Vers une architecture d'adaptation automatique des applications reparties basées composants

Makhlouf Derdour[1], Philippe Roose[2]
Marc Dalmau[2], Nacéra Ghoualmi Zine[1], Adel Alti[3]

[1] Univérsité Annaba - Algérie
{m.derdour, ghoualmi}@yahoo.fr
[2] LIUPPA - IUT de Bayonne Pays Basque
{roose, dalmau}@univ-pau.frautre-adresse
http://www.iutbayonne.univ-pau.fr/~roose/V3
[3] Université Sétif
altiadel2002@yahoo.fr

Résumé. Les systèmes informatiques d'aujourd'hui sont de plus en plus pervasifs, composés de composants hétérogènes fournissant des fonctionnalités avec des interactions complexes. Les recherches existantes sur le développement à base de composants ont surtout porté sur la structure des composants, les interfaces et les fonctionnalités de ces derniers. Le domaine de l'architecture logicielle traite, entre autres, de l'importance significative des interactions des composants, y compris la notion de connecteurs logiciels. Si les travaux sur l'assemblage des composants ne manquent pas, peu d'approches considèrent l'hétérogénéité des interactions en matière de types et de formats des données manipulés permettant ainsi d'assurer la compatibilité technique et sémantique des échanges. Dans ce papier, nous proposons une architecture basée sur les données de type multimédia pour l'adaptation des composants hétérogènes. Nous proposons dans un premier temps un typage des interactions des composants afin de pouvoir présenter les déférents formats de média (image, texte, son, vidéo). Nous développons ensuite un service d'adaptation permettant de détecter et de résoudre le problème de l'hétérogénéité entre les composants incompatibles. Nous proposons de voir l'adaptation comme une propriété non fonctionnelle assurée par un connecteur appelé « connecteur d'adaptation ».
Mots clés : Architecture Logicielle, Connecteur, Service, Multimédia, Hétérogénéité.

1 Introduction

Depuis plusieurs années, le multimédia s'invite dans la plupart des applications. Depuis peu s'ajoute à ces applications la mobilité des utilisateurs. Cette évolution n'est pas neutre. En effet, elle implique de concevoir différemment les applications afin de prendre en compte l'ubiquité, ce qui pose de nombreux problèmes puisque cette mobilité des utilisateurs s'accompagne d'un accroissement des types de terminaux mobiles (*ou MID – Mobile Internet Device*) et donc d'une hétérogénéité matérielle importante.

L'implémentation d'applications sur de tels MID est fréquemment réalisée à base de composants et/ou services permettant aux applications de s'adapter aux évolutions du con-

texte (*utilisateurs, matériel, etc.*). Malheureusement, la plupart des ces adaptations sont essentiellement basées sur l'aspect fonctionnel et ne prennent pas en compte l'hétérogénéité des données (*de type : image, son, texte, vidéo et de format d'encodage pour chaque type*). Prenons deux exemples simples :

On dispose sur un PC d'un composant de capture de photo au format JPG et d'un composant de transmission. On dispose sur un PDA d'un composant de restitution d'images au format PNG. L'assemblage fonctionnel entre ces trois composants est parfaitement valable, néanmoins il ne peut fonctionner pour cause d'incompatibilité de format. Il est nécessaire de réaliser une adaptation soit à la source, soit à la destination (voire même sur un hôte jouant le rôle de proxy).

Le deuxième exemple pose est celui d'un composant d'acquisition qui réalise des captures en 1024x768, format incompatible avec la taille de l'écran du PDA, capable de n'afficher que du 640x480, ceci se traduira par un affichage tronqué. Fonctionnellement, l'assemblage est cohérent même au niveau du format d'encodage, ce qui se pose là est un problème de qualité de service, le service rendu par l'affichage n'est pas celui souhaité. Il faut donc pouvoir résoudre, au-delà du format d'encodage, le problème de la donnée et de sa pertinence

Dans le cas d'applications en environnement pervasifs, ce telles adaptations sont complexe à anticiper puisque par nature elles dépendent d'une part de l'assemblage en cours de déploiement mais également de l'hôte sur lequel les données sont exploitées. Nous souhaitons proposer une solution qui permette de détecter de façons automatique les points d'hétérogénéités entre les composants d'une application, et de rechercher les adaptateurs nécessaires pour réaliser les transformations afin de pouvoir exploiter les données. Ces adaptateurs devront à la fois permettre un ajustement de format des données (*interopérabilité technique*) et également sur leur type (*interopérabilité sémantique*). Nous centrons nos travaux sur les applications réalisées à base de composants logiciels et proposons une solution à base de connecteurs assurant ces deux types d'adaptations.

Dans cet article nous présentons brièvement la conception d'applications à base de composants et de services, puis nous présentons un modèle de composant et de connecteur (*vue en tant que composant de première classe*) typant l'interface particulièrement adapté aux données orientées multimédia (image, son, texte, vidéo). Nous présentons ensuite une modélisation d'architecture logicielle permettant la découverte de l'hétérogénéité, la recherche du service d'adaptation approprié, et leur intégration pour construire le connecteur d'adaptation.

2 Conception des applications basées composants

Le développement d'applications pour l'informatique pervasive présente un certain nombre de défis pour l'ingénierie logicielle, dont en particulier, l'adaptation des applications sensibles au contexte : adaptation à l'environnement (*localisation, temps, condition, etc.*), à la connectivité (*débit, protocole, etc.*), aux limites de l'appareil (écran, son, etc.) et même aux utilisateurs (*handicap physique, choix personnels, etc.*). La prise en compte de l'ensemble de ces facteurs liés au contexte est une tâche complexe pour le programmeur.

Parmi les architectures logicielles pour applications pervasives on trouve les architectures logicielles à base de composants, qui présentent l'intérêt de permettre le raisonnement sur des systèmes logiciels complexes à un niveau abstrait, c'est-à-dire en faisant abstraction des détails de conception et d'implantation. Les propriétés fonctionnelles et/ou non fonction-

nelles peuvent concerner des composants assemblés dans les architectures ou bien les assemblages eux-mêmes. Elles couvrent aussi bien les aspects, dynamiques que structurels des applications. Nous nous plaçons à un niveau plus haut, dans lequel l'architecture est une description abstraite et modulaire du système. A ce niveau, l'architecture est perçue comme une collection de composants (*au sens d'entités logicielles*), une collection de connecteurs (*pour décrire les interactions entre composants*) et des configurations, c'est-à-dire des assemblages de composants et de connecteurs. L'accent est mis sur la modularité, qui est un critère important pour la réutilisabilité, la fiabilité et l'évolutivité du logiciel.

Des ADLs (*Architecture Description Language*) sont utilisé pour spécifier ces architectures. [MED 00] a essayé de présenter la différence entre un ADL et une spécification formelle. [CLE 96] et [BAR 05] a présenté un aperçu sur les ADL. De même, les types de connecteurs ont été étudiés par [MEH 00] qui a présenté une taxonomie permettant la prise en charge des propriétés non-fonctionnelles par des connecteurs de plusieurs catégories (communication, sécurité, conversion, facilitation, coordination, interaction). Les modèles couvrent tout ou partie des besoins en termes de langage, de sémantiques et d'outils. Dans [MEH 00] les auteurs relèvent des insuffisances pour la spécification des propriétés non-fonctionnelles des systèmes, un manque de fondement sémantique pour l'expression des contraintes et du raffinement (*composant, connecteur et configuration*), un manque d'outils pour la reconfiguration et l'évolution, et un manque dans les spécifications (manifestes) des éléments de l'architecture afin de détecter la non interopérabilité comportementale (sémantique) entre ces éléments, cette non interopérabilité est la conséquence de l'utilisation des données multimédia, avec plusieurs type (image, son, vidéo, texte) et plusieurs format (exemple pour l'image : BMP, JPEG, PNG, etc.).

L'adaptation qui représente l'objet de notre travail est l'une des propriétés non fonctionnelles à développer, et que doivent assurer les connecteurs de composants opérationnels. Ce choix est justifié du fait que l'adaptation est un problème générique qui n'est pas lié à un composant spécifique.

3 Modélisation des éléments de notre architecture

Dans l'objectif de répondre à des problèmes récurrents en informatique tels que la réutilisation, la maintenance, l'évolution et l'intégration des systèmes, l'ingénierie logicielle n'a cessé de produire de nouvelles approches et de nouveaux paradigmes. Ainsi, l'ingénierie logicielle à base de composants se positionne depuis quelques temps comme une approche prometteuse pour faire face à ces types de problèmes. Ces différentes approches visent une meilleure réutilisation des modèles à base de composants. En revanche la complexité inhérente aux besoins croissants des utilisateurs et à la maintenabilité des systèmes reste un champ d'investigation pour les chercheurs. Le problème d'incompatibilité sémantique entre composants est lié beaucoup plus à l'utilisation des données multimédia, ces données avec leurs diversités de formats et de types engendre pas mal de problème lors de la communication et la coopération des composant. Face à ce problème, nous avons obligé de travailler sur trois dimensions : (I) les flux multimédia afin de spécifier les relations sémantiques et structurelles entre les différents types, et citer les différents passages possibles entre les types et formats de données (II) les composants afin d'avoir une spécification adéquate au besoin de détection d'hétérogénéité au niveau d'assemblage ou de configuration (III) les connecteurs d'adaptation afin de métriser les différents types d'hétérogénéités.

3.1 Flux multimédia

Dans les environnements hétérogènes, les appareils peuvent demander tout type de contenu allant d'un contenu textuel pauvre à des documents multimédias complexes et riches. Assurer une livraison adaptée à toute la communauté des utilisateurs, exige des techniques efficaces qui prennent en considération les médias et la structuration des flux. C'est pourquoi une modélisation des médias manipulés est nécessaire à l'adaptation de contenus et au passage d'un type de média à un autre. Pour représenter la structure hiérarchique des médias, nous avons utilisé un diagramme UML (cf. figure 1). Les médias peuvent être classés en deux catégories : les médias continus qui sont caractérisé par la durée de présentation tels que le son et la vidéo et les médias discrets tels que les images et le texte. Il existe autre type de lien entre média, tel que les liens de transformation/Conversion de type (jpeg-->png) et de format (son-->texte). La notion du temps même s'il n'a pas de sens pour les médias tels que le texte et l'image, elle est très importante pour les flux, puisque un flux contient plusieurs médias reliés par des relations structurelles, temporelles et/ou logiques.

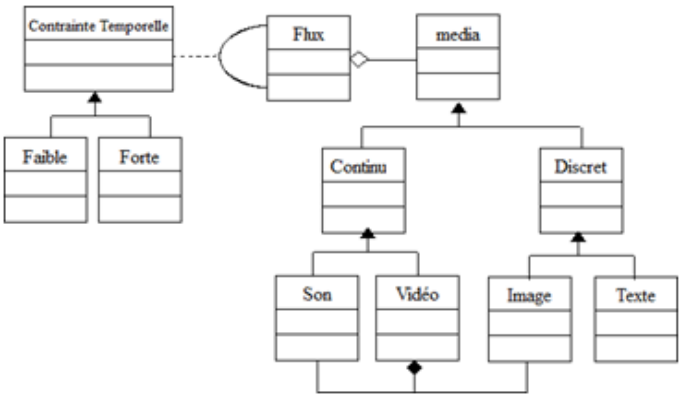

FIG.1- *Modèle de flux multimédia*

L'adaptation est un traitement (cf. Tab 1) permettant de modifier le type de média (transmodage), le format d'encodage et/ou le contenu du média (transcodage).

Catégorie	Texte	Image	Vidéo	Audio
Transcodage	- conversion de format - réduction de la taille de police - changement de police - couleur ...etc.	- conversion de format - réduction de la taille - réduction des dimensions -réduction de couleur-profondeur - image en niveau de gris	- conversion de format - réduction du taux de défilement - réduction de la résolution spatiale - réduction résolution temporelle - réduction de la profondeur de la couleur	- conversion de format -changement d'échantillonnage
Transmodage	- transformation texte-à-audio - transformation texte à image	- Image au texte	- transformation de vidéo-à-image - transformation de vidéo-à-texte - transformation vidéo-à-audio	- transformation audio à texte

TAB. 1 – ADAPTATIONS DES MEDIA

3.2 Composant & service

Plusieurs langages de description de composants ont été proposés, certains sont destinés à des implémentations spécifiques, telles que COM, EJB, CCM et .NET tandis que d'autres se veulent plus universels comme Fractal et UML. Le problème de ces descripteurs est la dépendance aux langages de programmation et/ou aux plateformes d'exécution, spécialisation, rigidité, ainsi que compatibilité. La plupart des langages proposés utilisent comme moyens de représentation le texte et/ou le graphique et/ou les arbres syntaxiques.

Parmi les modèles de composants qui existent, on trouve dans [GRA 04] des modèles en UML qui ont été élaborés pour décrire les différentes facettes de composants logiciels dans les deux middlewares EJB et CORBA. [HAI 06] propose un modèle de composants basé sur le concept Vue/Point de vue. Cette vision limite les choix d'adaptation aux points de vue proposés lors de la conception et ne permet pas d'avoir d'autre point de vue lors de l'exécution. Dans le modèle Kmelia, un composant se caractérise par un nom (l'identificateur du composant), un état (variables et prédicat invariant sur eux) et une interface pour les services et la description des services. L'interface spécifie les interactions entre les composants avec leur environnement. [ATT 06 & 09] [AND 07& 08] proposent des améliorations sur les interfaces d'un composant Kmelia afin de présenter les services fournis et les services requis. Des travaux en relation à ce travail, tel que [PAV 05] [PLA 02] [SUD 05] proposent l'association de comportements dynamiques (ou protocoles) aux composants et aux services. Ceci complique encore la tâche de maintenabilité et engendre des cas d'incohérence difficile à gérer dans les composants modifiés.

Le modèle de flux multimédia appelé « Korrontea » [BOU 04] propose une spécification basée sur des graphes fonctionnels qui permet d'exprimer une architecture abstraite en termes de rôle et de flux de données échangés. Ce modèle intègre les préoccupations non fonctionnelles du transport avec conservation de la synchronisation qui permet à une plateforme de superviser et de gérer la qualité de service requise et fournit. Un modèle de composants multimédia « Osagaia » a été proposé par [BOU 05] sur la base de « Korrontea ». Osagaia permet une implémentation de l'architecture logicielle. La plateforme propose des reconfigurations au niveau des graphes fonctionnels afin de prendre en considération le contexte d'utilisation. Les adaptations proposées sont assurées par des composants logiciels.

La diversité des médias manipulés par les composants (*image, son, vidéo, texte*), leurs types (*ex pour les images : PNG, JPG, GIF*) et leurs propriétés (*résolution, nb de couleur, etc.*) doivent être pris en compte lors de la conception, ce qui rend complique la tâche (il faudrait prévoir à la conception toutes les adaptations possibles selon les périphériques et les déploiements possibles).

Ce qui nous intéresse dans cet article est de permettre la caractérisation des composants du point de vue de l'adaptation au moment de l'assemblage ou à l'exécution. Nous en donnons donc une définition minimale basée essentiellement sur deux points clés. Le premier fait référence à l'utilisation d'une forme de typage d'interface, adéquate à nos besoins en termes de définitions d'un modèle de composant permettant la détection des conflits et des points d'incompatibilité comportementale entre composants fonctionnellement compatibles. Dans un souci de clarté et pour éviter toute confusion avec l'approche classique du typage, nous désignerons cette notion de « type » plus large par le terme « média ». L'autre élément central de notre modèle relève de la description explicite des modifications souhaitées dans le cadre d'adaptations : les schémas d'adaptation permettant de rechercher, construire et intégrer les services d'adaptation dans un connecteur. On dit que deux composants ont une

hétérogénéité comportementale, s'ils ont des interfaces de flux (entrant/sortant) incompatibles.

Nous proposons la modélisation des ports d'E/S de média par des formes géométriques (cf. tableau 1), chaque média sera représenté par une forme, et chaque type par une couleur. Cette modélisation permet d'identifier visuellement les points d'hétérogénéité par type de média et de mettre en évidence une nécessité de recherche de connecteurs d'adaptation.

Type	Sortie Entrée	Format		
Texte		DOC	DOCX	ODT
Image		JPEG	BMP	PNG
Son		WAVE	RM	MP3
Vidéo		MPEG	AVI	MP4

TAB. 2 – *Port d'interface multimédia*

Le problème d'adaptation n'est pas un problème de protocole de communication, mais aussi un problème du langage de communication entre composants (flux continu/scalaire, média, format, propriétés, caractéristiques). La modélisation que l'on souhaite pour le composant ne doit pas seulement concerner les interfaces du composant, mais aussi le détail des services qu'il propose et ses interfaces en matière de flux multimédia tout en détaillant les caractéristiques de chaque flux entrant/sortant (propriétés, caractéristiques, etc.).

La figure 2 montre qu'un service est proposé pour satisfaire un besoin: il peut être proposé par un composant (besoin fonctionnel), un connecteur (besoin non fonctionnel) ou comme un service d'adaptation (besoin d'interaction). Un connecteur d'adaptation est composé d'un ensemble de services d'adaptation et peut être attaché à d'autres connecteurs. Un connecteur d'adaptation est un connecteur spécifique entre deux composants hétérogènes.

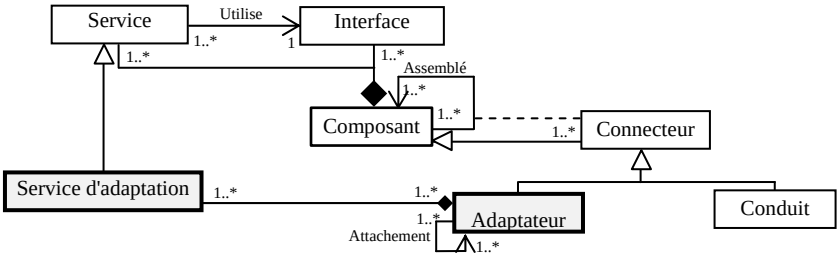

FIG. 2**:** *Relations entre acteurs de notre architecture*

Pour la représentation graphique d'un composant (cf. figure 3), nous avons choisi l'utilisation du modèle « Osagaia» [BOU 05]. En effet, le conteneur Osagaia intègre des unités de traitement spécifiques afin de gérer les flux de données et les buffers associés. Ce conteneur offre un certain nombre de commandes de supervision permettant de le connecter/déconnecter/déplacer à volonté et d'en remplacer la partie fonctionnelle (le composant métier). Il offre en outre tout un ensemble d'informations sur son fonctionnement qui permettent de décider d'éventuelles reconfigurations. Enfin, dans sa version actuelle, ce conteneur possède plusieurs implantations lui permettant d'être utilisé sur des périphériques plus

ou moins contraints. Cet ensemble de caractéristiques en font un candidat intéressant pour nos travaux. Néanmoins, le modèle Osagaia ne propose pas de typage des ports d'entrées/sorties pour représenter les divers médias, point qu'il faut impérativement résoudre dans le cadre de nos travaux.

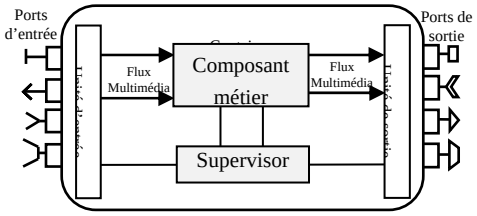

FIG. 3: *Modèle de composant multimédia*

Dans les ADLs, l'application est un assemblage de composants. Pour faire cet assemblage on a besoin d'informations sur les composants, ces informations sont souvent présentées sous forme d'un manifeste qui contient les informations techniques sur le composant.

Dans ce travail, on s'intéresse aux composants multimédia, ce qui nécessite l'ajout de nouvelles contraintes techniques au manifeste. Ainsi nous avons besoin de vérifier non seulement la possibilité d'un assemblage du point de vue fonctionnel mais aussi comportemental (sémantique). Pour ces raisons nous avons décidé d'enrichir les manifestes des composants afin de prendre en considération l'aspect multimédia.

Un manifeste (cf. figure 4) permet de décrire des composants selon un modèle abstrait (*sans détail d'implémentation*). Il permet de séparer la description abstraite de la fonctionnalité offerte par un composant, des détails concrets du composant tels que «comment» et «où» cette fonctionnalité est offerte, et de décrire les données manipulées par les composants (*type de données, format, contrainte temporelle – forte/faible -, etc.*).

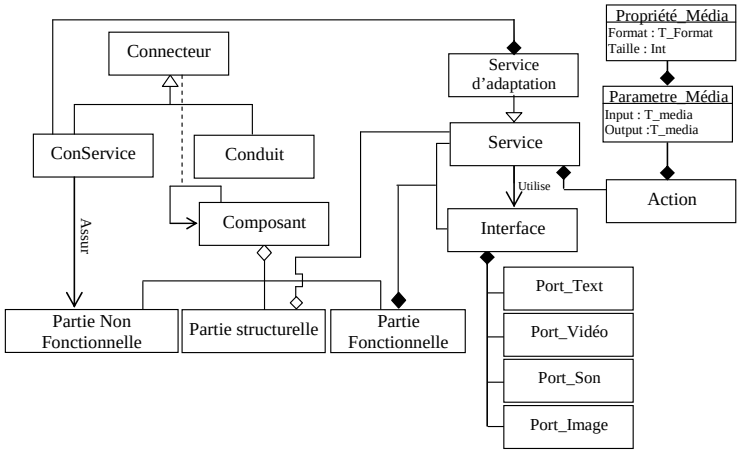

FIG. 4**:** *Manifeste de composant en UML*

3.3 Connecteur

Comparés à ceux des langages de description d'architectures [ALL 97] [MEH 00] [MEH 00] les connecteurs que nous proposons peuvent être simples ou composés et peuvent même assurer des services. Autrement dit, ces connecteurs n'assurent pas uniquement les liens de communications mais également la correspondance entre composants.

Les connecteurs d'adaptation utilisent les mêmes notations présentées dans le tableau 2 pour relier des composants multimédia.

Exemple 1: pour relier deux composants. L'un qui fournit des média sous forme de vidéo et l'autre requiert des média sous forme de son et d'image. On a besoin d'un connecteur d'adaptation pour éclater le fichier vidéo en son et images (cf. figure 5).

FIG. 5: *assemblage de composants hétérogènes*

Si l'on veut remplacer la vidéo par des images contenant du texte, il faut assurer la transformation du son en texte et l'insertion de ce dernier dans les images correspondantes (cf. figure 6).

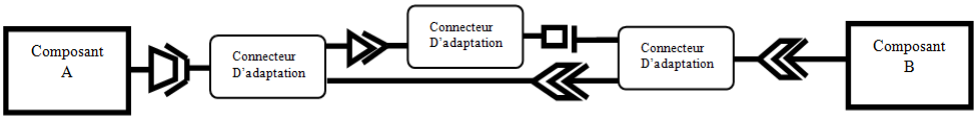

FIG. 6: *Exemple d'une application multimédia*

Exemple 2: Soit un système de surveillance (cf. figure 7) automatique d'une société contenant des caméras de surveillance, un système à base de composants et une base de données contenant les visages des employés de la société. Le système doit déclencher une alarme lors de la détection d'un étrange.

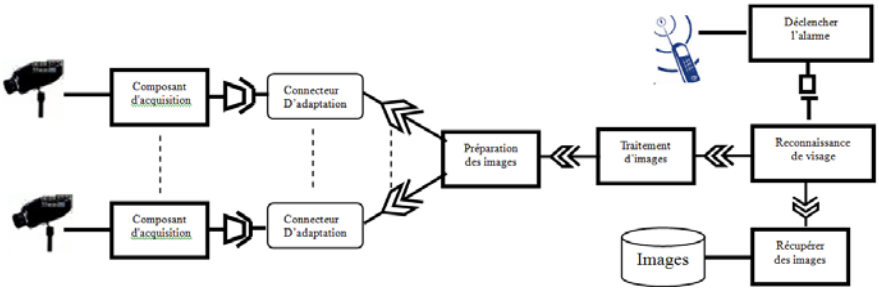

FIG. 7: *Adaptation des composants d'une application multimédia*

Les médias sont transportés au sein de l'application à l'aide d'un connecteur qui s'appelle conduit [BOU 05]. Les conduits sont utilisés pour connecter les processus métier des compo-

sants entre eux. Pour notre approche nous allons utiliser deux types de connecteurs. Le conduit et l'adaptateur. L'adaptateur constitue l'entité d'adaptation dans notre approche c'est-à-dire qu'il est capable de transférer des données multimédia entre les différents composants tout en assurant l'adaptation de ces données par rapport au contexte.

Les composants manipulent et fournissent des médias dans plusieurs formats ayant chacun des propriétés (temporelles, structurelles, sémantiques, etc.) et caractéristiques propres (résolution, fréquence, etc.) ce qui soulève un problème supplémentaire : chaque adaptation doit être associée à une qualité de service. En effet, certaines transformations ont des effets sur les données produites. Ainsi, la conversion d'une image JPG en GIF va diminuer le nombre de couleurs pour le limiter à 256 couleurs. Si dans certains cas cela ne pose aucun problème (cas d'une visioconférence par exemple), ceci peut avoir des effets dramatiques si l'information représente des dégradés utilisés en visualisation scientifique par exemple.

L'adaptation est considérée comme un besoin non fonctionnel d'un composant, cette tâche doit être assurée par un autre service. Le connecteur fournit les aspects non fonctionnels dont le composant a besoin. Le rôle du connecteur d'adaptation (cf. figure 8) est de recevoir les données, de les adapter selon les directives du gestionnaire de QoS et de les acheminer vers le composants/connecteur destinataire. Dans le cas ou le gestionnaire de la QoS du connecteur n'arrive pas à générer la qualité exigée, la plateforme demande du gestionnaire de la QoS globale (hébergé sur la plateforme) de trouver un autre service d'adaptation à partir du graphe d'adaptation [DER 09].

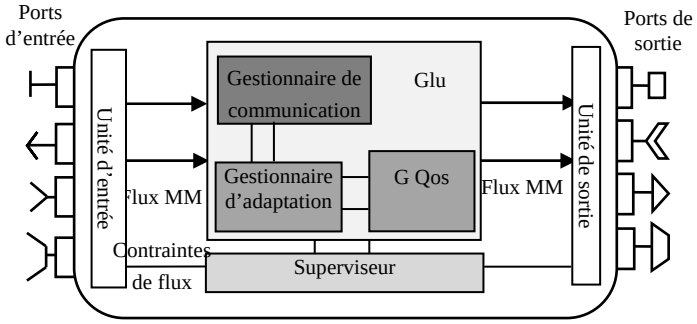

FIG. 8**:** *Modèle de connecteur d'adaptation*

Au niveau des ports de communication, nous avons ajouté des types pour clarifier les différentes liaisons entre composants en matière de flux de données. Au niveau interne nous avons enrichi la glu pour qu'elle réalise l'adaptation. L'adaptation est assurée par un gestionnaire d'adaptation qui coopère avec un gestionnaire de QoS local, permettant ainsi la maitrise de la qualité des flux de données par rapport aux contraintes de l'environnement.

4 Modélisation de l'architecture d'adaptation

Des approches [ALL 97] [MAX 05] [ATT 09] permettant la séparation des préoccupations fonctionnelles ont été proposées dans le but de capitaliser les besoins fonctionnels dans des entités modulaires. Dans cette perspective plusieurs idées ont été proposées. On distingue principalement deux catégories d'approches pour les architectures logicielles: celles inspirées

du développement de logiciels à base de composants (CBSE) et celles orientées service (SOA). Dans le premier cas [SZY 97] [ALL 97] [BER 00] l'accent est mis sur la structure statique du système: les éléments logiciels sont des composants assemblés par des connecteurs dans des configurations. Dans le second cas [PAP 03] [MAX 05] [AND 09] [ASR 09] l'accent est mis sur la structure fonctionnelle du système : les éléments logiciels sont des fonctionnalités (des services) liés par des relations de type collaboration ou composition. Des travaux de recherche proposent des SOA pour l'adaptation multimédia dans des environnements ubiquitaires, comme DCAF [BER 05], PAAM [ZAK 06] et [ROM 06], ces travaux traitent le problème d'adaptations au niveau applicatif, et selon un processus sensible au contexte d'exécution. Notre travail vise a proposé des mécanismes d'adaptation à un niveau plus abstrait, ce qui permet de régler le problème d'hétérogénéité au niveau des configurations et des assemblages de composants.

L'architecture que nous proposons est composée de deux modules (cf. figure 9): le module d'adaptation présenté dans [DER 09] qui fournit les services d'adaptation nécessaires à l'assemblage de composants hétérogènes, et le module de configuration des applications à base de composants. Nous mettons l'accent ici sur le module de configuration.

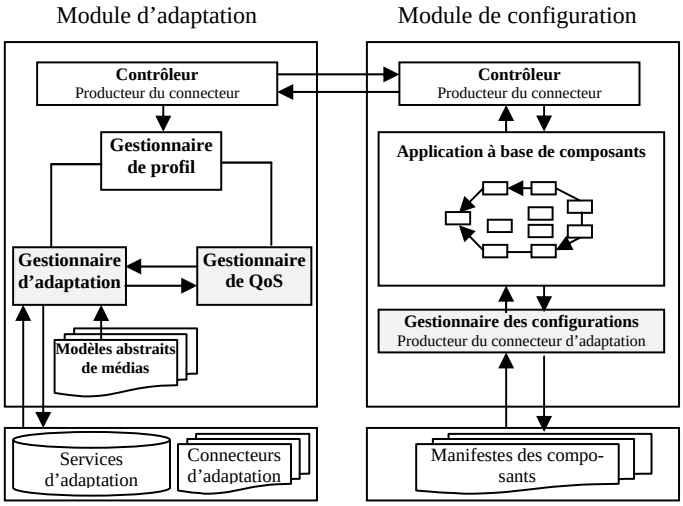

FIG. 9: *Architecture d'adaptation des composants*

Le module de configuration est composé d'un contrôleur qui joue le rôle d'un superviseur de l'environnement de configuration et des assemblages de composants. Ce contrôleur transmet les besoins en adaptation au module d'adaptation afin de récupérer les adaptations demandées. Le gestionnaire des configurations assure la configuration de l'application basée composants et, éventuellement, les reconfigurations et réassemblages des composants. Dans ce qui suit, nous donnons de plus amples détails sur le module de configuration.

L'objectif de notre architecture est de fournir des mécanismes permettant l'adaptation des flux de données multimédia (*une adaptation est l'ajout d'un ou plusieurs traitements permettant de modifier le format d'encodage et/ou de modifier le contenu du média pour l'adapter au composants destinataire*) dans des applications à base de composants hétérogènes (*on parle des composants fonctionnellement compatibles et sémantiquement incompatibles par ce lorsqu'ils fournissent/requièrent des flux de données de différents formats et types*). Il

s'agit donc d'assurer l'assemblage de ces composants. Pour cela, l'architecture doit être capable de vérifier la possibilité d'assemblage (*au sens de compatibilité des données/formats au niveau des interfaces/ports*) des composants à partir des manifestes (*un manifeste doit contenir les informations techniques d'un composant - cf.section 3.2*). L'assemblage peut ne pas fonctionner pour deux raisons : incompatibilité fonctionnelle et/ou comportementale (*les interfaces de flux entrant/sortant sont incompatibles, les composants requièrent/fournissent différents types et/ou formats de données : Son, Vidéo, Texte, Image*). Par exemple lorsqu'un composant fournit des images de type PNG, alors que l'autre n'accepte que des images JPEG. L'adaptation du flux et alors nécessaire pour qu'ils soient compatibles.

Afin de modéliser l'aspect fonctionnel de l'architecture d'assemblage (module de configuration), nous avons utilisé un diagramme de cas d'utilisation qui nous permet de citer les fonctions utilisés par chaque acteur dans le système (cf. figure 10). Nous avons pris la définition d'un composant proposé par [ATT 06] qui a défini un composant comme un ensemble de services interagissent entre eux afin de réaliser l'objectif du composant. Ainsi, un composant est une agrégation d'un ou plusieurs services. Sachant qu'un connecteur est une composition de services d'adaptation, il est possible de construire un ou plusieurs connecteurs à partir d'un ou plusieurs services d'adaptation.

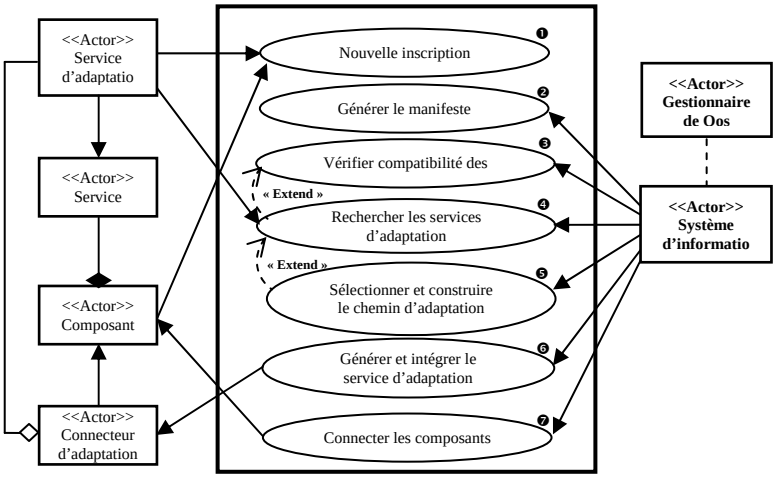

FIG. 10: *Diagramme fonctionnel d'une architecture basée composants multimédia*

❶&❷ Un service est proposé par un composant ou un connecteur d'adaptation. Le composant est accompagné d'une description (manifeste), qui doit fournit toutes les informations nécessaires à son intégration. Par exemple pour l'intégrer dans un assemblage nous avons besoin de sa spécification fonctionnelle (*services et rôle*), comportementale (*formats et types de données d'E/S)*, et de ses restrictions techniques.

❸A partir d'un manifeste, le système doit être capable d'évaluer les assemblages fonctionnellement possibles entre composants, et de détecter les points d'incompatibilité comportementale entre ces composants.

❹ Après la détection d'une incompatibilité comportementale, le système cherche les services d'adaptation qui existent, et construit le graphe des adaptations possibles.
❺ A partir d'un ensemble de critères qui gèrent la qualité de service, le service d'adaptation choisit le meilleur schéma d'adaptation [DER 09], qui sera par la suite transféré au module de configuration.
❻ Après l'assemblage des services nécessaires à l'adaptation des composants. Deux solutions sont envisageables:

- L'adaptation peut être vue de deux manières différentes fonctionnelle et non fonctionnelle. Si elle est vue comme une propriété fonctionnelle, cela implique l'intégration des services d'adaptation dans le composant. Cette solution est difficile à implanter dans le cas de composants hébergés sur périphériques hétérogènes. En effet, lors de l'adaptation de l'application afin de permettre de respecter une certaine QoS, soit parce que l'utilisateur modifie ses choix, soit parce qu'il souhaite poursuivre son application sur une autre hôte (PC -> PDA par exemple), les composants logiciels peuvent être migrés sur des périphériques plus ou moins limités et sur lesquelles l'adaptation telle qu'implémentée dans le composant n'est pas possible (pour cause de puissance limité par exemple). De plus, les adaptations ne sont pas figées et dépendent de la configuration (et donc du déploiement) en cours.
- L'adaptation est donc considérée comme une propriété non fonctionnelle, ce qui implique l'intégration des services d'adaptation dans les connecteurs de composants. Ceci, rend possible les changements éventuelles lors de l'exécution de l'application (adaptation dynamique et en temps réel), et conserve la spécification abstraite du composant.

❼ Enfin, les composants pourront être assemblés en utilisant les connecteurs d'adaptation.

5 Conclusion

Notre travail porte sur la configuration des architectures à base de composants, afin de permettre la vérification de l'une des propriétés non fonctionnelles des composants, celle de l'adaptation des données échangées, et la fourniture d'adaptateurs pour les composants hétérogènes. Dans cet article, nous avons défini comment intégrer l'architecture logicielle de génération dynamique d'adaptateurs [DER 09] dans les architectures à base de composants. Les services d'adaptation composés par cette architecture permettent aux composants qui n'ont pas été spécialement conçus pour communiquer entre eux d'établir des interactions.

Nous avons montré comment cette architecture fournit l'adaptation adéquate et comment elle pouvait être intégrée dans un connecteur. Un typage de données multimédia a été présenté afin de permettre une représentation graphique riche et significative des composants multimédia, ainsi que des connecteurs et des services.

Références

[ALL 97] Allen, R., Garlan D., (1997). « A Formal Basis for Architectural Connection », ACM Transactions on Software Engineering and Methodology, vol. 6, no 3, 1997, p. 213–249.

[AND 07] André, P., Ardourel, G., and C.Attiogbé, (2007). Defining Component Protocols with Service Composition: Illustration with the Kmelia Model. 6th International Symposium on Software Composition, SC'07, volume 4829 of LNCS. Springer.

[AND 08] André, P., Ardourel, G., and Attiogbé, C (2008). Composing Components with Shared Services in the Kmelia Model. 7th International Symposium on Software Composition, SC'08, volume 4954 of LNCS. Springer.

[ATT 06] Attiogbé, C, André, P., and Ardourel, G. (2006). Spécification d'architectures logicielles en Kmelia : hiérarchie de connexion et composition. In 1ère Conférence Francophone sur les Architectures Logicielles, pp 101–118. Hermès, Lavoisier.

[ATT 09] Attiogbé, C. André, P. and Messabihi, M (2009). Correction d'assemblages de composants impliquant des interfaces paramétrées. 3ième Conférence Francophone sur les Architectures Logicielles. Hermès, Lavoisier.

[BAR 05] Barais O (2005)- Construire et maîtriser l'évolution d'une architecture logicielle à base de composants - PhD thesis, LIFL, Université Lille 1, novembre.

[BER 05] Berhe, G.Brunie, L.Pierson, (2005). Distributed content adaptation for pervasive systems, Information Technology: Coding and Computing, ITCC. PP 234- 241 Vol. 2

[BOU 04] Bouix Emmanuel, Philippe Roose, Marc Dalmau (2004). OSAGAIA : un modèle de composants pour les traitements de flux synchrones. Workshop OCM-SI- Inforsid - Biarritz, France.

[BOU 05] Bouix Emmanuel, Dalmau Marc, Roose Philippe, Luthon Franck (2005) - A Multimedia Oriented Component Model - AINA 2005 - The IEEE 19th International Conference on Advanced Information Networking and Applications - Tamkang University, Taiwan.

[CLE 96] Clements P. C (1996). A Survey of Architecture Description Languages. IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design, Washington, DC, USA, IEEE Computer Society, page 16.

[DER 09] Derdour Makhlouf, Nacira Ghoualmi-Zine, Philippe Roose, Marc Dalmau (2009). Toward a dynamic system for the adaptation multimedia fluxes in the P2P architectures. The Fifth International Symposium (FINA) helded in the IEEE 23rd International Conference onAdvanced Information Networking and Applications (AINA-09), 2009.

[ASR 09] El Asri B., Kenzi A., Nassar M.et Kriouile A. (2009). Vers une architecture MVSOA pour la mise en œuvre des composants multivue. 3e Conférence francophone sur les Architectures Logicielles, pp 1-17. RNTI.

[HAI 06] Hair Abdellatif (2006). Towards a multi-visiens software components modéls. Revue INIST-CNRS, RIST Vol 16 N°02.

[GRA 03] M. Graiet, M.T. Bhiri, J-P. Giraudin., (2003). Modélisation en UML des composants logiciels et non logiciels. Dans les Actes du Viieme congrès ALCAA 2003, Bayonne 4-5 septembre 2003.

[MAX 05] E. M. Maximilien and M. P. Singh. Self-Adjusting Trust and Selection for Web Services. In Proceedings of International Conference on Autonomic Computing (ICAC), pages 385–386, Seattle, WA, 2005. IEEE Computer Society.

[MED 00] Medvidovic N., Taylor R. N (2000). A Classification and Comparison Framework for Software Architecture Description Languages - IEEE Transactions on Software Engineering, vol. 26, no 1, p. 70–93.

[MEH 00] Mehtan N. Medvidovic N., R., Phadke S (2000). Towards a taxonomy of software connectors. ICSE '00: Proceedings of the 22nd international conference on Software engineering, ACM Press, p. 178–187. (b)

[PAV 05] Pavel S., Noyé J., Poizat P., and Royer J.C. (2005). A Java Implementation of a Component Model with Explicit Symbolic Protocols. Proceedings of the 4th International Workshop on Software Composition (SC'05), volume 3628 of LNCS.

[PAP 03] Papazoglou M. P (2003). Service-Oriented Computing: Concepts, Characteristics and Directions. WISE, IEEE Computer Society, p. 3–12.

[PLA 02] Plasil F. and S. Visnovsky (2002) Behavior Protocols for Software Components - IEEE Transactions on SW Engineering, IEEE Transactions on Software Engineering, volume 28, pp. 1056 – 1076. 2002.

[SUD 05] Sudholt M. (2005). A Model of Components with Non-regular Protocols - Software composition. International workshop No4, Edinburgh, UK, vol. 3628, pp.99-113.

[SZY 97] Szyperski C. (1997). Component Software: Beyond Object-Oriented Programming – AddisonWesley Publishing Company.

[ROM 06] D. Roman, J. de Bruijn, A. Mocan, H. Lausen, J. Domingue, C. Bussler, and D. Fensel. (2006). Www: Wsmo, wsml, and wsmx in a nutshell. In ASWC, pages 516–522.

[ZAK 06] Zakia Kazi-Aoul, Isabelle Demeure et Jean Claude Moissinac. (2006). "Towards a Peer-to-peer Architecture for the provision of Adaptable Multimedia Composed Documents". DFMA (Distributed Frame for Multimedia Applications), Penang, Malaysia.

Summary

The computing systems of today are more and more pervasive, composed of heterogeneous components providing of the functionalities with complex interactions. Existing research on the component-based development focused on the structure of components, the interfaces and the functionalities of the latter. The field of software architecture milked, among others, the significance of the interactions of components, including the concept of software connectors. If work on the assembly of the components are not lacking, few approaches consider the heterogeneity of interactions in the data types and formats, thus allowing to ensured technical compatibility and semantic data exchange. In this paper, we propose an architecture based data of multimedia type to adapt it to the heterogeneous components. We propose in the first time a typing of the interactions of components in order to present the deferential media formats (image, text, sound, and video). We then develop an adaptation service to solve the problem of heterogeneity between incompatible components. We propose to see

adaptation as a non-functional property secured by a connector called "adaptation connector".