Aula 06

Processo para Projeto Arquitetural

Análise e Projeto Arquitetural de Software

Prof. Thiago

Pós-Graduação em Gestão de Sistemas de Informação

Objetivos

- Após esta aula, você deverá ser capaz de:
  - Definir arquitetura de software;
  - Descrever e utilizar um processo para projeto arquitetural;
  - Descrever atributos de qualidade para uma arquitetura de software;
  - Utilizar técnicas para projeto arquitetural.

Definição

- Segundo Sommerville (2011), **projeto arquitetural** o processo que identifica quais subsistemas compõem o sistema e qual é o *framework* de comunicação entre eles

- Entrada
  - Os modelos do sistema gerados na fase de análise

- Saída
  - Descrição da arquitetura do software, composta de uma série de representações gráficas dos modelos do sistema, associadas ao texto descritivo

Definição alternativa

Arquitetura é a visão da modelagem na qual você não pode eliminar mais nada sem deixar de entender o propósito do software

Adaptado de Phillipe Krutchen – The Rational Unified Process: An Introduction

Arquitetura e requisitos

Há uma considerável sobreposição entre o processo de Engenharia de Requisitos e o Projeto de Arquitetura

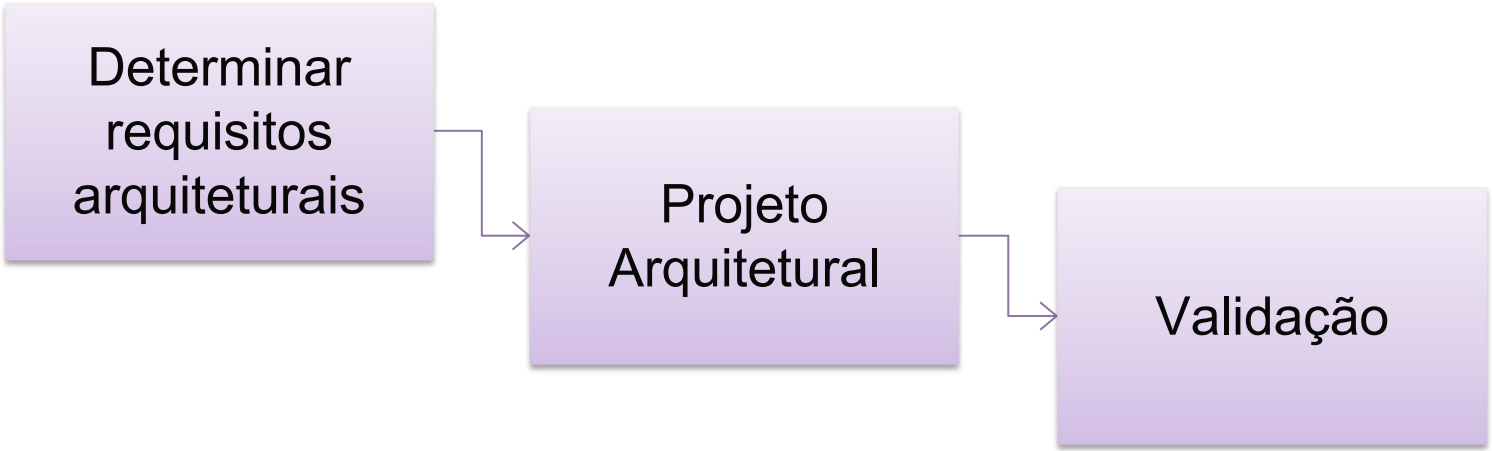
Requisitos funcionais → atendidos pelo detalhamento dos modelos estruturais e comportamentais [Análise]

Requisitos não-funcionais → atendidos pelas decisões de projeto tomadas para a arquitetura [Projeto]

Processo de Projeto Arquitetural

Processo para Projeto Arquitetural

Gorton (2006) propõe as seguintes atividades para o processo de Projeto Arquitetural:



Processo para Projeto Arquitetural

- Determinar requisitos arquiteturais
  - Identificar e priorizar requisitos que tem influência no projeto de arquitetura
- Projeto Arquitetural
  - Aplicar estruturas e padrões estruturais e de controle para a definição da arquitetura
- Validação
  - Utilizar especificação de cenários e prototipação para verificar a validade das decisões de projeto tomadas

Vamos discutir alguns atributos de qualidade, associados a requisitos não-funcionais, e como eles afetam a arquitetura

Aspectos de qualidade

1. Desempenho e escalabilidade
2. Segurança
3. Tolerância a falhas
4. Possibilidade de modificação
5. Operabilidade
6. Possibilidade de compreensão

Desempenho

Mesmo que a maioria das aplicações de TI não opere, efetivamente, em tempo real (como alguns sistemas de robótica), frequentemente há a necessidade de processar grandes quantidades de dados em tempos reduzidos

Ex: Finanças, Telecom, Governo

Medidas frequentes:

- ***Throughput***: quantidade de trabalho por unidade de tempo. Tipicamente, transações por segundo ou mensagens por segundo
- **Tempo de resposta**: ex. para sistemas de venda online
- **Tempo limite de processamento**: ex. sistemas batch, folha de pagamento, previsão do tempo

Escalabilidade

Envolve “em que grau a solução do problema continua funcionando quando o tamanho do problema aumenta”.

O que pode aumentar em uma solução baseada em TI?

- **Carga de requisições**
- **Número de conexões simultâneas**
- **Tamanho dos dados a processar** (como comprimento de mensagens e tamanho de repositórios de dados)
- **Implantação** (ex: quantidade de máquinas para implantação do sistema)

Segurança

No nível arquitetural, tratar a segurança significa prover mecanismos estruturais que dão suporte a requisitos de segurança.

Os principais, segundo Gorton (2006), são:

- **Autenticação**: como no login de usuários
- **Autorização**: para execução de operações no sistema
- **Encriptação**
- **Integridade**: garantia que uma mensagem não será alterada
- **Não-repúdio**: garantia da identidade de um remetente

Tolerância a falhas

Refere-se à capacidade de manter alguma qualidade de serviço em caso de falha de software, ou comportamento imprevisto de usuários, software e hardware.

Segundo Barbosa (2009), algumas formas de mensurar o grau de tolerância a falhas:

- Serviço continua funcionando em caso de falha de n servidores;
- Variação no tempo de resposta ou no número de usuários simultâneos no caso de uma falha de hardware;
- Como o sistema se composta em caso de entrada de dados inválidos.

Possibilidade de modificação

“All capable software architects know that along with death and taxes, modifications to a software system during its lifetime are simply a fact of life” (Gorton, 2006)

É a capacidade do software sofrer modificações com baixo impacto em áreas não-relacionadas à mudança

Prever a possibilidade de modificação exige gerar medidas de esforço e/ou custo para executar uma mudança

Muito relacionada às métricas clássicas de software, como níveis de coesão e acoplamento e complexidade ciclomática

Operabilidade

Capacidade de o usuário operar ou controlar o sistema;

Muito importante em grandes sistemas de software, onde há um tipo de usuário que é o administrador do sistema.

Algumas operações que estão no escopo deste aspecto de qualidade são as operações administrativas e vinculadas a decisões de arquitetura, tais como:

- ligar, desligar ou verificar estado de servidores;
- realizar backup dos dados;
- “barrar” conexões.

Possibilidade de compreensão

Relacionada à quantidade de conceitos ou operações que o usuário precisa aprender para fazer com que o software funcione

Também ligada à quantidade e qualidade da documentação do sistema

Da mesma forma que a operabilidade, está ligada à usabilidade do sistema

Priorização de requisitos arquiteturais

Em geral uma categorização dos requisitos como descrito abaixo é suficiente para o processo de projeto arquitetural:

- **Alto**: a aplicação deve dar suporte ao requisito pois ele irá direcionar o projeto arquitetural
- **Médio**: o requisito será suportado em algum momento, mesmo que não seja no primeiro release
- **Baixo**: suporte ao requisito é desejável mas ele não é um direcionador do projeto arquitetural

Priorização de requisitos arquiteturais

Muitas vezes os requisitos arquiteturais podem ser conflitantes como, por exemplo:

- Reutilização de componentes versus rápido lançamento
- Redução de gastos em componentes “de prateleira” versus redução do tempo de desenvolvimento

Não há solução simples para esses conflitos; é papel do arquiteto discutir as questões com os stakeholders e considerar os pontos de equilíbrio possíveis.

Como aplicar o conceito de padrões para o projeto arquitetural?

Padrões arquiteturais

Soluções gerais baseadas em problemas recorrentes do projeto arquitetural

Em geral, focam em uma das seguintes alternativas:

- Estruturação dos subsistemas
- Fluxo de controle dos subsistemas

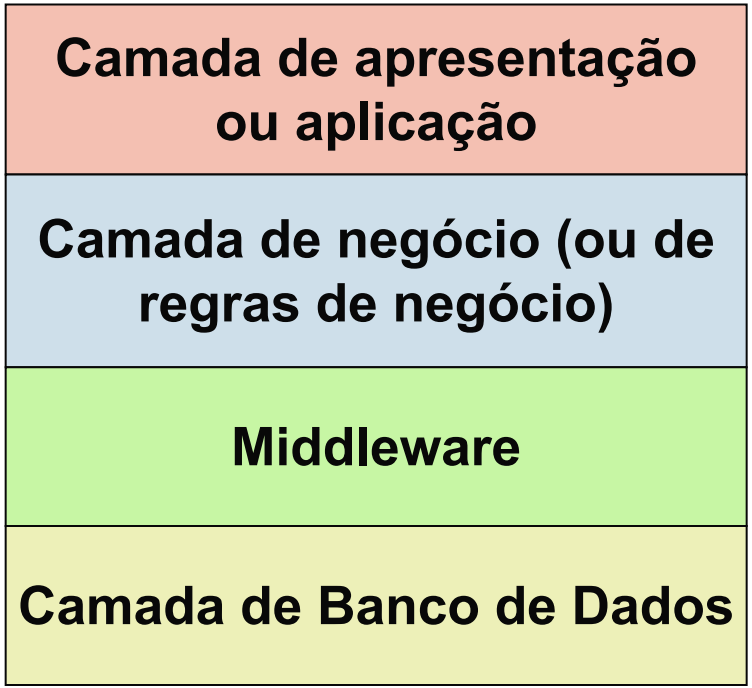
Modelo em camadas (N-tier)

Solução clássica para garantir separação de responsabilidades

A comunicação entre as camadas é **síncrona** (baseada no modelo requisição-resposta)

Desenvolvimento e implantação flexível: cada camada pode rodar em uma estrutura física independente das demais (e garantindo redundância)

Modelo em camadas (N-tier)



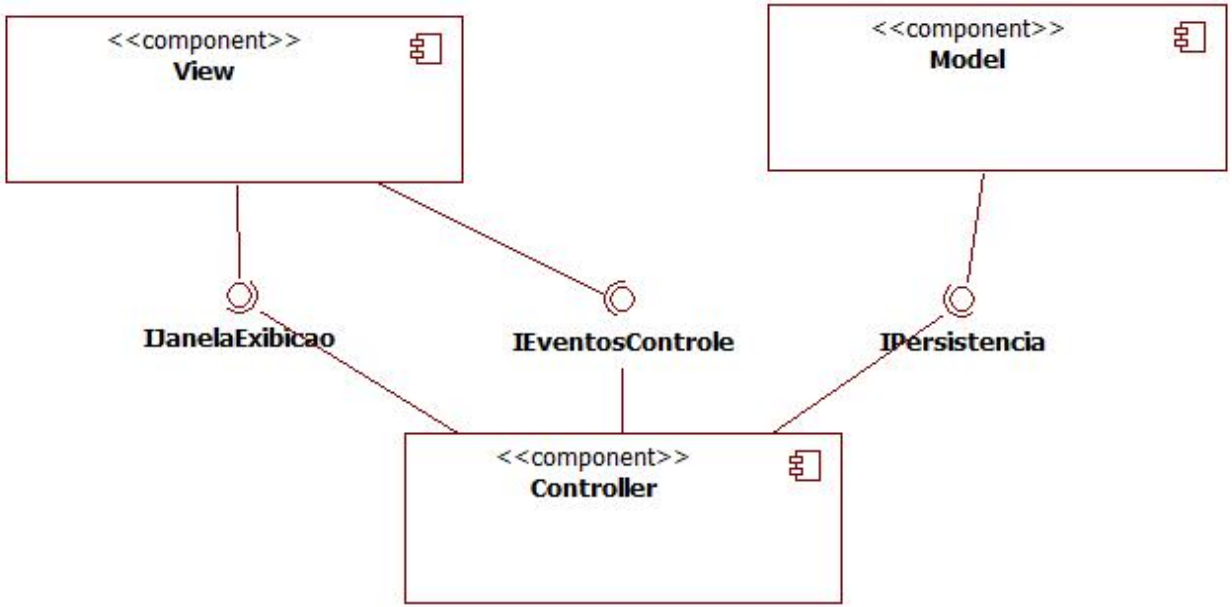
Engloba os subsistemas que lidam com a interface final com o usuário e apresentação das informações

Engloba as regras para processamento das informações

Oferece subsistemas para classes utilitárias e serviços para intercomunicação de objetos distribuídos em ambientes heterogêneos, entre outros.

Provê serviços de armazenamento e recuperação de informações

Diagrama de componentes do MVC



Cada componente realiza (implementa) uma interface, que pode ser utilizada por outros componentes.

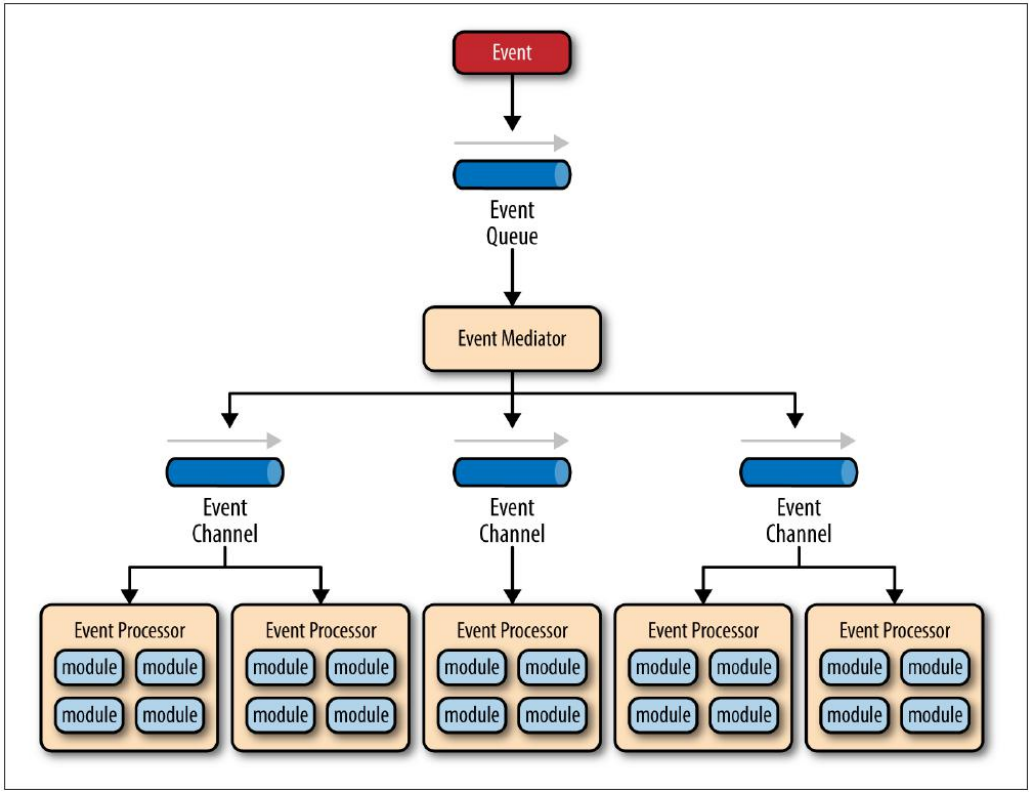
Arquitetura orientada a eventos

Modelo focado na comunicação entre componentes, que usualmente é **assíncrona**

Visa a escalabilidade e alto nível de desacoplamento entre componentes

Depende de um componente **Mediador de Eventos**, centralizado (próxima figura) ou um Broker, fila descentralizada de eventos

Arquitetura orientada a eventos



Arquitetura orientada a eventos

Algumas soluções para implementação do Mediador de Eventos:

- Spring Integration
  https://projects.spring.io/spring-integration/
- Apache Camel
  http://camel.apache.org/

- Apache ODE – orquestração de eventos baseada em BPMn
  http://ode.apache.org/

Quais técnicas de projeto arquitetural podem ser utilizadas para garantir cada aspecto de qualidade?

Alocação de componentes

- Identificar os maiores componentes da aplicação e como eles se “encaixam” no padrão escolhido
- Identificar a interface ou serviços que cada componente oferece
- Identificar as responsabilidades do componente, definindo qual o comportamento garantido quando ele recebe uma requisição
- Identificar dependências entre componentes
- Identificar partições dos componentes que sejam candidatas a distribuição em servidores de rede

Técnicas de Projeto Arquitetural

- Minimizar dependências entre componentes
- Projetar componentes que agrupam um conjunto de funcionalidades altamente coesas
- Isolar dependências de middleware ou componentes de prateleira
- Use a decomposição para estruturar as estruturas internas de um componente de forma hierárquica
- Minimizar chamadas entre componentes, especialmente se estas se relacionarem a componentes distrubuídos

Como garantir que a arquitetura funcionará conforme esperamos?

Validação

Duas técnicas de validação usuais para o projeto de arquitetura são:

- **Cenários**
- **Prototipação**

Cenários

Técnica simples que consiste na identificação de estímulos que podem exercitar aspectos do projeto

Pode estar relacionado aos requisitos prioritários do projeto de arquitetura

Atributo de qualidade	Estímulo	Resposta
Disponibilidade	A conexão de rede com os consumidores de mensagens falhou	Mensagens são armazenadas no servidor até que a conexão seja restabelecida. Mensagens só serão perdidas se o servidor falhar antes da conexão retornar
Segurança	Nenhuma requisição é feita em uma conexão de usuário por mais de 10 minutos	O sistema trata a sessão como potencialmente insegura e invalida as credenciais de segurança associadas à sessão. O usuário deve se logar novamente.
Possibilidade de modificação	O fornecedor do engine de transformação saiu do mercado	Um novo engine deve ser adquirido; a camada de serviço abstrata que fornece acesso ao engine deve ser reescrita para acompanhar as mudanças no componente. Os componentes clientes não são afetados pois utilizam apenas a camada abstrata de acesso

Prototipação

Podem ser de dois tipos:

- **Prova de conceito**: identificam se a arquitetura pode ser construída para satisfazer os requisitos
- **Prova de tecnologia**: verificam se a tecnologia escolhida (middleware, bibliotecas, aplicações integradas) se comportam da maneira esperada

Referências

- BARBOSA, G. M. G. **Um livro-texto para o ensino de projeto de arquitetura de software**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, 2009.

- GORTON, I. **Essential Software Architecture**. Heidelberg: Springer, 2011

- SOMMERVILLE, I. **Engenharia de Software**. 9ª ed. São Paulo: Pearson, 2011.

- RICHARDS, M. Software Architecture Patterns. O'Reilly, 2015.