INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Engenharia de software

João Caldeira

Professor Convidado

O email. joaocarlos.caldeira@my.istec.pt

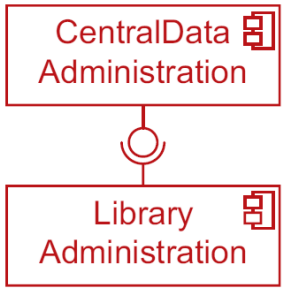
Mob. +351 917769544

5 de janeiro de 2023

INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS DE LISBOA

Software Construção

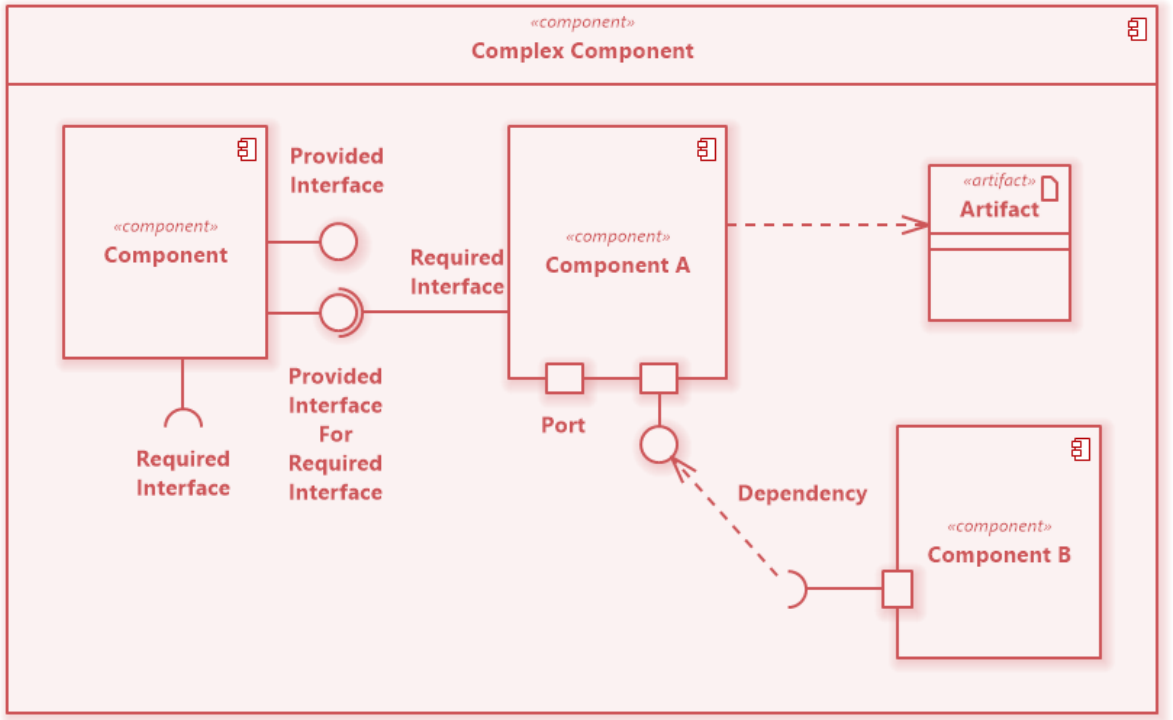
5 de janeiro de 2023



- **Componentes (Diagramas)**

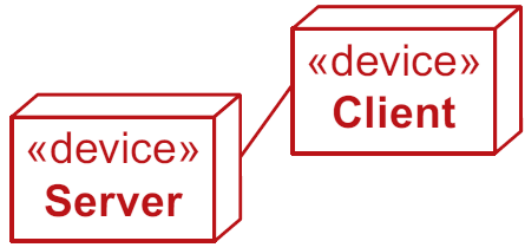
1. Um componente é uma unidade executável independente que fornece serviços a outros componentes ou usa os serviços de outros componentes

2. Ao especificar um componente, você pode modelar duas visualizações explicitamente:
   - **A visão externa.** Que representa a especificação do componente
   - **A visão interna.** Que define a implementação do componente



Construção de software

Modelos Estruturais

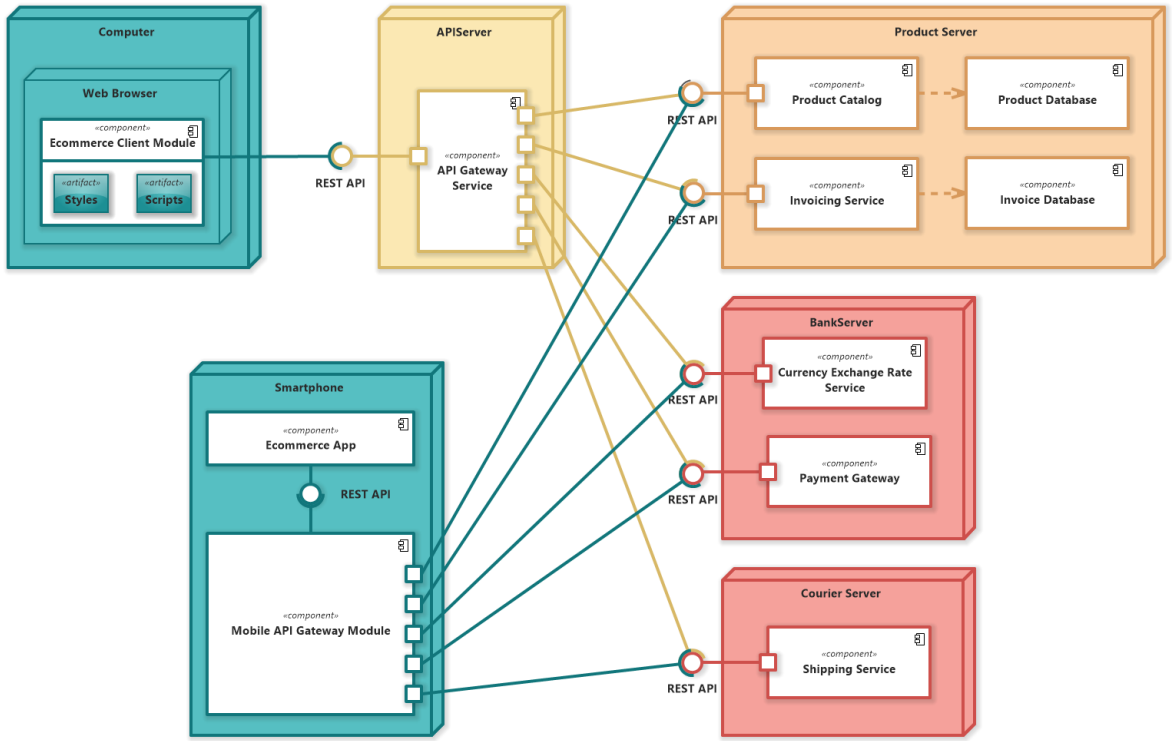


- **Implantação (diagramas)**

1. Representa a topologia de hardware usada e o sistema de tempo de execução atribuído

2. O hardware engloba unidades de processamento na forma de nós, bem como relações de comunicação entre os nós

3. Um sistema de tempo de execução contém artefatos que são implantados nos nós



Construção de software

Problemas de implementação

- **Preocupações**

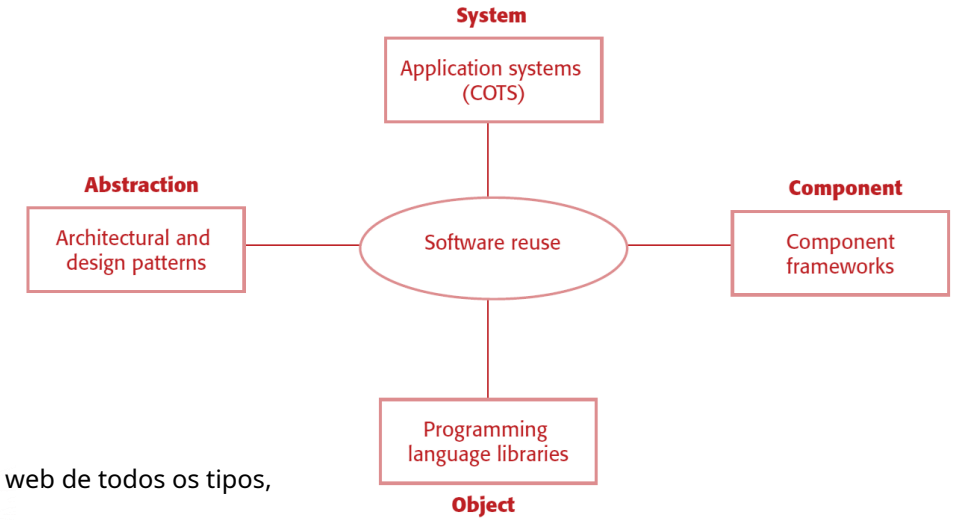
1. **Reuso.** A maioria dos softwares modernos é construída reutilizando componentes ou sistemas existentes. Quando você está desenvolvendo um software, **você deve fazer o máximo uso possível do código existente.**

2. **Gerenciamento de configurações.** Durante o processo de desenvolvimento, muitas versões diferentes de cada componente de software são criadas. Se você não acompanhar essas versões em um sistema de gerenciamento de configuração, **você está sujeito a incluir as versões erradas desses componentes em seu sistema.**

3. **Desenvolvimento do host-alvo.** O software de produção geralmente não é executado no mesmo computador que o ambiente de desenvolvimento de software. Em vez disso, você o desenvolve em um computador (o sistema host) e o executa em um computador separado (o sistema de destino). **Às vezes, os sistemas host e de destino são do mesmo tipo, mas geralmente são completamente diferentes.**

Construção de software

Reutilização de software



- **Reuso**
  - Dos anos 1960 aos anos 1990, a maioria dos novos softwares foi desenvolvida do zero
  - Uma abordagem baseada em reutilização é agora amplamente utilizada para sistemas baseados na web de todos os tipos, software científico e, cada vez mais, em engenharia de sistemas embarcados.

1. **O nível de abstração.**Nesse nível, você não reutiliza o software diretamente, mas usa o conhecimento de abstrações bem-sucedidas no design do seu software
2. **O nível do objeto.**Nesse nível, você reutiliza objetos diretamente de uma biblioteca em vez de escrever o código sozinho.
3. **O nível do componente.**Componentes são coleções de objetos e classes de objetos que operam juntos para fornecer funções e serviços relacionados. Freqüentemente, você precisa adaptar e estender o componente adicionando algum código próprio
4. **O nível do sistema.**Nesse nível, você reutiliza sistemas de aplicativos inteiros. Essa função geralmente envolve algum tipo de configuração desses sistemas

Construção de software

Gerenciamento de configurações

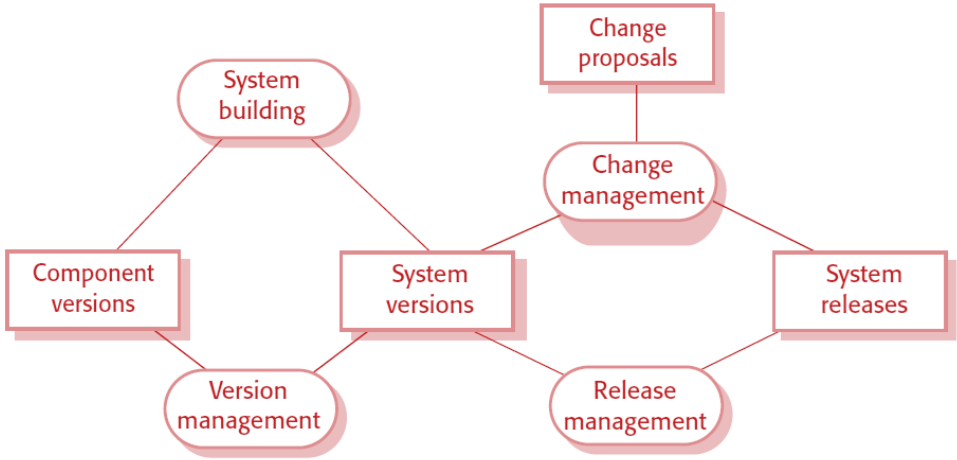
- ## Atividades

1. **Gerenciamento de versão.**Onde o suporte é fornecido para acompanhar as diferentes versões dos componentes de software

2. **Integração do sistema.**Onde o suporte é fornecido para ajudar os desenvolvedores a definir quais versões de componentes são usadas para criar cada versão de um sistema.

3. **Rastreamento de problemas.**Onde o suporte é fornecido para permitir que os usuários relatem bugs e outros problemas e para permitir que todos desenvolvedores para ver quem está trabalhando nesses problemas e quando eles são corrigidos.

4. **Gestão de lançamentos.**Onde novas versões de um sistema de software são liberadas para os clientes. Preocupa-se com o planejamento da funcionalidade de novos lançamentos e com a organização do software para distribuição.

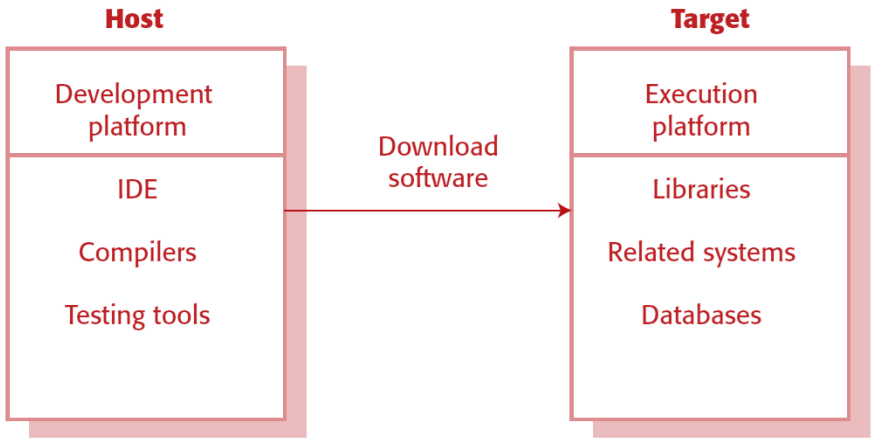


Construção de software

Desenvolvimento de Host-Destino

- **Plataforma de Desenvolvimento de Software**

1. Um compilador integrado e sistema de edição direcionado à sintaxe que permite criar, editar e compilar código
2. Um sistema de depuração de linguagem
3. Ferramentas de edição gráfica, como ferramentas para editar modelos UML
4. Ferramentas de teste, como JUnit, que podem executar automaticamente um conjunto de testes em uma nova versão de um programa
5. Ferramentas para dar suporte à refatoração e visualização do programa
6. Ferramentas de gerenciamento de configuração para gerenciar versões de código-fonte e para integrar e construir sistemas



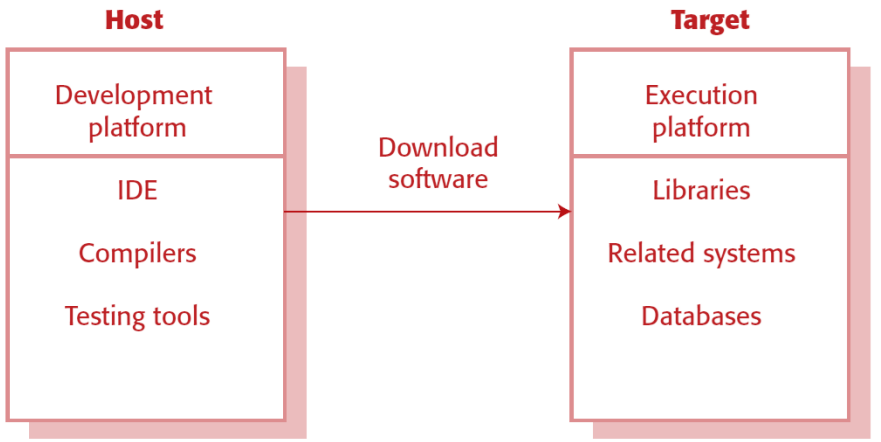
Desenvolvimento de Host-Destino

- # Plataforma Alvo

1. **Os requisitos de hardware e software de um componente**
   - Se um componente for projetado para uma arquitetura de hardware específica ou depender de algum outro sistema de software, ele deve obviamente ser implantado em uma plataforma que forneça o suporte de hardware e software necessário.

2. **Os requisitos de disponibilidade do sistema**
   - Os sistemas de alta disponibilidade podem exigir que os componentes sejam implantados em mais de uma plataforma.

3. **Comunicações de componentes**
   - Se houver muita comunicação entre componentes, geralmente é melhor implantá-los na mesma plataforma ou em plataformas fisicamente próximas umas das outras.



- **Definição**

1. O desenvolvimento de código aberto é uma abordagem para o desenvolvimento de software em que o código-fonte de um software**sistema é publicado e voluntários são convidados a participar**no processo de desenvolvimento

2. Suas raízes estão no**Fundação de Software Livre (www.fsf.org)**, que defende que o código-fonte**não deve ser proprietário**mas deveria sempre**estar disponível para os usuários examinarem e modificarem**como eles desejam

3. Havia uma suposição de que o código seria controlado e desenvolvido por um pequeno grupo principal, em vez de usuários do código

Construção de software

Desenvolvimento de código aberto

- **Licenciamento**

1. Um princípio fundamental do desenvolvimento de código aberto é que o código-fonte deve estar disponível gratuitamente

2. Isso não significa que qualquer um pode fazer o que quiser com esse código

3. Legalmente, o desenvolvedor do código (seja uma empresa ou um indivíduo) é o proprietário do código

4. Eles podem impor restrições sobre como ele é usado, incluindo condições juridicamente vinculativas em uma licença de software de código aberto

- **Variantes de Licenciamento (3 modelos gerais)**

1. **Licença Pública Geral GNU (GPL).** Esta é a chamada licença recíproca que significa de forma simplista que, se você usar software de código aberto licenciado sob a licença GPL, deverá tornar esse software de código aberto.

2. **Licença Pública Geral Menor GNU (LGPL).** Esta é uma variante da licença GPL em que você pode escrever componentes vinculados ao código-fonte aberto sem ter que publicar o código-fonte desses componentes. No entanto, se você alterar o componente licenciado, deverá publicá-lo como software livre.

3. **Licença de Distribuição Padrão Berkley (BSD).** Esta é uma licença não recíproca, o que significa que você não é obrigado a republicar quaisquer alterações ou modificações feitas no código-fonte aberto. Você pode incluir o código em sistemas proprietários que são vendidos. Se você usar componentes de código aberto, deverá reconhecer o criador original do código. **A licença MIT é uma variante da licença BSD com condições semelhantes.**

Construção de software

Desenvolvimento de código aberto

- **Problemas de licenciamento**

1. Se você usa software de código aberto como parte de um produto de software, **você pode ser obrigado** pelos termos da licença **para fazer seu próprio produto de código aberto**

2. Se você está tentando **vender seu software, você pode querer mantê-lo em segredo**. Isso significa que você **pode querer evitar o uso de código aberto licenciado pela GPL** software em seu desenvolvimento

3. Se você está construindo um software que **é executado em uma plataforma de código aberto** mas que não reutiliza componentes de código aberto, **então as licenças não são um problema**

4. No entanto, se você **incorpore software de código aberto em seu software**, você precisa de processos e bancos de dados para acompanhar **o que foi usado e suas condições de licença**