3
Sistema de Componentes de Software

A tecnologia de componentes de *software* é vista como uma extensão da tecnologia de *software* orientado a objeto que se preocupa com os quesitos de interoperabilidade e reutilização. Não existe na comunidade de *software* uma única definição para termo componente de *software* (33).

Uma definição amplamente aceita é de Szyperski: (33)

> Um componente de *software* é uma unidade de composição com interfaces contextualmente especificadas e somente dependências explicitas de contexto. Um componente de *software* pode ser implantado independentemente e está sujeito a composição por terceiros.

Outra definição existente é de Councill: (06)

> Um componente de *software* é um elemento de *software* que segue um modelo de componente e pode ser independentemente implantado e composto sem modificação de acordo com um padrão de composição.
>
> Um modelo de componentes define as interações e as normas de composição. O modelo de implementação de componente é um conjunto específico de elementos de *software* executáveis necessários para apoiar a execução de componentes que estejam em conformidade com o modelo.
>
> Uma infraestrutura de componentes de *software* é um conjunto de *softwares* interativos concebidos para assegurar que um sistema ou subsistema de *software* construído usando seus componentes e as suas interfaces irão satisfazer as especificações de desempenho definidas.

As duas definições concordam que os componentes podem ser compostos por terceiros e são independentes. A primeira definição é mais específica a respeito das interfaces e dependências, a segunda deixa estas características a serem tratadas pelo modelo de componente utilizado. A principal diferença

entre elas é que a segunda menciona composição sem modificação, para esta finalidade o modelo de componente deve definir mecanismos de customização que descrevem como os componentes podem ser estendidos sem modificação (06).

Como exemplos de modelo de componentes de *software* se destacam: COM (19), JavaBeans (31), e CORBA (24). COM, *Component Object Model*, é o padrão desenvolvido pela Microsoft, que, mais recentemente, introduziu um novo padrão de componentes conhecido como .NET (20), que utiliza ideias de COM e Java. A Sun Microsystems desenvolveu o JavaBeans e o Enterprise JavaBeans (32) baseados na sua linguagem de programação Java. O CORBA, *Common Object Request Broker Architecture*, é uma especificação de componentes distribuídos multiplataforma, ou seja, suporta a interação de componentes implementados em diferentes linguagens, distribuídos em uma rede com diferentes sistemas operacionais.

Um componente pode ser formado por objetos de uma ou mais classes, assim, ele pode ser mapeado em um conjunto de objetos. Ou qualquer outro artefato de programação. Isso mostra o nível de abstração da abordagem de componentes e de *software* e a sua força.

3.1 SCS: Modelo de Componentes

O desenvolvimento deste trabalho está apoiado sobre o uso de um sistema de componentes distribuídos conhecido como SCS (*Software Component System*) (34) (01) que é desenvolvido sobre o *middleware* de comunicação CORBA. A compatibilidade do SCS com a *middleware* CORBA permite aos componentes a possibilidade de interagir com componentes de terceiros que podem ser construídos em linguagens ou plataformas diferentes. Neste estudo entendemos que componentes de terceiros podem ser componentes reais de um radar.

Este capítulo descreve brevemente as duas partes principais do SCS: o modelo de componentes e a infraestrutura de execução.

3.1.1 Modelo de Componentes

O modelo do sistema de componentes do SCS define facetas , que são acessos para serviços fornecidos, e receptáculos, que são acessos para serviços solicitados. O modelo disponibiliza três facetas que possibilitam a interação, configuração e introspecção, essas facetas são: *IComponent*, que permite a identificação, ativação e desativação do componente; *IReceptacles*, que controla as conexões entre componentes; e *ImetaInterface*, que provê o acesso às facetas

e receptáculos do componente. O projeto do SCS disponibiliza implementações para as linguagens Lua (12), Java e C++.

A interação entre os componentes é baseada na conexão dos receptáculos com as facetas. Através dos receptáculos os componentes acessam os serviços oferecidos por outros componentes sem a necessidade de se criar conexões estáticas. De fato, um componente não referencia diretamente a um componente alvo, mas solicita o serviço ao seu receptáculo. Então o componente tem uma visibilidade autônoma através de seu receptáculo que pode se conectar dinamicamente com as facetas que proveem os serviços requeridos.

Para se desenvolver um componente SCS deve se descrever suas interfaces em OMG IDL (24) (do Inglês, *Interface Description Language*), que retratam as suas facetas e os seus receptáculos, implementar essas interfaces em uma das linguagens suportadas, implementar um código que crie uma instância dos componentes, e atribuir-lhe um identificador único que é composto por um nome, uma versão e um campo para detalhes da plataforma.

Para criar uma instância de um componente é necessário carregá-lo por meio de seu IOR (do Inglês, *Interoperable Object Reference*) ou utilizando a faceta *ComponentLoader* de um *container* de componentes, para tal é necessário passar o identificador do componente; os descritores das facetas que são os nomes das interfaces em OMG IDL que o componente disponibiliza; e os descritores dos receptáculos que são os nomes das interfaces em OMG IDL que o componente requer.

3.1.2 Infraestrutura de Execução

O SCS disponibiliza uma infraestrutura de execução distribuída para os componentes. Essa infraestrutura possibilita a instanciação, configuração, suspensão e execução dos componentes.

Ela é composta pelos componentes: *Container*, que é responsável pela carga e descarga dos componentes, representa um processo comum a um ou mais componentes SCS e permite carga de diferentes tipos de componentes; *ExecutionNode*, que é responsável por criar os *Containers* e gerenciar os acessos aos recursos locais, ele representa um acesso à máquina física e atua como um nó na rede; e *Repository*, que é um repositório de componentes que podem ser armazenados e requisitados remotamente.