Arquitetura de Software

Marcos Monteiro, MBA, ITIL V3

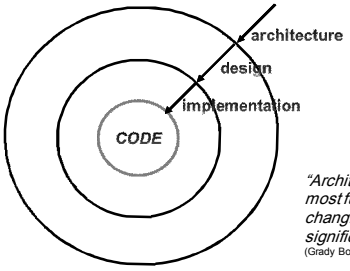
http://www.marcosmonteiro.com.br
contato@marcosmonteiro.com.br

Arquitetura

(Um pouco de cultura inútil)

- Grécia, 500 a.C.:
  - Arkhitekton – construtor chefe
  - Arkhi – chefe ou mestre
  - Tekton – trabalhador ou construtor
  - Tekhne – arte ou habilidade

Arquitetura de Software

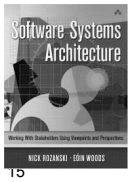


"Architectural decisions are the most fundamental decisions; changing them will have significant ripple effects."
(Grady Booch)

Definição de Arquitetura de Software

O que é arquitetura de software?

– "Arquitetura de software é o conjunto de decisões que, se incorretas, podem causar o cancelamento do seu projeto."



Definição de Arquitetura de Software

O que é arquitetura de software?

– "A arquitetura de software (…) é a estrutura (…) composta por elementos de software com suas propriedades externamente visíveis e o relacionamento entre eles."

– Bass, Clements e Kazman em *Software Architecture in Practice*

16

Definição de Arquitetura de Software

- Enquanto produto:
  - Modelo abstrato que representa a estrutura de um software.
  - NÃO é o software operacional.
  - NÃO é uma pilha de tecnologias.

- Enquanto disciplina:
  - Campo da engenharia de software que trata das estruturas de software, procurando reduzir complexidades através de abstrações e separação de interesses.

17

Utilidade da Arquitetura de Software

Para que serve arquitetura de software?

– A arquitetura permite:
- Analisar a eficácia do design em resolver os requisitos.
- Considerar alternativas arquitetônicas em estágios iniciais.
- Reduzir os riscos associados com a construção do software.

22

Utilidade da Arquitetura de Software

Qual a importância da arquitetura de software?

➢Como construir uma casa sem as plantas?

– A arquitetura de software fornece a visão geral (*big picture*) para garantir que se está no caminho correto.

– Habilita a comunicação entre os interessados sobre como o software vai ser construído.

– Logo no início do ciclo de vida, ressalta decisões de *design* que têm profundo impacto em todo trabalho.

– Constitui um modelo simples e intelectualmente compreensível dos componentes do software.

23

POC = Proof of Concept

- O que é um POC arquitetural?
  - É uma solução, mesmo que conceitual, para satisfazer um requisito de qualidade.
- Pode apresentar-se em alguma das formas:
  - Lista de tecnologias conhecidas.
  - Esboço de modelo conceitual (ex: usando UML).
  - Simulação de solução.
  - Um protótipo executável.
- O POC deve ter critérios de avaliação
  - Critério de sucesso deve ser definido antes da execução da prova.
  - O arquiteto deve ser capaz de analisar se os objetivos foram atingidos.

Arquitetura e processos de Desenvolvimento de Software

Alguns Processos de Desenvolvimento de Software :

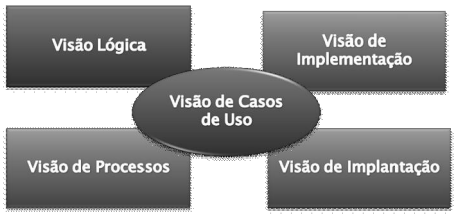
- RUP – Rational Unified Process
- XP – Extreme Programming
- OpenUP – Versão "magra" do RUP com filosofia ágil

Exemplos de Arquitetura de Software

- Cliente-Servidor
- Computação distribuída
- P2P
- Quadro Negro
- Criação implícita
- Pipes e filtros
- Plugin
- Aplicação monolítica
- Modelo em três camadas
- Analise de sistema estruturada
- Arquitetura orientada a serviço
- Arquitetura orientada a busca

Representação da Arquitetura de Software

Franco padrão: Modelo de Visualização 4+1 de Philippe Kruchten



Projetando uma Arquitetura de Software

Nosso exemplo:

Sistema para uma rede de hotéis

Projetando uma Arquitetura de Software

Nosso processo didático (e simplicista):

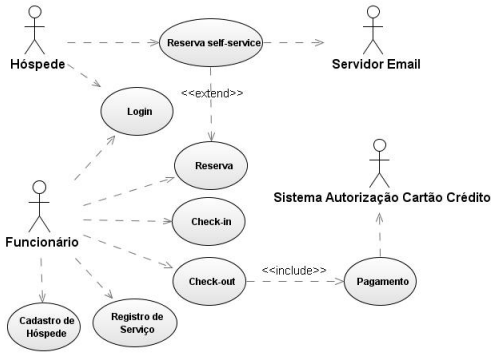
- Identificar requisitos relevantes à arquitetura
- Classificar e priorizar requisitos relevantes à arquitetura
- Identificar riscos e restrições
- Atacar maiores riscos técnicos
- Desenvolver visão geral da arquitetura
- Definir organização de alto nível
- Identificar mecanismos de análise
- Promover o reuso
- Avaliar a arquitetura

Projetando uma Arquitetura de Software

1. Identificar requisitos relevantes à arquitetura

- Requisitos funcionais – Casos de Uso:
  - Reserva self-service
  - Reserva
  - Pagamento
  - Login
  - Check-in
  - Check-out
  - Cadastro de hóspede
  - Registro de serviços

Visão de Casos de Uso



Projetando uma Arquitetura de Software

1. Identificar requisitos relevantes à arquitetura

- Principais requisitos de qualidade (não-funcionais):
  - O hóspede pode fazer sua reserva remotamente (WWW) e de forma segura.
  - Pico de 1000 acessos simultâneos em alta temporada.
  - Tempo de resposta para reserva pelo hóspede: 6 seg.
  - Tempo de resposta para reserva por funcionário: 2 seg.
  - A interface para o funcionário deve ser rica (gráfica, *drag-n-drop*,etc) e acessível pela intranet.
  - Disponibilidade do sistema de 99,9%.
  - Estimativa de crescimento de 20% ao ano.

32

Projetando uma Arquitetura de Software

1. Identificar requisitos relevantes à arquitetura

- Identificação dos casos de uso críticos
  - ❖**Reserva self-service**
  - ❖**Reserva**
  - ❖**Pagamento**
  - Login
  - Check-in
  - Check-out
  - Cadastro de hóspede
  - Registro de serviços

➢Casos de uso críticos são também conhecidos como arquiteturalmente significativos.

Projetando uma Arquitetura de Software

2. Classificar e priorizar requisitos relevantes à arquitetura

Requisito de Qualidade	Classificação
Reserva remota segura.	Segurança
Usuários identificados.	Segurança
Pico de 1000 acessos simultâneos	Performance
Confirmação de reserva por email.	Interoperabilidade
Tempo de resposta reserva hóspede 8 seg.	Performance
Tempo de resposta reserva funcionário 5 seg.	Performance
A interface para o funcionário deve ser rica.	Usabilidade
Disponibilidade do sistema de 99,9%.	Disponibilidade
Estimativa de crescimento de 20% ao ano.	Escalabilidade

34

Projetando uma Arquitetura de Software

2. Classificar e priorizar requisitos relevantes à arquitetura

- Priorização dos casos de uso arquiteturalmente significativos.

Ordem	Caso de Uso	Requisitos Suplementares
1	Reserva self-service	Segurança, Performance, Interoperabilidade, Disponibilidade
2	Reserva	Usabilidade
3	Pagamento	Interoperabilidade, Segurança

35

Projetando uma Arquitetura de Software

3. Identificar riscos e restrições

- Principais restrições identificadas:
  - O sistema de autorização de cartões de crédito (SACC) é provido por terceiros e as transações serão realizadas através de webservices.
  - O servidor de aplicações deve ser o Sun GlassFish.
  - O SGBD deve ser o Oracle 10G.
  - A aplicação web deve ser compatível com IE 6 e Firefox 2 ou versões superiores.
- Principais riscos identificados:
  - Apesar de ser compatível com J2EE, o servidor de aplicações não é conhecido pelos desenvolvedores.

36

Projetando uma Arquitetura de Software

4. Atacar maiores riscos técnicos

- Ações de mitigação a riscos e aos requisitos de qualidade mais severos.

Risco ou Requisito severo	Mitigação
Performance	Teste de stress sobre UC Reserva self-service
Disponibilidade	Teste de maturidade
Disponibilidade email	POC Email – Componentes assíncronos
Interoperabilidade	POC Integração com Sistema de Cartão de Crédito
Escalabilidade	POC Cluster
Avaliação servidor de aplicação	POC Avaliação, Capacitação da Equipe

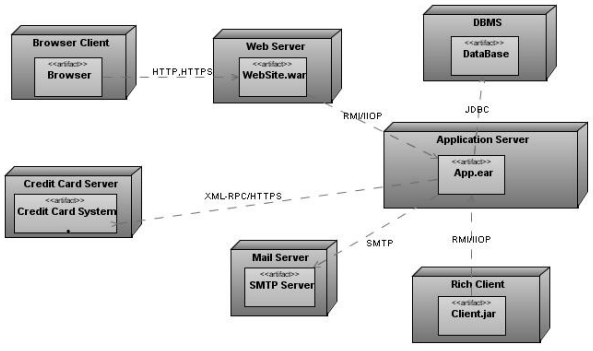
Projetando uma Arquitetura de Software

5. Desenvolver a visão geral da arquitetura

- Explorar e avaliar opções arquiteturais de alto-nível.
- Prover um entendimento da estrutura para os patrocinadores e demais stakeholders.
- Levar em consideração:
  - design da rede pré-existente
  - bancos de dados pré-existentes
  - ambiente web
  - configuração dos servidores
  - uso de padrões
- Produto típico: modelo de implantação preliminar

38

Visão de Implantação



39

Projetando uma Arquitetura de Software

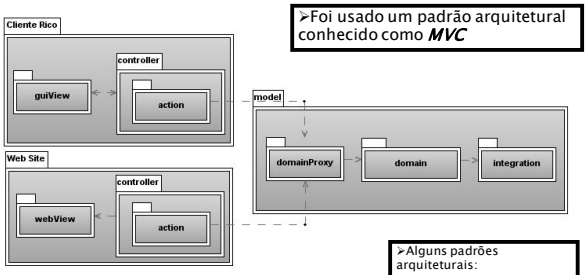
6. Definir organização lógica de alto nível

- Criar uma estrutura inicial para o modelo de design.
- Mostrar pacotes de desenho arquiteturalmente significativos.

➢Considerar o uso de padrões arquiteturais.
➢*Patterns of Enterprise Application Architecture* de Martin Fowler
➢*POSA4*

Visão Lógica

➢Foi usado um padrão arquitetural conhecido como ***MVC***



➢Alguns padrões arquiteturais:
- ➢Layers
- ➢Broker
- ➢Pipes and Filters
- ➢Microkernel

Projetando uma Arquitetura de Software

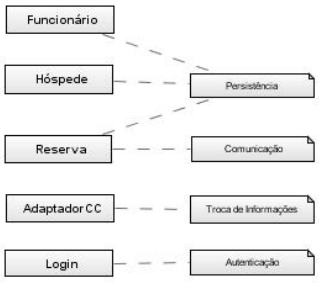
7. Identificar mecanismos de análise

- O que é necessário para "dar vida" aos componentes.
- Alguns mecanismos de análise:
  - Persistência
  - Roteamento de mensagens
  - Gerência de transações
  - Redundância
  - Troca de informações

➢Boas referências:
- ➢Livro *Analysis Patterns* de Martin Fowler
- ➢Artigo *Capturing Architectural Requirements* de Peter Eeles

Projetando uma Arquitetura de Software

7. Identificar mecanismos de análise



Projetando uma Arquitetura de Software

8. Promover o reuso

- Identificar ativos pré-existentes que possam colaborar para a solução.
- Realizar uma avaliação preliminar.

➢Redução de custos e riscos.
➢Não reinventar a roda!

44

Projetando uma Arquitetura de Software

9. Avaliar a arquitetura

- O objetivo é rever o resultado e analisar alternativas.
  - Avaliação do grau de atendimento dos requisitos de qualidade.
  - Utilização de checklists para validação.
  - Métodos clássicos: ATAM, CBAM, SAAM e ARID (SEI / Carnegie Mellon)

➢Ferramentas de análise estrutural e arquitetural podem auxiliar na avaliação arquitetura e conformidade do código.
  ➢Rational Software Architect
  ➢SA4J
  ➢Metrics (plugin do Eclipse)
  ➢SourceMonitor
  ➢JDepend

45

Projetando uma Arquitetura de Software

- Após a primeira análise arquitetural, ao fim da fase de **Iniciação**, temos a **Arquitetura Candidata.**

- E depois?
  - Refinamento da arquitetura na fase seguinte (se o projeto não foi abortado).

- E ao final da **Elaboração**
  - A arquitetura do software deve estar **Estável.**

46

Papel do Arquiteto de Software

"O arquiteto ideal deve ser uma pessoa de letras,
um matemático,
familiar com estudos históricos,
um diligente estudante de filosofia,
interessado em música,
não ignorante em medicina e proficiente em questões jurídicas,
familiar com a astronomia e os cálculos astronômicos."

– Vitruvius, 25 d.C.

47

Papel do Arquiteto de Software

- O arquiteto deve combinar as habilidades:
  - Experiência
  - Liderança
  - Comunicação
  - Orientado a objetivos
  - Pró-atividade
- Necessita abranger as capacidades de *designer* (ou projetista), mas:
  - tende a ser mais generalista que especialista;
  - deve tomar decisões técnicas mais abrangentes.
- Está em constante aprendizado.
- Arquiteto ou Time de Arquitetura?

48

Tao do Arquiteto de Software

O arquiteto observa o mundo.
Mas confia em sua visão interior.
Ele permite que as coisas venham e vão.
Seu coração é aberto como o céu.

O arquiteto não fala, age.
Quando o trabalho está pronto,
o time diz: "Incrível:
nós fizemos tudo sozinhos!"

Quando o arquiteto lidera, o time
dificilmente percebe que ele existe.
O segundo melhor é o arquiteto que é amado.
Depois, o que é temido.
O pior deles é aquele desprezado.

Lao-Tsu, revisited by Philippe Kruchten

49

Algumas Referências Extras

- dearchitectura.wordpress.com
- pangeanet.org
- www.sei.cmu.edu/architecture
- www.ibm.com/developerworks/architecture
- www.booch.com/architecture
- www.bredemeyer.com
- IBM Rational Unified Process
- OpenUP
- *Arquitetura Ágil* – Scott Ambler
- *The "4+1" View Model of Software Architecture* - Philippe Kruchten
- *Who Needs an Architect?* - Martin Fowler
- *Enterprise Architects Join the Team* - Rebecca J. Parsons
- *Software Architecture in Practice* – Clements, Bass, Kazman
- *An Introduction to Software Architecture* - Garlan, Shaw
- *Software Systems Architecture* – Rozanski, Woods

50

Conclusões

- Definição de arquitetura de software?
- Importância de arquitetura de software?
- Perspectiva perante processos de desenvolvimento?
- Modelo de Visualização 4+1?
- Ops! Alguma visão faltante?
- Habilidades do arquiteto?

51