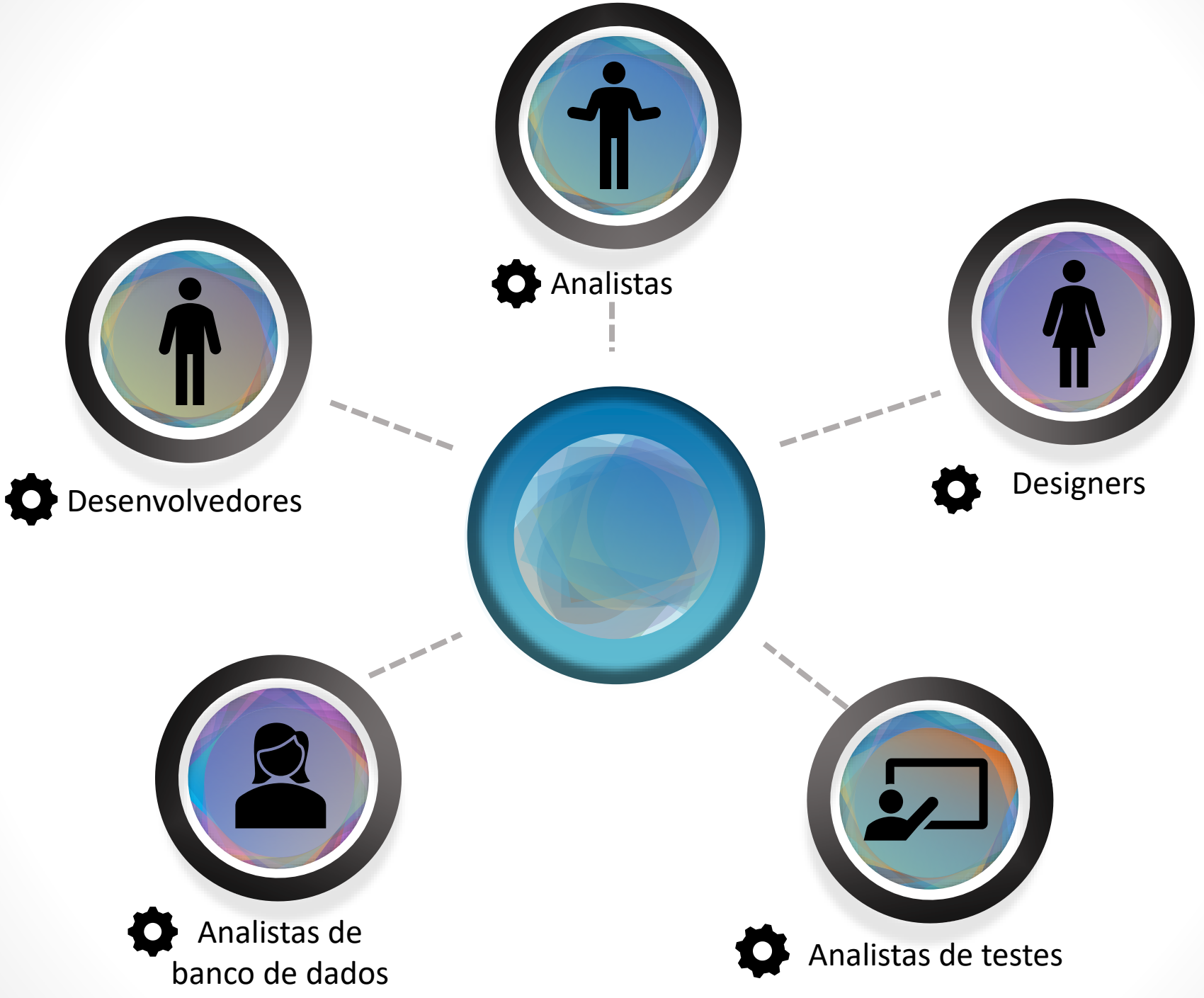
GSI003 - 2019-2

MÉTODOS

Guilherme Goulart
Paulo Diego
Rafaela Peres
Vinicius Silva Martins


Analistas
Desenvolvedores
Designers
Analistas de banco de dados
Analistas de testes


Cliente
Metodologia
Desenvolvedores

❖ Manifesto ágil

❖ Mais adaptáveis que previsíveis

❖ Mais orientado a pessoas que processos

Métodos ágeis

“A XP é uma maneira leve, eficiente, de baixo risco, flexível, previsível, científica e divertida de desenvolver software”

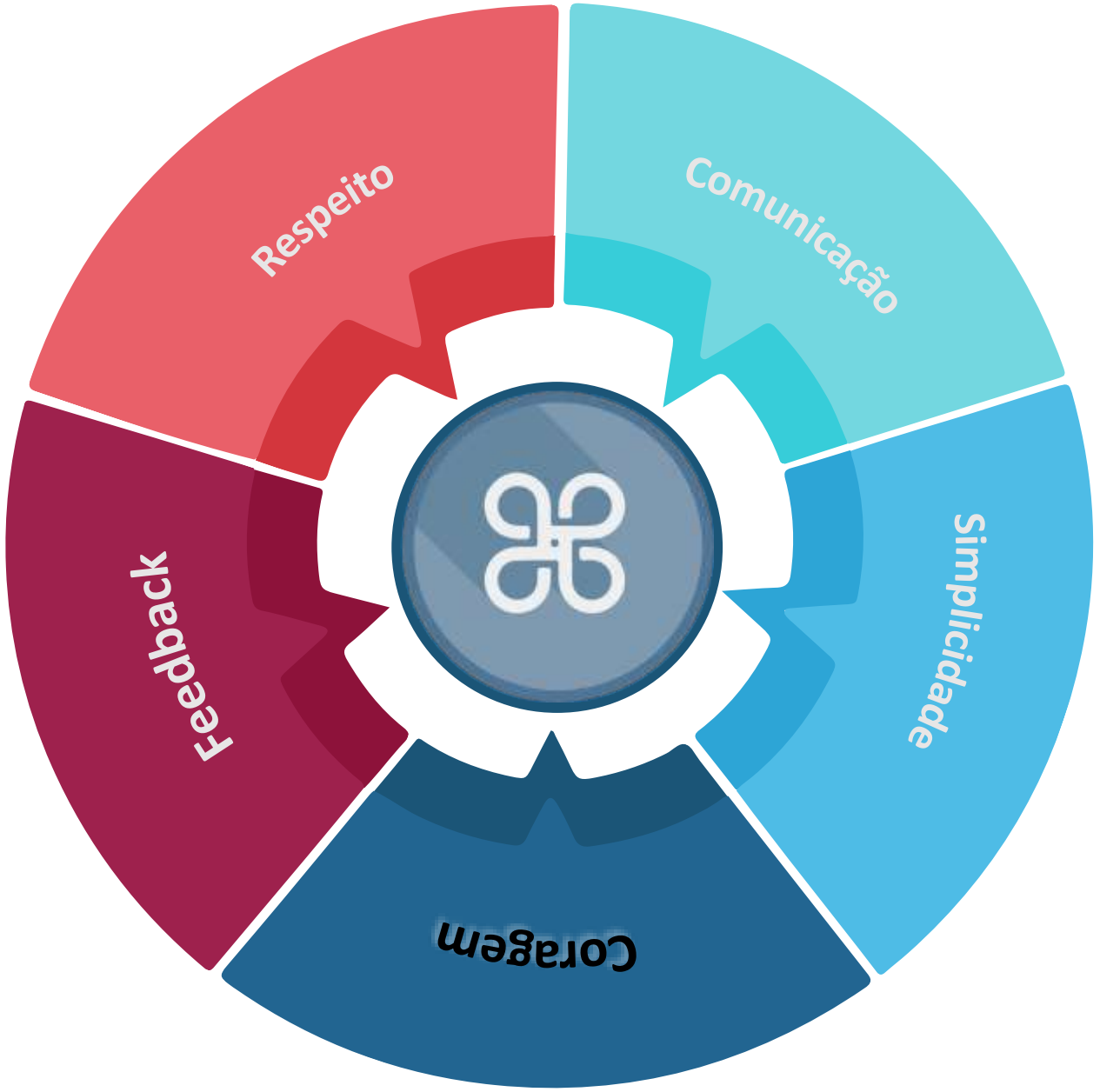
Kent Beck

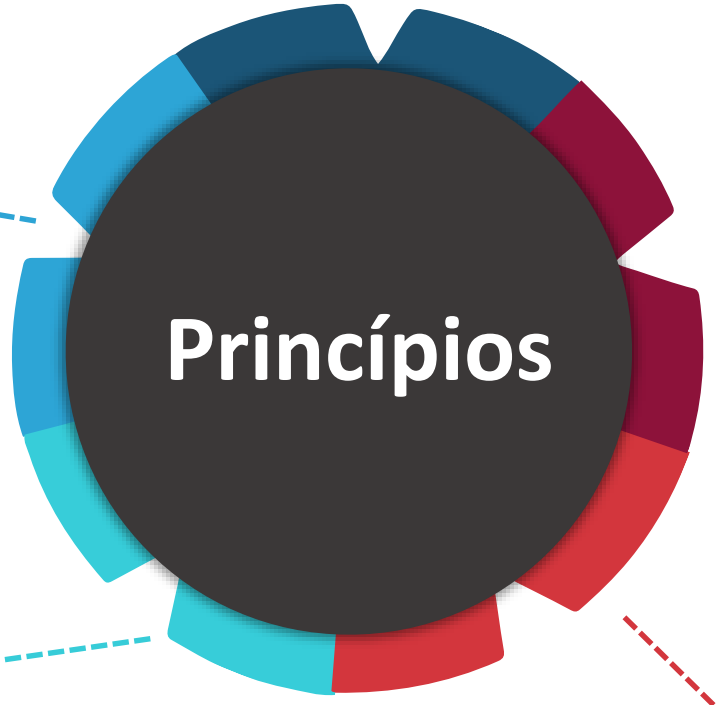
Características

- Equipes pequenas e médias
- Projetos com requisitos vagos
- Orientado a objetos
- Desenvolvimento incremental
- Codificação

Ênfases

- Menor em processos formais
- Maior em disciplina rigorosa


Respeito
Comunicação
Simplicidade
Coragem
Feedback



❖ Fazer mudanças incrementais

❖ Prover feedback

❖ Abraçar mudanças

❖ Assumir simplicidade

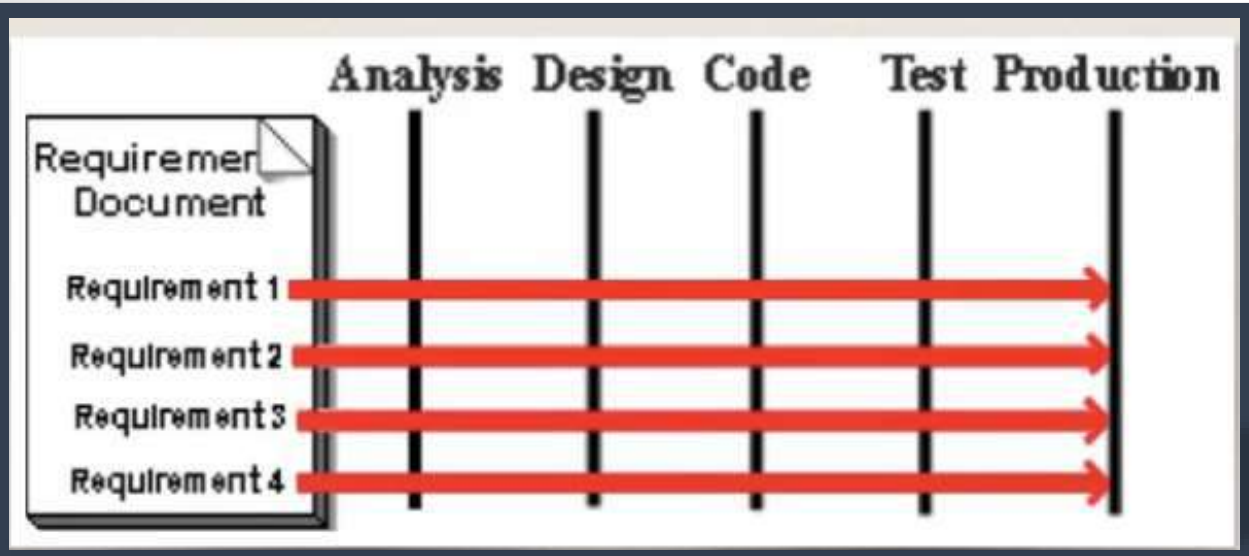
❖ Trabalho de qualidade

REGRAS

Extreme Programming Planning (Planejamento)

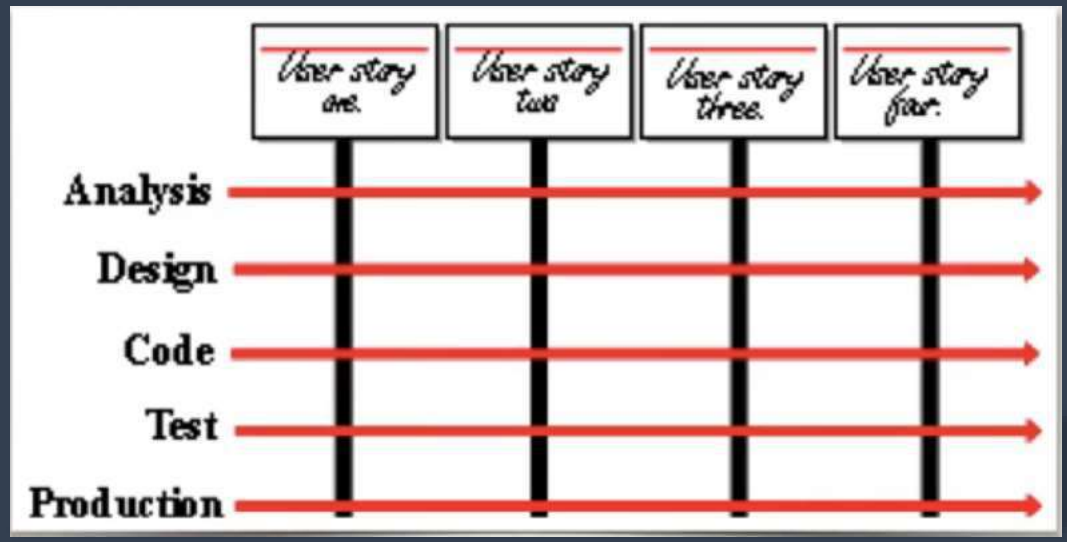
- Estórias dos usuários;
- Será criado um planejamento de liberação para criar a programação de liberação;
- Projeto é dividido em iterações e lançamentos frequentes;
- O planejamento de iteração vai iniciar cada iteração.

REGRAS



Tradicional

XP



REGRAS

Estória do Usuário
<Nome do Projeto>

Estimativa da Estória em horas
X hrs

<Nº> Nome da Estória

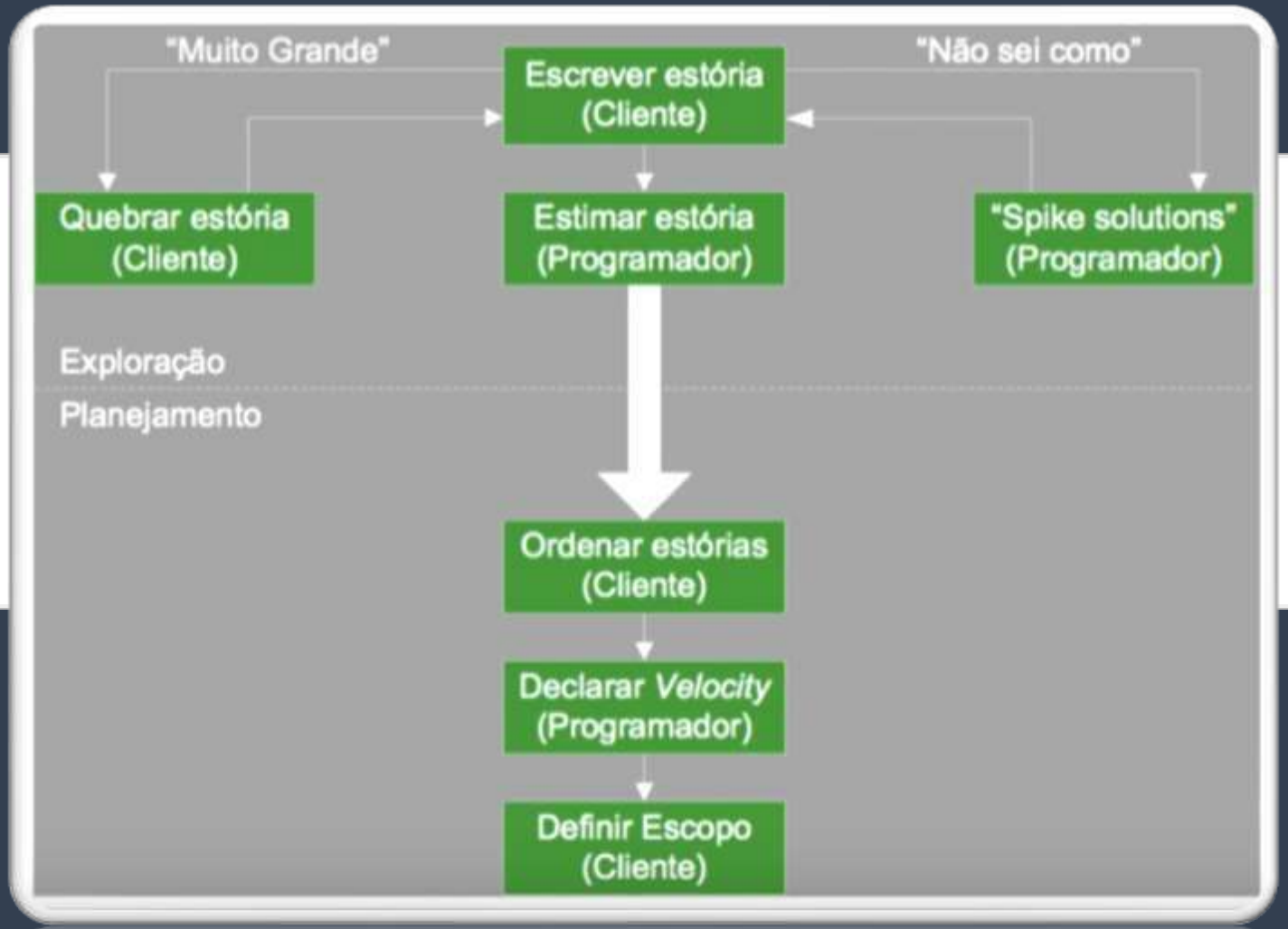
Prioridade da Estória
☐ essencial ☐ importante ☐ desejável

Descrição da Estória. Frase curta definindo a funcionalidade ou restrição.

Tarefas:

- Lista de Tarefas da Estória
- As tarefas são as decomposições da historia
- ou seja, atividades que seguidas, implementam a funcionalidade.

Planning Game



REGRAS

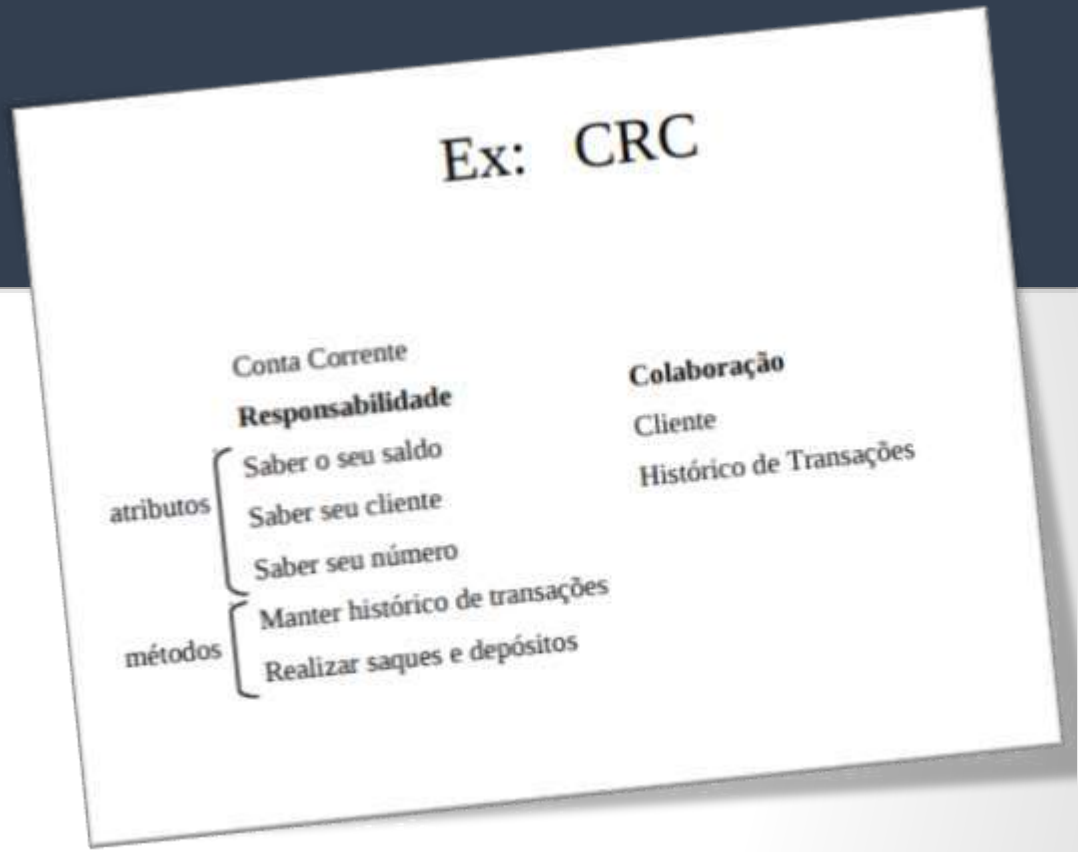
Gerenciamento

- Espaço de trabalho aberto e dedicado;
- Definição de ritmo sustentável;
- Reunião stand up todos os dias;
- Medição da velocidade do projeto;
- Movimento das pessoas;
- Correção do XP, quando necessário.

REGRAS

Projetando

- Escolher metáfora do sistema;
- Usar cartões CRC para sessões de design;
- Criar soluções de pico para reduzir riscos;
- Cuidado na adição das funcionalidades com muita antecedência;
- Refatorar sempre que possível, para aprimorar a concepção do software.



REGRAS

Codificação

- O cliente esteja disponível e presente;
- Desenhar diagramas (Extreme Programming Diagram) que serão inscritos nos códigos;
- Padronização do código;
- Codificar o teste de unidade primeiro;
- Equipes em pares;
- Apenas um par vai integrar um código por vez;
- Integração contínua;
- Propriedade coletiva.

REGRAS

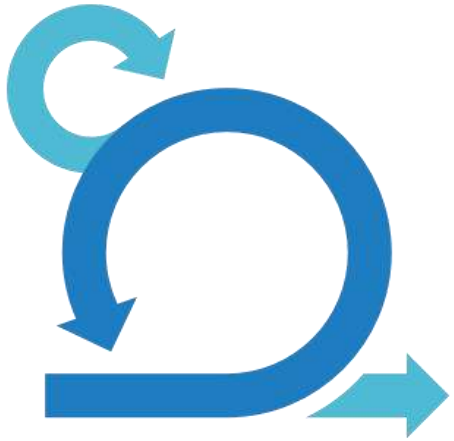
Testes

- Quando um erro é encontrado, os testes são criados;
- Os testes de aceitação são realizados frequentemente e a pontuação é executada.

REGRAS

- O cliente sempre disponível
- Uso de metáforas no projeto
- Testes de Aceitação
- Primeiro os testes
- Refatoração - melhoria constante do código

- Pequenas versões
- Planejando o jogo
- Integração Contínua
- Simplicidade de Projeto



Scrum

Scrum

Scrum é uma metodologia ágil para gestão e planejamentos de projetos.

Scrum - Características

Sprint Planning Meeting – Reunião de planejamento com o Product Owner.

Sprint – Ciclos tipicamente mensais para realizar atividades.

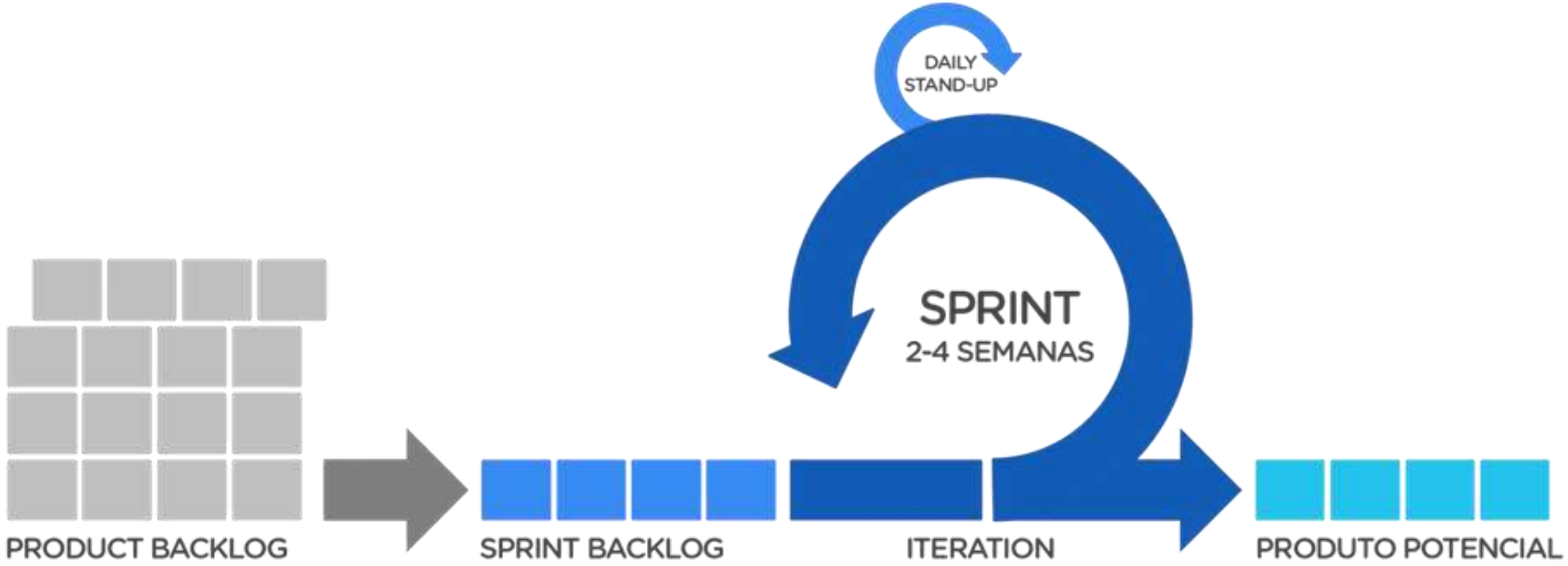
Product Backlog – Lista de todas as funcionalidades a serem implementadas.

Sprint Backlog – Tarefas a serem realizadas na Sprint.

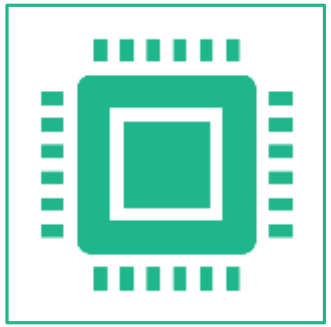
Daily Scrum – Reunião diária para alinhamento.

Sprint Review Meeting – Apresentação das implementações ao fim da Sprint.

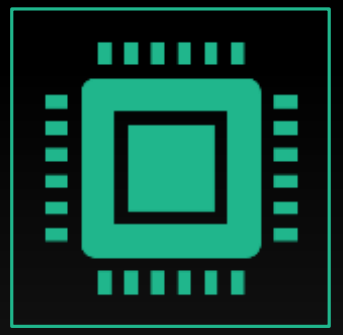
Sprint Retrospective – Reunião para rever pontos positivos e de oportunidades.


DAILY
STAND-UP
SPRINT
2-4 SEMANAS
PRODUCT BACKLOG
SPRINT BACKLOG
ITERATION
PRODUTO POTENCIAL

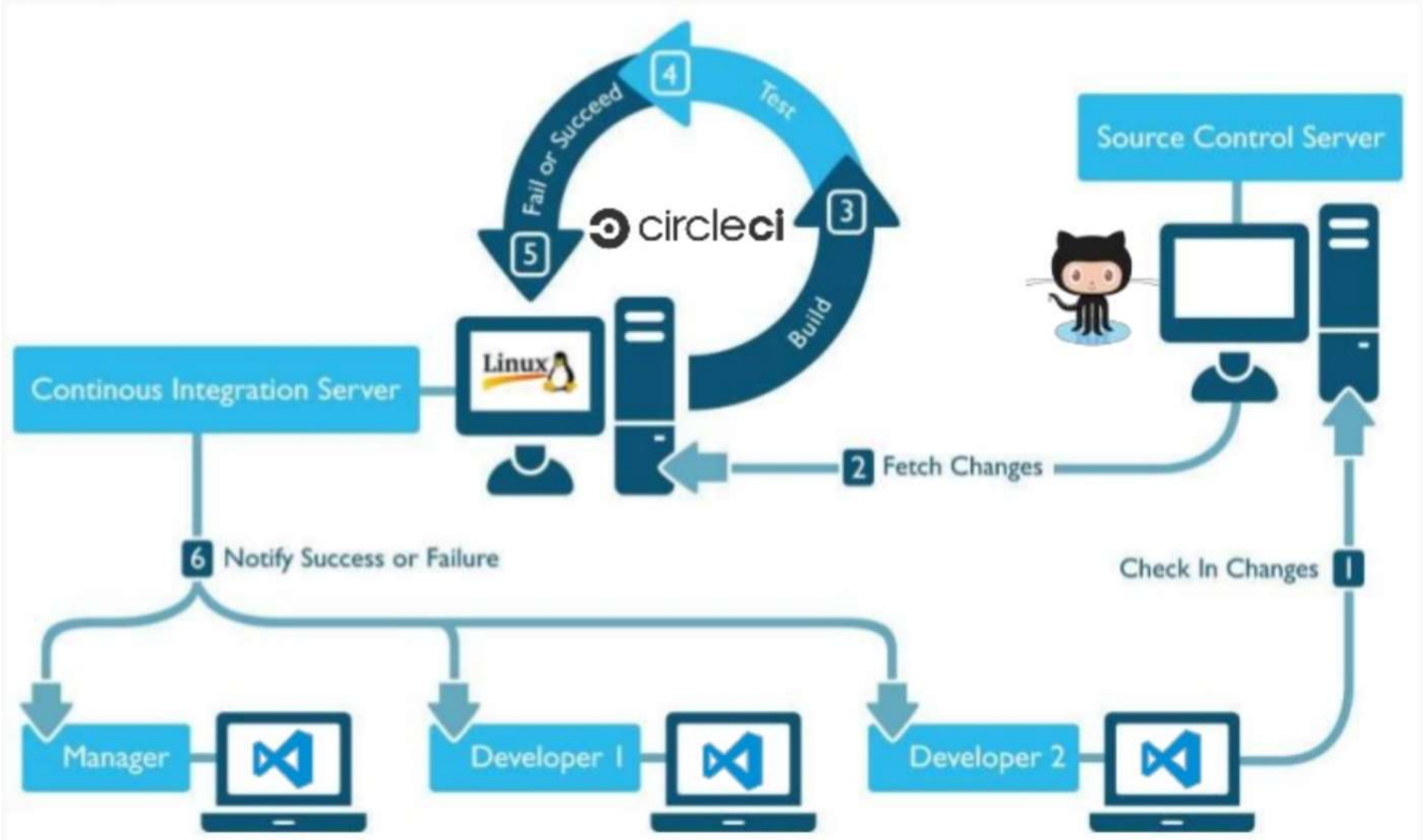
Integração contínua



É a integração do código a um repositório compartilhado



Execução do ciclo de vida do software, conhecido como BDT (Build – Deploy – Test);


Source Control Server
circleci
4
Test
3
Build
5
Fail or Succeed
Linux
Continous Integration Server
2 Fetch Changes
6 Notify Success or Failure
Check In Changes 1
Manager
Developer 1
Developer 2

Vantagens

Rapidez no desenvolvimento do software

- Check-in frequentes

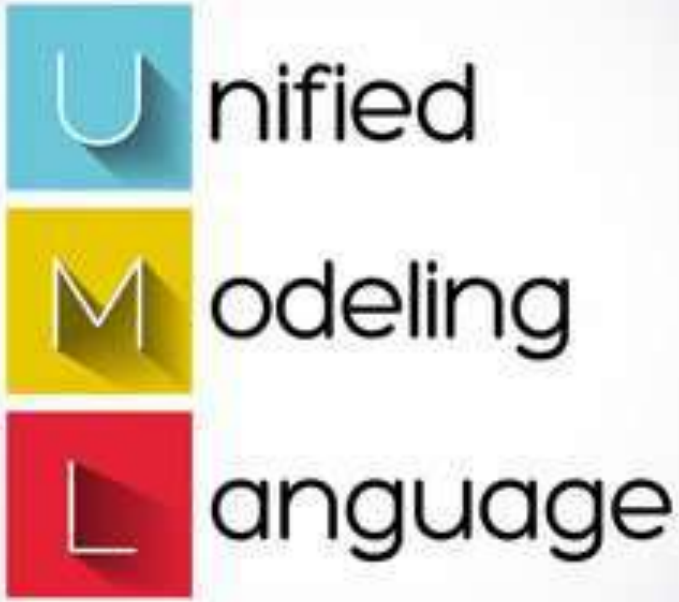
Detecção de erros no código isolado e unificado

Testes automatizados:

- Feedback mais confiáveis

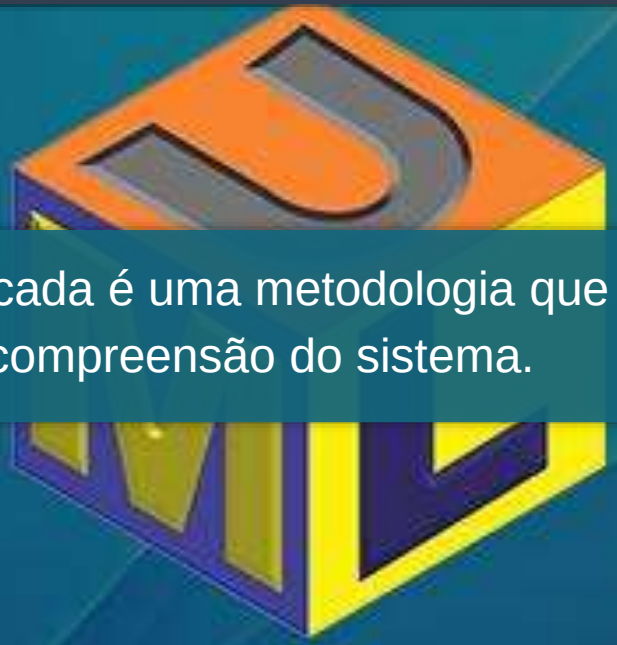
“inferno da integração”

- Diminui as dificuldades de mesclagem de código e conflitos


Unified
Modeling
Language

Definição

A linguagem de modelagem unificada é uma metodologia que permite representar o sistema de forma padronizada facilitando assim a compreensão do sistema.



UML

Como aplicar no meu projeto?

Selecione uma metodologia: uma metodologia define formalmente o processo que você usa para reunir requisitos, analisá-los e projetar um aplicativo que os atenda de todas as formas.

Selecione uma ferramenta de desenvolvimento UML: como a maioria (embora não todas) as ferramentas baseadas em UML implementam uma metodologia específica, em alguns casos pode não ser prático escolher uma ferramenta e, em seguida, tentar usá-la com uma metodologia para a qual não foi criada.

Diagramas

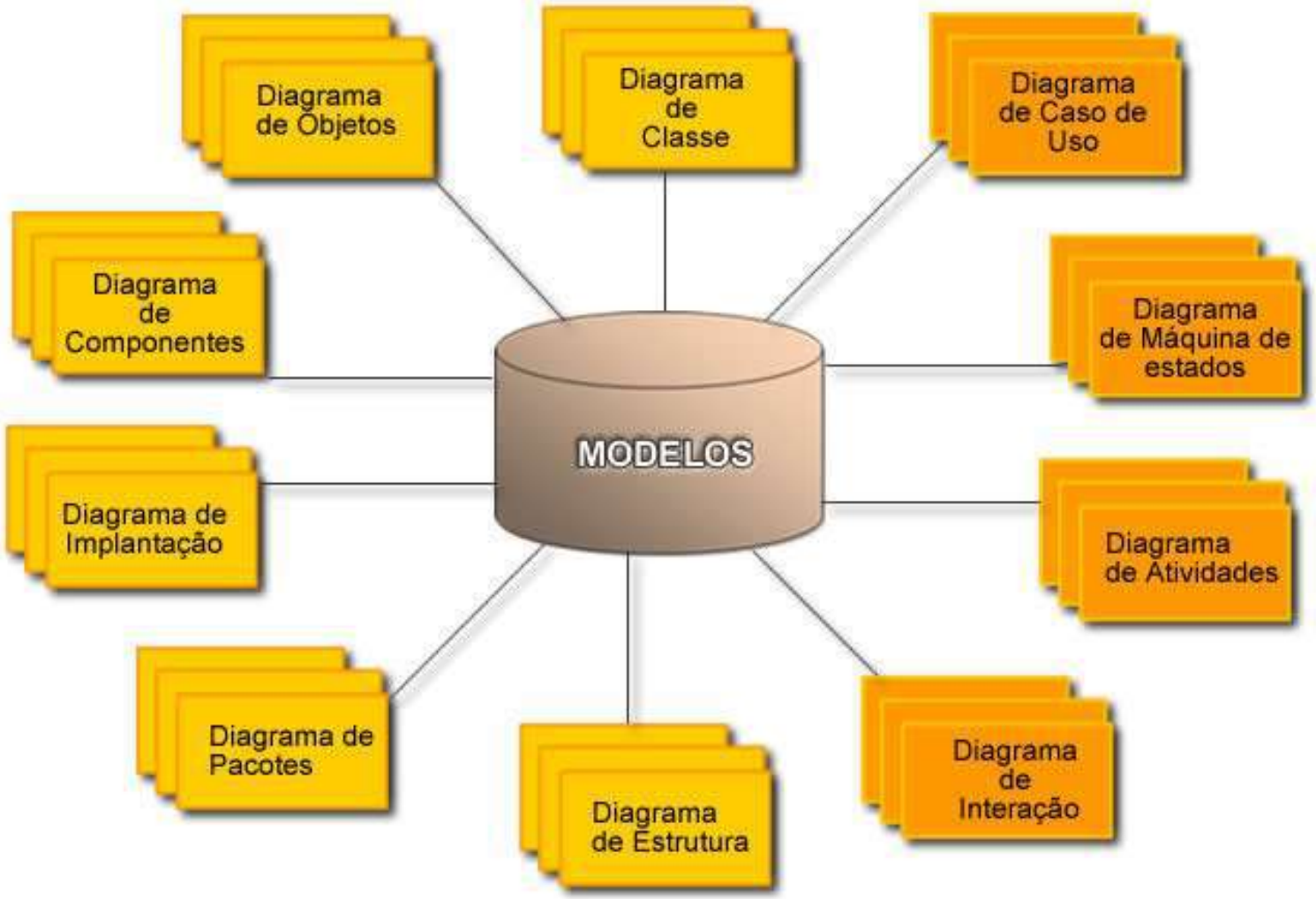


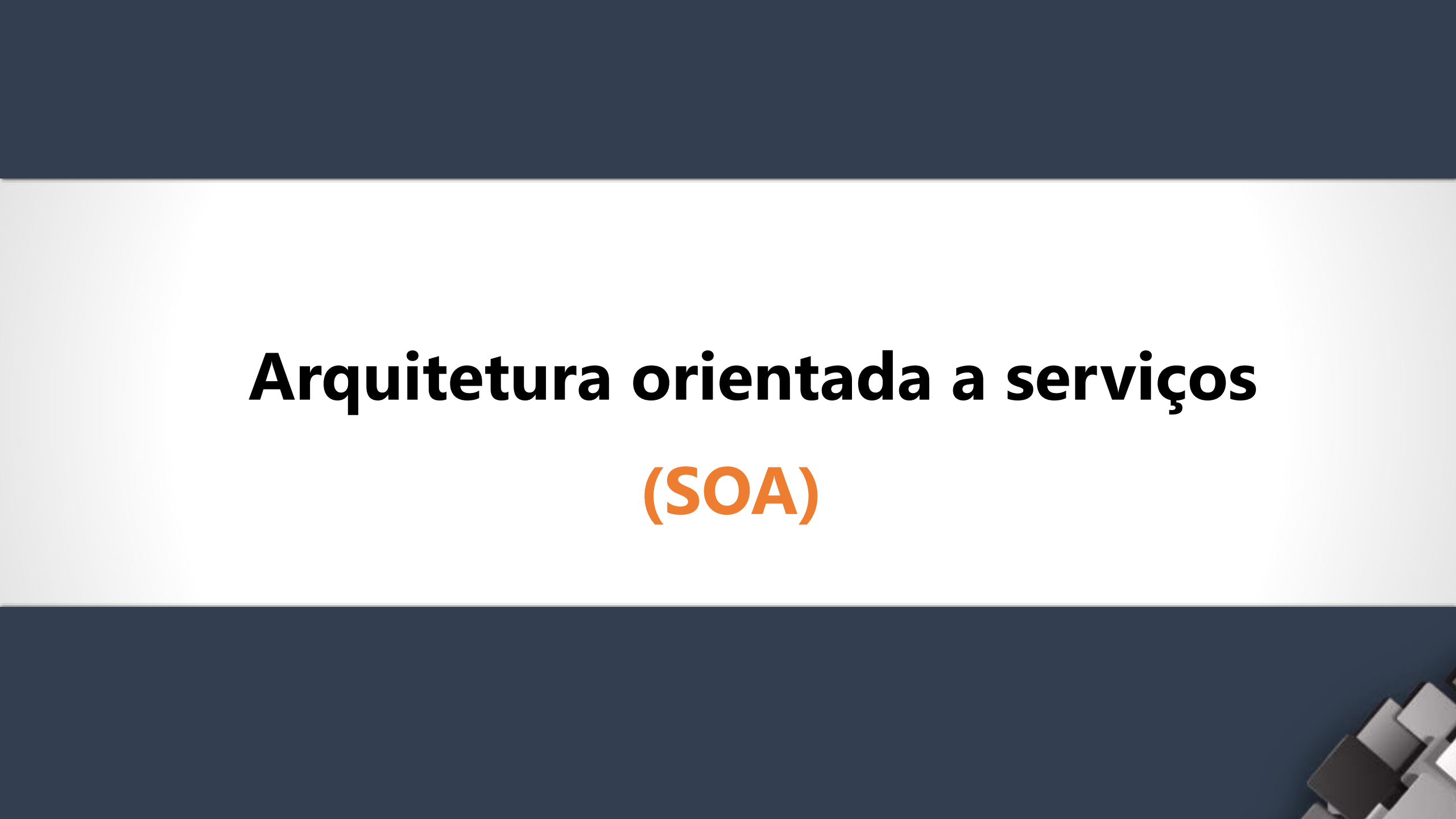
Diagrama de Objetos
Diagrama de Classe
Diagrama de Caso de Uso
Diagrama de Componentes
Diagrama de Máquina de estados
MODELOS
Diagrama de Implantação
Diagrama de Atividades
Diagrama de Pacotes
Diagrama de Estrutura
Diagrama de Interação

Onde é mais utilizada?

- Sistemas de informações corporativos
- Serviços bancários e financeiros
- Telecomunicações
- Transportes
- Defesa/Espaço Aéreo
- Vendas de Varejo
- Eletrônica médica
- Serviços distribuídos

Arquitetura orientada a serviços

(SOA)



CAPÍTULO

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 **19** 20 21 22 23 24 25 26

Arquitetura orientada a serviços

Objetivos

O objetivo deste capítulo é apresentar a arquitetura de software orientada a serviços como uma forma de criação de aplicações distribuídas usando *web Services*. Com a leitura deste capítulo, você:

- compreenderá as noções básicas de *web Service*, padrões de *web Services* e arquitetura orientada a serviços;
- entenderá o processo da engenharia de serviços que se destina a produzir *web Services* reusáveis;
- conhecerá a noção de composição de serviços como um meio de desenvolvimento de aplicações orientadas a serviços;

IAN SOMMERVILLE

9ª Edição

Pág. 355

Um web Service é uma instância de uma ideia mais geral de um serviço, que é definido (LOVELOCK et al., 1996) como:
um ato ou desempenho oferecido de uma parte para outra. Embora o processo possa ser vinculado a um produto físico, o desempenho é essencialmente intangível e normalmente não resulta na posse de qualquer um dos fatores de produção.

Tecnologia que permite a comunicação entre aplicações de uma maneira independente do sistema operacional e da linguagem de programação.

Um software prestando serviços a outro software

Figura 19.1 Arquitetura orientada a serviço

Diretório de serviços

Para Web services

Serviço publicado em linguagem de interface

Protocolo de comunicação

Modelo de arquitetura!

Figura 19.1 Arquitetura orientada a serviço

- Princípios de computação distribuída
- Paradigma request/reply
- Interface de serviços fracamente acoplados
- Facilita o reúso

Figura 19.2 Padrões de *web Services*

Conceitos-Chave:

- Visibilidade
- Interação
- Efeitos

Quadro 19.1 Parte de uma descrição WSDL para um *web service*

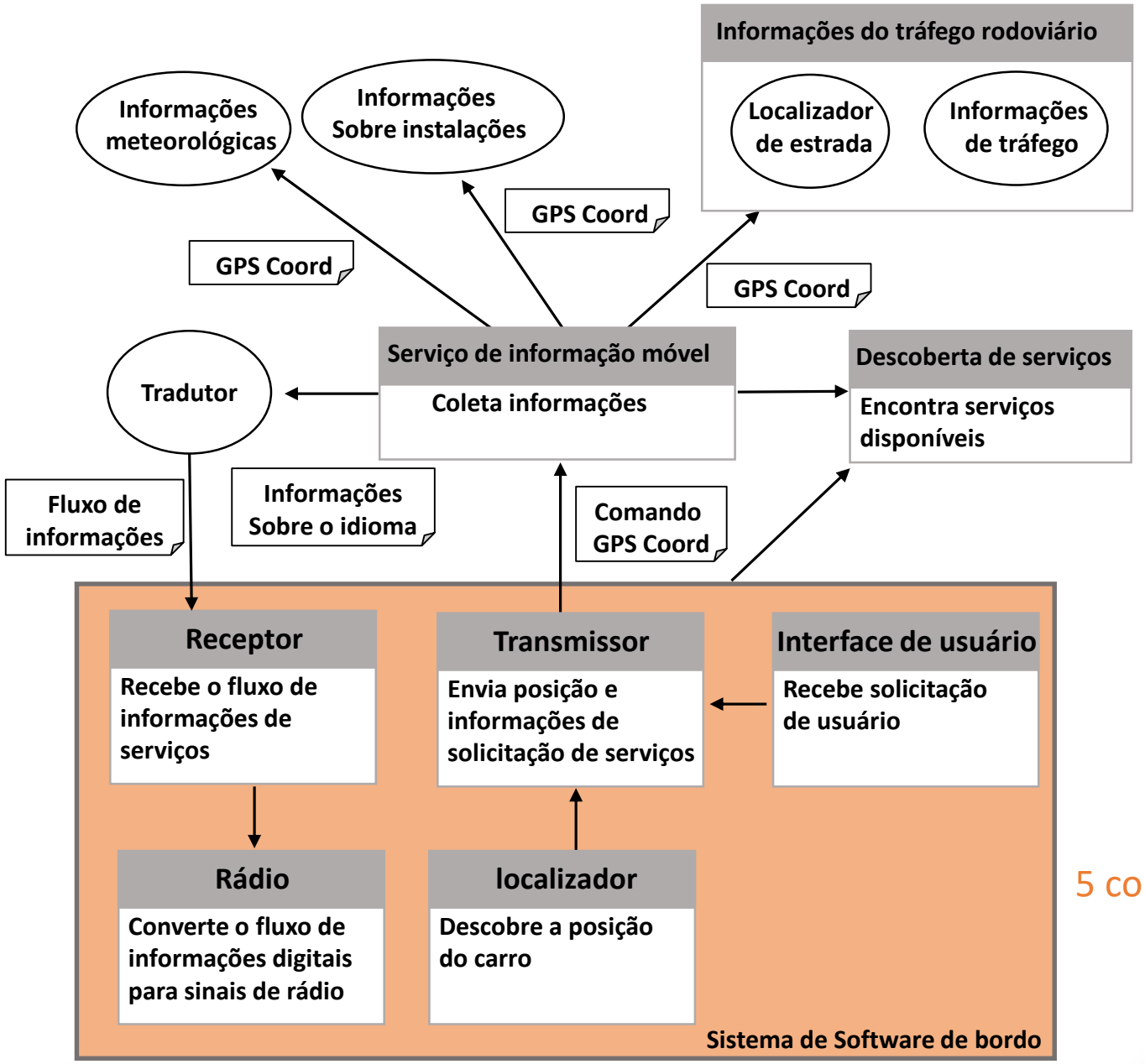
Defina alguns dos tipos usados. Suponha que o prefixo de *namespace* 'ws' se refira ao *namespace* URI para esquemas XML e o prefixo de *namespace* associado com essa definição seja *weathns*.

```
<types>
  <xs: schema targetNameSpace = "http://.../weathns"
        xmlns: weathns = "http://.../weathns" >
        <xs:element name = "PlaceAndDate" type = "pdrec" />
        <xs:element name = "MaxMinTemp" type = "mmtrec" />
        <xs:element name = "InDataFault" type = "errmess" />
        <xs:complexType name = "pdrec"
        <xs:sequence>
              <xs:element name = "cidade" type = "xs:string"/>
              <xs:element name = "país" type = "xs:string"/>
              <xs:element name = "dia" type = "xs:date"/>
          </xs:complexType>
          Definições de MaxMinType e InDataFault aqui
  </schema>
</types>
```

Agora, defina a interface e suas operações. Nesse caso, existe apenas uma operação para retornar as temperaturas máximas e mínimas.

```
<interface name = "weatherInfo" >
  <operation name = "getMaxMinTemps" pattern = "wsdlns: in-out">
  <input messageLabel = "In" element = "weathns: PlaceAndDate" />
  <output messageLabel = "Out" element = "weathns:MaxMinTemp" />
  <outfault messageLabel = "Out" element = "weathns:InDataFault" />
</operation>
</interface>
```

Um sistema de informações de bordo baseado em serviços



5 componentes

Um sistema de informações de bordo baseado em serviços

Parceiros

Informações meteorológicas

Informações Sobre instalações

Informações do tráfego rodoviário

Localizador de estrada

Informações de tráfego

GPS Coord

GPS Coord

GPS Coord

Tradutor

Serviço de informação móvel

Coleta informações

Descoberta de serviços

Encontra serviços disponíveis

Fluxo de informações

Informações Sobre o idioma

Comando GPS Coord

Receptor

Recebe o fluxo de informações de serviços

Transmissor

Envia posição e informações de solicitação de serviços

Interface de usuário

Recebe solicitação de usuário

Rádio

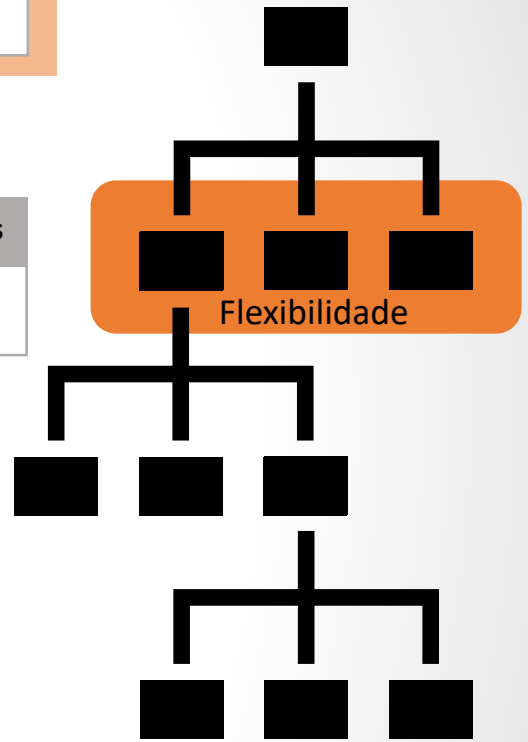
Converte o fluxo de informações digitais para sinais de rádio

localizador

Descobre a posição do carro

Sistema de Software de bordo

Composição de serviços



Engenharia de serviços