5
Considerações Finais

5.1. Conclusões

O objetivo deste trabalho foi realizar um estudo abrangente sobre o desenvolvimento orientado a serviços que, segundo alguns estudos, tem um grande potencial para ser referência no desenvolvimento de software em alguns anos.

O primeiro passo do trabalho foi identificar, através de estudos sobre aplicações orientadas a serviços, as características que podem ser consideradas relevantes na construção dessas aplicações. Um conjunto de nove características foi identificado e, em seguida, estudadas isoladamente. O trabalho buscou também diferenciar o conceito de serviços do conceito de componentes, e concluiu que são conceitos similares. Identificou-se que o termo "serviço" é empregado de formas distintas na literatura acadêmica e na literatura não acadêmica. O resultados desses estudos deram origem ao capítulo dois deste trabalho.

Paralelamente, foram investigados alguns *frameworks* que fornecem suporte a construção de aplicações orientadas a serviços. Dentre as várias opções existentes foram escolhidos três *frameworks*, sobre os quais os estudos foram aprofundados. Os três *frameworks*, Vinci, Jini e XML Web Services, são significativos uma vez que possuem diferenças na maneira que foram projetados, e mais ainda, na maneira como suas aplicações são construídas. Existem outros *frameworks*, porém todos encontrados possuem alguma similaridade com um dos três escolhidos para este trabalho. O estudo de XML Web Services serviu principalmente para desmistificar essa nova tecnologia, que vem sendo apresentada, principalmente pelo mercado de desenvolvimento de software, como uma grande inovação tecnológica e que irá alterar a maneira como o software é desenvolvido atualmente. A conclusão que o trabalho chegou é que XML Web Services não apresentam grande inovação em termos de tecnologia, e sim uma inovação no que diz respeito a padronização de protocolos. Devido a essa padronização, é possível que a adoção dos XML Web Services em larga escala se

concretize, diferentemente do que ocorreu em propostas anteriores de soluções para o desenvolvimento de sistemas distribuídos.

A última parte do trabalho foi construir um *framework* que fornecesse suporte ao desenvolvimento orientado a serviços. Tirando proveito dos estudos feitos nas fases anteriores do trabalho, o objetivo foi criar um *framework* que resolvesse grande parte dos problemas de orientação a serviços com simplicidade. Além disso, o *framework* foi construído para ser uma extensão de um servidor de aplicações J2EE e com isso, deixar que suas aplicações pudessem utilizar todas as funcionalidades que um servidor de aplicações dessa categoria fornece.

Atualmente o *framework* já é utilizado em duas aplicações de médio porte. Em ambas aplicações, a escolha do XMLTalk ocorreu principalmente devido a sua facilidade para construir aplicações que descrevem seus dados em XML e a necessidade de integração com componentes já existentes desenvolvidos para a plataforma J2EE. O uso do XMLTalk gerou uma grande produtividade, principalmente pela possibilidade de programação declarativa do uso dos serviços através de *sitemaps* em XML. O mecanismo de disponibilização automática acelerou o desenvolvimento, pois re-iniciar o servidor de aplicação a cada alteração de código passou a ser desnecessário. Já o mecanismo de cache se mostrou fundamental em uma das aplicações onde o alto desempenho era mandatório. A não obrigatoriedade da explicitação dos atributos de um serviço acabou complicando o desenvolvimento, pois os clientes precisavam saber quais atributos deveriam ser enviados para que o serviço executasse sua tarefa adequadamente. Por não ser obrigatória, a explicitação desses atributos quase sempre não foi feita, dificultando o trabalho de desenvolvimento dos clientes. Talvez, o uso de uma estratégia de *proxies* dinâmicos, criados em tempo de execução com interfaces completas e bem definidas fosse mais apropriada, pois obrigaria a formalização dos atributos de um serviço,. O XMLTalk precisa ainda de uma experimentação maior para assim atingir um grau de maturidade requerido para que possa ser adotado no desenvolvimento de sistemas de missão critica e alta disponibilidade, como é seu objetivo.

A conclusão final é que através do estudo de algumas alternativas existentes para a construção de aplicações baseadas em serviços, e principalmente, da elaboração de um *framework* completo, que fornece suporte a sistemas orientados a serviços, foi possível investigar as características relevantes para aplicações

orientadas a serviços. Apesar do surgimento de arquiteturas orientada a serviços, o desenvolvimento de sistemas distribuídos continua sendo um desafio para desenvolvedores. As questões envolvidas, como desempenho, concorrência, heterogeneidade, entre outras, ainda não foram totalmente resolvidas. Porém, com certeza o uso de *frameworks* concebidos com o intuito de facilitar tais tarefas é muito proveitoso. Mais do que isso, se basear no conceito de serviços para a concepção de novas aplicações pode trazer um ganho real em médio prazo, pois o reuso de serviços já constituídos mesmo em ambientes heterogêneos é altamente factível.

A seção a seguir apresenta sugestões de trabalhos futuros com o objetivo de encorajar a continuidade do trabalho desenvolvido

5.2. Trabalhos Futuros

Em relação a trabalhos futuros este trabalho fornece algumas opções no que diz respeito a continuidade do desenvolvimento do *framework*. Abaixo são citadas algumas das possibilidades:

- **Maturidade do *framework***

  Para o *framework* atingir sua maturidade e conseqüente aumento da usabilidade, é necessário que o XMLTalk seja testado mais intensamente. Durante o desenvolvimento ele foi testado em dois sistemas de médio porte, porém seria interessante aumentar a quantidade de testes de unidade e criar mais aplicações para uso do mesmo. Além disso, a criação de um conjunto maior de serviços a serem distribuídos como parte do pacote básico (*core*) do *framework* também é importante.

- **Criação de novos protocolos**

  Atualmente, o coordenador de serviços só se comunica através do protocolo RMI-IIOP, o que segrega a sua utilização a clientes para o universo Java e CORBA. É importante fazer com que o coordenador entenda ao menos o protocolo SOAP, para que assim, clientes desenvolvidos em outras tecnologias possam fazer uso do XMLTalk. Além disso, é importante

também a criação de outros *proxies* para serviços, fazendo com que o *framework* possa se comunicar com serviços implementados em outras plataformas que não Java, SOAP ou JavaScript, como é possível atualmente.

- **Uso do *framework* em containeres diversos**

Durante o desenvolvimento do trabalho, uma série de "servidores de componentes" Java, como Spring [Spring, 2004] e Avalon [Avalon, 2004], amadureceram e passaram a disputar as atenções com os servidores de aplicação J2EE para construção de aplicações de alta disponibiliade. Esses containeres pregam que a aplicação não deve ser penalizada por funcionalidades que não estão sendo utilizadas. Por exemplo, se sua aplicação não precisar de transação, não é justo, como acontece em containeres EJB, que a aplicação perca desempenho relativo ao controle transacional oferecido. O trabalho proposto é investigar a possibilidade de estender a infra-estrutura fornecida por esses containeres utilizando o XMLTalk.