Context-Aware Deployment of multi-component applications

Dhouha Ayed, Chantal Taconet, and Guy Bernard

GET / INT, CNRS Samovar 5157
9 rue Charles Fourier, 91011 Évry, France
{Dhouha.Ayed, Chantal.Taconet, Guy.Bernard}@int-evry.fr

Abstract. The deployment refers to all activities that make software available to its users. These activities become more difficult for software built from heterogeneous distributed components. In this paper, we propose a deployment platform for distributed multi-component software. In this platform, the deployment is carried out by components which are separated from software functionalities. The deployment solution that we propose includes adaptation components which collect information from sensors and act on deployment components.

1 Introduction

Software deployment is the complex process that covers all the activities performed after the development of a software system. These activities include configuring, installing, updating, reconfiguring and even de-installing a software [1]. The principal deployment goal is to assemble usable software system's configuration on the consumer site and maintain the integrity of the deployed software configuration after changes on consumer or producer site.

There are several deployment solutions, which cover one or several of deployment activities such as presented in [2] and [3]. However, these solutions are made for monolithic applications, installed only on one site, and context free.

The software system's context is any detectable and relevant attribute of the software's host devices, the software's user and the host device's surrounding environment. Context sensitivity (or context awareness) is the software's ability to sense and analyse context from various sources; it lets software take different actions adaptively in different contexts, i.e. fluctuating network bandwidth, decreasing battery power, changes in location or device capabilities.

Many research efforts have focused on finding solutions to adapt software to the changes of the context [4][5][6]. But the current deployment solutions do not use yet context-awareness and remain rather static.

Context awareness plays a significant role in the software deployment. It permits to automatically install and reconfigure a software system on the consumer site depending on the user needs and the environmental constraints.

In this article, we propose a deployment platform for the multi-component applications whose components may be distributed on different machines, this

platform will cover the various stages of deployment. We will focus our work on deployment context-awareness.

The remainder of the paper is structured as follows: Section 2 describes an application example used to illustrate the concepts throughout the paper. Section 3 presents the architectural model of our deployment solution. In Section 4, we introduce the overall architecture of the system. In Section 5, we discuss related work before concluding the paper in Section 6.

2 Application Example

To illustrate our approach, we give the example of an emergency application intended to be used in the ambulance of the future. Let us suppose that in this ambulance, the paramedics are provided by a PDA, which enables them to observe the patient state. The patient state is detected by a set of sensors placed on his body, linked to a monitoring machine. As PDAs have limited capacities, and the software medical applications are various, PDAs could not contain all the software useful for the paramedics. It should be possible to dynamically deploy these applications on PDAs in proportion to paramedic needs, its profile and to the patient state.

We require to deploy a medical software constituted by a set of components. Each component represents a medical branch e.g. cardiology component, neurology component, and pneumology component.

3 Architectural Model

The architecture model that we propose for our context-aware deployment tool has three levels (cf. Figure 1): (1) the application-level which contains application components to be deployed, (2) the meta-level which we call deployment level, and (3) the meta-meta-level which we call adaptation level.

The role of components at the meta-level consists on performing different deployment activities of application-level components: Installation, update, deinstallation, and reconfiguration activities. The role of components at the adaptation level consists on facilitating sensitivity of application-level components' deployment to the context.

Adaptation level components are interfaced with sensors which collect context information and send events to components of the deployment level indicating them the occurrence of a specific context change. These events contain generally the necessary parameters for the deployment activity such as the architecture of the software to deploy or some context information that will help to make a choice between component versions (implementations).

This reflective model plays a significant role in facilitating context sensitivity of the deployment at runtime. We detail in Section 4 the role of components belonging to the deployment and adaptation levels.

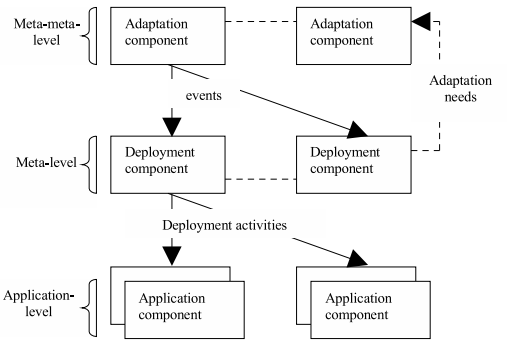


Fig. 1. Architectural Model.

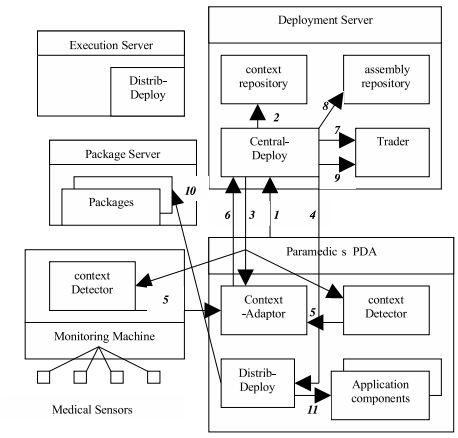


Fig. 2. Deployment steps.

4 Deployment Platform

To offer a dynamic and context-aware deployment of multi-component applications, the producer site, which offers software to be deployed, must provide a deployment platform. This section describes this platform architecture as well as the different steps of a context-aware deployment.

4.1 Deployment Platform Architecture

The platform architecture consists of nine primary components (cf. Figure 2) : The package servers, the execution servers, the assembly repository, the context repository, the trading (discovery) service, the CentralDeploy component, the DistributedDeploy component, the ContextAdaptor component and the ContextDetector component. The deployment is essentially performed by CentralDeploy and DistributedDeploy components which are meta-level components and the context-awareness is ensured by ContextAdaptor and ContextDetector components which are meta-meta-level components. In what follows, we describe the role of each component of the platform.

The package servers represent a storage space for component packages. Each package contains a different implementation version of a component dedicated to a particular execution environment.

The execution servers provide an execution environment for instantiating components of an application. If the user terminal is very loaded or has limited resources, a part of the application's components must be instantiated on these servers.

The deployment platform needs a description of the applications' structure. For this purpose, each application provides an assembly descriptor which describes the application architecture i.e. the applications' components and connections as well as deployment information such as constraints on the servers

where components may be placed after deployment. The assembly descriptors are placed in the assembly repository.

Each application has also a deployment context descriptor, which describes the context to which the deployment application is sensitive. The context descriptor is useful for creation of adaptation components i.e. ContextDetector and ContextAdaptor and it is placed in a context repository. We detail in Section 4.2 how the context descriptor of the Figure 3 is used in the deployment process. For each context, this descriptor contains its name, the name of the resource from which we can obtain the information on this context e.g. a sensor or a file, and its type that may be of two types: the 'AlwaysRelevant' context and the 'SpecificChangeRelevant' context. In the first type, any context change affects the deployment. In the second type, the deployment will be affected only after particular context changes. So, the context descriptor must indicate the conditions for which context changes will affect the deployment. Finally, the context descriptor must indicate the deployment activities generated by a context change as well as the deployment component will carry out these activities.

The trading service plays a significant role in adapting deployment to the context. It permits to find the component package, which contains the best implementation according to the context. It also allows to find an execution server to lodge components, which are not installed on the user terminal. The choice will be made according to the execution servers load and component dependencies.

The CentralDeploy component plays the role of a head of orchestra. It performs different software deployment activities: Installation, update, reconfiguration after runtime evolutions, and de-installation of the software. For each deployed application, the CentralDeploy component will instantiate a DistributedDeploy and a ContextAdaptor component in the user terminal and one or more ContextDetector components on different machines which contain context information sources (including the user terminal).

The goal of the DistributedDeploy component is to decentralise the deployment and reduce CentralDeploy tasks. There's a DistributedDeploy component installed on each execution server which performs component packages transfer, components unpacking, instantiation, and configurations on these servers.

ContextAdaptor and ContextDetector components are dynamically created from the context descriptor of the software and they have the role to adapt the deployment to the context. The number of ContextDetector components depends on the context information sources computers dispersion. These sources are not necessarily placed on the user terminal; some of them can be distributed on several computers. For our medical example, PDAs properties are accessible from the PDA. But, medical sensors are not connected to paramedics' PDAs. Sensors may be distributed over any number of computers, thus CentralDeploy will instantiate a ContextDetector on each of these computers. These ContextDetectors will detect context changes from context sources and if a relevant condition is verified, they send events to ContextAdaptor. The ContextAdaptor collects

all local and distant events and sends events to CentralDeploy or LocalDeploy components to perform the appropriate deployment.

4.2 Deployment Steps

All the deployment activities must gather information, from the producer and from the target consumer sites. In this section, we illustrate different deployment steps and the interaction between the different components of the platform with our medical example.

To ensure deployment activities, the ambulance must be equipped with the deployment platform (cf. Section 4.1). In Figure 2, we placed the trading service, the assembly and context repositories, and CentralDeploy component on the same computer that we call Deployment Server. The PDA that we presented is used, in this scenario, by a nurse speaking English. Let us describe the different steps of this interaction with the system.

1. On the nurse demand, the PDA sends a deployment request to CentralDeploy component for the medical software.

2. The CentralDeploy recovers the context descriptor of the software from the context repository. Figure 3 shows a part of our example descriptor that describes three different contexts: The patient pulse, the language used by the paramedic and the patient respiratory state. (a) Information on the pulse are recovered thanks to the sensor of pulse placed on the patient thorax and related to a monitoring machine. Pulse context is 'SpecificChangeRelevant' context: It will have effect on the deployment only if the patient pulse is greater or equal to 100. If it is the case, we must add a cardiology component to the software to deploy. The name of this component (found in the descriptor) will be sent as a parameter of the deployment activity that will be performed by CentralDeploy. (b) User Language context is however recovered from a property file situated on the user terminal. It is an 'AlwaysRelevant' context i.e. each time that the paramedic changes the language used, we must deploy again the components' versions made for this language. (c) Respiratory state context resembles the pulse context. Other contexts are described in this descriptor as the paramedic profile and his terminal properties, but they are not presented here.

3. CentralDeploy uses the context descriptor to instantiate a ContextAdaptor on the paramedic PDA, a ContextDetector on the monitoring machine to get the patient state, and another ContextDetector on the paramedic PDA to get terminal properties.

4. CentralDeploy instantiates a DistributedDeploy component on the PDA.

5. ContextAdaptor collects from the two ContextDetectors, context information. If the pulse is greater than or equal to 100, it indicates to CentralDeploy that the software to deploy have to contain a graphical interface (GUI) component and a cardiology component for English speaker nurse using a PDA.

6. CentralDeploy must initially perform installation. Firstly, it sends a research request to the trading service to find an assembly of the application which adapts best to the context : containing only a GUI and a cardiology component.

7. CentralDeploy recovers the chosen assembly descriptor from the assembly repository.

8. For each component of the assembly, CentralDeploy has to: (a) Use the trading service in order to find the best implementation which adapts to the context i.e. for a nurse and English speaker using a PDA. (b) Use the trading service in order to find an appropriate execution server to instantiate this component, if it should not be installed on the user terminal. (c) Send a request to the DistributedDeploy component placed on this execution server to locally transfer component implementation files i.e. the component package. Consequently, DistributedDeploy will unpack the package and instantiate the component.

9. CentralDeploy connects the two components instantiated as described on the assembly descriptor and activates the application.

```
<DeploymentContext>
<Context >
<ContextName>Pulse </ContextName>
<SensorId> <MachineName> MonitoringMachine </MachineName>
<SensorName> PulseSensor </SensorName></ SensorId >
<RelevantContext >
<RelevantContextName>RCPulse</RelevantContextName>
RelevantContextType>SpecificChangeRelevant</RelevantContextName>
<RelevantContextOperator>greater than or equal</RelevantContextOperator>
<RelevantContextValue>100</RelevantContextName>
<DeploymentActivityParameters >
    <MetaLevelComponent>CentralDeploy</MetaLevelComponent>
    <Assembly> <Component> CardioComponent</Component></Assembly>
</DeploymentActivityParameters >
</ RelevantContext >
</Context>
<Context >
<ContextName> UserLanguage </ContextName>
<ContextSource><MachineName> UserTerminal </MachineName>
                <File>UseProperties.txt</File></ ContextSource >
<RelevantContext >
<RelevantContextName>RCUserLanguage</RelevantContextName>
<RelevantContextType>AlwaysRelevant</RelevantContextName>
<DeploymentActivityParameters >
    <MetaLevelComponent>CentralDeloy</MetaLevelComponent>
    <ImplementationProperty>
    <UserLanguage> UserLanguageValue</UserLanguage>
    </ ImplementationProperty >
</DeploymentActivityParameters >
</ RelevantContext >
</Context>
<Context >
<ContextName> RespiratoryState </ContextName>
<SensorId><MachineName> MonitoringMachine </MachineName>
    <SensorName> RespiratorySensor </SensorName></ SensorId >
<RelevantContext >
<RelevantContextName>RC RespiratoryState</RelevantContextName>
<RelevantContextType>NotAlwaysRelevant</RelevantContextName>
<RelevantContextValue>bad</RelevantContextName>
<DeploymentActivityParameters >
    <MetaLevelComponent>CentralDeploy</MetaLevelComponent>
    <Assembly>
    <Component> RespiratoryComponent</Component></Assembly>
</DeploymentActivityParameters >
</ RelevantContext >
</Context></DeploymentContext>
```

Fig. 3. The context descriptor of the medical software.

5 Discussion and related work

The deployment domain has already been investigated during the last years. SDI [7] supports multi-component deployment and context-awareness but the SDI deployment server is actually specific to the software to be deployed whereas the CentralDeploy and LocalDeploy components that we propose are generic and can deploy any type of software. Software Dock [1] covers the totality of deployment activities but does not support multi-component applications assemblies description. It generally supports adaptation and reconfiguration but do not grant specific interest to context-awareness. [8] describes an approach that supports software component deployment onto networks of distributed and mobile devices which is based on the principles of software architectures. This approach does not support context-awareness either.

6 Conclusion

Context-awareness is a key factor for just in time deployment. This paper presents a platform covering the deployment of multi-component distributed software. This deployment is managed by components separated from software functionalities which add flexibility to our architecture. The asset of this platform is the context awareness based on a reflective model which permits to components of the deployment level to use components of the adaptation level in order to adapt their deployment activities. This model permits to modify the behaviour as well as the architecture of a software during runtime. Context-awareness is ensured thanks to an event system and a consistent description for the deployment context.

As a next step, we will work more on assembly descriptors, study in a detailed way the various activities of multi-components software's deployment and prototype the deployment platform that we propose. More precisely, we will study the treatment of the errors of deployment.

The issue of performance is central to the deployment especially in emergency domain. During the development of the architecture and prototype, we must then respect this constraint and offer a lightweight deployment and adaptation components in order to be able to place them on resource constrained devices.

References

1. Hall, R., Heimbeigner, D., van der Hoek, A., Wolf, A.: An architecture for Post-Development Configuraion Management in a Wide-Area Network. The International Conference on Distributed Computing Systems (1997)
2. Ewing, M., Troan, E.: The rpm packaging system. The first Conference on Freely Redistributable Software, Cambridge, MA, USA (1996)
3. InstallShield. (http://www.installshield.com)
4. Stephen, S., Yu, W., Fariaz, K.: Development of Situation-Aware Application Software for Ubiquitous Computing Environments. COMPSAC'02, England (2002)
5. Marmasse, L., Schmandt, C.: Location-Aware Information Delivery with ComMotion. The Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000, pages 151-171, Bristol, UK (2000)
6. Rakotonirainy, A., Indulska, J., Loke, S., Zaslavsky, A.: Middleware for Reactive Components: An Integrated Use of Context, Roles, and Event Based Coordination. IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany (2001)
7. Taconet, C., Putrycz, E., Bernard, G.: Context Aware Deployment for Mobile Users. Accepted for publication, COMPSAC'03 (2003)
8. Mikic-Rakic, M., Medvidovic, N.: Architecture-Level Support for Software Component Deployment in Resource Constrained Environments. The first International IFIP/ACM Working Conference on Component Deployment. Berlin, Germany (2002)