

20105

Arquitetura de Software

Uma arquitetura de software

pode ser definida como uma estrutura que representa a organização de um software.

Representação de
relacionamentos existentes

* Arquitetura é construída e/ou modificada tanto no desenvolvimento quanto na manutenção

Princípios Arquiteturais

representam boas práticas que podem ser considerados no desenvolvimento e manutenção de uma arquitetura.

Boas práticas ou catálogo de práticas

- Uso de Padrões ; } revisão.
- Uso de soluções conhecidas ; }
- Revisões } qualidade
- Uso de métricas } formas que medem a arquitec-
tura para auditar.

Anomalias Arquiteturais

apresentam possíveis problemas observados durante a construção ou manutenção de uma arquitetura.

Anomalias:

- Classes longas, - Métodos longos
- Código duplicado; - Declarações switch.

Classes longas: quando uma apresenta um excesso de responsabilidades (atributos/métodos que não devem ser considerados pela classe).

Solução: extração de classes

Código Duplicado: quando um mesmo trecho de código em mais de duas regiões do software.

(herança)

Solução: reaproveitamento de código.

Métodos Longos: Observa quando o método possui mais de uma responsabilidade (retorno de resultados não relacionados).

Solução: extrair de métodos.

Declaroções Switch: Estruturas de seleção (switches cases, muitos ifs) que possuem a finalidade de realizar algum tipo de validação.

Solução: Aplicar o princípio de polimorfismo.

→ Todas as soluções podem ser ditadas pelo refatoração (melhorar a estrutura do software) e melhorar a arquitetura.

22/05

Projeto Arquitetural (de arquitetura)

Conjunto de ações, que são realizadas com objetivo de identificar os principais elementos de uma arquitetura.

O projeto pode ser utilizado em diferentes situações para:

- desenvolvimento;
- manutenção
- atividades de análise
- comunicação com stakeholders
- ...

Obs: usualmente, registra-se os resultados das ações em um documento que é consultado quando necessário

Abastecimento de Arquitetura

Considera inicio e representação e o desenvolvimento da arquitetura.

Representação da Arquitetura.

Uma arquitetura de software pode ser representada de diferentes maneiras.

- Diagrama de Bloco (antigo)
- Diagrama UML. (utilizado na disciplina)

Diagramas de Bloco:

Busca representar os elementos de uma arquitetura como bloco.

- Diferentes relacionamentos entre os blocos pode ser considerados.

Exemplo no slide.

Desenvolvimentos de Arquitetura.

Possuem diferentes objetivos, que ajudam com versões iniciais e podem ser reutilizados, considerando os reportes da literatura e podem ser entendidos como padrões.

Padrões da Arquitetura

Representam a estruturação do solução ou organização do arquitetura.

- Fazem desenvolvimentos e aplicados em diferentes projetos.
- Cada padrão apresenta uma área prática.

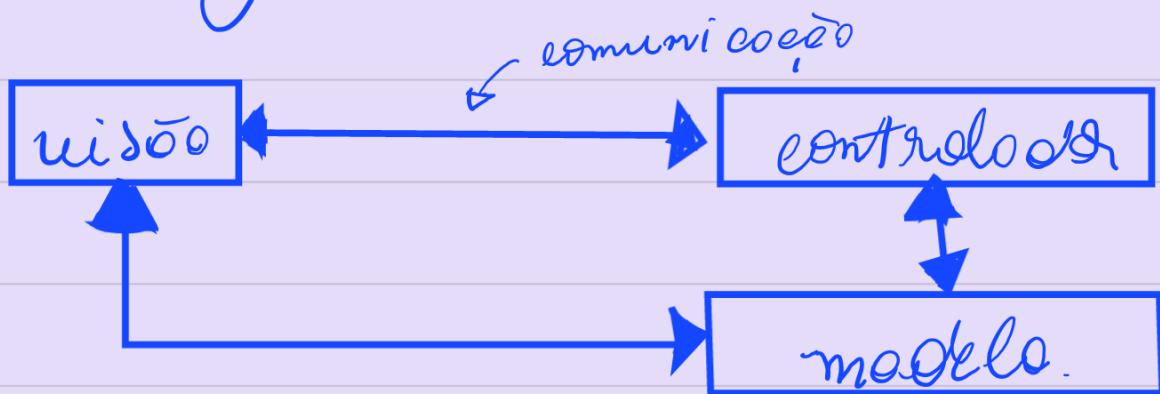
Padrões Gerais (ou clássicos)

Modelo - Visão - Controlador;
Camadas

Modelo Visão Controlador

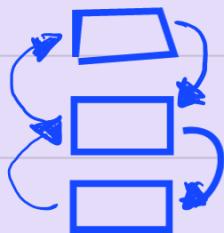
Agrupa as responsabilidades dos elementos em:

- representação (visão)
- processamento (controlador)
- armazenamento (modelo)



Camadas

Busca agrupar as funcionalidades pelo eixo das estruturas específicas conhecidas como camadas.



Cliente-Servidor.

Introdução em sistemas web

Cliente → solicita dados (realiza requisições).

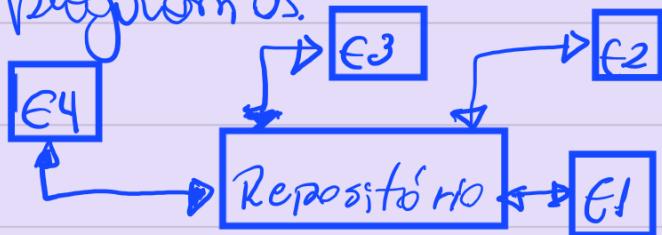
Servidor → realiza processamento e retorna resultados.

→ O fluxo de comunicação entre cliente e servidor é bem definido. O cliente inicia a interação e o servidor retorna o que é solicitado.



Repositórios

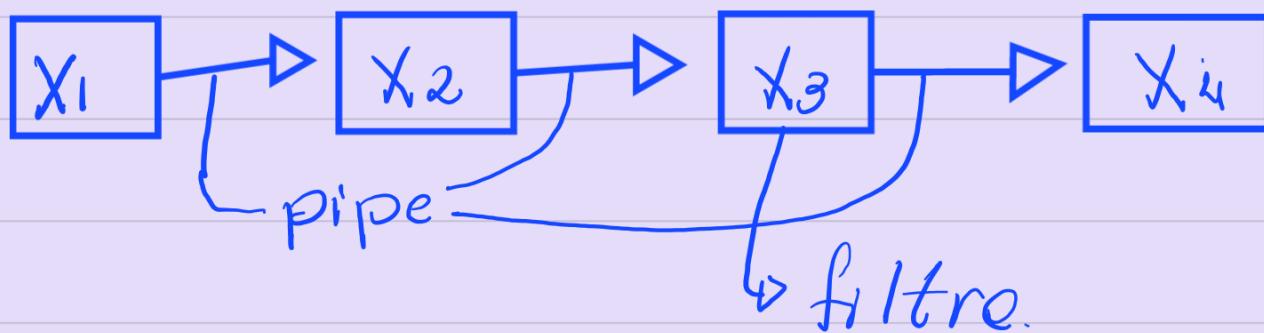
Utilizado para armazenamento de dados e execução de programas.



- Se o repositório guarda dados, se obtém-lha a um banco de dados;
- Se o repositório tem dados e processos é semelhante a um servidor.

Pipes e Filtros.

Utilizado quando a necessidade de um processamento sequencial (processamento em lote)



↳ Cai na prova, um caso x, e qual o
processo mais se adequa e por quê?

Arquitetura de Aplicações.

Podem ser definidas como arquiteturas
usualmente utilizadas em um deter-
minado tipo de software.
Tipos de software (aplicações).

Processamento de dados.

- ↳ Processamento sequencial.
sem interrupção do usuário.

Processamento de Eventos.

- ↳ monitoramento de respostas
cada evento passa sua resposta.

Processamento de Troncos

- Tronco: conjunto de operações
- Acesso ao banco de dados.

Processamento de Linguagem

- Reclama uma linguagem de entrada
- Realiza uma tradução
- Retorna os resultados em outra linguagem.

Processamento de Troncos

Posuem diferentes aplicações relacionadas

Ex: Arquitetura de sistemas de informação

- Observar o software comercial ou empresarial.

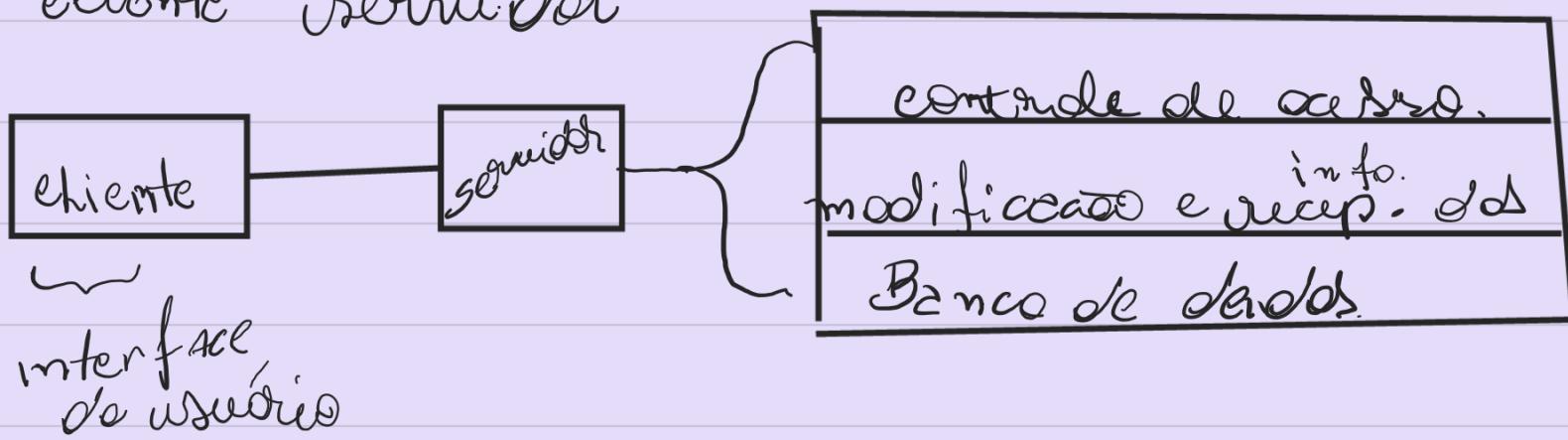
- Possível organização em comodato.

Pg 20 slide.

Obs: A popularização da internet fez os sistemas de informaçãoarem utilizados de modo unificado.

► Sistemas de informação baseados na web.

- Organizados de acordo com o padrão cliente - servidor



Processamento de Linguagem

- Observados em softwares básicos, tais como compiladores e interpretadores.
- Projetados a partir dos padrões:
 - Repositório.
 - Pipes e filtros.

Projeto e Implementação 03/06

Contextualização:

Considerando que numa primeira visão da arquitetura do software este disponibiliza planejor e implementação da arquitetura (e software).

... perdi coisa no caminho.

Reuso de Software

Definido como reaproveitamento de artefatos.

resultados gerados ou utilizados durante o ciclo de vida do software.

- As práticas de reuso estão sendo adotadas há dezenas.
- Com passar do tempo as práticas de reuso foram agrupadas
- Desse agrupamento surgiram 3 níveis de reuso.

Níveis de Reuso.

- ↳ Abstração: modelos ou soluções gerais, que podem ser adoptados ao projeto de desenvolvimento produtivas.
- ↳ Objeto/função: reaproveitamento de códigos desenvolvidos anteriormente.
scanner e scmf
- ↳ Componente: reaproveitamento de subsistemas (partes de um software).
Elemento autônomo e reutilizável.
frameworks
- ↳ Sistemas: reaproveitamento de aplicações completas (com ou sem configuração) *pacote office*.

Custos de Reuso.

- Existem diferentes custos associados com o reuso.
- ↳ Tempo: associado com a pesquisa de práticas que podem ser adoptadas no projeto e na implementação.

- ↳ Integração: "ligação" de elementos do software em construção com os elementos reusados.
- ↳ Aquisição: compra / assinatura de determinado elemento.
- ↳ Adaptação / configuração: customização de um elemento que será utilizado no software.

Desenvolvimento Open Source

- Ainda no contexto de reuso, pode-se considerar práticos mais próximos da implementação:
- É o caso do desenvolvimento open source.
- Neste reuso pode ser observado em diferentes níveis.

Características

- Criação de projetos;
- Formação de equipes de desenvolvimento;
- reaproveitamento de códigos;
- reaproveitamento de resultados;

- * No geral, o modo de negócio associado com o desenvolvimento open source é talvez baseado em suporte.
 - ↳ Código e executável gratuitos;
 - ↳ manutenções e/ou consultorias mediante pagamentos.;

05/06

Benefícios com reuso de software.

Existem diferentes benefícios com reuso de software.

- ↳ Desenvolvimento acelerado (já presente certos fatos).
- ↳ Tempo reduzido de projeto (conjunto de variações realizadas com tempo definido).

Desafios com Reuso de software

Aplicação de reuso não é trivial

Desafios ou problemas para o reuso.

- ↳ Dependências de terceiros: dificuldades de customizar o software para nossas necessidades.
- ↳ Dificuldade com manutenção e suporte: problemas observados durante o uso de softwares.

Soluções com reúso de software.

Antes de começar o estudo das soluções, vamos checar a romoroma de reúso
Exemplo

- Padrões Arquiteturais;
- Padrões de Projetos; Anexo 5
- Frameworks; Anexo 6.

Padrão de Projeto

Pode ser entendido como uma solução genérica que resolve problemas comuns, observados durante o projeto de software.

Características:

- Modelo de solução (reúso a nível de abstração)
- Independência de tecnologia.

Estrutura de um padrão

Cada padrão é estruturado, ou descrito de um modo genérico.

Informações consideradas:

- nome;
- problema considerado;
- solução desenvolvida;

Classificação de Padrões.

Realizada com base nos seguintes critérios

↳ Escopo:

- ↳ classe;
- ↳ Objeto;

↳ Propósito:

- ↳ criação;
- ↳ estrutura;
- ↳ comportamento;

→ a utilização das classificações não é só, de vez em quando entre si.

Outros exemplos de padrões.

- + Adoptador: comunicação entre interfaces incompatíveis
- + Builder: construção de objetos complexos construídos em etapas
- + Mediator: simplificar a comunicação entre os elementos do software.
- + Observer: atualizar múltiplos representantes de um conjunto de dados
 - Gráficos: tabuleiro, pizza;
- + Template Method: customizar possos específicos de um determinado processo ou algoritmo

Exemplo Templete Method

Problema: Ordenação de conjunto de dados

Processo:

① Receber os dados do conjunto.

- memória principal

- memória secundária (arquivo ou banco dados)

② Realizar a ordenação (processamento)

- selection, bubble...

- quick sort, merge, radix

③ Retornar o conjunto ordenado

- memória principal

- memória secundária

Observações

Orientação a Objetos.
↓

Características } Coesão (independência)
importantes } Acoplamento (dependência)

Framework

Um framework pode ser definido como uma aplicação - esqueleto que possui uma finalidade específica.

Aplicação incompleta ↪ falta
e customizável principal

Ex: construção de interfaces

- Persistência de dados

Classificações

- Frameworks de infraestrutura: auxilia na comunicação entre interfaces
- Frameworks de integração: possibilita a formação de uma estrutura a partir de outras estruturas menores

- Framework de Aplicações corporativas:
Aplicações financeiras.

Obs: É comum que determinados domínios considerem frameworks durante o desenvolvimento. Nesse contexto, espera-se que tais frameworks ofereçam a implementação de elementos essenciais ao domínio.

Exemplos em sistemas web

Framework Web

Offerce suporte a determinados características (implementadas por diferentes elementos).:

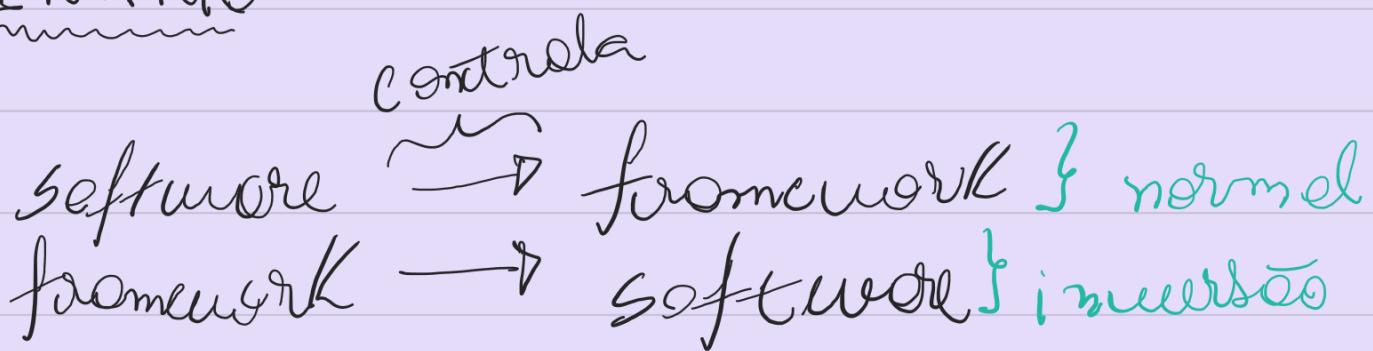
- **Proteção:** segurança e controle de acesso.
- **Páginas dinâmicas:** gerenciamento de conteúdo mediante a interrogação
- **Suporte ao Banco de dados:** persistência
- **Gerenciamento do tempo:** tempo de interrogação
- **Interrogação:** modo de interrogação

Inversão de controle

Acontece quando o software é universalizado pelo framework.

Nesse cenário, as classes não relacionados com framework são chamados quando necessário.

CENÁRIO



Possibilidades de comunicação entre framework e softwares.

- Herança
- Classes Abstratas.
- Interfaces.

Linha de Produto de Software LPS

É uma abordagem de reuso sistemática

- TAMBÉM CONHECIDA COM "FAMÍLIA DE APLICAÇÕES"

Características:

- ↳ Foco em uma área de atuação
 - domínios específicos

- ↳ Reuso a nível de componente e sistema
 - aplicações geradas de reusos de resultados.

- ↳ Definição e desenvolvimento de artefatos:
 - São necessários à geração de produtos.

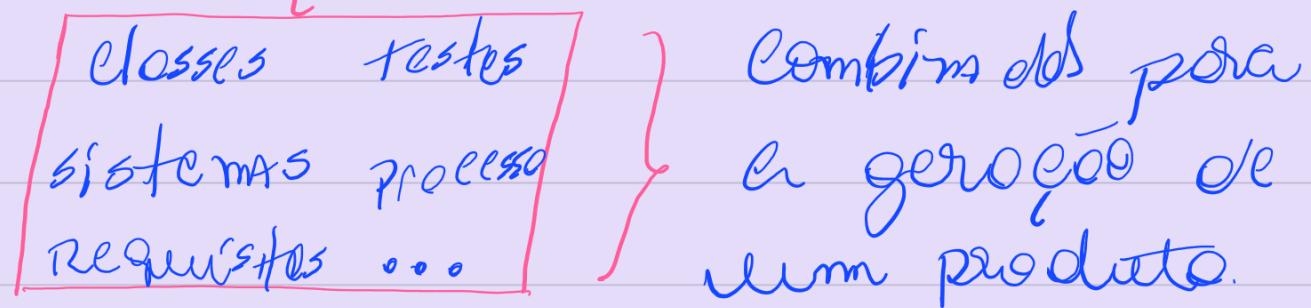
Linha de Produto de Software
abordagem sistemática e planejada
de reúso.

- Foco em uma área de atuação
- Consideração de reúso a nível de componente e sistema.

Tudo o que é desenvolvido no ciclo de vida do software (artefatos)

Obs: os diferentes artefatos associados com uma LPS devem ser monitorados em um repositório (núcleo do artefato).

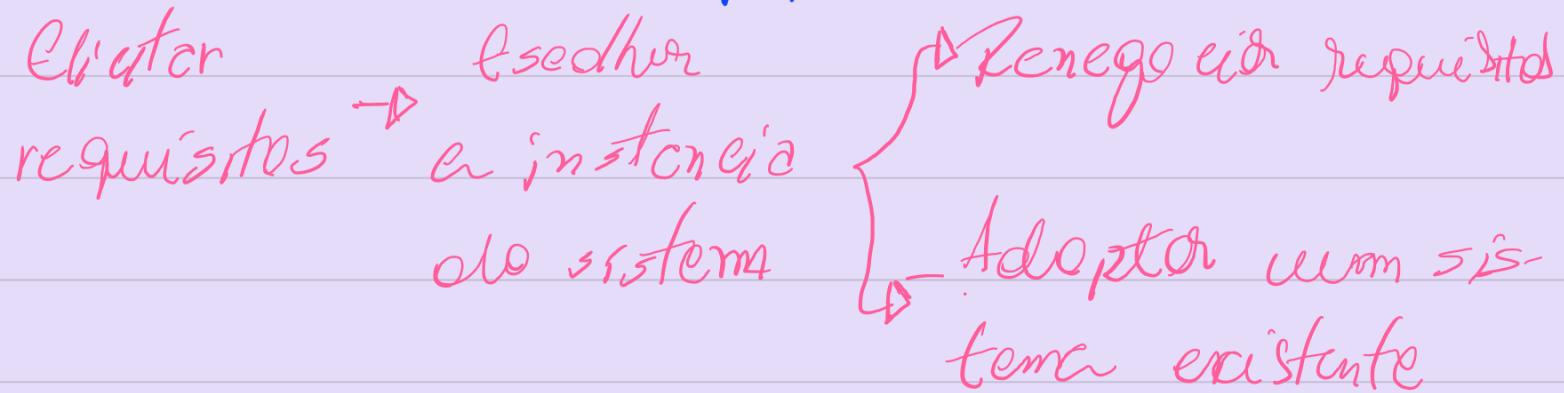
Ilustração:



Instanciación de Produto

Consiste em um processo a ser executado com objectivo de gerir um produto (software).

- Modelo de processo, por Sommerville.



Outra alternativa:

Elaborar requisitos → Identificar artefatos → Combinar associações com requisitos Artefatos → Gerir produto.

Obs: Configurações complementares acontecem durante a gerência de um produto.

Existem situações conhecidas em que as configurações complementares são necessárias:

→ **Tempo de Projeto:** desenvolvimento de produto

→ **Tempo de implantação:** após o desenvolvimento da 1^a versão do sistema.

15/08 - 1º prova → 0º dia 26/08 ou 11/07

21/08 - 2º prova

Produtos COTS

comercial off the shelf.

São softwares **genéricos** para um **determinado domínio**.

↳ Reúso a nível de sistema.

Exemplos:

- Microsoft Office
- Corel Draw, Photoshop.

Benefícios

- Implementação mais rápida
- Avaliação mais rápida (^{execução dos} funções desejadas)
- Redução de riscos de desenvolvimento

Problemas

- Falta de controle sobre a evolução
- Possível falta de suporte
- Problemas com funcionalidades

Modalidade de Reuso

No contexto de produtos COTS, duas modalidades de reuso são consideradas:

- + Sistemas de solução
- + Sistemas integrados COTS.

↳ Sistemas de Solução

Consiste no reuso de um produto COTS.

- Oferecimento de diferentes funcionalidades (geradas baseando-se em conteúdos).
- Customização mediante configurações

↳ Sistemas Integrados COTS.

É considerado quando um único produto COTS não atende todas as necessidades de um cliente.

É comum o desenvolvimento de sobrasistemas de integração.

↳ estabelecer comunicação entre os produtos COTS.

↳ Até aqui é os conteúdos da 1º Prac.

Engenharia de Software Baseada em componentes

É uma abordagem de desenvolvimento que prioriza a construção e reutilização de componentes.

- ↳ Componente Based Software Engineering. (CBSE).

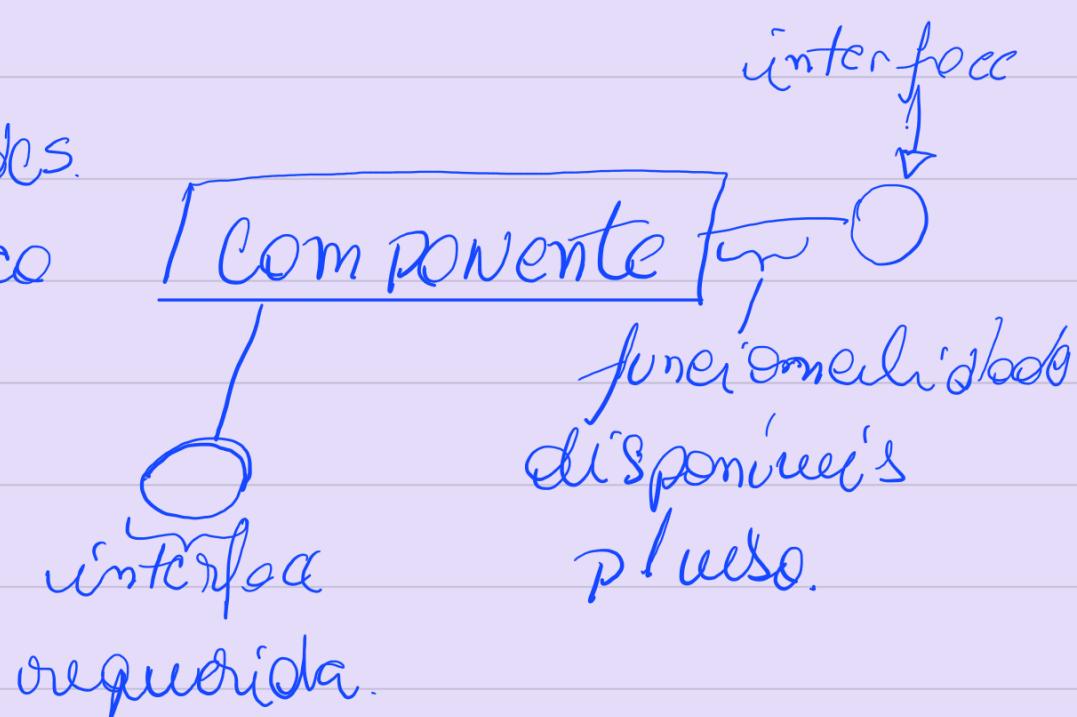
Componente

Elemento autônomo e executável
em nível espacial
de independência de execução

- ↳ Usualmente, um componente representa agregados de outros elementos, tais como classes.

- ↳ Pode ser combinado com outros componentes, de modo a gerar um outro componente resolutivo.
- ↳ Pode ser chamado de entidade prestadora de serviços.

Componente (meio codificado) } equívoco
API (mercado) } lento.
funcionalidades.
necessárias ao
componente.



Padrões de Componentes

(modelos)...

Representam modelos de organização e funcionamento para os componentes.

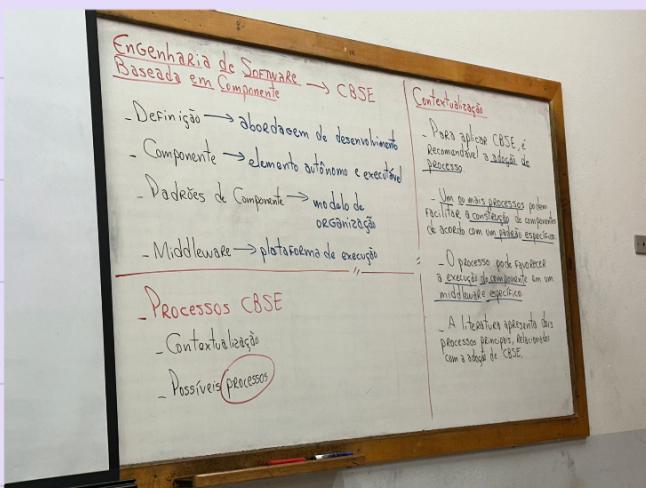
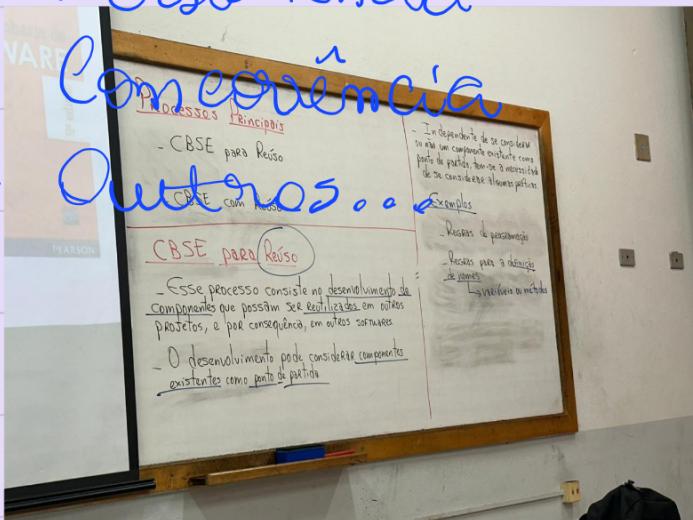
- ↳ Informações esperadas de um componente só de finidas por padrões.
- ↳ Destaque para padrões de componentes:
 - CORBA
 - Enterprise Java Beans (EJB)
 - .NET

Middleware

- Plataforma de execução de componentes.
- Visualmente, cada middleware está associada com um ou mais padrões de componentes.

- É possível considerar a comunicação entre diferentes componentes a partir de um middleware.
 - Além de oferecer suporte às funcionalidades dos componentes, um middleware também oferece funcionalidades serviços
- Exemplos.**

- Proteção
- Persistência

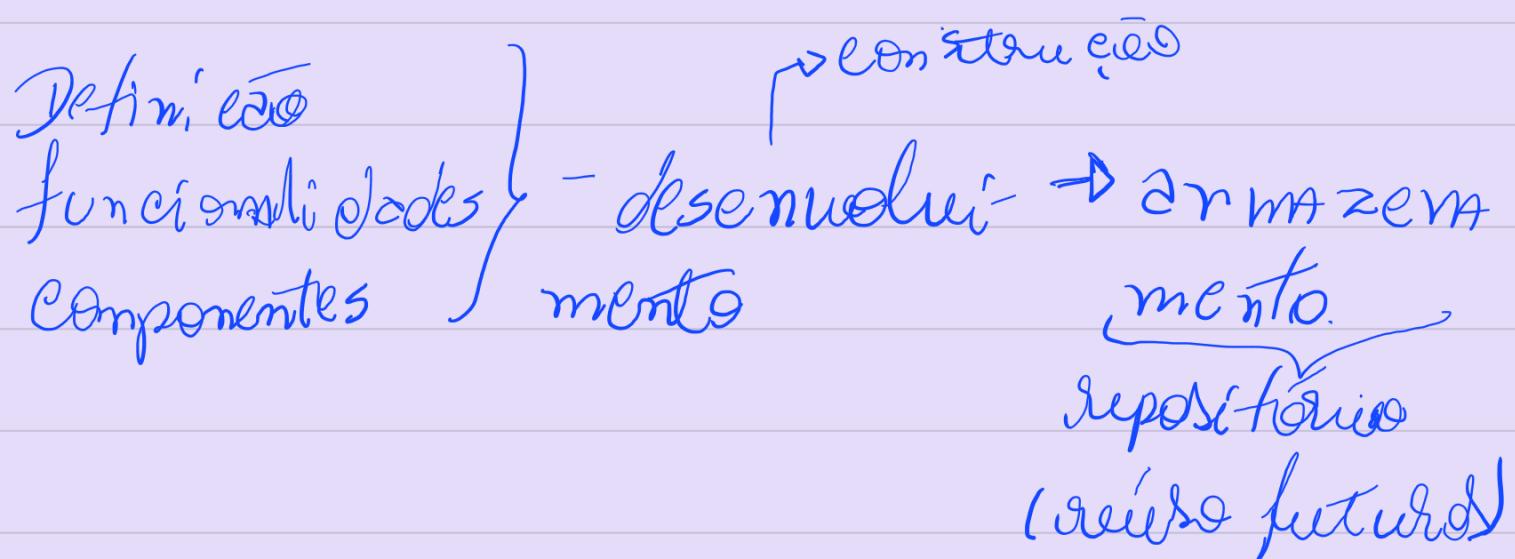


Possui mais conteúdos anteriormente. 03/07

Inde pendente de se considerar ou não um componente existente como ponto de partida, tem-se a necessidade de se considerar alguns práticos.

Ex: - regras de programação
- regras de nomeação de variáveis.

Uma possível estrutura:



Obs: Diferentes métodos e ferramentas podem ser considerados durante o processo de CBSE para reuso.

• É interessante audiar a necessidade do componente antes de iniciar o processo.

CBSE com Reuso

Construção de software a partir da combinação de composição de componentes.

- ↳ A combinação considera a comunicação entre componentes.
- ↳ A combinação considera o agrupamento entre dois ou mais componentes, com o objetivo de construir um componente reutilizável.

O processo de CBSE com reuso pode considerar as seguintes etapas:

Bobar
requisitos.
↓

Projeta
arquitetura
↑
Identifica
componentes.
↓

Identifica
componentes
condicionais.
→ modifica
requisitos

Compor compõem
os para criar
o software.

Obs: A combinação de componentes pode ser realizada de diferentes maneiras.

↳ Sequencial:

Inclusão em sequência

Dessf: leitor com diferentes interfaces

Solução: Adaptador de interfaces requeridas.

↳ Hierarquia: hierarquia entre componentes - fluxo de comunicações definido.

↳ Aditiva: combinação de diferentes interfaces e diferentes componentes.

Engenharia de Software Distribuído.

Sistemas Distribuídos: considera elementos de software e hardware localizados em diferentes lugares.

- ↳ para o usuário não faz diferença
Ex: whatsup, telegram.
- ↳ É preciso conciliar questões de hardware

Características:

- Compartilhamento de recursos } rentes elementos
- abertura (adição de novos elementos)
- concorrência (múltiplos processamentos)
- escalarabilidade (atendimento ao demanda)
- tolerância a defeitos (manter sistema funcionando mesmo com problema).

uso de um recurso por dife-

Projeto de Sistemas Distribuídos.
realiza atividades que favorecem os seguintes resultados:

• Transparência: permite ao usuário que os elementos estão localizados em diferentes lugares.

• Abertura:

Facilidade na comunicação entre elementos

• Escalabilidade:

atendimento de demandas futuras

• Proteção:

Aspectos de Segurança

• Qualidade de Serviço:

Desempenho (velocidade de processamento)

• Gerenciamento de Falhas:

Atividades p/ garantir a tolerância a defeitos

Modelos de Interação

É a comunicação entre sistemas distribuídos

• Modelo Interação Procedural

Os elementos envolvidos na comunicação devem estar disponíveis exclusivamente para interação

↳ comunicação síncrona;

• Modelo Interação Baseada em Mensagens.

Os elementos da comunicação podem estar envolvidos em diferentes interações

↳ comunicação assíncrona.

Modelo de Computação

↳ processamento.

O desenvolvimento de software distribuído pode considerar padrões arquitetônicos como ponto de partida.

- Cliente - Servidor

- Comod.

Padrão Arquitetural Mestre-Escravo
Este padrão estabelece que cada elemento da arquitetura de um software distribuído deve possuir um único papel.

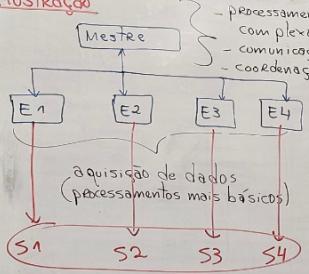
- O padrão estabelece os seguintes papéis: **mestre** e **escravo**.

- O elemento mestre é responsável pelos processamentos mais complexos, além de estabelecer a comunicação e realizar a coordenação dos elementos escravos.

- O elemento escravo é responsável por processamentos mais básicos, geralmente relacionados com a aquisição de dados.

Observação: no geral, se considera somente um elemento mestre.

Ilustração



Aula de Hoje

- Padrões Arquiteturais Específicos
- Software Como um Serviço

Padrões Arquiteturais Específicos

- A literatura apresenta diferentes padrões arquiteturais para softwares distribuídos.

- Tais padrões representam modificações de padrões arquiteturais clássicos, em especial cliente-servidor a camadas.

- Destaques para:
 - * Mestre - Escravo
 - * Cliente - Servidor de Duas Camadas
 - * Cliente - Servidor Multicamadas
 - * Arquitetura Distribuída de Elementos

* Arquitetura Ponto-a-Ponto

Cliente - Servidor de Duas camadas.

Uma comanda para o cliente e outra para o servidor.



↳ Número de comandos

é uma limitação para lidar com vários, são criados usuários:

Cliente-Foco: maior parte dos processamento será feita no cliente.

- Variante Usada quando se conhece o poder de processamento do cliente.

Cliente Magro: Maior parte dos procedimentos será feita no servidor.

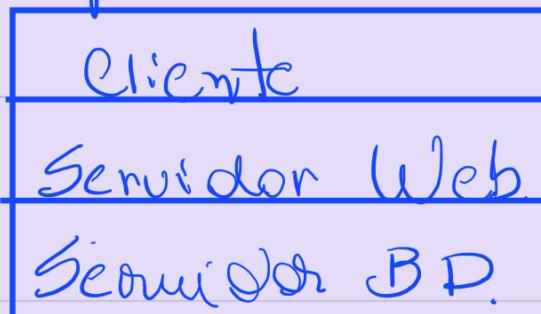
- É o mais utilizado.

Cliente-Servidor Multicomodos.

Número variável de comandos.

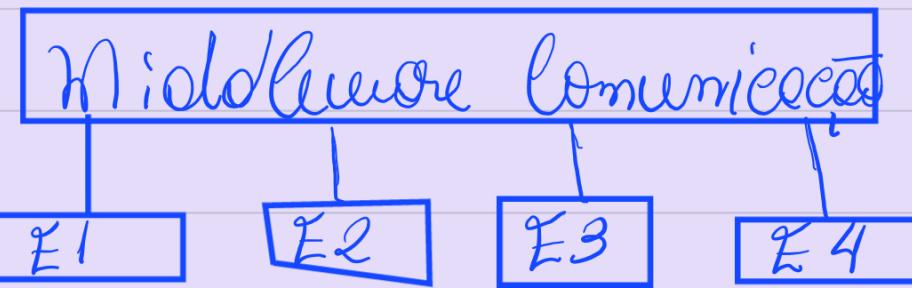
- Em geral uma comanda para o cliente e múltiplas para o servidor.

Ex:



Arquitetura Distribuída de Elementos
É uma estrutura para estabelecer a comunicação entre os diferentes elementos da arquitetura.

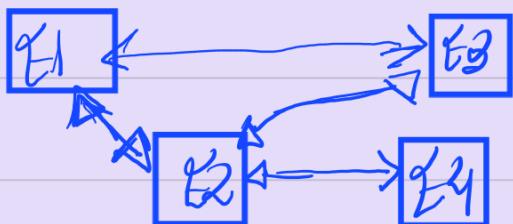
- Cada elemento da arquitetura pode atuar tanto como cliente e servidor.



Arquitetura Ponto-va-Ponto

Conhecida Peer-to-Peer

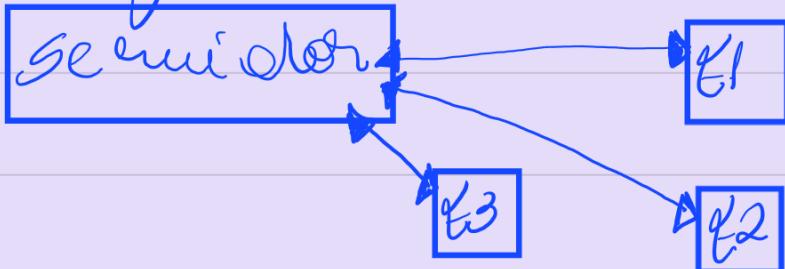
- ↳ Cada elemento pode atuar como cliente ou como servidor.



• Usado muito para
pirataria

- ↳ O p2p original, considera
uma arquitetura descentralizada

- ↳ Para gerenciamento considero servidor



Software como um serviço

- ↳ representa uma mudança de uso do software.

- ↳ Era tratado como produto, no contexto distribuído tornou-se um serviço

O software pode ser instalado em um local (servidor) e ser acessado por diferentes locais.

↳ Conhecidos como SaaS (Software as a Service)

Características

- Acessado via web.
- Com múltiplos usuários simultâneos, tem:
 - configuração de uso personalizada
 - customização de recursos com suas necessidades.
- Multilocação: relacionada com o armazenamento de dados.
- Escalabilidade: aumenta as demandas.

Engenharia de Softwares Distribuídos.

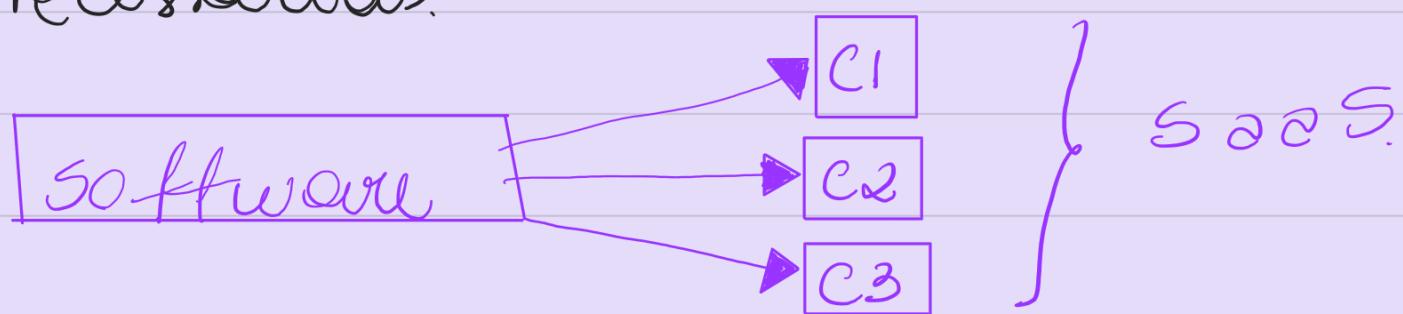
Arquitetura Orientada a Serviço (SOA)

representam uma possibilidade de aplicação de software distribuído.

Existem diferenças entre arquitetura de software OS e o software como serviços.

Por exemplo:

Arquitetura Orientado a Serviços
podem combinar serviços provenientes de diferentes fornecedores para oferecer todos as funcionalidades necessárias.



Arquitetura Orientado a Serviços (SOA)



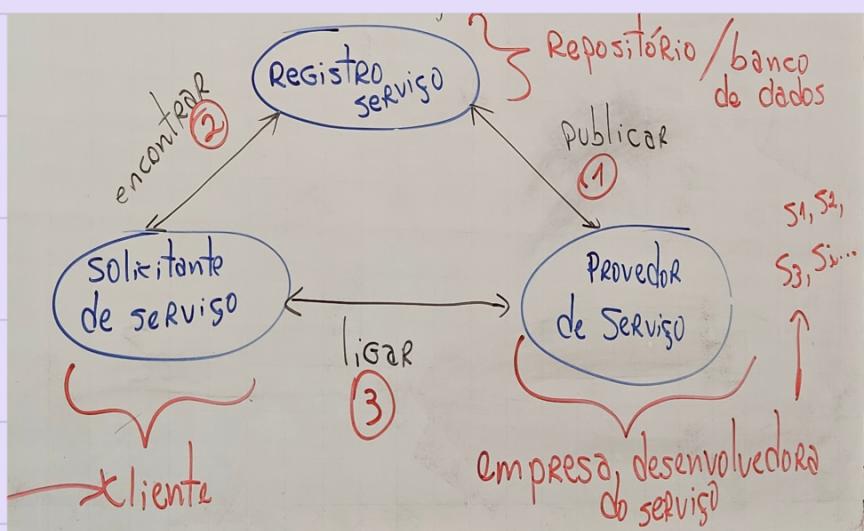
↳ Elemento básico de SOA é o serviço elemento autônomo e executável, ↳ independência

↳ Independencia tecnológica.

- ↳ Possibilidades de atender múltiplos clientes
- ↳ quando o serviço é oferecido via web é chamado de web service.

Organização do SQA.

organizações por um modelo de organização



- * As comunicações
ocorrem a partir
de protocolos:

- WS-BPEL (combinado)
desarrollado

- WSDL (Descrição de Serviços)

- SOAP: utilização de serviços (troco de mensagem).

A consideração de diferentes compromissos
e desempenho de SOA.

↳ Nesse contexto que as **arquiteturas REST**
foram criadas

Uso de **protocolo**
http

29/10

Arquitetura
Orientada
a serviços

{ Arquitetura SOA
Arquitetura REST
Arquitetura _____

Observações



Atividades consideradas na construção
de uma Arquitetura orientada serviço.

- ↳ Engenharia de Serviços
- ↳ Desenvolvimento de Softwares como serviço

Engenharia de Serviços

Busco desenvolver serviços genéricos
que podem ser usados em diferentes
aplicações:



- ① Identificação de serviços candidatos.
compreender a finalidade dos serviços
- ② Projeto de serviço
tecnologia, dados, repositórios, interface.
- ③ Implementação e implantação de serviço
construção e instalação do serviço.

Desenvolvimento de software como serviço

Busca auxiliar no desenvolvimento de software com base na composição de serviços

- ① Formulários de workflow preliminar.
definição de requisitos de serviços
- ② Descoberto de serviços
busca de serviços que atendem os requisitos
- ③ Seleção de serviços
escolha de serviços a ser utilizados
- ④ Refinamento do workflow
Evolução do workflow (adição, remoção, alteração das atividades/serviços).

⑤ Criação do programa baseado no workflow

- contratos do serviço
- consideração de diferentes tecnologias.
LP(REST) e WS-BPEL(SOA)

⑥ Teste de Serviço

- teste do serviço isolado.
- teste do software.



Microserviços

Conteúdo

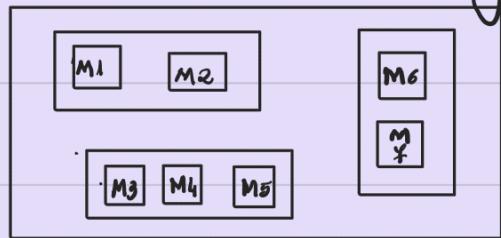
- Durante muito tempo os softwares foram entendidos como monólitos. único processo a ser executado
- Começa esse processo receber diferentes recursos do sistema operacional.
Ex: CPU, memória

↳ A manutenção de monólitos vêm com apresentar problemas e surge a arquitetura baseada em microserviços

- Microserviços é um serviço independente, tal como o pratico anterior.

Construção de microserviços

A partir de agrupamentos de módulos.



Cada microserviço
pode funcionar
como software inde-

pendente, com finalidade específica.

Vantagens:

- Maior facilidade de manutenção
- Maior facilidade para escalabilidade.
microserviços não vai na prona!

Software Embutido.

Um software embutido pode ser definido como um software que pode ser utilizado em diferentes dispositivos, assim como computadores e geladeiras.

Características.

- Executação contínua e ininterrupta.
- Interações são controladas e sempre visíveis.

Necessidade de pensar no gerenciamento de problemas

- Limitações variadas

recursos computacionais { Processamento
Armazenamento

Obs:

- Um sistema é composto por hardware e software.

- Um software em barcode é parte de um sistema embutido (embarcado).

Projeto.

O projeto de softwares embutidos deve considerar diferentes tipos de informação.

↳ Requisitos Não-Funcionais.

Representam restrições que o software, embutido deve respeitar.

Exemplos.

◦ Tempo de Resposta.

Tempo máximo para retorno de resultado.

◦ Confidabilidade:

Retorno de um resultado correto.

◦ Disponibilidade:

Garantir a execução contínua

◦ Segurança:

Garante a integridade dos dados.

Tecnologias

As tecnologias são combinadas sob diferentes perspectivas de vista em um projeto de software embutido.

Exemplos:

- Linguagens de Programação
C, Assembly.

- Armazenamento de Dados.
Arquivos (binários), estruturas.

Quais veem em ORD.

Observações

- softwares embarcados são considerados em diferentes contextos.

Exemplo: interações em tempo real.

Sistema de Tempo Red.

- São Reativos.

Respondem a evento.

eee.

- Respostas baseadas em estímulos.
- Periódicos: possuem frequência conhecida
- Aperiódicos: frequência desconhecida.

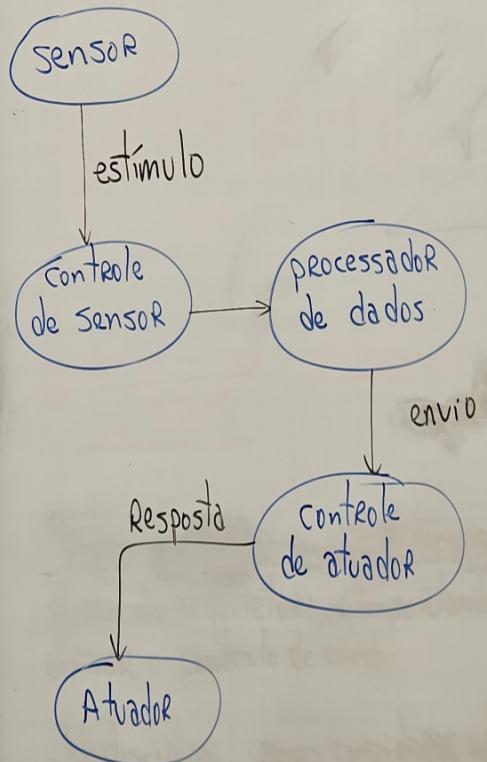
Software Embutido.

Viremos considerar softwares embutidos que são formados por sensores e atuadores, representados por processo programados em execução

No contexto de sensores e atuadores os seguintes elementos são necessários:

- Controle de sensores (colha e agrupamento de dados).
- Processador de dados (realiza cálculos e representa respostas).
- Controle de atuadores (envio de sinal de comando para atuadores).

Ilustração - Relação entre os Elementos



Uma vez que os sensores e atuadores foram representados, é importante entender sobre os processos de desenvolvimento.

Tais processos consideram tanto o desenvolvimento do software quanto o de desenvolvimentos do sistema embarcado.

Existem diferentes modelos de processo

- ↳ Destaque para os seguintes atuadores:
 - Seleção de Plataforma, ambiente e plataforma.

- Identificação de estímulos e respostas.
eventos ocasionados
- Análise de Timing.
requisitos de tempo.
- Projeto do Processo.
Fluxo de execução
- Projeto de Algoritmo.
algoritmos para implementação flexível.
- Projeto de Dados.
identifica os dados necessários.
- Programação de Processo.
implementações do fluxo.

Obs

O sistema embeded pode ser composto por múltiplos processos, logo o processo representado anteriormente pode ser unatriziado. múltiplos usos.

Para auxiliar no desenvolvimento de softwares em veículos, padrões de arquitetura foram desenvolvidos.

Rodrões Arquitetuais para Softwares Embutidos

- Observar e Reagir.
sistemas de monitoramento.
- Controle de Ambiente.
sistemas de controle.
observação e reação.
- Pipeline de Processos.
Aquisição de dados coletados em tempo real.