1
Introdução

Em Engenharia de Software o conceito de componentes pode assumir diferentes interpretações [1]. Uma definição amplamente aceita para *componentes de software* é dada por Szyperski [2]:

> Um componente de software é uma *unidade de composição* com, ao menos, *interfaces* contratualmente especificadas e *dependências de contexto* explícitas. Um componente de software pode ser implantado independentemente e está sujeito a composição por terceiros.

As *dependências de contexto* podem ser de dois tipos: *paramétricas* e *estáticas*, conforme Szyperski [3]. As paramétricas são aquelas resolvidas pela relação entre os serviços providos (facetas) e os serviços requeridos (receptáculos) pelo componente. Já as dependências estáticas representam os artefatos de software (bibliotecas dinâmicas, executáveis, arquivos de configuração, etc.) e os recursos específicos da plataforma (processador, memória, sensores, banco de dados, etc.). As dependências estáticas de um componente são indispensáveis à execução do componente, enquanto as dependências paramétricas, normalmente, podem ser configuradas em tempo de execução. É inevitável ter dependências estáticas, uma vez que não é prático sempre optar pela parametrização [3].

A fim de ser implantado independentemente, um componente tem que ser uma *unidade de implantação* (do Inglês, *unit of deployment*) e, em cenários dinâmicos, também tem que ser uma *unidade de versionamento e substituição* [3]. Como uma *unidade de implantação*, um componente de software deve ser um artefato pronto para execução sem intervenção humana, mediante um descritor de implantação. Por outro lado, enquanto uma *unidade de versionamento e substituição*, um componente de software deve manter-se invariante à forma como foi instalado em cada sistema. Para isso não deve carregar estados observáveis para viabilizar sua troca.

O processo de implantação de componentes envolve diversas etapas interdependentes como a publicação, instalação, ativação, atualização, reconfiguração, desativação, remoção e revogação [4]. Entre essas, a instalação, a atualização e a remoção são críticas, pois podem interferir na manutenibilidade

dos outros softwares pré-existentes [5]. Portanto, para implantar componentes de software, mesmo em uma única máquina, é fundamental sistematizar a instalação dos componentes e das suas dependências estáticas.

Em cenários distribuídos existem diversos desafios envolvidos que dificultam a implantação das aplicações baseadas em componentes [6], como: (i) heterogeneidade de hardware, sistemas operacionais e softwares dependentes; (ii) complexidade no monitoramento e na gestão dos recursos amplamente distribuídos; e (iii) escalabilidade da implantação. Esses desafios mostram que é inviável a gerência da implantação distribuída por atores humanos. Portanto, o processo de implantação precisa ter fases automáticas que sejam capazes de lidar com esses desafios e viabilizar a adoção das tecnologias de componentes nesses cenários [7].

Por exemplo, na Computação em Grade (do Inglês *grid computing*), o usuário tem acesso a um grande poder computacional amplamente distribuído e comumente utilizado em aplicações científicas como modelagem, visualização, simulações climatológicas, entre outras [8]. As tecnologias para componentes de software podem ajudar o desenvolvimento das aplicações de grade contanto que permitam fases automáticas de implantação para lidar com a grande escala de distribuição dos recursos [7]. Para um usuário da grade, a implantação automática deve permitir o monitoramento, suspensão, reinício e término da execução dos componentes, invariante aos nós onde os componentes executarão.

Este trabalho avalia algumas tecnologias de componentes como EJB [9], CCM [10], OpenCom [11] e Fractal [12] investigando como suas ferramentas de implantação enfrentam os desafios listados acima. Dentre essas tecnologias, destacam-se as ferramentas: DeployWare [6] que usa um metamodelo, uma linguagem específica de domínio e uma máquina virtual para executar os planos de implantação de componentes Fractal; e DAnCE [13] que implementa a especificação OMG D&C [14] (do Inglês, *Deployment & Configuration*) para implantar componentes CCM com suporte à configuração de QoS e conexão automática entre componentes, mediante o uso de vários descritores.

Além dessas tecnologias, foram considerados trabalhos da área da Computação em Grade, pois há pesquisas recentes [15, 16] sobre implantação distribuída. Os *middlewares* de grade avaliados neste trabalho não utilizam o paradigma orientado a componentes, mas suas ferramentas apresentam soluções interessantes para implantação distribuída e, portanto, têm semelhança com a motivação deste trabalho. Lacour et al. propõem o ADAGe [17], que usa descrições genéricas de aplicações construídas sob diferentes tecnologias (como CCM e MPI) e conta com um planejador automático de implantação. O ADAGe é destinado apenas a implantações estáticas e não conta com uma

interface de programação para gerir a implantação. Nesse sentido, Cudennec desenvolveu o CoRDAGe [18], que usa o ADAGe e coordena a expansão e contração dos planos de implantação em tempo de execução.

A partir desse amplo estudo sobre implantação distribuída observa-se que, dentre as tecnologias avaliadas, apenas o CCM especifica como descrever dependências estáticas mas não apresenta alternativas para sistematizar seu tratamento. Além disso, os sistemas de componentes avaliados não oferecem estratégias automáticas para implantação e não apresentam facilidades para o mapeamento dos componentes nas máquinas físicas.

Dessa forma, o administrador de uma aplicação baseada em componente é obrigado a lidar manualmente com os desafios impostos pelo cenário distribuído, entre eles a grande escala e a heterogeneidade dos sistemas computacionais envolvidos. Por outro lado, as tecnologias da Computação em Grade têm mostrado alternativas para automatizar tanto o mapeamento das aplicações distribuídas num conjunto de máquinas quanto a seleção dos recursos necessários à execução dessas aplicações.

1.1 Objetivos e Contribuições

Este trabalho apresenta uma *infraestrutura de implantação* que permite a implantação remota e descentralizada de componentes de software distribuídos multi-linguagem e multi-plataforma. Por multi-linguagem, entendemos que uma aplicação pode ser composta por componentes implementados em diferentes linguagens. Por multi-plataforma, entendemos que uma aplicação pode ser composta por componentes que executam em diferentes plataformas. As principais contribuições dessa infraestrutura são:

1. oferecer suporte à implantação de componentes com dependências estáticas, que podem ser diferentes a depender da linguagem e da plataforma, através do reuso de um sistema de pacotes (do Inglês, *Package Management System*);

2. oferecer ao ator da implantação uma interface de programação que:

   2.1. possa ser usada remotamente a partir de diferentes linguagens e plataformas;

   2.2. implemente todas as fases de um processo de implantação;

   2.3. simplifique a manipulação do ambiente de execução do sistema de componentes;

2.4. permite o mapeamento dos componentes na infraestrutura de execução e nas máquinas envolvidas através de níveis de detalhamento;

2.5. permite o planejamento e a implantação automática dos componentes em um parque de máquinas;

2.6. permite a implementação de novas estratégias automáticas para implantação.

É importante destacar que este trabalho oferece uma interface de programação para linguagens imperativas ao invés de uma optar por uma abordagem declarativa. A decisão por uma abordagem imperativa foi motivada pela possibilidade de aproveitar recursos presentes na maioria dessas linguagens, como estruturas de controle condicional, laços e funções. Entende-se que o uso desses recursos facilita a construção de roteiros (planos) de implantação dinâmicos que manipulem os componentes mesmo em tempo de execução. Por outro lado, outros trabalhos que sigam abordagens declarativas, como aqueles que usam Linguagens de Descrição de Arquitetura [19, 20] (do Inglês, *Architecture Description Language - ADL*) podem usar a interface de programação oferecida por este trabalho para implantar os seus componentes.

Uma outra contribuição é o amplo estudo sobre as atuais metodologias usadas na implantação de componentes distribuídos. A fim de organizar esse estudo, este trabalho define critérios de classificação que abrangem diversos aspectos relevantes à implantação distribuída. Esse estudo engloba não apenas tecnologias de componentes mas também algumas ferramentas da área de Computação em Grade, que oferecem suporte à implantação distribuída.

1.2 Estrutura do Documento

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta o estudo e classificação dos trabalhos relacionados. O Capítulo 3 apresenta uma visão geral do sistema de componentes utilizado e sua infraestrutura de execução já existente. O Capítulo 4 explica o suporte a dependências estáticas e sua relação com os sistemas de pacotes. O Capítulo 5 detalha a infraestrutura de implantação proposta. O Capítulo 6 apresenta exemplos de uso da infraestrutura de implantação. Por fim, o Capítulo 7 traz as considerações finais e as indicações de trabalhos futuros.