OTES12 – Tópicos Avançados em Engenharia de Software

Universidade do Estado de Santa Catarina
Centro de Ciências Tecnológicas – DCC

Prof. Dr. William Alberto Cruz Castañeda

2020/2


PREVENTION
PLANNING
DEVELOPMENT
FIXING
OPTIMIZATION
ANALYSIS
BUG
DEFECTS
TESTING
SOFTWARE
MAINTENANCE
PERFORMANCE
MAINTENANCE
ISSUE
IMPROVE
ADAPTIVE
CORRECTION
ENHANCEMENT
LIFE-CYCLE
FUNCTIONALITY
PROCESS
FAULTS

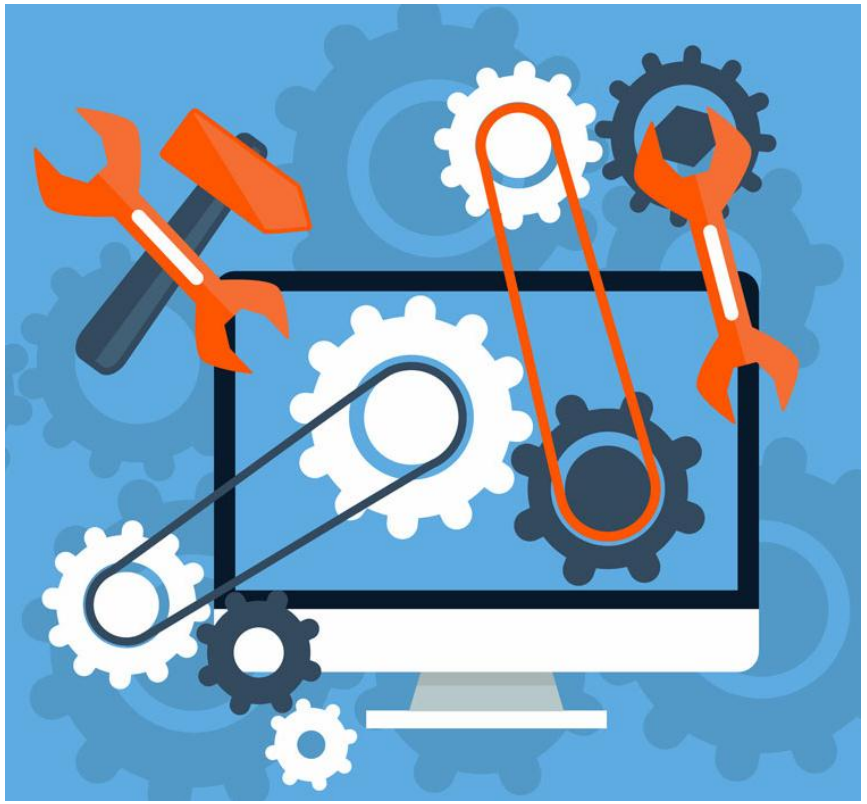

TEST
MONITOR
UPGRADE
RESPOND
FIX

Much of the software we depend on today is on average 10 to 15 years old. Even when these programs were created using the best design and coding techniques known at the time [and most were not], they were created when program size and storage space were principle concerns. They were then migrated to new platforms, adjusted for changes in machine and operating system technology and enhanced to meet new user needs—all without enough regard to overall architecture. The result is the poorly designed structures, poor coding, poor logic, and poor documentation of the software systems we are now called on to keep running . . .

Manutenibilidade é uma indicação qualitativa da facilidade com os quais o software existente pode ser corrigido, adaptado ou aprimorado.

Manutenção de software:

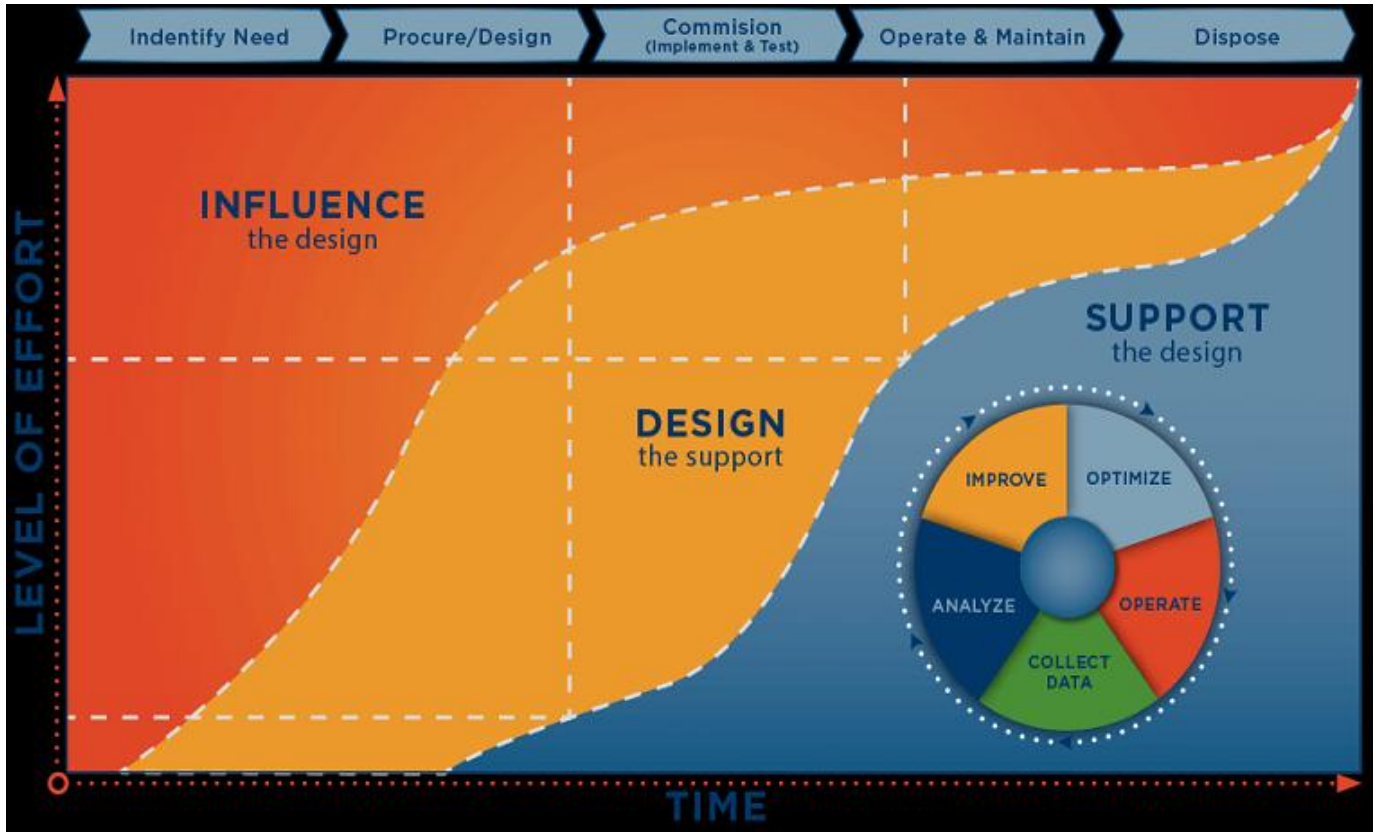
- Apresenta modularidade.
- Utiliza padrões de projeto para facilitar o entendimento.
- Define código fonte autodocumentado e compreensível.
- Passou por técnicas de garantia de qualidade.
- O projeto e a implementação do software devem ajudar a realizar a mudança.



Suportabilidade é a capacidade de suportar um sistema de software durante toda a vida útil do produto.

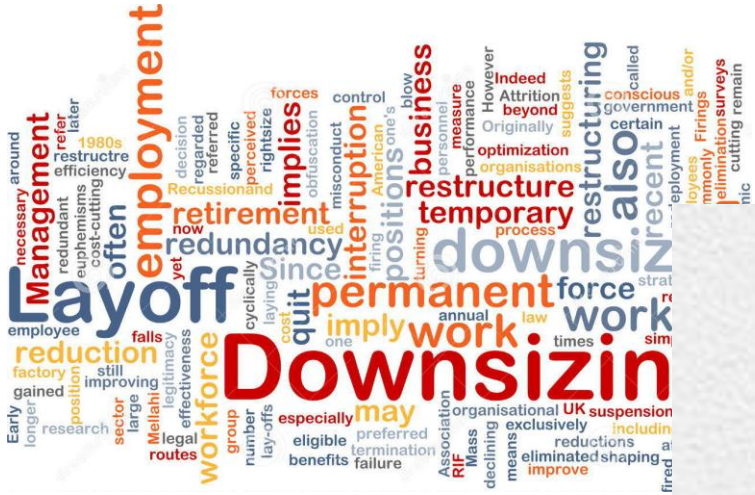
Implica satisfazer:

- Requisitos, fornecer equipamentos, infraestrutura de suporte, software adicional, instalações, mão de obra ou qualquer outro recurso para manter o software operacional e capaz de satisfazer sua função.



É um dos fatores de qualidade que devem ser considerados durante as ações de análise e design que fazem parte do processo de software.

[Reengenharia]

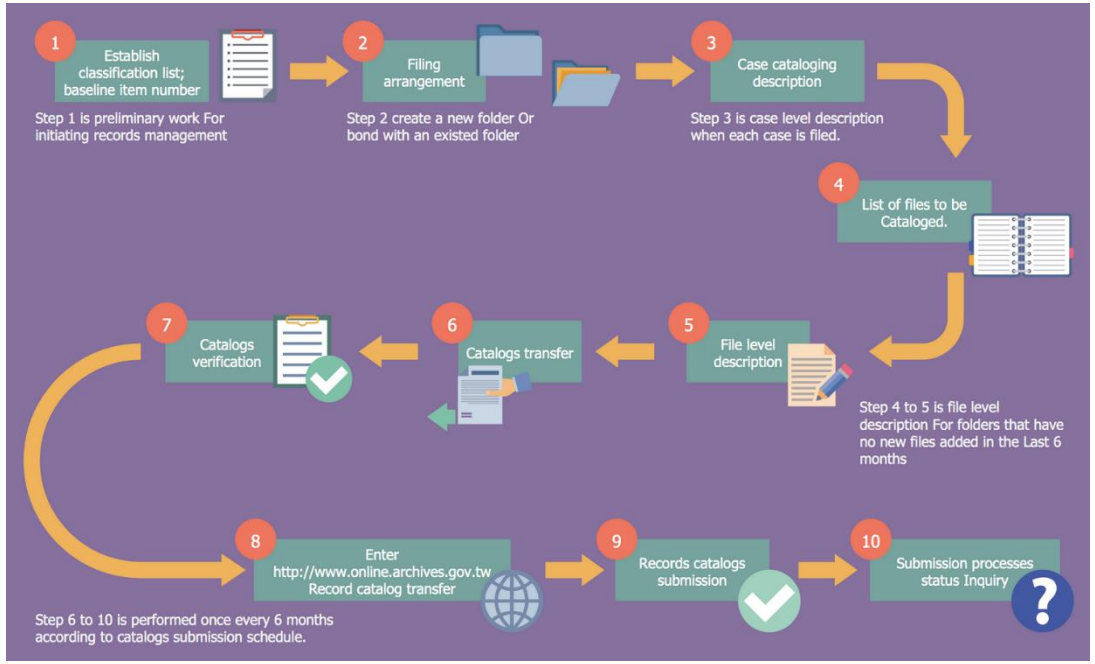


Software Re-engineering

A ligação entre reengenharia de negócios e reengenharia de software baseia-se em uma visão de sistema

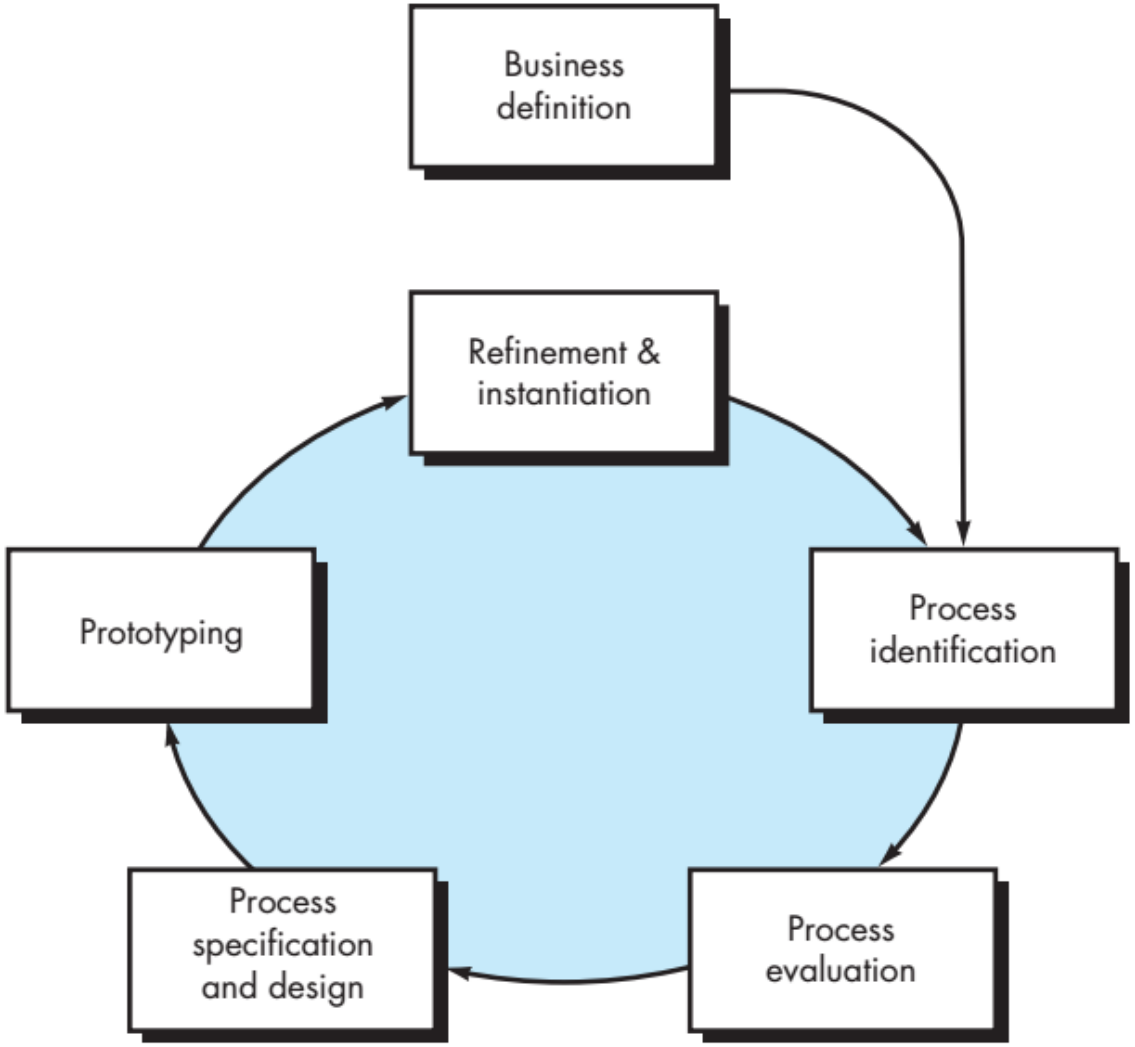
Processo de Negócio

Conjunto de tarefas relacionadas logicamente e executadas para alcançar um resultado comercial definido.



O negócio → sistemas de negócio → processos de negócio → subprocessos de negócio

Business Process Reengineering (BPR) – Reengenharia de Processo de Negócio

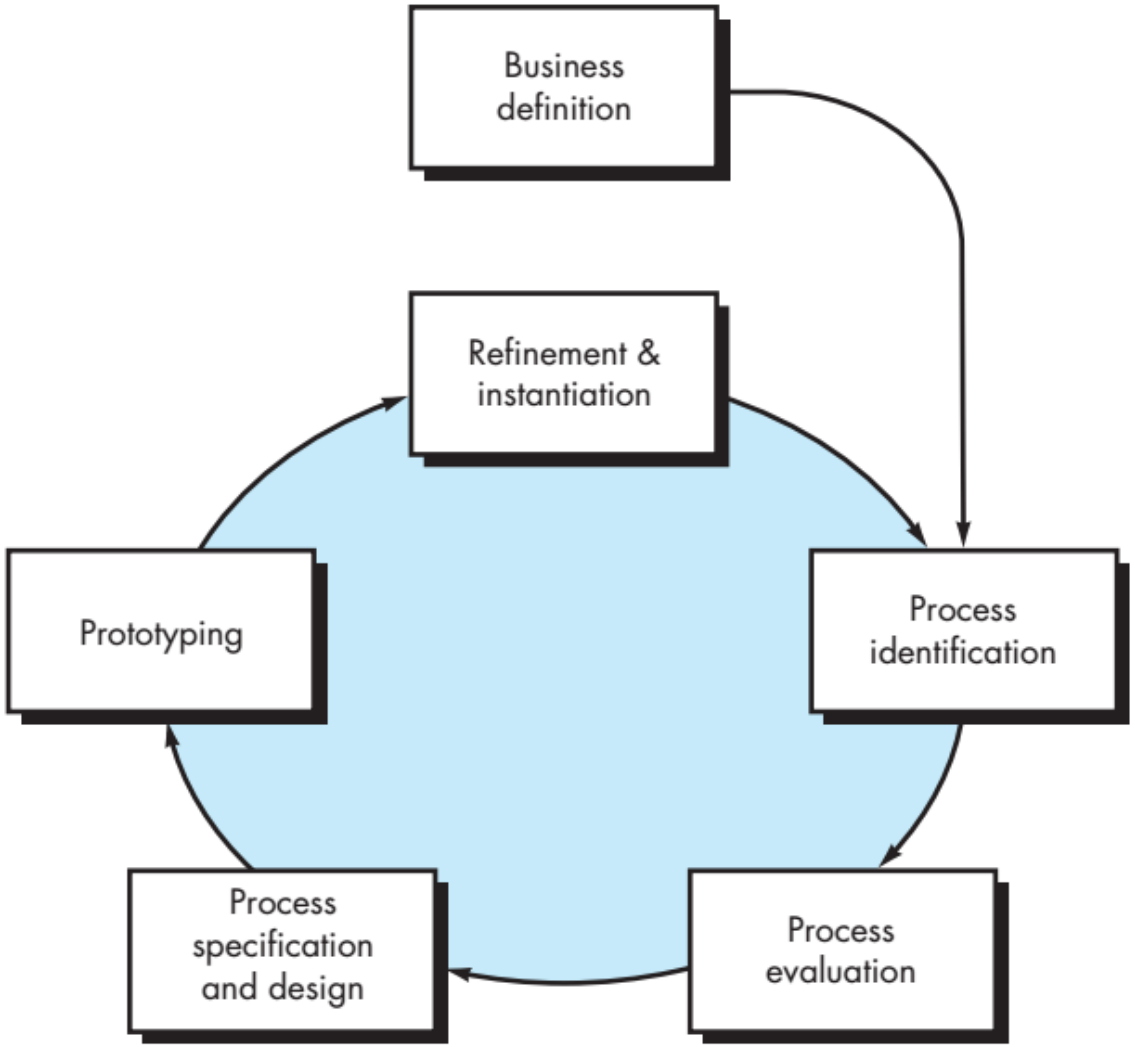


Iterativo e Evolucionário

Definição do negócio – metas identificadas dentro do contexto de redução de custos, redução de tempo, melhoria de qualidade e desenvolvimento de pessoal.

Identificação do processo – classificados por importância, por necessidade de mudança ou qualquer outra forma que seja apropriada para a atividade de reengenharia.

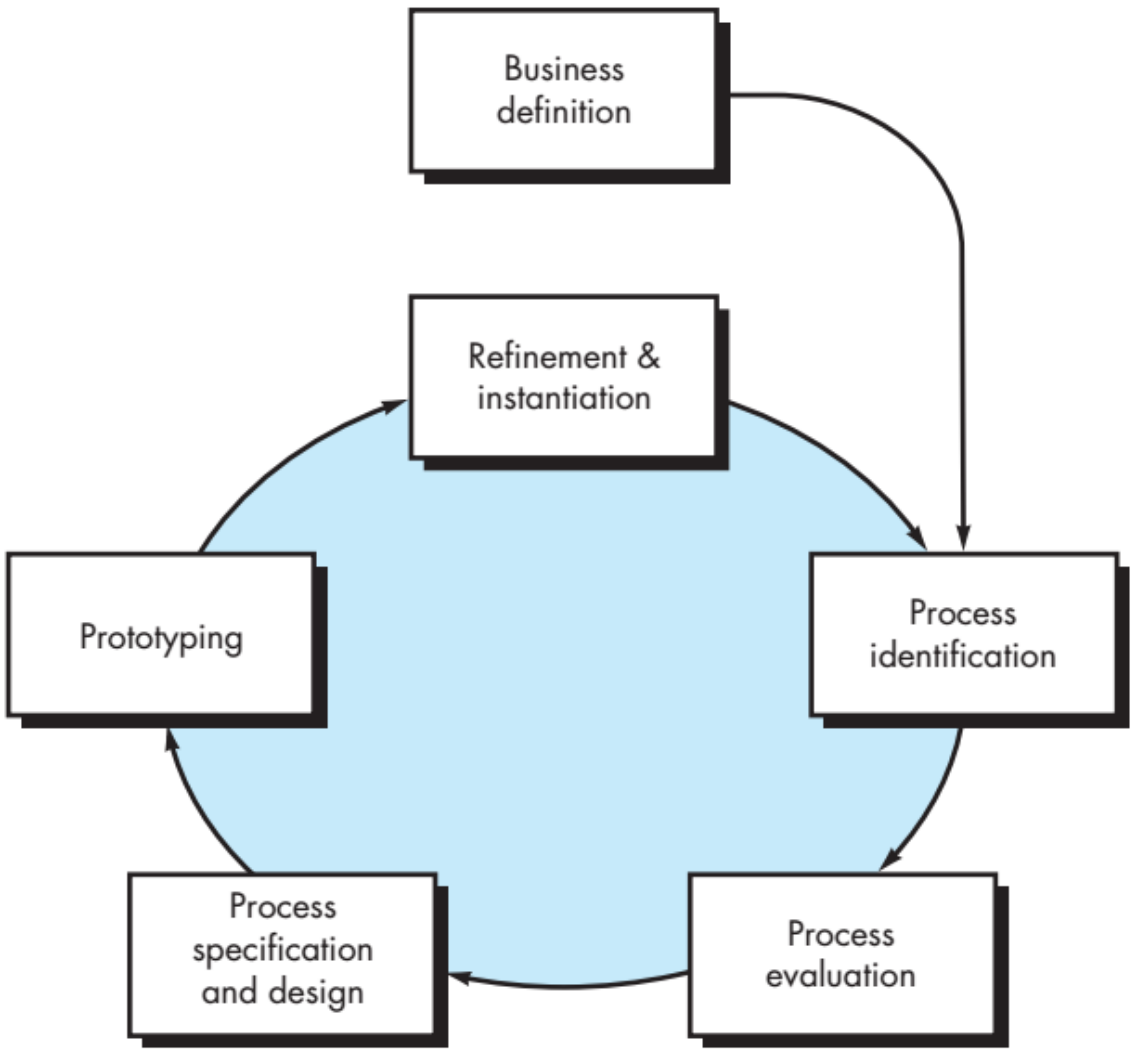
Avaliação de processo – processo existente analisado e medido (custos, tempo, problemas de qualidade / desempenho).



Especificação e projeto do processo – casos de uso são preparados para cada processo que será reprojetado. Casos de uso identificam um cenário que entrega algum resultado para um cliente.

Prototipagem – atividade que testa o processo para que os refinamentos possam ser feitos.

Refinamento e instanciação – feedback do protótipo, o processo de negócios é refinado e, em seguida, instanciado em um sistema de negócios.



[Reengenharia de Software]

Um cenário muito comum...

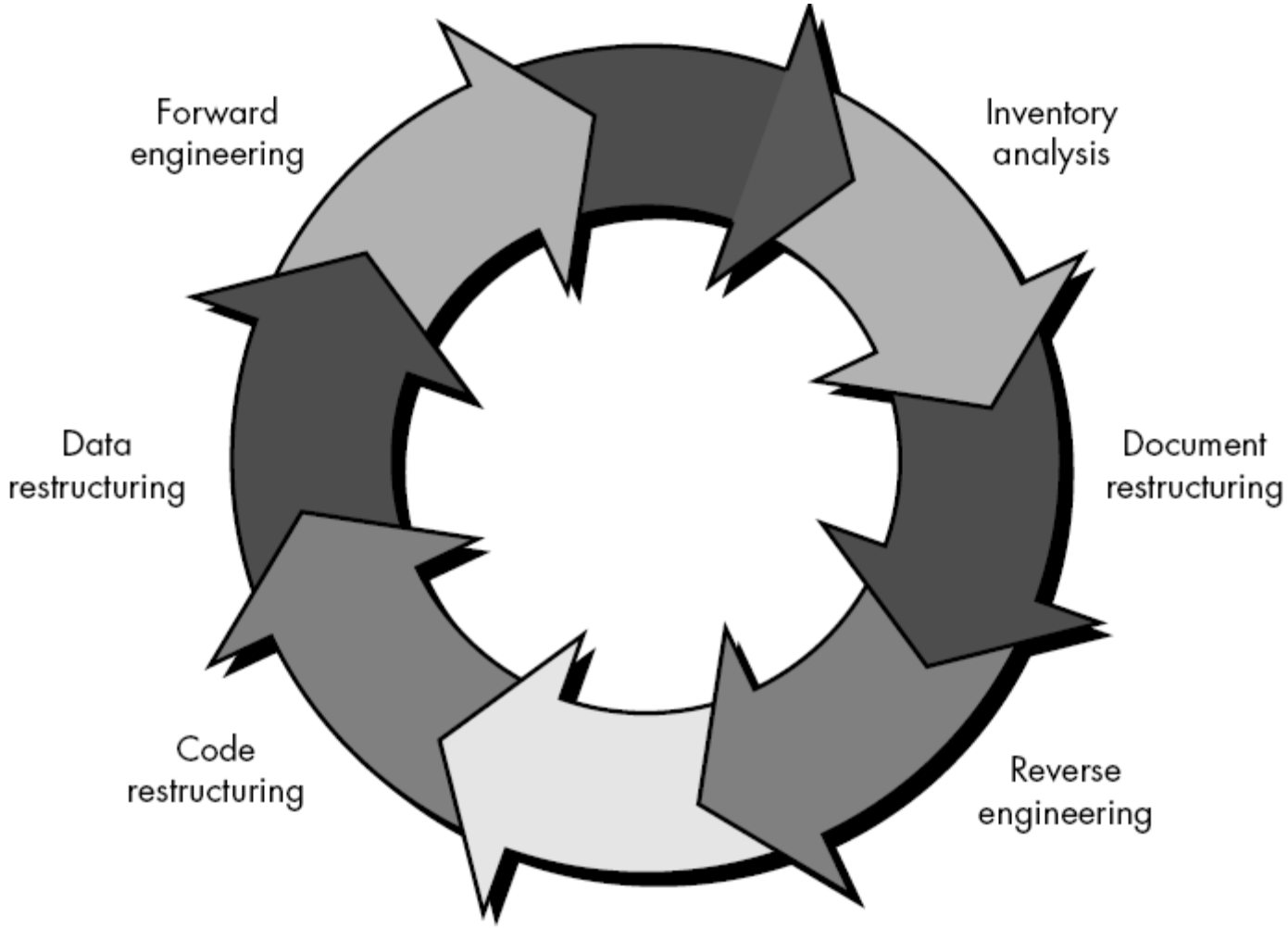
Um aplicativo atendeu às necessidades comerciais de uma empresa por 10 ou 15 anos. Durante esse tempo, ele foi corrigido, adaptado e aprimorado muitas vezes.

As pessoas abordavam esse trabalho com as melhores intenções, mas boas práticas de engenharia de software foram sempre deixadas de lado (devido à urgência de outros assuntos).

Agora o aplicativo está instável. Ainda funciona, mas sempre que se tenta fazer uma alteração, ocorrem efeitos colaterais sérios e inesperados. No entanto, o aplicativo deve continuar evoluindo. O que fazer?

- Reengenharia toma tempo, tem um custo e absorve recursos;
- Realizada em alguns meses ou mesmo anos;

Software Reengineering Process Model



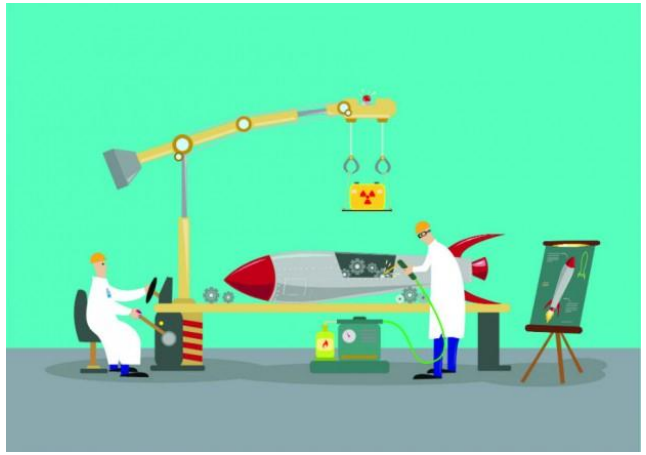
Análise de inventario – toda organização de software deve ter um inventario de todos os aplicativos.

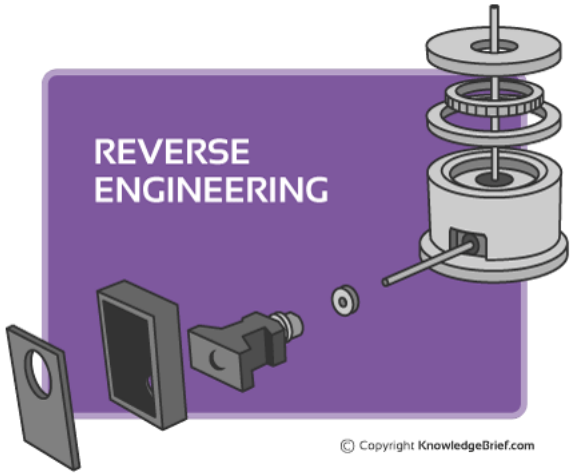
Reestruturação dos documentos – documentação pobre é marca registrada de muitos sistemas legados.

- Criar documentação consome muito tempo, se o sistema funciona, pode optar por conviver com aquilo que tem. Atitude correta em alguns casos.
- Não é possível recriar documentação para centenas de linhas ou programas.
- Programa relativamente estático → fim da sua vida util.
- Documentação deve ser atualizada mas a organização apresenta recursos limitados – Abordagem documentar quando usar mesmo quando o sistema é critico

Engenharia Reversa – resulta em uma ou mais especificações de projeto e fabricação para um produto.

- Utilizada quando nenhuma especificação foi desenvolvida;
- Processo para analisar um programa na tentativa de criar uma representação do programa em um nível mais alto de abstração do que o código fonte;
- Processo de recuperação do projeto;
- Ferramentas extraem informações do projeto de dados, da arquitetura e procedural com base em um programa existente;





Reestruturação do código – código fonte analisado e violações das construções de programação são registradas e código reestruturado

- Código resultante é revisado e testado para evitar introduzir anomalias;

Reestruturação dos dados – atividade que começa como atividade de engenharia reversa.

- Arquitetura de dados atual dissecada e modelos de dados definidos;
- Objetos, estruturas de dados identificados e revisados quanto à qualidade;
- Estrutura de dados fraca passa por uma reengenharia;

Engenharia direta – recupera informações do projeto existente, mas também utiliza-as para alterar ou reconstruir o sistema existente para melhorar a qualidade.

- Arquitetura de dados atual dissecada e modelos de dados definidos;
- Objetos, estruturas de dados identificados e revisados quanto à qualidade;
- Estrutura de dados fraca passa por uma reengenharia;

[Engenharia Reversa]

Processo de criação de um projeto de software para discernir suas regras olhando apenas para o software e seu comportamento.

Esse processo envolve pegar algo que você pode não entender completamente tecnicamente quando começa, e chegar a entender completamente sua função, seus aspectos internos e sua construção.

Uma boa engenharia reversa tenta entender os detalhes do software, o que necessariamente, envolve a compreensão de como funciona o maquinário de computação geral no qual o software funciona.

Uma engenharia reversa requer um profundo conhecimento do hardware e do software e de como tudo funciona junto.

Por que fazer engenharia reversa?

Permite aprender sobre a estrutura de um programa e sua lógica.

Leva a percepções críticas sobre como um programa funciona.

Útil quando explora software.

Vantagens

- Aprender o tipo de funções do sistema que um programa está usando.
- Aprender os arquivos que o programa de destino acessa.
- Aprender os protocolos que o software alvo usa e como ele se comunica com outras partes da rede alvo.

Alterar a estrutura de um programa e, assim, afetar diretamente seu fluxo lógico.

Tecnicamente, essa atividade é chamada de *patching*, porque envolve a colocação de novos patches de código (de uma maneira contínua) sobre o código original, como um patch costurado em um cobertor.

O patch permite adicionar comandos ou alterar a maneira como funcionam as chamadas de funções específicas. Isso permite adicionar recursos secretos, remover ou desativar funções e corrigir bugs de segurança sem código-fonte.

Um uso comum de patching envolve a remoção dos mecanismos de proteção contra cópia.

Como qualquer habilidade, a engenharia reversa pode ser usada para fins bons e ruins.

Ferramentas e Conceitos de Engenharia Reversa

Depurador

Programa que se conecta e controla outros programas de software.

Permite uma única etapa do código, rastreamento de depuração, definição de pontos de interrupção e visualização de variáveis e estado da memória no programa de destino à medida que é executado em etapas.

Determinam o fluxo lógico do programa.

Se enquadram em duas categorias:

- **Modo de usuário** – executados como programas normais no sistema operacional e estão sujeitos às mesmas regras, no entanto, podem depurar apenas outros processos no nível do usuário.
- **Modo kernel** – parte do sistema operacional e pode depurar drivers de dispositivo e até mesmo o próprio sistema operacional.

Ferramentas de Injeção de Falha

Ferramentas que podem fornecer entrada malformada ou formatada incorretamente para um processo de software de destino para causar falhas.

As falhas do programa podem ser analisadas para determinar se existem erros no software de destino.

Algumas falhas têm implicações de segurança, como falhas que permitem que um invasor acesse diretamente o computador host ou a rede.

As ferramentas de injeção de falhas se enquadram em duas categorias:

- **Host** – operam como depuradores e podem ser anexados a um processo e alterar os estados do programa.
- **Rede** – manipulam o tráfego da rede para determinar o efeito no receptor.

Desmontador

Ferramenta que converte código legível por máquina em linguagem assembly.

Desmontadores revelam quais instruções de máquina estão sendo usadas no código. O código de máquina geralmente é específico para uma determinada arquitetura de hardware.

Escritos expressamente para a arquitetura de hardware de destino.

Compilador Reverso ou Descompilador

Ferramenta que converte código de montagem ou código de máquina em código-fonte em uma linguagem de alto nível.

Uteis para determinar a lógica de nível superior, como loops, interruptores e instruções if-then.

[Abordagens de Engenharia Reversa]

Análise de Caixa Branca

Envolve a análise e compreensão do código-fonte.

Às vezes, apenas o código binário está disponível, mas se você descompilar um binário para obter o código-fonte e, em seguida, estudar o código, isso também pode ser considerado uma espécie de análise de caixa branca.

Eficaz em encontrar erros de programação e erros de implementação em software.

Atividade equivale à correspondência de padrões e pode até ser automatizada.

Desvantagem

Pode relatar uma vulnerabilidade potencial onde não existe nenhuma (chamado de falso positivo).

Análise de Caixa Preta

Análise de um programa em execução testando-o com várias entradas.

Esse tipo de teste requer apenas um programa em execução e não faz uso de análise de código-fonte de nenhum tipo.

Se o programa for interrompido durante um teste específico, um problema de segurança pode ter sido descoberto.

Possível mesmo sem acesso ao código binário (um programa pode ser testado remotamente em uma rede).

Não é tão eficaz quanto o teste de caixa branca para obter conhecimento do código e seu comportamento, mas o teste de caixa preta é muito mais fácil de realizar e geralmente requer muito menos experiência do que o teste de caixa branca.

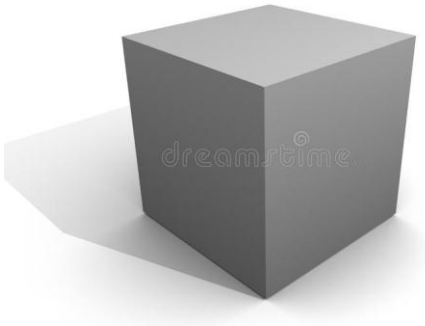
Analista tenta avaliar todos os caminhos de código interno significativos que podem ser diretamente influenciados e observados de fora do sistema.

Análise de Caixa Cinza

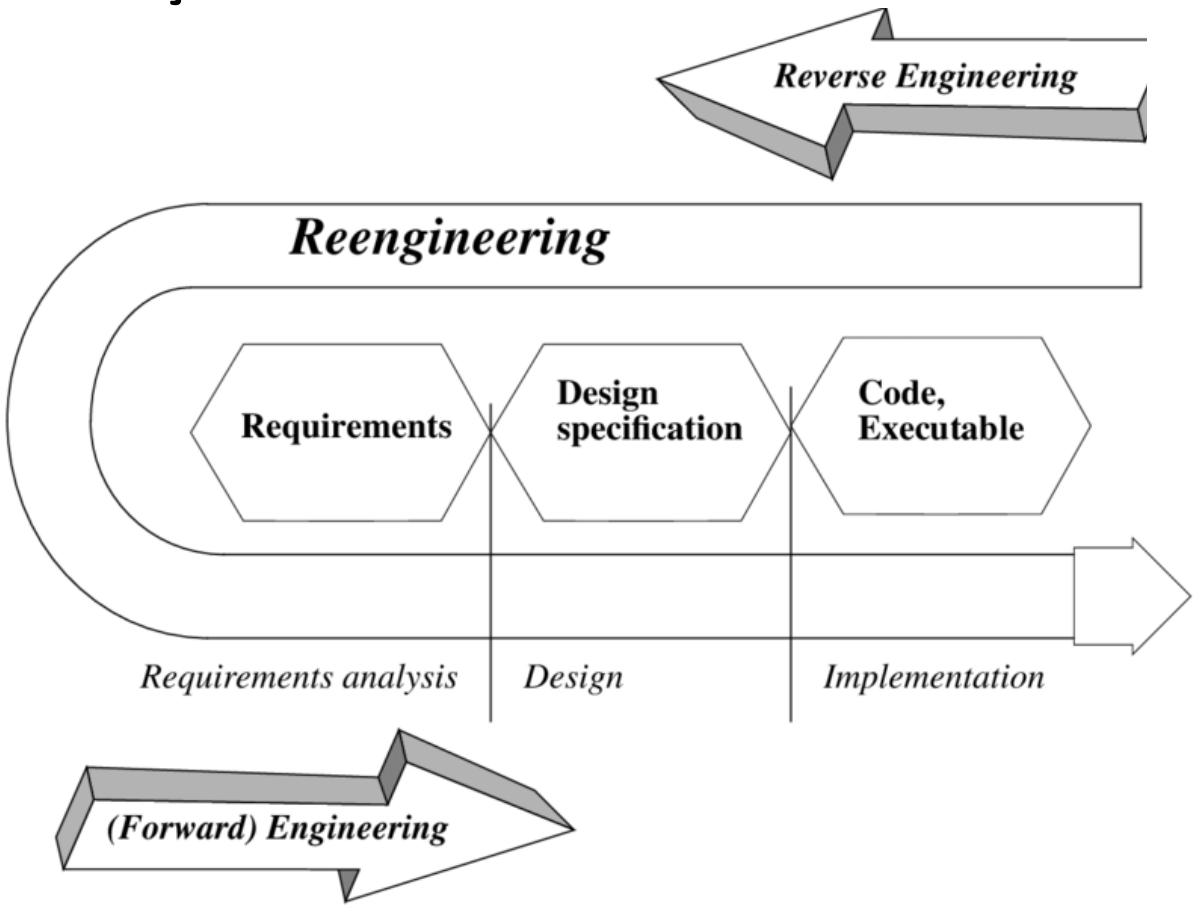
Combina técnicas de caixa branca com teste de entrada de caixa preta.

Requere o uso de várias ferramentas juntas.

Um bom exemplo de uma análise de caixa cinza simples é executar um programa de destino em um depurador e, em seguida, fornecer conjuntos específicos de entradas para o programa.



Pode extrair informações do projeto do código-fonte, mas o nível de abstração, a completeza da documentação são variáveis.



Nível de Abstração

Sofisticação das informações de projeto que podem ser extraídas do código-fonte.

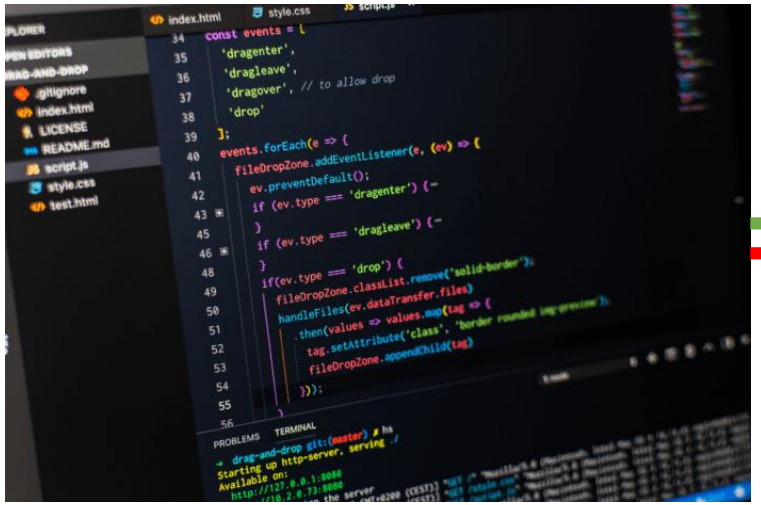
Deve ser capaz de:

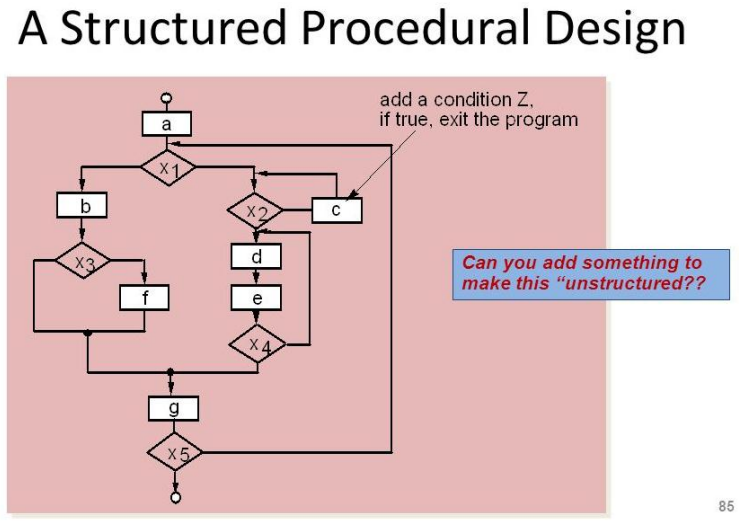
- Derivar representações de projeto procedural;
- Informações de programa e de estrutura de dados;
- Modelos de objeto, dados e modelos de fluxo de controle;
- Modelos de entidade-relacionamento;

Abstração aumenta → entender o programa

Completeza

- Nível de detalhe fornecido em um nível de abstração.
- Diminui conforme abstração aumenta.





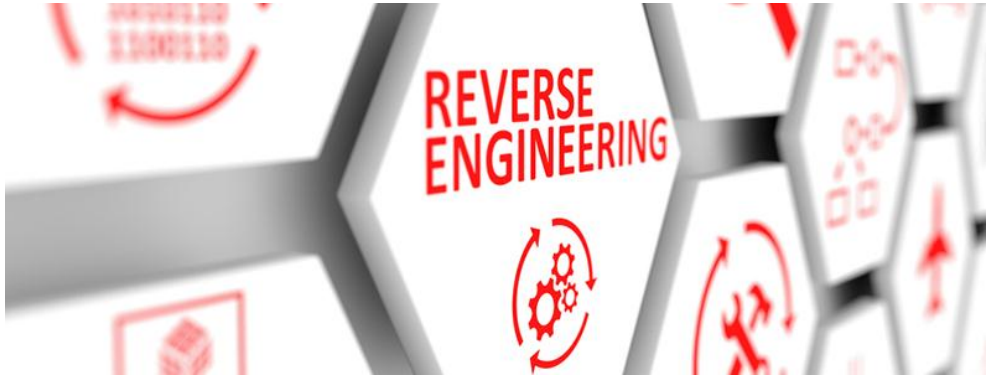
Processo de Engenharia Reversa

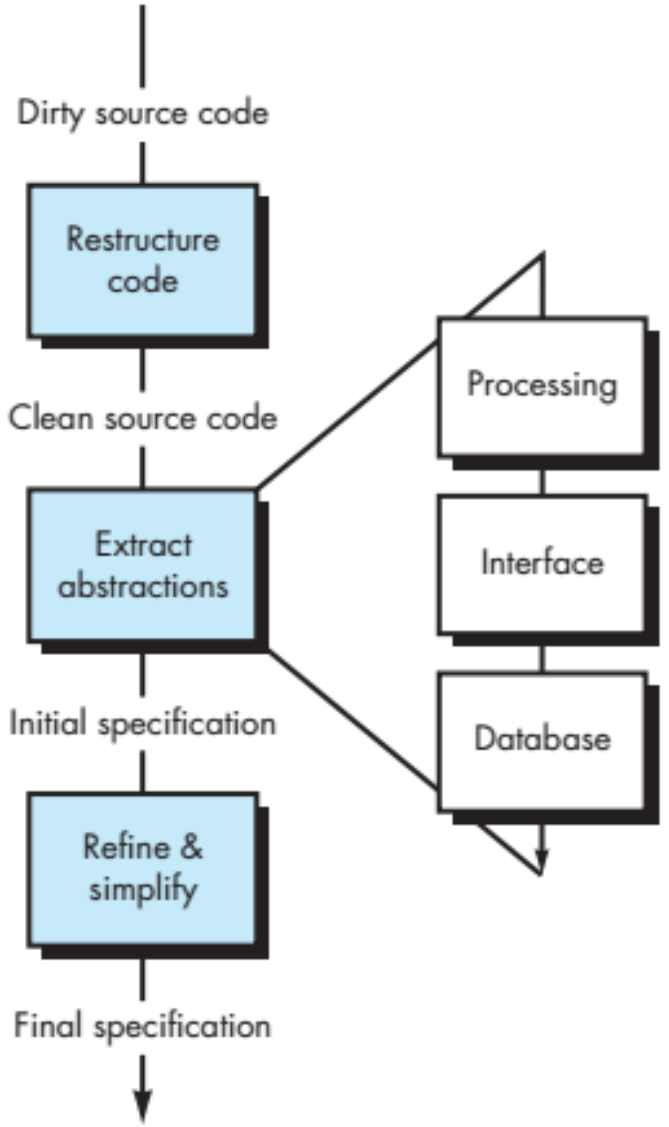
Engenharia reversa para entender os dados:

- Estruturas internas de dados;
- Estrutura de base de dados;

Engenharia reversa para entender o processamento

Engenharia reversa das interfaces de usuário



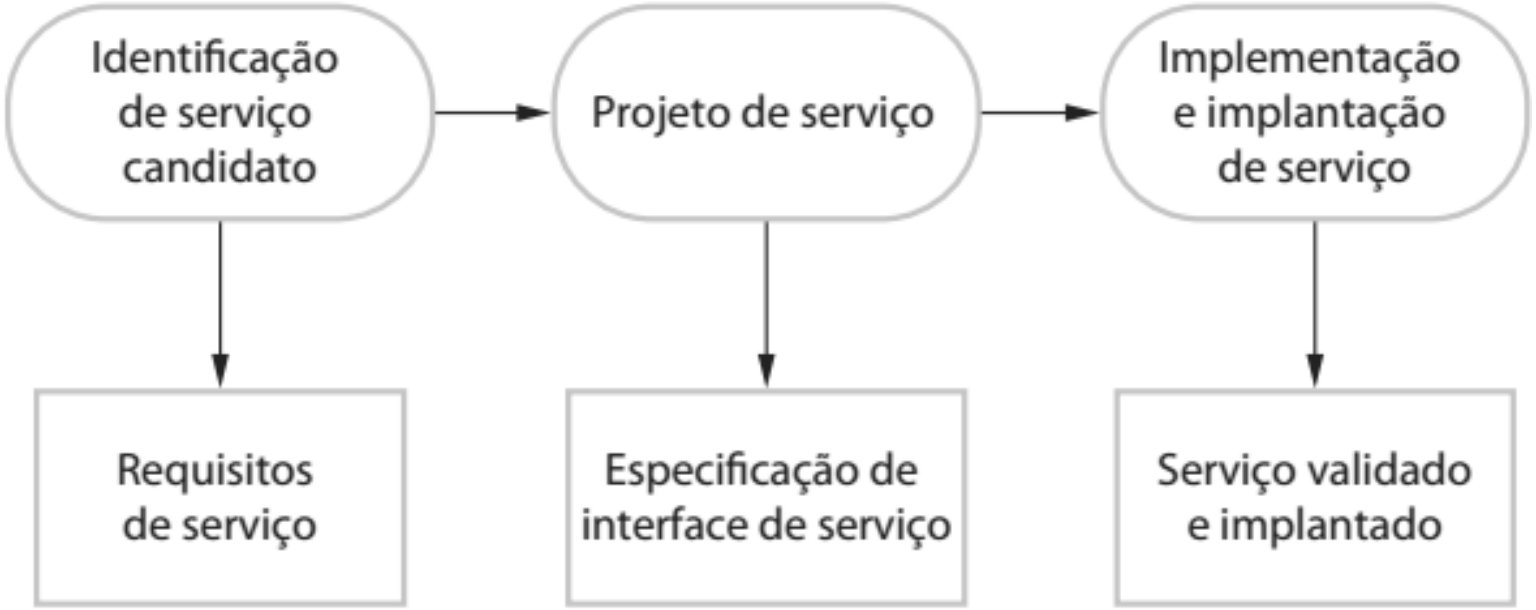


[Engenharia de Serviços]

Processo de desenvolvimento de serviços para reúso em aplicações orientadas a serviços.

- Em comum com a engenharia de componentes, já que o serviço deve representar uma abstração reusável que poderia ser útil em diferentes sistemas, além de robusto e confiável.
- Documentar o serviço para que possam ser descobertos e compreendidos pelos usuários.

1. Identificação de serviço candidato – identifica os possíveis serviços que podem ser implementados e define os requisitos de serviço.
2. Projeto de serviço – projeta a lógica e as interfaces de serviço (SOAP e/ou REST).
3. Implementação e implantação de serviço – implementa e testa os serviços tornando-os disponíveis para uso.



Identificação de Serviço Candidato

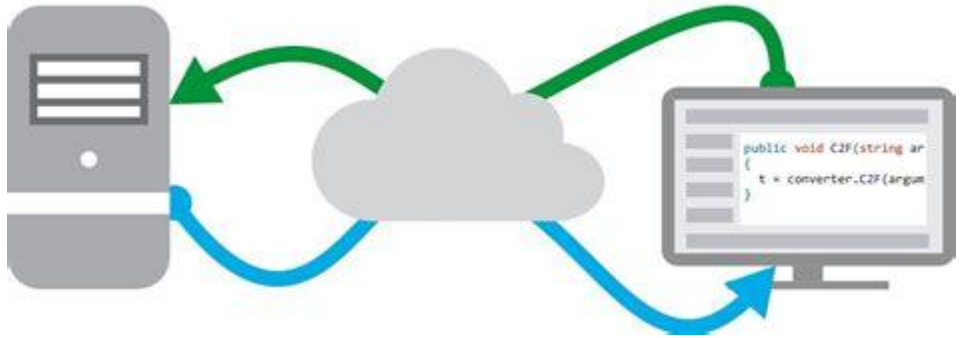
Envolve compreensão e análise dos processos de negócios para decidir quais serviços reusáveis poderiam ser implementados para dar suporte a esses processos.

Três tipos fundamentais de serviços que podem ser identificados:

1. **Serviços utilitários** – implementam alguma funcionalidade geral, que pode ser usada por diferentes processos de negócios.
2. **Serviços de negócio** – associados a uma função específica dos negócios.
3. **Serviços de coordenação ou de processo** – suportam um processo de negócios amplo que geralmente envolve atividades e atores diferentes.

Projeto de Serviço

- Envolve a definição das operações associadas com o serviço e seus parâmetros.
- Objetivo é minimizar o número de trocas de mensagens que deve acontecer para se completar a solicitação de serviço.
- Garante que a informação seja passada por uma mensagem, em vez de interações síncronas.



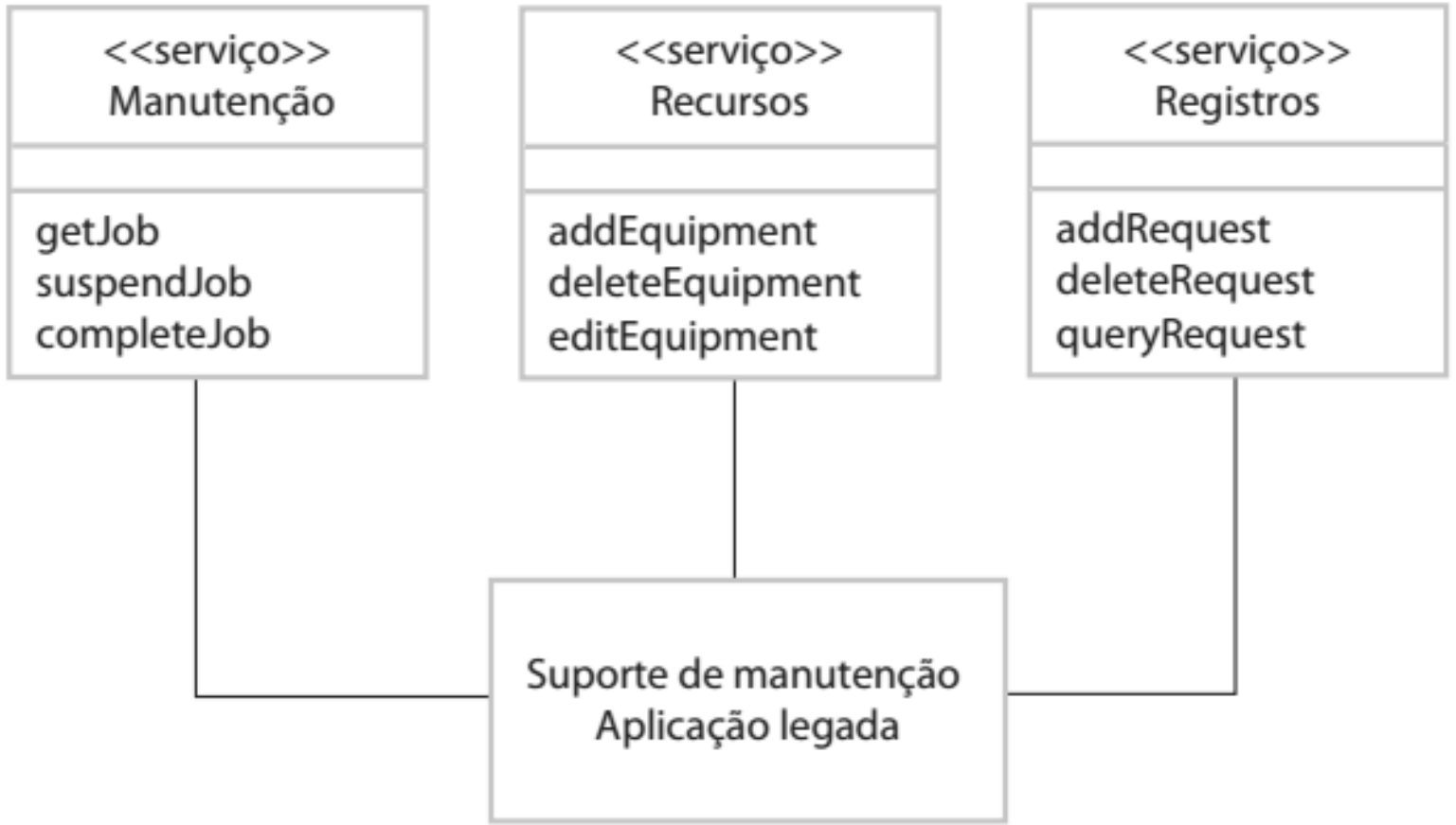
Existem três estágios de projeto de interface de serviço:

1. Projeto lógico de interface, em que você identifica as operações associadas ao serviço, suas entradas e saídas e as exceções associadas a essas operações.
2. Projeto de mensagem, em que você cria a estrutura das mensagens que são enviadas e recebidas pelo serviço.
3. Desenvolvimento WSDL e/ou REST, em que você traduz o projeto lógico e de mensagem para uma descrição da interface abstrata escrita em WSDL e/ou REST.

Implementação e Implantação de Serviço

Envolver programação de serviço usando uma linguagem de programação padrão. Inclui o uso de bibliotecas com suporte para o desenvolvimento de serviços.

Serviços fornecendo acesso a um sistema legado



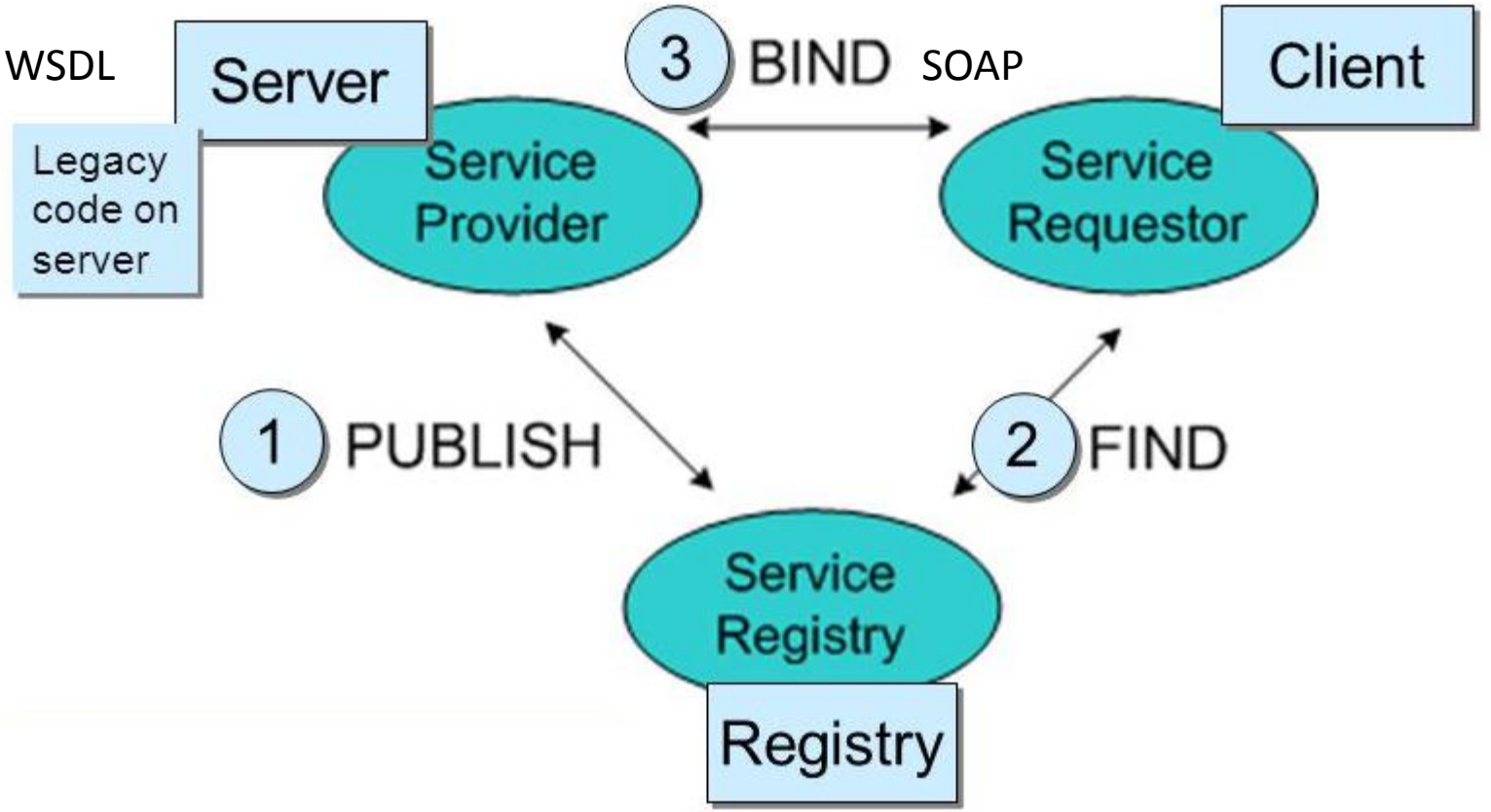
[Arquitetura Orientada a Serviços]

Objetivos de aprendizado

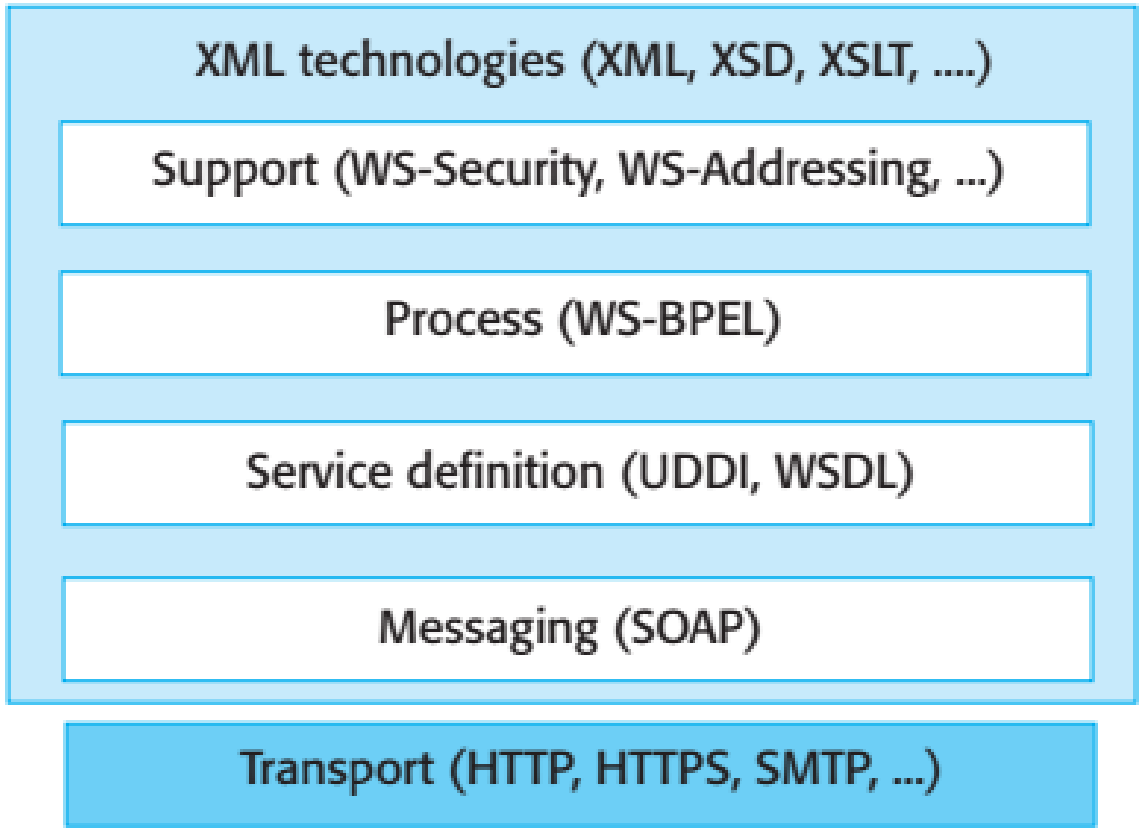
- Apresentar a arquitetura de software orientada a serviços como uma forma de criação de aplicações distribuídas usando web services;
- Compreender conceitos de web service, padrões de web services e arquitetura orientada a serviços;
- Compreender a ideia de serviços RESTful e as diferenças entre serviços RESTful e SOAP;
- Entender o processo da engenharia de serviços que se destina a produzir web services reusáveis;
- Composição de serviços como um meio de desenvolvimento de aplicações orientadas a serviços;

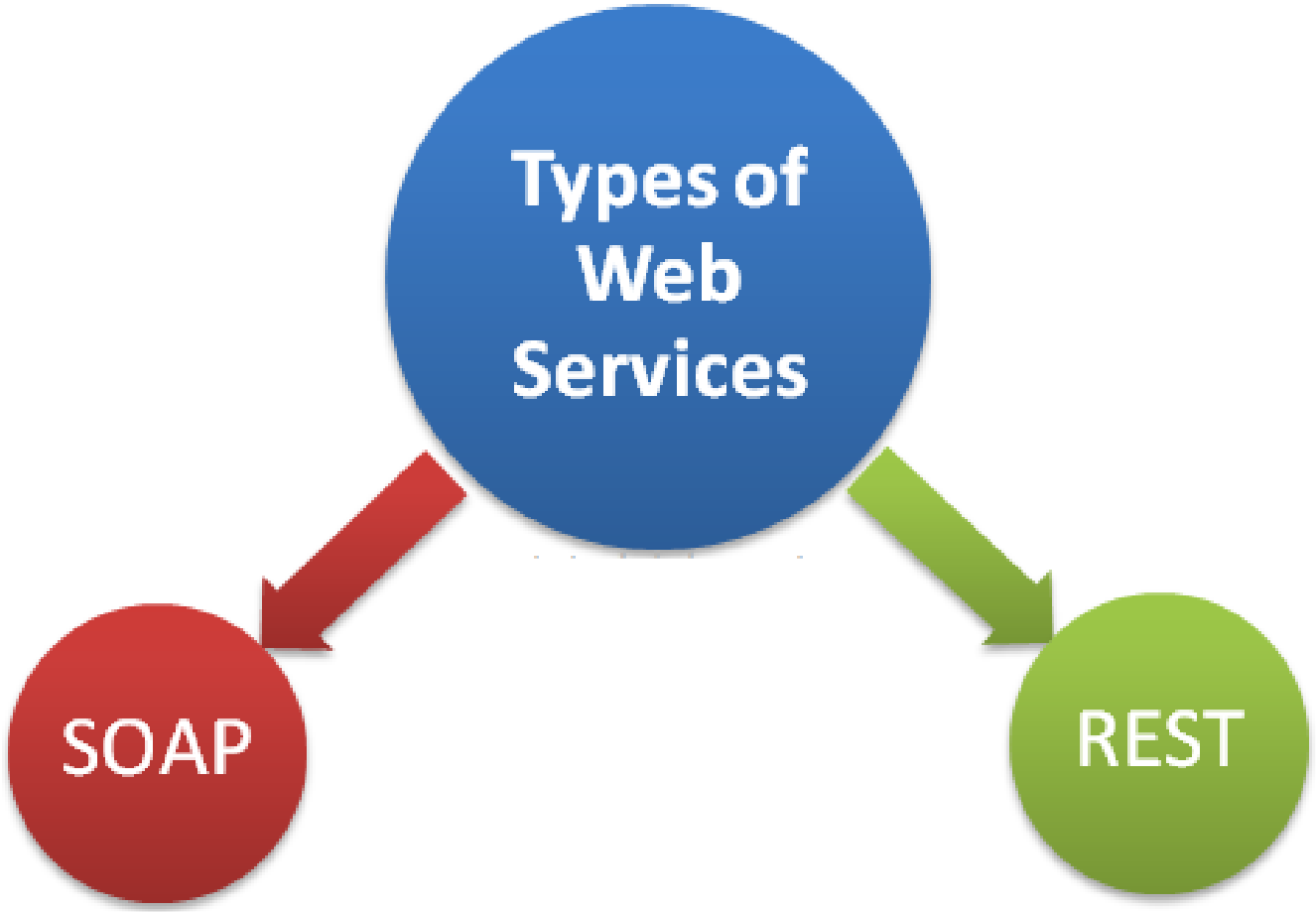
Arquitetura Orientada a Serviços – SOA

- Estilo de arquitetura baseado na ideia de que serviços executáveis podem ser incluídos em aplicativos.
- Os serviços têm interfaces bem definidas e publicadas, e os aplicativos podem escolher se são apropriados ou não.
- O serviço pode estar disponível em diferentes provedores e que os aplicativos podem tomar uma decisão em tempo de execução de qual provedor de serviços usar.


WSDL
Server
Legacy code on server
3 BIND
SOAP
Client
Service Provider
Service Requestor
1 PUBLISH
2 FIND
Service Registry
Registry

Padrões Internacionais para oferecer suporte a Web Services




Types of Web Services
SOAP
REST

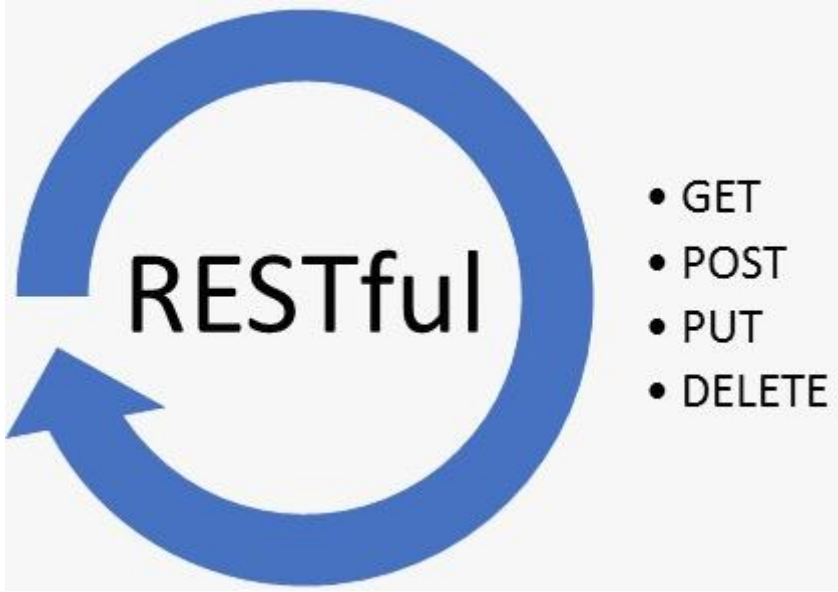
REST

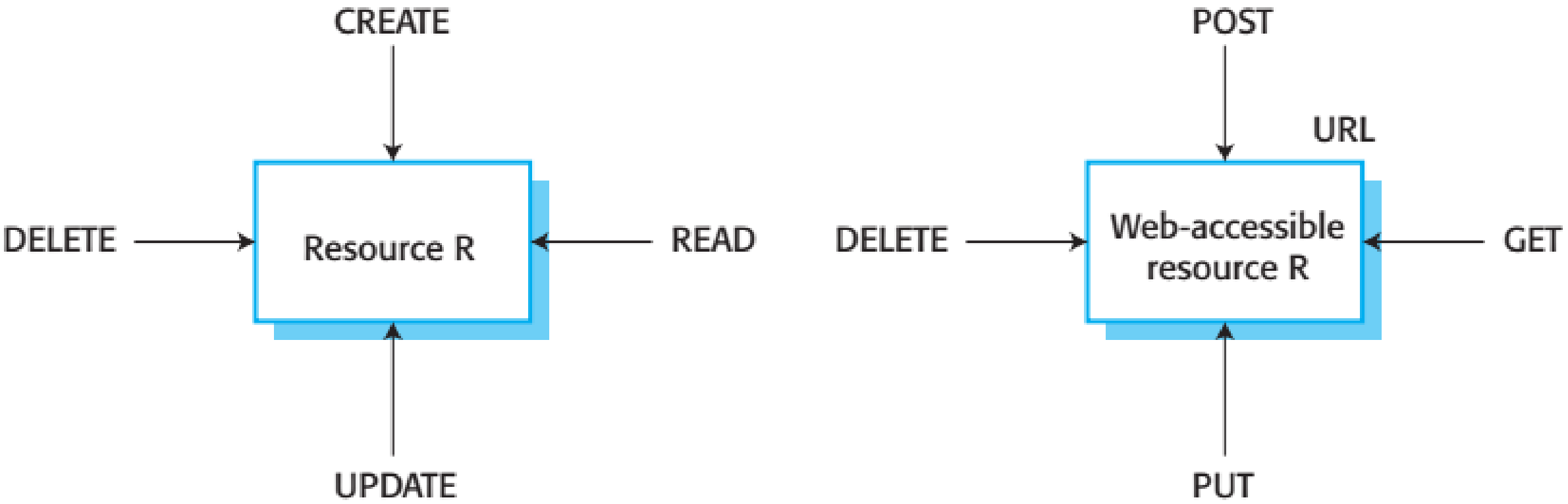
- Estilo de arquitetura baseado na transferência de representações de recursos de um servidor para um cliente.
- Elemento fundamental em uma arquitetura REST → Recurso;
- Em uma arquitetura RESTful, tudo é representado como um recurso;
- Recursos possuem identificador exclusivo (URL);

RESTful
Web Services

Operações Fundamentais

1. ***Create*** – coloca um recurso em existência.
2. ***Read*** – retorna uma representação do recurso.
3. ***Update*** – altera o valor do recurso.
4. ***Delete*** – torna o recurso inacessível.

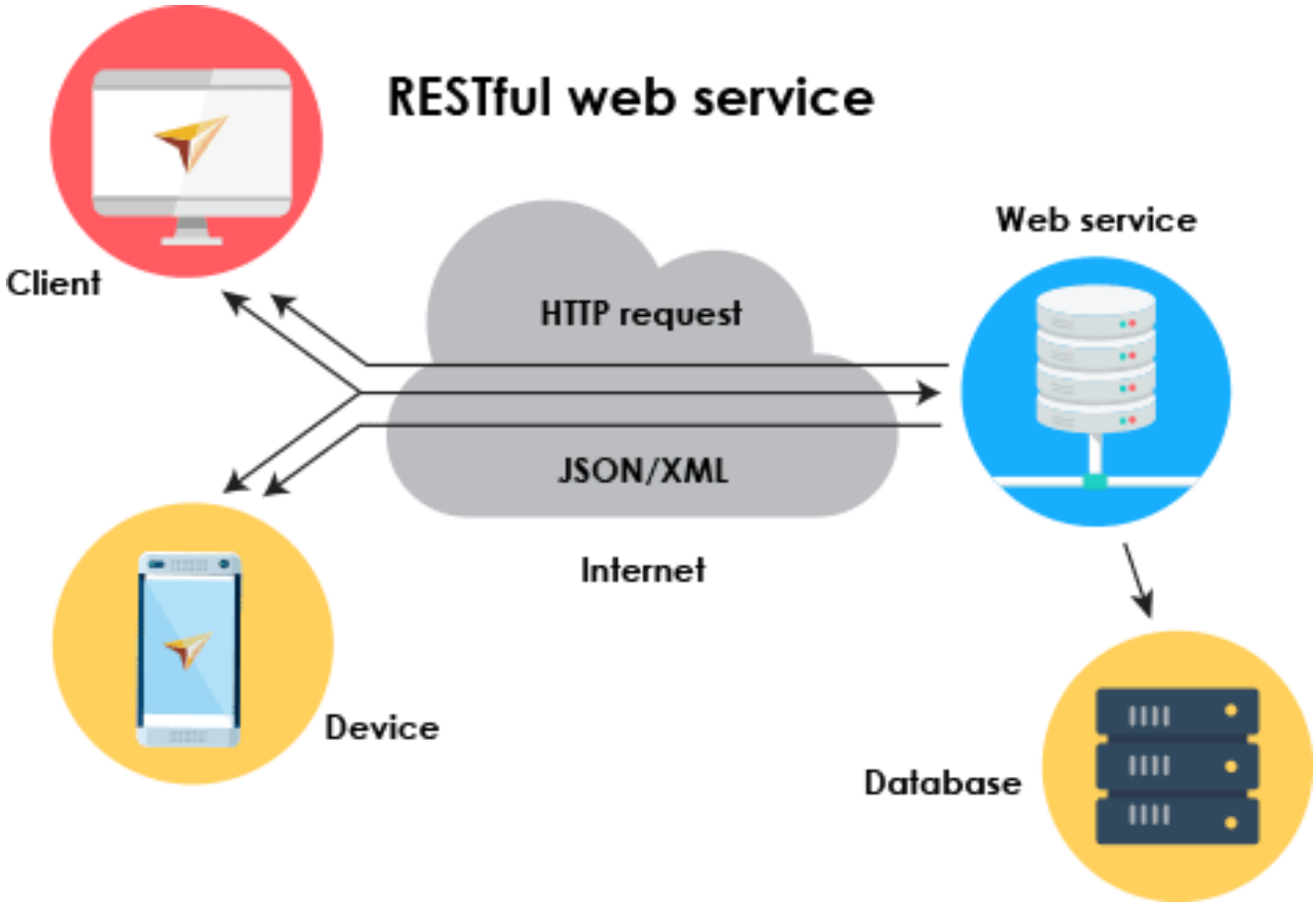



CREATE
DELETE
Resource R
READ
UPDATE
POST
URL
DELETE
Web-accessible resource R
GET
PUT

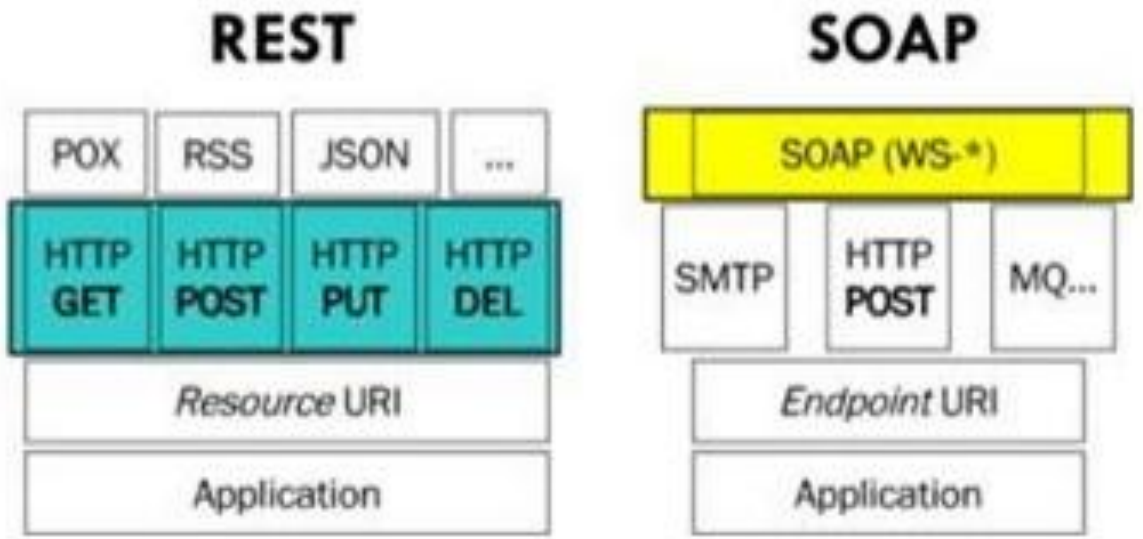
Abordagem no qual os clientes usam URLs e as operações HTTP **GET**, **PUT**, **DELETE** e **POST** para manipular recursos representados em XML/JSON.

- Ênfase na manipulação de recursos de dados e não nas interfaces;

- Quando um novo recurso é criado, tem um novo URL pelo qual pode ser acessado ou atualizado;



Protocol Layering



Consider the cutie on the left as your data

SOAP

Client  +  =  <- -> Server

Data + SOAP Standards

Your data sent to the server would become as huge as the chubby above

REST

Client 

<- ->

Rest is like sending the DATA as such

Server

OTES12 – Tópicos Avançados em Engenharia de Software

Universidade do Estado de Santa Catarina
Centro de Ciências Tecnológicas – DCC

Prof. Dr. William Alberto Cruz Castañeda

2020/2