Arquitetura de Software

Introdução

NAIRON NERI SILVA

Primeira Analogia:

O que é Arquitetura de Software?

- Significa coisas diferentes para pessoas diferentes...
  - Para **um desenvolvedor**... estrutura do sistema a ser construído
  - Para o **criador de um framework**... a forma dos sistemas criados com o framework
  - Para **um testador**... a forma do sistema a ser testado
  - Para **os demais envolvidos**... estrutura de alto nível para a solução do problema que o cliente quer ver resolvido

Definição 1

"O conjunto de decisões significativas sobre a organização de um sistema de software, a seleção de elementos estruturais e suas interfaces que compõem o sistema, juntamente com o seu comportamento, conforme é especificado nas colaborações entre os elementos, a composição desses elementos estruturais e comportamentais em subsistemas progressivamente maiores e o estilo de arquitetura que orienta essa organização – esses elementos e suas interfaces, suas colaborações e sua composição."

(*Tradução de Grady Booch, James Rumbaugh, Ivar Jacobson*)

Definição 2

"A arquitetura de software não está relacionada somente com a estrutura e o comportamento, mas também com a utilização, funcionalidade, desempenho, flexibilidade, reutilização, abrangência, restrições e ajustes econômicos e tecnológicos e questões estéticas."

(*Tradução de Grady Booch, James Rumbaugh, Ivar Jacobson*)

Motivação

- Crescente tamanho e complexidade dos sistemas

- Como organizar e controlar
  - a estrutura dos sistemas de software?
  - os protocolos de comunicação, sincronização e acesso a dados?
  - a atribuição de funcionalidade à elementos do projeto?
  - a distribuição física?
  - a composição dos elementos de projeto?
  - a escalabilidade e a performance?
  - a escolha entre alternativas de projeto?

Princípios Arquiteturais

- Regras ou padrões que ajudam os engenheiros de software a tomar as decisões corretas

- De onde surgem essas “regras” e “padrões”?

1. Aplicação prática de propostas de organização
2. Avaliação empírica (“baseada na experiência”)
3. Identificar soluções recorrentemente bem sucedidas
4. Catalogar para utilização futura

Definindo a Arquitetura de Software

- Inicia com o entendimento dos requisitos do sistema a ser desenvolvidos:
  - Identificação dos requisitos arquiteturalmente significantes – têm impacto na arquitetura
  - Tomar decisões com base nesses requisitos

- As decisões tomadas durante a criação da arquitetura são fundamentais para o sistema
  - Pois elas irão embasar todas as demais decisões seguintes

Definindo a Arquitetura de Software

- Algumas arquiteturas de software são melhor descritas como ”grandes bolas de lama”
  - Difícil de construir
  - Difícil de manter
  - Podem não atender aos requisitos do cliente

Componentes de uma Arquitetura de Software

1. Metas e ”filosofia” do sistema
2. Premissas e dependências arquiteturais
3. Requisitos arquiteturalmente significantes
4. Instruções de empacotamento para os subsistemas e componentes
5. Subsistemas e ”camadas” críticas
6. Referências aos elementos de projeto arquiteturalmente significantes
7. Interfaces críticas para o sistema
8. Cenários chave que descrevem comportamentos críticos para o sistema

Documento de Arquitetura

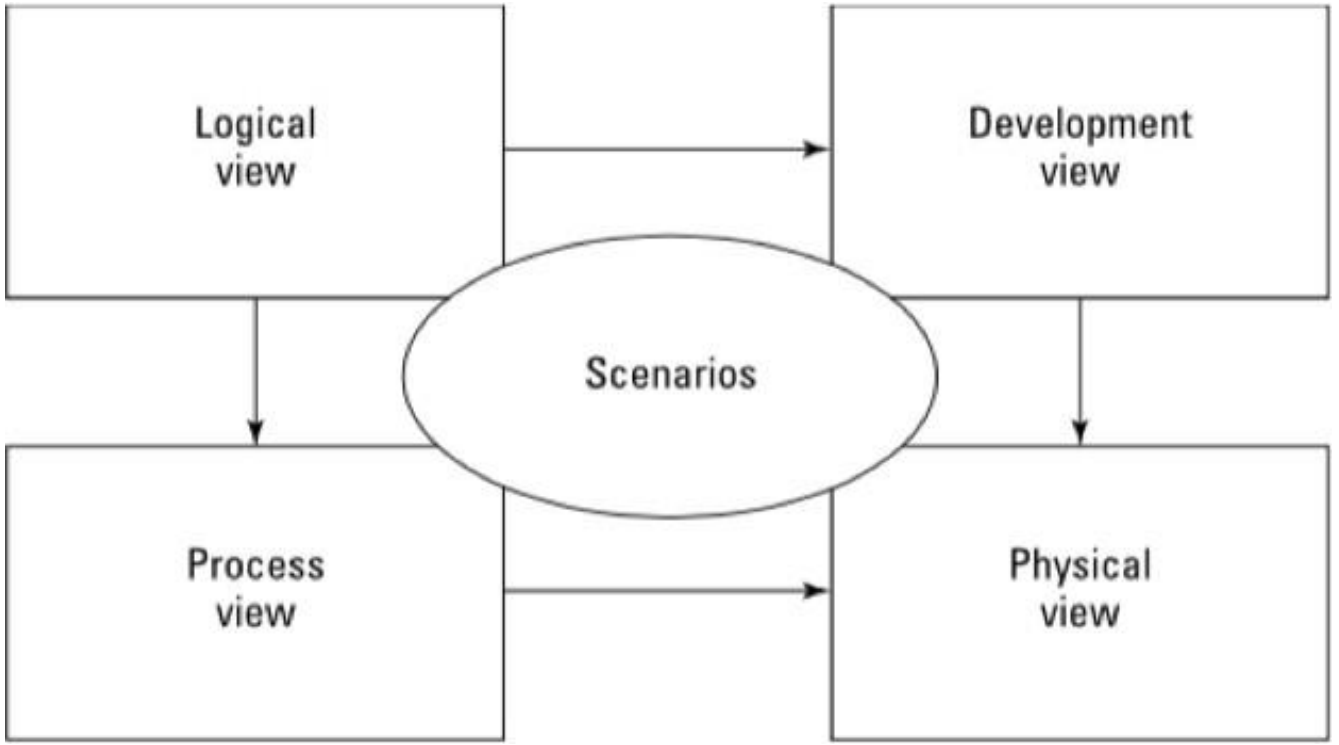
- Todos os componentes descritos anteriormente estarão presentes no documento de arquitetura, incluindo:
  - Decisões chave que dão forma a arquitetura
  - Como as partes trabalham juntas
  - Como o sistema será empacotado
  - Ilustração das visões arquiteturais (pontos de vista)

Visões Arquiteturais

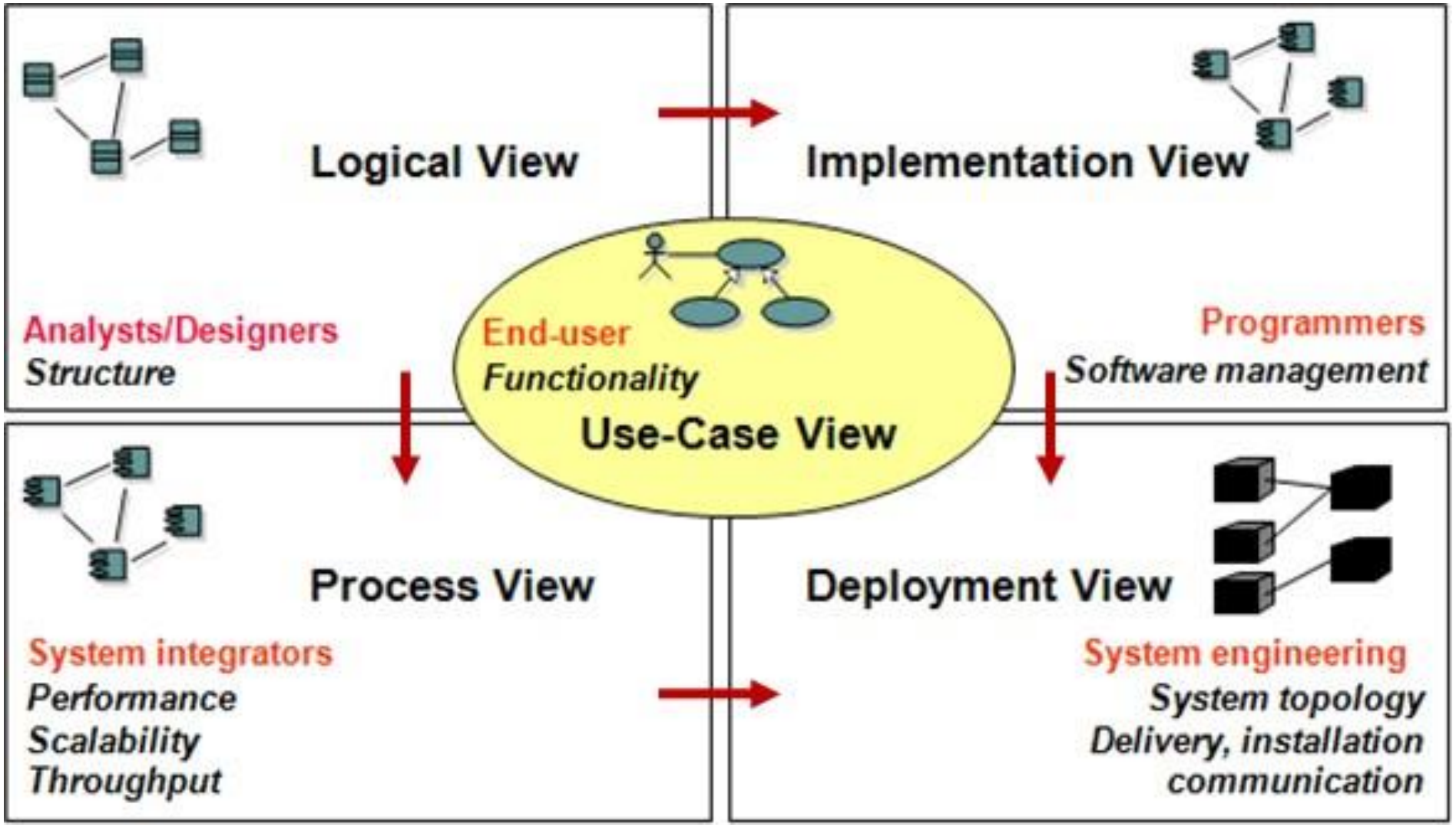
1. Lógica
   - Mapeia o sistema em classes e componentes, partes que oferecem as funcionalidades ou que interagem com o usuário
2. Processos
   - Explica como as partes da arquitetura trabalham juntas e permanecem sincronizadas – mapeia unidades de computação
3. Física
   - Explica como o software que implementa o sistema é mapeado em plataformas computacionais (e distribuídas)
4. Desenvolvimento
   - Explica como será a gerência do software durante o desenvolvimento do mesmo – permite o trabalho em equipe

Visões Arquiteturais

- Modelo 4+1 para uma arquitetura:



Fonte: https://www.igti.com.br/blog/arquitetura-de-software-definicao-negocios/


Logical View
Implementation View
Analysts/Designers
Structure
End-user
Functionality
Use-Case View
Programmers
Software management
Process View
Deployment View
System integrators
Performance
Scalability
Throughput
System engineering
System topology
Delivery, installation
communication

Identificando o Problema a ser Resolvido

- Qual o Problema eu estou Resolvendo?
- Principal questionamento para a definição da arquitetura de software a ser utilizada
- Só identificando o problema é possível
  - Encaminhar a solução do problema
  - Atender às necessidades do cliente

Atributos do Problema

1. **Função**: descreve o problema a ser resolvido

2. **Forma**: descreve o formato da solução e como ela se encaixa no ambiente operacional e tecnologias

3. **Economia**: descreve quanto custa a construção, operação e manutenção da solução

4. **Tempo**: descreve como se espera que o problema mude com o passar do tempo

Atributos do Problema

- Pergunte ao cliente o que ele quer e por que ele quer dessa forma

- Por fim, a arquitetura proposta deve:
  - Fazer o que o cliente quer
  - Ao custo que o cliente está disposto a pagar
  - E no calendário que satisfaça as necessidades do cliente

Declaração do Problema

- Exemplo: sistema de folha de pagamento

- **Passo 1**: estabeleça as metas do processo de definição do problema

Decida quanto tempo você pode gastar no desenvolvimento da declaração do problema e quanto detalhe a declaração do problema precisa de ter.

Para um sistema de folha de pagamento, você deverá identificar as restrições para a solução (questões que afetam sua forma, economia e tempo) e certifique-se de que você entenda a função de alto nível: obter trabalhadores remunerados.

Declaração do Problema

- **Passo 2**: reúna os fatos

Nessa etapa, você trabalha com o cliente para entender suas necessidades, como são satisfeitas tais necessidades no momento, e qual a plataforma computacional que espera ser usada na solução. Você também deve identificar os demais envolvidos e sistemas relacionados, conhecidos como atores, que irão interagir com o sistema. Seu objetivo é descobrir o máximo que puder sobre o problema, a necessidade e as expectativas sobre o sistema.

Para o sistema exemplo de folha de pagamento, você iria reunir fatos sobre o número de empregados, com que frequência eles são pagos, como são calculados os salários, e que as potenciais deduções que são retiradas dos salários.

Declaração do Problema

- **Passo 3**: descubra os conceitos que são essenciais para a solução e que irão moldar sua arquitetura

Nesta etapa, você irá procurar pelos principais conceitos envolvidos. Você descobrirá pressupostos, equações, regulamentos, modelos de processos, restrições de uso, e outros conceitos fundamentais.

Para o sistema de folha de pagamento, você irá descobrir as equações utilizadas para calcular pagamento de um empregado e determinar como irregularidades no pagamento serão comunicadas ao sistema.

Declaração do Problema

- **Passo 4**: determine o que o cliente deve ter para estar satisfeito com a solução

Esta etapa envolve a compreensão das necessidades e expectativas do cliente com base nos conceitos subjacentes que você encontrou na terceira etapa. Qual é o mínimo que o cliente precisa ter para estar satisfeito com a solução que você projetar?

O sistema exemplo de folha de pagamento precisa receber (1) as horas que cada funcionário trabalhou, (2) a taxa base dos salários e (3) as deduções relacionadas, para calcular os valores a serem pagos. O sistema também deve realizar a transferência ou de alguma outra forma efetuar os pagamentos aos empregados.

Declaração do Problema

- **Passo 5**: escreva a declaração do problema

Com base na sua compreensão do problema ao completar os últimos quatro passos, você pode escrever uma declaração do problema que traz os quatro atributos de função, forma, economia e tempo de uma forma que o cliente possa entender.

Para o sistema exemplo de folha de pagamento, a declaração do problema seria ”Calcular e pagar os funcionários de acordo com o trabalho realizado [Função] por meio de um sistema interativo para a entrada das horas trabalhadas, e que possibilite fazer o pagamento através de depósito direto [Forma]. A solução deve estar disponível em três meses [Tempo] segundo o preço negociado [Economia].”

Definindo os Casos de Uso Importantes

- A declaração do problema precisa ser refinada

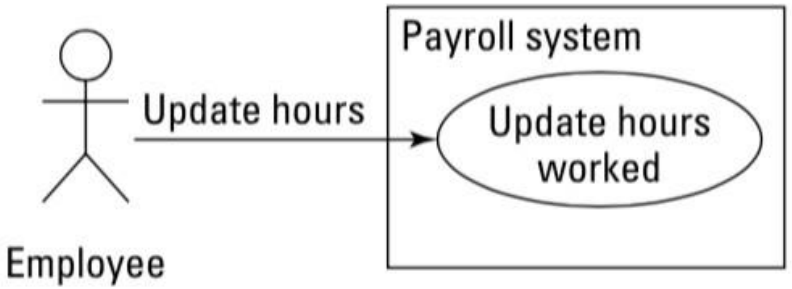
- Precisa ser detalhado o que é necessário para a resolução do problema
  - A forma natural de fazer isso é escrevendo casos de uso
  - Os cenários apresentados nos casos de uso conectam as diferentes visões da arquitetura

Escolhendo que Funcionalidades Capturar

- Dicas:
  - Funcionalidades que agregam ”valor” ao sistema
  - Funcionalidades que envolvam atores externo

Escolhendo que Funcionalidades Capturar

- Exemplo do sistema de folha de pagamento



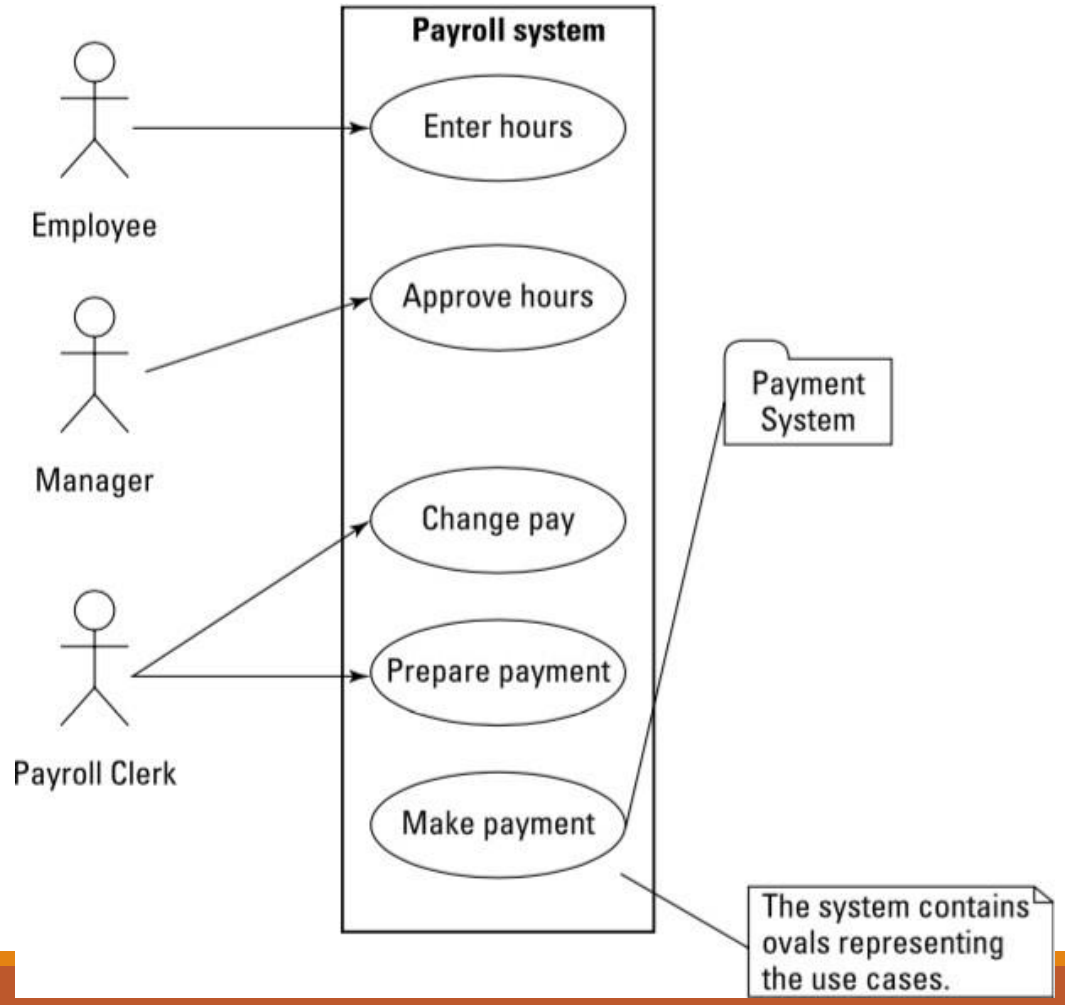
Use case: Update hours worked.
Summary: The Employee actor updates the number of hours that they have worked during a pay period.
Main course:
1. The actor accesses the web interface to the system
1.1 System determines Actor identification
2. Actor specifies the time period to be updated, i.e. which work week
2.1 The System provides a display of weeks to be updated
3. Actor specifies hours worked
3.1 System performs sanity checks
3.2 System acknowledges entry
3.3 System prompts for next action
4a. Actor logs off
4b. Actor restarts from Step 2

Identificando os Atores

1. Atores realizam funções descritas nos CDUs
2. Atores executam vários papéis: cliente, usuário, funcionário, gerente, atendente, etc.
3. Atores podem se relacionar a vários casos de uso
4. Atores podem ser outros sistemas
5. Nenhum dos atores deve representar componentes internos do sistema

Diagramando o Sistema

- Um diagrama de casos de uso provê uma visão alto-nível de como os atores interagem com o sistema



Identificando os Requisitos

- O problema a ser resolvido necessita ser traduzido em requisitos detalhados
  - Lista de coisas que você precisa incluir na solução
- Definição dos requisitos funcionais
  - Tudo o que o usuário pode obter utilizando o sistema
- Definição dos requisitos não-funcionais
  - Características que o sistema deverá ter (manutenabilidade, alta performance, reusabilidade, etc.)

Revisando os Requisitos

- Busca evitar requisitos pouco claros e incompletos

- Algumas ações:
  1. Busque identificar e descrever os requisitos implícitos ou ocultos
  2. Valide todos os pressupostos
  3. Não estenda premissas
  4. Evite especificações dúbias
  5. Evite requisitos inconsistentes ou conflitantes
  6. Case o escopo da solução com o escopo do problema

Praticando...

- Crie a declaração do problema para a situação abaixo, baseando-se na função, forma, economia e tempo:

- Um provedor de internet deseja uma solução para que seus técnicos possam ser avisados via aplicativo em qual cliente eles devem ir;
- Os técnicos quando terminarem o atendimento informam no App que estão disponíveis;
- O sistema precisa ser inteligente o suficiente para enviar o técnico no cliente que ele está mais próximo;