Administration de composants logiciels : application aux systèmes sans fil

Fabien Romeo
LIUPPA
Université de Pau et des Pays de l'Adour
F-64000 Pau, France
+33 5 59 40 76 52
fabien.romeo@univ-pau.fr

Franck Barbier
LIUPPA
Université de Pau et des Pays de l'Adour
F-64000 Pau, France
+33 5 59 40 77 43
franck.barbier@franckbarbier.com

RESUME

La grande diversité des dispositifs sans fil amène à concevoir les applications mobiles et ubiquistes comme un assemblage de composants logiciels interchangeables, adaptés à l'environnement de déploiement du logiciel. Afin de pouvoir s'assurer du bon fonctionnement de l'assemblage logiciel et opérer un réel contrôle de l'application en cas de défaillances, la mise en place des concepts, modèles et outils nécessaires à l'administration de ces composants est cruciale. Cet article propose une méthode de gestion du comportement des *composants sans fil*, s'inscrivant dans un système global d'administration d'applications à composants. Cette contribution, qui suit la mouvance de l'ingénierie dirigée par les modèles (MDA/MDE), s'appuie sur une librairie J2ME (Java 2 Micro Edition) permettant l'exécutabilité des statecharts UML (Unified Modeling Language) et leur contrôle via JMX (Java Management eXtensions).

Mots-clés

Composants logiciels, administration, statecharts, UML, MDA/MDE, J2ME, JMX.

1. INTRODUCTION

Une des grandes tendances dans les systèmes mobiles est de concevoir leur logiciel comme un assemblage de composants (composants Java dans les environnements J2ME [16], composants C# dans les environnements Windows CE…). Les composants sont interconnectés grâce à leurs interfaces, tout en cachant leur implémentation afin d'augmenter leur réutilisabilité et permettre leur déploiement par des tiers. Le déploiement s'effectue sur des dispositifs très variés tels que des téléphones mobiles, des PDAs, des boîtiers de télévision numérique, des cartes à puce, etc. Compte tenu du fait que les environnements de déploiement sont différents des environnements de développement, des compor-tements anormaux et/ou des disfonctionnements se produisent, ce qui, par conséquent, met en évidence le besoin crucial d'un système d'administration et de supervision.

Le but de cet article est de présenter le système d'administration de composants logiciels sans fil que nous avons imaginé pour répondre à cette problématique. La suite de l'article décrit une vue globale de notre système, puis nous détaillons en particulier sa méthode de gestion du comportement des composants ainsi que sa mise en œuvre technique.

2. SYSTEME GLOBAL

La Figure 1 est une vue globale du système que nous proposons pour l'administration de "composants logiciels sans fil". Par cette expression peut-être abusive de "composants logiciels sans fil", nous désignons simplement les composants logiciels déployés sur des systèmes sans fil. L'activité d'administration se concentre sur la configuration des composants et leur possible reconfiguration dynamique afin d'assurer un réel contrôle des appareils mobiles.

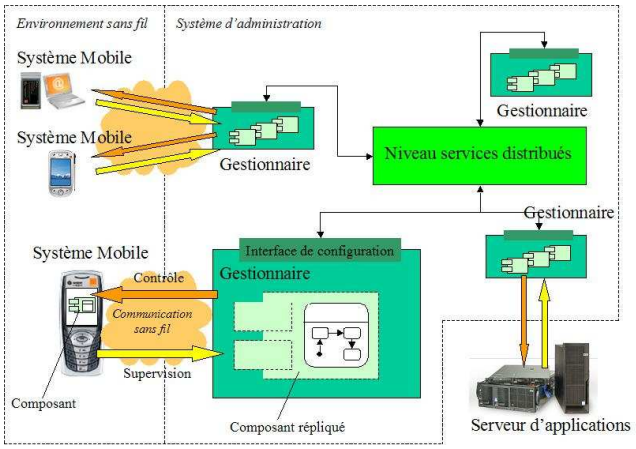

Figure 1. Système d'administration de composants logiciels sans fil

La supervision et le contrôle sont deux activités duales intervenant dans l'administration. Alors que le contrôle implique que le gestionnaire (*manager*) change l'état du composant géré, la supervision consiste pour le gestionnaire à récupérer des informations sur l'état actuel du composant. Etant donné que ces activités doivent être réalisées en utilisant une communication sans fil en relation avec des appareils fortement contraints, notre intention est de minimiser l'*overhead* généré côté mobile par notre système d'administration. La qualité de service doit être le moins possible affectée par l'exécution des fonctionnalités d'administration. Nous créons donc dans ce but une image, une sorte de cache, répliquant les composants à gérer dans le système d'administration. C'est ensuite ce composant image qui est manipulé par le gestionnaire et fait donc office de mandataire au composant sans fil.

Du fait de l'existence de comportements complexes à gérer côté administration, nous représentons le comportement des composants avec des statecharts [7] UML que nous implémentons grâce à une librairie Java qui supporte l'exécution de diagrammes d'états UML. Ainsi nous communiquons directement les événements de la machine à états des composants en exécution à leurs images et nous tenons ainsi à jour la vue du comportement de l'application sur le système d'administration. Le mécanisme de gestion du comportement par événement sera détaillé dans la section suivante. La Figure 2 en présente une mise en œuvre sur un composant sans fil J2ME [18] (A) communiquant par message (B) avec son composant répliqué J2SE (C). Le système de message utilise la technologie WMA [15].

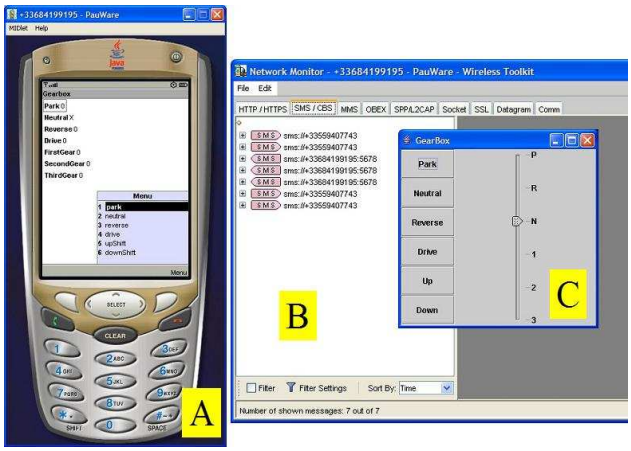

Figure 2. Mise en oeuvre Java du système de gestion du comportement des composants sans fil

Le grand avantage et la singularité de notre approche est que l'on profite de l'expressivité des statecharts pour superviser les composants. A la différence d'autres approches de supervision qui représentent l'état du système par la valeur de quelques variables ou attributs, on a ici accès aux états **abstraits et hiérarchisés** des statecharts qui ont été définis lors de la modélisation du comportement du composant et possèdent donc un **sens** pour l'application. Cela permet d'avoir une vision accrue de l'état global de l'application et permet d'envisager de nouvelles politiques d'administration basées sur ces états logiques. De même, l'utilisation des statecharts offre également de nouvelles possibilités dans l'activité de contrôle. En attaquant directement la machine à états des composants en exécution, on peut contraindre leur comportement et réellement manipuler l'application.

L'architecture globale du système d'administration se base sur le standard d'administration JMX (Java Management eXtensions) [10], [17], [19], et étend des idées de la technologie de test intégré BIT (Built-In Test) [2], [3], qui ne disposait d'aucune notion d'administration.

3. GESTION DU COMPORTEMENT PAR EVENEMENT

Dans l'esprit de l'ingénierie dirigée par les modèles (MDE / MDA), une recommandation préconisée par l'OMG [13], nous modélisons le comportement des composants sans fil par des statecharts UML directement exécutables.

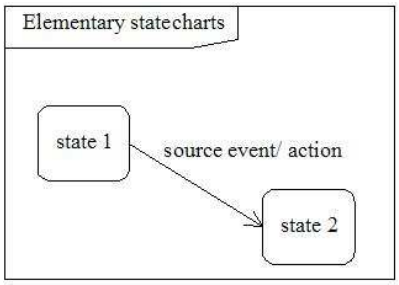

Figure 3. Statecharts élémentaire

La Figure 3 représente un statecharts élémentaire composé de deux états, *state 1* et *state 2*, et d'une transition de *state 1* vers *state 2* initiée par un événement de type *source event* qui génère en réaction une action de type *action*. Dans l'architecture que nous proposons, l'enjeu consiste à garder le statecharts du composant répliqué (*Replicated component*) en phase avec le statecharts du composant sans fil (*Wireless component*) lorsqu'un événement apparaît dans le système.

Utiliser pour cela une technique classique de *monitoring* se basant sur les états du statecharts et qui vérifie périodiquement l'état dans lequel se trouve le *Wireless component*, n'est pas une solution satisfaisante. En effet, des événements peuvent ne pas être observés s'ils se produisent dans la même période de rafraîchissement du moniteur et réduire la durée de la période pour minimiser ce phénomène se révèle de plus en plus coûteux en communication.

Nous nous sommes donc basés sur les événements des statecharts. Les statecharts du *Wireless component* et du *Replicated component* étant les mêmes, mêmes états et mêmes transitions définis, le principe de base est de notifier chaque événement intervenant dans le statecharts du *Wireless component* au *Replicated component* pour mettre à jour son statecharts. Ainsi toute l'information utile est transmise. Cependant tous les événements pouvant intervenir dans le système ne sont pas identiques et leur gestion doit donc être adaptée à leurs particularités.

Dans les sous-sections suivantes, nous définissons les différents types d'événement que nous avons identifiés et décrivons leur gestion dans notre système d'administration.

3.1 Evénements simples

Définition

"Un événement simple est un événement dont la source est locale au système sans fil et qui engendre une action interne au composant, i.e. une action ne nécessitant aucune autres ressources que celles disponibles dans le système sans fil considéré."

C'est par ce type d'événement que l'on provoque en particulier l'affichage de nouvelles informations sur un système disposant d'une interface graphique [8], ou l'exécution d'un traitement propre au composant, une fonction de calcul par exemple.

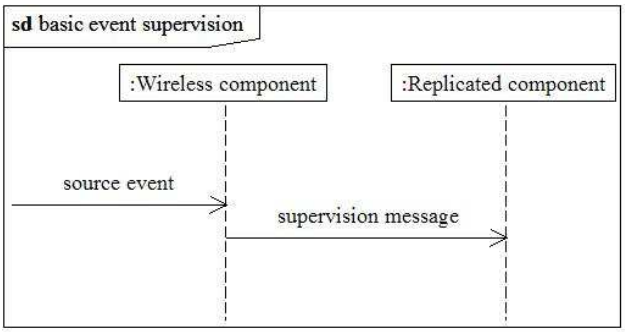

Figure 4. Supervision d'un événement simple

Le diagramme d'intéraction de la Figure 4 décrit la communication entre le *Wireless component* et le *Replicated component* au déclenchement d'un événement *source event* : un message de type *supervision message* est envoyé au *Replicated Component* ce qui a pour effet de synchroniser les statecharts des deux composants. C'est ainsi qu'est réalisée la supervision d'un événement simple.

Le contrôle d'un événement simple est décrit par le diagramme de la Figure 5. Contrairement à la Figure 4, le *source event* n'est pas reçu par le *Wireless Component* mais bien par le *Replicated*

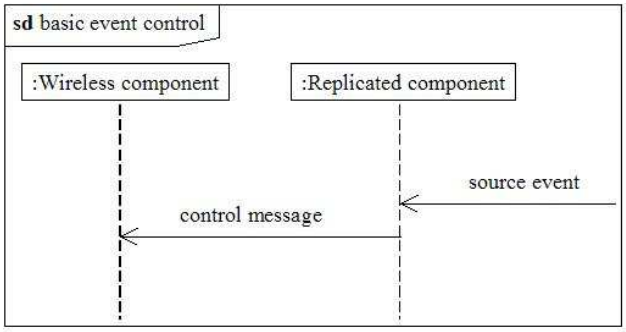

Figure 5. Contrôle d'un événement simple

component puisque c'est lui qui est en relation directe avec le gestionnaire. A la réception d'un tel événement, le statecharts du *Replicated component* est modifié et il émet un *control message* au *Wireless component* pour se synchroniser et déclencher les actions correspondantes à la transition effectuée. Le *control message* est interprété différemment d'un événement afin de ne pas déclencher le processus de supervision qui, via un nouveau *supervision message*, déclencherait une seconde fois l'événement sur le *Replicated component* et désynchroniserait les statecharts si la réaction à cet événement n'est pas idempotente.

3.2 Evénements de communication

Définition

"Un événement de communication est un événement qui engendre une communication sans fil, entrante ou sortante, sur le système sans fil considéré avec un composant distant."

Les composants logiciels déployés sur des systèmes sans fil sont fortement enclins à faire appel aux possibilités de communication sans fil dont sont pourvus leurs systèmes hôtes pour réaliser des applications distribuées.

Les diagrammes des figures 6, 7, et 8 décrivent la supervision et le contrôle des événements de communication. Ils font intervenir une troisième entité, le *Remote component* qui est le composant avec lequel le *Wireless component* est en communication sans fil. On peut s'étonner de l'absence de diagramme pour le contrôle d'un événement de communication entrant mais cela est parfaitement logique puisque le *Replicated component* sur lequel on initie le contrôle est une réplique du *Wireless component* et n'a par conséquent aucune prise sur le *Remote component* déployé sur un système distant. Pour contrôler un événement de communication entrant il faut pour cela que le *Remote component* soit également administré par le système : le *Remote component* est alors vu comme le *Wireless component* et cela correspond au

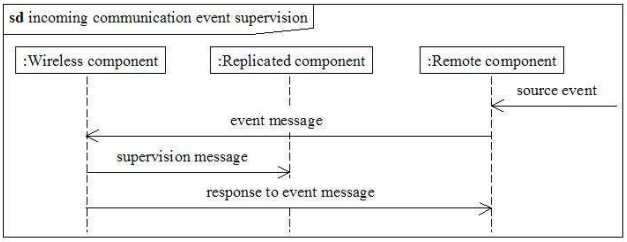

Figure 6. Supervision d'un événement de communication entrante

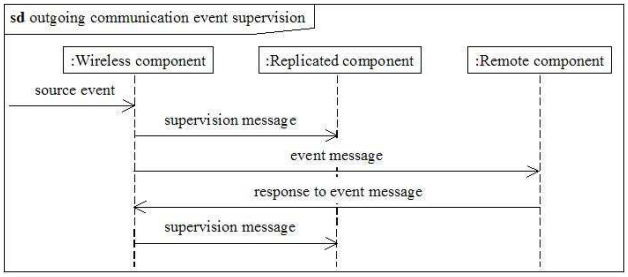

Figure 7. Supervision d'un événement de communication sortante

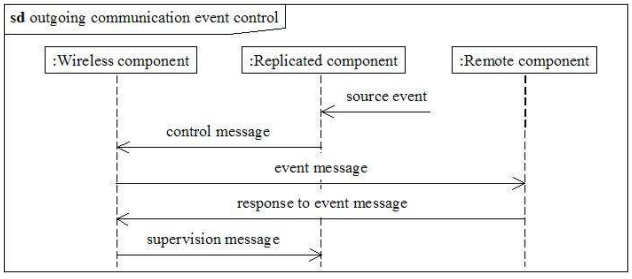

Figure 8. Contrôle d'un événement de communication sortante

diagramme de la Figure 8.

3.3 Evénements de service

Définition

"Un événement de service est un cas particulier d'événement simple, c'est un événement dont la source provient d'un service propre au système sans fil considéré."

Les *timers*, les notifications d'appel téléphonique sur les téléphones mobiles ou les interruptions systèmes sont des exemples d'événements de service que l'on peut avoir à gérer. Pour obtenir le service, le *Wireless component* en fait la requête (*service request*) au *Service component* correspondant qui génère ensuite les événements de service (*service event*) correspondants.

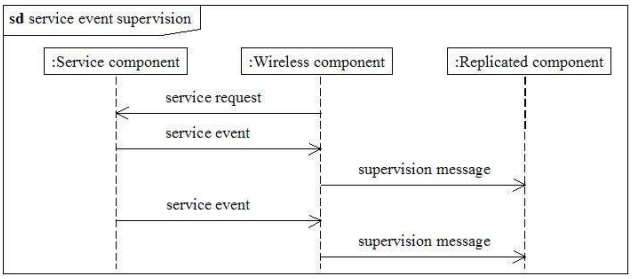

Figure 9. Supervision d'événements de service

Les événements de service étant un cas particulier des événements simples, ils peuvent être gérés comme ces derniers (cf. Figure 9). Cependant les services comme les timers par exemple génèrent de nombreux événements qui seront donc coûteux en messages de supervision. Il peut donc être intéressant de répliquer le service (*Replicated service component*) sur le système d'administration afin d'éviter ces communications (cf. Figure 10). Bien que les statecharts des Wireless et Replicated components puissent différer dans le temps, ils doivent néanmoins rester logiquement équivalent.

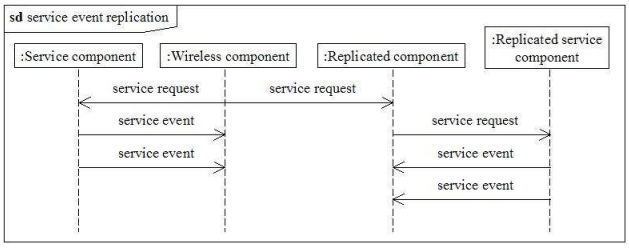

Figure 10. Réplication d'événements de service

4. TRAVAUX CONNEXES

Les travaux les plus proches des nôtres sont ceux de [14] qui définissent une architecture de middleware permettant d'administrer des appareils sans fil. Cette architecture utilise le principe d'externalisation de l'état, de la structure et de la logique du système pour contrôler pleinement l'application en exécution. Notre approche utilise en fait ce même principe mais à un plus haut niveau d'abstraction via l'utilisation des composants logiciels et des statecharts ce qui permet d'intégrer notre approche dans une méthode de développement basée sur UML [12].

Dans le domaine des composants logiciels, l'article [6] utilise également une approche basée sur les modèles, en l'occurrence des réseaux de Petri, pour administrer le comportement des composants, mais traite seulement la supervision et pas le contrôle.

D'autres travaux utilisent également les statecharts dans les systèmes mobiles mais plus dans le but de modéliser la mobilité comme dans [1], [5], [9] ou [11].

Au niveau technique, l'article [4] utilise également JMX pour administrer le cycle de vie des composants d'une plate-forme de services utilisable sur des systèmes sans fil.

5. CONCLUSION

Dans cet article, nous avons présenté une architecture pour la supervision et le contrôle du comportement des composants logiciels sans fil. Le comportement des composants est décrit par des statecharts directement exécutés par le système sans fil et répliqués sur le système d'administration.

Ce système fonctionne actuellement pour l'administration du comportement des composants pris individuellement. Le but est de l'étendre en prenant en compte les relations entre composants afin d'exercer un contrôle plus fin de l'application assemblée. En effet, analyser une défaillance d'un système de composants ou alors reconfigurer un système à composants est plus épineux. Cette perspective de recherche s'appuie sur la formalisation d'une relation de composition basée sur les états des composants, c'est-à-dire des règles d'assemblage sur lesquelles les règles d'administration elles-mêmes peuvent s'appuyer.

6. REFERENCES

[1] Acharya, S. *et al*. (2002). MOBICHARTS: A Notation to Specify Mobile Computing Applications. In Proceedings of the 36th HICSS. IEEE Computer Society.

[2] Barbier, F. *et al*. (2003). Incorporation of Test into Software Components. In Proceedings of the 2nd International Conderence on COTS-Based Software Systems, Ottawa, Canada, Lecture Notes in Computer Science, Springer.

[3] Edler, H. & Hörnstein, J. (2003). Component+ Final Report. http://www.component-plus.org/pdf/reports/Final report 1.1.pdf.

[4] Frénot, S. & Stefan, D. (2004). Instrumentation de plates-formes de services ouvertes – Gestion JMX sur OSGi. Mobile and Ubiquitous Computing, Nice, France. ACM.

[5] Grassi, V. *et al*. (2004). A UML Profile to Model Mobile Systems. The Unified Modelling Language: Modelling Languages and Applications. 7th International Conference, Lisbon, Portugal, LNCS 3273, Springer.

[6] Grosclaude I. (2004). Model-based monitoring of component-based software systems. In Proceedings of the 16th European Conference on Artificial Intelligence. ECAI, Valencia, Spain, IOS Press.

[7] Harel, D. (1987). Statecharts: a Visual Formalism for Complex Systems. Science of Computer Programming, 8: 231-274.

[8] Horrocks, I. (1999). Constructing the User Interface with Statecharts. Addison-Wesley Professional; 1st edition.

[9] Knapp, A. *et al*. (2004). Refining Mobile UML State Machines. 10th Intl. Conf. Algebraic Methodology and Software Technology (AMAST 2004). Springer-Verlag, LNCS.

[10] Kreger, H. *et al*. (2003). Java and JMX. Addison Wesley.

[11] Latella, D. *et al*. (2004). Mobile UML Statecharts with Localities. Global Computing, IST/FET International Workshop, GC 2004, Rovereto, Italy. LNCS 3267, Springer.

[12] Mellor, S. *et al*. (2002). Executable UML: A Foundation for Model Driven Architecture. Addison-Wesley.

[13] Miller, J. & Mukerji, J. (2003). Technical Guide to Model Driven Architecture: MDA Guide Version 1.0.1. OMG.

[14] Román, M. & Islam, N. (2004). Dynamically Programmable and Reconfigurable Middleware Services. In Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Toronto, Canada, Springer-Verlag.

[15] Siemens AG. (2003). Java Wireless Messaging API (WMA) Specification Version 1.1.

[16] Sun Microsystems. (2000). Connected Limited Device Configuration. Specification Version 1.0a. J2ME.

[17] Sun Microsystems. (2002). Java Management Extensions Instrumentation and Agent Specification, v1.2.

[18] Sun Microsystems. (2002). Mobile Information Device Profile for J2ME. Version 2.0. JSR118 Expert Group. Java Community Process.

[19] Sun Microsystems. (2003). Java Management Extensions Remote API 1.0 Specification.