ANR programme ARPEGE 2008

Systèmes Embarqués et Grandes Infrastructures

Projet SELKIS : Une méthode de développement de systèmes d'information médicaux sécurisés : de l'analyse des besoins à l'implémentation.

ANR-08-SEGI-018

Février 2009 - Décembre 2011

INTEGRATION DES PROPRIETES DE SECURITE DANS UML

Livrable n° 2.1

Jacky Akoka (ISID CEDRIC CNAM Paris)
Tatiana Aubonnet (ISID CEDRIC CNAM Paris)
Jean Sylvain Bucumi (ISID CEDRIC CNAM Paris)
Isabelle Comyn-Wattiau (ISID CEDRIC CNAM Paris)
Akram Idani (VASCO LiG Grenoble)
Mohamed Amine Labiadh (VASCO LiG Grenoble)
Nadira Lammari (ISID CEDRIC CNAM Paris)
Yves Ledru (VASCO LiG Grenoble)
Jean-Luc Richier (VASCO LiG Grenoble)

Février 2010

SOMMAIRE

1. Introduction

La recherche en ingénierie de systèmes d'information a mené à différentes propositions de (méta) modèles et de processus. Le volet sécurité est devenu important dans les critères de qualité des nouveaux systèmes d'information. Certaines approches ont intégré quelques aspects de la sécurité, notamment les contrôles d'accès, mais aucune ne les prend tous en considération.

Dans le projet SELKIS, nous avons l'ambition de couvrir l'ensemble des propriétés de base, à savoir la Disponibilité, l'Intégrité, la Confidentialité et la Traçabilité (DICT). Notre postulat de départ est que le fait de prendre en charge les propriétés de sécurité très tôt permet d'obtenir au final un système de meilleure qualité, qui répond parfaitement aux besoins des différents utilisateurs et intègre les considérations de sécurité pour toutes les parties prenantes dans le système d'information.

L'analyse des systèmes d'information révèle que les besoins découlent des objectifs (buts) de l'organisation, des acteurs et de l'environnement dans lequel évolue le système d'information. De plus, les propriétés DICT sont très générales et indépendantes des plateformes et modèles d'implémentation.

SELKIS se propose de prendre en compte la sécurité en combinant une approche dirigée par les buts et un processus d'ingénierie des systèmes d'information fondé sur MDA. Un état de l'art, qui sera détaillé par la suite, a permis de constater qu'aucune approche ne prend en compte tous les aspects de sécurité. Une grande partie de ces approches est plutôt orientée vers l'analyse des risques de sécurité (malveillances, mauvais usages, etc.) et vers la protection.

Pour permettre une bonne appréhension de la variété des travaux menés dans ce domaine, nous présentons un état de l'art incluant les taxonomies de la sécurité, les processus de sécurité et, enfin, les méthodes de conception incluant, en partie au moins, la sécurité.

A partir de cette synthèse et d'un tour d'horizon des méta-modèles des différents aspects, nous proposons la définition d'une approche de conception intégrant la sécurité (MoSIS) et d'un Métamodèle Intégré des Systèmes d'Information Sécurisés.

2. Etat de l'art

La sécurisation des systèmes d'information constitue, de nos jours, l'une des thématiques majeures de l'ingénierie des systèmes d'information (SI). Elle nécessite une prise en compte méthodologique dès la conception du SI. Dans cet état de l'art, nous commençons donc par un survol des méthodes de conception et de développement des SI sécurisés. Nous proposons une approche dirigée par les modèles. C'est la raison pour laquelle cet état de l'art fait aussi une synthèse des différents méta-modèles de la littérature, en nous limitant à ceux qui nous semblent pertinents pour la représentation des concepts et propriétés de la sécurité. Enfin, nous proposons une synthèse comparative des profils UML liés à la sécurité.

2.1 Méthodes de conception et développement de systèmes d'information sécurisés

Avant de proposer notre approche fondée sur une architecture MDA, nous présentons un état de l'art relatif aux méthodes existantes dans la littérature. Nous proposons ci-après une catégorisation de ces méthodes. Rappelons que ces dernières couvrent tout ou partie des étapes du cycle de vie d'une application. Pour chacune des catégories, nous procéderons à une comparaison sur la base des éléments suivants :

- les étapes du cycle de vie couvertes par l'approche,
- les types de besoins de sécurité pris en compte (contrôle d'accès, traçabilité, etc.),
- les modèles et langages utilisés,
- le degré d'intégration entre besoins fonctionnels et besoins non fonctionnels.

La littérature dans ce domaine réunit des travaux de recherche proposant des processus et des méthodes1 (processus et modèles) pour la prise en charge de la sécurité dans le cycle de développement des applications. Parmi les processus existant nous pouvons citer le processus VOSREP (View Point Oriented Security Engineering) de [Agarwal, 2008], processus CLAPS2 (Comprehensive Lightweight Application Security Process) et le processus de [Haley et al., 2008]. Quant aux méthodes, c'est-à-dire les démarches disposant aussi des modèles, on peut les catégoriser en trois : les méthodes orientées but, les méthodes orientés modèles et les méthodes orientées aspect. La plupart des méthodes orientées but s'appuient sur les méta-modèles de KAOS, I*, SI* ou encore ceux de SecureTropos. Les méthodes orientées modèles utilisent UMLsec ou SecureUML.

Pour faciliter la comparaison de ces méthodes, nous les différencions de la façon suivante :
- méthodes intégrant l'élicitation et/ou la spécification des besoins de sécurité,
- méthodes pour la conception d'applications sécurisées,
- méthodes englobant la spécification des besoins et la conception,
- méthodes couvrant les étapes de conception et d'implémentation d'applications.

[1] Selon Booch [Booch, 1991] une méthode est un processus permettant de générer un ensemble de modèles qui décrit divers aspects d'un logiciel en cours de construction en utilisant une certaine notation bien définie.
[2] http://searchsoftwarequality.techtarget.com/searchAppSecurity/downloads/clasp_v20.pdf

2.1.1. Méthodes intégrant l'élicitation et/ou la spécification des besoins en sécurité

Nous avons recensé sept méthodes principales qui entrent dans cette première catégorie ([Mead, 2008a], [Fontaine, 2001], [Massacci, 2008], [He, 2009], [Lamsweerde, 2009], [Mellado, 2007], [Liu, 2003], [Sindre, 2005]).

Dans [Mead, 2008a] les auteurs proposent une intégration du processus SQUARE (Security Quality Requirements Engineering) dans le RUP. SQUARE est une approche pour l'élicitation, l'analyse, la catégorisation, la prioritisation et la documentation des besoins en sécurité [Mead, 2005]. Il permet à la suite de l'élicitation des objectifs de sécurité et de l'analyse des risques de mettre en évidence les besoins en sécurité. Notons aussi qu'une proposition d'intégration de SQUARE avec la méthode agile DSDM est publiée dans [Mead, 2008b].

En se fondant sur le framework de SI* [Zannone, 2008], [Massacci, 2008] décrit une approche dirigée par les modèles de spécification des besoins de sécurité en contrôle d'accès. Bien qu'elle ne couvre qu'une étape du cycle de vie d'une application, cette approche intègre les besoins fonctionnels avec ceux liés au contrôle d'accès. Son point fort aussi est la prise en compte du contexte social.

Dans [He, 2009], les auteurs proposent une approche intitulée ReCAPS (Requirement-based access Control Analysis and Policy Specification), guidée par des heuristiques. Ces dernières permettent de dériver des politiques d'accès à partir de plusieurs sources (spécification des besoins fonctionnels, schéma conceptuel de la base de données, documents de conception, etc.). Une fois dérivés, les besoins en contrôle d'accès sont intégrés aux besoins fonctionnels.

KAOS est aussi une méthode d'élicitation et de spécification des besoins fonctionnels et non fonctionnels, incluant la sécurité [Lamsweerde, 2009]. L'élicitation des besoins se fait par raffinement des buts. Le résultat de ce raffinement est une instanciation du méta-modèle des buts. D'autres diagrammes sont aussi utilisés pour la spécification des besoins, laquelle est décrite à l'aide du diagramme des cas d'utilisation UML. Ce dernier est obtenu au moyen du mécanisme d'opérationnalisation des buts. Notons que, bien que KAOS offre la possibilité de décrire, dans le modèle des buts, les buts fonctionnels, les buts de sécurité ainsi que les anti-buts, elle ne propose cependant qu'un algorithme de traduction des besoins fonctionnels en cas d'utilisation. Elle ne permet pas de dériver les cas de sécurité.

L'approche décrite dans [Mellado, 2007] a la particularité d'intégrer dans le processus UP, le processus Security Requirements Engineering Process (SREP) d'expression des besoins de sécurité et les standards critères communs CC (common criteria) pour la spécification des besoins de sécurité. Elle se fonde sur le méta-modèle SSR (Security Ressources Repository). Dans ce dernier, sont représentés et classifiés les besoins en sécurité, les menaces ainsi que les cas d'utilisations associés.

[Liu, 2003] s'appuie sur le « framework » de i* pour la spécification des besoins fonctionnels et non fonctionnels, y compris ceux liés à la sécurité. L'utilisation de i* rend possible la prise en compte du contexte organisationnel dans lequel évoluera la future application dès la phase de spécification des besoins.

[Sindre, 2005] utilise UMLSec pour la spécification des besoins fonctionnels et de sécurité. Ainsi les cas fonctionnels, ceux liés aux contrôle d'accès et les anti-cas (misuse cases) sont représentés dans un seul diagramme des cas de l'application à sécuriser. Cependant, les auteurs ne proposent pas de guide facilitant l'obtention des cas de sécurité.

Mentionnons enfin, dans cette catégorie, la méthode proposée dans [Fontaine, 2001]. C'est une méthode de transformation du modèle des buts de KAOS en Ponder, langage de spécification des politiques de sécurités dans le contexte des systèmes distribués. Comme beaucoup d'autres méthodes, celle-ci ne procède pas à une intégration des besoins de sécurité et des besoins fonctionnels dans sa démarche.

Le tableau ci-dessous (tableau 1) synthétise ces approches. La plupart d'entre elles ont tenté d'aller au-delà de la spécification des besoins de sécurité de contrôle d'accès et d'usage. De plus la majorité d'entre elles proposent une intégration entre besoins fonctionnels et de sécurité.

APPROCHE	ETAPES DU CYCLE DE VIE	BESOINS EN SECURITE	MODELES/ LANGAGE/DOC	INTEGRATION DES BESOINS
[Massacci, 2008]	Spécification et analyse des besoins	Contrôle d'accès	modèles de SI* [Zannone, 2008]	oui?
[He, 2009]	Elicitation et spécification des besoins en contrôle d'accès	Contrôle d'accès	modèle E/R documents sur la spécification des besoins fonctionnels Documents sur la conception	oui
[liu, 2003]	Spécification et analyse des besoins en sécurité	sécurité	modèle i*	oui
KAOS [Lamsweerde, 2009]	Spécification et analyse des besoins dont la sécurité	sécurité	KAOS UML	oui?
[Mellado, 2007]	Elicitation et spécification des besoins	tous les aspects de sécurité	modèle SRR (Security resources repository) éventuellement UMLsec	non
[Sindre, 2005]	Elicitation et spécification des besoins	contrôle d'accès et d'usage, menaces, contre-mesures	UMLsec	
[Mead, 2008a]	Elicitation et spécification des besoins	sécurité	cas et anti-cas (misuse cases)	oui
[Fontaine, 2001]	spécification et analyse des besoins	Contrôle d'accès	KAOS langage Ponder	non

Tableau 1. Les méthodes couvrant l'élicitation des besoins

2.1.2. Méthodes pour la conception d'applications sécurisées

Dans cette catégorie, nous avons recensé quatre méthodes ([Georg, 2009], [Breu 2007], [Crook, 2005], [He, 2003]).

[Georg, 2009] propose une démarche orientée aspect pour la conception d'une application sécurisée. La première phase de cette démarche consiste en une analyse des risques servant à identifier les menaces. Ces dernières, une fois identifiées, sont modélisées sous forme d'aspects. La seconde phase permet de générer un modèle d'anti-cas (misuse model) à partir

de la composition des aspects avec le modèle des cas d'utilisation. Le modèle généré est ensuite analysé pour évaluer sa résistance aux attaques. Le résultat de cette analyse est l'identification des contre-mesures. Le point fort de cette démarche est sa résistance aux changements des besoins de sécurité. Cependant, le problème principal réside dans sa limitation à une phase du cycle de vie d'une application.

Dans [Breu 2007], seuls les besoins en droit d'accès sont pris en compte. Ils sont intégrés juste avant la conception de l'application selon une approche objet fondée sur la notation UML. La conception des droits d'accès est réalisée avec P-MOS, langage supportant les prédicats de premier ordre. Elle est réalisée à partir d'un modèle de droits d'utilisateur obtenu sur la base des cas d'utilisation, du diagramme des classes et d'une expression textuelle des droits. Cette approche a l'avantage d'être multi-plateforme.

[Crook, 2005] s'appuie sur un méta-modèle décrivant le contexte organisationnel de l'application pour concevoir les contrôles d'accès. Ce méta-modèle, qui est une extension de i*, est formalisé en Z. En plus de sa limitation à un aspect de la sécurité, cette approche est mono-plateforme.

Enfin [He, 2003] propose une démarche orientée but pour la conception des contrôles d'accès selon le modèle RBAC (Role-Based Access Control). Pour ce faire, les auteurs utilisent un méta-modèle de données orienté contexte dans lequel les données sont contextualisées par les éléments de sécurité. Dans leur démarche, les auteurs utilisent la documentation sur les processus métiers et la spécification de besoins.

Le tableau ci-dessous (Tableau 2) résume les points saillants de chaque méthode. Comme le montre ce tableau, la quasi-totalité des approches se focalisent sur les contrôles d'accès.

APPROCHE	ETAPES DU CYCLE DE VIE	BESOINS EN SECURITE	MODELES/ LANGAGE/DOC	INTEGRATION DES BESOINS
[Breu 2007]	Conception	Contrôle d'accès	modèle de droits d'utilisateurs (basé sur cas d'utilisation, diagramme des classes et expression textuelle des droits) P-MOS (langage d'expression prédicats de premier ordre)	oui
[Crook, 2005]	conception	Contrôle d'accès	méta-modèle décrivant le contexte organisationnel en Z	non
[He, 2003]	conception de la sécurité	contrôle d'accès	modèle de données orienté contexte RBAC	non
[Georg, 2009]	conception	sécurité	cas et anti-cas (misuse cases) UML aspects génériques décrits en UML	oui

Tableau 2. Les méthodes de conception d'applications sécurisées

2.1.3. Méthodes englobant la spécification des besoins et la conception

Nous nous contentons dans cette catégorie d'analyser deux méthodes qui nous semblent les plus représentatives de leur catégorie : MSBE [Jürjens, 2008] et SecureTropos [Mouratidis, 2009].

MBSE (Model-Based Security Engineering) est une approche de spécification et conception des besoins de sécurité pour les applications mobiles [Jürjens, 2008]. La spécification des besoins dans MBSE consiste en le recensement des besoins fonctionnels et des besoins en sécurité, dans deux modèles séparés : le modèle d'usage pour les besoins fonctionnels et le modèle des besoins de sécurité pour l'autre catégorie. Le processus de conception consiste en la génération du modèle d'implémentation, exprimé en UMLSec, par adaptation du modèle des besoins en sécurité au modèle d'usage. Bien que couvrant deux étapes du cycle de vie d'une application, MBSE ne guide pas le concepteur dans sa démarche. Cependant, elle présente l'avantage de ne pas se limiter aux contrôles d'accès et des usages.

[Mouratidis, 2009] décrit SecureTropos, une approche orientée agent pour le développement de systèmes d'information pour la santé. Cette approche est fondée sur la méthode Tropos pour le développement de systèmes multi-agents. SecureTropos, au même titre que Tropos, construit un système à travers une série de transformations depuis la spécification des besoins jusqu'à la conception détaillée tout en tenant compte du système cible et de son environnement organisationnel. C'est une approche orientée but de par sa vision du système en tant qu'ensemble d'acteurs dépendant les uns des autres pour accomplir un but. Dans SecureTropos, les besoins fonctionnels et de sécurité sont représentés dès la phase de spécification des besoins à travers le diagramme des acteurs. En effet, sont modélisés dans ce dernier aussi bien les acteurs avec leurs buts et leurs dépendances que les contraintes de sécurité associées aux buts. Notons que dans [Matulevicius, 2008] les auteurs étendent SecureTropos à la gestion des risques.

Le tableau ci-après (Tableau 3) résume ces approches. Bien que les deux méthodes prétendent prendre en charge tous les besoins en sécurité, il n'apparaît pas évident comment est prise en charge, par exemple, la propriété de traçabilité.

APPROCHE	ETAPES DU CYCLE DE VIE	BESOINS EN SECURITE	MODELES/ LANGAGE/DOC	INTEGRATION DES BESOINS
SecureTropos [Mouratidis, 2009]	spécification des besoins à la conception détaillée	sécurité	SecureTropos Patterns AUML	oui
Approche MBSE [Jürjens, 2008]	spécification des besoins à la conception	sécurité	Modèle d'usage Modèle des besoins en sécurité Modèle d'implémentation UMLsec	oui

Tableau 3. Méthodes englobant spécification des besoins et conception

2.3.4. Méthodes couvrant les étapes de conception et d'implémentation d'application

Dans cette catégorie d'approches nous avons retenu celle de [Basin, 2006] et celle de [Hafner, 2008].

La première est une démarche fondée sur l'architecture MDA. Elle permet de traduire une modélisation conceptuelle des contrôles d'accès, exprimée en SecureUML, selon le modèle RBAC.

La seconde approche a été mise en œuvre pour prendre en charge les contrôles d'accès et des usages dans le secteur médical. Dans [Hafner, 2008] les auteurs décrivent un processus fondé sur SECTET. Ce dernier est un « framework » conforme à MDA. Pour la conception des aspects sécuritaires, il s'appuie sur trois méta-modèles : de sécurité, de ressources et de rôles. Ces trois méta-modèles sont utilisés pour le passage vers le modèle UCON. Un prototype permettant de mettre en œuvre le modèle UCON à l'aide du langage XACML est prévu.

Le tableau ci-après (Tableau 4) synthétise cette famille d'approches. Notons que, même si ces deux approches sont fondées sur l'architecture MDA, elles n'intègrent pas les besoins de sécurité et les besoins fonctionnels. De plus elles ne prennent pas en charge toutes les propriétés DICT.

APPROCHE	ETAPES DU CYCLE DE VIE	BESOINS EN SECURITE	MODELES/ LANGAGE/DOC	INTEGRATION DES BESOINS
[Basin, 2006]	Conception et implémentation	Contrôle d'accès	modèle RBAC SecureUML	?
[Hafner, 2008]	Conception et implémentation	Contrôle d'accès et des usages	Méta-modèle de sécurité Méta-modèle de ressource (données) méta-modèle des rôles UCON SECTECT-PL XACML	non

Tableau 4. Les méthodes de conception et implémentation des applications sécurisées

En résumé, notre revue de l'état de l'art nous a permis de constater la richesse de la littérature dans le domaine de la conception et du développement d'applications sécurisées. Un bon nombre de ces méthodes adoptent l'architecture MDA mais se focalisent cependant sur le passage du niveau Platform Independent Model (PIM) vers le niveau Platform Specific Model (PSM) ou encore du PIM vers le codage. De plus, une bonne partie de ces travaux de recherche s'intéressent à la spécification des besoins de sécurité mais, le plus souvent, ils ne les intègrent pas avec les besoins fonctionnels et négligent le contexte organisationnel dans lequel auront à opérer ces applications. Aussi, la plupart des approches sont centrées sur les contrôles d'accès et d'usage et ne prennent, par conséquent, pas en compte toutes les propriétés DICT. Enfin, une grande attention est portée au développement d'applications sécurisées, c'est à dire à la création d'applications « from scratch » alors que les organisations, à l'heure actuelle, sont la plupart du temps demanderesses d'évolution de leurs applications existantes par intégration de nouveaux besoins dont ceux de sécurité.

2.2 Les méta-modèles

Notre approche est fondée sur l'ingénierie dirigée par les modèles. Il s'agit de construire un modèle de conception du SI et de le dériver successivement à l'aide de règles de transformation. Ces règles opèrent sur des méta-modèles de SI sécurisés. La représentation des propriétés de sécurité nécessite la possibilité d'intégrer dans ces méta-modèles : 1) les processus de gestion pour lesquels les SI servent de support, 2) les besoins fonctionnels et non fonctionnels que les SI vont prendre en charge et enfin 3) la structure de l'organisation dans laquelle le SI est exploité.

2.2.1 Les méta-modèles de processus

La littérature comprend un nombre important de méta-modèles de processus. On peut les classifier selon l'objectif attendu de ces méta-modèles. La thèse [Hug 2009] fait, par exemple, un état de l'art très exhaustif des méta-modèles de processus utilisés dans l'ingénierie des systèmes d'information (SI). Un grand nombre de méta-modèles ont été proposés à un niveau plus générique pour modéliser les processus métiers. [Morley et al. 2005] est un exemple de ce type. Les efforts de l'OMG autour de BPDM (Business Process Definition Meta-Model) ont aussi pour objectif la modélisation des processus métiers [OMG, 2006]. D'autres méta-modèles de processus ont été proposés pour la représentation de processus au moyen de « workflows » pour les systèmes de gestion de « workflows ». Les travaux de [Kradolfer, 2000], [Brambilla et al., 2007] et [Eder et al. 2002] peuvent être placés dans cette catégorie. [Breton et al. 2000] compare les méta-modèles de processus industriels. Enfin, plusieurs articles ont proposé des méta-modèles permettant la comparaison et l'évaluation de langages de modélisation de processus ou d'outils de représentation des processus [List et al. 2006, Lei et al. 1997]. Enfin, certains méta-modèles ont intégré de façon succincte ou détaillée pour représenter la sécurité dans les processus [Zannone 2008, Saïdani et al. 2006]. Nous décrivons brièvement chaque méta-modèle étudié en insistant sur ses spécificités. Puis nous proposons une comparaison de ces modèles. Enfin, nous utilisons les résultats de cette comparaison pour proposer un méta-modèle de processus qui sera le support d'une intégration de la sécurité du système d'information dès la conception.

a) Méta-modèle de Morley

L'objectif de la méta-modélisation proposée par Morley et al est pédagogique. Elle prépare à la modélisation des processus métiers. Le méta-modèle décrit des processus qui obéissent à un objectif. Ils peuvent être globaux ou détaillés. Les processus détaillés peuvent présenter des variantes qui sont aussi des processus. Un processus détaillé peut faire l'objet de scenarii, qui représentent les simulations du processus. Un processus peut être décomposé en activités. Morley propose aussi une classification des événements en : déclencheurs, modificateurs, interrupteurs et une seconde classification en : temporels, externes, internes. Les activités sont classées en trois sous-classes : production, contrôle et communication (Figure 1).

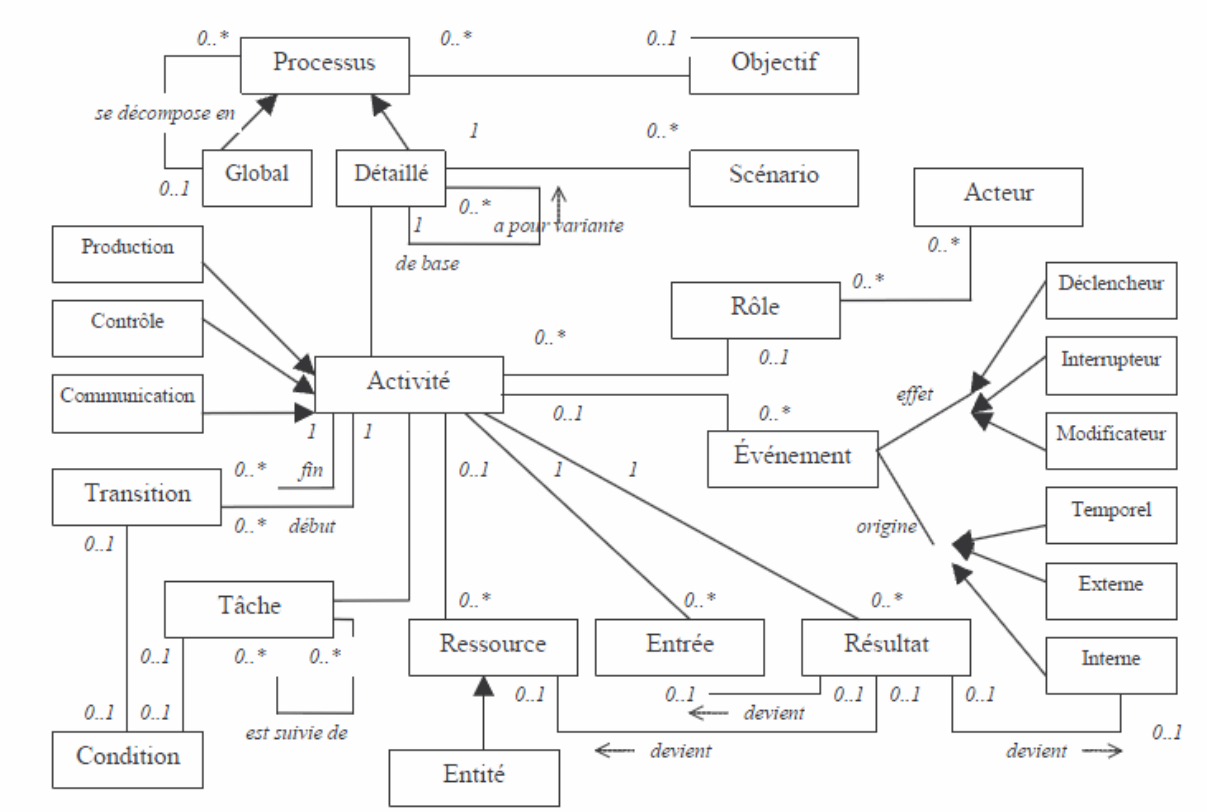

Figure 1. Méta-modèle de processus [Morley et al. 2005]

b) Le méta-modèle de Kradolfer

L'objectif de la méta-modélisation proposée par Kradolfer est la représentation de workflows. Le workflow est décrit au travers de différents aspects :

1) L'aspect fonctionnel/structurel décrit le workflow en termes d'activités, de variables et de blocs.
2) L'aspect informationnel décrit les interfaces et leurs paramètres. Il comprend les flux de données entrants et sortants entre les différents workflows.
3) L'aspect comportemental représente les conditions notamment temporelles d'exécution du workflow.
4) L'aspect organisationnel considère les entités organisationnelles qui prennent en charge les activités du workflow.

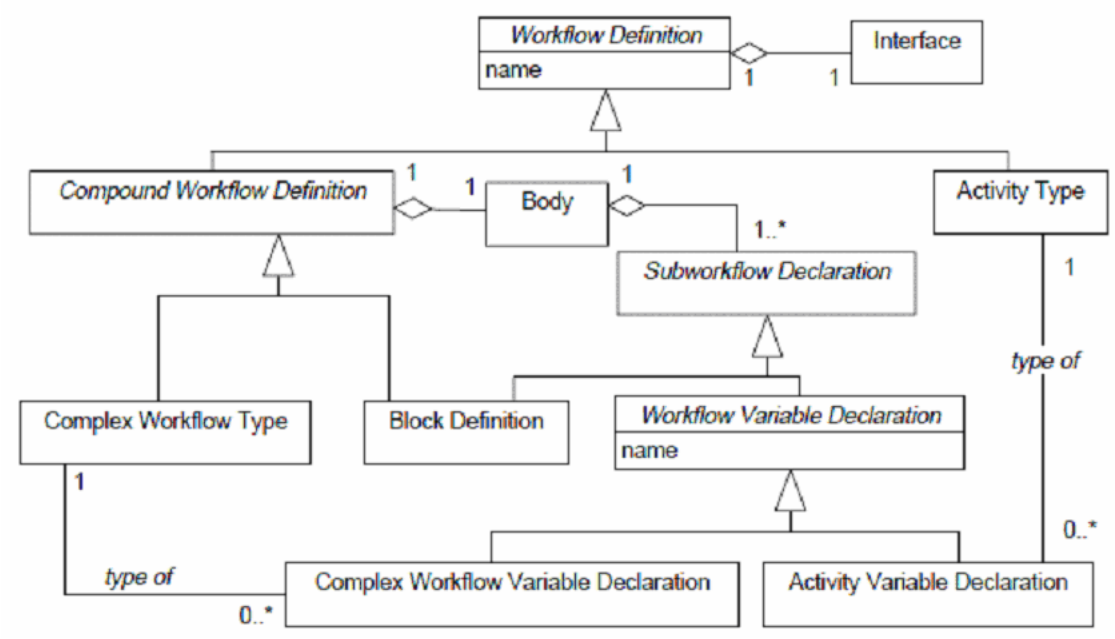

Figure 2. L'aspect fonctionnel/structurel du méta-modèle

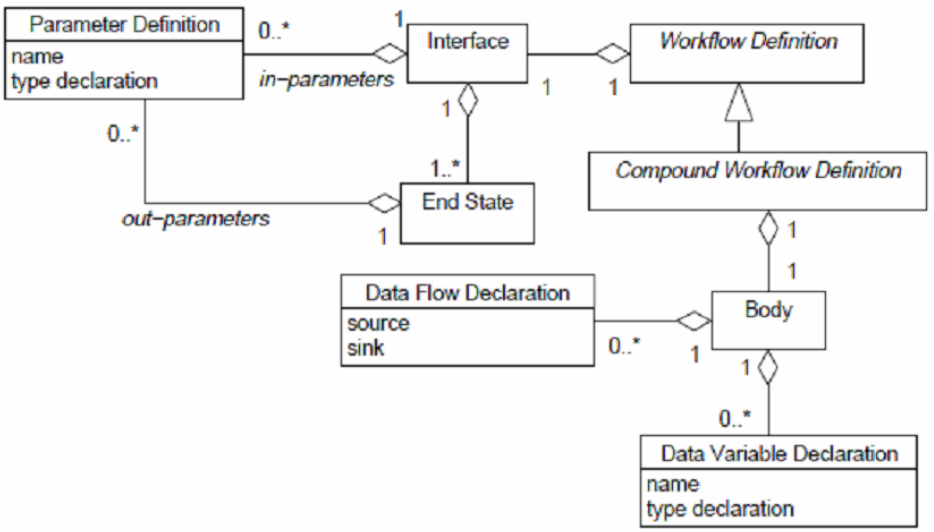

Figure 3. L'aspect informationnel du méta-modèle

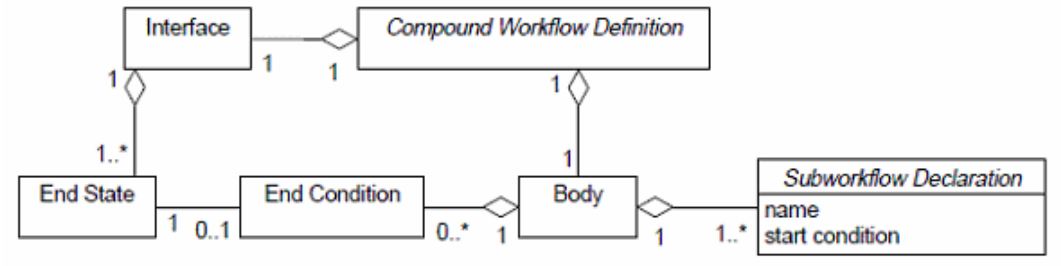

Figure 4. L'aspect comportemental du méta-modèle

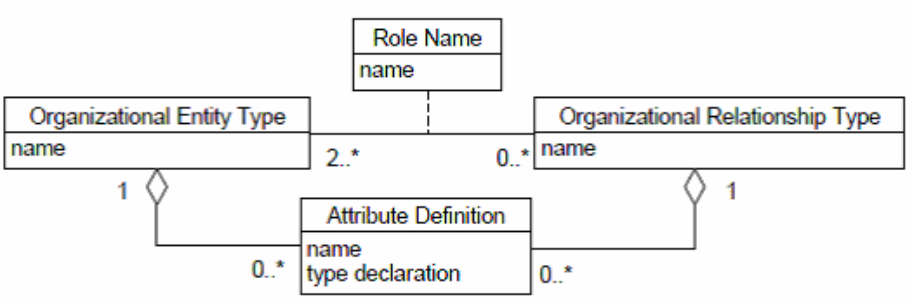

Figure 5. L'aspect organisationnel du méta-modèle

c) Le méta-modèle de List

[List et al. 2006] propose un méta-modèle générique dans le but d'évaluer les différents langages de modélisation de processus. Il s'appuie sur le framework de Curtis [Curtis et al., 1992] qui, le premier, a proposé de modéliser les processus selon quatre perspectives : fonctionnelle, comportementale, organisationnelle et informationnelle. List et al. l'ont enrichi d'une cinquième perspective, appelée contexte du processus sur le plan business : buts, mesures, livrables, propriétaire, type, client. Le méta-modèle de List est manifestement celui qui prend en compte le plus grand nombre de perspectives et qui les intègre avec une granularité fine.

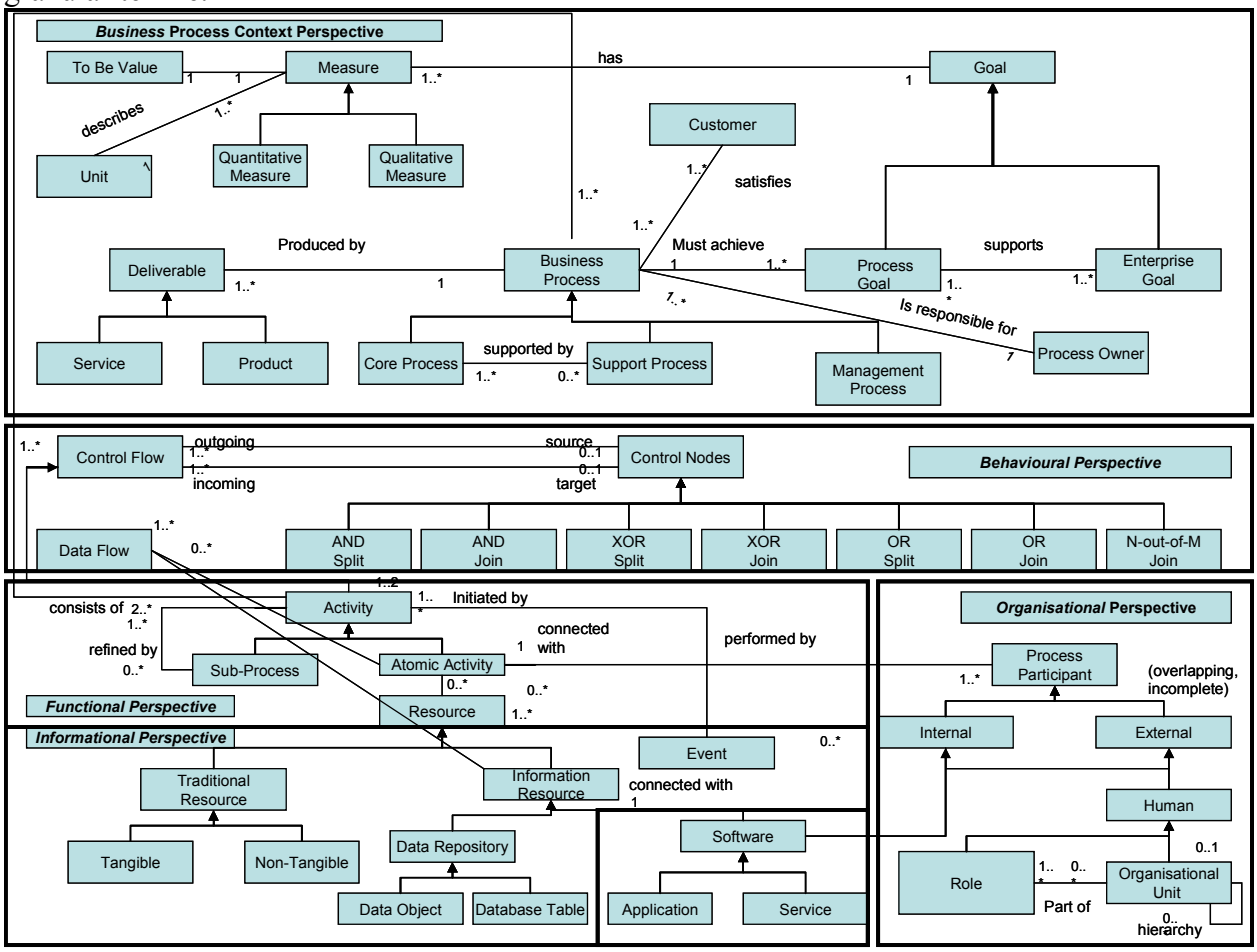

Figure 6. Le méta-modèle de List

d) Le méta-modèle de Saidani

Saïdani et al ont pour objectif la représentation des processus flexibles [Saidani et al. 2006]. Les processus sont décrits très succinctement dans ce méta-modèle. Ils sont composés de missions répondant à des buts opérationnels qui satisfont des opérations.

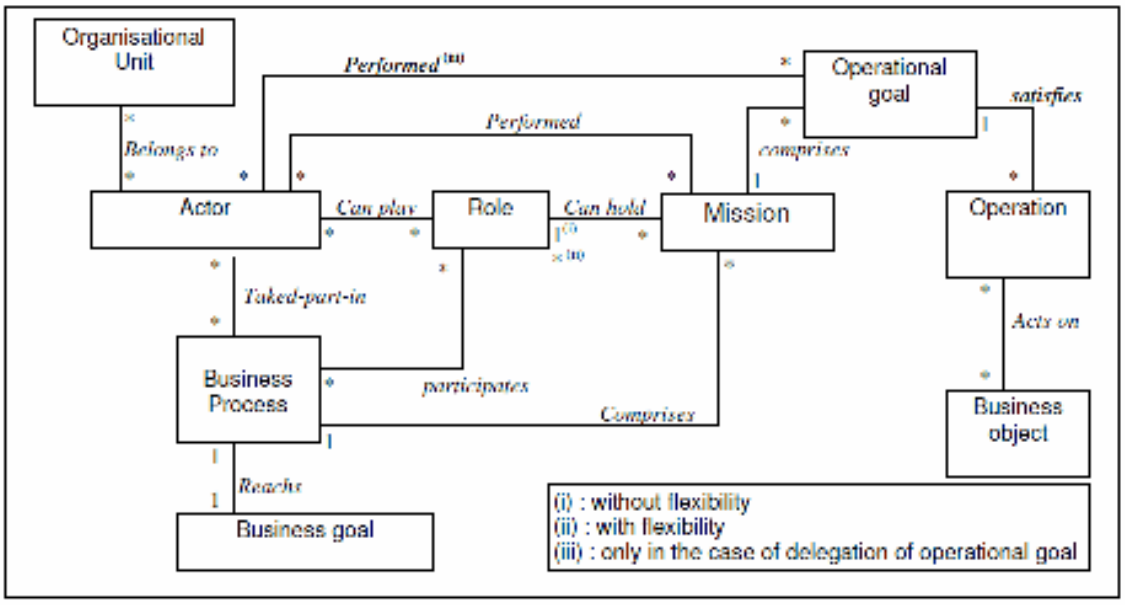

Figure 7. Le méta-modèle de Saidani

La prise en compte de la flexibilité comprend en particulier la possibilité de déléguer la réalisation des processus. Plus précisément, un acteur peut déléguer à un autre acteur, à un rôle. Un rôle peut lui aussi déléguer à un rôle. Le méta-modèle de délégation est représenté ci-dessous. Il intègre le contexte, qui peut par exemple exprimer l'urgence, le conflit d'intérêt ou tout autre information.

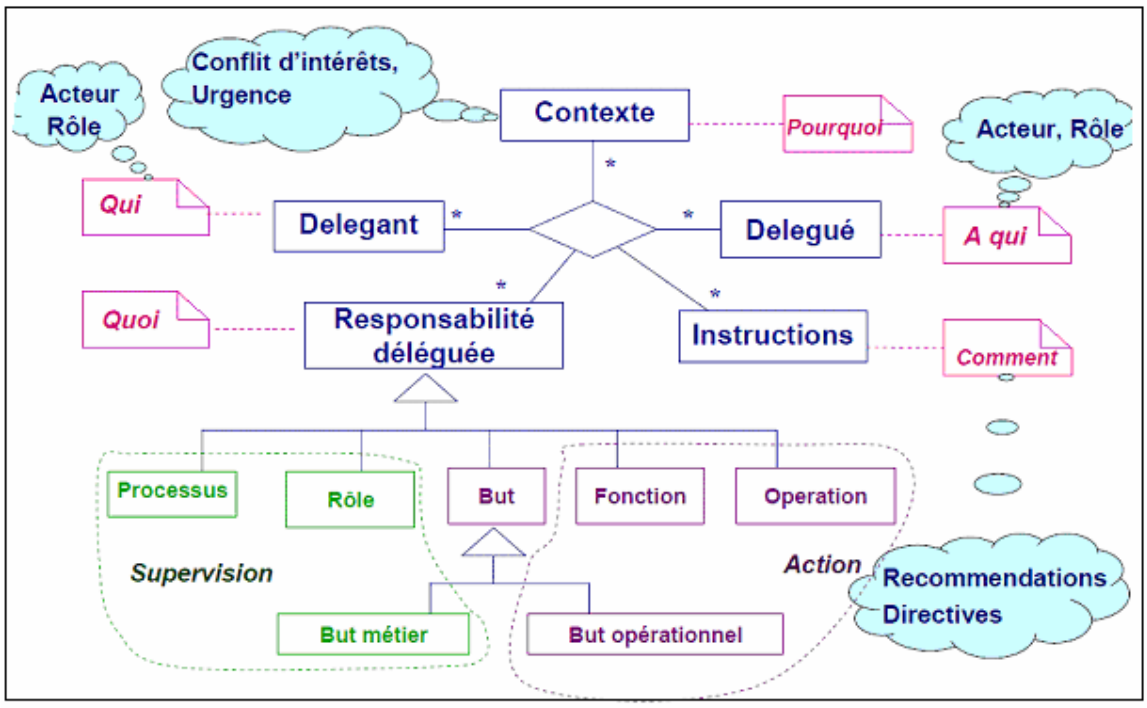

Figure 8. Le méta-modèle de la délégation

Les auteurs proposent aussi une description du contexte qui peut impacter de nombreux éléments du modèle. Le contexte intègre le temps (durée, heure, date, jour de la semaine, fréquence, historique), le lieu, les ressources (disponibilité ou proximité des objets, motivation, compétence, expérience, état des acteurs et enfin l'organisation (nature de la relation, proximité des acteurs).

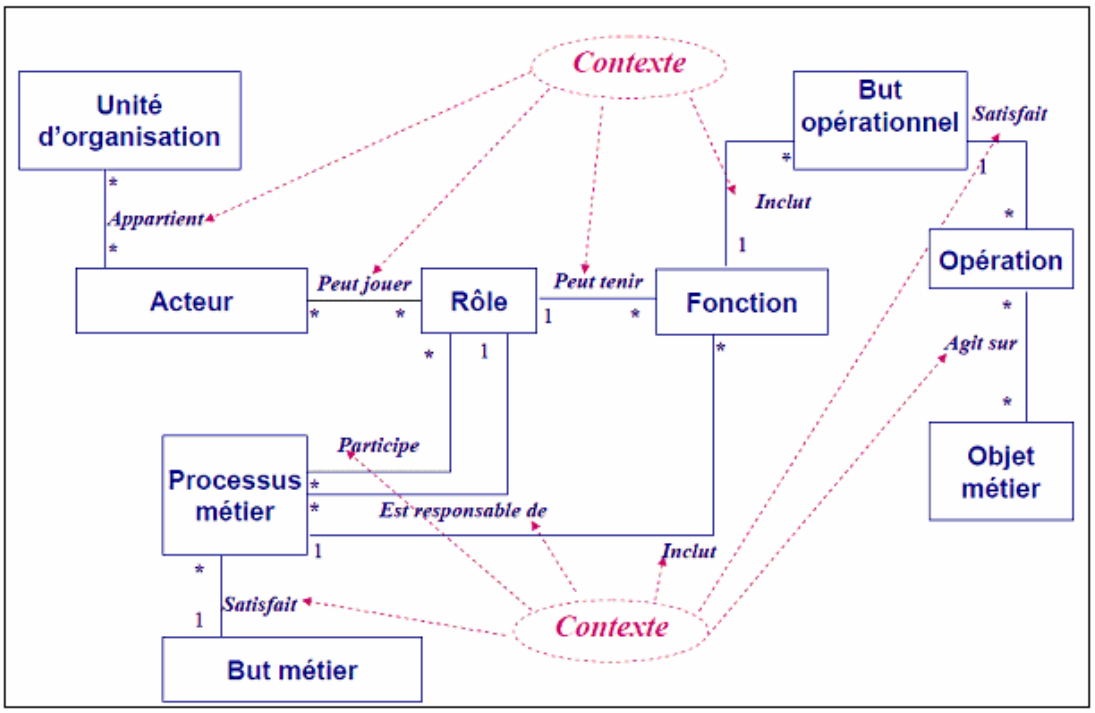

Figure 9. Représentation du contexte du processus

e) Le méta-modèle d'Eder

Le méta-modèle d'Eder et al est exclusivement centré sur les transformations de workflows. Il décrit les aspects statiques des workflows. Un workflow comprend des activités qui sont des workflows ou des activités, élémentaires ou complexes. Les activités complexes sont composées d'autres activités représentées comme des occurrences dans la composition d'une activité complexe. Une activité complexe est décrite au moyen d'une structure de contrôle : séquentielle, parallèle ou conditionnelle.

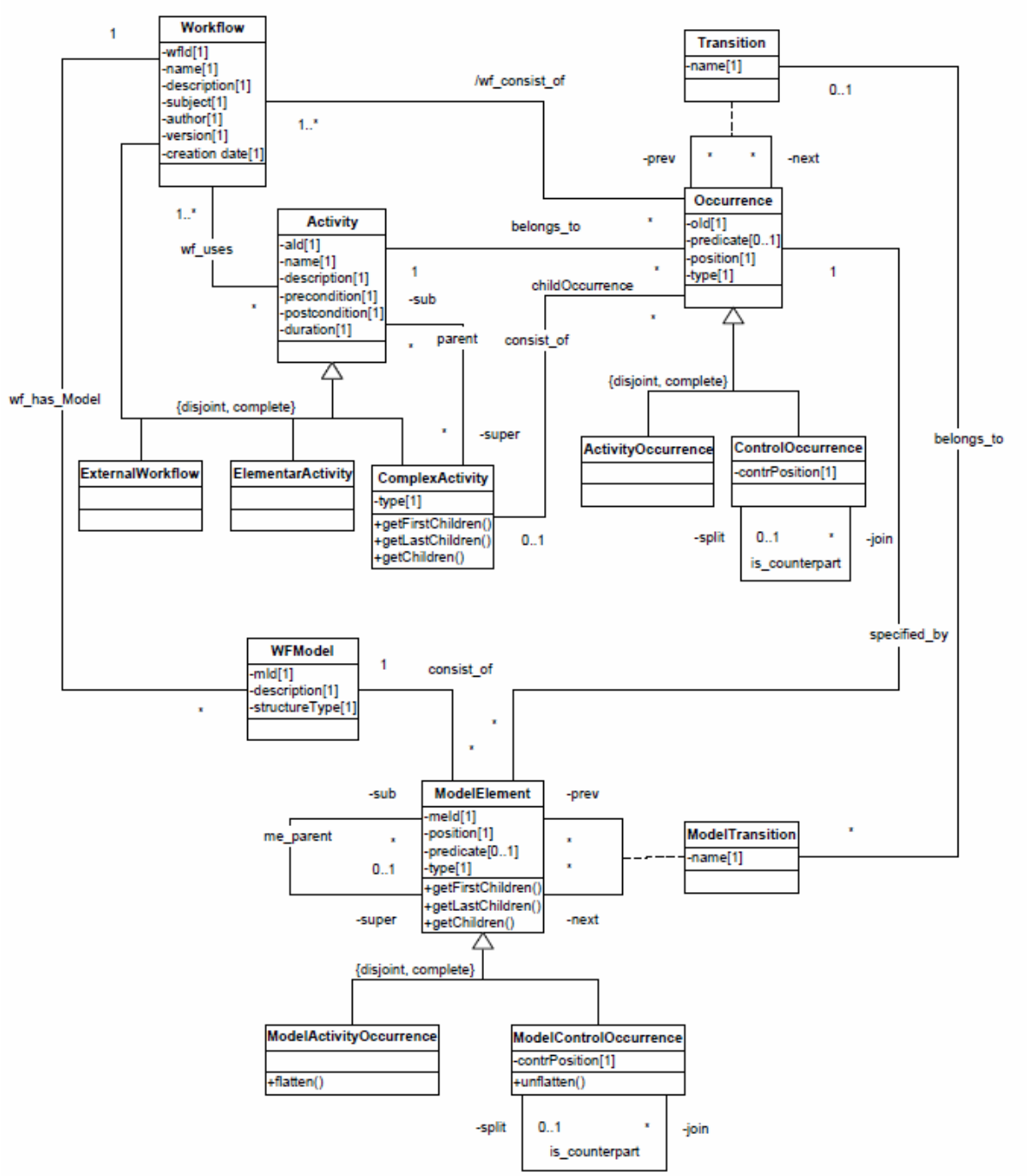

Figure 10. Le méta-modèle d'Eder

f) Le méta-modèle de Ouyang

Le méta-modèle de Ouyang est trop succinct dans sa description du processus [Ouyang et al., 2007]. Toutefois, il a pour avantage de découper le processus en activités qui sont contrôlées par des politiques (policies). Ces politiques définissent la gestion des ressources, des applications et des rôles. Elles se réfèrent à des modèles, notamment des modèles de données et des modèles organisationnels.

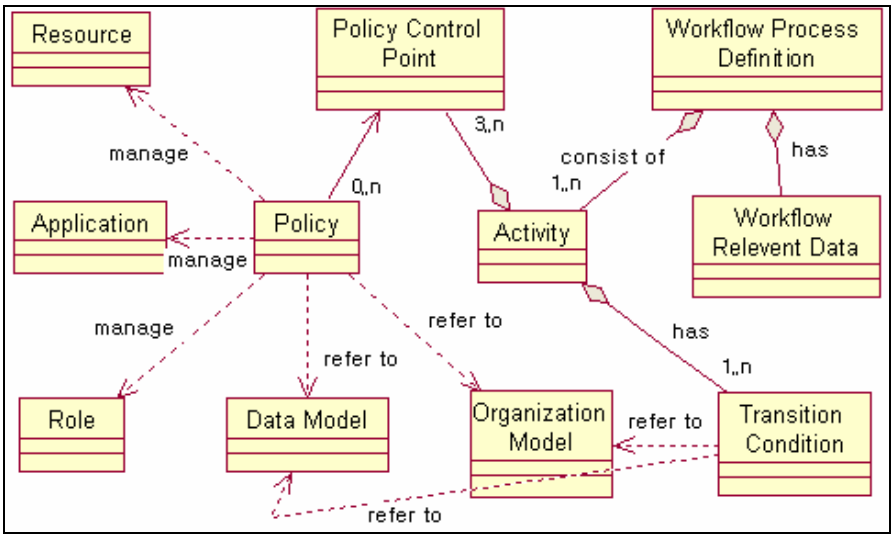

Figure 11. Méta-modèle de Ouyang

g) Le méta-modèle de Brambilla

Brambilla modélise les processus pour automatiser la génération de modèles de domaines [Brambilla et al., 2007]. L'intérêt de ce modèle réside justement dans le lien qu'il établit entre le processus et le domaine fonctionnel. Un exemple d'un tel modèle est donné ci-dessous. Toutefois, la description du processus reste très sommaire.

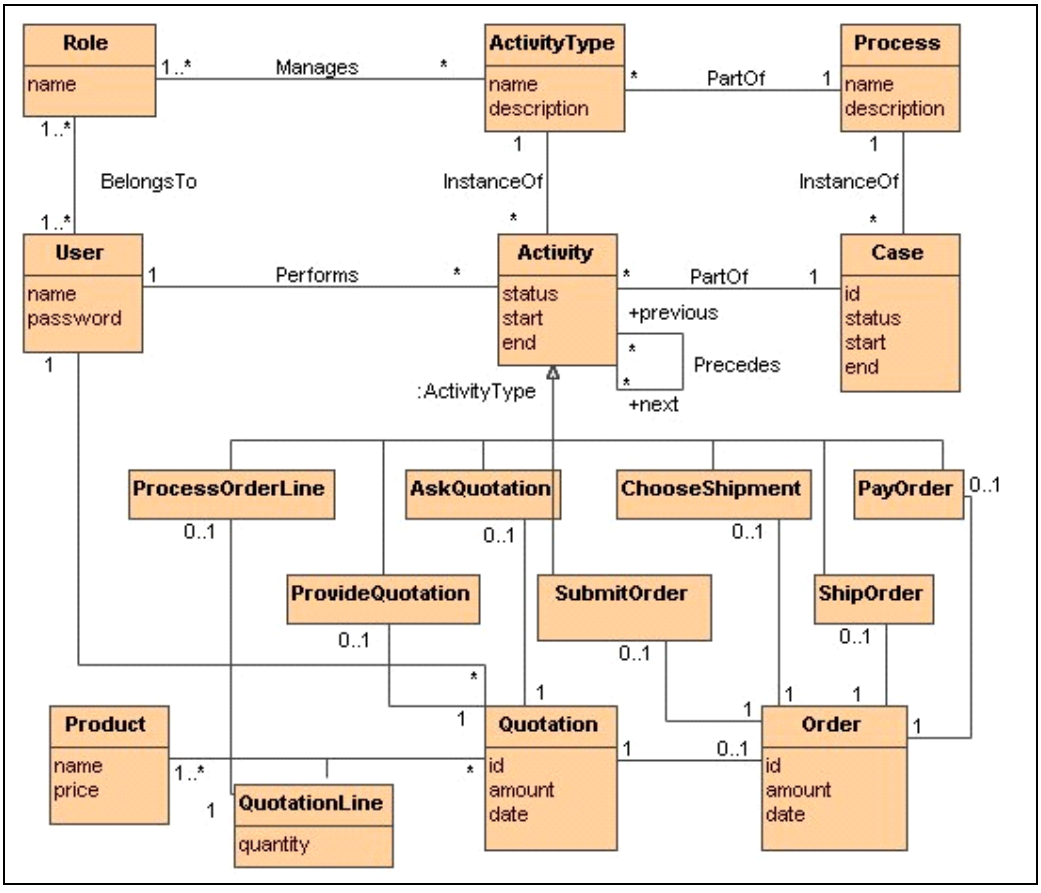

Figure 12. Le méta-modèle de Brambilla

h) Le méta-modèle de Kofod-Petersen

Kofod-Petersen et al modélisent le contexte en reprenant la théorie de l'activité, fondée notamment sur le triangle initial Artefact-Sujet-Objet et le triangle CHAT (Cultural Historical Activity Theory). Le triangle CHAT inclut Règles, Communauté, et Division du travail dans le triangle initial [Kofod et al. 2006].

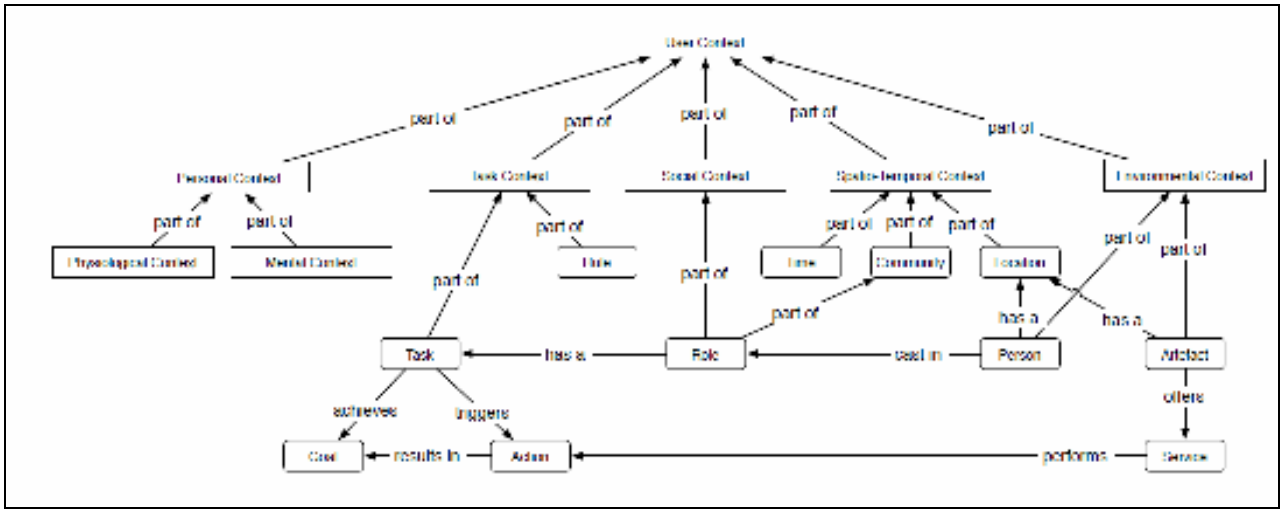

Figure 13. Le méta-modèle de Kofod-Petersen

2.2.1.1 Comparaison des méta-modèles

Le tableau ci-dessous synthétise les différentes approches décrites ci-dessus. Les méta-modèles considérés peuvent être classés selon l'objectif qu'ils visent : modélisation des processus métiers, modélisation des workflows, etc. Nous avons retenu ce sous-ensemble parmi une littérature abondante, en raison d'une ou plusieurs de leurs spécificités qui vont trouver leur application dans notre modélisation de la sécurité.

Approche	Objectif	Spécificité
Morley	modélisation des processus métiers	classification des événements et des activités
Kradolfer	représentation des workflows	le processus est décrit par son workflow selon 4 perspectives
List	évaluation comparative de langages de modélisation de processus	5 perspectives issues de Curtis et enrichies
Saidani	modélisation des processus flexibles	concepts de délégation et de contexte
Eder	modélisation de workflows	centré la description fine des activités du processus
Ouyang	modélisation des workflows	intègre les politiques de contrôle des activités
Brambilla	automatisation de la génération de modèles de domaines	établit un lien entre le processus et le domaine fonctionnel
Kofod-Petersen	modélisation du contexte pour l'informatique ambiante	utilise la théorie de l'activité

Figure 14. Comparaison des méta-modèles de processus

Le tableau ci-dessous (Figure 15) recense les concepts principaux utilisés dans les différents modèles. Il apparaît aisément que le modèle de List est celui qui couvre le nombre maximal de concepts pertinents.

Concept	Morley	Kradolfer	List	Saidani	Eder	Ouyang	Brambilla	Kofod-Petersen
Processus/workflow	x	x	x	x	x	x	x	
Activité	x	x	x		x	x	x	
Rôle	x		x	x		x	x	x
Evénement	x		x					
Objectif/but	x		x	x				x
Tâche/opération	x		x	x				x
Transition	x				x	x		
Ressource	x		x			x		
Flux de données		x	x					
Interface		x						
Entité organisationnelle		x		x				
Participant/acteur/utilisateur			x	x			x	x
Flux de contrôle			x					
Logiciel			x					
Contexte				x				x
Politique/règle							x	x

Figure 15. Les concepts présents dans les modèles

2.2.2 Les méta-modèles des besoins

Dans la littérature, les besoins sont déduits des buts ou objectifs à travers les opérations d'élucidation, d'élaboration, de structuration, de spécification, d'analyse et de documentation. Ainsi, on trouve plusieurs définitions du but. [Pohl, 1997] définit un but comme étant « *un objectif qu'un acteur veut atteindre quand il demande un certain service* ». Pour [Lamsweerde, 1998], un but est « *un objectif que le système composite doit satisfaire* » alors que [Rolland, 1998] définit un but comme étant « *une chose que certaines parties prenantes souhaitent atteindre dans le futur* ».

Dans la littérature, nous avons pu relever principalement quatre classifications de buts :

- les buts *fonctionnels* (ce que prévoit le système) et *non fonctionnels* (besoins en termes de qualité que devrait fournir le système),
- les buts « *hard* » (buts mesurables et vérifiables à travers des techniques formelles) et « *soft* » (buts dont le critère de satisfaction n'est pas clairement défini et dépend généralement de la satisfaction des utilisateurs),
- les buts *comportementaux* (décrivent le comportement du futur système) et *de qualité de développement* (décrivent le processus permettant de produire et faire évoluer le système),
- les buts *organisationnels* (liés au rôle des acteurs dans l'organisation) et « *crowd* » (dépendant d'un acteur ou d'un groupe d'acteurs).

Certains méta-modèles décrivent uniquement les besoins de sécurité (méta-modèle SRR de [Sindre et al, 2003], taxonomie de [Mazhar, 2007], etc.) alors que d'autres intègrent les besoins fonctionnels et non fonctionnels dont la sécurité (ceux de i*, Tropos, Secure Tropos,

Si*, O-RGPS, KAOS, etc.). Dans cet état de l'art, nous nous limitons à la description des méta-modèles les plus pertinents.

a) Méta-modèle SRR

Ce méta-modèle est un entrepôt de ressources de sécurité réutilisables dans lequel sont décrits les objectifs de sécurité ainsi que les besoins qui en découlent. Ces besoins sont regroupés en clusters et associés à des spécifications sous forme textuelle et/ou en UMLSec. Ce méta-modèle identifie aussi les menaces, les besoins de sécurité permettant de les atténuer et leur associe une spécification textuelle et/ou au moyen des anti-cas (misuse case). Il met aussi en évidence les artefacts à sécuriser.

b) Méta-modèle i*

Dans [Yu, 95] l'auteur propose l'approche i*, une approche intentionnelle et orientée agent d'ingénierie des besoins. i* est fondée sur un modèle de dépendance stratégique et sur un modèle rationnel stratégique. Le modèle rationnel stratégique représente le raisonnement de chaque acteur à propos de ses relations intentionnelles et permet d'identifier les besoins du système. Un méta-modèle pour la représentation des modèles rationnels stratégiques simplifiés de i* a été présenté dans [Yu, 2006]. Ce méta-modèle présente les concepts de *buts, buts soft, tâches, ressources et étiquettes d'évaluation.* Dans ce méta-modèle, un *but* (*soft* ou non) se décompose en *tâches* et peut contribuer à la réalisation d'un autre but. Les tâches sont exécutées et réparties sur les ressources mais peuvent également contribuer à d'autres tâches en cas de processus complexes. Une évaluation des *buts, tâches et ressources* est faite à travers des *étiquettes d'évaluation* afin de s'assurer de la satisfaction des buts. Ce méta-modèle permet donc de déterminer si un but est satisfait ou non. L'approche i* intègre l'origine des buts et des besoins du système mais ne précise pas les notions de sécurité bien que certains concepts (le degré de dépendance par exemple) puissent être intéressants pour la sécurité. Différentes approches ultérieures se sont appuyées sur les concepts de cette approche en fournissant des extensions dont certains aspects de sécurité.

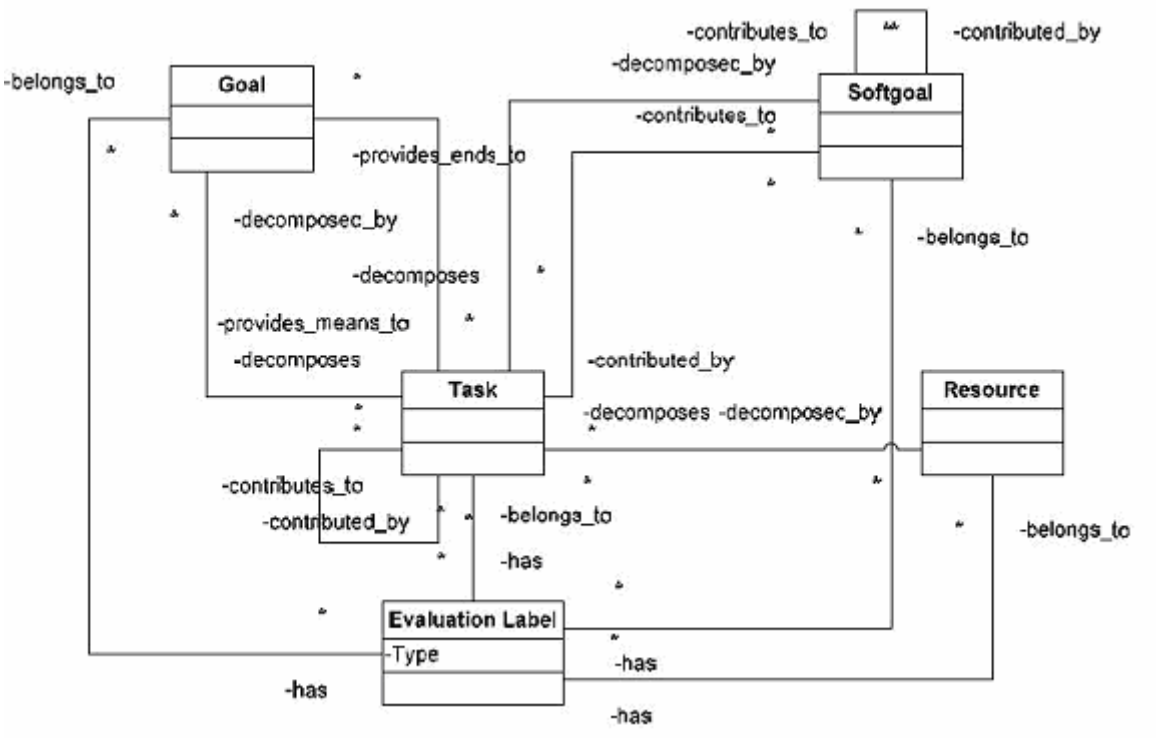

Figure 16. Le méta-modèle i*

Quelques chercheurs se sont intéressés à la modélisation et l'analyse des exigences pendant la première phase du développement de logiciels, phase qui précède la spécification des exigences du système à construire. C'est le cas notamment de [Bresciani et al, 2004] qui a proposé le méta-modèle Tropos, fondé sur i* et intégrant les concepts d'acteur, de but, de plan, de ressource et de dépendance sociale.

c) Méta-modèle de Tropos et SecureTropos

Tropos est une méthodologie orientée agent qui couvre le cycle de vie du développement du logiciel. Elle utilise les agents de type BDI (Belief - Desire - Intention) et les états mentaux dans la prise de décision propre par les agents. Elle accorde une importance à l'ingénierie des premières exigences qui sont élaborées par les acteurs de l'application ainsi qu'à leurs intentions.

Dans [Susi, 2005], les auteurs présentent le *diagramme de buts* qui est un graphe dont les nœuds sont les buts ou plans (tâches en i*) alors que les liens entre buts/plans indiquent la contribution positive (+) ou négative (-) à la réalisation d'un but. Dans ce méta-modèle, un but est réalisé par la contribution des *acteurs*, des *plans* et des *ressources*. Cette contribution peut être positive par la participation à la réalisation du but ou négative lorsqu'il s'agit d'un obstacle à la réalisation du but. Contrairement à i*, les auteurs séparent les buts des tâches surtout au niveau des décompositions. Ici, les buts se décomposent entre eux et les tâches entre elles. Une analyse des moyens de réaliser les buts est proposée à travers *Means-end analysis* pour l'évaluation des buts.

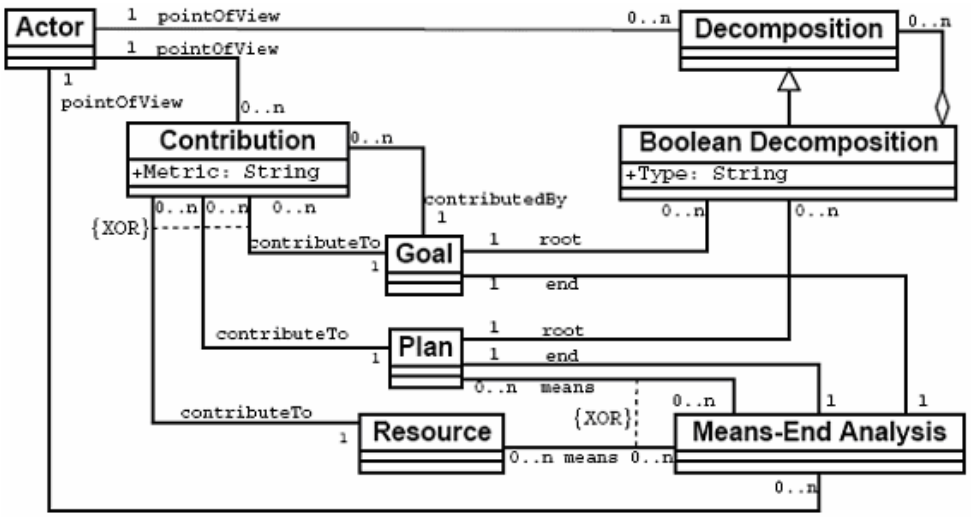

Figure 17. Diagramme de classe UML des concepts de but du méta-modèle Tropos

d) Métamodèle Si*

Le méta-modèle i* ne présentant pas les aspects de sécurité, d'autres chercheurs l'ont étendu en les y incorporant. C'est le cas de [Zannone, 2008] qui définit le méta-modèle Si* et montre les différences avec les méta-modèles d'origine, i* et Tropos.

Si* se décline en cinq diagrammes : *diagramme d'acteurs, diagramme social, diagramme de dépendances d'exécution, diagramme de délégation de permission et diagramme de buts/tâches*. Dans cette partie, nous nous intéressons à ce dernier diagramme. Le méta-modèle pour les *buts/tâches* présente les concepts de *but, tâche, rôle, ressource, moyen d'analyse, décomposition et de contribution.* Contrairement à i*, la *contribution* prend place entre le *rôle* qui représente un acteur, un *but* et une *tâche*. Une *tâche* va consommer ou produire une ressource. Elle peut se décomposer en *sous tâches* alors que la décomposition des buts est de deux types : AND et OR.

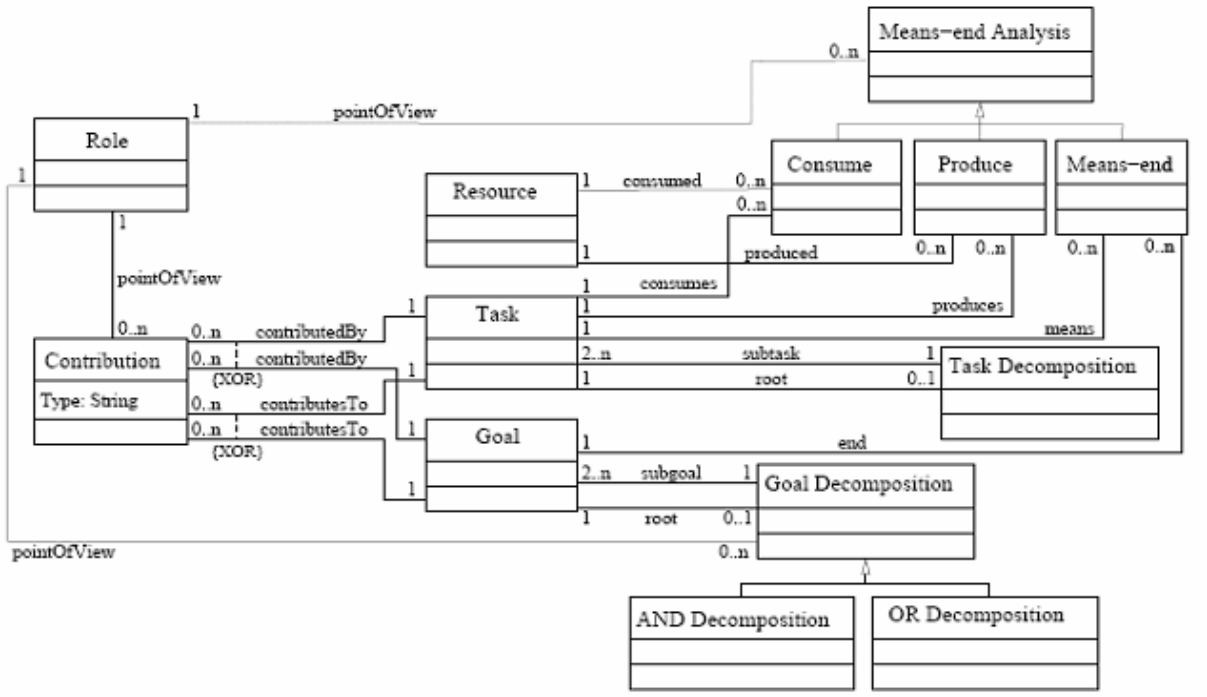

Figure 18. Méta-modèle du diagramme But/Tâche

Toutes ces approches fondées sur i* ne présentent pas un processus d'ingénierie allant de la spécification des besoins jusqu'au développement du produit. Aussi, le mécanisme d'obtention des buts n'est pas explicite et l'influence du contexte organisationnel et du domaine n'est pas mise en évidence.

e) Méta-modèle O-RGPS

Pour pallier cette lacune, [Keqing, 2008] et [Jingbai, 2009] présentent un méta-modèle O-RGPS qui se fonde sur le contexte du domaine pour la modélisation des besoins incluant la sécurité. Il peut être mis en œuvre au moyen d'une approche en cinq couches. Celle-ci est fondée sur une ontologie du domaine qui, elle aussi, se décline en cinq couches : *Ontologie, Rôle, But (Goal), Processus* et *Service.* L'*Ontologie* gère le passage d'une couche à l'autre en définissant une correspondance des concepts entre couches. La couche *Rôle* représente les relations dans l'organisation dont certaines permettent d'identifier les buts. La couche *Goal* représente les buts dont vont découler les besoins opérationnels. La couche *Processus* présente l'opérationnalisation des buts identifiés dans la couche précédente, tandis que la couche *Service* décrit le service obtenu à la fin de la couche *Processus.*

Les besoins sont représentés par le méta-modèle G-Net dans la couche *Goal* au moyen des concepts : *but* (fonctionnel vs. non fonctionnel, crowd goal vs. but organisationnel, but opérationnel), *processus*, *décomposition*, *acteur*. Les auteurs classent les buts selon deux axes : *fonctionnel* et *non fonctionnel ;* (rôle) *organisationnel* et *personnel* (groupe d'acteurs). Les buts possèdent des contraintes d'*exclusion* et de *dépendance* mais également une décomposition de la variabilité (« *mandatory », alternative, optionnel, OR*). Différents liens connectent les concepts : *contribution* (entre but non fonctionnel et processus), *réalisation* (entre but fonctionnel et processus), etc. La décomposition des buts *fonctionnels* et *non fonctionnels* aboutit aux buts opérationnels liés à un processus. Le but *fonctionnel* réalise un processus alors que le but *non fonctionnel* contribue au processus.

Dans [Keqing, 2008], le méta-modèle de *but* représente les buts de *confiance* qui sont chargés d'assurer la gestion des différentes *fautes*. O-RGPS met en évidence l'importance du domaine mais ne présente pas un processus complet d'ingénierie des besoins. Le méta-modèle présente les différentes couches sans donner une démarche pour guider l'utilisateur.

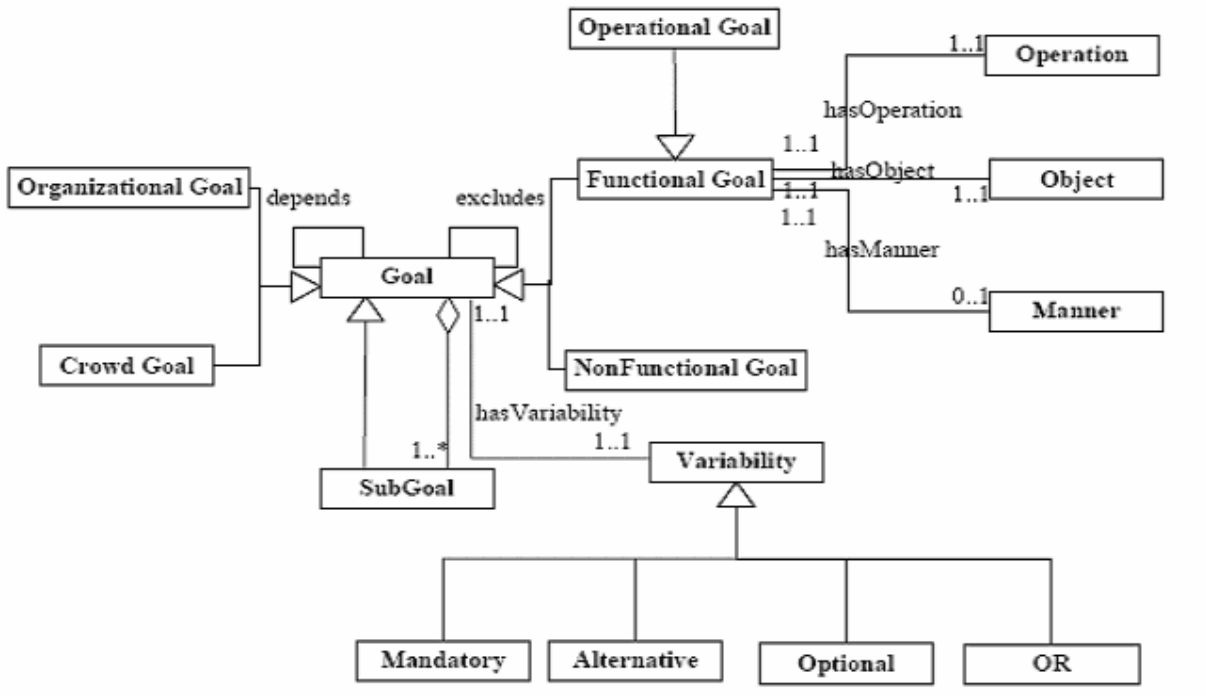

Figure 19. Le méta-modèle de G-net [Wang07]

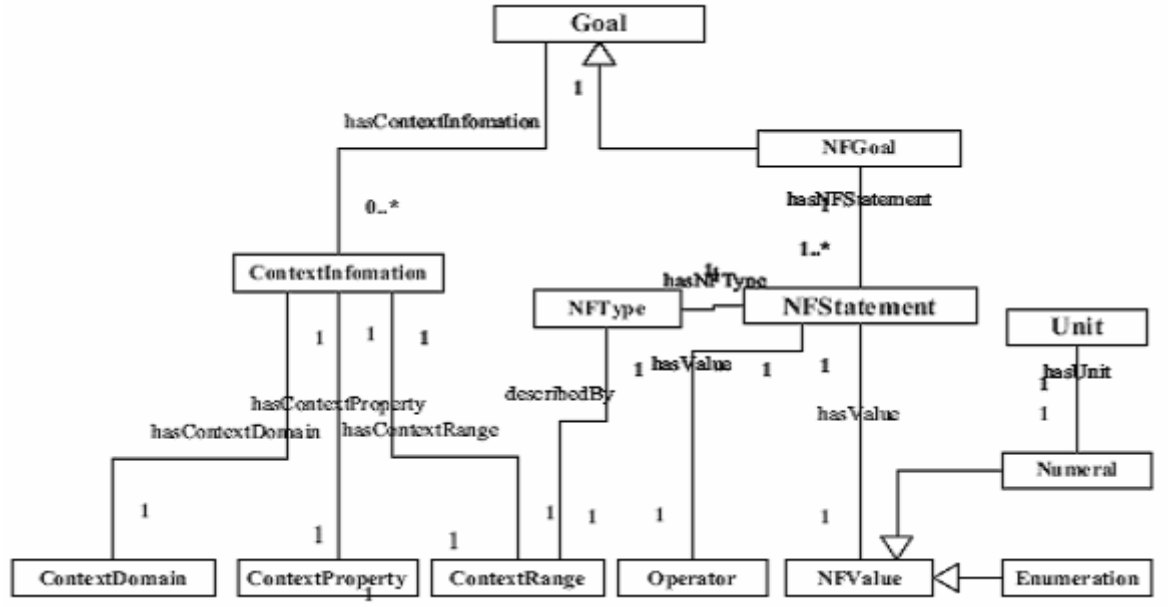

Figure 20. Le méta-modèle des besoins non fonctionnels incluant le contexte

f) Méta-modèle de KAOS

[Lamsweerde 09] propose l'approche KAOS, un processus complet d'ingénierie des besoins allant de la spécification jusqu'au développement et prenant en compte les influences du domaine et de l'environnement. Ce processus va de la spécification des besoins jusqu'à la spécification du produit. Dans [Lamsweerde, 2009], cinq méta-modèles sont présentés : *méta-modèle des buts, méta-modèle d'agent, méta-modèle d'opération, méta-modèle d'objet et méta-modèle de comportement*.

Le *méta-modèle de buts* présente le raffinement des buts en besoins et attentes en tenant compte du contexte du domaine afin d'écarter les obstacles à la réalisation de ces derniers. Un but peut être *soft* (réalisé par la satisfaction d'un ou plusieurs critères) ou *comportemental* (réalisé par un changement d'états à travers les opérations : *achieve, avoid, maintain*). Les attentes sont affectées aux agents tandis que les besoins sont sous la responsabilité des agents logiciels qui assurent la réalisation de ces buts à travers l'*opérationnalisation* des besoins. KAOS présente différents types de raffinement notamment AND, OR, *milestone* (ordonné), etc. Ce méta-modèle des buts KAOS a été étendu dans [Semmak et al, 2008] pour prendre en compte la variabilité dans le processus de développement des applications. Cette variabilité est obtenue par l'introduction des concepts de *variant, variant point* et *facet. Variant* est l'attribut sur lequel il est possible de faire varier le processus de raffinement. *Variant point* est le point sur lequel peut s'appliquer la variabilité en présentant par exemple différentes manières de réaliser un même but. *Facet* est un point de vue. Cette variabilité est surtout liée à la variabilité des raffinements et des attributions (affectation) de la responsabilité des buts aux agents. Elle permet ainsi de représenter explicitement les différentes manières de réaliser un but donné.

Concepts	I*	Tropos	Secure Tropos	SI*	O-RGPS	KAOS	VarKAOS
Contribution	*Contribution*	*Contribution*	*Contribution*	*Contribution*	*'Depends' link*	*Refinement*	*Refinement*
Variabilité	*Degré de dépendance*	-	-	-	*variability*	-	*point de variation (facettes)*
Besoin	*Hard goal*	*Hard goal*	*Hard goal*	*Hard goal*	*Functional goal*	*Requirement*	*Requirement*
Attentes	*Soft goal*	*Soft goal*	*Soft goal*	*Soft goal*	*Non functional goal*	*Expectation*	*Expectation*
Obstacle		*Contribution (-)*	*Contribution(-)*	*Contribution(-)*	*Lien 'exclude'*	*Oui*	*Oui*

Figure 21. Les concepts des méta-modèles de besoins

2.2.3 Les méta-modèles de contexte organisationnel

Pour la prise en charge des contrôles d'accès dans le processus de conception et développement d'applications, des méta-modèles décrivant le contexte organisationnel ont été proposés dans la littérature. Ces méta-modèles peuvent être classifiés selon leur niveau d'abstraction qui est le reflet de l'étape du cycle de vie de l'application. Certains, tels que ORBAC et ORBAC étendu, sont du niveau d'abstraction logique du fait de leur exploitation dans la conception des politiques de sécurité ([Miége, 2005], [Coma, 2008], [Cuppens, 2003]). D'autres, tels que ForBAC [Saïdani, 2006], les méta-modèles de SI* [Zannone, 2008], le méta-modèle R-NET de O-RGPS [Wang, 2007] et ceux des méta-modèles de KAOS [Lansweerde, 2009], sont d'un niveau conceptuel car ils sont utilisés dès la phase de spécification des besoins.

a) Les méta-modèles ORBAC et ORBAC étendu

Dans le méta-modèle ORBAC, la structure organisationnelle, nommée « organisation », est au centre du modèle. Les autres concepts servant à décrire le contexte organisationnel sont : sujet, rôle, organisation, action (lecture, écriture, envoi), activité, contexte, vue, objet. Dans ce méta-modèle sont décrites les habilitations des acteurs (sujet) à jouer des rôles au sein des organisations. Les droits d'accès sont exprimés au moyen des permissions, interdictions et les droits d'usage via le lien d'utilisation.

Le méta-modèle ORBAC a par la suite été étendu pour prendre en compte d'autres aspects organisationnels tels que le concept de hiérarchie des organisations, des rôles, la notion de supervision, le concept d'héritage des permissions et interdictions, de délégation de privilèges, le concept d'activation dynamique des privilèges à travers la définition de contexte, etc. [Coma, 2008], [Cuppens, 2003]

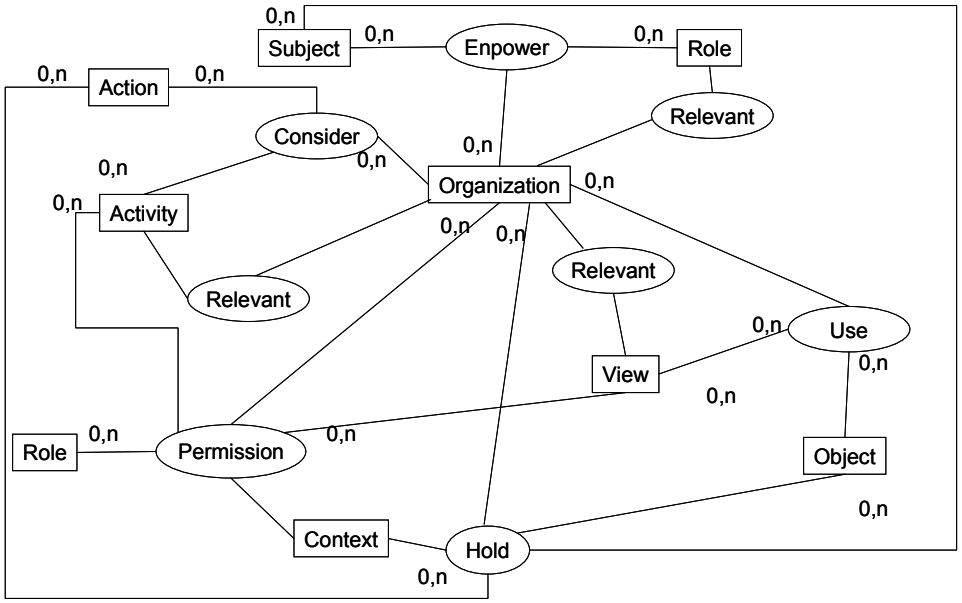

Figure 22. Méta-modèle ORBAC [Miége, 2005]

b) Méta-modèle FORBAC

Le méta-modèle FORBAC (Figure 7), défini dans le cadre de la modélisation de processus métiers flexibles, est globalement moins riche dans la description du contexte organisationnel que le méta-modèle ORBAC étendu. Cependant, il introduit le concept de flexibilité des délégations, qui est, à notre avis, une pratique assez courante dans les organisations et, par conséquent, il peut impacter les politiques d'accès à des applications.

c) Méta-modèles de Si*

Le contexte organisationnel est représenté, dans Si*, à travers trois diagrammes : le diagramme d'acteurs pour modéliser les acteurs externes et internes à une organisation, le diagramme social pour décrire les relations sociales entre les acteurs, enfin le diagramme des permissions-délégations pour les autorisations d'acteurs à acteurs.

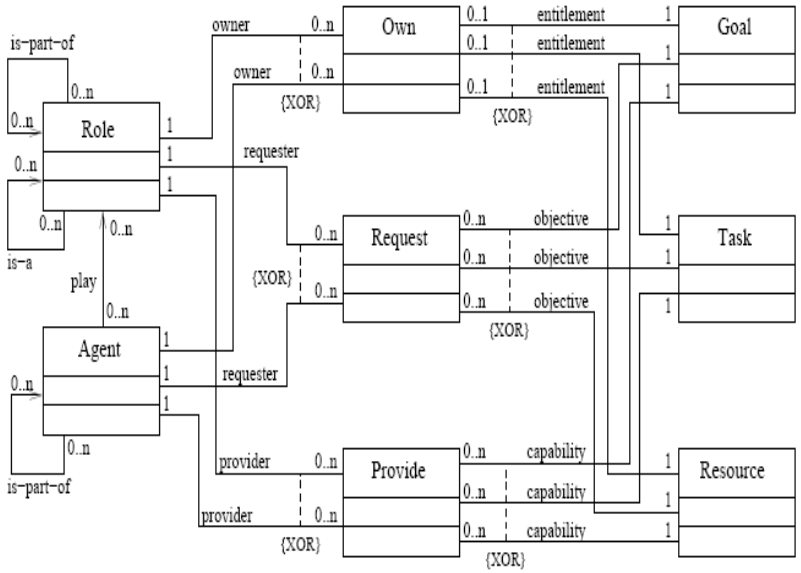

Figure 23. Diagramme d'acteur de Si*

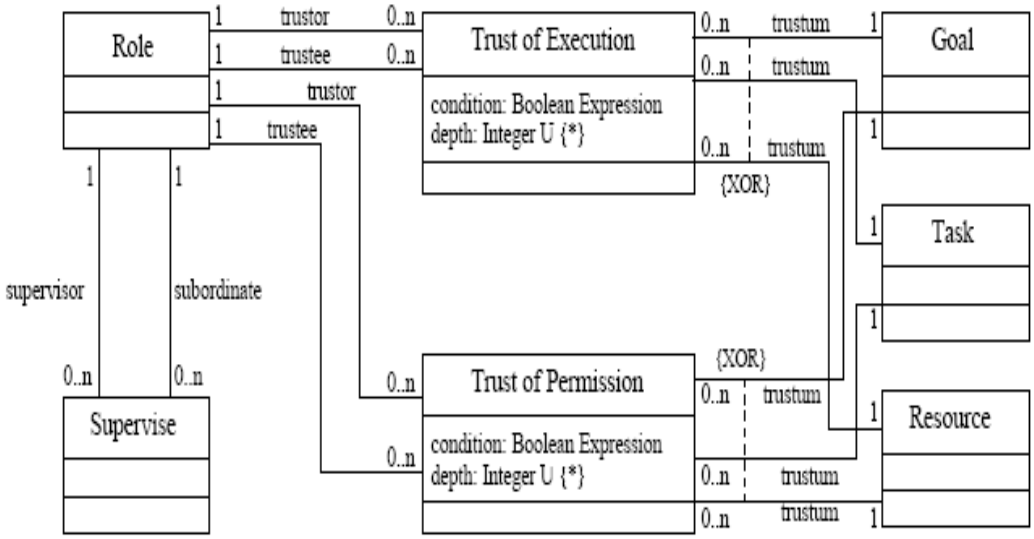

Figure 24. Diagramme social de Si*

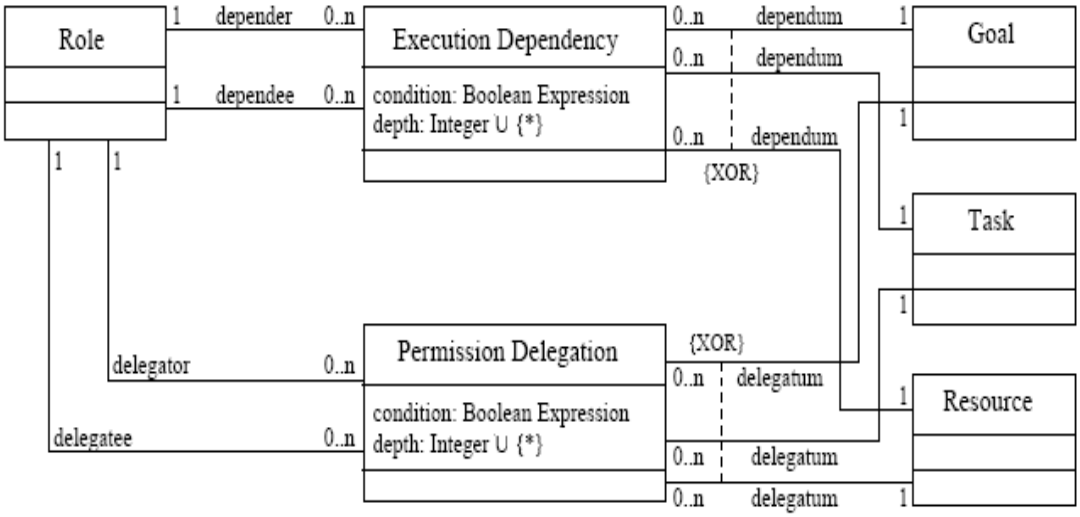

Figure 25. Diagramme de dépendance de Si*

Dans ces méta-modèles, le concept d'organisation est absent. Cependant, en plus du concept d'habilitation, sont représentées de façon explicite les aptitudes des acteurs, la notion de confiance (c'est-à-dire l'attente d'acteur par rapport au comportement d'autres acteurs) ainsi que les possibilités de retransmission de privilèges reçus par des acteurs.

d) Méta-modèle R-Net

C'est l'un des modèles les plus pauvres dans la description du contexte organisationnel. La hiérarchisation des rôles et des organisations est ignorée. Cependant, une distinction entre acteur humain et logiciel est faite.

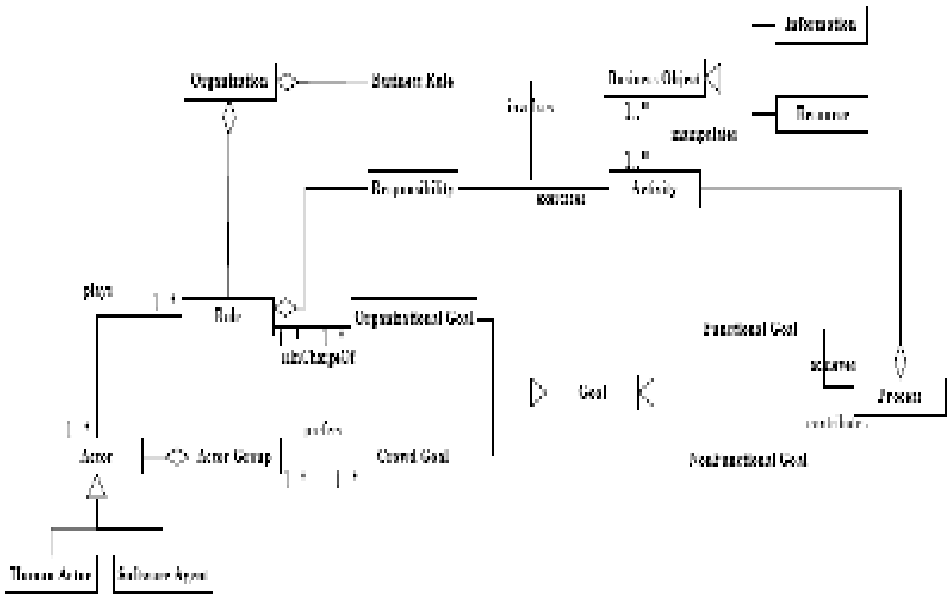

Figure 26. Méta-modèle R-Net

e) Méta-modèles de KAOS

Dans KAOS, la description du contexte organisationnel est répartie dans le méta-modèle des agents et le méta-modèle des opérations. Cette description est, à notre avis, moins riche que les précédentes. Le concept de rôle existant dans les organisations est assimilé au concept d'agent. Le concept d'acteur et d'organisation est absent. Seules les habilitations sont représentées.

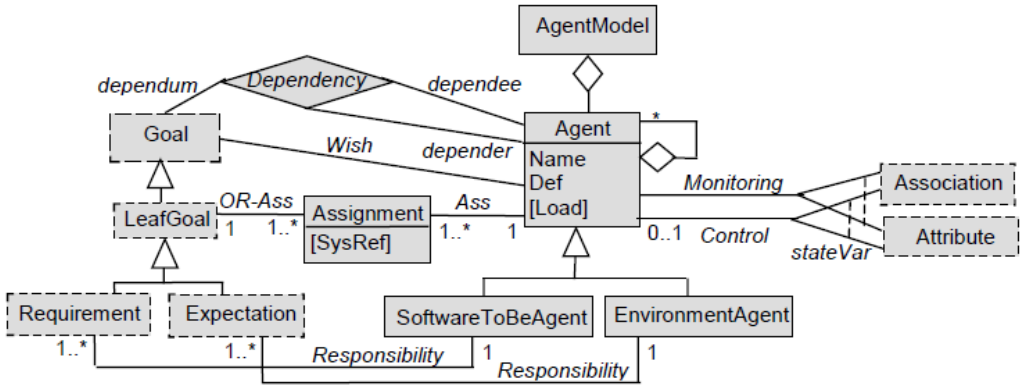

Figure 27. Méta-modèle des agents

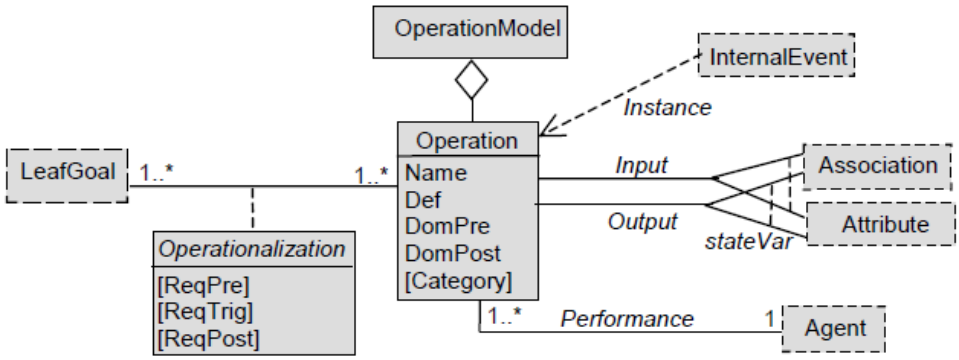

Figure 28. Méta-modèle des opérations

Le tableau ci–après regroupe les concepts utilisés dans les méta-modèles décrits ci-dessus.

Concept	Méta-modèle de KAOS	Méta-modèles de SI*	R-Net	OrBac étendu	Forbac
Organisation	-	-	Organisation	organisation	Unité organisationnelle
Agent / acteur	-	Agent autonome	Agent humain et logiciel	Sujet	acteur
Rôle	Agent	Rôle	Rôle	Rôle	rôle
Tâche	Opération	Tâche	Activité	Tâche informatique	mission
Habilitation	Habilitation	Habilitation	Responsabilité	Habilitation	habilitation
Permission	-	-	-	permission	-

Interdiction	-	-	-	interdiction	-
Délégation	-	Rôle/Rôle Option de transfert de délégation	-	délégation	Délégation flexible
Confiance	-	Confiance	-	-	-
Aptitude	-	Aptitude	-	-	-
Liens entre rôles	Partie de	Is-A / Partie de/Supervision	-	Is-A / Partie de/Supervision	-
Hiérarchie des organisations	-	-	-	Is-A	-
Contexte	-	-	-	Contexte	-

Tableau 5. Comparaison des méta-modèles organisationnels

2.3 Les profils UML pour la sécurité

Ce paragraphe est dédié à l'état de l'art sur les profils UML pour la sécurité. Rappelons à cet effet qu'un profil permet d'adapter le langage de modélisation UML à un domaine particulier. Il est constitué d'un ensemble de concepts (stéréotypes, TaggedValues, Contraintes) permettant d'établir une correspondance entre les concepts standards d'UML (packages, classes, associations, etc) et ceux propres au domaine concerné par le profil. Pour la construction de profils la littérature propose deux méta-modèles : celui de l'OMG [OMG, 2009] ainsi que celui de la société SoftTeam [SOF]. Une description détaillée de ces deux méta-modèles est fournie en annexe 2 de ce document. L'étude de ces méta-modèles a permis de dresser le tableau de comparaison ci-après. Ces deux méta-modèles ont été comparés selon trois points de vue : le point de vue présentation à travers les deux critères de lisibilité et d'expressivité, le point de vue usage en industrie et enfin le point de vue technique en termes de spécification de la sécurité et d'opérationnalisation des profils à travers la spécification de la génération de code, la validation de modèles et la construction de diagrammes. Il a été constaté que du point de vue de l'usage dans l'industrie logicielle, le méta-modèle de profil de l'OMG est le plus standard. Celui de SofTeam est utilisé essentiellement dans les produits propres à la société SofTeam à savoir l'outil Objecteering et MDAModeler. Du point de vue présentation, le méta-modèle de profil de l'OMG est plus lisible puisqu'il contient moins de concepts techniques que celui de SofTeam, qui lui, présente une meilleure expressivité dans la mesure où il contient tous les concepts nécessaires à la description d'un profil et surtout la partie de description opérationnelle permettant la transformation de modèles, etc. Du point de vue technique, le méta-modèle de profil de SofTeam présente une partie de spécification de sécurité au moyen de trois mécanismes d'extension (stéréotypes, TaggedValues et Contraintes). Celui de l'OMG se contente des stéréotypes. De plus, le méta-modèle de profil de SofTeam, contrairement à celui de l'OMG, suggère une partie de description opérationnelle matérialisée par un ensemble de règles offrant la possibilité de valider, transformer et générer automatiquement du code source dans la plateforme retenue.

Critères de comparaison	Méta-modèle de SofTeam	Méta-modèle de l'OMG
Lisibilité	Moins Lisible	Lisible
Expressivité	Meilleure	Moindre
Usage en Industrie	Dans les produits propres à SofTeam	Officiel
Spécification de la sécurité	Stéréotypes Tagged Values Constraints	Stéréotypes
Validation de modèles	Oui	Non
Construction de diagrammes	Oui mais Partielle	Non
Spécification de la génération de codes	Oui	Non

Tableau 6. Comparaison des profils OMG et SofTeam

Dans le cadre de la sécurisation des systèmes d'information, un certain nombre de profils ont été définis dans la littérature. Nous nous intéressons dans cet état de l'art aux profils exploités au niveau de l'analyse et de la spécification d'une application. Les profils pour la modélisation de la qualité de service tel que celui de l'OMG [OMG, 2008] ainsi que ceux associés à la conception ou l'implémentation de la sécurité sont hors champ d'étude de ce document. Dans cette catégorie de profils on retrouve les profils de cas ainsi que les profils pour la description statique des classes. Nous allons nous limiter aux profils suivants : ceux de secureUML et de UMLsec, les anti-cas ou «Misuse Cases », les cas d'abus ou « Abuse Cases » ainsi que les « Security Use Case ».

a) Le méta-modèle de SecureUML

Le méta-modèle SecureUML [Lodderstedt et al., 2002] est défini comme étant une extension du méta-modèle UML . Les concepts de la politique de contrôle d'accès RBAC (la description du modèle RBAC est fournie en annexe 1 de ce document) sont directement représentés dans ce méta-modèle. Il a été conçu initialement dans le cadre d'une recherche dont l'objet est de proposer une approche garantissant la sécurité dans le e–commerce. SecureUML se concentre initialement sur le contrôle d'accès.

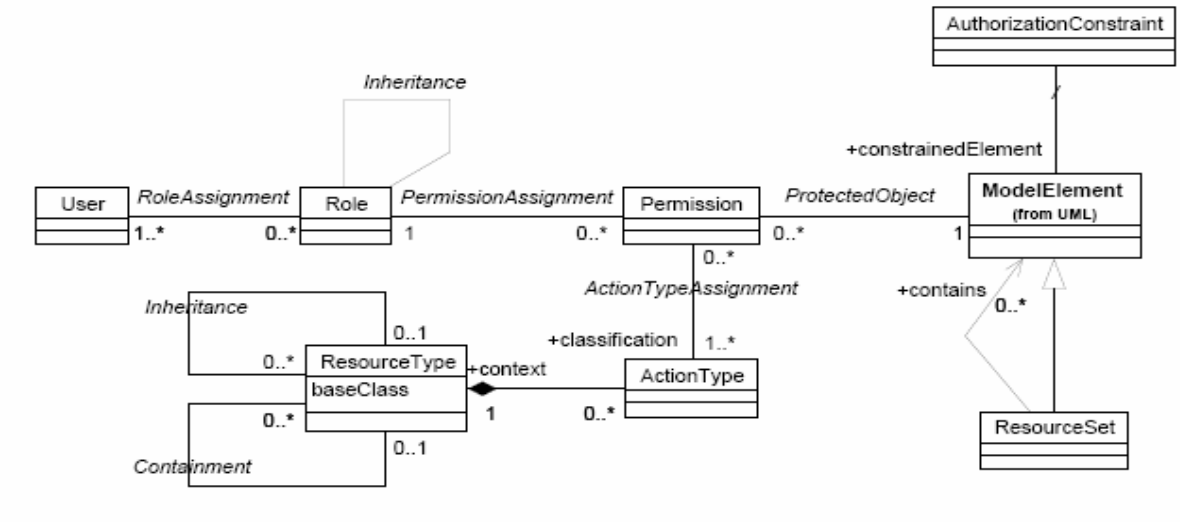

Figure 29. Méta – Modèle de sécurité SecureUML

Le méta–modèle SecureUML reprend intégralement les concepts de la politique de sécurité RBAC notamment les méta classes User, Role et Permission. A noter que la permission porte sur un ensemble d'actions (ActionType). Chaque ActionType représente des opérations pertinentes définies sur un type particulier de ressources. Les actions lire, écrire et exécuter sont des exemples d'ActionType.

b) Les méta-modèles de UMLSec

UMLSec est aussi une extension UML qui autorise l'expression d'informations de sécurité au moyen de diagrammes UML dès la phase de spécification des besoins d'une application.

Le profil UMLSec [Jurjens et al., 2002] est établi au moyen des trois mécanismes d'extension d'UML qui sont les stéréotypes, les valeurs marquées ou TaggedValues (rattachées aux stéréotypes) et les contraintes qui servent à raffiner la sémantique des éléments stéréotypés. Ces mécanismes ont été ajoutés aux diagrammes UML pendant la phase de conception dans le but de garantir certains besoins basiques de sécurité et plus particulièrement :

- La confidentialité et l'intégrité,
- La sécurité du flux d'information,
- Le contrôle d'accès,
- L'auditabilité, la traçabilité en conséquence et
- La sécurité du protocole d'analyse.

UMLSec facilite l'évaluation des spécifications UML, l'encapsulation de l'ingénierie de sécurité dans des « patterns » et représente un moyen simple de conception pour les développeurs non initiés à la conception de systèmes sécurisés dès les premières phases. L'extension UMLSec a pour objectif principal de développer des systèmes d'information sécurisés. Elle garantit particulièrement certains buts bien précis :

- Etant donné un modèle décrit en UML, il est possible de l'évaluer automatiquement pour détecter ses vulnérabilités.

- UMLSec permet à ses utilisateurs, souvent non spécialisés en sécurité, de définir aisément et de manière non ambiguë les caractéristiques et les besoins du système en termes de sécurité.

- Les utilisateurs peuvent appliquer certaines règles d'ingénierie de sécurité préétablies.

- Les besoins de sécurité essentiels et récurrents comme la confidentialité et l'intégrité sont intégrés dans le modèle.

- UMLSec définit une sémantique formelle et précise pour exprimer la sécurité à l'aide des construits UML en utilisant les machines à états UML (AbstractStateMachines) permettant de modéliser globalement le comportement des objets. Elles autorisent la construction de descriptions formelles et simples de modèles UML (diagrammes de cas d'utilisation, diagrammes de classes, diagrammes de séquences, etc) en incluant des renseignements sur tous les diagrammes et à tous les niveaux d'abstraction.

UMLSec présente un avantage majeur ; il fournit au développeur des éléments de notation représentant les exigences de sécurité dans le modèle, avec leurs définitions formelles. UMLSec introduit certains profils au moyen de mécanismes d'extension.

c) Les autres méta-modèles (« Misuse Cases », « Abuse Cases » et « Security Use Case »)

Pour spécifier des scénarios d'attaques qui peuvent violer la sécurité des systèmes, plusieurs travaux proposent d'étendre UML. Les modèles de *Misuse Cases* [Sindre *et al.,* 2000], *Abuse Cases* [McDermott *et al.,* 1999] et *Security Use Case* [Firesmith, 2003] permettent d'intégrer les menaces de sécurité dans le formalisme UML de cas d'utilisation. Les cas d'utilisation spécifient une séquence d'actions que l'entité réalise, en interagissant avec les acteurs de l'entité.

Dans ces modèles une menace peut être initiée par un utilisateur du système autorisé ou non. L'ensemble des cas d'utilisation exprime la vulnérabilité du système, les déroulements des opérations qu'un système doit exécuter dans le cas de chaque attaque. Ces cas d'utilisation incluent également les impacts possibles si l'attaquant réussit.

Le tableau 7 récapitule notre étude de cet état de l'art. La comparaison des modèles de sécurité permet de distinguer : les profils de use cases des profils de classes. Les concepts, tels que le mapping avec les éléments UML, usage, acteurs externes et buts font apparaître la similitude des approches UMLsec et SecureUML pour ces aspects et la complémentarité des modèles "Misuse Cases", "Abuse Cases" et "Security Use Case" pour les concepts traités.

Concept	« Misuse Cases"	« Abuse Cases"	"Security Use Cases"	UMLsec	SecureUML
Mapping avec l'élément UML	Use Cases	Use Cases	Use Cases	Use cases, Classes, Etats transition	Classes, Etats transition
Usage	Analyse et spécification « Threats »	Analyse et spécification des besoins : point de vue des attaquants	Analyse et spécification des besoins de sécurité	Analyse et spécification des besoins de sécurité	Analyse et spécification des besoins de sécurité
Acteurs externes	Misuser, utilisateur	Plusieurs utilisateurs simultanément comprenant les attaquants internes ou	utilisateurs	acteurs	acteurs
Buts	Threats objectifs	Business objectif accompli par les acteurs et un workflow particulier	Business objectif	Business objectif	Business objectif

Tableau 7. Comparaison des profils de sécurité

3. Description générale de notre approche

3.1 Introduction

Dans le cadre du projet SELKIS, nous proposons une démarche de **Mo**délisation des **S**ystèmes **d'I**nformation **S**écurisé,appelée MoSIS, où les propriétés DICT sont prises en charge depuis l'étape de spécification des besoins jusqu'à l'implémentation. Pour garantir une indépendance entre la logique métier dont est porteur le système d'information et les plateformes technologiques sur lesquelles il reposera, nous avons opté pour une approche dirigée par les modèles fondée sur une architecture MDA (*Model Driven Architecture)*. Un tel choix nous permet aussi une meilleure prise en charge des changements dans les systèmes d'information. En effet, avec une documentation à jour des différentes étapes de construction du système d'information, le processus de rétro-ingénierie est facilitée toute les fois où il s'impose suite à des changements majeurs dans le système d'information. Enfin, nous avons choisi UML comme langage de modélisation.

A titre de rappel, MDA prévoit trois niveaux d'abstraction du système d'information : CIM pour «Computation Independent Model», PIM pour « Plateform Platform Independent

Model », PSM pour «Platform Specific Model » et CODE [Bézivin, 2002]. A chacun de ces niveaux, on peut associer des modèles. Le niveau CIM fournit une vision métier du système d'information. Le niveau PIM est une description informatique du système d'information, indépendante de toute plateforme technologique. Le niveau PSM est une variante de description informatique du système selon une plateforme donnée. Le passage d'un modèle de niveau supérieur vers un modèle de niveau inférieur est réalisé à l'aide de transformations. De plus, il est possible d'itérer dans un même niveau grâce à des procédés de raffinement (voir figure 30).

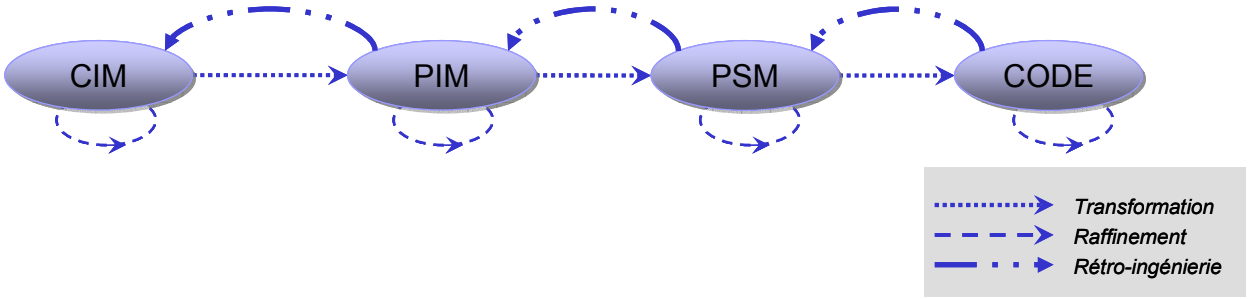

Figure 30. Approche MDA

A l'aide de MoSIS, nous construisons un système d'information sécurisé selon quatre étapes. Chacune d'elles correspond, tel que le montre la figure 31, à un niveau d'abstraction MDA. La première étape comprend la description du système d'information et de son environnement. C'est dans cette étape que s'effectue l'analyse de l'environnement organisationnel où le futur système va opérer. A titre d'exemple, il s'agira dans cette étape de décrire ce que les acteurs doivent faire. Ceci nous permettra, dans les étapes suivantes, de déduire ce qu'ils ont le droit de faire. Cette analyse contient, d'une certaine manière, la justification (« pourquoi ») du choix des mécanismes de sécurité préconisés et fournit la réponse au «comment» et «quand» les utiliser. La seconde étape correspond à l'analyse et la conception abstraite du système d'information. C'est dans cette étape que les besoins fonctionnels et de sécurité sont intégrés et modélisés à l'aide d'un diagramme des cas d'utilisation. La troisième phase est la conception détaillée dans laquelle la plateforme technologique sur laquelle reposera le système d'information a été prise en compte. La dernière phase correspond au codage.

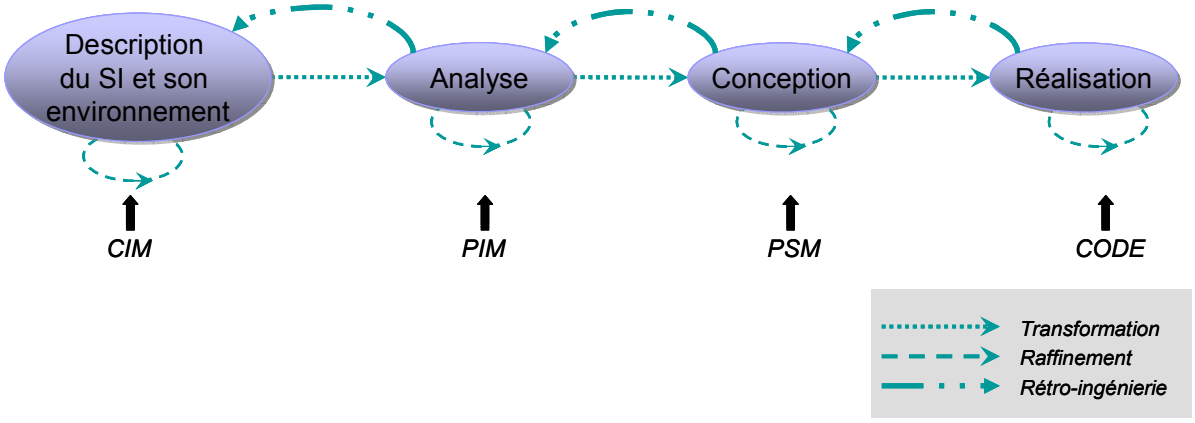

Figure 31. Les étapes de MoSIS

Pour faciliter l'exécution des différentes transformations et/ou raffinements, MoSIS s'appuie sur une base de connaissances qui contient une ontologie des buts et besoins de sécurité génériques ainsi que les différents types de spécification associés, la documentation des applications existantes, les différents types de règles de transformation, ou tout autre source d'information servant le processus.

Dans les paragraphes qui suivent, nous présentons, dans un premier temps le méta-modèle de niveau CIM de MoSIS. Dans un second temps, nous présentons deux méta-modèles de niveaux PIM, décrivant respectivement les fonctionnalités du système d'information sécurisé et sa structure statique. Ces méta-modèles sont respectivement des extensions du diagramme des cas et du diagramme des classes de UML. Ces extensions ont été opérées via le concept de profil. Enfin, nous fournirons quelques règles de passage du CIM vers le PIM.

Une instanciation des différents méta-modèles est proposée sur la base de l'exemple décrit en annexe 4 de ce rapport.

3.2. Le méta-modèle CIM de notre approche

Un système d'information est l'articulation de trois sous-systèmes indissociables : (a) un système social représenté par les acteurs concernés, (b) un système organisationnel constitué de règles, méthodes et procédures et enfin (c) un système technique qui est un dispositif pouvant se matérialiser par un réseau ou un outil informatique [Guyot, 2006]. Mis en place au sein d'une organisation, il permet de couvrir toutes les tâches informationnelles nécessaires à l'exécution des activités opérationnelles, de management et de prises de décision. En d'autres termes, il doit répondre à des besoins fonctionnels et non fonctionnels issus des objectifs qui lui ont été assignés. Parmi ces besoins fonctionnels, on retrouve ceux liés à la sécurité et plus particulièrement aux propriétés DICT.

Une représentation du système d'information, au niveau CIM, consiste donc à le décrire du point de vue social, organisationnel et du point de vues besoins qu'il est censé satisfaire. Pour chacun de ces points de vue, nous proposons respectivement une description sous forme de méta-modèles : un méta-modèle organisationnel, un méta-modèle de processus et un méta-modèle des besoins. La fusion de ces trois méta-modèles constitue le méta-modèle de niveau CIM associé à notre système d'information.

Dans ce qui suit, nous décrivons chacun de ces méta-modèles. Dans chaque méta-modèle nous distinguons les concepts d'ancrage permettant de le relier aux autres méta-modèles. Nous fournissons aussi des exemples d'instanciation sur la base du cas bibliothèque décrit en annexe 4. L'instanciation complète est fournie en annexe 5.

3.2.1 Le méta-modèle de processus

Pour pouvoir représenter l'ensemble des propriétés DICT, une modélisation des processus métier supportés par le système d'information s'avère nécessaire. Cette modélisation prend en compte la structure des processus, leur comportement, les informations qu'ils manipulent.

Le résultat de notre étude de l'état de l'art nous a permis de dresser une première version du méta-modèle (Figure 32). En bleu, figurent les concepts d'ancrage avec le méta-

modèle des besoins. En orange, sont représentés les classes d'ancrage avec le méta-modèle organisationnel.

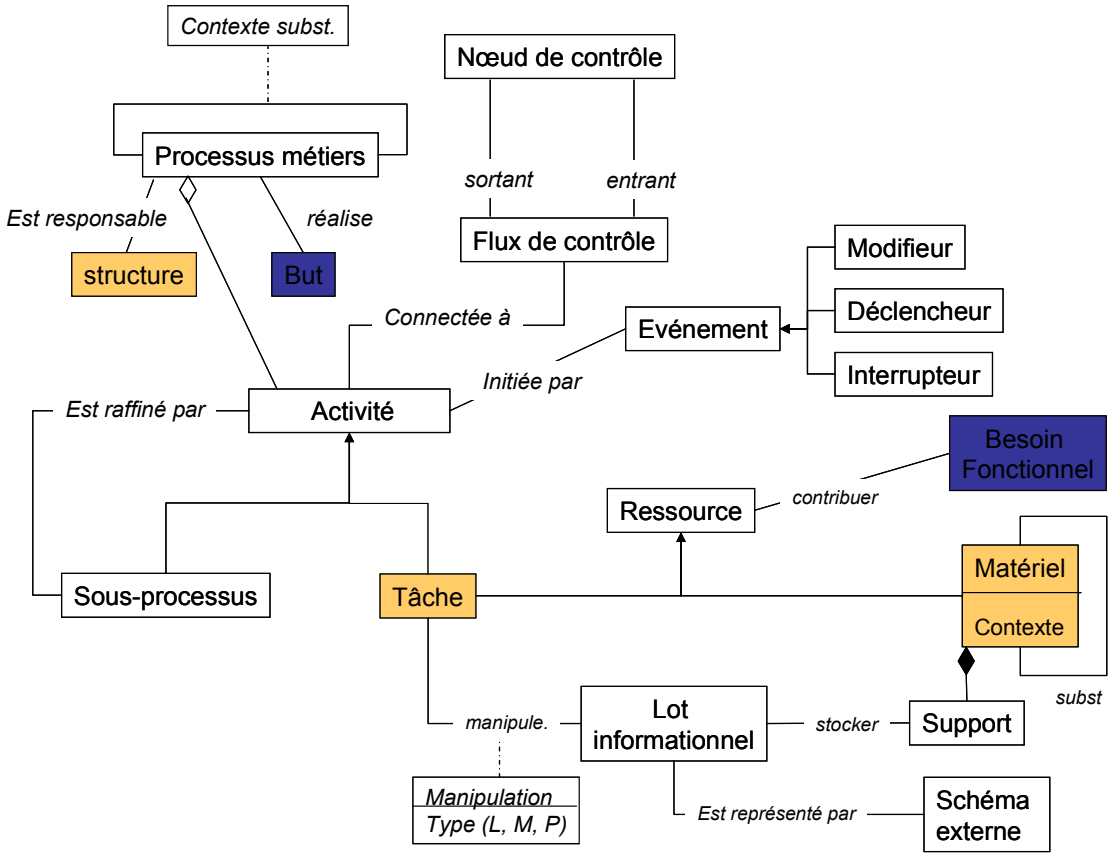

Figure 32. Méta-modèle de processus

Un processus dans notre méta-modèle est composé d'activités. Les activités peuvent être raffinées en sous processus jusqu'au niveau le plus fin appelé tâche. Dans le cas bibliothèque, par exemple, le processus « achat de livre » comporte huit activités : « enregistrement de la demande d'achat », « traitement de la demande d'achat », « enregistrement avis 1 », enregistrement avis 2 », « enregistrement avis 3 », « traitement annulation demande », « traitement annulation commande », « traitement enregistrement commande ». L'activité « traitement demande », par exemple, se décompose en deux tâches : « choix approbateur » et « envoi mail aux approbateurs ». Dans ce méta-modèle sont décrits les flux de contrôle des activités et l'ordre dans lequel elles s'enchaînent dans les processus. Certains processus se substituent à d'autres, sous certaines conditions exprimées dans le méta-modèle par la propriété « contexte » de l'association de substitution.

3.2.2 Le méta-modèle des besoins

Il s'agit, dans ce méta-modèle, de décrire les objectifs attendus du système d'information. Ces objectifs (ou buts) se déclinent en buts élémentaires que l'on nomme besoins, par un processus de raffinement. Dans ce rapport, nous nous focalisons, sur les besoins fonctionnels que doit assurer le système d'information et sur les besoins de sécurité garantissant les

propriété DICT de sécurité. Les autres besoins non fonctionnels (qualité, performance, etc.) sont hors champs d'étude.

La figure 33 présente notre méta-modèle des besoins. Comme le montre cette figure, il peut exister plusieurs façons de raffiner un but. Ces différentes possibilités de raffinement sont décrites par le biais du concept «Variante_raffinement». Etant donné qu'un raffinement n'est possible que sous certaines conditions, nous proposons de renseigner le concept «Variante_raffinement» par la propriété contexte qui décrit la condition d'application de ce raffinement. A noter que but et besoin fonctionnel sont des concepts d'ancrage avec le méta-modèle de processus et que, d'autre part, le besoin fonctionnel est aussi un concept d'ancrage avec le méta-modèle organisationnel.

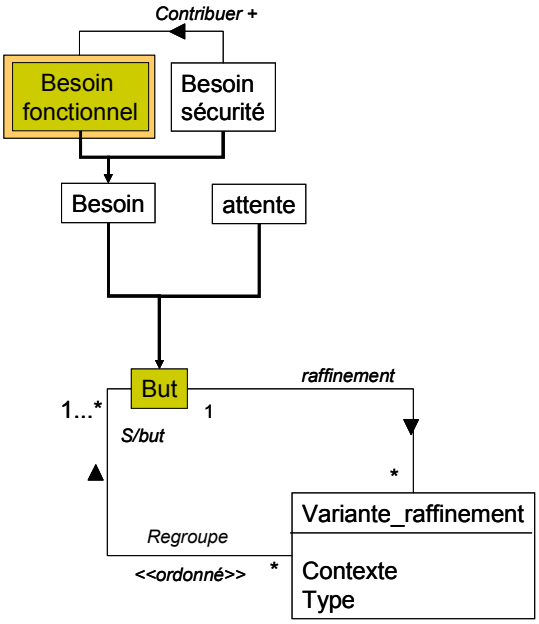

Figure 33. Méta-modèle des besoins

A titre d'exemple, dans le cas bibliothèque, le but « gestion des demandes d'approbation de livres » se raffine de deux façons : soit en « gestion de demandes d'approbation de livres pour la bibliothèque des sciences exactes » (V1111 dans le schéma en annexe 5) ou encore en « gestion de demandes d'approbation de livres pour la bibliothèque des sciences humaines » (V1112 dans le schéma en annexe 5). Chacune des deux variantes n'est valide que si la demande a été adressée respectivement à bibliothèque concernée par la demande (BSE pour V1111 et BSH pour V1112).
Une tâche d'un système d'information contribue à la réalisation d'un ou plusieurs besoins fonctionnels. C'est le cas par exemple de la tâche « enregistrer avis 1 » qui contribue à la réalisation des deux besoins fonctionnels cités ci-dessus. La tâche « enregistrer avis 2 » contribue, quant à elle, uniquement à la réalisation du besoin fonctionnel « gestion de demandes d'approbation de livres pour la bibliothèque des sciences exactes ».

Les buts pouvant obéir à un ordonnancement spécifique, nous proposons de les ordonnancer par défaut et de renseigner la propriété type associée à la variante pour maintenir ou non cet ordonnancement. Dans notre exemple, les deux variantes de raffinement du but « gestion des achats » exigent l'ordonnancement des buts issus du raffinement.

Nous précisons aussi la contribution des besoins de sécurité aux besoins fonctionnels. Le degré de contribution d'un besoin de sécurité sera explicité dans nos recherches futures.

Enfin, mentionnons que les buts de sécurité sont aussi raffinés jusqu'à obtention de besoins de sécurité. Cependant, le rattachement des besoins de sécurité aux tâches de sécurité appropriées se fait, à la seconde itération, à l'aide d'un processus d'enrichissement. Ce dernier exploite l'ontologie existante dans notre base de connaissances.

3.2.3 Le méta-modèle organisationnel

Il s'agit dans ce méta-modèle de regrouper tous les acteurs directement concernés par le système d'information. Ces acteurs, se trouvant dans une structure organisationnelle, ont à jouer des rôles bien définis, à exécuter des tâches déterminées relatives aux rôles qu'ils leur sont attribués. Ils peuvent aussi avoir un certain nombre de privilèges tel que celui de déléguer des tâches qui leur sont affectées.

Après un processus d'intégration et d'enrichissement des méta-modèles étudiés dans l'état de l'art, nous avons obtenu le méta-modèle de la figure 34. Ce méta-modèle, ainsi que ceux relatifs aux autres vues du système d'information, se veut être suffisamment générique pour pouvoir être instancié pour n'importe quel système d'information et n'importe quelle organisation. Par exemple, la prise en compte de deux catégories d'acteurs : les humains et agents logiciels (autonomes et non autonomes) contribue à la généricité du méta-modèle. Un deuxième exemple est la prise en charge de la flexibilité des délégations. Un troisième exemple serait de séparer l'affectation géographique des acteurs, des habilitations qu'ils peuvent avoir.

Comme le montre la figure 34, nous considérons les acteurs, nommés « agents » dans notre méta-modèle, et les rôles des acteurs comme des composants des structures organisationnelles concernées par le système d'information. Dans notre « cas bibliothèque », le bibliothécaire, le bibliothécaire en chef, l'étudiant, l'enseignant-chercheur sont des rôles. Monsieur X est un agent humain. L'application web « GesBiblio » est un agent logiciel.

Un agent humain a des aptitudes. Les stocker peut être utile dans la répartition des tâches dans un système d'information. De même, un agent logiciel a un niveau d'autonomie. Il réalise des besoins fonctionnels du système d'information. Il est hébergé par un matériel. A titre d'exemple, l'application web «GesBiblio» est non autonome et est hébergée par un serveur de l'université. Cette application web réalise un certain nombre de besoins fonctionnels des deux bibliothèques tels que «l'enregistrement d'une demande d'achat de livre ».
Un agent (humain ou logiciel) a un contexte qui peut impacter sa disponibilité pour l'exécution des tâches qui lui sont attribuées. De même, un matériel a un contexte qui peut influer sur la disponibilité des agents logiciels qu'il héberge. Enfin, pour assurer une meilleure disponibilité du système d'information, il est intéressant de savoir si un matériel est substituable. Cette information est fournie par l'association réflexive « substitue ». Le besoin fonctionnel est un concept d'ancrage avec le méta-modèle des besoins. Tâche et matériel sont les concepts d'ancrage avec le méta-modèle de processus.

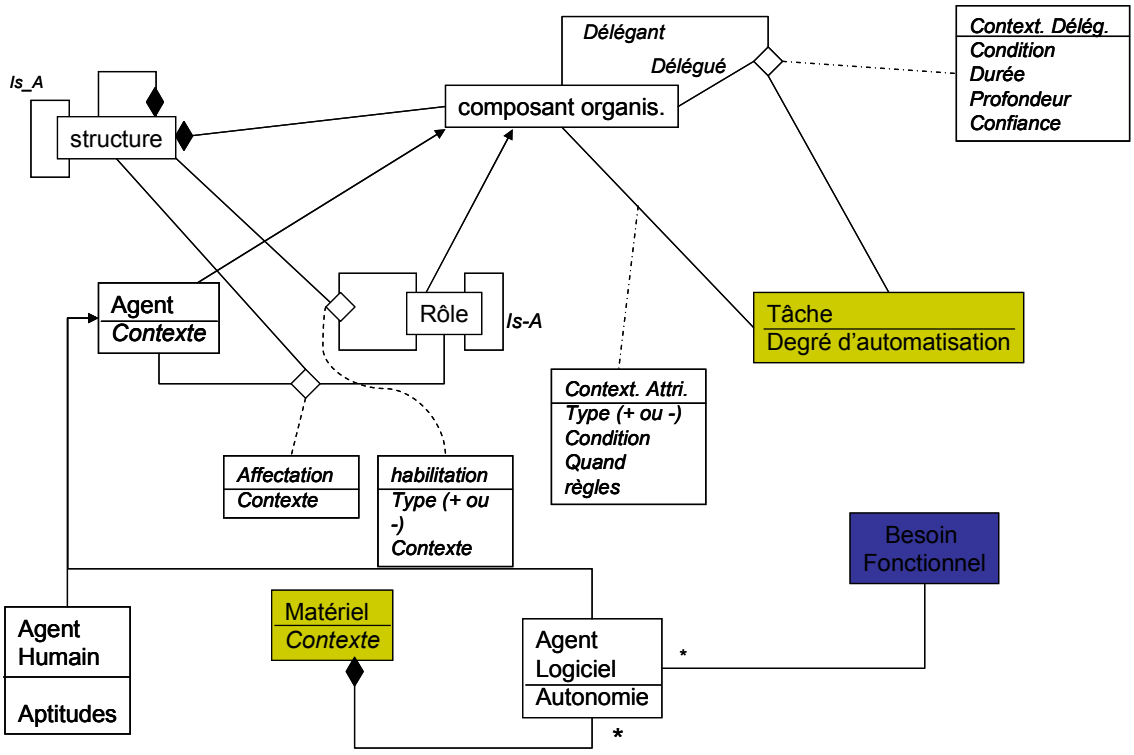

Figure 34 : Perspective organisationnelle du méta-modèle

Nous définissons deux types de classification des structures : la hiérarchie et l'agrégation. Dans notre exemple, la bibliothèque des sciences exactes fait partie de la faculté des sciences. Les deux bibliothèques ainsi que les deux facultés sont des structures universitaires.

Les rôles dans une organisation peuvent être aussi hiérarchisés. Tel est le cas du rôle « doctorant » qui est un sous rôle de «étudiant ». De même, le rôle de « doyen » est sous rôle de «enseignant-chercheur permanent» qui, à son tour, est sous-rôle de « enseignant-chercheur ».

Un *agent* est affecté dans une structure pour jouer un *rôle* donné. Son affectation peut être contrainte. Cette contrainte est exprimée dans le méta-modèle à l'aide de l'attribut «contexte» de l'association « affectation ». A titre d'exemple, « Monsieur X » est affecté en tant qu'«étudiant » dans la « faculté des sciences ». Il est aussi affecté en tant que « bibliothécaire » dans la bibliothèque des sciences selon le contexte « contrat en cours avec la bibliothèque des sciences exactes ».

De plus, un *rôle* peut être habilité, ou non, à jouer un autre *rôle* dans une *structure*. Cette habilitation peut aussi être contrainte. Cette contrainte est exprimée par l'attribut « contexte » de l'association « habilitation ». L'attribut « type » de cette association nous renseigne sur le type d'habilitation : positive ou négative. Une habilitation positive correspondra à un type de permission faisant partie des politiques de sécurité à mettre en œuvre. Une habilitation négative correspondra, quant à elle, à une interdiction.
Pour notre cas bibliothèque, le rôle « étudiant » est habilité à jouer le rôle de « bibliothécaire » dans la structure « bibliothèque » si et seulement si « il est sous-contrat avec cette bibliothèque, que cette bibliothèque fasse partie de la faculté à laquelle il appartient et que son contrat est en cours ». Cependant, le rôle « étudiant » ne peut en aucun cas être habilité à jouer le rôle « bibliothécaire en chef ».

Une tâche est attribuée à une composante organisationnelle (c'est-à-dire à un rôle ou un agent bien déterminé). Cette attribution peut être contrainte. L'attribut contexte de l'association « attribution » représente cette contrainte. Dans le cas bibliothèque, la tâche « enregistrement demande d'achat » est attribuée au rôle « enseignant-chercheur » avec la contrainte suivante : « la demande est soumise à la bibliothèque de la faculté dont dépend le demandeur ». La tâche «choix d'approbateur » est, en revanche, attribuée à l'application web «GesBiblio» qui est un agent.

L'exécution d'une tâche peut être déléguée à un délégué par un déléguant. Le délégué, ainsi que le déléguant, sont des composantes organisationnelles (rôle ou agent). Cette délégation peut être, dans certains cas, contrainte (sur la durée par exemple). Elle peut aussi être autorisée à être retransmise par le délégué. L'attribut « profondeur » de l'association « contexte de délégation » permet de limiter le nombre de niveaux de délégation autorisé. Dans le cas où il est égal à zéro, il y a interdiction de retransmission d'une délégation reçue. Le déléguant peut avoir un degré de confiance (> 0) sur le délégué pour l'exécution d'une tâche. Ce degré de confiance peut avoir une influence sur l'exécution d'une délégation lorsque deux délégués sont candidats à l'exécution d'une tâche, dans le cas d'indisponibilité du déléguant par exemple. Dans le cas bibliothèque, la tâche « enregistrement avis 2 » peut être déléguée par le rôle « doyen » à un rôle « enseignant-chercheur ». Cette délégation n'est possible que si le déléguant (c'est-à-dire le doyen) est absent, si le délégué (c'est-à-dire l'enseignant chercheur) n'a pas déjà été sollicité pour la même demande et que le déléguant fait partie de la faculté à laquelle appartient la bibliothèque réceptrice de la demande. D'après l'énoncé du cas, l'enseignant-chercheur délégué ne peut transmettre le privilège reçu par le doyen pour cette tâche.

3.3. Profils UML pour le niveau PIM

Comme décrit précédemment, la seconde étape de MoSIS est une étape d'analyse et de conception abstraite du système d'information sécurisé. De cette analyse découlent les modèles UML d'analyse. Parmi ces modèles, on peut citer le diagramme des cas d'utilisation pour la description des fonctionnalités du système et le diagramme des classes pour la description de la structure statique du système. Ces modèles constituent le niveau PIM du système d'information sécurisé. Ils doivent prendre en compte aussi bien les concepts fonctionnels intrinsèques à l'application que les politiques de sécurité entreprises pour sécuriser cette application. Ces modèles sont des instanciations de méta-modèles de niveau PIM qui eux sont des extensions des méta-modèles de UML. Ces extensions sont fondées sur des profils.

Les paragraphes 3.3.1 et 3.3.2 fournissent respectivement une description des profils des cas et des classes. Le paragraphe qui suivra (3.3.3) présente les règles de passage du niveau CIM au niveau PIM.

3.3.1 Profil UML pour la description des fonctionnalités

Un profil, comme défini précédemment, décrit la syntaxe et la sémantique des concepts utilisés dans un domaine particulier. Les concepts du domaine sont représentés en utilisant les stéréotypes. Comme une classe, un stéréotype peut avoir des propriétés. Lorsque un stéréotype est appliqué à un modèle les valeurs de ces propriétés sont appelées TaggedValues

La figure 35 présente quelques stéréotypes définis pour la description des fonctionnalités d'un système sécurisé.

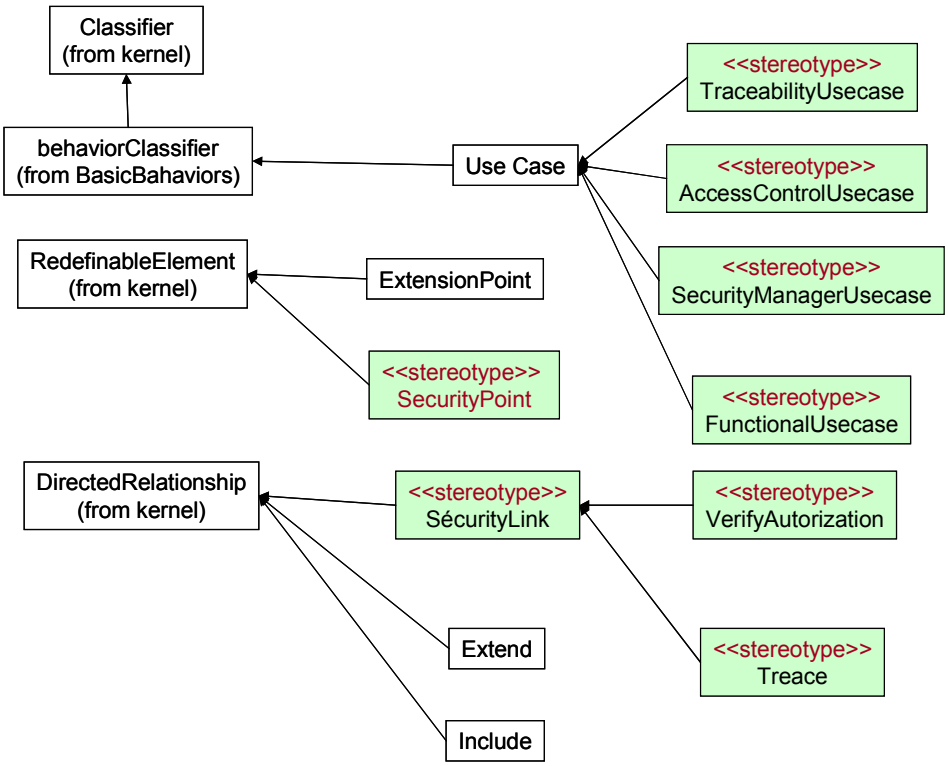

Figure 35. Stéréotypes pour la sécurité

Dans l'ensemble des cas d'utilisation de UML, nous distinguons les cas décrivant les fonctionnalités du système des cas de sécurité. Les cas d'utilisations de sécurité sont de trois types : les gestionnaires de la sécurité (securityManagerUseCase), les contrôleurs d'accès (*AccesscontrôlUsacase)* et les traceurs des actions exécutées dans les use cases (*TraceabilityUsecase*). Les gestionnaires servent à l'administration des politiques de sécurité définies dans l'organisation. Il s'agit par exemple de la définition des rôles, des affectations, des habilitations, des délégations, des attributions positives (permissions) ou négatives (interdictions), etc. Les contrôleurs d'accès sont en charge de l'analyse et du respect des politiques de sécurité définies par les gestionnaires. Les traceurs permettent d'enregistrer les informations pertinentes sur les éléments importants pour le système et qui ont un impact sur la vulnérabilité du système.
Le langage UML propose généralement deux types de liens entre cas d'utilisation : le lien d'extension (*extend*) et le lien (*include*). Pour prendre en charge les besoins de sécurité, nous proposons de l'étendre en rajoutant des liens de sécurité sous le stéréotype « *secure* ». Ils permettent de relier des cas fonctionnels aux cas de sécurité. Au même titre que les points d'extension (extension point) existant dans UML, on définit les points de sécurité (security point) matérialisant les points d'entrée ou d'appel des cas d'utilisation de sécurité par les cas fonctionnels.

Notre méta-modèle PIM pour la description des fonctionnalités d'un système d'information sécurisé est représenté par la figure 36.

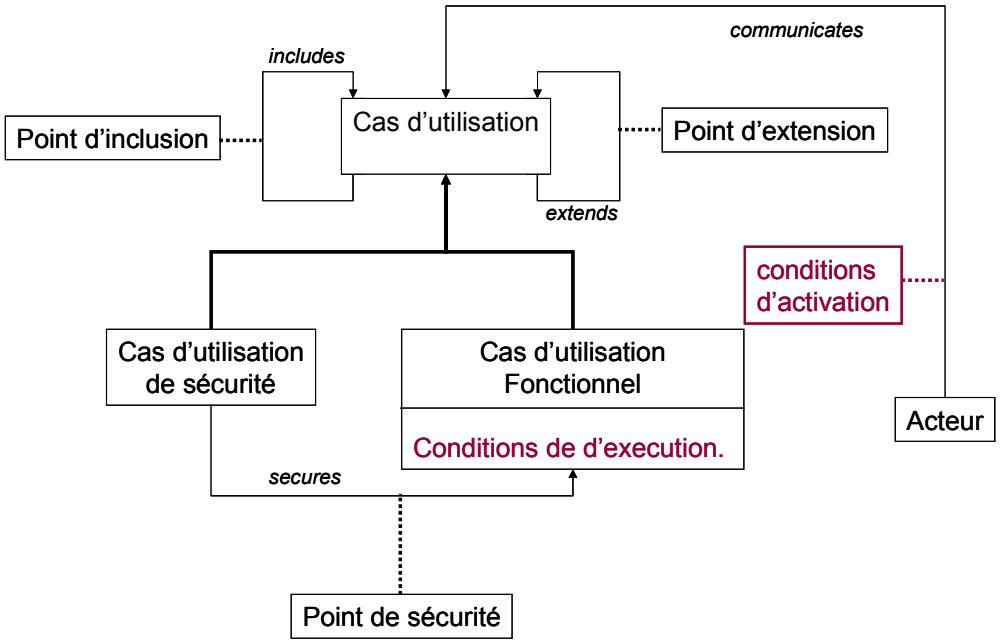

Figure 36. Méta-modèle PIM pour la sécurité

Comme le montre la figure 36, un diagramme des cas d'utilisation d'une application sécurisée contient des cas fonctionnels et des cas de sécurité. Ces derniers sont sollicités par les cas fonctionnels. Un cas fonctionnel obéit à des règles de déclenchement dictées par le processus dont il fait partie. Ces règles sont représentées au niveau CIM à l'aide des nœuds de contrôle. Un acteur, ayant le droit d'activer un cas, ne peut l'activer que sous certaines conditions. Généralement les conditions d'activation découlent des contextes de permission et délégation mentionnées au niveau CIM.

3.3.2 Profil UML pour la description de la structure statique

L'élaboration du méta-modèle pour la description de la structure statique du système d'information sécurisé a été guidée par l'étude de cas fournie par Ifremmont. La version du méta-modèle présentée dans ce document s'adapte donc à ce cas d'étude et sera élargie au fur et à mesure si de nouveaux concepts doivent être pris en compte.

Les concepts fonctionnels du système Res@mu sont décrits principalement par des diagrammes de classes UML. Notre effort porte donc en grande partie sur l'élaboration d'un formalisme permettant d'exprimer une politique de sécurité sur les entités de ces diagrammes de classe. Pour ce faire, nous avons proposé une extension d'UML permettant d'intégrer un contrôle d'accès à base de rôles de type RBAC. Dans notre proposition de méta-modèle dédié au niveau PIM, nous avons décidé d'intégrer le modèle RBAC1. Les modèles RBAC et RBAC1 sont décrits en annexe 1.

En vue d'exprimer des propriétés de sécurité au niveau des diagrammes de classes UML, nous proposons d'étendre le méta-modèle d'UML associé aux diagrammes de classes en y intégrant des mécanismes de contrôle d'accès. Le méta-modèle dédié à cet effet sera décrit par la syntaxe abstraite présentée dans la figure 37, ainsi que la sémantique associée à chaque entité abstraite. Nous proposons également une syntaxe concrète permettant d'enrichir la représentation graphique des diagrammes de classes UML par des informations de sécurité.

La figure 37 spécifie les concepts utilisés pour sécuriser au moyen de règles de contrôle d'accès les classes du modèle UML. Notons que les méta-classes « *class* » et « *Operation* » sont issues du méta-modèle d'UML.

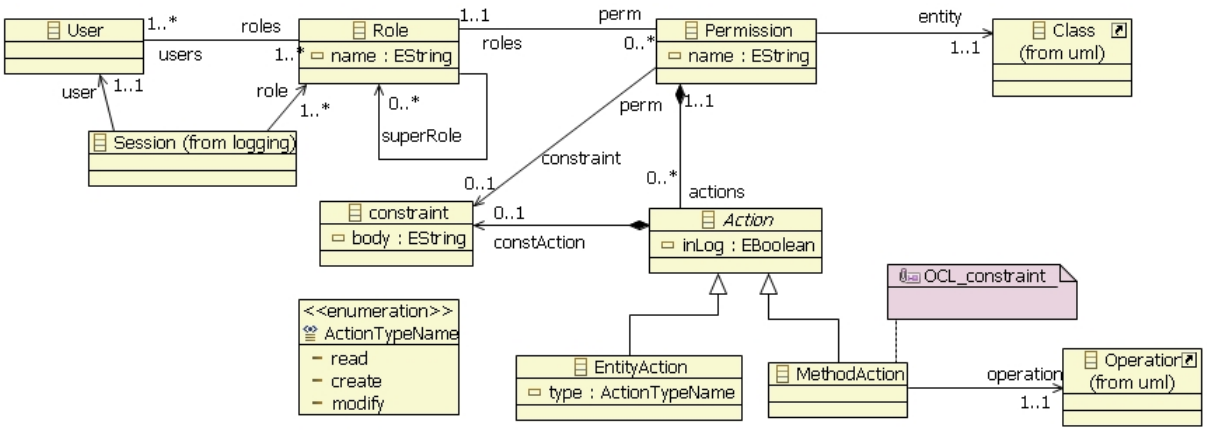

Figure 37. Méta-modèle de sécurité au niveau PIM

Ce méta-modèle reprend en partie les notions issues de RBAC [ANSI, 2004], notamment les notions de rôle, de permission, d'utilisateur et de session. Dans ce qui suit, nous décrivons la sémantique associée à chacun des concepts.

a) Les rôles

La sémantique que nous procurons au concept de rôle est identique à celle définie dans RBAC. Cependant, dans RBAC l'association d'un utilisateur à un rôle est définie par la relation :

$$UserAssignement \subseteq User \times Role$$

Dans l'état actuel de notre contribution, nous interdisons l'interaction d'un utilisateur avec le système sans être associé à un rôle particulier. L'existence d'un utilisateur est donc conditionnée par l'existence d'un rôle auquel cet utilisateur sera associé. C'est ce qui explique la multiplicité (1..*) du côté de la méta-classe *ROLE* entre *USER* et *ROLE*. Dans le système Res@mu, la méta-classe *ROLE* structure les différents types d'utilisateurs du système et est instanciée par : *Administrator, Operator, Team Member,* etc. Quant à la hiérarchie entre rôles, elle indique qu'un rôle hérite toutes les permissions du rôle parent et peut disposer de permissions qui lui sont propres.

Au niveau de la syntaxe concrète, nous proposons de représenter les rôles par des classes décorées avec le stéréotype « Rôle ». La figure 38 présente les rôles *RESCUER* et *TEAM MEMBER* avec une relation de hiérarchie.

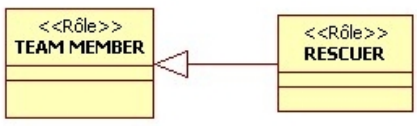

Figure 38. Syntaxe concrète associée à la méta-classe ROLE

Notons que le marquage des propriétés de sécurité à l'aide de stéréotypes permet de les différencier des entités fonctionnelles du système. Ce faisant, l'héritage d'UML est réutilisé pour exprimer la hiérarchie de rôles car la méta-classe *ROLE* est définie comme étant une extension de la méta-classe « *Class* » d'UML.

b) Les permissions

La méta-classe ***PERMISSION*** permet de définir des droits d'accès de *ROLE* à *CLASSE*. Elle vise ainsi à sécuriser les entités du modèle fonctionnel en indiquant les actions autorisées sur ces entités. Contrairement au modèle RBAC qui permet l'association d'une permission à plusieurs rôles, notre modèle n'admet qu'un seul rôle par permission. Cela n'affecte en rien l'expressivité de notre modèle vu qu'il est possible de simuler les classes associatives par des associations simples et de factoriser les permissions redondantes. On peut aboutir ainsi à un modèle strictement conforme à RBAC. Néanmoins, ce choix de spécification a été guidé par la syntaxe concrète qu'on a associée à la méta-classe ***PERMISSION***. En effet, nous proposons d'exprimer les permissions sur le diagramme de classes en utilisant des classes associatives décorées par le stéréotype « Permission ». De ce fait, une permission ne peut exister qu'entre un seul rôle et une seule classe UML.

La figure 39 présente un exemple de permission (*ManagementActPerm*) destinée à contrôler l'accès à l'entité *ManagementAct* par un membre d'une équipe médicale (rôle *TEAM MEMBER*). Ce contrôle d'accès est réalisé au moyen d'un ensemble d'actions autorisées listées dans la permission *ManagementActPerm.*

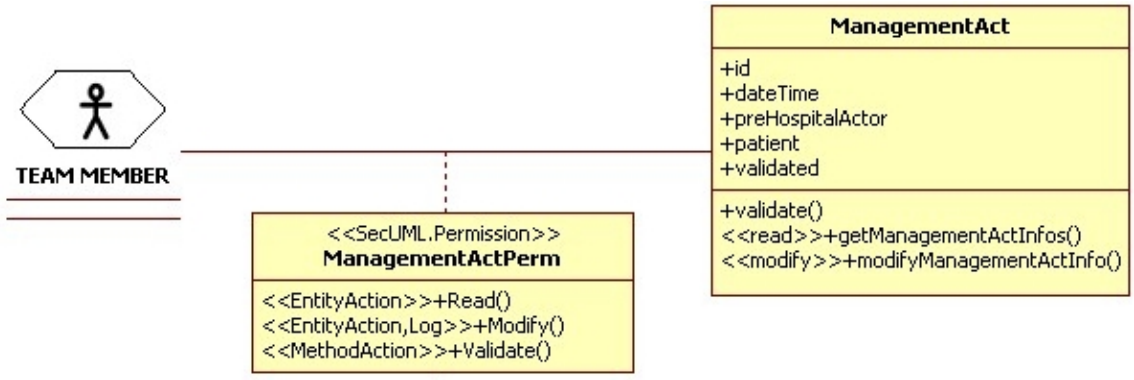

Figure 39 Exemple de permission associé à l'entité ManagementAct

c) Les actions

Une permission permet d'autoriser une ou plusieurs actions sur la classe fonctionnelle à laquelle elle est associée. Dans le méta-modèle (Figure 37), cela se traduit par un lien de composition entre la méta-classe *PERMISSION* et la méta-classe abstraite *ACTION*. Nous distinguons deux catégories d'actions que nous représentons par des spécialisations de la méta-classe *ACTION* :

- Les *EntityAction* : correspondent à des actions sur toute la classe UML désignée par la permission. Nous identifions trois types d'actions associées à une entité : de lecture, de modification et de création. Cela est justifié par le fait qu'à un niveau d'abstraction élevé, les services de bases relatifs à la manipulation des données ne sont pas explicités par des opérations, car ils sont souvent liés aux choix d'implémentation.

C'est-à-dire que ces détails n'apparaîtront qu'au niveau du PSM. Pour s'affranchir de cette absence, nous proposons un type d'action plus abstrait qui permettra d'identifier certaines de ces opérations de base. Nous introduisons ainsi l'entité *EntityAction* qui permet de modéliser les actions de bases sur les classes.

- Les ***MethodAction*** : autorisent l'exécution de méthodes encapsulées dans la classe UML désignée par la permission. Une *MethodAction* doit donc explicitement faire référence à une opération de la classe UML. En effet, pour intégrer le modèle de sécurité au modèle fonctionnel, il est important d'identifier les actions à protéger. Si l'on se place au niveau du diagramme de classes, ces actions seront les services fournis par les classes aux utilisateurs et qui seront modélisés par des opérations de classes. Pour référencer ces actions dans notre modèle nous introduisons donc la méta-classe *MethodAction*.

Dans la syntaxe concrète que nous proposons pour représenter les actions nous considérons que ces actions sont des opérations de permission. *EntityAction* et *MethodAction* sont donc des extensions de la méta-classe UML « *Operation* ». Ceci a un double avantage, tant au niveau de la sémantique associée aux actions qu'au niveau de leur syntaxe concrète. En effet, le fait de considérer que les actions sont des opérations nous permettra par la suite d'indiquer les pré-conditions de leur déclenchement. Par exemple, on peut considérer que la validation d'un *ManagementAct* ne peut être réalisée que sous certaines conditions. Au niveau de la syntaxe concrète, cela permettra de stéréotyper les opérations d'une permission en leur associant une catégorie d'action.

La permission illustrée dans la figure 39 donne le droit à un *TEAM MEMBER* de réaliser trois actions :

- A1: «lire un acte de soin »
- A2 : « modifier un acte de soin »
- A3 : « valider un acte de soin ».

La figure 40 reprend l'exemple de la figure 39 sous la forme d'un diagramme d'objets instanciant notre méta-modèle de sécurité. On y voit apparaître la permission *ManagementActPerm* entre le rôle *TEAM MEMBER* et la classe *ManagementAct*. Cette permission est liée à trois actions. Les actions A1 et A2 sont des *EntityAction* qui confèrent à un *Team Member* le droit de lire ou de modifier les données d'un *ManagementAct*. Cela peut être réalisé soit directement soit au moyen des méthodes *getManagementActInfo* et *modifyManagementActInfo*} qui sont respectivement des méthodes de lecture et de modification. Quant à l'action A3, elle correspond à une *MethodAction* associée à la méthode *validate* de *ManagementAct*. Un Team Member a donc le droit d'exécuter la méthode *validate*}.

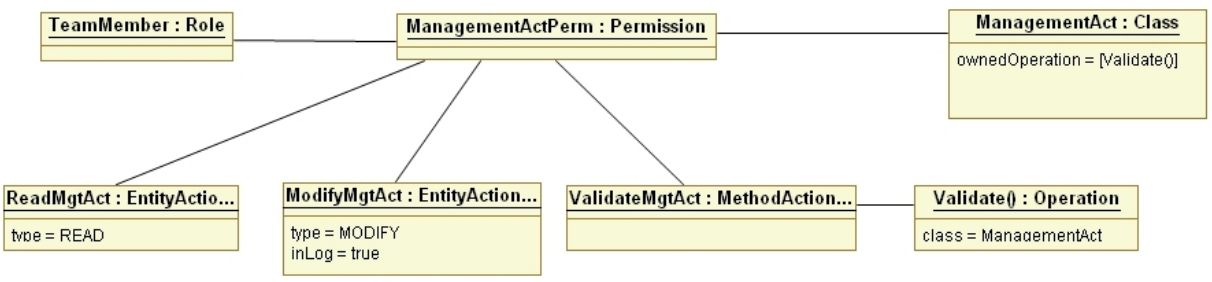

Figure 40. Instance du méta-modèle

d) La propriété *inLog*

Les deux types d'actions, présentées précédemment, héritent de la méta-classe abstraite *ACTION*. Cette dernière introduit une propriété structurelle fondamentale au concept d'action : *inLog.* Il s'agit d'un attribut d'une action, permettant d'introduire la notion de traçabilité explicite dans le modèle de sécurité. En effet, nos préoccupations de sécurité couvrent les propriétés (DICT). Les trois premières sont couvertes par les mécanismes de contrôle d'accès autour duquel s'articule le modèle de sécurité. Tandis que la propriété de traçabilité est considérée de façon implicite. Cependant, nous proposons d'introduire explicitement ce mécanisme pour pouvoir spécifier les actions pour lesquelles la traçabilité est impérative. Par exemple, le modèle de la figure 39 indique que toutes les actions de modification doivent obligatoirement être tracées, contrairement aux actions de lecture ou de validation.

e) Les contraintes

La propriété *constAction* d'une action fait référence à la méta-classe *CONSTRAINT* et permet d'étendre le modèle de base pour y introduire la notion de contrainte d'autorisation. Cette dernière se présente sous la forme d'un prédicat qui définit un contexte particulier à la réalisation d'une action et qui dépend de l'état du système.

Nous proposons de spécifier les contraintes comme des invariants OCL exploitant les données d'exécution. Ces données incluent, en plus des données fonctionnelles de l'application (instance du PSM), les données de contrôle d'accès dont les données de session essentielles pour identifier l'utilisateur en cours ainsi que ses rôles actifs. Concrètement, les contraintes doivent pouvoir être évaluées durant l'exécution afin de conditionner l'application des règles lors du contrôle d'accès. En effet, si une contrainte est appliquée à l'une des actions d'une permission (et non à la « vraie » action du système) alors l'action ne sera autorisée que si le prédicat de la contrainte est vrai au moment de l'appel de l'action système correspondante.

La notion de contrainte, et plus généralement de contexte, répond à un besoin impératif dans l'expression des politiques de sécurité. En effet, si nous reprenons l'exemple de la figure 39, la politique qui en découle autorise n'importe quel utilisateur jouant le rôle de *TeamMember* à lire le dossier de n'importe quel patient. Ceci est, bien évidemment, contraire aux conventions d'usage présentées ci-dessous. D'où la nécessite d'introduire le concept de contrainte. Pour illustrer l'utilisation des contraintes, nous proposons de reprendre l'exemple de la permission ***ManagementActPerm*** et la politique qu'elle implémente en y introduisant les contraintes qui s'imposent. Les règles correspondantes à nos trois actions deviendraient ainsi :

- R1 : « un membre d'équipe a le droit de lire un acte de soin s'il a participé à la prise en charge du patient. »
- R2 : « un membre d'équipe a le droit de modifier un acte de soin s'il a participé à la prise en charge du patient et si l'acte n'a pas encore été validé. »
- R3 : « un membre d'équipe a le droit de valider un acte de soin s'il l'a réalisé »

Comme illustré dans la figure 41, les contraintes associées à ces règles peuvent être exprimées dans une note liée à la permission *ManagementActPerm*. Cette note contient essentiellement l'expression de ces contraintes en OCL. Ce sont concrètement des pré-conditions portant sur nos trois actions (opérations de la permission).

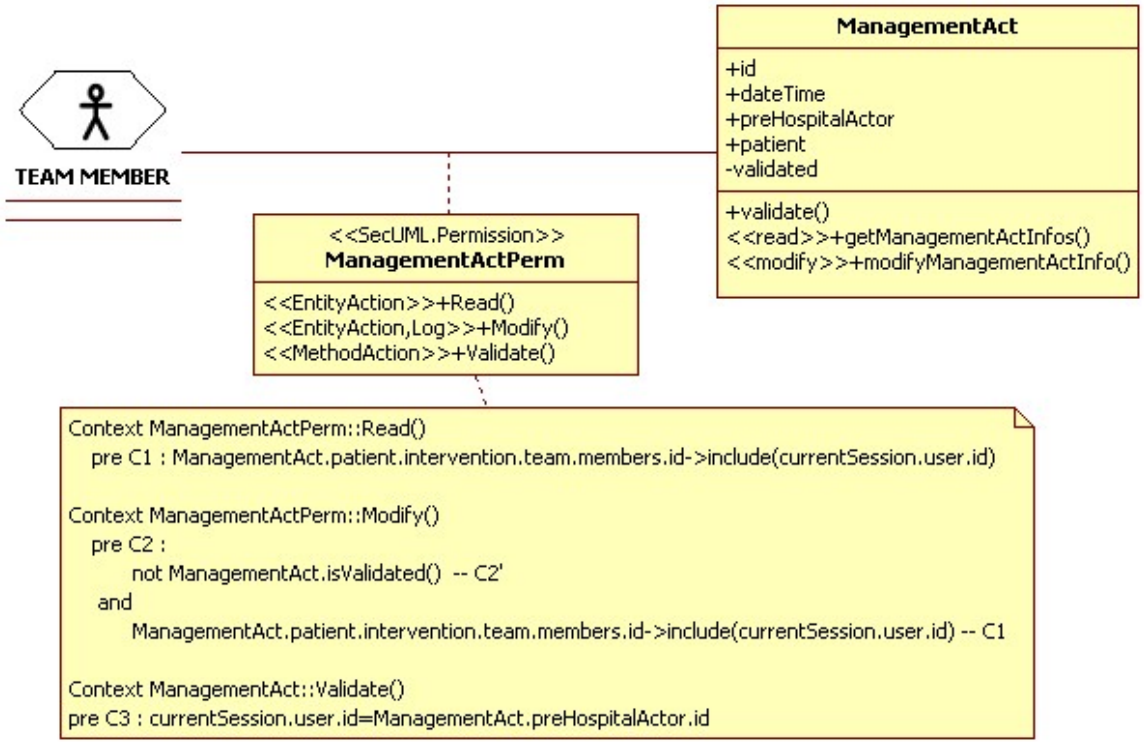

Figure 41. Exemple de contraintes d'autorisation

Si nous nous intéressons de plus près aux expressions de ces contraintes d'actions, nous voyons apparaître les données fonctionnelles avec la classe protégée comme point d'entrée (*ManagementAct* dans l'exemple). Mais également l'entité *currentSession* qui représente la session de l'utilisateur demandant l'autorisation pour l'action contrainte. A partir de cette entité, il est possible d'identifier l'utilisateur et de faire le lien avec sa représentation dans le modèle fonctionnel.

Une autre constatation pouvant découler de l'exemple concerne l'intersection (logique) entre les trois contraintes. Nous remarquons, en effet, qu'il est possible de factoriser la contrainte C1 (associée à R1) pour les trois règles. En effet, pour accéder au dossier d'un patient -- que ce soit pour le lire, le modifier ou le valider -- il faut que la personne participe à la prise en charge du patient. En d'autres termes on a les propriétés suivantes :

$$C3 \Rightarrow C1 \ , \ C2 \Leftrightarrow C1 \wedge C2'$$

Notons que C1, C2 et C3 représentent respectivement les contraintes issues des règles R1, R2 et R3 et que la contrainte C2' vérifie que l'acte n'a pas encore été validé.

Ce cas de figure justifie l'introduction de la propriété *constraint* au niveau de la méta-classe *Permission*. Cette contrainte de permission permet ainsi d'appliquer une contrainte sur l'ensemble des actions d'une permission. L'action d'une permission n'est donc autorisée que si la contrainte spécifique à l'action ainsi que la contrainte de la permission sont vérifiées.

La figure 42 illustre la factorisation des contraintes de nos trois règles dans l'instance du méta-modèle. On y voit apparaître la contrainte de permission *InterventionInvolvment* qui instancie

la contrainte C1. On note, par conséquent, l'absence de contrainte d'action au niveau de l'action de lecture *ReadMgtAct* (A1) ainsi que la simplification de la contrainte au niveau de la deuxième action (A2).

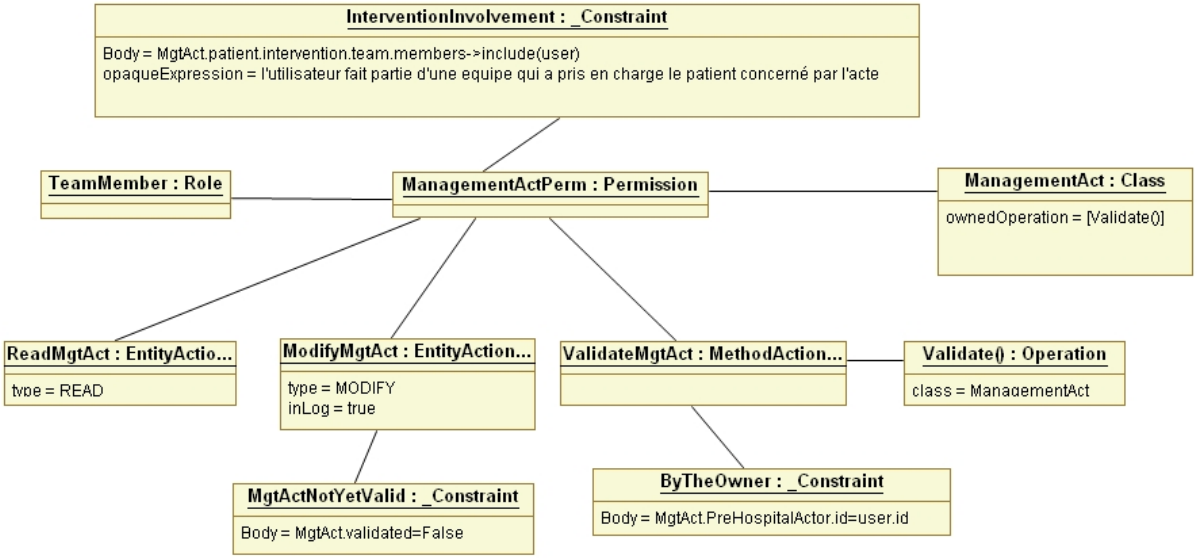

Figure 42. Exemple de contraintes de permission

Notons que l'apport de la notion de contrainte de permission dans le méta-modèle de sécurité n'est pas fondamental. Cependant, elle s'impose comme mécanisme de simplification dans l'expression des règles.

3.4. Les règles de transformation CIM-PIM

Bien que les méta-modèles de niveau CIM et PIM ne soient pas totalement finalisés, ils contiennent néanmoins les concepts clés des mécanismes de contrôle d'accès. Dans ce paragraphe nous présentons de façon informelle quelques règles de transformation d'un modèle CIM vers un diagramme des cas de niveau PIM. Puis nous décrivons, de façon informelle aussi, les liens entre les concepts de niveau CIM et ceux du diagramme des classes du niveau PIM.

3.4.1 Transformation d'un modèle de niveau CIM vers un diagramme des cas

Les besoins fonctionnels et de sécurité d'une application représentant un système d'information sécurisée peuvent être regroupés en packages : le package décrivant les besoins constituant le « front office » de l'application et celui décrivant les besoins constituant son « back office». Dans le premier package on retrouve tous les cas d'utilisation correspondant aux besoins métiers et ceux les sécurisant. Dans le second package, on retrouve ceux permettant de définir et d'évaluer les politiques de sécurité.

La génération du diagramme des cas d'une application sécurisée à partir de sa description au niveau CIM se fait en trois temps (Figure 43).

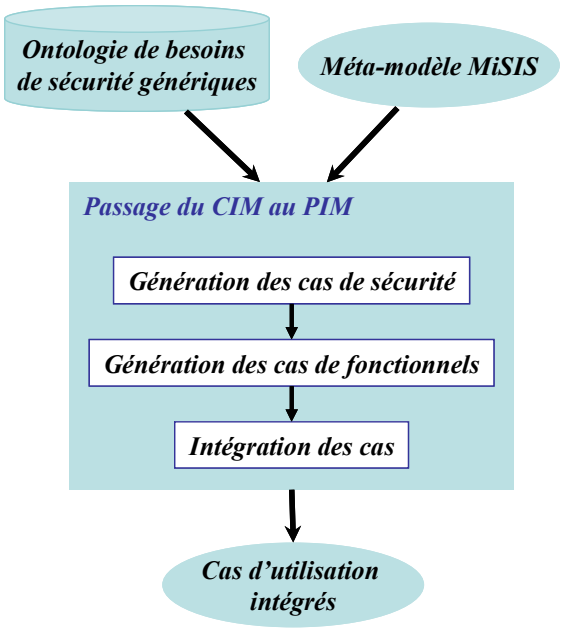

Figure 43. Génération du diagramme des cas

Dans un premier temps, on génère l'ensemble des cas de sécurité : ceux sécurisant les besoins métiers ainsi que ceux de définition et d'évaluation des politiques de sécurité. Pour ce faire, on exploite la description des besoins de sécurité du méta-modèle de MoSIS et l'ontologie des besoins de sécurité génériques. Cette dernière est un constituant de la base de connaissances. On procède à une mise en correspondance des besoins de sécurité spécifiés dans le méta-modèle avec ceux se trouvant dans l'ontologie. Cette mise en correspondance permet de retrouver les cas d'utilisation génériques correspondants.

La seconde étape consiste à déduire les cas d'utilisation fonctionnels. Pour cela on s'appuie sur des règles de transformation de concepts du méta-modèle en concepts de diagramme de cas fonctionnels. Parmi ces règles on peut citer les deux règles suivantes :

Règle 1 : Si une tâche automatisable T1 dans le méta-modèle a été attribuée à un rôle R1
alors :
- R1 devient acteur dans le diagramme des cas
- A1 est un cas d'utilisation fonctionnel
- les conditions se trouvant dans l'association *attribuer* entre R1 et T1 constituent les conditions d'exécution du cas associé à T1.

Règle 2 : Toute tâche T1 attribuée à l'application fait partie du cas d'utilisation de l'activité ou de la tâche s'exécutant juste avant T1.

Enfin dans un troisième temps, on procède à l'intégration des cas fonctionnels avec les cas de sécurité appropriés. Cette intégration s'appuie sur les liens existant entre les besoins fonctionnels et ceux de sécurité se trouvant dans le méta-modèle des besoins. En effet, si un besoin de sécurité contribue à la réalisation d'un besoin fonctionnel, alors un lien de sécurité est mis en place du cas fonctionnel correspondant au besoin fonctionnel vers le cas de sécurité correspondant au besoin de sécurité.

3.4.2 Liens entre les concepts de niveau CIM et le diagramme des classes

Dans le CIM sécurisé, les politiques de sécurité apparaissent principalement dans la perspective organisationnelle du méta-modèle. Cette dernière inclut les concepts assez similaires à ceux du modèle de contrôle d'accès dans une organisation tel que « ***OrBac*** » ([Abou El Kalam, 2003]), mais modélisés à un haut niveau d'abstraction. Il inclut également les concepts des approches pertinentes du CIM tels que l'ingénierie des besoins et la modélisation des processus métiers (Business Process Management). L'écart entre le niveau d'abstraction du modèle CIM et celui du modèle PIM produira des pertes d'information durant la transition entre CIM et PIM. En effet, dans la proposition actuelle, le CIM est plus général et couvre un contexte plus large que le PIM. Par exemple, le modèle PIM ne considère pas les processus, ni plusieurs niveaux d'automatisation des tâches. Par conséquent, cette section est consacrée à la clarification des concepts partagés et ceux qui ne sont pas couverts entre les modèles de ces deux niveaux.

a) Concepts couverts (partagés)

Dans cette section, nous présentons les concepts du méta-modèle CIM qui correspondent à un ou plusieurs concepts du méta-modèle PIM. La figure 43 donne un aperçu de ces liens de correspondance.

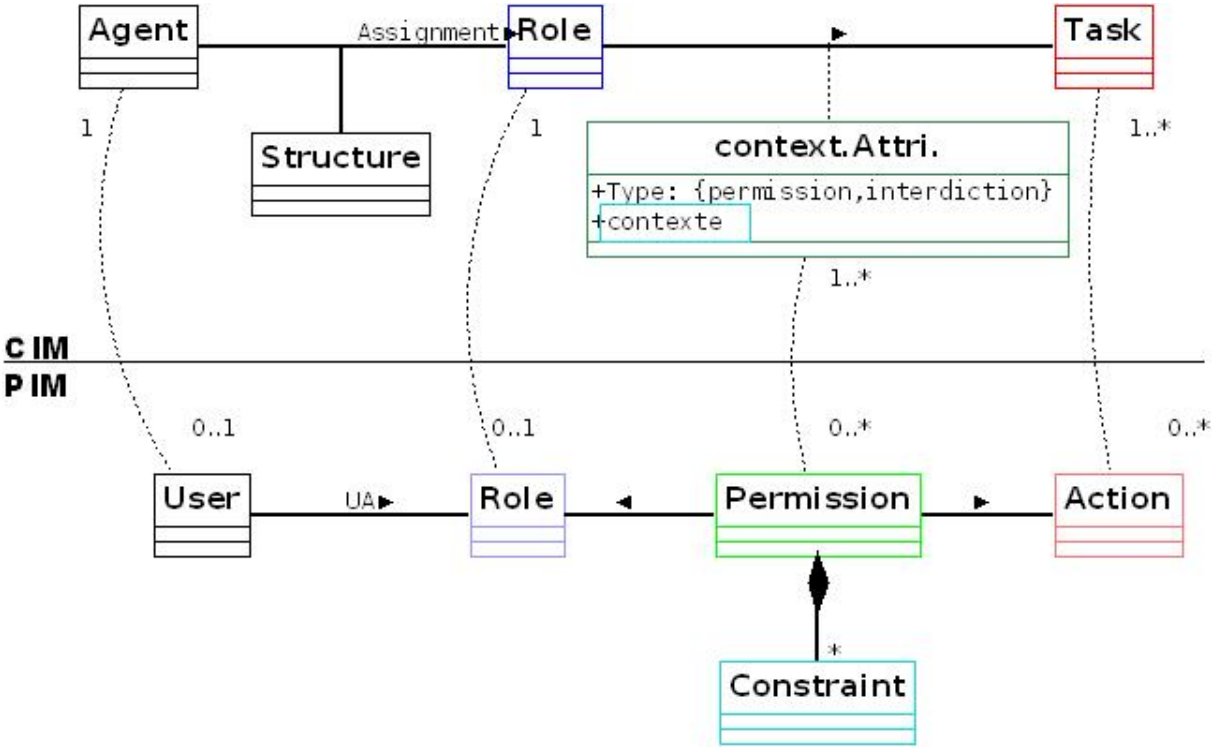

Figure 43. Traçabilité de concepts CIM et PIM

Rôles
Bien que le concept de rôle apparaisse dans les deux méta-modèles, sa sémantique diffère selon le niveau d'abstraction. En effet, il représente, dans le CIM, tous les types d'acteurs, humains ou non, impliqués dans le processus métier de l'organisation alors qu'au niveau PIM, il représente le type d'utilisateurs ayant des interactions avec le système d'information. Ainsi,

le concept de rôle dans le modèle PIM est un sous-ensemble de celui du méta-modèle CIM et uniquement les rôles représentant les utilisateurs du système (véritables acteurs du système) sont représentés dans les rôles du modèle PIM. En outre, dans les deux niveaux, les rôles possèdent une hiérarchie d'héritage à travers le lien réflexif 'Is-A' dans le méta-modèle CIM et '*superRole*' dans le modèle PIM.

Agents
Le concept de 'sujet' (selon la définition de *ORBAC*) ou d'utilisateur (selon la définition dans notre modèle PIM) est utilisé pour modéliser les utilisateurs du système. Dans le modèle sécurisé CIM, il est représenté par le concept d'*agent*. Ce concept, généralement utilisé dans l'ingénierie des besoins, est plus générique que la notion de d'utilisateur (ou sujet) parce qu'il inclut aussi bien les acteurs humains que les acteurs logiciels. En plus, ces agents humains peuvent être des agents de l'environnement sans nécessairement être des utilisateurs du système. Nous admettons que seul le sous-ensemble d'agents du CIM, associés aux rôles représentés par les rôles dans le PIM, va apparaître comme utilisateur dans le niveau PIM.

Affectations
L'affectation est une association reliant les entités *Rôle, Agent* et *Structure*. Il permet d'affecter un *rôle* à un *agent* dans une structure (aussi appelée *Organisation*). Au regard de l'entité *Structure* reflétant les aspects organisationnels qui ne sont pas présents dans le méta-modèle PIM, ce lien est équivalent dans le PIM au concept d'affectation entre l'*utilisateur* et le *rôle*. Le lien d'affectation peut se traduire par *UserAssignment* sans l'entité *Structure*. Cependant, cela peut générer des effets de bord et doit donc être faite avec prudence. Pour y remédier, nous admettons que le contrôle d'accès basé sur les rôles est spécifique à une seule structure. Alors, l'association ternaire *Agent/Structure/Rôle* peut se décomposer et le modèle PIM est dérivé pour chaque instance de la structure de la méta-classe.

Tâches
La notion de tâche n'apparaît pas dans le méta-modèle PIM. Cependant, elle est très proche du concept d'*action*. En effet, une tâche est définie dans le domaine de la modélisation des processus métiers comme étant une activité atomique affectée à « *un utilisateur ou application* » (OMG). Pour le méta-modèle PIM, une tâche peut être considérée comme étant une ou plusieurs actions.

Selon le point de vue de sécurité, les tâches en interaction avec les humains (par exemple ceux qui impliquent les utilisateurs du système) sont sensibles et nécessitent une politique de contrôle d'accès. Dans le méta-modèle CIM, ces tâches font référence aux utilisateurs autorisés dans la classe association '*Context Attr.*'. Cette association entre la méta-classe *Rôle* et la méta-classe *Tâche* peut être assimilée dans le PIM sécurisé à une permission. La correspondance entre le contexte d'affectation d'une tâche et une permission est conditionnée par la correspondance entre *Rôle* et *tâche* dans le niveau CIM et *Rôle* et *Action* dans le niveau PIM. Le sous-ensemble de tâches pertinent pour le niveau PIM exclut les tâches automatiques et manuelles. Dans ce PIM, ne sont représentées que les tâches nécessitant une interaction avec un utilisateur.

b) Concepts non couverts

Les méta-modèles CIM et PIM partagent plus ou moins les concepts de RBAC. Ceci permet de considérer que RBAC est un élément clé pour la transition du CIM au PIM. Cependant, quelques concepts présents dans le CIM sécurisé ne sont pas couverts par le PIM sécurisé.

Ceci est justifié par le fait que ces concepts n'apparaissent pas dans le modèle de base de RBAC qui est la charnière du méta-modèle PIM. Les concepts non couverts permettent, peut être, d'enrichir le modèle RBAC par introduction d'autres concepts avancés.

Structure
Généralement appelé *organisation*, cette méta-classe représente des groupes structurés de sujets ayant des rôles divers. Dans la version courante du méta-modèle CIM, ce concept est sollicité dans le lien d''*affectation*' de rôles. Un agent peut alors jouer plusieurs rôles dans une structure spécifique. Ce concept n'est pas ainsi considéré dans le PIM sécurisé mais peut être simulé par l'utilisation de nouveaux rôles entre le *Structure* et les *Rôles* du CIM[3]. Cependant, l'utilisation de l'entité *Structure* est importante dans l'affectation des permissions ou dans la dynamique de séparation de devoirs (*separation of duty*).

Habilitation
Ce lien entre deux rôles et la Structure permet d'introduire une certaine séparation de devoirs statique par activation ou non de deux rôles pour un seul agent dans la même structure. Ainsi, un agent peut être habilité à jouer un autre rôle dans la même organisation selon dans certains contextes et dans certaines situations.

Délégation
Ce concept présenté dans le CIM sécurisé, permet la description des tâches de délégation entre *rôles* et *agents*. Du point de vue de la sécurité, les permissions du délégant relatives aux tâches sont cédées au délégué. Ainsi, en cas d'absence d'un agent, il peut déléguer ses droits (permissions/interdictions) à un agent ou un groupe d'agents (*Rôle*) pour réaliser les tâches qui lui ont été assignées. Ce manque dans le PIM sécurisé pourrait être résolu par réplication des permissions au délégué en utilisant des contraintes appropriées afin de refléter le contexte de délégation.

4. Conclusion

Dans ce livrable, relatif au projet SELKIS, nous prenons en compte l'ensemble des propriétés DICT de la sécurité en combinant une approche dirigée par les buts et un processus d'ingénierie des systèmes d'information fondé sur MDA. Notre hypothèse est la suivante : le fait de prendre en charge les propriétés de sécurité très tôt dans le cycle de vie permet d'obtenir au final un système de meilleure qualité. Ainsi, ce système sera en mesure de répondre parfaitement aux besoins des différents utilisateurs et intègrer les considérations de sécurité pour toutes les parties prenantes dans le système d'information.

L'analyse des systèmes d'information permet d'affirmer que les besoins découlent des objectifs (buts) de l'organisation, des acteurs et de l'environnement dans lequel évolue le système d'information. De plus, les propriétés DICT sont très générales et indépendantes des plateformes et modèles d'implémentation.

Dans ce travail, nous avons réalisé un état de l'art. Cela nous a permis de constater qu'aucune approche ne prend en compte tous les aspects de sécurité. Une grande partie de ces approches est plutôt orientée vers l'analyse des risques de sécurité (malveillances, mauvais usages, etc.) et vers la protection.

[3] $(\text{PIM ROLES} \subseteq \text{STRUCTURE} \times \text{CIM ROLES})$

Nous avons réalisé aussi un état de l'art incluant les taxonomies de la sécurité, les processus de sécurité et, enfin, les méthodes de conception incluant, en partie au moins, la sécurité.

A partir de cette synthèse et d'un tour d'horizon des méta-modèles des différents aspects, nous avons proposé la définition d'une approche de conception intégrant la sécurité (MoSIS) et d'un Métamodèle Intégré des Systèmes d'Information Sécurisés. A l'aide de MoSIS, nous construisons un système d'information sécurisé selon quatre étapes. Chacune d'elles correspond, à un niveau d'abstraction MDA. La première étape comprend la description du système d'information et de son environnement. La seconde étape correspond à l'analyse et la conception abstraite du système d'information. C'est dans cette étape que les besoins fonctionnels et de sécurité sont intégrés et modélisés à l'aide d'un diagramme des cas d'utilisation. La troisième phase est la conception détaillée dans laquelle la plateforme technologique sur laquelle reposera le système d'information a été prise en compte. La dernière phase correspond au codage.
Pour faciliter l'exécution des différentes transformations et/ou raffinements, MoSIS s'appuie sur une base de connaissances qui contient une ontologie des buts et besoins de sécurité génériques ainsi que les différents types de spécification associés, la documentation des applications existantes, les différents types de règles de transformation, ou tout autre source d'information servant le processus.

Notre travail futur portera principalement sur la formalisation des contextes ainsi que sur l'enrichissement des règles de transformation d'un modèle CIM vers un modèle PIM.

Références

[Abou El Kalam, 2003] Anas Abou El Kalam, Rania El Baida, Philippe Balbiani, Salem Benferhat, al Et. *Organization based access control.* Dans : *Policies for Distributed Systems and Networks (POLICY 2003)*, *Como*, *01/01/03-31/12/03*, H. Lutfiyya, J. Moffett, F. Garcia (Eds.), Institute of Electrical and Electronics Engineers, p. 120-131, janvier 2003.

[Agarwal, 2008] Ashish Agarwal Daya Gupta, Security Requirements Elicitation Using View Points for Online System. Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology. Pages: 1238-1243

[ANSI, 2004] ANSI. *American national standard for information technology – role based access control*. ANSI INCITS 359-2004, February 2004 ANSI.

[Anton, 2004] Annie I. Anton, Julia B. Earp. A requirements taxonomy for reducing Web site privacy vulnerabilities. RE journal, vol , 2004

[Basin, 2006]David Basin, Jürgen Doser, Torsten Lodderstedt. Model driven security: From UML models to access control infrastructures TOSEM 15(1), 2006

[Best, 2007] Bastian Best, Jan Jürjens, Bashar Nuseibeh: Model-Based Security Engineering of Distributed Information Systems Using UMLsec. ICSE 2007: 581-590

[Bézivin, 2002], Bézivin, J., Blanc, X. MDA Vers un nouveau paradigme. MDA vers un nouveau paradigme. Journal Développeur Référence, pages 7-11 July 2002

[Booch, 1991] Booch, G., *Object Oriented Analysis and Design with Application*, Benjamin/Cummings, 1991.

[Brambilla et al.,2007a] Brambilla M., Cabot J., Comai S., Automatic Generation of Workflow-Extended Domain Models. MoDELS 2007, LNCS 4735, 2007.

[Brambilla et al.,2007b] Brambilla M., Cabot J., Comai S. Generating Extended Conceptual Schemas from Business Process Models. Technical Report, 2007, Politecnico di Milano.

[Bresciani et al, 2004] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. ”Tropos: An Agent-Oriented Software Development Methodology”. Autonomous Agents and Multi-Agent Systems, 8(3):203–236, 2004

[Breton et al., 2000] Breton E., Bézivin J. An Overview of Industrial Process Meta-models. ICSSEA 2000.

[Breu, 2007] Ruth Breu, Gerhard Popp Muhammad Alam Model based development of access policies. International Journal on Software Tools for Technology Transfer 9 (5), 2007

[Coma et al., 2008] Context Ontology for Secure Interoperability. ARES’ 2008

[Crook, 2005] Crook, R.; Ince, D.; Bashar Nuseibeh. On modelling access policies: relating roles to their organisational context. RE 2005

[Cuppens, 2003] F. Cuppens et A. Miège. Modelling contexts in Or-BAC Model. ACSAC’ 2003

[Curtis et al., 1992] Curtis B., Kellner M.I., Over J. Process Modeling. Communications of the ACM, Vol 35(9), 1992.

[Damianou,2001] Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman: The Ponder Policy Specification Language. POLICY 2001: 18-38

[Debnath et al., 2006a] Debnath N., Riesco D., Montejano G., Cota M.P., Garcia J.B., Romero D., Uva M. Supporting the SPEM with a UML Extended Workflow Metamodel. Proceedings of the IEEE International Conference on Computer Systems and Applications (ICCSA) 2006.

[Debnath et al., 2006b] Debnath N., Riesco D., Montejano G. An ArgoUML metamodel extension for the workflow systems, Journal of Computational Methods in Sciences and Engineering, Volume 6 , Issue 5,6 Supplement 1 (April 2006).

[Doan, 2004] Thuong Doan, Steven A. Demurjian, T. C. Ting, Andreas Ketterl: MAC and UML secure software design. FMSE 2004

[Eder et al., 2002] Eder J., Gruber W. A Meta Model for Structured Workflows Supporting Workflow Transformations, LNCS 2435, Springer-Verlag Berlin Heidelberg, 2002.

[Firesmith, 2003] Firesmith D., « Security Use Cases. », *Journal of Object Technology*, vol. 2, n° 1, p. 53-64, 2003.

[Fontaine, 2001] Fontaine P.J. Goal-oriented elaboration of security requirement. Thèse, 2001

[Georg, 2009] Geri Georg, Indrakshi Ray, Kyriakos Anastasakis, Behzad Bordbar, Manachai Toarcienne, Siv Hilde Houmb: *An aspect-oriented methodology for designing secure applications*. Information & Software Technology 51(5): 846-864 (2009)

[Giorgini et al, 2004] Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone."Requirements Engineering meets Trust Management: Model, Methodology and Reasoning". volume 2995, pages 176–190, 2004.

[Hafner, 2008] *Modeling and Enforcing Advanced Access Control Policies in Healthcare Systems with SectetSource*. Models in Software Engineering: Workshops and Symposia at MoDELS 2007, Nashville, TN, USA, September 30 - October 5, 2007, Reports and Revised Selected Papers, Section: Model-Based Design of Trustworthy Health Information Systems, Pages: 132 – 144, 2008

[Haley, 2008] Charles B. Haley, Robin C. Laney, Jonathan D. Moffett, Bashar Nuseibeh: *Security Requirements Engineering: A Framework for Representation and Analysis*. IEEE Trans. Software Eng. 34(1): 133-153 (2008)

[He, 2003] He, Q., Antón, A.I., *A framework for modelling privacy requirements in role engineering*, Int'l Workshop on Requirements Engineering for Software Quality (REFSQ), 16-17 June, Klagenfurt/Velden, pp.115-124, 2003.

[He, 2009] He Q. and Annie I. Antón. *Requirements-based Access Control Analysis and Policy Specification (ReCAPS)*. Information and Software Technology, Volume 51 , Issue 6 (June 2009)

[Hug, 2009] Hug C., Méta-modèle de processus pour l'ingénierie des systèmes d'information, Thèse de doctorat, Grenoble, octobre 2009

[Jingbai 2009] *An approach to generation of process-oriented requirement spécification* J. Software Engineering & Applications, 2009, 2:13-19

[Jingbai08] : Tian Jingbai, He Keqing, Wang Chong, and Liu Wei. A context awareness non-functional requirements metamodel based on domain ontology. Semantic Computing and Systems, IEEE International Workshop on, 0:1–7, 2008

[Jürjens, 2002] UMLSec: Extending UML for secure systems development, software and systems engineering, J. JURJENS, department of informatics, Munich University of Technologies, Germany, 2002

[Jürjens, 2007] Jan Jürjens, Pasha Shabalin: *Tools for secure systems development with UML*. STTT 9(5-6): 527-544 (2007)

[Jürjens, 2008] Jan Jürjens, Jörg Schreck, Peter Bartmann: *Model-based security analysis for mobile communications*. ICSE 2008: 683-692

[Keqing, 2008] *RGPS: Metamodel for On-Demand Model Selection*

[Kiyavitskaya, 2008] Nadzeya Kiyavitskaya, Nicola Zannone: *Requirements model generation to support requirements elicitation: the Secure Tropos experience*. Autom. Softw. Eng. 15(2): 149-173 (2008)

[Kofod-Petersen, 2006] Kofod-Petersen A., Cassens J. *Using Activity Theory to Model Context Awareness*, Lecture Notes on Artificial Intelligence In Modeling and Retrieval of Context, Vol. 3946 (2006), pp. 1-17.

[Konopacky, 2009] Konopacky P., Frappier M., Laleau R. *Modélisation de politiques de sécurité à l'aide d'une algèbre de processus*, INFORSID, 2009.

[Kradolfer, 2000] Kradolfer M. *A Workflow Metamodel Supporting Dynamic Reuse-Based Model Evolution*. Ph.D Thesis, 2000.

[Lamsweerde, 2009] Axel van Lamsweerde . *Requirements Engineering: From System Goals to UML Models to Software Specifications*. ISBN: 978-0-470-01270-3. 2009

[Lei, 1997] Lei Y., Singh M.P. *A Comparison of Workflow Metamodels*. ER Workshop on Behavioral Modeling (1997).

[List, 2006] List B., Korherr B. *An Evaluation of Conceptual Business Process Modelling Languages*. SAC'06, 2006.

[Lodderstedt et al., 2002] SecureUML: A UML – Based Modeling Language for Model – Driven Security,T. LODDERSTEDT, D. BASIN, J. DOSER, University of Freiburg, germany, 2002

[Lui, 2003] Lin Liu, Eric S. K. Yu, John Mylopoulos: *Security and Privacy Requirements Analysis within a Social Setting*. RE 2003: 151-161

[Martínez, 2009] Alicia Martínez, Oscar Pastor, John Mylopoulos, Hugo Estrada: From Organizational Models to Software Requirements. SEKE 2009: 61-66

[Massacci, 2008] Fabio Massacci, Nicola Zannone: *A Model-Driven Approach for the Specification and Analysis of Access Control Policies*. OTM Conferences (2) 2008: 1087-1103

[Matulevicius, 2008] Raimundas Matulevicius, Nicolas Mayer, Haralambos Mouratidis, Eric Dubois, Patrick Heymans, Nicolas Genon: *Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development*. CAiSE 2008: 541-555

[McDermott *et al.,* 1999] McDermott J., Fox C., « Using Abuse Case Models for Security Requirements Analysis », *15th Annual Computer Security Applications Conference, Phoenix, Arizona, USA, 1999*, 1999.

[Mead, 2005] Mead N.R, Stehney T, Security Quality Requirements Engineering (SQUARE) Methodology, Software Engineering for Secure Systems – Building Trustworthy Applications (SESS'05), 2005.

[Mead, 2008a] Mead, N. R., Viswanathan, V., & Zhan, J. *Incorporating Security Requirements Engineering into the Rational Unified Process* 537–542. 2008 International Conference on Information Security and Assurance (ISA), Busan, Korea, April 26–28, 2008. IEEE Computer Society, 2008.

[Mead, 2008b] Mead, N. R., Viswanathan, V., & Padmanabhan, D. *Incorporating Security Requirements Engineering into the Dynamic Systems Development Method*, 949–954. COMPSAC (International Computer Software and Applications Conference) 2008, IWSSE Workshop (International Workshop on Security and Software Engineering), July 28, 2008, Turku, Finland. IEEE Computer Society, 2008.

[Mellado, 2006] Daniel Mellado, Eduardo Fernández-Medina and Mario Piattini. *Applying a Security Requirements Engineering Process*. ESORICS 2006

[Mellado, 2007] Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini: *A common criteria based security requirements engineering process for the development of secure information systems*. Computer Standards & Interfaces 29(2): 244-253 (2007)

[Miége, 2005] Miège, *Définition d'un environnement formel d'expression de politiques de sécurité. Modéle Or-BAC et extensions*. Thèse 2005

[Morley, 2005] Morley C., Hugues J., Leblanc B., Hugues O. *Processus métiers et S.I. – Evaluation, modélisation, mise en œuvre*, Dunod, 2005.

[Mouratidis, 2007] Mouratidis H., *Secure information systems engineering: a manifesto Source* International Journal of Electronic Security and Digital Forensics Volume 1 , Issue 1. Pages: 27-41, 2007

[Mouratidis, 2009] H. Mouratidis (2009), *Secure Tropos: An Agent Oriented Software Engineering Methodology for the Development of Health and Social Care Information Systems*, International Journal of Computer Science and Security, 3(3), pp. 241-271.

[Mouratidis, 2010] H. Mouratidis, J. Jurjens,*From Goal Driven Security Requirements Engineering to Secure Design*, International Journal of Intelligent Systems, Willey, (accepted for publication November 2009 / to be published 2010).

[OMG, 2006] Business process modeling notation specification. http ://www.bpmn.org/, August 2006.

[OMG, 2008] UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, v1.1.

[OMG, 2009] OMG Unified Modeling Language (OMG UML) Infrastructure, Version 2.2, Février 2009, http://www.omg.org/docs/formal/09-02-04.pdf

[Ouyang, 2007] Ouyang S., Xu H. *A Process Meta Model to Support Policy Based Management in Workflow System*. ICCS 2007.

[Saidani, 2006] *FORBAC : A role based approach for modeling flexible business processes*. The 7th Workshop on Business Process Modelling, Development, and Support (BPMDS'06, (in association with the CAISE'06 Conference), Springer Verlag (pub), Juin 5-6, 2006, Luxembourg.

[Saidani, 2006] Saidani O., Nurcan S. A Role-Based Approach for Modelling Flexible Business Processes, BPMDS'06, 2006.

[Sandhu, 1996], Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. *Role-based access control models*. IEEE Computer, 29(2) :38–47, 1996

[Semmak et al, 2008] Farida Semmak, Christophe Gnaho, and Régine Laleau. "Extended Kaos to Support Variability for Goal Oriented Requirements Reuse". In MoDISEEUS, pages 22–33, 2008

[Sindre et al., 2003] Sindre, G., Firesmith, D.G., and Opdahl, A.L."A Reuse-Based Approach to Determining Security Requirements". 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03) , Austria, 2003

[Sindre *et al.,* 2000] Sindre G., Opdahl A. L., « Eliciting Security Requirements by Misuse Cases », *TOOLS (37)*, IEEE Computer Society, p. 120-131, 2000.

[Sindre, 2005] Guttorm Sindre, Andreas L. Opdahl: *Eliciting security requirements with misuse cases*. Requir. Eng. 10(1): 34-44 (2005)

[SOF, 2009] Description du méta – modèle de l'outil Objecteering de la société SofTeam, http://support.objecteering.com/objecteering6.1/help/us/metamodel/toc.html

[Susi et al, 2005] Angelo Susi, Anna Perini, John Mylopoulos, and Paolo Giorgini. "The Tropos Metamodel and its Use". Informatica, 29:401–408, 2005.

[Susi, 2005] : Angelo Susi, Anna Perini, John Mylopoulos, and Paolo Giorgini. "The Tropos Metamodel and its Use". Informatica, 29:401–408, 2005.

[Wang07] : Jian Wang, Keqing He, Bing Li, Wei Liu, and Rong Peng. *Meta-models of domain modeling framework for networked software*. In GCC ’07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing, pages 878–886, Washington, DC, USA, 2007. IEEE Computer Society

[Wang07] : Jian Wang, Keqing He, Bing Li, Wei Liu, and Rong Peng. Meta-models of domain modeling framework for networked software. In GCC ’07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing, pages 878–886, Washington, DC, USA, 2007. IEEE Computer Society

[Yu, 1999] E. Yu *Strategic Modelling for Enterprise Integration*. Proceedings of the 14th World Congress of International Federation of Automatic Control (IFAC’99), July 5-9, 1999, Beijing, China. pp. 127-132. Permagon, Elsevier Science

[Yu, 95] Eric Siu-Kwong Yu. “Modelling strategic relationships for process reengineering”. PhD thesis, Toronto, Ont., Canada, Canada, 1995. Adviser-Mylopoulos, John.

[Yu06] : E. Yu, M. Strohmaier, X. Deng Exploring Intentional Modeling and Analysis for Enterprise Architecture Proceedings of the Workshop on Trends in Enterprise Architecture Research (TEAR'06), at the Enterprise Computing Conference (EDOC) Hong Kong, October 2006

[Zannone, 2008] The SI* Modeling Framework: Metamodel and Applications. –IJSEKE journal. April 13, 2008 12:10

Annexe 1 : Description du modèle R-Bac

Le contrôle d'accès à base de rôles (RBAC) vise à protéger les ressources d'un système par la mise en œuvre de politiques de sécurité centrées sur le concept de rôle.
La version standardisée de l'ANSI [ANSI, 2004] fournit un modèle de référence qui définit l'ensemble minimale de concepts (utilisateur, rôle, session, permission, opération et objet) pour tout système RBAC ainsi que les relations entre ces concepts.

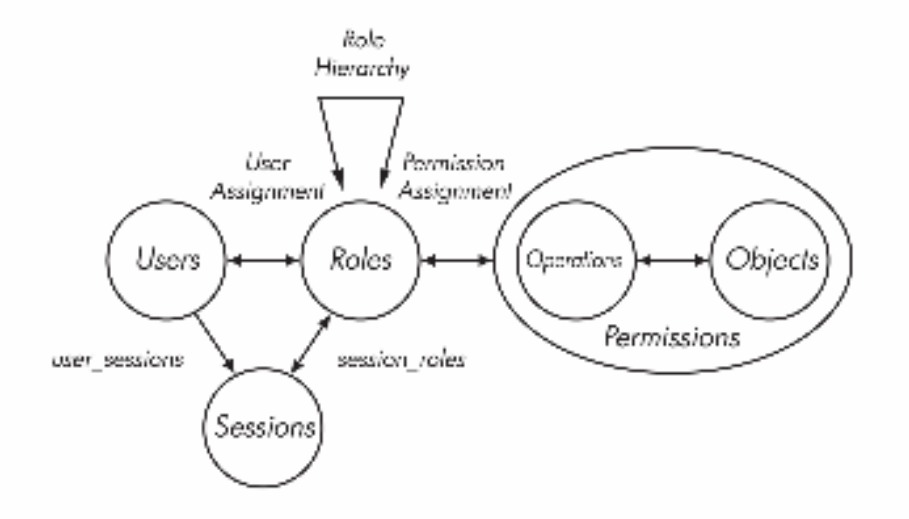

Figure 44. Schéma illustratif de RBAC (ANSI-RBAC)

La figure 44 schématise les entités et les relations du modèle que nous proposons de détailler brièvement.

- ***Users*** : ce sont les agents humains ou logiciels qui peuvent accéder aux ressources du système. Ils sont généralement assimilés aux utilisateurs humains dont les actions au sein du système restent toujours imprévisibles, d'où le besoin de contrôle.
- ***Objects*** : les objets représentent les ressources du système susceptibles de faire l'objet d'accès et qui ont besoin d'être protégées. Ces ressources peuvent être des fichiers, des enregistrements de bases de données, des imprimantes ou toute autre entité «contenant ou recevant de l'information» [ANSI, 2004].
- ***Operations*** : les opérations représentent les types d'accès (actions) qu'un utilisateur du système peut invoquer sur une ressource. Par opération, on entend toute action du système qui récupère ou communique des informations à un objet.
- ***Permissions***: une permission représente un privilège dans le système autorisant des opérations sur des objets protégés. En fait une permission remplit une fonction structurante qui consiste à faire le lien entre des opérations et des objets pour modéliser ainsi des activités au sein du système.
- ***Roles*** : représentant, généralement, une fonction ou une autorité au sein d'une organisation, un rôle RBAC permet de regrouper les privilèges (permissions) liées à cette fonction et d'en faire jouir les utilisateurs disposant de ce rôle. Le concept de rôle est central dans RBAC car il permet la construction de la politique de contrôle d'accès en faisant le pont entre les utilisateurs et les permissions. En effet, la relation ***PermissionAssignment*** permet d'accorder des permissions aux rôles et ces rôles sont

attribués, au moyen de la relation ***UserAssignment***, aux utilisateurs qui acquerront les privilèges du rôle.

- ***Sessions*** : cet élément représente les sessions de travail des utilisateurs. Au cours de ces sessions, un utilisateur est associé à un sous-ensemble des rôles auxquels il a droit.

Le modèle RBAC est largement utilisé comme noyau pour mettre en œuvre des mécanismes de contrôle d'accès. Toutefois, la complexité croissante des systèmes d'information et des politiques de sécurité, mises en œuvre, a conduit le modèle RBAC à s'enrichir au fur et mesure avec des concepts plus avancés. D'ailleurs, le standard *ANSI* définit 4 variantes de RBAC. La première, appelée « *RBAC0* » par Sandhu et al, dans [Sandhu, 1996], représente le noyau de RBAC avec les concepts introduits précédemment. La deuxième variante, appelée « *RBAC hiérarchique* » ou « *RBAC1* », introduit la notion d'héritage de rôle. Cette dernière permet de simplifier la définition de rôles en permettant l'héritage des permissions assignées aux rôles. Les deux dernières variantes permettent d'introduire les notions de séparation de devoirs, respectivement statique et dynamique, afin d'éviter le cumul de certains rôles (conflictuels).

Annexe 2 : Les méta-modèles de profil UML

La description d'un profil passe le plus souvent par 3 points essentiels :

- *Le domaine du Profil* **:** les profils permettent d'adapter le langage de modélisation UML à un domaine particulier, le domaine de la sécurité des Systèmes d'Information dans les organismes de santé que nous étudions. La description du domaine peut se faire en définissant le méta – modèle du domaine. Ce méta – modèle devra décrire les concepts de notre domaine en question ainsi que les différentes relations existant entre ses concepts. Si un tel méta – modèle existe et qu'il est défini, la description du profil pourra alors le référencer.
- *La description technique du Profil* **:** techniquement, un profil n'est qu'un ensemble de stéréotypes, de TaggedValues et de Contraintes. Ces éléments permettent d'établir une correspondance entre les concepts prédéfinis d'UML (packages, classes, associations, etc) et les nouveaux concepts du domaine de sécurité dans des organisations de santé que nous allons représenter par un profil. Il est important de noter ici qu'en principe seule cette partie suffit pour définir un profil selon un standard de modélisation bien déterminé.
- *La description de la partie opérationnelle du Profil* **:** le profil ne peut se résumer uniquement par une liste de stéréotypes, TaggedValues et des Contraintes. C'est aussi et surtout un ensemble de règles qui permettent de rendre un profil *opérationnel*. Par exemple, nous pouvons, à partir de cette description, définir des règles qui permettent de générer automatiquement le code dans une plateforme donnée, générer aussi des fichiers de déploiement utiles à la mise en œuvre du système. Cette partie n'est pas souvent clairement identifiée dans la définition classique du profil UML, elle est pourtant fondamentale.

Dans le cadre de la modélisation de profils, la littérature offre deux méta-modèles de profils : celui de l'OMG [OMG, 2009] et celui de la société SoftTeam. Cette annexe décrit ces deux méta-modèles.

a) Méta-modèle de profil de l'OMG

Un profil doit être en conformité avec un méta – modèle. Ainsi, le méta-modèle de profil de l'OMG est relatif au méta – modèle UML. De ce fait, plusieurs paquetages et notions ont été importés dans le profil à partir du méta – modèle de référence afin de définir des extensions au domaine étudié et qui seront applicables à notre domaine médical.

Il convient ainsi de fournir un aperçu sur l'infrastructure du méta – modèle UML pour comprendre les mécanismes d'imports de construits.

L'infrastructure d'UML est définie par une structure appelée « InfrastructureLibrary », qui satisfait certains besoins de conception qui sont :

- La définition d'un méta – langage « Core » qui pourrait être réutilisable pour définir une variété de méta – modèles incluant à la fois UML, MOF et CWM,
- L'alignement des architectures UML, MOF et XMI de manière à supporter totalement l'échange de modèles et,

– L'autorisation de la personnalisation d'UML à travers les Profils et la création de nouvelles familles de langages basés sur le même méta – langage « Core » qu'UML.

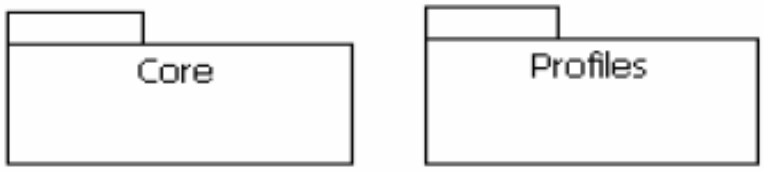

Figure 45. Les paquetages formant l'InfrastructureLibrary

L'«InfrastructureLibrary» est représenté par un paquetage qui renferme les deux paquetages « Core » et « Profiles ». Il définit, d'une part, un méta – langage réutilisable de conception de méta – modèles et d'autre part, il fournit un mécanisme d'extension permettant de rajouter des profils et de personnaliser UML à différentes plateformes et domaines sans avoir à recourir à des méta – modèles complets et prédéfinis.

Le paquetage « Core » est la partie centrale réutilisable de « InfrastructureLibrary ». Il est subdivisé en quatre sous – paquetages (Figure 46).

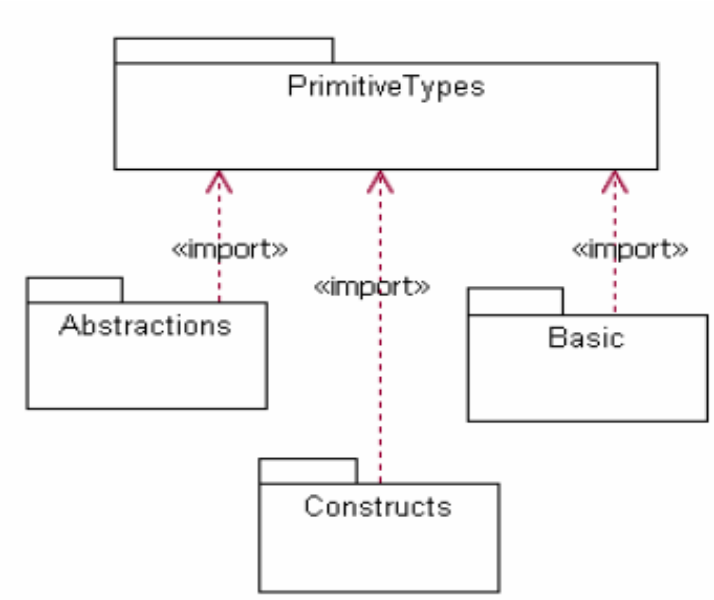

Figure 46. Les sous – paquetages formant le paquetage central « Core »

– PrimitiveTypes : est un paquetage de InfrastructureLibrary : Core qui contient un certain nombre de types prédéfinis utilisés pour définir la syntaxe abstraite des méta – modèles ; il renferme des classes abstraites à savoir INTEGER, BOOLEAN, STRING, etc,

– Abstractions : ce paquetage est divisé en un nombre de paquetages à un niveau de granularité plus fin permettant ainsi une réutilisation plus flexible lors de la création des méta – modèles, parmi les sous – paquetages qu'il contient, nous trouvons OwnerShips, Multiplicities, Constraints, Generalizations, etc,

– Constructs : ce paquetage dépend de plusieurs autres paquetages notamment ceux du Basic et Abstractions ; il fusionne plusieurs sous – paquetages à savoir RelationShips, StructuralFeatures, OwnerShips, Multiplicities, etc,

– Basic : il contient un sous – ensemble de paquetages de Constructs et est utilisé principalement pour des raisons d'échange de modèles en manipulant XMI.

L'« InfrastructureLibrary » d'UML contient aussi le paquetage « Profiles », qui est très important dans le sens où il contient des mécanismes qui permettent d'étendre les méta – classes des méta – modèles existants et ainsi de les enrichir et de les adapter à différents contextes.

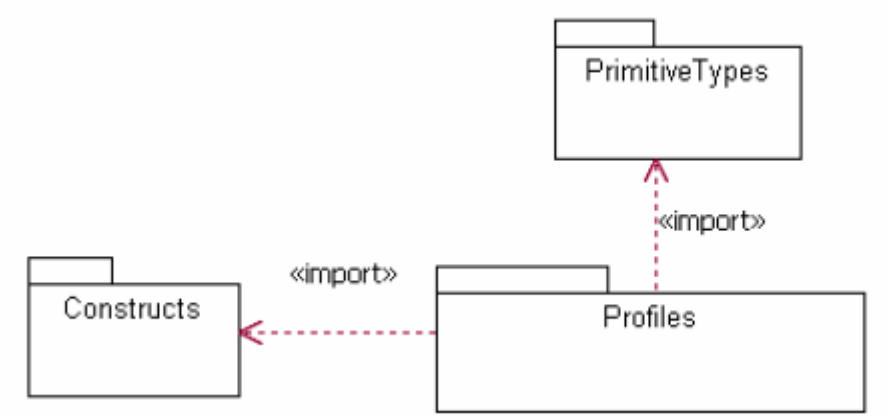

Figure 47. La dépendance du paquetage « Profile » vis-à-vis des sous-paquetages du « Core »

La figure 47 montre les dépendances qui existent entre le paquetage Profile et certains autres paquetages correspondants au Core. Ces dépendances, à travers un mécanisme d'import de concepts, permettent de récupérer des éléments UML à partir d'autres paquetages afin d'arriver finalement à construire un profil adapté au domaine d'étude retenu ou à une plate – forme de développement spécifique.

Le méta-modèle de profil proposé par l'OMG [OMG, 2009] est présenté ci – après avec une description détaillée de ses éléments constitutifs et de leurs interdépendances.

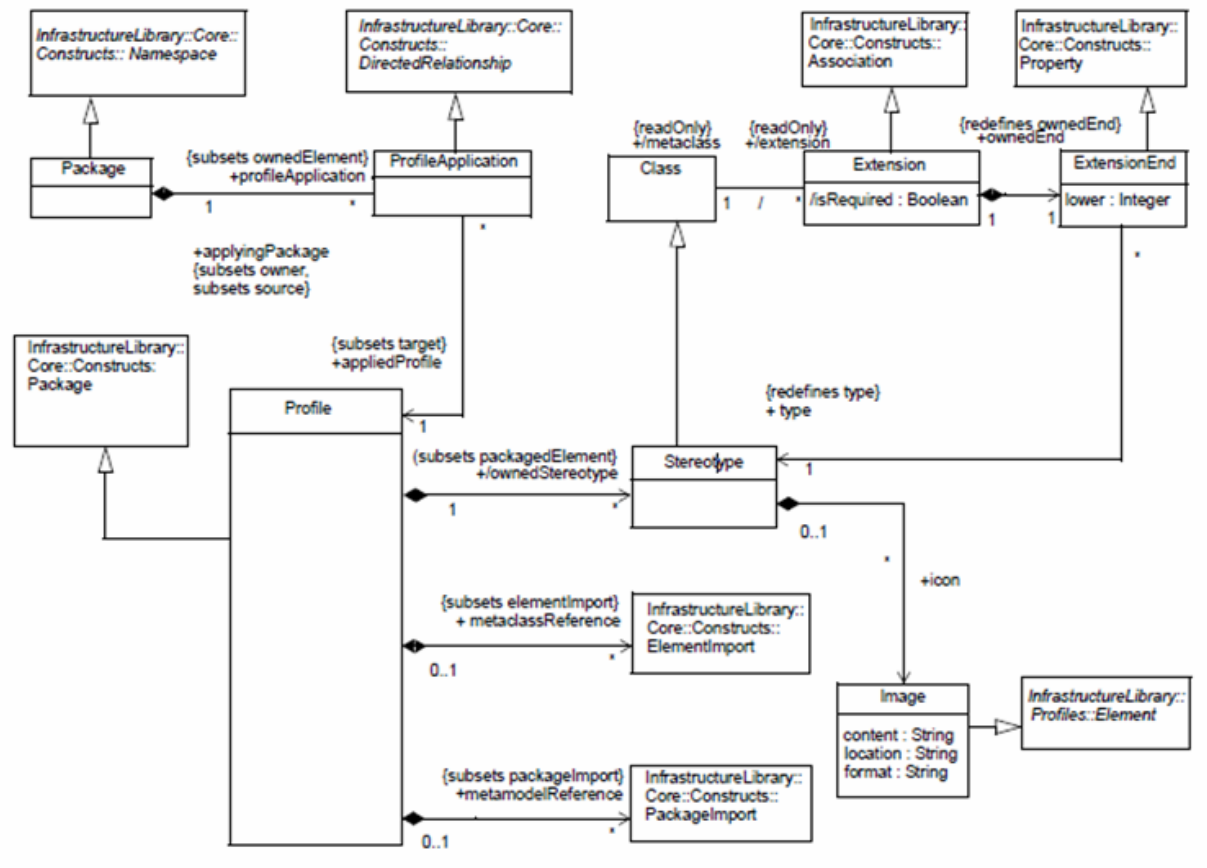

Figure 48. Méta-modèle de profil de l'OMG

A partir de la figure 48, nous pouvons décrire certains éléments qui rentrent dans la description d'un profil :

- La description du domaine : matérialisée par les méta-classes Profile qui étend un méta-modèle déjà existant et qui fait référence à un domaine d'étude bien déterminé à savoir, dans le cadre du projet SELKIS, la définition d'un Système d'Information Sécurisé, Package qui est une spécialisation du paquetage NameSpace (contenant l'identification des tous les éléments UML) et qui peut avoir un ou plusieurs ProfileApplication permettant d'indiquer quels profils sont appliqués sur ce Package et ProfileApplication qui relie ces deux méta- classes et rajoute la possibilité de spécifier qu'un profil donné est appliqué sur un package donné.

- La description technique : dans cette partie du profil, nous trouvons les méta-classes Class, Extension, ExtensionEnd et Stereotype. Une classe (Class) est un élément de modélisation UML. C'est un de la méta-classe Stereotype. Une classe peut avoir une ou plusieurs Extensions. Nous remarquons ici que le seul mécanisme d'extension possible dans le profil UML proposé par l'OMG est le stéréotype.

- L'identification des sources permettant l'import et l'utilisation des éléments UML à partir d'autres paquetages.

b) Méta-modèle de profil de SofTeam

Dans ce méta-modèle de profil [SOF], nous pouvons repérer les trois composants différents nécessaires à la description d'un profil :

- La description du domaine: comportant les méta-classes ProfileDomain, Package, ItemDictionary, ProfileInterface, ProfileImplementation avec tous les liens possibles entre elles ; la méta – classe ProfileDomain permet de spécifier un contexte bien défini, dans notre cas la conception d'un système d'information médical sécurisé. ProfileInterface, quant à elle, permet de structurer les extensions UML. Enfin, ProfileImplementation permet de spécifier les paquetages dédiés pour implémenter un profil ou un méta – modèle, il peut être créé en Java par exemple,

- La description technique du profil : qui comporte la méta- classe générique ElementUML ainsi que UMLExtension, UMLModelType, et les méta-classes stéréotypées Stereotype, TypeOfNote et TagType. UMLExtension est une généralisation utilisée pour définir de nouveaux concepts. Ces concepts sont structurés autour du ProfileInterface et regroupent des stéréotypes, des contraintes (TypeOfNote) et des valeurs marquées (TagType),

- La description opérationnelle du profil : qui fait intervenir un certain nombre de méta-classes à savoir ProfileOperation, ProfileRule, TransformationRule, ValidationRule, PresentationRule, PredefinedView et ProfileWorkProduct (représente des éléments externes générés par des composants et qui sont typiquement des fichiers sources, de la documentation, etc). On trouve plusieurs types de règles dans cette partie de description d'un profil dont notamment :

    - *Les règles de présentation* : qui permettent la manipulation des aspects graphiques d'un modèle. Deux types de règles de présentation sont identifiés à ce jour : 1) le filtrage permettant de masquer ou non certains éléments graphiques d'un modèle et 2) les vues prédéfinies qui identifient précisément un diagramme particulier (et non pas un type de diagramme). Ces vues sont utilisées pour faciliter la communication entre les membres d'une équipe de travail. Elles permettent aussi la construction automatique du diagramme correspondant à la vue.

    - *Les règles de validation* : permettent de vérifier la cohérence d'un modèle vis-à-vis d'un profil donné (par exemple la vérification de la présence des multiplicités dans les diagrammes de classes UML, de la présence des intervenants ou acteurs dans les diagrammes de Cas d'Utilisation UML). Ainsi, les règles de validation regroupent l'ensemble des contraintes à appliquer sur un modèle dans le cadre d'un profil donné,

    - *Les règles de transformation* : peuvent être de plusieurs types ; des règles de génération de code exécutable à partir de modèles, de transformation de code d'une plate-forme à une autre, etc. L'élaboration des règles de transformation se fait en comparant les concepts du profil source avec ceux du profil cible. L'automatisation du développement peut être réalisée à travers le développement de *work products*, des règles de génération de code et aussi des patrons de conception ou *design patterns*.

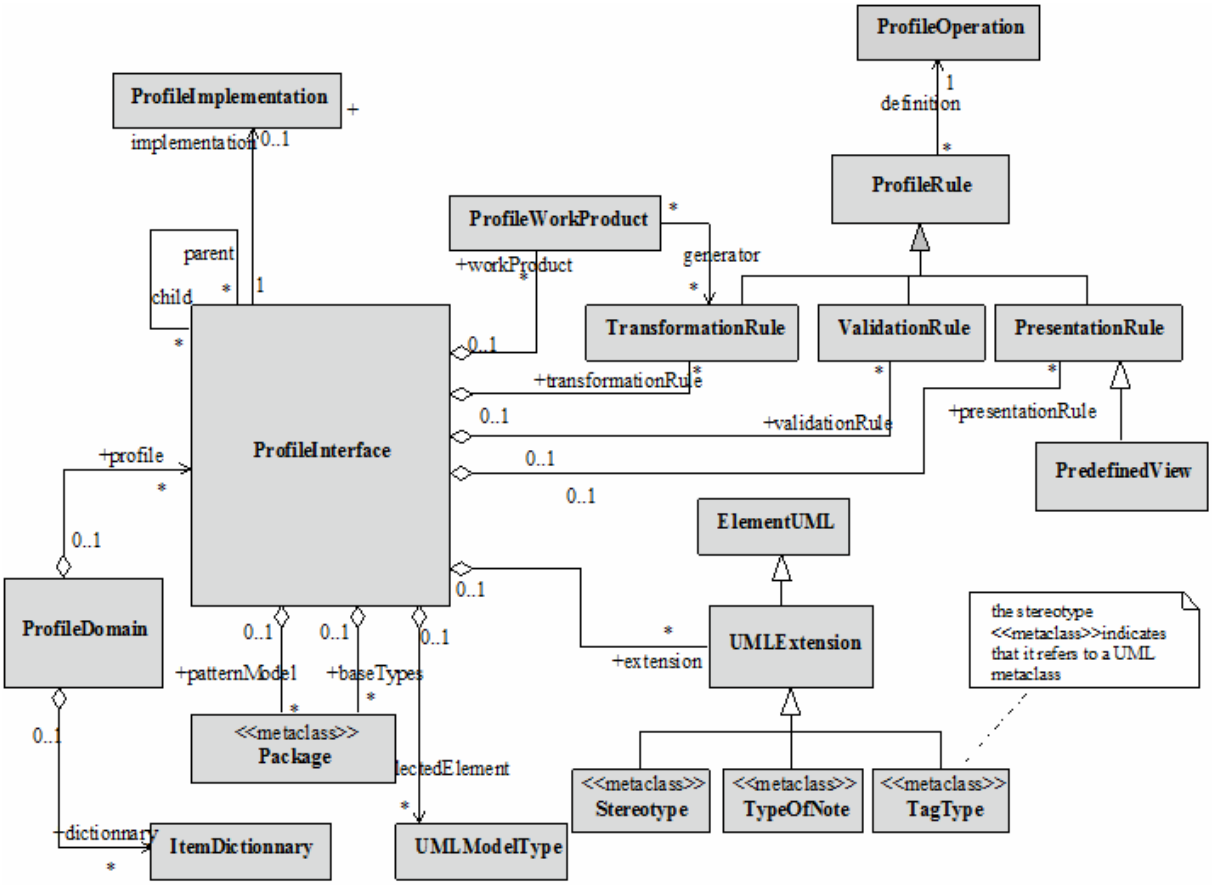

Figure 49. Méta-modèle de profil de SofTeam correspondant au méta-modèle de l'outil Objecteering

Annexe 3 : Notre méta-modèle et le glossaire

Méta-modèle intégré

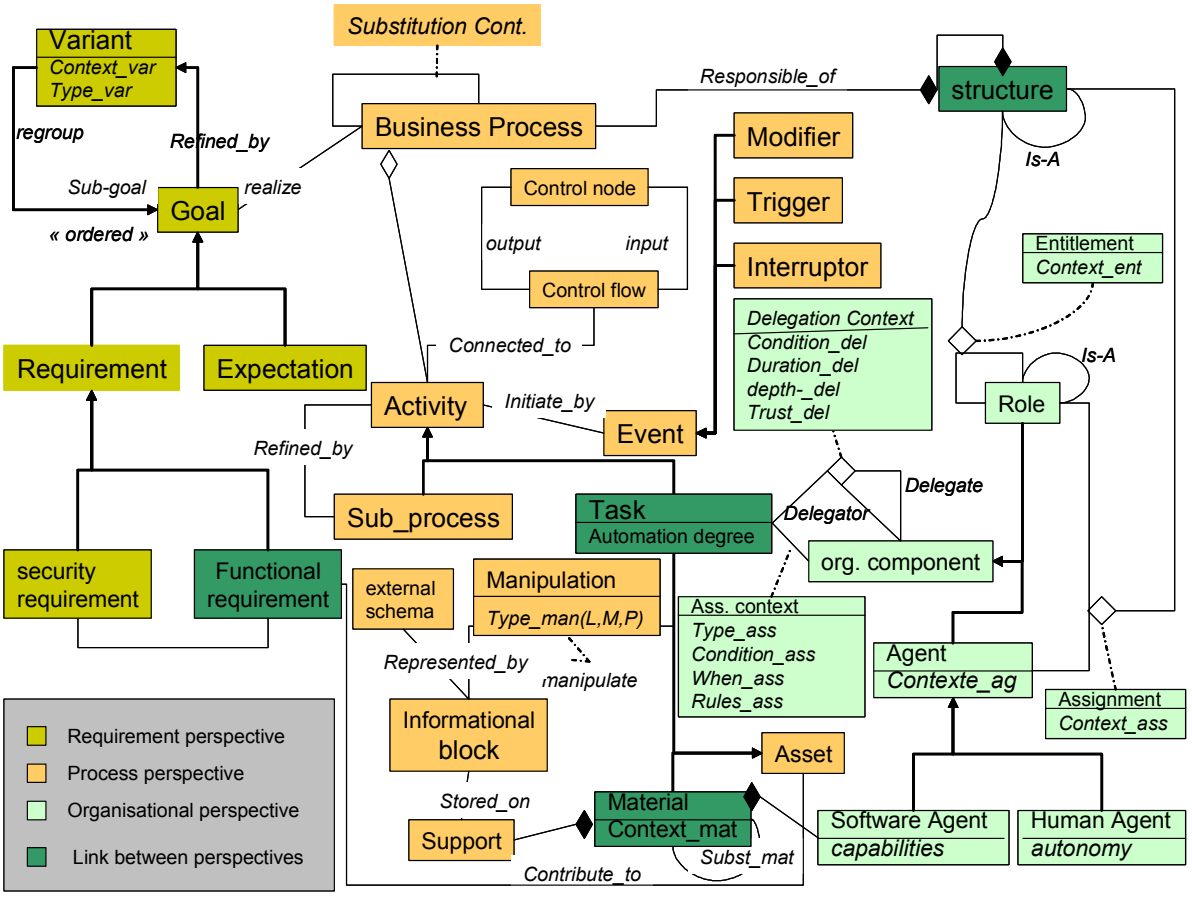

Figure 50. Méta-modèle intégré

Concepts	Expression	Définition ou description
Organisation	*Structure*	Entité regroupant le système d'information, le matériel et les agents jouant un rôle dans l'exécution d'une tâche
Composant organisationnel	*Organizational component*	Agent ou rôle ayant des attributions pour l'exécution d'une tâche
Délégant	*Delegator*	Agent/rôle cédant temporairement ses attributions à un autre agent/rôle
Délégué	*Delegate*	Agent/rôle ayant reçu des attributions pour réaliser des tâches
Contexte de délégation	*Delegation context*	Situation dans laquelle une délégation est réalisée
Condition de délégation	*Condition_del*	Condition dans laquelle une délégation est faite (par exemple : absence, etc.)
Durée de délégation	*Duration_del*	Période dans laquelle la délégation est valable
Profondeur	*Depth_del*	Niveau limite de la délégation dans la hiérarchie
Confiance de délégation	*Trust_del*	Le niveau de confiance que le délégant a envers son délégué
Agent	*Agent*	Acteur qui joue un rôle dans la réalisation d'une tâche
Contexte de l'agent	*Context_ag*	Indique par exemple la disponibilité de l'agent
Rôle	*Role*	Entité abstraite dans l'organisation ayant des attributions
Affectation	*Assignment*	Associer un agent à un rôle dans l'organisation
Contexte d'affectation	*Context_ass*	Description de la situation d'affectation (exemple : planification, mission, etc.)
Habilitations	*Entitlement*	Autorisation d'exercer un rôle
Contexte d'habilitation	*Context_ent*	Situation d'habilitation (par exemple : urgence, contrat d'embauche, etc.)
Contexte d'attribution	*Assignation context*	Précision des droits d'accès et d'usage
Type d'attribution	*Type_ass*	Indique les permissions (+) ou les interdictions (-)
Conditions d'attributions	*Condition_ass*	Donne l'état du système ou de l'environnement où les attributions sont valides
Quand	*When_ass*	Précise la situation d'application des attributions
Règles	*Rules_ass*	Vérification à faire sur les attributions
Tâche	*Task*	Activité élémentaire
Degré d'automatisation	*Automation degree*	Permet de distinguer le niveau d'automatisation de la tâche
Agent humain	*Human agent*	Personnes exécutant des tâches dans l'organisation
Aptitudes	*Capabilities*	Compétences d'un agent humain
Agent logiciel	*Software agent*	Application pouvant réaliser certaines tâches
Autonomie	*Autonomy*	Indique si l'application nécessite une interaction avec un humain pour son fonctionnement
Matériel	*Material*	Indique le matériel informatique qui héberge les agents logiciels
Contexte	*Context_mat*	Indique l'état du matériel ou sa disponibilité

matériel		
Besoin fonctionnel	*Functional requirement*	But métier élémentaire
Besoin de sécurité	*Security requirement*	But de sécurité élémentaire
contribue	*contribute*	Représente l'apport des besoins de sécurité aux besoins fonctionnels
Besoins	*Requirement*	Propriétés mesurables qui doivent être assurées par le système
Attentes	*Expectation*	Propriétés non mesurables qui doivent être satisfaites par le système
But	*Goal*	Objectif organisationnel permettant de définir les besoins et les attentes
Raffiné par	*Refined_by*	Lien de décomposition d'un but
Variante	*Variant*	Une manière de réaliser un but
Contexte de variation	*Context_var*	Situation qui permet de distinguer les variantes
Type de variantes	*Type_var*	Indique le type de raffinement de buts (par exemple : *milestone, etc.*)
Regroupe	*regroup*	Lien indiquant l'obtention de 2 ou plusieurs sous-buts obtenus par raffinement
Ordonné	*Ordered*	Précise s'il existe une contrainte d'ordre des sous buts obtenus
Sous buts	*Sub goal*	But moins complexe obtenu par raffinement
Contexte de substitution	*Substitution context*	Situation dans laquelle un processus peut remplacer un autre
Processus métier	*Business process*	Description des règles et de l'enchaînement des tâches dans une organisation
Responsable de	*'responsible_of'*	Indique le responsable du processus métier
Réalise	*'realize'*	Indique que le processus permet de satisfaire un but
Activité	*Activity*	Ensemble de tâches et de processus dans l'organisation
Connecté à	*'connected_to'*	Indique la connexion entre activités
Flux de contrôle	*Control flow*	Permet d'enchaîner des activités
Entrant	*'input'*	Flux fourni en entrée d'un processus
Sortant	*'output'*	Flux fourni en sortie d'un processus
Nœud de contrôle	*Control node*	Nœuds permettant d'agencer et de synchroniser un ensemble d'activités
Initié par	*'initiated_by'*	Indique l'initiateur de l'activité
Evénement	*Event*	C'est l'initiateur d'une activité
Modifieur	*Modifier*	Evénement chargé de gérer les changements au cours d'une activité
Déclencheur	*Trigger*	Evénement initiateur d'une activité
Interrupteur	*Interrupter*	Evénement chargé d'arrêter une activité
Sous processus	*Sub_process*	Activité complexe qui se raffine pour obtenir des tâches
Raffiné	*'refined'*	Décomposé en activités élémentaires (tâches)
Ressource	*Asset*	Ensemble de matériels et de tâches contribuant à la satisfaction d'un besoin fonctionnel
Contribue à	*'contribute_to'*	Indique l'apport de la ressource à la satisfaction d'un besoin fonctionnel

Manipule	*'manipulate'*	Indique les actions de manipulation des données
Manipulation	*Manipulation*	Opérations de manipulation (lecture, modification, etc.)
Type de manipulation	*Type_man*	Type d'opération de manipulation (lecture par exemple)
Lot informationnel	*Informational block*	Bloc de données sur lequel s'effectuent les manipulations
Stocké sur	*'stored_on'*	Indique le lieu de stockage
Support	*Support*	Document ou espace sur lequel sont stockés les lots informationnels
Substitution matérielle	*Substitution_mat*	Désigne la possibilité qu'un matériel soit remplacé par un autre pour assurer le fonctionnement
Représenté par	*'represented_by'*	Indique la représentation des lots informationnels
Schéma externe	*External schema*	Fichier décrivant la structure des lots informationnels

Annexe 4 : Cas bibliothèque

Présentation générale

Comme cas d'étude, on considère le système de gestion des bibliothèques d'une université. Dans notre exemple, la bibliothèque est répartie sur deux sites distincts :
1. la bibliothèque des sciences exactes ;
2. la bibliothèque des sciences humaines.
La bibliothèque des sciences exactes est rattachée à la faculté des sciences, tandis que celle des sciences humaines est rattachée à la faculté de droit. Chacune de ces facultés comporte plusieurs départements. L'université possède aussi un système d'information pour le service financier.
L'objectif de ce cas d'étude est de modéliser le système de gestion des bibliothèques, indépendamment des systèmes de gestion des facultés et du service financier. Toutefois, le système de gestion des bibliothèques pourra avoir des accès aux données contenues dans les autres systèmes d'information.

Utilisateurs, ressources, actions

Les rôles des utilisateurs du système de gestion des bibliothèques sont les suivants :
– Enseignants-chercheurs : les enseignants-chercheurs des facultés peuvent être classifiés en deux catégories, les permanents et les non permanents ;
– Etudiants : parmi ces utilisateurs, on distingue les étudiants au doctorat des autres étudiants ;
– Bibliothécaires : il est important de noter que certains bibliothécaires peuvent être des étudiants de l'université engagés sur un emploi étudiant à travers un contrat d'embauche;
– Bibliothécaires en chef : chaque bibliothèque de l'université est dirigée par un bibliothécaire en chef, qui ne peut pas être un étudiant ;
– Doyens des facultés.
Dans cet exemple de bibliothèque, les ressources à emprunter ne sont constituées que de livres. Il en existe de deux types : les livres réguliers et les livres précieux.
Les actions qui peuvent être réalisées au sein d'une bibliothèque sont les suivantes : achat et suppression d'un livre de la bibliothèque, inscription et résiliation d'un membre, emprunt, réservation et retour d'un livre.
L'achat et la suppression de livres doivent être obligatoirement approuvés. De plus, tout achat de livres peut être annulé selon certaines conditions. Les réservations et les emprunts de livres peuvent être modifiés ou annulés.
Les principales actions du système sont réalisées sur place, par l'intermédiaire d'un bibliothécaire. Les membres de la bibliothèque peuvent toutefois consulter sur Internet l'état courant de leur compte en termes d'emprunts et de réservations, et ils peuvent aussi interagir directement sur le site web « GesBiblio » pour réserver à distance un livre qui est déjà emprunté.

Règles de fonctionnement

Le type de propriété à modéliser est indiqué en caractère gras.

1. Seule une personne ayant le rôle de bibliothécaire peut réaliser des actions directement sur le système. Par exemple, lorsqu'un membre emprunte un livre, l'emprunt est enregistré par le bibliothécaire au sein du système. (**Permissions**)

2. Les emprunts de livres sont limités dans le temps. Dans la bibliothèque des sciences exactes, l'emprunt dure au maximum 3 semaines pour les étudiants, 5 semaines pour les étudiants au doctorat et 7 semaines pour les enseignants-chercheurs permanents. Dans la bibliothèque des sciences humaines, il peut durer 2 semaines pour un étudiant, 4 semaines pour un doctorant et 6 semaines pour les enseignants-chercheurs permanents. Dans tous les cas, pour les personnels non-permanents, la durée de l'emprunt ne doit pas excéder la fin du contrat de travail ou la fin de l'année universitaire. (**Contraintes temporelles**)
3. Toute personne s'inscrivant dans une des bibliothèques est automatiquement membre de l'autre. Il peut alors emprunter des livres dans l'une ou l'autre. (**Permissions**)
4. Tout livre emprunté dans une bibliothèque doit être retourné dans la même bibliothèque. (**Contrainte d'intégrité**)
5. La demande d'achat de livres doit être effectuée auprès de la bibliothèque par un enseignant-chercheur. La bibliothèque doit obtenir l'avis des enseignants-chercheurs potentiels ayant le droit de donner des approbations pour l'achat des livres.
6. Dans la bibliothèque des sciences humaines, l'achat d'un livre doit obligatoirement être approuvé par un enseignant-chercheur de la faculté de droit différent de celui qui a fait la demande d'achat. (**Workflow sécurisé**)
7. Dans la bibliothèque des sciences exactes, l'achat d'un livre doit être approuvé par deux enseignants-chercheurs distincts et différents de celui qui a fait la demande, dont le doyen de la faculté des sciences. En cas d'absence, le doyen peut déléguer son pouvoir d'approbation à un enseignant-chercheur de sa faculté. Dans tous les cas, les deux personnes qui approuvent l'achat d'un livre doivent être distinctes. (**Workflow sécurisé + délégation**)
8. En cas d'annulation de la demande faute d'avis favorables, d'enregistrement de la commande ou de son annulation dans les délais pour erreur de saisie ou autre raison, le demandeur doit être informé.
9. En cas de délégation du doyen de la faculté des sciences, on doit être capable de retrouver les personnes qui ont autorisé les achats. (**Traçabilité**)
10. Si l'achat d'un livre dépasse un certain montant fixé par le service financier, il faut demander l'approbation de ce dernier avant de pouvoir acheter ce livre. (**Contrainte sur un attribut**)
11. Dans le cas d'une erreur de manipulation lors de la saisie d'un achat de livres, seul le bibliothécaire en chef a le droit d'annuler la commande, dans un délai d'une heure après la saisie. **(Contrainte temporelle + contexte particulier**)
12. Certains livres de la bibliothèque, qui sont considérés comme précieux, ne peuvent être empruntés que par des chercheurs de la faculté à laquelle est rattachée la bibliothèque dans laquelle se trouvent ces livres. Les chercheurs sont soit des enseignants-chercheurs, soit des étudiants en doctorat. (**Permissions**)
13. Un livre qui n'est pas emprunté doit nécessairement se trouver sur le site de la bibliothèque à laquelle il appartient. (**Contrainte d'intégrité**)
14. Sur le site web « GesBiblio », un membre authentifié peut consulter uniquement ses propres emprunts et réservations. Il a la possibilité de réserver un livre déjà emprunté et de modifier ses réservations. (**Confidentialité + permissions**)
15. Dans le cadre d'un emploi étudiant, ce dernier peut travailler en tant que bibliothécaire. Dans ce cas, l'étudiant ne peut pas gérer son propre dossier. En particulier, il n'a pas le droit d'emprunter directement des livres ; il doit passer par un autre bibliothécaire. (**Séparation des devoirs**)

16. De manière générale, toute action du système doit être atteignable. Autrement dit, pour toute action, il existe un utilisateur ayant les permissions à un certain moment pour exécuter cette action. (**Atteignabilité ou faisabilité des actions**)
17. Chaque utilisateur doit pouvoir exécuter à certains moments les actions pour lesquelles il a les permissions. (**Disponibilité des actions**)

Annexe 5 : Instanciation complète du méta-modèle CIM

Instance du méta-modèle des besoins

Comme représenté dans notre méta-modèle, un but se raffine par spécification des différentes variantes de sa réalisation. Le raffinement des buts continue jusqu'à l'obtention des buts dont il existe une ou un ensemble de tâches capables de le réaliser. Ces buts représentent les besoins du système d'information.
Pour le cas de la bibliothèque, nous identifions une variante non ordonnée où le but « gestion bibliothèque » est affiné à travers trois buts « gestion livre », « gestion membre » et « gestion usage ». Dans cette partie, nous nous limitons au but « gestion livre ». Ce but se réalise à travers les buts de « gestion achat » et « gestion emprunt ». Le but premier but est détaillé dans l'exemple par raffinement en deux variantes ordonnées, selon les approbations ou désapprobations. Les buts obligatoires sont communs aux deux variantes mais se distinguent par le but « annulation demande » pour manque d'approbation. Dans le cas contraire, le but « gestion commande » donne lieu à l'enregistrement de la commande ou à l'annulation de cette commande en cas d'erreur.
Le raffinement est arrêté lorsque le but obtenu représente un besoin du système qui peut être réalisé par une ou un ensemble de tâches. Dans ce cas de la bibliothèque, « gestion demande d'achat », gestion demande d'approbation », « gestion approbation d'achat », etc. représentent des buts finaux correspondant aux besoins du système.

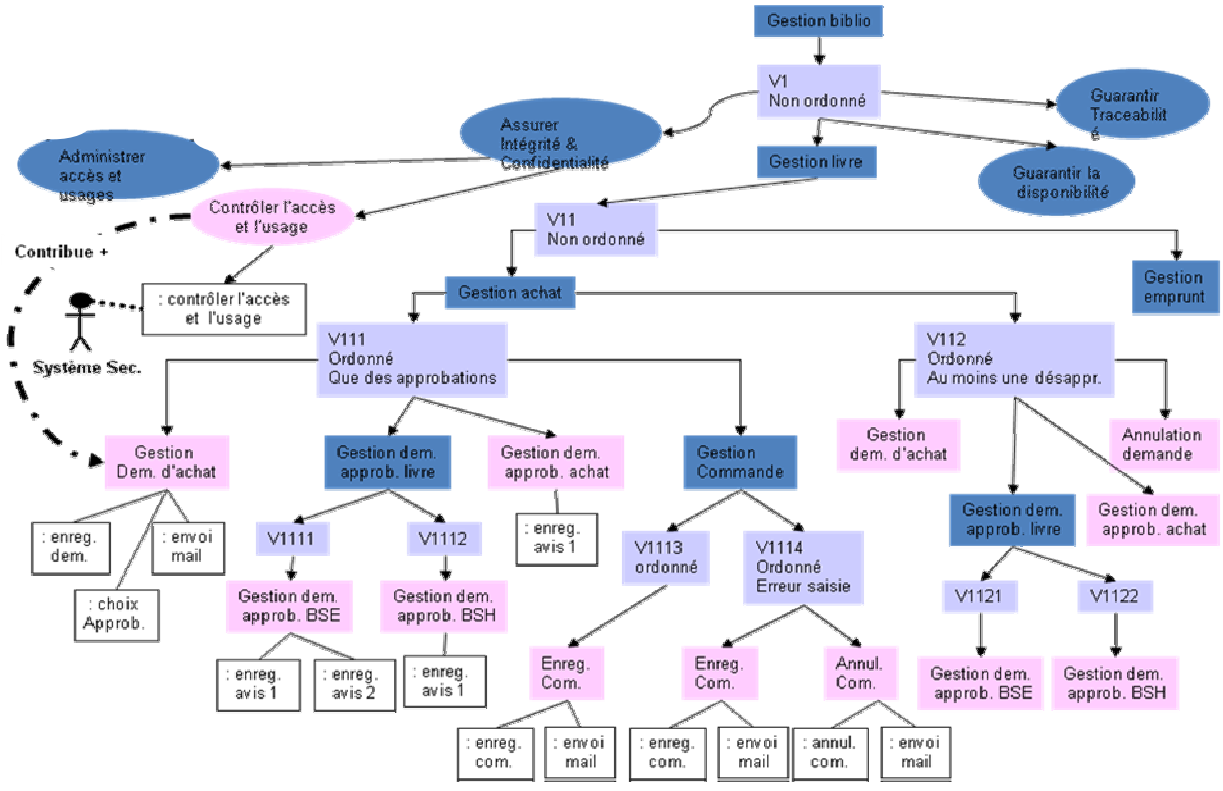

Figure 51. Instance de la perspective de besoins du méta-modèle pour le cas bibliothèque

Instance du méta-modèle organisationnel

Dans l'exemple de la bibliothèque, l'université est composée d'un service financier et de deux facultés qui possèdent chacune une bibliothèque. La faculté des sciences possède la bibliothèque des sciences exactes tandis que la faculté de droit possède la bibliothèque des sciences humaines. La faculté possède des enseignants-chercheurs permanents (*Doyen*) ou non permanents. Elle possède également des étudiants inscrits qui peuvent qui peuvent parfois se spécialiser en étudiants-engagés (sur contrat d'embauche) ou doctorant (pour la recherche). Les agents en interaction avec la bibliothèque jouent un des rôles suivant : bibliothécaire, bibliothécaire-en-chef, Doyen, Enseignant-chercheur, étudiant et le service financier. Le rôle de membre regroupe les différents acteurs ou rôles ayant le droit d'effectuer les opérations d'emprunt de livres. L'exemple présenté ici illustre les habilitations positives qui sont des permissions à l'opposé des habilitations négatives qui traduisent des interdictions. En effet, un étudiant-engagé est habilité positivement (*permission*) de jouer le rôle de bibliothécaire dans la bibliothèque où il est engagé. De même, un étudiant a une habilitation négative (*interdiction*) de jouer le rôle de bibliothécaire-en-chef.

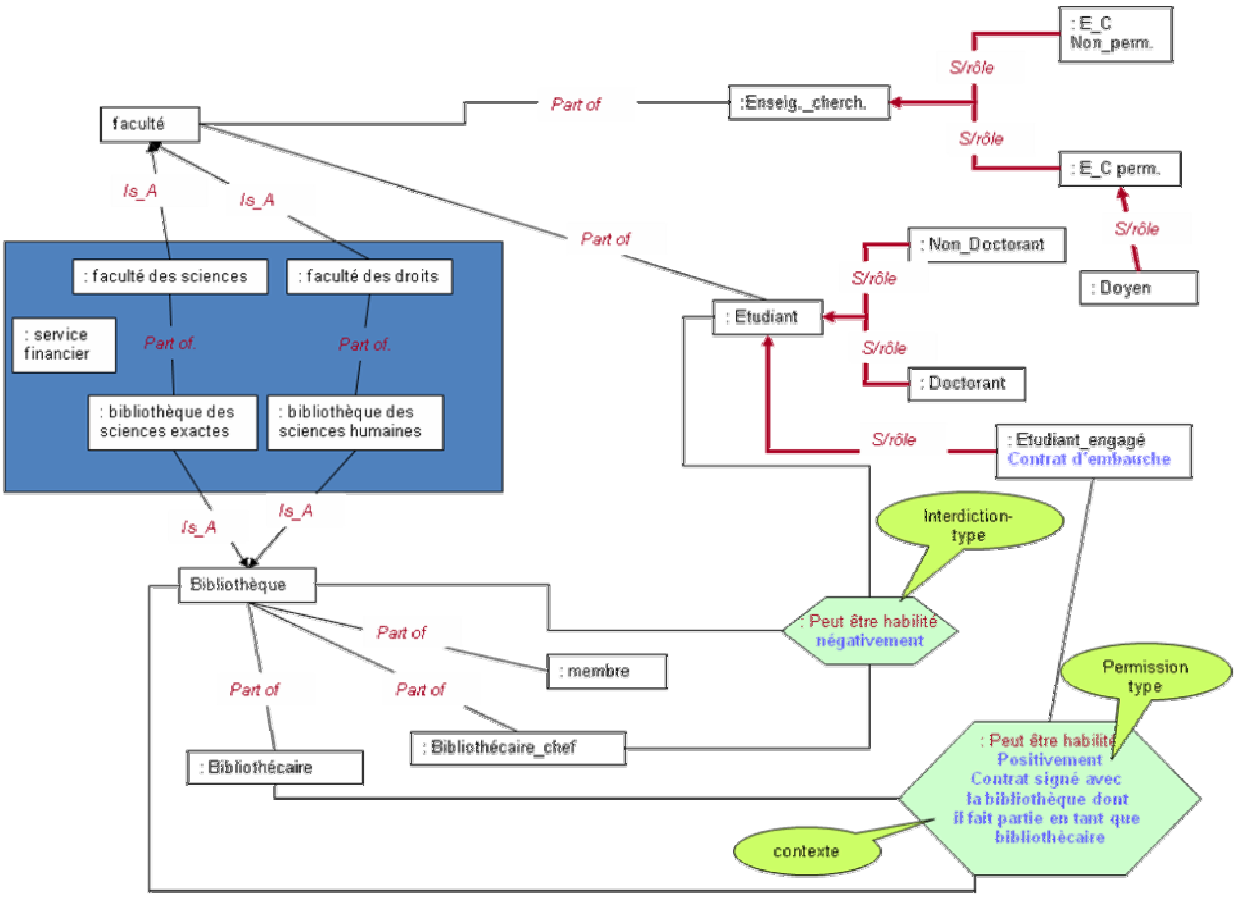

Figure 52. Instance de la perspective organisationnelle du méta-modèle pour le cas bibliothèque

Instance du méta-modèle de processus

Dans l'étude de cas de la bibliothèque, l'intérêt est mis sur le processus d'achat de livre que nous avons étendu pour faciliter la mise en évidence des aspects de sécurité.

Dans ce processus, l'enseignant-chercheur enregistre une demande d'achat d'un livre. La bibliothèque, à travers son application, traite la demande en envoyant des mails aux approbateurs potentiels. Si la demande d'achat est faite dans la bibliothèque des sciences humaines, l'approbation (avis 1) est donnée par un enseignant-chercheur différent de celui qui a fait la demande d'achat.

Pour une demande d'achat de la bibliothèque des sciences humaines, deux approbations sont nécessaires : l'approbation (avis 1) d'un enseignant-chercheur différent du demandeur d'achat et l'approbation (avis 2) du doyen de la faculté ou de son délégué (enseignant-chercheur non demandeur de cet achat) en cas d'indisponibilité. Cette situation est représentée 'ET' qui est un nœud de contrôle de type *AndSplit* comme défini dans [van der Aalst, 2003] puisque les deux avis doivent être obtenus en parallèles et indépendamment l'une de l'autre. Tout avis négatif conduit à l'annulation de la demande et l'envoi de mail au demandeur. Pour la bibliothèque des sciences exactes, un seul avis négatif suffit pour annuler la demande et se traduit par le 'OU' qui représente un nœud de contrôle de type *OrJoin* (voir [Van der Aalst, 2003]). L'expression « **a. BSH ⊕ a.b.BSE** » représente un nœud de contrôle de type *OrJoin* puisque la satisfaction d'une seule des deux conditions permet de passer à la tâche suivant. Pour le cas de la bibliothèque des sciences exactes, l'expression représente un nœud de contrôle de type *AndJoin* parce qu'il faut obligatoirement deux avis favorables pour passer à la tâche suivante.

Si le prix du livre est n'est pas élevé, la commande est enregistrée par le bibliothécaire et le demandeur est informé par mail. En revanche, si le prix du livre s'avère élevé, il faut l'approbation (avis 3) du service financier pour passer la commande. S'il y a une erreur de saisie, le bibliothécaire-en-chef peut annuler la commande dans une durée ne dépassant pas une heure après la saisie. Le demandeur est également informé en cas d'annulation de la commande.

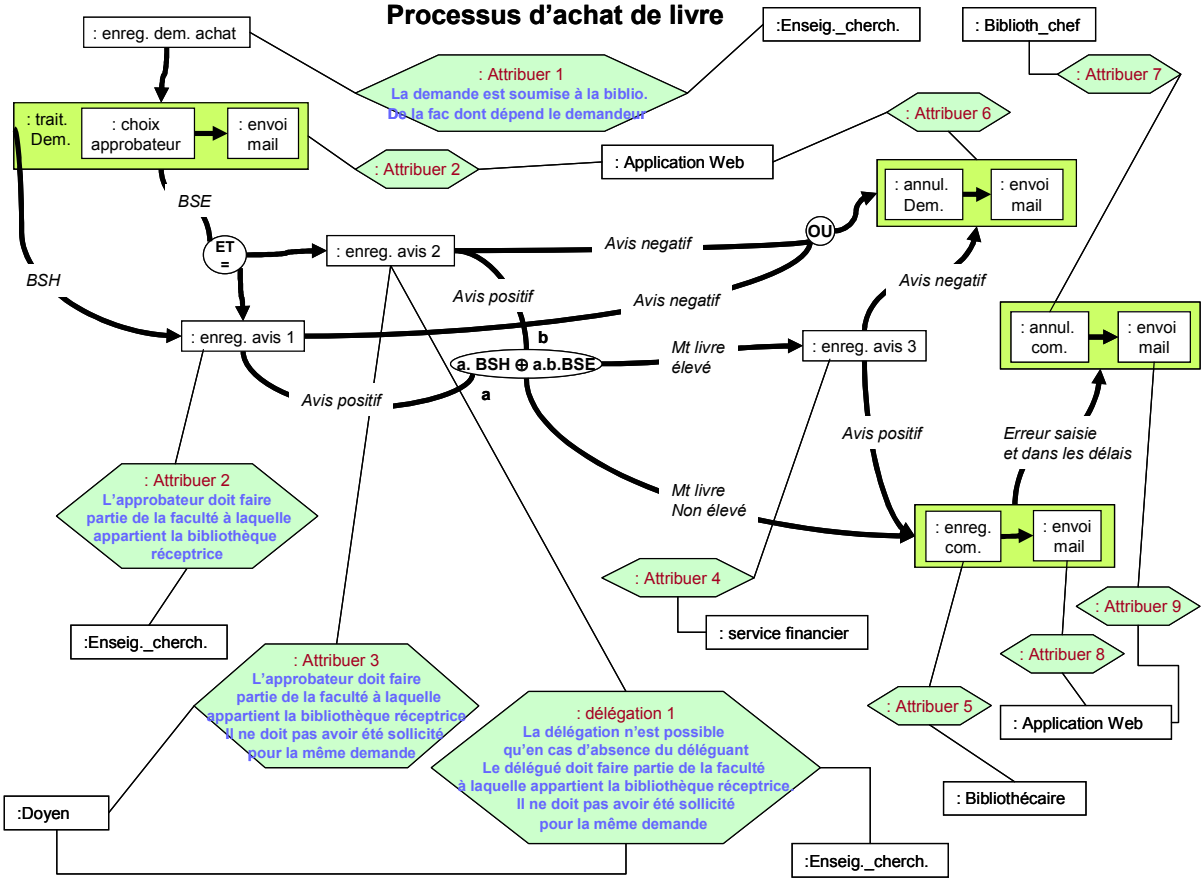

Figure 53. Instance de la perspective processus du méta-modèle pour le cas bibliothèque