Le développement des IHM dirigé par les modèles

Wided Bouchelligua[1], Adel Mahfoudhi[1], Mourad Abed[2]

1 : Faculté des sciences, département d'informatique
Route Soukra Km 3.5 Sfax (Tunisie)
wided_bouchelligua@yahoo.fr, adel.mahfoudhi@fss.rnu.tn

2 : LAMIH (UMR CNRS 8530) Université de Valenciennes
BP : 311 – 59304 Valenciennes cedex 9 (France)
mourad.abed@univ-valenciennes.fr

RÉSUMÉ. Cet article présente une approche à base de modèles pour la génération d'Interface Homme-Machine (IHM). Il décloisonne les recherches unissant l'ingénierie des modèles IDM (Ingénierie Dirigée par les Modèles) autour du problème de génération à partir de spécifications. L'objectif est d'une part de montrer que les concepts de l'Ingénierie Dirigée par les Modèles, s'appliquent au domaine des IHM et d'autre part, de faire bénéficier l'ingénierie des IHM de deux formes générative et intégrative de l'IDM. Nous proposons une démarche de développement des applications interactives visant l'unification de la conception et de l'exécution d'IHM. La conception d'une IHM y est vue comme une série de correspondances entre quatre métamodèles : les tâches utilisateur, les objets du domaine, l'espace interactif et finalement la maquette de l'implémentation.

ABSTRACT. This article presents an approach containing models for the Man-machine generation of Interface (IHM). It decompartmentalizes research linking the Model Driven Engineering MDE around the problem of generation starting from specifications. The objective is on the one hand to show that the concepts of the Engineering Directed by the Models, apply to the field of the IHM and on the other hand, to make profit engineering from the IHM of two forms generative and integrative of the IDM. We propose a step of development of the interactive applications aiming at the unification of the design and the execution of IHM. The design of a IHM is seen there like a series of correspondences between four metamodels: the tasks user, objects of the field, interactif workspace and finally the model of the implementation.

MOTS-CLÉS : Interface Homme-Machine, Ingénierie Dirigée par les Modèles, génération des Interfaces Homme-Machine.

KEYWORDS: Man-machine interface, Model Driven Engineering, Man-machine generation of the Interfaces.

1. Introduction

Un système interactif est une application proposant une interface dirigée par l'utilisateur; application qui ne prédéfinit aucune séquence d'opérations et se contente de répondre aux requêtes qu'elle reçoit de ses utilisateurs. Une interface peut être vue comme un dispositif qui sert de limite commune à plusieurs entités communicantes. Elle doit assurer à la fois la connexion physique entre les entités et effectuer des opérations de traduction entre les formalismes des parties communicantes. Dans le cas de l'interface Homme Machine la connexion a lieu entre l'image externe du système et les organes sensoriaux de l'utilisateur. La réalisation d'une telle interface suppose donc la connaissance précise du comportement de chacune des entités à relier, ce qui rend cette tâche complexe et souvent empirique. Malgré les récents progrès dans les domaine de génie logiciel et de l'ingénierie de la conception, la conception des applications interactives (grand public, embarquée, télécommunication, informatique, industriels,…) révèle de problèmes conceptuels, méthodologiques et technologiques. Pour cela, la conception des IHM constitue, aujourd'hui, un domaine de recherche qui exige des progrès visant la résolution de ces problèmes. Les travaux élaborés dans cet axe de recherche ont menés à de nombreux outils, formalismes et méthodes assurant une couverture plus ou moins complète du cycle de développement des applications interactives. Ces travaux qui sont menées depuis le milieu des années 90, s'appuient sur le paradigme de la conception d'interface utilisateur à base de modèles (Model-Based user interface Design ; MBD) [Szekely., 1996]. C'est une description de la sémantique de l'application et toutes les connaissances nécessaires à la spécification tant de l'apparence de l'application que du comportement du système interactif. Celle-ci décrite dans un langage de haut niveau spécialisé engendre une génération totale ou partielle du code de l'application. Les environnements qui soutiennent ces approches sont appelés MB-IDE (Model Based – Interface Development Environment), [Szekely, 1996] traduit en environnement de développement d'interface à base de modèles. Selon [Szekely, 1996], les principaux composants d'un MB-IDE sont le modèle, les outils de modélisation, les outils de conception automatisée et les outils d'implémentation. Le modèle représente les caractéristiques de l'application du point de vue de l'interface, au sens large incluant les concepts de dialogue et de représentation. Il est le composant central des approches basées sur le modèle.

La Littérature a apporté des réponses afin d'assurer un développement continu de l'analyse à l'implémentation des systèmes interactifs. Plusieurs approches MBD ont été élaboré tel que : MASTERMIND [Szekely et al., 1995], ALACIE [Gamboa-Rodriguez, 1998], DIANE+ [Tarby, 1993], etc. Cependant, aucun travail concret n'a porte ces travaux qui se basent sur les modèles d'interaction au service de l'ingénierie dirigée par les modèles. L'ingénierie des modèles est une forme d'ingénierie générative, par laquelle tout ou partie d'une application informatique est générée à partir de modèles. Une des solutions reconnue actuellement pour atteindre ce but est de passer d'une approche orientée code vers une approche orientée modèles. De ce fait, notre ambition dans cet article est de rapprocher le domaine d'IHM avec celui de l'IDM, par la proposition d'un cadre méthodologique pour le développement des applications interactives. L'intérêt porte particulièrement sur l'unification de la conception et de l'exécution d'IHM par transformation de modèles (Modèle de la tâche, Modèle des objets du Domaine, Modèle de l'Utilisateur…) qui peuvent être productifs au sens de l'IDM [Bézivin et al., 2005].

Après une première section qui pose les concepts de base de l'approche IDM, la section 3 présente une synthèse justifiant notre vision d'intégration de deux communautés IHM et IDM. La section 4 présente la démarche proposée détaillant les métamodèles à la base de la conception de l'IHM. Un cas d'étude simple sera donné et sera employé à titre d'illustration tout au long de l'articule. La section 5 se focalise sur les aspects de transformations possibles entre les différents modèles de la démarche. Enfin, la section 6 fournit une conclusion et développe les perspectives de ce travail.

2. Concepts essentiels de l'IDM

Depuis l'adoption récente du Model Driven Architecture (MDA) [MDA] par l'OMG [OMG], l'approche orientée modèle a suscité un grand intérêt. En fait le MDA a mis un accent fort sur des notions fondamentales telles que les modèles, les métamodèles et les transformations. Cette approche focalisée sur les architectures à base de composants, traite séparément la logique métier et les contraintes technologiques. La logique métier, habituellement cachée dans le code, est ainsi modélisée de manière abstraite et indépendante de toute implémentation (PIM – Platform Independent Model). Puis ce modèle, par des règles de transformation, est automatiquement traduit vers la ou les plateformes souhaitées (PSM – Platform Specific Model). Les intérêts principaux d'une telle approche sont une meilleure qualité logicielle due à la prise en compte rapide des évolutions technologiques (par création ou modification des règles de transformation). Le MDA peut se résumer à l'élaboration de modèles indépendants de plates-formes et à la transformation de ceux-ci en modèles dépendants de plates-formes [Bézivin et Blanc, 2002]. Ensuite, l'approche MDA est devenu une variante particulière de l'ingénierie dirigée par les modèles (IDM en français ou MDE pour "Model Driven Engineering") pour recouvrir aussi les aspects méthodologiques.

L'IDM est basée sur trois concepts essentiels : les modèles, les métamodèles [Terrase et al., 2005] et les transformations. Ces termes fréquemment utilisés dans l'IDM et les relations entre ces termes ont été largement discutés dans la littérature [Seidewitz, 2003], [Bézivin, 2004], [Kleppe et Warmer, 2003], [Bézivin et Gerbé, 2001] et [Atkinson et Kühne, 2003]. Dans [Bézivin, 2004], Bézivin identifie deux relations fondamentales : la première relation, appelée ReprésentationDe est liée à la notion de modèle, la deuxième appelée EstConformeA définit la notion de modèle par rapport à celle de métamodèle (voir Figure 1).

La littérature a donné plusieurs définitions pour le concept modèle, cependant il existe une convergence entre ces définitions. Elles visent toutes à faire référence à la notion de modèle et système modélisé. En effet, un aspect d'un système est capturé par un modèle. La relation qui lie un modèle et un système modélisé est appelée ReprésentationDe. Cette relation est notée μ. Un métamodèle est un modèle d'un langage de modélisation. La notion de métamodèle mène à l'identification d'une seconde relation nommée EstConformeA [Bézivin, 2004] [Favre, 2004]. Cette relation est notée χ. Cette relation permet d'assurer la productivité d'un modèle puisqu'il est conforme à son métamodèle. Ceci facilite la transformation de modèles. La notion de transformation est un autre concept central pour l'IDM, le mécanisme de transformation permet d'utiliser les deux notions Modèle et Métamodèle. La puissance de l'IDM consiste à créer les modèles de transformation. Ces modèles se basent sur les métamodèles correspondant au modèle source et au modèle cible. Ainsi la relation EsttransforméEn notée τ permet d'automatiser la transformation d'un modèle vers un autre.

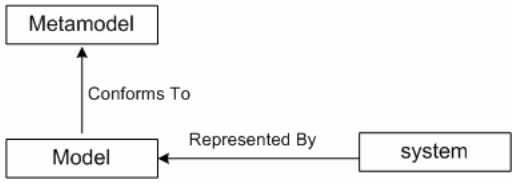

Figure 1: Basic Notions in Model Engineering

Dans cette section, nous avons choisi de présenter les différents concepts et relations de l'IDM, afin de montrer leurs correspondances avec ce qui sera utilisé dans la suite côté de l'IHM. Ainsi, nous optons à une nécessité d'intégration entre les deux communautés IHM et IDM. Tout au long de la démarche de conception de l'IHM proposée les différents modèles seront conformes à leurs métamodèles. Nous serons conforme aussi avec l'architecture pyramidale de l'OMG qui permet de classer et d'organiser les méta-méta-modèles, les méta-modèles, les modèles et les données utilisateurs.

Les niveaux manipulés par notre démarche sont le niveau **M**éta-**M**odèle noté **MM** et le niveau **M**odèle noté **M**.

3. La démarche de conception de l'IHM

Afin d'illustrer les différents modèles et métamodèles de la démarche proposée, nous avons choisi un cas d'étude simple. Il souvent difficile d'appréhender l'approche IDM dans le cadre d'un article, notamment ici avec un nouveau domaine. Ainsi, l'exemple donné a une seule visée de montrer la vision globale de la démarche. Il concerne le développement d'une application permettant aux clients d'un complexe cinématographique de consulter les films à l'affiche et de réserver des billets. Les contraintes suivantes ont été fixées :

- les clients peuvent réserver directement des billets sans passer par une phase de consultation ;

- les clients peuvent réserver des billets pour la séance du jour même, ou pour un autre jour, mais toujours avant le mercredi suivant, jour de sortie des nouveaux films ;
- un client peut rechercher un film à partir de son titre, de sa catégorie, des acteurs présents dans le film, de l'heure de passage du film, ou de la date du film. Un seul critère sera utilisé à la fois ;
- après avoir recherché un film, les clients peuvent réserver des billets pour le film ou obtenir des informations concernant le film.

La méthode proposée s'articule sur une transformation d'une série de modèles couvrant le processus de développement allant de la spécification du domaine (objets et tâches) à la l'IHM finale. Quatre modèles sont à la base de la méthode :

- *Modèle des Objets du Domaine (**MOD**),*
- *Modèle de la Tâche (MT),*
- *Modèle Espace Interactif (MEI),*
- *Modèle d'Implémentation (MI).*

La figure 2 montre une vision globale sur la démarche proposée et représente l'ensemble de transformations (par des flèches) envisagé entre les modèles couvrant la démarche dans une approche inspirée de l'IDM.

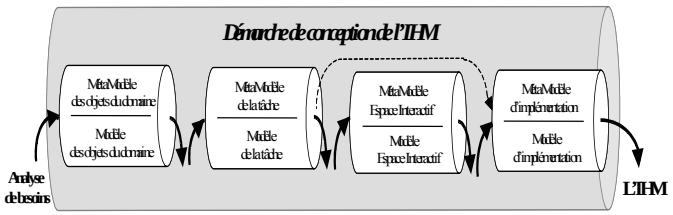

Figure 2: Processus de développement de l'IHM

3.1 (Méta)-Modèle des Objets du Domaine (MM-OD et M-OD)

Le modèle des objets du domaine décrit les entités du domaine, recensées à partir de l'analyse de l'existant et des besoins, manipulées dans les tâches. Il explicite par exemple que le "serveur de cinéma" est composé de "salles" et qu'à un "Film" participe un ou plusieurs "acteurs". La salle est définie par un numéro et un nombre de places maximal. Le métamodèle des objets s'appuie sur les relations d'héritage et d'association telles que définies dans le diagramme des classes UML. Chaque OD est lié à une ou plusieurs tâches dans lesquelles il accomplit à leurs réalisations. La figure 3 propose un Métamodèle et un Modèle des Objets du Domaine, illustré sur le cas d'étude.

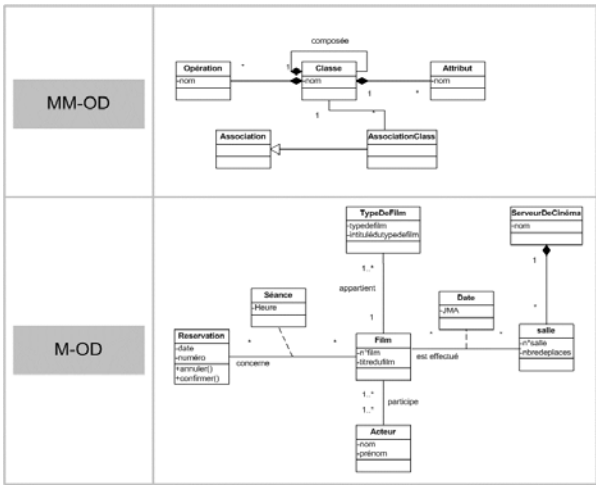

Figure 3: Métamodèle et modèle des objets de domaine

3.2 (Méta)-Modèle de la Tâche (MM-ST et M-ST)

Le Modèle de la Tâche est obtenu à partir d'une décomposition du travail utilisateur en éléments significatifs appelés tâches. Chaque tâche est caractérisée par un but que l'utilisateur vise atteindre. Une telle tâche peut atteindre son but à travers l'exécution d'une procédure qui définit un ensemble des sous-tâches nécessaires pour son accomplissement. Une procédure définie la décomposition récursive des tâches jusqu'à l'obtention des tâches terminales ; c'est-à-dire des tâches qui ne seraient décomposables qu'en actions élémentaires. Les relations entre les tâches sont définies par des opérateurs (séquence, choix et parallélisme). Comme le montre la figure 4, l'application serveur de cinéma se décompose en deux tâches principales : la tâche « chercher un film » et la tâche « chercher un film ». Cette dernière est elle même décomposée en trois sous-tâches : de recherche, de réservation et de collecte d'informations supplémentaires sur les films.

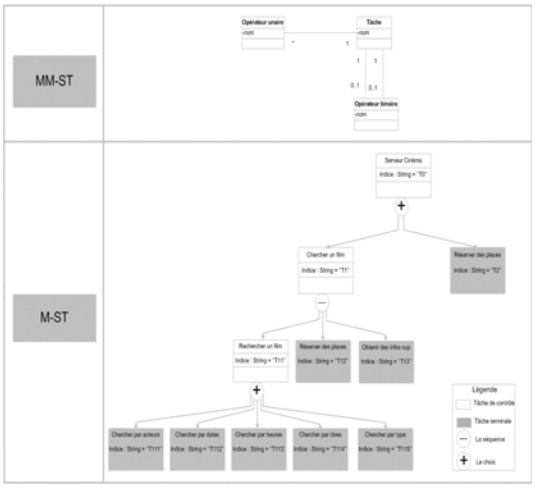

Figure 4: Métamodèle et modèle de décomposition hiérarchique de la tâche racine « serveur de cinéma »

La tâche nécessite un ensemble d'objets qui supporte son exécution nommé ressource. Cette ressource est représentée par une classe composite de la Classe_Tâche (voir figure 5).

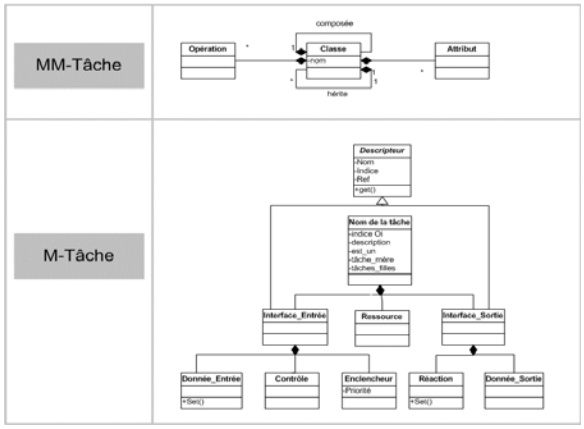

Figure 5: Métamodèle et modèle des objets de domaine

Comme le montre la figure 5, chaque tâche possède aussi :

- un ensemble de descripteurs généraux : nom, description, indice et type,
- une *Interface_Entrée* (IE) spécifiant les objets de domaine nécessaires pour sa réalisation : enclencheurs, données de contrôle et données d'entrée,
- une *Interface_Sortie*, composée des objets de domaine résultant de la réalisation de la tâche qui sont de deux types : évènements de réaction et données de sortie.

Une fois la décomposition est réalisée la méthode prévoit d'affecter les objets de domaine nécessaires pour l'accomplissement de chaque tâche.

3.3 (Méta)-Modèle des espaces interactifs

L'articulation entre modèle de système et modèles de tâches est très importante dans la construction des systèmes interactifs. La réalisation d'une tâche interactive fait intervenir des espaces (ou objets) interactifs et un utilisateur qui sont ses ressources (voir figure 5). Ainsi, notre démarche préconise de décrire le contenu de chaque tâche terminale, identifiée dans le modèle de la tâche, par deux modèles : de l'interface d'utilisation appelé modèle local de l'interface (**MLI**) et le modèle du comportement (cognitif et physique) de l'utilisateur appelé (**MU**). Ces modèles qui décrivent le comportement de l'utilisateur en situation d'utilisation du système permettent de spécifier précisément les différentes stratégies d'appréhension des espaces interactifs, leur enchaînement et leur contenu conceptuel.

Le diagramme d'états/transitions UML est utilisé pour modéliser la dynamique de ces deux modèles.

3.3.1 Modèle de l'Utilisateur (MM-U et M-U)

Le Modèle de l'utilisateur décrit l'ensemble des stratégies qu'un utilisateur est sensé effecteur lors de la tâche. Comme le montre la figure 6, les états du diagramme d'états/transition peuvent être étiquetés par trois états de l'opérateur : Perception (Lecture), Cognition (Evaluation) ou Action (Physique). Ces états correspondent aux trois principaux sous-systèmes de l'opérateur humain (visuel, cognitif et moteur). Alors que les transitions modélisent les actions que l'utilisateur est amené à entreprendre pour exécuter la tâche terminale en question, et qui permettent d'évoluer d'un état opérateur à un autre.

3.3.2 Modèle Local de l'Interface (MM-LI et M-LI)

Le Modèle Local d'Interface décrit le comportement des objets interactifs conformément au comportement de l'utilisateur dans la réalisation de sa tâche. Les états du diagramme représentent les différents états de l'objet. Le comportement de l'objet interactif est défini par l'enchaînement d'états et de transitions. Ainsi, une transition est un changement d'état induit par une occurrence d'événement. Une transition est conditionnée par une garde, ayant pour forme Evénement [Condition]/Action, qui peut exécuter une action et générer des événements.

3.3.4 (Méta)- Modèle Abstrait de l'Interface (MM-AI et M-AI)

Le Modèle Local de l'Interface est construit par la spécification des objets interactifs en rapport avec des tâches terminales indépendamment les unes des autres. En réalité, l'interface utilisateur, et par conséquent chaque objet interactif, ne se limite pas à une tâche spécifique ou à une transition particulière. Pour ce faire, nous définissons le Modèle Abstrait de l'Interface qui décrit des espaces interactifs (ou IHM). La construction d'un espace d'objets IHM suggère l'agrégation de tous les objets interactifs de même nom, du Modèle Local de l'Interface. La procédure d'agrégation est incrémentale, c'est-à-dire elle prend les deux premiers modèles locaux de l'interface en entrée pour en produire un modèle agrégé qui sera par la suite intégré avec un troisième modèle local de l'interface et ainsi de suite. Avant d'effectuer l'agrégation, il faut vérifier si les deux diagrammes d'états/transitions qui constituent les deux modèles locaux de l'interface ont des hiérarchies d'états cohérentes. Notons que si le même état apparaît plus d'une fois à des niveaux différents dans le diagramme d'états/transitions, une erreur de conception est signalée.

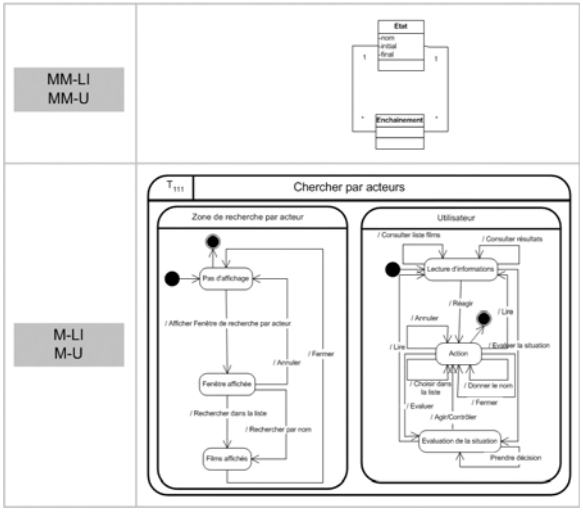

Figure 6: Métamodèle Modèle de la tâche terminale T_{111} : "Chercher par acteur"

L'agrégation de deux diagrammes d'états/transitions se fait par fusion d'états et de transitions. La construction du Modèle Abstrait de l'Interface s'effectue ainsi par l'agrégation d'objets interactifs ayant des caractéristiques semblables (attributs, opérations, etc.).

3.4 (Méta)-Modèle d'Implémentation (MM-I et M-I)

La construction du Modèle d'Implémentation est déduite du modèle de la tâche et du modèle opérationnel. Pour ce faire un ensemble de règles est établi tel que :

Règle 1 : associer à chaque tâche racine et terminale une fenêtre ;

Règle 2 : masquer les fenêtre inutiles qui correspondent à des tâches de contrôle (tâche gestionnaire se trouvant à un niveau hiérarchique) ;

Règle 3 : les boutons associés à chaque fenêtre sont issus du modèle de l'utilisateur (à chaque action possible par l'utilisateur, on associe un bouton qui porte le nom de l'action) ;

Règle 4 : les objets interactifs qui supportent chaque tâche terminale seront placés dans des conteneurs avant d'être placé dans la fenêtre ;

Règle 5 : les objets de l'utilisateur seront également placés dans la fenêtre associée à chaque tâche terminale. Ils seront regroupés dans l'onglet de l'objet interactif qui les manipule ;

Règle 6 : pour la tâche racine et les tâches de contrôle, nous nous basons sur les relations inter-tâches (voir figure 4). Dans le cas où on a une relation de choix, nous associons des boutons radio qui permettent la sélection de la tâche à effectuer. Dans le cas où on a une relation de séquencement, la fenêtre associée à la tâche mère sera inutile et elle sera donc masquée. Par contre pour une relation de parallélisme, les fenêtres associées aux tâches filles s'affichent en même temps et la fenêtre associée à la tâche mère sera masquée.

La figure 7 présente une illustration simplifiée appliquée sur le cas d'étude. Elle présente la fenêtre liée à la tâche terminale "Chercher par acteur".

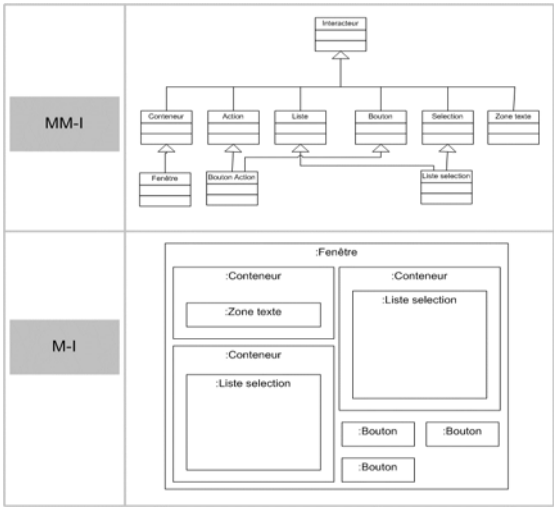

Figure 7 : Métamodèle et Modèle d'Implémentation de la tâche terminale T_{111} : "Chercher par acteur"

4. Ingénierie de la démarche

La démarche de développement des IHM proposée se définit par une série de transformations de modèles. Chaque transformation saisit des modèles en entrée et produit des modèles en sortie, jusqu'à l'obtention de l'interface exécutable. Ceci permet aux modèles d'interaction (Modèles des Objets du Domaine, Modèle de la Tâche et Modèle des espaces interactifs) de passer du stade productif, ce qui rend ces modèles dépendants les uns des autres.

Dans la suite, nous décrivons quelques transformations entre modèles constituant notre démarche. Nous débutons par la transformation Tâche – Espaces Interactifs. Du Modèle de la Tâche MST, les fenêtres associées aux tâches de contrôles et aux tâches terminales sont déterminées. Dans cette correspondance, on applique les règles 1 et 2 du modèle d'implémentation. La figure 8 illustre notre cas d'étude. Les objets interactifs de chaque fenêtre seront placés en appliquant les règles 3, 4 et 5 du modèle d'implémentation pour les tâches terminales et en appliquant la règle 6 pour les tâches de contrôles.

Les objets interactifs de chaque fenêtre seront placés en appliquant les règles 3, 4 et 5 du modèle d'implémentation pour les tâches terminales et en appliquant la règle 6 pour les tâches de contrôles.

La figure 9 modélise le passage du Modèle Statique de la Tâche vers le Modèle d'Implémentation de la tâche de contrôle "Rechercher un film". La relation de choix est transformée sous forme des boutons radio représentant les tâches filles. On a choisit d'associer à chaque objet d'interaction un conteneur.

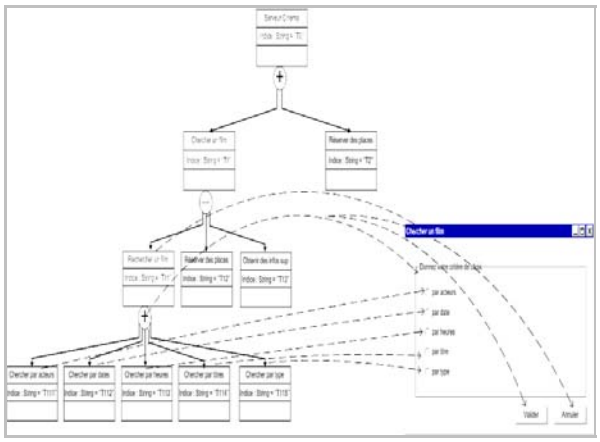

Figure 8 : Correspondances tâches-espaces d'interactifs (fenêtres)

Ces différentes correspondances entre les différents modèles de la démarche doivent être spécifier et implanter à l'aide des langages de transformations. Les différentes transformations sont traduites sous forme des règles qui sont à base des métamodèles.

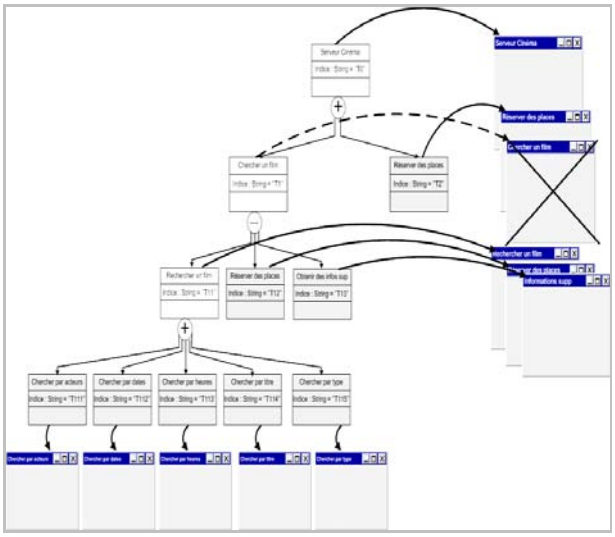

Figure 9 : Correspondances entre le Modèle de la Tâche et le Modèle d'Implémentation de la tâche de contrôle "Rechercher un film".

5. Conclusion

Dans cet article, nous avons présenté une démarche de développement d'Interface Homme Machine basée sur un ensemble de modèles et de métamodèles dépendants les uns des autres. Notre objectif primordial était l'unification des travaux issus de l'ingénierie dirigée par les modèles avec ceux d'ingénierie d'interface Homme Machine.

La mise en évidence des correspondances entre modèles n'est pas suffisant et ne s'agit pas là d'une vision opérationnelle. Il est intéressant d'utiliser un langage de transformation pour modéliser et implémenter cette transformation. Il est aussi nécessaire d'ajouter le métamodèle des programmes pour la liaison avec les artefacts informatiques utilisés par les programmeurs.

Cet article a montré clairement l'intérêt de l'ingénierie des Interfaces Homme-Machine de bénéficier de l'Ingénierie Dirigée par les Modèles notamment lorsqu'on considère la plasticité des IHM pour lesquelles le changement des plates - formes est dynamiques.

6. Bibliographie

[Atkinson, Kühne, 2003] C. Atkinson, T. Kühne: *Model-Driven Development: A Metamodeling Foundation*. IEEE Software. Septembre 2003.

[Bézivin et al., 2005] J. Bézivin, M. Blay, M. Bouzeghoub, J. Estublier, and J.-M. Favre. Rapport de synthèse. Action spécifique CNRS sur l'Ingénierie Dirigée par les Modèles, janvier 2005. http ://wwwadele.imag.fr/mda/as/rapport/AS-MDA-IDM-Synthese-1.1.pdf.

[Bézivin et Blanc, 2002] J. Bézivin, X. Blanc (2002). MDA : *Vers un important changement de paradigme en génie logiciel.* Développeur Référence – juillet.

[Bézivin et Gerbé, 2001] J. Bézivin, O. Gerbé: Towards a Precise Definition of the OMG/MDA Framework. ASE'01 Automated Software Engineering, San Diego, USA, 26-29 Novembre, 2001.

[Bézivin, 2004] J. Bézivin: *In Search of a Basic Principle for Model-Driven Engineering*. Journal Novatica,Issus Spécial. Mars- Avril 2004

[Favre et Nguyen, 2004] J.M. Favre, T. NGuyen, Towards a Megamodel to Model Software Evolution Through Software Transformation, Workshop on Software Evolution through Transformation, SETRA 2004, Rome, Italy, October 2, 2004, Electronic Notes in Theoritical Computer Science, Volume 127, Issue 3, ENTCS ELSVIER.

[Favre, 2004] J.M. Favre, *Towards a Basic Theory to Model Model Driven Engineering.* Workshop on Software Model Engineering, WISME @ UML2004, Lisboa, Portugal, October 11, 2004,

[Gamboa-Rodriguez, 1998] F. Gamboa-Rodriguez (1998). Spécification et implémentation d'ALACIE : *Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques.* Thèse en informatique, Université de Paris Sud – Paris XI, Octobre.

[Kleppe et Warmer, 2003] [KWB03] A. Kleppe, J. Warmer, J. Bast: MDA *Explained-The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[MDA] OMG. MDA Guide Version 1.0.1. 2003.

[OMG] OMG, http://www.omg.org

[Seidewitz, 2003] E. Seidewitz: *What Models Mean*. IEEE Software. Septembre 2003

[Szekely et al., 1995] P. Szekely, N. Sukaviriya, P. Castells, J. Muthukumarasamy, E. Salcher (1995). *Declarative interface models for user interface construction tools: the MASTERMIND approach*. In L. Bass, C. Unger (Eds.), Engineering of Human Computer Interaction, pp. 120-150, Chapman & Hall, London.

[Szekely., 1996] Szekely, P. (1996). *Retrospective and challenge for Model Based Interface Development*. [CADU'96], Namur, pp.xxi-xliv.

[Tarby, 1993] J.C. Tarby (1993). *Gestion Automatique du Dialogue Homme-Machine à partir de spécifications Conceptuelles*. Thèse en informatique, Université de Toulouse I, Septembre.

[Terrase et al., 2005] M. Terrase, M. Savonnet, E. Leclercq, T. Grison, G. Beker. *Points de vue croisées sur les notions de modèle et métamodèle*. IDM'05 Premières Journées sur l'Ingénierie Dirigée par les Modèles, Paris 30 juin, 1 juillet 2005.

[UML] OMG. *Unified Modeling Language* : Superstructure Version 2.0. 2003. OMG