Ingénierie Dirigée par les Modèles pour la Spécification d'un système de paiement de frais scolaire dans des écoles privées de la République Démocratique du Congo

Model Driven Engineering for the Specification of a School Fees Payment System in Private Schools in the Democratic Republic of Congo

NYAMI Nyate Ruphin
Institut Supérieur de Commerce d'Ilebo
République Démocratique du Congo
ruffinnyami125@gmail.com

Date de soumission : 30/03/2021

Date d'acceptation : 26/05/2021

Pour citer cet article :

NYAMI. N. R (2021) «Ingénierie Dirigée par les Modèles pour la Spécification d'un système de paiement de frais scolaire dans des écoles privées de la République Démocratique du Congo», Revue Internationale du chercheur «Volume 2 : Numéro 2» pp : 851 - 875

Résumé

Cet article concerne l'Ingénierie Dirigée par les Modèles (IDM) pour la spécification d'un Système de Paiement de Frais Scolaire (SPFS) dans des établissements privés d'enseignements. L'obsolescence perpétuelle des Framework de développements de logiciels fait pencher la balance au profit de la sédentarité du savoir-faire de besoins métiers, afin de permettre aux établissements d'enseignements de capitaliser leurs connaissances sur les procédées de la perception de frais scolaires ; sans inquiétude sur le choix technologique. Dans cet article nous allons modéliser une architecture logicielle pour le système (SPFS) en se basant sur les techniques d'Architecture Dirigée par les Modèles (ADM) ou MDA en anglais (Model Driven Architecture). L'allégeance au MDA est faite par les modèles d'exigences réalisés grâce aux cas d'utilisations UML afin de décrire les flux et actions du métier, le diagramme de classes du domaine et participantes pour le modèle de conception et le modèle de codes est réalisé à partir de classes de conception. Pour élaborer ces modèles, nous allons utiliser les diagrammes du Langage de Modélisation Unifié UML 2.0.

Mots clés : ***MDE ; MDA ; modélisation ; cas d'utilisation ; séquences ; classe participante***.

Abstract

This article is about Model Driven Engineering (MDI) for the specification of an Education Fee Payment System (SPFS) in private educational institutions. The perpetual obsolescence of software development frameworks tilts the balance in favor of the sedentary nature of the know-how of business needs, in order to allow educational establishments to capitalize their knowledge on the procedures for collecting school fees; without worrying about the technological choice. In this article we will model a software architecture for the system (SPFS) based on the techniques of Model Driven Architecture (ADM) or MDA (Model Driven Architecture). Allegiance to MDA is made by requirements models realized through UML use cases to describe the flows and actions of the business, the domain and participant class diagram for the design model and the code model is made from design classes. To build these models, we will use the UML 2.0 Unified Modeling Language diagrams.

Keywords: MDE; MDA; modelization ; use case ; sequences; participating class.

Introduction

De nos jours, la sédentarisation du métier des entreprises par rapport aux technologies ou Framework de développement des applications informatiques a donné naissance à la technique d'Ingénierie Dirigée par les Modèles (IDE) ou Model Driven Engineering (MDE) en anglais. Fort de ce cette de disparité, il est évident que séparer les aspects métier des aspects techniques lors de la construction d'une application est une réponse adéquate. Cela permet sans doute de rendre les spécifications métier pérennes, indépendamment des spécifications techniques(Jézéquel et al., 2012). Pour note SPFS, quel que soit les réformes du système éducatif, ses compétences dans les écoles privées restent donc immuables ; ce qui corrobore avec l'IDE dans son exemple d'application de l'approche MDA (Model Driven Architecture)(Jézéquel et al., 2012). Aujourd'hui la pérennisation du savoir-faire et compétences dans ce domaine repose sur l'application de l'approche MDA qui est recommandée par l'OMG (Object Management Group)(A. Harbouche, 2018)(Abdelhedi et al., 2016). Elle spécifie trois niveaux d'abstractions exprimés en termes de modèles dont le premier niveau concerne la capture des exigences logicielles nommé CIM (Computation Independent Model). Le second niveau s'apaisante sur la conception générique sans détails d'implémentation donc les modèles sont indépendants des plateformes (PIM : Platform Independent Model), et le dernier niveau enfin incorpore dans le PIM les contraintes techniques selon les plateformes de développement nommé PSM (Platform Specific Model). Le processus de paiement de frais scolaires dans les établissements privés d'enseignement va de l'inscription des élèves à la perception périodique des contributions d'élève. La maitrise de ce processus influe directement sur la qualité d'enseignement et la motivation du personnel enseignant. Cependant, il peut devenir un motif d'abandons scolaires parmi tant d'autres si sa pérennité est mise en cause (Biyouda et al., 2021). Dans cet article, nous voulons pérenniser des spécifications métier du Système de Paiement de Frais Scolaire (**SPFS),** dont l'épineuse question est celle de savoir « Comment sédentariser les connaissances métiers face à l'évolutivité technologique et quelles en sont les différents vues ? ».

De ce qui précède, cet article vise à répondre à cette problématique en s'attelant dans la première section sur les différents concepts du MDA. Et nous proposons une deuxième section où nous modélisons par approche MDA le métier des paiements de frais dans les écoles privées. Cette dernière section apporte une réponse à la problématique en spécifiant pour notre SPFS les différents modèles (CIM, PIM, PDM et le PSM).

1. Etat de l'art

La mise en cause de méthodologies d'analyse fonctionnelle et systémique dans les années 80 a donné naissance au principe sacré « tout est objet », il faut donc analyser, modéliser, concevoir « l'objet » au lieu de représenter différemment ses aspects. Il est avéré que le paradigme objet a donné naissance au principe de modélisation. C'est ainsi que l'ingénierie logicielle se déploie désormais vers l'ingénierie dirigée par les modèles. Un modèle est donc une représentation abstraite à partir des faits caractérisant un objet du monde réel. Le modèle représente donc un système selon un certain point de vue, à un niveau d'abstraction facilitant par exemple la conception et la validation de cet aspect particulier du système(Jézéquel et al., 2012). À cet égard, la spécificité du domaine de l'ingénierie est que les ingénieurs construisent des modèles d'artefacts qui en général n'existent pas encore (ne serait-ce que parce que le but ultime est de construire ces artefacts). Dans cette optique, l'approche UML adoptée et normalisée est à la base de plusieurs techniques orientées modèle, avec le méta-modèle comme langage de spécification de modèle c'est-à-dire un modèle qui décrit un langage de méta modélisation. Il existe plusieurs méta-modèles dans différents domaines du génie logiciel avec un point centripète le MOF (Meta-Object Facility), ayant pour objectif de définir de méta-modèles. Ce principe sacré est décrit par l'OMG sous forme pyramidale dont le niveau le plus concret se trouve à la base élargie tandis que le MOF est au sommet (Combemale, 2008) (Belaunde et al., s. d.).

1.1. L'approche MDA (Model Driven Architecture)

Motivé par l'essence de promouvoir les bonnes pratiques d'exploitation de modèles, l'OMG définit et promulgue le MDA comme exemple concret de l'IDM. En 2014, les membres ont adopté la dernière version révisée de la spécification(*MDA Specifications | Object Management Group*, s. d.) donnant une définition détaillée de l'architecture dont UML 2.0 est maintenant la version officielle actuelle adaptée aux exigences de MDA(Abbas, 2018b). Cette approche vise à mettre en valeur les qualités intrinsèques des modèles, telles que pérennité, productivité et prise en compte des plateformes d'exécution. Le principe clé et initial du MDA consiste à s'appuyer sur le standard UML pour décrire séparément des modèles pour les différentes phases du cycle de développement d'une application. Cette approche prône l'utilisation de modèles indépendants de toute plate-forme et Technologie(O. Harbouche et al., 2008). L'approche MDA propose un cadre méthodologique et architectural pour le développement de systèmes qui assure la pérennisation des architectures métier en les

découplant des préoccupations technologiques. Ainsi, l'approche MDA se focalise d'abord sur les fonctionnalités et le comportement d'une application, sans se préoccuper de la technologie avec laquelle l'application sera implémentée. L'implémentation de l'application se fait par le biais des transformations des modèles métiers en modèles spécifiques à une plate-forme cible. Le MDA préconise l'élaboration de modèles d'exigence (CIM), d'analyse (PIM) et de conception et un modèle de code (PSM).

1.1.1. ***Le CIM (Computation Independent Model)***

Les exigences ou les besoins du système sont modélisés dans le modèle CIM, il est indépendant de tout système informatique. C'est le modèle métier ou le modèle du domaine d'application. Le CIM permet la vision du système dans l'environnement où il opèrera, mais sans rentrer dans le détail de la structure du système, ni de son implémentation. Il aide à représenter ce que le système devra exactement faire. L'indépendance technique de ce modèle lui permet de garder tout son intérêt au cours du temps et il est modifié uniquement si les connaissances ou les besoins métier changent (Vandeputte, 2018)(Brahim et al., 2020)(Kharmoum, Ziti, Rhazali, & Omary, 2019).

1.1.2. Le PIM (Platform Independent Model)

Le modèle PIM couvre les phases d'analyse et de conception du cycle de développement. Il est indépendant de toute plate-forme et ne contient pas d'informations sur les technologies qui seront utilisées pour déployer le système(*Understanding the Model Driven Architecture (MDA) for Software Development*, s. d.). Le PIM représente la logique métier spécifique au système ou le modèle de conception. Il représente le fonctionnement des entités et des services. Il doit être pérenne et durer au cours du temps. Il décrit le système, mais ne montre pas les détails de son utilisation sur la plate-forme. A ce niveau, le formalisme utilisé pour exprimer le modèle PIM est le langage UML. De plus MDA ne donne aucune indication sur la méthode utilisée pour l'élaboration des PIM.

1.1.3. Le PSM (Platform Specifique Model)

Il est dépendant de la plate-forme technique spécifiée par l'architecte. Le PSM sert essentiellement de base à la génération de code exécutable vers la ou les plates-formes techniques. Il existe plusieurs niveaux de PSM. Le premier, issu de la transformation d'un PIM, se représente par un schéma UML spécifique à une plate-forme. Les autres PSM sont obtenus par transformations successives jusqu'à l'obtention du code dans un langage spécifique(*MDA Specifications | Object Management Group*, s. d.)(Caron et al., 2006).

Notons aussi qu'un autre model existe (PDM) pour contenir les informations pour la transformation de modèles vers une plate-forme en particulier et il est spécifique à celle-ci(Abdelhedi et al., 2016) (Kharmoum, Ziti, Rhazali, & Fouzia, 2019).

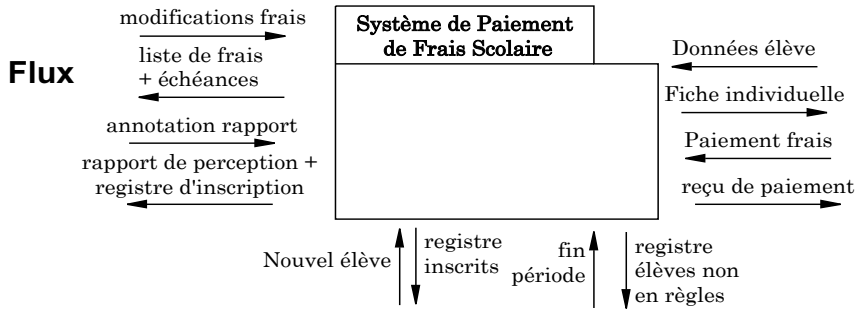
2. Conception du SPFS basé sur MDA

Dans le cadre de cet article, nous allons réaliser quatre modèles du paradigme MDA dont le premier mettra l'accent sur l'identification des modèles du CIM de notre **SPFS** en utilisation le diagramme de contexte statique et dynamique ; le second porte sur la conception du PIM du **SPFS** en utilisant les diagrammes UML (cas d'utilisation, séquences, classes et composants) ; La description de la plate-forme technique pour avoir le PDM constituera le troisième modèle En fin un affinement successif sera fait de nos modèles du PIM pour avoir le PSM. La transformation du PIM vers PSM est réalisée automatiquement grâce à l'outil StarUML qui intègre les principes MDA.

2.1. Elaboration du Computation Indepenent Model (CIM) du SPFS

Dans ce paragraphe nous allons décrire le modèle d'analyse de base du domaine avec un focus sur les concepts métier (élève, reçu, paiement, frais scolaire) et règles de gestion des écoles privées. Suite à la réception d'un calendrier scolaire, les responsables des établissements privés d'enseignement fixent de commun accord avec le comité de parents d'élèves les différents frais et échéances de paiement. Au début de l'année scolaire, chaque élève (ancien ou nouveau) confirme son inscription par le paiement d'une somme d'argent (Biyouda et al., 2021). Le service d'inscription dresse un rapport d'inscrit par classe tout en spécifiant les effectifs (Garçons et Filles) et le soumet aux responsables de l'établissement. A la fin d'une période (mois, trimestre) chaque élève verse sa contribution selon les modalités fixées comme indiqué dans la **figure N°1** suivante où nous représentons un modèle de flux(Tsanga, 2019) afin de se focaliser sur les interactions et entre le système et son environnement. Le système est représenté dans un package portant dans le compartiment haut le son nom. Le système **SPFS** émet des messages suivants : Listes des élèves inscrits ; Fiche individuelle de paiement (relevé de paiement) d'un élève ; Liste des élèves par classes ; Rapport succinct de paiement / d'inscription ; Liste frais fixé ; Reçu de paiement ; Liste des élèves non en règles dans une période donnée. Les flux entrants (reçus) dans le système sont : Fixation de frais scolaire ; Inscription élève ; Annotation rapport de paiement / d'inscription ; Donnée élève ; Paiement frais ; Fin période

Figure N° 1: *Diagramme de flux du SPFS*

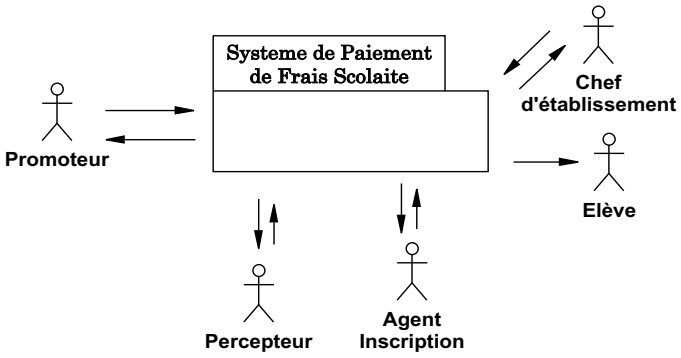


Source 1—auteur

2.1.1. Modélisation du contexte

Par rapport à l'organisation (domaine d'étude), la modélisation du contexte détermine les rôles à jouer par les travailleurs (acteurs) qui participent dans ce processus de paiement de frais comme le démontre la ***Figure n°2*** suivante où le rôle de « *Percepteur* » reorésente un employé ayant dans ses responsabilités la perception de frais versé par les élèves pour le compte de l'établissement et en fait rapport au chef d'établissement ; le rôle du « *Promoteur* » désigne une personne physique ou morale ayant dans ses responsabilités la gestion de son école ; le rôle du « *Chef d'établissement* » pointe un agent ayant la responsabilité sur le déroulement des enseignements, contrôle de paiement, sensibilisation sur le paiement de frais scolaire ; le rôle « d'*Elève* » représente une personne physique qui vient pour payer le frais ; Et enfin le rôle « d'*Agent inscription* » désigne un travailleur interne ayant la responsabilité d'inscrire ou réinscrire les élèves au début d'une année scolaire.

Figure N° 2*:Diagramme de contexte d'organisation du SPFS*



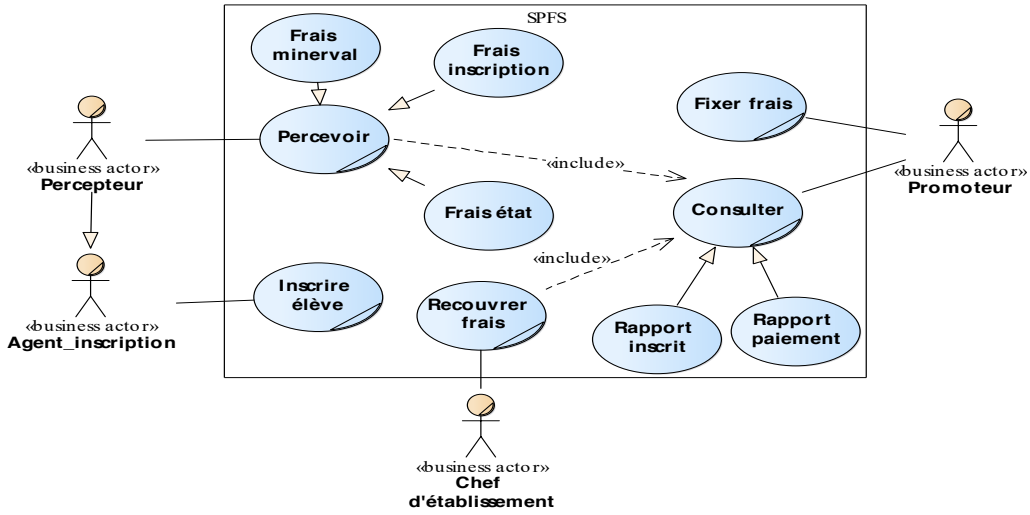
Source 2—conçu par nous-mêmes

2.1.2. Identification des cas d'utilisation

Partant du modèle de contexte réalisé précédemment, nous allons identifier les fonctionnalités nécessaires aux utilisateurs du système. Par définition un cas d'utilisation représente une

d'actions réalisées par le système, et le lien entre ces séquences d'actions est précisément l'objectif métier de l'acteur(Roques, s. d.). Les cas d'utilisations de notre SPFS sont les suivants : *Fixer frais, Percevoir (Frais d'inscription, minerval ou frais de l'état), Inscrire élève, Recouvrer frais*, *Consulter* (rapport paiement ou d'inscription). Chacun de ces cas d'utilisation est relié à un seul acteur par un lien d'association symbolisé par un trait plein pour désigner l'acteur principal de ce cas. L'acteur *Promoteur* est responsable de deux cas d'utilisations (*Fixer frais et Consulter*) ; l'acteur *Percepteur* est responsable lui de la perception de différents frais et il généralisé en agent inscription par une flèche orientée plaine, avec possibilité d'inscrire des élèves. L'acteur *Agent-inscription* est responsable d'inscription des élèves ; le chef d'établissement lui est responsable de recouvrement de frais (sensibilisation des enfants au paiement). Par ailleurs, les cas d'utilisation « *Percevoir frais* » et « *Recouvrer frais* » inclus le cas d'utilisation « *consulter rapport* » cela se justifie du fait que généralement pour connaitre l'état de paiement de chaque élève le *Chef d'établissement* doit jeter consulter le rapport de *paiement*. **La figure N°3** suivante représente le diagramme de cas d'utilisation de notre ***SPFS***.

Figure N° 3:*Diagramme de cas d'utilisation*



Source : conçu par nous-mêmes avec l'outil Enterprise Architect

2.1.3. Description textuelle de cas d'utilisation

La compréhension du déroulement d'un scénario de cas d'utilisation passe par spécification textuelle décrivant toutes les interactions de l'acteur et du système du début à la fin. La fin d'un cas d'utilisation est tributaire au résultat escompté par l'acteur principal en suivant un chemin de succès. Cependant, certaines situations inhabituelles font à ce que le cas d'utilisation n'arrive pas à une fin normale.

❖ ***Cas d'utilisation « fixer frais »***

Ce cas d'utilisation va permettre au promoteur de fixer les différents frais à payer conformément aux accords conclus avec le comité de parents. Le scénario suit les étapes reprises dans le ***Tableau N°1*** suivant :

Tableau N° 1 : *Scénario nominal du cas d'utilisation « fixer frais »*

1. Le Promoteur recherche la liste de frais recommandé.	1.1. Le système affiche la liste de frais autorisés ;
2. Le Promoteur sélection un frais donné.	2.1. Le système affiche les détails sur ce type de frais.
3. Le Promoteur saisit pour chaque type de frais le montant annuel.	3.1. Le système compare le montant saisi avec celui conclu dans le contrat avec les parents
4. Il spécifie le montant et échéance (mensuel ou semestriel ou encore annuel).	4.1. Le système avalise le délai
5. Le promoteur valide la liste de frais et publie	5.1. Le système prend en charge la liste de frais.

Source : auteur

- ***Flux alternatifs***

3-1a. *[Montant non approuvé] : dans ce cas le processus reprend l'étape 1-5 du scénario nominal.*
4a. *[Délai inacceptable] : le processus reprend l'étape 4 à 5 du scénario nominal. La description formelle de ce scénario est consolidée dans le diagramme de séquences de la* ***Figure N°4.***

❖ ***Cas d'utilisation « Inscrire élève »***

Ce cas d'utilisation permettre à l'Agent inscription de réinscrire des anciens élèves ou nouveaux élèves selon la disponibilité des effectifs dans chaque classe. Le cheminement normal est donné dans le ***Tableau N°2*** *suivant :*

Tableau N° 2 : *Scénario nominal du cas d'utilisation « Inscrire élève »*

1. *L'Agent inscription demande au système d'inscrire un élève*	1.1. *Le système affiche la liste de classes et option disponibles*
2. *L'Agent inscription choisi la classe pour l'élève*	2.1. *Le système vérifie la disponibilité de places dans la classe choisie*
3. *L'Agent inscription renseigne les informations administratives de l'élève (nom, postnom, prénom, lieu et date de naissance, nom père, mère, adresse parent, etc…)*	3.1. *Le système vérifie si l'élève existe déjà avec ces mêmes données*
4. *L'Agent inscription rempli la fiche de scolarité de l'élève*	4.1. *Le système affiche les conditions et fournitures à pourvoir*
5. *L'Agent inscription rempli la fiche de paiement individuelle pour cet élève*	5.1. *Le système affiche les détails de paiements (mensuel, trimestriel, annuel)*
6. *L'Agent inscription valide l'inscription*	6.1. *Le système affiche le reçu d'inscription + journal + esquissons*

Source : auteur

- ***Scénarii alternatifs***

2a. [Exception1 : effectif_atteint] *: Dans ce cas le processus se termine en échec.*

3-6a : [Nouvel_élève] *: dans ce cas, le système averti l'Agent inscription de l'obligation de dépôt du dossier, si oui alors le processus poursuit l'étape 5 à 7 du scénario nominal. Sinon le processus se termine en échec. La description formelle de ce scénario est représentée dans le diagramme de séquences de la* ***Figure n°5.***

❖ ***Cas d'utilisation « Percevoir frais scolaire »***

Avec ce cas d'utilisation, le *Percepteur* veut pouvoir encaisser les paiements de frais versés par chaque élève selon le montant et les modalités de paiements fixés par le promoteur. Dans le Tableau N03 suivant, nous donnons l'enchaînement nominal des actions du *Percepteur*.

Tableau N° 3 : Scénario nominal du cas d'utilisation « Percevoir frais »

1. Le Percepteur vérifie si l'élève existe dans registre d'inscription	1.1. Le système affiche le résultat de la vérification
2. Le Percepteur recherche / crée la fiche de paiement individuelle	2.1. Le système affiche le résultat trouvé
3. Le Percepteur sélection le type de frais	3.1. Le système affiche le montant et l'échéance fixé pour ce frais
4. Le Percepteur rempli les informations nécessaires à savoir : - Les coordonnés de l'élève (numéro d'ordre, nom, post-nom, prénom, classe, niveau) - Les coordonnés de paiements (mois, type de frais, montant versé, la date de paiement, signature)	4.1. Le système vérifie la correspondance entre le taux fixé et le montant payé
5. Le Percepteur met à jour la fiche individuelle de paiement	5.1. Le système affiche la situation de paiement de l'élève
6. Le percepteur valide le paiement	6.1. Le système affiche le reçu

Source : auteur

- **Scénarios alternatifs**

1-2. a. [*Elève inconnu*] : dans ce cas le processus se termine en échec en attente d'inscription

2a. [*Fiche non trouvée*] : Dans ce cas le système averti le Percepteur que l'élève ne dispose d'aucune fiche de paiement individuelle et lui invite d'en crée une à l'étape 2 création.

4.a. [*montant insuffisant*] : le processus se poursuit à l'étape 4-6 du scénario nominal

La description formelle de ce scénario est représentée dans le diagramme de séquences de la ***Figure N°6.***

2.2. Elaboration du Plateform Independent Model (PIM) de notre SPFS

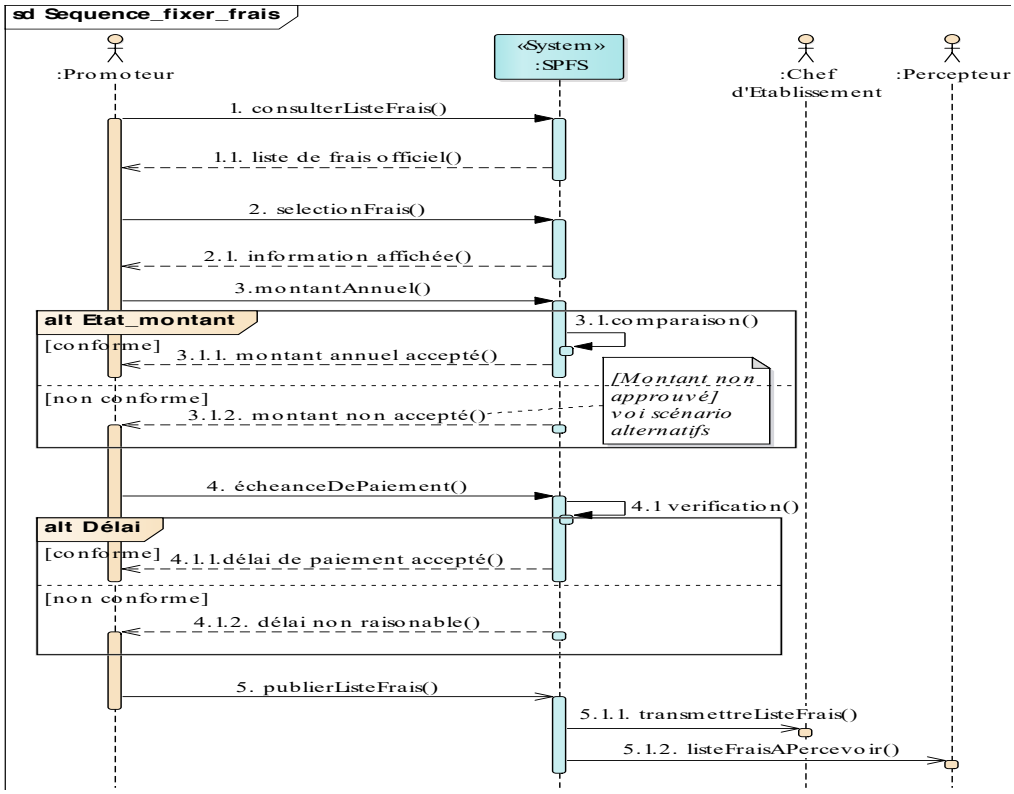
Le modèle de conception de notre système de paiement de frais scolaire s'articule premièrement sur les besoins des utilisateurs appelés cas d'utilisation du système ou use case en anglais. Ensuite chaque cas d'utilisation sera détaillé par une spécification textuelle pour améliorer le contenu informatif des utilisateurs et développeur. Ces scénarios sont consolidés

par de diagrammes de séquences UML. Le déroulement d'un cas d'utilisation entraine la manipulation des structures d'informations contenues dans les documents du domaine. Ces documents et concepts du domaine sont modélisés sous forme de classes UML afin d'avoir une vision statique des informations manipulées.

2.2.1. Consolidation des descriptions textuelles de cas d'utilisation

Dans ce paragraphe il sera question de modéliser chaque scénario sous-forme de séquences de messages échangés entre l'acteur et le système. Le digramme de séquence UML est bien indiqué pour ce genre de situations. Dans le diagramme de la figure 4 suivante, l'acteur *Promoteur* en face du système SPFS pour réaliser le scénario du cas d'utilisation « *Fixer frais* » décrit dans la section précédente. Le message *3 « montantAnnuel() »* déclenche une action interne dans le système consistant à la comparaison du montant fixer avec celui du consensus du comité de parent, ce qui donne lieu à deux alternatifs du fragment « **alt »** : 3.1.1 « *conforme* » ou 3.1.2. « non-*conforme* » avec des actions à exécutées. La fin du scénario au point 5 « *publierListeFrais* » donne lieu à deux actions internes au système (5.1 et 5.2) avec comme messages « *transmissionListeFrais* » auprès du *chef d'établissement* et au près du *Percepteur.*

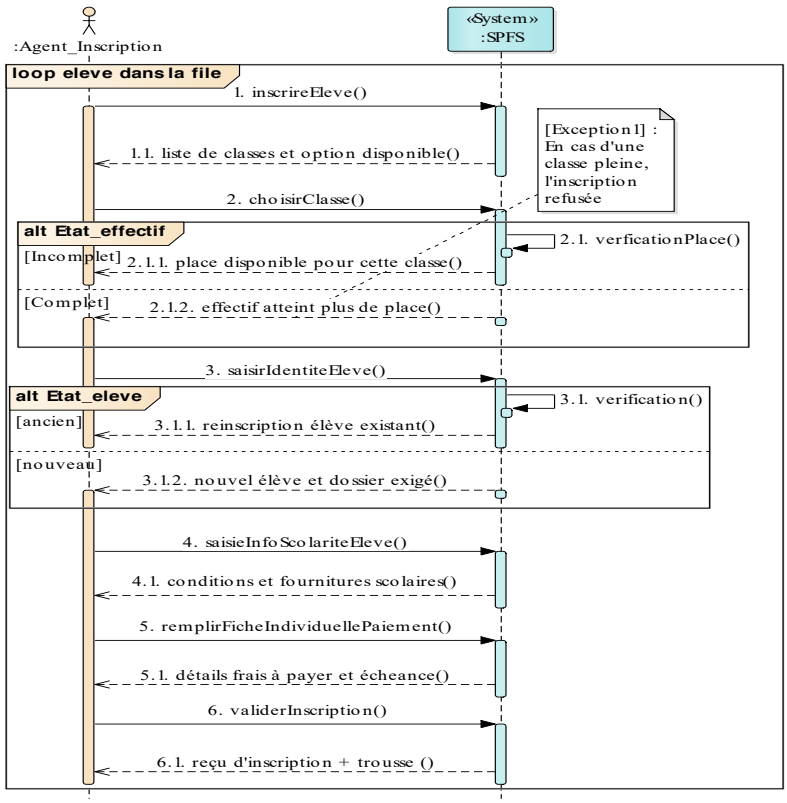
Figure N° 4 : *diagramme de séquences système de fixation frais scolaire*



Source : conçu par nous-mêmes avec l'outil Enterprise Architect

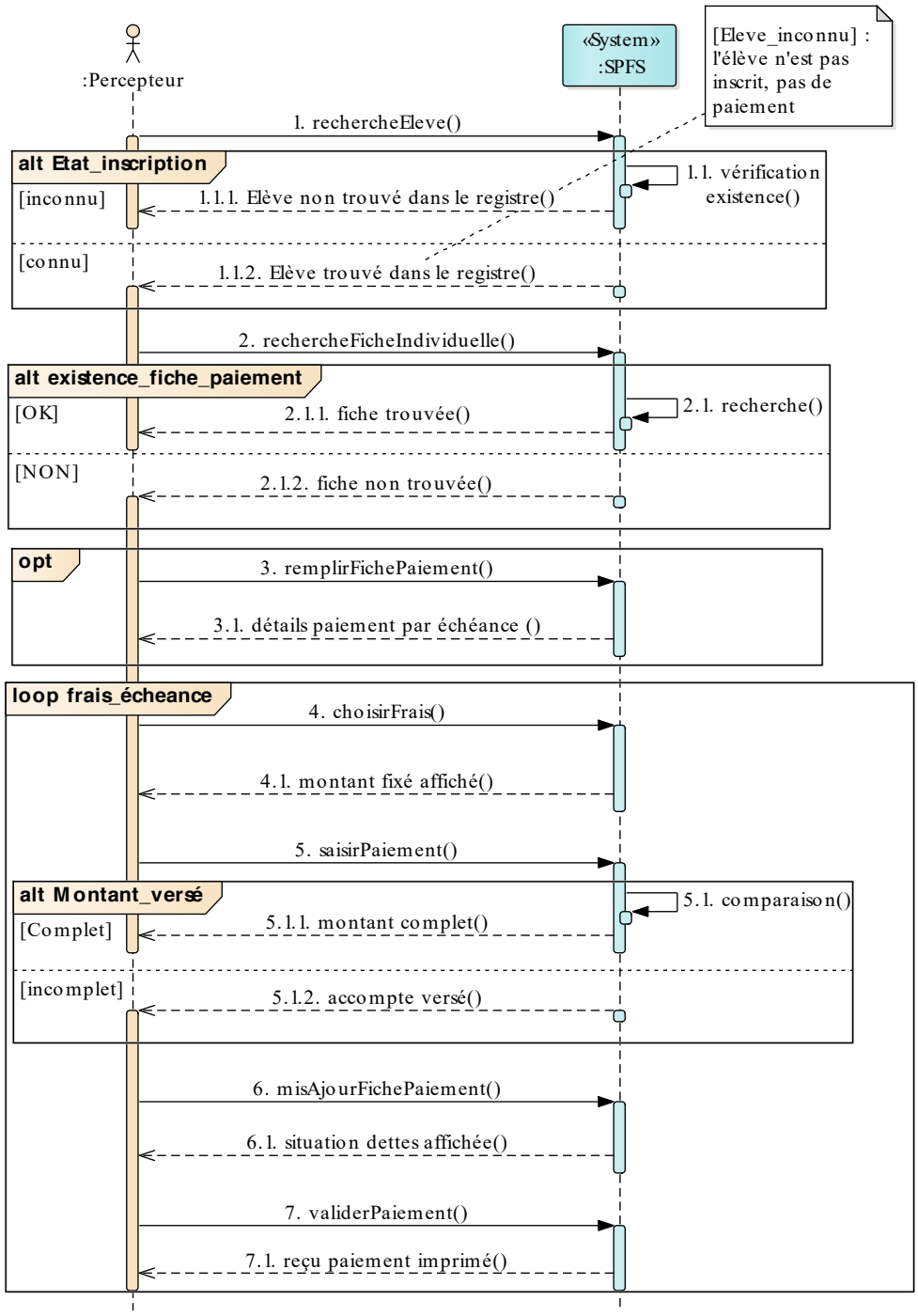
Suite à la réception du barème de frais, *l'Agent-inscription* procède sur instruction du à l'inscription et réinscription respectivement pour les nouveaux élèves et anciens élèves. Dans le diagramme de séquences de la ***Figure 5*** suivant, il détaillé le scénario d'inscription d'élève où l'acteur *Agent-inscription* est en face du système « SPFS » pour inscrire. Le message *1.* « *inscrireEleve()* » est placé dans une boucle « ***loop*** » qui indique la répétition des tâches d'inscriptions selon le nombre d'élèves dans la file d'attente. Par ailleurs, les messages 2, 3 et 4 engendrent chacun un message à soi-même au système *« self message »* donnant ainsi lieu à deux alternatifs. La fin de ce scénario produit pour chaque inscrit un reçu d'inscription et une trousse scolaire. Une fois acquise, l'inscription donne droit à l'élève de participer aux cours et un devoir de payer les frais selon les échéances fixées. Ce qui nous amène au scénario du cas d'utilisation « *Percevoir frais »* de la ***Figure 6*** où l'acteur *Perceteur* est devant le système « SPFS » pour percevoir le frais scolaire. La première interaction « *1. rechercheEleve »* donne lieu à deux messages de retour mutuellement exclusifs « ***alt »***, qui généralement l'élève a déjà prix l'inscription sinon il doit rentrer le faire avant de poursuivre un paiement quelconque dans le système. A la fin de ce scénario de perception de frais, le *Percepteur* doit mettre à jour la fiche individuelle de l'élève.

Figure N° 5 : *Diagramme de séquences paiements inscription élève*



Source : auteur

Figure N° 6 : *Diagramme de séquence perception frais*



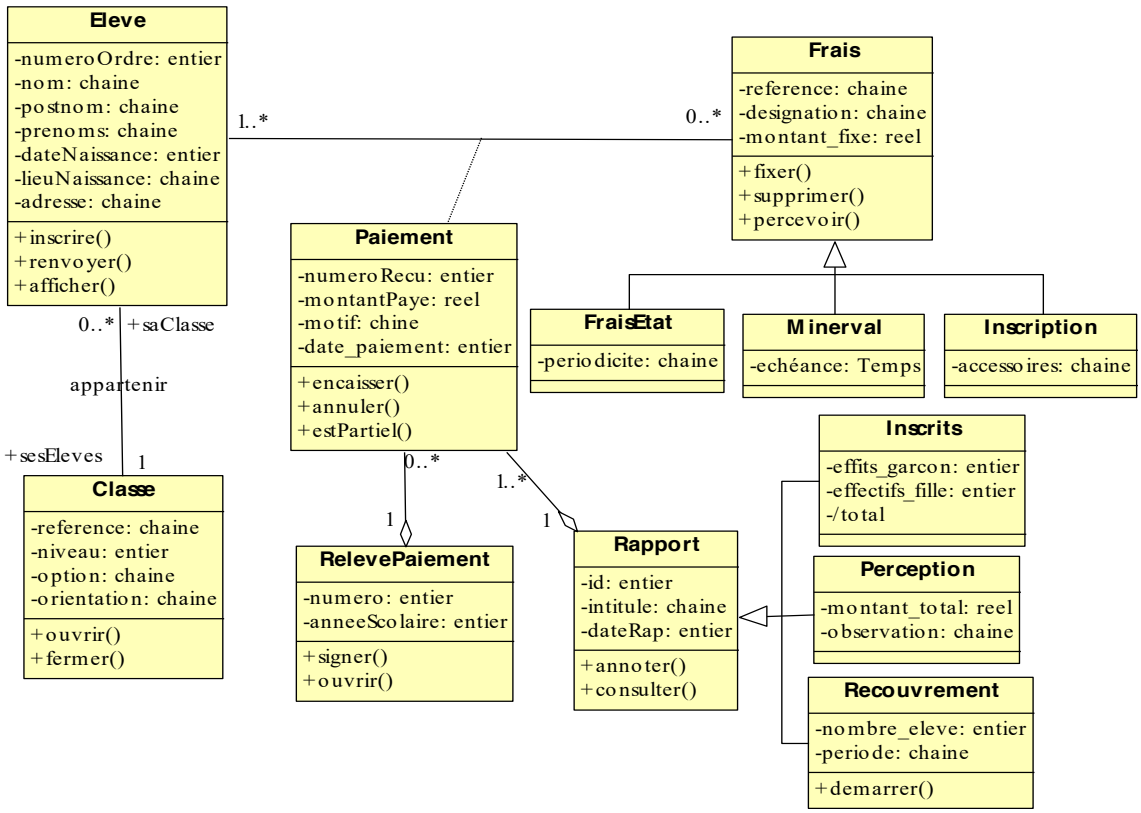
Source : conçu par nous-mêmes avec l'outil Enterprise Architect

2.2.2. Diagramme de classe du domaine

Le diagramme de classes de la **Figure N° 7** suivante fournit une vue globale des structures d'informations manipulées dans le système en spécifiant ses classes et les relations entre elles(Abbas, 2018a) (CROZAT, 2018). Chacun de concepts du domaine représente une classe

conceptuelle du domaine. Chacune peut être décrite par un ou plusieurs attributs et des opérations que peut réaliser ces instances. Un attribut est une information élémentaire qui caractérise une classe et dont la valeur dépend de l'objet instancié(CROZAT, 2018)(Roques, 2008). L'ensemble des valeurs attributs forme l'état de l'objet de cette classe tandis que les opérations constituent le comportement. En ce qui est de notre SPFS, les concepts du domaine manipulé lors de la description textuelle de cas d'utilisation sont les suivants : *Elève, Frais, Paiement, Classe, FraisEtat, Minerval, FraisInscription, RapportInscription, RapportPerception, RapportRecouvrement et Relevé de paiement*. Ces concepts sont modélisés sous forme de classes UML contenant les attributs et opérations pertinents, avec des types primitifs sans tenir compte des contraintes technologiques. La classe Eleve est en lien direct avec la Classe. Une classe définit la structure commune d'un ensemble d'objets qui partagent les mêmes caractéristiques en termes de fonctionnalités, contraintes et sémantique(Booch et al., 2003).

Figure N° 7 : *Diagramme de classe du domaine*

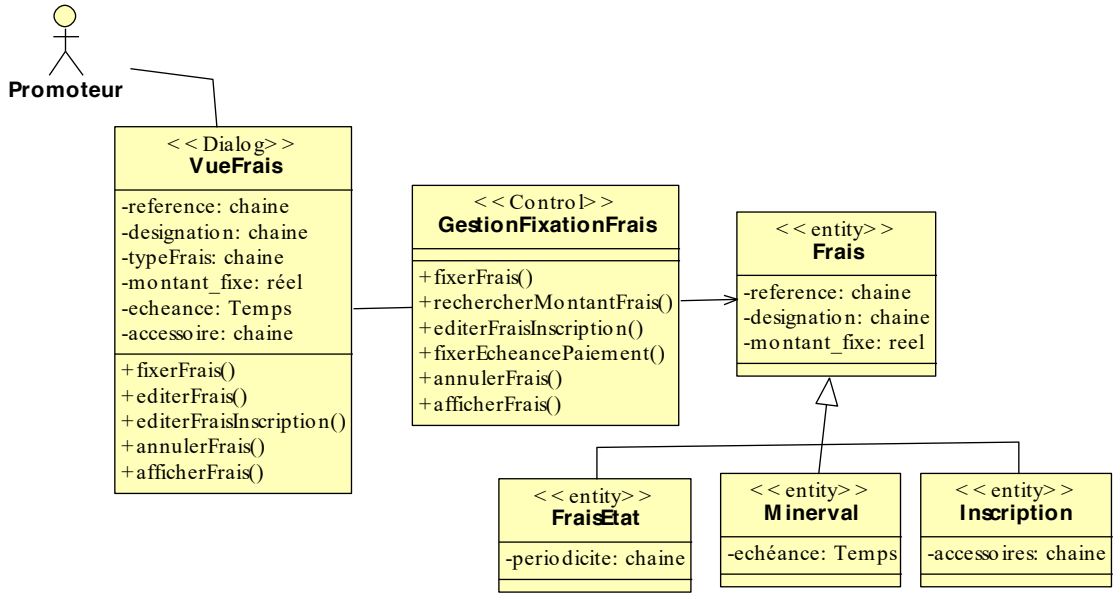


Source : auteur

2.2.3. Identifications des classes participantes du SPFS

Outre les classes du domaine identifiées précédemment, les classes participantes sont réalisées cas d'utilisation par cas d'utilisation et interviennent dans les interactions de l'application. Les trois principales classes d'analyse et leurs relations(Roques, 2008) : les dialogues, les contrôles et les entités. Ces classes sont particulièrement importantes car ils font la jonction entre les cas d'utilisation, la maquette et les diagrammes de conception logicielle. Dans le paragraphe suivant, nous allons modéliser ces trois classes d'analyse de notre SPFS où les concepts du domaine deviennent des entités « *entity* » en anglais contenant seulement les attributs, les règles de gestion constituent une logique applicative stéréotypé « Control » contenant seulement les opérations et les classes d'interface graphiques ou formulaire stéréotypé « *dialog* ». Pour chaque cas d'utilisation nous plaçons l'acteur principal devant le formulaire contenant des attributs et les opérations. Dans la ***Figure N° 8***, l'acteur *Promoteur* est relié au formulaire « *VueFrais »* lequel est relié au contrôleur « *GestionFixationFrais »*, en son tour relié à l'entité « *Frais »*.

Figure N° 8 : *Diagramme de classes participantes du promoteur*



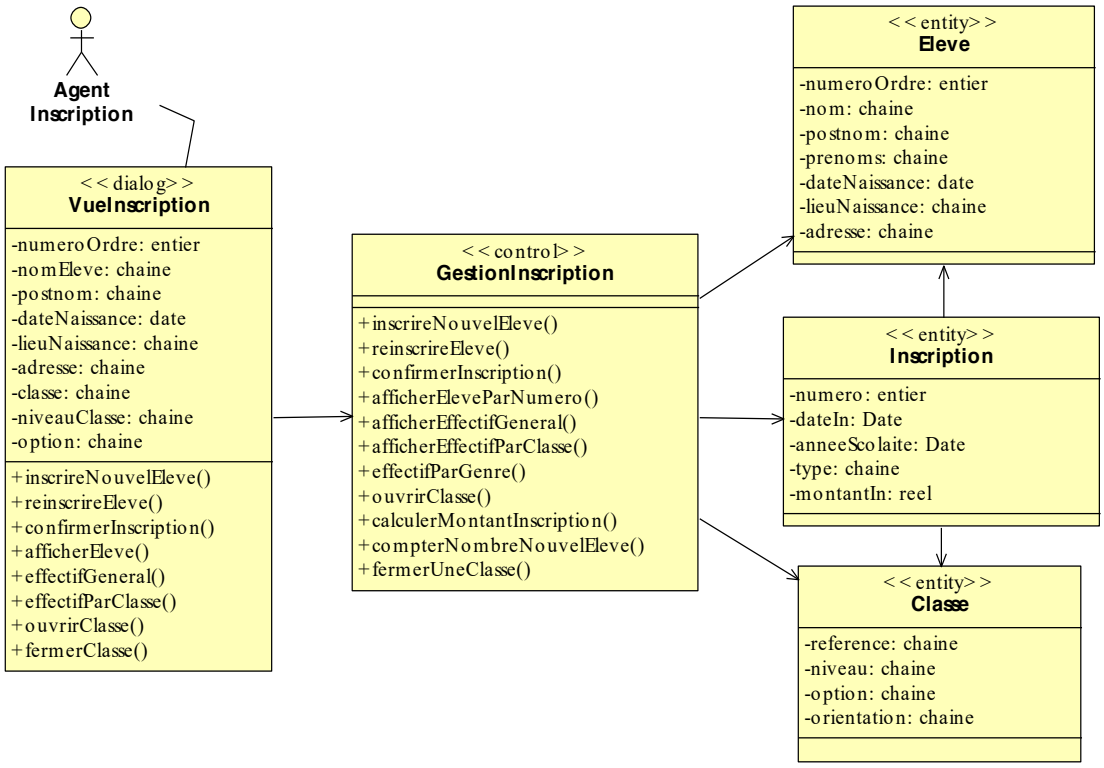
Source : conçu par nous-mêmes avec l'outil StarUML

Pour inscrire un élève, l'acteur *Agent-Inscription* se connecte au formulaire « *VueInscription* » relié au contrôleur *« GestionInscription »* lequel relié en son tour aux entités (Eleve, Classe et Inscription). *Le contrôleur « GestionInscription »* contient les opérations suivantes : *fixerFrais(), recherchcerMontantFixe(), editerFraisInscription(),*

fixerEcheancePaieent(), annulerFrais(), afficherFrais(). Les classes participantes du cas d'utilisation *« Inscrire élèves »* sont reprises dans le digramme de la ***Figure N° 9***.

Dans le diagramme de la ***Figure N° 10***, l'acteur *Percepteur* se trouve en face d'une classe dialogue « VuePerceptionFrais » où il renseigne les informations administratives de l'élève, le montant versé et le type de frais payé. Toutefois, le *Percepteur* peut se connecter au formulaire de modification « *VueModification* » en cas de modification d'un paiement ou de suppression « *vueSuppression* » dans le cas d'annulation d'un paiement déjà enregistré. A la validation de l'un de formulaire, les données sont soumises au contrôleur « *GestionPerception* » pour un contrôle avant de les acheminer vers les entités respectives. La mise en œuvre de cas d'utilisation fait intervenir les entités suivantes : La classe ***Elève*** concernée par le paiement, le ***Frais*** payé par l'élève, la classe ***Paiement, le Rapport.***

Figure N° 9 : *Diagramme de classes participantes du cas d'utilisation « Inscrire élève »*



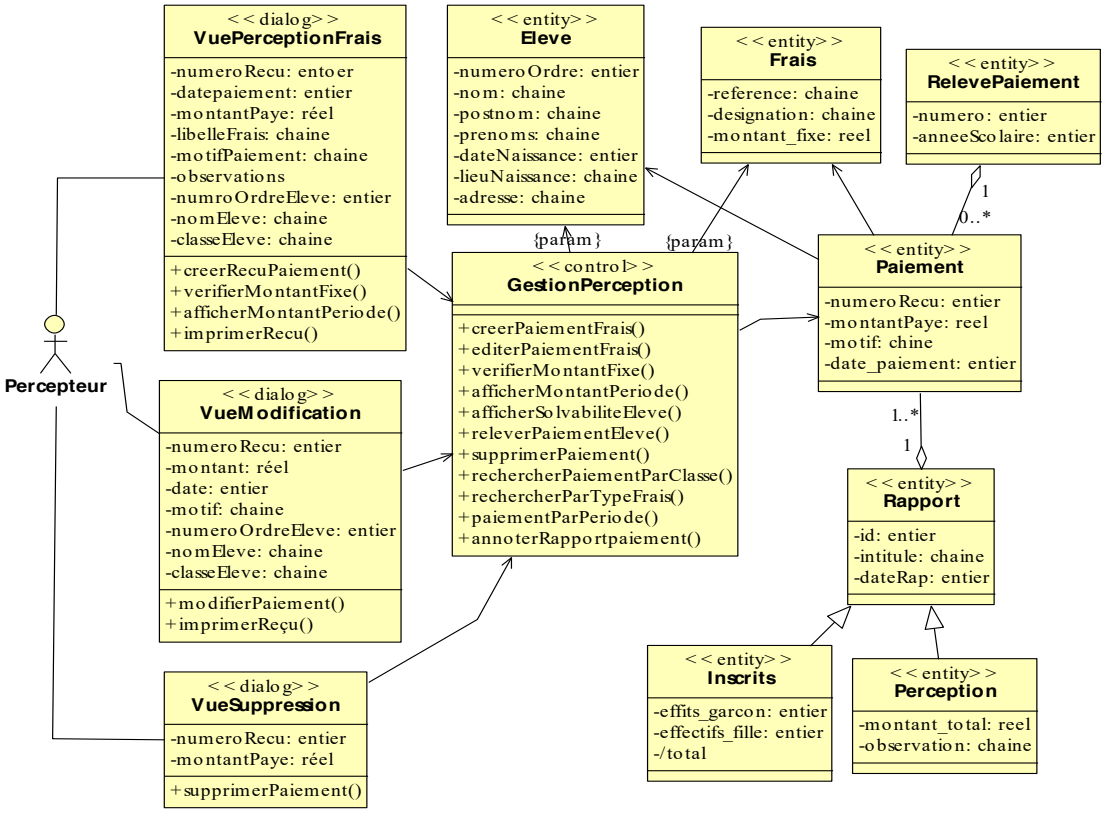
Source : conçu par nous-mêmes avec l'outil StarUML

2.3. Elaboration du Plateforme Dependent Model (PDM) de notre SPFS

Dans le diagramme de navigation ***Figure N° 11*** suivant, nous décrivons l'architecture technique qui sera appliquée au PIM pour obtenir le PSM. Par définition, une plate-forme est un ensemble de technologies, y compris les fonctionnalités fournies par le biais d'interfaces et

de modèles d'utilisation(*Understanding the Model Driven Architecture (MDA) for Software Development*, s. d.). Les plates-formes peuvent être génériques (comme le traitement d'objets ou de lots), spécifiques à la technologie (comme la plate-forme Java) ou spécifiques au fournisseur (comme la plate-forme Microsoft .NET). Un modèle de plate-forme décrit une plate-forme, y compris ses pièces et les services qu'elle fournit(*Understanding the Model Driven Architecture (MDA) for Software Development*, s. d.). Le schéma global sur la perception de frais oblige le *Percepteur de* frais va comprendre l'ensemble des pages, frames et actions principales du cas d'utilisation *Percevoir frais*. Le schéma ainsi obtenu (***Figure N° 11)*** est très intéressant, car il fournit sur une seule page les règles fondamentales de déplacement dans la partie graphique de SPFS. Le *Percepteur* commence par la page d'accueil où deux groupes d'options (*recherche et liste de frais à payer disponibles*). Dans le premier bloc, le *Percepteur* peut consulter un paiement individuel ou par classe. Le deuxième groupe d'options contient le cadre pour le frais de l'Etat, un cadre pour le frais de participation, frais connexe. Ce cadre est lié au cadre de paiement contenant les options (*paiement, apurement, recherche)* qui aboutit dans le connecteur du Reçu.

Figure N° 10 : *Diagramme de classes participantes du percepteur*



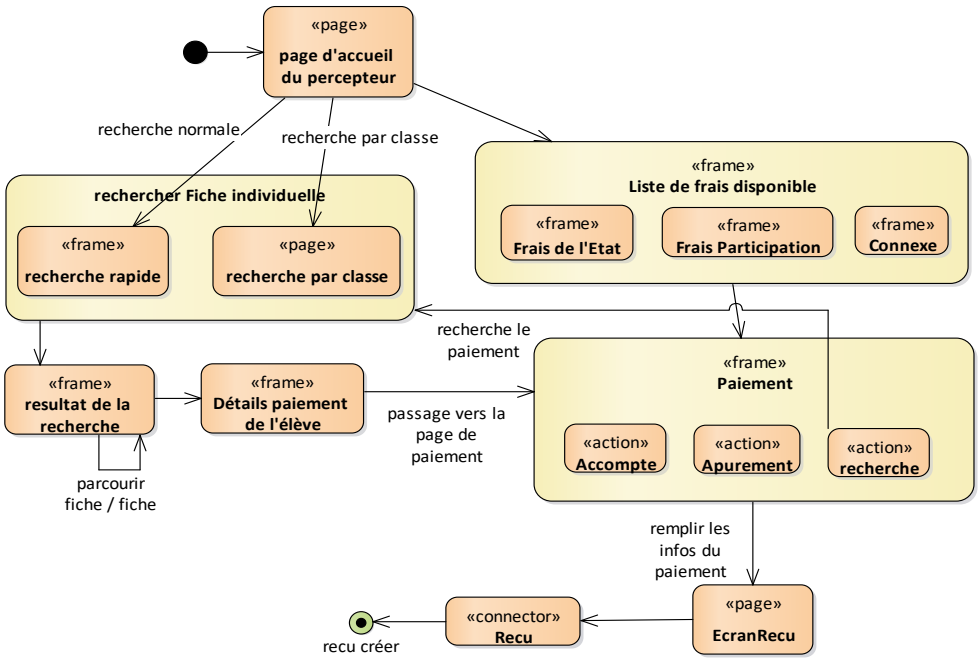
Source : auteur

2.4. Plateforme Specific Model PSM de notre SPFS

Contrairement au modèle d'analyse ou de conception, le modèle de code est lié à une plateforme

d'exécution(Caron et al., 2007). Il sert principalement à faciliter la génération de code. Les PSM peuvent être obtenus par application de profils UML, c'est-à-dire l'adaptation d'UML à un domaine particulier, ou par l'utilisation de modèles de plates-formes (PDM) lors de la transformation du PIM. Etant liés à une plate-forme d'exécution, les modèles de code n'ont pas pour vocation d'être pérennes(Brun, 2010). En partant du modèle de classes réalisé dans le PIM nous allons affiner et compléter les diagrammes de classes participantes obtenus précédemment. Ajouter ou préciser les opérations dans les classes (un message ne peut être reçu par un objet que si sa classe a déclaré l'opération publique correspondante). Nous allons également ajouter des types aux attributs et aux paramètres et retours des opérations. Enfin nous allons affiner les relations entre classes : associations (avec indication de navigabilité), généralisations ou dépendances.

Figure N° 11 : *Diagramme de navigation du cas d'utilisation du Percepteur*



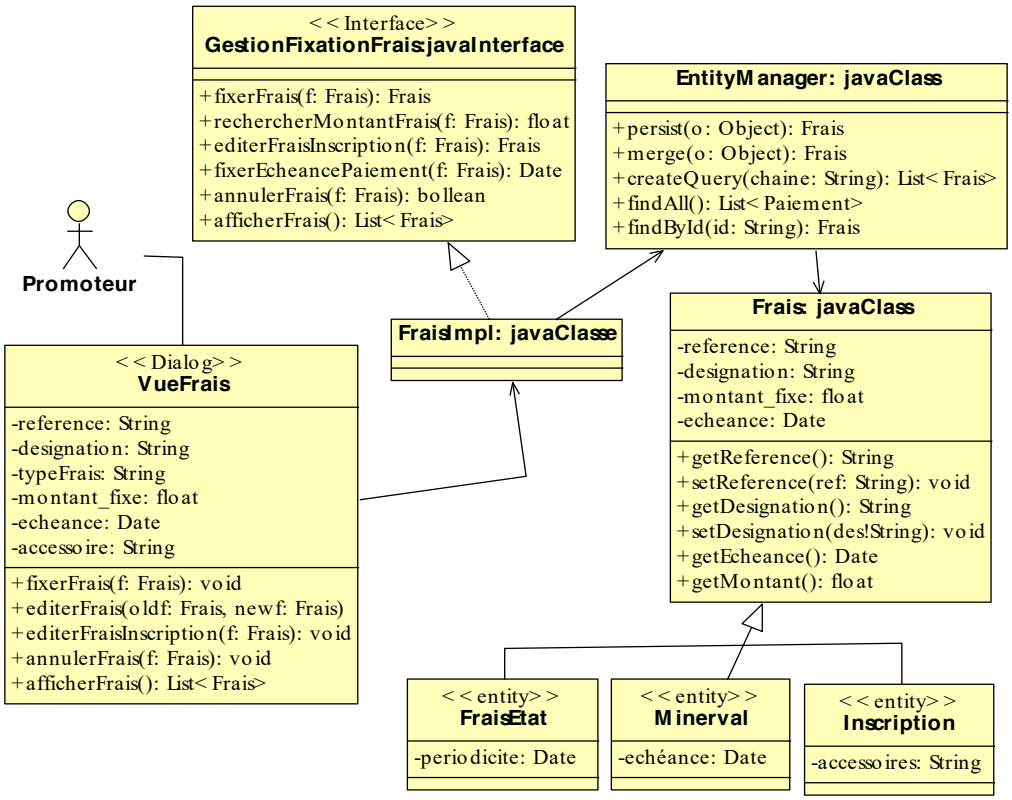
Source : auteur

2.4.1. Modèle PSM du cas d'utilisation « Fixer frais »

Le modèle PSM du cas d'utilisation « *Fixer frais* » s'est ajouter une interface « GestionFixationFrais » du type « *javaInterface* ». Cette interface contient les déclarations

des opérations sur la fixation de frais et est implémentée par la classe technique « FraisImpl » du type javaClass. La persistance des données est réalisée grâce au Mapping Objet Relationnel(Goman, 2012) symbolisé par la classe « EntityManager » laquelle est reliée aux entités (*Frais, Classe)*. Le diagramme ainsi obtenu est représenté dans le ***Figure N° 12*** suivante :

Figure N°12 : *Modèle PSM du cas d'utilisation Fixer frais*

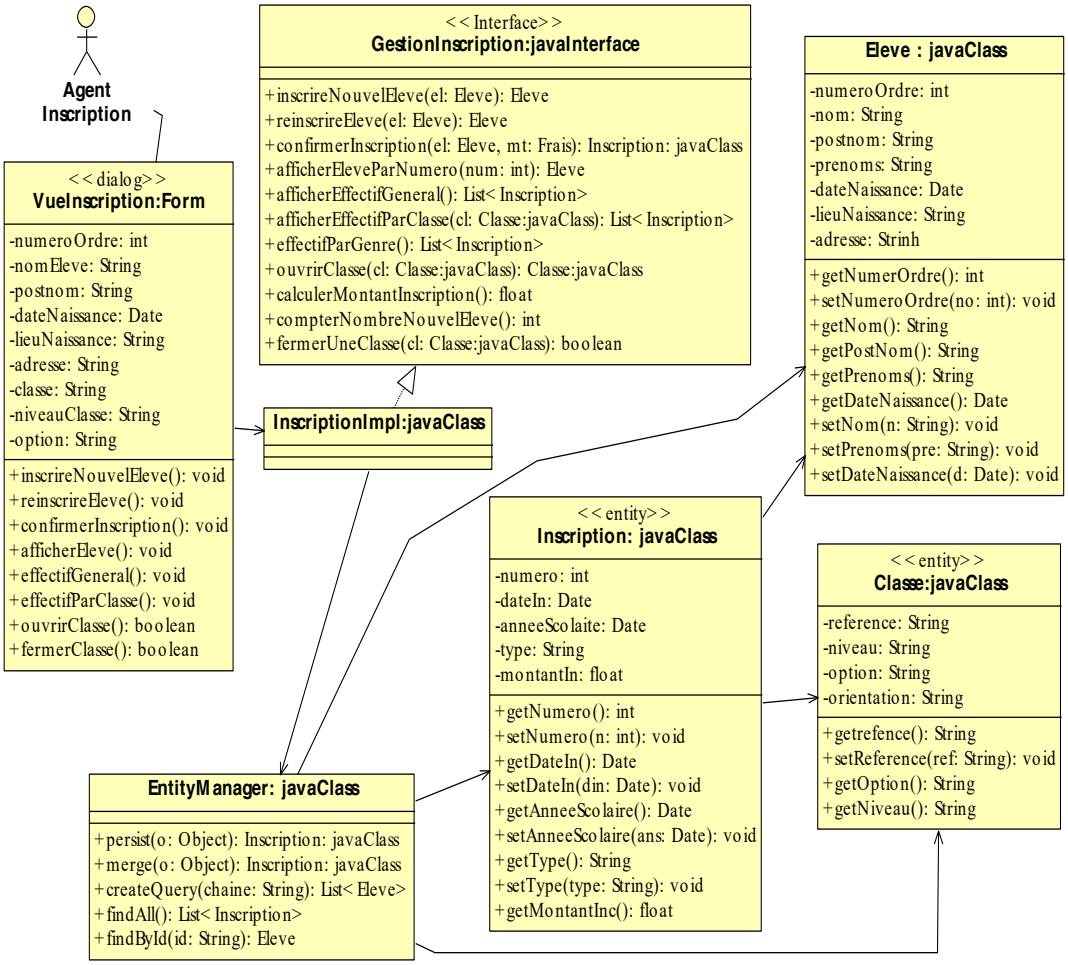


Source : auteur

2.4.2. Modèle PSM du cas d'utilisation « Inscrire élève »

Le schéma obtenu à la ***Figure 13*** représente le modèle de codes du cas d'utilisation « *Inscrire* élève ». L'interface « *GestionInscription »* contient les déclarations des opérations de gestions des inscrits dont chacune contient une interface précise (types attendues, types de retour). Cette interface est rendue opérationnelle par une classe java « InscriptionImpl.java » qui en son tour relié à « *EntityManager.java* » pour la persistance de données des entités (*Eleve, Inscription, Classe*). Les types d'interfaces graphiques sont également spécifiés pour chaque attributs et actions.

Figure N°13: *Modèle PSM du cas d'utilisation «Inscrire élève »*

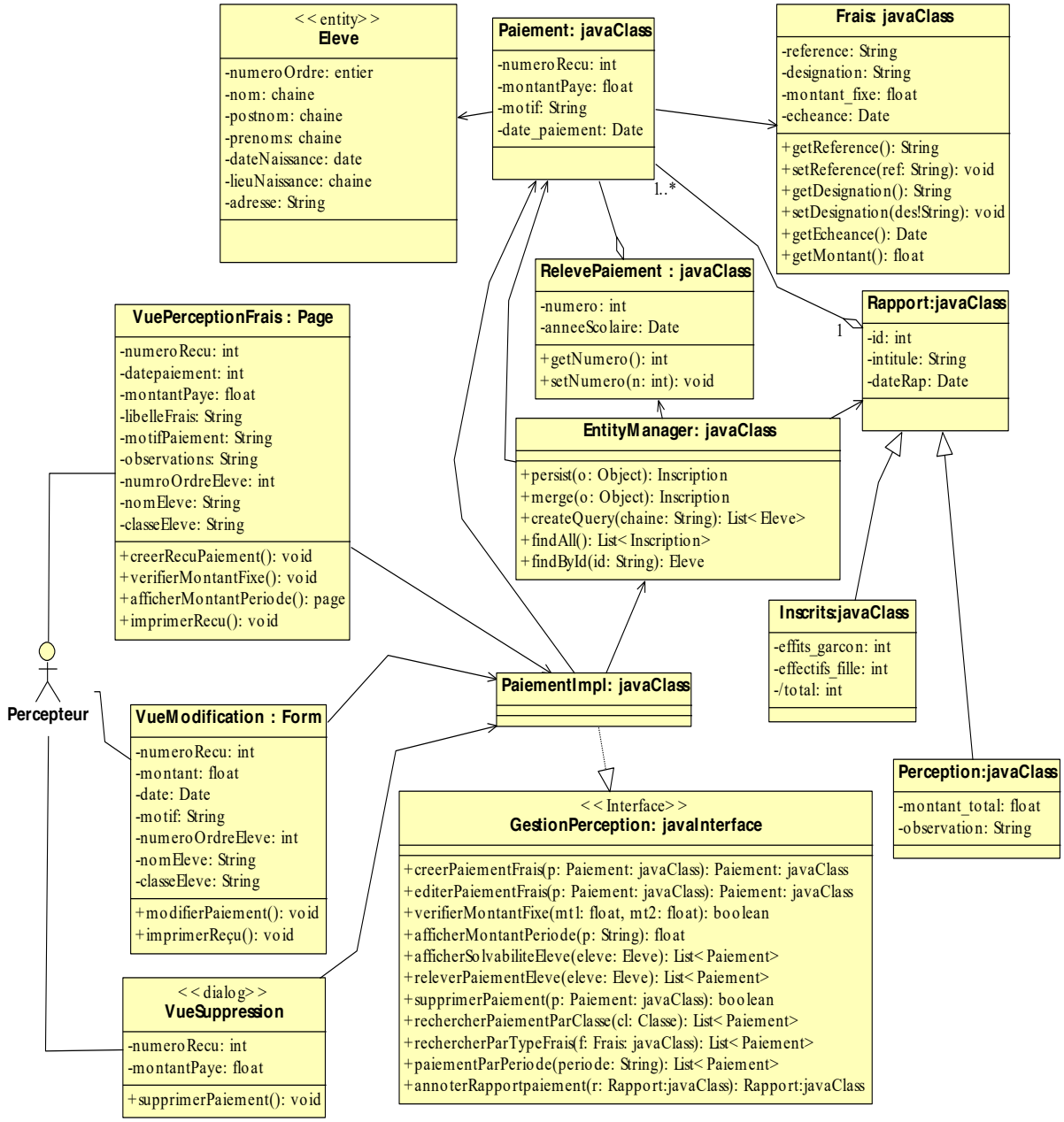


Source 3—auteue

2.4.3. Modèle PSM du cas d'utilisation « Percevoir frais »

Le modèle de codes obtenu dans la ***Figure N°14 suivante*** contient en plus de l'interface graphique et des entités, une interface java « GestionPerception.java » contenant le comportement abstrait de la gestion de perception. Pour la rendre opérationnelle, cette interface est implémentée par la classe « *PaiementImpl.java* » ; l'implémentation de cette interface instancie en son tour la classe technique « EntityManager.java » qui s'occupe de la persistance des entités (*Paiements, RelevePaiement, Eleve, Frais*).

Figure N° 14 : *Modèle PSM du cas d'utilisation Percevoir frais*



Source : auteur

Conclusion

De nos jours, l'évolution technologique a modifié le mode de paiement de frais scolaires dans les écoles, cependant le processus suit son cours normal de l'inscription d'élèves au dit paiement. En RDC, Elle est au centre d'intérêts des promoteurs d'écoles, de parents d'élèves et du gouvernement qui priorise la gratuite d'enseignement.

Cet article avait pour objectif de démontrer comment le savoir-faire métier des établissements privés d'enseignement fonctionne face aux innovations technologiques. Nous avons construit les trois modèles de base de l'approche MDA à avoir (CIM, PIM et PSM). Pour le modèle de besoins du métier de notre système SPFS nous avons modélisé les flux métiers non intrinsèques aux technologies avec le diagramme de contexte dynamique et le cas d'utilisation UML suivi d'une description textuelle de cas d'utilisation. Le PIM a été réalisé avec le diagramme de séquences UML modélisant est les interactions entre les acteurs et le système afin de consolider les descriptions textuelles, les classes du domaine et classes participantes aux interactions du SPFS. L'obtention du PSM de notre SPFS est rendue possible par l'affinement des successif des modèles PIM des différents cas d'utilisations et du modèle dépendant de la plateforme. La transformation du PIM vers PSM est faite automatiquement par l'outil StarUML. Il été également question de démontrer l'intérêt du MDA dans le système éducatif face aux mutations technologiques. Il sied de signaler que nonobstant les Framework, technologies nouvelles de développement, les rôles et capacités des acteurs dans le SPFS restent plus au moins sédentaires dans les établissements privés d'enseignement de la RDC. Selon les perspectives de cet article, il est impérieux à l'avenir de se pencher aux problématiques intrinsèques au savoir-faire des SPFS face la gratuité d'enseignement généralisée, la formation à distance des élèves et les capacités des établissements privées d'enseignements et mode de paiement de frais scolaire.

Bibliographie

Abbas, M. (2018a). *L'environnement FoCaLiZe au service d'UML/OCL* [PhD Thesis]. Université des Sciences et de la Technologie Houari Boumediene (USTHB ….

Abbas, M. (2018b). *L'environnement FoCaLiZe au service d'UML/OCL* [Theses, Université des Sciences et de la Technologie Houari Boumediene (USTHB), Alger, Algérie. ; L'École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), Paris, Evry.]. https://hal.archives-ouvertes.fr/tel-02007777

Abdelhedi, F., Ait Brahim, A., Atigui, F., & Zurfluh, G. (2016). Processus de transformation MDA d'un schéma conceptuel de données en un schéma logique NoSQL. *34e Congrès Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID 2016)*, 15-30. https://hal.archives-ouvertes.fr/hal-01514630

Belaunde, M., Casanave, C., DSouza, D., Kaim, W. E., Frank, W., Hauch, R., Hettinger, M., Consulting, M., Hybertson, D., Jourdan, J., Kurokawa, T., Mellor, S., Mischkinsky, J., Mullins, C., Schwab, C., Rioux, L., Seidewitz, E., Siegel, J., Smith, D., … Wood, B. (s. d.). *This is largely a presentation of the OMG MDA Guide. The Guide was prepared by the ORMSC, under the supervision of the AB. Many folk contributed to the Guide. I'm to blame for what is presented here that is not in the Guide (and for what is in the Guide).* 162.

Biyouda, S., Zahid, A., & Touhami, Z. O. (2021). Les déterminants sociodémographiques et scolaires du risque de décrochage scolaire chez des collégiens marocains. *Revue Française d'Economie et de Gestion*, *2*(3), Article 3. https://www.revuefreg.fr/index.php/home/article/view/235

Booch, G., Rumbaugh, J., & Jacobson, I. (2003). *Le guide de l'utilisateur UML*. Eyrolles.

Brahim, A. A., Ferhat, R. T., & Zurfluh, G. (2020). Approche dirigée par les modèles pour l'extraction automatique du modèle NoSQL. *Business Intelligence & Big Data: 15ème Edition de la conférence EDA, Montpellier France 2019*.

Brun, M. (2010). *Contribution à la considération explicite des plates-formes d'exécution logicielles lors d'un processus de déploiement d'application* [PhD Thesis]. Nantes.

Caron, P.-A., Blay-Fornarino, M., & Le Pallec, X. (2007). La contextualisation de modèles, une étape indispensable à un développement dirigé par les modèles? *Obj. Logiciel Base données Réseaux*, *13*(4), 55-71.

Caron, P.-A., Pallec, X. L., & Derycke, A. (2006). Une architecture conceptuelle pour la construction de système dirigée par les modèles. *Poster IDM*, *6*(2).

Combemale, B. (2008). *Ingénierie Dirigée par les Modèles (IDM)–État de l'art*.

CROZAT, S. (2018). *TN29 : Introduction à la modélisation UML pour la conception bases de données*.

Goman, D. (2012). *Analyse dynamique de l'architecture de Hibernate en lien avec les stratégies de mapping* [PhD Thesis]. Haute école de gestion de Genève.

Harbouche, A. (2018). *Une approche dirigée par les modèles pour une conception flexible des systèmes distribués* [PhD Thesis]. UNIVERSITE MOHAMED KHIDER BISKRA.

Harbouche, O., Chenaoui, A., & Harbouche, A. (2008). Conception de systèmes multi-agents basée sur La démarche MDA. *Avril*, 15-16.

Jézéquel, J.-M., Combemale, B., & Vojtisek, D. (2012). *Ingénierie Dirigée par les Modèles : Des concepts à la pratique...* Ellipses.

Kharmoum, N., Ziti, S., Rhazali, Y., & Fouzia, O. (2019). A method of model transformation in MDA approach from E3value model to BPMN2 diagrams in CIM level. *IAENG International Journal of Computer Science*, *46*(4).

Kharmoum, N., Ziti, S., Rhazali, Y., & Omary, F. (2019). An automatic transformation method from the e3value model to uml2 sequence diagrams : An mda approach. *International Journal of Computing*, *18*(3), 316-330.

MDA Specifications | Object Management Group. (s. d.). Consulté 31 mars 2021, à l'adresse https://www.omg.org/mda/specs.htm

Roques, P. (s. d.). *UML 2.5 par la pratique : Etudes de cas et exercices corrigés*. 1.

Roques, P. (2008). *UML 2 : Modéliser une application Web*. Eyrolles.

Tsanga, R. C. N. N. (2019). Logistique hospitalière et performance des Structures Hospitalières Publiques. Quel impact? *Journal d'Economie, de Management,*

d'Environnement et de Droit, *2*(1), 63-70. https://doi.org/10.48398/IMIST.PRSM/jemed-v2i1.15466

Vandeputte, C. (2018). *L'accueil des adolescents à la MDA : L'anonymat et la confidentialité déterminent-ils le recours de l'adolescent à ce dispositif?*