# 6
# Conclusão

Nesse trabalho apresentamos os principais conceitos de sistemas de gerenciamento de *workflow* e de tolerância a falhas por meio de replicação. A partir desses conceitos, identificamos requisitos para um mecanismo que pudesse ser usado para instrumentar uma aplicação com tolerância a falhas. Como estudo de caso, caracterizamos um sistema de automação de procedimentos como um sistema de gerenciamento de *workflow* com foco em processos produtivos e utilizamos o mecanismo para torná-lo tolerante a falhas. Além disso, desenvolvemos um conjunto de *workflows* de testes para esse sistema que exercitasse as principais características de sistemas de gerenciamento de *workflow* e analisamos os resultados obtidos para avaliar o impacto do uso do mecanismo no desempenho do sistema. Apesar do sistema usado no estudo de caso ser utilizado realização de processos produtivos, nada no mecanismo foi feito especificamente para esse tipo de processo. Sendo assim, podemos afirmar que o mecanismo pode ser usado para permitir a tolerância a falhas na realização de processos administrativos.

Apesar dos resultados indicarem claramente aumento no tempo médio para realização do controle quando há uma falha na réplica principal, esse aumento foi da ordem do intervalo definido para identificação da falha. Isso indica que, se for possível utilizar valores menores para esse intervalo o tempo, o mecanismo pode ser usado para execução de processos com tempo de resposta menor. Todavia, o mecanismo foi tolerante a falhas e manteve a execução do controle com seguidas interrupções na réplica principal. Os testes com falha na réplica principal foram realizados em sequência e o controle continuou sendo realizado corretamente.

O processo modelado de um sistema que utilize esse mecanismo irá continuar sendo realizado mesmo que uma falha interrompa a execução de uma de suas máquinas. Entretanto, falhas de outra natureza, por exemplo, falhas que impliquem no envio de mensagens incorretas, não poderão ser evitadas e demandam soluções específicas. Nesse exemplo, essas falhas podem ser superadas usando alguma forma de replicação ativa para as operações das réplicas e realizar o consenso em relação ao resultado das operações e do novo

estado que deve ser aplicado. Além disso, se houver a necessidade de realização de ações que não sejam idempotentes, será necessário buscar uma forma de execução atômica dessas ações para que a recuperação de uma falha nesse ponto não interfira no resultado do processo.

Na sua implementação atual, o mecanismo fica vulnerável à presença de aplicativos não-confiáveis na rede. Aplicativos que conheçam as interfaces usadas pelo mecanismo e a localização do motor do sistema e dos replicadores podem influenciar a operação dos processos. Para evitar esse problema seria necessário um serviço de certificação do acesso que, além do tempo de implementação, poderia impactar o desempenho. Entretanto, no sistema utilizado para o estudo de caso, o nível de segurança foi mantido, pois o motor já disponibilizava uma interface de controle para a rede local para acesso mediante a senha, ficando sujeito apenas à segurança da rede em que opera.

Além de possibilitar a operação do sistema na presença de falhas, o mecanismo de tolerância a falhas permite, também, que a manutenção das máquinas seja feita sem interromper a execução dos processos. Para tal, basta que o replicador da máquina em que será feita a manutenção seja removido do grupo. Entretanto, deve-se confirmar se o replicador em questão é o replicador principal por meio da localização de serviço. Se for o caso, pode ser necessário pausar a execução dos processos antes da remoção do replicador para que nenhuma atualização seja perdida. Uma vez que os processos do replicador ou do motor sejam interrompidos (explicitamente ou pelo desligamento da máquina), os demais replicadores irão eleger uma nova réplica principal e retomar a execução dos processos.

Um ponto que observamos e que também é mencionado em [15] é a dificuldade em se adaptar um sistema já em operação para que fique tolerante a falhas. A maior parte dessa dificuldade está na obtenção, redefinição e atualização de estado. No caso de sistemas de gerenciamento de *workflow* a manipulação do estado é o ponto mais trabalhoso das modificações necessárias. De acordo com a implementação do motor que executa os processos, a descrição da ação que está sendo executada pode não estar disponível. Entretanto, essa característica não é observada em motores que fazem a varredura de todas as ações e verificam seu estado, determinando se estão ativas. Nesses casos, bastaria armazenar o estados das ações, o estado das instâncias dos *workflows* e os dados relevantes para os *workflows*.

Em relação ao tempo gasto nas modificações do sistema do estudo de caso para a obtenção de estado e para a interação com o mecanismo, percebemos que o maior desafio foi na obtenção de estado. Para esse sistema foi necessário alterar a forma que as ações eram executadas para que todas as informações

necessárias estivessem disponíveis ao retomar a execução em outra máquina. Se tivéssemos que estimar, a proporção entre modificações para gerenciamento de estado e para interação com o mecanismo é de 3 para 1.

No estudo de caso, uma questão não foi observada nos testes de desempenho: o custo do envio do estado inicial para uma réplica. Como os modelos de processo e de planta desenvolvidos no MPA não são alterados durante a execução, esses elementos não estão sendo incluídos no estado relevante utilizado para atualização e inclusão de novas réplicas. Sendo assim, a entrada de uma nova réplica pode ficar mais lenta se o estado enviado incluir o estado inicial. Entretanto, esse caso apresentaria uma vantagem em relação a implementação atual, pois não seria mais necessário reinicializar o sistema para atualizar os modelos de processo e de planta.

Outro ponto que pode ser modificado no sistema MPA que traria um ganho em relação ao tempo para obtenção do estado e possivelmente para o tempo de identificação de falha é a inclusão de comandos de espera incompletos no estado completo. No modelo atual do sistema MPA comandos de espera são funções comuns, e não é possível distinguir programaticamente um comando de espera de um comando que possua uma espera interna. Se fosse definido um elemento de espera aos fluxogramas do MPA, a entrada de uma nova réplica não precisaria aguardar a conclusão de esperas em andamento. Nessa caso, bastaria armazenar o tempo restante das esperas no estado com o tempo. Para isso, o tempo restante deveria ser armazenado de forma independente do relógio da máquina, pois o relógio da máquina origem e o relógio da máquina destino poderiam estar fora de sincronia.

Apesar dos resultados indicarem que o mecanismo não apresenta impacto significativo em uma execução sem falhas, não é possível afirmar que esse comportamento será observado para todos os sistemas de gerenciamento de *workflow*. Para sistemas e processos que demandem um tempo de resposta menor, o impacto no desempenho pode ser impeditivo, demandando melhorias no mecanismo.

Para que o mecanismo possa ser usado em conjunto com o sistema do estudo de caso em soluções que tenham requisitos de tempo de resposta menores, é necessário utilizar uma versão da infra-estrutura de comunicação que não deixe de enviar mensagens enquanto tenta re-estabelecer o canal de comunicação com réplicas em falha. Substituindo esse elemento, será possível utilizar intervalos para identificação de falha menores, sem o risco de suspeitar incorretamente da falha de uma réplica.

Futuramente, o mecanismo pode ser estendido para tolerar falhas bizantinas e falhas de particionamento. Além disso, seria interessante avaliar a via-

bilidade de definir o tipo de replicação por ação ou por *workflow*, ou seja, *workflows* menos importantes poderiam ter esquemas de replicação menos agressivos enquanto fluxogramas críticos manteriam o comportamento atual em que o resultado de cada ação é encaminhado para as réplicas. Alternativamente o mecanismo poderia utilizar uma ou as duas flexibilizações para replicação passiva quente mencionadas no capítulo 3. Com isso o mecanismo deixaria de interromper a execução do motor em um ou dois momentos da replicação: durante o envio da atualização para os replicadores e durante a aplicação da atualização nos replicadores.

Na seção 4.5 discutimos algumas limitações do mecanismo e indicamos algumas melhorias que podem ser avaliadas para que o mecanismo seja usado com sistemas de gerenciamento de *workflow* que demandem tempos de resposta e recuperação menores. A manutenção de *timeouts* por replicador, a propagação de mensagens por *heap* e a flexibilização das garantias da replicação poderiam ser utilizadas em conjunto. O mecanismo poderia ordenar a lista de replicadores e definir os candidatos de acordo com o *timeout* observado. Dessa forma, a réplica principal poderia utilizar a flexibilização do número de respostas necessárias para retomar a execução assim que enviasse a atualização para as duas réplicas com o menor *timeout*. Essa ordenação acabaria deixando a réplica principal independente do tempo que leva para enviar as mensagens de atualização para réplicas mais distantes minimizando a sobrecarga no tempo para o realização das ações.