3

Adaptação Dinâmica

Adaptação dinâmica é a capacidade de um sistema ser modificado durante sua execução para se adequar a novas necessidades. Recentemente, esse tem se tornado um tópico de pesquisa proeminente [5, 18]. Neste capítulo, é feita uma introdução a alguns conceitos relacionados à adaptação dinâmica, além de uma discussão sobre algumas de suas técnicas. Esses conceitos serão utilizados para caracterizar as formas de adaptação tratadas neste trabalho.

Um sistema de computação é adaptado através de alterações em sua estrutura. Essas alterações podem ser feitas em níveis diferentes, caracterizando graus diferentes de flexibilidade. Neste trabalho será feita a distinção entre adaptações de acordo com o nível onde a adaptação é feita. Serão chamadas *adaptações em ponto grande*, as adaptações que resultem em alterações na forma como os componentes da aplicação são interconectados (*i.e.*, na estrutura macro do sistema), como por exemplo, a forma como os objetos de um sistema orientado a objetos ou os módulos de um sistema modular se relacionam. Em contrapartida, serão chamadas *adaptações em ponto pequeno*, as adaptações que resultem em alteração nos próprios componentes da aplicação (*i.e.*, na estrutura interna dos componentes), como por exemplo, alterações de suas interfaces ou implementações. Adaptações em ponto pequeno fornecem um nível maior de detalhes, fornecendo maior flexiblidade, porém com maior complexidade. Além disso, adaptações em ponto pequeno geralmente resultam em adaptações em ponto grande. Note que essa classificação é relativa ao que é considerado componente do sistema, não sendo portanto uma classificação precisa.

3.1

Reflexão Computacional

A capacidade de alterar dinamicamente sistemas de computação não deixa de ser uma forma de desenvolvimento de aplicações. Por essa razão, é necessário que os mecanismos e modelos de adaptação dinâmica apresentem abstrações adequadas a esse tipo de programação. Nesse sentido, uma alternativa é a utilização de mecanismos de *reflexão computacional*[19], que é a capacidade de um sistema de se manipular, ou seja, de manipular sua própria implementação. Essa manipulação é feita através de mecanismos de

introspeção, através dos quais o sistema pode observar e analisar sua própria estrutura, e mecanismos de *intercessão*, através dos quais o sistema pode intervir e alterar sua própria estrutura (*i.e.*, implementação).

Mecanismos de introspeção são geralmente fornecidos através de interfaces de programação, que permitem ao sistema obter informações sobre sua estrutura ou sobre a estrutura de seus componentes. Um exemplo disto é a API de reflexão de Java, que permite que a aplicação possa obter informações sobre a estrutura de classes e objetos que a compõem. Outro exemplo é o repositório de interfaces da arquitetura CORBA, que permite obter informações sobre a interface oferecida por um determinado objeto.

Mecanismos de intercessão são geralmente fornecidos através de *pontos de variação*[4], que definem pontos da estrutura de execução do sistema que podem sofrer algum tipo de variação introduzindo uma alteração. Essa variação pode ser feita alterando o elemento sendo executado (*e.g.*, uma função ou procedimento) ou através do uso de *interceptadores*, que são elementos que interceptam a execução no ponto de variação para executar um outro elemento (*e.g.*, executando uma função antes da execução do código original). O uso de interceptadores permite uma alteração menos destrutiva, uma vez que não elimina o elemento original da execução, mas também exige um nível de indireção maior, que pode afetar o desempenho do sistema. Já a alteração definitiva do elemento de execução é mais adequada a alterações permanentes, como correções de erros na implementação de um sistema.

3.2

Adaptação em Ponto Grande

Neste trabalho será chamada adaptação em ponto grande a restruturação dos componentes que formam a estrutura macro do sistema. Esses componentes podem ser entidades diferentes dependendo do modelo adotado. Por exemplo, em um sistema orientado a objetos, os componentes podem ser instâncias de classes (*i.e.*, objetos) que possuem relacionamentos entre si para compor o sistema. Já num sistema baseado em componentes distribuídos (*e.g.*, EJB, CCM), os componentes que formam a aplicação são os próprios componentes.

Como as adaptações em ponto grande são feitas num nível mais alto, elas são limitadas às estruturas daquele nível. Por exemplo, na restruturação de um sistema baseado em componentes é necessário respeitar as estruturas de cada componente, ou seja, seus serviços e dependências. Para alterar as estruturas de cada componente, é necessário que a alteração seja feita num nível menor, ou seja, dentro de cada componente.

Para utilizar adaptações em ponto grande, é necessário que a estrutura do sistema seja projetada tendo em vista possíveis restruturações. Para tanto, é necessário modularizar adequadamente o sistema, separando suas responsabilidades e funcionalidades em componentes distintos que possam ser reorganizados para realizar alterações. Por essa razão, a adaptação através da restru-

turação é geralmente utilizada em virtude de alterações esperadas. Em especial, essa é uma abordagem muito utilizada na construção de aplicações auto-adaptáveis, ou seja, aplicações que se adaptam automaticamente em virtude da percepção de alterações esperadas em seu ambiente de execução [20, 21].

3.2.1

Padrões de Projeto

Padrões de projeto [3] são modelos para serem aplicados em situações recorrentes no projeto de sistemas computacionais. Tais modelos visam primordialmente a flexibilização do sistema através de estruturas de classes e objetos formadas para esse fim. Essa flexibilidade é dada pela capacidade dessas estruturas de acomodarem reconfigurações que podem, por exemplo, alterar o objeto que recebe determinada mensagem ou fazer com que a mensagem seja recebida por vários objetos.

Com o uso de mecanismos que permitam carregar código dinamicamente (*e.g.*, DLL's, classes Java, etc.), é possível adicionar um maior grau de flexibilização aos padrões de projeto, pois assim novas funcionalidades podem ser inseridas à aplicação. Contudo, ainda assim, é necessário estruturar o sistema de forma a acomodar certas alterações [22].

3.2.2

Sistemas Baseados em Componentes

A modularização e o baixo acoplamento do modelo de componentes de software[4] se adequam bem ao modelo de adaptação em ponto grande. Em sistemas baseados em componentes, essa forma de adaptação pode ser feita através da substituição de componentes ou apenas reconectando-os de forma a alterar suas interações.

Substituição de Componentes

Como tipicamente cada componente é responsável por implementar uma determinada funcionalidade do sistema, a substituição de componentes permite alterar a forma com que cada funcionalidade do sistema é implementada. A substituição de componentes é o mecanismo de adaptação mais fundamental, pois efetivamente é capaz de mudar o comportamento do sistema, através da alteração na forma como o sistema é implementado. O componente substituído pode corrigir uma versão anterior, assim como apresentar uma implementação mais eficiente.

Entretanto, a substituição de componentes ainda é dependente da arquitetura do sistema. É necessário que o sistema seja projetado adequadamente, ou seja, é necessário que as funcionalidades sejam adequadamente separadas em componentes distintos e que as interdependências sejam bem definidas de

forma que essa estrutura não se altere com o tempo. A substituição de um componente por outro mais funcional, que forneça serviços adicionais ou semanticamente diferentes, ou inclusive apresente dependências divergentes do componente anterior, resulta em alterações em outras partes do sistema, através de outras substituições ou inclusão de novos componentes.

A substituição de componentes tem sido aplicada com sucesso na manutenção de sistemas através de substituições por novas versões [23]. Muitos modelos de componentes de software distribuídos apresentam recursos que permitem que essa substituição seja feita de forma dinâmica. O modelo de componentes de CORBA (apresentado na seção 2.2) permite instanciar novos componentes, assim como desfazer e estabelecer dinamicamente conexões entre eles.

Um dos grandes problemas da substituição de componentes é a transferência de estado. Quando um componente que apresenta estado é substituído por outro, o estado do componente antigo deve ser transferido para o novo componente. Entretanto, a transferência do estado pode introduzir a possibilidade de inconsistência, pois essa transmissão nem sempre é atômica com a substituição do componente propriamente dita.

Reconexão de Componentes

A reconexão de componentes tem por objetivo alterar a forma de interação entre os componentes. A substituição de um componente geralmente é feita através de uma reconexão de componentes, mas a reconexão também pode ser feita para inserir novos componentes ou desviar o fluxo das interações entre componentes. Particularmente, a simples reconexão dos componentes de um sistema (*i.e.*, sem inclusão de novos componentes) é um mecanismo de adaptação ainda mais limitado, pois exige uma arquitetura especialmente projetada para acomodar certas reconfigurações.

Suponha um componente de acesso a uma rede de transmissão de pacotes que dependa de um outro componente para fazer a cobrança sobre o número de pacotes recebidos. Alterando a conexão entre o componente de acesso e o componente de cobrança é possível alterar a forma de cobrança. Entretanto, as possíveis formas de cobrança devem ser previstas no projeto, pois as interações entre os dois componentes não devem ser alteradas. Suponha agora que se deseje permitir que a cobrança seja feita sobre o tipo do dado contido no pacote. Nesse caso é necessário definir novas interações entre os componentes. Uma alternativa seria fazer com que o componente de acesso avaliasse o tipo do dado e repassasse essa informação para o componente de cobrança através de uma nova forma de interação (*e.g.*, novo canal). Outra alternativa seria introduzir um novo componente de avaliação de pacotes que fosse responsável por classificar cada pacote transmitido e informasse isso ao componente de cobrança. Em ambos os casos seria necessário alterar a estrutura das interações entre os componentes, ou seja, as interconexões entre os componentes.

Alguns trabalhos [24, 25, 26] abordam que uma separação entre as ta-

refas relacionadas com as interações entre componentes, (*i.e.*, coordenação), e o processamento realizado pelo componente (*i.e.*, computação propriamente dita), seja a chave para adaptação de sistemas baseados em componentes, dado que o processamento das *regras do negócio* sejam mais estáveis do que o comportamento resultante das interações entre os componentes (*e.g.*, a seqüência em que as etapas são realizadas, etc.). Nesse caso, as adaptações são feitas através da alteração na coordenação, ou seja, reconexão entre os componentes e alteração do código relacionado às interações entre os componentes que são feitas através das conexões. Entretanto, não há uma clara distinção entre o processamento relacionado à coordenação e o relacionado à computação. Caso o sistema sendo desenvolvido não faça a distinção apropriadamente ou caso a própria computação seja muito sujeita a alterações, então é necessário fazer a substituição de componentes ou uma adaptação num nível menor (*i.e.*, ponto pequeno).

3.3
Adaptação em Ponto Pequeno

A adaptação em ponto pequeno é uma restruturação do sistema num nível menor, ou seja, em estruturas mais fundamentais, dando um maior grau de flexibilidade, uma vez que a adaptação pode atingir mais detalhes da estrutura do sistema. Assim como as adaptações em ponto grande, as estruturas manipuladas durante a adaptação podem variar de acordo com o modelo e com o nível de detalhe adotado. Por exemplo, uma linguagem de programação reflexiva pode adotar estruturas de fluxo de controle e variáveis como elementos capazes de serem manipulados pelo programa em tempo de execução. Entretanto, esse nível de granularidade não parece ser apropriado por expor muitos detalhes do sistema e outros modelo mais adequados são utilizados em mecanismos de adaptação em ponto pequeno.

3.3.1
Nível da Linguagem de Programação

A adaptação em ponto pequeno pode se dar através da alteração na implementação do sistema por meio de recursos de reflexão da linguagem em que o sistema está implementado. Em linguagens que apresentam funções como elementos de primeira classe (*e.g.*, Lisp e Lua), é possível substituir as funções que implementam os componentes do sistema, caracterizando uma alteração em ponto pequeno. Entretanto, a substituição de implementação geralmente é irreversível, o que não é adequado em adaptações temporárias.

Na linguagem orientada a aspectos AspectJ [27] é possível definir interceptadores que são associados à leitura de atributos ou a chamadas de métodos em classes ou objetos Java. O código executado pelos interceptadores definem um *aspecto* do sistema e, juntamente com a implementação de objetos e classes, compõem o comportamento do sistema. A troca dos aspectos caracteriza uma

alteração na implementação do sistema e portanto uma forma de adaptação em ponto pequeno.

3.3.2

Nível Arquitetural

Alguns trabalhos propõem o uso de arquiteturas com recursos de reflexão computacional, denominadas de *arquiteturas reflexivas* [28, 29]. As arquiteturas reflexivas representam um meio termo entre a adaptação em ponto grande e a adaptação em ponto pequeno, pois permitem realizar adaptações restruturando a arquitetura da infra-estrutura sobre a qual o sistema computacional é construído (*e.g.*, middleware). Entretanto, aqui classificaremos arquiteturas reflexivas como mecanismos de adaptação em ponto pequeno pois a adaptação é feita nas estruturas que formam os componentes da aplicação, ou seja, em subcomponentes.

Arquiteturas reflexivas têm sido utilizadas no desenvolvimento de middlewares adaptativos, ou seja, ambientes para construção de aplicações adaptáveis através de alterações no próprio middleware sobre o qual a aplicação foi construída. Como os middlewares fornecem abstrações e mecanismos que atendem certos requisitos não-funcionais como distribuição, persistência, segurança, eficiência, etc. as adaptações feitas através de middlewares adaptativos geralmente alteram características não-funcionais do sistema [30, 31].

3.3.3

Nível dos Componentes

O modelo de componentes de software define um nível estrutural maior que o das linguagens de programação, pois um componente de software é uma entidade menos fundamental que funções ou objetos. O componente de software é uma entidade com funcionalidades e responsabilidades específicas e portanto promove uma melhor modularização do sistema. Por essa razão, componentes são considerados uma boa unidade de construção de sistemas [4]. Isso faz com que a adaptação no nível estrutural dos componentes de uma aplicação se mostre como uma alternativa bastante natural.

Para realizar alterações nas estruturas de um componente é necessário definir mecanismos de reflexão que permitam manipular essa estrutura, ou seja, é preciso definir um modelo de componentes reflexivo. Através de um modelo de componentes reflexivo, é possível construir componentes que possam ser alterados dinamicamente. Neste trabalho, chamaremos esses componentes de *componentes dinâmicos*. A estrutura de um componente é dada pelos serviços que oferece e pelas conexões que pode estabelecer (*i.e.*, suas dependências). Logo, um modelo de componentes reflexivo deve apresentar os seguintes recursos:

Mecanismos de introspeção para obter informações sobre a estrutura do componente, assim como manipulá-la (*e.g.*, estabelecer conexões).

Mecanismos de intercessão que permitam alterar a estrutura do componente (*e.g.*, adicionar conectores), assim como sua implementação (*e.g.*, alterar pontualmente a implementação de um serviço).

Além dos mecanismos de adaptação em ponto grande utilizados em sistemas baseados em componentes, como substituição e reconexão, o uso de um modelo de componentes reflexivo na construção de sistemas adaptáveis permite uma forma de substituição parcial do componente, uma vez que nem toda implementação do componente precisa ser alterada. Essa substituição parcial permite também que o componente alterado mantenha seu estado, pois se a alteração não interfere no estado o componente, esse estado é preservado. A alteração de um componente também pode ocasionar uma reconfiguração, pois a nova estrutura do componente pode permitir o estabelecimento de novas conexões. Além disso, o uso de componentes dinâmicos fornece recursos para alteração da coordenação dos componentes, pois permite alterar suas conexões, assim como o código relacionado às interações entre eles.