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

UPDATE OF L2.1: A UML PROFILE FOR SECURITY CONCEPTS

Livrable n°2.2

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 2011

SOMMAIRE

1. Introduction

Le projet SELKIS a pour objectif de développer une méthode, fondée sur MDA (Model Driven Architecture), d'analyse et de conception de systèmes d'information (SI) sécurisés intégrant les propriétés de sécurité de base, à savoir la Disponibilité, l'Intégrité, la Confidentialité et la Traçabilité (DICT), dans les différentes phases du cycle de développement d'une application depuis la spécification des besoins jusqu'à l'implémentation.

Dans notre précédent livrable 2.1, nous avions présenté notre approche, nommée MoSIS (Modélisation des Systèmes d'Information Sécurisés) pour la prise en compte des propriétés DICT depuis l'étape de spécification des besoins jusqu'à l'implémentation. MoSIS est une approche dirigée par les modèles, fondée sur une architecture MDA. Cette dernière prévoit trois niveaux d'abstraction du système d'information : (1) CIM pour «Computation Independent Model», (2) PIM pour « Platform Independent Model », et (3) PSM pour «Platform Specific Model ». 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. A chacun de ces niveaux, on peut associer des modèles. 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.

Pour permettre une intégration au niveau CIM des aspects fonctionnels (business rules) et des aspects sécurité, nous avions, sur la base de notre étude de l'état de l'art (décrite dans le livrable 2.1), proposé un méta-modèle CIM plus riche que ceux de la littérature, qui englobe la vision métier et qui représente, à un grain très fin, les liens entre les exigences fonctionnelles et non fonctionnelles (notamment la sécurité). De plus, afin de garantir la traçabilité verticale de notre approche, nous avons intégré dans notre méta-modèle les dimensions organisationnelle, comportementale, informationnelle et structurelle des processus métiers. Ce méta-modèle a été appliqué au système d'information pré-hospitalier de IFREMMONT; ce qui nous a permis de produire une nouvelle version améliorée et enrichie de notre méta-modèle. Les améliorations et enrichissements apportés constituent ce livrable 2.2.

Ce livrable contient aussi une nouvelle version des profils UML pour le niveau PIM : (1) celui relatif à la description des fonctionnalités ainsi que (2) celui associé à la description de la structure statique du système d'information. Enfin, il présente une nouvelle version du processus de transformation d'une instance du méta-modèle CIM en une instance du méta-modèle des cas d'utilisation.

2. Version 1.1 du méta-modèle CIM de MoSIS

La représentation des propriétés de sécurité, à différents niveaux d'abstraction, nécessite d'intégrer dans le méta-modèle CIM : 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. Ces trois aspects ont été décrits, dans la version 1.0 du méta-modèle CIM de MoSIS (section 2.1).

2.1 Rappel de la version 1.0 du méta-modèle CIM de MoSIS

A partir d'un état de l'art sur les modèles exploités par les méthodes de conception et de développement de Systèmes d'Information (SI), nous avions présenté, dans le livrable 2.1, la version 1.0 de notre méta-modèle CIM décrivant un SI au niveau métier. Notre description du SI a été faite selon trois points de vue que le méta-modèle est censé satisfaire. Pour des raisons de lisibilité, nous avions représenté chaque point de vue à l'aide d'un méta-modèle : un méta-modèle des processus, un méta-modèle des besoins et un méta-modèle organisationnel. La fusion de ces trois méta-modèles constitue le méta-modèle CIM associé à un SI.

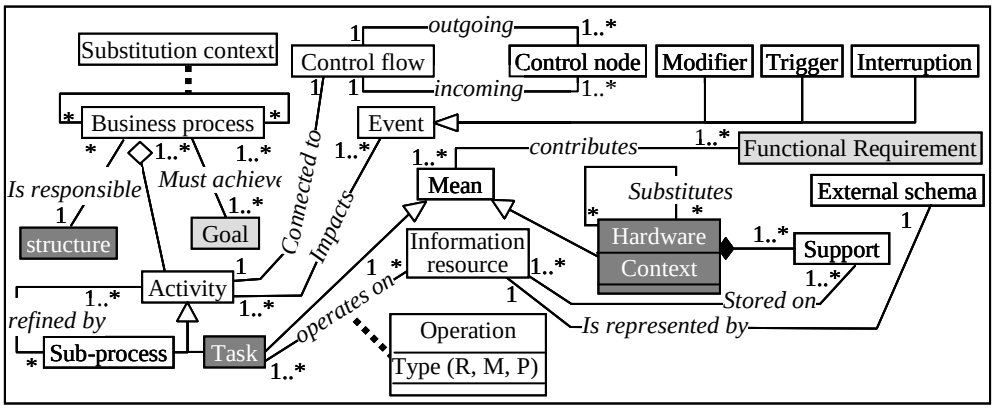

Fig. 1 : Méta-modèle des processus - version 1.0 du méta-modèle CIM de MoSIS

Les figures 1, 2 et 3 décrivent respectivement chacun des méta-modèles. A noter que dans ces derniers, figurent les concepts d'ancrage avec les autres méta-modèles (en gris clair les concepts d'ancrage du méta-modèle des processus avec celui des besoins et en gris foncé ceux liant le méta-modèle des processus au méta-modèle organisationnel).

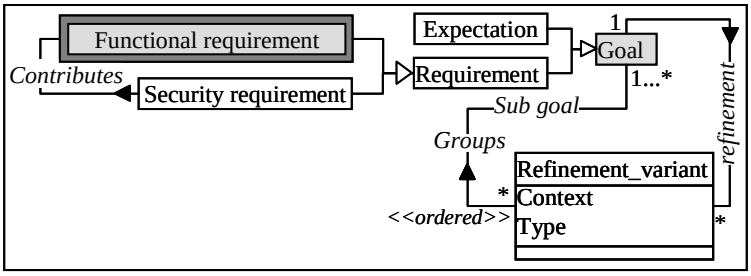

Fig. 2 : Méta-modèle des besoins - version 1.0 du méta-modèle CIM de MoSIS

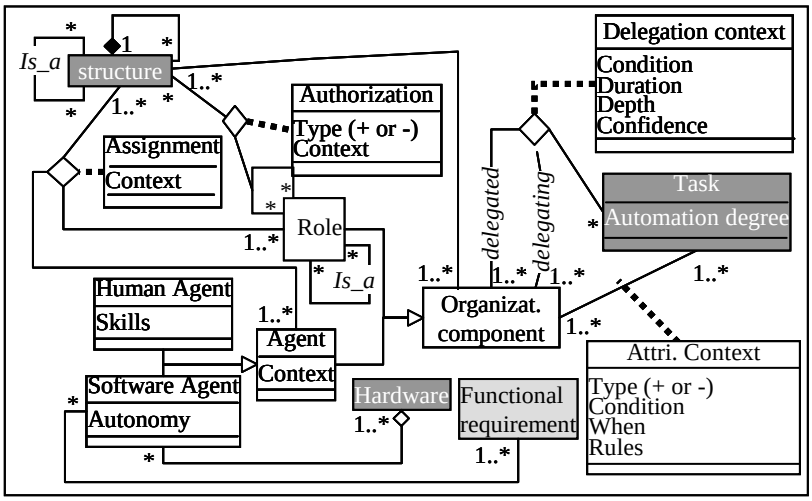

Fig. 3 : Méta-modèle organisationnel - version 1.0 du méta-modèle CIM de MoSIS

Dans notre *méta-modèle des processus*, un processus métier est composé d'activités. Ces dernières peuvent être raffinées en sous processus jusqu'au niveau le plus fin appelé tâche. Un processus permet la réalisation d'un but. Il est sous la responsabilité d'une structure. Une activité peut être modifiée, déclenchée ou encore interrompue par un événement. Le déroulement des activités est décrit par les flux de contrôle. Un processus peut être substitué par un autre processus selon le contexte. Les tâches agissent sur des ressources informationnelles. Les tâches ainsi que les matériels sont des ressources (moyens) contribuant à la réalisation de besoins fonctionnels du SI.

Notre méta-modèle prend en compte toutes les connaissances qui peuvent impacter les propriétés DICT du SI. La disponibilité du SI est, par exemple, impactée dans la mesure où l'on décrit à travers notre méta-modèle la possibilité de substitution de processus et/ou de matériels selon un contexte (exemple: indisponibilité pour une durée donnée). On assure l'intégrité en définissant le type d'usage autorisé des ressources informationnelles (Lecture, Modification, Production) par les tâches. La confidentialité est aussi prise en charge à travers la définition des ressources informationnelles pouvant être manipulées par les tâches. La description comportementale du processus permet de tracer les activités tel que l'exigent certaines applications.

La *méta modélisation des besoins* s'inscrit dans une approche par les buts. Par conséquent, nous représentons les objectifs (ou buts) d'un système d'information que l'on raffine en objectifs élémentaires appelés «besoins». Tous les besoins (fonctionnels et non fonctionnels) sont représentés au moyen de notre méta-modèle. Toutefois, l'accent est mis (via la hiérarchie des besoins) sur les besoins fonctionnels et de sécurité, afin d'identifier la contribution d'un besoin de sécurité à la réalisation d'un besoin fonctionnel. Notre méta-modèle prend en compte les différents raffinements possibles d'un but à travers le concept de variante de raffinement. Etant donné qu'un raffinement n'est possible que sous certaines conditions, nous proposons une propriété «contexte», pour le concept «Variante_raffinement», afin de décrire la condition d'application d'un raffinement donné. De plus, si pour une variante donnée, l'ordonnancement des sous-buts n'est pas exigé alors la propriété «type» de cette variante est positionnée à «non-ordonné» ; cette propriété a par défaut la valeur «ordonné».

Notre méta-modèle des besoins reprend les concepts clés des méta-modèles existants en les enrichissant au moyen, par exemple, du concept de «variante de raffinement» absent des modèles les plus connus. Cependant, il ne prend pas en compte les anti-buts, autre moyen de modélisation des préoccupations de sécurité. Cet aspect pourrait faire partie de nos travaux

futurs. Pour simplifier notre description, nous n'avons pas incorporé dans notre méta-modèle les différentes classifications des buts, utiles pour le guidage lors de la spécification des besoins. Cette omission volontaire de la typologie rend notre méta-modèle plus lisible. Notre objectif principal est l'intégration des besoins de sécurité avec les besoins fonctionnels dès la première phase de développement d'une application. C'est pourquoi, nous mettons l'accent sur les liens entre ces deux catégories de besoins. La connaissance de ces liens sert aussi l'objectif de traçabilité dans le cas d'une évolution du SI.

La composante sociale d'un système d'information est représentée à travers le *méta-modèle organisationnel*. Ce méta-modèle regroupe tous les acteurs directement concernés par le système d'information. Nous avons préféré le terme «agent» à celui d'«acteur» car sa sémantique plus générique permet d'englober aussi bien les acteurs humains (caractérisés par leurs aptitudes) que les agents logiciels (caractérisés par leur degré d'autonomie). Un agent logiciel répond à des besoins fonctionnels du système d'information. Il est hébergé par un matériel. Tout agent (humain ou logiciel) se trouve dans une structure organisationnelle. Il est amené à jouer des rôles bien définis, à exécuter des tâches déterminées, relatives aux rôles qui lui sont attribués. Il peut aussi avoir un certain nombre de privilèges tels que celui de déléguer des tâches. Il est caractérisé par un contexte qui peut impacter sa disponibilité lors de 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.

Notre méta-modèle fait la distinction entre l'affectation géographique des acteurs et les habilitations dont ils peuvent jouir. L'affectation d'un agent à une structure peut être contrainte. Cette contrainte est exprimée dans le méta-modèle à l'aide de l'attribut «contexte» de l'association « affectation ». 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 une permission faisant partie des politiques de sécurité à mettre en œuvre. Une habilitation négative correspondra, quant à elle, à une interdiction.

Une tâche est attribuée à une composante organisationnelle (c'est-à-dire à un rôle ou à un agent). Cette attribution peut être contrainte. L'attribut contexte de l'association «attribution» représente cette contrainte. 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 transférée 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 reflète l'interdiction de transmission 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 le choix de la délégation lorsque deux délégués sont candidats à l'exécution d'une tâche.

2.2 Changements apportés à la version 1.0

Une première instanciation de notre méta-modèle CIM a été réalisée sur le cas « bibliothèque » décrit en annexe 4 du livrable 2.1. Elle a été présentée en annexe 5 de ce même livrable. Une seconde instanciation a concerné le système d'information pré-hospitalier d'IFREMMONT, un de nos partenaires dans le projet. Dans un premier temps, nous nous sommes fondés sur les documents fournis par notre partenaire [1, 2] pour effectuer une instanciation. Dans un second temps, nous avons refait cet exercice en nous fondant sur des échanges en plénière lors de notre réunion du 14 janvier 2011.

Suite à cette instanciation, nous avons procédé à des changements dans notre méta-modèle et cela afin de gagner en généricité. Les changements ont concerné le méta-modèle organisationnel. Le premier changement effectué a été l'introduction du concept «type de structure», l'objectif étant de faire la distinction entre les règles génériques, rôles génériques, etc. (i.e. applicables à toute structure de même type) et ceux spécifiques (i.e. applicables à des structures particulières). En effet, certains types d'organisation, tels qu'IFREMMONT, regroupent des structures de même type, c'est-à-dire des structures qui disposent de procédures de travail semblables, qui obéissent aux mêmes règles de fonctionnement et qui regroupent des acteurs dont les rôles sont identiques. La connaissance de ces règles et rôles applicables à tout type de structure permettrait de les répercuter de façon systématique au niveau des structures. Autrement dit, nous en avons déduit l'intérêt d'introduire le concept de type de structure, permettant de factoriser les caractéristiques communes à un ensemble de structures fonctionnant sur des bases identiques.

L'ajout, dans le méta-modèle, du concept «type structure» a, par conséquent, impacté les liens existant entre une structure et les autres concepts du modèle. A titre d'exemple, un agent fait partie d'une structure alors qu'un rôle fait partie d'un type de structure. De plus, la hiérarchisation des structures n'a plus lieu d'être car elle peut être obtenue par calcul à partir de la hiérarchisation des types de structures.

L'instanciation de notre méta-modèle au cas IFREMMONT nous a aussi permis d'être plus précis dans la caractérisation des structures. En effet, certaines organisations, telles qu'IFREMMONT, sont amenées, pour la réalisation de leurs objectifs, à mettre en place des structures temporaires (c'est-à-dire des structures dont la durée de vie est très limitée, par exemple une équipe ad hoc constituée pour une intervention en urgence aura une existence de quelques heures au maximum). La mise en place de telles structures sous entend aussi une affectation temporaire d'agents à ces structures, des attributions temporaires de rôles et de tâches, etc. Pour une meilleure administration de la sécurité, nous avons jugé utile de prendre en charge dès le niveau CIM cet aspect.

Le nouveau méta-modèle organisationnel de MoSIS est représenté à la figure 4.

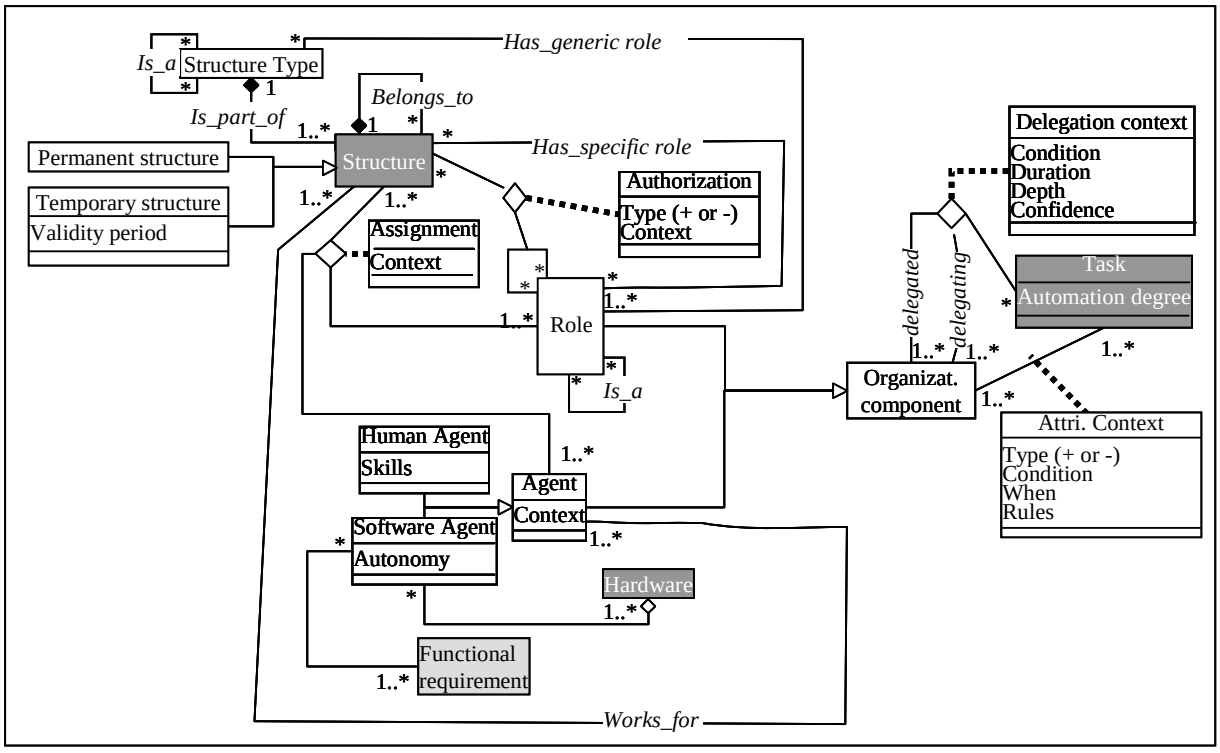

Fig. 4 : Méta-modèle organisationnel - version 1.1 du méta-modèle CIM de MoSIS

3. Profils UML pour le niveau PIM

A l'aide de MoSIS, nous construisons un système d'information sécurisé selon trois étapes. Chacune d'elles correspond à un niveau d'abstraction MDA. La première étape a pour but la description du système d'information et de son environnement. C'est dans cette étape que s'effectue l'analyse de l'environnement organisationnel dans lequel 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. Le résultat de cette étape est une instance du méta-modèle CIM.

La seconde étape correspond à l'analyse et la conception abstraite du système d'information. 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 la sécuriser. Ces modèles sont des instanciations de méta-modèles de niveau PIM qui, eux, sont des extensions des méta-modèles d'UML. Ces extensions sont fondées sur des profils.

La troisième étape est dédiée à la conception détaillée dans laquelle la plateforme technologique, qui accueillera le système d'information, a été prise en compte.

Nous avions, dans le livrable 2.1, proposé une première version des profils des cas d'utilisation pour la description des fonctionnalités d'un SI ainsi que celui des classes pour la

description de la structure statique du SI. Ces profils ont été enrichis. Nous les présentons respectivement dans les sections 3.1 et 3.2.

3.1 Profil UML pour la description des fonctionnalités

Un profil décrit la syntaxe et la sémantique des concepts utilisés dans un domaine particulier. Les concepts du domaine sont représentés en utilisant des 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 5 présente les stéréotypes définis pour la description des fonctionnalités d'un système sécurisé.

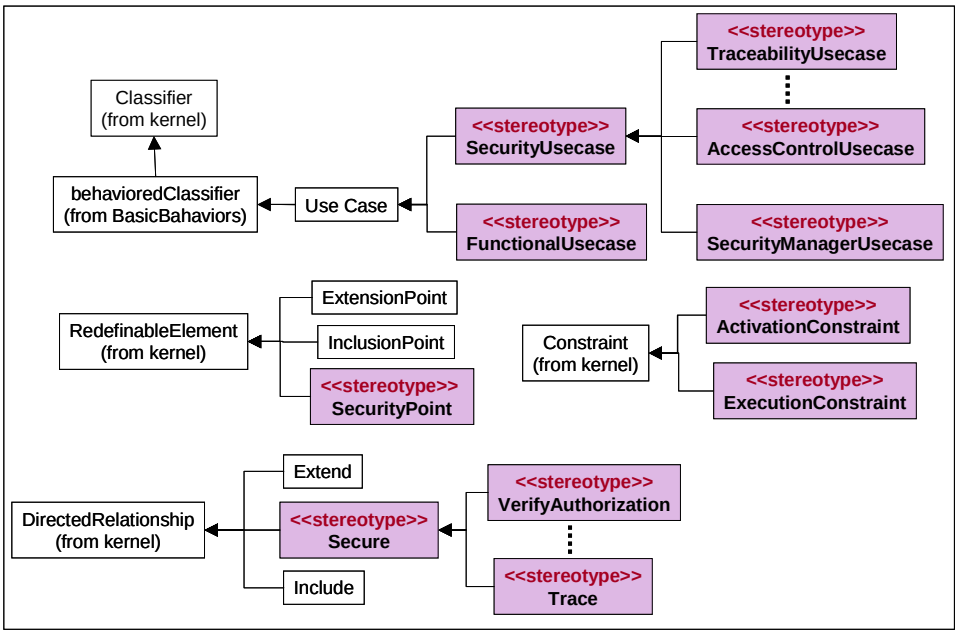

Fig. 5 : Stéréotypes utilisés pour la description des fonctionnalités d'un SI sécurisé

Dans l'ensemble des cas d'utilisation d'UML, nous distinguons les cas décrivant les fonctionnalités du système de ceux décrivant les exigences de sécurité. Les cas d'utilisation de sécurité sont de trois types : les gestionnaires de la sécurité (securityManagerUseCase), les contrôleurs d'accès (AccesscontrolUsecase) 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 ont en charge l'analyse et le respect des politiques de sécurité définies par les gestionnaires. Les traceurs permettent d'enregistrer les informations pertinentes relatives aux éléments importants du système et qui ont un impact sur sa vulnérabilité.

Le langage UML propose généralement deux types de liens entre cas d'utilisation : le lien d'extension (extend) et le lien d'inclusion (include). Pour prendre en charge les besoins de sécurité, nous proposons d'étendre ces liens en y ajoutant 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) et d'inclusion (inclusion 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. Nous avons aussi défini deux stéréotypes pour l'expression des contraintes : ceux conditionnant l'exécution des cas d'utilisation (Execution Constraint) et ceux contraignant la communication d'un acteur avec un cas (Activation Constraint).

Notre méta-modèle PIM pour la description des fonctionnalités d'un système d'information sécurisé est représenté à la figure 6.

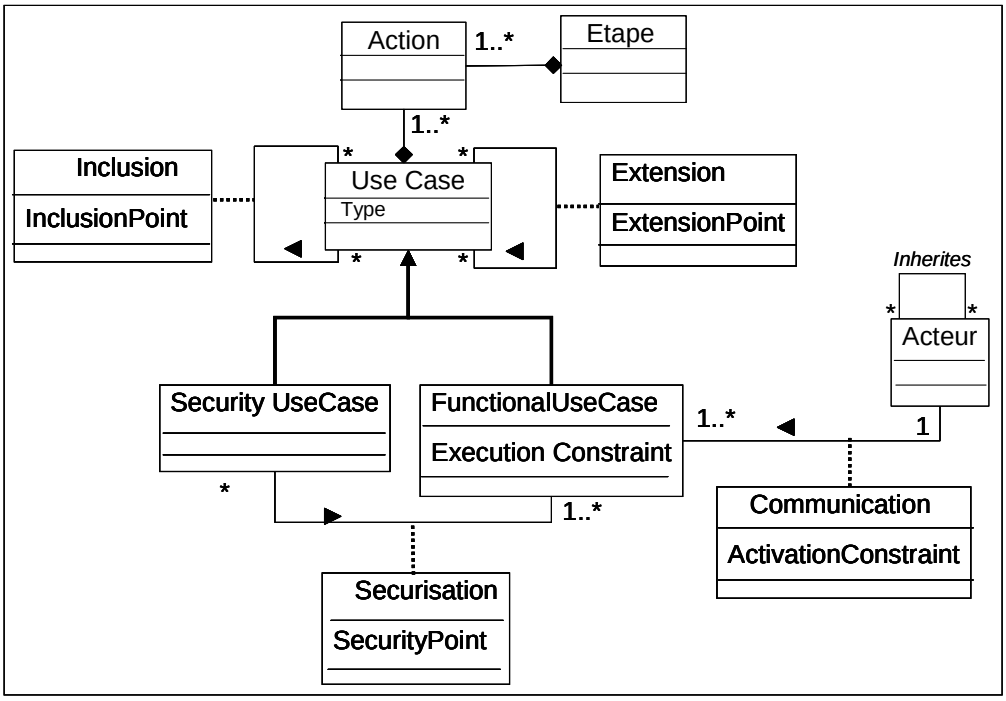

Fig. 6 : Profil UML pour la description des fonctionnalités d'un SI sécurisé

Comme le montre la figure 6, 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 regroupe un certain nombre d'actions élémentaires. Ces actions peuvent être aussi regroupées en étapes. Un cas fonctionnel obéit à des règles de déclenchement dictées par le processus dont il fait partie (Execution Constraint). 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 (Activation Constraint). Généralement les conditions d'activation découlent des contextes de permission et de délégation mentionnés au niveau CIM.

3.2 Profil UML pour la description de la structure statique

Le méta-modèle de sécurité de la figure 7 intègre les concepts de sécurité couverts à un niveau conceptuel indépendant des plates-formes (PIM). Outre les concepts de base de Role Based Access Control (RBAC), pris en compte par l'approche SecureUML, nous avons intégré les notions d'organisation, d'affectation et de délégation évoquées lors de la modélisation des exigences de sécurité au moyen des modèles CIM.

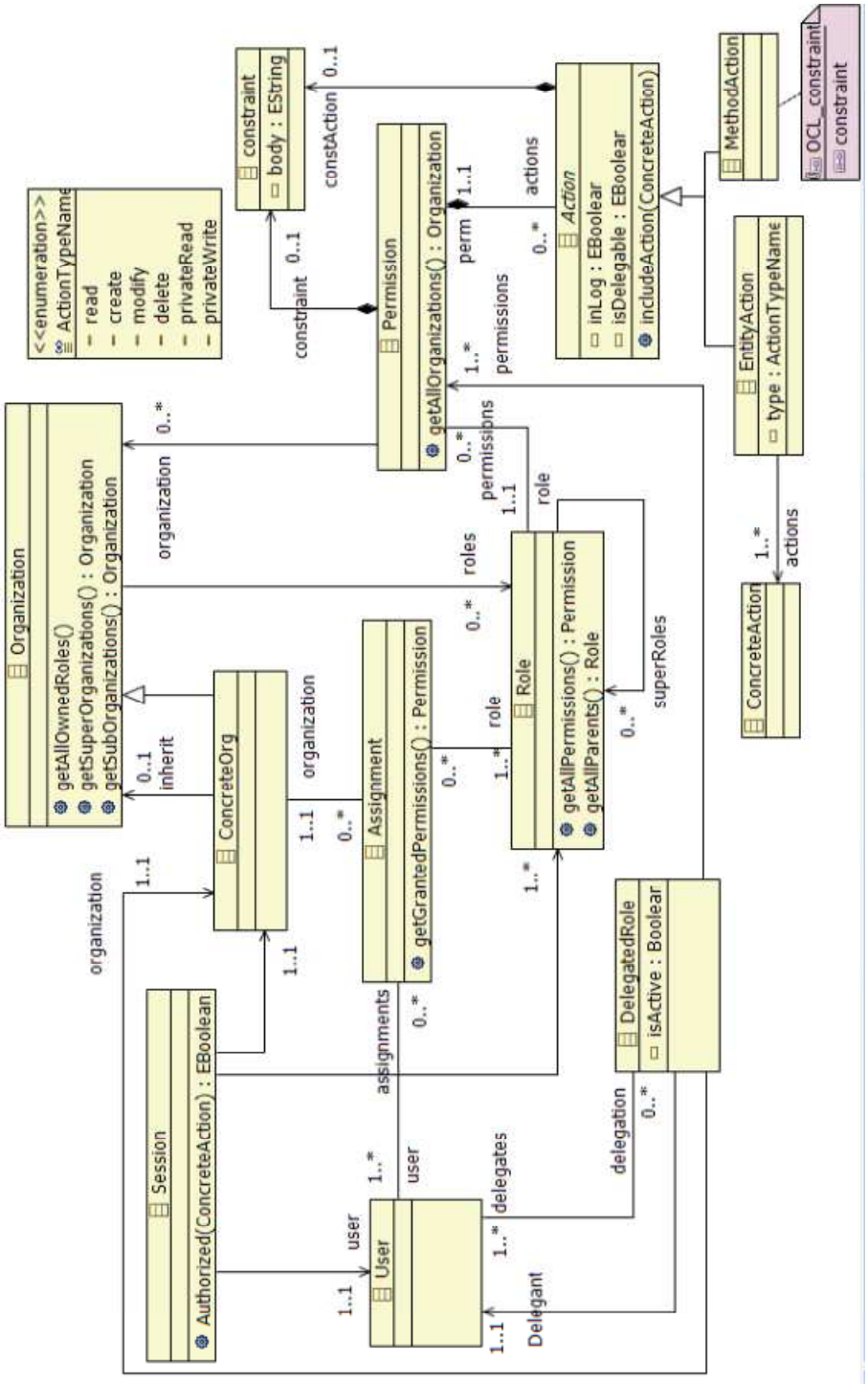

Fig. 7 : Profil UML pour la description statique d'un système d'information sécurisé

3.2.1. Concepts de base : Rôle, Permission et Action

La figure 8 présente un exemple simple de modèle de sécurité couvert par notre méta-modèle. Dans cet exemple, nous illustrons les notions de rôle (Nurse, Doctor, etc), de permission (MedicalAdvicePerm, etc) ainsi que les actions (EntityAction et MethodAction). Comme indiqué au niveau du méta-modèle, les concepts sécuritaires sont rattachés aux entités fonctionnelles en vue d'indiquer les droits d'accès à ces entités. Dans la figure 8, nous considérons que l'entité MedicalRecord dispose d'attributs publics et privés ainsi que des méthodes Close et Validate.

La permission Medical_MRPerm liant le rôle Medical et l'entité MedicalRecord indique que tout personnel médical (qu'il soit interne, médecin ou infirmier) a le droit de lire les données d'un acte de soin. En effet, une EntityAction de type PrivateRead donne une autorisation de lecture des attributs publics et privés de l'entité.

L'héritage entre rôles permet de factoriser les permissions communes aux sous-rôles. En effet, outre le droit de lecture provenant du rôle Medical, un interne a la possibilité (via la permission Intern_MRPerm) de modifier les données de l'acte de soin. Une EntityAction de type Modify permet la modification des attributs publics uniquement. Ainsi, un interne peut modifier l'attribut public data de MedicalRecord. Cependant, il n'a pas le droit d'agir sur les attributs privés, ni sur les opérations. La permission Doctor_MRPerm liant le rôle Doctor à MedicalRecord indique qu'un docteur peut créer (<<EntityAction>> Create) de nouveaux actes de soin et aussi modifier (<<EntityAction>>Modify) leurs données. Cependant, le docteur agit sur les attributs privés de MedicalRecord (isClosed et isValid) de façon contrôlée via les opérations Close et Validate. En effet, les MethodAction permettent d'indiquer des autorisations d'exécution d'opérations.

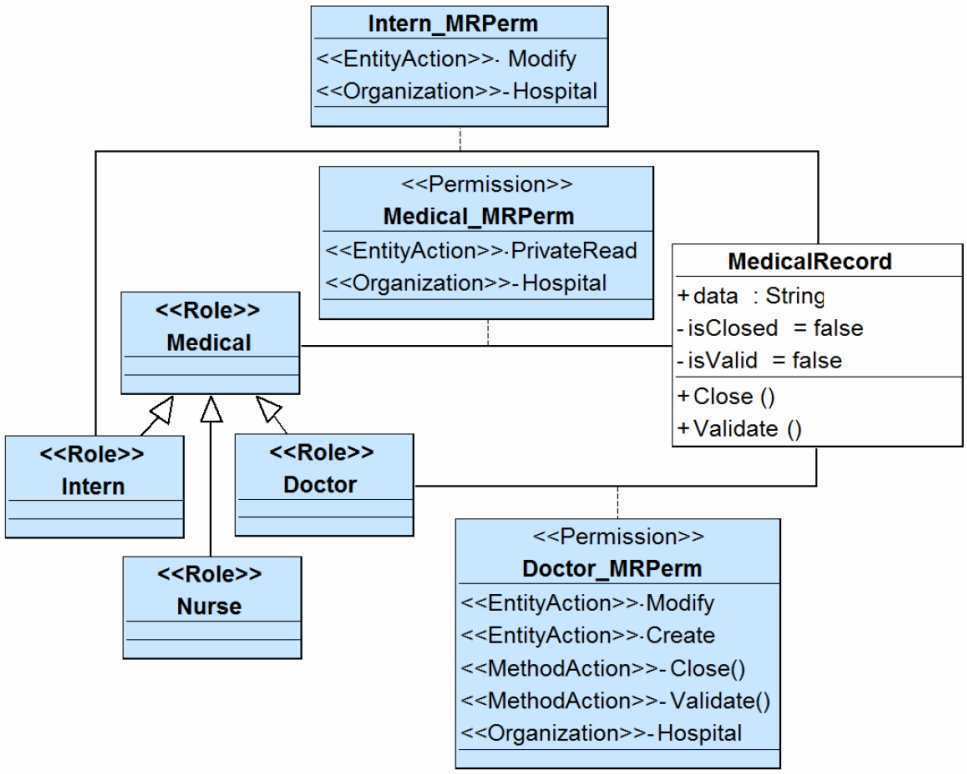

Fig. 8 : Exemple de modèle de sécurité

3.2.2. La prise en compte des organisations

Dans la figure 8, nous ne considérons qu'une seule organisation qui est « Hospital ». Dans la syntaxe concrète, nous choisissons de représenter les organisations de deux façons différentes – mais aboutissant à une même instance du méta-modèle de sécurité. La première considère qu'une organisation est un attribut stéréotypé par <<Organization>> ou <<ConcreteOrg>> et qui est encapsulé dans les permissions. La deuxième considère qu'une organisation est une classe portant l'un de ces stéréotypes. La classe est liée à ses permissions. Dans la figure 9 nous considérons cette deuxième représentation : le stéréotype <<Organization>> indique qu'il s'agit d'une organisation abstraite alors que le stéréotype <<ConcreteOrg>> labellise une organisation concrète.

La notion d'organisation abstraite permet simplement de factoriser certaines permissions. En effet, dans l'exemple de la figure 9, la permission Nurse_MRPerm1 rattachée à l'organisation abstraite « Hospital » est commune aux organisations concrètes « H1 » et « H2 ». Cependant, la permission Nurse_MRPerm2 est spécifique à l'organisation concrète « H2 ». L'exemple considère donc qu'un infirmier a le droit de lire les informations d'un acte de soin, et ce, qu'il soit dans l'hôpital H1 ou dans l'hôpital H2. A l'inverse, l'hôpital H2 permet, de plus, à ses infirmiers de modifier les données d'un acte de soin.

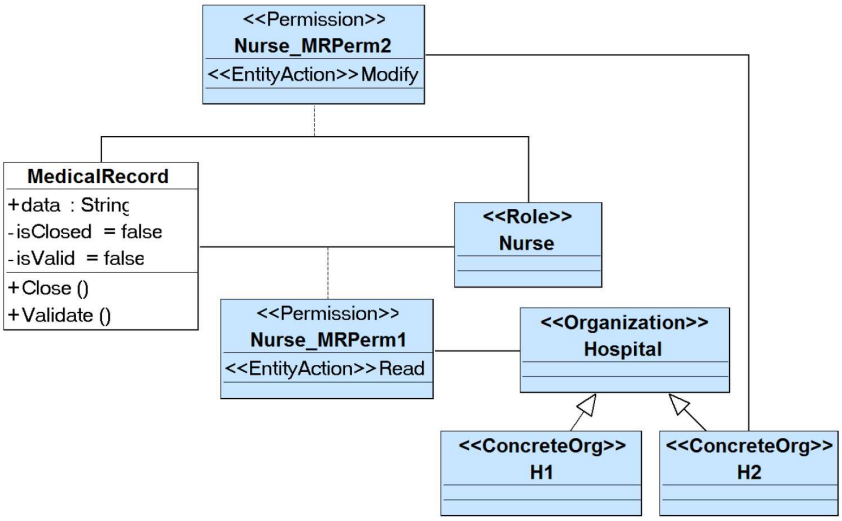

Fig. 9 : Exemple de modèle illustrant les organisations

Les organisations abstraites permettent également de factoriser les rôles communs aux organisations concrètes. La figure 10 illustre le fait que les rôles TEAM_MEMBER, RESCUER, NURSE et DOCTOR sont des rôles communs à l'hôpital Purpan et à l'hôpital Joseph Ducoing. D'autres rôles peuvent être définis de façon spécifique pour les organisations concrètes.

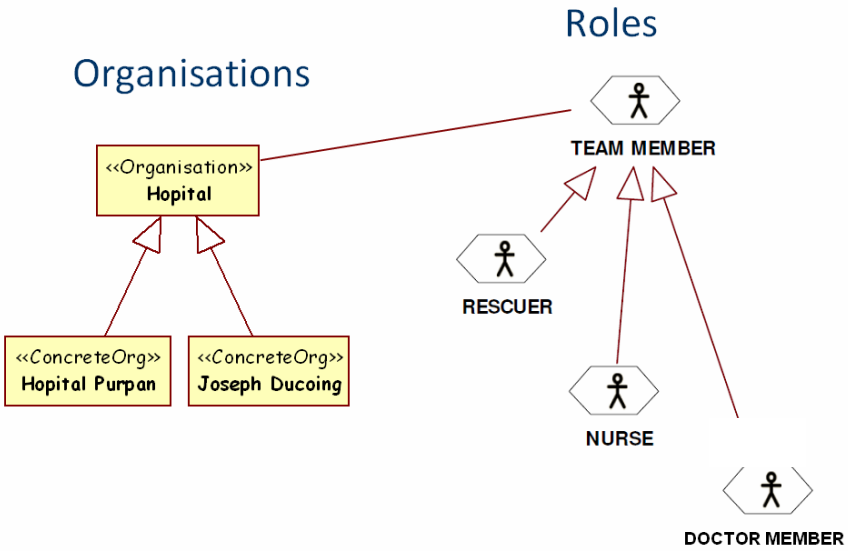

Fig. 10 : Liens entre organisations et rôles

3.2.3. Les affectations

Les affectations sont spécifiées dans le méta-modèle au moyen de la méta classe « Assignment ». Elles lient des utilisateurs, des rôles et des organisations concrètes. Chaque affectation s'exprime dans une et une seule organisation concrète et lie un ensemble

d'utilisateurs à un ensemble de rôles. Dans l'exemple de la figure 11, Jack joue le rôle « DOCTOR » dans l'hôpital Joseph Ducoing et le rôle « TEAM_MEMBER » dans l'hôpital Purpan. En revanche, les utilisateurs Bob et Paul ne peuvent jouer que le rôle « TEAM_MEMBER » au sein de l'hôpital Joseph Ducoing.

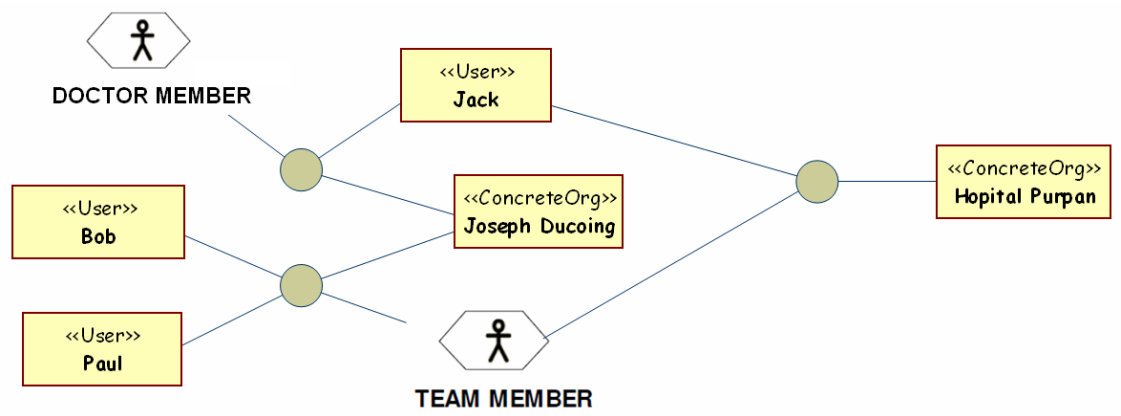

Fig. 11 : Exemple d'affectations

Enfin, la notion de session (méta-classe « Session » de la figure 7) est étroitement liée aux affectations. Le méta-modèle de sécurité indique qu'une session est spécifique à un utilisateur et une organisation concrète. Dans une session, l'utilisateur peut activer un ou plusieurs rôles, parmi les rôles qui lui sont affectés dans l'organisation concrète.

3.2.4. La délégation

Au niveau CIM, nous avons eu le souci de décrire le concept de délégation sous une forme la plus générale possible. Au niveau PIM, nous nous limitons à une vision de la délégation plus restreinte dans le but de permettre une vérification formelle. En effet, nous ne couvrons pas le cas où un utilisateur peut déléguer ses rôles ou toutes ses permissions. Ceci est justifié par le fait que les modèles PIM seront traduits dans des notations formelles en B et en Z pour effectuer des vérifications/validations automatisées. Le mécanisme général de délégation est complexe. En conséquence, il peut rendre ces vérifications/validations délicates. De plus, la délégation peut engendrer des failles de sécurité si elle n'est pas réalisée avec précaution.

Dans le méta-modèle de sécurité, nous limitons ce mécanisme à la méta classe « DelegatedRole » qui indique que le concepteur d'une politique de sécurité doit explicitement créer un rôle (dit délégable) pour chaque utilisateur (dit déléguant) ayant la possibilité de déléguer certaines de ses permissions. Le rôle délégable fera donc référence aux permissions que le déléguant a le droit de déléguer à d'autres utilisateurs (dits délégués). Nous ajoutons la contrainte selon laquelle les permissions d'un rôle délégable doivent être un sous-ensemble des permissions accordées à l'utilisateur déléguant par le biais de tous ses rôles. En effet, le lien de référence entre « DelegatedRole » et permission a pour simple but de référencer des permissions déjà créées et assignées à des rôles du modèle.

Quand un utilisateur s'absente, alors que certaines de ses tâches sont nécessaires au bon fonctionnement du système, il est amené à rendre effectives les délégations nécessaires. Ceci est spécifié par l'attribut isActive de « DelegatedRole » qui indique si la délégation est effective ou non.

Cette description complète l'ensemble des profils UML définis pour le niveau PIM. Ainsi, les évolutions précédemment décrites tant au niveau CIM qu'au niveau PIM vont nous permettre, dans la suite, de définir le passage d'un niveau à l'autre.

4. D'un modèle de niveau CIM vers un diagramme des cas

Il existe plusieurs chemins de passage du niveau CIM vers le niveau PIM. Nous nous intéressons ici à la transition consistant à utiliser les diagrammes de cas d'utilisation.

La génération des diagrammes de cas d'une application sécurisée à partir de sa description au niveau CIM s'effectue en deux temps (voir figure 12). Dans un premier temps, sont générés les cas d'utilisation fonctionnels et, parallèlement, les cas de sécurité. Dans un second temps, on procède à leur intégration. Celle-ci s'appuie sur les liens existant entre les besoins fonctionnels et les exigences de sécurité. Les uns et les autres sont décrits dans l'instance du 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 vers le cas de sécurité correspondant.

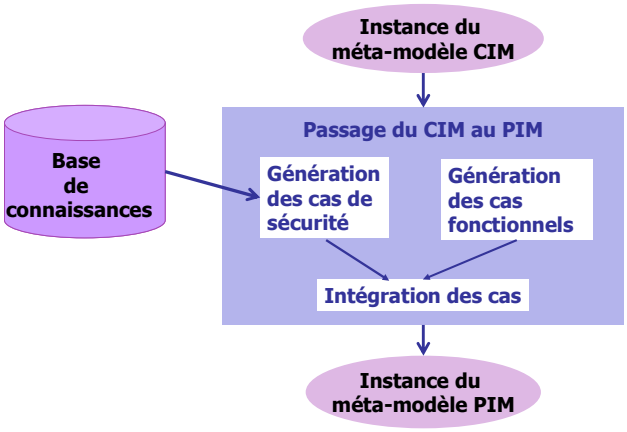

Fig. 12 : Approche de transformation d'un modèle de niveau CIM vers un diagramme des cas d'utilisation

Pour la génération des cas de sécurité, on exploite la description des besoins de sécurité (instance du méta-modèle CIM de MoSIS) de l'application ainsi que l'ontologie des besoins de sécurité génériques de la base de connaissances de MoSIS (voir figure 13). On procède à une mise en correspondance des besoins de sécurité spécifiés dans l'instance du méta-modèle CIM avec ceux se trouvant dans l'ontologie. Cette mise en correspondance permet de retrouver les cas d'utilisation génériques correspondants.

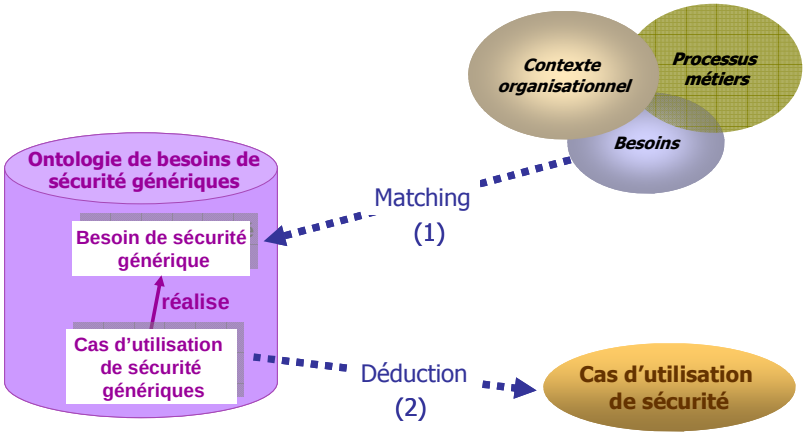

Fig. 13 : Génération des cas d'utilisation de sécurité

La génération de cas d'utilisation fonctionnels à partir d'une instance du méta-modèle CIM s'appuie sur des règles de transformation de concepts du méta-modèle CIM en concepts

propres aux diagrammes de cas fonctionnels. Elle se déroule en deux étapes. La première étape consiste en la dérivation du diagramme des cas (acteurs, cas fonctionnels, scénarios des cas, liens de communication). La seconde étape procède à une re-structuration des cas suite à des fusions ou encore à l'introduction de liens d'extension ou d'inclusion.

Les règles de transformation sont catégorisées selon les concepts qu'elles permettent de générer. Parmi ces règles, certaines permettent de déduire des packages fonctionnels, d'autres de les structurer en cas d'utilisation. D'autres règles encore vont permettre de déduire les différents acteurs et les liens d'héritage existant entre eux. De plus, nous fournissons des règles pour la déduction de cas d'utilisation. Enfin, d'autres règles concernent la structuration par fusion ou via des liens d'inclusion ou d'extension. Certaines de ces règles sont fournies en annexe 1 de ce document.

5. Conclusion

Nous avons présenté, dans ce livrable, les évolutions apportées à notre démarche MoSIS de **Mo**délisation des **S**ystèmes d'**I**nformation **S**écurisés. Ces évolutions concernent aussi bien les méta-modèles de niveau CIM que ceux de niveau PIM, c'est-à-dire les méta-modèles de profil UML pour la description fonctionnelle et statique d'un système d'information.

D'autres enrichissements pourraient être apportés au méta-modèle CIM. Parmi ceux-ci, on peut citer la prise en compte des anti-buts ainsi qu'une caractérisation des buts de sécurité (exemple : priorité ou importance). Celle-ci pourrait être intégrée dans notre démarche au sein d'une phase d'analyse des risques préalable. Cela permettrait de procéder à une itération supplémentaire au niveau CIM pour prendre en compte ces anti-buts et leurs liens avec les autres buts. Un autre enrichissement serait de fournir au concepteur un guidage pour l'instanciation du méta-modèle ainsi que pour la formalisation du contexte à travers des contraintes.

Il est aussi prévu de procéder à la formalisation des règles de transformations permettant de déduire, à partir d'un modèle de niveau CIM, la description des fonctionnalités d'un système d'information sécurisé et leur application au cas IFREMMONT. De plus, nous comptons, à l'aide de MoSIS, produire des règles de transformation permettant de déduire la description statique d'un système d'information sécurisé.

Références

[1] Quentin Switsers et Pascal Zellner. « Flux et données d'un SI pré hospitalier». Meeting projet SELKIS – Grenoble 6 et 7 juillet 2009. IFREMMONT, http://lacl.univ-paris12.fr/selkis/sites/default/files/SELKIS_IFREMMONT_Description%20du%20contexte.pdf

[2] Quentin Switsers et Pascal Zellner. «Processus de gestion ManagementAct». IFREMMONT, 3 juin 2010.

Annexe 1 : Quelques règles de transformation d'un modèle de niveau CIM en cas d'utilisation fonctionnels

R1 : Tout processus se transforme en un package fonctionnel
R2 : Toute composante organisationnelle donne naissance à un acteur du diagramme des cas
R3 : Toute tâche T automatique ou semi-automatique devient un cas élémentaire K ayant pour étape T
R4 : Les conditions d'exécution de toute tâche T correspondant à un cas élémentaire K deviennent les conditions d'exécution de K.
R5 : Deux cas élémentaires correspondant chacun à des tâches d'un même processus feront partie d'un même package
R6 : **SI** K est un cas élémentaire
ET SI A est un acteur du diagramme des cas
ET SI la tâche correspondant au cas K est liée à la composante organisationnelle C à travers le lien d'attribution L
ALORS créer un lien de communication entre K et l'acteur A issu de C
R7 : **SI** K est un cas élémentaire
ET SI A est un acteur du diagramme des cas
ET SI la tâche correspondant au cas K est liée à la composante organisationnelle C à travers le lien d'attribution L
ALORS renseigner les conditions d'activation du lien de communication entre K et l'acteur A issu de C par le contexte de L
R8 : **SI** K est un cas élémentaire
ET SI la tâche correspondant au cas K peut être déléguée par une composante organisationnelle C1 à la composante organisationnelle C2 à travers le lien de délégation D
ET SI C1 correspond à l'acteur A communiquant avec K
ALORS créer un lien de communication entre K et l'acteur correspondant à C2 et intégrer dans les conditions d'activation de ce lien le contexte de D ainsi qu'avec les conditions d'attribution de C1 à K
R9 : **SI** K est un cas élémentaire
ET SI la tâche correspondant au cas K peut être déléguée par une composante organisationnelle C1 à la composante organisationnelle C2 à travers le lien de délégation D
ET SI C1 correspond à l'acteur A communiquant avec K
ALORS intégrer dans les conditions d'activation du lien de communication entre K et l'acteur correspondant à C2, le contexte de D ainsi que les conditions d'activation mentionnées entre C1 et K.
R10 : SI une tâche T2 est automatique
ET SI T2 est la tâche qui suit T1
ET SI T1 correspond à l'étape E1 d'un cas K1
ET SI T2 correspond à l'étape E2 d'un cas K2
ET SI T1 et T2 réalisent les mêmes besoins fonctionnels
ET SI K1 et K2 font partie d'un même package
ALORS intégrer l'étape E2 au cas K1 et la supprimer de K2
R11 : SI une tâche T2 est automatique
ET SI T2 est la tâche qui suit T1
ET SI T1 correspond à l'étape E1 d'un cas K1

ET SI T2 correspond à l'étape E2 d'un cas K2
ET SI T1 ET T2 réalisent les mêmes besoins fonctionnels
ET SI K1 et K2 font partie d'un même package
ALORS supprimer l'étape E2 de K2

R12 : **SI** deux cas K1 et K2 font partie d'un même package
ET SI K1 et K2 communiquent avec les mêmes acteurs
ET SI l'ensemble des tâches correspondantes aux étapes de K1 ont les mêmes besoins fonctionnels que les tâches correspondantes aux étapes de K2
ET SI K1 et K2 sont dans le même package
ALORS mettre les étapes de K1 dans K2 et supprimer K1

R13 : **SI** deux cas K1 et K2 font partie d'un même package
ET SI K1 et K2 communiquent avec les mêmes acteurs
ET SI l'ensemble des tâches correspondantes aux étapes de K1 ont les mêmes besoins fonctionnels que les tâches correspondantes aux étapes de K2
ET SI K1 et K2 sont dans le même package
ALORS supprimer K2