6
Conclusões

O estudo realizado neste trabalho procurou analisar o uso de contratos em sistemas de componentes de software, especialmente os aspectos relacionados à avaliação em tempo de execução. Foram apresentados benefícios, bem como algumas dificuldades e limitações relacionadas ao contexto de contratos e componentes distribuídos.

Embora não sejam técnicas recentes, a utilização em conjunto de contratos e componentes não é muito comum nos projetos de desenvolvimento. Ela depende da disponibilidade de ferramentas adequadas que a viabilizem, e de uma conscientização em relação aos seus benefícios. Poucas linguagens e sistemas de *middleware* atuais possuem suporte a estes mecanismos de forma integrada, principalmente os de grande adoção no âmbito comercial. Portanto, as ferramentas existentes precisam evoluir no sentido de oferecer melhor suporte às tecnologias de contratos e componentes, sem deixar de considerar os fatores que dificultam o seu uso no contexto de componentes distribuídos.

Este capítulo apresenta uma avaliação geral da discussão realizada no escopo desta dissertação, e lista alguns pontos a serem explorados em trabalhos futuros.

6.1
Avaliação do Estudo

Ao longo do texto, foram apresentados diversos aspectos favoráveis ao uso de contratos associados a componentes de software. O aumento do formalismo da especificação leva o desenvolvedor a uma maior reflexão sobre as interfaces elaboradas, contribuindo para a sua completude e consistência, e reduzindo sua ambigüidade. Com isso, melhora a compreensão da especificação tanto por parte dos seus responsáveis, como pelos analistas que a implementam e por aqueles que a utilizam em seus projetos. Por si só, esta característica dos contratos já representa uma vantagem significativa da sua adoção. Os exemplos apresentados no capítulo

5 ilustram alguns casos em que os contratos auxiliaram na revisão e melhora de interfaces existentes, promovendo a correção de defeitos.

Ao elaborar uma especificação usando contratos, o analista deve se voltar para *o que* o software deve fazer, e não *como*, o que ajuda a ter uma visão melhor da abstração que o componente ou objeto em questão representa. Além disso, a definição de um contrato promove uma divisão mais clara entre consultas e comandos, uma vez que os efeitos colaterais devem ser considerados. Isto contribui para a criação de interfaces com métodos mais objetivos e com uma semântica mais precisa, além de evitar que sejam criadas operações que concentrem muitas funcionalidades e que se tornem confusas para a sua implementação e utilização.

Além do efeito positivo sobre a especificação, os contratos também promovem a verificação da consistência do software em tempo de execução. Os contratos de sincronização representam um papel ainda mais crítico, pois tornam-se parte integrante do sistema em questão, uma vez que, além de especificar os aspectos relativos à concorrência, são responsáveis por efetivamente implementá-los. Porém, isto faz com que sejam mais intrusivos que os contratos comportamentais, o que, de certa forma, representa uma mudança em relação aos objetivos originais dos contratos, apenas de especificação. A dificuldade de estabelecer a granularidade adequada na sincronização de operações representa um problema adicional no uso deste tipo de contrato.

Em consonância com a programação orientada a aspectos, os contratos também promovem a separação de interesses nos componentes, principalmente com relação à sincronização. Como apresentado no capítulo 5, o código da implementação pode ficar isento de questões relacionadas à concorrência, delegando ao contrato o seu tratamento. Além disso, os contratos comportamentais favorecem a separação entre o código que verifica condições de erro e o código que implementa as funcionalidades.

Entretanto, os fatores que influenciam o uso de contratos em ambiente distribuído não devem ser ignorados. As dificuldades de se obter um mecanismo robusto de verificação de assertivas que dependam de objetos remotos são significativas, pois dependem de suporte a transação e bloqueio distribuídos. Ainda assim, os experimentos realizados demonstram que, mesmo com limitações, é possível obter alguns benefícios da utilização de contratos neste contexto.

Por outro lado, as limitações decorrentes do uso de contratos em ambientes distribuídos tratados na seção 4.6 não são problemas novos, pois mecanismos de bloqueio e transação de objetos distribuídos são necessários em diversos tipos de aplicações, mesmo quando não há nenhum contrato envolvido. Portanto, nos sistemas que já contam com este tipo de recurso, seria necessário levar a mesma funcionalidade para o subsistema de contratos, para permitir a criação de assertivas capazes de lidar com este cenário.

Uma possível mudança de abordagem dos contratos em ambiente distribuído, em que a manutenção do estado de um objeto remoto entre a sua avaliação na pré-condição e a execução da operação (ou entre a operação e a pós-condição) é difícil de ser garantida, seria o contrato deixar de ser um instrumento de verificação da correção para ser um "monitor" do comportamento do sistema. Ao invés de lançar exceções ou interromper a execução, as violações de contrato poderiam ser registradas e acompanhadas pelos analistas para correlacionar a sua distribuição com a incidência de falhas do sistema. Aos poucos, seriam eliminados os falsos positivos das violações, contribuindo para a correção dos defeitos do sistema.

Um outro ponto a ser considerado no uso de contratos é que nem sempre eles são simples de escrever ou compreender. A elaboração de contratos pode ser difícil em situações nas quais o comportamento do objeto em questão é complexo para ser descrito em algumas assertivas. Ou então, quando elas dependem de informações não presentes na interface, por não serem públicas ou por serem específicas da implementação subjacente. Neste caso, os benefícios provenientes da utilização de contratos são mais limitados. Um exemplo desta dificuldade ocorre no contrato da interface *EventChannel* na seção 5.4, que possui poucos elementos públicos para expressar sua semântica.

Em algumas situações, é possível que as assertivas sejam mais complexas ou mais custosas do que a própria implementação da funcionalidade. No contrato da interface *List* na seção 4.4.2, por exemplo, a operação `List::sublist()` possui uma especificação de contrato de complexidade comparável à sua própria implementação. Nestes casos, cabe ao analista avaliar e decidir se o seu uso é adequado.

A interação dos contratos com o mecanismo de herança em orientação a objetos também é um aspecto importante da discussão apresentada. Os contratos devem manter a consistência dos subtipos que compõem uma hierarquia, e não violar o princípio da substituição de um objeto por outro que implemente uma interface derivada. Verificar as implicações lógicas entre pré e pós-condições para garantir a consistência da hierarquia de objetos é um desafio para os sistemas de contratos, uma vez que envolvem a prova de expressões lógicas que podem ser de difícil tratamento. Na maioria dos casos, cabe ao analista que especifica o contrato cuidar para que as pré-condições sejam mais fracas e as pós-condições sejam mais fortes quando ocorre herança de interfaces ou classes. Como visto no capítulo 2, existem trabalhos que argumentam que, em alguns casos, as implicações lógicas das pré e pós-condições são mais rigorosas que o necessário para que uma hierarquia de classes seja considerada válida.

Outro tópico relevante abordado no texto é o uso de operações sem efeitos colaterais nos contratos, para não modificar o comportamento do objeto sendo avaliado. Como foi visto, poucas linguagens ou ambientes oferecem mecanismos que

permitam especificar métodos que sejam comprovadamente sem efeitos colaterais, o que dificulta para os sistemas de contratos garantir que as assertivas não serão afetadas por este problema.

Os problemas relacionados ao mecanismo de herança e ao uso de operações sem efeitos colaterais são, na maioria dos sistemas de contratos, tratados de modo a depender do programador para que funcionem corretamente. De certa maneira, este representa um aspecto paradoxal dos sistemas de contratos, pois visam a redução da incidência de erros na especificação e implementação de software e, ao mesmo tempo, dependem parcialmente dos analistas para garantir o seu funcionamento correto. Embora exista este tipo de restrição, o uso de contratos apresenta benefícios, e espera-se que estes problemas sejam reduzidos com a evolução das ferramentas disponíveis.

Uma contribuição relevante do trabalho realizado é a análise do uso de contratos em componentes, baseado na implementação do estudo de caso, que contou com recursos para a especificação de facetas e receptáculos. Estes recursos permitem determinar não só o comportamento de uma faceta, mas também a relação existente entre as facetas de um mesmo componente, colaborando para uma especificação mais precisa da abstração que o componente representa. Além disso, a definição de máquinas de estados e contratos de sincronização também ajudam a melhorar a qualidade da especificação em questão.

6.2 Trabalhos Futuros

Nesta seção, são apresentadas algumas possíveis extensões do trabalho realizado no escopo desta dissertação. Algumas dessas extensões foram citadas previamente no texto, principalmente quando foram abordadas as limitações da implementação atual.

Entre as possíveis mudanças, um mecanismo de transação distribuída seria uma importante contribuição ao sistema de contratos implementado, pois aumentaria o potencial de uso da ferramenta em ambientes distribuídos onde a concorrência possui forte influência.

Os contratos de sincronização representam um tema relevante para estudos mais profundos, visando determinar as dificuldades existentes na sua utilização em sistemas de maior porte, já que a influência que exercem sobre os mecanismos de controle de concorrência pode ter grande impacto sobre a sua usabilidade.

A elaboração de um mecanismo de verificação de consistência da hierarquia de contratos para detectar erros na especificação de interfaces derivadas também seria

uma possível extensão, semelhante ao trabalho realizado por Findler e Felleisen [22].

O suporte a transição de estados das interfaces também poderia ser estendido para permitir a fusão de máquinas de estados quando ocorre a herança de interfaces, como visto na seção 4.6. A implementação do Tamago [29] e outros trabalhos podem servir de modelo para a inclusão desta funcionalidade.

A verificação de compatibilidade entre interfaces poderia ser melhorada através do suporte à compatibilidade estrutural, em que interfaces diferentes poderiam ser verificadas quanto à equivalência entre operações e parâmetros para permitir, por exemplo, a conexão de objetos a receptáculos cuja interface não pertença à mesma hierarquia que a interface especificada. Este mecanismo poderia ser incorporado ao sistema de contratos e à implementação do SCS para que a conexão fosse aceita.

As características da linguagem de especificação de contratos poderiam ser mais integradas, uma vez que não é possível, por exemplo, referenciar os estados da interface nas assertivas ou nas condições de sincronização. Futuramente, a implementação destas funcionalidades poderia ser revista para permitir a sua utilização em conjunto.

Uma característica para aumentar a usabilidade do sistema de contratos seria a criação de uma ferramenta gráfica que facilitasse a especificação dos contratos em conjunto com arquivos de interface, permitindo a visualização de ambos num só ambiente e fazendo a verificação da consistência entre eles. Isto tornaria a elaboração de contratos mais simples e menos sujeitas a erros, auxiliando também na manutenção simultânea de interfaces e contratos durante o desenvolvimento de um sistema. Outra possibilidade seria criar uma linguagem mais específica para expressar os contratos e componentes de forma integrada. Esta linguagem poderia servir de base para gerar a descrição das interfaces em IDL e a representação interna dos contratos no sistema.

A integração do LUC à biblioteca LOOP também poderia contribuir para a criação de uma ferramenta de desenvolvimento orientado a objetos em Lua bem completa, com suporte a contratos embutido. Com isso, a elaboração de contratos para objetos fora do ambiente do OiL poderia ser mais consistente e padronizada.

Um outro tema para trabalhos futuros seria avaliar a inclusão de especificações de segurança nos contratos de componentes, de forma semelhante aos aspectos de sincronização. Dentre os trabalhos encontrados na literatura, poucos correlacionam o conceito de contratos com a segurança, o que poderia ser um ponto interessante a ser explorado.

Um trabalho futuro que poderia enriquecer significativamente a discussão contida nesta dissertação seria o desenvolvimento de um sistema utilizando o ambiente do OiL/SCS com suporte a contratos, desde a sua especificação até a implementa-

ção e manutenção. O projeto poderia ser acompanhado de métricas para avaliar os benefícios obtidos e as dificuldades do uso de contratos.

O uso de contratos no cliente não foi avaliado com mais detalhes no estudo realizado, por não ter havido uma necessidade natural nos experimentos realizados de utilizar esta funcionalidade. Futuramente, seria interessante explorar as possibilidades e analisar melhor os benefícios da avaliação distribuída de contratos.

Por fim, a ferramenta implementada pode servir de base para experimentos em disciplinas ligadas ao tema, em que equipes distintas poderiam desenvolver um mesmo trabalho com e sem o uso de contratos, simultaneamente. Os resultados poderiam ser comparados em função de critérios como número de defeitos, qualidade do código e prazo de conclusão, para avaliar o impacto dos contratos sobre o processo de desenvolvimento baseado em componentes.