Engenharia de Software III

Aula 5
Padrões de Arquitetura
Service-Oriented Architecture

l.bertholdo@ifsp.edu.br

Conteúdo

- Arquitetura Orientada a Serviços (SOA)
- Desenvolvimento de Software com Serviços
- Projeto e Implementação de Workflow
- Testes de Serviços
- SOA e Software como Serviço (SaaS)
- Considerações sobre Escalabilidade

l.bertholdo@ifsp.edu.br

Arquitetura Orientada a Serviços

- Na década de 1990, as máquinas clientes podiam acessar informações de servidores remotos, de outras organizações, apenas via *browser*, o que dificultava o acesso direto de programas a estas informações.

- Nesse contexto, surgiu a ideia de ***web service***, na qual as organizações podem especificar e publicar uma **interface** para definir quais dados são disponibilizados e como podem ser acessados por programas externos.

Web services *são componentes de software que permitem a integração entre aplicações de diferentes plataformas e linguagens, por meio do envio e recebimento de dados via protocolos, como* ***SOAP (Simple Object Access Protocol)*** *e* ***REST (Representational State Transfer)****.*

Eles traduzem formatos de dados específicos das aplicações envolvidas para um formato genérico como ***XML (Extensible Markup Language)****,* ***WSDL (Web Service Definition Language)*** *e* ***JSON (JavaScript Object Notation)****, tornando possível a comunicação entre elas.*

l.bertholdo@ifsp.edu.br

Arquitetura Orientada a Serviços

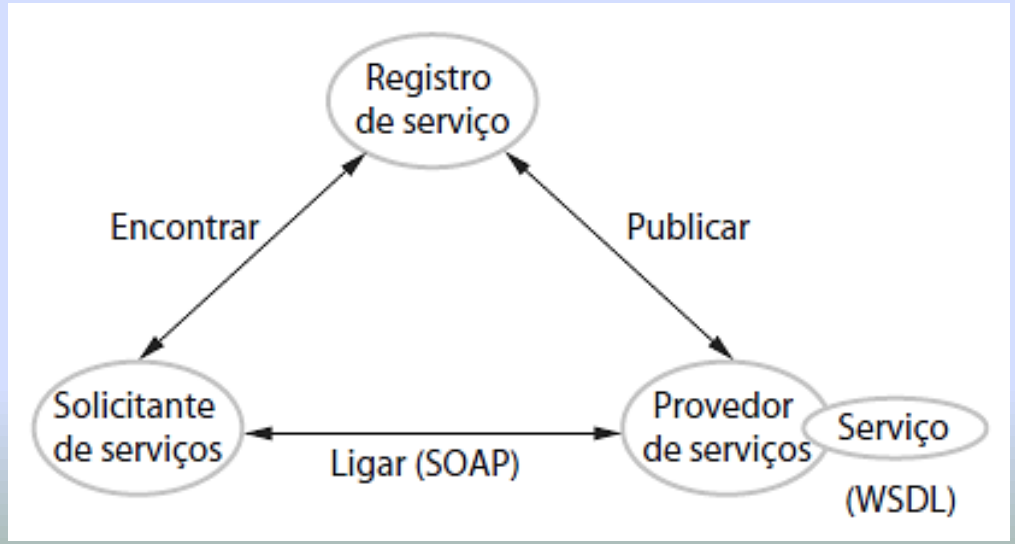
- A **arquitetura orientada a serviços (*Service-Oriented Architecture* – SOA)** é baseada na ideia que os componentes de software são **serviços autônomos**, executados em máquinas geograficamente distribuídas.
- Segundo esse conceito, os sistemas podem ser construídos pela junção de **serviços locais** e **serviços externos** de diferentes provedores.
- Aplicações baseadas em serviços permitem que empresas e outras organizações cooperem e façam uso das **funções de negócios** umas das outras.

*Por exemplo, em **sistemas de cadeia de suprimentos**, que envolvem muitas trocas de informações entre clientes e fornecedores, a comunicação pode ser facilmente automatizada com o uso de uma arquitetura orientada a serviços.*

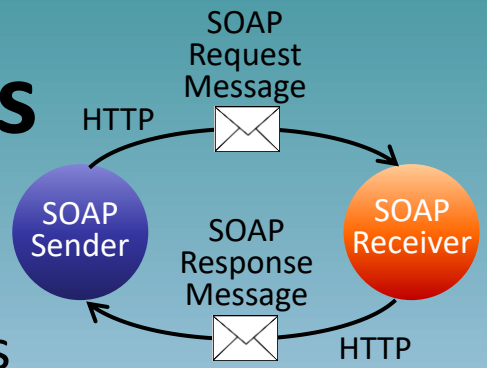
l.bertholdo@ifsp.edu.br

Arquitetura Orientada a Serviços

- Em uma arquitetura SOA, os **provedores de serviços** especificam as interfaces dos serviços (XML, WSDL, JSON etc). Esses serviços são publicados em um **registro**, junto com suas especificações.
- Os **solicitantes de serviço** descobrem a especificação do serviço desejado e vinculam sua aplicação a ele, usando protocolos como SOAP e REST para comunicar-se com o serviço.



Arquitetura Orientada a Serviços



- Um dos protocolos mais conhecidos para troca de informações estruturadas entre aplicações e serviços é o **SOAP**, desenvolvido no final da década de 1990 pela Microsoft.
- Neste protocolo, as mensagens usam o formato **XML**, uma notação legível por máquina e humanos que permite a definição de dados estruturados por meio de marcações.
- As mensagens SOAP podem ser transmitidas com base em diversos protocolos, como HTTP, SMTP, TCP, entre outros.
- Outro padrão usado na arquitetura SOA é o **WSDL**, uma linguagem baseada em XML para definição de interfaces de *web services*. Um documento WDSL descreve o serviço, especifica como acessá-lo e quais as operações ou métodos ele disponibiliza.

Arquitetura Orientada a Serviços

- Nos últimos anos, arquiteturas baseadas no padrão **SOAP** sofreram críticas por serem consideradas pesadas e ineficientes para criar, transmitir e interpretar as mensagens XML associadas.

- Por essa razão, muitas organizações passaram a adotar abordagens mais simples e eficientes para comunicação entre serviços, usando os chamados **serviços RESTful** (baseados no estilo arquitetural **REST**).

- O padrão REST é baseado exclusivamente no protocolo HTTP, e aceita, além de XML, outros formatos de mensagens, mais leves, como JSON, CSV, texto etc.

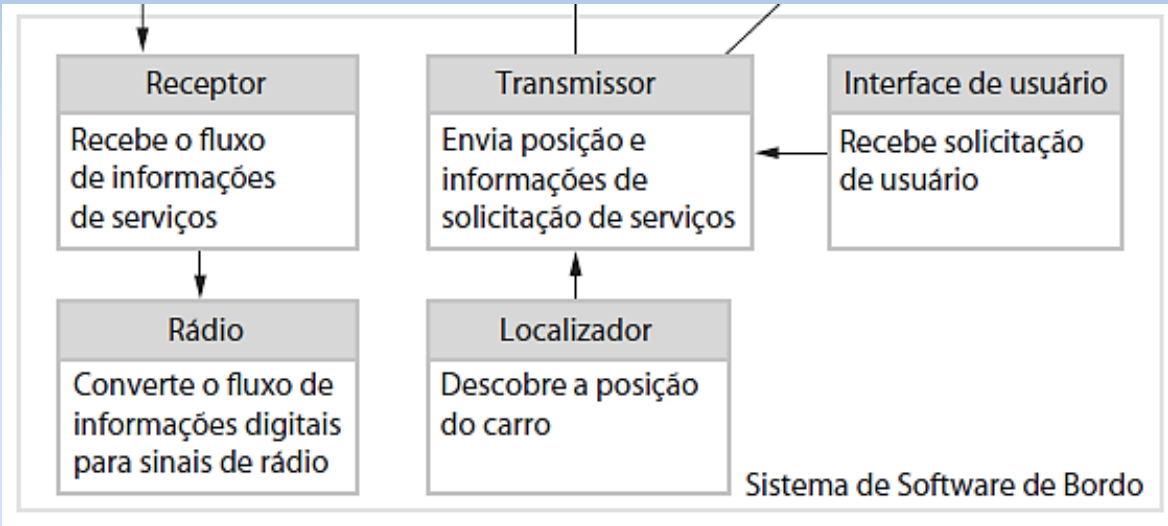
Atualmente, grandes empresas como Google, Facebook, LinkedIn e Netflix disponibilizam APIs Web baseadas na tecnologia REST.

l.bertholdo@ifsp.edu.br

Arquitetura Orientada a Serviços

- Alguns sistemas podem ser construídos apenas com *web services*, e outros podem mesclar *web services* com componentes desenvolvidos localmente.

Exemplo: Sistema de bordo de um carro baseado em componentes locais e serviços



*O sistema de bordo de um carro usa cinco componentes para fornecer ao motorista informações como clima, condições de tráfego, edificações locais etc. O **transmissor** e o **receptor** lidam com todas as comunicações com os serviços externos.*

*O **receptor** é integrado ao **rádio** do carro para que as informações sejam entregues como um sinal em um canal de rádio específico. O **localizador** descobre a posição do carro e, com base nela, o **transmissor** solicita uma gama de serviços de informação. O **transmissor** também envia pedidos de serviços solicitados via **interface de usuário**.*

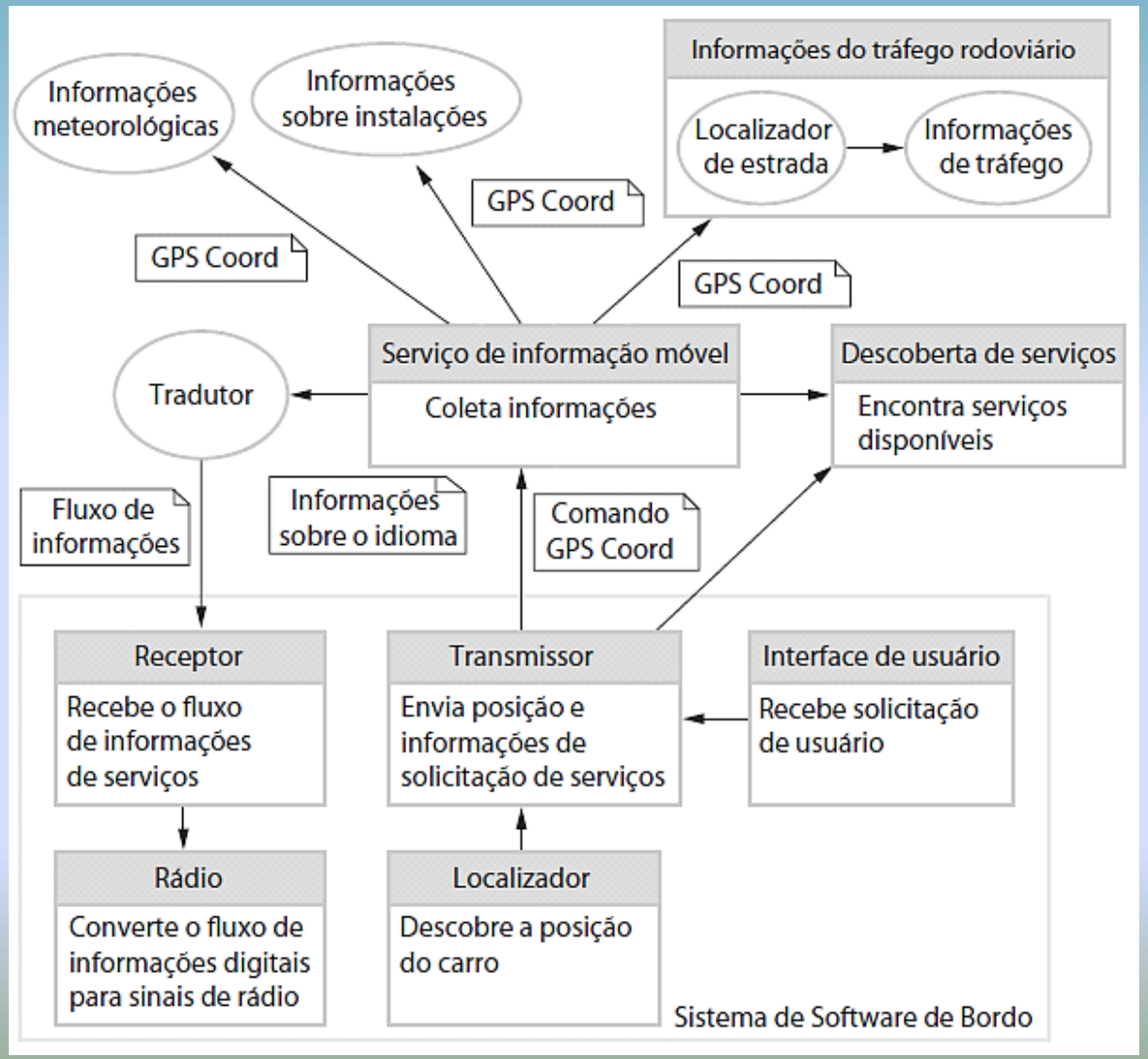
Arquitetura Orientada a Serviços

*O sistema comunica-se com um **serviço de informações** que agrega dados oriundos de vários outros serviços. Ele também usa um **serviço de descoberta** para localizar serviços adequados. O **serviço de descoberta** é usado também pelo **serviço de informação** para conectar os serviços adequados.*

*Os dados agregados pelo **serviço de informações** são enviados para o carro por meio de um **tradutor** que os configura no idioma especificado pelo motorista.*

*Esse exemplo ilustra a **flexibilidade modular** oferecida pela abordagem SOA. Não é necessário decidir qual provedor de serviço deve ser usado, quais serviços devem ser acessados e nem o idioma das informações. Conforme o carro se move, o sistema usa o serviço de descoberta para encontrar o serviço mais adequado.*

Os serviços trocam mensagens entre si, como as coordenadas geográficas do carro.



Desenvolvimento de Software com Serviços

- A ideia de compor e configurar serviços para criar **serviços compostos** aumenta a possibilidade de **reúso** generalizado dentro da organização.
- Por isso, atualmente, muitas empresas estão convertendo suas aplicações corporativas em **sistemas orientados a serviços**, em que cada bloco básico da aplicação é um serviço, e não um componente.
- A composição de serviços pode integrar processos de negócios separados, a fim de oferecer uma gama maior de funcionalidades.

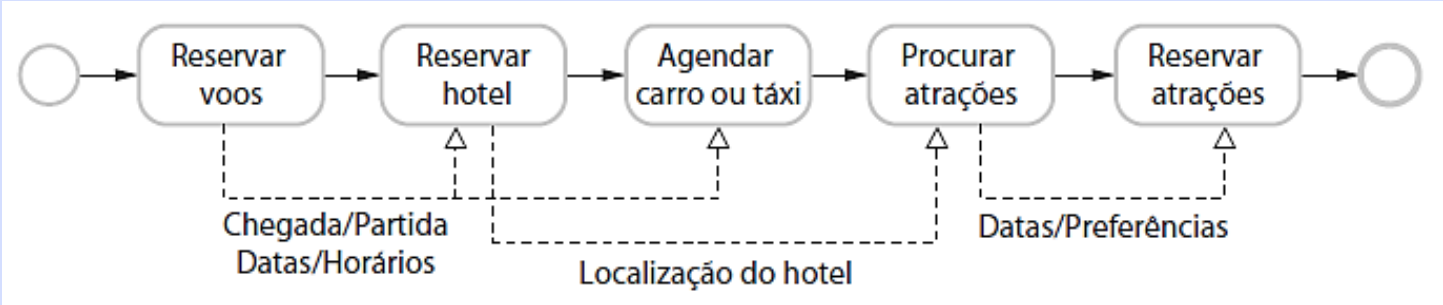
Uma companhia aérea poderia fornecer um pacote de férias completo para os viajantes, ou seja, além de reservar seus voos, os viajantes poderiam também reservar hotéis, alugar carros, reservar táxi no aeroporto, fazer reservas para visitar atrações locais.

Para criar essa aplicação, a companhia aérea poderia criar um serviço único, composto pelo seu próprio serviço de reserva de voos e os serviços oferecidos por outras empresas, como hotéis, locadoras de carro, companhias de táxi e plataformas de venda de ingresso.

Desenvolvimento de Software com Serviços

- O exemplo da companhia aérea pode ser visto como um ***workflow***, em que as informações são passadas de uma etapa para a próxima.
- Um *workflow* é um modelo de processo de negócios que define as etapas envolvidas para alcançar um objetivo particular e importante para uma empresa, no caso do exemplo, o serviço de pacote de férias.

Workflow da reserva do pacote de férias



*Embora o cenário representado no workflow pareça simples, na prática, a composição de um serviço composto é muito mais complexa do que isso. Por exemplo, é preciso incluir mecanismos para lidar com **falhas nos serviços**. Também podem ser necessários **serviços extras** para atender demandas especiais dos viajantes, como o aluguel de uma cadeira de rodas para locomoção no aeroporto.*

Desenvolvimento de Software com Serviços

- O *workflow* também precisa estar preparado para situações em que a execução de um dos serviços é incompatível com um serviço anterior. Por exemplo:

Um voo é reservado para sair em 15 de junho e voltar em 20 de junho. Em seguida, o workflow passa para a fase de reserva de hotel. Porém, o serviço de reserva relata "falta de disponibilidade", pois o hotel está recebendo uma convenção até 17 de junho.

Embora não se trate de uma falha, o sistema precisa desfazer a reserva do voo e passar as informações sobre a falta de disponibilidade para o usuário, que precisará decidir se deseja alterar as datas ou o hotel.

*Na terminologia do workflow, a situação acima é chamada "ação de compensação". **Ações de compensação** são usadas para desfazer ações que já foram concluídas, mas que devem ser alteradas como resultado de etapas posteriores do workflow.*

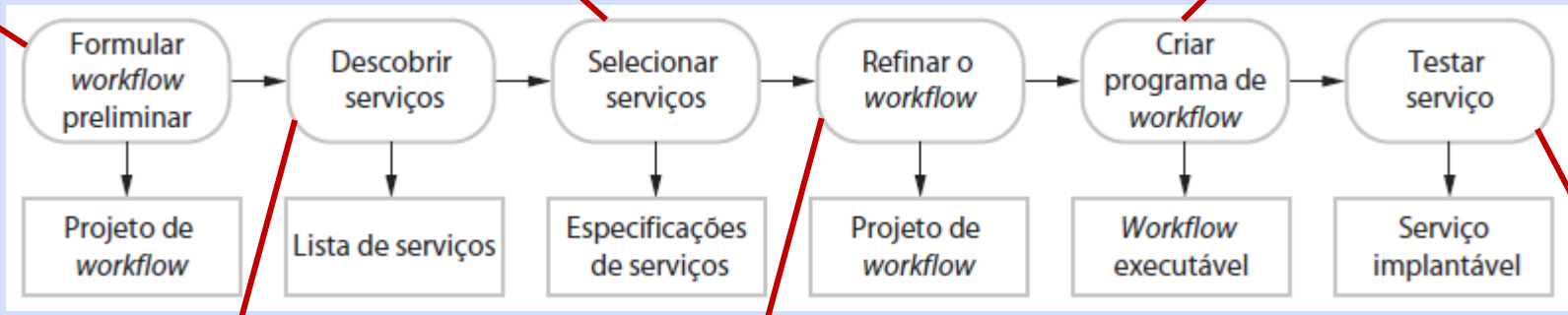
Desenvolvimento de Software com Serviços

- A construção de serviços reusando serviços existentes é antes de mais nada um **processo de projeto de software**. Seus estágios básicos são:

Nesse estágio, é criado um projeto abstrato baseado nos requisitos do serviço composto, com a intenção de adicionar detalhes após saber mais sobre os serviços disponíveis.

Dos serviços descobertos, são selecionados os que contemplam as atividades de workflow, com base em sua funcionalidade, custo e qualidade oferecidos.

O projeto abstrato de torna um programa executável, e a interface do serviço é definida. Pode envolver a criação de interfaces de usuário para acesso ao novo serviço.



Nessa etapa, é feita uma pesquisa para descobrir quais serviços existem, quem fornece esses serviços e os detalhes de fornecimento deles.

Envolve adicionar detalhes e, talvez, remover ou incluir atividades de workflow. Nesse caso, as fases de descoberta e seleção de serviços podem se repetir.

A aplicação e seus serviços são testados juntos. Se houver uso de serviços externos, o processo de teste é mais complexo do que um teste comum de componentes.

Projeto e Implementação de Workflow

- O projeto de *workflow* envolve a análise de processos de negócios existentes ou planejados, para compreender as diferentes atividades realizadas e como as informações são trocadas durante seus fluxos.

- *Workflows* são geralmente representados por meio de uma notação gráfica, como o **diagrama de atividades** da UML ou o **diagrama de *workflow*** da BPMN (*Business Process Model and Notation*).

***BPMN**, ou "Modelo e Notação de Processos de Negócio", é uma notação gráfica para modelar processos de negócios. É provável que o diagrama de workflow da BPMN e o diagrama de atividades da UML sejam integrados no futuro, gerando um padrão único para modelagem de workflows.*

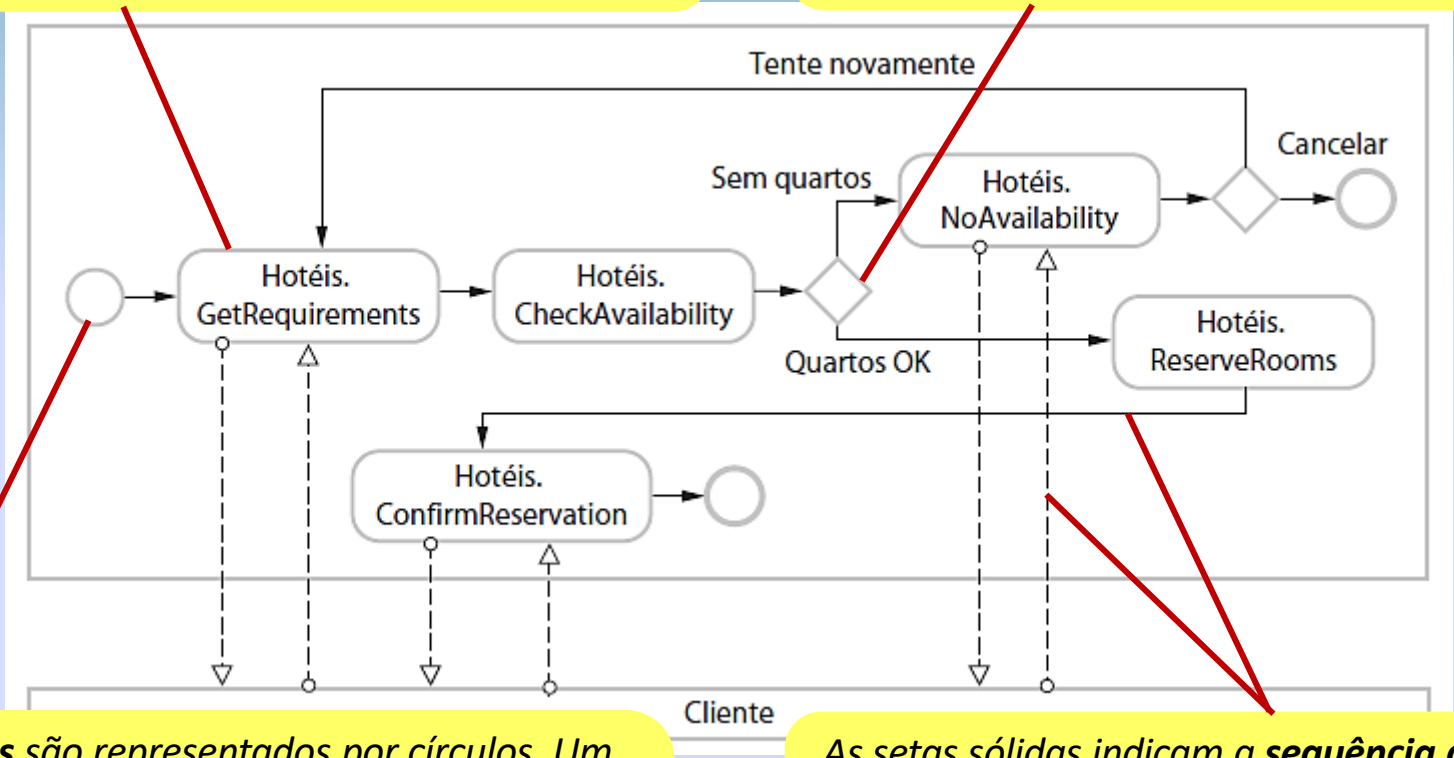
l.bertholdo@ifsp.edu.br

Projeto e Implementação de Workflow

Diagrama BPMN mostrando o *workflow* de **reserva de hotel** do sistema de pacote de férias

*As **atividades** são representadas por retângulos com cantos arredondados. Uma atividade pode ser executada por humanos ou por serviços externos.*

*Os **pontos de decisão** são representados por um diamante. Aqui, são representados os dois caminhos possíveis a respeito da disponibilidade de quartos.*



*Os **eventos** são representados por círculos. Um evento é algo que acontece durante o processo de negócios. Um círculo com borda fina representa o início de um evento, e com borda grossa, o fim.*

*As setas sólidas indicam a **sequência de atividades**. Já as setas tracejadas representam o **fluxo de mensagens** das atividades. Aqui, as mensagens são passadas entre o serviço de reservas de hotel e o cliente.*

Projeto e Implementação de Workflow

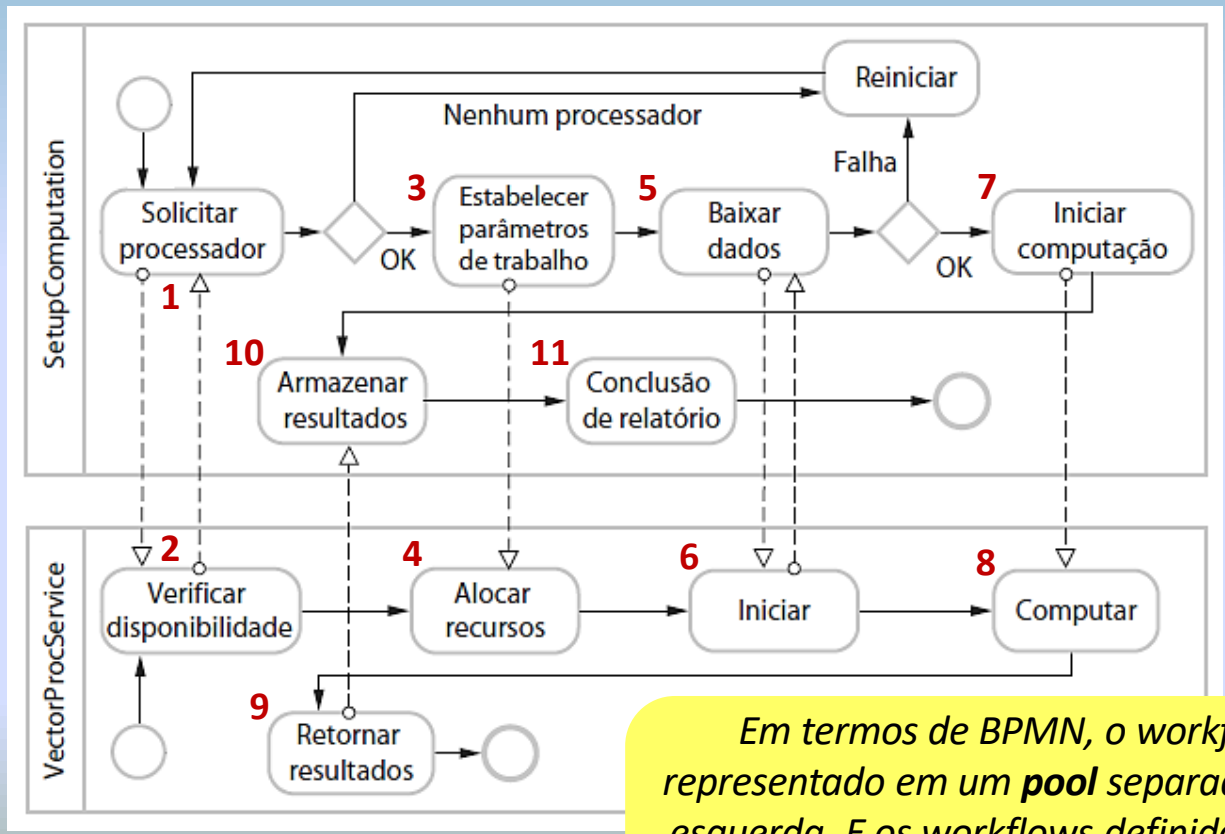
- A BPMN inclui ainda recursos que permitem a conversão automática para um serviço executável. Portanto, *web services*, baseados em composições de serviços descritos em BPMN, podem ser gerados diretamente a partir de um modelo de processo de negócios.
- No entanto, o principal benefício da **modelagem orientada a serviços** é que ela oferece suporte à **computação interorganizacional**, ou seja, processamentos envolvendo serviços em diferentes empresas.
- Isso é representado em BPMN criando *workflows* separados para cada uma das organizações envolvidas.

l.bertholdo@ifsp.edu.br

Projeto e Implementação de Workflow

- Suponha que um computador de processamento de vetores seja oferecido como um serviço (**VectorProcService**) por um laboratório de pesquisa, sendo acessado por outro serviço chamado **SetupComputation**.

Workflows para compartilhamento de computadores de alto desempenho



*O workflow do serviço **SetupComputation** solicita acesso a um processador. Se um processador estiver disponível, estabelece os parâmetros requeridos e envia os dados para o serviço de processamento. Ao concluir a computação, os resultados são armazenados no computador local e um relatório de resultados é gerado.*

*O workflow do serviço **VectorProcService** verifica se há um processador disponível, aloca recursos para a computação, inicia o sistema, realiza o processamento e retorna o resultado para o serviço cliente.*

*Em termos de BPMN, o workflow dos serviços de cada organização é representado em um **pool** separado, com o nome escrito na vertical, na borda esquerda. E os workflows definidos em cada pool são integrados via **troca de mensagens** (o **fluxo entre atividades** dos diferentes pools não é permitido).*

Testes de Serviços

- Testes são importantes em todos os processos de desenvolvimento de software, pois demonstram se o sistema satisfaz a seus requisitos funcionais e não funcionais, além de detectar eventuais defeitos.

- Porém, muitas técnicas de testes, como inspeções de programa e testes de desenvolvimento, dependem da análise do código-fonte do sistema.

- Quando os serviços são oferecidos por um fornecedor externo, o código-fonte não está disponível. Logo, para testar sistemas baseados em serviços externos, não é possível usar técnicas baseadas em código.

- Além disso, por uma série de motivos, os testadores também podem ter mais dificuldades ao testar serviços e composições de serviços.

Testes de Serviços

- Dificuldades enfrentadas ao testar serviços e composições de serviços:

1. *Serviços externos estão sob o **controle do fornecedor do serviço**. Ele pode alterar ou retirar os serviços a qualquer momento, o que invalida quaisquer testes de aplicações anteriores.*
2. *Na visão de SOA, os serviços devem ser **vinculados dinamicamente às aplicações**. Isso significa que uma aplicação talvez não use sempre o mesmo serviço ao ser executada. Logo, os testes podem ser bem-sucedidos quando uma aplicação é vinculada a um determinado serviço, mas não se pode garantir que esse serviço será usado durante a execução real do sistema.*
3. *O **comportamento não funcional** de um serviço não depende simplesmente de como ele é usado pela aplicação que está sendo testada. Por exemplo, um serviço pode executar bem durante o teste, simplesmente porque ele está com pouca demanda de outros clientes naquele momento.*
4. *Se o **modelo de pagamento** pelos serviços for baseado na quantidade de uso, os testes podem ser tornar caros. Ainda que o serviço seja gratuito, seu provedor pode não aceitar que o serviço seja sobrecarregado por aplicações em testes.*
5. *Pode ser difícil simular **falhas específicas** em serviços externos durante o processo de teste, o que pode prejudicar a cobertura completa dos testes.*

SOA e Software como Serviço (SaaS)

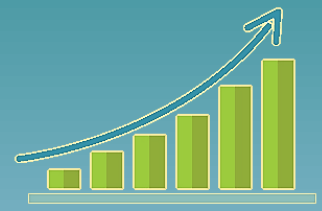
- **SaaS** baseia-se na ideia que um software pode ser pensado como um **serviço remoto**, o qual pode ser acessado de qualquer dispositivo equipado com um *browser*. Alguns exemplos são sistemas de correio, como Gmail, e aplicações de escritório, como Google Docs.

- Nesse sentido, a noção de **SaaS** e as **arquiteturas orientadas a serviços (SOAs)** relacionam-se, porém são conceitos diferentes:

- ***SaaS*** *é uma forma de fornecer funcionalidade a partir de um servidor remoto, em geral, por meio de um browser. O servidor mantém os dados e o estado do usuário durante a sessão de interação que, geralmente, envolve transações longas (por exemplo, edição de um documento).*
- ***SOA*** *é uma abordagem para estruturar um sistema como um conjunto de serviços separados, sem estado do usuário, os quais podem ser fornecidos por vários provedores. Normalmente, envolvem transações curtas, em que um serviço é chamado, faz algo e retorna um resultado.*

l.bertholdo@ifsp.edu.br

Considerações sobre Escalabilidade



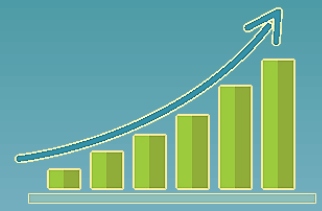
- **Escalabilidade** é um requisito não funcional que reflete a capacidade do sistema de manter a qualidade de seus serviços quando a demanda por eles aumenta. A escalabilidade tem três dimensões:

- ***Tamanho:*** *Deve ser possível adicionar mais recursos a um sistema para lidar com um número crescente de usuários.*
- ***Distribuição:*** *Deve permitir dispersar geograficamente os componentes de um sistema sem comprometer seu desempenho.*
- ***Capacidade de gerenciamento:*** *Deve ser possível gerenciar um sistema à medida que ele aumenta de tamanho, mesmo que partes dele estejam localizadas em organizações diferentes.*

Em termos de ***tamanho****, existe uma distinção entre* ***escalabilidade para cima*** *e* ***escalabilidade para fora****. O primeiro significa a substituição de recursos no sistema por recursos mais poderosos, por exemplo, aumento de memória em um servidor. O segundo refere-se a adicionar recursos ao sistema, por exemplo, incluir um servidor extra para aumentar o número de transações que podem ser processadas em paralelo.*

l.bertholdo@ifsp.edu.br

Considerações sobre Escalabilidade



- Algumas diretrizes gerais para a implementação de sistemas escaláveis são:

1. *Desenvolver aplicações em que cada componente é implementado como um **serviço simples**, o qual pode ser executado em qualquer servidor. Assim, um usuário pode interagir com instâncias do mesmo serviço em execução em diversos servidores.*
2. *Projetar o sistema para que tenha um **comportamento assíncrono** e não precise esperar o resultado de uma interação (por exemplo, uma solicitação de leitura). Isso permite que a aplicação continue a realizar um trabalho útil enquanto está aguardando a interação.*
3. *Gerenciar recursos, como conexões de rede e banco de dados, em um **pool de processadores**, para que nenhum servidor específico corra o risco de ficar sem recursos.*
4. *Projetar o banco de dados para permitir o **bloqueio de baixa granularidade**, ou seja, para não bloquear registros inteiros no banco de dados quando apenas alguns campos deles estão em uso.*

Referências

- PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

- SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.