Orquestração vs Coreografia em microserviços

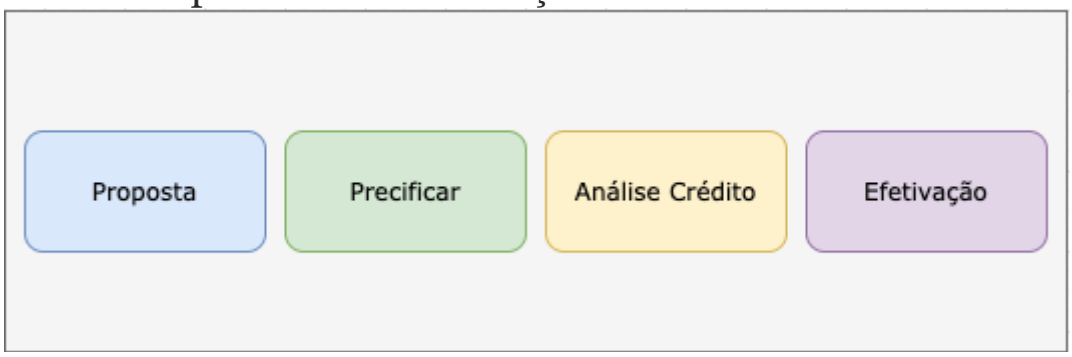
Cristiano Altmann

Em arquiteturas de microserviços e principalmente orientadas a eventos é comum termos que compor processos complexos com a interação entre diferentes serviços. Isso ocorre tanto na composição de um processo de negócio, workflows e nas demais interações entre os serviços.

Normalmente, os micros serviços possuem interfaces de comunicação de entrada e saída. Como exemplo podemos REST APIs (síncronas) ou o processamento e geração de eventos (assíncronas). Desta forma. Quando temos este tipo de arquitetura é comum que um processo complexo seja dividido e composto por diversos serviços. Além disso estes serviços podem fazer parte de mais de um workflow, dependendo do domínio que este represente.

Para compor um processo de negócio, usualmente temos duas maneiras de realizar o design deste: Orquestração ou Coreografia.

Para exemplificar, vamos imaginar um sistema responsável por contratar um crédito. Existem diferentes etapas que serão realizadas por diferentes serviços:

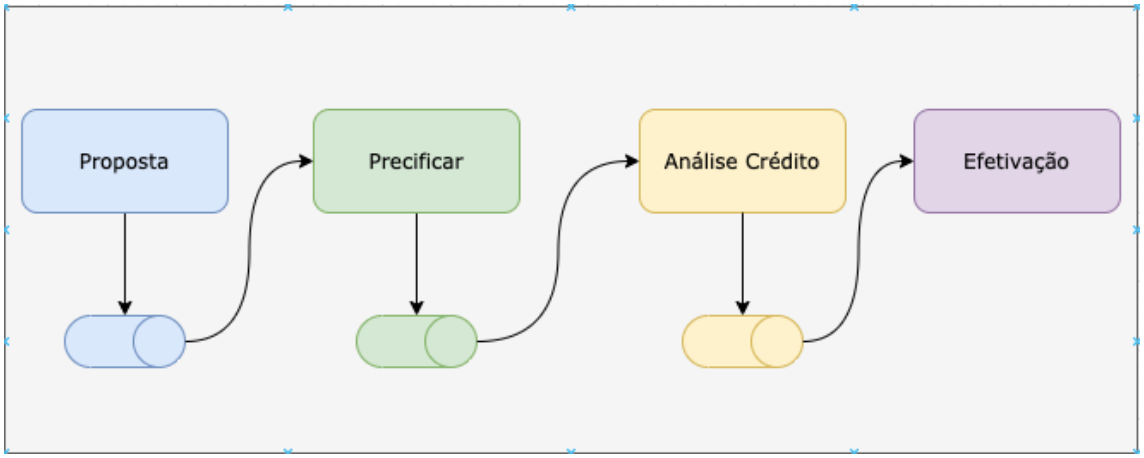


Neste exemplo temos os seguintes serviços que compõem o processo de contratação de crédito:

- Proposta: Armazena os dados de uma proposta como o valor, quantidade de parcelas, etc
- Precificar: Sistema responsável por calcular a taxa de juros de uma proposta
- Análise de Crédito: Responsável por aprovar ou reprovar uma proposta
- Efetivação: Responsável por fazer as movimentações financeiras do crédito.

Coreografia

Na coreografia o processo se dá pela comunicação direta entre os serviços. Não existe um controle central lembrando uma dança em que cada dançarino realiza a sua parte da tarefa mas olhando o todo temos algo mais complexo ocorrendo.



Coreografia

O serviço que produz um evento esta produzindo um resultado de processamento ou uma alteração de estado interno nos casos de event sourcing. Este evento pode ser consumido por diferentes serviços. No caso do workflow o próximo serviço na cadeia do workflow, processa o evento do serviço anterior, realiza a sua parte do processamento e gera uma saída. Claro que estamos falando aqui de eventos, mas podemos ter outros tipos de comunicação envolvidas.

É uma maneira elegante de arquitetar uma solução onde os serviços de domínio através do processamento de eventos reagem a este e realizam a sua parte da tarefa.

Cada serviço então esta interessado em sua etapa e não é necessária a criação de serviços acessórios para controle de fluxo.

Apesar de mais simples esta arquitetura gera alguns problemas.

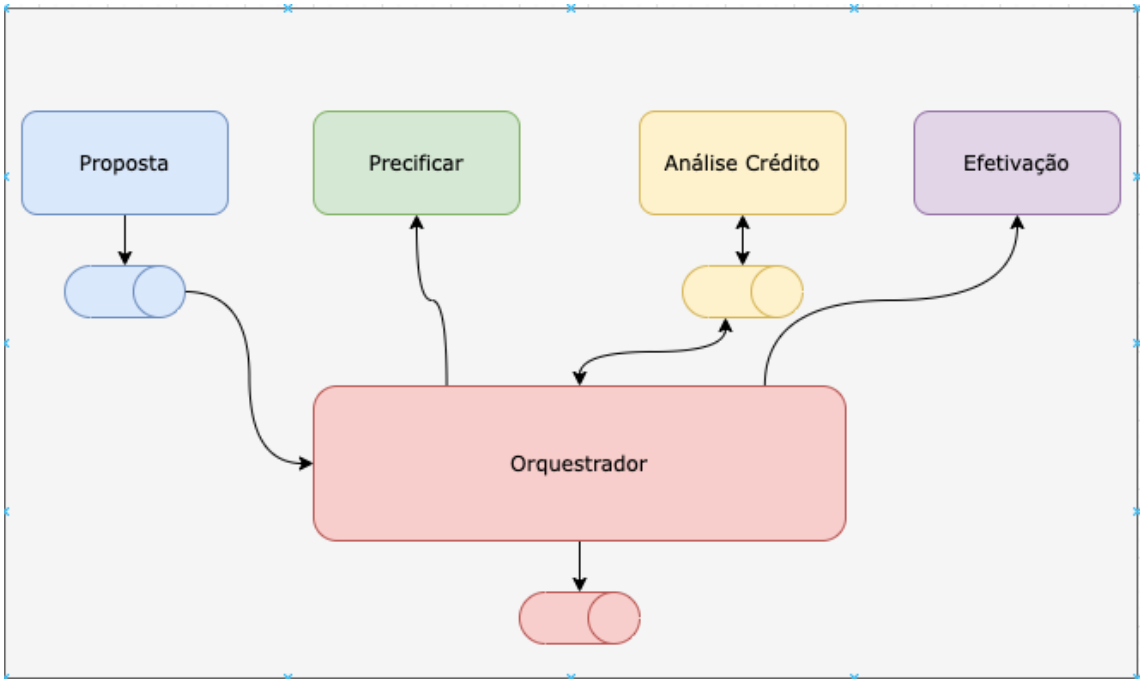
O serviço de precificação do exemplo esta consumindo eventos do serviço de proposta gerando um acoplamento entre eles. Se precisarmos alterar este workflow, adicionando uma etapa de formalização por exemplo, teríamos que alterar o serviço seguinte da cadeia pois temos um acoplamento entre a responsabilidade de negócio do serviço e o processamento do workflow.

É muito mais difícil conseguirmos encontrar problemas dado que não existe um controle neste fluxo, é tudo *coreografado*. Você precisa de um bom sistema de tracing para conseguir identificar erros do fluxo, buscando em logs ou outras ferramentas os pontos de falha.

Há uma dificuldade maior para reexecutar uma etapa ou tratar erros. Como não existe um comando e os sistemas reagem a eventos, temos dificuldade em reprocessar algo pois regerar eventos de domínio podem causar impactos. Os serviços então além de controlar a sua lógica de negócio do seu domínio, precisam ter um bom controle de erro, salvando eventos com erro em cada etapa do workflow para poder gerir de maneira eficiente.

Orquestração

Na orquestração, temos diferentes microserviços responsáveis por realizar uma parte de um processo de negócio. Cada microserviço então realiza esta etapa e não tem conhecimento dos demais componentes do sistema.



Há neste caso, um serviço responsável por gerenciar o fluxo. A este serviço se dá o nome de orquestrador.

Note que os serviços agora não possuem nenhuma ligação entre si. Toda a dependência esta centralizada no orquestrador. Isso gera um acoplamento menor nos serviços que não precisam gerir a responsabilidade de acompanhar um fluxo. Inclusive, os serviços de domínio por vezes podem participar de mais de um fluxo de negócio e com a figura do orquestrador isso fica transparente para estes.

O orquestrador então, por ter este controle, facilita mudanças no fluxo, acompanhamento de erros e também facilita o uso de diferentes protocolos de comunicação entre os diferentes serviços. Comunicações síncronas em sistemas coreografados são pouco resilientes e suscetíveis a falhas. No orquestrador isso pode ser melhor gerenciado.

Outra vantagem do orquestrador é a visualização da execução do workflow. Como temos um sistema central responsável pela execução é simples termos a visualização das etapas realizadas seus sucessos e erros.

Por fim, no orquestrador é muito mais simples reprocessarmos ou executarmos uma etapa novamente visto que os comandos de execução substituem a reatividade, como é um comando podemos organizar a arquitetura permitindo a repetição de comandos, onde cada um é processado isoladamente.

São duas formas interessantes de se trabalhar. Cada uma com suas características e peculiaridades.