7
Conclusões

O desenvolvimento de aplicações adaptáveis traz novos desafios em relação ao desenvolvimento de software convencional. Em parte, isso está relacionado às diferentes características das diversas técnicas de adaptação dinâmica. Por exemplo, é necessário definir o grau de flexibilidade das adaptações, assim como os impactos no desempenho do sistema. Além disso, a definição de alterações em tempo de execução tornam a complexidade desse tipo de sistema ainda maior. Portanto, a construção de aplicações dinamicamente adaptáveis deve ser auxiliada através de ferramentas adequadas.

Neste trabalho é apresentado o LOAF, que é um *framework* composto pelo modelo de componentes reflexivo LuaCCM e por ferramentas que utilizam esse modelo para fornecer recursos de adaptação dinâmica através de diferentes abordagens. Por exemplo, através dos recursos de papéis e protocolos, é possível realizar adaptações em ponto pequeno, que oferecem uma maior flexibilidade através de alterações na estrutura interna dos componentes, permitindo alterar sua definição e implementação. Entretanto, adaptações em ponto pequeno geralmente têm um impacto no desempenho do sistema, particularmente devido ao grau de indireção e verificações em tempo de execução que costumam ser introduzidas. Portanto, em alguns casos, adaptações através de reconfigurações são mais adequadas. Para tanto, o LOAF fornece o recurso de manipuladores, que facilitam a definição de reconfigurações através da linguagem Lua.

Para mostrar a flexibilidade das aplicações construídas através das ferramentas do LOAF, foram apresentados exemplos de uso através de alterações num sistema em execução. As alterações apresentadas permitiram adicionar novos recursos ao sistema sem alterar sua implementação (sincronização de fluxo), assim como definir novos recursos de forma intrusiva, ou seja, interagindo com a implementação dos componentes do sistema (depurador distribuído). Além disso, também foi mostrado como o recurso de adaptações em níveis pode ser utilizado para adicionar um novo comportamento em todas as instâncias de um componente (replicação passiva).

Todos os recursos do LOAF apresentados nesse trabalho foram implementados satisfatoriamente num protótipo. Esse protótipo foi utilizado em testes de interoperabilidade com outros componentes CCM implementados em C++ através do MICO CCM [38], permitindo criar sistemas de componentes formados por componentes dinâmicos LuaCCM e outros componentes CCM. Além disso, o protótipo também permitiu comprovar a aplicabilidade do mo-

delo através da implementação dos exemplos de uso apresentados no captítulo 6.

Entretanto, apesar da flexibilidade e modularização das alterações feitas através do LOAF, ainda há pontos em aberto. Em especial, uma das limitações do LOAF está relacionada as alterações em componentes LuaCCM. Como as operações da interface de adaptação do LuaCCM permitem alterar apenas uma porta do componente por vez, não há atomicidade nas alterações feitas por um papel, ou seja, entre a adição de duas portas num componente é possível que este receba alguma requisição. Isso gera problemas, pois se os segmentos de duas portas são dependentes (*e.g.*, um acessa campos do outro), uma das novas portas pode receber uma requisição antes que a outra seja adicionada ao componente.

Além disso, o mecanismo de adaptação em níveis não abrange mais de um servidor de componentes. Todas as adaptações no nível da definição de um componente são limitadas a um único servidor. Apesar de ser possível aplicar um mesmo papel em diversos servidores, o LOAF não garante que essas adaptações sejam feitas de forma atômica, o que pode resultar em instâncias de um mesmo componente com versões diferentes. Uma maneira de evitar que esse tipo de inconsistência seja percebida pelos clientes é alterar a versão do componente apenas através da definição de novas portas e não através da re-implementação de portas existentes. Dessa forma, a versão anterior do componente é mantida juntamente com as novas funcionalidades, que se tornam disponíveis através das portas adicionadas. Entretanto, uma alternativa mais adequada seria utilizar protocolos que permitissem executar transações de forma atômica, para disparar adaptações em servidores diferentes, de forma que dentro de cada servidor, o mecanismo de herança do LOOP permita que as adaptações sejam refletidas em todas as instâncias dentro daquele servidor.

Outra questão importante relacionada ao LuaCCM é a complexidade inerente do modelo de componentes adotado, que está diretamente relacionada à pretensão do modelo CCM ser um modelo de componentes satisfatoriamente completo. Essa complexidade vai de encontro à filosofia apresentada neste trabalho e adotada por outros projetos como o LuaOrb e a própria linguagem Lua. Entretanto, este trabalho não está focado necessariamente em simplificações ao modelo CCM, mas na introdução de mecanismos de adaptação dinâmica simplificados e na avaliação do modelo CCM quanto ao suporte a esse tipo de adaptação. Isso resultou numa implementação consideravelmente fiel ao modelo CCM originalmente proposto pela OMG. Todavia, acreditamos que o LuaCCM possa inspirar simplificações no modelo CCM resultantes de facilidades fornecidas pela linguagem Lua, apesar desse não ser o foco deste trabalho.

Apesar dessas limitações, o LOAF apresenta recursos importantes para o estudo de adaptação dinâmica de aplicações. Por exemplo, os mecanismos reflexivos do LuaCCM permitem alterar a implementação de componentes sem perda ou alteração do seus estado interno, evitando assim a necessidade de transferência de estado na substituição de componentes. Além disso, os recursos do LOAF fornecem um ambiente de experimentação onde é possível

investigar e validar abstrações e modelos de adaptação dinâmica, de forma similar a abordagem utilizada para validar a utilização do modelo de papéis e protocolos no modelo CCM.

7.1 Trabalhos Relacionados

Grande parte dos mecanismos de adaptação apresentados na literatura são relacionados a adaptações em ponto grande, através de reconfigurações, utilizando diversas abordagens como discutido em [39]. A linguagem Lua tem sido utilizada como linguagem de configuração para aplicações baseadas em componentes na arquitetura CORBA [29, 40, 41, 36]. Entretanto, o modelo de objetos de CORBA apresenta deficiências em relação à reconfiguração [41]. Um exemplo disso é a falta de definição explícita das dependências de um componente do sistema. As ferramentas propostas neste trabalho estendem a linguagem Lua para facilitar a sua utilização como linguagem de configuração do modelo de componentes de CORBA, que tenta solucionar alguns problemas da utilização de CORBA, em particular problemas relacionados a reconfiguração.

Entretanto, as vantagens da utilização de linguagens de *script* na configuração dinâmica de componentes distribuídos já é conhecida, inclusive a OMG especifica uma linguagem de *script* baseada na sintaxe de IDL denominada IDLScript[42]. Em [43] é apresentada uma arquitetura para implantação dinâmica de componentes CCM através de uma implementação da linguagem IDLScript, denominada CorbaScript[44]. Nessa arquitetura é definido o conceito de *servidor de contâiner genérico*, que basicamente permite obter a implementação de um componente através de *Web Services*, sendo que estes componentes são implementados de forma especial, ou seja, utilizando um compilador de IDL projetado especificamente para tornar a implementação portável entre diferentes ORBs, porém com algumas limitações. Desta forma a linguagem CorbaScript é utilizada como um mecanismo para definição de roteiros de implantação mais flexíveis que o modelo de implantação de CCM através de pacotes de componentes e arquivos de descrição de montagem. Entretanto, diferentemente da linguagem Lua, a linguagem IDLScript não fornece mecanismos de extensão que permitam fornecer novos recursos específicos para o modelo CCM, como é o caso da ferramenta de manipuladores proposta pelo LOAF. Além disso, o LOAF pode ser estendido para fornecer ferramentas para implantação de componentes, que implementem toda a arquitetura definida pelo modelo CCM. Isso permitiria utilizar a linguagem Lua como um mecanismo flexível e dinâmico de implantação de componentes em qualquer servidor que seguisse o padrão CCM .

Em [45], é proposto um mecanismo para construção de objetos CORBA dinâmicos, ou seja, objetos que são capazes de ter sua implementação alterada em tempo de execução. Essa alteração é feita no nível dos métodos oferecidos pelo objeto, ou seja, a adaptação é feita substituindo a implementação de cada operação do objeto. O LuaCCM estende esse trabalho através da inclusão dos

conceitos do modelo CCM, que se propõem a solucionar diversos problemas da arquitetura CORBA. Em particular, da mesma forma que conceitos como contêiners e *homes* do modelo CCM fornecem mecanismos de padronização de uso e disponibilização de recursos da arquitetura CORBA, esses mesmos conceitos são utilizados no LuaCCM como mecanismos para utilização e a disponibilização dos recursos fornecidos pelo modelo de objetos dinâmico. Como exemplo, temos o recurso de adaptação em níveis do LuaCCM que é feita através de um contêiner (nível da definição do componente) ou de um *home* de componentes. Além disso, o modelo CCM define um novo mecanismo de extensão através do conceito portas, o que permite fornecer recursos de adaptação num nível menos granulado que o nível das operações dos objetos dinâmicos. O recurso de portas do modelo CCM também é utilizado para estender componentes de forma que estes forneçam os recursos de adaptação dinâmica, como é caso da faceta adaptation dos componentes LuaCCM.

A flexibilidade da linguagem Lua e o uso de objetos dinâmicos permitem a construção de arquiteturas de sistemas dinamicamente adaptáveis, como é apresentado em [20]. Tais arquiteturas utilizam o recurso do *Lua Monitor*, que é um mecanismo de monitoramento extensível implementado em Lua, e o conceito de *smart proxies*, que permite redirecionar requisições de um servidor para outro. Através dos recursos de papéis do LOAF, esse modelo de arquitetura pode ser estendido para permitir a inclusão do recurso de monitoramento dentro dos próprios componentes, permitindo que adaptações sejam disparadas em conseqüência de alterações tanto no contexto de execução da aplicação, como também no próprio estado interno dos componentes. Além disso, o modelo de conexões explícitas de CCM permite que a tarefa de redirecionamento de requisições feitas pelos *smart proxies* seja feita através de reconexões das portas do componente (*e.g.*, receptáculos).

Os conceitos de papel e protocolo implementados pelo LOAF são propostos por [14], onde eles são implementados num *middleware* baseado em Java chamado Comet. O modelo de componentes do Comet define que as conexões entre componentes sejam feitas através de canais de eventos assíncronos, através dos quais é possível definir outras formas de comunicação [46], inclusive síncronas. Além disso, os canais de comunicação de um componente são identificados unicamente pelo tipo de evento emitido, não permitindo que um componente emita o mesmo tipo de evento por dois canais distintos. Neste trabalho as mesmas abstrações de papéis e protocolos são aplicadas a um modelo de componentes mais completo e portanto mais complexo, como é o caso do modelo CCM. Isso fornece meios para validar a aplicabilidade dessas abstrações na adaptação de sistemas baseados em componentes que sejam implementados ou possam ser integrados com o modelo CCM.

Em [24, 25] são apresentados modelos e ferramentas para evolução de sistemas baseados em componentes através da separação entre computação e coordenação. Essa separação é feita através do uso de contratos, que definem toda a computação relacionada às interações entre os componentes, denominada de coordenação. A computação que efetivamente realiza as tarefas do sistema de computação é realizada pelos componentes do sistema, que são in-

tegrados através dos contratos. As ferramentas apresentadas são baseadas na geração automática do código que implementa os contratos e são posteriormente integrados aos componentes do sistema. Essa geração de código limita a aplicação dessa abordagem na adaptação dinâmica de aplicações. Entretanto, o uso de componentes dinâmicos apresentado neste trabalho pode eliminar a necessidade de geração de código e permitir a geração dinâmica de contratos através de alterações dinâmicas em componentes que compõem um contrato. Em particular, a coordenação do sistema pode ser feita através de interceptadores associados aos diversos componentes do sistema, podendo inclusive serem alterados em tempo de execução.

Em [47] é apresentado o modelo Lasagne, que apresenta o modelo de adaptações em níveis, como apresentado no LuaCCM. Além disso, o modelo Lasagne utiliza uma abordagem que permite manter consistentemente diferentes extensões num mesmo componente utilizadas por clientes com diferentes necessidades, inclusive num ambiente distribuído. Essa consistência é mantida por informações contidas em cada requisição que identificam quais as extensões do componente que devem ser aplicadas no tratamento da requisição. Apesar do LuaCCM não fornecer mecanismos para adaptações consistentes em níveis de abrangência que extrapolem um servidor de componentes, nenhuma informação adicional é associada às requisições CORBA para indicar a adição de extensões num determinado componente. Além disso, o LuaCCM fornece mecanismos para alteração permanente de componentes através da troca de implementação.

7.2 Trabalhos Futuros

Como continuidade da pesquisa desenvolvida neste trabalho, pode-se avaliar os recursos oferecidos pelo LOAF na construção de aplicações dinamicamente adaptáveis, de forma a validar os conceitos e proposições deste trabalho em problemas reais e complexos. Como o LOAF está baseado num modelo de componentes comercial e adotado internacionalmente, esse tipo de experimentação se torna ainda mais relevante. Além do desenvolvimento de aplicações, outra linha de continuidade é a avaliação dos recursos do LOAF no desenvolvimento de arquiteturas adaptáveis através dos conceitos do modelo CCM. Como exemplo, pode-se citar uma adaptação da arquitetura apresentada em [20], podendo inclusive definir mecanismos de monitoração através de papéis, o que permitiria realizar consultas sobre informações da implementação do componente, como a memória utilizada ou número de linhas de execução (*threads*).

Ainda como continuidade dos trabalhos relacionados ao LOAF, pode-se propor a avaliação do uso de outras abstrações em ferramentas do LOAF. Como exemplo, o modelo de desenvolvimento orientado a coordenação apresentado em [24], pode ser incorporado ao LOAF através de *contratos dinâmicos*, que seriam ferramentas para adaptação dinâmica da coordenação de componentes.

Os contratos dinâmicos podem ser implementados utilizando os recursos oferecidos pelo LuaCCM, que permitem alterar as interações entre os componentes através de adição de novas portas ou interceptação das portas existentes. Esse novo recurso permitiria utilizar as abstrações do modelo de evolução de sistemas baseado em coordenação para adaptar dinamicamente sistemas baseados em componentes.

Além de recursos relacionados a adaptações dinâmicas do LOAF, pode-se incorporar ferramentas para implantação de componentes de forma flexível através de roteiros em Lua. Como exemplo de tais ferramentas, pode-se citar uma ferramenta para obter a implementação de um componente a partir de um repositório (*e.g.*, *Web Service*) e torna-la disponível no ambiente do servidor de componentes onde o componente deva ser instalado. Essa ferramenta poderia fazer uso dos recursos de descrição de dados de Lua para permitir especificar dinamicamente estruturas complexas de sistemas baseados em componentes.

Outra linha de trabalho futuro é reduzir as limitações do modelo LuaCCM, como por exemplo, avaliar mecanismos que permitam realizar um conjunto de adaptações de forma atômica, impedindo que o componente possa receber requisições enquanto estiver num estado inconsistente durante uma adaptação. Além disso, estudar como esses mecanismos podem permitir adaptações atômicas em níveis que possam abranger mais de um servidor. Ou seja, realizar uma adaptação em todas as instâncias de um componente em diversos servidores de componentes LuaCCM. Outros novos recursos importantes no LuaCCM incluem a extensão da interface de adaptação para permitir a gerência de vários interceptadores numa mesma porta, assim como a interceptação de várias portas por um único interceptador.

Em relação aos contêiners Lua, é necessário estender sua implementação de forma que ofereça todos os recursos definidos pelo modelo CCM. Em especial, os contêiners Lua devem oferecer suporte aos serviços de objetos CORBA na forma especificada, ou seja, tanto através de interfaces oferecidas pelo contêiner, quanto pelas políticas do modelo CCM. Além disso, deve-se oferecer o suporte às demais categorias de componentes, como é o caso dos componentes de processo e entidade, fornecendo inclusive o recurso de persistência gerenciada pelo contêiner. Já em relação ao *framework* de implantação do modelo CCM, a extensão dos contêiners Lua deve implementar as demais interfaces, em especial a interface Components::ComponentInstallation, que permite disponibilizar pacotes de componentes no ambiente de execução do servidor. Para permitir que a implantação de componentes LuaCCM possa ser feita através de outras ferramentas de implantação, deve-se implementar também o suporte aos arquivos XML definidos pelo modelo CCM. Também é importante estender o modelo de pacotes do LuaCCM para uma estrutura mais robusta, onde seja possível incluir outros recursos além da própria implementação do componente (*e.g.*, imagens).

Devido ao alto grau de intrusão dos mecanismos adaptativos do LuaCCM, outra linha de trabalho seria a investigação de recursos de segurança, que permitam restringir adaptações em componentes. Particularmente, esses mecanismos devem permitir a definição de quais os tipos de alterações

são autorizadas, por exemplo impedindo o acesso dos segmentos ou interceptadores adicionados aos segmentos que compõem o componente original. Esses recursos podem ser definidos com o auxílio do serviço de segurança de CORBA.

Por fim, vale ressaltar que os recursos oferecidos pelo LOAF podem servir como base para o desenvolvimento de outros projetos que investiguem diferentes aspectos e abordagens para o problema de adaptação dinâmica de aplicações baseadas em componentes.