3
SCS: Sistema de Componentes de Software

O mecanismo para acompanhamento das chamadas remotas se baseia em informações coletadas durante a execução da aplicação. Para a coleta dessas informações é necessário realizar a instrumentação da aplicação. Neste trabalho utilizaremos a infra-estrutura de monitoramento dos componentes SCS [14] (Sistema de Componentes de Software) em razão de sua simplicidade e flexibilidade. Essa infra-estrutura facilita a instrumentação e pode ser facilmente adaptada às necessidades específicas da aplicação.

3.1
O modelo do componente SCS

O modelo de componentes de software do SCS presa a separação entre a especificação de um serviço, representada pela sua interface, e a sua implementação. Essa separação permite definir as dependências de um serviço como uma coleção de interfaces. Dessa forma, um componente de software é um artefato de software com definições explícitas dos serviços que oferece e dos serviços das quais depende.

Esse modelo é seguido por um componente SCS, que possui uma coleção de interfaces bem definidas que representam os serviços oferecidos, e outro conjunto de interfaces que representa as dependências que devem ser satisfeitas para que os serviços oferecidos possam ser utilizados. As interfaces do componente são chamadas de *facetas* e as dependências são representadas pelos *receptáculos*. Cada receptáculo de um componente poderá ser associado a uma faceta de outro componente em um processo denominado conexão, desde que a interface oferecida pela faceta seja compatível com interface esperada pelo receptáculo. Esse processo de conexão representa a dependência de um componente sendo suprida pelo serviço oferecido por outro componente.

O sistema de componentes SCS utiliza o padrão CORBA para comunicação entre os componentes, de forma que cada componente possui uma coleção de facetas e uma coleção de receptáculos. Cada uma dessas facetas implementa um conjunto de métodos que podem ser invocados e cada um dos receptáculos pode ser conectado a uma faceta, de forma que os métodos im-

plementados por esta possam ser utilizados. A figura 3.1 apresenta o esquema de um componente SCS.

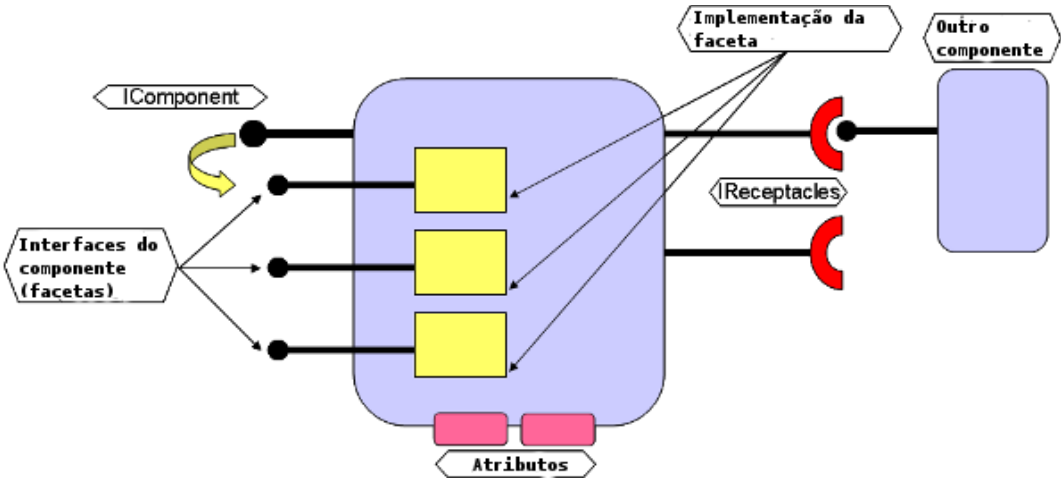

Figura 3.1: Modelo de componente SCS

As aplicações distribuídas baseadas em componentes que usam o SCS podem fazer uso de uma infra-estrutura de execução [15] que conta com um mecanismo de monitoramento de execução, proposto por Fonseca [16]. Esse mecanismo permite a definição de um conjunto de métricas que serão coletadas durante a execução do sistema e disponibilizadas em um canal de eventos para que possam ser monitoradas ou analisadas por um consumidor conectado ao canal.

3.2 Infra-estrutura de execução

A infra-estrutura de execução do SCS permite o carregamento, execução, suspensão e instrumentação dos componentes. Ela é composta pelos seguintes elementos principais: *ExecutionNode* e *Container*.

O *Container* é um processo do sistema e executa um ou mais componentes SCS que compartilham um mesmo espaço de endereçamento. O *ExecutionNode* gerencia os *Containers* que executam em uma mesma máquina e pode criar e destruir *Containers*. Em uma aplicação distribuída, um *ExecutionNode* tipicamente está associado a uma máquina e representa um nó da rede.

3.3 Infra-estrutura de monitoramento

O SCS conta com uma infra-estrutura de monitoramento que permite a coleta de dados a partir dos *Containers* que hospedam os componentes da aplicação. Os dados a serem coletados são descritos por meio de métricas.

Algumas métricas, consideradas gerais o suficiente para serem úteis para uma grande parte das aplicações distribuídas, são oferecidas de antemão por essa infra-estrutura:

- Tempo total de utilização de CPU pelo *Container* desde que foi criado.
- Total de memória utilizada pelo *Container*.
- Tempo total de execução do *Container*.
- Tempo de resposta de uma operação invocada (medido no servidor).
- Tempo de propagação na rede da requisição para invocação de uma operação.
- Tempo de uso da CPU pelo processo que executou a operação invocada.

O levantamento dessas métricas foi realizado por Fonseca [16] tendo como base uma criteriosa análise de alguns aspectos de sistemas distribuídos.

No entanto, em alguns casos a análise de uma aplicação vai exigir a coleta de uma métrica específica, relacionada ao domínio da aplicação, e que não foi prevista pela infra-estrutura de monitoramento. Por exemplo, a análise de uma aplicação que manipula grandes volumes de dados armazenados em arquivos exigiria a coleta de dados que possam quantificar a utilização do disco.

Para esses casos em que as métricas oferecidas não são suficientes, essa infra-estrutura apresenta a flexibilidade necessária para ser adaptada ao tipo de aplicação a ser instrumentada. Essa adaptação é feita pelo desenvolvedor através da definição de um conjunto de métricas consideradas relevantes para sua aplicação.

Para que essas novas métricas possam ser coletadas junto daquelas já definidas, o programador deve associar cada nova métrica à um método de coleta de dados, e depois, plugá-la à infra-estrutura de monitoramento. É importante mencionar que tais métricas podem ser adicionadas e retiradas em tempo de execução.

A coleta de algumas métricas, como o tempo de CPU do processo que executou uma operação invocada, é feita pela infra-estrutura de monitoramento através do uso dos interceptadores CORBA. Como o uso desses interceptadores também é necessário neste trabalho para a coleta dos dados necessários ao acompanhamento das sequências de interações entre os componentes, nós iremos estender os interceptadores da infraestrutura de monitoramento para incluir a coleta desses dados. Assim, vai ser possível fazer a associação de uma sequências de interações com algumas métricas, como por exemplo, o tempo de CPU utilizado por um método remoto.

A instrumentação para o monitoramento de uma aplicação causa um inevitável impacto no desempenho dessa aplicação. Uma solução para minimizar esse problema seria colocar o componente responsável pelo tratamento dos dados coletados em um nó da rede não utilizado pelos componentes da aplicação. Para isso, a infraestrutura de monitoramento disponibiliza as métricas coletadas em um canal de eventos CORBA, para que possam ser obtidas por qualquer aplicação cliente conectada ao canal. Assim, qualquer processamento adicional não vai onerar os nós da rede em que rodam os componentes da aplicação.

3.3.1 Os interceptadores definidos por CORBA

A arquitetura CORBA permite o registro de um objeto que pode atuar em conjunto com o ORB no fluxo que envolve a requisição e a resposta de uma chamada remota. Esse objeto é denominado de *interceptador*, podendo ser um interceptador cliente, que é chamado quando uma aplicação cliente CORBA realiza uma requisição de chamada remota, ou um interceptador servidor, quando uma aplicação servidora CORBA recebe uma requisição para invocação de método remoto.

O ORB invoca operações do interceptador em quatro pontos de interceptação ao longo do caminho de invocação entre cliente e servidor, que podem ser vistos na figura 3.2. No lado do cliente são definidos dois pontos de interceptação: Um no momento em que o cliente envia uma requisição (*SendRequest*) e outro quando ele recebe a resposta de uma requisição feita (*ReceiveReply*). No lado servidor os pontos de interceptação são: *ReceiveRequest*, no momento na qual o servidor recebe a requisição e *SendReply*, quando o servidor processou a requisição e vai enviar uma resposta para o cliente.

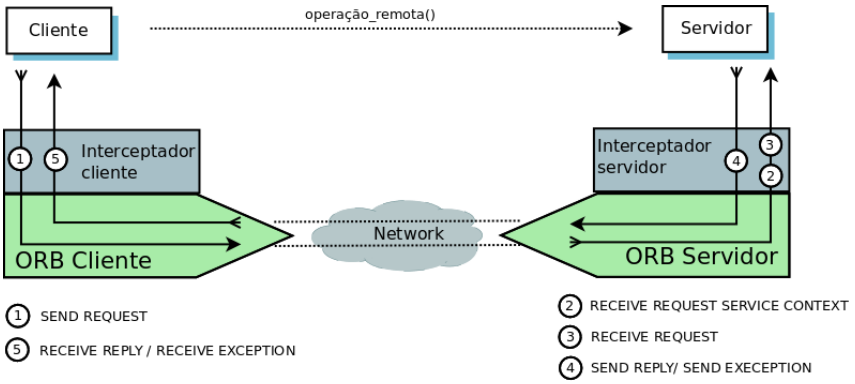

Figura 3.2: Pontos de interceptação

Existem outros pontos de interceptação para o caso de ocorrência de uma exceção: *SendException*, quando o servidor vai enviar a exceção para o cliente e o *ReceiveException* quando o cliente recebe como resposta a exceção

levantada. O objeto que implementa o interceptador possui métodos que o ORB vai invocar em cada ponto de interceptação.

Em tempo de execução o interceptador pode examinar o estado da requisição e decidir que ações realizar com base nas informações associadas com a invocação, por exemplo, o nome da operação, parâmetros, lista de exceções, identificador da requisição e valores de retorno. O interceptador não pode modificar os parâmetros ou valores de retorno, mas pode inserir ou extrair dados de uma lista de contexto de serviço da requisição. O contexto de serviço é um campo do tipo *sequence* em uma mensagem GIOP, que pode transmitir informação adicional associada à requisição, tais como credenciais de autenticação, contexto de transações ou prioridade das operações. Cada entrada no contexto de serviço tem um único identificador que pode ser usado pelas aplicações e pelos componentes CORBA para extrair o contexto de serviço apropriado.

A especificação de interceptadores do CORBA permite modificar o comportamento das aplicações sem mudanças no código e pode ser utilizada para desenvolvimento de aplicações adaptativas. Pode ser utilizada também para o monitoramento da utilização dos recursos pela aplicação, para a coleta de informações para depuração em tempo de execução e para o registro (*log*) de informações cuja necessidade só foi detectada após a implantação da aplicação. Uma vantagem é que não é necessário mudança no código da aplicação. Porém existe um impacto negativo no desempenho da aplicação, como mostrado por Baldoni e Marchetti [17], já que um código adicional será executado a cada invocação remota.

Uma alternativa para superar as limitações quanto ao desempenho do uso de interceptadores é a utilização de *proxies* modificados para inclusão de código de instrumentação [17]. Dessa forma, seria possível filtrar os métodos que seriam interceptados, minimizando o impacto global no desempenho. No entanto, a ferramenta desenvolvida neste trabalho deve reconstruir a sequência de chamadas de todos os métodos da aplicação, sendo assim necessário a interceptação de todas as requisições remotas. Por isso a abordagem utilizando *proxies* seria mais trabalhosa, além de não ser tão vantajosa na questão do desempenho, já que seria necessário inserir código em todos os *proxies* da aplicação.