1
Introdução

Sistemas de software estão presentes em diversas partes das nossas vidas. Atualmente eles podem ser encontrados em lugares comuns do dia-a-dia, como automóveis, elevadores, equipamentos hospitalares, etc. Estamos cada vez mais dependentes do funcionamento correto desses sistemas e isso implica em uma exigência natural por softwares mais robustos e confiáveis. Por isso, buscamos incessantemente aperfeiçoar o processo de desenvolvimento desses sistemas.

Nessa procura por um processo de desenvolvimento mais eficiente, a área de pesquisa de metodologias de programação tem provido diversos resultados. Alguns autores se referem a essas metodologias como Paradigmas de Programação. Uma rápida busca na Internet revela inúmeros paradigmas diferentes, por exemplo, programação orientada a objetos, programação orientada a aspectos, programação baseada em componentes e meta-programação. Provavelmente não existe uma metodologia melhor que as outras, mas sim, a mais adequada para um dado problema.

Particularmente a programação baseada em componentes tem se mostrado interessante no desenvolvimento de sistemas distribuídos. Essa metodologia tem atraído atenção de desenvolvedores por todo o mundo, pois permite que desenvolvedores construam sistemas de software complexos de forma rápida e estruturada.

A literatura existente sobre o assunto é rica e diversificada. Diversos modelos de componentes já foram propostos no meio acadêmico, como por exemplo, EJB [1], CCM [2], Fractal [3], COM [4] e OpenCom [5]. A ideia principal dessa técnica é a construção de sistemas maiores a partir da composição de artefatos de software menores [6]. A adoção dessa prática enfatiza naturalmente a separação de interesses.

Frequentemente, essa composição é descrita com a utilização de uma linguagem de descrição arquitetural. Nessa descrição estão informações sobre como os diversos componentes interagem entre si. As interações podem ser locais (mesmo processo) ou distribuídas (processos/máquinas diferentes), e são realizadas através de Conectores de Software [7].

Segundo *Metha et al* [8], esses conectores se manifestam em sistemas

PUC-Rio - Certificação Digital Nº 0912899/CA

de software como acessos a variáveis compartilhadas, *buffers*, instruções para o *linker*, chamadas de procedimentos, protocolos de rede, *pipes* e assim por diante. Nesse mesmo texto os autores descrevem e classificam diversos tipos de Conectores de Software conhecidos.

O significado do termo Conectores de Software geralmente causa dúvidas entre profissionais da Ciência da Computação. Comumente falta uma distinção da relação entre conectores de alto nível e conectores do nível de implementação. Além disso, às vezes conectores são modelados como componentes, tornando o significado ainda mais confuso. Neste trabalho, o termo Conectores de Software segue a definição dada por *Shaw e Garlan* [9]:

> *"Conectores mediam as interações entre os componentes, isto é, estabelecem as regras que governam a interação de componentes e especificam quaisquer mecanismos auxiliares necessários."*

Mais especificamente, conectores neste trabalho fazem parte de um componente, mas não são um componente por si só.

Em sistemas de componentes distribuídos os Conectores de Software mais utilizados são os baseados em *Remote Procedure Call* (RPC) [10]. Geralmente, implementações desses modelos utilizam um middleware de comunicação para alcançar esse objetivo. Um exemplo é o *CORBA Component Model* (CCM) [2] construído sobre o padrão *CORBA* [11]. No entanto, grande parte dos modelos de componentes distribuídos não possui suporte a Conectores de Fluxos de Dados (CFD). Quando esse tipo se mostra necessário, os desenvolvedores frequentemente implementam soluções específicas para o sistema que está sendo desenvolvido, sem modificar o atual modelo de componentes. Em especial, o CCM possui uma proposta de extensão em andamento que especifica Conectores de Fluxo de Dados [12].

Sistemas distribuídos geralmente são compostos por diversos computadores que interagem entre si para atingir um objetivo em comum, como por exemplo, disseminação de informação. Diversos autores, como *Drucker* [13], sugerem que a sociedade esteja migrando para uma economia baseada no conhecimento, ou seja, baseada na produção, distribuição e utilização do conhecimento. Sugerem também que a tecnologia da informação provavelmente seja uma das forças motrizes por trás dessa nova economia. É fácil notar que a conectividade provida por redes de computadores como a Internet tem facilitado a comunicação e disseminação da informação. No entanto, o tamanho e a quantidade das informações distribuídas tem crescido com o tempo. É comum assistirmos vídeos e escutarmos música em nossos computadores ou dispositivos móveis. Os sistemas de software e as metodologias utilizadas precisam estar habilitados para lidar com esses diversos tipos de comunicação.

PUC-Rio - Certificação Digital Nº 0912899/CA

Conectores de Fluxo de Dados possuem uma grande relevância em sistemas que lidam com transmissão de áudio, vídeo e dados de sensores, devido a natureza dessas informações. Também se mostram importantes em alguns sistemas que trabalham com aquisição, filtragem e armazenamento de dados. A dinâmica da interação através desse tipo de conector difere consideravelmente da interação via *RPC*. Na primeira, uma grande quantidade de dados é transmitida e recebida de forma contínua. Na segunda, os dados são transmitidos seguindo um padrão de requisição e resposta. Essa diferença normalmente influencia na implementação e no modelo de execução de um componente.

Este trabalho foi motivado originalmente por uma demanda de uma aplicação comercial que utiliza o framework CSBase [14], um software utilizado para a construção de ambientes de grades personalizados. A aplicação em questão possui um módulo executor de fluxos de algoritmos, no qual, dado um arquivo de dados inicial, o usuário pode planejar uma sequência de algoritmos por qual esses dados devem ser executados. Esta aplicação permite conectar a saída de um algoritmo na entrada de outros. Atualmente ela utiliza arquivos e *pipes* para realizar a comunicação entre os diferentes algoritmos. No entanto, esta forma de comunicação limita a execução de fluxos a apenas uma máquina. Existe um grande interesse dos desenvolvedores e usuários em utilizar essa infraestrutura para execuções distribuídas.

Analisando a questão, pensamos em transformar a aplicação acima em uma aplicação SCS [15, 16], transformando cada algoritmo em um componente SCS e aproveitando o trabalho recente de implantação desenvolvido por *Barbosa* [17]. Porém, na época, não havia uma maneira de modelar no SCS a forma de comunicação utilizada por esses algoritmos. Então, um estudo sobre conectores que permitissem esse tipo de comunicação foi iniciado, resultando na ideia de estender o modelo de componentes do SCS com Conectores de Fluxo de Dados.

O SCS foi escolhido por três motivos principais: a familiaridade do autor com o código do middleware, por ser utilizado em sistemas comerciais onde a necessidade de conectores de fluxo de dados foi identificada e por ser um modelo utilizado em diversas pesquisas acadêmicas.

1.1 Objetivos e Contribuições

Tendo esse cenário como motivação, o objetivo deste trabalho foi estudar e implementar o suporte a Conectores de Fluxo de Dados no modelo SCS. Para avaliar as funcionalidades e a utilização do resultado obtido, uma aplicação

PUC-Rio - Certificação Digital Nº 0912899/CA

de execução de fluxos de algoritmos distribuídos baseada nesses conectores foi desenvolvida. A intenção é que essa aplicação sirva de modelo para guiar uma futura implementação no CSBase. Além disso, também foi realizada uma avaliação de desempenho que procurou comparar os resultados da solução desenvolvida com os resultados de outros métodos de transferência de dados e, com isso, validar objetivamente a aplicabilidade da implementação.

São duas as contribuições esperadas como resultado deste trabalho. A primeira é a extensão do modelo de componentes SCS com dois tipos de conectores, aumentando sua expressividade e permitindo seu uso em aplicações com requisitos de comunicação através de fluxos de dados. A segunda é o relato das dificuldades e decisões tomadas, que podem servir como base para projetos futuros. Espera-se que o modelo desses novos conectores promova o reuso de bibliotecas de fluxo de dados e forneça uma maior flexibilidade para aplicações construídas com o SCS.

1.2 Estrutura do Documento

Os próximos capítulos estão organizados da seguinte forma: o capítulo 2 analisa trabalhos relacionados com o objetivo desta pesquisa. O capítulo 3 apresenta uma visão geral do modelo de componentes SCS e as características particulares dos Conectores de Fluxos de Dados, detalhando os aspectos que influenciaram as decisões de projeto tomadas. O capítulo 4 começa enumerando os requisitos levantados para o suporte de CFDs e segue ilustrando o novo modelo de componente proposto. Ainda nesse capítulo, são apresentados detalhes e decisões da implementação dos conectores no modelo estendido. O capítulo 5 faz uma avaliação sobre a implementação com um estudo de caso e medidas de desempenho. Por fim, o capítulo 6 apresenta as conclusões e possíveis trabalhos futuros.

PUC-Rio - Certificação Digital Nº 0912899/CA