KALIMUCHO : Architecture logicielle pour périphériques mobiles contraints

Cyril Cassagnes, Philippe Roose, Marc Dalmau, Christine Louberry

KALIMUCHO : Architecture logicielle pour périphériques mobiles contraints

Cyril Cassagnes, Philippe Roose, Marc Dalmau, Christine Louberry

LIUPPA/IUT de Bayonne
2, Allée du Parc Montaury
64600 Anglet
Prenom.Nom@iutbayonne.univ-pau.fr

---

1. Introduction

La mise en place de méthodes de conception et de modèles d'architectures logicielles pour concevoir, développer et déployer des applications sur des supports mobiles et/ou à capacité réduite est un enjeu du fait de l'implantation massive de ces équipements dans notre quotidien (lieu de travail, transports en commun, voitures, électroménager). Dans ce domaine en pleine effervescence, de tels outils existent mais sont encore trop liés aux technologies sous-jacentes. La diversité des matériels visés induit une contrainte importante pour notre modèle car l'implantation d'OSGi n'est pas possible sur les machines virtuelles JAVA n'offrant pas la classe *java.lang.ClassLoader*. des machines virtuelles n'implantant pas le chargeur de classes, nous instancierons les composants par une entité Isolate (JSR121). En termes de dynamisme, les capacités sont, certes, plus restreintes mais il reste possible d'arrêter, démarrer et configurer des composants en cours d'exécution.

2. Composants à services

Les modèles de composants classique séparent les préoccupations fonctionnelles et non-fonctionnelles mais ne fournissent pas d'outil pour supporter le dynamisme. L'approche à service a donné naissance à un nouveau modèle de composants appelé modèle de composants à service. Dans notre modèle de composants, les propriétés fonctionnelles forment le cœur applicatif. Les propriétés non-fonctionnelles concernent les mécanismes qui permettent à l'architecture d'être dynamiquement reconfigurée et supervisée par la plate-forme. La plate-forme gère la QoS de l'application au niveau utilisateur, matériel, logiciel et réseau et son évaluation est la source des reconfigurations.

4. La plate-forme Kalimucho

L'objectif de la plate-forme distribuée, Kalimucho, est d'une part de superviser les différents composants (Osagaia, Korrontea) et d'autre part de reconfigurer l'architecture de l'application : soit en agissant sur le cycle de vie des composants, soit en les déplaçant (migration), soit en modifiant leurs connexions d'entrée et/ou de sortie. Une reconfiguration a lieu lorsque la QdS optimale n'est plus assurée. Dans la section suivante nous présentons l'architecture de la plate-forme Kalimucho et les fonctionnalités qu'elle offre. Le deuxième enjeu est de déterminer les services que notre plate-forme doit proposer et comment ils peuvent être répartis sur les différents types d'hôtes en fonction des ressources dont ils disposent. En effet, tous les services ne peuvent être déployés de la même manière sur l'ensemble des périphériques. De ce fait, notre plate-forme ne doit pas avoir une architecture figée, c'est pourquoi l'approche à services constitue un atout. Kalimucho doit répondre aux besoins suivant :

1. Créer des conteneurs Osagaia qui encapsulent un composant métier de l'application (fabrique),
2. Créer des connecteurs Korrontea qui établissent un canal de communication entre deux composants (fabrique).
3. Capter les événements de QoS émis par les composants et les connecteurs et déterminer s'il faut reconfigurer.
4. Générer un schéma de déploiement initial. Puis, si nécessaire, générer de nouvelles configurations de l'architecture par ajout, suppression, substitution, déplacement de composants et des liens de communications.
5. Assurer la cohérence entre les services répartis sur les hôtes (découverte, unicité, ajout, suppression, etc.).
6. Gérer la mobilité du code métier.

Les services Usine à Conteneur et Usine à Connecteur répondent aux besoins 1 et 2. Quelle que soit l'usine employée, l'entité créée est adaptée à son environnement de déploiement. Les services de Supervision et de Génération de Reconfiguration répondent aux besoins 3 et 4 : Suite à la capture d'un événement, le service de Supervision procède à l'évaluation de la configuration en cours. Ensuite c'est au service de Génération de Reconfiguration que revient le travail de générer une nouvelle configuration. Enfin, pour répondre au besoin 5, nous mettons en place le service de Registre des Composants qui connaît et tient à jour la liste des composants disponibles pour les périphériques mobiles et/ou contraints qui ne bénéficient pas des apports d'OSGi. Ainsi *Kalimucho* dispose d'une vision globale des services de l'application. Les services que nous venons de définir permettent de piloter une application à base de composants encapsulés dans les conteneurs que nous avons définie. Cette supervision assure un niveau de qualité de service durable et/ou optimal (selon les choix de l'utilisateur). Chacun de ces services nécessite plus ou moins de ressources. Le *Générateur de Reconfiguration* demandera de nombreux calculs. De ce fait, le choix de ne pas embarquer certains services sur certains types d'hôtes se justifie par la différence des ressources dont disposent ces hôtes. L'objectif est de distribuer au maximum la plate-forme et les tâches qui lui incombent, pour fournir une autonomie maximale à chaque périphérique en fonction de son potentiel (mémoire, énergie, cadence processeur). Nous représentons ici un exemple d'application construite à base de composants ainsi que les liens entre les interfaces de la partie non-fonctionnelle et la plate-forme.

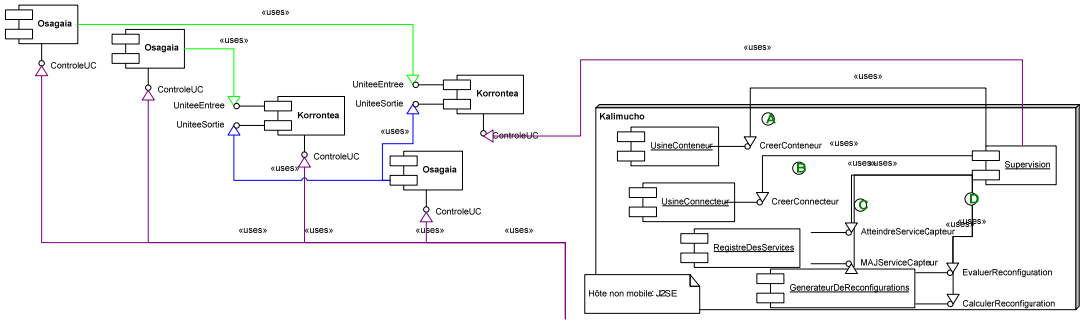

FIG. 1 - Liens plate-forme/application (cas d'un hôte fixe)

Dans le cas d'hôtes mobiles l'implémentation est adaptée aux possibilités offertes par l'API CDC/CLDC. Par exemple, le service Supervision n'a plus exactement le même rôle que sur un hôte non contraint car il n'a plus la possibilité d'accéder au service de *génération de reconfiguration*. Ceci signifie que l'évaluation des configurations devra être demandée à la *Supervision* de l'hôte non contraint le plus proche. En revanche, Les services usines gardent le même rôle mais leur implémentation est adaptée à l'interface de programmation.

5. Perspectives & Conclusion

La conception d'applications ubiquitaires doit prendre en compte l'hétérogénéité des hôtes d'un point de vue matériel (puissance, mobilité variable) mais également d'un point de vue fonctionnel. Afin de permettre la conception d'applications dans ce cadre là, il est nécessaire de disposer de supports d'exécution permettant d'assurer les reconfigurations nécessaires dans un environnement mobile et versatile. C'est ce que nous offrons au travers de notre modèle de composants et de notre plate-forme à services conçue pour être adaptée aux capacités matérielles des hôtes d'un point de vue structurel et fonctionnel. Ces travaux se réalisent dans le cadre d'un accord de coopération avec *Sun MicroSystems (SunSpot)*, du *Conseil Général 64* et font l'objet d'un contrat de transfert de technologie avec la société *Dev 1.0*. Des versions de conteneurs, de composants, de connecteurs et de la plateforme *Kalimucho* ont été réalisées sur PC (*OSGi Felix*), PDA (*iPaq – OSGi Concierge*) et capteurs *SunSpot* (*MV Squawk*).