Vers un environnement de déploiement autonomique

Mohamed El Amine Matougui, Sébastien Leriche

Vers un environnement de déploiement autonomique

Mohammed El Amine Matougui
Institut Télécom ; Télécom SudParis
UMR 5157 CNRS SAMOVAR
Évry, France
mohammed_el_amine.matougui@it-sudparis.eu

Sébastien Leriche
Institut Télécom ; Télécom SudParis
UMR 5157 CNRS SAMOVAR
Évry, France
sebastien.leriche@it-sudparis.eu

Résumé
Le déploiement autonomique de logiciel dans les systèmes répartis à grande échelle (grilles, systèmes pervasifs, P2P...) est une problématique actuelle ouverte. Quelques solutions existent aujourd'hui, mais ne sont exploitables que dans le cadre d'architectures figées. De ce fait, elles ne prennent pas en compte la réalité des évolutions dynamique (variation de QoS, pannes, apparition et disparition de nœuds, de services...) des environnements ouverts.

Nous présentons une approche originale, basée sur nos précédents résultats de recherche dans le domaine des agents mobiles adaptables (AMA). Les propriétés de ces agents logiciels (autonomie, proactivité, adaptation dynamique au contexte...) nous permettent d'envisager la construction d'un intergiciel capable de déployer des applications dans les environnements les plus instables, en limitant au minimum les interventions humaines.

Mots-clés
Intergiciel, déploiement logiciel, gestion autonome, auto-déploiement, auto-reconfiguration.

ABSTRACT
Autonomic software deployment in large scale distributed systems (grids, pervasive systems, P2P systems) is an open issue. Some solutions exist, but they are usable only in the context of static architectures. They do not take into account the dynamic changes of QoS, hosts failures, appearance and disappearance of nodes and services occurred in open environments.

In this paper we present an original approach, based on our previous research on adaptable mobile agents (AMA). The properties autonomy and dynamic adaptation to context of mobile agents allow us to consider building a middleware for deploying applications in the most unstable environments, with minimal human intervention.

Keywords
Middleware, Software deployment, autonomic deployment, self-deployment, self-reconfiguring

1. INTRODUCTION
Le déploiement de logiciel est défini comme un processus complexe qui comporte un ensemble d'activités de déploiement [3], a l'heure actuelle il n'existe pas de consensus autour des activités de déploiement. Le cycle de vie de déploiement regroupe toutes les activités depuis la validation du logiciel par le producteur jusqu'à sa désinstallation des sites cible de déploiement, cela inclue les activités de mise en place sur les sites cible comme l'installation et l'activation ainsi que les activités de maintenance du logiciel comme reconfiguration, mise à jour. La figure 1 représente le cycle de vie du déploiement [2].

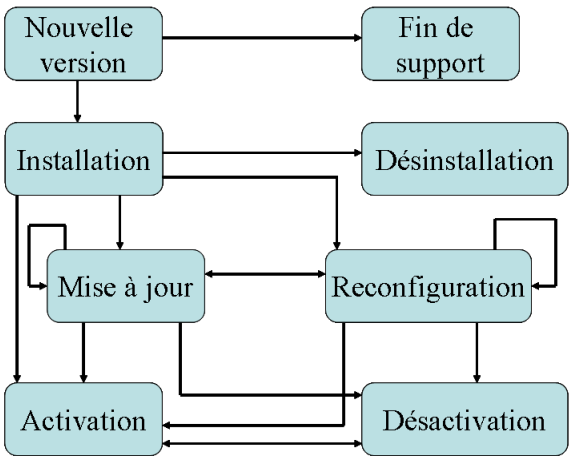

Figure 1: Cycle de vie du déploiement logiciel

Les principales activités de déploiement sont l'installation, la désinstallation, l'activation, la désactivation, la mise à jour et la reconfiguration.

L'activité d'installation [2] a pour objectif la préparation de l'environnement du logiciel à déployer. Elle permet d'insérer un logiciel sur les sites cible de déploiement. Elle comporte plusieurs opérations telles que la résolution des dépendances logicielles et matérielles du logiciel, le transfère du logiciel

vers les sites cible et la configuration du système afin que le logiciel déployé soit activable.

La désinstallation [2] est une activité qui consiste à remettre à l'état initial le système et d'enlever toutes les modifications introduites sur les sites cible de déploiement par les activités de déploiement.

L'activité d'activation [2] regroupe toutes les opérations nécessaires pour que le logiciel installé puis être exécuté. Elle consiste au lancement de l'exécution du logiciel déployé. Dans certains cas l'activation d'un certain nombre d'éléments logiciels peut être nécessaire pour l'exécution du logiciel (des serveurs web, autres logiciels installés,Ĕ) avant de pouvoir activer le logiciel déployé.

L'activité de désactivation[2] est une activité utilisée en général avant les activités de mise à jour, de reconfiguration et de désinstallation. Elle consiste à interrompre l'exécution du logiciel ou une partie du logiciel déployé.

L'activité de mise à jour [2] consiste à introduire des modifications sur le logiciel installé en utilisant la configuration courante. Un exemple de cette activité est le déploiement d'une nouvelle version du logiciel déployé.

La reconfiguration [2] modifie un logiciel déjà déployé mais en utilisant une configuration différente à la configuration existante. Ces modifications peuvent être la suppression, l'ajout ou le remplacement d'une entité logiciel et la réparation d'une panne.

Le terme déploiement de logiciel est souvent utilisé et laisse penser que l'unité de déploiement est le logiciel, tandis que souvent les éléments qui sont effectivement déployés sont des unités de déploiement (des composants, des services, Ĕ). Une unité de déploiement est une entité logicielle packagée sous la forme d'un package exécutable dans un environnement d'exécution sans besoin d'intervenir pour le modifier au moment du déploiement afin de le rendre prêt à être déployé [3]. Notons qu'un logiciel peut être une unité de déploiement en lui-même ou être constitué de plusieurs unités de déploiement.

Dans notre proposition de déploiement de logiciel nous nous intéressons au déploiement autonomique de logiciel à grande échelle. Les systèmes ciblés par ce déploiement sont du type grille de calcule, systèmes P2P et systèmes pervasifs. Ces systèmes sont caractérisés par un grand nombre de sites équipés d'environnements matériels et logiciels hétérogènes. Une partie de ces systèmes est caractérisée aussi par la mobilité des sites (cas des systèmes pervasifs) et des changements de topologie du réseau (disparition et apparition des pairs dans les systèmes P2P).

Il existe plusieurs plate-formes de déploiement dédiées à ce type de systèmes tels que Software Dock [6], DeployWare [5] et JADE [8], mais elles ne sont exploitables que dans le cadre de topologies statiques et ne prévoient pas des solutions d'auto-adaptation et de reconfiguration prenant en compte les variations de QoS et/ou par exemple les situations de pannes de machines et des liens du réseau qui caractérise les environnements ouverts. En plus l'utilisateur de ces plates-formes de déploiement doit gérer manuellement la plupart du temps les activités de déploiement ainsi que les opérations de reconfiguration ce qui représente une intervention humaine très importante dans le processus de déploiement. Ces interventions deviennent très vite une tâche très complexe qui présente un besoin de rendre autonome le processus de déploiement pour réduire au minimum les interventions humaines.

Dans cet article, après avoir présenté deux scénarios motivants nos travaux (section 2), nous présentons notre ébauche de proposition de déploiement autonomique de logiciel à grande échelle qui se base sur un système d'agents mobiles adaptable pour la réalisation des activités de déploiement (section 3). Dans la section 4 nous présentons l'état d'avancement de l'implémentation du prototype. Enfin, dans la section 5 nous présentons une conclusion et les perspectives de notre proposition.

2. SCÉNARIOS MOTIVANTS

Pour pouvoir exprimer au mieux les différents problèmes et les spécificités du déploiement de logiciel à grande échelle nous présenterons deux scénarios de déploiement. Le premier scénario consiste au déploiement d'une application de supervision de l'état d'activités pour un ensemble d'équipements connectés à un réseau WiFi en utilisant l'intergiciel de gestion de contexte COSMOS[7], développé dans notre équipe. Le deuxième scénario consiste au déploiement d'une application de simulation sur une grille.

2.1 Scénario 1

Considérons un logiciel réparti de supervision de l'état d'activités d'un ensemble d'équipements pour un ensemble de participants à une conférence. Ce logiciel fournit à l'ensemble des participants des informations statistiques sur leurs équipements connectés à un réseau WiFi au moment de l'expérimentation au moyen de l'intergiciel de gestion de contexte COSMOS. Le nombre de participants à cette expérimentation est imprévisible, il peut être à l'échelle d'une dizaine ou d'une centaine de participants.

COSMOS comporte plusieurs versions, une version installable sur des téléphones mobiles, une version installable sur des Smartphones équipés d'Android et une version java standard installable sur des ordinateurs équipés d'une machine virtuelle java standard.

Les informations récoltées dans le cadre de cette expérimentation sont la taille de la RAM disponible, le type de l'OS utilisé, le pourcentage de l'occupation du CPU et la taille du disque disponible pour chaque équipement impliqué. Les équipements impliqués dans l'expérimentation sont de natures différentes, ils peuvent être des ordinateurs portables, des Smartphones, des netbooks et des Macbook...

Chaque équipement connecté au réseau WiFi est potentiellement un site cible de déploiement.

L'application à déployer est constituée de 8 composants (nœud de contexte COSMOS), chaque composant de l'application représente une unité de déploiement. La figure 2 présente l'application sous la forme de nœuds de contexte COSMOS. Le composant OS-Type permet de retourner le

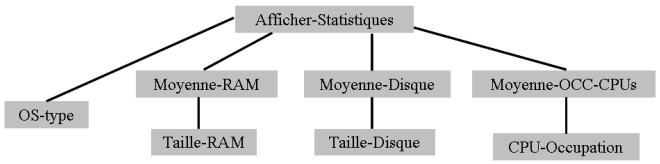

Figure 2: Composition de nœuds de contexte de l'application

type du système d'exploitation utilisé. Le composant Taille-RAM permet le calcule de la taille de la RAM disponible dans chaque site. Le composant Taille-Disque permet de calculer la taille disque disponible dans chaque site. Le composant CPU-Occupation permet de calculer le pourcentage de l'occupation des processeurs. Les composants Moyenne-RAM, Moyenne-Disque et Moyenne-CPUS-Occupation permettent de calculer respectivement la moyenne de RAM disponible, la moyenne de l'espace disque disponible et la moyenne du pourcentage d'occupation des processeurs. Enfin le composant Afficher-Statistiques a pour rôle l'affichage des informations statistiques calculées au courant de cette expérimentation.

Le plan de déploiement souhaitable de cette application est le suivant : on distingue une machine dans lequel on installe tous les composants de l'application puis on installe les composants OS-type, Taille-RAM, Taille disque et CPU-occupation sur toutes les autres machines impliquées.

Pour initialiser le fonctionnement global du système, nous aurons besoin d'un mécanisme de "bootstrap" qui aura pour rôle la préparation de l'environnement de l'exécution pour le système de déploiement, ainsi que pour résoudre le problème de l'administration multiple. Cela pourrait être réalisé par l'intermédiaire d'un programme dédié, volontairement installé par l'administrateur d'une machine, et mis à sa disposition par d'autres moyens (push sur bluetooth, publicité, SMS. . .)

Le système de déploiement doit couvrir toutes les activités de déploiement. Il doit permettre l'installation du logiciel selon le plan de déploiement défini. Dans cette activité le système de déploiement doit résoudre les dépendances logicielles et matérielles, puis transfère les composants sur les sites cible et enfin il procède à leurs mises place dans les sites cible de déploiement.

Le système de déploiement doit aussi permettre l'activation du logiciel déployé. Une fois le logiciel activé il récupère les informations sur les sites (taille de la RAM et du disque,. . .), puis il calcule les moyennes et affiche les résultats obtenus.

Le système de déploiement doit aussi fournir les mécanismes nécessaires pour la désactivation, la désinstallation et la mise à jour du logiciel déployé. Enfin le système de déploiement doit aussi prévoir un mécanisme de reconfiguration dynamique au moment de l'exécution pour prendre en charge les situations de pannes de machine, les déconnexions et les nouvelles connexions.

2.2 Scénario 2

Le deuxième scénario présente le déploiement d'une application de simulation à grande échelle sur une grille. Le processus de simulation commence par la préparation de la simulation, cette étape consiste à développer le modèle de simulation ainsi que les différents scénarios de tests de ce modèle. La deuxième étape consiste au déploiement du modèle de simulation et les différents scénarios de tests sur une grille. La troisième étape consiste à analyser les résultats de la simulation.

Dans ce scénario l'étape qui nous intéresse est la deuxième. Pour déployer cette application de simulation, la majorité des outils de déploiement existants exige que la personne en charge du déploiement fournisse un ou plusieurs fichiers de description dans lesquelles il déclare un certain nombre d'informations. L'utilisateur doit spécifier les propriétés du logiciel à déployer, par exemple le chemin vers l'archive de l'application, les ports de communication utilisés et les dépendances vers d'autres logiciels. L'utilisateur doit aussi souvent fournir un plan de déploiement précis de son application, il doit décrire comment installer et activer son application. Enfin l'utilisateur doit lister à la main tous les nœuds physiques dans lesquels les unités de déploiement doivent être déployées. La déclaration des sites cible et la mise en place du plan de déploiement manuellement une tâche très lourde à réaliser par un opérateur humain dans le cadre d'un système de grosse taille, tel que les grilles de calcul.

Une fois que l'utilisateur a réalisé toutes les déclarations, le processus de déploiement peut commencer. Dans ce type de système, la personne en charge du déploiement n'a pas besoin d'un mécanisme de détection de sites cible de déploiement par ce qu'elle connaît au préalable les différents sites. De même, le problème d'administration multiple posé par le premier scénario n'est plus pertinent, étant donné que l'on est dans le cadre d'un système mono-administrateur.

Les outils de déploiement existants pour ce type de système ne couvrent pas en général toutes les activités de déploiement, ces dernières sont souvent réalisées manuellement ou d'une manière semi-automatique à l'aide de scripts. En plus la majorité de ces systèmes n'offrent pas généralement des mécanismes de reconfiguration et d'auto-adaptation pour la prise en charge des situations de pannes de machines et des liens du réseau.

2.3 Synthèse

Dans cette section nous avons présenté deux scénarios de déploiement de logiciel sur des environnements répartis à grande échelle. Le premier scénario traite le déploiement d'une application dans un environnement ouvert. Le déploiement de logiciel dans ce type d'environnement engendre des activités supplémentaires.

1. La première activité est la découverte du réseau. Dans ce type de système (P2P, pervasif) on ne connaît pas au préalable les sites de déploiement. Donc on a besoin de détecter les sites cible de déploiement pour pouvoir déployer un logiciel.
2. La deuxième activité est la satisfaction des problèmes d'administration et de droit d'accès pour que notre

logiciel soit déployable sur les sites détectés.

3. Enfin les outils de déploiement pour ce type d'environnement doivent prévoir des mécanismes d'auto-adaptation et de reconfiguration pour la prise en charge des différentes situations de pannes et de déconnexions.

Le deuxième scénario présente le déploiement d'une application de simulation à grande échelle. Le déploiement de ce logiciel avec les plates-formes de déploiement existantes tel que DeployWare, exige la fourniture de plusieurs descripteurs de déploiement dans lesquelles l'utilisateur déclare les propriétés du logiciel à déployer, la liste des sites cible et son plan de déploiement. En plus l'utilisation de ces outils nécessite un effort supplémentaire non négligeable qui ce manifeste dans la gestion manuelle des activités de déploiement et les opérations de reconfiguration.

De ce fait, la plate-forme de déploiement qui répond aux problématiques et spécificités de ce type d'environnement (grille, P2P, pervasif) doit être :

1. Capable de détecter, gérer, et accéder aux sites cible d'une manière automatique.
2. Capable de gérer les hétérogénéités logicielles et matérielle des sites détectés.
3. Fournir un moyen pour la déclaration des dépendances logicielles, les préférences matérielles et les contraintes de déploiement.
4. Capable de calculer un plan de déploiement qui satisfasse les contraintes de déploiement d'une manière automatique.
5. Capable d'exécuter les activités de déploiement avec un minimum d'intervention humaine.
6. Capable de s'auto-adapter et offre des mécanismes de reconfiguration automatiques au moment de l'exécution pour prendre en charge les situations de pannes de machine et des déconnexions.
7. Utilisables dans des topologies à grande échelle.

3. PROPOSITION POUR LE DÉPLOIEMENT AUTONOMIQUE

Dans cette section nous allons présenter brièvement l'architecture de notre intergiciel pour le déploiement autonomique de logiciel. Nous proposons un intergiciel qui utilise OSGi [9] comme support de déploiement, un langage dédié pour l'expression des contraintes de déploiement, un solveur de contraintes pour le calcul d'un plan de déploiement qui satisfasse les contraintes et un système d'agents mobiles adaptables qui prend en charge l'exécution et la supervision des activités de déploiement.

Les objectifs de l'intergiciel sont de permettre le déploiement réparti à grande échelle en réduisant au minimum les interventions humaines dans le processus de déploiement.

3.1 Support de déploiement

Notre intergiciel utilise le Framework OSGi comme support de déploiement. OSGi [9] est une spécification qui définit une plate-forme java qui permet le déploiement d'unités de déploiement appelées bundle dans l'environnement d'exécution OSGi. Un bundle est une archive java au format JAR, contenant également un fichier de méta-données qui décrit le JAR, les classes et les bibliothèques de code de l'application et une classe d'activation.

Les activités de déploiement qu'on peut réaliser avec le Framework OSGi sont l'installation, l'activation, la désactivation, la mise à jour et la désinstallation des bundles. En plus l'environnement prend en charge la gestion des dépendances de code qui doivent être satisfaites après l'installation du bundle afin de pouvoir l'activer. Les applications OSGi sont construites à partir des bundles connectés à travers des services. Les services peuvent dynamiquement apparaître ou disparaitre au courant de l'exécution de l'application.

L'utilisation d'OSGi comme support de déploiement va nous permettre le déploiement de logiciel sur plusieurs types d'équipements dotés d'environnements matériels et logiciels hétérogènes y compris des matériels de petite taille (type smartphones / tablettes PC / UMPC...). En plus la réutilisation des fonctionnalités de déploiement telles que l'installation et la désinstallation offertes par l'environnement d'exécution OSGi nous permettent de nous concentrer sur les autres aspects.

3.2 L'unité de déploiement

L'unité de déploiement est actuellement celui d'un bundle OSGi. Le déploiement d'autres types d'unités de déploiement sera envisagé ultérieurement.

3.3 Description des contraintes de déploiement

Comme toutes les plates-formes de déploiement existantes, l'utilisateur de notre intergiciel doit fournir des informations sur l'application à déployer et un ensemble de contraintes de déploiement. Les informations (propriétés du logiciel à déployer) et les contraintes de déploiement seront utilisées par la suite pour le calcul d'un plan de déploiement qui satisfasse les contraintes de déploiement.

Les informations de l'application à déployer ainsi que ses différentes contraintes de déploiement seront déclarées en utilisant un langage dédié pour description de contraintes de déploiement. Ce langage est fortement inspiré du langage DELADAS [4].

Les informations à déclarer sont respectivement le chemin vers l'archive de l'application, le nom et la version du logiciel, les composants qui forment l'application, les services requis et fournis par l'application, les dépendances vers d'autres logiciels et enfin les préférences matérielles et logicielles des sites cible de déploiement (la taille de la RAM et du disque, le type de l'OS, l'état de la batterie,....

Les contraintes de déploiement sont le nombre minimum des sites cible de déploiement et les contraintes liées aux activités de déploiement, par exemple les contraintes d'installation et

les contraintes d'activation. La figure 3 présente un exemple de déclaration des contraintes d'installation.

```
ConstraintsSet {

InstallConstraintes {
Install exemple1 forall hosts;
Install exemple2 in 2 hosts;
}
…
}
```

Figure 3: Déclaration de contraintes d'installation

Dans cet exemple les contraintes d'installation sont, installer le composant exemple1 sur tous les hôtes détectés et installer le composant exemple2 sur seulement deux hôtes détectés.

3.4 Système d'agents mobiles adaptable

Dans notre intergiciel de déploiement, l'entité qui prend en charge l'exécution des activités de déploiement est un système d'agents mobiles adaptables. Un agent mobile adaptable est une entité autonome capable de communiquer et de se déplacer disposant des connaissances et d'un comportement privé et d'une capacité d'exécution propre, en plus il dispose des mécanismes d'adaptation au contexte de l'exécution [1].

À chaque activité de déploiement on associe un ou plusieurs agents spécialisés. Les opérations de supervision et de contrôle du processus de déploiement seront assurées par un autre agent. Donc dans notre système on a des agents d'installation, des agents de désinstallation, des agents d'activation, des agents de désactivation, des agents de mise à jour et des agents de supervision.

L'agent d'installation a pour objectif l'installation de bundle selon le plan de d'installation calculé. Les tâches confiées à cet agent sont : le téléchargement du package de l'application sur les différents sites cible de déploiement, résolution des dépendances de l'application, l'installation physique du package et enfin notifier à l'agent de supervision la fin de l'installation.

L'agent de désinstallation a pour objectif d'enlever toutes les modifications introduites par les différentes activités de déploiement. Cet agent doit être en mesure de détecter si une dépendance installée est en cours d'utilisation par une autre application afin de ne pas la désinstaller et avoir causé des situations de conflits.

L'agent d'activation a pour but de démarrer l'exécution de l'application selon le plan d'activation à la demande de l'agent de supervision.

L'agent de désactivation arrête sur la demande d'un autre agent tout ou une partie de l'application déployée.

L'agent de mise à jour réalise les mises à jour de l'application sur la demande de l'agent de supervision.

L'agent superviseur a pour rôle, le contrôle du processus de déploiement et la reconfiguration de l'application dans le cas de pannes et de déconnexions. L'agent de supervision offre des interfaces d'échanges avec les utilisateurs du système de déploiement. Ces interfaces permettent de consulter l'état du processus de déploiement et d'intervenir dans une ou plusieurs activités de déploiement si l'utilisateur le souhaite.

L'exécution et la création ainsi que les communications des agents mobiles sont assurés par un système d'accueil d'agents installé et exécuté sur l'environnement d'exécution d'OSGi.

3.5 Exécution du processus de déploiement

La figure 4 résume toutes les étapes de l'exécution du processus de déploiement.

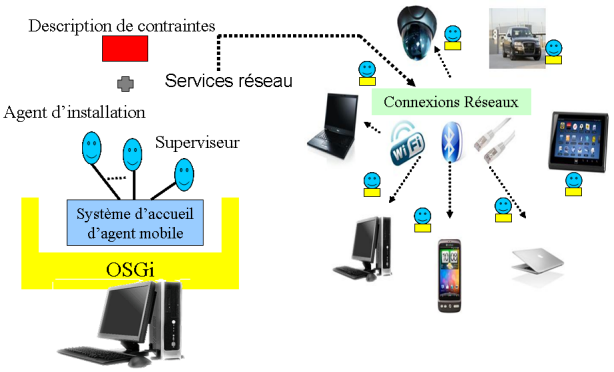

Figure 4: Processus de déploiement

Le processus de déploiement commence dès que l'utilisateur fournit ça description de contraintes de déploiement.

L'intergiciel commence par le lancement du service de découverte du réseau pour la détection des sites cible de déploiement. Une fois les sites détectés l'intergiciel lance le solveur de contraintes en luit passant comme argument les contraintes de déploiement. Si le solveur réussit à calculer un plan de déploiement qui satisfasse les contraintes, le système d'agents mobiles est déclenché, les agents de déploiement sont créés et envoyés vers les sites cible afin d'exécuter les différentes activités de déploiement. Le contrôle du processus de déploiement est assuré par les agents superviseurs.

4. PROTOTYPE

Pour valider notre approche, nous avons procédé au développement d'un prototype. Dans ce prototype le système d'accueil d'agents mobile utilisé est JavAct[1].

Nous avons transformé JavAct sous la forme d'un bundle OSGi pour permettre son installation et son activation dans l'environnement d'exécution OSGi. Le bundle de JavAct est testé et exécuté sur les Framework Equinox[2] du projet Eclipse et le Framework Appache Felix[3].

[1]http://www.javact.org
[2]http://www.eclipse.org/equinox/
[3]http://felix.apache.org/site/index.html

Nous avons aussi développé les agents d'installation, d'activation, de désactivation, de désinstallation et de mise à jour. L'ensemble de ses agents permet le déploiement d'applications composées d'un ou plusieurs bundles sur l'environnement d'exécution OSGi. La version actuelle du prototype permet d'envoyer nos agents de déploiement à d'autres systèmes d'accueil JavAct exécutés sur d'autres machines virtuelle java afin d'exécuter des activités de déploiement.

Nous allons poursuivre l'amélioration du prototype en développant les agents de supervisions de déploiement afin de contrôler tout le processus de déploiement et offrir la possibilité de reconfiguration et d'auto-adaptation à notre système de déploiement.

Entre autres nous avons commencé le développement d'un service de découverte du réseau ainsi que l'étude et l'extension du langage de description de contraintes de déploiement DELADAS et son solveur de contraintes qui seront intégré dans la prochaine version du prototype.

5. CONCLUSION

Dans cet article nous avons présenté une approche pour le déploiement autonomique de logiciel à grande échelle. Cette approche consiste à un assemblage de technologies pour le développement d'un intergiciel pour le déploiement autonomique de logiciel à grande échelle.

L'intergiciel est constitué d'un langage de description de contraintes de déploiement, d'un service de découverte de réseau, d'un service de bootstrap pour la préparation de l'environnement de l'exécution, d'un solveur de contraintes de déploiement pour calculer un plan de déploiement et d'un système d'agents mobiles adaptable qui exécute les activités de déploiement.

Nous avons choisi le Framework OSGi comme plate-forme de déploiement pour permettre le déploiement de logiciels sur les matérielles de petite taille comme les Smartphones et les téléphones portables et même d'autres types d'équipement comme les calculateurs des voitures.

Grâce aux caractéristiques des agents mobiles (comportement autonome et capacité de déplacement) nous pouvons réaliser des reconfigurations et des adaptations dynamiques au moment de l'exécution sans aucune intervention de l'utilisateur de notre système de déploiement. Nous envisageons par l'assemblage des différentes technologies répondre aux problématiques des environnements ouverts comme les systèmes P2P et les systèmes pervasifs en réduisant au minimum les interventions humaines dans le processus de déploiement.

Nous poursuivons actuellement nos travaux sur une nouvelle version du prototype. Nous travaillons sur la génération du plan de déploiement à partir d'une description de contraintes écrite à l'aide du langage de description de contraintes de déploiement. Nous travaillons aussi sur les comportements des agents de superviseurs et des algorithmes de déploiement par des agents pour résoudre au mieux les problématiques de déploiement à grande échelle et offrire un système qui permet de réaliser des opérations d'auto-adaptation dynamique au moment de l'exécution.

6. REFERENCES

[1] J.-P. Arcangeli, S. Leriche, and M. Pantel. Un framework à composants et agents pour les applications réparties à grande échelle. *L'OBJET*, 12(4):103–132, 2006.

[2] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. van der Hoek, A. L. Wolf, A. V. Der, E. L. Wolf, and E. L. Wolf. A characterization framework for software deployment technologies. Technical report, Dept. of Computer Science, University of Colorado, 1998.

[3] A. Dearle. Software deployment, past, present and future. In *FOSE*, pages 269–284, 2007.

[4] A. Dearle, G. N. C. Kirby, and A. J. McCarthy. A framework for constraint-based deployment and autonomic management of distributed applications. In *ICAC*, pages 300–301, 2004.

[5] A. Flissi, J. Dubus, N. Dolet, and P. Merle. Deploying on the grid with deployware. In *CCGRID*, pages 177–184, 2008.

[6] R. S. Hall, D. Heimbigner, and A. L. Wolf. A cooperative approach to support software deployment using the software dock. In *Proceedings of the 21st international conference on Software engineering*, ICSE '99, pages 174–183. ACM, 1999.

[7] R. Rouvoy, D. Conan, and L. Seinturier. Software architecture patterns for a context-processing middleware framework. *IEEE Distributed Systems Online*, 9(6):1, 2008.

[8] C. Taton, S. Bouchenak, N. De Palma, D. Hagimont, and S. Sicard. Self-sizing of clustered databases. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, WOWMOM '06, pages 506–512, Washington, DC, USA, 2006. IEEE Computer Society.

[9] The OSGi Alliance. OSGi service platform core specification, release 3. version 4.2, 2009.