3
SCS: Modelo de Componentes e Infraestrutura de Execução

O desenvolvimento deste trabalho está apoiado sobre o uso e extensão da infraestrutura proposta por Augusto [42]. Nesse trabalho, Augusto usa um sistema de componentes distribuídos conhecido como SCS (*Software Component System*) [43] que é construído sobre o *middleware* de comunicação CORBA [27]. A compatibilidade do SCS com a versão 2 de CORBA oferece aos componentes a capacidade de interoperar com componentes de terceiros que podem ser desenvolvidos em diferentes linguagens ou plataformas, além de facilitar a integração de um componente SCS com aplicações CORBA já existentes, como, por exemplo, os serviços padrões *CosEvent e CosTrading*.

Este capítulo descreve brevemente as duas partes principais do SCS: o *modelo de componentes* e a *infraestrutura de execução*. De forma análoga à classificação dos trabalhos relacionados, aplicaremos os mesmos critérios definidos na Seção 2.2 para orientar a discussão sobre o suporte previsto no SCS para implantação distribuída. Essa discussão permitirá o entendimento das soluções tecnológicas dos Capítulos 4 e 5 deste trabalho.

3.1
Modelo de Componentes

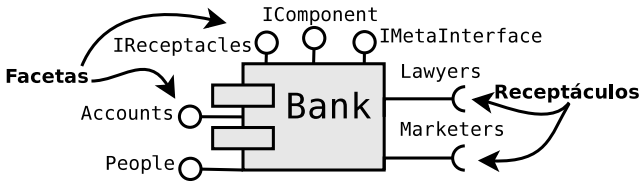

Figura 3.1: Exemplo de componente SCS com facetas e receptáculos

O modelo de componentes do SCS foi inspirado no modelo de componentes do Microsoft COM [44] e da OMG CCM [10], contudo tenta evitar a complexidade imposta por esses modelos [42]. O modelo do SCS define *facetas* (portas de provimento de serviço) e *receptáculos* (portas para serviços requeridos). O modelo permite a interação, configuração e introspecção através de três facetas básicas, ilustradas na Figura 3.1: (i) *IComponent*, que permite a identificação, ativação e desativação do componente; (ii) *IReceptacles*, que

gerencia as conexões entre componentes (remotos ou locais); e (iii) *IMetaInterface*, que provê operações para introspecção. O projeto do SCS fornece uma implementação do modelo e uma biblioteca de apoio à programação para as linguagens Lua [45], Java e C++. Essas implementações usam, respectivamente, as seguintes implementações CORBA: OiL [46], JacORB e Orbix.

O desenvolvimento de um componente SCS consiste em: (i) especificar as interfaces em OMG IDL que representarão as facetas e os receptáculos; (ii) implementar essas interfaces em uma das linguagens suportadas; e (iii) implementar um código que use o método *newComponent* presente na respectiva biblioteca de apoio. O modelo do SCS não obriga que as implementações das facetas e receptáculos implementem uma interface básica. Isso proporciona um menor acoplamento de código e simplifica a implementação do componente.

As bibliotecas de apoio disponibilizam o método *newComponent* para instanciar o componente. Esse método espera: (i) um identificador para o componente; (ii) descritores das facetas (apelido, nome da interface em OMG IDL e nome do artefato da linguagem que a implementa); e (iii) descritores dos receptáculos (apelido e nome da interface em OMG IDL). O identificador de um componente (conhecido como *ComponentId*) deve ser único e é formado por um nome, uma versão e um campo para detalhes da plataforma (conhecido como *platform_spec*) que é útil para diferenciar implementações de um mesmo tipo de componente.[1] Na prática, uma faceta é apenas um *objeto* CORBA, portanto basta que o serviço a ser provido esteja descrito na forma de interfaces OMG IDL. Um receptáculo, por sua vez, é uma estrutura que agrupa um ou mais *proxies* CORBA, os quais referenciam facetas de mesma interface em outros componentes. A associação entre um receptáculo e uma ou mais facetas é representada por uma *conexão* que é gerida pela faceta *IReceptacles*.

3.2 Infraestrutura de Execução para Componentes

Augusto [42] desenvolveu uma *infraestrutura de execução* para aplicações distribuídas baseadas em componentes. Essa infraestrutura permite a instanciação, configuração, suspensão, interceptação e execução de componentes SCS e é formada pelos seguintes componentes, ilustrada na Figura 3.2:

– *Container*: representa um processo de sistema que disponibiliza um espaço de memória comum a um ou mais componentes SCS e com a responsabilidade de controlar sua carga e descarga. Adicionalmente, também pode monitorar os componentes, provendo facilidades para interceptação, suspensão e resumo da invocação de métodos remotos.

[1] Entende-se por "tipo do componente" o conjunto de facetas e receptáculos.

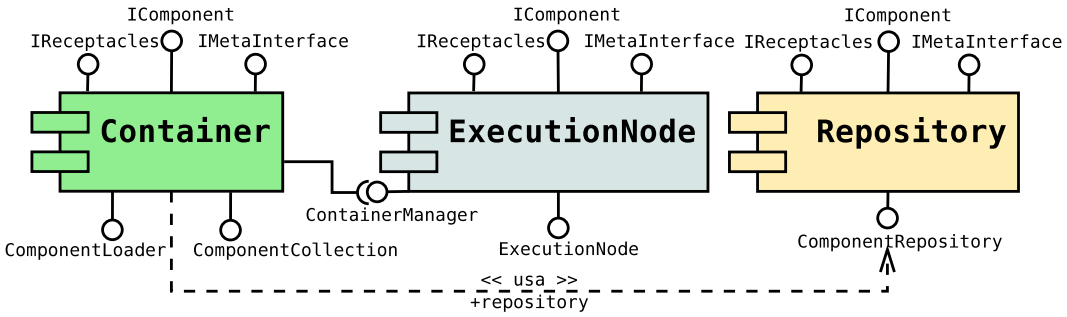

Figura 3.2: Componentes da infraestrutura de execução para aplicações SCS

- *ExecutionNode*: representa um nó na rede e atua como porta de entrada na máquina física. Esse componente é responsável por gerenciar a criação dos *Containers* e controlar o acesso desses aos recursos físicos locais.
- *Repository*: armazena implementações de componentes que podem ser publicados e obtidos remotamente. Esse componente é usado pelo *Container* para que este tenha acesso às implementações dos componentes e assim possa carregar os componentes solicitados.

É interessante ressaltar que essa infraestrutura de execução pode ser administrada remotamente, pois todas as facetas desses componentes são, na verdade, objetos CORBA.

Diferente de outras abordagens como EJB ou CCM, um *Container* da infraestrutura de execução do SCS permite a carga de diferentes componentes, não obrigando que o componente seja de apenas um tipo. Esse recurso é uma flexibilidade interessante ainda que possa trazer um impacto na segurança, pois componentes danificados podem causar a parada do processo do *Container*. Entretanto, Augusto [42] prevê o uso de serviços adicionais para controle de acesso podem ser combinados aos componentes da infraestrutura de execução.

É importante notar que apenas o *Container* precisa estar programado nas diferentes linguagens. Os componentes *ExecutionNode* e *Repository* podem estar programados em apenas uma linguagem. Atualmente o *ExecutionNode* e *Repository* estão programados em Lua, usam o ORB OiL e podem ser executados em diversas plataformas (sejam baseadas em Windows ou Unix).

3.3 Análise do Suporte Previsto à Implantação

O modelo do SCS não define o que é uma unidade de implantação. O uso da infraestrutura de execução considera a unidade de implantação como um pacote de componente (formato ZIP) que agrupa os artefatos binários da implementação (.lua, .class, .jar, .dll, .so). A fim de organizar a carga dos

componentes nos *Containers*, o programador deve implementar uma fábrica de componentes que encapsule o uso do método *newComponent* da biblioteca de apoio e realize as configurações mais elementares (caso necessárias) antes de permitir que o componente seja ativado. Durante a publicação no *Repository* é preciso fornecer o identificador do componente, o pacote do componente e o nome do artefato que age como a fábrica de componentes. Assim, ao solicitar a instanciação de um componente no *Container* solicita-se automaticamente a obtenção da unidade de implantação a partir do *Repository*.

O ator da implantação precisa coordenar manualmente as entidades da infraestrutura de execução, proposta por Augusto [42], para implantar seus componentes SCS. Da mesma forma que na maioria das outras tecnologias, o SCS não oferece suporte a instalação das dependências estáticas e assume que o programador ou administrador deve cuidar manualmente dessa questão. No SCS a instalação de implementações é tarefa do *Container*, que obtém os códigos binários a partir do *Repository*, salva-os localmente e então procede à carga do componente. Essa abordagem mostra-se inadequada, pois a instalação deixa de ser um procedimento padrão, precisa ser reprogramada desnecessariamente por todos os implementadores de *Containers* e prejudica, assim, a extensibilidade da infraestrutura de execução. Por outro lado, a simplicidade é um ponto forte da infraestrutura de execução do SCS. O componente *Repository*, por exemplo, é uma entidade interessante no ambiente, pois é possível mantê-lo permanentemente disponível a fim de potencializar seu reuso em todas as fases do ciclo de vida dos componentes (projeto, implantação e execução), como Lau e Wang [24] afirmam ser o ideal. Por fim, a Tabela 3.1 resume a classificação do SCS segundo os critérios propostos na Seção 2.2.

Critério Classificatório	**Avaliação**
Controle da Implantação	dinâmico por acesso direto
Suporte à Composição	execução
Repositório de Implementações	**acesso:** remoto **uso:** dinâmico **fase:** projeto, implantação, execução
Abstração da Distribuição	manual
Modelo de Dependências	paramétricas
Abrangência Tecnológica	**linguagem:** Lua, Java e C++ **acesso:** serviço próprio

Tabela 3.1: Classificação do SCS incluindo o uso da infraestrutura de execução