

Gestion de Personnalités de Services Techniques dans un Modèle à Composants

Colombe Hérault — Sylvain Lecomte

LAMIH / ROI / SID, UMR CNRS 8530

Université de Valenciennes - Le Mont Houy,

59313 Valenciennes Cedex 9 - France

{colombe.herault, sylvain.lecomte}@univ-valenciennes.fr

L'émergence des intergiciels fut en grande partie motivée par le besoin d'interopérabilité dans un contexte fortement distribué entre des modules logiciels, écrits à des époques différentes, avec des langages hétérogènes. Une fois ce problème en partie résolu, on a voulu apporter une certaine qualité de service à ces modules logiciels ainsi qu'une plus grande facilité de développement et de réutilisabilité. L'une des solutions adoptées fut de séparer code fonctionnel et code non-fonctionnel. Ainsi le développeur peut se focaliser sur la logique applicative et déléguer l'implantation des services techniques (appelés aussi services non-fonctionnels) à la plate-forme d'exécution.

Cependant, l'émergence simultanée des outils informatiques personnels portables et des réseaux sans fil a considérablement augmenté le nombre de terminaux connectés aux réseaux. Aujourd'hui, un utilisateur peut être connecté en permanence à des applications distribuées à sa maison ou au travail, en utilisant des outils puissants, mais également des ordinateurs portables comme des assistants personnels ou des téléphones mobiles (qui, bien que connectés au réseau et offrant des capacités de stockage et de traitement, sont bien moins puissants), ou encore d'autres outils informatiques présents à la maison comme les terminaux de télévision numérique, les consoles de jeux qui partagent beaucoup de caractéristiques avec les ordinateurs portables. En conséquence, pour développer des applications portables sur ces terminaux, il devient nécessaire de prendre en compte plusieurs contraintes liées à leur hétérogénéité. Ces applications sont déployées dans des environnements extrêmement distribués, très sujets aux pannes et aux problèmes de localisation d'informations et de services. Enfin, des réseaux pair à pair remplacent de plus en plus souvent les architectures classiques Client/Serveur. Les différents participants de l'application peuvent alors successivement passer du rôle de client à celui de serveur, pour fournir soit des données, soit un service applicatif, ceci même si chacun de ces terminaux a des caractéristiques qui lui sont propres en terme de connexion et de puissance de calcul.

Une solution pour gérer cette hétérogénéité au niveau de l'application est d'utiliser le modèle de programmation par composants. Cependant, les modèles à composants actuels ne permettent pas d'adapter les services techniques à l'environnement d'exécution : le développeur peut uniquement décider d'utiliser ou non un service. On ne lui offre pas le choix du modèle implantant le code non-fonctionnel. Les composants qu'il écrit utilisent donc toujours les mêmes services techniques quel que soit leur environnement. Or, en fonction des spécificités, certaines implantations d'un service sont plus efficaces que d'autres. Nous proposons donc de réaliser également les services techniques à l'aide du modèle à composants et plus précisément le modèle Fractal: un service technique devient alors un assemblage de composants élémentaires, que l'on peut remplacer pour faire évoluer le service. Cela permet l'adaptabilité statique ou dynamique (au moment de l'exécution, en fonction des terminaux utilisés et de ses besoins) des services techniques.

Afin de localiser, choisir, configurer et d'utiliser les personnalités de services techniques (e.g. différentes versions, modèles d'un même service technique), nous définissons des composants de gestion du système : des contrats, un coordinateur du système, un annuaire et des moniteurs. Un *contrat* passé entre des personnalités et une application représente le fait que cette application bénéficie des services de ces personnalités (i.e. que cette application est composée avec les services techniques). De plus, il met à jour cette coopération en trouvant la personnalité la plus adaptée par rapport à l'environnement d'exécution et aux besoins de l'application. Le rôle du *coordinateur* est tout d'abord de gérer les contrats. Il les crée, les prévient lorsqu'ils ont besoin d'une mise à jour et les détruit. Il reçoit les informations concernant

l'environnement grâce aux moniteurs auxquels il a souscrit en définissant ses préférences. Un *annuaire* de personnalités fournit un service de nommage et courtage afin de trouver les personnalités des services techniques. Il permet au contrat de trouver une ou plusieurs personnalités en fonction de l'environnement d'exécution et des besoins de l'application. Pour cela les personnalités doivent s'enregistrer à son service. Un *moniteur* reçoit des informations de son capteur et fait une sélection en fonction des préférences que lui a exprimé le coordinateur. Ainsi le coordinateur n'est pas surchargé par des informations inutiles. Ensuite il envoie les informations pertinentes au coordinateur.

En conclusion, nous proposons d'adapter des services techniques, pour permettre le développement des architectures à composants dans de nombreux domaines tels que le M-commerce où les applications s'exécutent dans les environnements divers (pour l'adaptabilité statique) et très changeants (pour l'auto-adaptabilité). Nous avons développé un premier prototype basé sur l'implantation Julia du modèle de composant Fractal qui montre la faisabilité d'une telle solution. Les bénéfices d'une telle solution sont l'amélioration de la qualité de service et la réutilisabilité du code tout en maintenant les caractéristiques des composants tels que l'interopérabilité et la facilité d'implantation. Des problèmes subsistent, tels que la séparation des services techniques ou la description de la qualité de service fournie par un composant. Notre solution apporte de plus une description haut niveau du comportement du système tels que les moniteurs, le coordinateur, etc. Les règles de fonctionnement du coordinateur restent à formaliser.