Ubiquité des interacteurs

Dibon Pierre
IUT de Bayonne
Université de Pau
Anglet, France
+33559574312
pierre.dibon@univ-pau.fr

Dalmau Marc
LIUPPA
Université de Pau
Anglet, France
+33559574326
marc.dalmau@univ-pau.fr

Roose Philippe
LIUPPA
Université de Pau
Anglet, France
+33559574348
philippe.roose@univ-pau.fr

RÉSUMÉ

La large diffusion des terminaux mobiles modernes entraine de nouvelles habitudes dans l'utilisation et la conception des applications logicielles. Ces systèmes pervasifs font partie de nos vies et les objets connectés procurent aux logiciels de plus en plus de possibilités. Il en résulte que nous devons repenser la façon par laquelle nous abordons les interactions avec ces systèmes. Ces interactions sont potentiellement distribuées sur une multitude de terminaux et utilisent plusieurs modalités d'interaction différentes. Dans cet article, nous proposons de concevoir ces interactions grâce une architecture basée sur des containers logiciels et à une plateforme gérant dynamiquement le cycle de vie des composants qu'ils encapsulent.

ABSTRACT

Modern mobile devices spreading introduce a new way of using and designing software applications. These pervasive systems are now part of our lives. More and more connected objects provide us more and more software capabilities. In this environment we have to rethink the way we traditionally consider our interactions with this software. These interactions are now potentially spread over multiple devices with different interacting modalities. We propose, in this paper, a software container component based architecture to design such interactions and a software platform to manage their dynamic life-cycle.

Keywords

ubiquitous widgets; interactors; human-computer interactions; pervasive applications; context-aware applications ; multimodal interactions

1. INTRODUCTION

La conception de logiciels pour les environnements mobiles actuels place le développeur devant plusieurs difficultés. Ces nouvelles applications bouleversent les habitudes du génie logiciel classique, pour relever les défis qu'elles soulèvent nous devons tout d'abord identifier leurs spécificités. La première d'entre elles est qu'elles sont réparties sur plusieurs terminaux mobiles et s'exécutent, de surcroît, dans des milieux mouvants où la présence de ces dispositifs n'est pas garantie. La seconde est que ces applications prennent de plus en plus de place dans nos vies quotidiennes et que nous les utilisons pour tous types de tâches, à tout moment de la journée et souvent en concurrence avec d'autres utilisateurs. C'est ce qui nous amène à les qualifier d'applications éternelles. Il faut aussi noter que ces applications ont la capacité d'utiliser un nombre croissant de nouveaux objets communicants qui leur permettent de mesurer leur milieu ambiant ou encore d'agir sur leur environnement afin de fournir des services adaptés au contexte. Nous dirigerons donc nos recherches vers les applications distribuées et pervasives ainsi que leurs capacités d'adaptation. Le premier défi consiste en la définition précise des besoins de ces nouveaux logiciels. Nous avons déjà soulevé celui de leur distribution sur plusieurs terminaux : ils devront donc être exécutables sur différents systèmes d'exploitation. Comme ils s'exécutent dans des environnements changeants ils seront capables de s'adapter automatiquement en fonction du contexte. Ceci nous conduit donc à caractériser le contexte, nous en distinguerons trois : le contexte environnemental, le contexte de l'utilisateur et le contexte propre aux terminaux. Concrètement les applications devront avoir la capacité de s'adapter par exemple à un changement d'atmosphère sonore ou à une variation importante de la charge de la batterie d'un terminal ou, encore, à une demande des utilisateurs. Pour pouvoir s'adapter, elles devront pouvoir être déployées dynamiquement sur plusieurs terminaux tout en supportant la migration à chaud de certains services ainsi que de profondes restructurations. Nous réalisons aujourd'hui ces adaptations grâce à une plateforme logicielle, Kalimucho, basée sur une architecture à composants logiciels de type pipe&filter. Cette plateforme ne gère à l'heure actuelle que les aspects fonctionnels, notre but est donc de l'utiliser pour produire des interfaces distribuées, multiutilisateurs, multiplateformes et sensibles au contexte. De la même manière que les applications que nous adressons, ces interfaces devront être duplicables dynamiquement et leur cycle de vie pourra être modifié durant leur utilisation. Nous ajouterons que la répartition des interactions sur plusieurs types de terminaux et d'objets nous amène à envisager une certaine dynamique quant à leur multimodalité. Cet article s'attachera donc à présenter nos recherches visant à fournir une réponse à la conception de ces interfaces et interactions émergeantes. La suite de ce document s'organise de la manière suivante : la deuxième partie propose une série de définitions qui nous amèneront à réaliser ces interactions par des interacteurs dotés d'ubiquité. Dans un troisième temps nous présenterons notre plateforme et une architecture qui permettra de procurer aux interacteurs leur caractère ubiquitaire en réutilisant les méthodes de conception du noyau fonctionnel. La quatrième partie illustrera nos propos à travers un exemple de la vie courante. Le cinquième chapitre présentera nos recherches sous un jour plus formel alors que le sixième décrira certains travaux réalisés dans des domaines connexes. Nous conclurons finalement cet article dans la dernière partie

2. CARACTERE UBIQUITIARE DES INTERACTEURS: DEFINITIONS

2.1 Les interactions

Comme nous avons parlé d'objets d'interaction nous devons définir ce qu'est une interaction. Tout d'abord il faut rappeler

qu'une interaction homme-machine est en fait la relation entre un humain et le système qu'il utilise, une application mobile dans notre cas, afin de réaliser une tâche. Puisque nous traitons d'interactions douées d'adaptabilité il faut ajouter le contexte environnemental à cette relation. Dans notre approche nous distinguerons trois types d'interactions entre l'application l'utilisateur et l'environnement.

- Le premier s'intéresse simplement à la présentation des données de l'application à l'utilisateur.
- Comme nous travaillons sur des systèmes adaptatifs le second type porte sur l'interaction entre l'application et son environnement, celle-ci déterminant son contexte en le mesurant grâce à ses capteurs.
- Le troisième est lié à la transmission des décisions, des questions et des préférences de l'utilisateur à l'application.

Comme nous l'avons dit dans l'introduction nous devrons garantir que nos interacteurs auront, comme tous les composants de nos applications, la capacité d'être déployés dynamiquement sur plusieurs terminaux et de s'adapter aux changements de contexte.

2.2 Modalités des interactions homme-machine

Comme nous avons évoqué plus haut la notion de multimodalité il convient ici de la définir clairement. En premier lieu posons la définition rapide pour le terme modalité d'interaction. Selon [15] une modalité d'interaction est un couple impliquant un dispositif d'acquisition d'une grandeur physique se rapportant à un sens humain et un langage de représentation de l'interaction. Ainsi si l'on considère une interface permettant d'avancer ou de reculer d'une page vers une autre dans l'historique d'un navigateur web, on peut imaginer sur une tablette tactile moderne avoir accès à ces deux fonctionnalités soit en appuyant sur deux boutons tactiles, suivant et précédent, soit en faisant deux gestes tactiles glissés, l'un de gauche à droite l'autre dans le sens inverse. Ces deux manières représentent deux modalités différentes car même si le dispositif physique, l'écran tactile, est le même, les langages d'interprétation sont différents. Le premier langage exprime les deux actions en les mettant en relation avec l'appui de deux zones précises et différentes de l'écran, les boutons, le deuxième avec deux « trajets » horizontaux du doigt sur des zones indéterminées de cet écran. Ainsi une application proposant par exemple ces deux types de modalités verra ses interactions avec l'utilisateur qualifiées de multimodales on parlera également de la multimodalité de ses interactions. Alors même qu'une interaction avec une application est qualifiée de multimodale il faut savoir de quelle manière ces modalités sont combinées. Ces combinaisons de modalités peuvent être exprimées à l'aide des propriétés CARE [8] qui listent la Complémentarité, l'Assignation, la Redondance et l'Equivalence comme piliers de l'assemblage de modalités. Ainsi l'équivalence propose plusieurs modalités pour une même interaction, l'utilisateur ayant le choix d'utiliser l'une ou l'autre. L'assignation représente l'absence de choix. La complémentarité demande d'exprimer une interaction par parties représentées selon des modalités différentes et obligatoirement complémentaires qui une fois additionnées produiront l'interaction complète. Pour la redondance une addition de modalités équivalentes doit être obligatoirement effectuée pour produire une interaction valable. Il faudrait évidemment détailler plus avant ces combinaisons notamment au niveau de leur différentiation pour des interactions en entrée ou en sortie et de leur capacité à réaliser la fusion ou la fission des données voir [15] mais, ces notions de propriétés CARE seront suffisantes pour introduire les opérateurs de combinaison que nous détaillerons dans la section 5.

2.3 L'interacteur

Les développeurs ont l'habitude de programmer les interactions homme-machine des applications mobiles sous la forme de widgets qui couvrent généralement un sous-ensemble des fonctionnalités de ces dernières. Ici nous préférons le terme d'interacteur qui gomme la connotation par trop graphique du widget en le généralisant à tous les nouveaux modes d'interactions que permettent les terminaux mobiles modernes. Nous définissons alors un interacteur comme un composant logiciel qui donne la capacité à un utilisateur d'effectuer une tâche grâce à une application logicielle. Ce composant fournit des entrées et des sorties du côté de l'utilisateur ainsi que de celui de l'application mais est aussi capable de mesurer son environnement [11]. Dans cet article nous considérerons un interacteur comme pouvant indifféremment représenter une interaction, avec l'utilisateur, en entrée ou en sortie laissant aussi la charge au développeur de décider de leur orientation. De cette orientation dépendra la pertinence de l'utilisation conjointe ou séparée des connecteurs d'entré et de sortie de l'interacteur en tant que composant logiciel. Comme nous l'avons déjà dit les applications mobiles actuelles doivent prendre en compte plusieurs modalités pour leurs entrées/sorties. Mais on peut également envisager que plusieurs utilisateurs pourront utiliser simultanément plusieurs interacteurs, répartis sur plusieurs terminaux différents, représentant la même interaction. L'association de cette multiplicité de modalités à cette distribution représente le caractère ubiquitaire de nos interacteurs. En effet ils ont le don d'ubiquité dans l'espace classique à trois dimensions où sont situés les différents terminaux comme dans l'espace des modalités.

3. LA PLATEFORME KALIMUCHO

3.1.1 Présentation de la plateforme

Pour développer ces applications nous utilisons une approche flexible et modulaire s'appuyant sur une architecture basée sur les composants logiciels. Pour superviser ces applications nous avons conçu une plateforme logicielle nommée Kalimucho [13] (qui traduit littéralement du Basque signifie « beaucoup de qualité »). L'architecture d'une application repose alors sur le modèle de composant Osagaia [3] et le modèle de connecteur Korrontea [3]. Korrontea est en fait un connecteur de première classe qui inclut un composant métier dédié aux communications. Ces modèles encapsulent des composants métier (CM) dans des conteneurs qui nous permettent de surveiller leur contexte (activité, QdS, délais etc.) et de contrôler leur cycle de vie (démarrage, arrêt, connexion, déconnexion, migration). Une application est donc construite grâce à des composants fonctionnels Osagaia reliés entre eux par des connecteurs Korrontea. Comme nous le voyons (Figure 1) une entité fonctionnelle est un composant métier (CM) encapsulé dans un conteneur Osagaia dont l'unité de contrôle (UC) expose l'interface nécessaire à la gestion de son cycle de vie. Cette unité de contrôle permet également l'initialisation des données de configuration du composant ainsi que la réception des commandes et l'envoi d'évènements de et vers la plateforme. Les reconfigurations sont liées aux événements de gestion du contexte. Ces événements sont captés et évalués par la plateforme qui déclenchera, si besoin est, les reconfigurations éventuelles en

ajoutant, enlevant ou migrant des composants de l'application. Les unités d'entrée (UE) et de sortie (US) assurent au conteneur la possibilité d'être connecté dynamiquement aux connecteurs conformément aux commandes de configuration envoyées par la plateforme. Précisons ici que Kalimcuho supervise les composants Osagaia/Korrontea en agissant sur leur cycle de vie et en les migrant d'un hôte à un 'autre pour reconfigurer les applications. La plateforme est distribuée sur tous les terminaux impliqués dans l'application. Cet assemblage de composants reliés entre eux permet donc de construire des applications conformes à l'architecture pipe&filter.

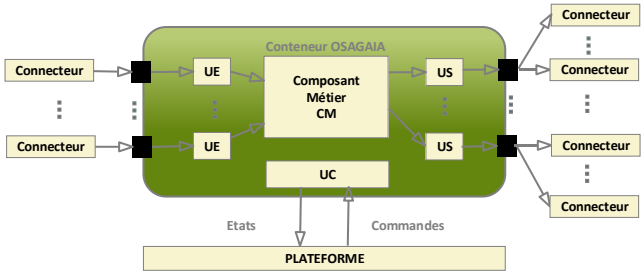

Figure 1. Composant métier

3.1.2 Kalimucho et les interacteurs

Le but de nos travaux est d'utiliser les architectures, modèles de composants et plateforme cités précédemment afin de développer les interactions dont nous avons parlé précédemment. Les interacteurs ont des entrées-sorties et sont adaptables. Comme notre modèle de composant est étudié pour l'adaptabilité nous avons choisi de l'utiliser pour réaliser ces interacteurs. Ceux-ci seront donc, comme les composants métiers, encapsulés dans un conteneur Osagaïa et leurs entrées-sorties seront reliées à des connecteurs. Nous les connecterons donc, via ces connecteurs, entre eux ou aux composants métier de l'application afin de réaliser la présentation des données et des états de l'application à l'utilisateur et de capter les événements d'interaction. Nous pourrons alors utiliser notre plateforme pour fournir l'adaptabilité évoquée plus haut.

4. EXEMPLE DE LA VIE COURANTE

Pour illustrer notre vision du comportement des applications mobiles et notre travail sur l'ubiquité des interacteurs nous allons présenter ici un cas d'utilisation de ces applications dans la vie courante. Il devient évident que les applications mobiles sont aujourd'hui utilisées par la plupart des gens tout au long de leur journée de travail ou à titre personnel. Nous constatons aussi que ce type d'applications est disponible sur nombre de nouveaux terminaux comme les consoles de salon ou les téléviseurs connectés. Nous pouvons alors imaginer un scénario dans lequel un utilisateur travaille chez lui sur son ordinateur personnel dans son bureau tout en écoutant de la musique. Son travail terminé il décide de naviguer sur le web pour trouver une recette de cuisine pour son repas du soir. La recette choisie il va passer dans la cuisine pour la préparer ; ce faisant il ne peut plus utiliser son ordinateur fixe il va donc continuer à consulter la recette mais depuis sa tablette tout en continuant à écouter de la musique. Comme le son de la tablette n'est pas satisfaisant, il écoutera la musque sur la chaine hifi du salon tout en disposant des contrôles de lecture sur sa tablette. Mais la pratique de la cuisine ne se prête guère à l'utilisation d'une tablette tactile, de même que l'écoute de musique produit une ambiance sonore qui exclut les commandes vocales. Il est donc préférable de proposer à l'utilisateur de contrôler la navigation et la lecture de la musique par des commandes utilisant la reconnaissance gestuelle via le périphérique de type Kinect connecté à son téléviseur afin. Nous pouvons également imaginer que la musique, trop forte, dérange un second utilisateur qui souhaitera donc en modifier le volume soit par son téléphone portable soit directement depuis la chaine hi-fi. Nous proposons donc deux diagrammes de cas d'utilisation pour représenter ces deux scénarios. La Figure 2 illustre la situation initiale de manière très simple car les composants métier et leurs interacteurs sont situés sur un seul terminal à savoir l'ordinateur personnel. En revanche quand l'utilisateur s'est déplacé dans la cuisine, les divers composants et interacteurs se retrouvent répartis sur plusieurs terminaux. A partir de ce moment nous nous trouvons en présence de deux applications s'exécutant sur deux terminaux et qui sont contrôlées par plusieurs interacteurs. En d'autres termes, les interacteurs deviennent polymorphes dans la mesure où ils sont dupliqués sur plusieurs terminaux tout en utilisant des modes d'interaction différents. Par exemple l'interacteur du contrôle du volume sonore est present à la fois en mode tactile, sur la tablette du premier utilisateur et le téléphone du second, et en mode gestuel sur la télévision, via le Kinect (voir Figure 3).

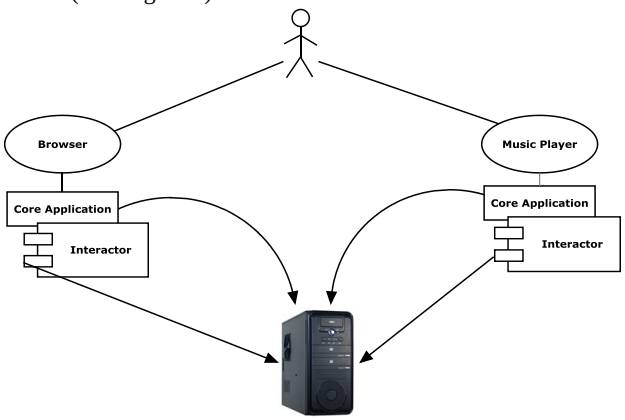

Figure 2. Use Case 1

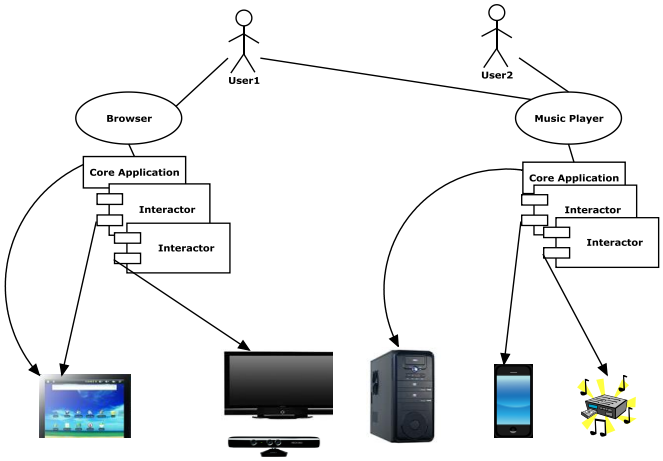

Figure 3. Use Case 2

Par ailleur nous pouvons remarquer que, dans ce scénario, certains composants, comme le navigateur, migrent de l'ordinateur vers la tablette. De la même manière, des interacteurs

peuvent être dupliqués ou migrés à chaud d'un terminal vers l'autre comme le sont, par exemple, les contrôles du son. Ce fonctionnement est rendu possible grâce à l'encapsulation des composants fonctionnels de l'application et des interacteurs dans des conteneurs Osagaia. Ces derniers sont gérés par notre plateforme et les liens qui leur sont nécessaires fournis par des connecteurs Korrontea qui permettent d'établir dynamiquement les connections à travers le réseau.

5. NOTRE PROPOSITION

5.1 Une architecture pour les interacteurs

Dans les cas d'utilisation précédents nous nous sommes focalisés sur l'adaptabilité des interfaces mobiles. Nous avons vu que nous pouvions traiter les interacteurs comme des composants métier en les encapsulant dans des conteneurs. Tout comme nous avions nommé les composants fonctionnels « Composants Métier » (CM) nous nommerons les interacteurs « Composants d'interaction » (CI) c'est-à-dire des composants donc la fonction a uniquement trait aux interactions. Un CI est responsable de la présentation des données, des événements et des effecteurs (boutons ou gestes) du cœur fonctionnel des applications comme dans l'exemple du lecteur de musique de la section 4. Un CI est également capable d'utiliser plusieurs modalités d'interaction pour adapter la communication entre les utilisateurs et l'application en prenant en compte le contexte. Dans la section 2 nous avons précisé que l'interacteur possédait des entrées-sorties. Nous retrouvons ce type d'architecture pour les interacteurs dans [14] où l'équivalent du CI est nommé Unité d'Abstraction et d'Affichage et possède des unités de contrôle pour les entrées-sorties. La principale différence avec notre travail tient au fait que nous laissons l'abstraction côté utilisateur à la charge du développeur du code de l'interacteur pour nous concentrer sur les connections avec le cœur fonctionnel de l'application. En relation avec le caractère ubiquitaire des interacteurs évoqué dans la section 2.3 nous devons introduire un troisième composant logiciel pour traiter la combinaison de modalités. Ce composant nommé Composant de Combinaison (CC) sera de la même manière encapsulé dans un conteneur Osagaia et sera spécialisé dans les opérations de combinaison en relation avec les propriétés CARE, cet opérateur aura aussi la charge des fusion et fission de données qu'il nous faudra implémenter dans la suite de nos travaux. Pour nous les entrées des interacteurs viennent uniquement de la partie fonctionnelle, via un CC, alors que les sorties sont la conséquence des choix de l'utilisateur et des données produites via le CI et seront injectées, toujours au travers d'un CC, dans l'application qui les traitera ensuite. Dans notre exemple nous avons montré qu'un interacteur pouvait être dupliqué sur plusieurs terminaux mais aussi que plusieurs instances pouvaient être utilisées simultanément par plusieurs utilisateurs. Ceci nous oblige à envisager la coopération entre ces diverses instances d'un même interacteur. Le problème essentiel étant celui de l'utilisation concurrente de la même interaction par deux utilisateurs. Les deux instances devront donc être à l'écoute l'une de l'autre afin de présenter les choix d'interaction effectués aux autres instances. Pour ce faire nous introduisons une boucle de retour qui connecte les entrées des diverses instances du même interacteur à la sortie de l'opérateur qui les relie au noyau fonctionnel. Afin d'assurer ce fonctionnement nous avons modifié notre modèle Osagaia pour lui donner la capacité de connecter simultanément plusieurs connecteurs sur chacune de ses UE. Nous présentons ces modifications sur la Figure 4 et spécialisons donc le CM en CI. La même spécialisation sera implémentée pour les CC. Ceci nous conduit à une vision des applications mobiles sous la forme d'une architecture à trois couches présentes non seulement au moment de la conception mais également à celui du déploiement. Nous envisageons bien évidemment ici des applications comprenant un ou plusieurs composants métier adaptatifs (CM) mais également un ou plusieurs composants d'interaction (CI) reliés par un Composant de Combinaison (CC). Les trois couches de cette architecture sont : une couche pour les CM et une couche pour les CI (voir Figure 5) et une couche pour les CC qui communiquent grâce à des connecteurs. La plateforme Kalimucho est alors en charge de la décision de l'ajout, de la suppression et de la migration des CM, des CI et des CC. Ces décisions sont prises en se référant à des règles qui prennent en compte le contexte (environnement, désirs de l'utilisateur …). Ces règles indiquent si un CM peut ou non accepter plusieurs CI et si un CI peut ou non être connecté à plusieurs CMs. Même si il nous reste à définir ces règles dans le futur nous pouvons d'ores et déjà imaginer qu'elles prendront en compte le contexte lié à la modalité d'interaction et permettront en particulier de remplacer une modalité par une autre ou éventuellement de remplacer une interaction simple par une combinaison de deux modalités complémentaires afin de palier par exemple à la baisse de fiabilité inhérente à un changement de contexte physique, par exemple un son de la musique trop fort entrainant une baisse de la fiabilité de la modalité de reconnaissance vocale. La règle régissant cet exemple aura la forme « **si** contexte sonore dépasse seuil s, **alors** remplacer reconnaissance vocale par reconnaissance vocale + complément acquittement par bouton graphique ok ». On voit ici que notre plateforme aura la charge de fournir un service de contexte d'interaction et de prendre les décisions de remplacement de l'opérateur et migration ou d'ajout de modalité sur la base de cette règle.

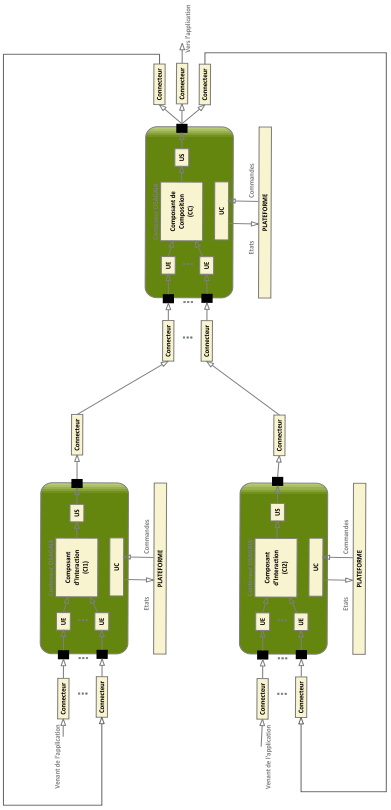

Figure 4. Interacteurs, combinaison et boucle de retour

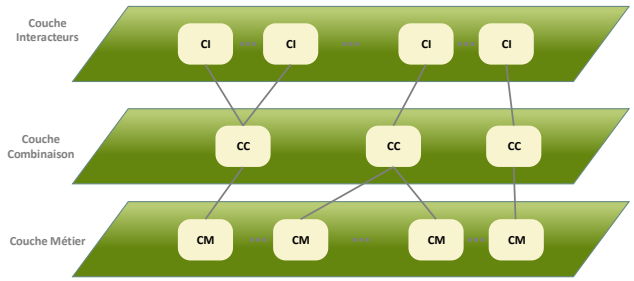

Figure 5. Architecture en trois couches

Quand plusieurs CI sont connectés au même CM par un CC d'équivalence ils voient une de leur entrée automatiquement reliée à ce dernier par une boucle de retour (voir Figure 4). Jusqu'ici les exemples choisis utilisaient des interactions en entrée il faut inverser tous les raisonnements et toutes les connexions pour envisager le cas des interactions en sortie.

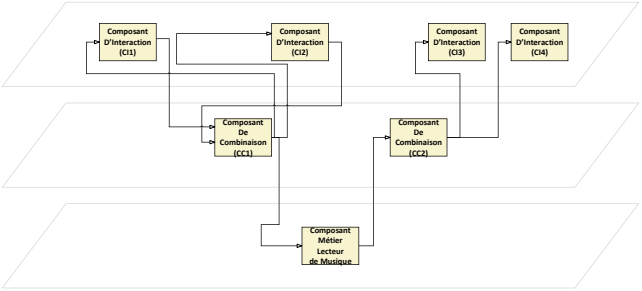

Figure 6. Implémentation de l'architecture pour le lecteur de musique

5.2 L'exemple du lecteur de musique

Dans cette partie nous détaillerons une implémentation concrète de notre architecture en trois couches basée sur la partie du scénario évoqué plus haut concernant le contrôle du lecteur de musique. Dans cet exemple nous considérons un interacteur composé d'un des contrôles classiques comme start, stop, pause, volume+- et d'un interacteur de sortie pour l'affichage du volume sonore. Quand l'utilisateur agit sur les contrôles, le premier interacteur envoie par ses sorties des événements d'interaction au composant métier du lecteur. Comme cet interacteur est dupliqué sur un autre terminal nous avons représenté une autre instance de ce CI sur le schéma de la Figure 6. Ces deux instances sont reliées au composant métier du lecteur par deux connexions qui transitent par un composant de combinaison d'équivalence CC1 et fournissent la même fonctionnalité d'interaction sur les deux terminaux sur lesquels elles sont déployées, cependant elles utilisent deux modalités différentes. Par les mêmes connexions inversées nous retrouverions deux instances de l'interacteur en sortie sur deux terminaux (chaine hi-fi et téléphone portable) opérant selon deux modalités différentes (leds sur la chaine hi-fi et barre de niveau graphique sur le téléphone) et recevant la même donnée de volume sonore depuis l'opérateur de combinaison CC2

6. TRAVAUX CONNEXES

Nous pouvons tout d'abord citer ici l'adaptation hypermédia comme première approche de l'adaptabilité des interfaces. Cet axe de recherche [4] fit son apparition au début des années 90 et avait pour but principal l'adaptation des interfaces web du point de vue de l'utilisateur. Nous pouvons, même si elle peut paraître éloignée de nos travaux, citer en particulier l'architecture AHA! [9] qui fournissait un framework permettant de réaliser ce type d'adaptation d'interfaces. Ces travaux sont à l'origine des adaptations modernes d'interfaces web pour les dispositifs mobiles. En regardant ces travaux de plus près nous pouvons nous apercevoir qu'ils sont, en fait, assez proches de nos préoccupations puisqu'ils évoquent la distribution d'applications sur plusieurs terminaux et la nécessité d'adapter les modalités de données à l'utilisateur et au contexte environnemental.

Nous souhaitons également évoquer le prototype QuickSet [7] réalisé à la fin du siècle dernier. Il fut conçu pour illustrer la possibilité d'envisager les interactions homme-machine par une approche multimodale et adaptative. Ce prototype était assez proche des systèmes distribués classiques puisqu'il utilisait Corba et DCOM comme intergiciels de communication. Le but final semble assez proche du nôtre mais l'angle choisi est très différent. Quickset utilisait des agents autonomes pour traiter de l'adaptabilité tout en se basant sur une architecture et des intergiciels complexes. Ceci est complétement à l'opposé de notre démarche qui vise à utiliser la plateforme Kalimucho développée en interne pour éviter le millefeuille d'intergiciels, de frameworks et d'architectures complexes et ainsi concevoir des applications mobiles distribuées simples relativement légères et performantes.

Pour illustrer un exemple de ce que nous appelons un millefeuille applicatif nous pouvons citer l'approche DynaMo [1] qui propose également un ensemble d'outils permettant de résoudre nos problématiques avec une approche très similaire puisqu'elle propose notamment des composants de médiation respectant les propriétés CARE qui jouent un rôle très proche de nos opérateurs de combinaison. Cette approche propose également d'utiliser un gestionnaire autonomique se chargeant de faire appliquer des règles en tenant compte du contexte pour générer, déployer et reconfigurer les interactions. Ces travaux sont bien plus aboutis que les nôtres, en particulier sur la partie modélisation des interactions, mais nous semblent illustrer parfaitement l'empilement de couches applicatives puisque basés sur la conjonction de la plateforme OSGI du middleware ROSE et du runtime IPOJO. Loin de faire ici une critique de ces assemblages, qui utilisent des standards éprouvés de l'industrie, notre approche est avant tout orientée sur la recherche d'efficacité et de simplicité quant à la conception d'application mobiles s'exécutant sur des terminaux mobiles. Ces terminaux n'ont pas forcément la puissance de calcul nécessaire à l'exécution de telles infrastructures, il n'est même pas garanti que le code de leurs briques soit porté pour les systèmes d'exploitation mobiles. De plus cette approche propose une multiplicité de services (Upnp, web service, DPWS) utilisant chacun un protocole différent et demandant certainement la possibilité d'accès à un grand nombre de ports réseau là où notre plateforme en nécessite seulement un petit nombre. Pour finir il semble que ces travaux soient axés sur un dynamisme basé sur la découverte et la versatilité des périphériques d'interaction là nous adressons une dynamique d'exécution fondée sur la versatilité des plateformes (les

terminaux) et la migration du code des applications et de leurs interactions.

Nous pouvons également mentionner les travaux sur la plasticité des interfaces [6,5] qui nous paraissent très proches des nôtres. Nous souhaitons tirer de ces travaux une base théorique pour la conception orientée modèles des interfaces adaptatives. En effet, leur définition des interacteurs, nommés COMETS, pour COntext of use Mouldable widgETs, est globalement proche de la nôtre. Cet axe de recherche a pour but de proposer un framework permettant au concepteur de réifier des interfaces à travers divers niveaux d'abstraction jusqu'à l'implémentation concrète du code en raffinant successivement les modèles des différents niveaux par des transformations. Les auteurs parlent également d'interactors et utilisent un framework général d'abstraction (CAMELEON-RT [2]) pour gérer les aspects de sensibilité au contexte et d'adaptabilité.

Finalement nous pouvons citer une approche récente dont le but est de fournir une méthodologie permettant la conception d'interactions multimodales [12]. Les auteurs ont réalisé un atelier logiciel complet de conception des interactions basé sur les modèles et outils du framework OpenInterface [16]. Cet atelier comprend une palette qui va de l'outil de conception graphique jusqu'à la plateforme d'exécution. Ces travaux se focalisent sur la conception des applications alors que nous nous concentrons sur leur implémentation pratique au travers de notre plateforme. Il n'est donc pas impossible que nous reposions sur eux pour les niveaux d'abstraction plus élevés de la réalisation de nos interacteurs ubiquitaires. C'est à dire pour la modélisation de l'environnement, des utilisateurs et des tâches afin de fixer les modèles des règles qui seront nécessaires aux prises de décisions concernant la gestion du cycle de vie des interacteurs et de leurs relations avec le cœur fonctionnel des applications. Ces préoccupations de modélisation et de description des interactions dynamiques sont un point crucial du processus de conception et représentent un axe de recherche particulièrement actif qui s'appuie généralement sur des langages dérivés du XML comme par exemple SMIUML [10].

Pour terminer, notre travail adresse également, de par la nature répartie des applications qu'il étudie, les problématiques de répartition et de synchronisation des interfaces utilisateur distribuées. Ces mécanismes peuvent être régis par des graphes distribués [15] mais uniquement pour les aspects de distribution dynamique alors que nous devons tenir également compte des changements de media ou de modalité.

7. CONCLUSION

Nous avons, dans cet article, posé les bases de nos futurs travaux sur la conception des interactions multi-utilisateurs et multimodales pour les applications mobiles pervasives. Ces applications sont, comme nous l'avons vu, très différentes des applications distribuées conventionnelles et changent, en outre, la façon par laquelle les utilisateurs interagissent avec elles. Ces changements rendent possible l'utilisation d'une multiplicité d'interfaces dupliquées selon des modalités différentes. La probabilité importante d'apparition ou de disparition incontrôlée de terminaux mobiles parmi ceux impliqués dans leur fonctionnement rendent obligatoire un contrôle dynamique du cycle de vie des interactions de ces applications. C'est pourquoi nous proposons, dans cet article, l'utilisation de nos architecture et plateforme qui fournissent déjà des fonctionnalités dynamiques aux application mobiles et pervasives. Etant au début de nos travaux dans ce domaine nous aurons à aller beaucoup plus loin sur les aspects de génie logiciel et aurons, notamment, à proposer les règles qui permettront de régir le cycle de vie des interactions ainsi que les cardinalités des associations entre interacteurs et noyau fonctionnel. Il nous faudra, pour ce faire, envisager la création, sur notre plateforme, de métriques appropriées à l'exécution de telles règles afin de fournir les mécanismes d'adaptation des interactions au contexte.

8. REFERENCES

[1] Avouac, P. A., Lalanda, P., & Nigay, L. Service-oriented autonomic multimodal interaction in a pervasive environment. In Proceedings of the 13th international . ACM conference on multimodal interfaces (2011, November),pp. 369-376.

[2] Balme, L., Demeure, A., Barralon, N., Coutaz, J., & Calvary, G. (2004). Cameleon-rt: A software architecture reference model for distributed, migratable, and plastic user interfaces. In Proceedings of Second European Symposium of Artificial Intelligence (November 2004), pp. 291-302

[3] Bouix, E., Dalmau, M., Roose, P., & Luthon, F. (2005, March). A multimedia oriented component model. In Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on (Vol. 1, pp. 3-8). Tamkang University, Taiwan - March 28 - March 30, 2005.

[4] Brusilovsky, P. (2001). Adaptive hypermedia. User modeling and user-adapted interaction, User Modeling and User-Adapted Interaction archive Volume 11 Issue 1-2, 2001 Pages 87 – 110 Kluwer Academic Publishers Hingham, MA, USA.

[5] Calvary, G., Coutaz, J., Dâassi, O., Balme, L., & Demeure, A. (2005). Towards a new generation of widgets for supporting software plasticity: the" comet". In Satellite Proc. of the ACM/IEEE 8th International Conf. on Models Driven Engineering Languages and Systems, MoDELS/UML 2005.

[6] Calvary, G., Coutaz, J., & Thevenin, D. (2001). A unifying reference framework for the development of plastic user interfaces. In IFIP WG2.7 (13.2) Working Conference, EHCI01,Toronto, Springer Verlag Publ., LNCS 2254, M. Reed Little, L. Nigay Eds.

[7] Cohen, P.R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. (1997) QuickSet: Multimodal interaction for distributed applications. In Proceedings of the Fifth ACM International Multimedia Conference ACM, NY, 31–40.

[8] Coutaz, J., & Nigay, L. (1994). Les propriétés CARE dans les interfaces multimodales. In IHM (Vol. 94, pp. 7-14)

[9] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, & D., Stash, N. (2003, August). AHA! The adaptive hypermedia architecture. HYPERTEXT '03 Proceedings of the fourteenth ACM conference on Hypertext and hypermedia Pages 81–84 - ISBN:1-58113-704-4.

[10] Dumas, B., Lalanne, D., & Ingold, R. (2010). Description languages for multimodal interaction: a set of guidelines and its illustration with SMUIML.Journal on multimodal user interfaces, 3(3), 237-247.

[11] Faconti, G., & Paternò, F. (1990). An approach to the formal specification of the components of an interaction. In proceedings of Eurographics (Vol. 90, pp. 481-494), Montreaux.

[12] Lawson, J.-Y. L. , Al-Akkad, A.-A. , Vanderdonckt, J. and B. Macq. An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-the-Shelf Heterogeneous Components. In Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2009).

[13] Louberry C, Dalmau M., Roose P. Kalimucho: Adaptation Platform for Mobile Applications - 11th International IFIP Conference on Distributed Applications and Interoperable Systems (DAIS 2011), June 6th - 9th, 2011, in Reykjavik, Iceland, Springer, LCNS 6723, ISBN 978-3-642-21386-1, pp 43-56.

[14] Markopoulos, P. Interactors: formal architectural models of user interface software. In Kent, A., and Williams, J.G., (Eds.) Encyclopedia of Microcomputers,Volume 27 (Supplement 6), Marcel Dekker, New York, 2001, pp 203-235.

[15] Nigay, L., & Coutaz, J. (1996). Espaces conceptuels pour l'interaction multimédia et multimodale. TSI, spéciale Multimédia et collecticiel, AFCET &HERMES.

[16] Serrano, M., Nigay, L., Lawson, J. Y. L., Ramsay, A., Murray-Smith, R., & Denef, S. (2008, April). The openinterface framework: a tool for multimodal interaction. In CHI'08 extended abstracts on Human factors in computing systems (pp. 3501-3506). ACM.