A Data Model for Context-aware Deployment of Component-based Applications onto Distributed Systems

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

Abstract

Recent advances in wireless networking technologies and the growing success of mobile computing devices are affecting how software deployment is being performed. Deployed applications have to be suited to the user needs, the resources of his terminal and the surrounding environment. In this paper, we propose a schema for the just-in-time deployment of component-based applications which supports context-aware adaptation. This schema presents a deployment plan which specifies how to create component instances of the application to deploy, where to instantiate them, and how they are connected to each other according to the context. This latter is intended to be used by the components performing the adaptation of the deployment to the context.

1 Introduction

Deployment refers to all activities, performed after the development of the software, which make it available to its users. These activities consist essentially in installing and configuring the software but can also include software reconfiguration, update and even deinstallation.

Component-based applications are divided into reusable components connected to each other via ports [15] [7] [10]. To deploy a component-based application, each subcomponent must first be instantiated, then interconnected and configured. The automation of the deployment tasks is very important because it saves the user from the repetitive tasks of the installation and the configuration. To automate the deployment activities, it is necessary to define a schema which provides semantic knowledge about the software to deploy.

Context-awareness is a key factor for the just in time deployment of applications because it allows the installation and the configuration of an application suited to the user needs, the resources of his terminal and the surrounding environment. Context-aware deployment becomes more important with recent advances in wireless networking technologies and the growing success of mobile computing devices, such as laptop computers, mobile phones, personal digital assistants and the like. These devices are likely to have scarce resources, such as little memory and a slow CPU, and they are usually used by mobile users. Which means the deployment has to consider the context in which the deployment process occurs.

The purpose of this paper is to define a schema in order to describe the just-in-time deployment of the component-based applications for mobile users confronted with deploying applications in different contexts. Applications’ subcomponents might be distributed among a set of nodes. Section 2 briefly introduces the requirements and issues of software deployment languages and schema and discusses some related works. Section 3 presents the different types of contexts that can affect the deployment as well as the different deployment parameters which vary according to the context. The data model for context-aware deployment is presented in Section 4. Section 5 provides a deployment scenario before concluding the paper in Section 6.

2 Background and Related Work

According to [4], a software deployment language and schema must be able to semantically describe software systems or components and consumer sites where software is to be deployed. Software description consists in specifying information about software artifacts, constraints, configuration, dependencies and specialized activities required to complete deployment. Consumer site description consists in specifying information about site properties and resources. Given such descriptions, generic deployment solutions are possi-
ible by interpreting the constraints, dependencies, and various configurations of an application with respect to the constraints and resources available at the consumer site.

Many efforts have tried to create a standard schema to describe software systems in order to facilitate deployment over networks such as the Open Software Description (OSD) [16], and the Software Dock [5]. The purpose of such software deployment language and schema is to provide semantic knowledge to support automation and simplification of the software deployment.

A component-based application is generally represented by a component assembly. Hence, languages cited above describe the schema of only one component but do not describe the overall architecture of the application in terms of a set of components and a set of connectors. ADLs (Architecture Description Language) [6] present a convenient way to specify, textually or graphically, software architectures of component-based applications. These languages are useful for tests and simulations and they are generally not designed for the deployment.

The CCM (CORBA Component Model) [10] presents a component packaging and deployment solution which specifies four descriptors:

- The Component Software Descriptor describing general information about a component and its different implementations.
- The CORBA Component Descriptor providing components characteristics used to determine the type of container in which the component needs to be installed.
- The Component Assembly Descriptor which describes a deployment plan specifying how components making up the assembly are partitioned and how they are connected to each other.
- The component property file detailing component attribute settings. These descriptors are not sufficient for the mobile setting, since they do not consider the context.

The OMG Specification of the Deployment and Configuration [11] presents a data model for the description of a deployment plan which contains information about artifacts that are part of the deployment, how to create component instances from artifacts, where to instantiate them, and information about connections between them. This specification also presents a data model for the description of the domain into which applications can be deployed as a set of interconnected nodes with bridges routing between interconnects. These data models still insufficient for the mobile setting and do not support a description of the rules achieving the adaptation of the deployment.

What the data model described in this paper does, is to add context-awareness to the previous approaches, as will be made clear in the following sections.

3 Application Deployment: A Context-aware Service

A context-sensitive or a context-aware application, is an application able to sense and analyze the context from various sources and which takes different actions suited to different contexts. We consider just-in-time deployment as a context-sensitive service, since we identify three context categories which can affect the deployment. In the following paragraphs we present these categories and we show how these contexts can affect the deployment service.

The first category presents information about the target environment into which applications can be deployed i.e. the nodes’ properties and their topology at the deployment time, such as the operating system, the free memory, the network bandwidth, the battery power, etc.

The second category presents information about the user i.e. his requirements, his preferences and his activity at the deployment time.

The third category includes many other types of contexts which are specific to the application to deploy. For instance, in an ambulance, a paramedic having a PDA, may need to deploy a medical application adapted to the patient’s state. In a commercial application, the location of the user can be used to deploy a Data Base component providing information about the stock of the nearest store.

The contexts cited above can act on five deployment variable parameters:

- The choice of the components’ implementation versions: Let us take the case of two users, one using a laptop, the other one using a PDA, who want to deploy a graphical user interface component. The context-aware deployment will install a normal screen version for the former and a small screen version for the second one.
- The components’ placement selection: If the resources of the user terminal are insufficient to install the overall application components, only a part of the components will be installed on the user terminal, the others will be installed on distant servers. These servers can be selected according to several context parameters such as the availability of their resources and the user’s location.
- The application composition structure (or architecture): The components making up the deployed
assembly vary according to the context. Indeed, several components can be specified in the deployment plan of an application but only needed components will be deployed. For example, if the network connection undergoes fluctuations, we can envisage the deployment of an additional logging component on the user’s terminal.

- The configuration property values of the components: If we have for instance a component having a property which represents the user’s language, the value of this property depends on the user’s language used at the deployment time.

- The connections between components: Establishment of connections at the deployment time depends on the context. If we have, for instance, to deploy three components A, B and C and there are possible connections between A and B, B and C and A and C, if according to the context, the connection between A and C is not necessary it must not be established. The establishment of useless connections can be bandwidth consuming when the components are placed in distant computers.

Once these deployment variables are identified, we aim at providing, in the following section, a set of rules allowing the deployment tool to act on these parameters according to the context.

4 The Data Model of the Context-aware Deployment

A data model is a model of descriptive information which can be used to generate XML schemas [17] or IDL [12] data types and structures for the purpose of using the modelled data as parameters in the runtime interfaces. To perform deployment adaptation and act on the five deployment variable parameters cited in Section 4, we propose the use of two data models detailed in the following subsections.

4.1 The Data Model of the Relevant Context Descriptors

We consider that the acquisition of the context data from sensors, and the processing of this data to obtain high-level context information, are out side the scope of this paper. For more detailed information refer to [3] and [14]. High level contexts supported by a producer site are described by a name, a reference, a method to acquire and process the context in order to obtain its high level value and the interval of time necessary to refresh the context information (cf. Table 1). The latter information is useful for deployment reconfiguration.

Before deploying an application we need to know which contexts can affect its deployment and from the context information acquired what is relevant for the deployment purpose in order to be filtered. This information type is called a relevant context information. It is specified by the application developer in what we call a ”Relevant Context Descriptor”. The data model of this descriptor is presented in Figure 1.

Each piece of relevant context information is associated to a context supported by the producer site and requires the specification of an operator such as greater, equal, smaller, etc, and a relevant value of the context. A relevant context may consist of a combination of a set of relevant contexts collected from several resources. For example, if we consider the context representing the user language, the English language relevant context, can be described as follows:

```xml
<relevantcontext id= "EnglishLanguage" contextref= "UserLanguage">
  <operator value="equal">
    <relevantvalue>english</relevantvalue>
  </operator>
</relevantcontext>
```

Information described in a relevant context descriptor will be used by a filter component [2] which receives context information and delivers only relevant context information for the deployment.

4.2 The Data Model of the Context-aware Deployment Plans Descriptors

The context-aware deployment plan describes the deployment adaptation rules which will be applied, once the relevant contexts occur. It allows a deployment adapter component [2] to act on the five deployment parameters presented in Section 4. According to the context, it provides a deployment adapter component the ability to determine the components that make up the application assembly, how to create the component instances, where to instantiate them, and how they are connected to each other.
The context-aware deployment plan describes the component instances forming the application to deploy.

General information about the component package of each component type described in the deployment plan is given in a file presented by a "ComponentFile" element. This file is an OSD [16] schema providing the component name, purpose, licence and specific implementations and requirements.

For each component instance, the context-aware deployment plan specifies (cf. figure 2):

- A set of possible implementation versions and their associated relevant contexts which require these implementations.
- A set of properties which can have different values according to the context. It is possible to map a property value to a context value.
- A set of possible destination sites and their associated relevant contexts which require the choice of these placements.

One of the hardest tasks of the context-aware deployment is the modification of the application architecture (composition structure) to be deployed according to the context because it needs the description of all its possible assemblies for all the context combinations [1]. This description becomes very complicated in the case of large applications having numerous components and sensitive to many contexts.

In order to be able to modify the architecture of the application according to the context, without describing all the possible assemblies, the context-aware deployment plan provides the ability to mention for each component instance a set of relevant contexts for which the instance will exist in the final assembly to deploy in what we call an "Existence-Context" element. Component instances whose existence context was not specified will exist in the final assembly for any context state.

The deployment plan provides also the ability to describe for each component instance a set of possible connections and their associated relevant contexts which require the establishment of these connections. The connection between two components is established by relating a "provides" port of one component to a "uses" port of the second one. The specification of a relevant context associated to a connection is at the same time used for not establishing the useless connections between the deployed components nor establishing, by error, connections with components which are not deployed.

It is possible to specify general adaptation rules to be applied on each deployed component of the application. These rules may specify a general placement for the overall components instances described in the deployment plan. For example: If a given context state obliges the deployment of all the components into the same site, it is not necessary to rewrite this rule at the description of each component instance. In the same way, all the components of an application often have joint properties. For given contexts, these properties can have the same value for all the components. The general rules specify the possible values of these properties according to the different relevant contexts.

The context-aware deployment plan allows the use of a discovery service [8] or a naming service in order to find a particular site where to instantiate a component, an implementation version or an already instantiated component by providing the ability to respectively describe discovery constraints and names associated to relevant contexts. General rules having to be applied on all types of applications are directly supported by the adaptation component provided by the deployment service provider and must not be specified in the context-aware deployment plan. Only the adaptation rules which are specific to a given application are described in this latter.

### Table 1. Context description examples

<table>
<thead>
<tr>
<th>context idref</th>
<th>context type</th>
<th>acquisition</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserLanguage</td>
<td>string</td>
<td>PreferenceAcq, getUserLang()</td>
<td>false</td>
</tr>
<tr>
<td>CPUuse</td>
<td>integer</td>
<td>NodeAcq.getProcessorUse()</td>
<td>true</td>
</tr>
<tr>
<td>UserTerminal-Type</td>
<td>string</td>
<td>NodeAcq.getNodeType()</td>
<td>false</td>
</tr>
<tr>
<td>Connection-Option</td>
<td>string</td>
<td>PreferenceAcq.getConnOpt()</td>
<td>true</td>
</tr>
<tr>
<td>UserConnection-Zone</td>
<td>string</td>
<td>Location.getUserZone()</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 2. The context-aware deployment plan data model
The data model described in this Section comprises two submodels. The first one describes the relevant contexts able to impact the deployment of an application and the second one describes the assembly of the application to deploy by specifying how to create component instances, where to instantiate them, and how they are connected to each other once the relevant contexts occur. The ability to describe discovery requests adds extensibility and dynamism to the model.

5 Application Example

We are currently implementing a filter component which uses information described in the relevant context descriptors to filter relevant contexts and an adaptation component which uses information described in the context-aware deployment plan to adapt the deployment of different applications to the context. The component platform that we are using is OpenCCM [13].

Let us take the case of a simple application having four components: a Graphical User Interface (GUI) component, a processing component, a Data Base (DB) component, and a logging component. All the processing results have to be stored in the DB. Figures 3 and 4 show an extract of the XML schema of respectively the relevant context descriptor and the context-aware deployment plan of this application.

The relevant context descriptor describes the following relevant contexts: InsufficientCPU represents the case where the CPU use is greater than 50 per cent, EconomicOption represents the case where the user does not want to pay for the network connection during the overall processing, FluctuatingConnectionZone represents the case where the user is situated in a zone where the network connection undergoes fluctuations, LittleScreen represents the case where the user uses a PDA, etc.

The relevant context descriptor allows also to combine a set of relevant contexts received from several sources thanks to relevantcontextcombination elements.

The context-aware deployment plan (cf. Figure 4) describes a general rule allowing the configuration of the value of the property representing the language of the user of all the components of the application to be deployed. This configuration is made by mapping (linking) this property to the context representing the language used. All the components will exist in the final assembly, except for the logging component, which will be deployed only if the DB component was not deployed on the user terminal and the user was in a zone where the network connection undergoes fluctuations or the user chose the economic option. Consequently, the logging component instance requires the specification of these relevant context impacting its existence in the final assembly to deploy in an "ExistenceContext" Element. Once the user finishes his work, he reconnects and the logging component will store the results of the calculations in the DB component.

The DB component will not be deployed on the user terminal if its CPU use is greater than 50 per cent. To find the nearest site to the user to deploy the DB component, the context-aware deployment plan describes a request which will be sent to a discovery service. This request allows a server whose zone is the same as the user to be found.

The GUI component has two implementations versions, the first one is dedicated to PDAs and the second one is dedicated to terminals having normal screens.

The deployment adaptation component collects the relevant contexts, combines them and decides the architecture, the placements, the implementations and configuration of the components, using this deployment plan.

6 Conclusion

Context-awareness is a key factor for just-in-time deployment. This paper defines a schema allowing context-aware adaptation of just-in-time deployment of component-based applications. This schema specifies relevant contexts for the deployment and presents a deployment plan which describes the assembly of applications to deploy by specifying how to create component instances, where to instantiate them, and how they are connected to each other according to the context.

What this model adds, compared to the existing deployment solutions of component based applications in particular the OMG Specification of the Deployment and Configuration, and the CCM deployment solution, is context-awareness. It allows the description of a set of rules in order to vary deployment variable parameters according to the context. The asset of the context-aware deployment plan that we propose is the ability to specify discovery requests which provide more flexibility in context-aware adaptation.

The context aware deployment tool that we propose is compliant with the Model Driven Architecture (MDA) [9] defined by the OMG. The data model presented in this paper constitutes, with the management model of the filter
Figure 4. Example of a context-aware deployment plan

<contextawaredeploymentplan>
<generalsecurity/>
<contextawaredeploymentplan>
<componentproperties>
<property name="UserLanguage" type="string">
<contextmapping contextref="UserLanguage"/>
</property>
</componentproperties>
<componentinstances>
<componentinstance idref="GUIinstance">
<componentfile idref="gULib.rdf">
<componentimplementation>
<impl implementref="PDAimplref" relevantcontextref="NormalScreen"/>
<impl implementref="NormalScreenimplref" relevantcontextref="NormalScreen"/>
</componentimplementation>
<destination>
<naming>
<discovery>
<namingservice name="UserTerminal"/>
</discovery>
</naming>
</destination>
<contextawareconnections>
</contextawareconnections>
</componentinstance>
<componentinstance idref="DBinstance">
<componentfile idref="DBinfo.rdf">
<componentimplementation>
<componentproperties>
</componentimplementation>
<destination>
<discovery>
<namingservice name="DBserver">
<contextmapping contextref="UserConnectionZone"/>
</discovery>
</naming>
</destination>
<contextawareconnections>
</contextawareconnections>
</componentinstance>

</componentinstances>

</contextawaredeploymentplan>

As next steps, we intend to extend the data model of context-aware deployment to allow reconfiguration of the architecture and properties of an already deployed application during runtime.

The performance issue is essential to the deployment, adaptation components must not add important deployment delays and have to be lightweight to be able to install them onto resource constrained devices.

References


