Arquitetura de Software

Engenharia de Software I

Estagiária PAE: Lina María Garcés Rodríguez

Profa. Dra. Elisa Yumi Nakagawa

29-06-2015
São Carlos

Conteúdos

- Introdução à Arquitetura de Software
- Funções do Arquiteto de Software
- Processo de Desenvolvimento e a Arquitetura de Software
- Requisitos Arquiteturais
- Projeto da Arquitetura de Software
- Documentação da Arquitetura de Software
- Avaliação da Arquitetura de Software

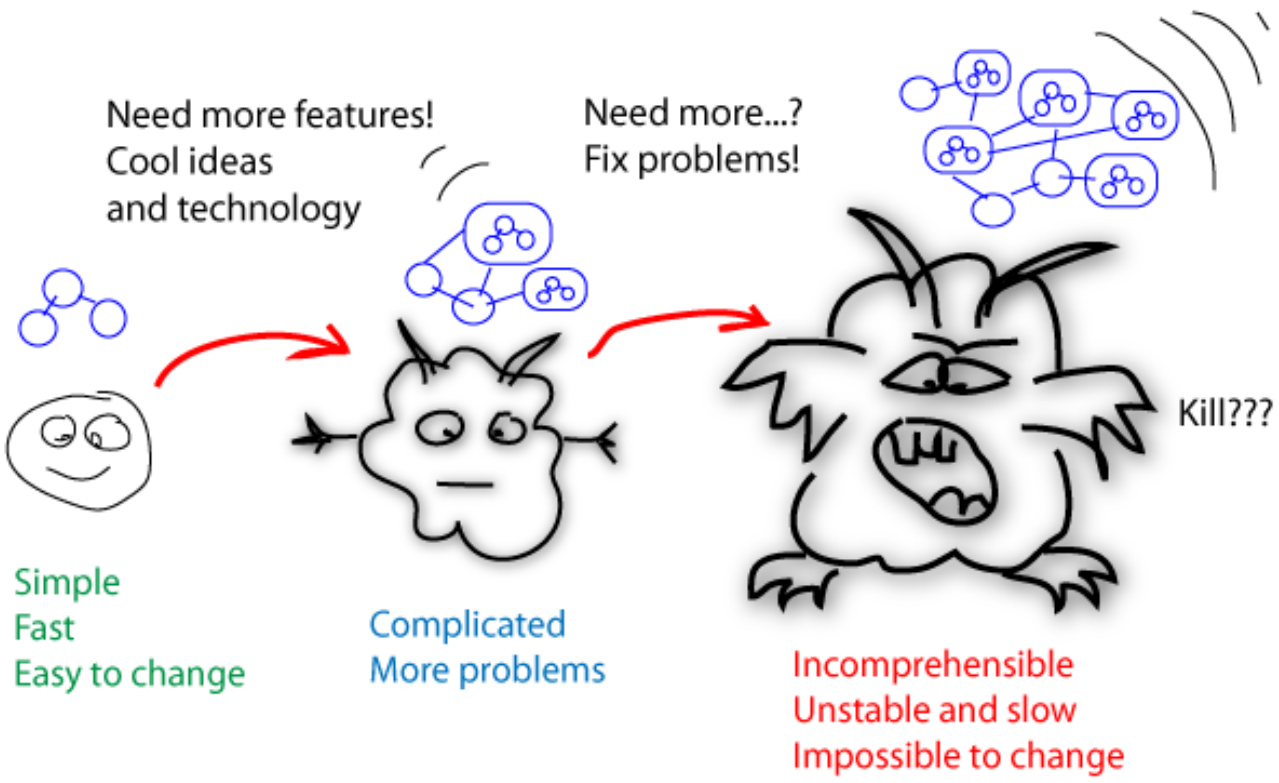
Conteúdos
Primeira parte

- **Introdução à Arquitetura de Software**
- **Funções do Arquiteto de Software**
- **Processo de Desenvolvimento e a Arquitetura de Software**
- **Requisitos Arquiteturais**
- Projeto da Arquitetura de Software
- Documentação da Arquitetura de Software
- Avaliação da Arquitetura de Software

Introdução à Arquitetura de Software

- Arquitetura de software tem emergido como uma sub-disciplina importante da engenharia de software (Clements et al. 2010).

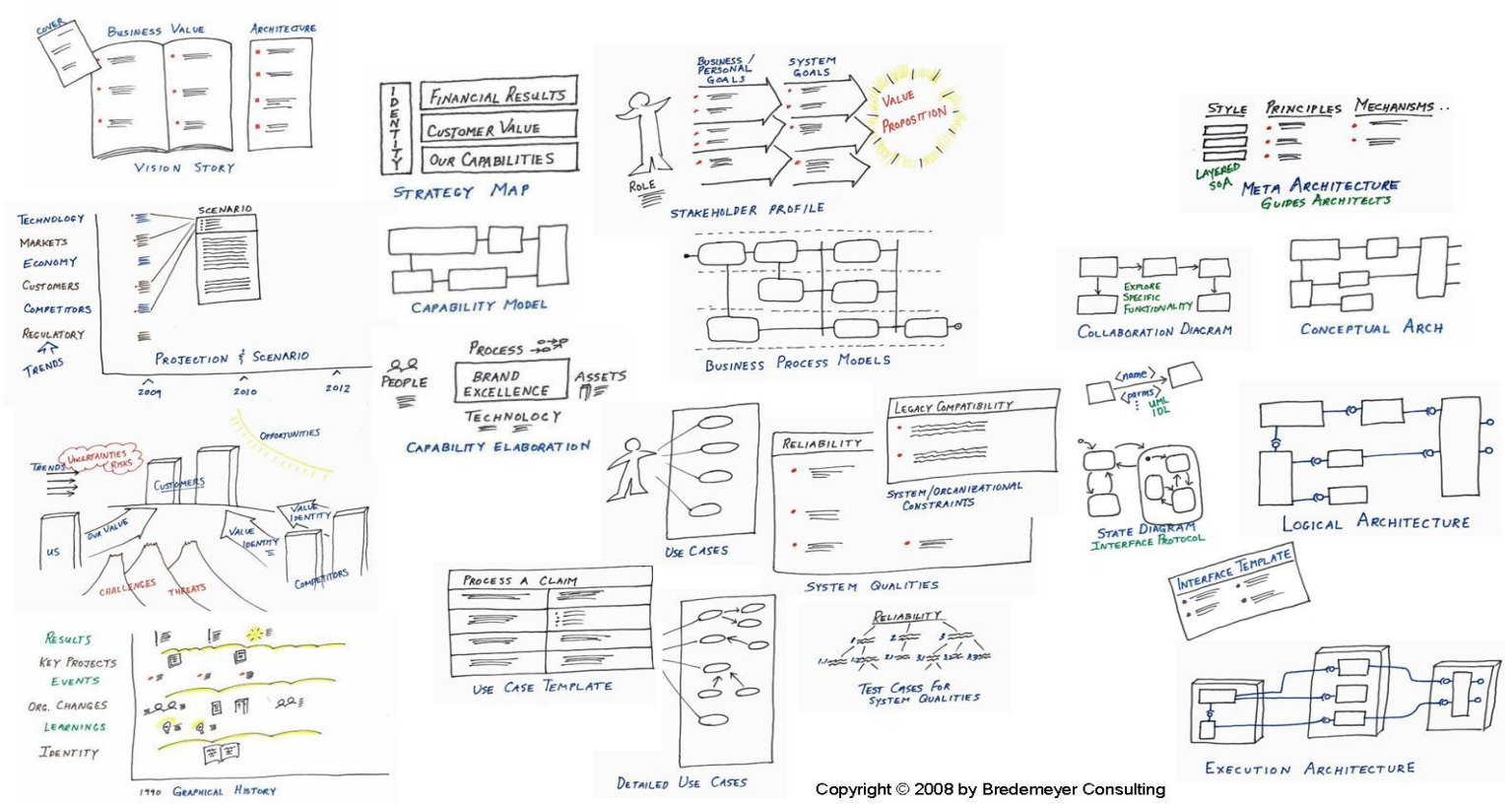
- De forma geral, arquitetura é uma divisão prudente do “todo” em partes, com relações específicas entre as partes (Clements et al. 2010).



Definição

- **Arquitetura de software:** (SEI 2005; Garlan et al. 2000)

  Estrutura de componentes de um programa/sistema, os relacionamentos entre esses componentes, os princípios e diretrizes que governam os projetos e a evolução dos softwares.



Diferença entre Arquitetura de Software e Design de Software

- A arquitetura é *projeto*, mas nem todo projeto pode ser considerado arquitetura.

- *Ou seja, muitas decisões de projeto não são consideradas na arquitetura, mas são deixadas para que os programadores ou outros projetistas façam.*

- *A arquitetura estabelece restrições nas atividades futuras, e essas atividades devem produzir artefatos (código ou projeto mais detalhados) conforme à arquitetura.* (Clements et al. 2010).

Importância da Arquitetura de Software

(Bass et al. 2003).

Comunicação entre stakeholders

A arquitetura de software representa uma abstração comum de um sistema que a maioria (ou todos) os stakeholders podem utilizar como base para o entendimento, negociação, consenso, e comunicação.

Abstração transferível do sistema

A arquitetura de software constitui um modelo relativamente pequeno e compreensível, de como o sistema é estruturado e como seus elementos trabalham juntos. Esse modelo é transferível no sistema.

Esse modelo pode ser utilizado em outros sistemas que tenham atributos de qualidade e requisitos funcionais similares.

Funções do Arquiteto de Software

As funções e responsabilidades do arquiteto variam dependendo do tipo de sistema. As mais comuns são (Garland and Anthony 2003).:

Estabelecimento dos requisitos:

O arquiteto tipicamente é o responsável pelo entendimento e gestão dos requisitos não funcionais do sistema.

O arquiteto pode trabalhar diretamente com os stakeholders.

Avaliação do risco técnico do sistema:

O arquiteto deve fornecer um plano de gestão do risco.

Deve poder avaliar o impacto que uma mudança nos requisitos terá no sistema, e avaliar o risco dessas mudanças.

Análise do domínio do sistema:

O arquiteto deve ser capaz de dividir os problemas em partes e estruturar soluções que possam abordar as necessidades da organização.

Revisor dos entregáveis do sistema.

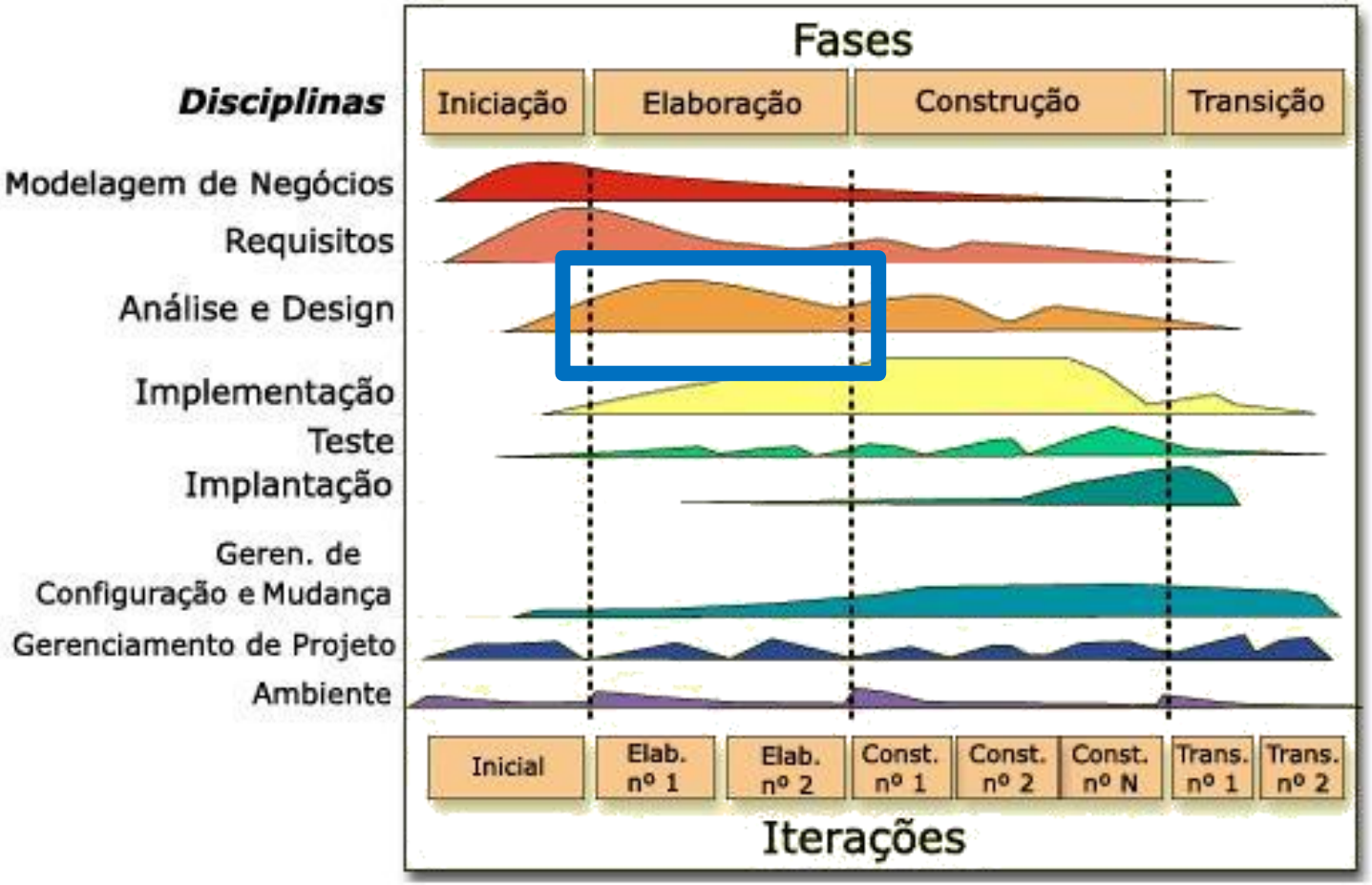
Mentor de projetistas e desenvolvedores

Desenvolvedor (em projetos pequenos)

Líder da equipe

Processo de Desenvolvimento e a Arquitetura de Software

Projeto da Arquitetura de Software



IBM Rational Unified Process. Fonte: Wikipedia

Processo de Projeto da Arquitetura de Software

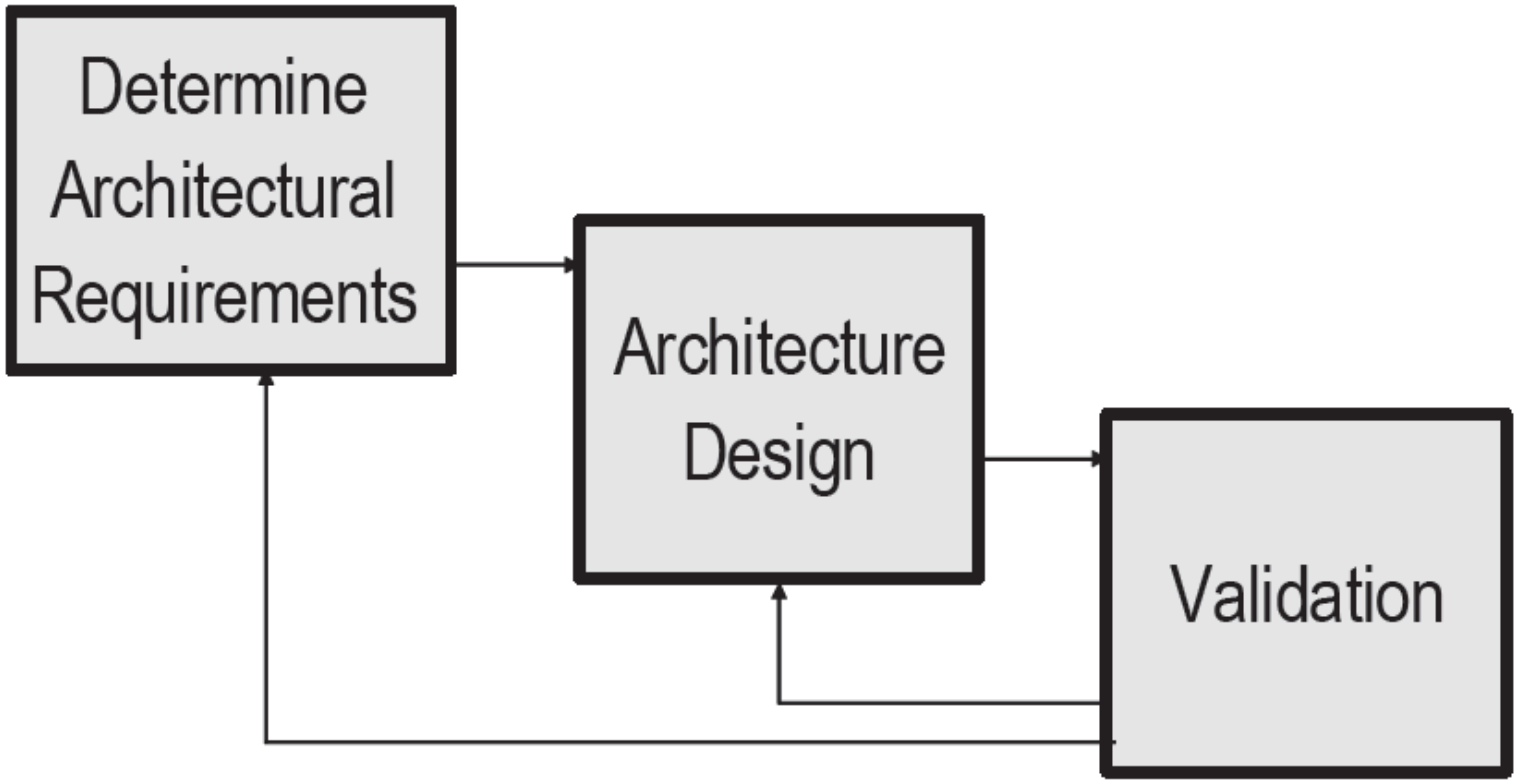


Fig. 36. A three step architecture design process

(Gorton 2006).

Processo de Projeto da Arquitetura de Software

1. Requisitos Arquiteturais

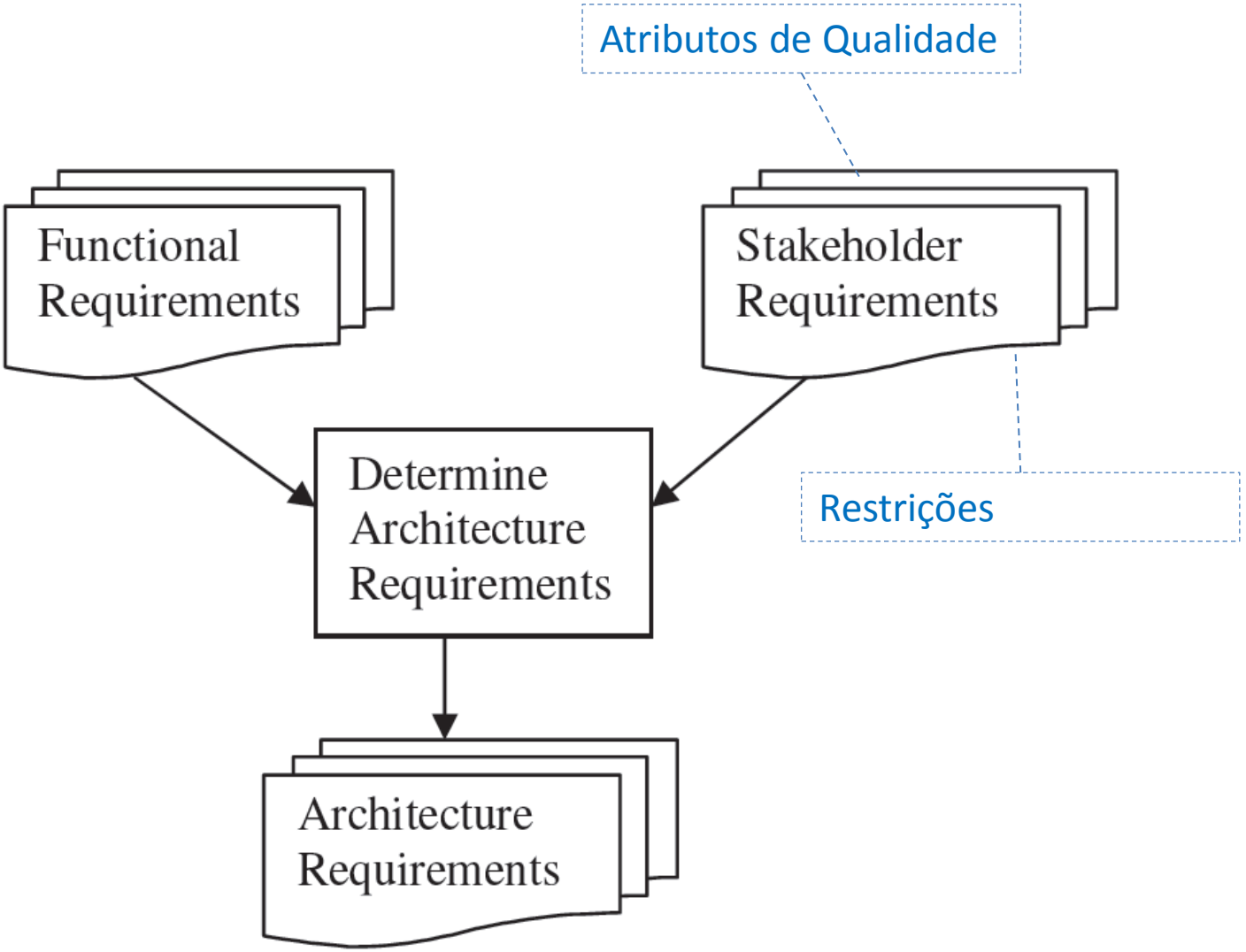


Fig. 37. Inputs and outputs for determining architecture requirements (Gorton 2006).

Exemplos de Requisitos Arquiteturais

A typical architecture requirement concerning **reliability of communications** is:

"Communications between components **must be** guaranteed to succeed with no message loss"

Some architecture requirements are really **constraints**, for example:

"The system **must use** the existing IIS-based web server and use Active Server Page to process web requests"

Qualidade de Software

Qualidade de software :

Grau de satisfação que um produto software alcança quando é utilizado em condições específicas (ISO/IEC 25000:2005)

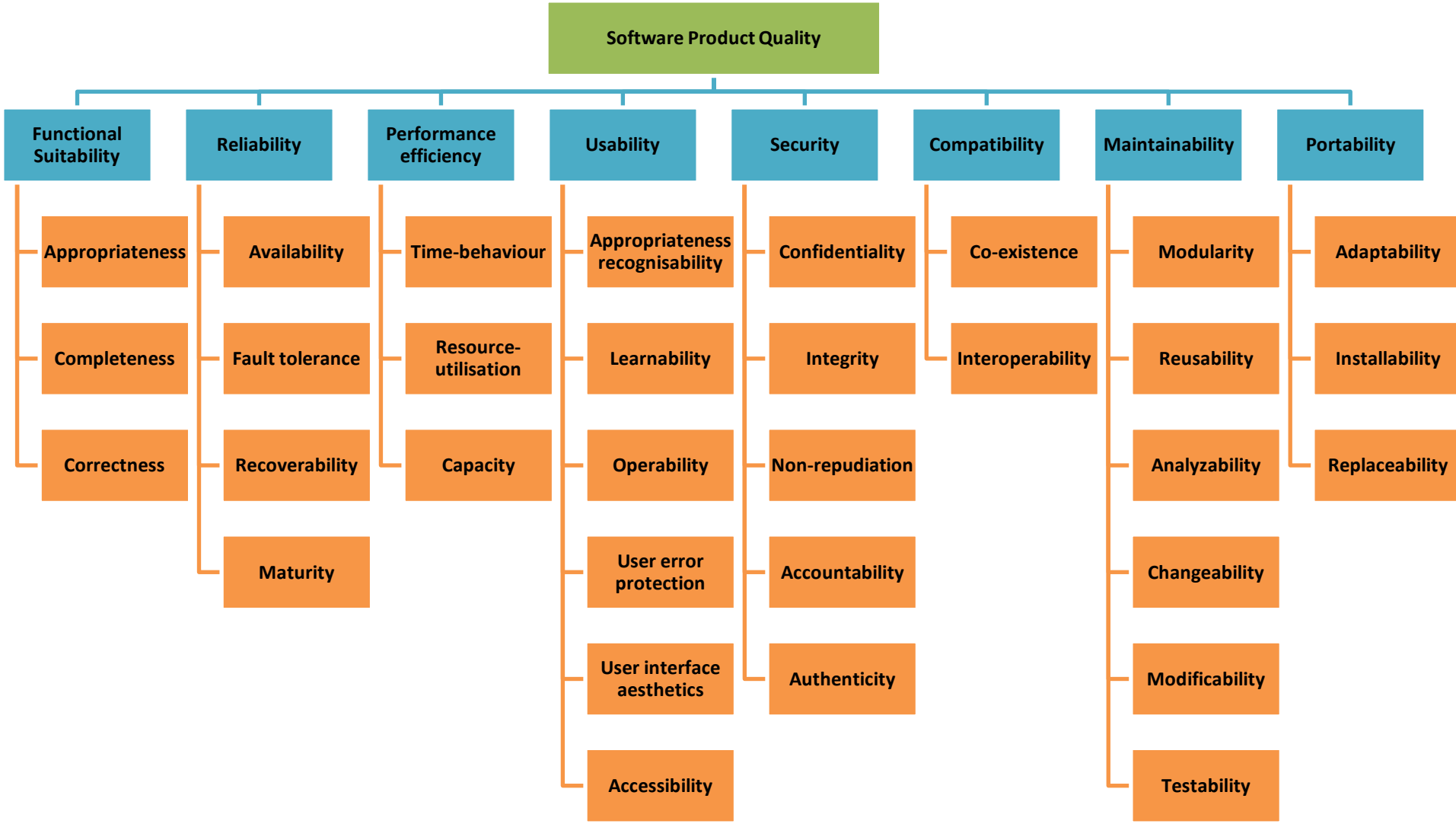
Atributo de Qualidade:

Uma caraterística de software que especifica o grau que deve possuir um atributo que afeta a qualidade do software (ISO 2001).
Exemplos: Usabilidade, confiabilidade, performance, etc.

Modelo de Qualidade:

Conjunto de caraterísticas, e relacionamento entre elas, que fornecem um marco de referência para especificar requisitos de qualidade e avaliar a qualidade do software (ISO/IEC 25000:2005).

Modelo de Qualidade ISO/IEC 25010



Modelo de Qualidade ISO/IEC 25010

Atributo de Qualidade	Definição	Exemplo de Requisito Arquitetural
Functional Suitability	degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions	A aplicação deve permitir o pagamento por cartão de crédito de forma segura.
Reliability	degree to which a system, product or component performs specified functions under specified conditions for a specified period of time	Perda de pacotes de dados menor que o 0.01%
Performance efficiency	performance relative to the amount of resources used under stated conditions	Tempo de processamento menor de 0.01 segundo.
Usability	degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use	Fornecer interfaces para usuários com deficiências e que sejam fáceis de utilizar
Security	degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization	Utilizar senhas criptografadas.
Compatibility	degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment	Que a aplicação possa compartilhar informações com redes sociais facebook, twitter, instagram.
Maintainability	degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers	O tempo de atualização deve ser menor de 2 horas.
Portability	degree of effectiveness and efficiency with which a system, product or component can be transferred fromone hardware, software or other operational or usage environment to another	Que a aplicação possa ser utilizada em plataformas windows, mac, linux, android, etc.

Os atributos de qualidade **não são ortogonais.**

Eles interagem de formas sutis, ou seja, um projeto que está em conformidade com um atributo de qualidade pode ter um efeito prejudicial sobre outro requisito.

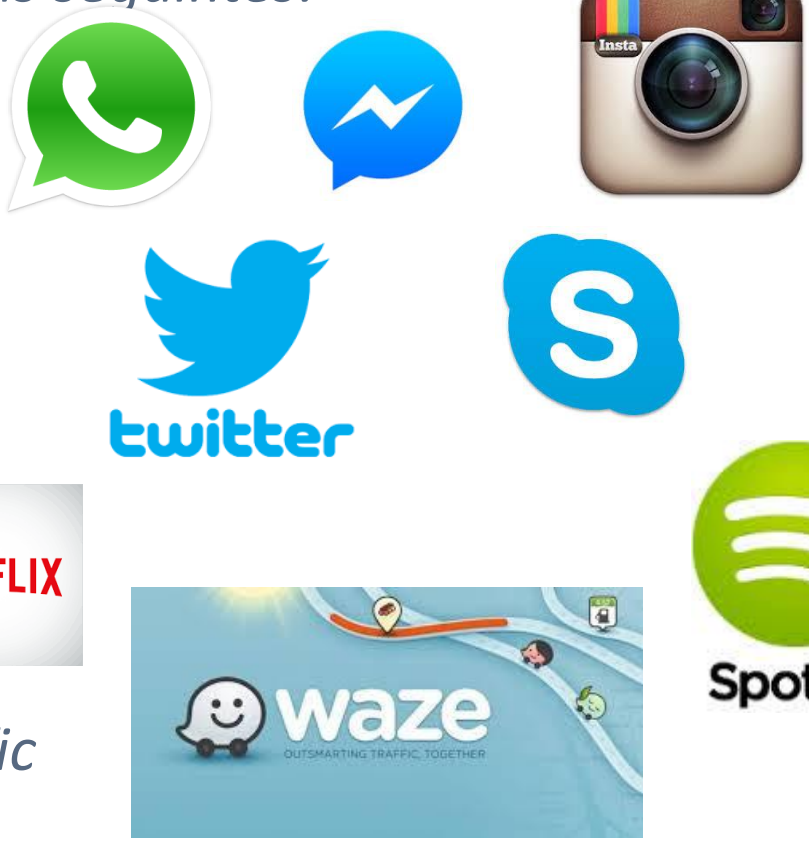
Exemplo: Alto performance vs Portabilidade.

Simplesmente **não é possível** satisfazer completamente todos os atributos de qualidade desejados em um sistema software.

Exercício de aula

Selecione uma aplicação dentre as seguintes:

- *Whatsapp*
- *Facebook messenger*
- *Instagram*
- *Youtube*
- *Twitter*
- *Skype*
- *Spotify*
- *Dropbox*
- *Netflix*
- *Waze social GPS Maps & Traffic*



Pense em pelo menos cinco requisitos que o arquiteto da aplicação considerou ao projetar sua arquitetura.

Conteúdos
Segunda parte

- Introdução à Arquitetura de Software
- Funções do Arquiteto de Software
- Processo de Desenvolvimento e a Arquitetura de Software
- Requisitos Arquiteturais
- **Projeto da Arquitetura de Software**
- **Documentação da Arquitetura de Software**
- **Avaliação da Arquitetura de Software**

Processo de design da Arquitetura de Software

2. Design

Design da Arquitetura = Alcançar as qualidades

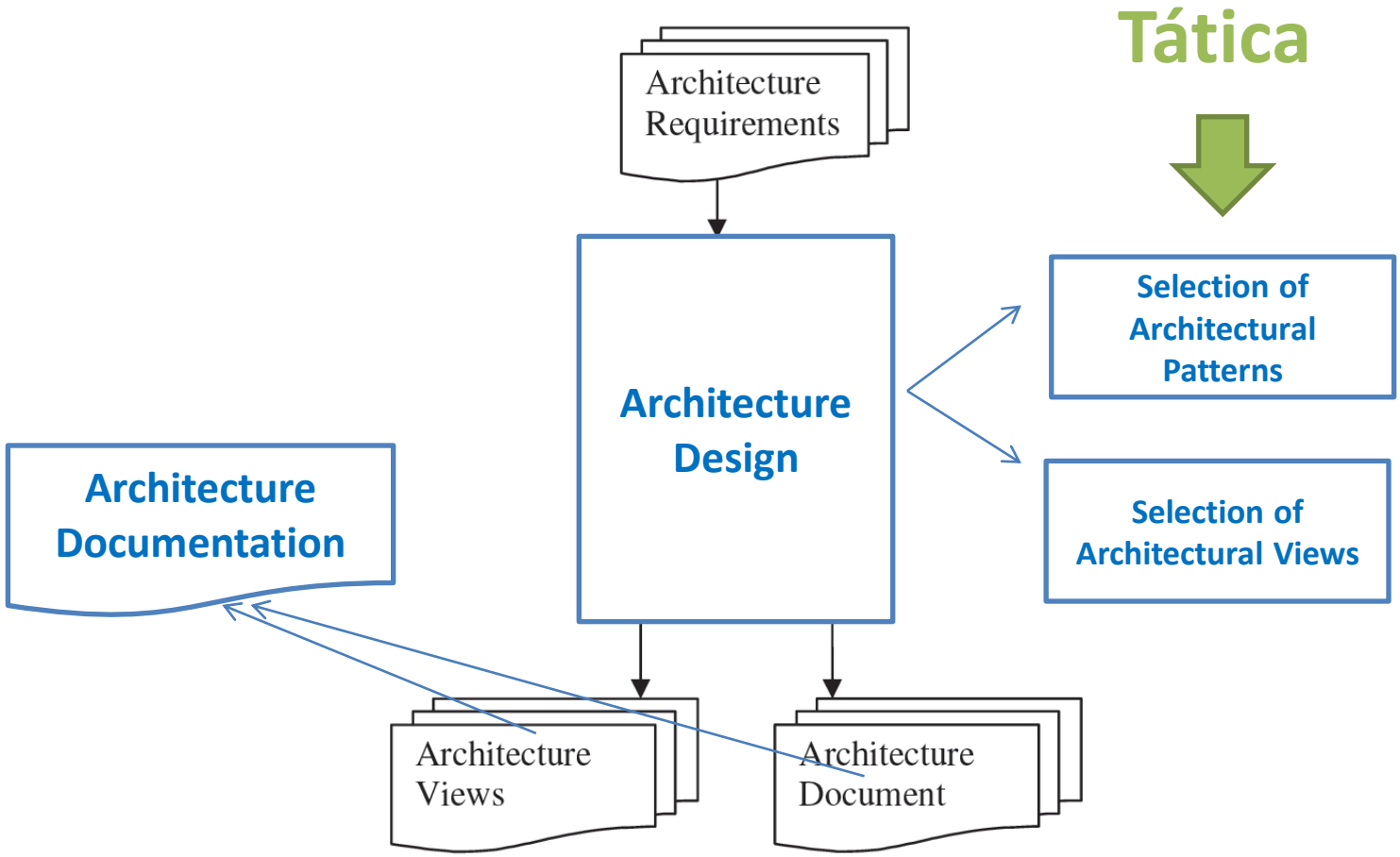


Fig. 38. Inputs and outputs of architecture design

Padrões Arquiteturais

"Most of the IT applications I've worked on in the last ten years are based around a small number of well understood, proven architectures. There's a good reason for this – they work".

Ian Gorton, 2006.

Aproveitar soluções conhecidas **minimiza os riscos** que uma aplicação possa falhar devido a uma arquitetura inadequada.

Assim a etapa inicial envolve a seleção de um padrão arquitetural que ajude a **satisfazer os requisitos essenciais**.

Para aplicações pequenas, somente um padrão pode ser suficiente. Para aplicações mais complexas, o projeto irá incorporar um ou mais padrões conhecidos, sendo que o arquiteto especificando como esses padrões se integram para formar a arquitetura geral.

Não existe nenhuma fórmula mágica para projetar uma arquitetura, porém, um requisito é entender como cada padrão aborda os atributos de qualidade.

Padrões Arquiteturais

Classificação segundo Bass et al. (2010) :

Padrões Módulo	Padrões Componente & Conectores	Padrões Alocação
• Mostra a arquitetura em termos de módulos;	• Mostra a arquitetura em termos de componentes e conectores; • Possibilitam estruturar o software como um conjunto de elementos com comportamentos e interações em tempo de execução.	• Mostra a arquitetura como uma combinação de elementos de software e elementos que não são de software (e.g., servers, networks, etc)

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão fluxo de dados

- Os componentes atuam como transformadores e os conetores transferem dados desde a saída de um componente para a entrada de outro.
- Esses padrões são comuns em domínios onde o processamento de fluxos de dados ocorre, e onde toda a computação pode dividir-se em um conjunto de passos transformadores.

Padrão call-return

- Padrões onde os componentes interagem mediante a invocação síncrona de capacidades fornecidas por outros componentes.
- Um componente que invoca um serviço, é pausado até que o serviço seja concluído.
- Os conectores são responsáveis de transferir a solicitação do serviço e de retornar os resultados.

Padrão baseado em eventos

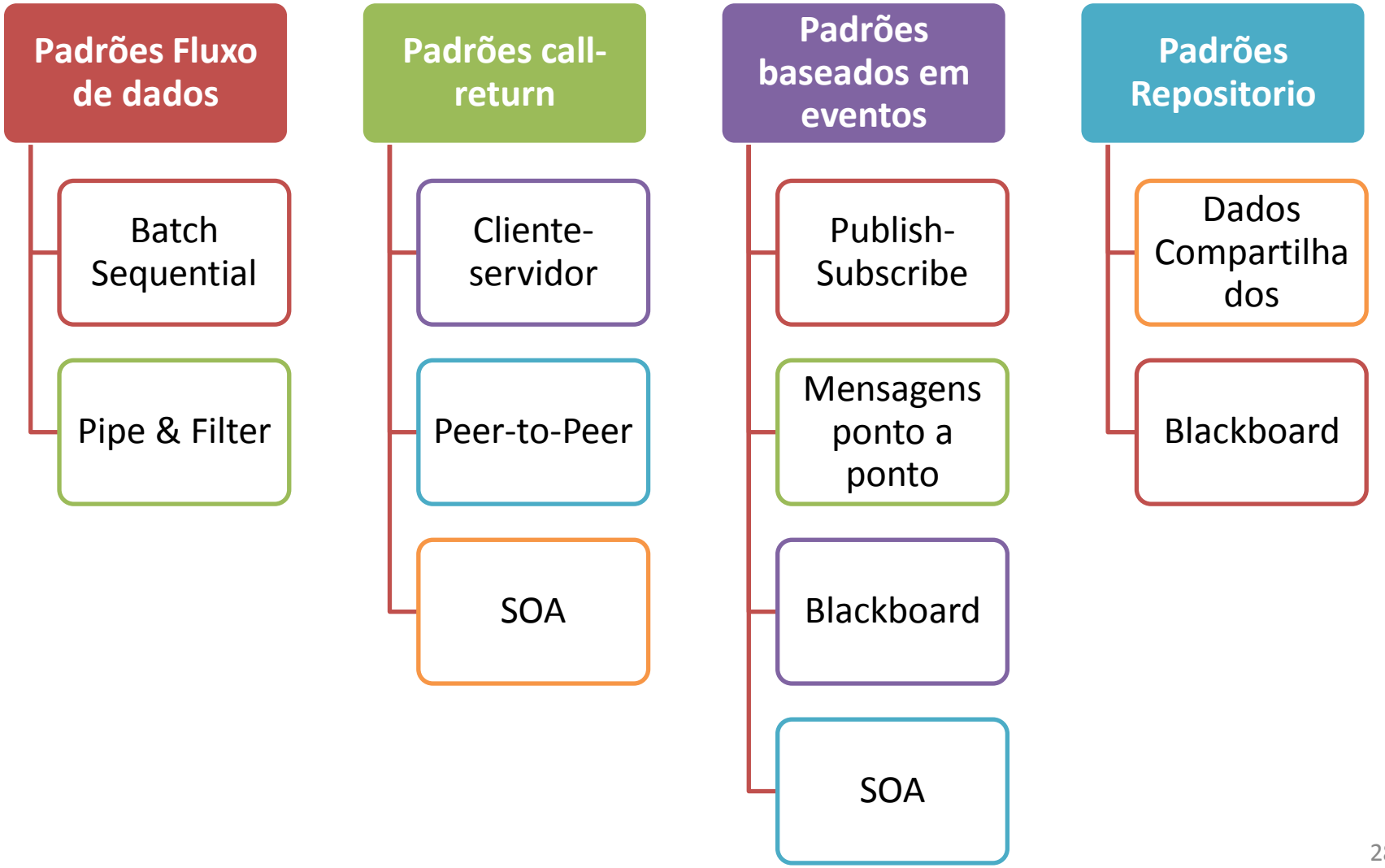
- Padrões onde os componentes interagem mediante eventos ou mensagens asíncronos.
- Os sistemas que utilizas esses padrões são organizados como uniões com acoplamento fraco de componentes que acionam o comportamento em outros componentes através de eventos.

Padrão Repositorio

- Padrões onde os componentes interagem mediante grandes coleções de dados compartilhados persistentes.
- Em muitos casos, o acesso ao repositório é mediado pelo DBMS que fornece uma interface call-return para a recuperação e gestão de dados.

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)



Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão: Cliente - Servidor

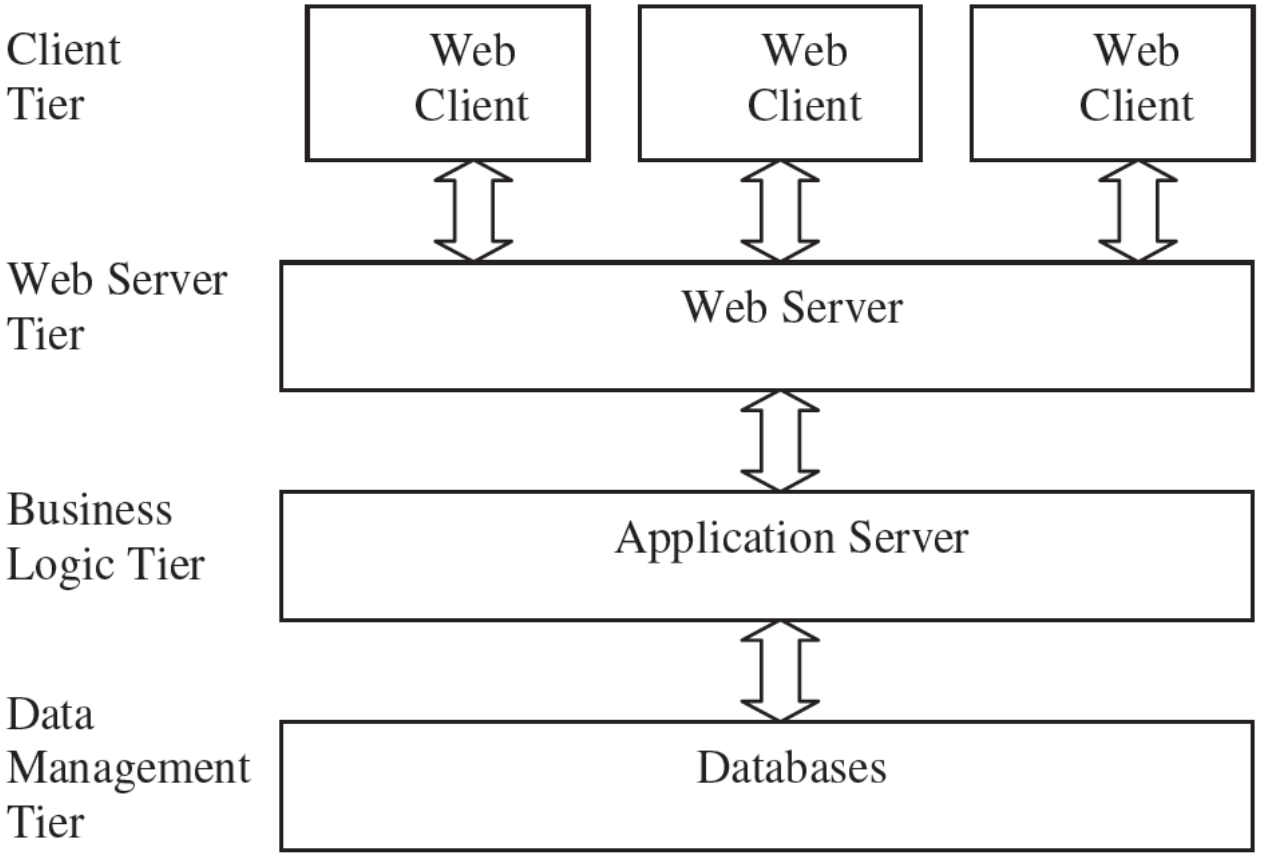


Fig. 39. N-tier client-server example

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão: Cliente - Servidor

Relations	The *attachment* relation associates client service-request ports with the request role of the connector and server service-reply ports with the reply role of the connector.
Computational Model	Clients initiate interactions, invoking services as needed from servers and waiting for the results of those requests.
Constraints	• Clients are connected to servers through request/reply connectors. • Server components can be clients to other servers. • Specializations may impose restrictions: – Numbers of attachments to a given port – Allowed relations among servers • Components may be arranged in tiers.
What It's For	• Promoting modifiability and reuse by factoring out common services • Improving scalability and availability in case server replication is in place • Analyzing dependability, security, and throughput

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão: Cliente - Servidor

Propriedades do Padrão C-S Multi-nível

Separação de interesses: A lógica de apresentação, negócio, e gestão dos dados se divide claramente em diferentes níveis.

Comunicação síncrona: A comunicação entre os níveis é síncrona, os pedidos emanam em uma direção, e cada nível espera por uma resposta do outro nível antes de prosseguir.

Deployment flexível: Não há restrições sobre a implantação de uma aplicação multi-nível. Todos os níveis podem ser executados na mesma máquina ou em máquinas diferentes.

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão: Cliente - Servidor

Quando utilizar o padrão C-S Multi-nível?

Quando uma aplicação tem de suportar um grande número de clientes e solicitações simultâneas, e cada pedido leva um intervalo relativamente curto (alguns milissegundos ou segundos) para processamento.

Padrões Arquiteturais tipo Componente & Conector

(Bass et al., 2010)

Padrão: Cliente - Servidor

Atributos de Qualidade abordados:

Disponibilidade	Tolerância a falhos	Modificabilidade	Desempenho	Escalabilidade
• Servers em cada nível podem ser replicados, por se um deles falha outros estejam disponíveis.	• Implementação transparente de controle de falhas. • As solicitações do cliente são direcionadas às replicas.	• Separação de preocupações possibilita as mudanças, sem precisar mudar outros níveis. • Encapsulação das preocupações.	• Alto desempenho • Solicitações dos clientes são processadas pelos servers com menor carga de trabalho. • Um server pode atender múltiplas solicitações.	• Servers podem ser replicados • Múltiplas instâncias do server são executadas no mesmo ou diferente server. • Possível gargalho: Administração de dados (DBMS)

Processo de design da Arquitetura de Software

3. Documentação da Arquitetura

Muitas vezes, arquiteturas de software são criadas e não são documentadas (e conseqüentemente, comunicadas) de forma efetiva, isto é, desenvolvedores e outros com interesse no sistema (*stakeholders*) não têm acesso a uma representação adequada da arquitetura.

Assim, a documentação de arquitetura de software torna-se o artefato principal em todas as fases do desenvolvimento em que a arquitetura é usada.

"*Software architecture documentation speaks for the architect, today, tomorrow and 20 years from now.* " (SEI)

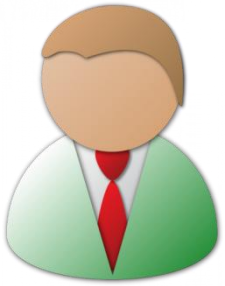
Requisitos arquiteturais

Seleção de padrões arquiteturais

Desenvolvedor

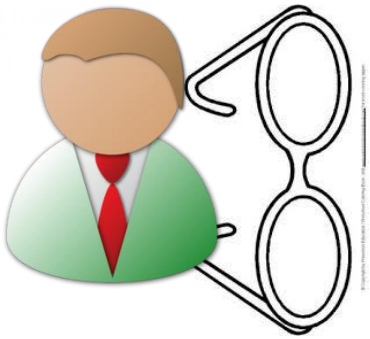
Cliente

Analista



Manager da equipe

Usuários finais



Visões arquiteturais

"Uma visão é uma representação de um conjunto de elementos e relações do sistema que lhes estão associados".

As visões importantes dependem da finalidade do arquiteto

Visões arquiteturais

Documentar uma arquitetura é uma questão de **documentar as visões** relevantes e, em seguida, a adição de documentação que se aplica a mais de uma visão.

Visões arquiteturais

Quais visões usar?

4+1 Views

- Visão de módulos
- Visão em tempo de execução
- Visão de implantação
- Visão de implementacão
- Visão de dados

Views & Beyond

- Visão de módulos
- Visão de componentes e conectores
- Visão de alocação

https://wiki.sei.cmu.edu/sad/index.php/The_Adventure_Builder_SAD

Visões arquiteturais

OBS: Nem todas as visões são relevantes para todos os sistemas.

Quais são as visões relevantes?

Isso depende dos objetivos do arquiteto!

As visões que você deve documentar dependem do uso que se espera para a documentação. Diferentes visões destacam diferentes elementos do sistema e/ou relações.

Exemplo:

Uma visão com o padrão de camadas esta orientado a descrever a portabilidade.

Uma visão de deployment tem como objetivo oferecer informações sobre o desempenho e confiabilidade do sistema.

Visões arquiteturais - Notações

Notações para documentar as visões diferem consideravelmente em seu grau de formalidade:

Notações informais: Visões são representadas (frequentemente com gráficos) utilizando ferramentas como power point, paint, etc.
As semântica da descrição são caraterizados em linguagem natural e **não** pode ser formalmente analisado.

Notações semi-formais: Visões são representadas com uma notação padronizada que prescreve elementos gráficos e regras de construção, mas não fornece um tratamento semântico completo do significado desses elementos.
Análise rudimentar pode ser aplicado para determinar se uma descrição satisfaz propriedades sintácticas. UML é uma notação semi-formal neste sentido.

Notações formais: Visões são representadas com uma notação que tem uma semântica precisa (usualmente baseada em matemática). É possível uma análise formal.
Existem variedades de notações formais, geralmente conhecidas como ADLs (Architectural Description Languages).

Visões arquiteturais - Notações

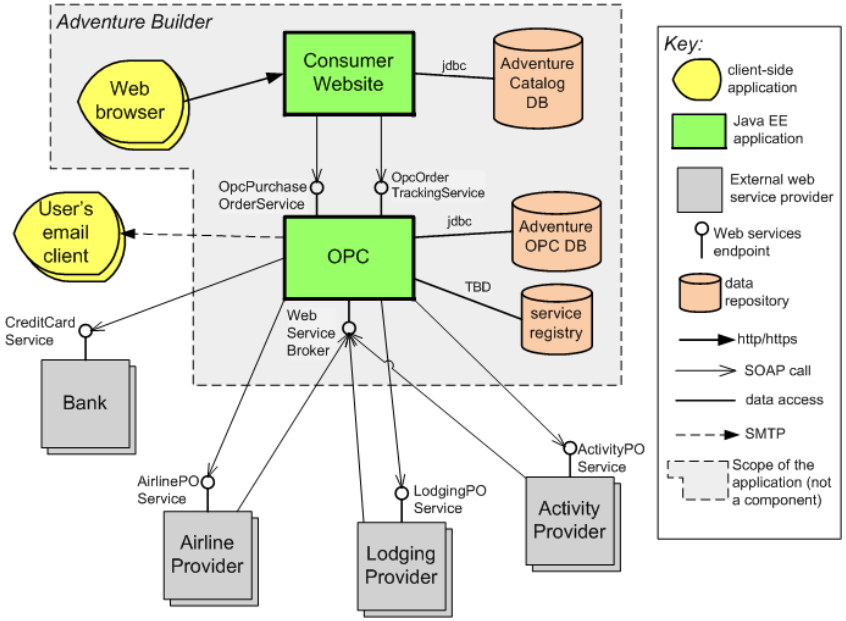
Determinar qual forma de notação utilizar envolve a tomar compromissos (trade-offs).

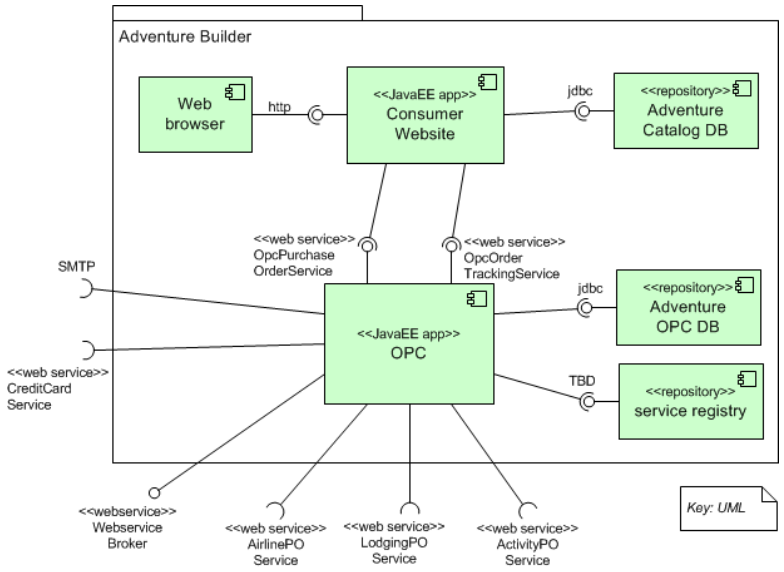
Tipicamente as notações mais formais precisam mais tempo e um esforço para criá-las, mas elas retribuem o esforço na redução da ambiguidade e possibilitam uma melhor análise.

Inversamente, notações mais informais são fáceis de criar, mas fornecem menos garantias.

Visões arquiteturais - Notações

Exemplo usando notação informal e UML

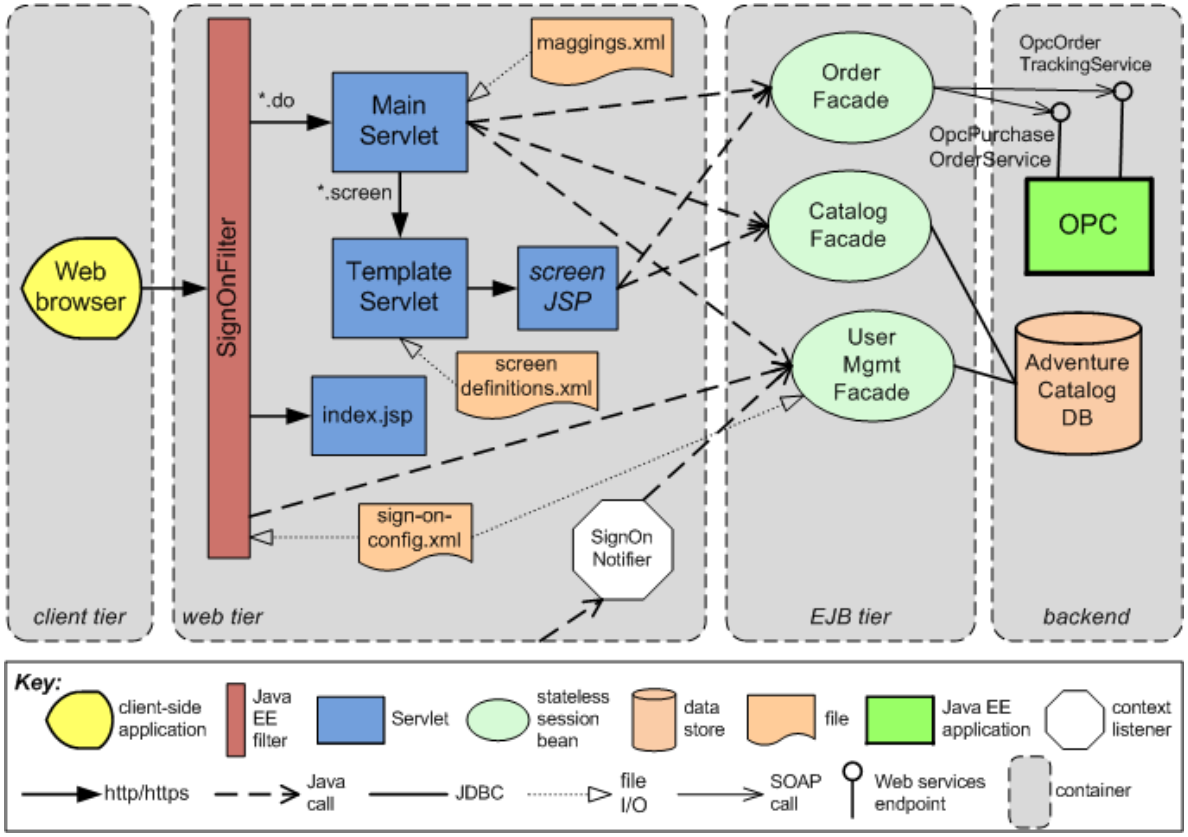




Exercício de Aula

Usando notação informal, estabeleça uma visão geral da arquitetura da aplicação selecionada na última aula. A visão deve ser feita visando que um usuário final quaisquer, possa entender o funcionamento da aplicação.

Fornecer uma legenda para os símbolos utilizados no diagrama, por exemplo:



Processo de design da Arquitetura de Software

4. Avaliação da Arquitetura

Evaluando a Arquitetura de Software

Architectures are not inherently good or bad...
... they are only well-suited or not with respect to a particular set of goals.

Architecture Evaluation
Checks
Architectural-significant decisions
Against
Architectural-significant requirements

Tomado da palestra do professor Paris Avgeriou no ICMC-USP 3/10/2014

Tipos de Avaliação

- **Quantitative:** How much ...?
  - Estimation
  - Analytical or simulation models
  - Measurements on feasibility prototypes or products

- **Qualitative:** What if ...?
  - Questioning techniques: questionnaires & checklists
  - Based on scenarios: SAAM, ATAM, CBAM, ...
  - Prototyping (proof-of-concept)

Evaluation mostly uses scenarios to verify QAs

Tomado da palestra do professor Paris Avgeriou no ICMC-USP 3/10/2014

Métodos baseados em cénarios

ATAM (Architecture Trade-off Analysis Method)
- All quality attributes & tradeoffs
- Design decisions & alternative design options

SAAM (Software Architecture Analysis Method)
- Modifiability and areas of potential high complexity
- Complete architecture qualitatively

CBAM (Cost Benefit Analysis Method)
- Costs, benefits, schedule implications of architectural decisions

ARID (Active Reviews for Intermediate Design)
- Preliminary evaluation for partial architecture
- Engineering SHs buy-in

There are more methods in the literature...

Tomado da palestra do professor Paris Avgeriou no ICMC-USP 3/10/2014

Referências

Ian Gorton. 2006. Essential Software Architecture. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Jeff Garland and Richard Anthony. 2003. Large-Scale Software Architecture: A Practical Guide Using UML. John Wiley & Sons, Inc., New York, NY, USA.

Len Bass, Paul Clements, and Rick Kazman. 2003. Software Architecture in Practice (2 ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Len Bass, David Garlan, Felix Bachmann, James Ivers, Judith Stafford, Paul Clements, and Paulo Merson. 2010. Documenting Software Architectures: Views and Beyond (2nd ed.). Addison-Wesley Professional.

SEI *(Software Engineering Institute)*
http://www.sei.cmu.edu/

ISO/IEC 25010:2011
http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733