Modélisation conceptuelle des IHM. Une approche globale s'appuyant sur les processus métier

Arnaud Brossard, Mourad Abed, Christophe Kolski

Modélisation conceptuelle des IHM :

Une approche globale s'appuyant sur les processus métier

Arnaud Brossard – Mourad Abed – Christophe Kolski

Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (CNRS – UMR 8530)
Le Mont Houy
F-59313 Valenciennes Cedex 9
{arnaud.brossard, mourad.abed, christophe.kolski@univ-valenciennes.fr}

RÉSUMÉ. *La modélisation conceptuelle des IHM est un élément essentiel pour arriver à une ingénierie dirigée par les modèles capable de prendre en charge un grand nombre de types d'applications. S'il est possible de trouver aujourd'hui des méthodes de modélisation des IHM capable de prendre en compte la modélisation de niveau conceptuelle, celles-ci sont bien souvent difficiles d'accès à travers l'utilisation de formalisme spécifiques qui ne facilitent pas les échanges avec les experts métier. Or ce sont les personnes les plus à même de définir ces modèles et de les valider. Pour répondre à cette problématique, nous avons défini les bases d'une méthode de modélisation des IHM qui, en s'appuyant sur les processus métier, permet de faciliter la communication entre les informaticiens et les experts métier.*

ABSTRACT. *The conceptual modeling of the HCI is an essential element for being able to reach a true model driven engineering for a large broad of applications. If it is possible to find modeling method which take into account the conceptual modeling level, they are often hard to understand because they use specific formalism which do not facilitate the exchanges with the domain's experts who however are the right people for defining and validating this kind of models. What we propose through our approach is a modeling method of the HCI which, by being based on business process, facilitate the communication between the computer scientists and the domain's experts.*

MOTS-CLÉS : *Ingénierie dirigée par les modèles, Interaction Homme-Machine, modélisation conceptuelle.*

KEYWORDS: *Model Driven Engineering, Human-Computer Interaction, conceptual modeling.*

1. Introduction

Un nouveau paradigme émerge aujourd'hui en matière de développement informatique : c'est l'ingénierie dirigée par les modèles ou IDM (Favre *et al.*, 2006). Celle-ci a pour but de permettre la création d'applications informatiques à partir de modèles conceptuels et surtout à partir d'un assemblage de modèles. Dans ce cas, chaque modèle traite une ou plusieurs problématiques métier bien définies ; chaque problématique représentant un point de vue différent sur un même processus métier comme la sécurité des accès, la répartition des tâches entre les intervenants, etc.

Le processus métier étant un élément important de l'approche IDM, il est indispensable d'en donner une définition. Pour (Johansson *et al.*, 1993) : *"Un processus métier est un ensemble d'activités qui prend un élément en entrée et le transforme en un élément de sortie. Idéalement, la transformation qui se passe durant le processus doit ajouter de la valeur à l'élément en entrée et créer un élément de sortie qui est plus utile et plus adapté aux besoins du destinataire que celui-ci se trouve en amont ou en aval."*. Pour (Davenport, 1992) :*"Un processus métier est un ensemble d'activités structurées et mesurables conçues pour produire une sortie spécifique pour un client ou un marché déterminé. Ceci impose d'étudier en détail la façon de faire le travail au sein d'une organisation et pas le résultat produit par ce travail. Un processus est donc un assemblage ordonné d'activités dans le temps et l'espace avec un début, une fin, et des éléments d'entrée et de sortie clairement définis : c'est une structure pour l'action […] Avoir une approche processus implique d'adopter un point de vue client."*

Il est important de noter que la notion de processus métier n'est pas liée à des interactions avec l'utilisateur. Un exemple de processus métier avec des interactions pourrait être l'ensemble des actions réalisées par un usager pour acheter un titre de transport sur un site internet. Un exemple de processus métier sans interactions pourrait être l'impression automatique d'un rapport journalier sur l'ensemble des retards constatés chaque jour sur les trains ; ce dernier processus pouvant entrer dans la composition d'un processus métier plus large comme l'ensemble des opérations de suivi journalier faites par le contrôleur du réseau de transport.

Une autre notion importante de l'IDM est celle de modèle conceptuel. Pour notre part, nous avons retenu la définition donnée par l'OMG pour le niveau CIM (Computation Independent Model) de l'approche Model Driven Architecture (OMG, 2003) : *"Le*

modèle CIM est une vue du système d'un point de vue indépendant des éléments informatiques permettant de le réaliser. Le modèle CIM ne montre pas les détails de la structure des systèmes. Un modèle CIM est parfois appelé un modèle de domaine et un vocabulaire qui est familier aux experts de ce domaine en question est utilisé pour le spécifier."

L'objectif des modèles conceptuels est donc de permettre aux experts du domaine, qui ne sont pas des informaticiens mais des spécialistes métier, de définir eux-mêmes les modèles des processus métier qu'ils utilisent et qu'ils veulent voir repris dans une application informatique. Dans la suite de cet article, pour plus de clarté, nous utiliserons le terme d'expert métier à la place d'expert de domaine.

Le passage des modèles conceptuels aux applications concrètes, manipulables par des utilisateurs finaux, se fait à travers une succession de transformations de modèles en s'appuyant sur une approche de type MDA (cf. figure 1).

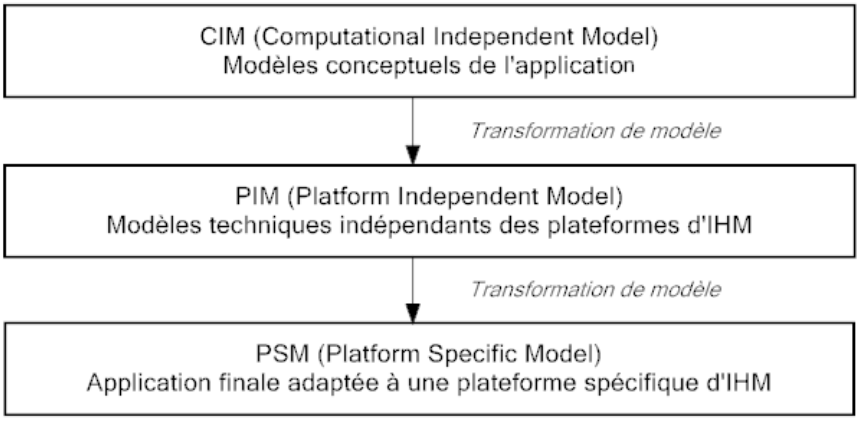


Figure 1. *Représentation globale de l'Approche MDA vue sous l'angle de l'IHM*

Malgré le fait que l'approche MDA (Model Driven Architecture) existe depuis plusieurs années (Oya, 2002) (OMG, 2001), celle-ci n'est encore utilisée que trop rarement dans le cadre de la réalisation d'applications et encore moins dans le cadre du développement d'IHM ; la société d'étude de marchés Gartner Group, dans son étude annuelle "Hype Cycle 2006", ne voyant MDA que comme une technologie émergente dont l'adoption ne devrait pas être généralisée avant 5 ou 10 ans (Gartner, 2006). Et, lorsqu'elle est utilisée, c'est bien souvent de façon incomplète malgré le fait que de nombreux travaux aient montré l'intérêt de ce type d'approche (Eyermann *et al.*, 2004) (Frankel, 2005) (Muller *et al.,* 2005) notamment dans le domaine des IHM (Sottet *et al.,* 2006). Ce qui limite aujourd'hui son utilisation, c'est en partie le manque d'outils et de méthodes permettant de réaliser une modélisation conceptuelle des applications (Tariq *et al.,* 2004*)* indépendante des techniques utilisées pour leur réalisation ; la plupart des solutions existantes se contentant d'utiliser les niveaux PIM et PSM.

En fait, la question qui se pose aujourd'hui est de savoir si cette modélisation conceptuelle des applications est réalisable et surtout quels en sont les avantages, les limites et les contraintes. Plutôt que de vouloir répondre de manière globale à cette question, il apparaît plus pertinent d'essayer, dans un premier temps, d'apporter des éléments de réponse pour un domaine particulier. C'est ce que propose cet article à travers la présentation d'une méthode de modélisation conceptuelle des IHM.

2. La problématique de la modélisation conceptuelle des IHM

La modélisation des IHM à l'aide de modèles n'est pas récente et de nombreux travaux en ont montré les avantages : indépendance par rapport aux langages d'implémentation, réduction des coûts de développement, meilleure qualité des IHM produites, plus grande expressivité ou encore possibilité de représenter des connaissances de manière unique (Palanque *et al.*, 1998) (Szekely, 1996). Mais, à notre connaissance, aucune solution existante ne permet de modéliser des IHM de manière conceptuelle même si certaines propositions les associent, sous une certaine forme, à des composants métier (Dery-Pinna, 2004). On peut se demander si la modélisation conceptuelle est vraiment nécessaire. En fait, selon (Frankel, 2003), elle est indispensable pour assurer la pérennité des investissements informatiques. S'affranchir des contraintes techniques à travers un modèle conceptuel permet de manipuler des concepts directement associés aux métiers qui ont une durée de vie beaucoup plus longue que les technologies utilisées pour réaliser les applications. Mais surtout les modèles conceptuels permettent d'avoir une vision complète des processus métier associés au domaine applicatif considéré et, c'est certainement là leur principal intérêt.

La modélisation conceptuelle étant très certainement un élément clé pour généraliser l'utilisation de l'ingénierie dirigée par les modèles, la question se pose de savoir comment la réaliser. Pour l'OMG (Watson, 2005), celle-ci doit se faire en utilisant un ensemble de modèles complémentaires permettant chacun de modéliser un aspect bien précis de l'application avec un formalisme et une notation adaptés. Cette approche n'est pas totalement nouvelle et de nombreux travaux ont déjà abordé cette problématique (Bardon *et al.*, 2003) (Jardim Nunes, 2001) (Schewe *et al.*, 2005) notamment au niveau des IHM (Wilson *et al*., 1993) mais ceux-ci ne se basent pas sur une approche standard de type MDA ; ce qui en limite leur portée. De plus, ces différents travaux ne proposent pas de modèles conceptuels. Ce qui est clair aujourd'hui c'est que pour pouvoir traiter la problématique des IHM, il faut avoir une approche multidisciplinaire (Bézivin *et al.*, 2004) car il est impossible d'être expert dans tous les domaines liés à l'IHM (modèle de tâches, modèle des utilisateurs, modèle de l'environnement, ergonomie, etc.). D'autre part, il faut aussi être capable de

saisir toute la complexité des IHM et donc de comprendre l'environnement dans lequel elles sont utilisées, ainsi que les objectifs qui leurs sont attribués. Enfin, il faut être capable de prendre en compte l'ensemble des contextes d'utilisation associés à l'application (Calvary *et al.,* 2003).

Ce type d'approche, c'est ce que propose aujourd'hui le langage de description d'interfaces homme-machine UsiXML (Vanderdonckt, 2005). Celui-ci s'appuie sur le framework CAMELEON (Calvary *et al.,* 2003) qui, en définissant un niveau « Tâches et concepts », permet de prendre en charge les tâches de l'utilisateur mais aussi tout un ensemble de concepts annexes permettant de modéliser les utilisateurs ou encore l'environnement (cf. figure 2). Dans UsiXML, les tâches de l'utilisateur sont modélisées à l'aide de ConcurTaskTrees (Paterno, 2000) qui utilise une structure hiérarchique des tâches orientée action utilisateur, des opérateurs temporels et une manipulation possible d'objets communiquant entre les tâches. ConcurTaskTrees, qui utilise une notation graphique pour en faciliter la manipulation, possède quatre catégories de tâches : les tâches utilisateurs réalisées par l'utilisateur, les tâches applications effectuées complètement par le système, les tâches d'interaction réalisées par les interactions entre les utilisateurs et le système et les tâches abstraites qui n'appartiennent pas aux autres catégories définies précédemment. UsiXML, qui s'appuie sur le langage XML, permet de définir des interfaces utilisateurs à un haut niveau abstraction permettant de s'affranchir des contraintes liées aux plateformes techniques utilisées par les interfaces finales. Le passage des interfaces abstraites aux interfaces concrètes est réalisé à travers une succession de transformations et d'enrichissements des modèles abstraits initiaux. UsiXML, par son approche, est compatible avec une démarche de modélisation de type MDA. Pour une présentation plus précise d'UsiXML, on pourra se référer à la thèse de (Florins, 2006) et au site Internet http://www.usixml.org .

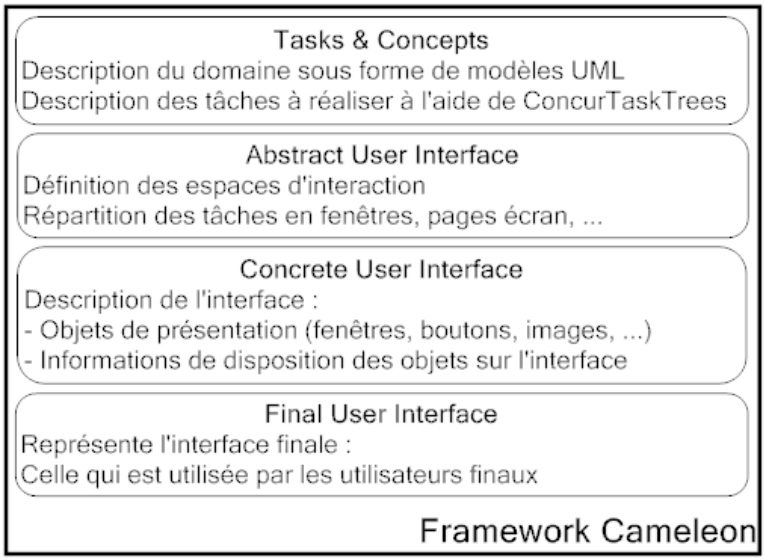


Figure 2. *Schéma global du framework Cameleon utilisée par UsiXML*

L'approche UsiXML étant, d'après nos recherches bibliographiques, l'approche la plus avancée en matière de modélisation des IHM, nous l'utiliserons dans la suite de notre article comme élément de comparaison avec notre proposition d'approche. D'autres approches sont aussi possibles comme celle de (Schattkowsky *et al.,* 2005) qui propose de faire une séparation stricte des aspects visuels de l'interface des autres éléments du modèle global et plus particulièrement des traitements mais celles-ci sont moins avancées qu'UsiXML.

Un autre élément important de l'IDM réside dans la réutilisation des modèles créés en vue d'accélérer et de fiabiliser les développements. Dans ce domaine, des travaux ont déjà été réalisés au niveau des IHM mais sont restés cantonnés à une approche à travers des patrons de conception (Forbrig *et al.,* 2004) ou des assemblages de composants (Schlee *et al.,* 2004) alors que l'IDM tend à une réutilisation complète des modèles conceptuels au sein d'autres applications ou alors au sein d'autres modèles conceptuels plus complexes.

Aujourd'hui, un des problèmes de la modélisation des IHM réside dans le fait que les personnes qui connaissent le mieux les besoins du point de vue applicatif sont les experts métier qui ont souvent beaucoup de mal à comprendre des modèles qui ont été conçus par des informaticiens et sont bien souvent incapables de les manipuler (Selic, 2003) (Jardim Nunes, 2004). On pourra noter qu'UML, en étant trop lié aux concepts de la programmation orientée objets ne permet pas de créer des modèles vraiment indépendants des solutions techniques utilisées et surtout est loin d'être reconnu comme un langage compréhensible par l'ensemble des experts métier (Cook, 2004) (Palano *et al.,* 2006) même si l'utilisation des profils permet d'en simplifier la lecture. Pour faciliter la communication entre les experts métier, les ergonomes et les informaticiens, il est donc indispensable de créer des modèles et outils qui soient destinés aux experts métier et surtout qui soient facilement compréhensibles et manipulables par tous les intervenants (Bernonville *et al.,* 2005).

En résumé, une approche de modélisation IHM devrait se baser sur : (1) Une approche de type MDA afin de permettre la mise en place d'une Ingénierie Dirigée par les Modèles dans le domaine des IHM. (2) Des formalismes et des notations facilement compréhensibles par des experts métier. (3) Une prise en compte de l'environnement complet d'exécution des IHM à travers un ensemble de modèles ; ce que ne permet pas UsiXML qui, par exemple, ne traite pas de la sécurité d'accès aux applications. (4)

La modélisation de l'ensemble des flux d'information ; ce que ne fait pas UsiXML qui ne prend pas en compte par exemple les échanges verbaux d'informations entre deux intervenants d'un même processus métier.

Notre approche, en se basant sur les approches existantes propose une solution de modélisation conceptuelle des IHM respectant ces quatre grands principes.

Le cadre général étant fixé, il faut pouvoir évaluer notre approche. Malheureusement, il n'existe pas de grille d'évaluation universelle pour les méthodes de modélisation mais plutôt des axes d'évaluation tels que ceux qui sont définis par (Selic, 2003) pour les outils et méthodes :

L'abstraction : ils doivent permettre de simplifier la réalité à travers une certaine abstraction qui permet aux personnes en charge de la modélisation de n'avoir à se concentrer que sur les problématiques métiers et pas sur les problématiques annexes comme des problématiques techniques par exemple.

La compréhensibilité : ils doivent permettre de créer des modèles faciles à comprendre et à manipuler par des informaticiens mais aussi par des experts métier.

La pertinence : ils doivent pouvoir modéliser la réalité dans sa globalité.

La prévisibilité : ils doivent permettre de prévoir le comportement de l'application à partir des modèles conceptuels.

Le faible coût : ils doivent permettre des gains de productivité en matière de développements informatiques.

Dans la troisième partie de cet article, nous présenterons notre approche de modélisation des IHM dans une approche cible plus globale. Puis, nous présenterons l'ensemble des modèles conceptuels spécifiques aux IHM. Ensuite, nous développerons la notion de services fonctionnels. Enfin, nous expliquerons comment nous passons des modèles conceptuels d'IHM aux applications finales.

3. Description générale de notre approche

Dans le cadre de la modélisation conceptuelle d'une application, la modélisation des IHM ne représente qu'une partie de l'ensemble des modèles. Or, tous les modèles sont en interaction les uns avec les autres (Watson, 2005). Aussi, il est impossible de proposer une méthode de modélisation des IHM sans l'inscrire dans une méthode plus globale de modélisation des applications. Pour répondre à cette problématique, et en nous basant sur nos travaux précédents (Brossard, 2006) (Brossard *et al.*, 2007) nous avons développé une architecture globale à plusieurs niveaux respectant une approche de type MDA (cf. figure 3).

Le premier niveau, correspondant au niveau CIM du MDA, contient l'ensemble des modèles conceptuels permettant de modéliser une application dans sa globalité.

Le deuxième niveau, correspondant au niveau PIM, présente la particularité de contenir un framework de services fonctionnels et techniques utilisables de manière explicite ou de manière implicite dans les modèles de niveau CIM. Ces services, que nous présenterons en détail dans la partie 5, contiennent des fonctionnalités qui permettent l'exécution d'un processus métier mais qui ne lui sont pas spécifiques. Au niveau du processus, l'important est d'indiquer qu'on va utiliser cette fonctionnalité et pas de la définir complètement. Un exemple de services fonctionnels explicites pourrait être l'appel à des fonctions de personnalisation des informations retournées à l'utilisateur en fonction de sa langue d'origine. Un exemple de service implicite pourrait être la gestion des informations de session de l'utilisateur. Celui-ci est utilisé par tous les processus de manière automatique pour gérer les variables de session et n'a pas besoin d'être appelé de manière explicite par le processus. L'ensemble de ces services sont configurables à travers l'utilisation de règles métier qui sont gérées de manière centralisée au niveau du framework PIM. Une règle métier permet de décrire précisément les critères de décision utilisés au sein des différents services fonctionnels et techniques et qui influent sur leur déroulement (OMG, 2005).

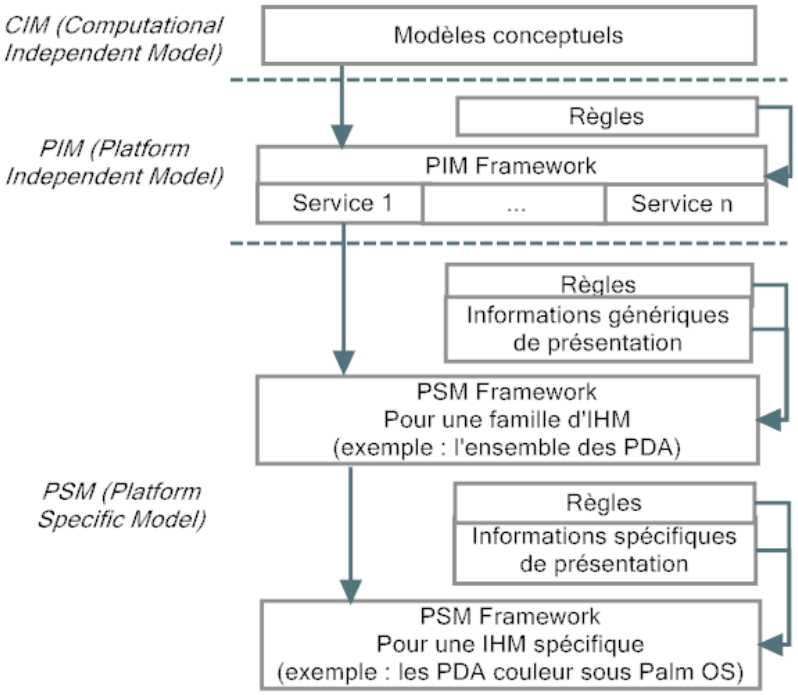


Figure 3. *Représentation de notre proposition d'architecture globale*

Le troisième niveau, correspondant au niveau PSM de l'approche MDA, contient en fait deux sous-niveaux. Le premier permet de définir des règles métier et des informations génériques de présentation pour une famille de plateformes d'interaction par exemple pour l'ensemble des PDA. Les règles métier permettent de préciser la façon dont les éléments d'IHM définis dans les modèles de niveau PIM seront traduits en éléments d'IHM concrets pour une famille de plateformes d'interaction spécifique. Pour donner un exemple, les règles pourront être utilisées pour indiquer à partir de combien d'éléments dans une liste de valeur celle-ci doit être présentée sous la forme d'une liste d'éléments défilants de type Listbox ou sous la forme d'une boîte de sélection de type ComboBox. Le deuxième sous-niveau permet de définir des règles métier et des informations de présentation pour une plateforme d'interaction bien précise comme par exemple pour les PDA couleur avec Palm OS. Au niveau PSM, les différents framework utilisés permettent de spécialiser le framework de services défini au niveau PIM mais permettent aussi de définir les éléments permettant de prendre en compte de manière concrète et physique les éléments d'IHM définis au niveau conceptuel. Par rapport à l'approche classique proposant la génération du PSM à partir d'un tissage du modèle PIM avec un modèle de plateforme PDM (Plateform Definition Model), notre approche intègre ce tissage à travers l'utilisation de règles et d'informations de présentation dans les deux sous-niveaux du PSM.

Cet article ayant pour but de présenter une méthode de modélisation conceptuelle des IHM, nous allons maintenant présenter brièvement l'ensemble des modèles conceptuels que nous avons défini pour pouvoir modéliser une application (cf. figure 4). Sur la figure, les modèles d'IHM, sur lesquels portent cet article, sont indiqués en caractères gras et sont entourés d'un cadre en pointillés.

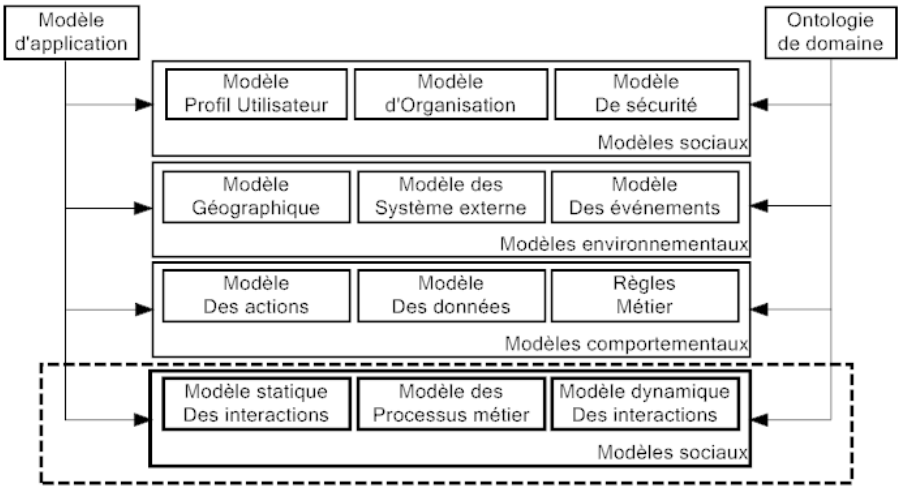


Figure 4. *Représentation de l'ensemble des modèles conceptuels de notre approche globale*

Le modèle d'application permet de modéliser les éléments spécifiques à l'application qui se sont pas repris dans les autres modèles et de définir les propriétés directement associées à l'application (langue par défaut affichée, ensembles de données utilisées, etc.).

L'ontologie de domaine permet de modéliser l'ensemble des concepts métier du domaine de l'application. Celle-ci, en se basant sur la notation OWL DL (W3C, 2004) introduit des relations et des contraintes entre concepts qui permettent d'utiliser une sémantique métier complète dans les autres modèles. L'ontologie de domaine est un élément clé de notre approche car c'est elle qui permet de faire le lien entre les différents modèles conceptuels de l'application et l'ensemble des concepts associés au domaine métier dans lequel va évoluer l'application. Pour l'importance des ontologies en matière de modélisation conceptuelle, on pourra se reporter à (Guarino *et al.,* 2000) ou (Furtado *et al.*, 2001).

Les modèles sociaux permettent, en premier lieu, de modéliser les utilisateurs à travers un modèle de profil utilisateur générique. Le modèle d'organisation permet pour sa part de définir l'environnement humain dans lequel l'application va être utilisée sous la forme de structures d'organisations hiérarchiques ; chaque élément de ce modèle étant rattaché à un autre élément du modèle à travers un lien hiérarchique ou un lien d'appartenance. Pour une même application, on pourra avoir plusieurs modèles d'organisation différents. Ainsi, pour une application fournissant des informations sur les horaires de trains, on pourra avoir un modèle d'organisation pour les employés de la société et un autre pour les clients externes. Enfin, le modèle de sécurité permet de définir, à partir des éléments définis dans le ou les modèle(s) d'organisation les accès à l'application sous la forme de rôles utilisateurs. Pour tenir compte du fait que les modèles sociaux peuvent évoluer en fonction de l'utilisateur, les modèles d'organisation et les modèles de sécurité peuvent utiliser des informations du profil utilisateur et des règles métier pour faire évoluer de manière automatique les accès des utilisateurs en fonction de l'évolution dans le temps des informations contenues dans leur profil comme par exemple le changement d'un groupe d'appartenance ou d'un rôle d'accès en fonction de l'âge de l'utilisateur.

Les modèles environnementaux permettent de définir l'environnement de l'application. Au sein de cet ensemble, le modèle géographique permet de modéliser l'environnement géographique associé à l'application. Ce modèle géographique est constitué de couches d'informations regroupant des informations de même nature du point de vue du domaine métier ; les couches étant reliées les unes aux autres à travers l'ontologie de domaine. Ainsi, dans le cadre d'une application dans le domaine des transports, on pourrait définir une couche d'informations contenant les coordonnées géographiques des arrêts de bus de toutes les lignes d'une compagnie de transport et une couche d'informations définissant chaque zone tarifaire ; une zone tarifaire étant définie au niveau de l'ontologie de domaine comme étant composée d'un ensemble d'arrêts. Le modèle de systèmes externes permet de modéliser l'ensemble des systèmes, généralement des applications ou des systèmes techniques comme des imprimantes, avec qui l'application est susceptible de communiquer. Chaque système est vu comme une boîte noire qui peut recevoir et émettre des informations, qui peut recevoir des consignes, émettre et réagir à des événements. Pour être qualifiée de système externe, une application doit ne pas pouvoir être modifiée par les personnes en charge du domaine d'application. C'est le cas par exemple d'un progiciel de gestion qu'une société aurait acheté à un tiers. Enfin, le modèle d'événements permet de définir l'ensemble des événements pouvant survenir dans l'environnement d'exécution de l'application. Dans le cadre de notre approche, un événement est un message d'information qui est émis à destination d'une ou de plusieurs applications et qui peut interrompre ou influer sur le fonctionnement d'un processus en cours au sein de cette ou de ces application(s).

Les modèles comportementaux permettent de modéliser les éléments internes à l'application qui influent sur son fonctionnement mais avec lesquels les utilisateurs n'ont aucune interaction. Dans cet ensemble, le modèle des actions est utilisé pour modéliser les tâches métier effectuées sans aucune interaction avec l'utilisateur. Un exemple d'action pourrait être le calcul du total d'une commande ou l'envoi d'une facture à l'impression après mise en page automatique. Le principe des actions est de permettre une définition et une réutilisation simple, lors de la modélisation des processus métier, d'un ensemble de tâches. Le modèle de données permet de modéliser les données auxquelles va accéder l'application en s'appuyant sur les modèles conceptuels de données définis dans l'approche MERISE. Chaque modèle de données est rattaché aux concepts de l'ontologie de domaine à l'aide d'un langage d'association de type XML. La particularité des modèles de données et de pouvoir être regroupés en ensembles de données qui pourront ensuite être rattachés au modèle d'application. Ainsi, il est possible de définir, pour un même concept de l'ontologie de domaine, plusieurs modèles de données différents se trouvant chacun dans des ensembles de données distincts et ainsi d'envisager l'utilisation, pour chaque application, de sources de données différentes pour un même concept. Enfin, le modèle de règles métier permet de définir l'ensemble des règles métier de validation, d'action et de sélection utilisées par l'application. Ces règles, qui sont utilisables par les autres modèles de notre approche, sauf par l'ontologie de domaine, permettent d'agir facilement et rapidement sur les applications en centralisant une partie de la logique métier au sein d'une même structure.

Enfin, les modèles des interactions permettent de modéliser pour chaque application l'ensemble des interactions homme-machine : processus et workflow métier, modèle statique d'interaction et modèle dynamique d'interaction. L'objectif de cet article étant de présenter la modélisation conceptuelle des interactions homme-machine, nous n'allons par la suite ne présenter que ces trois modèles conceptuels d'interactions. Afin de limiter la taille de cet article, le traitement des relations entre ces modèles et les autres modèles de notre approche globale ne sera abordé que de manière succincte dans le cadre des différents exemples présentés.

Conformément à une architecture de type trois tiers, nous avons dans notre approche une séparation entre les données, représentées par les modèles de données et par l'ontologie de domaine, et les traitements représentés par les modèles des actions et les modèles des systèmes externes. La couche de présentation est pour sa part prise en compte au niveau des trois modèles d'interactions. Afin d'assurer cette séparation, le modèle des processus métier ne modélise que des processus métier nécessitant des interactions avec les utilisateurs ; les processus ne nécessitant pas d'interaction étant repris au niveau des modèles d'actions. Notre approche a aussi la particularité de proposer des modèles de type support qui peuvent être utilisés par les autres modèles.

Ainsi, le modèle de sécurité peut être utilisé pour modifier le comportement d'un traitement, d'un modèle d'action, ou peut être utilisé pour modifier le contenu d'une interface en fonction de l'utilisateur connecté.

4. Les différents modèles conceptuels utilisés pour la modélisation des IHM

4.1. Les éléments communs aux différents modèles

Comme nous l'avons indiqué dans la partie 2, un des problèmes de la modélisation réside aujourd'hui dans le fait que les modèles ne sont pas facilement compréhensibles par tous les intervenants d'un projet. Ceci est particulièrement vrai pour les experts métier qui n'ont pas toujours une culture informatique ou technique et qui n'ont pas forcément le temps ou l'envie d'apprendre à lire des modèles techniques. Aussi, il est indispensable de proposer et/ou de développer des langages et outils facilement utilisables et facilement compréhensibles par tous mais suffisamment complets et rigoureux pour prendre en charge les problématiques associées aux différents modèles concernés. Pour les modèles d'interactions, nous avons identifié, dans le cadre de nos travaux, trois types de modèles : (1) Les modèles des processus métier associés aux interactions : ils permettent de savoir pourquoi on a besoin de réaliser des interactions entre le système et les utilisateurs et surtout quel est le cheminement à effectuer pour arriver à réaliser complètement chaque processus métier (correspond en partie au modèle de tâches dans une approche de type UsiXML). Les tâches qui sont définies dans les processus métier sont les tâches qui nécessitent une interaction avec l'utilisateur ou alors font appel à des actions (cf. partie 3) ; c'est-à-dire à des ensembles de tâches sans interactions avec l'utilisateur. (2) Les modèles associés aux ensembles d'éléments d'interactions qui permettent de regrouper des éléments qui ont une cohérence forte aussi bien du point de vue des interactions que du point de vue de la logique métier. Pour une application disposant d'une interface graphique, un ensemble d'éléments d'interaction se traduit généralement par des écrans (correspond à l'Abstract User Interface dans un modèle de type UsiXML). Ces ensembles définissent ce que nous appellerons dans la suite du document les modèles statiques d'interaction. (3) Les modèles dynamiques qui permettent de modéliser, pour un groupe d'éléments d'interactions donné, par exemple une page écran, tous les liens entre les éléments d'interactions qui constituent ce groupe (correspond à un modèle de validation des interfaces qui n'existe pas dans l'approche UsiXML).

Dans le cadre de notre approche, les modèles des processus métier servent de base à la définition des modèles statiques d'interactions et des modèles dynamiques d'interaction. En conséquence, ils doivent être définis en premier. Si aujourd'hui on dispose de nombreuses possibilités pour modéliser les tâches à l'aide d'outils comme CTTE (Paterno, 2000) ou K-MADe (Baron *et al.,* 2006), ceux-ci ne sont malheureusement pas facilement manipulables par des experts métiers. Mais surtout, ils ne prennent pas en charge l'ensemble des flux d'informations et des intervenants d'un processus notamment lorsque les tâches à considérer ne sont pas directement interprétables au niveau informatique comme par exemple un échange d'informations verbales entre deux personnes.

Or, il existe aujourd'hui un formalisme, théoriquement compréhensible par des experts métier permettant de définir des processus métier, y compris des workflow métier, et surtout capable de prendre en charge l'ensemble des flux d'information : c'est la notation Business Process Modeling Notation ou BPMN (BPMI, 2004). Celle-ci est d'ailleurs recommandée aujourd'hui fortement par l'OMG (Watson, 2005) pour la modélisation des applications. Ce formalisme présente en outre l'avantage de pouvoir s'intégrer dans une approche de type MDA (Smith, 2003). Notre approche, afin de respecter les critères de simplicité de lecture et de prise en compte de l'ensemble des flux d'information, s'appuie sur cette notation pour définir l'ensemble des modèles conceptuels d'IHM.

Notre approche utilisant comme point central une ontologie de domaine, nous allons présenter rapidement les choix effectués pour la définition de cette ontologie : (1) L'ontologie contient tous les concepts métier du domaine applicatif. Pour donner un exemple, dans le domaine des transports, on pourrait trouver le concept "Ligne de bus", le concept "Arrêt de bus" ou encore le concept "Ville desservie". Elle contient aussi l'ensemble des relations entre ces concepts ainsi que les restrictions qui leur sont associées. Par exemple, dans le domaine des transports, on peut indiquer dans l'ontologie qu'une "Ligne de bus" et constituée d'au moins deux "Arrêt de bus" et que chaque "Arrêt de bus" et associé à au moins une "Ville desservie". (2) L'ontologie est définie par des experts du domaine, c'est-à-dire par des experts métier. (3) Le formalisme choisi pour cette ontologie est un langage sémantique pour le web recommandé par le W3C : OWL (W3C, 2004). L'objectif d'OWL est de faciliter le partage d'informations entre les services et les agents web. Avec OWL, il est possible de définir des classes et des propriétés pour ces classes ainsi que des individus pour les classes et faire des inférences sur ces individus. Enfin, on peut définir des règles pour les classes qui s'appliqueront à tous les individus. Le W3C propose trois versions d'OWL. La version OWL-Lite qui correspond à une structure de classement de concept de type hiérarchique avec un nombre limité de possibilités de définition de relations. La version OWL-DL qui permet d'utiliser des règles d'inférence sur les différents individus de l'ontologie en s'assurant que le résultat de l'application de ces règles d'inférence donnera toujours un résultat dans un temps fini. La version OWL-Full qui permet de définir ses propres règles d'inférences et des synonymes entre concepts mais qui ne permet plus de s'assurer d'avoir un résultat dans un temps fini lors de l'utilisation de règles d'inférences. Dans le cadre de notre approche, nous choisi d'utiliser la version OWL-DL qui représente un juste compromis et surtout qui permet de définir des règles d'inférence qui pourront être assimilées à des vues spécifiques sur des ensembles de données.

Dans notre approche, l'ontologie de domaine est utilisée de manière à permettre une séparation stricte entre les modèles et les données, le seul modèle faisant le lien entre les concepts du domaine et les données étant le modèle de données. Ceci permet d'utiliser, pour un même concept, différentes sources de données ; chaque source de données devant pouvoir exprimer le concept auquel elle est rattachée. Ainsi, si on définit que le concept client comporte seulement deux propriétés "numéro client" et "nom

client" alors toute source de données qui contient des informations clients et qui contient au moins deux champs contenant le numéro du client et le nom du client pourra être rattaché au concept client même si elle possède d'autres champs ; ceux-ci n'étant alors tout simplement pas utilisés dans l'application. Les concepts métier évoluant en théorie moins vite que les données, ceci permet de rendre les modèles plus stables dans le temps et aussi plus compréhensibles ; l'ontologie de domaine représentant un véritable dictionnaire des termes métier utilisés par les applications. De ce point de vue, l'ontologie de domaine facilite grandement la communication entre les différents intervenants du projet ; chacun utilisant alors un vocabulaire commun.

4.2. Le formalisme BPMN

La notation BPMN (BPMI, 2004), qui est assez récente, a pour but de permettre de modéliser l'ensemble des processus métier associés à une application. Elle permet de modéliser aussi bien les tâches humaines que les tâches non-humaines et surtout elle permet de préciser clairement les différents intervenants pour chacune des tâches définies. Elle permet aussi de modéliser l'ensemble des flux d'information associés à un processus métier bien défini.

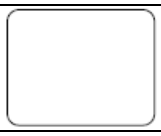
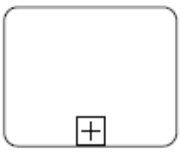
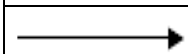
Notation	Signification
	**Événement de départ :** marque le début d'un processus métier. Il ne peut y avoir qu'un événement de départ par diagramme.
	**Événement de fin :** Il marque la fin d'un processus métier. Il ne peut y avoir qu'un seul événement de fin par diagramme.
	**Activité/Processus/Sous-Processus :** Une activité représente un travail effectué. Si elle n'est pas composée d'autres activités, on dit que c'est une tâche élémentaire sinon, c'est un processus ou sous-processus standard dans lequel l'ordre d'exécution des activités est fixé.
	**Processus/sous-processus non étendu :** Un processus non étendu est la représentation simplifiée d'un processus (sur le diagramme ne sera visible que le nom du processus). À tout processus non étendu est associé un processus/sous-processus "étendu" (représenté sous la forme d'un processus / sous-processus standard) qui permet d'en préciser le contenu.
	**Processus "Ad-Hoc" :** Une activité "Ad-Hoc" représente un processus dans lequel tous les sous-processus peuvent être appelés indépendamment les uns des autres (il n'y a pas d'ordre contrairement à un processus standard). Le signe "+" précise que le processus "Ad-Hoc" est représenté de manière non étendue.
	Représente une porte logique de type OU exclusif permettant de faire des branchements conditionnels lors de l'exécution d'un processus métier.
	Représente une association vers un élément qui, dans notre méthode, devra être précisé à travers l'utilisation d'un artefact adapté
	Représente un flux normal entre deux éléments
	Représente le flux par défaut (à n'utiliser qu'avec des flux conditionnels)
	Représente un flux de type "envoi d'informations" entre deux éléments
	Un ensemble d'activité représente l'ensemble des processus/activités effectués par un ou plusieurs intervenants (humains ou non humains). Par défaut, on recommande d'utiliser pour le nom de l'ensemble d'activité le nom du rôle métier associé aux tâches qu'il contient.

Tableau 1. *Le formalisme BPMN*

Une des particularité de la notation BPMN est d'avoir été conçue en se basant sur les réseaux de Petri et donc de permettre, en théorie, une validation des modèles à deux niveaux : vérifier qu'il n'y a pas de blocage dans le processus métier qui empêche la bonne exécution de celui-ci et vérifier que tous les éléments définis au niveau du processus métier sont accessibles. Malheureusement, les outils disponibles actuellement ne permettent pas de faire ce type de validation bien que celle-ci soit un avantage important de BPMN par rapport aux autres notations existantes. A ce sujet, on pourra se reporter aux travaux de (Dikjman et al., 2007) et à l'outil "BPMN to Petri Net Transformer" qui a été développé dans le cadre de leurs recherches et qui est disponible à l'adresse http://is.tm.tue.nl/staff/rdijkman/cbd.html.

Afin de faciliter la compréhension des modèles présentés dans la suite de l'article, nous allons préciser la signification de quelques éléments de notation de BPMN. Ainsi, un processus métier représente une ou plusieurs tâches métiers réalisées par un ou plusieurs systèmes et/ou par une ou plusieurs personnes. Il est représenté par une association d'actions/de tâches élémentaires. Il a toujours un début et une fin. Il a toujours un but au niveau métier. Un événement représente quelque chose qui se passe durant un processus métier. Un événement influe sur le déroulement du processus métier. Un événement doit toujours être nommé.

4.3. Description du cadre d'application

Dans le cadre de la présentation des différents modèles, un scénario d'application simple, tout en étant représentatif de recherches et développements actuels (Anli *et al.,* 2004), a été développé pour servir de base à une présentation concrète de chaque modèle. Ce scénario est le suivant : *« Sur un site internet d'une société de transport, un utilisateur désire obtenir, pour un trajet particulier entre deux lieux distants, tous les horaires possibles à partir d'un jour et d'une heure de départ déterminés. Pour cela, il ouvre sur le site internet un formulaire de saisie lui permettant d'indiquer la date, l'heure et le lieu de départ ainsi que le lieu d'arrivée. Une fois les informations saisies, il valide son choix. Les informations sont envoyées au système qui calcule l'ensemble des itinéraires possibles et les retourne à l'utilisateur sous la forme d'une liste d'horaires. Cette liste est affichée à l'utilisateur à travers une page de résultats. Après consultation, l'utilisateur peut ensuite décider de refaire une nouvelle demande ou de fermer la page de résultats. »*

4.4. Le modèle des processus métier

Le premier modèle conceptuel d'interaction de notre approche est le modèle des processus métier. Il permet de définir l'ensemble des processus métier associés aux applications en se concentrant uniquement sur le cœur des processus même et sur les acteurs. Ainsi, on ne trouvera pas dans ces modèles toutes les interactions possibles mais uniquement les interactions nécessaires et suffisantes pour le bon déroulement du processus métier qui leur est attaché. Par exemple, lors de la modélisation d'un processus métier associé à une saisie d'informations, l'indication des tâches de validation des informations entrées par l'utilisateur ne doivent être indiquées que si elles sont utiles à la bonne compréhension du processus métier. Dans le cas contraire, elles ne seront reprises qu'au niveau des modèles conceptuels dynamiques associés à ce processus métier (cf. 4.5 de cet article). Ceci permet aux experts métier en charge de la modélisation de se concentrer sur les processus métier même et pas sur la définition de l'ensemble des interactions associées à ces processus ; ce qui conduit à une plus grande lisibilité au niveau des modèles. Cette simplification relative des modèles conceptuels de processus métier présente l'inconvénient de lier très fortement les différents modèles conceptuels d'interaction entre eux ; l'ensemble des tâches associée au processus métier complet se trouvant dans deux types de modèles différents ce qui n'en facilite pas la maintenance.

Lors de la modélisation des processus métier, notre méthode impose de définir des ensembles logiques cohérents de tâches. Pour cela, nous avons ajouté à la notation BPMN des éléments complémentaires de notation qui nous permettent de définir et de manipuler de manière explicite les tâches associées à ces ensembles logiques. Ainsi, chaque ensemble commence par une tâche dont le nom est précédé par le préfixe « Ouvrir » suivi du nom de l'ensemble d'interaction et se termine par une tâche commençant par le mot-clé « Fermer » suivi du nom de l'ensemble d'interaction. Ces ensembles seront par la suite utilisés au niveau des modèles statiques d'interactions afin de permettre d'établir le lien avec les modèles de processus métier ; chaque modèle statique représentant un ensemble d'interaction. Les ensembles d'interactions étant définis, pour décrire l'ensemble des tâches effectuées à l'intérieur de ces ensembles d'interaction, on utilise une notation spécifique définie dans le cadre de notre approche. Le point important au niveau de la notation réside dans le fait que les informations manipulées ne peuvent être que des concepts définis au sein de l'ontologie de domaine associée à l'application. La définition des tâches associées à la manipulation de ces concepts se fait à l'aide d'un certain nombre de mots-clés prédéfinis. Ainsi le préfixe « Entrer » suivi du nom et de la propriété associée du concept indique une tâche de saisie explicite d'information pour un individu particulier du concept (cf. figure 5). Le préfixe « Lire » suivi du nom du concept et de la propriété associée indique qu'on va afficher l'information associée à une propriété d'un concept pour un individu donné. D'autres préfixes, comme « Supprimer » suivi du nom du concept permettent de supprimer un individu associé à ce concept dans l'ontologie. Enfin, l'utilisation du préfixe « Action » suivi du nom d'un modèle conceptuel d'action permet d'indiquer qu'on va exécuter un ensemble de tâches définies dans un modèle d'actions. Pour les actions, il est possible d'indiquer les informations nécessaires à l'entrée et les informations fournies en sortie en utilisant la notation BPMN d'envoi de flux d'information sur laquelle on précise le concept de domaine qui transite dans ce flux d'information. Ainsi, dans l'exemple de la figure 5, les informations en entrée de l'action « CalculerHoraires » sont un individu du concept « DemandeHoraire » et les informations retournées en sortie sont un individu du concept « ListeHoraires ».

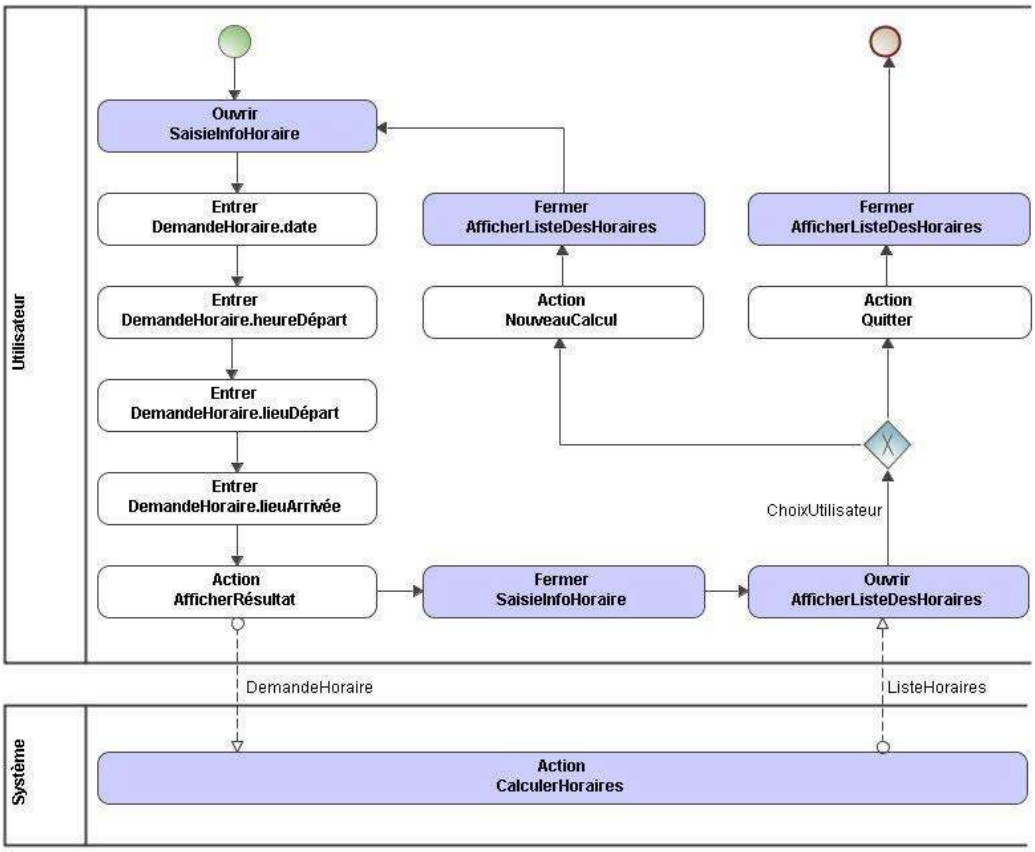


Figure 5. *Exemple de modèle de processus pour le scenario présenté en 4.3*

Si un modèle de processus métier ne peut définir qu'un seul processus métier à la fois, il peut néanmoins utiliser d'autres processus métier dans le cas où ceux-ci peuvent être considérés comme étant des sous-processus du processus principal. Pour donner un exemple, si on suppose qu'on modélise une application de consultation d'informations sur les transports en commun pour des usagers, on peut trouver au sein de cette application un processus métier pour la consultation des tarifs, un processus métier pour la visualisation des lignes de transport et un processus métier pour la demande d'horaires. Dans notre approche, tous ces processus métier devront être définis chacun dans un modèle de processus différent. Pour pouvoir utiliser ces processus, un utilisateur devra passer lui-même par un autre processus métier qui lui permettra de choisir d'exécuter un des trois processus définis précédemment. Pour ce processus métier, les trois processus définis précédemment sont considérés comme étant des sous-processus et peuvent être appelés en utilisant une tâche spécifique dont le nom commence par le préfixe « ExécuterProcessus » suivi du nom du processus métier à exécuter. Cette notation est spécifique à notre approche et, par défaut, l'appel d'un sous processus par un processus père entraîne la mise en suspens de l'exécution du processus père jusqu'à la fin de l'exécution du sous-processus. En termes de réutilisation, une application est définie comme étant un appel à un processus métier particulier. De cette manière, les processus métier ne sont pas définis par rapport à une application spécifique mais par rapport à une problématique métier bien déterminée. Ainsi, il peuvent être utilisés par plusieurs applications différentes si celles-ci font appel à la même problématique métier.

Au niveau de la représentation graphique, l'utilisation d'un code couleur permet de différencier les tâches réalisées de manière interactive par les utilisateurs, les échanges d'information informels comme des échanges verbaux et les tâches réalisées de manière automatique par le système. Dans l'exemple de la figure 5, les tâches réalisées de manière interactive par l'utilisateur sont sur fond blanc tandis que les tâches réalisées de manière automatique par le système sont sur fond gris (ce choix de couleur correspond à un choix technique pour rendre l'article plus lisible).

Toujours au niveau de la notation utilisée, il est important de noter que pour les portes logiques, qui définissent des choix effectués soit par l'utilisateur soit par le système en fonction de certains critères, on doit toujours indiquer le type de choix pris en compte dans le cadre du modèle. Ainsi si on indique, au niveau de la connexion précédant la porte logique, le mot-clé « ChoixUtilisateur » ceci signifie que c'est l'utilisateur qui procède lui-même au choix de la tâche à exécuter dans le cadre de la suite du processus métier. Le processus est alors interrompu jusqu'à ce l'utilisateur fasse ce choix. Sur une interface de type graphique sur une plateforme PC ce type de choix se traduira généralement par la possibilité pour l'utilisateur de sélectionner une option dans un menu ou de cliquer sur un des boutons à l'écran.

Par rapport aux approches existantes, notre modèle des processus métier correspondrait au modèle de tâches qu'on trouve dans une approche de type UsiXML (Vanderdonckt, 2005) à travers l'utilisation de CTT ou dans une approche de type k-MADe (Baron 2006) qui elle n'est pas spécifique aux IHM. Les avantages de notre approche pour modéliser les processus métier plutôt que CTT ou k-MADe sont : (1) La notation BPMN est la seule notation de modélisation de tâches, de processus métier, qui soit normalisée (spécifications adoptées le 6 février 2006 par l'OMG). (2) En théorie, les modèles développés sont plus faciles à lire et

à manipuler ; le formalisme BPMN ayant été conçu pour être utilisé par des experts métier. Ceci devra être étudié en détail dans le cadre de notre approche lorsque nous passerons à la phase de validation de nos outils avec des experts métier. (3) Possibilité, au niveau des processus, d'indiquer graphiquement, pour chaque intervenant l'ensemble des tâches qu'il doit réaliser et les échanges d'information entre intervenants. Cette façon de visualiser la répartition des tâches n'existe pas dans les approches comme UsiXML et k-MADe. Elle est réalisable, sous une certaine forme, dans un diagramme de séquence sous UML mais, avec UML, il est très difficile de modéliser l'ensemble des actions d'un intervenant ou la sélection d'un élément de navigation à travers des classes ou des objets. (4) Possibilité de regrouper les différentes tâches et sous-tâches en groupes homogènes d'ensembles d'interaction. (5) BPMN étant, par sa genèse, basé sur les réseaux de Petri, il devrait être possible de tester, une fois les outils développés, chaque processus dès sa conception (cf. 4.2). (6) Enfin, BPMN offre la possibilité de représenter de manière simple les échanges d'informations se faisant avec des intervenants externes ou entre intervenants humains comme, par exemple, les échanges d'informations verbales. Aucune autre méthode existante ne permet de modéliser ce type de flux d'informations or ceux-ci sont indispensables pour pouvoir modéliser un processus métier complètement.

Malgré tout, BPMN présente un certain nombre d'inconvénients qui pour l'instant ne permettent pas de conclure sur sa supériorité éventuelle par rapport aux autres approches existantes. Ces inconvénients sont : (1) BPMN ne possède pas de méta-modèle normalisé (celui-ci sera à définir dans la suite de nos travaux à l'aide d'un outil comme EMF) ; ce qui implique que le passage du niveau CIM au niveau PSM à travers une transformation de modèles reste à définir. En principe, à partir de la définition d'un méta-modèle créé à partir de MOF, ce passage devrait pouvoir se faire en utilisant un langage de transformation de modèles comme QVT, ATL ou encore Kermeta. L'absence de méta-modèle basé sur MOF ne permet pas non plus de nous appuyer sur les outils standards de l'IDM pour définir les relations entre les différents modèles utilisés dans notre approche. (2) Dans le cadre de l'avancement de nos travaux, nous n'avons pas encore pu vérifier la complétude du langage BPMN associé à notre formalisme de notation et donc nous ne pouvons pas affirmer que celui-ci soit capable de modéliser tous les types de processus métier existants. Ce point fait, pour l'instant, l'objet de perspectives de recherche. (3) Le choix effectué dans notre approche de limiter le contenu des processus métier au minimum afin de faciliter la lecture des modèles, ne permet pas d'avoir des modèles de processus vraiment exhaustifs ; ce qui peut conduire à de mauvaises interprétations. Pour limiter ce risque, nous avons établi, pour l'instant de manière empirique, un certain nombre de règles de conception qu'il conviendra de faire évoluer par la suite. Pour donner un exemple de règle, nous avons défini que les tâches de validation des informations entrées par l'utilisateur ne sont pas à indiquer dans les processus métier sauf si elles font appel à une action. Ainsi, si la validation consiste à vérifier que l'information entrée par l'utilisateur est non vide et possède une taille supérieure à quatre caractères, celle-ci sera définie dans le modèle dynamique d'interaction (cf. 4.5). Par contre si la validation de l'information entrée par l'utilisateur consiste à vérifier que celle-ci correspond bien à une fiche client qui est toujours valide et qui a déjà passé une commande alors dans ce cas, on utilisera une action pour faire cette validation et celle-ci sera indiquée au niveau du modèle de processus.

Le modèle conceptuel des processus métier ayant été présenté, nous devons maintenant étudier comment passer de ces processus métier aux interfaces utilisateurs ; c'est-à-dire comment créer des ensembles cohérents d'interaction.

4.5. Le modèle statique des interactions

4.5.1 Le cadre général

Le modèle statique d'interaction permet de définir l'ensemble des éléments d'interactions avec lesquels les utilisateurs peuvent interagir. Ce modèle s'appuie beaucoup sur les propositions et idées utilisées par UsiXML pour modéliser de manière conceptuelle les interfaces graphiques. Une présentation globale de la hiérarchie de composants d'IHM que nous utilisons dans le cadre de nos modèles statiques d'interaction est donnée dans la figure 6.

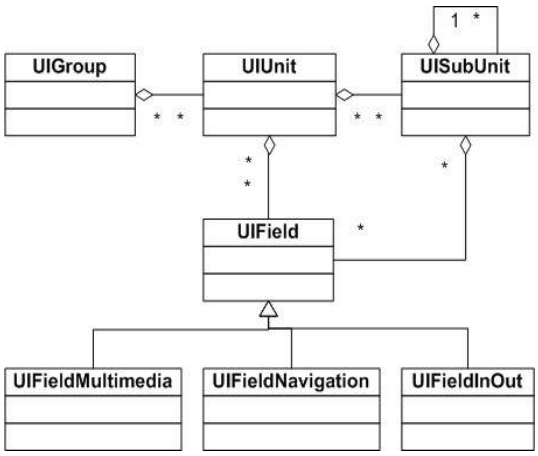
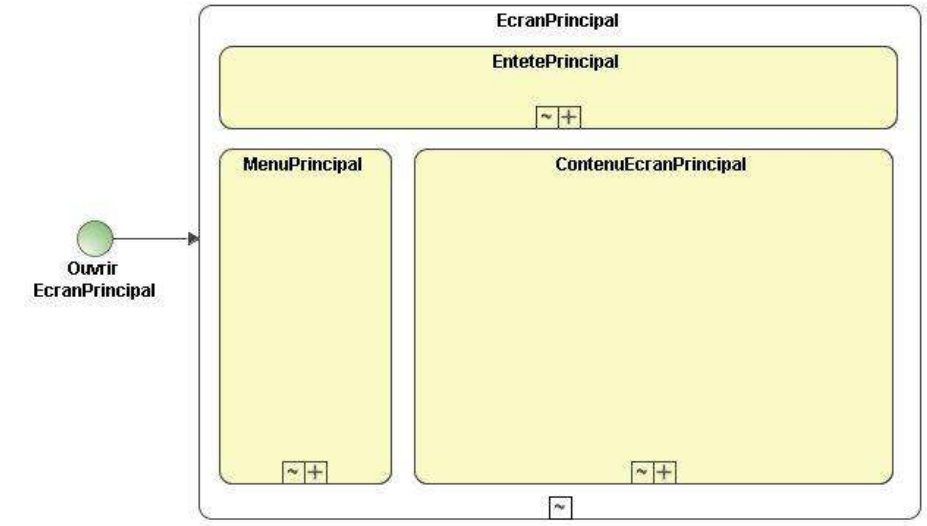


Figure 6. *Extrait de la hiérarchie de composants d'IHM.*

Les éléments d'interactions sont d'abord regroupés dans des éléments de type UIGroup qui correspondent à des groupes logiques d'interactions du point de vue métier. Ces groupes logiques contiennent ensuite des unités logiques d'interactions dénommées UIUnit qui regroupent des éléments d'interactions qui ne peuvent pas être séparés du point de vue de la logique

métier lorsqu'ils sont présentés à l'utilisateur contrairement aux UIGroup qui peuvent se présenter sous la forme de plusieurs écrans par exemple (un écran par UIUnit). Un exemple de définition du contenu d'un UIGroup et un exemple possible de représentation graphique de cet UIGroup sont donnés à la figure 7.

Dans cet exemple, la plateforme cible choisie est une application de type PC. Si l'application devait être déployée sur un téléphone portable, chaque UIUnit pourrait être représenté, par exemple, par un onglet qui une fois sélectionné permettrait d'accéder au contenu de l'UIUnit en question. Ce qui signifie, que sur un téléphone portable, les trois UIUnit définis dans l'exemple précédent ne pourraient pas être vus simultanément mais uniquement de façon séparée. Par contre, dès qu'un UIUnit est sélectionné, le contenu de celui-ci est affiché en un bloc.



(a)

(b)

Figure 7. *Exemple de modèle statique d'interface (a) avec une de ses représentations graphiques possible (b)*

Les UIUnit sont constitués d'UIElement, qui sont des éléments d'interactions entre l'utilisateur et l'application comme des éléments de navigation, des éléments d'entrée de données ou encore des éléments de type multimédia. Ils peuvent aussi être constitués d'UISubUnit qui sont des regroupements d'UIElement réutilisables au sein de plusieurs UIUnit différents. L'ensemble de ces éléments (UIGroup, UIUnit, UISubUnit et UIElement) sont prévus pour être facilement réutilisables par plusieurs applications permettant ainsi d'envisager la création d'application par simple assemblage d'éléments existants. Ils disposent aussi de propriétés qui permettent de préciser leur comportement ainsi que, si nécessaire, leur lien avec l'ontologie du domaine. Ils disposent aussi d'un certain nombre d'actions par défaut qui leur permettent d'hériter de comportements facilement adaptables par la personne en charge de la modélisation.

Les UIElement disposent aussi de propriétés permettant d'indiquer, au niveau du modèle, un certain nombre d'informations concernant leur adaptation aux différents contextes physiques d'utilisation ; c'est-à-dire par rapport aux différentes plateformes

utilisateurs. Ces propriétés permettent de prendre en compte aussi bien une adaptation du contenu qu'une adaptation au contenant en prenant en compte le fait que certains UIElement ne soient pas disponibles pour tous les types de plateformes. Ainsi, pour des éléments de type graphiques, il est possible d'associer des messages de remplacement dans le cas où l'image ne pourrait être affichée. Dans les modèles statiques d'interactions, il est possible de définir des UIElement qui ne font pas partie des processus métier comme par exemple une image pour afficher le logo d'une société ou alors un champ texte pour afficher des informations d'aide relative à l'UIUnit en cours d'utilisation.

Le cadre général étant fixé, nous allons maintenant présenter les grands principes d'adaptation des éléments d'interaction que nous avons retenus dans notre approche.

4.5.2 Les grands principes d'adaptation choisis dans le cadre de notre approche

En premier lieu, tous les éléments d'interactions peuvent être définis suivant trois niveaux de plateformes utilisateurs (High, Medium et Low), ce qui permet par exemple de définir trois images de qualité différentes pour un élément d'interactions de type UIPicture (élément d'affichage d'une image). Pour donner un exemple, pour un champ de type texte, le niveau de plateforme « High » signifie que la longueur du champ n'est pas limitée en nombre de caractères affichables. Le niveau « Medium » signifie que la longueur des champs texte affichés par cette plateforme est limité à 20 caractères maximum. Le niveau « Low » signifie que la longueur des champs texte affichables sur cette plateforme est limitée à 8 caractères. Il est important de noter que les choix de 20 et de 8 caractères ont pour l'instant été faits de manière arbitraire dans le cadre de notre approche et qu'ils devront être validés par une étude comparative des capacités techniques des différentes plateformes d'IHM existantes. Afin de pouvoir gérer le maximum de cas, les niveaux de plateforme pour les différents types d'éléments d'interactions ne sont pas obligatoirement les mêmes pour une plateforme bien définie. Ainsi, il est tout à fait possible de dire qu'une plateforme n'est capable de prendre en charge que des champs texte de niveau « Medium », des images de niveau « High » et des vidéo de niveau « Low ».

En second lieu, chaque type d'élément d'interaction peut avoir une propriété d'importance vis-à-vis du processus métier qui lui est associé permettant de définir si cet élément doit ou ne doit pas être implémenté au niveau de la plateforme finale en fonction des capacités de celle-ci. Par exemple, si on a dans un UIUnit des éléments textuels d'information qui ne servent pas directement à la réalisation du processus métier associé, on pourra indiquer que ceux-ci ont une priorité basse et que par conséquent ils pourront ne pas être utilisés et affichés dans des plateformes fortement contraintes en taille. Ainsi, supposons que dans la définition d'un UIUnit, on indique que celui-ci contient des images mais qu'elles ne sont pas indispensables à l'exécution du processus métier associé comme par exemple l'affichage de photos d'un lieu sur une fiche d'information touristique. Il est alors tout à fait possible d'indiquer que ces images ne seront affichées que si la plateforme utilisateur dispose d'un écran suffisamment grand pour permettre l'affichage des photos et des informations textuelles en simultané. Dans le cas contraire, seules les informations seront affichées. Ainsi, sur un PC, on aurait les images et les informations textuelles et sur un téléphone portable que les informations textuelles.

Si ce type d'adaptation n'est pas nouveau dans ses principes puisqu'on retrouve une partie de ces éléments dans d'autres approches comme UsiXML, la possibilité de mixer les différents niveaux de chaque élément d'interactions pour définir une plateforme permet d'assurer une bonne adaptation à chaque type de plateforme et ainsi de pouvoir, en théorie, rapidement intégrer une nouvelle plateforme à une application sans avoir besoin de changer les modèles conceptuels.

4.5.3 Exemple de définition d'un modèle statique d'interaction

La figure 8 présente un exemple de définition d'un modèle statique d'interaction pour notre cadre d'application. Cet exemple ne traite que de l'interface associée à l'entrée des informations de recherche d'itinéraire par l'utilisateur. Dans cet exemple, l'élément central d'interaction est un élément de type UIUnit portant le nom de SaisieInfoHoraire, conformément à ce qui a été indiqué dans le modèle de processus métier. Cet UIUnit contient aussi un certain nombre d'UIElement qui permettent d'afficher des contenus textuels statiques, de définir des champs de saisie d'information ou encore de définir des éléments d'interaction permettant à l'utilisateur de lancer des actions à exécuter.

Dans cet exemple, on pourra remarquer qu'on a associé à l'élément d'interaction de type UIUnit, à travers un artefact « UIUnitArtifact » une propriété indiquant qu'il est lié au concept de domaine « DemandeHoraire ». Pour des raisons de simplification, nous n'avons pas présenté sur cet exemple les propriétés des différents UIElement présents sur le diagramme mais chaque UIElement de ce modèle dispose de propriétés qui lui sont propres. Si celles-ci ne sont pas définies de manière explicite alors ce sont des propriétés par défaut qui sont utilisées.

Afin de faciliter la lecture des modèles statiques, notre approche a défini un code couleur permettant d'identifier facilement chaque type d'éléments (non présenté car non visible dans cet article en noir et blanc).

Par rapport aux solutions déjà existantes, l'utilisation de la notation BPMN apporte un certain nombre d'avantages : (1) Elle permet d'utiliser la notion de zoom sur des éléments d'interactions (un signe + sur un élément indiquant que celui-ci est détaillé dans un autre modèle). (2) Elle permet d'assurer une certaine cohérence avec le modèle de processus métier à travers l'utilisation d'un formalisme similaire.

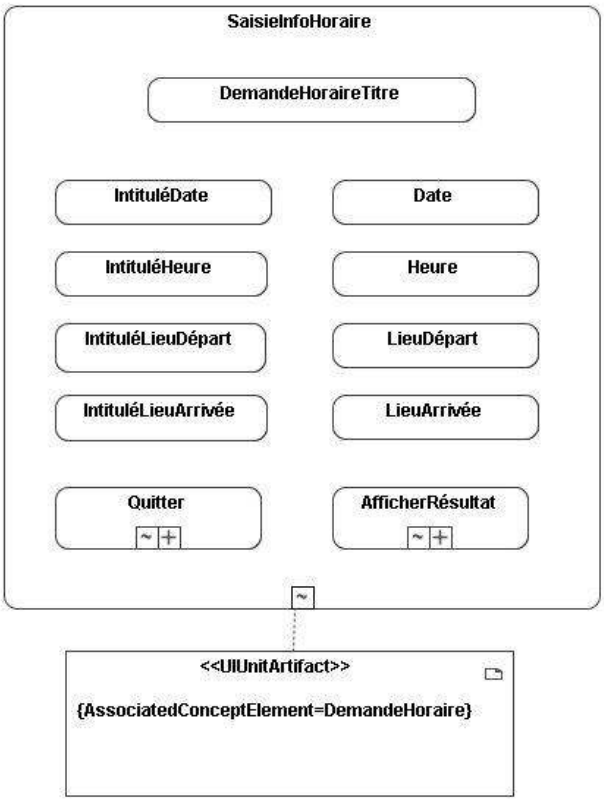


Figure 8. *Exemple de modèle statique d'interface pour notre exemple de demande d'application de demande d'horaires de transport*

Dans l'état actuel d'avancement de nos travaux, nous avons pu vérifier la validité de notre approche dans le cadre de la réalisation d'une application interactive dans le domaine des transports (voir figure 7 et 11) dont les interfaces sont constituées de fenêtres, de boutons, de zones de saisie, de zones d'affichage et d'éléments graphiques de type images.

4.6 Le modèle dynamique d'interface

Le modèle dynamique d'interface, complémentaire au modèle des processus métier, permet de modéliser l'ensemble des possibilités d'interactions au sein d'un élément de type UIUnit. Les UIUnit étant assimilés à des ensembles d'éléments d'interactions, ils ne sont pas obligatoirement liés à un seul processus métier mais peuvent être utilisés par plusieurs processus métier et chaque processus métier n'a pas forcément besoin de tous les UIElement contenu dans l'UIUnit pour son déroulement correct. Pour prendre un exemple simple, un UIUnit contenant un menu de navigation pourra être utilisé par plusieurs processus, chaque processus pouvant par exemple être associé à une entrée de ce menu.

Il est important de noter ici que les UIGroup ne disposent pas de modèles dynamiques d'interface spécifiques mais utilisent les modèles dynamiques d'interfaces des différents UIUnit qui les composent. Ceci s'explique par le fait que les UIUnit sont composés d'éléments d'interactions et donc peuvent être affichés à l'écran alors que les UIGroup ne sont composés que d'UIUnit et ne possèdent aucun autre type d'élément d'interaction. Au niveau d'un UIgroup, la seule possibilité offerte à un utilisateur est de sélectionner un des UIUnit contenu dans l'UIGroup. La problématique de synchronisation entre les différents modèles pouvant être importante, nous avons décidé de considérer dans notre approche que, dans un UIGroup, les différents modèles dynamiques des UIUnit qui le composent sont indépendants les uns des autres. Ceci signifie que les UIUnit peuvent être manipulés par les utilisateurs indépendamment les uns des autres. Par exemple, si dans un UIGroup un UIUnit contient un menu général de navigation au sein de l'application et qu'un autre UIUnit représente un formulaire de saisie, l'utilisation du menu par l'utilisateur est totalement indépendante de l'utilisation du formulaire. Dans le cas ou des interactions devraient être établies entre deux modèles dynamiques d'UIUnit, comme par exemple un passage d'informations, celles-ci devront être modélisées de manière explicite au niveau du modèle dynamique de chaque UIUnit concerné.

Pour rappel (cf. 4.4), les modèles dynamiques d'interaction contiennent des tâches provenant des modèles de processus mais aussi des tâches, comme par exemple des tâches de validation, qui complètent les modèles de processus. Ces tâches ne sont pas indispensables à la définition et à la compréhension du processus métier associé mais le sont à son bon fonctionnement. Le choix d'inclure ou non une tâche dans le modèle de processus est défini par un guide de modélisation des processus métier qui a été établi dans le cadre de notre approche et qui est prévu pour être évolutif afin de pouvoir prendre en compte à l'avenir des situations non encore rencontrées dans le cadre de la réalisation de nos applications de test.

Comme pour la modélisation des processus métier, les modèles dynamiques s'appuient sur : (1) L'utilisation d'une notation complémentaire à BPMN comme par exemple au niveau des portes logiques l'utilisation du préfixe « Valider » pour indiquer qu'on va valider la valeur d'une propriété d'un concept. Si cette propriété porte un nom unique pour tous les concepts du domaine,

il n'est pas nécessaire de préciser le concept. Dans le cas contraire, on doit utiliser une notation du type « Valider nomconcept.nompropriété » en remplaçant « nomconcept » par le nom du concept et « nompropriété » par le nom de la propriété de ce concept qu'on veut valider. Dans le cadre d'une porte logique de validation, celle-ci peut avoir deux sorties : une sortie par défaut qui signifie que la validation a été effectuée avec succès et une sortie « erreur » qui indique les tâches à réaliser en cas d'erreur. (2) L'utilisation de tâches prédéfinies qui permettent d'effectuer certaines actions déterminées comme la gestion des erreurs après une validation à l'aide d'une tâche ayant comme préfixe le terme « TraiterErreur » suivi du nom de la propriété. De manière pratique, on associe au traitement d'erreur l'affichage d'un message à l'utilisateur lui permettant de connaître la raison de l'erreur et lui indiquant comment y remédier. (3) L'utilisation d'un code couleur pour différencier les tâches effectuées par l'utilisateur des tâches effectuées par le système.

L'utilisation de BPMN introduit, en théorie la possibilité, de vérifier, à l'aide de réseaux de Petri, qu'il n'y a pas de point de blocage et qu'il n'y a pas de zones non accessibles au niveau de l'UIUnit modélisée dans le modèle dynamique. Si ceci permet une validation de l'UIUnit du point de vue de son contenu et des interactions qui lui sont associées, elle permet aussi de valider, de manière théorique, les fonctionnalités métier qui sont associées à l'UIUnit. Il est important de noter qu'on ne procède pas ici à une validation ergonomique. Ceci représente néanmoins un gros avantage par rapport aux autres méthodes de modélisation conceptuelle qui elles ne disposent pas d'outils de validation au niveau conceptuel. Pour l'instant, nous n'avons pu vérifier ce principe qu'à travers des validations manuelles des modèles dynamiques ; ce qui a permis d'en vérifier tout l'intérêt. Dans le cadre de nos futurs travaux, nous avons l'intention de développer des outils de validation permettant de faire ces validations à l'aide des réseaux de Petri de manière automatique.

Dans la figure 9, on trouvera un exemple de modèle dynamique d'interaction pour notre cadre d'application. Sur cette figure, on pourra noter l'utilisation particulière du formalisme BPMN à travers l'indication de liens représentés par un rond avec une flèche à l'intérieur qui permettent d'établir des relations entre deux diagramme différents ; dans notre cas entre deux modèles dynamiques d'interaction. La notation associée à ce lien est spécifique à notre approche. Elle commence par un préfixe « UIUnitStart » suivi du nom du diagramme dynamique vers lequel se fait le lien. Ceci impose de commencer chaque diagramme dynamique par un lien commençant par le préfixe « UIUnitStart » suivi du nom du diagramme dynamique.

Pour résumer, le modèle dynamique d'interaction présente deux gros avantages : (1) il permet de modéliser l'ensemble des interactions possibles au niveau d'un ensemble d'éléments d'interaction de type UIGroup ; (2) en s'appuyant sur le formalisme des réseaux de Petri associés à la notation BPMN, il devrait permettre de pouvoir valider les interfaces du point de vue dynamique.

Mais il présente aussi deux inconvénients : (1) il reprend un certain nombre d'informations déjà contenues dans les modèles des processus métier ce qui entraîne des doublons et donc des risques d'erreur et des lourdeurs au niveau de la maintenance des modèles ; (2) il ne permet pas pour l'instant de prendre en compte les différents niveaux d'interfaces utilisateur or comme les niveaux influent sur le contenu de l'UIUnit affiché à l'utilisateur, ils influent aussi sur les différentes possibilités d'interactions de l'UIUnit pour chaque niveau d'interface.

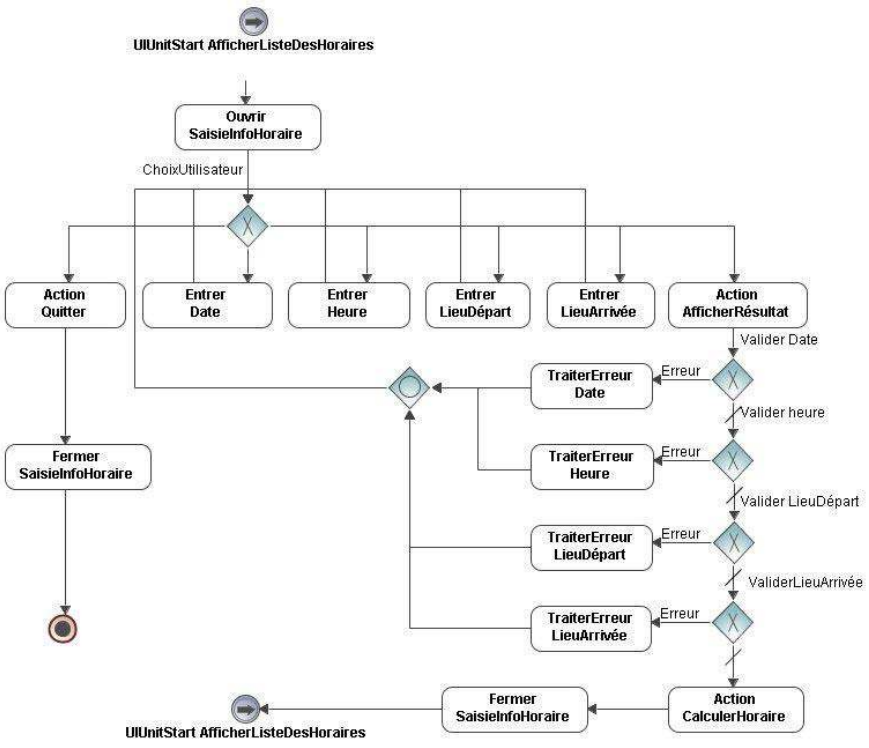


Figure 9. *Exemple de modèle dynamique d'interface pour l'interface de saisie des informations pour une application de demande d'horaires de transport*

L'ensemble des modèles conceptuels d'interactions ayant été présenté, il convient maintenant de s'intéresser au passage des modèles conceptuels d'interactions vers les modèles concrets d'interaction directement manipulables par les utilisateurs.

5. Le passage du niveau CIM au niveau PSM

5.1 La notion de services de niveau PIM

Une des originalités de notre approche, qui ne se trouve dans aucune autre approche actuelle de modélisation des IHM, réside dans le principe de définition et d'utilisation de services fonctionnels au niveau PIM.

Un service fonctionnel est un ensemble de traitements permettant d'exécuter une opération précise sur l'élément auquel il est rattaché. Un service fonctionnel peut être attaché à n'importe quel élément d'interaction. Pour donner un exemple, la gestion des informations linguistiques, et plus précisément de l'adaptation des interfaces à la langue de l'utilisateur, est un service fonctionnel qui peut être utilisé au niveau CIM mais qui n'a pas besoin d'y être défini. Ce qui est important pour la personne en charge de la modélisation, ce n'est pas de savoir comment fonctionne cette adaptation mais juste d'indiquer si on veut utiliser ou pas ce type de personnalisation dans tel ou tel processus métier.

La différence entre les services fonctionnels et les éléments d'adaptation, présentés en 4.5.2, réside dans le fait que les éléments d'adaptation, du point de vue métier, ne correspondent à aucun traitement. En fait, ils servent uniquement à la prise en compte de la diversité technique des plateformes d'exécution des applications à travers la définition de deux propriétés conceptuelles qui sont le niveau technique de la plateforme et le niveau d'importance de l'élément d'interaction.

Pour qu'une opération ou un ensemble d'opérations devienne un service fonctionnel, il faut que : (1) il ne fasse pas directement partie d'un processus métier sinon c'est une action ; (2) il puisse être associé à un ou plusieurs éléments d'interaction ; principalement à des UIElement ; (3) il soit susceptible d'être utilisé par plusieurs applications. Si un ensemble d'opérations n'est utilisé que par une application, il n'est pas forcément pertinent de le transformer en service fonctionnel.

Au niveau de leur nature même, les services fonctionnels peuvent être classés en deux catégories :

- Les services explicites qui peuvent être appelés directement au niveau des modèles en fonction des besoins de la personne en charge de la modélisation comme par exemple la personnalisation des contenus. Pour information, ce type de personnalisation correspond à une adaptation des contenus d'informations retournés à l'utilisateur en fonction de son profil, de ses préférences ou encore en fonction de recommandations issues d'une communauté d'utilisateurs ayant un profil assez similaire à celui de l'utilisateur. On pourra donner comme exemple de personnalisation des contenus la personnalisation effectuée sur un site web commercial comme celui de la société Amazon (http://www.amazon.fr).

- Les services implicites qui sont automatiquement appelés par les applications comme par exemple la gestion des sessions utilisateurs ou encore la gestion des accès aux données. La définition de l'utilisation ou non des services implicites se fera au niveau PIM à travers la définition de règles d'utilisation de ces services.

Si le contenu et le fonctionnement des services fonctionnels sont définis avec les experts métier, leur orientation technique implique une modélisation et un développement classique par des informaticiens (passage par un modèle de type UML de niveau PIM qui est transformé ensuite en modèles de niveau PSM). Afin de permettre un développement homogène de ces services, un guide de développement a été établi pour indiquer l'ensemble des règles à respecter aussi bien au niveau de la définition du service fonctionnel que de son développement. Ce guide, qui pour l'instant ne reprend qu'un certain nombre de règles techniques simples, devra dans la suite de nos travaux être revu et étendu afin de pouvoir assurer que les services fonctionnels développés soient facilement intégrables dans l'architecture de niveau CIM, soient sans interaction avec les autres services fonctionnels déjà développés et surtout respectent des critères de qualité aussi bien du point de vue métier que du point de vue technique. Ceci nécessitera de définir un cadre de validation de l'ensemble de ces règles.

Pour fixer les idées sur ce que peut être un service fonctionnel et comment il est appelé au niveau CIM, nous allons prendre comme exemple un service de personnalisation des contenus tel que nous l'avons défini précédemment. Du point de vue des processus métier, la personnalisation des contenus ne fait pas partie des processus à moins qu'elle ne soit directement demandée par l'utilisateur mais, même dans ce cas, le processus métier considéré ne définit pas la fonctionnalité de personnalisation mais ne fait que l'utiliser. Ceci signifie que ce qui est intéressant d'indiquer au niveau des processus métier c'est juste d'indiquer qu'on veut utiliser la fonctionnalité de personnalisation des contenus et pas de définir le fonctionnement de cette personnalisation.

Le service de personnalisation des contenus pourra par exemple prendre en charge plusieurs types de personnalisation : une personnalisation géographique dépendant de la localisation de l'utilisateur, une personnalisation linguistique dépendant de la langue de l'utilisateur, une personnalisation en fonction des préférences personnelles de l'utilisateur et une personnalisation en fonction de préférences communautaires. Au niveau des modèles conceptuels, ceci se traduira par la possibilité de définir la valeur de quatre propriétés booléennes au niveau du service de personnalisation indiquant l'activation ou pas d'un des quatre types de personnalisation définis ci-dessus (cf. figure 10).

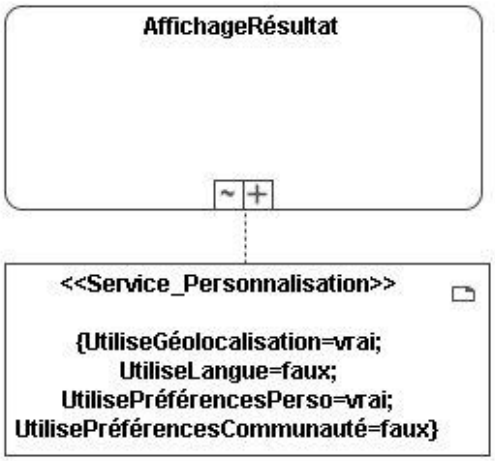


Figure 10. *Exemple d'association d'un service de personnalisation à un élément d'interaction de type UIUnit au niveau d'un modèle statique d'IHM*

Afin de faciliter l'utilisation des services fonctionnels, chaque service disposera de valeurs par défaut valables pour l'ensemble des applications et de valeurs par défaut valables spécifiquement pour une application ou un ensemble d'applications. Ainsi, au niveau des modèles, on n'affichera que les propriétés qui ne correspondent pas aux valeurs par défaut afin d'en faciliter la lecture. Ainsi, si on définit pour une application que toutes les informations seront personnalisées en fonction de la localisation géographique de l'utilisateur, il ne sera plus nécessaire de l'indiquer dans les différents modèles d'interaction associés à cette application à moins qu'on veuille indiquer explicitement qu'on ne veut pas utiliser cette personnalisation géographique pour un type d'information bien particulier.

La définition des valeurs des propriétés des services pouvant varier en fonction de règles métier prédéfinies ou de choix effectués dynamiquement par l'utilisateur, il est possible de définir pour un même élément d'interaction plusieurs ensembles de propriétés qui seront activés ou pas en fonction de l'évaluation des règles métier et/ou en fonction des choix de l'utilisateur. Afin d'assurer un fonctionnement cohérent des applications, notre méthode impose la création de valeurs par défaut qui seront sélectionnées dans le cas où aucune des règles ne seraient valides.

En résumé, ces services, qui sont utilisables au niveau des modèles conceptuels permettent de simplifier les modèles mais surtout permettent d'apporter une certaine homogénéité au sein des applications. Ainsi, si on prend à nouveau comme exemple un service de personnalisation des contenus, la définition de cette personnalisation en tant que service permet d'assurer : (1) une cohérence de fonctionnement de la personnalisation des contenus au niveau de l'ensemble des applications étant donné que celle-ci sera effectuée de la même façon pour toutes les applications ; (2) une plus grande souplesse au niveau de la maintenance du service de personnalisation, celle-ci ne se faisant qu'à un seul endroit pour l'ensemble des applications.

En conclusion, la définition de services fonctionnels et techniques au niveau PIM présente les avantages suivants : (1) une simplification des modèles conceptuels, l'appel des services se faisant de manière explicite à travers l'utilisation de propriétés ou de manière implicite directement lors du passage du niveau CIM au niveau PIM ; (2) une homogénéisation du fonctionnement des services fonctionnels, ceux-ci n'étant définis qu'à un seul endroit pour l'ensemble des applications ; ce qui en facilite aussi la maintenance ; (3) une fiabilisation accrue des applications à travers la possibilité de valider ces services de niveau PIM avant de permettre leur utilisation dans les modèles de niveau CIM. A ce sujet, il est important de rappeler que, dans notre approche, les services fonctionnels et techniques de niveau PIM ne sont pas modifiables au niveau CIM.

La notion de service fonctionnel et technique ayant été présentée, nous allons pouvoir expliquer maintenant comment dans notre méthode nous passons des modèles conceptuels aux applications concrètes.

5.2 Les transformations effectuées pour passer du niveau CIM au niveau PSM

Avant toute chose, il est important de rappeler que notre approche repose sur l'utilisation de plusieurs niveaux de framework (cf. 3) :

- Un framework de niveau PIM capable de prendre en charge l'ensemble des services fonctionnels et techniques explicites et implicites (cf. 5.1).

- Un premier framework de niveau PSM permettant de prendre en charge l'ensemble des services fonctionnels qui sont alors adaptés, si nécessaire, à chaque famille d'interface. Ce framework permet aussi de traduire les éléments de type UIField en éléments graphiques adaptés à la plateforme ; un UIField pouvant être associé à un ou plusieurs éléments graphiques différents en fonction des propriétés qui lui ont été associées au niveau du modèle statique d'interaction. Pour donner un exemple, un élément d'interaction de type zone de texte devant afficher des informations contenues dans une liste pourra être affiché sur une plateforme sous la forme d'une liste déroulante et sur une autre sous la forme d'une zone de texte multilignes possédant des ascenseurs de navigation pour pouvoir visualiser l'ensemble des informations.

- Un deuxième framework de niveau PSM qui permet de préciser, si nécessaire, les différents éléments en fonction d'une plateforme utilisateur bien précise. Ceci permet de prendre en compte les particularités qui peuvent exister sur chaque plateforme d'une même famille comme une taille d'écran un peu différente.

Dans le cadre de nos travaux, une première application simple d'affichage d'informations transport utilisant l'ensemble des modèles d'interactions a été réalisée nous permettant d'en valider les grands principes (cf. figures 7 et 11).

L'objectif de notre approche étant de générer de manière semi-automatique les applications à partir des modèles conceptuels, la génération des applications se fait aujourd'hui en plusieurs étapes :

- Étape 1 : Transformation des modèles statiques d'interaction en fichiers XML. Pour chaque élément, le fichier XML associé contient l'ensemble des informations concernant le contenu, les propriétés de l'élément et des services techniques et fonctionnels éventuellement associés et les actions qui lui sont attachées.

- Étape 2 : Au niveau PIM, modification des contenus des fichiers XML par ajout, manuellement pour chaque type d'UIElement, d'informations concernant le positionnement général des éléments sur une interface de type graphique (positionnement à gauche, à droite, en haut, en bas, etc.). Remarque : Un positionnement par défaut est proposé lors du passage du niveau CIM au niveau PIM en fonction du positionnement des éléments au niveau des modèles statiques d'interactions. Aussi, la redéfinition manuelle du positionnement de chaque élément n'est pas obligatoire.

Figure 11. *Exemples d'interfaces réalisées pour (a) un formulaire de création d'un profil utilisateur et (b) pour une application d'information voyageur*

- Étape 3 : Passage des fichiers XML au premier niveau PSM regroupant les règles et la charte graphique associées à chaque type de famille de plateformes utilisateur et ajout, de manière automatique dans le fichier XML, à partir des règles de transformations, pour chaque famille de plateforme utilisateur cible, d'informations génériques permettant de faire un premier positionnement graphique de chaque élément (positionnement en X et Y pour une interface en deux dimensions). Puis on

applique la charte graphique à l'ensemble des éléments. Dans le cadre de notre approche, nous avons défini la charte graphique comme étant un ensemble d'informations permettant de savoir quelles sont les polices de caractères standard à utiliser dans l'interface (nom, taille, couleur, affichage en majuscules, etc.) et quelles sont les couleurs de base d'un élément d'interaction (couleur de fond d'écran, couleur de fond des éléments graphiques de type bouton de navigation, couleur des barres de titre, etc.).

- Étape 4 : Passage des fichiers XML au deuxième niveau PSM contenant les règles et la charte graphique spécifique à un type précis de plateforme utilisateur et ajout automatique éventuel, à partir de règles de transformation, d'informations permettant de préciser le positionnement graphique de certains éléments ainsi que les éléments spécifiques de la charte graphique à appliquer par rapport à ceux de la famille de plateformes ajoutés lors de l'étape précédente.

- Étape 5 : Modification manuelle, si nécessaire, des informations de positionnement des différents éléments graphiques et ajout des informations de présentation spécifiques pour chaque élément à particulariser (taille des caractères, polices de caractères, couleur, etc.). Cette modification manuelle peut se faire aussi bien au premier niveau qu'au deuxième niveau de notre architecture PSM.

Que ce soit au niveau PIM ou au niveau PSM, la prise en charge des fichiers XML se fait à travers l'utilisation des différents frameworks (cf. 3). Ceci signifie qu'à part pour les actions, qui ne dépendent pas directement des modèles d'interactions mais qui sont utilisées par celui-ci, le passage du niveau CIM au niveau PSM se traduit par une succession de transformations et d'ajouts dans les fichiers XML de définition des interactions.

Le choix du langage XML, dans le cadre de la transformation des modèles conceptuels en applications concrètes, correspond à un choix technique effectué pour permettre une première validation de notre approche dans le cadre de la réalisation d'un premier démonstrateur. Dans la suite de nos travaux, lorsque nous aurons définis les méta-modèles de l'ensemble de nos langages de modélisation conceptuels, la pertinence du choix de l'utilisation du langage XML sera réévaluée.

Pour pouvoir être utilisée, notre approche impose, en préalable, la définition d'un certain nombre de fonctionnalités et de services au niveau des frameworks (cf. 5.1). Cette façon de faire présente les avantages suivants :

- Diminution de la complexité des modèles au niveau CIM à travers la prise en charge d'une partie de la complexité liée à la modélisation des applications aux niveaux PIM (services fonctionnels et techniques) et PSM (gestion des informations associées à la représentation graphique des éléments d'interaction).

- Plus grande souplesse au niveau de la maintenance des fonctions et services techniques et fonctionnels offerts par les frameworks car ceux-ci ne sont pas directement liés aux modèles conceptuels des applications.

- Permet d'assurer des expériences cohérentes pour les utilisateurs des différentes applications à travers une certaine homogénéisation des comportements des applications à travers l'utilisation des services techniques et fonctionnels offerts au niveau PIM et surtout à travers une homogénéisation de l'apparence graphique des applications sur chaque plateforme utilisateur.

- Possibilité de créer des applications qui fonctionnent de manière dynamique ; chaque élément d'interaction étant redéfini à l'écran lors de son utilisation en fonction des informations contenues dans les fichiers XML associés aux éléments d'interaction. Ceci permet d'envisager la répercussion immédiate au niveau de l'application physique de toute modification effectuée au niveau des modèles conceptuels.

L'utilisation des frameworks présente aussi un certain nombre d'inconvénients : (1) nécessité d'avoir des framework définis et fonctionnels avant de pouvoir créer le moindre modèle. La question est alors de savoir comment les définir ; (2) si on a besoin de définir un nouveau service fonctionnel, on doit le définir dans un modèle à part, le développer et tester avant de pouvoir l'utiliser ce qui peut entraîner un retard au niveau des développements ; (3) les fonctions et les services pouvant être spécifiques à un domaine applicatif, le transfert de modèles d'un domaine applicatif à un autre peut poser problème.

L'utilisation des frameworks, si elle simplifie beaucoup la création et la lecture des modèles nécessite une assez longue phase de mise au point et de stabilisation afin de pouvoir s'assurer qu'on dispose bien de l'ensemble des fonctions nécessaires à la prise en charge complète d'un domaine applicatif. Notre méthode ne sera donc vraiment utilisable que lorsqu'on aura défini des frameworks de base, standardisés au niveau de leur contenu, capables de prendre en charge les éléments communs de l'ensemble des domaines applicatifs existants. Une fois cette étape réalisée, il sera alors possible de définir des frameworks spécialisés adaptés aux spécificités de chaque domaine pour lesquels on pourra établir des règles de correspondance afin de faciliter le passage des modèles d'application d'un framework de domaine à un autre. Il est intéressant ici de noter que la notion de framework de services est reprise en partie dans des produits commerciaux comme « Microsoft Enterprise Library 3.0 » (Microsoft, 2007).

6. Évaluation de notre approche

Dans l'état actuel de nos travaux, nous n'avons pas encore établi de grille d'analyse complète nous permettant d'évaluer notre approche par rapport à une approche idéale de modélisation conceptuelle d'IHM. Néanmoins, à travers le développement d'une application dans le domaine des transports, nous avons pu réaliser une première validation de celle-ci (la transformation des modèles conceptuels en modèles de niveau PIM ayant été faite de manière manuelle lors de cette première validation). À partir de cette première expérience, nous avons pu faire une première évaluation de notre approche suivant les cinq critères de (Selic, 2003).

Abstraction : À travers l'utilisation d'une ontologie, de notre notation enrichissant BPMN et de modèle statiques définis indépendamment des technologies, elle permet de définir des modèles indépendants des solutions techniques qui seront utilisées lors de la réalisation concrète de l'application.

Compréhensibilité : Le formalisme utilisé, BPMN, est prévu pour être facilement compréhensible et manipulable par des experts métier (c'est en effet un des objectifs que s'étaient fixé les concepteurs de cette notation). Ceci a pu être informellement confirmé lors d'une première série de validation de notre approche initiée auprès d'experts métier. Néanmoins, le nombre limité d'experts métier interrogés pour l'instant ne nous permet pas encore de conclure sur ce point.

Pertinence : Les trois modèles proposés permettent d'appréhender la problématique de la modélisation des IHM suivant trois angles complémentaires. Ainsi, on dispose d'un premier type de modèle qui permet de prendre en compte les processus métier et de rattacher ceux-ci à des IHM ; ce qui permet d'associer aux IHM des objectifs métiers. Ensuite, on dispose d'un deuxième type de modèle, le modèle statique, qui permet de définir l'IHM du point de vue statique, c'est-à-dire d'indiquer comment les éléments d'interaction sont regroupés entres eux. Puis on trouve un troisième type de modèle, le modèle dynamique, qui permet de définir toutes les interactions possibles au niveau d'un ensemble d'éléments d'interaction regroupés au sein d'un UIUnit. Enfin, on dispose des autres modèles de notre approche globale qui permettent d'utiliser au niveau des modèles d'interactions des règles métier, des informations géographiques, etc. (cf. 3).

Prévisibilité : À travers l'utilisation des réseaux de Petri au niveau des modèles d'interaction, notre approche devrait permettre d'offrir un certain niveau de validation fonctionnelle des IHM dès la modélisation conceptuelle. Pour l'instant, les validations étudiées, de manière manuelle, sont : la possibilité de vérifier que tous les éléments d'interactions sont bien utilisés et la possibilité de vérifier que le déroulement d'un processus métier ne conduit pas à un blocage ; c'est-à-dire à l'impossibilité de terminer celui-ci.

Faible coût : Chaque modèle est conçu pour être réutilisable. Ainsi, les processus métier et les ensembles d'interaction de type UIUnit ne sont pas liés à une application spécifique et peuvent de ce fait être utilisés par plusieurs applications. Ceci permet de traduire le fait qu'un ensemble d'éléments d'interaction comme un formulaire de saisie puisse être utilisé par plusieurs applications différentes et dans le cadre de plusieurs processus métier différents. Cette réutilisation possible des modèles diminue les coûts de développement en permettant de créer de nouvelles IHM par simple assemblage de modèles existants. D'autre part, la possibilité de valider en partie les modèles au niveau conceptuel permet de diminuer très fortement les coûts au niveau des tests fonctionnels. Enfin, les modèles sont prévus pour être manipulables par des experts métier ; ce qui permet de diminuer le risque d'erreur lors du passage d'informations entre les experts métier et les personnes en charge de la réalisation de l'application du point de vue informatique.

En conclusion, dans l'état actuel de nos travaux, notre approche semble donc bien répondre aux différents critères énoncés par (Sottet, 2003) et, en ce sens, représente une certaine avancée dans le domaine de la modélisation des IHM.

Néanmoins, il est clair que nous devrons à l'avenir définir des grilles d'évaluation beaucoup plus précises afin de permettre de comparer notre approche par rapport aux autres approches existantes, aussi bien dans le domaine de la recherche que dans le domaine des produits commerciaux.

7. Conclusions et perspectives

L'ingénierie dirigée par les modèles, ou IDM, va très certainement devenir, dans les années qui viennent, le nouveau paradigme en matière de développements informatiques. Par rapport aux techniques actuelles de développement, les promesses de l'IDM sont grandes à travers la possibilité de créer des applications par simple assemblage de modèles existants et surtout la possibilité pour des non informaticiens, experts dans un domaine métier, de pouvoir créer eux-mêmes leurs propres applications à partir de modèles conceptuels en utilisant un formalisme adapté facile à comprendre et à manipuler pour eux. Malheureusement, pour l'instant, l'IDM n'étant un sujet de recherches que depuis quelques années, cette approche est loin d'être arrivée à maturité. Voilà pourquoi il n'existe pas encore, à notre connaissance, de méthodes ni d'outils vraiment utilisables qui permettent de créer de bout en bout des applications informatiques à partir d'une approche de type IDM. En fait, une des grandes questions qui se pose encore est de savoir si cette approche est vraiment réaliste ou si elle comporte une certaine dose d'utopisme notamment dans les aspects de modélisation conceptuelle qui, en principe, devraient être pris en charge par des experts métier. Pour pouvoir répondre en partie à cette question, nous avons décidé de proposer une méthode de conception d'IHM de type IDM qui, en s'appuyant sur une infrastructure de type MDA, permet de générer de manière semi-automatique des IHM pour différentes plateformes utilisateurs.

Notre approche, qui se base sur la notion de processus métier au niveau conceptuel et qui s'appuie sur un framework de services fonctionnels et techniques, présente, dans l'état actuel de nos travaux de recherche de nombreux avantages. En premier lieu, à travers l'utilisation de la notion de processus métier, les modèles sont théoriquement plus facilement compréhensibles et manipulables par des utilisateurs métier ; ce qui correspond bien à un des objectifs de l'IDM. D'autre part, les modèles conceptuels sont plus simples à créer grâce à l'utilisation du framework de services fonctionnels et techniques qui permet d'utiliser au niveau conceptuel, de manière explicite ou implicite, des ensembles techniques complexes comme par exemple des fonctionnalités de personnalisation des contenus affichés à l'utilisateur. Notre approche offre aussi, en théorie pour l'instant, la possibilité de valider les modèles conceptuels du point de vue fonctionnel en utilisant des réseaux de Petri ce qui devrait apporter des gains en temps et en qualité au niveau des développements informatiques. Elle permet une réutilisation des différents modèles

conceptuels permettant de simplifier les mises à jour et le développement de nouvelles applications ; chaque modèle pouvant être utilisé par une ou plusieurs applications. Enfin, à terme, elle devrait permettre une adaptation semi-automatique des interfaces au contexte d'utilisation à travers l'utilisation de règles d'adaptation définies au niveau PSM. Pour l'instant, seuls quelques types d'adaptations ont été validés, ce qui ne permet pas d'affirmer que tous les types d'adaptation conceptuelle pourront être pris en charge.

Dans l'état actuel de nos recherches, de nombreux travaux restent encore à faire pour arriver à une méthode réellement exploitable par des utilisateurs métier. En premier lieu, nous devons finaliser l'ensemble de nos modèles conceptuels définis au niveau de notre architecture globale. Ensuite, nous devons définir une approche de modélisation globale indiquant comment utiliser l'ensemble de nos modèles. Après cela, nous devons créer les outils capables de prendre en charge nos différents modèles et surtout capables de prendre en charge notre architecture globale. Étant donné que nous n'avons testé notre approche que dans le cadre limité d'une application de diffusion d'informations dans le domaine des transports, nous devons rendre notre formalisme plus générique de manière à lui permettre de prendre en charge un plus grand nombre de typologie d'applications. Étant donné que nous n'avons réalisé pour l'instant nos tests qu'en laboratoire et qu'au sein d'une équipe d'informaticiens, nous devrons à l'avenir valider notre formalisme et notre approche avec des experts métier dans le cadre de la réalisation concrète d'applications. De manière plus globale, nous devrons aussi étudier plus en détail les apports de la modélisation conceptuelle de niveau CIM par rapport aux modélisations de niveau PIM proposées aujourd'hui par les approches existantes. Nous devrons aussi étudier les avantages, les contraintes et les limites associées à une modélisation des applications réalisées par les experts métiers eux-mêmes. Enfin, nous devrons créer une grille d'évaluation des méthodes de modélisation des applications nous permettant de comparer notre approche avec les autres approches existantes.

En conclusion, nous pouvons dire que notre approche ouvre une nouvelle voie qui s'avère prometteuse pour une utilisation plus large des outils de modélisation et de génération semi-automatique des applications. Néanmoins, les nombreuses perspectives de recherche identifiées ne permettent pas pour l'instant de rendre celle-ci directement utilisable. À travers la réalisation d'une application complexe dans le domaine des transports, nous espérons pouvoir finaliser un grand nombre de points dans un proche avenir ; ce qui nous permettra de confirmer ou d'infirmer un certain nombre d'hypothèses posées dans le cadre de la réalisation de nos travaux.

8. Remerciements

Ce travail de recherche a été partiellement soutenu par le "Ministère de l'Education Nationale, de la Recherche et de la Technologie", la "Région Nord Pas-de-Calais" et le FEDER (projets MIAOU, EUCUE, SART), l'ANR ADEME (Viatic.Mobilité) et la PREDIM (MouverPerso). Les auteurs remercient l'ensemble de ces institutions pour leur support.

9. Bibliographie

Anli A., Petit-Rosé C., Grislin-Le Strugeon E., « Plate-forme d'intégration de services personalises à base d'agents logiciels », *Génie Logiciel*, n° 71, 2004, pp. 34-39

Bardon D., Bjerke C., Vredenbourg K., OVID Tutorial : Mastering the complexity of creating highly satisfying user experiences, 2002, société IBM, accessible à http://www-03.ibm.com/easy/page/2943.

Baron M., Lucquiaud V., Autard D., Scapin D.L., « K-MADe : un environnement pour le noyau du modèle de description de l'activité », 18ème conférence francophone sur l'Interaction Homme-Machine – IHM 2006, Montréal, Canada, 2006, ACM International Conference Proceeding Series, Vol. 133, pp 287-288.

Bernonville S., Kolski C., Beuscart-Zéphir M., « Contribution and limits of UML models for task modelling in a complex organizational context: case study in the healthcare domain », Proceedings of The 5th International Business Information Management Association Conference – IBIMA 2005, Le caire, Egypte, 2005, pp. 119-127

Brossard A., L'ingénierie dirigée par les modèles appliquée au développement d'interfaces graphiques personnalisées : une approche à travers les processus métier, Mémoire de Master Recherche, Université de Valenciennes, France, 2006.

Brossard A., Abed M., « Modélisation des IHM : Une approche basée sur les processus métier », Proceedings of Nouvelles tendances technologiques en génie électrique & informatique, GEI'2007, Sfax, Tunisie, 2007, pp 91-101.

Bézivin J, Blay M., Bouzhegoub M., Estublier J, Favre J.-M., Gérard S., Jézéquel J.-M, Rapport de synthèse de l'AS CNRS sur le MDA, AS MDA, décembre 2004, CNRS.

BPMI, Business Process Modeling Notation 1.0, 2004, accessible à http://www.bpmn.org .

Calvary G., Coutaz J., Daassi O., Balme L., Demeure A., « Towards a new Generation of widgets for supporting software plasticity: the comet », 9th IFIP Working Conference on Engineering for Human-Computer Interaction, Hambourg, Allemagne, 2003

Cook S., « Domain-Specific Modeling and Model Driven Architecture », *MDA Journal*, 2004, pp. 2-10.

Davenport H., Process Innovation: Reengineering Work Through Information Technology, Harvard Business School Press, Boston, Etats-Unis, 1992.

Dery-Pinna A.M., Fierstone J., « Amusing : Outil d'assemblage et d'adaptation d'IHM », Rencontres Jeunes Chercheurs en Interaction Homme-Machine, Lacanau, France, 2004

Dikjman R. M., Dumas M., Ouyang C., Formal Semantics and Analysis of BPMN Process Models using Petri Nets, Technical Report, Queensland University of Technology, 2007, accessible à https://eprints.qut.edu.au/archive/00007115/.

Eyermann L.J., Smith A.B., Roberts E.F., « What Senior Management Needs to Know about the Value of MDA », 2004, société IMI, accessible à http://www.omg.org/news/whitepapers/OMG-MDA-Final-Ariticle_June-2004v6.pdf .

Favre J.-M., Estublier J., Blay-Fornarino M., L'ingénierie dirigée par les modèles, Hermès-Lavoisier, Paris, France, 2006.

Florins M., *Graceful Degradation: a Method for Designing Multiplatform Graphical User Interfaces*, Ph. D. Thesis, Université Catholique de Louvain, Belgique, 2006.

Forbrig P., Dittmar A., Reichart D., Sinnig D., « From Models to Interactive Systems Tool Support and XIML », Proceedings of the First International Workshop on Making model-based user interface design practical: usable and open methods and tools, Funchal, Portugal, 2004, pp. 1-14.

Frankel D., *Model driven architecture : applying MDA to enterprise computing*, Wiley, 2003.

Furtado E., Furtado J.J.V., Silva W. B., Rodrigues D.W.T., da Silva Taddeo L., limbourg Q., Vanderdonckt J., « An Ontology-Based Method for Universal Design of User Interfaces », Proceedings of Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends, A. Seffah, T. Radhakrishnan & G. Canals (éds.), Lille, France, 2001.

Gartner Inc., *Gartner's 2006 Emerging Technologies Hype Cycle Highlights Key Technology Themes*, accessible à http://www.gartner.com/it/page.jsp?id=495475 .

Guarino N., Welty C., « Conceptual Modeling and Ontological Analysis », *Tutorial présenté à AAAI-2000*, Austin-Texas, USA, 31 juillet 2000.

Jardim Nunes D.N., Object Modeling for User-Centered Development and User Interface Design : The Wisdom Approach, Ph. D. Thesis, Universidade da Madeira, 2001.

Jardim Nunes D.N., Campos P., « Toward Usable Analysis, Design and Modeling Tools », Proceedings of the First International Workshop on Making model-based user interface design practical: usable and open methods and tools, ISSN 1613-0073. Vol. 103. online http://CEUR-WS.org/Vol-103 , 2004.

Johansson H., McHugh P., Pendlebury A., Wheller W., Business Process Reengineering: Breakpoint Strategies for Market Dominance, Wiley, Chichester, Angleterre, 1993.

Microsoft, *Enterprise Library 3.0*, 2007, accessible à http://msdn2.microsoft.com/en-us/library/aa480453.aspx .

Muller P.-A., Studer P., Fondement F., Bezivin J., « Platform independent Web application modeling and development with Netsilon », *Software & System Modeling*, Vol. 4, n° 4, 2005, pp. 424-442.

Myers B., Hudson S.E., Pausch R., « Past, Present and Future of User Interface Software Tools », *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7, n° 1, 2000, pp. 3-28.

OMG, *Model Driven Architecture*, Model Driven Architecture (MDA), Document number ormsc/2001-07-01. Technical report. 2001

OMG, *Semantic of Business Vocabulary and Business Rules (SBVR)*, 2005, accessible à http://www.omg.org/technology/documents/bms_spec_catalog.htm .

OMG, MDA Guide v1.0.1, 2003, accessible à http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Oya M., MDA and System Design, *MDA Information Day,* OMG Technical Meeting, 2002, accessible à http://www.omg.org/mda/mda_files/E_MDADay_Oya.pdf .

Palano R., Pandurino A., Guido A.L., « Conceptual design of web application families : the BWW approach », *6th Workshop on Domain Specific Modeling*, Portland, USA, 2006, pp. 23-32.

Palanque P., Paterno F., *Formal Methods in Human-Computer Interaction*, ouvrage collectif, Springer-Verlag Telos, 1998

Paterno F., *Model-Based Design and Evaluation of Interactive Applications*, Springer-Verlag, Londres, Angleterre, 1999.

Schattkowsky T., Lohmann M. « Towards Employing UML Model Mappings for Platform Independent User Interface Design », Satellite Events at the MoDELS 2005 Conference: MoDELS 2005 International Workshops OCLWS, MoDeVA, MARTES, AOM, MTiP, WiSME, MODAUI, NfC, MDD, WUsCAM, Montego Bay, Jamaica, Volume 3844, 2006, pp. 201 - 209

Schewe K.-D., Thalheim B., « Conceptual modelling of web information systems », *Data & Knowledge Engineering*, vol. 54, n° 2, 2005, pp. 147-188.

Schlee M., Vanderdonckt J., « Generative Programming of Graphical User Interfaces », Proceeding of the working conference on Advanced visual interfaces, AVI 2004, Gallipoli, Italie, 2004, pp. 403-406.

Selic B., « The Pragmatics of Model-Driven Development », *IEEE Software*, vol. 20, n° 5, 2003, pp. 19-25.

Smith H., « BPM and MDA : Competitors, Alternatives or Complementary», *Business Process Trend Whitepaper*, 2003, accessible à http://www.bptrends.com/publicationfiles/07-03%20WP%20BPM%20and%20MDA%20Reply%20-%20Smith.pdf .

Sottet J.-S., Calvary G., Favre J.-M., Coutaz J., « IHM & IDM : Un tandem prometteur », Poster, Ergo'IA 2006, Biarritz, France, 2006.

Szekely P., « Retrospective and Challenges for Model-Based Interface Development », Proceedings Design, Specification and Verification of Interactive Systems '96, DSV-IS 96, Vienne, Autriche, 1996, pp. 1-21.

Tariq N.A., Akhter N. « *Comparison of Model Driven Architecture (MDA) based tools* », Proceedings of the 13th Nordic Baltic Conference on Biomedical Engineering and Medical Physics, IFMBE, Umea, Suède, 2004, pp. 43-44.

Vanderdonckt J., « A MDA-Compliant Environment for Developing User Interfaces of Information Systems », Proceedings of the 17th Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal, 2005, pp. 16-31.

W3C, *OWL Web Ontology Language Overview*, 2004, accessible à http://www.w3.org/TR/owl-features .

Watson A., *OMG's new modeling specifications***,** Présentation faite à ECMDA-FA, Nuremberg, Allemagne, 2005.

Wilson S., Johnson P., Kelly C., Cunningham J., Markopoulos P., « Beyond Hacking: a Model Based Approach to User Interface Design », Proceedings of HCI'93, Loughborough, Grande Bretagne1993, pp. 217-231.