5
Conclusões

Propomos neste trabalho um modelo de componentes com suporte a múltiplas versões. Os principais fatores que influenciaram o desenvolvimento deste trabalho foram as demandas de evolução contínua dos componentes de software já existentes [1], a necessidade de realizar atualizações parciais em sistemas distribuídos, e a inexistência de um modelo de componentes que ofereça um suporte adequado a essas questões.

O modelo proposto foi concebido estendendo um modelo de componentes tradicional, atentando para os conceitos de modularização [14] de compreensibilidade, continuidade e ocultação de informação. Os modelos de componentes tradicionais apresentam limitações com relação ao suporte a múltiplas versões de interfaces, pois, analisando pela perspectiva da modelagem e organização do modelo e do seu uso, as soluções por eles oferecidas para suporte a múltiplas versões de interfaces ocasionam a perda da identidade da abstração de componentes, e são contrárias à diretiva de modularização do sistema, pois não aglutina informações relacionadas, no caso as diferentes versões de uma interface, em uma mesma entidade ou contexto. O modelo proposto foi implementado através da extensão do sistema de componentes SCS [13].

Para solucionar esses problemas, nós redefinimos o conceito e a representação das facetas, introduzindo o conceito de interfaces de acesso e criando uma interface (*IFacet*) para representar a faceta, onde o conjunto de interfaces de acesso representam as diferentes versões disponibilizadas pela faceta. O conceito das interfaces de acesso foi inspirado no UPSTART [15, 16], que é um protótipo de um modelo que provê suporte a múltiplas versões de serviços em sistemas distribuídos, porém não adota uma abstração de componentes de software.

Realizamos um estudo de caso aplicando o modelo proposto sobre uma aplicação real, que precisa prover suporte a múltiplas versões de uma mesma interface e é desenvolvida utilizando o SCS. Nesse estudo de caso, adaptamos duas versões distintas da aplicação para usar o SCS-MV, retratando a evolução natural da mesma.

Constatamos com o estudo de caso que o modelo proposto se mostrou

adequado para todos os cenários de uso da aplicação e a modelagem proposta se mostrou mais coesa e trouxe bons benefícios, principalmente para a gerência de configuração do sistema. Com esse modelo o sistema pode evoluir sem provocar uma recomposição do sistema, mantendo uma visão mais abstrata de sua arquitetura.

Existem várias frentes para realização de trabalhos futuros. Uma possibilidade seria desenvolver um framework para realização de atualizações dinâmicas inspirado no framework do UPSTART (explicitado no apêndice C). Com isso, seria possível aprimorar o mecanismo de conexão, para solicitar a instalação de uma versão dinamicamente quando alguma versão não é encontrada.

Outra opção seria estender a ferramenta de implantação do SCS [32] de forma a levar em consideração as informações de múltiplas versões das facetas no momento de configurar e implantar o sistema. Também é importante implementar o modelo proposto nas demais linguagens suportadas pelo SCS, e isso pode levantar problemas relacionados ao suporte que essas linguagens dão à propria abstração de componentes de software. Seria especialmente interessante levar esse modelo para o ASCS [33] e rever as anotações Java, para dar suporte a multiplas versões.

Outra possibilidade de trabalho futuro é procurar melhorar o suporte de programação oferecido ao desenvolvedor do componentes. Como por exemplo, prover uma maneira de não obrigar o uso da indireção da interface de acesso se a faceta não oferecer suporte a múltiplas versões, ou se possuir uma versão padrão. Um item que não pode ser esquecido é desenvolver algum mecanismo para garantir que os nomes dos sub-serviços sejam únicos no sistema. Os nomes dos sub-serviços ganharam uma importância semântica muito forte, e o SCS-MV não oferece nenhum apoio para garantir sua consistência.

Por fim, seria interessante realizar mais estudos de casos para experimentar e avaliar o modelo em outros sistemas, com mais versões e com maior distância entre as versões. Isso pode levar à identificação de outros mecanismos de apoio à implementação das interfaces de acesso.