CONTROLE DE CONSISTÊNCIA EM UM AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE DISTRIBUÍDO (ADSD) COM OBJETOS IMUTÁVEIS

Carlos A. M. Pietrobon
Arndt Von Staa
Departamento de Informática - PUC/RJ
Rua Marquês de São Vicente, 225, Gávea
22453 - Rio de Janeiro - RJ

Ambientes de Desenvolvimento de Software Distribuídos (ADSD), tem sido propostos para aumentar a produtividade (software de qualidade assegurada produzida por unidade de tempo), das equipes envolvidas com produção de softwares grandes e complexos.

Para vencer a complexidade dos sistemas, estes tem sido decompostos em subsistemas (ou objetos de projeto), menores, de complexidade reduzida. Cada um destes objetos é então desenvolvido por diferentes pessoas ou equipes, as quais podem estar geograficamente distribuídas. Cada objeto pode ser subdividido em outros objetos ainda menores e, cada um destes, pode ter diferentes versões e representações (e.g. especificação funcional e o código fonte correspondente). Além disso, um mesmo objeto pode ser usado em diversas partes do projeto ou mesmo em projeto distintos. Deste maneira, é necessário que um ADSD consiga controlar o compartilhamento destes diversos objetos com fim de evitar inconsistências entre os mesmos.

Tradicionalmente, o controle de consistência tem sido feito pelas operações de requisição ("check-out") e devolução ("check-in"), do objeto à base de dados. Estas operações exigem serialização do acesso e destroem a informação anterior.

Uma outra proposta para o controle de consistência, quando existe acesso concorrente, considera os objetos de projeto como imutáveis, ou seja, o objeto uma vez criado não pode ser alterado. Qualquer tentativa de atualização gera uma nova versão do objeto. Usando esta técnica, a consistência não é controlada quando da

retirada ou inclusão de um objeto da/na base de dados mas, através do controle de versão e configuração.

Embora esta técnica consiga contornar vários problemas gerados por técnicas tradicionais de controle de acesso concorrente (e.g. serialização de acesso), ela não garante que a nova versão seja consistente com sua especificação e com as outras versões com as quais ela se relaciona. Além disso, neste ambiente existe a possibilidade de se gerar uma nova versão pela obtenção de informações localizadas em outras versões (junção de versões), o que acarreta a necessidade de se checar todos os objetos com os quais as versões se relacionam, além de checar conflitos diretos entre versões que são combinadas (objetos que tiveram uma mesma porção alterada concorrentemente).

As ferramentas existentes atualmente para apoiar o controle de consistência não são adequadas. Elas, quando existem, atuam mais a nível sintático (e.g. compilador) do que semântico e, em sua maioria, atuam a nível de código fonte, sendo poucas as ferramentas existentes para o controle de outros tipos de representações. O controle a nível semântico é bem mais difícil de ser conseguido, sendo necessário se dispor de informações a respeito da natureza da entidade sendo representada, o que é difícil de se representar e manipular automáticamente. Assim, possivelmente ter-se-á que contar com a participação do projetista. Finalmente, existe carência de suporte no que tange a consistência entre duas representações de um mesmo objeto. Usualmente, este último controle é deixado a cargo do projetista.

Concluindo, a técnica de manipular objetos de projeto como sendo imutáveis, facilita a manutenção de consistência destes objetos. No entanto, muito ainda tem que ser feito. São necessárias novas ferramentas e técnicas que controlem a consistência nos níveis semântico e sintático, de todas as representações, de todos os objetos de projeto que compõem o sistema, ao longo do ciclo de vida deste sistema. Pelo menos, estas ferramentas devem alertar para a possibilidade de ocorrência de uma inconsistência quando uma alteração for realizada.