Composition d'Interfaces Homme-Machine dirigée par la composition du Noyau Fonctionnel

Cédric Joffroy

I3S – CNRS – Université Nice-Sophia Antipolis
930, route des Colles - BP 145
06903 SOPHIA ANTIPOLIS Cedex
joffroy@polytech.unice.fr

RESUME
Cet article présente une approche permettant de composer des Interfaces Homme-Machine suite à la composition effectuée au niveau du Noyau Fonctionnel. Cette approche s'appuie sur la mise en place d'un méta-modèle fédérateur et d'un moteur de composition.

MOTS CLES : IHM, NF, Composition, Méta-Modèle

ABSTRACT
This article presents an approach to compose User Interfaces from a composition of the Functional Core. This approach is based on a federative metamodel and a composition engine.

CATEGORIES AND SUBJECT DESCRIPTORS: H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

GENERAL TERMS: Theory

KEYWORDS: HCI, FC, Composition, Metamodel

INTRODUCTION

La composition d'applications existantes nécessite de composer à la fois la partie fonctionnelle et la partie Interface Homme-Machine (IHM). Dans le cadre de nos travaux, nous nous intéressons aux compositions effectuées au niveau fontionnelqui amènent le développeur à devoir réécrire toute l'IHM afin de pouvoir utiliser le résultat de la composition fonctionnelle. Les IHM existantes associées aux différents Noyau Fonctionnels (NF) composés ne sont donc pas exploitées. Afin de pouvoir réutiliser au maximum les éléments présents : NF, IHM, liens entre les deux ; nous avons différents travaux présentés dans l'état de l'art. Dans cet article, l'approche proposée s'appuie sur certains de ce ces résultats afin de proposer une solution qui permet de raisonner au mieux sur l'existant afin de déduire la composition des IHM existantes.

ETAT DE L'ART

L'état de l'art présente les travaux menés dans différents domaines que sont : la composition au niveau du NF, les travaux effectuées autour de la compostion d'IHM et sur les différents niveaux de conception existants, et pour finir les travaux qui s'intéressent à faire le lien entre le NF et l'IHM.

Noyau Fonctionnel (NF)

Les travaux sur la composition au niveau du NF décrivent de manière implicite un modèle de tâches simplifié qui correspond à l'enchaînement des différentes opérations sans se préoccuper de l'enchaînement des tâches utilisateurs. Que l'on se place dans un monde à composants avec Fractal [11] ou SCA [7] ou dans un monde de services avec les travaux sur BPEL4WS [8], on a la description des liens qu'il y a entre les différentes données utilisées et l'enchaînement des opérations.

Cependant, ces travaux qui résolvent le problème de la composition au niveau NF n'étudient pas la répercussion de cette composition au niveau des IHM existantes liées aux différents éléments du NF.

Interface Homme-Machine (IHM)

Afin de pouvoir réutiliser les différentes IHM existantes en vue de les composer, il est donc nécessaire de pouvoir les abstraire afin d'obtenir une base homogène. Les travaux sur la plasticité des IHM distinguent différents niveaux d'abstraction des IHM. Le projet CAMELEON a découpé la partie présentation, donc l'IHM, en quatre niveaux d'abstraction définis dans un cadre de référence [3]. Les quatre niveaux d'abstraction des IHM sont les suivants : (i) les tâches utilisateur et concepts du domaine, (ii) l'IHM Abstraite qui décrit l'IHM de manière structurelle, (iii) l'IHM Concrète qui précise l'IHM Abstraite par l'introduction des interacteurs et (iv) l'IHM Finale qui correspond à l'implémentation dans un langage donné. Au travers de ces différents niveaux d'abstraction, des transformations ont été mises en place afin de pouvoir passer facilement d'un niveau à un autre.

Les travaux sur la composition d'IHM s'appliquent au niveau abstrait dans le cadre d'Amusing avec le langage SunML [12] ou au niveau concret dans le cadre de ComposiXML [9] avec le langage UsiXML [10]. Ces compositions s'effectuent par combinaison de deux descriptions

d'IHM. On a donc une composition structurelle qui utilise des opérateurs tels que l'union, l'intersection, ...

Cependant, la composition effectuée au niveau de l'IHM n'implique pas de répercussion quant aux éléments du NF sous-jacents.

Lien NF–IHM

Afin de conserver une cohérence entre le NF et l'IHM après composition, il est important de décrire clairement les liens qui existent entre l'IHM et le NF. Ces liens décrivent aussi bien les données qui transitent entre les deux niveaux que les événements qui vont déclencher les appels au NF. Les architectures telles que Arch [1], MVC [15] ou PAC [4] séparent le NF et l'IHM mais également les liens qui sont faits entre les deux au-travers du C (Contrôleur/Contrôle) de MVC ou PAC ou dans le Contrôleur de Dialogue d'Arch.

Enfin, des travaux prennent en considération à la fois le NF et l'IHM dans leur composition. Dans le cas des *mashups* proposés par iGoogle ou Netvibes, la seule composition possible est de placer sur une même fenêtre différentes applications sans qu'il n'y ait d'échanges d'informations entre elles. Dans le cas de SOAUI [16], la composition des IHM se base sur le *workflow* décrit au niveau des services. Cette composition ne prend pas en considération des applications existantes qui arrivent avec leur NF et leur IHM. Ce qui est également le cas des travaux sur la composition de fonctionnalités volatiles [6] qui est dédiée au cadre spécifique des applications web décrites en JSP et qui s'appuie sur la programmation aspect pour composer les IHM. Enfin les travaux sur la plannification [5] se destinent aux utilisateurs finaux et proposent de fournir tous les résultats possibles de composition correspondant aux besoins de l'utilisateur.

Bilan de l'état de l'art

Dans le cadre de nos travaux, nous nous intéressons au développeur et au fait de pouvoir réutiliser au maximum les IHM existantes et ce dirigé par la composition du NF. A partir de cette composition, nous souhaitons obtenir l'enchaînement de tâches ainsi que les données nécessaires à son fonctionnement. Pour cela, nous pouvons exploiter les liens décrits entre le NF et l'IHM qui permettent de conserver une cohérence entre les deux niveaux. Enfin, les travaux sur la plasticité et plus particulièrement ceux effectués au cours du projet CAMELEON fournissent les mécanismes de transformation qui sont nécessaires pour réutiliser les IHM existantes. Ces transformations permettent d'abstraire les IHM afin d'effectuer les compositions de celles-ci. Se placer au niveau d'une description d'IHM abstraite pour raisonner sur la composition est un point essentiel.

Notre approche s'appuie essentiellement sur la déduction à partir du NF du modèle de tâches restreint aux seules tâches système et sur la description d'IHM au niveau abstrait associée aux abstractions ou concrétisations d'IHM. Enfin, elle s'appuie également sur la puissances d'expressions des liens entre le NF et l'IHM.

COMPOSITION D'IHM DIRIGEE PAR LE NF

L'approche mise en œuvre afin de réaliser la composition d'IHM s'appuie sur les connaissances qu'il est possible d'obtenir de la composition du NF. En utilisant les différents points forts de l'état de l'art, nous avons mis en place un méta-modèle qui permet de décrire aussi bien l'IHM que le NF, ainsi que les différents liens existants. Ce méta-modèle s'appelle ALIAS (pour Abstract Languages for Interfaces AssemblieS). Un second point important afin d'effectuer la composition est la réalisation d'un moteur de composition qui s'appuie sur des faits Prolog afin de déduire la composition d'IHM.

ALIAS : un Méta-Modele fédérateur

ALIAS [14] est un méta-modèle qui permet de décrire à la fois le NF et l'IHM. En effet, ALIAS s'appuie sur la description des différentes entrées, sorties et "actions" disponibles. Celles-ci possèdent des significations différentes selon que l'on décrit le NF ou l'IHM. Dans le cas de la description du NF, les entrées (resp. sorties) correspondent aux différentes entrées (resp. sorties) des opérations disponibles. Les opérations disponibles sont quant à elles fournies par les "actions". Tandis que pour la description de l'IHM, les entrées correspondent aux éléments sur lesquels l'utilisateur peut agir pour fournir de l'information au NF, les sorties correspondent aux retours du NF, et enfin les "actions" sont des éléments qui permettent de déclencher l'appel d'une opération d'un service.

ALIAS permet également de décrire les liens de données qui existent entre la partie IHM et la partie NF, mais également les liens d'événements. La Figure 1 illustre sur une représentation à composants la partie IHM et NF ainsi que les liens entre les deux. Dans un souci de lisibilité, les flots de données et d'événements sont disjoints, mais l'envoi de toutes les données doit se faire avant l'appel de l'opération. Dans le cas de services, cet envoi se fait en même temps que l'appel de l'opération ; dans le cas de composants il faut que tous les attributs soient affectés avant l'appel.

ALIAS permet également de représenter les compositions de services sur lesquelles nous nous appuyons afin d'effectuer les compositions d'IHM. Ces compositions peuvent être représentées sous la forme de composants-composites. Cela va se traduire dans ALIAS par : (i) des liens entre un composant-composite (résultat de la composition) et ses sous-composants (services composés) et (ii) des liens entre sous-composants (entre services composés).

L'intérêt d'avoir représenté dans ce méta-modèle uniquement les entrées/sorties et les "actions" ainsi que les liens qui existent entre le NF et l'IHM est de pouvoir se re-

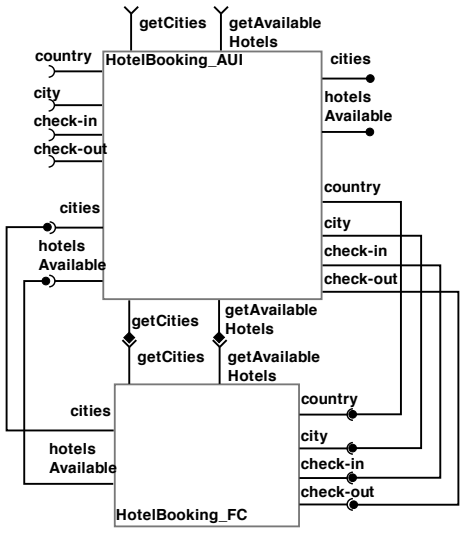


Figure 1 **:** Composants NF et IHM et liens entre les deux

streindre aux éléments nécessaires pour la composition. En effet, la composition de l'IHM étant dirigée par la composition effectuée au niveau du NF, seuls les éléments d'IHM liés aux entrées/sorties et "action" du NF sont nécessaires afin de pouvoir raisonner dessus.

Pour finir, afin de vérifier les résultats de la composition et afin de réutiliser des IHM existantes, différentes transformations ont été mises en place [2], [13]. Ces transformations permettent de faire l'abstraction d'IHM décrites dans des langages de types XML (Flex, Xaml, ...) dans ALIAS, et de concrétiser ALIAS dans différents langages. Il existe également des transformations qui permettent de générer des faits Prolog qui sont les éléments de base pour le moteur de composition.

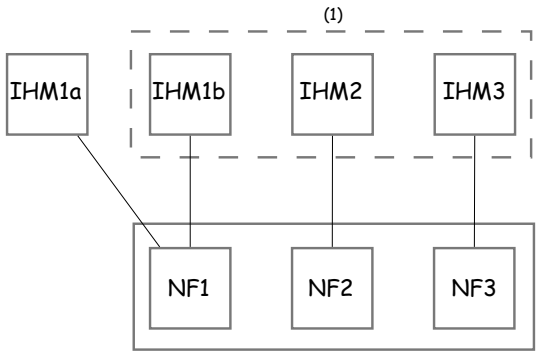
Moteur de composition

Basé sur les différentes informations décrites précédemment (NF, IHM, liens, composition), le moteur de composition va déduire la composition d'IHM suite à la composition du NF.

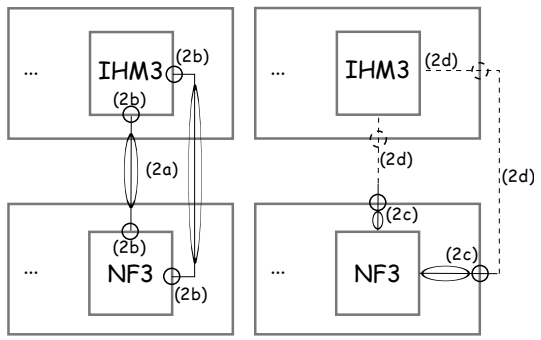
Le moteur s'appuie sur différents types de faits : (i) les faits de bases qui correspondent aux différents NF avec leurs IHM et liens associés, (ii) les faits déclencheurs de la composition qui correspondent à la composition au niveau NF, (iii) les faits résultants de la composition qui correspondent à la composition IHM avec les liens entre les deux niveaux et (iv) les faits propres au moteur qui permettent de faire le raisonnement sur les faits précédents.

Le principe du moteur de composition est de raisonner sur les divers éléments fournis pour réaliser la composition d'IHM et également de fournir des informations quant à la justesse du résultat. Pour se faire, la composition se déroule en plusieurs étapes :

1. identifier toutes les IHM associées aux NF utilisés dans la composition. Si plusieurs IHM sont associées à un même NF pour une même fonctionnalité, il faudra créer les différentes combinaisons possibles (Fig. 2(a));
2. créer les différentes compositions d'IHM correspondantes aux différentes combinaisons. Cette étape se décompose en plusieurs sous-étapes :
   (a) pour chacune des IHM d'une combinaison donnée, identifier pour chacune des fonctionnalités utilisées les liens entre le NF et l'IHM (Fig. 2(b));
   (b) pour chaque lien identifé, extraire les ports d'entrée/sortie et d'événement de l'IHM et du NF (Fig. 2(b));
   (c) identifier pour chacun des ports d'entrée/sortie et d'événement du NF sa liaison avec le composant-composite NF ainsi que les ports impliqués du composant-composite NF (Fig. 2(c)) ;
   (d) créer les entrées/sorties et événements nécessaires du composant-composite IHM et câbler comme il faut pour relier les différents éléments entre eux (Fig. 2(c)).



(a) Identification d'une combinaison



(b) Identification des liens et ports utilisés entre NF-IHM

(c) Identification des liens entre NF et composite NF puis création

Figure 2 **:** Illustration schématique de notre composition

La Figure 2 illustre de manière simplifiée et schématique les différentes étapes de la composition décrites précédemment.

La composition NF peut introduire des fusions au niveau des entrées/sorties, ce qui implique de devoir faire un choix au niveau de l'IHM. Il est possible de pouvoir lever des problèmes de composition afin d'être sûr que le résultat obtenu est bien le résultat escompté. Par ailleurs, selon les besoins de l'utilisateur, il pourrait être intéressant de conserver tous les éléments de la partie IHM et donc de dupliquer la liaison entre la composition NF et la composition IHM. Ceci peut être résolu par l'introduction de critères ergonomiques qui est une réponse possible à la levée des inconsistances.

BILAN ET PERSPECTIVES

Nous avons présenté dans cet article une solution qui permet de composer des IHM suite à la composition du NF. Cette solution s'appuie sur un méta-modèle ALIAS qui permet de décrire à la fois le NF, l'IHM, les liens et les différentes compositions. Ce méta-modèle mis en place permet de réconcilier le monde de l'IHM et du NF en proposant une vision commune de ces deux niveaux. Le moteur de composition quant à lui se base sur les différentes descriptions afin de pouvoir déduire la composition d'IHM.

Cette composition reste relativement simple puisque la composition décrite ne s'appuie que sur la description des liens de données et d'événements ainsi que sur les tâches systèmes. La prise en compte du *workflow* permettra d'aller plus loin dans l'expression des compositions fonctionnelles et donc dans les résultats obtenus au niveau de l'IHM. Le modèle mis en place pourrait être complété par les tâches utilisateurs ou autre modèle de dialogue si ces informations sont disponibles lors de la récupération du service. Dans notre cas, nous sommes partis de l'hypothèse que ces informations n'étaient pas toujours disponibles et avons basé notre solution sur le minimum d'informations fournies. Ainsi, il est déjà possible d'obtenir des résultats qui peuvent être enrichis par le développeur afin d'affiner ceux-ci.

BIBLIOGRAPHIE

1. Bass, L. J., and Coutaz, J. A metamodel for the runtime architecture of an interactive system: the uims tool developers workshop. *SIGCHI Bull.*, 24(1):32–37, 1992.
2. Bihler, P., Fotsing, M., Kniesel, G., and Joffroy, C. Using conditional transformations for semantic user interface adaptation. In *10th iiWAS International Conference*. ACM, Nov 2008.
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting With Computers Vol. 15/3*, pages 289–308, 2003.
4. Coutaz, J. Pac: an implementation model for dialog design. In *Conference Proceedings of Interact'87*, Stuttgart, 1987.
5. Gabillon, Y., Calvary, G., and Fiorino, H. Composing interactive systems by planning. In *UbiMob'08*, pages 37–40, Saint Malo, France, 28 - 30 mai 2008.
6. Ginzburg, J., Rossi, G., Urbieta, M., and Distante, D. Transparent interface composition in Web Applications. *Lecture Notes in Computer Science*, 4607:152, 2007.
7. IBM. Service Component Architecture. http://www-128.ibm.com/developerworks/library/specification/ws-sca/, 2006.
8. Khalaf, R., Mukhi, N., and Weerawarana, S. Service-oriented composition in bpel4ws. In *WWW (Alternate Paper Tracks)*, 2003.
9. Lepreux, S., Hariri, A., Rouillard, J., Tabary, D., Tarby, J., and Kolski, C. Towards Multimodal User Interfaces Composition Based on UsiXML and MBD Principles. *Lecture Notes in Computer Science*, 4552:134, 2007.
10. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., and Trevisan, D. Usixml: A user interface description language for context-sensitive user interfaces. In *ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"*, pages 55–62, Gallipoli, May 2004.
11. Objectweb Consortium. The Fractal Component Model. http://fractal.objectweb.org/, 2008.
12. Pinna-Déry, A.-M., and Fierstone, J. Component model and programming: a first step to manage Human Computer Interaction Adaptation. In *Mobile HCI'03*, volume LNCS 2795, pages 456–460, Udine, Italy, Sept. 2003. L. Chittaro (Ed.), Springer Verlag.
13. Pinna-Déry, A.-M., Joffroy, C., Occello, A., Renevier, P., and Riveill, M. L. In *IDM'09*, pages 131–146, Nancy, FR, Mar. 2009.
14. Pinna-Déry, A.-M., Joffroy, C., Renevier, P., Riveill, M., and Vergoni, C. ALIAS: A Set of Abstract Languages for User Interface Assembly. In *SEA'08*, pages 77–82, Orlando, Florida, USA, Nov. 2008. IASTED, ACTA Press.
15. Reenskaug, T. M. H. Mvc xerox parc 1978–1979. http://heim.ifi.uio.no/ trygver/themes/mvc/mvc-index.html, 1979.
16. Tsai, W.-T., Huang, Q., Elston, J., and Chen, Y. Service-oriented user interface modeling and composition. In *ICEBE '08*, pages 21–28, Washington, DC, USA, 2008. IEEE Computer Society.