Le SNI : un modèle de haut niveau pour la conception et le maquettage des IHM

SNI : High Level Model for IHM Design and Mock-up

Jean-Bernard Crampes

IRIT, UPS Toulouse / IUT de Blagnac 1, Place George Brassens, 31703 Blagnac, France
jbcrampes@gmail.com

Nicolas Ferry

IRIT, UPS Toulouse / Capgemini Sud 8, rue Paul Mesplé - BP 1155, 31036 Toulouse, France
ferry@iut-blagnac.fr

Résumé

Cet article présente le SNI, Schéma Navigationnel d'Interactions, comme un modèle de représentation des interfaces homme-machine permettant d'une part, de concevoir une IHM tenant compte des exigences des utilisateurs, et d'autre part de générer automatiquement, par transformation de modèles, un squelette du logiciel souhaité. En fait, deux niveaux de modèles sont proposés. Le niveau conceptuel (SNI) permet de spécifier la navigation fonctionnelle dans le logiciel indépendamment de toute plateforme technique. Le niveau logique décrit l'enchaînement dynamique et la définition détaillée des composants de l'IHM en fonction d'une plateforme de développement particulière (*GUI*, *WEB*, multimodal …). Cet article présente ces deux niveaux de modèles, leurs méta-modèles respectifs et l'outil logiciel "VisualSNI" permettant de construire le SNI et de générer automatiquement une maquette dans le framework JSF.

Abstract

This paper presents SNI, Interactions Navigational Diagram, as a UI representation model allowing to design a CUI in accordance with users requirements and to generate automatically a skeleton of the wished software, using model transformation. To do so, two model levels are proposed. The conceptual level (SNI) specifies the functional navigation in the software independently of any technical platform. The logical level describes the dynamic chaining and the detailed definition of UI components according to a particular development platform (GUI, WEB, Multimodal…). This paper presents these two model levels, their respective metamodels and the software tool "VisualSNI" allowing to build the SNI and to generate a mock-up in the JSF framework automatically.

Mots-clés

IHM, MACAO, MDA, méta-modèle, eclipse, SNI, SEF, règles de transformation, eTI, revue électronique, technologie de l'information

Keywords

HCI, UI, MACAO, MDA, metamodel, eclipse, SNI, SEF, transformation rules, eTI, electronic journal, information technology

1. Introduction

1.1 Objet

La méthode de génie logiciel MACAO, Méthode d'Analyse et de Conception Orientée-Objet, (Crampes, 2003), (JBCC, 2008) propose une démarche itérative basée sur la conception et le développement de prototypes incrémentaux. Elle préconise une participation active des utilisateurs à chaque itération en positionnant l'Interface Homme-Machine (IHM) comme une composante majeure de la démarche. La prise en compte de cet aspect impose la définition d'un modèle commun utilisateur–concepteur afin de dialoguer de façon constructive pour l'acquisition et la compréhension des exigences.

Actuellement, divers modèles sont utilisés pour la modélisation des IHM. On trouve les diagrammes d'activités, d'états-transitions (Wasserman, 1995), (Horrocks, 1999), d'interaction, les réseaux de Pétri (Palanque et Bastide, 1995). Tous ces modèles, très généraux, proposent des représentations souvent complexes et difficilement utilisables lors d'un dialogue avec un utilisateur. Une tentative d'adaptation d'UML à la modélisation des IHM a été faite par Paulo da Silva avec UMLi (Pinheiro Da Silva et Paton, 2001). Cependant, cette modélisation ne semble pas satisfaisante en matière de lisibilité par l'utilisateur final car elle ne fait que rajouter de nouveaux symboles (triangles, cubes…) aux modèles UML souvent déjà surchargés.

C'est dans ce contexte que nous proposons le modèle SNI (Schéma Navigationnel d'Interactions) qui devient un support de communication supplémentaire pour les intervenants d'un projet et dont les objectifs essentiels sont :

- proposer un langage commun adapté à la communication entre les réalisateurs et les utilisateurs,
- modéliser l'IHM en termes de navigation entre ses différents objets, de droits d'accès et de couverture fonctionnelle du logiciel,
- permettre une génération quasi-automatique du code de l'IHM conformément à l'approche MDA (Miller *et al.*, 2003) (Dupuy-Chessa et Dubois, 2005) (Sedogbo, Grisvard *et al.*, 2006) en utilisant les transformations de modèles. Le passage du SNI au code est réalisé via un modèle intermédiaire adapté à chaque type d'IHM.

1.2 La gestion de l'IHM dans MACAO

Dans la méthode MACAO, les modèles d'IHM sont utilisés lors de l'étape d'analyse globale pour l'acquisition des exigences, lors de la conception globale pour décrire l'architecture de l'IHM et lors du développement, pour réaliser des maquettes et générer le code du logiciel. Ces modèles vont dans le sens d'une amélioration de l'utilisabilité des logiciels comme exposé en (Constantine et Lockwood, 1999), (Crampes et Nonne, 2005) et (Nielsen 2003). Pour cela, MACAO propose deux niveaux de modèles dont l'intérêt est de pouvoir préciser les frontières de l'IHM entre le « Quoi-Fonctionnel » de la phase d'analyse-conception et le « Comment-Technologique » de la phase de réalisation [1]:

[1] Ces différents modèles sont à rapprocher des DSL (*Domaine Specific Language*) qui ont pour objectif de formaliser de façon dédiée chaque aspect d'un domaine à modéliser. (Palanque et Bastide, 1994), (Jouault et Kurtev, 2005) et (Abouzahra, Bezivin *et al.*, 2005).

Niveau	Modèles / Diagrammes
Quoi-Fonctionnel : Analyse-Conception	SNI
Comment-Technologique : Réalisation	SEF, SEP, SEM DEF, DEP, DEM
Comment-Visuel Présentation	Maquettes, Code

Tableau 1. Les niveaux de modèles d'IHM de la méthode MACAO

C'est au niveau de l'analyse que l'on capture les exigences du client et que l'on définit la navigation globale dans l'IHM. Le SNI utilisé en mode "esquisse" (tracé en temps réel au fur et à mesure de la découverte des besoins) est le modèle adapté à ce niveau. Utilisé en mode conception, mettant en œuvre des patrons de conception, il permet de structurer complètement la navigation dans l'IHM.

La réalisation raffine le niveau précédent et permet de prendre en considération des domaines technologiques spécifiques. Ainsi, le Schéma d'Enchaînement des fenêtres (SEF) et le modèle Définition de Fenêtres (DEF) sont-ils adaptés à la représentation d'une IHM de type fenêtré (client lourd ou riche) alors que le Schéma d'Enchaînement des pages Web (SEP) et le modèle Définition des pages (DEP) sont mieux adaptés à la représentation *Web* (client léger). Enfin, le Schéma d'Enchaînement Multimodal (SEM) et le modèle DEfinition Multimodale (DEM) sont utilisés pour décrire des IHM faisant intervenir plusieurs modalités telles que le vocal ou le tactile (type *iPhone* d'Apple). Le passage d'un niveau à l'autre se fait au moyen d'outils et de langages de transformation de modèles pour permettre de générer *in fine* le squelette du code source d'un logiciel interactif.

La suite de cet article est structurée en cinq sections :

- une présentation du formalisme SNI en section 2,
- un descriptif de son méta modèle en section 3,
- une présentation du modèle logique SEF et de son méta-modèle en section 4,
- un examen rapide des règles de transformation du SNI en SEF ou en JSF, en section 5,
- une brève description du module Visual-SNI de l'AGL MACAO en section 6.

2. Le formalisme du SNI

2.1 Objet

Le SNI est défini comme un profil UML dérivé du diagramme d'activité UML 2.0 (Fowler, 2004). 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, *etc.*), ni les aspects matériels mis en œuvre (clavier, type d'écran, souris, *etc.*) qui sont du ressort du niveau réalisation. Par ailleurs, il ne prend pas en compte l'exécution des traitements (calcul, accès aux données, *etc.*).

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

LES UNITES DE DIALOGUE ELEMENTAIRES (UDE)

Une UD est dite élémentaire si elle correspond à une opération d'IHM indécomposable. Le tableau 2 décrit une classification des UDE en cinq catégories telle qu'elle apparaît dans MACAO et illustre chaque UDE avec un exemple.L'expérience acquise dans de nombreux projets concrets a montré que ces cinq catégories étaient nécessaires et suffisantes pour modéliser conceptuellement tout type d'IHM quelle que soit sa nature et sa complexité. Paulo da Silva dans UMLi est d'ailleurs arrivé à la même conclusion en proposant une catégorisation semblable présentée à la 2ème remarque de la section 3.1.

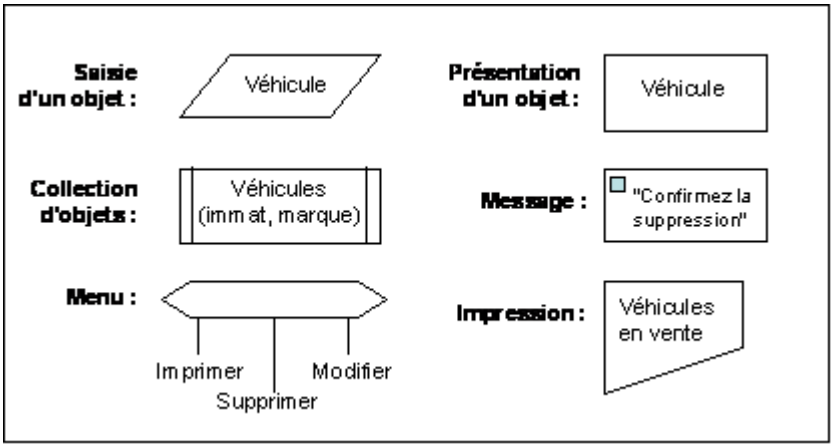

Tableau 2. Classification des UD élémentaires

LES UNITES DE DIALOGUE COMPOSEES (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, et d'autre part, de paralléliser plusieurs interactions conformément à la façon de procéder 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 ou en utilisant une boîte de groupage. Ces deux représentations sont sémantiquement proches (*cf.* section 3.1). L'utilisation de l'une ou de l'autre dépend essentiellement d'une facilité de représentation dans le SNI.

La figure 1 montre la juxtaposition d'un affichage et d'une saisie pour former une UDC de modification.

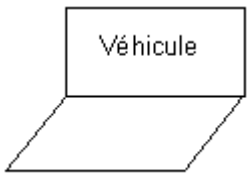

Figure 1. Exemple de composition par juxtaposition

La composition d'une UDC par groupage est illustrée en figure 2 avec l'affichage simultané d'un annonceur et de la liste des voitures mises en vente.

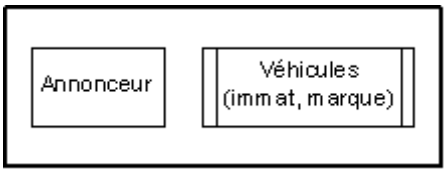

Figure 2. Exemple de composition d'une UDC par groupage

2.2 Exemple de SNI

Pour illustrer notre propos, prenons l'exemple d'une classe "Véhicule" enregistrée dans une base de données de ventes de véhicules d'occasion.

Figure 3. Description de la classe "Véhicule" comportant six attributs

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.

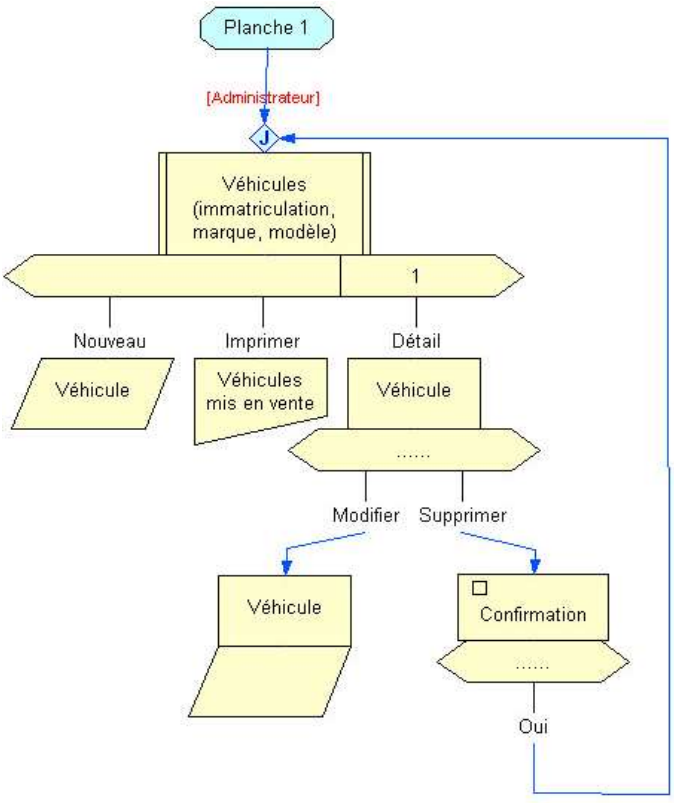

Figure 4. Diagramme SNI réalisé avec l'outil VisualSNI de l'AGL MACAO

2.3 Compléments de modélisation

Des concepts supplémentaires ont été définis dans le modèle SNI pour d'une part, permettre la prise en compte d'exigences particulières des utilisateurs du logiciel concernant l'IHM, et d'autre part d'intégrer de la flexibilité dans la structuration des diagrammes.

Les compléments liés aux exigences des utilisateurs sont :

- l'utilisation de fonctions "FILTRE" et "TRI" lors de l'affichage d'une collection d'objets,
- la notion de "rôle" indiquant les autorisations ou les interdictions de parcours de certaines branches du SNI suivant le type d'utilisateur,
- la notion de condition ou "garde" permettant de parcourir certaines branches en fonction de l'état des objets.

Les compléments liés à la flexibilité d'écriture des diagrammes concernent l'invocation d'un sous-SNI à plusieurs niveaux ainsi que la décision et la jonction qui permettent de parcourir différentes branches sous des conditions indépendantes des choix utilisateur. Il s'agit aussi du découpage en planches permettant de construire un SNI complexe tout en gardant une bonne lisibilité, et des étiquettes de branchement sur une UD ou une planche particulière.

3. Le métamodèle

Le métamodèle du SNI se présente sous forme de trois paquetages le SNI_UD, le SNI_Nœud et le SNI_Port. Remarquons que, pour simplifier sa présentation, nous n'avons pas fait apparaître ici les compléments de modélisation.

3.1 Le paquetage SNI_UD

Ce paquetage présente la structuration des différentes unités de dialogues composées et élémentaires du modèle SNI. La métaclasse UDE est directement dérivée de la métaclasse UML 2.0 "Action" (OMG, 2004). Il en est de même pour UDC qui dérive de "Structured-ActivityNode".

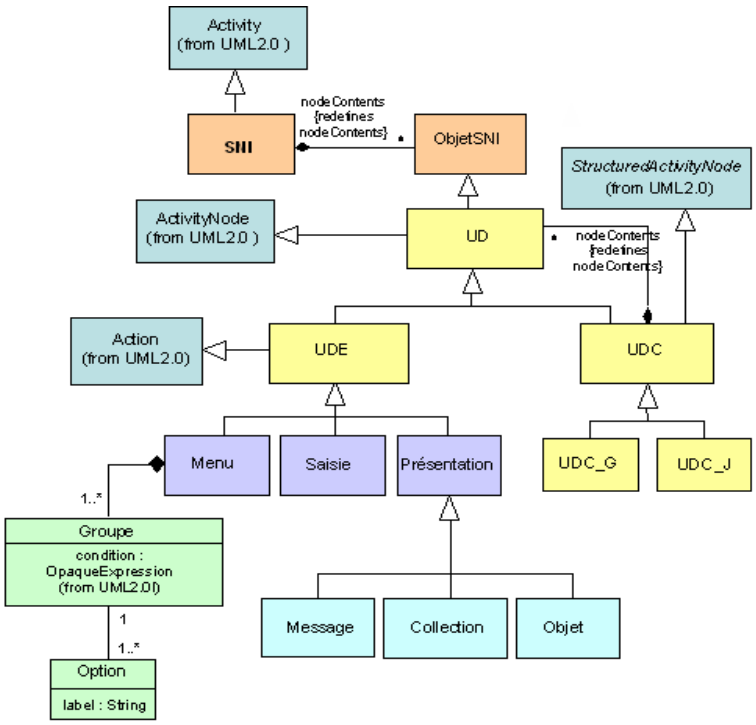

Figure 5. Le paquetage SNI_UD

Comme nous venons de le voir, les éléments de base du SNI sont des UDE et des UDC. Les UDE représentent les actions du diagramme d'activité. Une UDC est composée de plusieurs UD. Une UDC peut être spécialisée en juxtaposition (UDC_J) ou 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 métaclasse 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.

REMARQUES :

a. Bien que le SNI soit en réalité un diagramme d'activités UML *profilé*, il a été nécessaire d'en réaliser un métamodèle (DSL) complet car les premiers essais de mise en œuvre réalisés en profilant simplement le diagramme d'activités dans l'AGL "Together" de Borland ont montré de gros problèmes d'utilisabilité.

b. Les concepts d'UD, d'UDE et UDC se retrouvent dans le modèle UMLi de Paulo da Silva (Pinheiro Da Silva et Paton, 2001). Ils sont appelés respectivement *InteractionObject*, *Primitive-InteractionObject* et *Container*. Les PrimitiveInteractionObjects recouvrent les *Inputters* (Saisie), *Displayers* (Présentation), *Editors* (Modifications : combinaison d'une présentation et d'une saisie) et *ActionInvokers* (Menu).

3.2 Le paquetage SNI_Nœud

Le paquetage SNI_Nœud structure les objets de connexion comme le montre la figure 6. 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.

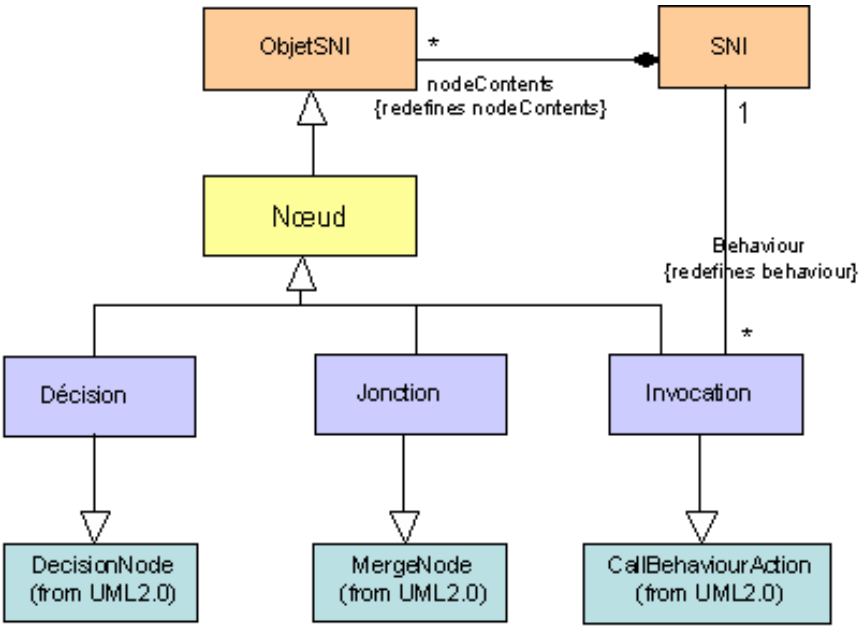

Figure 6. Le paquetage SNI_Nœud

3.3 Le paquetage SNI_Port

Un port est un objet abstrait attaché à une UD ou à un nœud permettant d'établir les connexions. Chaque ObjetSNI 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 options du menu.

A un port d'entrée, il est possible d'associer une précondition qui joue un rôle de garde en interdisant l'entrée dans un objet. De la même manière, à un port de sortie, on peut associer une postcondition qui bloque la sortie tant qu'elle n'est pas vérifiée. Par exemple, dans un site de e-commerce, le client ne peut pas commander si son panier est vide.

En figure 7, 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 métaclasse UML "ObjectFlow". A un port d'entrée, on peut attacher une précondition et à un port de sortie une postcondition. L'une et l'autre sont des expressions UML2.0.

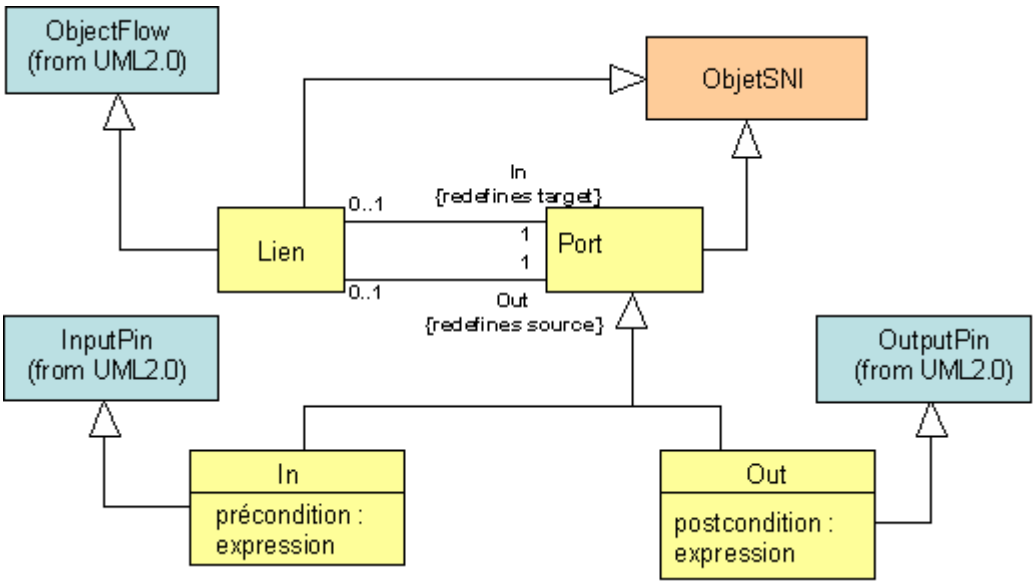

Figure 7. Le paquetage SNI_Port

4. Les modèles de niveau logique

4.1 Objet

Alors que le modèle de haut niveau (SNI) manipule des *unités de dialogue* (UD), les modèles de niveau logique mettent en œuvre des *composants* d'IHM (ou Unités Logiques de Dialogue) qui représentent la mise en œuvre des UD sur une plateforme technique spécifique (*Domain-Specific Language*) : fenêtre pour les IHM de type Windows, page pour les IHM de type WEB, message vocal pour les IHM multimodales, toucher pour les entrées tactiles, *etc.*

Le niveau logique se compose en réalité de deux types de modèles. Le Schéma d'Enchaînement (SE) modélise la dynamique d'ouverture des composants en réponse aux événements tels que une action utilisateur, une condition remplie, la détection d'une erreur ou une interruption système.Le 2ème modèle intitulé « Définition des Composants » (DC) décrit d'une part, le contenu statique des composants en termes de *widgets* (menus, contrôles, textes,

icônes, messages vocaux, entrées vocales…), et d'autre part, leur positionnement dans le composant. Dans cet article, nous ne nous intéresserons qu'à l'aspect dynamique des IHM de type Windows (*GUI*). Dans ce contexte, les composants d'IHM sont les fenêtres et la modélisation de leur enchaînement est réalisée grâce à un profil UML spécifique appelé « Schéma d'Enchaînement des Fenêtres » (SEF).

4.2 Typologie des objets graphiques GUI

Classiquement, les IHM *GUI* sont composées de deux grandes catégories d'objets à savoir les conteneurs qui représentent les composants (menu ou fenêtre) et les contrôles qui sont les objets élémentaires d'interaction avec l'utilisateur.

Nous distinguons quatre types de fenêtres : Fenêtre Primaire (FP ou Principale de l'application), Fenêtre Secondaire (FS), Boîte de Dialogue (BD) et Boîte de Message (BM). Les cinq types de menus sont :

- la Barre d'actions (BA),
- la Barre d'Outils (BO),
- le Menu déroulant (MD),
- le Menu Contextuel (MC),
- le Groupe d'Onglets (GO).

Les contrôles se répartissent en diverses classes telles que :

- Statiques (STC pour Statique Texte Constant, STV pour Statique Texte Variable…).
- Boutons (BP pour Bouton Poussoir, BC pour Bouton Case à cocher, BR pour Bouton Radio).
- Entrées telles que l'Entrée Simple (ES) et l'Entrée Multi-lignes (EM).
- Listes et Collections qui regroupent la Liste déroulante Simple(LS), la Liste déroulante Développable (LD), la Liste Combinée (Combo) Simple (LCS), la Liste Combinée Développable (LCD), la Table, l'Arbre, *etc.*,
- Options regroupant les diverses options de Menu, Onglet, Icône, *etc.*

Cette liste bien qu'elle soit non exhaustive car d'autres types de contrôles personnalisés peuvent être définis en fonction des exigences des utilisateurs, sera la base pour l'élaboration du méta-modèle.

4.3 Le SEF

Le SEF permet de représenter la dynamique d'ouverture des fenêtres de l'application. Il n'utilise qu'un seul type de symbole graphique pour représenter les conteneurs : un rectangle divisé verticalement en deux parties. La partie gauche contient les caractéristiques générales du conteneur telles que le type (FP, FS, BD, BM, MD, MC), le nom, les droits d'accès ou les paramètres d'ouverture. La partie droite est divisée en autant de lignes que de contrôles ou d'options situés dans le conteneur.

Si le nombre de contrôles est trop important, on n'indiquera que ceux qui participent à un événement utilisateur impliquant l'ouverture d'un nouveau conteneur. Un dessin de fenêtre (DEF) est alors réalisé pour présenter la totalité d'entre eux ainsi que leur disposition géographique dans la fenêtre.

Les événements sont représentés par des liens ayant pour origine les contrôles ou options générateurs d'événements et pour destination les conteneurs à ouvrir. Les types d'événements sont indiqués sur les liens (l'événement par défaut est le clic gauche souris). Des paramètres peuvent également être transmis à la fenêtre destinataire. Ils sont indiqués sur le lien, entre parenthèses.

La figure 8 illustre une fenêtre dans le SEF de type « boîte de dialogue » qui présente le détail d'un véhicule. Le Dessin (DESSIN) explicite le contenu de la boîte de dialogue.

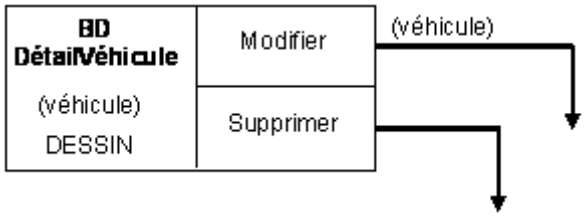

Figure 8. Représentation d'une fenêtre dans le SEF

4.4 Exemple de SEF

Reprenons, à titre d'exemple, le SNI de l'administration de la classe "Véhicule" vue en figure 4. L'administration s'effectue en trois boîtes de dialogue. Les UD de création et de modification d'un véhicule ont été regroupées en une seule boîte de dialogue : *BD-CreModifVéhicule*. Les boîtes de dialogue *BD-DétailVéhicule* et *BD-CreModifVéhicule* sont incomplètes, c'est pourquoi elles seront détaillées par un dessin non présenté ici. La boîte de dialogue *CreModifVéhicule* comporte un paramètre optionnel (entre crochets) selon qu'elle est ouverte à la suite d'une création (aucun paramètre) ou une modification (l'objet "véhicule" doit être présent).

L'impression est représentée par le même symbole que dans le SNI car il ne met pas en jeu de fenêtre si ce n'est une boîte de dialogue commune non envisagée ici, de choix de l'imprimante et des options d'impression.

Remarquons la présence du paramètre "véhicule" qui permet de réaliser la liaison entre les fenêtres.

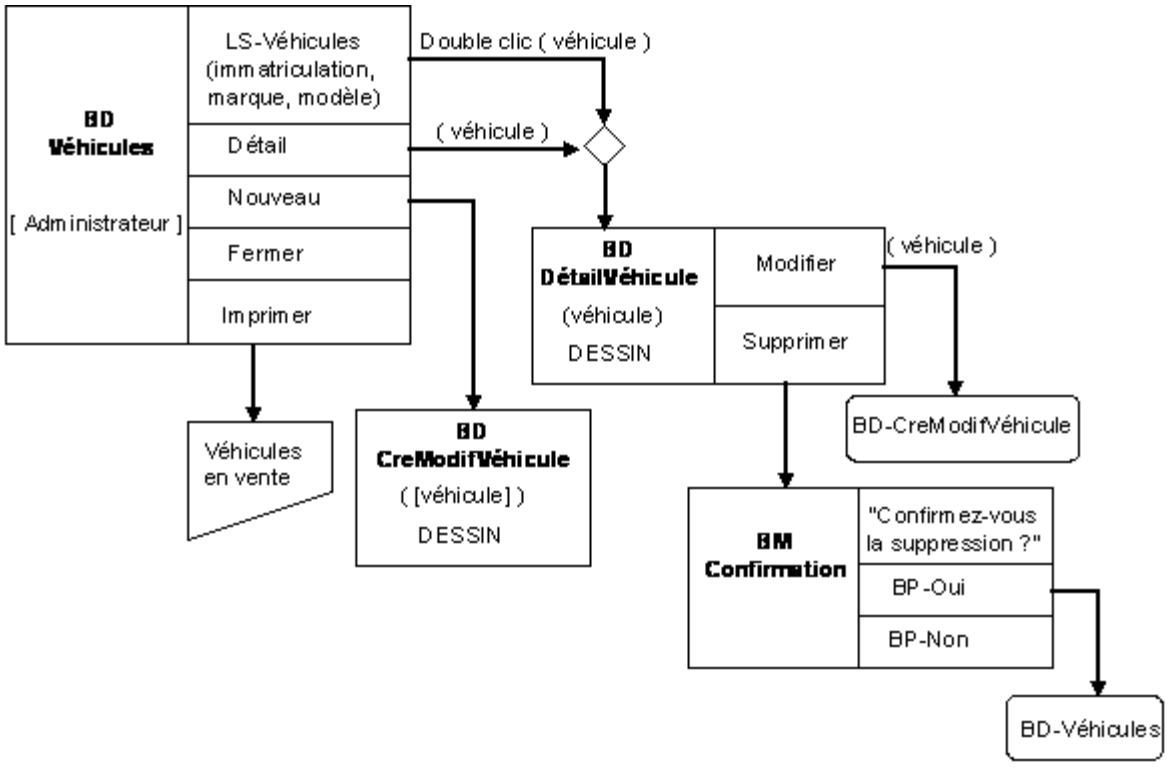

Figure 9. SEF de l'administration de la classe "Véhicule"

4.5 Le méta-modèle du SEF

Comme pour le SNI, nous ne présenterons ici qu'une partie du méta modèle afin de ne pas alourdir l'article. Sa totalité pourra être fournie à la demande.

Le méta modèle présente les principaux objets graphiques composant le SEF. Les liens peuvent connecter tout type de contrôle ou d'option de menu à un conteneur (menu ou fenêtre). La classe des contrôles est spécialisée en quatre sous-classes comportant des objets de différents types (par exemple STC pour Statique Texte Constant ou ES pour Entrée Simple). La classe Conteneur se spécialise en Menu et Fenêtre. Un menu peut être contextuel (MC), déroulant (MD, une barre d'actions (BA) ou un groupe d'onglets (GO).

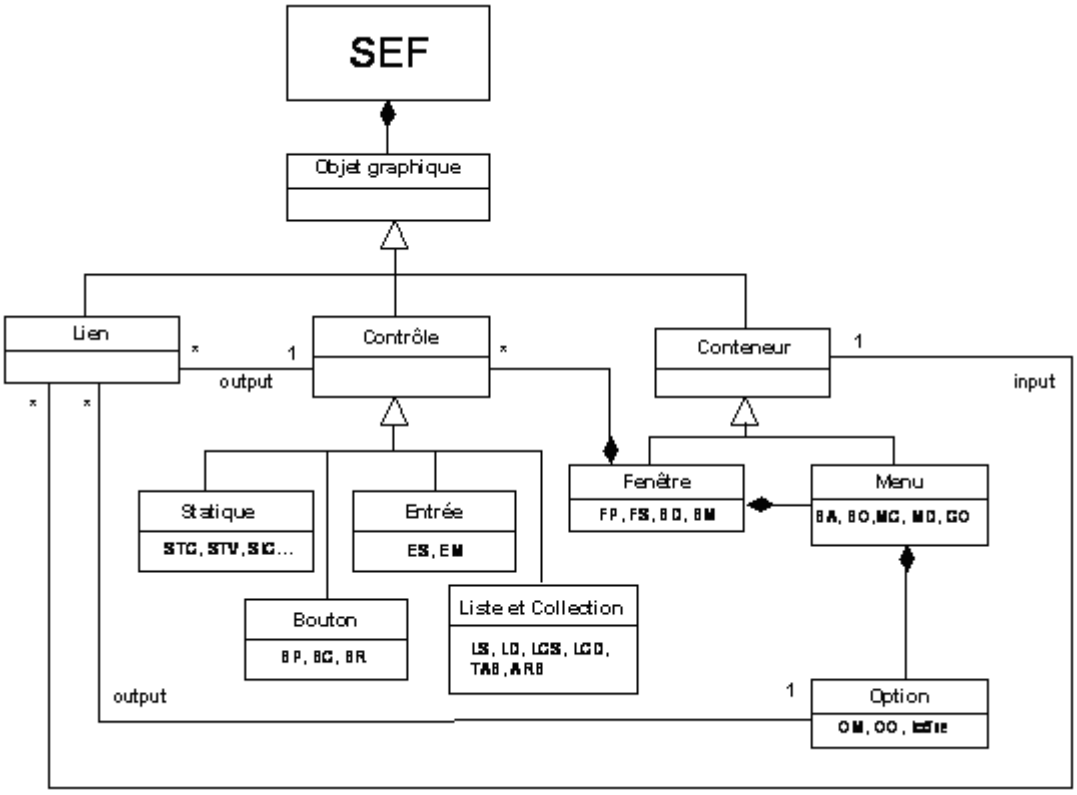

Figure 10. Métamodèle du SEF

5. Transformation de modèles

Deux types de transformation sont actuellement en cours de développement :

- SNI > SEF
- SNI > pages JSP dans le framework JSF de Sun.

5.1 Transformation SNI > SEF

Le but de cette transformation est de générer automatiquement le SEF à partir du SNI d'une application de façon à générer dans un deuxième temps le code du squelette du logiciel à réaliser. Remarquons que le SEF reprend toutes les informations du SNI (sans perte) mais qu'il s'enrichit de spécificités liées aux IHM *GUI* (type de contrôle, de fenêtre, de menu, *etc.*) auxquelles il faut ajouter la structuration des fenêtres (présence d'un dessin de fenêtre) et des informations de navigation (événement déclencheur, paramètres).

Un ensemble de règles permet une transformation quasi-automatique du SNI en SEF. Cependant, l'intervention de l'utilisateur est parfois nécessaire pour, par exemple, décider du

type de contrôle à utiliser ou pour choisir le type de représentation d'une collection sous la forme d'une liste déroulante, d'une table ou d'un arbre. Par souci de concision, nous ne présenterons pas ici la totalité des règles de transformation mais seulement celles qui permettent d'en expliquer le principe.

Le tableau 3 ci-dessous indique dans la colonne de gauche les types d'UD du SNI et dans la colonne de droite les objets graphiques correspondants. Chaque objet graphique est constitué d'un conteneur suivi entre parenthèses des objets qu'il contient (contrôles, barre d'outils, barre d'actions…). Lorsqu'un objet graphique n'apparaît qu'une seule fois dans un conteneur, il est précédé du chiffre "1" (par exemple, 1BA signifie une seule barre d'action). S'il peut apparaître plusieurs fois, il est précédé du signe "*". Par exemple, ***BO** signifie aucune ou plusieurs barres d'outils alors que **1*BP** signifie un ou plusieurs boutons poussoirs.

SNI	SEF
UDE	
Menu de premier niveau	FP (1BA, *BO)
Menu autre niveau	MD ou MC ou GO
Affichage	BD présentation (*STC, *STV, 1BP-Fermer)
Collection	BD présentation (1LS ou 1TAB ou ARB, 1BP-Fermer)
Saisie	BD saisie (*contrôle, *BP-Suite, 1BP-Valider, 1BP-Annuler)
UDC (Juxtaposition)	
Affichage avec menu	BD présentation (*STC, *STV, *BP-Suite, *LS, 1BP-Fermer)
Collection avec menu	BD présentation (1LS ou 1TAB ou 1ARB, *BP-Suite, 1BP-Fermer)
Modification	BD saisie (*contrôle, *BP-Suite, 1BP-Valider, 1BP-Annuler)
UDC (boîte de groupage)	
Boîte de groupage de 1ère niveau avec menu	FP (1BA, *BO, *contrôle)
Boîte de groupage autre niveau, sans saisie	BD présentation (*STC, *STV, *LS ou *TAB ou *ARB, 1BP-Fermer)

Tableau 3. Règles de transformation des UD du SNI en composants graphiques du SEF

La présence d'un "ou" indique que l'utilisateur devra faire un choix parmi plusieurs options possibles.

5.2 Transformation SNI > 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 de converger ainsi plus rapidement vers une solution satisfaisante en termes de couverture fonctionnelle, d'enchaînement des fonctions, de droits d'accès.

Remarquons que le choix de JSF en tant que langage cible a été dicté par son adoption dans la plupart des projets d'Airbus Industrie, principal client de Capgemini Sud.

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.

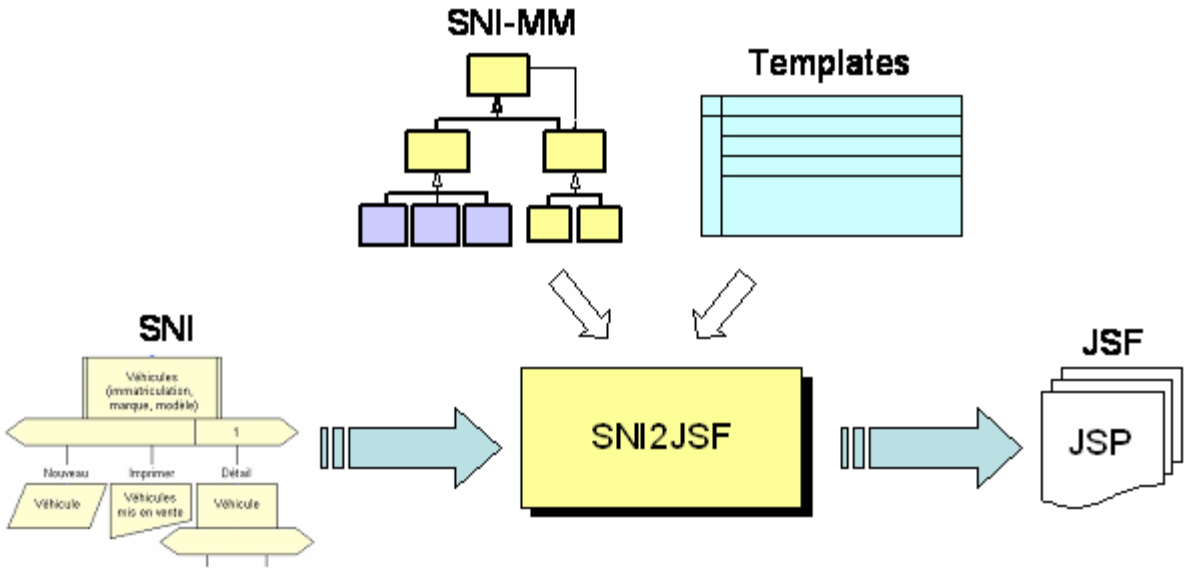

Figure 11. Génération de pages JSP

Comme le montre la figure 11, 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 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 (*open Architecture Ware*) du projet Eclipse GMT (*Generative Modeling Technologies*) (cf. http://www.eclipse.org/gmt/oaw/).

Un premier prototype du générateur SNI2JSF a été réalisé en collaboration avec Capgemini Sud et est en cours de validation.

6. L'éditeur graphique VisualSNI

Le métamodèle du SNI a permis la réalisation d'un éditeur graphique de SNI qui a été développé sous la forme d'un plugin Eclipse sous le nom de VisualSNI. Cet éditeur est disponible en licence GPL à l'adresse suivante : http://sourceforge.net/projects/visual-sni/.

Les trois paquetages du métamodèle ont été fusionnés afin qu'on puisse les reprendre par EMF (*Eclipse Modeling Framework*) (Moore, Dean *and al.*, 2004) pour générer l'ensemble des classes Java nécessaires d'une part, à la construction graphique d'un SNI, et d'autre part à la gestion de la persistance du modèle sous forme de fichiers XMI (*XML Model Interchange*).

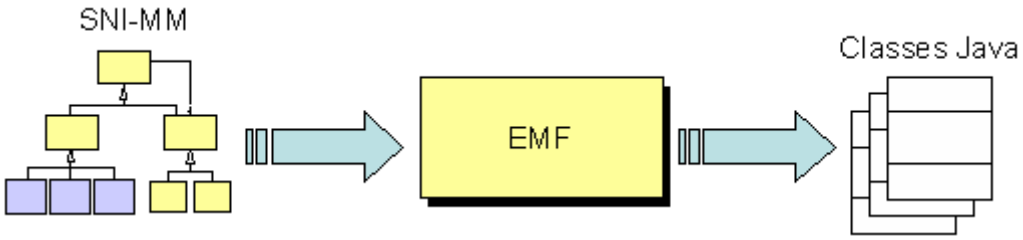

FIGURE 12. Génération des classes Java à partir du métamodèle du SNI

L'éditeur graphique a été construit en utilisant le framework GEF (*Graphic Editor Framework*) (Moore, Dean *et al*., 2004) 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.

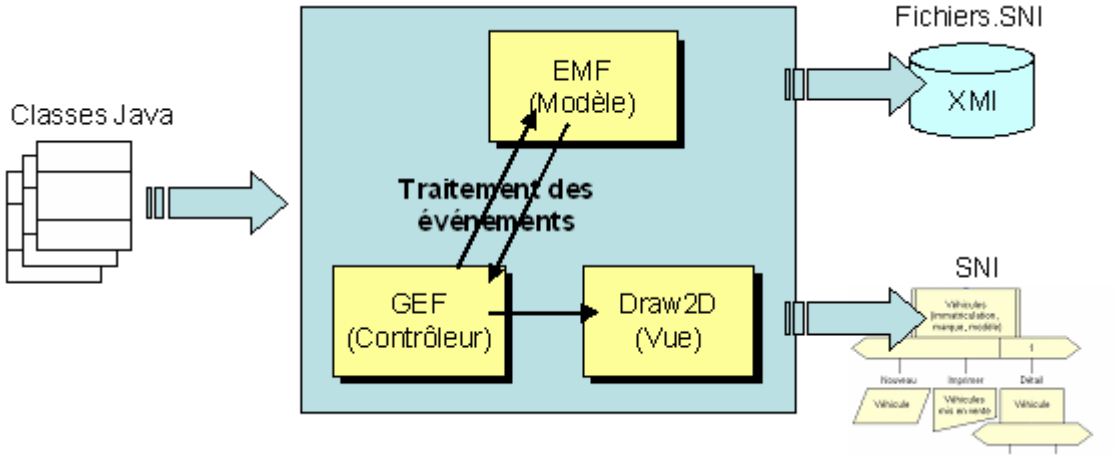

Figure 13. Architecture MVC réalisée par les plugins EMF, GEF et Draw2D

Comme le montre la figure 14, 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.

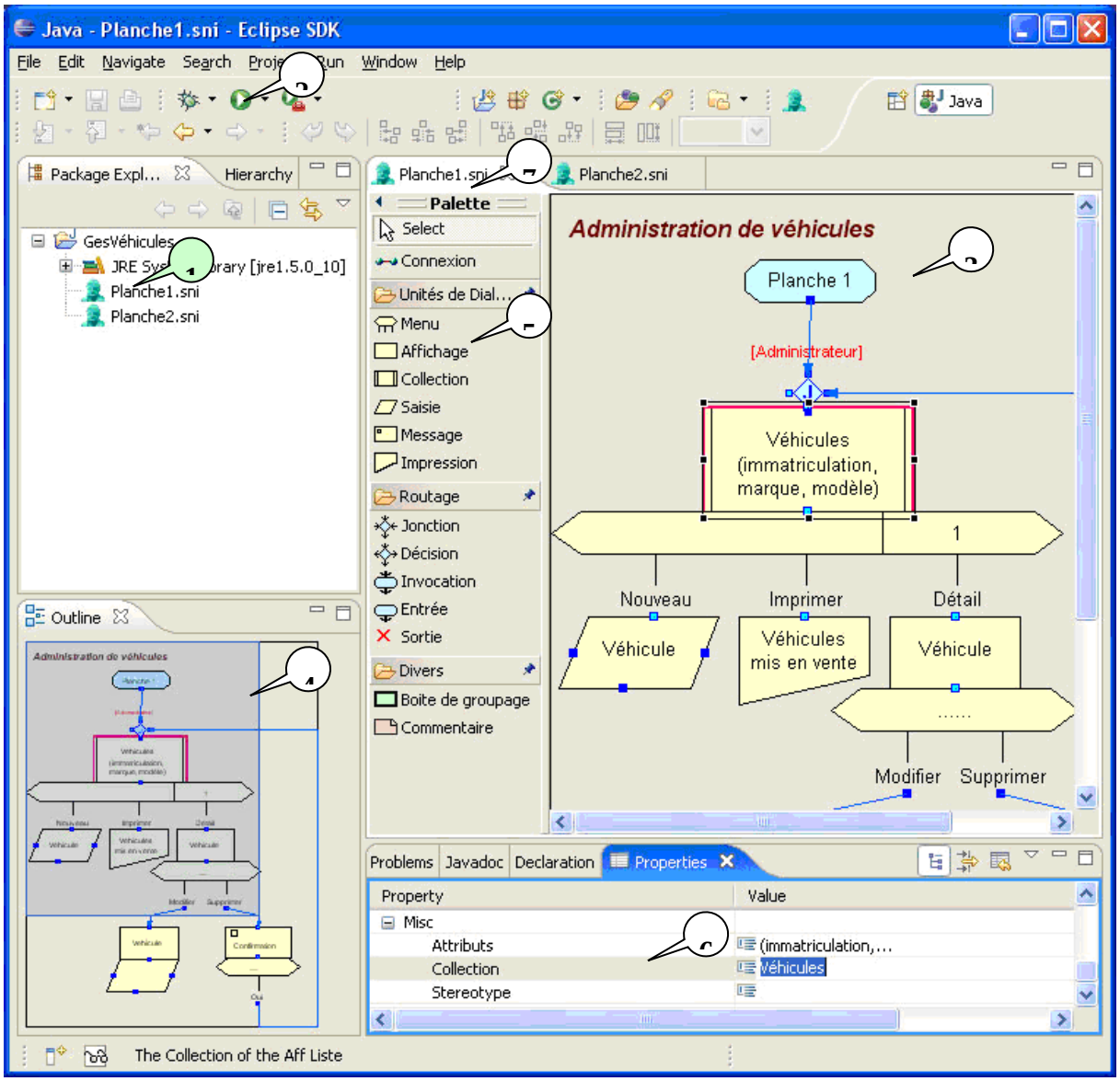

Figure 14. L'éditeur VisualSNI

Comme le montre la figure 14, l'éditeur VisualSNI est composé d'une zone de gestion des ressources (1), d'une barre d'outils (2), d'une zone d'édition graphique (3), d'une vue globale ou Outline (4), d' une palette d'outils (5) et d'une zone d'édition des propriétés des objets (6). Les planches de SNI apparaissent sous forme d'onglets (7).

La figure 15 montre la structure du fichier XMI générée lors de la construction du SNI présenté à la figure 14. On peut y voir la description des deux premiers objets : "Debut" (bulle 1) et "AffListe" (bulle 2).

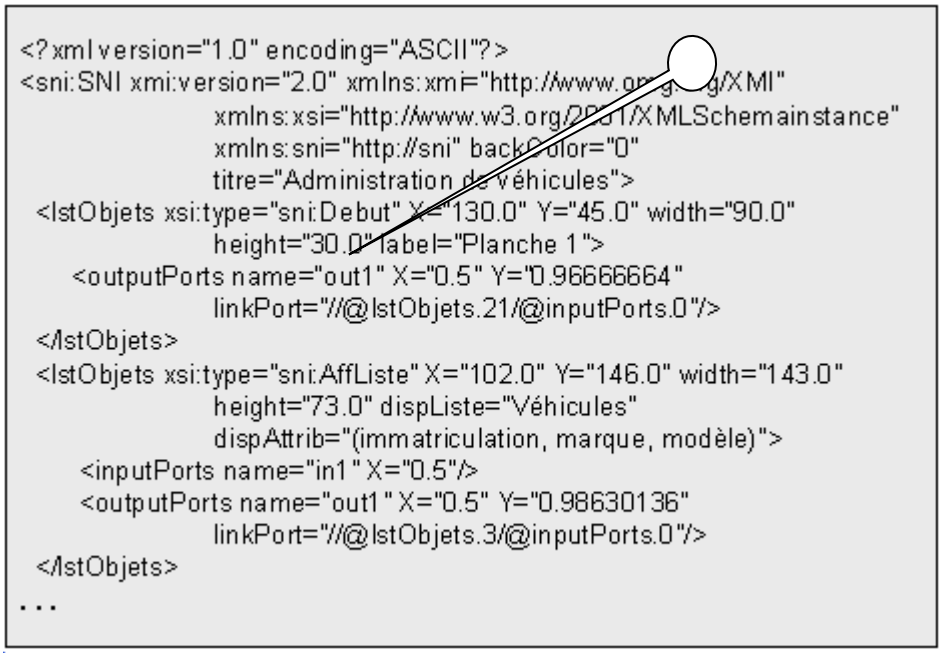

Figure 15. Début du fichier XMI généré par EMF

7. Conclusion

La mise en œuvre de MACAO et de ses modèles d'IHM, principalement le SNI, dans divers projets industriels (CNES, Capgemini Sud, Branche Recouvrement de la Sécurité Sociale regroupant une centaine d'URSSAF), a permis de mettre en évidence trois types d'utilisation :

- aide à l'acquisition et à la compréhension des exigences,
- conception détaillée de l'IHM (Rocques, 2006),
- refactoring de l'IHM.

A) AIDE A L'ACQUISITION DES EXIGENCES

Au travers de plusieurs projets récents (comme PDA de la société SICLI, suivi d'expériences à l'INRA, SARSAT pour l'armée), Capgemini Sud a pu constater que l'utilisation du SNI en tant que langage de dialogue avec les clients lors de la phase d'acquisition des exigences, a amélioré de façon sensible la compréhension mutuelle des besoins concernant les interactions homme-logiciel. En effet, la plupart des logiciels actuels s'articulent autour d'une IHM complexe (graphique, multimodale, multimédia, *etc.*), s'enchaînant suivant une logique choisie par l'utilisateur et intégrant de nombreux objets de dialogue et des résultats pouvant revêtir une grande variété de formes (fenêtres Windows, pages WEB, outils spécialisés de visualisation ou de saisie, *etc.*). 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 :

- l'IHM est la seule partie du logiciel visible par l'utilisateur et c'est donc sur elle que se focalisera son attention,
- un logiciel est rarement utilisé par une seule personne (diversité de 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.

Le SNI a permis d'améliorer ce dialogue mais a montré certaines limites, notamment celle d'une trop grande abstraction, même utilisé en mode esquisse (cf. section 1.2). C'est pour cette raison que la réalisation d'un générateur de maquettes JSF a été entreprise. Ce générateur étant actuellement en cours de réalisation, il n'a malheureusement pas pu, à ce jour, être éprouvé en situation réelle.

B) CONCEPTION DETAILLEE DE L'IHM

Le projet CARINS (CAlcul de Réseaux en régime INStationnaire) du Centre National d'Etudes Spatiales a pour objectif de réaliser un logiciel modulaire permettant de rendre compte des évolutions temporelles des grandeurs physiques caractérisant les systèmes propulsifs de lanceurs spatiaux pendant les différentes phases de leurs missions.

Le logiciel permet de construire graphiquement un réseau maillé, appelé "synoptique moteur" où chaque branche contient un objet du système propulsif étudié (vanne, canalisation, chambre de combustion, *etc.*). La représentation mathématique de chacun de ces objets doit se présenter sous la forme d'un système d'équations différentielles explicites issues des lois d'évolution relatives aux différents phénomènes de transfert thermodynamiques. CARINS doit ensuite produire le système d'équations général, c'est-à-dire finalement construire le simulateur mathématique du système propulsif étudié.

Le logiciel CARINS étant destiné aux ingénieurs motoristes de la Direction des Lanceurs du CNES, il était indispensable que son IHM leur soit parfaitement adaptée tant du point de vue de son ergonomie que de sa structure. Cela était d'autant plus complexe à obtenir que la première version de CARINS comprenait plus de 500 fonctions utilisateur réparties dans 43 cas d'utilisation.

La méthode MACAO a permis de construire un SNI complet de l'IHM en étroite collaboration avec les motoristes du CNES. Ce SNI se développe sur 35 planches avec le logiciel VisualSNI, chacune présentant un aspect particulier de l'IHM. Grâce à lui, il a été possible de développer le logiciel en respectant à la fois les délais impartis et les besoins exprimés.

C) REFACTORING DE L'IHM

Un besoin de plus en plus fréquent réside dans la reprise d'un logiciel ancien ayant une technologie d'IHM dépassée (par exemple de type client lourd ou riche) dans le but de la moderniser dans le cadre des nouvelles technologies (par exemple de type client léger WEB) et cela sans modifier ses fonctionnalités (isofonctionnalité).

L'utilisation du SNI en tant que langage pivot permet de réaliser une telle opération avec un minimum de risques.

Comme l'illustre la figure 16, le processus de refactoring débute par une rétro conception de l'IHM existante de façon à traduire l'ensemble des ses fonctionnalités et leur enchaînement sous forme de SNI. A partir du SNI obtenu, on peut alors entreprendre une phase classique de construction de la nouvelle IHM en passant par le SEF (*Windows-like*) ou le SEP (WEB) suivant le type d'IHM souhaité.

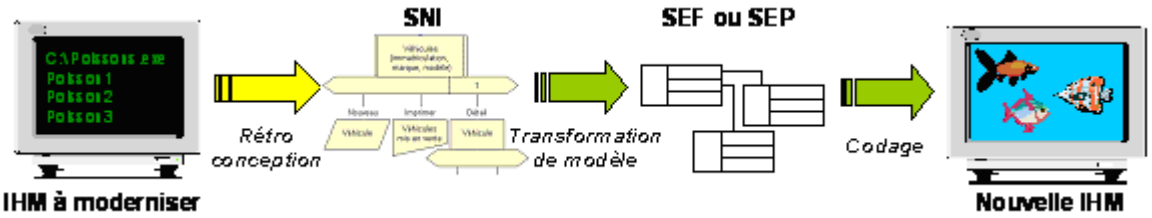

Figure 16. Processus de refactoring d'une IHM par rétro conception du SNI.

Ce travail de refactoring a été réalisé dans la Branche Recouvrement où une centaine d'URSSAF utilisaient le logiciel de *Workflow* WATT sous Lotus-Notes et Domino. Voulant évoluer vers un moteur de *Workflow* utilisant les standards actuels du BPM (SGBD relationnel, client léger, SOA-BPEL, Java…), il a fallu reprendre entièrement l'IHM Notes qui s'exprimait en termes de vues et de masques pour la traduire en pages JSP. Le passage par le SNI a permis de faciliter grandement cette migration qui a été réalisée en isofonctionnalité.

Références

Abouzahra, A., Bézivin, J., Didonet Del Fabro, M., Jouault, F. (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, Californie : 16-20 octobre.

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

Crampes, J.B. (2003). *Méthode Orientée-Objet intégrale MACAO*. Ellipses. ISBN 2-7290-1424-8.

Crampes, J.B. Nonne, L. (2005). Ingénierie d'utilisabilité, Capture des exigences pour l'IHM. *proceeding du congrès AIM'2005*. Monterey, Californie : 24-28 juillet.

Crampes, J.B. (2007). Site J.B. Crampes Consulting. www.jbcc.fr

Dupuy-Chessa, S., Dubois, E., (2005). Requierments and Impacts of Model Driven Engineering on Mixed Systems Design. *Proceeding des journées IDM 2005*. Paris : 30juin-1er juillet.

Fowler, M. (2004). *UML 2.0*. CampusPress. ISBN 2-74401-713-2.

Horrocks, I. (1999). *Constructing the User Interfaces with statecharts*. Addison-Wesley.

Jouault, F., Kurtev, I. (2005). Transforming Models with ATL. *Proceeding of MoDELS 2005 Conference.* MoDELS 2005 International Workshops OCLWS, MoDeVA, MARTES, AOM, MTiP, WiSME, MODAUI, NfC, MDD, WUsCAM, Jamaique : 2-7 octobre, 2005. Revised Selected Papers, LNCS 3844, édité par Jean-Michel Bruel. Springer. Berlin.

Miller, J. *et al.* (2003). *MDA Guides (v1.0.1).* OMG. OMG/2003-06-01.

Moore, B., Dean, D., *et al.* (2004). *Eclipse Development : using the GEF and EMF.* ibm.com/redbooks.

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

OMG. (2005). *Unified Modeling Language: Superstructure (v2.0)*. OMG formal 05-07-04.

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

Pinheiro da Silva, P., Paton, N.P. (2001). *Information Modelling and Knowledge Bases*. Proceeding de "10th European-Japanese Conference on Information Modellind and Knowledge Representation". Saariselka, Finland : May 2000. édité par IOS Press, Amsterdam. Pages 203-217.

Roques, P. (2006). *UML2 - Modéliser une application WEB.* Eyrolles. ISBN 2-212-11770-1.

Sedogbo, C., Grisvard, O., Landragin, F., Lard, J., Proud, S. (2006). HMI Engineering Productivity Thales Group. *Proceeding de l'atelier IDM.* IDM&IHM06. Montréal : 18 avril.

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