MASTER EN SCIENCES INFORMATIQUES

Analyse des standards HL7 et KMEHR et développement d'un outil de conversion

de Quirini, Cédric

Award date:
2014

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2013–2014

Analyse des standards HL7 et KMEHR et développement d'un outil de conversion

Cédric de Quirini

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Ce mémoire a pour objectif la mise en place d'une méthodologie de travail permettant la création d'un outil de conversion. Celui-ci devra permettre de passer un document XML formaté selon le standard KMEHR, standard de messages médicaux belge, au format HL7, standard d'origine américaine mondialement utilisé.

Cette méthode se déroule selon trois étapes principales. La première utilise la syntaxe des standards pour en extraire les concepts. La deuxième effectue un matching entre les concepts des deux standards. Finalement, la dernière produit l'outil de traduction du KMEHR au HL7.

Ce document est structuré selon ces trois étapes, à raison d'une par chapitre. Un chapitre supplémentaire considère les améliorations qui pourront être effectuées dans le futur.

La lecture de ce mémoire nécessite des notions de base en XML.

Mots-clés: KMEHR, HL7, SumEHR, CDA, ontologies.

Abstract

The objective of this thesis is the development of a working methodology for the creation of a conversion tool. It will allow to convert an XML document formatted according to the KMEHR standard, Belgian medical messages standard, into the HL7 format, worldwide used American standard.

This method involves three main steps. The first uses the syntax of the standard to extract concepts. The second performs a matching between the concepts of both standards. Finally, the last produces the translation tool from KMEHR to HL7.

This document is structured according to these three steps, one for each chapter. An additional chapter considers what improvements can be made in the future.

Basic XML concepts are a prerequisite for reading this document.

Keywords: KMEHR, HL7, SumEHR, CDA, ontologies.

Avant-propos

Je tiens à remercier M. Colin, qui m'a accompagné à travers ce travail avec un soutien parfait, dans les bons comme dans les mauvais moments de sa création. Il a été un promoteur idéal.

Je voudrais également remercier M. Petit qui a été d'une grande aide dans les deux premières étapes.

Merci également à M. Temans pour son aide et son intérêt dans ce mémoire, et aux membres de eHealth qui ont soutenu ce projet. Ils ont été une grande assistance et une motivation dans la réalisation de ce mémoire.

Finalement, je tiens à remercier Thomas Alarcia, Marie-Alice Maes et ma famille pour leur soutien et leurs nombreuses relectures.

Table des matières

Résumé **3**

Abstract **5**

Avant-propos **7**

Introduction **11**

1 Contexte **13**
1.1 État actuel des services de santé 13
1.2 Standards 15
1.3 Le standard KMEHR 16
1.4 Le standard HL7 20
1.5 Schémas XML 26
1.6 Situation du mémoire en contexte 28

2 Construction des ontologies **31**
2.1 Ontologies 31
2.2 Identification des besoins 34
2.3 Choix des outils 35
2.3.1 Langage 35
2.3.2 Outil de manipulation 37
2.4 Méthodologie 38
2.4.1 Règles méthodologiques 40
2.4.2 Règle de transformation 1 41
2.4.3 Règle de transformation 2 42
2.4.4 Règle de transformation 3 43
2.4.5 Règle de transformation 4 44
2.4.6 Règle de transformation 5 45
2.4.7 Règle de transformation 6 45
2.4.8 Règle de transformation 7 46

2.4.9 Règle de transformation 8 46
2.5 Résultats obtenus 47
2.6 Limites 52

3 Matching d'ontologies 53
3.1 Identification des besoins 53
3.2 Choix des outils 55
3.2.1 KitAMO 58
3.2.2 Anchor 58
3.2.3 AUTOMS 59
3.2.4 Falcon-AO 59
3.2.5 HMatch 59
3.2.6 Optima 60
3.2.7 AMW 61
3.2.8 COMA 3.0 61
3.3 Choix d'implémentation 63
3.3.1 Matching automatique 65
3.3.2 Noms et structures identiques 66
3.3.3 Noms différents, structures identiques 66
3.3.4 Noms et structures différents 66
3.4 Résultats obtenus 67

4 Du KMEHR vers le HL7 69
4.1 But poursuivi 69
4.2 Choix du langage 70
4.3 Langage XSL 70
4.4 Méthodologie de la transformation 77
4.4.1 Utilisation du matching 77
4.4.2 Utilisation des templates 77
4.4.3 Éléments non utilisés 77
4.4.4 Traitement des ID 78
4.4.5 Perte d'information et types par défaut 78
4.5 Résultats obtenus 79

5 Limites et améliorations possibles 85
5.1 Automatisation de la construction des ontologies 85
5.2 Automatisation de la création de l'outil 86
5.3 Utilisation des schémas XML plutôt que des ontologies 86
5.4 Prolongation du projet 87
5.5 Dictionnaires de données 88

5.6 Validation de l'outil 88

6 Conclusion **91**

Bibliographie **94**

A Contenu du CD-ROM **95**

B Ontologies réalisées **96**

Introduction

De tout temps, la communication -l'échange de données- est essentielle. Aujourd'hui, l'informatique en est sa principale alliée. Le domaine médical n'y fait pas exception.

L'informatique permet de communiquer des informations sur les patients entre les différents prestataires de soins. Cependant, lorsqu'il y a transmission d'informations se pose la question de l'interopérabilité des systèmes. Ces messages ont besoin d'être structurés - d'où l'importance des standards de communication. Les standards permettent l'interaction. Ils définissent clairement les messages envoyés afin de partager les informations de manière claire et non ambiguë. C'est là que le bât blesse : chacun a sa façon de gérer la communication. Les problèmes se rencontrent lorsque les standards ne sont plus les mêmes.

Le KMEHR, standard créé par l'état belge, est utilisé dans ce pays afin de répondre au besoin de communication entre les acteurs du milieu médical.

Le HL7 est un autre standard bien connu, largement répandu à travers le monde. Il s'impose aujourd'hui comme le plus connu des standards en soins de santé et est notamment utilisé en Amérique et dans plusieurs pays d'Europe.

Ce mémoire a pour but d'introduire une méthodologie de travail permettant de construire un outil de traduction du KMEHR vers le HL7. Cette méthode sera testée à chaque étape afin de la valider mais aussi de la perfectionner. Elle consiste à utiliser les ontologies pour relier les concepts des standards afin de structurer la mise en place de cet outil.

Après un chapitre présentant les standards, le chapitre 2 présente une analyse des standards qui permet d'en modéliser les concepts. Le chapitre 3 montre la technique de matching qui lie ces concepts d'un standard à l'autre. Le chapitre 4 introduit la méthode de création de l'outil de traduction du KMEHR vers le HL7. S'en suit un bref chapitre sur les améliorations possibles.

Ces trois étapes principales correspondent aux trois niveaux de travail : les données, la syntaxe et les concepts. La construction des ontologies permet de passer de la syntaxe aux concepts, le matching travaille sur ces concepts, et la traduction fait le lien entre les concepts et les données.

Chapitre 1

Contexte

Ce chapitre présente l'état actuel de l'informatisation des services de santé. Il aborde ensuite la notion de standard et d'interopérabilité, puis explique les standards qui seront étudiés ici. Il expose également brièvement les principes des schémas XML et présente finalement le but de ce mémoire dans ce contexte.

1.1 État actuel des services de santé

L'importance de l'informatique dans la vie de tous les jours n'étant plus à discuter, il est normal que nos données médicales soient informatisées. Lorsqu'un patient subit un traitement, il doit effectuer des radios, des examens divers et se voit prodiguer des soins. Toutes ces informations sont bien évidemment regroupées et centralisées afin de permettre l'accès à celles-ci au personnel médical qui en a besoin.

De nos jours, ce service s'est encore amélioré : les dossiers médicaux peuvent être aisément transmis entre les différents intervenants : médecins traitants, hôpitaux, etc. Cela permet un suivi du dossier d'un patient de manière plus complète et en s'assurant que tous les intervenants disposent des informations nécessaires aux soins. De plus, cela évite des examens redondants, parfois lourds à subir et à payer par le patient.

En Wallonie, le Réseau Santé Wallon (ou RSW, [RSW, 2010]) permet l'interaction et la transmission des informations à travers la Wallonie. La communication avec ses équivalents en Flandre et dans la région bruxelloise est en cours de développement. Le RSW est une plateforme pour l'échange de documents médicaux gérée par la FRATEM [1]. Il permet de simplifier la prise en charge des patients. Ce dernier reste maître de ses informations: il peut contrôler qui a accès à son dossier. L'accès aux documents n'est autorisé que dans le cadre des

1. Fédération Régionale des Associations de Télématique Médicale

soins. Le tout est bien entendu sécurisé (vérification et enregistrement de l'historique des accès) afin de respecter la loi sur la vie privée. Le principe se base sur un "lien thérapeutique", qui est créé avec l'aval des deux parties (médecin et patient). Celui-ci est du type "le docteur X suit le patient Y en consultation du 01/05/2014 au 01/05/2015". Une fois le lien créé, le médecin en question a accès aux documents déclarés par les autres médecins.

Il a été créé pour répondre à un besoin de transmission des données : avant la mise en place du réseau, l'échange était peu présent et, lorsqu'il y avait échange, c'était souvent à sens unique (de l'hôpital vers le généraliste). Le RSW comble ce manque en fournissant une interface de transmission qui permet une communication bidirectionnelle et une mise à disposition des intervenants de toutes les informations utiles. Le généraliste peut donc à présent fournir des informations à un hôpital et ainsi économiser du temps au patient et améliorer son traitement.

Cette communication entre intervenants permet à un patient de changer de médecin traitant ou d'hôpital en sachant que son dossier "le suivra", et que les médecins qui vont s'occuper de lui disposeront de toutes les informations nécessaires sur ses antécédents, son état actuel, etc. Cela permet, par exemple, de suivre un traitement dans un hôpital spécialisé sans devoir repasser des examens déjà réalisés dans un autre hôpital.

Le Réseau Santé Wallon ne stocke pas forcément l'information mais se contente souvent de la référencer, afin que celle-ci puisse être accessible par tous. Les données ne voyagent donc que lorsque cela est nécessaire, mais sont disponibles dans les hôpitaux concernés et accessibles via le RSW. Dans le cas des membres du réseau qui ne disposent pas de matériel adapté pour permettre un tel système (les médecins traitants qui n'ont pas d'outils informatiques, par exemple), le RSW se charge du stockage de l'information.

Cette information n'est donc pas multipliée inutilement, mais simplement mise à disposition de ceux qui en auraient besoin.

Au-dessus du RSW se situe eHealth, plateforme fédérale chargée de mettre en place le "méta-hub". Celui-ci relie le Réseau Santé Wallon aux différents services équivalents en Flandre et à Bruxelles. Cela permet la communication avec les autres régions pour le partage des données afin de ne pas limiter l'action du réseau de soins aux régions.

Cependant, le transport de l'information médicale d'un système à l'autre via le réseau ne vient pas sans contraintes. Tous ces composants informatiques doivent communiquer à l'aide du même langage, des mêmes codes. Et comme bien souvent en informatique, il n'en existe pas d'universel. Il faut alors parfois convertir les informations d'un format à un autre avant de pouvoir communiquer.

Afin d'accorder les formats de ces communications et de permettre l'inter-

opérabilité, il convient d'utiliser des standards. La section suivante en fait la description.

1.2 Standards

La notion d'interopérabilité est un fil rouge dans ce mémoire mais également le but avoué de l'utilisation de standards. Cette section l'introduit, avant de présenter les standards et leur utilisation.

La définition de l'interopérabilité est la suivante : *"Compatibilité des équipements, des procédures ou des organisations permettant à plusieurs systèmes, forces armées ou organismes d'agir ensemble."* (source : [Larousse, 2014])

Autrement dit, l'interopérabilité permet la communication : c'est la capacité à parler ensemble, à s'échanger des informations. Elle est nécessaire entre les équipements médicaux, mais aussi entre les différents intervenants médicaux belges : hôpitaux, médecins, spécialistes, service central.

Comme mentionné au point précédent, le matériel médical communique afin de transmettre directement les informations et centralise celle-ci dans le dossier du patient. Mais la communication entre tous ces appareils n'est pas innée : elle est régie par des codes, utilise un langage bien précis.

Dans le cas où chaque appareil utilise son propre langage, il nécessite une interface de communication avec tous les autres. Étant donné la diversité du matériel médical, il est facile d'imaginer le nombre incommensurable d'interfaces nécessaires, s'il en faut une différente pour chaque matériel devant communiquer avec un autre.

C'est ici qu'interviennent les standards : en formalisant le langage utilisé, ils permettent de s'entendre sur les codes de communication. À condition que les équipements informatiques utilisent le même standard, ils pourront aisément communiquer et partager de l'information. C'est ainsi que les hôpitaux s'échangent les informations sur les patients lorsque cela est nécessaire : en utilisant le même standard, qui régit leurs communications.

La définition d'un standard est la suivante : *"Règle fixe à l'intérieur d'une entreprise pour caractériser un produit, une méthode de travail, une quantité à produire, le montant d'un budget."* [Larousse, 2014]

Bien sûr, dans notre cas l'entreprise est en réalité plusieurs entreprises, un pays, voire même plusieurs, comme cela sera montré au point 1.6.

Les standards sont donc un outil indispensable pour permettre la communication et l'interopérabilité dans les services de soins, puisqu'ils accordent la transmission d'informations entre les parties communicantes.

Le problème est bien sûr la diversité des standards existants. Ce point sera détaillé plus loin dans ce chapitre, mais le HL7, standard bien connu dans le

milieu médical, tend à s'imposer mondialement. Il n'est cependant pas le seul existant et, même s'il prend de l'ampleur, il en existe actuellement bien d'autres.

Dès lors, permettre une uniformisation des systèmes, un rassemblement de l'information et une meilleure communication pour l'échange des données est un des défis actuels. Le but est bien sûr un meilleur suivi du patient dans le temps - et donc un soin plus adapté. Le rôle des standards de communication est vital dans cette évolution.

Les deux sections suivantes présentent les deux standards concernés par ce mémoire : le KMEHR, standard développé et utilisé par le réseau de santé en Belgique, et le HL7, standard d'origine américaine aujourd'hui largement mondialisé et peu à peu transformé en norme.

1.3 Le standard KMEHR

Le KMEHR, "Kind Messages for Electronic Healthcare Record", introduit en 2002, est actuellement maintenu par eHealth ([eHealth, 2014]) et utilisé dans le Réseau Santé Wallon. C'est l'implémentation de la quatrième recommandation de la *Belgian Health Telematics Commission*[2]

Il a été conçu par la Belgique afin de répondre à un besoin d'interopérabilité entre les acteurs du monde médical.

Il est composé de messages au format XML[3] avec une grammaire définie. Les messages expriment des transactions médicales, définies par la plate-forme eHealth. À ceci viennent se greffer les tables de référence qui peuvent être utilisées dans ces messages et un ensemble de webservices.

Le KMEHR se compose donc de trois éléments principaux :

- Le schéma XML (ou XSD, structure présentée au point 1.5) définit la structure du standard, sa grammaire. Elle permet de structurer le format des messages, les emplacements prévus dans le document pour les divers éléments qu'il contient.
- Les transactions médicales définissent le type de message envoyé : le dossier du patient, le résumé des médicaments, des images médicales (radios ou autre qui ont été effectuées) et bien d'autres. Pour comparaison, les schémas définissent où se trouvent les éléments et les transactions définissent quels éléments sont obligatoires ou optionnels selon le type de document utilisé.
- Les tables de référence contiennent des valeurs pour décrire une situation. Par exemple, différents codes existent pour exprimer la raison d'une

2. `http://www.health.belgium.be/eportal`, Mise en ligne : 2014, Consultation : 22/05/2014 16:25

3. XML : Extensible Markup Language, message composé de balises, utilisé par exemple en HTML.

incapacité de travail : maladie, accident, grossesse, etc. Tous ces codes sont conservés dans les tables de référence afin de permettre d'exprimer correctement les caractéristiques d'un patient.

Le KMEHR est utilisé dans le cadre du Réseau Santé Wallon qui est le service permettant la centralisation du dossier d'un patient. Il rassemble les informations sur les différents traitements de ce dernier, et les met également à sa disposition ainsi qu'aux médecins qui se chargent de lui. Le KMEHR est donc bien sûr un élément vital de son fonctionnement, permettant la transmission de ces informations.

Le KMEHR sert aussi aux équivalents du RSW en Flandre et à Bruxelles. C'est donc un standard utilisé de manière nationale dans les communications.

Les messages KMEHR sont composés d'un "header" et au minimum d'un "folder". Le premier donne des informations contextuelles (expéditeur, destinataire, date et heure de l'envoi, etc.) tandis que le deuxième fournit de l'information sur un patient. Il contient au moins une "transaction". Les transactions sont disponibles en ligne sur le site du KMEHR [4].

La figure 1.1 montre la structure d'un message KMEHR. On peut y retrouver le "header" (plus foncé sur l'image) et le "folder". Celui-ci a une cardinalité (identifiable à sa gauche) de 1 à ∞. Cela signifie, comme mentionné précédemment, qu'un message KMEHR en contient au moins un.

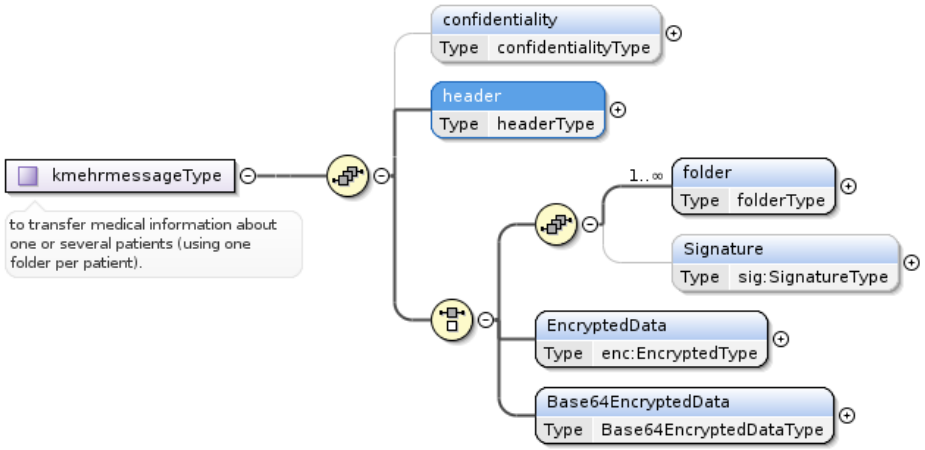

FIGURE 1.1 – Structure d'un message KMEHR

Les figures 1.2 et 1.3 présentent les éléments contenus dans les "folder" et "header" d'un message KMEHR de façon plus détaillée.

4. https://www.ehealth.fgov.be/standards/kmehr/content/page/transactions, Mise en ligne : inconnue, Consultation : 20/05/2014 07:40

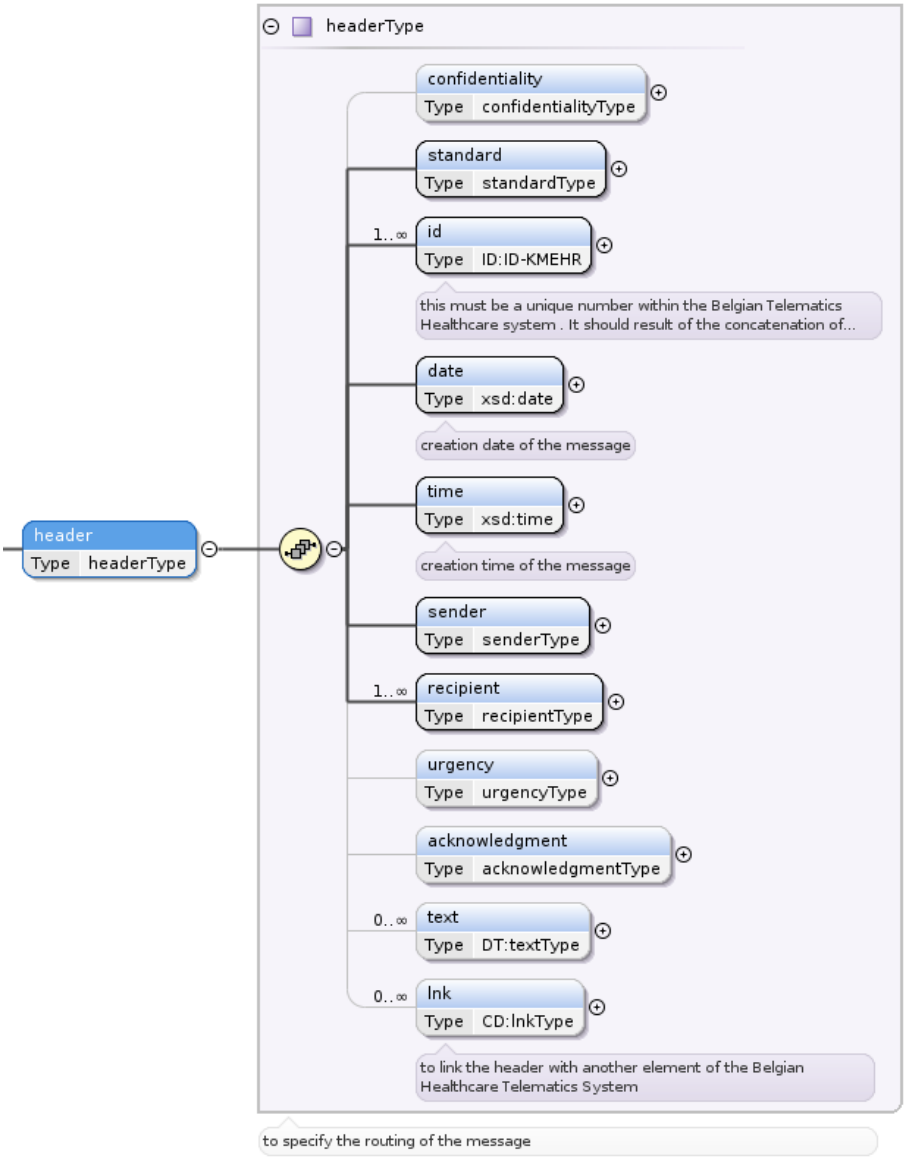

FIGURE 1.2 – "header" d'un message KMEHR

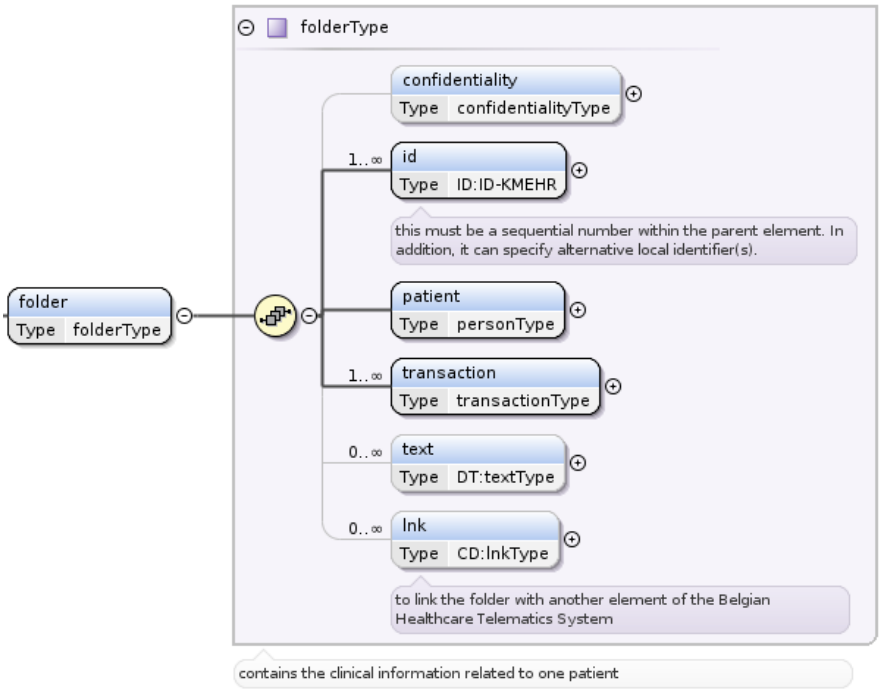

FIGURE 1.3 – "folder" d'un message KMEHR

Il existe plusieurs "niveaux" de standardisation d'un document. Plusieurs possibilités existent pour formater un document KMEHR : le document le plus simple est très peu structuré et sert juste au transport d'un texte brut incompréhensible par un ordinateur, tandis que le plus perfectionné exprime des données de manière entièrement interprétable par un système automatique. Les quatre niveaux possibles sont les suivants :

- Niveau 1 : Le message comporte un code de transaction afin de déterminer le type de message (dossier du patient, instruction de soin, ...), mais l'entièreté du message est un texte simple, non interprétable.
- Niveau 2 : Les informations dans la partie "header" sont codées et interprétables tandis que le reste du document est en texte simple. Les informations sur l'expéditeur et le patient peuvent donc être traitées automatiquement.
- Niveau 3 : Les différents éléments du "folder" sont eux aussi codés, même si leur contenu ne l'est pas. Un ordinateur peut donc différencier les types de contenus.
- Niveau 4 : Tous les éléments sont codés, même les informations contenues dans les "folder". L'entièreté du message est interprétable et donc utilisable par un ordinateur.

Une des transactions importantes du KMEHR est le SumEHR , ou "Summa-

rised Electronic Healthcare Record". Il fournit un résumé succinct sur le patient. Cette information doit permettre à un membre du personnel soignant de comprendre l'état du patient et ses besoins. Il est en quelque sorte similaire au dossier d'un patient.

Finalement, il faut noter que le standard KMEHR fournit également un système de définition de web services (programme d'échange de données entre des systèmes) permettant l'envoi des transactions aux autres membres du réseau.

La description formelle du standard, en dehors des informations textuelles qui le décrivent, est composée de schémas au format XML. Le format de fichier exprimant ces "schémas XML" est le XSD [5]. Ces fichiers expriment les règles de "grammaire" d'un document XML. Ils seront détaillés à la section 1.5.

1.4 Le standard HL7

Le HL7, ou "Health Level Seven" ([Health Level Seven, 2011a]) est une ASBL fondée en 1987, qui a pour but de développer des standards pour la transmission de messages médicaux. Initialement américaine, l'organisation s'est progressivement développée. Aujourd'hui accréditée par l'ANSI [6] et l'ISO [7], elle fournit des standards mondiaux. Le nom "HL7" est utilisé aussi bien pour désigner l'organisation que les standards eux-mêmes.

Le HL7 est à présent utilisé dans plusieurs pays du monde qui participent également à son développement. On peut supposer que son influence ira croissante étant donné sa normalisation de plus en plus grande. Il s'impose dès lors comme une norme mondiale incontournable dans les messages en soins de santé.

Il existe plusieurs versions du HL7. Les plus utilisées sont les versions 2.x, bien que la version 3.0 ait été développée depuis 1995 et publiée en 2005. La figure 1.4 montre que les différentes versions du HL7 sont utilisées de manière très éparses.

La version 3.0 du HL7 a comme but de compenser les insuffisances de la version 2, en formalisant beaucoup plus la méthodologie afin de faciliter les échanges. En effet, les multiples variations possibles dans une même version obligent les parties communicantes à s'accorder sur les messages avant de pouvoir réellement communiquer, ce qui est en soi un frein à l'interopérabilité. Alors que le but d'utiliser le même standard est de pouvoir aisément échanger des informations, le fait d'imposer une négociation à tous les participants ralentit le développement des infrastructures d'échange et nécessite un travail de transfor-

5. XSD : XML Schema, langage de description formelle de la structure et du contenu d'un document XML
6. American National Standards Institute
7. International Organization for Standardization

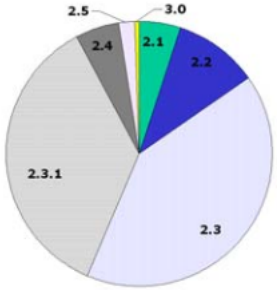

Figure 1: Approximate real-world usage of HL7 messaging standards. The vast majority of HL7 messaging is done using messages that approximate HL7 2.3 or HL7 2.3.1. Newer releases of HL7 (2.5, 3.0, and soon 2.6) represent a very small portion of real-world interfaces.

FIGURE 1.4 – usage du HL7, [Corepoint Health, 2006]

mation des messages de chaque côté, ce qui est plus conséquent. Cette situation a disparu avec la version 3.0 qui formalise les échanges.

La troisième version du HL7 est basée sur le "Reference Information Model" ou RIM et les messages respectant ce standard permettent beaucoup moins de libertés dans son implémentation. Le RIM est un modèle représentant les données cliniques contenues dans le HL7 et le cycle de vie d'un message. Il est donc à la base de tout message HL7 (depuis la version 3).

Cependant, cela n'est pas sans conséquences : tandis que les messages des versions 2.x sont rétro-compatibles (un message de la version 2.3 pourra être lu par un système utilisant la version 2.6), la version 3.0 casse le lien avec les versions précédentes : les messages de la version 2.x étaient composés de texte divisé par des séparateurs alors que ceux de la version 3 sont des messages XML.

De nos jours, la variation dans les versions est encore un frein au développement du HL7 : si deux entités souhaitent s'échanger des informations, il y a beaucoup de travail à réaliser sur la façon d'utiliser le HL7. Ce travail est grandement simplifié si la version 3 est utilisée, mais celle-ci n'est pas encore répandue. Il existe cependant des projets, notamment au niveau européen [8], destinés à implémenter une communication des informations médicales entre les différents acteurs à l'échelle européenne, mais ce projet n'en est qu'à ses débuts.

La version 3.0 du HL7 peine à percer dans le milieu médical notamment en raison de l'omniprésence de la version 2. La dernière version n'étant pas rétro-compatible, toute partie communicante qui utilise la version 3.0 doit également implémenter la version 2 pour des raisons de compatibilité avec le matériel existant. Cela ne favorise pas l'arrivée de la dernière version du HL7 sur le marché, alors que celle-ci est bien plus formelle et représente un gain de temps.

La définition du standard se base, comme dans le cas du KMEHR, sur des schémas XML afin d'exprimer la syntaxe des documents rédigés. Les fichiers

8. `http://www.epsos.eu/`, Mise en ligne : 18/01/2013, Consultation : 21/05/2014 12:10

XML servant à transmettre des informations sont donc “conformes” à cette description. Ces fichiers sont interprétables automatiquement, ce qui permet une vérification facile de la structure de tout document HL7.

Tout comme le KMEHR possède son SumEHR, le HL7 possède le CDA (“Clinical Document Architecture”). Celui-ci est lui aussi basé sur du XML, étant donné qu'il a été produit en parallèle avec la version 3 du HL7. La première version date de 2000 et la deuxième version date quant à elle de 2005. Il sert, comme le SumEHR, à transmettre l'information du dossier d'un patient afin d'assurer la continuité des soins. La figure 1.5 présente une partie du “Refined Message Information Model”, ou RMIM, du CDA. Il s'agit d'une version réduite du RIM, se limitant aux éléments contenus dans le CDA.

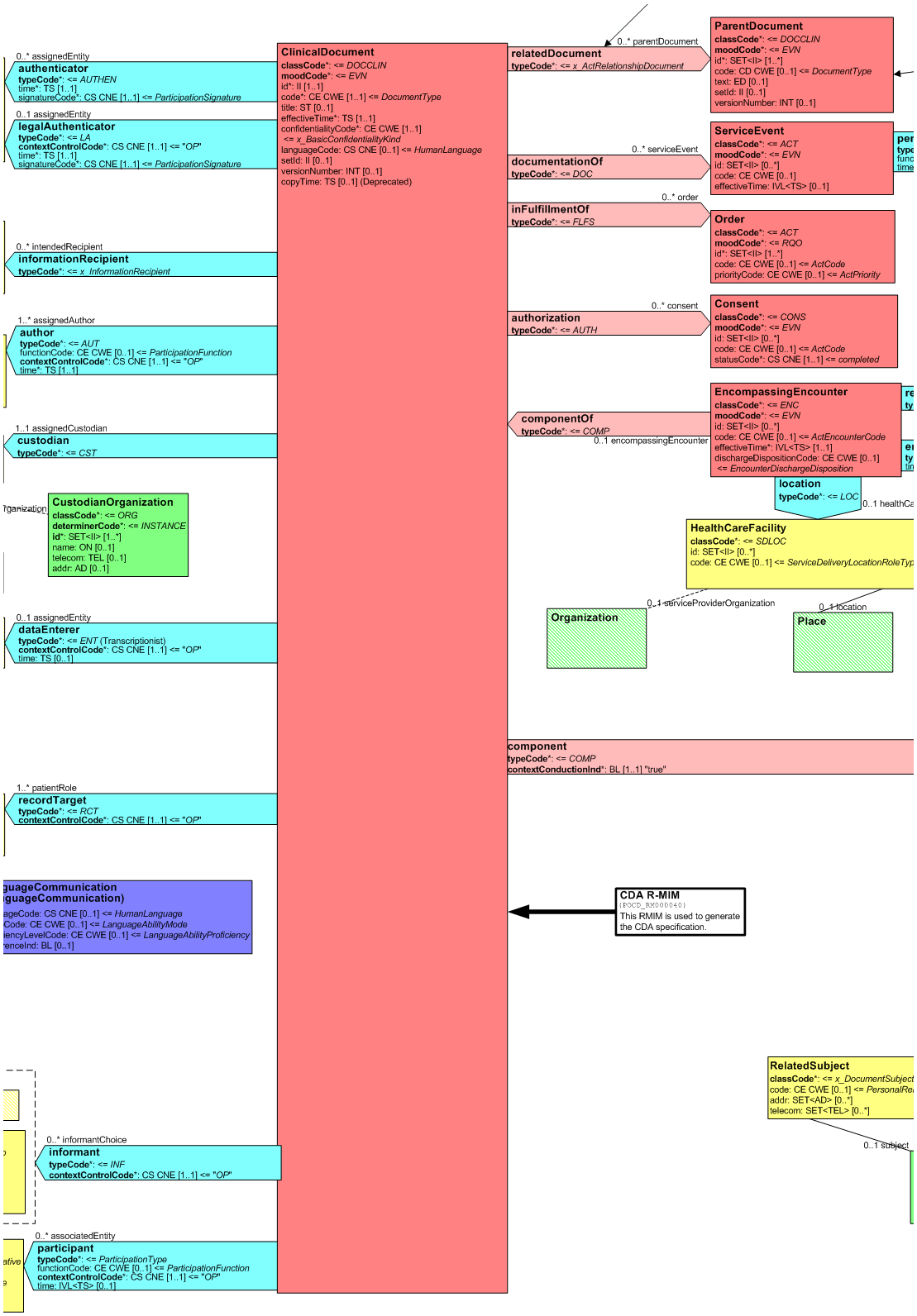

FIGURE 1.5 – RMIM du CDA

Le CDA est actuellement au centre de plusieurs projets et joue un rôle important dans l'échange d'informations médicales. Peu utilisé (car non nécessaire) à l'intérieur d'un même hôpital, il se retrouve de plus en plus dans les échanges entre infrastructures (hôpitaux, services centraux, médecins, ...) étant donné l'économie d'implémentation qu'il représente.

Pour ce qui est de la composition d'un CDA, elle est assez similaire à celle des messages KMEHR : un "header" contient les informations de base nécessaires à l'identification du patient, de l'auteur et du document. Ensuite vient le "body", qui contient les informations concernant le client. La figure 1.6 présente ces différents éléments.

La norme décrivant le CDA est disponible sur le site du HL7 ([Health Level Seven, 2011b]), les standards étant fournis gratuitement depuis 2013. Cette description contient plusieurs documents : descriptions littéraires, spécifications, XSD, documents sur l'implémentation, etc. Dans le cadre de ce mémoire, seule la partie relative aux XSD sera utilisée.

Le lecteur peut se référer à [Benson, 2012] pour plus de documentation sur le HL7 et le CDA. On y décrit notamment les différentes étapes de sa construction, ses différents éléments et son lien avec le "Continuity of Care Document", ou CCD, qui n'est pas abordé dans ce mémoire.

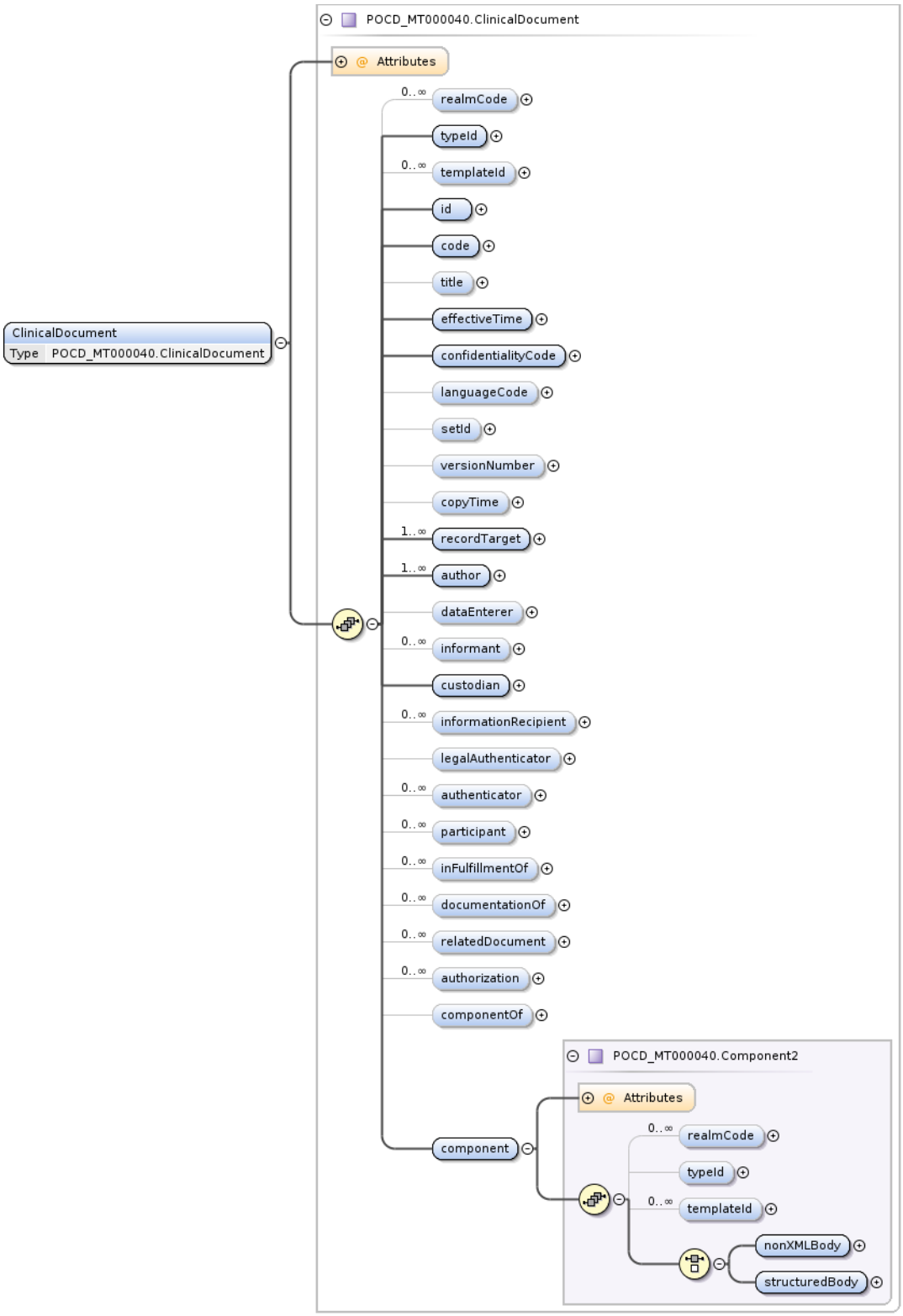

FIGURE 1.6 – Éléments d'un CDA

1.5 Schémas XML

Cette section présente le fonctionnement des schémas XML. Un bref rappel des notions du XML est préalablement présenté mais le lecteur est supposé avoir des connaissances de base dans ce domaine.

Les schémas XML, également appelés XSD, sont des documents permettant la compréhension des standards exposés ici. Ils décrivent la structure et le format des fichiers XML, mais permettent aussi de valider ceux-ci par rapport à cette structure. Ils contiennent en effet des descriptions des types autorisés dans le document.

Pour rappel, les fichiers XML sont écrits en texte clair, en utilisant des balises. Il est important de signaler que les fichiers XSD, bien que définissant la structure d'un document XML, sont eux aussi écrits en XML.

Voici un exemple de fichier XML provenant du KMEHR (largement simplifié) et mentionnant une rhinite :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kmehrmessage>
3   <transaction>
4    <item>
5      <id S="ID-KMEHR" SV="1.0">3</id>
6      <cd S="CD-ITEM" SV="1.0">healthcareelement</cd>
7      <content>
8        <cd S="ICD" SV="10">J00</cd>
9        <cd S="ICPC" SV="2">R74</cd>
10     </content>
11     <content>
12       <text L="en">acute infectious rhinitis</text>
13     </content>
14     <beginmoment>
15       <date>2007-09-03</date>
16     </beginmoment>
17     <lifecycle>
18       <cd S="CD-LIFECYCLE" SV="1.0">active</cd>
19     </lifecycle>
20   </item>
21  </transaction>
22 </kmehrmessage>
```

On y retrouve des éléments (kmehrmessage, transaction, item, etc.) et des attributs (S, SV, L). Ils contiennent des valeurs (3, healthcareelement, etc.). La

structure d'un tel élément est définie par son schéma XML ci-dessous. Plusieurs éléments ont volontairement été retirés afin de simplifier l'exemple.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2   <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       >
3    <xsd:element name="kmehrmessage">
4      <xsd:element name="transaction">
5        <xsd:complexType name="itemType">
6    <xsd:sequence>
7      <xsd:element name="id" type="ID:ID-KMEHR" maxOccurs="
           unbounded">
8      </xsd:element>
9      <xsd:element name="cd" type="CD:CD-ITEM" maxOccurs="
           unbounded">
10     </xsd:element>
11     <xsd:element name="content" type="contentType"
           minOccurs="0" maxOccurs="unbounded">
12     </xsd:element>
13     <xsd:element name="beginmoment" type="momentType"
           minOccurs="0"/>
14     <xsd:element name="lifecycle" type="lifecycleType"
           minOccurs="0"/>
15   </xsd:sequence>
16       </xsd:complexType>
17     </xsd:element>
18   </xsd:element>
19 </xsd:schema>
```

La table 1.1 explique les différentes balises et attributs rencontrés ici.

Ce schéma XML montre comment les paramètres du code XML en exemple sont définis : l'ordre des éléments et leurs nombres d'occurrences est précisément mentionné dans le schéma. C'est la raison d'être des schémas XML, qui structurent les fichiers XML.

Le codage de la structure d'un document à l'aide d'un schéma permet non seulement de respecter les règles de format lors de la création d'un document XML, mais aussi de vérifier un document reçu. Par exemple, une personne voulant vérifier la validité d'un document XML peut le faire contrôler par un "processeur de fichiers"[9] à l'aide du XSD correspondant.

Cette section n'étant qu'une introduction sur le sujet, de nombreuses fonc-

9. La notion de processeur XML sera vue plus en détail au chapitre 4

Éléments XSD	Définition
balise “element”	Définit un élément qui apparaîtra à cet endroit précis, entre des balises.
attribut “name”	Définit le nom de la balise. Exemple : un élément avec l'attribut name possédant la valeur “message” donnera la balise <message>.
balise “complexType”	Permet de définir le type de la balise en cours afin de déterminer les éléments qu'elle doit contenir.
balise “sequence”	Spécifie que chaque sous-élément doit apparaître dans l'ordre, de 0 à un nombre infini d'occurrences.
attribut “type”	référence une autre section du XSD où la définition de cet attribut se trouve.
attribut “min(max)Occurs”	Nombre minimum (maximum) d'occurrences de cet élément dans le document XML.

TABLE 1.1 – Tableau explicatif des éléments d'un XSD

tions des XSD ne sont pas abordées dans ce chapitre, comme les domaines de validité d'un champ et les types simples.

Les deux standards mentionnés dans ce chapitre fournissent tous deux les schémas XML servant à structurer les documents médicaux. Ces schémas permettent de vérifier les règles d'écritures d'un document lors d'un travail sur ces standards, mais aussi de contrôler les documents échangés pour vérifier qu'ils respectent le format établi. Ce sont ces fichiers qui vont servir de base de travail au chapitre 2.

1.6 Situation du mémoire en contexte

Le contexte ayant été posé, cette section présente la dynamique dans laquelle s'inscrit ce mémoire.

Comme le montre la section 1.1, l'interconnexion des services médicaux belges est bien coordonnée et en expansion. Du côté de la coopération internationale, par contre, l'échange de messages n'est pas aussi performant. De nos jours, il est actuellement impossible de transmettre automatiquement le dossier d'un patient à un service de soin étranger : le patient qui doit se faire soigner à l'étranger devra emmener avec lui son dossier médical s'il veut s'assurer d'un suivi dans les soins. Cet état de fait est bien évidemment très limitant et est appelé à évoluer dans le futur, au vu de l'interaction grandissante entre les services de santé.

Dans le cas de la Belgique, le fait d'utiliser le KMEHR pour l'échange de messages lui permet d'optimiser ce standard par rapport à ses besoins. A contrario, étant donné qu'elle est le seul pays à utiliser ce système, cela demande un gros

travail de transformation des messages pour être capable de les envoyer. Actuellement, aucun dispositif n'existe pour permettre la transmission des messages KMEHR à l'étranger.

Deux pistes s'offrent au réseau de santé belge pour permettre l'échange international : la construction d'un outil permettant la transformation d'un document KMEHR vers un document HL7 et inversement, ou l'abandon du KMEHR pour le HL7 sur l'entièreté du réseau. La deuxième alternative serait intéressante étant donné que plusieurs pays européens sont déjà affiliés au HL7. Le lecteur se référera à [Health Level Seven, 2014] pour la liste des pays utilisant le HL7. Cependant, les différentes versions du HL7 qui sont implémentées sont un obstacle à la transmission. Il est donc difficile de choisir une version spécifique du HL7 étant donné qu'il y aurait quand même un travail à faire pour la communication avec les autres pays. Le CDA et la version 3 du standard HL7 (qui est bien plus formelle que les versions antérieures) permettraient de contourner ces difficultés. La facilité d'intégration avec les réseaux étrangers dépend donc de la version du HL7 utilisée. C'est la raison pour laquelle la mise en place d'une structure pour l'ensemble de l'Europe serait très intéressante.

Bien sûr, aucune alternative n'est idéale : la première (un outil de conversion) demande un énorme travail qui devra être actualisé au fur et à mesure de l'évolution des standards, tandis que la deuxième (utilisation du HL7) obligerait à un remaniement complet de l'infrastructure actuelle. Autant dire que le choix ne se fait pas avec légèreté. Il faut également souligner que développer aujourd'hui des outils de conversion n'empêchera pas de choisir, dans le futur, de transformer le système pour utiliser le HL7.

Ce mémoire, inspiré du besoin d'interconnexion entre le standard national et le HL7, effectue l'analyse et la comparaison des deux standards. Le travail étant conséquent, il est impossible de traduire l'entièreté du KMEHR en HL7 et inversement dans ce mémoire.

Nous nous concentrerons donc sur le SumEHR en son équivalent en HL7, le CDA dans sa deuxième révision. Le but est de faciliter la recherche des similitudes sémantiques afin de développer un outil de conversion du SumEHR vers le CDA. La démarche ne sera pas la création de l'outil en lui-même, étant donné l'importance du travail, mais la mise en place d'une méthodologie permettant son approche et sa réalisation de manière performante et efficace. Un échantillon du convertisseur (limité à l'élément décrivant le patient) sera également réalisé afin de pouvoir tester la mise en place de la méthode et d'analyser les différents obstacles rencontrés. Ceux-ci permettront, à leur tour, d'améliorer cette même méthode.

Cette transformation du KMEHR en HL7 a été modulée en trois étapes, qui correspondent aux différents chapitres.

Le chapitre 2 aborde les ontologies pour l'analyse des standards, effectuée en transformant les deux standards en ontologies.

Le chapitre 3 expose le matching entre ces ontologies, afin d'évaluer les similarités sémantiques.

Le chapitre 4 crée un outil de conversion du KMEHR vers le HL7 en utilisant le matching réalisé.

L'outil créé en appliquant la méthode détaillée dans ce document permettra d'envoyer à l'étranger les documents au format HL7 afin de transmettre les informations médicales, en conservant à l'intérieur de la Belgique le système actuel. Cela permettrait par exemple la communication du Réseau Santé Wallon avec le Grand Duché de Luxembourg, qui utilise le HL7 pour la transmission des messages.

Chapitre 2

Construction des ontologies

Ce chapitre présente l'étape d'analyse qui débute le processus de ce mémoire. Celle-ci consiste à analyser les éléments contenus dans les standards KMEHR et HL7. Cette analyse se fait en utilisant des ontologies, permettant d'ajouter du sens aux informations contenues dans les schémas XML des standards.

La notion d'ontologie sera détaillée afin de familiariser le lecteur avec ce modèle. Ensuite, les besoins ayant poussé à l'utilisation des ontologies seront détaillés, suivis des choix effectués quant aux outils. Finalement, les résultats obtenus seront présentés, avec les limites inhérentes au processus détaillé ici.

2.1 Ontologies

Les ontologies sont un des outils majeurs utilisés dans ce mémoire, servant à identifier les concepts dans les modèles puis à établir des correspondances entre ceux-ci. Cette section est une introduction sur le sujet, permettant de comprendre le travail effectué dans ce chapitre.

Selon [IEEE, 2003] [1], la définition d'une ontologie est la suivante : *"Une ontologie est semblable à un dictionnaire ou un glossaire, mais avec plus de détails et une structure qui permettent aux ordinateurs de traiter leur contenu. Une ontologie se compose d'un ensemble de concepts, d'axiomes, et de relations qui décrivent un domaine d'intérêt."*

Autrement dit, une ontologie est un modèle de représentation des concepts, des individus, de leurs interactions et de leurs propriétés.

Une ontologie est composée d'individus liés par des propriétés. La figure 2.1 montre les relations possibles entre des individus : "Matthew", "Gemma" et "England". Ceux-ci sont liés par des propriétés exprimées de la manière suivante : "Matthew hasSibling Gemma" et "Matthew livesIn England". Autrement dit,

1. Institute of Electrical and Electronics Engineers

Matthew habite en Angleterre et a une soeur (ou un frère, le modèle de la figure 2.1 ne fournissant pas l'information) nommé(e) Gemma. Ce sont les données et les liens qui les relient. Les propriétés ont une série de caractéristiques qu'elles peuvent posséder (fonctionnelle, transitive, symétrique, asymétrique, réflexive ou irréflexive) mais qui ne sont pas présentées ici.

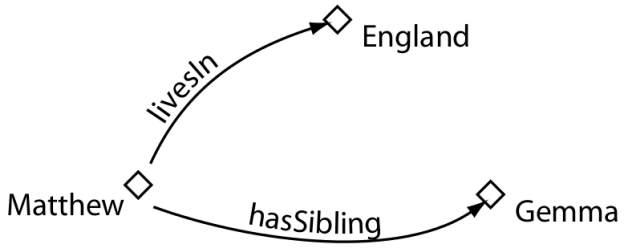

FIGURE 2.1 – Ontologies - propriétés et individus, [Horridge, 2011]

Les classes représentent des individus ayant les mêmes propriétés. Les individus sont alors regroupés dans la même classe. Ces classes peuvent avoir des propriétés, qui sont alors les propriétés de tout individu appartenant à la classe. La figure 2.2 montre un exemple d'individus regroupés en classes : les personnes, les animaux domestiques et les pays. On y voit apparaître une relation supplémentaire, hasPet, qui montre que Matthew a un animal nommé Fluffy. Les classes rassemblent des individus : Gemma et Matthew sont regroupés dans une classe "Person". L'Italie, l'Angleterre et les USA sont regroupés dans une classe "Country" et Fluffy et Fido sont regroupés dans la classe "Pet".

Cette figure 2.2 est donc une représentation en ontologie d'une partie du monde, qui décrit l'individu "Matthew".

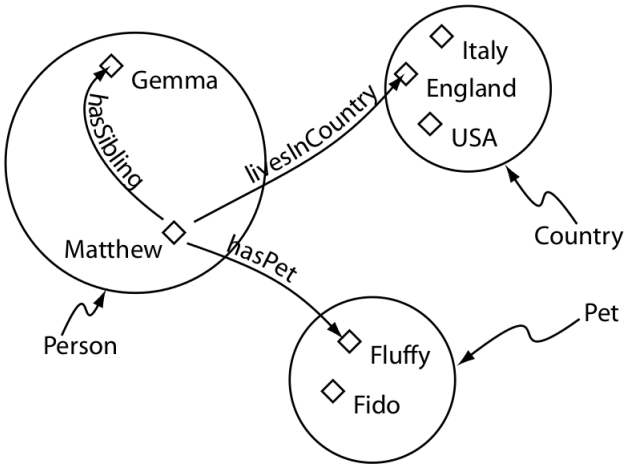

FIGURE 2.2 – Ontologies - classes, [Horridge, 2011]

Ces classes peuvent posséder des sous-classes. Par exemple, la classe "Patient" est la sous-classe de la classe "Personne". Elle raffine la classe mère en y ajoutant des propriétés. La classe enfant conserve bien sûr les propriétés de la classe parent (autrement dit, la classe "Patient" possède au moins les propriétés de la classe "Personne").

La figure 2.3 montre un exemple de sous-classes peuplées d'individus. Celles-ci représentent des garnitures de pizza. La première classe, "PizzaTopping", la plus générale, représente toute garniture de pizza. La suivante, "VegetableTopping", est une sous-classe de la première. On peut observer qu'il existe des individus de la première qui n'appartiennent pas à la deuxième. Ces individus situés en-dehors de la classe "VegetableTopping" sont donc bien des garnitures mais qui n'entrent pas dans la catégorie végétarienne.

Finalement, il faut souligner que les individus contenus dans la classe "TomatoTopping" possèdent les mêmes propriétés que la classe parent,"VegetableTopping". Celle-ci, à son tour, possède les mêmes propriétés que sa classe parent, "PizzaTopping". Dès lors, en imaginant qu'une propriété soit attribuée à la classe "PizzaTopping", du type "chaque PizzaTopping peut être mis zéro, une ou plusieurs fois sur une pizza". Dans ce cas, tous les individus de la classe "TomatoTopping" héritent de cette propriété étant donné que cette classe est une classe descendant de la classe "PizzaTopping".

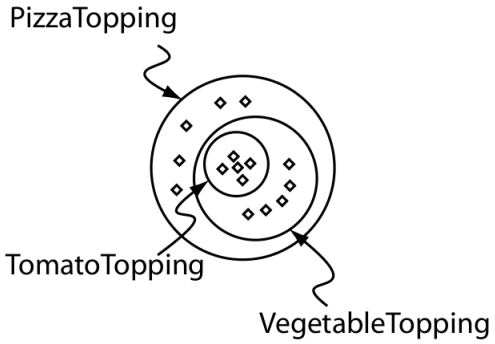

FIGURE 2.3 – Ontologies - sous-classes, [Horridge, 2011]

2.2 Identification des besoins

La première étape dans la construction de la méthodologie de travail est l'étude des standards mentionnés au chapitre 1. Elle est destinée à extraire des schémas XML les éléments qui représentent des concepts pour les dégager de leur implémentation technique. Le but ultérieur est de permettre la liaison entre ces concepts d'un standard à l'autre. Afin d'exprimer ces concepts, il est nécessaire de pouvoir les représenter dans un système qui permet de reproduire la structure existante tout en ajoutant du sens aux éléments analysés et en perdant les détails inutiles de l'implémentation.

Ce traitement s'inscrit dans la démarche méthodologique globale : ce chapitre expose la première partie de la méthodologie de ce mémoire, où les concepts sont extraits des standards afin de pouvoir les utiliser pour effectuer le matching au chapitre suivant, et ceci dans le but de mettre en place le traducteur du KMEHR. La méthode dans sa globalité sera exposée plus en détail au chapitre 3.

Il est apparu que les ontologies répondaient bien aux besoins exprimés ici. Cette section expose en détail les raisons qui justifient ce choix.

Il est nécessaire de rassembler les informations contenues dans le standard et d'y ajouter une sémantique. En effet, à l'intérieur d'un schéma XML, les éléments identiques peuvent être placés à des endroits différents : l'adresse, par exemple, est utilisée par différents éléments à des endroits variés (entre autres pour le médecin et pour le patient). Cette relation est à sens unique : lorsqu'on accède à l'élément "patient", on observe que celui-ci se réfère à l'élément "adresse", tout comme le fait l'élément "médecin". Mais la simple observation de l'élément "adresse" ne permet pas de connaître les endroits où celui-ci est utilisé.

Utiliser les ontologies fait aisément ressortir ces relations bidirectionnelles qui manquent aux schémas XML, à l'aide des propriétés des classes.

Par contre, il est vital, lors de l'analyse, de ne pas perdre l'ordre imposé par

les standards. Les schémas étant écrits en XML, ils sont structurés en arbre. Il y a donc une information structurée intrinsèque au document. En effet, un élément "text" dans l'entité patient n'est pas identique à un élément du même nom dans l'adresse de celui-ci. Il faut également souligner que l'arbre du schéma XML possède la notion de cardinalités : un sous-élément peut, selon les règles, exister une ou plusieurs fois, voire pas du tout. Dès lors, il est important de conserver cette information malgré le changement de format de l'information.

C'est en fonction de ces choix que l'utilisation d'une ontologie s'est imposé : comme montré à la section 2.1, les ontologies permettent d'utiliser des classes pour représenter les concepts. Ces classes englobent des individus. Elles peuvent être liées entre elles par des relations binaires qui représentent leurs liens, auxquelles s'ajoutent des cardinalités. De plus, elles possèdent une structure hiérarchisée : les classes peuvent posséder des sous-classes et les sous-classes héritent des propriétés de la classe parent.

Les ontologies offrent donc tous les outils pour représenter au mieux les informations contenues dans les XSD. Elles sont adaptées à la représentation de l'information de base et conservent l'information sous-jacente des schémas. Elles permettent, comme souhaité, d'ajouter de l'information à l'analyse lors de la transformation des schémas en ontologies. Elles s'imposent donc comme un excellent modèle de représentation pour l'expression des concepts des standards.

2.3 Choix des outils

Cette partie présente le langage et l'outil de travail qui ont été sélectionnés pour l'analyse des standards et l'écriture d'ontologies. En effet, il convient de sélectionner un langage de représentation des ontologies, mais aussi un outil permettant sa manipulation aisée pour éviter une perte de temps liée à un apprentissage de la syntaxe trop fastidieux.

2.3.1 Langage

Un état de l'art sur les langages utilisés montre que l'un d'entre eux offre les caractéristiques nécessaires et est très répandu dans la représentation d'ontologies. Il s'agit du langage OWL [2], fourni par le W3C [3] ([W3C, 2013]).

D'autres types de langages existent et permettent de représenter les ontologies sous des formats très variés. On peut citer, par exemple, RDF [4], OIL [5] ou

2. Web Ontology Language
3. World Wide Web Consortium
4. Resource Description Framework
5. Ontology Inference Layer

DAML+OIL [6].

RDF est un langage général (non limité aux ontologies), permettant de représenter des informations sur le Web. C'est une recommandation du W3C. Une de ses syntaxes bien connues est RDF/XML. Son but premier est de structurer les documents. Il s'accompagne du RDFS, qui est le schéma permettant de structurer les documents RDF.

OIL utilise les fonctionnalités de base de RDF/RDFS mais est destiné spécifiquement à la création d'ontologies. Cependant, le projet s'est terminé pour évoluer en DAML+OIL. Ce dernier ajoute au modèle une approche orientée objet pour décrire la structure d'un domaine. Le projet semble pourtant avoir été abandonné et le langage n'évolue plus depuis plusieurs années (la fin du projet datant de 2006). Il a clairement été dépassé par OWL.

OWL est celui qui est le plus utilisé dans la création d'ontologies et possède toutes les caractéristiques requises pour transcrire parfaitement celles-ci. Il est mis à jour régulièrement (sa deuxième version a été publiée en 2009 et mise à jour en 2012). De plus - cela sera exposé au point suivant -, de nombreux outils existent pour manipuler ce langage et il est facile de trouver de la documentation très complète dans le domaine.

Le langage OWL est une extension du RDF et du RDFS et les étend afin d'ajouter les possibilités de représentation qui manquent au RDF. Un "moteur d'inférence" peut être appliqué sur un document OWL. Celui-ci permet de raisonner sur des logiques de description afin d'inférer (donc de découvrir) des règles sous-jacentes à la représentation.

Il existe trois niveaux de langage OWL: OWL-Lite, OWL-DL et OWL-FULL. Le niveau de complexité augmente avec chacun des niveaux du langage.

OWL-Lite est la plus simple des trois, de complexité faible. Les requêtes sur un tel langage sont plus rapides au vu de sa simplicité.

OWL-DL est plus répandu, utilisant une logique décidable [7].

Finalement, OWL-Full est une version de OWL qui est indécidable. OWL-Full permet une expressivité maximale mais aucun moteur d'inférence ne peut y être appliqué.

OWL-DL est un sous-ensemble de OWL-Full qui est un bon choix pour la plupart des représentations, fournissant un bon rapport entre les possibilités du langage et son expressivité. Finalement, OWL-Lite est un sous-ensemble de OWL-DL et sert pour une représentation minimaliste du domaine.

Par exemple (mais ceci est un exemple simple), si la propriété suivante est déterminée : "Matthew hasSibling Gemma" et que la propriété "hasSibling" a été

6. DARPA Agent Markup Language

7. Dans notre cas, il s'agit d'une logique sur laquelle un ordinateur peut travailler afin de tirer des conclusions

identifiée comme "symétrique", le moteur inférera la relation suivante : "Gemma hasSibling Matthew". Dans les schémas de grandes dimensions, ces propriétés inférées automatiquement permettent une bien meilleure représentation du modèle.

Le langage OWL est donc apparu comme un choix judicieux dans le cadre de ce mémoire. La version OWL-DL est utilisée ici car elle est la plus adaptée à nos besoins.

2.3.2 Outil de manipulation

Une fois le langage choisi, l'étape suivante est bien sûr le choix de l'outil qui permettra de le manipuler. En effet, bien que le langage OWL utilise le XML, l'utilisation d'un outil adapté sera plus performante que l'apprentissage de la syntaxe du langage.

Il existe plusieurs programmes permettant la manipulation d'ontologies. Parmi ceux-ci, un des plus utilisés est "Protégé"
([Stanford Center for Biomedical Informatics Research, 2014]). Il n'est pas le plus ergonomique, mais il contrebalance ce désavantage par une forte personnalisation et une bonne communauté en ligne. Ces deux avantages ont prévalu dans le choix de l'outil. En effet, le premier permet, lors du matching d'ontologies (qui sera exposé dans le chapitre 3), d'utiliser des plugins permettant d'effectuer ce matching entre deux ontologies. Le deuxième avantage (la qualité du support sur le web), quant à lui, est un atout lors de l'apprentissage de l'outil et en cas de problème avec celui-ci. De fait, la documentation existante en ligne est importante et quelqu'un qui cherche à résoudre un problème rencontré trouvera facilement de l'aide en ligne pour contourner celui-ci.

De plus, comme "Protégé" utilise le langage Java, il est multi plate-forme et donc très portable. C'est un avantage supplémentaire puisque ce mémoire présente une méthodologie de construction d'un outil et que le travail de réalisation devra être effectué ultérieurement. De plus, ce programme est encore régulièrement mis à jour par l'université de Stanford et suit donc les évolutions du langage OWL. C'est également un atout pour la maintenance future de l'outil développé dans ce mémoire.

Le choix du programme pour réaliser cette étape s'est donc porté sur l'utilisation de "Protégé". La figure 2.4 présente l'interface de Protégé, sur l'onglet de gestion des classes. Le point suivant expose la méthodologie appliquée pour la transformation des standards en ontologies.

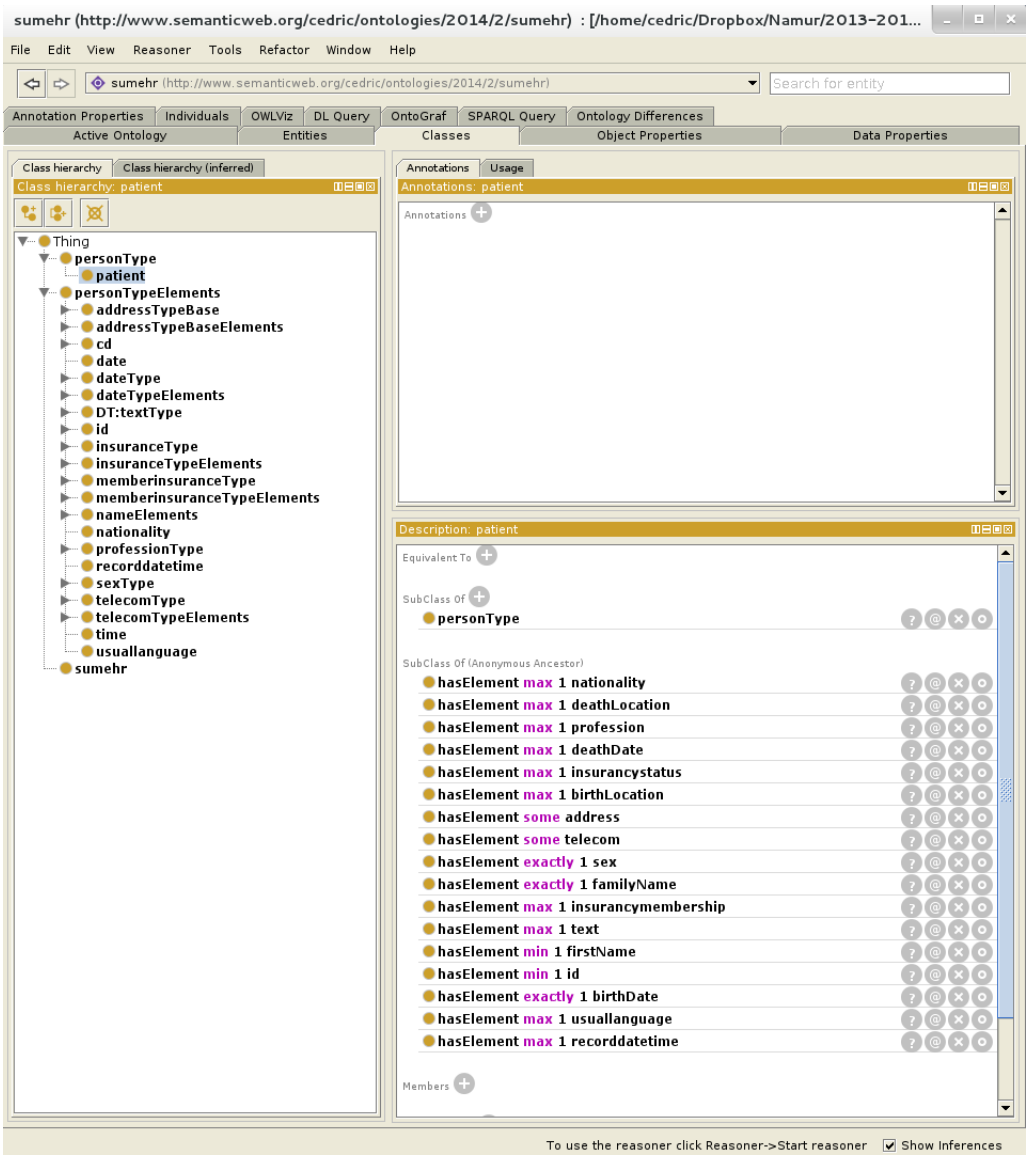

FIGURE 2.4 – Interface de Protégé,
[Stanford Center for Biomedical Informatics Research, 2014]

2.4 Méthodologie

Il est important de rappeler, avant toute chose, le but de la transformation en ontologies du SumEHR et du CDA. C'est le besoin d'un matching des concepts qui a dirigé la méthode de travail. La première étape doit donc permettre d'exprimer les concepts réprésentés par les standards afin que ceux-ci puissent être liés lorsque leur sémantique est identique. Cette deuxième étape (le matching,

qui sera étudié dans le prochain chapitre) permettra de structurer la création de l'outil de conversion d'un standard vers l'autre, but final de ce mémoire.

Dès lors, bien que l'analyse puisse être extrêmement complète et refléter parfaitement tous les concepts et structures fournis par les standards, elle ne sera pas utilisée si elle ne permet pas d'effectuer un matching entre ces concepts. De plus, toute information supplémentaire inutile au matching n'apporte rien et fait perdre du temps dans la construction de l'outil de conversion. Cela, uniquement du point de vue du but poursuivi, bien entendu, étant donné qu'une ontologie complète de ces standards pourrait se révéler extrêmement utile dans d'autres cas. Il existe d'ailleurs plusieurs projets qui étudient actuellement la potentialité de réaliser une interopérabilité entre des standards uniquement à l'aide d'ontologies (voir par exemple [Thuy et al., 2011]) - mais cela dépasse le cadre de ce mémoire.

Au vu du but poursuivi, seuls deux concepts principaux des ontologies doivent être utilisés dans cette transformation : les classes et les propriétés.

Les premières définissent les concepts sémantiques. Elles peuvent contenir des individus (dans notre cas, ceux-ci seraient des données), mais il n'est pas nécessaire de peupler les classes avec des individus dans le cadre de ce mémoire.

Ces classes peuvent également être liées par des relations ("Object Properties"). Celles-ci seront utilisées afin de reproduire la structure en arbre fournie par le schéma XML.

Les deuxièmes, les propriétés, vont servir à reproduire la structure en arbre des schémas XML.

Il existe bien d'autres concepts fournis par les ontologies qui, s'ils apportent un complément d'information, ne se révèlent pas utiles ici. Par exemple, la figure 2.5 montre le mapping d'un schéma XML vers un fichier OWL dans [Lacoste et al., 2011]. Bien entendu, ce type de transformation fournit une ontologie très complète, cohérente et permettant des analyses plus conséquentes des concepts utilisés, mais elle ne se prête pas du tout à un matching entre les concepts.

En effet, le chapitre suivant montre que seules les classes se révèlent utiles lors du matching.

XSD	OWL
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
named xsd:ComplexType	owl:Class
named xsd:SimpleType	owl:DatatypeProperties
xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sequence, xsd:all	owl:intersectionOf

FIGURE 2.5 – Mapping d'un schéma XML en OWL, [Lacoste et al., 2011]

Bien entendu, il ne s'agit pas seulement de décider quels éléments seront utilisés, il faut également formaliser la méthode avec laquelle les standards seront retranscrits en ontologies, afin de permettre une vérification de l'ensemble, de garder une cohérence au cours du travail et de permettre une implémentation du convertisseur KMHER - HL7 par différentes personnes.

2.4.1 Règles méthodologiques

Cette section décrit la méthode développée dans ce mémoire. Celle-ci est exprimée à la table 2.1. Ce tableau est un résumé condensé des règles de travail. Chaque transformation est explicitée aux sections 2.4.2 à 2.4.9. Elles sont identifiées par leur numéro dans la première colonne du tableau.

La méthode développée ici est inspirée de [Lacoste et al., 2011] (voir figure 2.5) et de [Bohring and Auer, 2005] qui sont deux transformations similaires. Elle est cependant, comme expliqué précédemment, bien plus réduite et modifiée par rapport à l'originale, étant donné les besoins actuels. Par exemple, les éléments "xsd:SimpleType" et "xsd:elements" (sans attributs ou sous-éléments) deviennent des "owl:DatatypeProperties" selon les paramètres exprimés à la figure 2.5. Cette méthode ne convient pas à l'analyse effectuée ici étant donné que tous les éléments doivent devenir des classes (owl:Class) afin de pouvoir être utilisés au chapitre 3.

Ces méthodes ayant servi de base à la méthode expliquée ici fournissent donc des ontologies trop complexes pour ce mémoire.

Il faut également noter qu'il existe actuellement des outils permettant d'obtenir automatiquement une ontologie à partir d'un schéma XML. Ce point sera abordé au chapitre 5

N°	Éléments du XSD	Éléments équivalents de l'ontologie
1	Types complexes	Classes
2	Noms des types complexes	Sous-classe de la classe représentant le type complexe
3	Éléments appartenant à des types complexes (et non utilisés dans d'autres types complexes) ou Éléments appartenant à des types complexes et utilisés dans d'autres types complexes qui sont des sous-classes	Sous-classes d'une classe nommée [nom de la classe représentant le type complexe]Elements
4	Éléments appartenant à des types complexes et utilisés dans d'autres types complexes n'étant pas des sous-classes	Les éléments partagés sont placés dans les éléments de la première super classe commune
5	Types complexes ne contenant qu'un seul type simple ou Types complexes ne contenant que des attributs et des "restriction base" ou "extension base"	Classes (du nom des éléments et non de leurs types) sans sous-classes
6	Attributs	Non représentés
7	Éléments de noms identiques qui ne contiennent pas les mêmes types de données	Classes avec, en suffixe, le type complexe de l'élément entouré de crochets
8	Relation parent-enfant dans l'arbre du XSD	Propriété d'objet "hasElement"

TABLE 2.1 – Tableau des règles de transformation

2.4.2 Règle de transformation 1

La première règle de transformation est la plus simple mais primordiale. Elle spécifie que les types complexes seront transformés en classes. Rappelons que

les types complexes sont des types XML composés (ils définissent des éléments).

Il faut porter une attention particulière à la différence entre un type complexe et le nom de celui-ci. Par exemple, dans un schéma XML, prenons un type complexe “MonTypePatient” :

```
1  <xsd:complexType name="MonTypePatient">
2    <xsd:sequence>
3      <xsd:element name="premierElement" type="String"/>
4      <xsd:element name="deuxiemeElement" type="String"/>
5    </xsd:sequence>
6  </xsd:complexType>
7  %
```

On peut voir que le type complexe définit simplement les éléments qu’il contient.

Il est à distinguer de son utilisation. Par exemple, dans une autre partie du même schéma XML, on pourra lire ceci :

```
1 <xsd:element name="MonEntitePatient" type="MonTypePatient
    "/>
```

Ce code signifie que, dans le document XML créé suivant les règles de syntaxe de ce schéma XML, on pourra trouver un élément “MonEntitePatient” :

```
1  <MonEntitePatient>
2    <premierElement>UnTexteQuelconque</premierElement>
3    <deuxiemeElement>UnAutreTexte</deuxiemeElement>
4  </MonEntitePatient>
```

Grâce au schéma XML, nous savons que “MonEntitePatient” est un élément, de type complexe “MonTypePatient”, contenant deux éléments (“premierElement” et “deuxiemeElement”). Pourtant, dans le fichier XML, il n’est marqué nulle part que “MonEntitePatient” est un type complexe - seul le schéma XML l’indique.

Cette règle ne concerne que les types complexes, pas les noms qui sont utilisés pour les représenter. Dans ce cas, il s’agit de “MonTypePatient”.

Exemple : Le type d’assurance est un type complexe : il contient plusieurs éléments (siscard, socialfranchiseperiod, etc.). Le type complexe “insuranceType” deviendra donc une classe “insuranceType”.

2.4.3 Règle de transformation 2

La deuxième règle associe les éléments de types complexes avec les noms des éléments qui les implémentent, en créant des sous-classes des classes qui représentent les types complexes.

Pour faire le lien avec la première règle, on traite ici “MonEntitePatient” et non plus “MonTypePatient”.

Exemple : l’assurance maladie du patient, nommée “insurancystatus”, est du type “insuranceType”. Dès lors, la classe “insurancystatus” sera une sous-classe de la classe “insuranceType”. Celle-ci contient deux sous-classes, “insurancystatus” et “hospitalizationinsurance”, qui sont tous deux des assurances de ce même type.

La figure 2.6 montre le résultat de la transformation sur l’entité adresse. On peut observer la classe “addressTypeBase” qui représente le type complexe. La classe “address”, qui est de ce type, est une sous-classe de celle-ci. Pour rappel, l’élément “addressTypeBase” est la définition d’un type complexe, tandis que “address” est le nom d’un élément de ce type. On peut également observer les éléments “birthLocation” et “deathLocation”, tous deux des sous-classes de “addressTypeBase” étant donné qu’ils sont également du même type (une adresse, donc).

FIGURE 2.6 – Transformation en ontologie de l’entité “insurancystatus”

2.4.4 Règle de transformation 3

La troisième règle sert à la structuration des ontologies. Si les ontologies étaient réalisées sans cette règle, celles-ci seraient toutes au même niveau logique (autrement dit, toutes regroupées au même endroit sans être imbriquées les unes dans les autres). Au vu du nombre d’éléments présents dans l’ontologie, celle-ci en deviendrait illisible. Il y aurait une grande quantité d’éléments se trouvant tous les uns à côté des autres, sans structure logique apparente. Ce genre de structure en bloc rend la lecture de l’ontologie fastidieuse et désagréable.

C’est ici un regroupement logique des éléments qui est effectué, en fonction de leur utilisation. De ce fait, les éléments seront regroupés dans une même classe, en fonction de leur contexte d’utilisation, afin de clarifier la lecture de l’ontologie.

Autrement dit, cette règle regroupe les éléments qui servent au même endroit. Les éléments contenus dans un type complexe, afin d’être associés à celui-ci, sont regroupés en sous-classes d’une seule et même classe, qui représente “les éléments contenus dans ce type complexe”.

Exemple : les éléments “socialfranchiseperiod”, “approvalnumber” et “personalpart” qui servent dans l’élément “insurancystatus”, de type “insuranceType”,

seront regroupés en sous-classes de la classe “insuranceTypeElements” (à condition qu’ils ne soient pas utilisés ailleurs, comme le montre la règle suivante).

La figure 2.7 représente cette situation. On peut y observer l’élément “insurancystatus”, de type “insuranceType”. Celui-ci contient plusieurs éléments, regroupés dans la classe “insuranceTypeElements”.

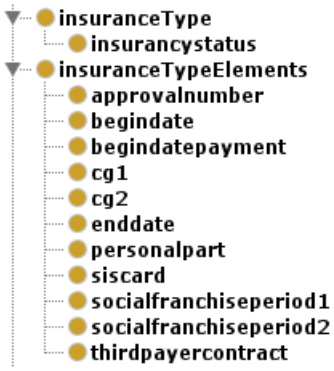

FIGURE 2.7 – Éléments de l’entité “insuranceType”

2.4.5 Règle de transformation 4

La quatrième règle permet de traiter les éléments qui sont utilisés à plusieurs endroits. Par exemple, un “ID”, qui sert à plusieurs endroits (celui du document, du patient et du médecin), ne peut pas être placé comme un sous-élément du document, du patient et du médecin. Il faut donc le placer à l’endroit le plus logique : dans la première classe commune. Il sera donc placé en sous-élément du document, puisque le patient et le médecin sont tous deux des sous-éléments du document.

On pourrait bien sûr vouloir faire autrement : si un “ID” sert au document, au patient et au médecin, pourquoi ne pas le mettre trois fois dans l’ontologie, dans “documentElements”, “patientElements” et “medecinElements” ? En réalité, il y a deux contre-arguements à cette méthode : premièrement, cela encombrerait inutilement l’ontologie et la rendrait moins lisible au vu de sa taille. De plus, cela est impossible. Par principe, une ontologie en OWL ne peut contenir deux éléments du même nom, étant donné que, dans une ontologie, deux classes ayant le même nom sont en fait une seule et même classe.

Dès lors, il faut obligatoirement placer un et un seul élément “ID”. Le choix a été de le placer dans l’endroit “le plus élevé” de l’ontologie.

Exemple : l’élément “Name”, utilisé par l’élément “Patient” et par l’élément “MédecinTraitant”, sera placé comme une sous-classe de l’élément “Header” qui les contient tous les deux.

2.4.6 Règle de transformation 5

La cinquième règle permet d'alléger le contenu de l'ontologie en supprimant des éléments qui n'apportent rien à la sémantique, lorsqu'ils ne contiennent que des éléments de type simple. Les types simples sont des types de base fournis par le XML : les strings, les entiers, les dates, etc. Puisque traiter ces éléments comme des types complexes ne ferait que surcharger l'ontologie inutilement, ceux-ci sont représentés par une simple classe au lieu de créer une classe pour le type et une classe pour le nom de l'élément qui l'implémente.

Jusqu'ici, le principe pour le traitement d'un type complexe était d'avoir le type, le nom de l'élément qui l'implémente en sous-classe et ses éléments dans une autre classe.

Mais dans le cas où l'élément ne contient quasiment aucune information (uniquement des attributs donnant la valeur, ou un unique type simple contenu dans le type complexe), la construction est inutile : la classe qui représente les éléments contenus dans le type complexe est alors vide ! Elle n'a dès lors aucune raison d'exister. De même, le type complexe rajoute une classe qui n'apporte rien au sens sémantique. La construction est donc inutile, et il est plus raisonnable de se contenter du nom de l'élément.

Exemple : l'élément "StreetLine" est de type complexe "StreetLineDetail" qui ne contient qu'un seul élément, un string. Alors qu'une classe "StreetLineDetail" aurait dû être créée avec "StreetLine" comme sous-classe, nous nous contentons d'ajouter la classe "StreetLine" directement, sans super classe.

2.4.7 Règle de transformation 6

Cette règle simple, dans la même veine que la précédente, allège l'ontologie. Les attributs étant rarement porteurs de sens du point de vue des concepts, ils ne sont pas ajoutés à l'ontologie.

En effet, cette étape de transformation des schémas XML en ontologies sert à extraire les concepts présents dans ceux-ci. Les attributs ne contiennent, en KMEHR et en HL7, que des informations d'implémentation : la valeur, le type, la période, etc. Du point de vue de l'ontologie qui est nécessaire ici, ces éléments sont des détails d'implémentation inutiles aux concepts. Ils ne sont donc pas repris dans l'ontologie.

Exemple : les attributs "UseablePeriod" et "Type" de l'élément "Address" ne seront pas intégrés à l'ontologie.

2.4.8 Règle de transformation 7

La septième règle permet de traiter les éléments de même nom. Il existe dans le standard plusieurs éléments dont le nom est identique mais pas le type. On peut citer le cas où l'élément représentant le nom du document et celui du nom du patient s'appellent tous deux "Name". Le premier sera de type "DocumentName" et le deuxième de type "PatientName". Ces deux éléments, selon les règles édictées plus haut, devraient avoir chacun le nom "Name" mais être traités de manière différente. Or, cela est impossible dans le langage OWL : il n'autorise pas l'existence de deux classes du même nom. Dès lors, il faut pouvoir modifier le nom de chacune des deux tout en conservant leur nom original et en y ajoutant de l'information.

La solution choisie ici est d'ajouter le type de l'élément entre crochets après son nom. Les crochets n'étant pas utilisés en KMEHR et en HL7, il n'y a pas de risque de confusion avec le nom d'un élément - les crochets sont obligatoirement des informations supplémentaires lorsqu'ils sont rencontrés dans l'ontologie. Cette technique permet de garder le nom de l'élément intact tout en apportant l'information supplémentaire concernant son type différent.

Les deux figures 2.8 et 2.9 montrent la situation, en HL7, où les deux éléments portent le même nom ("name"). Ceux-ci se voient ajouter leur type entre crochets.

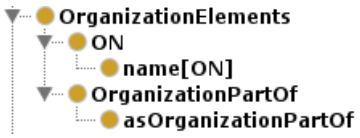

FIGURE 2.8 – Élément "name" - OrganizationElements

FIGURE 2.9 – Élément "name" - CRElements

Exemple : L'élément "Name" de type "DocumentName" devient "Name[DocumentName]" dans l'ontologie.

2.4.9 Règle de transformation 8

Cette huitième et dernière règle se contente de transcrire le lien de parenté du schéma XML : comme mentionné à la section 2.2, il est nécessaire de pouvoir retranscrire la forme en arbre du schéma XML. Celui-ci se retrouve dans l'ontologie au niveau de la propriété d'objet nommée "hasElement".

Exemple : L'élément "Address" est contenu dans l'élément "Patient". Dans l'ontologie, on ajoutera la règle "Patient hasElement Address". La règle devra bien sûr respecter la cardinalité exprimée dans le schéma XML.

En ce qui concerne les cardinalités en OWL, elles s'expriment de manière relativement différente de celles des schémas XML. En OWL, les cardinalités d'une relations sont exprimées par un attribut : "some", "only", "min", "max" et "exactly".

La table 2.2 montre la correspondance entre les cardinalités en OWL et celles des schémas XML.

Cardinalité du XSD	Cardinalité en OWL
0..∞	Some
1..∞	Only*
X..∞	Min X
0..X	Max X
X..X	Exactly X

TABLE 2.2 – Tableau des règles de cardinalités OWL - XSD

*: la règle "Only" est un peu différente d'une simple cardinalité puisqu'elle impose une restriction sur la propriété. Par exemple, la relation "Patient hasElement only address", cela signifie que l'élément Patient ne peut pas avoir une relation "hasElement" avec autre chose qu'un élément "address". Ne permettant pas d'exprimer correctement une cardinalité puisqu'elle impose cette restriction, elle n'est pas utilisée dans l'ontologie créée ici. On lui préférera la cardinalité "Min 1".

2.5 Résultats obtenus

Cette première partie d'analyse a été la partie la plus fastidieuse du mémoire étant donné la difficulté de trouver une méthodologie qui permettait de faire ressortir l'information essentielle dans les ontologies tout en éliminant les éléments inutiles et en restant systématique.

Pour rappel, le travail effectué dans ce chapitre se base sur [Lacoste et al., 2011] et [Bohring and Auer, 2005]. Ceux-ci ne sont pas les seuls projets existants, mais toutes les recherches en cours actuellement ont pour but de créer une ontologie générale et non spécifique au matching. C'est pour cette raison qu'elles ne convenaient pas aux besoins exprimés à ce stade du mémoire. Dans la même veine, les outils de création automatique d'ontologies ne convenaient pas puisqu'ils suivaient les mêmes méthodes. La détermination des règles nécessaires a

donc demandé beaucoup de documentation (voir notamment [Horridge, 2011]) et d'essais.

Plusieurs versions préalables de la modélisation ont dû être écartées car elles n'étaient pas appropriées. À part pour quelques règles évidentes, la plupart des règles énoncées ont été ajoutées avec chaque nouveau problème rencontré, celui-ci signifiant souvent un remaniement de l'ontologie.

Les tests d'implémentation ayant été réalisés uniquement sur l'entité "patient" du KMEHR, il sera peut-être nécessaire d'adapter légèrement les règles de transformation en fonction de situations rencontrées dans d'autres sections des XSD.

Les ontologies du KMEHR et du HL7 se trouvent en annexe sur le CD-ROM. La figure 2.10 montre le rendu de la partie du XSD représentant l'assurance médicale en KMEHR. L'image a été réalisée à l'aide du programme "<oXygen/> XML editor"[8]. Les deux figures 2.11 et 2.12 représentent l'ontologie du même élément "assurance" et le détail de ses relations.

On peut voir que l'ontologie est plus allégée en termes de concepts. Elle permet donc de se concentrer sur les concepts sans être gêné des détails d'implémentation, ce qui est le but visé.

Comme mentionné précédemment, le test de la méthode se fait sur l'entité "patient" du SumEHR. L'entièreté de la méthode exposée dans ce mémoire sera testée sur celle-ci. Celle-ci ne représente qu'une petite partie du SumEHR et du CDA, mais elle permet de vérifier la cohérence de la méthode proposée ici. Appliquer la méthode sur l'entièreté du SumEHR dépasse la portée de ce mémoire.

La figure 2.13 présente l'entité patient dans le schéma global d'un message KMEHR. L'annexe B présente les classes des ontologies réalisées (graphiquement, le format texte étant trop important).

8. `http://www.oxygenxml.com/`, Mise en ligne : 1/10/2002, Consultation : 23/05/2014 18:31

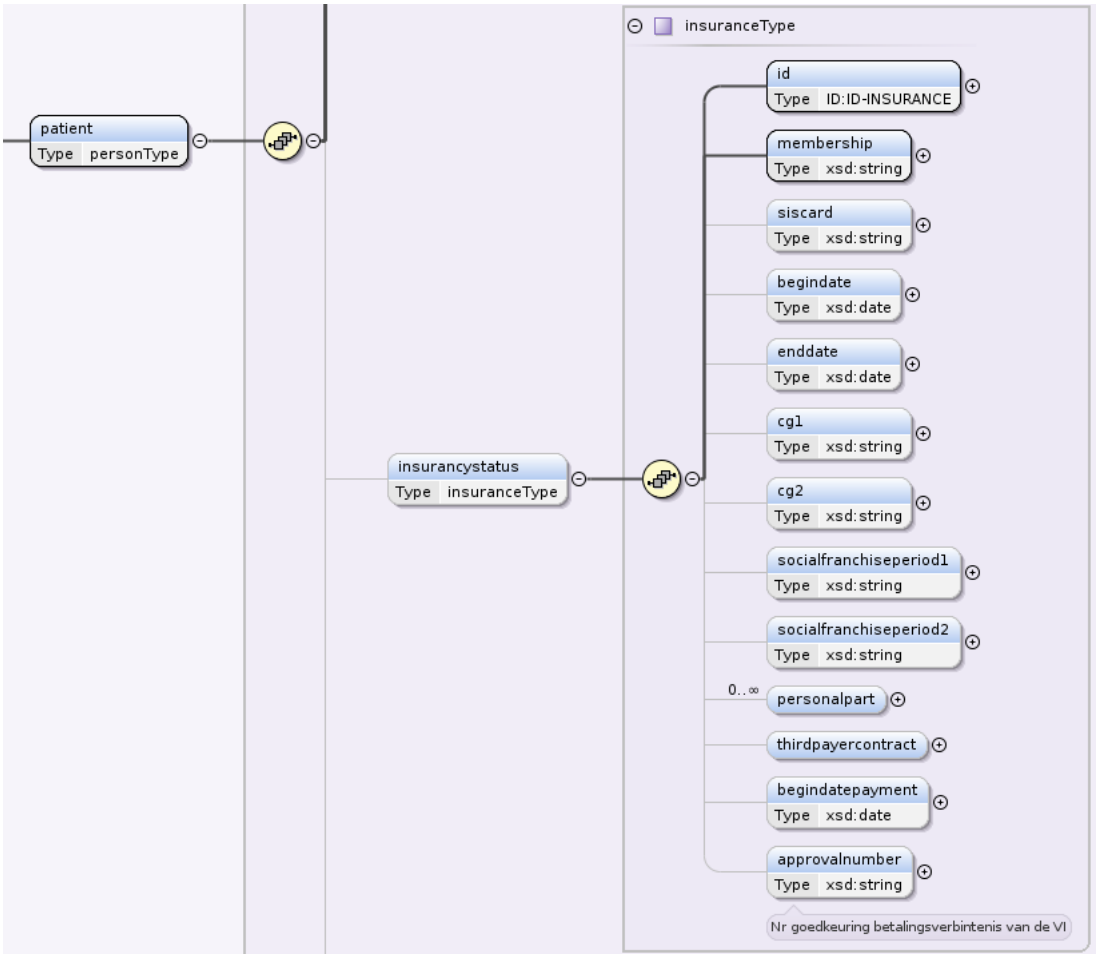

FIGURE 2.10 – représentation de l'entité assurance dans le schéma XML du KMEHR

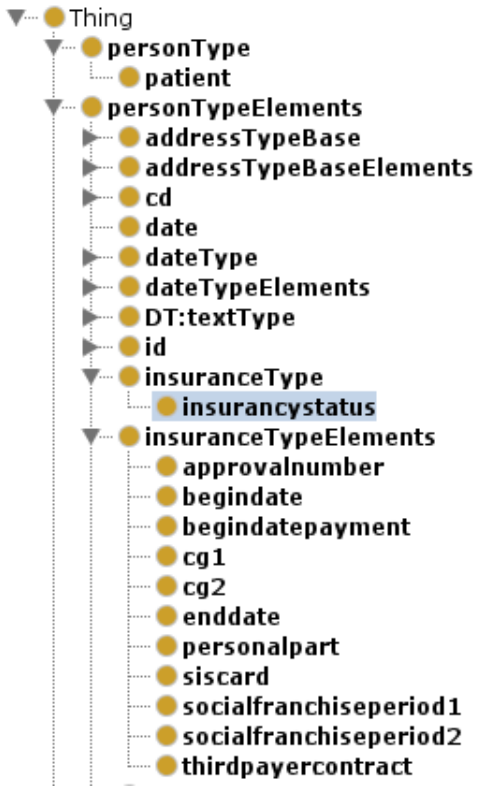

FIGURE 2.11 – représentation de l'entité assurance dans l'ontologie du KMEHR

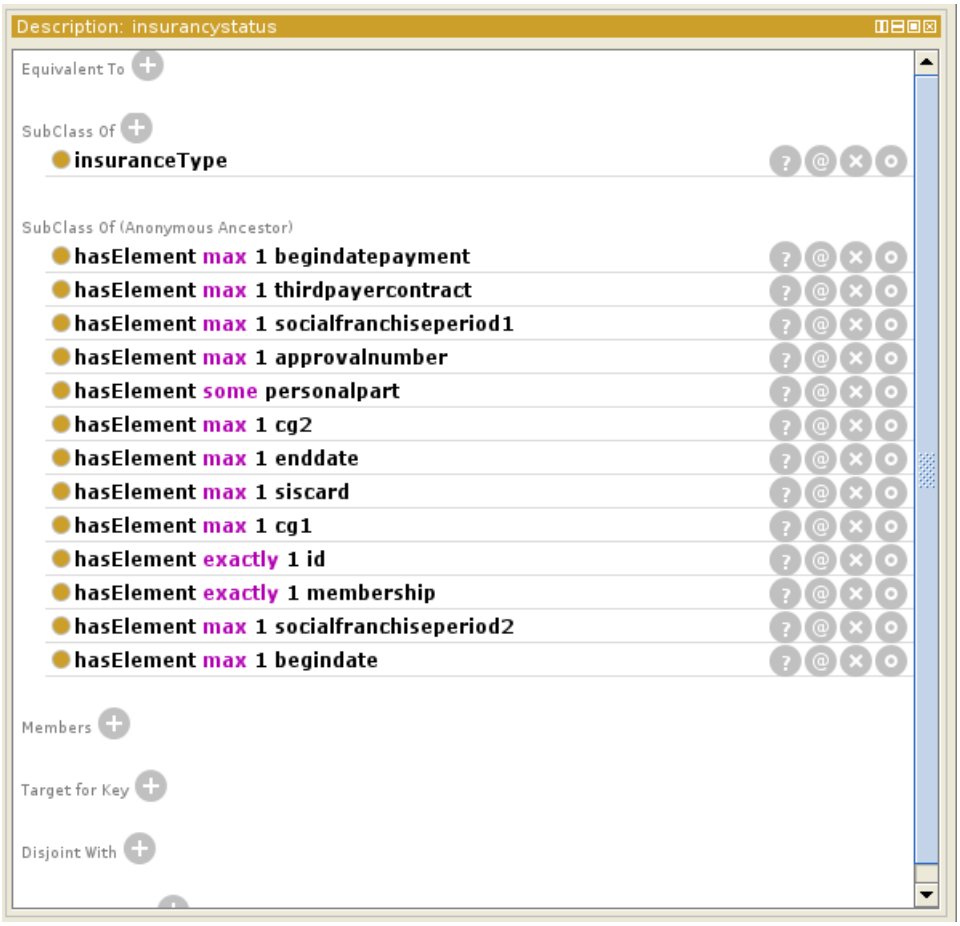

FIGURE 2.12 – Détail de la classe "insurancymembership" dans l'ontologie du KMEHR

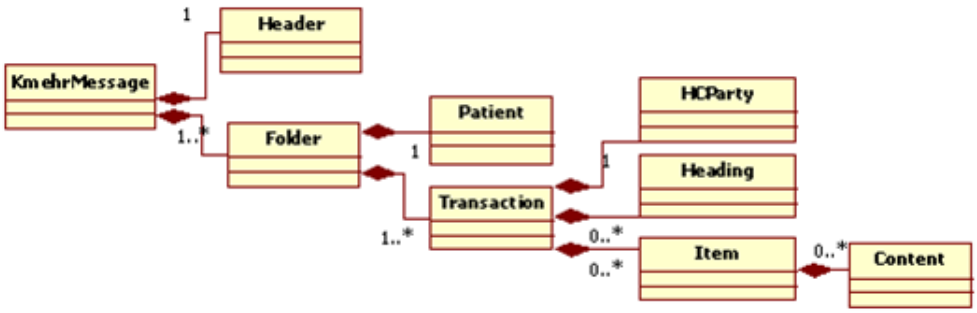

FIGURE 2.13 – Schéma d'un message KMEHR - Source: [eHealth, 2014]

2.6 Limites

Comme cela a été mentionné précédemment, ces ontologies ne représentent pas entièrement les standards étudiés. De fait, alors que le modèle par ontologie fournit un grand nombre d'outils et de nuances, nous nous contentons ici de transformer les éléments du schéma XML en classes. Nous utilisons une seule classe d'objet afin de conserver le système en arbre fourni par le schéma.

Cette approche ne donne pas une ontologie complète et optimale des standards. Elle fournit uniquement une structure minimaliste. Étant donné le but final du mémoire, cette ontologie est suffisante à l'interprétation des standards, raison pour laquelle elle est utilisée de la sorte. Cependant, il serait intéressant de considérer la modification des règles pour étoffer les ontologies. On pourrait également penser à l'automatisation de la transformation du schéma XML en ontologie - comme mentionné à la sous-section 2.4.1, de tels outils existent déjà mais sont inadaptés. Ces améliorations possibles seront détaillées dans le chapitre 5.

Finalement, le fait d'utiliser un modèle qui perd de l'information limitera les possibilités d'automatisation pour ce qui est de la traduction du KMEHR au HL7. Ce point sera détaillé dans le chapitre 4

Chapitre 3

Matching d'ontologies

Ce chapitre introduit tout d'abord les raisons du matching qui sera réalisé et les outils choisis pour ce faire. Il présente ensuite les choix d'implémentation effectués pour sa réalisation et termine par les résultats obtenus à cette étape du travail.

3.1 Identification des besoins

Ce chapitre reste au niveau d'abstraction supérieur que fournissent les ontologies. Rappelons que le travail effectué ici s'effectue sur trois niveaux.

Le premier est celui du XML. Le XML délimite les données d'un message à l'aide de balises. Il s'agit uniquement d'information brute, structurée selon des règles établies par le standard. Les documents XML servent donc à transporter des données. Quelques fichiers XML de test seront utilisés dans ce mémoire pour vérifier l'outil de conversion qui sera réalisé au chapitre 4. L'outil développé devra être capable de convertir un tel fichier du format KMEHR au format HL7, tout en conservant l'information à l'identique, autant que possible, grâce à des règles de traduction précises. C'est le niveau des "données".

Le deuxième niveau est celui des schémas XML, ou XSD. Ceux-ci, comme leur nom l'indique, sont les schémas décrivant la structure d'un fichier XML. Autrement dit, ils définissent à quels endroits peuvent être mis certains éléments. Il faut rappeler que ces schémas sont également écrits en XML, mais servent à définir un modèle à respecter pour d'autres fichiers XML. La figure 2.10 au chapitre précédent montrait un modèle imagé de ce qu'est un schéma XML, avec les éléments, leurs types, leurs cardinalités et leurs structures. Dans la méthodologie créée ici, ce sont ces schémas XML qui, dans chaque standard, sont utilisés pour obtenir les informations sur la structure des documents. Ils

ne contiennent pas la signification des éléments [1]. Mis en parallèle avec la documentation du standard, ils permettent la mise en place des messages. C'est le niveau de la "syntaxe".

Le troisième et dernier niveau, le plus élevé dans l'abstraction, est celui des ontologies. Celles-ci permettent, comme cela a été vu au chapitre précédent, de passer d'un modèle syntaxique à un modèle conceptuel, en ajoutant du sens à celui-ci. Nous ne nous attarderons pas sur leur description étant donné qu'elle a été faite précédemment, mais soulignons qu'elles permettent de refléter la sémantique en filigrane dans les schémas XML. C'est le niveau des "concepts".

Ce chapitre établit le lien entre le KMEHR et le HL7. Ayant élevé ceux-ci au niveau supérieur, de la syntaxe aux concepts, nous pouvons à présent aisément relier les concepts similaires d'un côté à l'autre. Le fait d'avoir utilisé des ontologies facilite grandement la tâche, étant donné que nous nous sommes débarrassés des spécificités de l'implémentation. Nous pouvons donc nous concentrer sur les principes exprimés par les standards, plus faciles à lier.

Pour cette étape, il est nécessaire d'utiliser un programme permettant d'effectuer un matching des ontologies. Celui-ci doit répondre à plusieurs critères.

Le premier critère, assez trivial, est une question de formats de fichiers. Le programme requis doit pouvoir utiliser en entrée les formats des fichiers générés par le programme "Protégé" choisi pour représenter les ontologies. En effet, plusieurs formats existent (RDF/XML, OWL/XML, OWL Functional Syntax, Manchester OWL Syntax, Turtle,...) et ne sont pas équivalents. "Protégé" permet l'enregistrement dans plusieurs formats de fichiers, mais pas l'entièreté. Il faut donc trouver un programme qui accepte en entrée au moins un des formats d'enregistrement de "Protégé". Il faut cependant souligner que le format RDF/XML est de nos jours le plus utilisé dans la représentation d'ontologies.

Le deuxième critère est un matching automatique. Celui-ci n'est pas obligatoire - un programme proposant uniquement un matching manuel peut être utilisé - mais il permet d'effectuer rapidement une bonne partie du travail. C'est donc un atout important.

Le troisième critère découle du précédent : le programme sélectionné doit permettre une modification manuelle du matching créé. Cela sera exposé à la section 3.2: certains outils proposent un matching entièrement automatique et n'ont pas d'interface destinée à la modification des résultats. Ce point est important puisque les concepts n'ont pas forcément le même nom d'un standard à l'autre et qu'un processus automatisé ne pourra pas les lier les uns aux autres correctement.

1. Ce point est discutable étant donné qu'il est possible de mettre des commentaires sur les éléments d'un XSD, mais les fichiers utilisés ici ne sont pas concernés.

3.2 Choix des outils

Une bonne partie des outils étudiés ici sont référencés sur le site "OntologyMatching.org" ([Shvaiko and Euzenat, 2005]), qui est une référence dans le milieu. Malheureusement, beaucoup de ces outils font partie de projets aujourd'hui abandonnés ou ne sont même plus disponibles. La table 3.1 présente un résumé des différents outils qui ont été considérés dans ce mémoire.

Bien que la tâche de sélection puisse paraître simple, elle a demandé énormément de travail : chaque outil a été testé (lorsqu'il était encore possible de le télécharger) afin de vérifier s'il répondait aux critères requis. La plupart des programmes marqués dans la table 3.1 comme "non fonctionnel" ont été analysés afin de tenter de les faire fonctionner (parfois avec succès). La plupart de ces projets étant réalisés en Java, les nombreux projets qui ne se lançaient pas ont été contrôlés pour tenter de corriger l'erreur du démarrage.

Étant donné l'importance du travail, la sélection d'un outil optimal pour le matching d'ontologies a représenté une charge très importante, la deuxième en ordre d'importance après la création d'une méthodologie formelle au chapitre 2.

TABLE 3.1 – Tableau des outils de matching

Nom	URL	Caractéristiques
AMW	`http://www.eclipse.org/gmt/amw/`	Non spécifique aux ontologies, interface peu adaptée au matching
Anchor-Flood	`http://www.kde.ics.tut.ac.jp/~hanif/res/anchor_flood.zip`	Projet abandonné, ne permet pas de travailler sur des fichiers locaux
AUTOMS	`http://www.icsd.aegean.gr/ai-lab/projects/AUTOMS/`	Projet supprimé
COMA 3.0	`http://dbs.uni-leipzig.de/Research/coma.html`	Fonctionne sous Windows uniquement, interface graphique de qualité, mapping automatique & manuel
CROSI	`http://www.aktors.org/crosi/`	Projet supprimé
CtxMatch	`http://dit.unitn.it/~zanobini/downloads.html`	Projet abandonné, non fonctionnel
eTuner/iMap-/Glue/LSD	`http://pages.cs.wisc.edu/~anhai/projects/schema-matching.html`	Non disponible au téléchargement

Falcon-AO	http://xobjects.seu.edu.cn/project/falcon/	Projet non fonctionnel
HMatch	http://islab.dico.unimi.it/hmatch/	Plugin pour "Protégé", non mis à jour (destiné à une très ancienne version de "Protégé"), non fonctionnel
KitAMO	http://www.ida.liu.se/~iislab/projects/KitAMO/	Non disponible en ligne
LOM	http://reliant.teknowledge.com/DAML/	Projet supprimé
Malasco	http://sourceforge.net/projects/malasco/	Projet non fonctionnel
Maponto	http://www.cs.toronto.edu/semanticweb/maponto/index.html	Plugin pour "Protégé", non mis à jour (destiné à une très ancienne version de "Protégé"), non fonctionnel
MetaQuerier	http://metaquerier.cs.uiuc.edu/	Projet non spécifique aux ontologies, non disponible en ligne
OLA	http://ola.gforge.inria.fr/	Projet ne permettant pas d'effectuer un mapping manuel
Optima	http://cobweb.cs.uga.edu/~uthayasa/Optima/Optima.html	Projet ne permettant pas d'effectuer un mapping manuel
PARIS	http://webdam.inria.fr/paris/	Projet ne permettant pas d'effectuer un mapping manuel
PJMapping-Tab	http://www.csd.abdn.ac.uk/~dcorsar/software/PJMappingTab/index.php	Plugin pour "Protégé", projet abandonné, non fonctionnel
PRIOR+	http://www.sis.pitt.edu/~mingmao/om07/index.html	Projet supprimé
RiMOM	http://keg.cs.tsinghua.edu.cn/project/RiMOM/	Projet ne permettant pas d'effectuer un mapping manuel

S-Match	`http://semanticmatching.org/`	Projet non fonctionnel (fonctionne, mais ne charge pas les fichiers OWL correctement)
SAMBO	`http://www.ida.liu.se/~iislab/projects/SAMBO/`	Non disponible en ligne
Similarity Flooding	`http://www-db.stanford.edu/~melnik/mm/sfa/`	Projet en développement, fonctionnel mais interface très peu user-friendly
SOMER	`http://somer.fc.ul.pt/`	Projet ne permettant pas d'effectuer un mapping manuel, interface peu user-friendly
ToMAS/Clio	`http://www.cs.toronto.edu/db/clio/`	Projet supprimé

Rappelons les critères de choix du programme, présentés au point précédent :

- Programme de matching d'ontologies.
- Capable de lire un des formats de fichiers fournis par "Protégé", si possible le RDF/XML.
- Possibilité de matching manuel.
- Optionnellement, matching automatique.

Après l'étude des différents programmes, il s'est rapidement avéré que le programme "COMA 3.0" ([Rahm et al., 2006]) était le plus adapté aux besoins de ce chapitre, en raison de la rareté de ce genre de programmes et du manque de suivi des projets qui sont souvent dépassés et ne fonctionnent plus.

Cette recherche n'a pas la prétention d'être exhaustive et il existe encore d'autres programmes qui n'ont pas été testés. De plus, d'autres projets en cours de développement répondront peut-être mieux, dans le futur, aux besoins du matching effectué ici. On peut citer, par exemple, KitAMO et SAMBO, mentionnés à la table 3.1 comme "non disponibles en ligne" - ils sont encore en projet.

Les sections suivantes présentent certains des programmes qui ont été analysés lors de la recherche. L'ensemble n'est pas présenté ici mais uniquement un échantillon représentatif des types de problèmes rencontrés.

3.2.1 KitAMO

Le programme KitAMO [2], pour ToolKit for Aligning and Merging Ontologies, est un exemple d'un projet référencé par [Shvaiko and Euzenat, 2005] qui est en cours de développement et ne permet pas le téléchargement de son programme.

Ce genre de site web de projet ne fournit en général que quelques publications, des informations sur le projet et sur leurs responsables. Ils ne sont donc pas utilisables dans ce mémoire.

3.2.2 Anchor

Ce projet n'est plus mis à jour depuis 2009. Ce n'est pas forcément un problème, mais cela signifie qu'il n'y aura plus d'évolution.

La documentation fournit les instructions suivantes :

```
1. For Instance Matching (benchmark)
   aflood -im <tbox_url1> <abox_url1> <tbox_url2> <abox_url2> <output_url>
OR
   aflood -im <atbox_url1> <atbox_url2> <output_url>
2. For Anatomy:
2a. Default Operation: Subtask #1
   aflood -omp <ref_url> <target_url> <output_url>
2b. Anatomy Subtask #2
   aflood -oa2 <ref_url> <target_url> <output_url>
2c. Anatomy Subtask #3
   aflood -oa3 <ref_url> <target_url> <output_url>
2b. Anatomy Subtask #4
   aflood -oa4 <ref_url> <target_url> <ref_align_url> <output_url>
3. For Benchmark, Conference, Directory or other general pair of ontologies
   aflood -omp <ref_url> <target_url> <output_url>
4. For evaluation
   aflood -eval <ref_align_url> <user_align_url>
```

Il faut noter deux informations fournies par cette documentation.

Premièrement, le programme ne permet d'utiliser que des URLs, ce qui ne convient pas pour le matching souhaité.

Deuxièmement, il doit obligatoirement se lancer en ligne de commande, ce qui est très peu agréable pour l'utilisateur. Il a donc été écarté.

2. `http://www.ida.liu.se/~iislab/projects/Ontologies/publications.html`, Mise en ligne : 21/02/2006, Consultation : 27/05/2014 08:41

3.2.3 AUTOMS

Ce projet est un exemple parmi les nombreux projets qui ont été abandonnés (ne sont plus mis à jour) et dont le site web a été entièrement supprimé. Ils sont donc inutilisables.

Dans ce cas précis, le lien redirige vers la page d'accueil du laboratoire qui avait créé le projet, mais un ancien site est toujours disponible [3]. Il est cependant impossible de télécharger le programme en question.

3.2.4 Falcon-AO

Ce programme, comme de nombreux autres, est disponible en ligne et a été mis à jour relativement récemment (celui-ci, par exemple, date de 2010) mais ne se lance pas. La plupart des programmes étant écrits en Java, il est possible d'obtenir l'erreur via la ligne de commande.

Voici l'erreur obtenue avec le programme Falcon-AO, à titre d'exemple :

```
1  Exception in thread "main" com.sun.xml.internal.ws.
       server.ServerRtException: Server Runtime Error: java.
       net.BindException: Cannot assign requested address
```

Dans tous les cas d'erreurs, une tentative a été faite pour corriger le problème, par l'ajout de librairies ou la consultation des sources. Le travail que représente cette correction est souvent fastidieux. De plus, la plupart des programmes qui ont pu être corrigés se sont révélés inadaptés et décevants car ils ne répondent pas à une des caractéristiques requises (pour rappel, la compatibilité avec les fichiers de "Protégé" ou les possibilités de matching manuel).

3.2.5 HMatch

Ce programme est en réalité un plugin destiné à l'utilisation dans "Protégé". Comme tous les plugins consultés, il n'est malheureusement compatible qu'avec la version 3 de Protégé. L'API de "Protégé" a beaucoup changé entre la version 3 et la version 4; c'est la raison pour laquelle elle n'est pas rétro-compatible. Il faut donc modifier les plugins créés pour les adapter. Ce travail n'est pas souvent réalisé dans le cadre des projets de plugins dans le domaine du matching d'ontologies.

Il a donc été nécessaire, afin de tester ces plugins, d'utiliser une ancienne version de Protégé. Cependant, les plugins se sont malgré tout révélés non fonctionnels et ne démarraient pas avec "Protégé".

3. `http://www.icsd.aegean.gr/kotis/AUTOMS/default.htm`, Mise en ligne : 31/12/2006, Consultation : 27/05/2014 09:20

A nouveau, comme “Protégé” et ses plugins fonctionnent en Java, l’erreur est affichée dans la ligne de commande. Cependant, aucun des plugins évalués dans ce mémoire n’a pu être corrigé.

3.2.6 Optima

De nombreux projets comme Optima se sont révélés inadaptés alors qu’ils étaient fonctionnels, simplement parce qu’ils ne permettaient pas un matching manuel des ontologies. Le matching automatique permet de relier plusieurs concepts mais n’est pas suffisant pour détecter les éléments similaires qui ont des noms différents. Ce genre de programme n’est donc pas adapté au matching recherché ici.

Optima existe en version *standalone* mais aussi en plugin pour “Protégé”. Cependant, comme celui-ci ne fonctionne pas, seule la version *standalone* est concernée par cette description.

La figure 3.1 montre l’interface graphique du programme Optima. Ce programme permet énormément de types de matchings différents - c’est un grand manque de ne pas permettre le matching manuel, étant donné la qualité du matching automatique qui pourrait être utilisé.

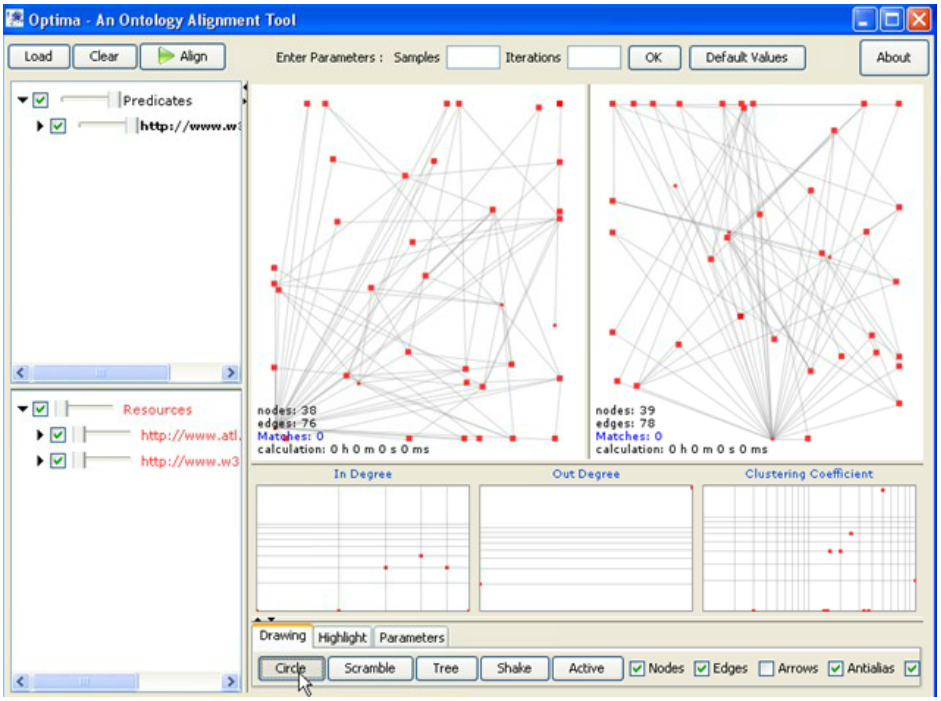

FIGURE 3.1 – Programme “Optima” - Interface principale, [Uthayasanker.T, 2010]

3.2.7 AMW

Le programme AMW (Atlas Model Weaver), basé sur l'environnement de développement bien connu "Eclipse", permet de relier des modèles. Fonctionnel et efficace, il permet de lier les éléments de deux modèles mais n'est pas spécifique aux ontologies. Il est par exemple possible de réaliser un schéma de migration d'un métamodèle vers un autre.

Son interface étant assez lourde à manipuler, il ne représente pas la meilleure option. Cependant, s'il s'avère que plus de fonctionnalités sont nécessaires pour le matching dans le futur, il reste un bon choix.

La figure 3.2 montre l'interface graphique de AMW: on peut voir les deux modèles (à droite et à gauche) et les liens entre eux sur le panneau du milieu.

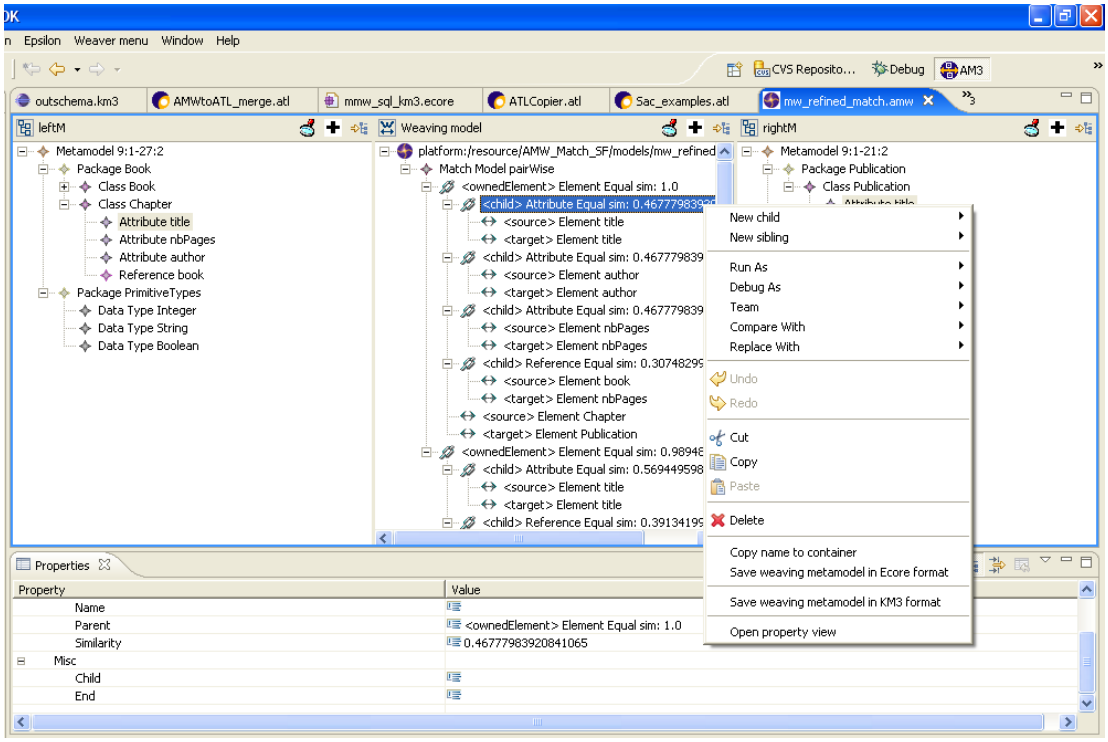

FIGURE 3.2 – Programme "AMW" - Interface principale, [INRIA, 2012]

3.2.8 COMA 3.0

Le programme sélectionné pour ce mémoire, "COMA 3.0", fonctionne sur Windows bien que, étant conçu en Java, il devrait être utilisable sur d'autres plate-formes (cela n'a pas été testé ici). Il nécessite une base de données MySQL [4] et l'environnement Java supérieur ou égal à 1.5.

4. `http://www.mysql.fr/`

La figure 3.3 présente l'interface graphique. Celle-ci est simple, permettant de charger rapidement des ontologies à partir de formats variés. La figure 3.4, quant à elle, montre les différentes possibilités de chargement d'un fichier que le programme offre - celui-ci supporte une large gamme de fichiers d'ontologies différents. Les XSD eux-mêmes peuvent être chargés dans le programme.

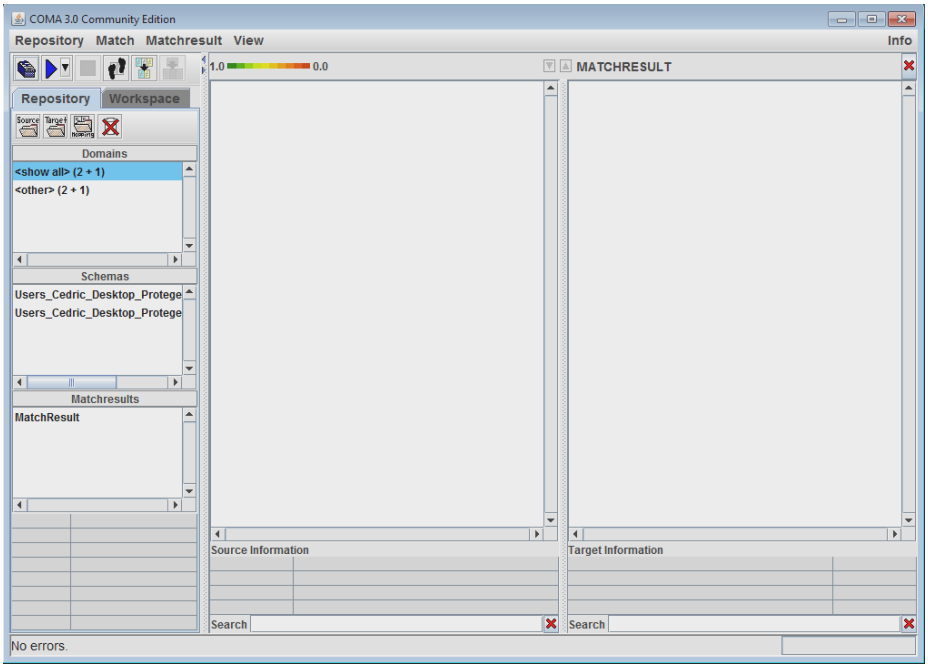

FIGURE 3.3 – Programme "COMA 3.0" - Interface principale

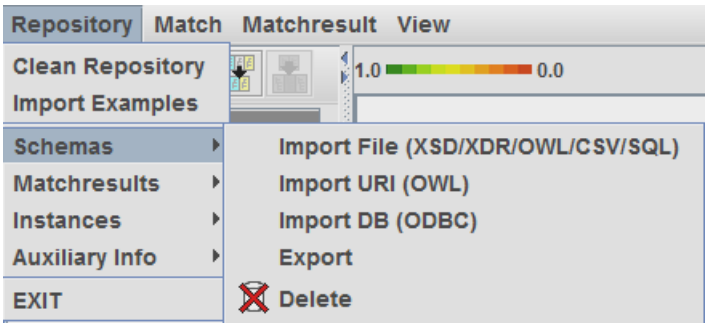

FIGURE 3.4 – Programme "COMA 3.0" - Importation de fichiers

Une fois les fichiers OWL chargés, ceux-ci sont stockés dans la base de données MySQL. Ils seront donc réutilisables par après, étant donné qu'il n'y a pas besoin de recharger les fichiers. De même, le matching une fois effectué pourra être sauvegardé, à condition que les schémas correspondants (ici, les ontologies) ne soient pas supprimés de la base de données.

Le matching peut se faire entièrement manuellement ou de manière automatique et être modifié par la suite. Le programme met à notre disposition plusieurs types d'analyses afin qu'il effectue un matching : en fonction du nom uniquement, de la structure de l'ontologie (si une classe ressemble à une autre), de l'arborescence des classes, etc.

Une fois le calcul effectué, les liens de correspondance d'une ontologie à l'autre qui ont été déduits automatiquement sont affichés. Ceux-ci sont accompagnés d'un pourcentage de similitude. Par exemple, l'élément "city" dans la première ontologie aura un pourcentage de similitude de 100 % avec l'élément "city" dans la deuxième puisqu'ils ont le même nom ; ils sont faciles à détecter automatiquement. Le programme suppose donc qu'ils sont strictement identiques. Par contre, dans le cas de l'adresse, qui se nomme "address" dans la première ontologie, la détection sera plus difficile étant donné qu'il correspond à l'élément "addr" dans la deuxième. Le lien sera tout de même détecté mais il aura une probabilité de seulement 53 %.

Des liens peuvent ensuite être manuellement ajoutés ou supprimés, afin de compléter les relations qui ont été calculées de manière automatique.

Le rendu final du matching est affiché à la figure 3.5. Comme l'interface étant orientée vers un rendu graphique agréable à l'utilisateur, elle permet de visualiser rapidement les liens entre les éléments. De plus, la quantité de liens affichés n'est pas trop importante étant donné que seuls ceux des éléments à l'écran sont affichés.

Par comparaison avec le programme AMW vu à la sous-section 3.2.7, l'interface est nettement simplifiée. Le choix s'est tout de même porté sur "COMA 3.0", étant donné que son interface permet une meilleure visualisation des liens qui sont créés et qu'il n'y a pas ici de nécessité d'utiliser des fonctions avancées.

3.3 Choix d'implémentation

A présent qu'un programme adapté à nos besoins a été sélectionné, cette section présente la méthodologie respectée lors du matching. Celle-ci doit rester la même tout au long de l'analyse afin que le résultat final soit cohérent. De plus, elle doit être reproductible pour permettre une réutilisation de ce travail dans le futur.

Il va de soi que la principale raison d'être du matching est de relier les concepts similaires. La méthode appliquée ici consiste principalement à relier les éléments qui ont la même signification. Par exemple, l'élément "sex" en KMEHR correspond au concept "administrativeGenderCode" en HL7. Ces deux éléments seront donc mis en relation dans le matching.

Nous verrons au chapitre suivant que des détails d'implémentation doivent

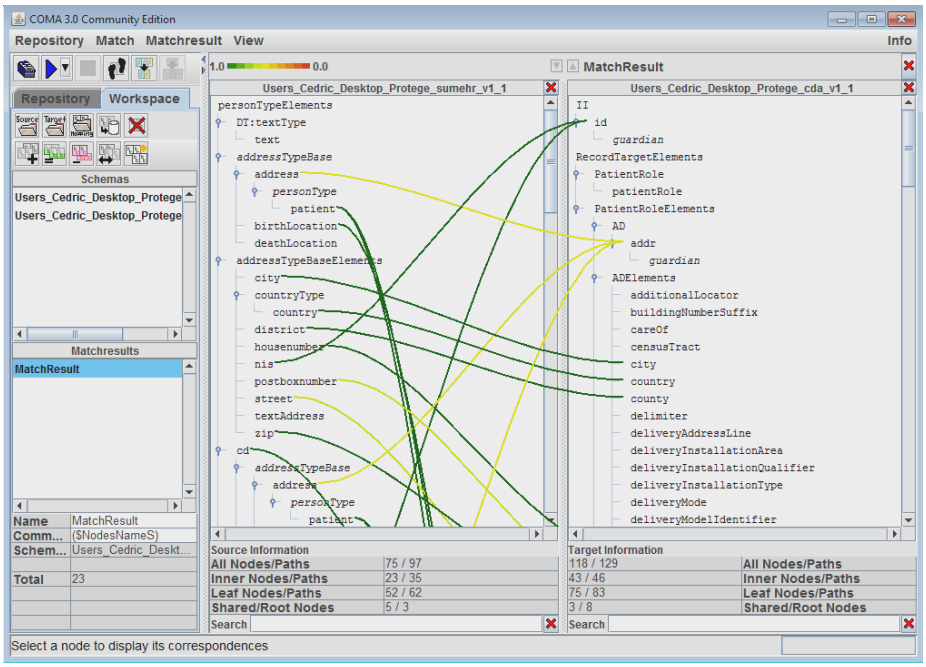

FIGURE 3.5 – Programme "COMA 3.0" - Matching

être réglés. En KMEHR, "sex" contient une chaîne de caractères qui peut prendre comme valeur "male", "female", "unknown" ou "changed". Il contient également des attributs, qui déterminent le système de codage utilisé. En HL7, "administrativeGenderCode" ne peut contenir que "F", "M" ou "UN" (qui signifie "undifferentiated"). Le fait de relier les concepts ne permet pas encore d'implémenter un outil de traduction du KMEHR vers le HL7.

Une chose importante à déterminer est la méthode à appliquer aux éléments qui ne peuvent pas être reliés. Par exemple, l'élément "deathLocation" présent en KMEHR n'existe pas à l'identique en HL7. Dans ce dernier standard, la mort n'est pas représentée par un élément mais par une transaction (la description des messages HL7 a été faite à la section 1.4).

Dès lors, le matching réalisé avec "COMA 3.0" ne permet pas d'exprimer une correspondance aussi subtile. Malheureusement, la limite des possibilités de cette analyse est atteinte à cause de cet obstacle. Si la possibilité existait d'exprimer une correspondance telle que celle-là, elle serait synonyme d'une manipulation compliquée et obligerait à un travail qui serait illisible par sa densité. Il n'est donc pas possible de continuer plus avant dans l'établissement de correspondances entre les éléments sans devoir redescendre au niveau de la syntaxe ou des données.

Ces éléments ne sont donc pas reliés dans le matching réalisé dans ce chapitre.

La méthodologie utilisée ici peut donc se résumer en deux points :

- Liaison des éléments sémantiquement identiques.
- Lorsqu'un élément ne peut pas être relié simplement à un autre parce que leur mode de représentation est trop différent, aucun lien n'est réalisé pour ne pas compliquer inutilement le matching.

À propos des versions du HL7, il est intéressant de remarquer que beaucoup d'éléments présents dans l'entité "patient" qu'on ne peut pas traduire du SumEHR au CDA existent, par contre, dans les données du patient de la version 2.

Pour rappel, la version 3 du HL7 (sur laquelle est basé le CDA) ayant changé de format (passage d'un texte brut pour la version 2 à du XML pour la version 3), il y a eu également un changement complet des différents éléments de chaque version. Plusieurs informations qui se trouvaient dans les caractéristiques du patient de la version 2 du HL7 étaient équivalentes à celles du KMEHR. Or, avec la version 3, ces éléments ne se trouvent plus dans l'entité "patient" et ne peuvent donc pas être reliés directement dans notre cas.

Ces éléments n'ont bien entendu pas disparu - ils sont simplement exprimés autrement. La nationalité, par exemple, n'existe plus dans la section qui représente le patient. Elle doit dès lors être exprimée comme la valeur d'une observation - la partie du CDA qui fournit les informations (en général médicales) sur le patient.

Les sections 3.3.1 à 3.3.4 présentent les différents cas rencontrés dans le matching et les liens établis en conséquence.

3.3.1 Matching automatique

Il est important de mentionner le matching réalisé automatiquement par le programme choisi. On peut voir dans la figure 3.6 des chiffres à côté de chaque lien. Par exemple, "Street" et "StreetName" sont marqués d'un 0.6, alors que "patient" et "Patient" ont comme valeur 1.0. Ces valeurs représentent la probabilité qu'il y ait réellement un lien entre ces deux éléments selon l'algorithme de "COMA 3.0".

L'algorithme appliqué ici analyse les noms des éléments et indique à l'aide de la valeur spécifiée le degré de ressemblance de ceux qu'il a matché automatiquement.

Les liens réalisés peuvent être supprimés s'ils sont erronés. Les liens ajoutés séparément se voient automatiquement attribués une valeur de 1.0 : étant donné qu'ils ont été manuellement encodés, ils sont supposés obligatoirement vrais.

3.3.2 Noms et structures identiques

Ce cas est trivial : il s'agit d'éléments identiques, dont la structure est la même. Par exemple, l'élément "country" a le même nom en KMEHR et en HL7 et il s'agit des deux côtés d'une classe simple et non pas d'une classe possédant des sous-éléments.

3.3.3 Noms différents, structures identiques

Ce cas est également simple à définir : lorsqu'un élément est identique à celui de l'autre ontologie mais de nom différent. Le lien peut alors être effectué sans poser de problème. Par exemple, "zip" et "postalCode" : ceux-ci ayant des noms très différents, ils ne sont pas matchés automatiquement mais se trouvent tous les deux dans l'élément "adresse" et représentent en réalité le même concept. Il s'agit du même élément qui peut donc être matché étant donné qu'ils ont la même sémantique.

3.3.4 Noms et structures différents

Ce cas est le plus complexe. Il se divise en deux : les situations où il est tout de même possible de matcher les éléments malgré leur différence de structure et les cas où cela n'est pas possible.

Dans le premier cas, on retrouve les éléments qui ont une structure plus complexe. Par exemple, "telecomNumber" et "telecom". Le premier contient trois éléments : deux servent à l'identification et le troisième fournit le numéro. Dans le deuxième, on ne retrouve qu'un élément servant à l'identification. La valeur du numéro est un attribut et n'existe donc pas dans l'ontologie. Les éléments peuvent être reliés mais ne sont pas strictement similaires. C'est le concept supérieur qui sera alors matché: au lieu de réaliser un matching sur les éléments contenus dans "telecomNumber" et dans "telecom", on va créer un lien directement sur ces deux éléments. Les éléments qu'ils contiennent ne seront donc pas reliés mais les concepts le seront tout de même.

Le deuxième cas concerne les éléments trop complexes pour effectuer un quelconque matching. La nationalité du patient en KMEHR n'existe pas en tant qu'élément dans le HL7. Elle doit être exprimée comme une "observation". Relier la nationalité avec l'observation n'aurait pas de sens au niveau du matching puisqu'il ne s'agit pas du même concept : une observation peut concerner bien plus d'éléments que la nationalité du patient. Ces relations ne peuvent pas être établies ici et devront donc être réalisées directement dans la phase de traduction.

3.4 Résultats obtenus

Une fois l'outil adapté trouvé, le matching en lui-même est très rapide. Il s'agit uniquement de lier les éléments entre eux - leurs concepts sont déjà connus étant donné qu'ils ont été étudiés lors du passage du schéma XML à l'ontologie dans le chapitre 2.

Les résultats du matching effectué sur l'entité patient (à des fins de test de la méthodologie) se trouvent en annexe sur le CD-ROM. La figure 3.5, montre le résultat du matching. La figure 3.6, quant à elle, montre celui-ci une fois exporté au format texte.

```
MatchResult [75,118] Users_Cedric_Desktop_Protege_sumehr_v1_1Users_Cedric_Desktop_Protege_cda_v1_1---
 - 376[address] <-> 832[addr]: 0.533333
 - 421[patient] <-> 820[Patient]: 1.0
 - 421[patient] <-> 875[patient]: 1.0
 - 383[birthLocation] <-> 836[birthplace]: 1.0
 - 393[city] <-> 842[city]: 1.0
 - 394[country] <-> 844[country]: 1.0
 - 401[district] <-> 845[county]: 1.0
 - 405[housenumber] <-> 862[houseNumber]: 1.0
 - 420[nis] <-> 864[id]: 1.0
 - 425[postboxnumber] <-> 878[postBox]: 0.583333
 - 434[street] <-> 895[streetName]: 0.6
 - 447[zip] <-> 879[postalCode]: 1.0
 - 384[cd] <-> 798[CD]: 1.0
 - 382[birthDate] <-> 835[birthTime]: 0.545455
 - 398[dateTypeElements] <-> 811[IVL_TS]: 1.0
 - 406[id] <-> 864[id]: 1.0
 - 437[telecomType] <-> 829[TEL]: 1.0
 - 403[familyName] <-> 856[family]: 0.6
 - 404[firstName] <-> 857[given]: 1.0
 - 428[recorddatetime] <-> 854[effectiveTime]: 1.0
 - 429[sex] <-> 833[administrativeGenderCode]: 1.0
 - 439[telecomnumber] <-> 899[telecom]: 1.0
 - 449[hasElement] <-> 911[hasElement]: 1.0
 + Total: 23 correspondences
--------------------------------------------------------
```

FIGURE 3.6 – Programme "COMA 3.0" - Exportation

Chapitre 4

Du KMEHR vers le HL7

Ce chapitre représente le point culminant de la méthodologie mise en place dans ce mémoire. Si l'analyse est une partie extrêmement importante de ce travail, elle a tout de même pour but la création d'un outil de traduction du KMEHR vers le HL7. Tout cela, bien sûr, en étant le plus rigoureux possible tout en permettant une automatisation du processus.

La première section revient sur le but de ce mémoire, afin de faire le lien entre les différentes étapes. Ensuite, le choix des outils destinés à la traduction du KMEHR vers le HL7 sera expliqué, suivi de la méthodologie destinée à cette traduction. Le chapitre conclut avec les résultats obtenus.

4.1 But poursuivi

Il est important de situer cette étape dans le contexte global. Elle conclut la méthode réalisée ici avec la création de l'outil de conversion. L'analyse faite aux chapitres 2 et 3 doit permettre d'optimiser le travail de traduction. L'idée initiale était de permettre la création automatique d'un traducteur à partir du matching. Une fois les ontologies réalisées et le matching effectué, celui-ci aurait été exporté et transmis à un outil. Cet outil, à partir du fichier exporté par "COMA 3.0", aurait produit automatiquement un traducteur de documents KMEHR vers le HL7 (et éventuellement dans le sens inverse également).

Cependant, étant donné la méthodologie utilisée pour la création d'ontologies, il n'est pas possible de créer un outil à partir du matching. Lors du passage aux ontologies, il y a une perte d'information vis-à-vis de l'implémentation. Les attributs, par exemple, ne sont pas retranscrits dans l'ontologie et les types simples (en XML) n'apparaissent pas dans l'ontologie. Le matching étant effectué sur les concepts des standards, il n'est donc pas possible, à partir du fichier exporté, de fournir un outil de traduction qui respecte les contraintes de l'im-

plémentation. Ce problème est contourné par la création manuelle d'un outil en utilisant le matching. Nous verrons au chapitre suivant les pistes pour améliorer cela afin de permettre l'automatisation du processus.

Dès lors, pour la traduction des standards, l'outil a été créé en se basant sur le matching pour savoir quels éléments relier. Celui-ci sert alors comme une "feuille de route" de l'implémentation pratique, pour aider à l'écriture du programme de traduction.

Pour réutiliser la description faite au début du chapitre 3, nous redescendons ici au niveau des "données". Les deux chapitres précédents portaient sur les niveaux de la "syntaxe" et des "concepts". Nous allons à présent, sur base des résultats de ceux-ci, redescendre au niveau de "données" et créer un outil de manipulation du XML permettant la transformation de documents respectant la syntaxe des standards.

4.2 Choix du langage

Cette section présente les choix qui ont mené à la sélection du XSL pour la réalisation du convertisseur du KMEHR au HL7.

Le langage choisi doit impérativement être capable de manipuler la structure d'un fichier XML, ses données et d'effectuer des transformations sur ces informations. Dans ce but, il doit également pouvoir stocker des informations temporairement afin qu'elles soient réutilisées à une autre étape du traitement.

Le premier langage considéré a été le langage Java. Celui-ci aurait permis une traduction automatique à partir du fichier de matching. Cependant, comme expliqué à la section précédente, cette solution n'est pas implémentable.

Dès lors, le langage XSL, très répandu pour le traitement de fichiers XML, est le plus adapté aux besoins de la tâche. Son but premier étant la transformation de documents XML, il est donc parfaitement adapté à notre situation, étant donné que les deux standards utilisent le XML dans leurs messages. De plus, il offre les fonctionnalités nécessaires à un matching complexe lorsque les éléments ne correspondent pas directement.

Il a donc été choisi comme langage pour la création de l'outil de traduction.

La section suivante présente le langage XSL et ses fonctionnalités.

4.3 Langage XSL

Le langage XSL [1] est une recommandation du W3C, tout comme le langage OWL et le XML. Il permet de décrire la transformation à appliquer sur un

1. XSL : eXtensible Stylesheet Language

document XML. Son implémentation est le langage XSLT [2] qui fait partie de la même recommandation du W3C.

Le langage XSLT se base sur XPath afin de parcourir le document XML. XPath est un langage permettant de sélectionner une partie d'un document XML. Les chemins XPath ressemblent aux chemins des fichiers sous UNIX : "/patient/deathlocation" pointe l'élément "deathlocation" qui se trouve dans l'élément "patient". Il permet également d'effectuer des commandes simples (if, for-each). L'association de XPath et de XSLT permettra ainsi de traiter des cas complexes de transformation des fichiers XML.

Étonnamment, le langage XSLT est lui-même écrit sous la forme d'un document XML. Avec celui-ci apparaît également la notion de contexte liée à XPath. Celle-ci doit être expliquée pour permettre de bien saisir la méthodologie qui sera appliquée à la section 4.4.

Prenons par exemple un document XML simple.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <kmehrmessage>
3   <patient>
4     <address>
5       adresseDuPatient
6     </address>
7   </patient>
8 </kmehrmessage>
```

Si nous souhaitons transformer ce document, nous pouvons utiliser un fichier XSLT afin de modifier la forme et même les données présentes dans celui-ci. Voici un exemple de fichier XSLT destiné à modifier le document XML original :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:template match="/">
3          <xsl:apply-templates select="kmehrmessage"/>
4  </xsl:template>
5  <xsl:template match="kmehrmessage">
6      <HL7_message>
7   <xsl:apply-templates select="patient"/>
8      </HL7_message>
9   </xsl:template>
10  <xsl:template match="patient">
11    <PatientTarget>
12  <address>
13    <xsl:value-of select="address" />
```

2. eXtensible Stylesheet Language Transformation

```
14    </address>
15      </PatientTarget>
16    </xsl:template>
```

Ceci mérite quelques explications. Nous voyons que le premier élément est un “template match”. Ce code XSLT permet de charger le contexte de l’élément voulu. Par charger le contexte, il faut comprendre “charger l’entièreté du code XML contenu dans l’élément”. Cela revient à dire que l’on “entre dans l’élément” pour le traiter.

Dans ce premier “template match”, nous chargeons l’élément “/”. Celui-ci désigne, par une expression XPath, la racine du document. En effet, tout document XML contient une et une seule racine. Dans notre document, la racine contient l’élément “kmehrmessage”. Autrement dit, charger la racine dans le contexte revient en quelque sorte à ouvrir le document XML à traiter.

Nous pouvons voir qu’à l’intérieur de cet élément, nous faisons appel à un autre template à l’aide de “apply-templates”, appliqué sur “kmehrmessage”. Le fichier XSLT donne donc comme instruction “chercher un élément nommé kmehrmessage et applique le template correspondant à celui-ci”.

Le document XML contenant effectivement un élément “kmehrmessage”, le template correspondant sera chargé. Dans ce deuxième template, seul le contexte correspondant sera chargé. Autrement dit, nous n’utilisons plus ici l’entièreté du document XML mais uniquement la partie contenue dans l’élément “kmehrmessage”. Pour l’instant, cela ne change bien sûr pas grand-chose étant donné que cet élément est à la racine du document.

Le template qui se charge ne se soucie plus, à présent, de la racine du document. Nous nous trouvons à la ligne 6 et nous travaillons sur une section plus réduite du document XML. Dans ce contexte, la situation est similaire à un fichier XSLT appiqué à ce document :

```
1  <patient>
2      <address>
3          adresseDuPatient
4      </address>
5    </patient>
```

Nous sommes donc dans une section réduite du document. La ligne 7 du document XSLT peut paraître étrange : elle n’est en réalité pas une instruction mais du texte, qui sera recopié à l’identique dans le document XML final. A ce stade, le document XML final contient donc une unique balise <HL7_message>. A l’intérieur de celle-ci, nous appliquons (l’instruction se trouve à la ligne 8) un template sur l’élément patient.

Nous nous trouvons maintenant à la dernière étape de notre exemple, dans

le template de l'élément "patient". Le contexte mis à jour est donc maintenant le suivant :

```
1       <address>
2          adresseDuPatient
3       </address>
```

Nous pouvons de nouveau voir quelques lignes qui ne sont pas des instructions et qui vont écrire dans le fichier XML transformé (il s'agit de "Patient-Target" et de "address"). La ligne "<xsl:value-of select="address" />" est une instruction de sélection qui écrit la valeur souhaitée se trouvant dans le "select". Dans celui-ci, nous voyons qu'il appelle la valeur de l'adresse du patient.

Cette instruction va donc retourner la valeur suivante : "adresseDuPatient".

Le traitement du fichier XML à l'aide du fichier XSLT étant maintenant terminé, nous obtenons le fichier suivant :

```
1       <HL7_message>
2          <PatientTarget>
3    <address>
4       adresseDuPatient
5    </address>
6          </PatientTarget>
7       </HL7_message>
```

Le fichier obtenu en sortie correspond donc bien à ce qui était voulu et, tout en contenant la même information que le fichier d'origine (ici, l'adresse du patient), il utilise un schéma différent du premier.

Il faut ajouter une remarque sur le processeur XSLT. En effet, un fichier de tranformation du XML n'est pas un programme, seulement un fichier. Dès lors, il faut qu'il soit exécuté par un programme capable d'interpréter ce type de fichiers et de l'appliquer à un document XML. C'est ce programme qui possède le nom de processeur XSLT.

Ce mémoire n'utilisant pas de commande complexes du langage XSLT, le choix du processeur n'a pas beaucoup d'importance.

L'outil de travail pour la rédaction du fichier XSLT est ici le même que celui utilisé au chapitre 2 pour manipuler les XSD : "<oXygen/> XML editor". La figure 4.1 montre la fenêtre d'édition des scénarios du programme, qui permet de choisir le fichier XML source (à transformer) et le fichier XSLT à appliquer pour la transformation. Dans le bas de la fenêtre, les différents processeurs sont affichés (appelés "Transformateur").

Le processeur utilisé pour ce mémoire est le processeur "Saxon", couramment utilisé et fort répandu. Il est référencé dans [Tidwell, 2008]

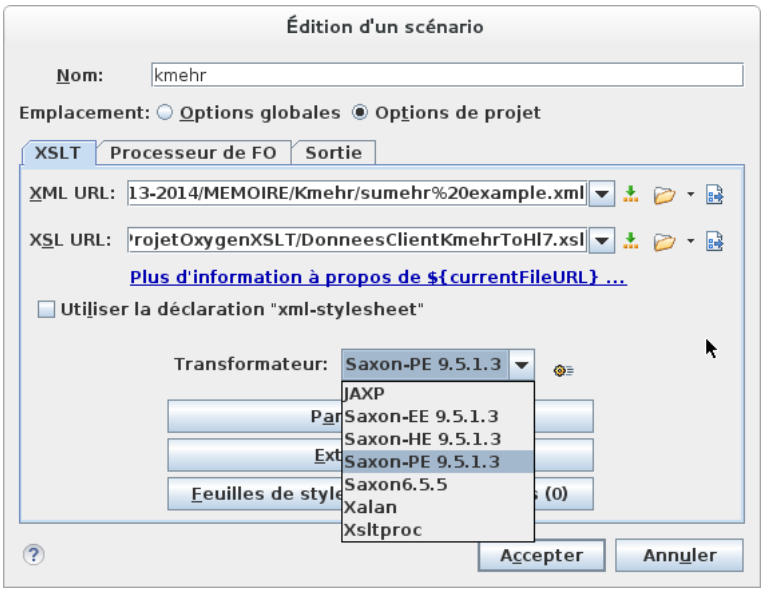

FIGURE 4.1 – Programme "<oXygen/> XML editor" - Processeurs XSLT

Le code suivant montre une partie du fichier XSLT réalisé pour la conversion du KMEHR vers le HL7. Dans celle-ci se trouve le début du traitement de l'entité patient, avec l'ID du patient et une partie du traitement de l'adresse (l'ID de l'adresse et son utilisation). On y trouve divers éléments qui n'ont pas été présentés dans les exemples. La table 4.1 les décrit brièvement pour permettre la bonne compréhension du code XSLT.

instruction	Effet
xsl:for-each	Permet d'effectuer un traitement sur tous les éléments d'un certain type (par exemple, toutes les adresses du patient)
xsl:if	Instruction classique "si - alors".
<!– –>	Commentaires dans le fichier XSLT. Ceux-ci n'apparaissent pas dans le fichier XML de sortie.
xsl:attribute	Permet d'ajouter un attribut à l'élément XML en cours.
xsl:choose	Permet d'effectuer un traitement en fonction de la situation (les balises "when" servant à décrire la situation).
xsl:comment	Permet d'insérer un commentaire dans le fichier XML de sortie.
@attribut	Permet de sélectionner la valeur d'un attribut

TABLE 4.1 – Tableau des instructions XSLT

```
1  <!-- Analyse de la racine du document KMEHR -->
2     <xsl:template match="/">
3         <xsl:apply-templates select="kmehrmessage"/>
4     </xsl:template>
5     <!-- element kmehrmessage -->
6     <xsl:template match="kmehrmessage">
7         <ClinicalDocument xmlns:ClinicalDocument="urn:hl7
              -org:v3" xmlns:voc="urn:hl7-org:v3/voc" xmlns:
              xsi="http://www.w3.org/2001/XMLSchema-instance
              "> <!-- xmlns:ClinicalDocument="urn:hl7-org:v3
              "? -->
8         <xsl:apply-templates select="header"/>
9         <xsl:apply-templates select="folder"/>
10        </ClinicalDocument>
11    </xsl:template>
12    <!-- elements header et folder -->
13    <xsl:template match="header">
14    </xsl:template>
15    <xsl:template match="folder">
16        <xsl:apply-templates select="id" mode="folderID"
              />
17        <xsl:apply-templates select="patient"/>
18        <xsl:apply-templates select="transaction"/>
19    </xsl:template>
20    <!-- elements id, patient et transaction -->
21    <xsl:template match="id" mode="folderID">
22    </xsl:template>
23    <xsl:template match="patient">
24      <!-- amelioration possible: traiter les nullFlavor
            dans les elements -->
25       <recordTarget>
26        <patientRole>
27         <xsl:for-each select="id">
28          <id>
29           <xsl:attribute name="root">
30            <xsl:value-of select="@S" />_V_<xsl:value-of
                  select="@SV" />
31             <xsl:if test="@SL">_L_<xsl:value-of select="
                  @SL"/></xsl:if>:<xsl:value-of select="."
                  />
```

```
32              </xsl:attribute>
33             </id>
34            </xsl:for-each>
35             <xsl:if test="address">
36              <addr>
37              <xsl:attribute name="use">
38               <xsl:choose>
39          <xsl:when test="address/cd='home'">H</xsl:when>
40          <xsl:when test="address/cd='work'">WP</xsl:when>
41          <xsl:when test="address/cd='temp'">TMP</xsl:when>
42          <xsl:when test="address/cd='old'">OLD</xsl:when>
43          <xsl:otherwise>H</xsl:otherwise> <!-- lorsque le
                type est inconnu, "home" est utilise par
                defaut -->
44               </xsl:choose>
45              </xsl:attribute>
46              <xsl:for-each select="address/id">
47               <xsl:comment> id adresse non ajoute en HL7: <
                     xsl:value-of select="@S" />_V_<xsl:value-
                     of select="@SV" />
48                <xsl:if test="@SL">_L_<xsl:value-of select="
                     @SL"/></xsl:if>:<xsl:value-of select="."
                     ></xsl:value-of>
49               </xsl:comment>
50              </xsl:for-each>
```

Finalement, il faut souligner que le passage par des templates n'est pas obligatoire. Nous aurions pu, en un seul template, accéder directement à l'élément qui nous intéresse :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:template match="/">
3   <HL7_message>
4     <PatientTarget>
5       <address>
6   <xsl:value-of select="kmehrmessage/patient/address" />
7       </address>
8     </PatientTarget>
9    </HL7_message>
10 </xsl:template>
```

Nous pouvons donc directement accéder à un élément en indiquant son che-

min d'accès (kmehrmessage/patient/address). Cette méthode fonctionne tout aussi bien, mais si aucun template n'est utilisé, le fichier XSLT est vite surchargé et illisible.

4.4 Méthodologie de la transformation

Afin de garder une structure identifiable dans le fichier XSLT, plusieurs choix ont dû être faits. Ceux-ci sont présentés dans cette section.

4.4.1 Utilisation du matching

Le matching effectué au chapitre 3 a été respecté lors de la construction du fichier XSLT. Les concepts similaires qui ont été reliés l'un à l'autre représentent les correspondances qui doivent être mises en place par l'utilisation du XSL. Le matching permet donc, comme une feuille de route, de visualiser rapidement comment les éléments doivent être placés dans le document HL7. Celui-ci devant, en sortie, être un fichier respectant le standard du CDA (rappelons que le CDA est l'équivalent du SumEHR, les deux standards traités dans ce mémoire), les détails d'implémentation sont traités à cette étape : modification éventuelle des données, transformation des attributs, sélection des éléments par défaut lorsqu'ils n'existent pas en HL7 ou en KMEHR, changement du format des dates, correspondance entre les sytèmes de codage.

Le matching permet donc d'accélérer la rédaction du fichier de transformation. Dans le futur, si les standards évoluent et que ce fichier doit être mis à jour pour correspondre à une nouvelle version de ceux-ci, le matching rendra également la modification plus facile.

4.4.2 Utilisation des templates

Nous l'avons vu à la section précédente : XPath permet d'utiliser deux types d'accès à une section du document XML : les templates et les chemins d'accès. Afin de clarifier le fichier XLST final, celui-ci est divisé à l'aide de plusieurs templates dès que cela est nécessaire, en fonction de la taille de la partie à traiter, afin de clarifier le code tout en faisant attention à ne pas trop diviser les sections (une surutilisation des templates compliquerait la lecture du document).

4.4.3 Éléments non utilisés

Les éléments qui ne peuvent pas être traduits (par exemple, l'élément "nis" dans l'adresse du patient) sont placés en commentaire dans le fichier HL7 afin de conserver l'information, bien qu'elle ne sera pas utilisée dans celui-ci. Cela

permet également de visualiser rapidement, dans le document final, les données qui n'ont pas été traitées et qui sont donc perdues en HL7.

Il sera intéressant, dans le futur, de copier également ces éléments dans la partie du message HL7 qui est non structurée. Pour rappel, un message CDA peut contenir une partie non structurée - et donc non identifiable par l'ordinateur, comme un PDF ou du texte brut. De cette manière, les informations seraient quand même disponible dans le dossier sans devoir pour autant accéder au code brut du message.

4.4.4 Traitement des ID

Le système de codage des identifiants étant radicalement différent en KMEHR et en HL7, il n'est pas possible de les convertir de manière simple. Dans la version actuelle du fichier, les ID sont simplement intégrés au fichier HL7. Cependant, bien que la structure du schéma soit correcte, cela ne permettra pas de le reconnaître dans un système utilisant le standard HL7.

Nous aborderons le travail restant concernant les ID dans le chapitre 5.

4.4.5 Perte d'information et types par défaut

A certains endroits du fichier KMEHR, des attributs expriment des informations qui existent de manière similaire en HL7. La chose se complique lorsque la liste des choix possibles est plus grande en KHMEHR qu'en HL7. Le code ci-dessous en montre un exemple.

```
1  <xsl:for-each select="address">
2    <addr>
3      <xsl:attribute name="use">
4        <xsl:choose>
5    <xsl:when test="cd='home'">H</xsl:when>
6    <xsl:when test="cd='vacation'">HV</xsl:when>
7    <xsl:otherwise>H
8        <xsl:comment>patient.address.use non reconnu(<xsl:
             value-of select="cd"/>), valeur H utilis e</xsl
             :comment>
9    </xsl:otherwise> <!-- lorsque le type est inconnu ou "
         other", "home" est utilise par defaut -->
10       </xsl:choose>
11     </xsl:attribute>
```

Nous pouvons voir que l'attribut “use” dans l'adresse du patient peut être exprimée de différentes manières en KMEHR. Ses valeurs peuvent être les suivantes : “home”, “work”, “vacation” ou “other”. En HL7, il existe également plusieurs valeurs possibles, mais aucune n'est de catégorie “other”. Dès lors, lorsque cette catégorie est rencontrée, la valeur utilisée par défaut est “H” (home).

Cette façon de faire cause forcément une perte d'information si l'adresse mentionnée est de type “other”. De nombreux cas similaires se rencontrent lors de l'analyse, et il est important que tout utilisateur de l'outil de conversion soit au courant des modifications qui seront apportées. Dans ce but, celles-ci sont toujours insérées en commentaire dans le fichier final lorsque le cas se produit.

4.5 Résultats obtenus

Cette section est l'aboutissement de ce travail. La méthodologie exposée jusqu'ici doit permettre la création de l'outil de traduction - c'est ce résultat qui est testé ici afin de vérifier la validité de la méthode. Il montre son bon fonctionnement.

Grâce à l'analyse effectuée préalablement aux chapitres 2 et 3, le travail de rédaction du fichier XSLT permettant le passage du SumEHR au CDA se révèle assez rapide.

La création d'une méthodologie stricte permettant de travailler efficacement s'est faite progressivement, au fur et à mesure des problèmes rencontrés lors de l'écriture. En effet, sans avoir une connaissance parfaite des standards et du langage XSLT, il est quasiment impossible de prévoir les obstacles qui vont devoir être contournés.

Cependant, il faut souligner que les difficultés rencontrées sont mineures et donc vite réglées. De plus, la structure en arbre du XML fait que les modifications nécessaires peuvent, la plupart du temps, être effectuées rapidement et sans impacter le reste du travail.

De plus, l'utilisation de l'outil "<oXygen/> XML editor" permet une construction rapide du fichier XSLT étant donné qu'il fournit une excellente interface de construction.

Pour rappel, la méthodologie présentée dans ce mémoire est appliquée uniquement à l'entité patient, afin de vérifier son bon fonctionnement. Le XSL réalisé ici transforme donc l'entité patient d'un SumEHR en entité patient du CDA.

Voici le code source d'un fichier test utilisé, fourni sur le site du KMEHR [3] :

```
1  <kmehrmessage>
2    <header>
3      <standard>
4        <cd S="CD-STANDARD" SV="1.0">20100601</cd>
5      </standard>
6      <id S="ID-KMEHR" SV="1.0"
           >14296612004.20060520005759598</id>
7      <date>2006-05-20</date>
8      <time>00:57:59</time>
9      <sender>
10       <hcparty>
11         <id S="ID-HCPARTY" SV="1.0">14296612004</id>
12         <cd S="CD-HCPARTY" SV="1.0">persphysician</cd>
13         <firstname>Johan</firstname>
14         <familyname>Brouns</familyname>
15       </hcparty>
16     </sender>
17     <recipient>
18       <hcparty>
19         <id S="ID-HCPARTY" SV="1.0">71099812</id>
20         <cd S="CD-HCPARTY" SV="1.0">orghospital</cd>
21         <name>Test Hospital</name>
22       </hcparty>
23     </recipient>
24   </header>
```

3. https://www.ehealth.fgov.be/standards/kmehr/content/page/transactions/94/summarised-electronic-healthcare-record-v11, Mise en ligne : inconnue, Consultation : 27/05/2014 14:07

```
25    <folder>
26      <id S="ID-KMEHR" SV="1.0">1</id>
27          <patient>
28        <id S="ID-PATIENT" SV="1.0">7031966696</id>
29        <id S="LOCAL" SV="1.0" SL="GPSMF-ID">51635</id>
30        <id S="LOCAL" SV="3.2" SL="MySoftware-Patient-ID"
              >51635</id>
31        <firstname>Patient</firstname>
32        <familyname>ForTestingPurpose</familyname>
33        <birthdate>
34          <date>1961-12-25</date>
35        </birthdate>
36        <birthlocation>
37          <country>
38            <cd SV="1.0" S="CD-FED-COUNTRY">it</cd>
39          </country>
40          <city>Napoli</city>
41        </birthlocation>
42        <sex>
43          <cd S="CD-SEX" SV="1.0">male</cd>
44        </sex>
45        <address>
46          <cd S="CD-ADDRESS" SV="1.0">home</cd>
47          <country>
48            <cd S="CD-FED-COUNTRY" SV="1.0">be</cd>
49          </country>
50          <zip>5000</zip>
51          <city>Namur</city>
52          <street>Rue de La Gare</street>
53          <housenumber>12</housenumber>
54        </address>
55        <telecom>
56          <cd S="CD-ADDRESS" SV="1.0">home</cd>
57          <cd S="CD-TELECOM" SV="1.0">phone</cd>
58          <telecomnumber>+3222345678</telecomnumber>
59        </telecom>
60        <telecom>
61          <cd S="CD-ADDRESS" SV="1.0">home</cd>
62          <cd S="CD-TELECOM" SV="1.0">email</cd>
63          <telecomnumber>patient.fortestingpurpose@provider
```

```
            .org</telecomnumber>
64        </telecom>
65        <telecom>
66          <cd S="CD-ADDRESS" SV="1.0">other</cd>
67          <cd S="CD-TELECOM" SV="1.0">mobile</cd>
68          <telecomnumber>0477777778</telecomnumber>
69        </telecom>
70        <usuallanguage>fr</usuallanguage>
71        <profession>
72          <text L="fr">plombier</text>
73        </profession>
74        <recorddatetime>2010-08-11T10:10:00</recorddatetime
              >
75      </patient>
```

Son équivalent en HL7, une fois traduit à l'aide du fichier XSLT construit à cette étape pour le test de la méthode, est celui-ci:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2 <ClinicalDocument xmlns:ClinicalDocument="urn:hl7-org:v3"
3                   xmlns:voc="urn:hl7-org:v3/voc"
4                   xmlns:xsi="http://www.w3.org/2001/
                        XMLSchema-instance">
5     <recordTarget>
6       <patientRole>
7         <id root="ID-PATIENT_V_1.0:7031966696"/>
8         <id root="LOCAL_V_1.0_L_GPSMF-ID:51635"/>
9         <id root="LOCAL_V_3.2_L_MySoftware-Patient-ID
              :51635"/>
10        <addr use="H">
11          <country>be</country>
12          <city>Namur</city>
13          <postalCode/>
14          <streetAddressLine>Rue de La Gare</
                streetAddressLine>
15          <houseNumber>12</houseNumber>
16        </addr>
17        <telecom use="home" value="phone:+3222345678"/>
18        <telecom use="home" value="email:patient.
              fortestingpurpose@provider.org"/>
19        <telecom use="other" value="mobile:0477777778"/>
```

```
20            <patient>
21                <name>
22                    <family>ForTestingPurpose</family>
23                    <given>Patient</given>
24                </name>
25                <administrativeGenderCode code="M" codeSystem
                      ="2.16.840.1.113883.5.1"/>
26                <birthTime value="19611225"/>
27                <birthPlace>
28                    <place>
29                        <addr use="H">
30                            <country>it</country>
31                            <city>Napoli</city>
32                        </addr>
33                    </place>
34                </birthPlace>
35                <languageCommunication>
36                    <languageCode code="fr"/>
37                </languageCommunication>
38                <!--
39                                    profession not in use in
                                        patientRole:
40
41                                    patient's profession:
                                        plombier (language fr)
42                                -->
43                <!--
44                                    recorddatetime not in use in
                                        patientRole:
45                                    2010-08-11T10:10:00-->
46            </patient>
47            <providerOrganization>
48                <name>ID-PATIENT_V_1.0</name>
49                <name>LOCAL_V_1.0_L_GPSMF-ID</name>
50                <name>LOCAL_V_3.2_L_MySoftware-Patient-ID</
                      name>
51            </providerOrganization>
52        </patientRole>
53    </recordTarget>
54 </ClinicalDocument>
```

Chapitre 5

Limites et améliorations possibles

Ce chapitre présente des pistes à suivre pour continuer ce mémoire. Beaucoup de travail reste à réaliser afin que l'outil de traduction soit mis en place et puisse être exploité de manière efficiente. Dans un but d'efficacité et d'automatisation du processus, les pistes existantes pour l'amélioration de la méthodologie de travail sont exposées ici.

5.1 Automatisation de la construction des ontologies

Le chapitre 2 mentionne les transformations d'un XSD en ontologie de manière automatique. Plusieurs projets existants permettent la transformation, entièrement automatisée, des fichiers. L'ontologie produite en sortie respecte les règles mentionnées dans [Lacoste et al., 2011] et dans [Bohring and Auer, 2005]. Une des implémentations possibles de ces paramètres peut se trouver en ligne sur [Garcia, 2005].

Réutiliser l'outil actuel n'est pas possible étant donné que les règles de transformation utilisées au chapitre 2 sont radicalement différentes de celles appliquées dans les programmes existants. La sous-section 2.4.1 montre que la méthode utilisée dans ce genre d'outils produit une ontologie complexe, correspondant au domaine, mais qui n'est pas adaptée au matching et n'est donc pas utile à ce mémoire. Cependant, il serait possible de s'inspirer de ceux-ci afin de créer un fichier XSL (ou un autre type de programme ou fichier) personnalisé qui transformerait un schéma XML en fichier OWL. Une des pistes intéressantes pour l'amélioration de la méthode serait la construction d'un tel outil.

Cette réalisation demanderait une démarche spécifique : elle nécessite un apprentissage de la syntaxe des XSD mais aussi de celle des fichiers OWL. Elle représente donc une grande quantité de travail. C'est pour cette raison qu'elle n'a pas été retenue pour ce mémoire.

5.2 Automatisation de la création de l'outil

Le chapitre 4 montre la méthode de passage du matching à un fichier XSL qui, à son tour, transforme un document XML qui respecte le standard KMEHR afin qu'il soit au format HL7.

Il serait intéressant d'étudier la possibilité d'automatiser la production de l'outil de traduction. Par exemple, sur base du fichier de matching retourné par "COMA 3.0" (ou un autre programme qui serait plus adapté à la tâche), le programme réalisé effectuerait lui-même la création du XSD permettant de la conversion du KMEHR au HL7.

Un tel outil permettrait de faciliter grandement la construction et la mise à jour de l'outil final, étant donné qu'il supprimerait la troisième étape de construction de l'outil de traduction. Cela permettrait d'effectuer très facilement des mises à jour lorsque les standards évoluent : il suffirait de mettre à jour les ontologies et d'adapter le matching en fonction pour pouvoir générer automatiquement un traducteur adapté.

La réalisation d'un tel projet serait cependant très lourde et complexe. Elle nécessiterait également de fournir au programme en entrée les informations sur l'implémentation, soit en modifiant la transformation des schémas XML en ontologies, soit en passant ces même schémas XML en entrée au programme, en parallèle avec les ontologies.

De plus, cela nécessiterait d'être extrêmement précis dans la création des ontologies, ou d'utiliser un programme spécifique, comme mentionné au point précédent. Toute erreur ou imprécision empêcherait la bonne construction du traducteur.

5.3 Utilisation des schémas XML plutôt que des ontologies

Il existe une solution intermédiaire à l'amélioration proposée à la section 5.2: l'utilisation des schémas XML pour le matching. Nous avons vu au chapitre 3 que le programme "COMA 3.0" permet d'effectuer un matching sur des schémas XML (au format XSD). Il serait intéressant d'évaluer la possibilité d'effectuer le matching directement à partir des schémas XML fournis par les standards

afin d'obtenir automatiquement un fichier XSL permettant la traduction du KMEHR. Un état de l'art des programmes existants actuellement ou la création d'un tel outil serait, certes, un travail conséquent, mais permettrait de gagner du temps et de faciliter la tâche de traduction.

Le revers de la médaille d'un tel système est qu'il gêne l'utilisateur par les détails de l'implémentation. En effet, cette méthode perd l'avantage acquis par le passage aux ontologies : l'abstraction.

On peut citer le programme "ALTOVA MapForce" [1] qui permet la création de fichiers XSLT directement à partir du matching des schémas XML. Ce genre de programme serait intéressant à tester et permettrait peut-être une accélération importante dans la création de l'outil de traduction.

5.4 Prolongation du projet

Cette section est la plus triviale : il s'agit de la continuation du travail réalisé jusqu'ici. Ce mémoire a posé les bases de la méthodologie nécessaire à la production d'un outil de conversion du KMEHR au HL7, mais seule la partie concernant l'identité du patient a été traitée. Il faudrait à présent continuer le travail en utilisant la méthode appliquée ici afin de pouvoir fournir un outil complet et fonctionnel, en collaboration avec le Réseau Santé Wallon, afin de pouvoir mettre en place l'interopérabilité recherchée.

La figure 5.1 présente le travail réalisé dans ce mémoire pour tester la méthodologie par rapport à l'entièreté des éléments à traiter : les essais ont été réalisés sur l'entité patient, entourée sur la figure. Le reste est encore à traiter pour traduire le SumEHR.

On peut supposer que, dans le futur, l'Europe viendra à formaliser les échanges de données internationales. Il sera intéressant de considérer, une fois une norme européenne choisie, de migrer du KMEHR vers celle-ci afin de faciliter les communications. En attendant, étant donné la bonne prise en charge du KMEHR et l'évolution des communications, il serait probablement fastidieux et inutile de migrer l'entièreté du système vers le HL7. Dans le contexte actuel, la construction d'un tel outil est la meilleure solution à court terme. Dans le contexte actuel, il est difficile de faire des prévisions à long terme.

5.5 Dictionnaires de données

Les dictionnaires de données sont un aspect important du travail de création de l'outil de traduction qui n'est pas abordé dans ce mémoire. Un message entre

1. `http://www.altova.com/mapforce.html`, Mise à jour : 2014, Consultation : 28/05/2014 10:36

les intervenants des services de soins utilise en effet un dictionnaire médical afin de référencer les éléments qu'il contient.

Par exemple, afin d'exprimer un type de maladie, un code est utilisé plutôt qu'un texte qui peut être ambigu ou incomplet. De manière similaire aux standards, il n'existe pas un dictionnaire unique. Dans les plus utilisés, on peut citer SNOMED CT ("Systematized Nomenclature of MEDicine Clinical Terms"), ICD ("International Classification of Diseases"), LOINC ("Logical Observation Identifiers Names and Codes"). Il faut donc considérer quel est le dictionnaire utilisé lors de la lecture du document.

Prolonger ce mémoire en réalisant l'outil de conversion fera probablement face à cet obstacle qu'est l'utilisation des dictionnaires dans les standards. Les codes exprimant les informations sont en effet très différents et ils nécessiteront une traduction spécifique. Bien sûr, une fois que le codage utilisé est retranscrit, le code en lui-même reste identique.

Par exemple, la section 1.5 a présenté un document XML au format KMEHR qui mentionnait l'élément suivant :

```
1  <cd S="ICD" SV="10">J00</cd>
```

Cet élément utilise le dictionnaire "ICD" dans sa version 10 et présente le code "J00". En HL7, les codes ressemblent à ceci :

```
1 <Code>
2   <Value>J00</Value>
3   <CodingSystem>ICD10</CodingSystem>
4 </Code>
```

Il faut donc posséder une table qui est capable de traduire "ICD v10" en "ICD". Le rassemblement des dictionnaires et de leurs codes dans les deux standards devra se faire afin de permettre la traduction.

5.6 Validation de l'outil

L'outil de traduction créé pour tester la méthode a été effectué en fonction des deux fichiers d'exemples fournis avec la spécification du SumEHR ([eHealth, 2014]). Grâce à la coopération du Réseau Santé Wallon, des fichiers de test ont été obtenus. Cependant, pour des raisons d'anonymat, ceux-ci ne possèdent pas d'informations sur l'entité patient. La méthodologie ayant été vérifiée uniquement sur l'entité patient, ces fichiers ne sont pas exploitables au stade de ce mémoire.

Il reste donc à vérifier l'outil mis en place sur un grand nombre de fichiers afin de tester son bon fonctionnement. A ce jour, n'ayant pas encore reçu de

données test sur le patient, il n'est pas possible de valider l'outil de manière complète. Ces tests devront donc être réalisés dans le futur.

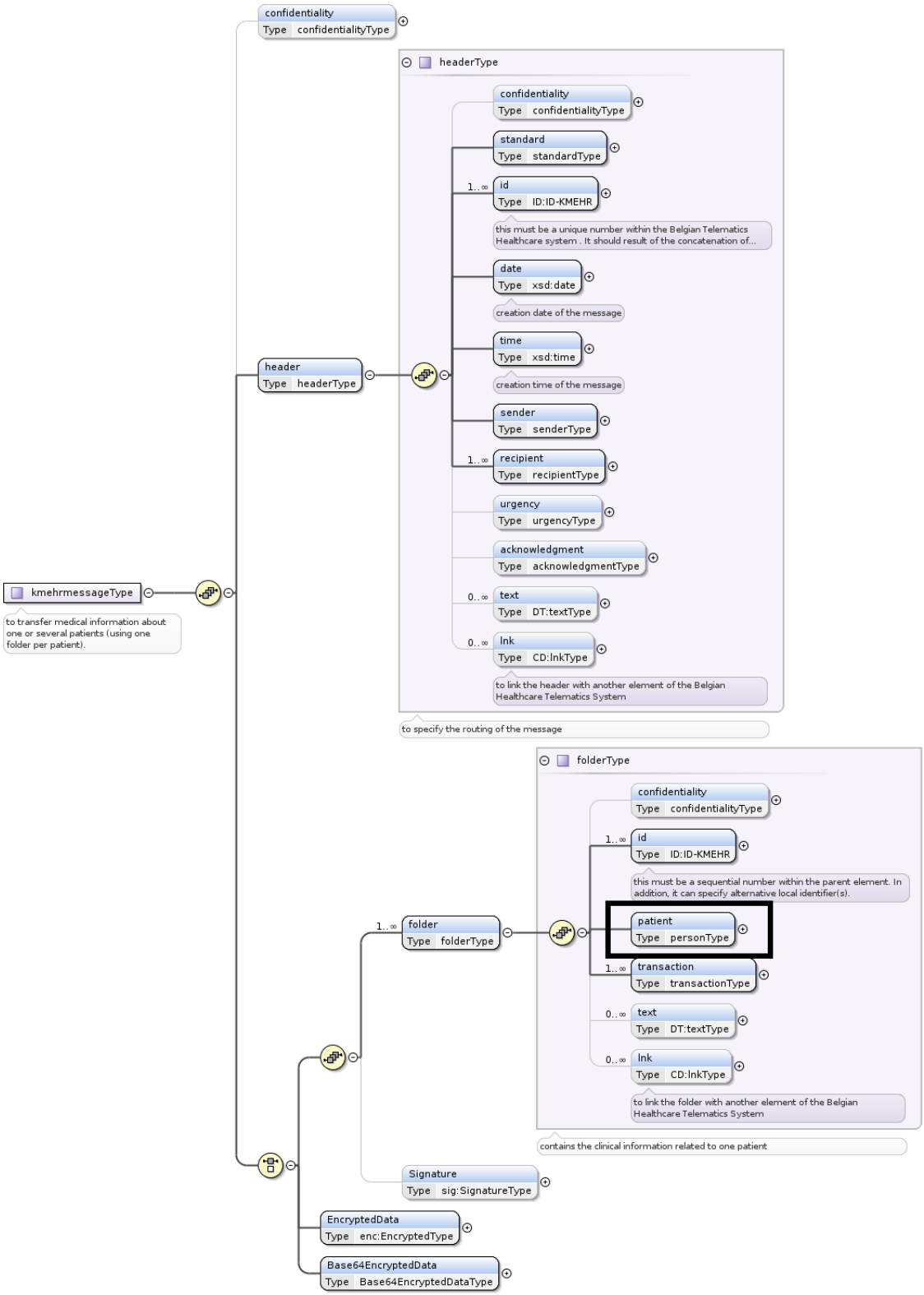

FIGURE 5.1 – Représentation du travail réalisé

Chapitre 6

Conclusion

Au cours de ce travail, nous avons pu suivre la création de la méthodologie de mise en place d'un outil de traduction du KMEHR en HL7, à travers chaque étape de conception : le passage aux ontologies, le matching de celles-ci et la création d'un outil de traduction. Ces trois étapes principales ont travaillé sur des niveaux d'abstractions différents : les données, la syntaxe et les concepts.

L'utilisation des ontologies a permis d'exprimer les concepts présents dans les standards de manière à faciliter leur compréhension et à permettre le matching. Celui-ci, à son tour, fait le lien entre les différents éléments des standards afin de structurer la construction du traducteur, qui est la dernière étape.

Si la fabrication d'un outil de traduction est rapide, elle ne peut se faire sans une analyse détaillée préalable. C'est le rôle de la méthode exposée dans ce mémoire. Celle-ci permet de structurer la construction de l'outil afin d'éviter des erreurs de conception et des pertes de temps en mettant en place les outils nécessaires à la construction du convertisseur KMEHR-HL7. Au vu de la quantité importante de travail et du manque de recherches sur le sujet, ce mémoire a été nécessaire afin d'ouvrir la voie. Il permettra dans le futur de créer, dans la continuité de ce projet, un service de liaison des services de santé des pays voisins.

Bien que l'élaboration de la méthodologie ait été parfois complexe ou déroutante et ait demandé une perpétuelle remise en question, elle représentera un acquis important pour la réalisation de la conversion d'un standard à l'autre.

On ne peut qu'espérer que, grâce à celle-ci, des tentatives de liaison internationales sur le plan médical verront le jour afin de permettre une meilleure connexion et un transfert aisé des informations médicales. Dans un futur proche, les membres d'un pays européen pourront aisément se faire soigner à l'étranger en ayant un transfert automatique de leur dossier médical, leur permettant un soin rapide et optimal, et la Belgique s'inscrira dans ce mouvement commun.

Ce travail aura apporté sa pierre à l'édifice de l'interopérabilité dans les services médicaux.

L'Europe viendra bientôt structurer officiellement les communications médicales internationales. Dans ce cadre, si l'utilisation du CDA se concrétise, il faudra peut-être organiser la migration des transmissions médicales du KMEHR vers le HL7. En attendant, on ne peut qu'être heureux que les services d'échanges d'information soient en bonne voie et que leur intégration progresse à travers le Réseau Santé Wallon via des messages KMEHR.

Bibliographie

[Benson, 2012] Benson, T. (2012). *Principles of Health Interoperability HL7 and SNOMED*. Springer.

[Bohring and Auer, 2005] Bohring, H. and Auer, S. (2005). Mapping xml to owl ontologies. Technical report, University of Liepzig.

[Corepoint Health, 2006] Corepoint Health (2006). The hl7 evolution. `http://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf`. Mise en ligne: 15/09/2006. Consultation : 16/05/2014, 09:56.

[eHealth, 2014] eHealth (2014). Introduction | kmehr. `https://www.ehealth.fgov.be/standards/kmehr/`. Mise en ligne: inconnue, Consultation : 16/05/2014, 10:42.

[Garcia, 2005] Garcia, R. (2005). Redefer. `http://rhizomik.net/html/redefer/#XSD2OWL`. Mise en ligne: 12/04/2005. Consultation : 20/05/2014, 17:12.

[Health Level Seven, 2011a] Health Level Seven (2011a). Health Level Seven International - Homepage. `http://www.hl7.org/index.cfm`. Mise en ligne: 02/08/2011. Consultation : 16/05/2014, 10:39.

[Health Level Seven, 2011b] Health Level Seven (2011b). HL7 Standards Product Brief - CDA Release 2. `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7`. Mise en ligne: 12/12/2011. Consultation : 16/05/2014, 11:40.

[Health Level Seven, 2014] Health Level Seven (2014). Hl7 - europe. `http://www.hl7.eu/`. Mise en ligne: inconnue, Consultation : 16/05/2014, 12:13.

[Horridge, 2011] Horridge, M. (2011). *A Practical Guide To Building OWL Ontologies, Using Protégé 4 and CO-ODE Tools*. The University Of Manchester, 1.3 edition.

[IEEE, 2003] IEEE (2003). Standard upper ontology working group (suo wg). `http://suo.ieee.org/`. Mise en ligne: 28/12/2003. Consultation : 18/05/2014, 08:15.

[INRIA, 2012] INRIA (2012). Amw. `http://wiki.eclipse.org/index.php/AMW`. Mise en ligne: 5/01/2012. Consultation : 21/05/2014, 11:56.

[Lacoste et al., 2011] Lacoste, D., Roy, S., and Prakash Sawant, K. (2011). An efficient xml to owl converter. Technical report, ISEC.

[Larousse, 2014] Larousse (2014). Définitions : interopérabilité - dictionnaire de français Larousse. `http://www.larousse.fr/dictionnaires/francais`. Mise en ligne: inconnue, Consultation : 16/05/2014, 10:10.

[Rahm et al., 2006] Rahm, E., Arnold, P., Do, H.-H., and Aumüller, D. (2006). Coma 3.0. `http://dbs.uni-leipzig.de/de/Research/coma.html`. Mise en ligne: 24/03/2006. Consultation : 18/05/2014, 15:13.

[RSW, 2010] RSW (2010). Réseau santé wallon. `https://www.reseausantewallon.be/Pages/default.aspx`. Mise en ligne: 21/09/2010. Consultation : 21/05/2014, 08:46.

[Shvaiko and Euzenat, 2005] Shvaiko, P. and Euzenat, J. (2005). Ontology matching. `http://www.ontologymatching.org/projects.html`. Mise en ligne: 18/09/2005. Consultation : 18/05/2014, 09:39.

[Stanford Center for Biomedical Informatics Research, 2014] Stanford Center for Biomedical Informatics Research (2014). Protégé. `http://protege.stanford.edu/`. Mise en ligne: 21/02/2014. Consultation : 17/05/2014, 09:16.

[Thuy et al., 2011] Thuy, P. T. T., Lee, Y.-K., and Lee, S. (2011). S-trans: Semantic transformation of xml healthcare data into owl ontology. Technical report, Kyung Hee University.

[Tidwell, 2008] Tidwell, D. (2008). *XSLT, Second Edition*. O'REILLY.

[Uthayasanker.T, 2010] Uthayasanker.T (2010). Optima. `http://cobweb.cs.uga.edu/~uthayasa/Optima/Optima.html`. Mise en ligne: 30/09/2010. Consultation : 21/05/2014, 10:30.

[W3C, 2013] W3C (2013). Owl - semantic web standards. `http://www.w3.org/2001/sw/wiki/OWL`. Mise en ligne: 11/12/2013. Consultation : 17/05/2014, 08:53.

Annexe A

Contenu du CD-ROM

Cette annexe présente le contenu du disque optique fourni avec le mémoire.

1. Fichier d'ontologie du SumEHR.
   Ontologie réalisée sur l'entité patient du SumEHR, dans le cadre du chapitre 2. Il a été réalisé avec le programme "Protégé".
2. Fichier d'ontologie du CDA.
   Fichier similaire pour le CDA.
3. Fichier exporté du matching de l'entité patient.
   Exportation du matching des ontologies réalisé au chapitre 3.
4. Fichier XSL de traduction de l'entité patient.
   Code XSLT permettant la traduction de l'entité patient du SumEHR vers l'entité du patient du CDA du chapitre 4.
5. Dossier du standard du CDA.
   Ce dossier, disponible sur le site du HL7, contient la définition du standard du CDA.
6. Dossier du standard du SumEHR.
   Ce dossier contient les spécifications du SumEHR et du KMEHR.
7. Mémoire en version PDF.

Annexe B

Ontologies réalisées

Cette annexe présente la représentation graphique par Protégé des ontologies des entités patients du SumEHR et du CDA.

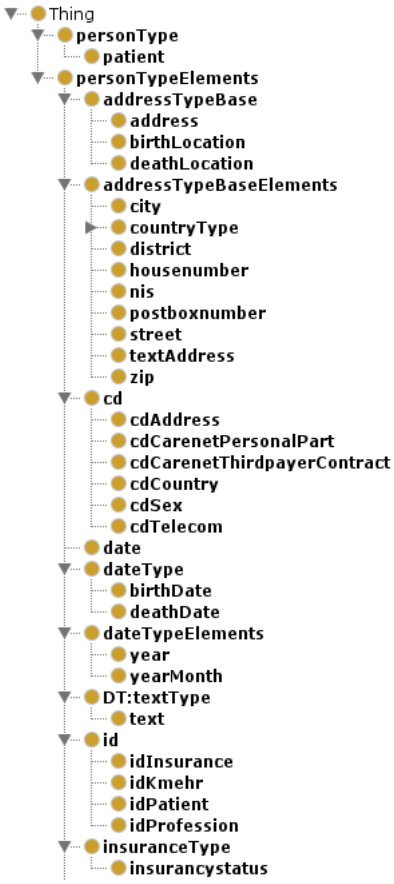

FIGURE B.1 – Ontologie de l'entité patient du SumEHR - partie 1

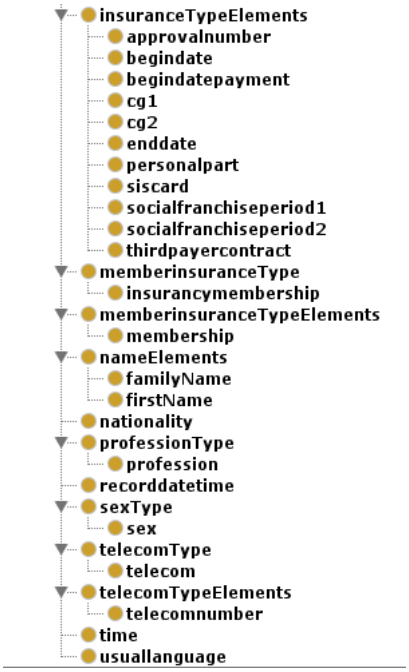

FIGURE B.2 – Ontologie de l'entité patient du SumEHR - partie 2

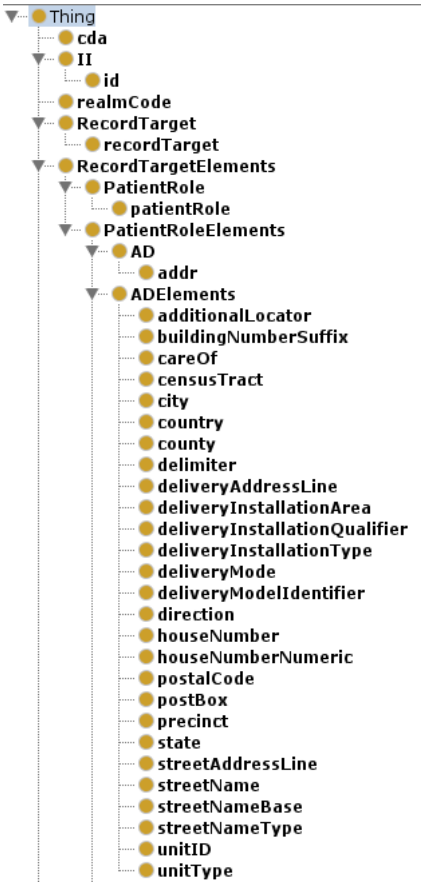

FIGURE B.3 – Ontologie de l'entité patient du CDA - partie 1

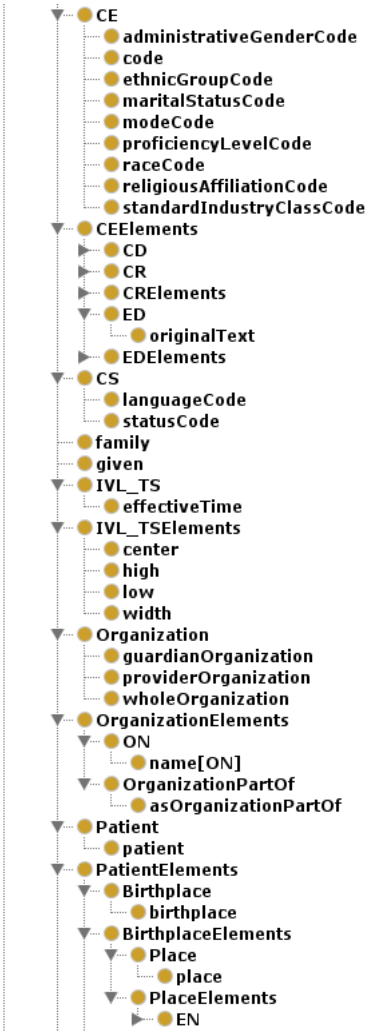

FIGURE B.4 – Ontologie de l'entité patient du CDA - partie 2

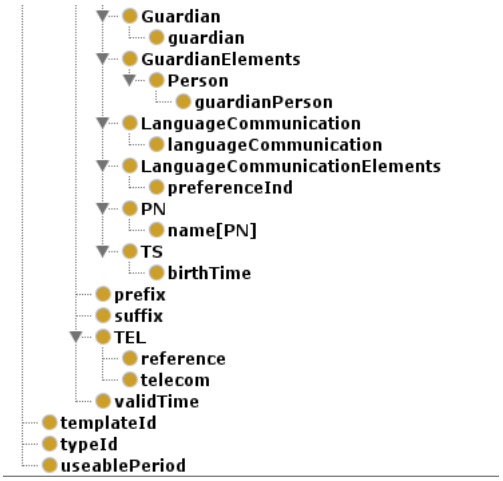

FIGURE B.5 – Ontologie de l'entité patient du CDA - partie 3