# 1
# Introdução

A construção de aplicações distribuídas baseadas em componentes de software vem ganhando espaço por propor um alto nível de reuso de artefatos de software. Originalmente, a idéia de componentes de software estava associada à produção de software em massa, onde os componentes representam uma forma modular de agrupar uma família de rotinas [1]. Atualmente, um componente pode ser definido como uma unidade de implantação cujas funcionalidades possuem uma interface pública bem definida e cujas dependências com o ambiente são obtidas a partir de uma interface de requisição [2, 3].

Tradicionalmente, o suporte à programação orientada a componentes está associado a frameworks disponibilizados para linguagens orientadas a objeto. O J2EE/EJB [4], o Julia/Fractal [5], o OpenCom [6] e o SCS [7] são exemplos de tecnologias baseadas em componentes de software.

Linguagens de programação orientadas a objeto já oferecem diversas abstrações, como interfaces, classes, herança e polimorfismo, que facilitam a decomposição das aplicações em módulos reusáveis. Entretanto, aplicações baseadas em componentes requerem outros tipos de abstrações que não estão presentes nas linguagens orientadas a objeto, como por exemplo a definição do próprio componente, do conjunto de serviços publicados por ele, das dependências necessárias para o seu funcionamento e de mecanismos próprios para manipulação das conexões entre componentes. Tipicamente, modelos de componentes usam abstrações das próprias linguagens OO para definir uma interface de programação para construção e uso dos componentes, criando frameworks e adotando padrões de projeto específicos. Como consequência, o código fonte mistura aspectos da funcionalidade do componente com os mecanismos de implementação específicos do modelo de programação, o que dificulta a reutilização deste componente em outros frameworks, além de incluir uma complexidade extra no código.

Vários trabalhos têm sido propostos com o objetivo de prover mecanismos e abstrações mais apropriados para a programação orientada a componentes, geralmente procurando isolar os aspectos funcionais do componente dos mecanismos de programação impostos por um modelo de componentes

em particular. A utilização de técnicas como a orientação a aspectos [8, 9], a utilização de metadados [10, 11], a definição de uma linguagem própria para componentes [12], e a utilização de arquivos externos ao código para introduzir relações com elementos do modelo de componentes, são exemplos de abordagens comumente adotadas por tais trabalhos.

Recentemente, observamos uma tendência à adição de metadados às implementações dos componentes, utilizando marcações específicas no código fonte. O SCA [13] e o Fraclet [14] são exemplos de trabalhos que adotam esta prática. Estes metadados visam prover as informações necessárias para que alguma ferramenta, seja baseada em geração de código ou em mecanismos de reflexão computacional da própria linguagem de programação, realize a integração da implementação do componente com a infra-estrutura de suporte do modelo de componentes. Alguns autores denominam esta abordagem de Programação Orientada a Atributos [10, 11]. Linguagens de programação como Java [15] e C# [16] já oferecem suporte nativo à técnica de programação orientada a atributos através das *Anotações* [10, 11].

O objetivo desta dissertação é investigar a adoção da técnica de programação orientada a atributos juntamente com uma linguagem orientada a objetos para construção de aplicações baseadas em componentes. Como parte do estudo, foi desenvolvido um novo mecanismo de programação baseado em atributos para a versão Java do middleware SCS [7].

O SCS [7](Software Component System) é uma infra-estrutura para dar apoio à implementação, instalação, carga e execução de componentes de software em um ambiente distribuído. Diferentemente de outros trabalhos, o SCS visa reunir em sua arquitetura apenas as funcionalidades essenciais para a criação de um sistema orientado a componentes, disponibilizando interfaces pequenas e simples de usar.

Atualmente, o SCS já é utilizado como ferramenta de ensino e para experimentação de novas técnicas relacionadas a componentes de software no nosso grupo de pesquisa [17, 18, 19]. Sendo assim, o SCS é a opção natural para utilizarmos como base deste trabalho.

O novo mecanismo desenvolvido, que funciona como um novo modelo de programação para o SCS, recebeu o nome ASCS (Annotated Software Component System). O ASCS é um modelo de programação baseado em anotações para a criação de componentes SCS. O desenvolvedor interage com uma camada de anotações para mapear os elementos do modelo de componentes com as construções do código Java.

O modelo de programação ASCS é equipado com um mecanismo de validação em tempo de compilação. Tal mecanismo, que é baseado na ferra-

menta APT [20](Annotations Processing Tool), permite que a maioria das dependências ocultas da API sejam eliminadas. Desta forma, o desenvolvedor do componente pode identificar de forma antecipada um possível erro estrutural do componente, antes mesmo do término de sua codificação. Assim, o tempo gasto com a identificação e correção de erros estruturais é reduzido.

Por fim, foi realizado um estudo analítico-comparativo entre a *API* proposta pelo modelo de programação original do SCS com a proposta pelo ASCS, ressaltando as vantagens e desvantagens de cada uma das abordagens. Para essa avaliação utilizamos, através de uma abordagem qualitativa, o CDN (Cognitive Dimensions of Notations Frameworks) [21], adaptando suas dimensões para a avaliação de *API's*. Também foi realizada, através de uma abordagem quantitativa, uma análise referente à diferença de quantidade de linhas de código em tarefas semelhantes nos dois modelos de programação.

Além do estudo relativo à *API* dos modelos de programação, também foi feita uma comparação quantitativa, entre os dois trabalhos, referente à diferença de desempenho para carga, conexão e acesso a componentes remotos.

Para estudo de caso e apoio na avaliação proposta, foi criada uma aplicação distribuída simples que implementa uma sala de bate-papo permitindo a conexão e comunicação de múltiplos usuários simultaneamente. Neste exemplo podemos ver como tipicamente se dá o uso de todas as anotações do ASCS, assim como a relevância das vantagens proporcionadas pelo novo modelo de programação, demonstradas em uma aplicação completa.

Esta dissertação está organizada da seguinte forma. O próximo capítulo apresenta alguns trabalhos relacionados ao tema desta dissertação. O capítulo 3 descreve a infra-estrutura SCS, identificando as abstrações presentes no seu modelo de componentes e detalhando seu modelo de programação. O capítulo 4 apresenta o ASCS, o novo modelo de programação do SCS que define um conjunto de anotações para serem usadas pelo desenvolvedor de componentes. A seguir, o capítulo 5 traz os exemplos, e as avaliações relativas à adoção de anotações para construção de aplicações baseadas em componentes, usando como base o novo modelo de programação implementado. Por fim, o capítulo 7 apresenta as conclusões e possíveis trabalhos futuros.