1
Introduction

The increasing use of complex technologies and large-scale software systems requires new approaches for developing software systems. Nowadays, these software systems need to become more flexible, scalable, robust, energy-efficient, configurable, and self-managed. In this context, Component-based Software Engineering(CBSE) continues gaining interest for the rapid assembly of software systems, combining concepts of software requirements engineering, architecture, design, verification, testing, configuration, and *deployment*. CBSE is defined as a process that emphasizes the design and construction of computer-based systems using reusable software components [Press01]. The notion of component is the key concept on CBSE and is encouraged by its reusability. Szyperski [Szy02] defines a software component as a unit of composition with contractually specified interfaces and only explicit context dependencies, it can be deployed independently and is subject to composition by third parties. According to this definition, developing complex software systems could be composed of a large number of heterogeneous components, offering and requiring services among themselves. It comprises the design of each component to reach a distributed, scalable and robust architecture, a reliable testing model and a valid *deployment plan*. Most of these applications are also *deployed* into distributed and heterogeneous target environments, becoming a challenge for the software industry.

Software deployment involves a set of activities that are performed to make a software system available to use. Dearle introduces several ideas concerning software deployment [Dearle07]. He defines software deployment as the processes between the acquisition and execution of software, performed by a software deployer. Whereas Heydarnoori describes software deployment as a sequence of related activities for placing a developed application into its target environment and making it ready to use [Heydarnoori08]. A typical software deployment begins with obtaining the software released, then it is installed and finally activated. Heydarnoori proposes a generic deployment process composed of ten well-defined activities: release, acquire, plan, install, configure, activate, update, deactivate, uninstall, and retire.

Heydarnoori also summarizes the main deployment techniques of component-based applications proposed by the research community. He classifies them into eight major deployment approaches. This work reviews two approaches focused on the deployment of applications into distributed environments: Model-Driven Deployment proposed by OMG and the Grid Deployment. The first, the Model-Driven Deployment is part of Model-Driven Architecture, which promotes the use of models for system specification and interoperability on the development of software systems. OMG Deployment and Configuration Specification [OMGDC06] describes metadata and interfaces to support the deployment and configuration of component-based applications into a heterogeneous distributed target environment and is compliant with MDA [Miller01]. The second, the Grid Deployment needs to deal with the complexity of grid environments. Grid environments are built upon a group of resources loosely coupled, heterogeneous, and geographically dispersed. Therefore, to take advantage of grid environments the Grid Deployment should be as automated as possible. All deployment techniques suppose a set of services grouped in a “Deployment Infrastructure” allowing users to manage all deployment activities.

Dearle argues that complexity in *deployment process* comes from interaction between the product being installed, the environment, and constraints of the execution policy. He proposes, to avoid this complexity, the creation of a perfect custom environment into which applications and components may be installed. These environments are possible using virtualization, which enables the execution of multiple operating systems sharing the same hardware. The virtualization layer is responsible for turning physical infrastructures into virtual resources, and the software that controls this layer is called a “virtual machine monitor” or “hypervisor” [RimalChLu09]. Some examples of hypervisors are Xen, KVM, VMware, and VirtualBox. They have gained great acceptance on the server virtualization market. In recent years, a new model called Cloud Computing has emerged as a new computing paradigm. Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort [NIST01].

In conclusion, applications have become more complex and they need more flexible target environments for their *deployment*. This flexibility can be described in terms of elasticity, scalability, and on-demand features. This work seeks to explore the advantages of using cloud infrastructures, a collection of hardware and software, to obtain a flexible systen in which to deploy distributed component-based applications. Additionally, we aim to use these computational resources to host the *deployment infrastructure*. The remaining of this chapter is organized as follows. Section 1.1 summarizes our proposal. Section 1.2 details the objectives. Section 1.3 describes our contributions. Finally, section 1.4 reviews the structure of the dissertation.

1.1 Proposal

We have described our main motivation to use cloud infrastructures as the target environment for the deployment of distributed component-based applications. On the one hand, the deployment process managed by a *deployment infrastructure* includes several generic activities, from packaging a component to installing in a target environment. Due to their complexity, many researches have focused on deployment intermediate activities. They are focus on increasing the performance of characteristics such as self-adaptability, self-configurability, and self-management. On the other hand, this deployment process needs a target environment to be executed. Typical target environments for deployment of applications are combinations of machines from single nodes to clusters and they are usually specified using description files. However, with the recently adoption of the Cloud Computing model, new target environments are being delivered on-demand, leveraging properties such as scalability, elasticity, flexibility, and so on. We aim to replace the typical description files of target environments with a pool of virtual machine instances in order to define a cloud-based target environment, and join it with the deployment activities. For example, once the application has been delivered to a software deployer for deployment, he writes a specification for the target environment according to its requirements ensuring some level of QoS(Quality of Service). In order to request a cloud-based target environment, the deployment infrastructure requires an overview of the cloud because it will decide whether consume or instantiate virtual machines on the cloud. Finally, the application is deployed following the activities defined by the deployment infrastructure.

Therefore, we have concentrated on two main topics: first, software deployment technologies and approaches focused on distributed component-based applications; second, the provision of computational resources provided by cloud computing.

1.2 Objectives

This work aims to combine two recent research areas: deployment of distributed component-based approaches and cloud computing. Approaches for deployment of distributed component-based applications have been designed to work with pre-defined target environments. Additionally, current deployment technologies provide limited support to define on-demand target environments. This limitation could be reduced using an *Infrastructure as a Service(IaaS)* because it allows the consumption of computational and storage resources.

Following this, our main objective proposes connection point where the deployment of distributed component-based applications and cloud infrastructures can combine. Thus, we need to provide mechanisms for the deployment process to be able to use cloud resources. These mechanisms should add capabilities to enable users to specify target environments on cloud infrastructures. Our tests comprise the deployment of a SCS MapReduce application, which allows us to assess the *deployment infrastructure* and its *target environment* deployed on the cloud. To carry out the purpose of this work we need to accomplish:

- Extension of the deployment activities of distributed component-based applications in order to acquire advantages of an elastic, scalable, and on-demand provision of computational resources.
- Set up and use a cloud infrastructure focused on providing compute and storage resources. Then, to consume these cloud resources based on user-specified deployment requirements.

1.3 Contributions

This work aims to achieve three major contributions to the deployment process of distributed component-based applications on cloud infrastructures. Our first contribution proposes a way for setting up a cloud-based target environment, which gives users the option of writing a set of parameters of computational resources. To specify these resources we adopt the use of *policies* combining them in a flexible form. For example, users could write a policy to instantiate the cloud-based target environment previously specified or only access it by using virtual machines running on the cloud. The second contribution is the design and implementation of a cloud API, where its main characteristic is the management of policies. For example, we can use both default or our own policies to request a customized set of virtual machine instances. Our third contribution, we got an experimental *Platform as a Service* as consequence of deploying the *SCS Deployment Infrastructure* [Junior09] using cloud infrastructures as its target environment. This means that we can extend it to provide a SCS-based cloud infrastructure to deploy distributed component-based applications.

1.4 Dissertation Structure

This work is organized in six chapters. Chapter 2 describes the fundamentals of Cloud Computing focused on the context of our work, and the related work studied. Chapter 3 details our reference scenario including the “Software Component System(SCS)” Component Model and its deployment infrastructure. Chapter 4 describes the challenges of our work, the elements developed to enable the use of cloud infrastructures, and our proposed architecture. Chapter 5 explains an example of the use of our proposed infrastructure using a SCS MapReduce application. Finally, chapter 6 details the conclusions and future works.