Architecture des IHM

Nicolas Ferry*, Jean-Bernard Crampes**
Salah Sadou*

*Valoria Lab
salah.sadou@univ-ubs.fr
nicolas.ferry@univ-ubs.fr
http://www-valoria.univ-ubs.fr/Salah.Sadou/
**IRIT Toulouse
jbcrampes@gmail.com
http://www.jbcc.fr

Résumé. Pour les sociétés de services en informatique (SSII), les phases de capture des exigences et de conception utilisent couramment l'ingénierie des modèles pour avoir une représentation de l'architecture de la solution. Malheureusement, l'architecture des interfaces homme-machine (IHM) est très peu étudiée dans ces phases et bien souvent l'architecture des interfaces est laissée à la responsabilité des développeurs qui n'ont généralement pas eu de réflexions avec le client et les utilisateurs finaux. Nous proposons un langage de description de l'architecture des IHM appelé SNI (Schéma Navigationnel des Interfaces) qui cherche à faciliter le dialogue avec les clients et définir l'architecture globale des interfaces. Dans ce cadre, nous présentons un outil qui permet de réaliser ce type de diagrammes et qui permet in fine de générer des maquettes afin d'obtenir un retour des utilisateurs finaux.

1 Introduction

Le nouveau standard de qualité industriel CMMi (Capability Maturity Model Integration) définit cinq niveaux d'industrialisation pour le processus de conception logiciels des sociétés de services en informatique (SSII). Durant toute la réalisation, une entente mutuelle basée sur des compromis est essentielle pour l'aboutissement des projets. Pour cela, il est recommandé de réaliser des ateliers participatifs afin d'échanger avec le client et les utilisateurs finaux sur la solution pour comprendre les besoins et les exigences.

Il existe plusieurs approches dont notamment la modélisation qui, avec l'ingénierie dirigée par les modèles (IDM), représente les informations dans une vue standardisée simplifiant la communication des intervenants. Les architectes en informatique utilisent régulièrement des diagrammes comme UML pour concevoir différents aspects du logiciel. Cependant, nous avons conduit une série d'interviews (Ferry, 2008) auprès de chefs de projets d'une grande société de services et nous avons remarqué que bien souvent les interfaces homme-machine (IHM) d'une application sont laissées uniquement à la responsabilité des développeurs. Hor-

mis certains projets utilisant des maquettes, dans l'ensemble peu de réflexions sont menées au préalable avec le client et les utilisateurs finaux sur l'architecture de l'IHM. Nous avons remarqué également qu'il n'y a peu ou pas d'acquisition des exigences IHM auprès des utilisateurs finaux. Cette approche introduit le risque de construire des interfaces qui ne sont pas en adéquation avec les attentes du client et respectant peu le critère « d'utilisabilité » définit par Constantine et L. Lockwood, (1999) et Nielsen (2003), générant une perte substantielle de temps et par conséquent d'argent. Nous constatons alors que l'interaction homme-machine n'est pas vraiment considérée dans la phases d'analyse et de conception. Certes divers modèles sont repris pour modéliser des IHM : nous pouvons citer les diagrammes d'activités, d'états-transitions (Wasserman, 1995), d'interaction, les réseaux de Pétri (Palanque et Bastide, 1995) mais ces modèles, non spécifiques à la modélisation des IHM, sont souvent très généraux et proposent des représentations souvent complexes et difficilement utilisables lors d'un dialogue avec un utilisateur.

Pour répondre à cette problématique, nous proposons d'introduire un langage permettant de représenter l'architecture des IHM répondant à deux propriétés soulevées par le problème précédent :

i) Il faut que le langage soit suffisamment expressif pour couvrir tous les types d'interactions. Il doit pour cela faire une abstraction des interactions.

ii) Il faut que le langage soit suffisamment simple pour être compréhensible par la plupart des futurs utilisateurs et/ou de la maîtrise d'ouvrage.

Ce langage a pour but de pouvoir d'une part dialoguer avec les futurs utilisateurs de l'application afin d'aboutir à l'interface qui correspond au mieux à la logique métier et au contexte de l'application ; d'autre part, pour la conception et rétro-conception pour permettre aux architectes de spécifier l'interface utilisateur. A partir de l'architecture de l'IHM faite avec le SNI (Schéma Navigationnel des Interfaces), nous générons le code nécessaire à son implémentation par une suite de transformations de modèles. Ainsi notre approche reste conforme à l'IDM.

Nous soulignons le manque de modélisation concernant la couche présentation. Afin de combler ce manque, nous présenterons le formalisme utilisé par le langage conceptuel SNI. Nous détaillons ensuite le méta-modèle qui a été développé pour supporter ce type de représentation. Nous verrons enfin un générateur de maquettes en JSF (Java Server Faces) à partir d'un modèle SNI.

2 Créer l'architecture des interfaces avec SNI

Nous rappelons l'intérêt de découper la couche présentation en trois niveaux d'abstraction en constituant un niveau conceptuel, un niveau logique et un niveau physique comme décrit par Coutaz avec PAC, ou Sedogbo et al.(2006) par exemple. L'intérêt est de pouvoir modéliser des IHM indépendamment de la technologie et de l'aspect final. Cette méthode introduit de la souplesse au niveau de l'IHM puisque celle-ci peut s'adapter à des évolutions technologiques et s'implanter sur un type de matériel différent.

L'un des objectifs du SNI est de permettre de décrire l'IHM en faisant abstraction de toutes contingences matériels ou techniques. Les DTI (Schémas de Défintion Techniques de

l'IHM) décrivent la conception du niveau logique, il existe un modèle par technologies (client lourd, client léger, client multimodal,...) et/ou par plate-formes d'implémentation. A ce niveau, il faut préciser pour une technologie donnée comment seront architecturées et représentées les IHM (au niveau de leur aspect). Enfin, le niveau physique supporte le programme comme il est classiquement développé par l'équipe informatique. Ces diagrammes sont donc un raffinement de modèles successifs et repose sur un raffinement de leur méta-modèle comme introduit dans Tarby et al. (2006) sur les Processus de Modélisation Incrémentaux (PMI).

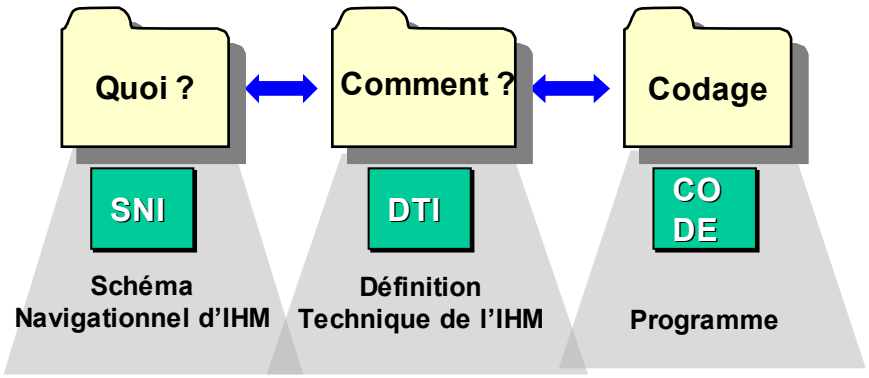

FIG 1. – *Architecture de la couche présentation décomposée en trois niveaux d'abstraction.*

2.1 Le formalisme du SNI

Le SNI est défini comme un profil UML dérivé du diagramme d'activité UML2.0 (Anas Abouzahra, Jean Bézivin et al, 2005). C'est un modèle de haut niveau d'abstraction permettant de représenter uniquement les exigences fonctionnelles, le "quoi" de l'analyse-conception. De ce fait, il est complètement indépendant de la plateforme physique et en particulier du type d'IHM envisagé pour la réalisation (Windows, Web, Mobile, Multimodal ou autres). De même, il ne représente ni les moyens d'interaction (menu déroulant, bouton poussoir, glisser-déposer…), ni les aspects matériels mis en œuvre (clavier, type d'écran, souris...) qui sont du ressort du niveau réalisation. Le premier objectif étant la construction d'une structure d'interaction respectant la logique métier de l'application. Par ailleurs, il ne prend pas en compte l'exécution des traitements (calcul, accès aux données…) car c'est un diagramme centrée sur l'IHM uniquement. Les modèles UML décrivent convenablement ces autres aspects.

Le SNI s'appuie sur le concept d'Unité de Dialogue (UD) introduit par (Crampes, 2000) qui représente une vue élémentaire de l'interaction homme-machine. Il existe deux types d'unités de dialogue, les élémentaires et les composées.

2.1.1 Les UD élémentaires (UDE)

Une UD est dite élémentaire si elle correspond à une opération d'IHM indécomposable. La figure 2 décrit une classification des UDE en six catégories. L'expérience acquise dans de nombreux projets concrets a montré que ces six catégories étaient nécessaires et suffisantes

pour modéliser conceptuellement tout type d'IHM quelle que soit sa nature et sa complexité. Dans ses travaux, paulo da Silva (2002) dans UMLi est arrivé à la même conclusion en proposant une catégorisation sensiblement identique.

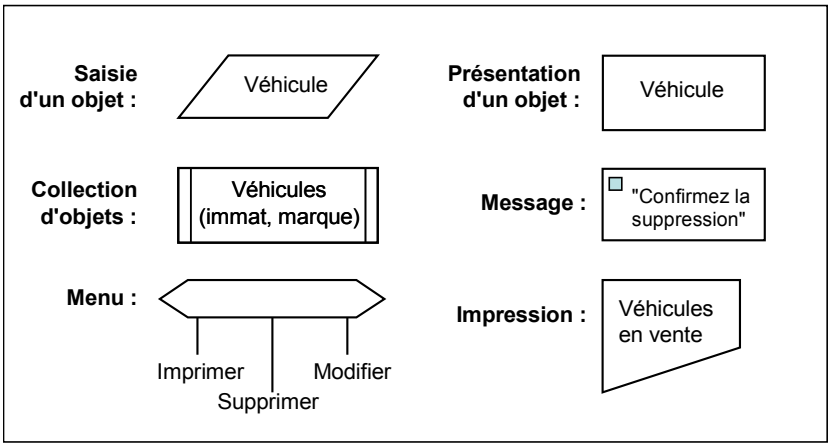

FIG 2. – *Classification des UD élémentaires. Chaque UDE est présentée avec un exemple.*

2.1.2 Les UD composées (UDC)

Plusieurs UD élémentaires peuvent être regroupées pour constituer une UD composée. Une UD composée permet d'une part de regrouper plusieurs informations de nature proche pour optimiser le travail de l'utilisateur, d'autre part de paralléliser plusieurs interactions conformément à la façon de faire de l'utilisateur dans l'exécution de ses tâches.

Dans les diagrammes SNI, la composition de plusieurs UD peut être réalisée simplement par juxtaposition de deux UDE (fig. 3) ou en utilisant une boîte de groupage (fig. 3). Ces deux représentations sont sémantiquement proches. L'utilisation de l'une ou de l'autre dépend essentiellement de l'architecte IHM. Notamment, s'il souhaite ajouter des éléments pour enrichir (juxtaposition) ou détailler un élément existant en précisant son contenu (boite de groupe).

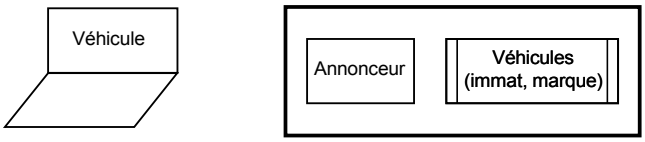

FIG. 3 – *Exemple de composition par juxtaposition (à gauche) et d'une boîte de groupage (à droite). La juxtaposition d'une unité d'affichage et d'une unité de saisie va former une UDC de modification. La boîte de groupage détaille une unité d'affichage élémentaire pour décrire que l'on affiche simultanément un annonceur et la liste des voitures mises en vente.*

2.2 Exemple de SNI

Pour illustrer nos propos, prenons l'exemple d'une classe "Véhicule" enregistrée dans une base de données de ventes de véhicules d'occasion. L'administration de cette classe doit permettre à une personne autorisée : l'administrateur, d'afficher tous les véhicules proposés à la vente, de modifier ou de supprimer un véhicule donné et d'ajouter des véhicules. Il pourra également imprimer la liste des véhicules mis en vente. Toutes ces opérations sont représentées par le SNI de la figure 4.

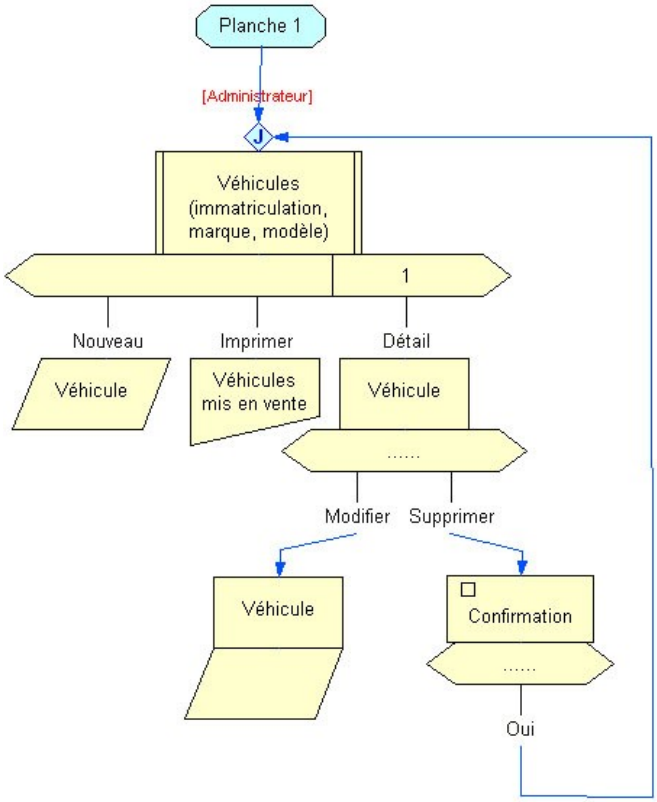

FIG. 4 – *SNI d'administration (ajouter, modifier, supprimer) de la classe véhicule. (Diagramme réalisé avec l'outil VisualSNI de l'AGL MACAO).*

La structure de la figure 4 est très utilisée dans les applications de gestion, c'est un patron d'IHM qui permet de créer une structure CRUD c'est-à-dire sur une collection d'élément, l'utilisateur pourra naviguer vers une unité de création d'un nouvel élément, modifier un élément sélectionné ou supprimer un élément particuliers.

2.3 Compléments de modélisation

Un certain nombre de concepts supplémentaires ont été définis dans le modèle SNI de la figure 4 pour d'une part permettre la prise en compte d'exigences particulières des utilisa-

teurs du logiciel concernant l'IHM, d'autre part d'intégrer une certaine flexibilité dans la structuration des diagrammes.

Les compléments liés aux exigences des utilisateurs : l'utilisation de fonctions "FILTRE" et "TRI" lors de l'affichage d'une collection d'objets pour permettre de prendre en compte des exigences clientes (Exemple : dans une collection ne pouvoir voir que les véhicules d'une marque ou d'un modèle donnée) ; la notion de "rôle" indiquant les autorisations ou les interdictions de parcours de certaines branches du SNI suivant le type d'utilisateur et la notion de condition ou "garde" permettant de parcourir certaines branches en fonctions de l'état de certains objets.

Les compléments liés à la flexibilité d'écriture des diagrammes concernent : l'invocation d'un sous-SNI qui peut être appelé à plusieurs niveaux ; la décision et la jonction qui permettent de parcourir différentes branches sous certaines conditions indépendantes des choix utilisateur ; le découpage en planches permettant de construire un SNI complexe tout en gardant une bonne lisibilité ; et les étiquettes permettant de se brancher sur une UD ou une planche particulière.

3 Le méta-modèle du SNI

Le méta-modèle du SNI se présente sous forme de trois paquetages SNI_UD, SNI_Nœud, SNI_Port. Pour simplifier sa présentation, nous n'avons pas fait apparaître ici les compléments de modélisation.

3.1 Le paquetage SNI_UD

Les éléments de base du SNI sont des UDE et des UDC. Les UDE représentent les actions du diagramme d'activité. La figure 5 montre qu'une UDC est composée de plusieurs UD. Une UDC peut être spécialisée par juxtaposition (UDC_J) ou par groupage (UDC_G). Une UDE peut se spécialiser en *Menu*, *Saisie* ou *Présentation*. La présentation peut elle-même se spécialiser en *Message*, *Collection* et *Objet*. La méta-classe « Menu » représente un ensemble d'options proposées de façon simultanée à l'utilisateur. Un menu est composé de groupes d'options actives sous la même condition. Par exemple, dans la figure 4, les options *Nouveau* et *Imprimer* représentent un premier groupe d'options activables dans tous les cas. Par contre, le groupe composé de la seule option *Détail* n'a d'existence que si un véhicule a été sélectionné dans la liste.

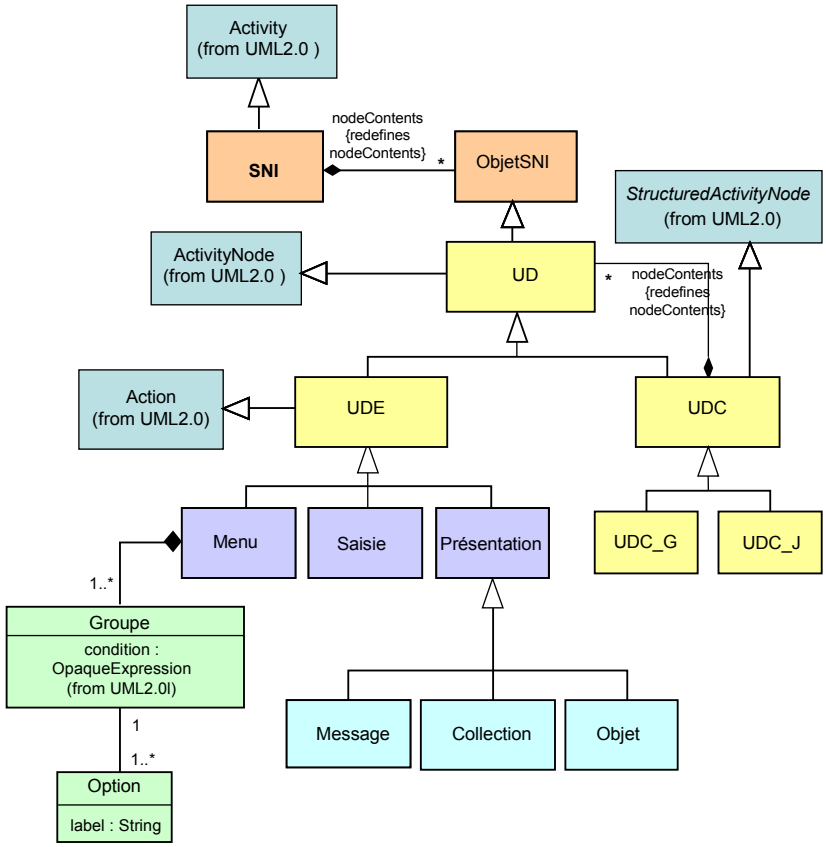

FIG. 5 – *Ce paquetage présente la structuration des différentes unités de dialogues composées et élémentaires du modèle SNI. La méta-classe UDE est directement dérivée de la méta-classe UML2.0 "Action" . Il en est de même pour UDC qui dérive de "Structured-ActivityNode.*

L'arborescence des unités de dialogues est gérée par un patron composite avec des UDE et des UDC. Chaque objet du SNI (ObjetSNI) contient plusieurs descriptions de questions : Qui doit réaliser cette unité ? Que représente cette unité et la problématique à résoudre ? Quoi (Quel est son type) ? Quand cette unité a t'elle été créée ? Comment est résolu le problème ?

3.2 Le paquetage SNI_Port

Un port est un objet abstrait attaché à une UD ou à un nœud permettant d'établir les connexions. Chaque objet (*ObjectSNI)* peut comporter un ou plusieurs ports d'entrée (*IN*) et un ou plusieurs ports de sortie (*OUT*). Par exemple, un menu comporte un port d'entrée et N ports de sortie qui correspondent aux choix possibles par l'utilisateur. Un port d'entrée peut être connecté à un port de sortie par un lien.

A un port d'entrée, il est possible d'associer une garde qui joue un rôle de condition transitoire en interdisant l'entrée dans un objet. De la même manière, à un port de sortie, on peut associer un garde qui conditionne la transition en sortie tant qu'elle n'est pas vérifiée.

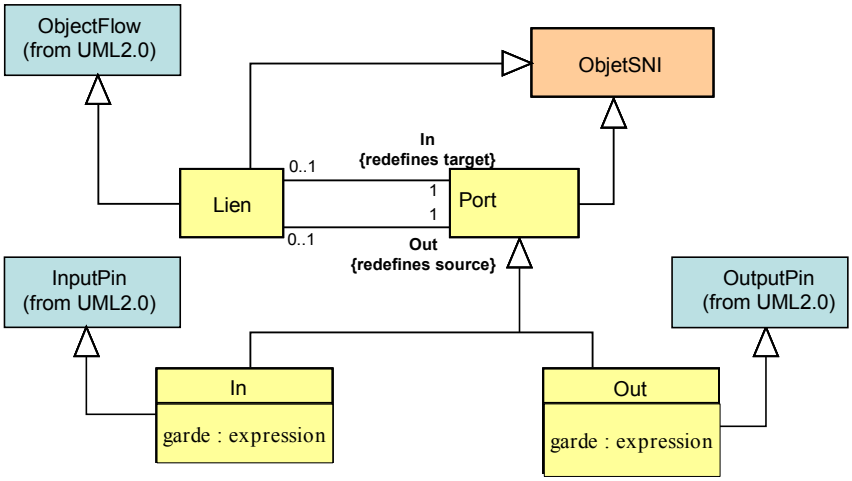

FIG. 6 – *Le paquetage SNI_Port montre qu'il existe deux types de ports, les ports d'entrée (In) et les ports de sortie (Out). Un port d'entrée peut être connecté à un port de sortie par un lien qui dérive de la méta-classe UML "ObjectFlow". Des gardes sont associés aux ports formant des conditions transitoires.*

Dans le SNI c'est une pile d'appels qui stocke l'ordre dans lequel les unités sont enchaînées. En pratique, c'est la plate-forme cible qui à la charge de gérer la dynamique des enchainements réels des contrôles graphiques implémentés.

3.3 Le paquetage SNI_Nodes

Le paquetage *Nodes* détermine toutes les unités de routage dans le diagramme. Le routage regroupe les objets du diagrammes qui permettent de naviguer d'une unité de dialogue à une autre (c-a-d les jonctions, les décisions ou les invocations).

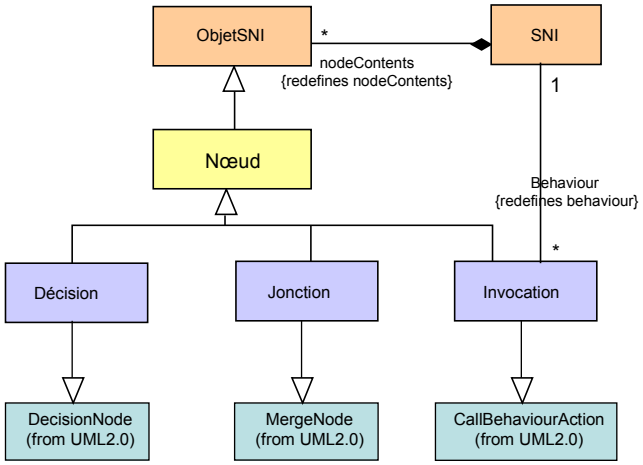

FIG. 7 – *Le paquetage SNI_Nodes structure les objets de connexions. Il existe trois types de nœuds dans le SNI : la Décision, la Jonction et l'Invocation qui dérivent directement des métaclasses "DecisionNode", "MergeNode" et "CallBehaviourAction" du diagramme d'activité UML 2.0.*

Le mécanisme d'invocation permet deux choses. D'une part, il permet de gagner en lisibilité sur des applications de taille standard et importante puisqu'il est ainsi possible de répartir les unités de dialogues sur plusieurs planches. Et d'autres part, il permet de factoriser des groupes d'unités afin de mieux contrôler les appels qui proviennent de multiples sources.

4 L'éditeur graphique VisualSNI

Le méta-modèle du SNI, a permis la réalisation d'un éditeur graphique de SNI que nous avons développé sous la forme d'un plugin Eclipse sous le nom de VisualSNI.

Les trois paquetages du méta-modèle ont été fusionnés afin de pouvoir être repris par EMF (Eclipse Modeling Framework) pour générer l'ensemble des classes Java nécessaires d'une part à la construction graphique d'un SNI, d'autre part à la gestion de la persistance du modèle sous forme de fichiers XMI (XML Model Interchange).

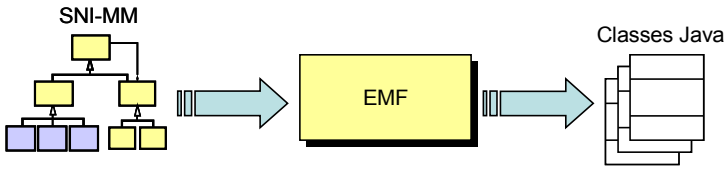

FIG. 8 – *Génération des classes Java à partir du méta-modèle du SNI (SNI-MM).*

L'éditeur graphique a été construit en utilisant le framework GEF (Graphic Editor Framework) selon une architecture de type MVC qui permet de découpler l'édition graphique, réalisée par le plugin Draw2D, de la gestion des modèles réalisée par EMF.

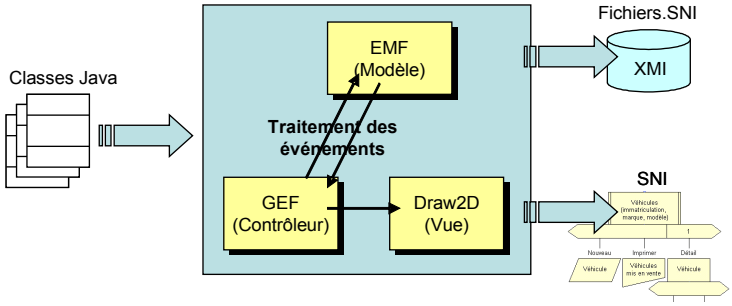

FIG. 9 – *Architecture MVC réalisée par les plugins EMF, GEF et Draw2D.*

Comme le montre la figure 9, l'éditeur VisualSNI est un plugin Eclipse permettant, dans le cadre d'un projet Java, de générer des planches de SNI graphiquement dans une zone d'édition et sous forme de fichiers au format XMI.

Ci-dessous la figure 11 montre la structure du fichier XMI générée lors de la sauvegarde du SNI présenté à la figure 10. On peut y voir la description des deux premiers objets : "Debut" et "AffListe".

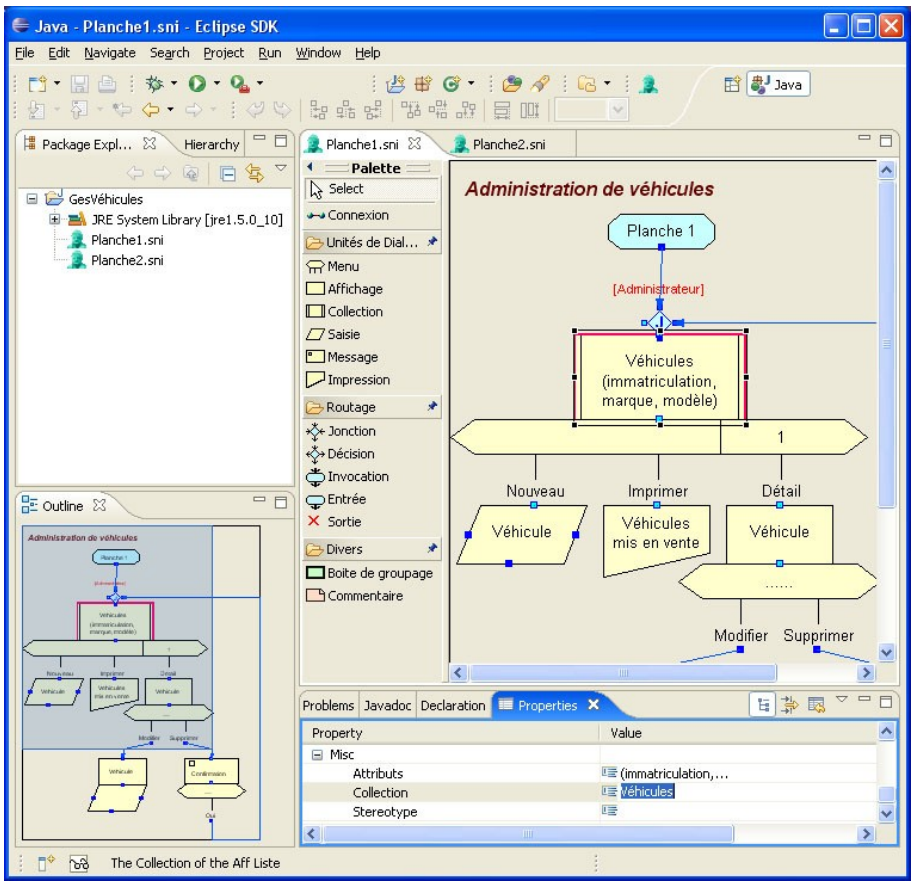

FIG. 10 – *L'éditeur VisualSNI est un plugin Eclipse manipulant les diagrammes SNI.*

```
<?xml version="1.0" encoding="ASCII"?>
<sni:SNI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
                    xmlns:sni="http://sni" backColor="0"
                    titre="Administration de véhicules">
  <lstObjets xsi:type="sni:Debut" X="130.0" Y="45.0" width="90.0"
                    height="30.0" label="Planche 1">
     <outputPorts name="out1" X="0.5" Y="0.96666664"
                    linkPort="//@lstObjets.21/@inputPorts.0"/>
  </lstObjets>
  <lstObjets xsi:type="sni:AffListe" X="102.0" Y="146.0" width="143.0"
                    height="73.0" dispListe="Véhicules"
                    dispAttrib="(immatriculation, marque, modèle)">
     <inputPorts name="in1" X="0.5"/>
     <outputPorts name="out1" X="0.5" Y="0.98630136"
                    linkPort="//@lstObjets.3/@inputPorts.0"/>
  </lstObjets>
...
```

FIG. 11 – *Exemple de structure d'un SNI stockée sous forme de fichier XMI.*

Dans la figure 10, L'éditeur VisualSNI est composé d'une zone de gestion des projets, d'une barre d'outils en icônes, d'une zone d'édition graphique du SNI, d'une vue globale pour la navigation rapide, d'une palette d'outils contenant les éléments de syntaxe et d'une zone d'édition des propriétés des objets.

La figure 11 montre l'instance du modèle qui est présenté figure 10, enregistré sous forme de fichier XMI.

5 Transformation SNI vers JSF

L'objectif de cette transformation est de générer automatiquement une maquette de site WEB dans le framework JSF de Sun. L'intérêt est de pouvoir montrer rapidement à l'utilisateur quel est le résultat final obtenu afin d'apporter immédiatement les modifications éventuelles au SNI et converger ainsi plus rapidement vers une solution satisfaisante en termes de couverture fonctionnelle, d'enchaînement des fonctions, et de droits d'accès.

A partir d'un SNI, le générateur SNI2JSF est capable de construire une maquette utilisateur avec les pages WEB et leur enchaînement. En fait, il gère la partie visuelle de l'application. Dans l'approche MVC ceci correspond à décrire la partie "View". Dans une application 3-tiers, cela correspond à la couche présentation.

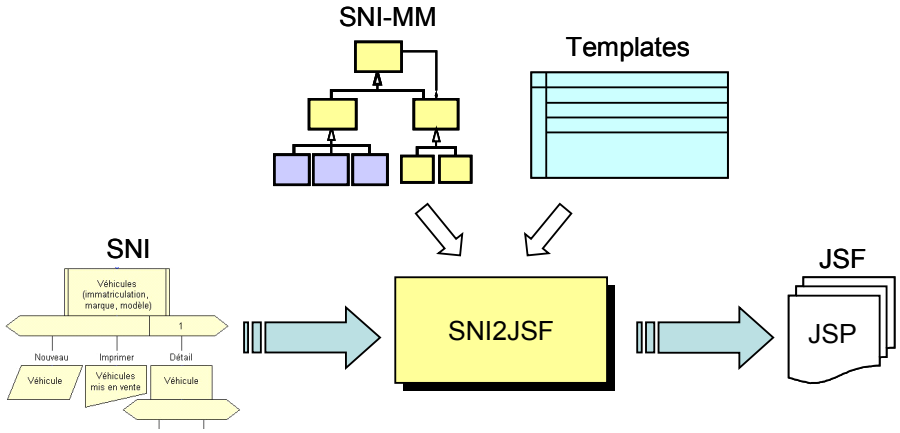

FIG. 12 – *Génération de pages JSP à partir d'un SNI, de son méta-modèle (SNI-MM) et d'un fichier Templates (modèle de pages et règles de transformation).*

Le générateur SNI2JSF transforme un SNI en un ensemble structuré de pages JSP du framework JSF en utilisant le méta-modèle du SNI (SNI-MM) et un *Template* décrivant la forme générale des pages JSP à générer. Les templates sont des patrons de génération écrits dans le langage Xpand2 (Efftinge et Kadura 2006) qui intègrent le savoir-faire des concepteurs-développeurs en matière de présentation finale des pages. En changeant de fichier *Template* il est possible de générer des pages avec des formats différents suivant les besoins de l'utilisateur.

La méthode de transformation s'appuie sur le framework oAW (openArchitectureWare) du projet Eclipse GMT (Generative Modeling Technologies).

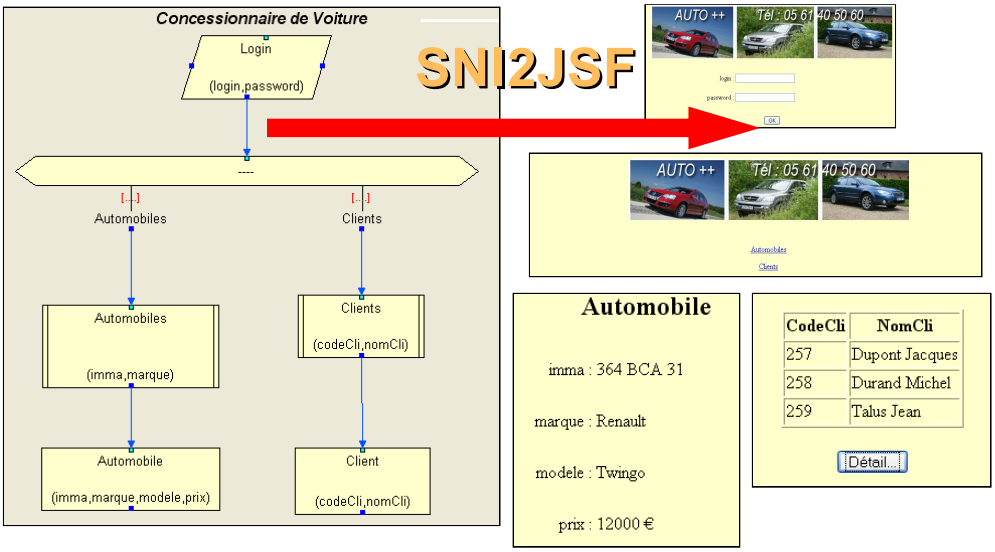

FIG. 13 – *Le générateur SNI2JSF transforme un modèle SNI (à gauche) en une maquette d'application Web fonctionnelle composée d'un ensemble de pages basée sur le modèle..*

Une maquette Web est entièrement générée, ce qui permet de donner une première vision aux utilisateurs afin d'obtenir un retour sur des questionnements généraux et sur l'adéquation avec leurs attentes. Une modification suivant l'approche des méthodes RAD (Rapid Application Development) peut ensuite être réalisée pour prendre notes de modifications précises sur les écrans.

6 Travaux connexes

Nos travaux se placent dans la continuité de l'utilisation de l'ingénierie dirigée par les modèles et des transformées de modèles appliquées aux interfaces hommes-machines. Le diagramme SNI réunit plusieurs objectifs dont les principaux sont :

- de capturer les exigences d'IHM de niveaux conceptuels.
- de formuler la compréhension de l'architecture de l'IHM du projet, ceci afin d'établir un dialogue constructif avec le client et d'obtenir son retour sur l'adéquation de la solution.
- de décrire la structure de l'IHM en terme de navigation, droits d'accès, sélection des informations à afficher et des intéractions d'enchaînements.
- d'utiliser la théorie des modèles pour donner une vision concrète en phase amont sans avoir à produire toute la solution.
- De générer une maquette à l'utilisateur pour obtenir ses retours.

Pour cette raison, nous nous ne considérons comme travaux annexes les seuls travaux dont l'objectif est la capture de la connaissance de l'utilisateur et ceux qui permettent la génération des interfaces.

Des travaux similaires sont entrepris dans ce domaine ou proche. Nous pouvons citer Coutaz et Calvary Gaëlle (2008) dans l'axe de recherche sur la plasticité des IHM. Ce sujet vise l'étude de nouvelles IHM composables dynamiquement par l'utilisateur final et capables de s'adapter à des variations du contexte d'usage qui réunit trois types de modèles : utilisateur, plate-forme, environnement.
La plasticité des IHM représente la capacité adaptative d'une IHM. L'IHM varie suivant le contexte d'usages de l'utilisateur. Les contextes sont définis par des ontologies sous forme de graphes orientés de contextes. La structure de gestion des contextes se compose en trois niveaux. Un graphe de contextes, un graphe par contexte pour décrire les situations, un graphe décrivant chaque situations avec les entités et les rôles joués qui relient ces entités. Un système plastique a une capacité d'adaptation au contexte d'usage. L'IHM peut évoluer par opération de remodelage, de distribution, ou de migration. Le modèle SNI est ouvert a pouvoir s'adapter aux problèmes de plasticité soit en reprenant ce concept soit en étant utilisé dans les graphes de plasticité.

Dubois (2005) traite de la réalité-mixte ou réalité augmentée. Leurs travaux traite respectivement d'utilisateurs en réalité modifiée. Les utilisateurs portant des casques virtuels ou des éléments spécifiques évoluent dans un système en réalité mixte ou augmentée. Le modèle SNI n'a pas vocation a traiter des IHM de la réalité-mixte ou augmentée.

D'autres modèles d'IHM existent comme CTT (Paternò F. et al, 2008) qui est un diagramme de tâches. Il reprend la sémantique du langage LOTOS (Language Of Temporal Ordering Specification) et applique les opérateurs d'ordonnancement temporel sur des tâches. Les tâches sont soit des tâches abstraites, interactives, ou informatisées et sont organisées en un découpage arborescent. Ce modèle n'entre pas en concurrence avec le SNI qui est un modèle conceptuel d'IHM car ce dernier ne modélise pas les tâches métiers du processus client.

Le langage usiXML (López-Jaquero et al., 2008) a le plus de ressemblance avec SNI. C'est un langage XML qui embarque la possibilité de définir plusieurs niveaux de description de l'IHM (abstraites jusqu'à concrètes). Le modèle SNI s'en distingue néanmoins sur plusieurs points tel que :

- l'évolution des interconnexion de l'IHM avec les couches métiers et données.
- la génération des interfaces qui est réalisé à base de patrons

Les patrons de transformation représentant une véritable base de l'expérience et du savoir-faire de l'entreprise.

7 Conclusion

La plupart des logiciels actuels s'articulent autour d'une IHM complexe (graphique, multimodale, multimédia…) s'enchaînant suivant une logique choisie par l'utilisateur intégrant de nombreux objets de dialogue et des résultats pouvant revêtir une grande variété de formes

(fenêtres Windows, pages WEB, outils de visualisation ou de saisie spécialisés…). La réalisation de telles IHM est parsemée de pièges qui sont générateurs de risques importants pour la rentabilité du projet en cours. Les pièges sont généralement de deux natures :

1) l'IHM est la seule partie du logiciel visible par l'utilisateur et c'est donc sur elle que se focalisera son attention,
2) un logiciel est rarement utilisé par une seule personne (multi-points de vue), aussi faudra-t-il obtenir un *modus vivendi* d'autant plus difficile à atteindre que l'aspect "artistique" d'une IHM joue souvent un rôle important. Le choix des couleurs, des polices, des icônes, l'incrustation d'images et d'animations, la disposition des objets, sont autant de motifs de mésentente et donc de dérapage du projet.

Afin de limiter ces inconvénients, il est essentiel d'établir un dialogue constructif avec les utilisateurs. Ceci nécessite l'utilisation d'un langage commun de communication permettant non seulement d'acquérir et de décrire leurs exigences, mais encore de leur rendre compte (feed-back) de la compréhension des choses et des solutions proposées. Nous avons réalisé un éditeur graphique VisualSNI comme un module d'Eclipse afin de gérer les diagrammes SNI de manière informatisé. Au préalable les diagrammes étaient créés à la main ou par des outils graphiques. Au niveau de l'architecture logicielle, la seule contrainte est l'indépendance de l'interface vis-à-vis du cœur de l'application.

On peut se demander si le SNI est vraiment adapté aux besoins initiaux. Des expériences en entreprises ont montré que cela dépendait du type de client. Par exemple, nous avons eu un client qui refusait de voir un modèle graphique qu'il soit UML ou non. La majorité cependant ont apprécié et ont reconnu l'intérêt du modèle et de l'outil. Ce refus pour certains clients provient d'une trop grande abstraction.

C'est pour cette raison que la réalisation d'un générateur de maquettes JSF a été entrepris. Il transforme un diagramme SNI en une maquette dans une technologie cible grâce au mécanisme de transformation de modèles de l'IDM. Le générateur permet ainsi de présenter rapidement un premier rendu pour l'interface. C'est aussi un moyen de capitaliser l'expérience de l'entreprise sous forme de patron de génération. Nous pensons ainsi que l'entreprise peut d'une part capitaliser son savoir-faire et d'autre part enrichir son catalogue de « patron » pour proposer différentes approches aux clients suivants.

Afin d'améliorer la phase d'acquisition des exigences, nous explorons la possibilité de transformer des diagrammes de tâches du type BPMN vers le SNI pour l'IHM et vers UML pour les autres couches. Cette approche propose d'adapter la modélisation des interfaces homme-machine à l'architecture des Services Web (Services Oriented Application). Le principe repose sur une définition d'un modèle du processus du métier client en utilisant le standard BPMN de l'OMG. L'ingénierie des modèles nous permettrait de séparer les composantes de l'architecture du processus métier vers une composante IHM avec le SNI notamment. Ce sujet fera partie d'un article futur.

Références

Abouzahra anas, Jean Bézivin, Marcos Didonet Del Fabro, Frédéric Jouault. (2005) *A Practical Approach to Bridging Domain Specific Languages with UML profiles* Proceedings of the Best Practices for Model Driven Software Development at OOPSLA'05, San Diego, California, USA, 2005.

Aniszczyk, Chris, (2005) *Using GEF with EMF*, article IBM, June 8, 2005.

Crampes, (2003) *Méthode orientée-objet intégrale MACAO*. Edition Ellipses. ISBN: 2729814248, 2003.

Constantine, L. Lockwood, (1999) *Software for Use : a practical guide to the models and methods of usage centered design*. Addison-Wesley.

Coutaz Joëlle, Calvary Gaëlle, (2008) *HCI and Software Engineering: Designing for User Interface Plasticity.* In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications. Pages 1107-1125. 2008. Second Edition, ISBN 9780805858709, Taylor & Francis CRC Press, Human Factor and Ergonomics series, A. Sears, J. Jacko Eds. http://www.isrc.umbc.edu/HCIHandbook/.

Dupuy-Chessa Sophie, Dubois Emmanuel (2005), *Requierments and Impacts of Model Driven Engineering on Mixed Systems Design* article aux premières journées à IDM 2005.

Efftinge S., Kadura C. (2006) *OpenArchitectureWare 4.1 – Xpand Language Reference,* Electronic document, r20_XpandReference.pdf, August 2006.

Ferry Nicolas (2008), *Formalisation des modèles de la méthode MACAO et réalisation d'un outil de génie logiciel pour génération de maquettes*, PhD's thesis, Department of Computer Science, University of Toulouse, France.

López-Jaquero, V., Montero, F., Molina, J.P. , and Vanderdonckt, J., (2008), *Computer-Aided Design of User Interfaces VI*, Proc. of 7th Int. Conf. of Computer-Aided Design of User Interfaces CADUI'2008 (Albacete, 15-17 September 2008), Information Systems Series, Springer-Verlag, Berlin, 2008, to appear.

Moore Bill, Dean David, Gerber Anna, Wagenknecht Gunnar, Vanderheyden Philippe, *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*, redbook, February 2004, International Technical Support Organization, http://www.redbooks.ibm.com/abstracts/sg246302.html.

Nielsen Jakob (2003) - *Usability Engineering* Morgan Kaufmann Publishers, ISBN 1-55860-561-4, 2003.

Palanque Philippe, Bastide Rémi (1995) *Spécifications formelles pour l'ingénierie des interfaces homme-machine* Technique et Science Informatique 14 n°4.

Paternò F., Santoro C., Mäntyjärvi J., Mori G., Sansone S. (2008) *Authoring Pervasive MultiModal User Interfaces.* International Journal of Web Engineering and Technology, Inderscience Publishers, pp.235-261, 2008.

Architecture des IHM

Paulo Pinheiro da Silva, Norman W. Paton. (2000) *User Interface Modelling with UML*. Information Modelling and Knowledge Bases XII, 10th European-Japanese Conference on Information Modellind and Knowledge Representation, Saariselka, Finland, May 2000.

Paulo Pinheiro da Silva (2002), *Object Modelling of Interactive Systems: The UMLi Approach*. PhD's thesis, Department of Computer Science, University of Manchester, United Kingdom, 2002.

Sedogbo Célestin, Olivier Grisvard, Frédéric Landragin, Jérôme Lard, Sébastien Proud (2006), *HMI Engineering Productivity* - Thales Group, IDM&IHM06 - Canada.

Tarby Jean-Claude, Le Pallec Xavier, Marvie Raphaël, Nebut Mirabelle (2006), *Processus de modélisation incrémental pour le développement d'applications interactives basées sur PAC,* 2006, White-Paper.

Wasserman A.I. (1995) *Extending State/Transition Diagrams for the specification on Human-Computer Interactions* IEEE Transaction on Software Engineering - Vol 11, n°8, August 1995.

Summary

For computer services societies, the requirement and architecture phases use widely the MDE (Model Development Engineering) approach in order to produce the global architecture. Unfortunately, few studies have been made on the HCI architecture in these phases and often the architecture's interface is the developer's responsibility, who generally have been involved with few reflections from the clients and the final users. We propose an HCI language called SNI (Schéma Navigationnel des Interfaces), which defines the global HCI architecture and seeks to facilitate dialogues with clients. Therefore, we present a tool to manipulate this kind of diagrams, and to allow the generation of mocks-up in order to acquire the final users feedback.