FullCycle

Fundamentos de Arquitetura de Software

FullCycle

Tipos de arquitetura

- Software
- Solução
- Tecnológica
- Corporativa

FullCycle

Arquitetura de Software

- É uma “disciplina" da engenharia de software
- Diretamente ligada ao processo de desenvolvimento de software
- Afeta diretamente na estrutura organizacional da empresa
  - Formação dos times
  - Estrutura dos componentes do software
  - "Organizações que desenvolvem sistemas de software tendem a produzir sistemas que são cópia das estruturas de comunicação dessas empresas. (Melvin Conway)"

FullCycle

Arquitetura de Software

É a relação entre os objetivos de negócio e suas restrições com os componentes a serem criados e suas responsabilidades visando sua evolução do software.

FullCycle

Arquitetura de Software

“É a organização fundamental de um sistema e seus componentes, suas relações, seu ambiente, bem como os princípios que guiam seu design e evolução.” (IEEE Standard 1471)

FullCycle

Arquitetura de Software

O processo de arquitetar um software estabelece que o que está sendo desenvolvido faça parte de um conjunto maior.

FullCycle

Papel do Arquiteto(a) de Software

- Transformar requisitos de negócio em padrões arquiteturais
- Orquestrar pessoas desenvolvedores e experts de domínio
- Entender de forma profunda conceitos e modelos arquiteturais
- Auxilia na tomada de decisão nos momentos de crise
- Reforça boas práticas de desenvolvimento
- Code reviews

FullCycle

Papel do Arquiteto(a) de Software

Apesar de nem todas as organizações possuírem o cargo de arquiteto de software, normalmente profissionais mais experientes como desenvolvedores seniors e tech leads acabam realizando esse papel baseado em suas experiencias anteriores.

FullCycle

Papel do Arquiteto(a) de Software

Há empresas que apesar de não possuírem o cargo de arquiteto(a) de software, possuem um departamento de arquitetura que auxilia os diversos times da organização com questões arquiteturais.

FullCycle

Por que aprender arquitetura de software?

- Poder navegar da visão macro para a visão micro de um ou mais softwares
- Entender quais são as diversas opções que temos para desenvolver a mesma coisa e escolher a melhor solução para determinado contexto
- Pensar a longo prazo no projeto e sua sustentabilidade
- Tomar decisões de forma mais fria e calculada evitando assim ser influenciado(a) por “hypes” de mercado
- Mergulho em padrões de projeto e de desenvolvimento e suas boas práticas
- Ter mais clareza do impacto que o software possui na organização como um todo
- Tomar decisões com mais confiança

FullCycle

Arquitetura vs Design

- Arquitetura: Escopo global ou alto nível
- Design: Escopo local

FullCycle

Arquitetura ou Design

"Atividades relacionadas a arquitetura de software são sempre de design. Entretanto, nem todas atividade de design são sobre arquitetura. O objetivo primário da arquitetura de software é garantir que os atributos de qualidade, restrições de alto nível e os objetivos do negócio, sejam atendidos pelo sistema. Qualquer decisão de design que não tenha relação com este objetivo não é arquitetural. Todas as decisões de design para um componente que não sejam "visíveis" fora dele, geralmente, também não são.”

Fonte: Elemar Jr. (https://www.eximiaco.tech/pt/2020/01/08/quais-sao-as-diferencas-entre-arquitetura-e-design-de-software/"

Arquitetura é uma abstração

Thus an architecture is foremost an abstraction of a system that selects certain details and suppresses others. In all modern systems, elements interact with each other by means of interfaces that partition details about an element into public and private parts. Architecture is concerned with the public side of this division; private details of elements—details having to do solely with internal implementation—are not architectural.

Bass, Len; Clements, Paul; Kazman, Rick. Software Architecture in Practice (SEI Series in Software Engineering) (p. 3). Pearson Education. Kindle Edition.

FullCycle

Arquitetura é um conjunto de estruturas de Software

FullCycle

A structure is architectural if it supports reasoning about the system and the system's properties. The reasoning should be about an attribute of the system that is important to some stakeholder(s).

Bass, Len; Clements, Paul; Kazman, Rick. Software Architecture in Practice (SEI Series in Software Engineering) (p. 2). Pearson Education. Kindle Edition.

FullCycle

Pontos importantes

- Todo software possui uma arquitetura
- Nem todas as arquiteturas são boas arquiteturas
- Arquitetura inclui comportamentos

FullCycle

Estruturas

Componente-conector

Módulos

Alocação

FullCycle

Estrutura componente-conector

- Interação entre elementos para garantir o funcionamento do sistema
- Componentes -> Comportamento
  - Services
  - Client
  - Servers
  - Pipelines
- Interações -> Conectores
  - Como os componentes se comunicam

FullCycle

Estrutura de Módulos

- Unidades de software
- Pacotes
- Responsabilidades funcionais
- Camadas
- Visão mais micro

Estrutura de Alocação

- É a relação das estruturas componente-conector e módulos e como elas se conectam com “não software”:
  - Tipo de computação
  - Ambientes
  - Testes
  - Build
  - Deployment
  - Controle de versão
  - Organização dos times

Efeito “Ivory Tower”

- Arquitetos tem que andar junto ao time de desenvolvimento

- Quando o arquiteto / time de arquitetura se isolam do dia a dia do time de dev, viram especialistas teóricos que pensam que estão resolvendo problemas, porém estão totalmente descolados da realidade

FullCycle

Decisões arquiteturais

FullCycle

Decisões “erradas" para iniciar o dev de um software

- Os problemas do waterfall
- Arquitetura de CRUD
- Começar pelas ferramentas
- Começar sem testes
- Começar sem processo de CI

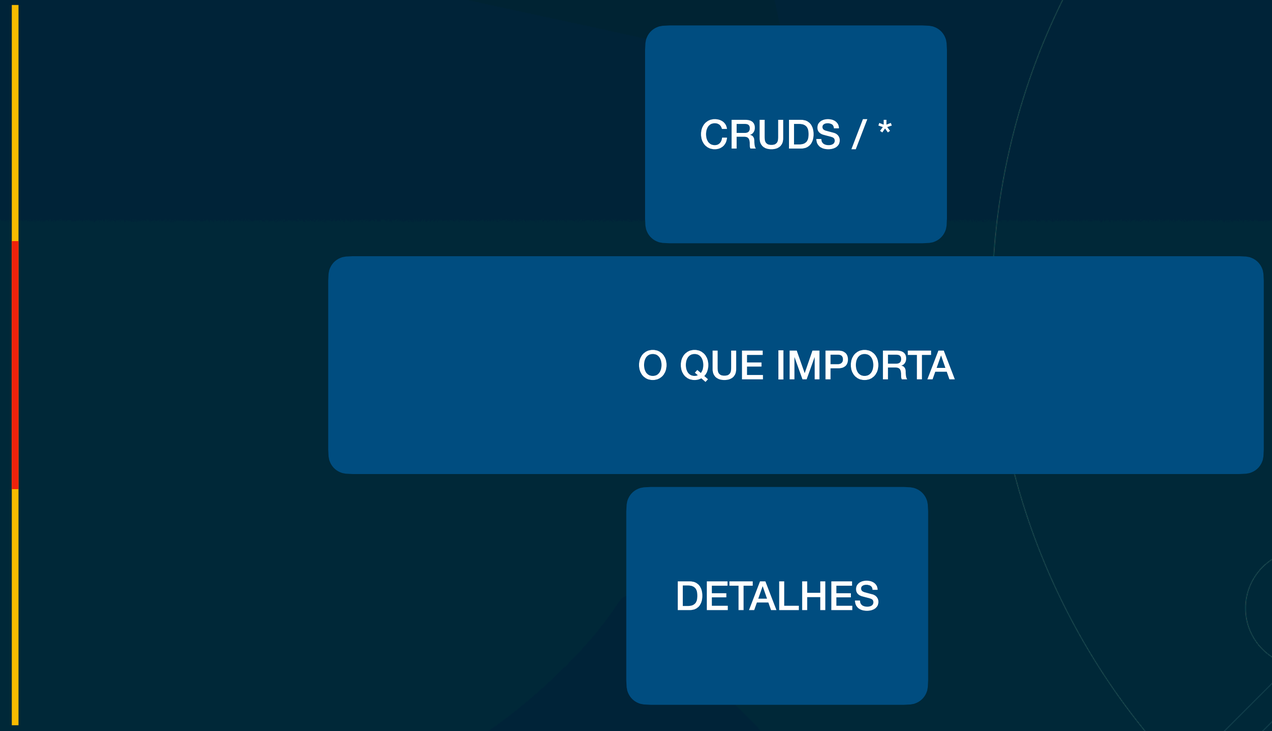
FullCycle

Por onde começar?

Middle Out!

FullCycle

Middle Out



SIMPLES ESTRUTURAÇÃO ARQUITETURAL

ARQUITETURA MAIS ELABORADA

FullCycle

Middle Out

- Foco no desenvolvimento dos componentes core e mais críticos do sistema
- Flexibilidade: Começar pelo meio permite fazer mais mudanças e eventualmente substituir ou incorporar novas tecnologias sem precisar mudar o core do sistema
- Desenvolvimento incremental
- Facilidade na escala
- Menor complexidade na integração com outros sistemas

FullCycle

Não existe arquitetura prematura no Middle Out

- O coração do software / regras de negócio sempre terão que possuir a melhor arquitetura possível

- Partes periféricas e de suporte não necessariamente terão que ter o mesmo peso.

- Isso é válido para componentes-conectores, módulos e alocações

FullCycle

Dimensões arquiteturais

FullCycle

Arquitetura Multidimensional

- Técnica
- Data
- Segurança
- Operacional

FullCycle

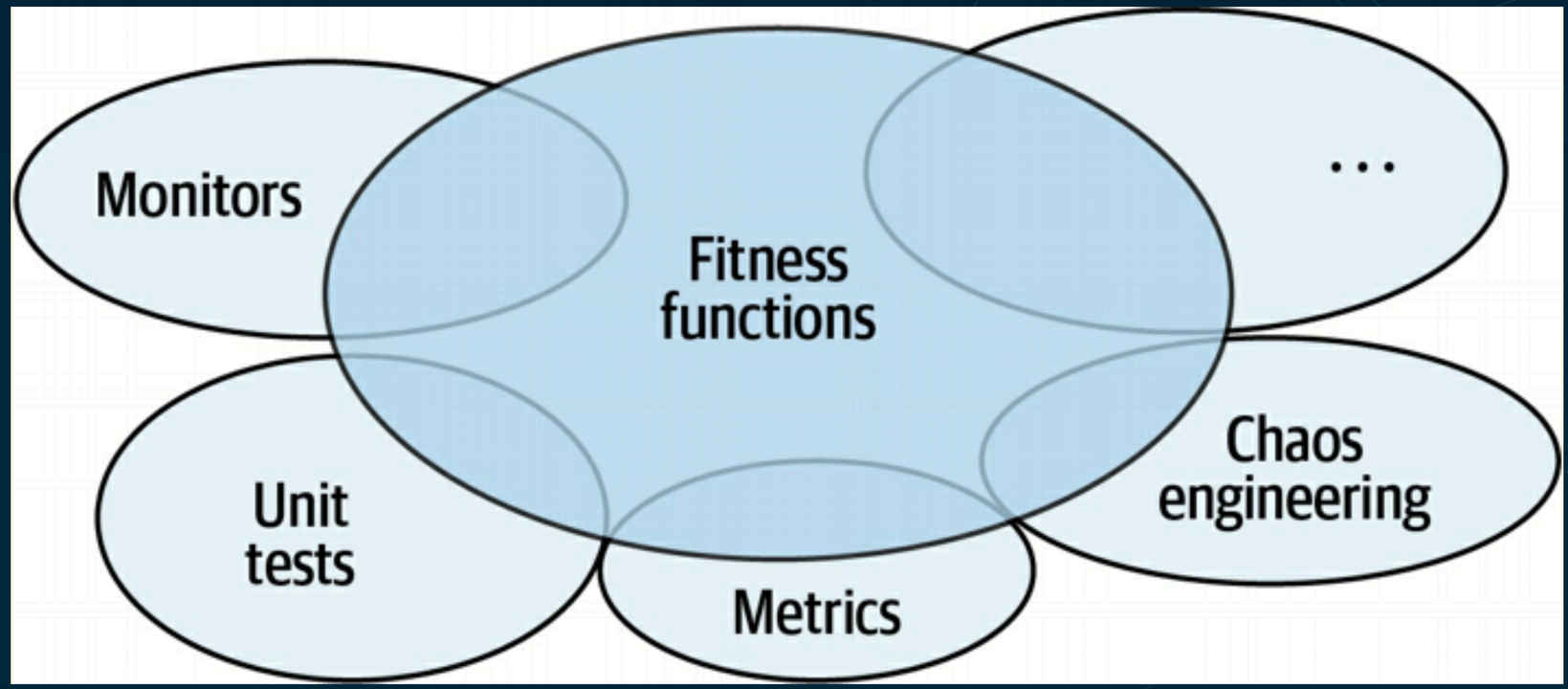
Fitness functions

An architectural fitness function is any mechanism that provides an objective integrity assessment of some architectural characteristic(s).

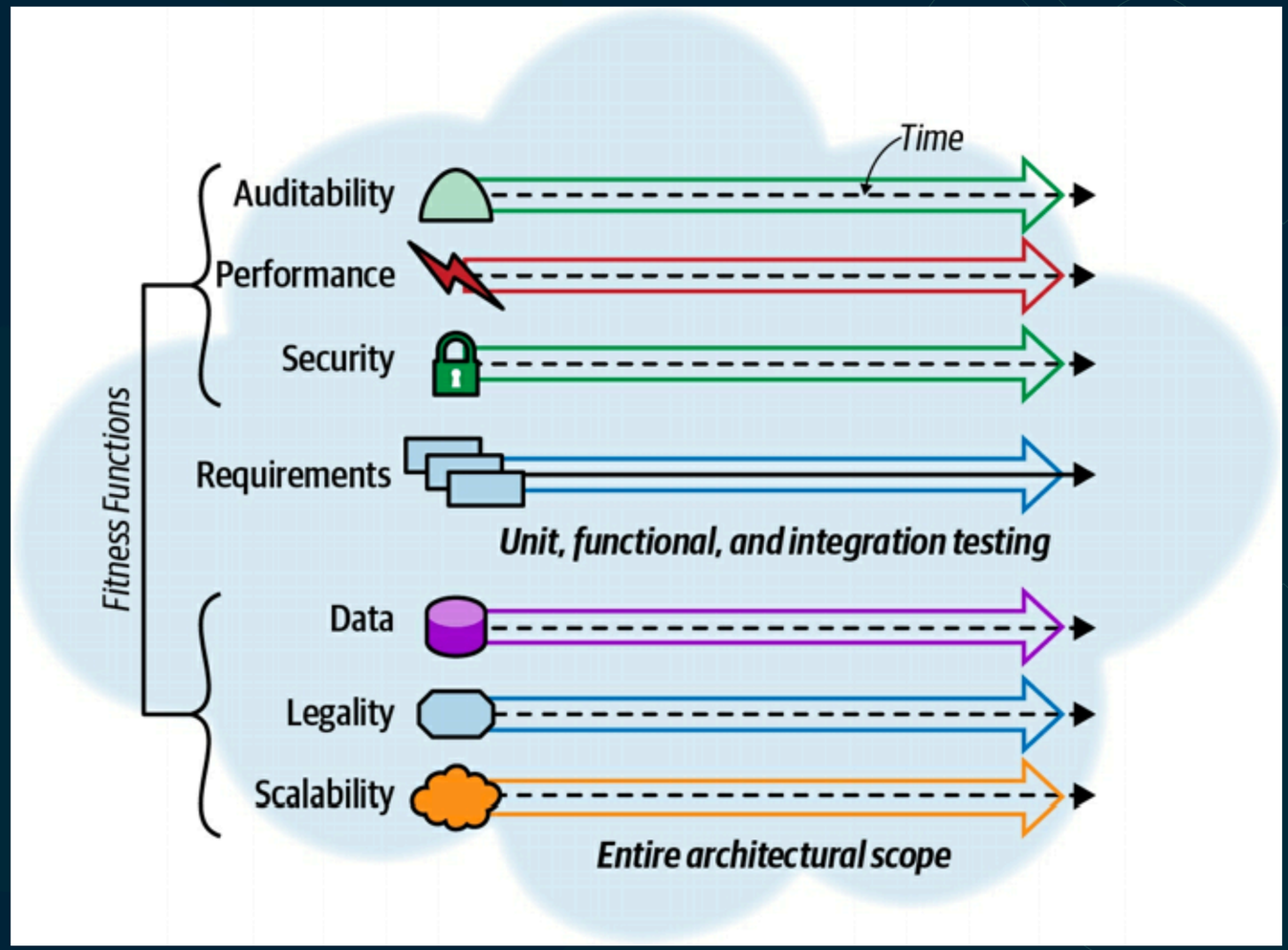
Ford, Neal; Parsons, Rebecca; Kua, Patrick; Sadalage, Pramod. Building Evolutionary Architectures . O'Reilly Media. Kindle Edition.

FullCycle

Fitness functions



Fitness functions



Ford, Neal; Parsons, Rebecca; Kua, Patrick; Sadalage, Pramod. Building Evolutionary Architectures . O'Reilly Media. Kindle Edition.

FullCycle

Fitness functions

- Forma de medir a efetividade, performance e fatores relevantes de acordo com a arquitetura de um software
- Elas são definidas baseadas em diversos critérios, incluindo performance, modularização, maneabilidade, escalabilidade, segurança, etc.
- Parte de acompanhamento do software conforme ele evolui
- Importante identificar os aspectos críticos do sistema e deixá-los de forma mensurável e com objetivos claros.

FullCycle

Acoplamento

- Tudo nesse mundo é acoplado
- Movimento do seu braço e ombro, um trem com vários vagões
- Acoplamento é algo que sempre existiu e sempre continuará existindo
- Mais importante é: Como conviver bem com ele em nossos códigos?

FullCycle

Evolução vs Acoplamento

- É comum fugirmos do acoplamento
- É difícil desenvolver um software complexo sem acoplamento
- Identificar, ter ciência do nível de acoplamento e tirar o máximo benefício disso em comparação com seus “malefícios"
- Se tudo tem um lado bom e ruim, vamos tentar fazer o bom ser muito melhor do que o ruim

Evolução vs Acoplamento

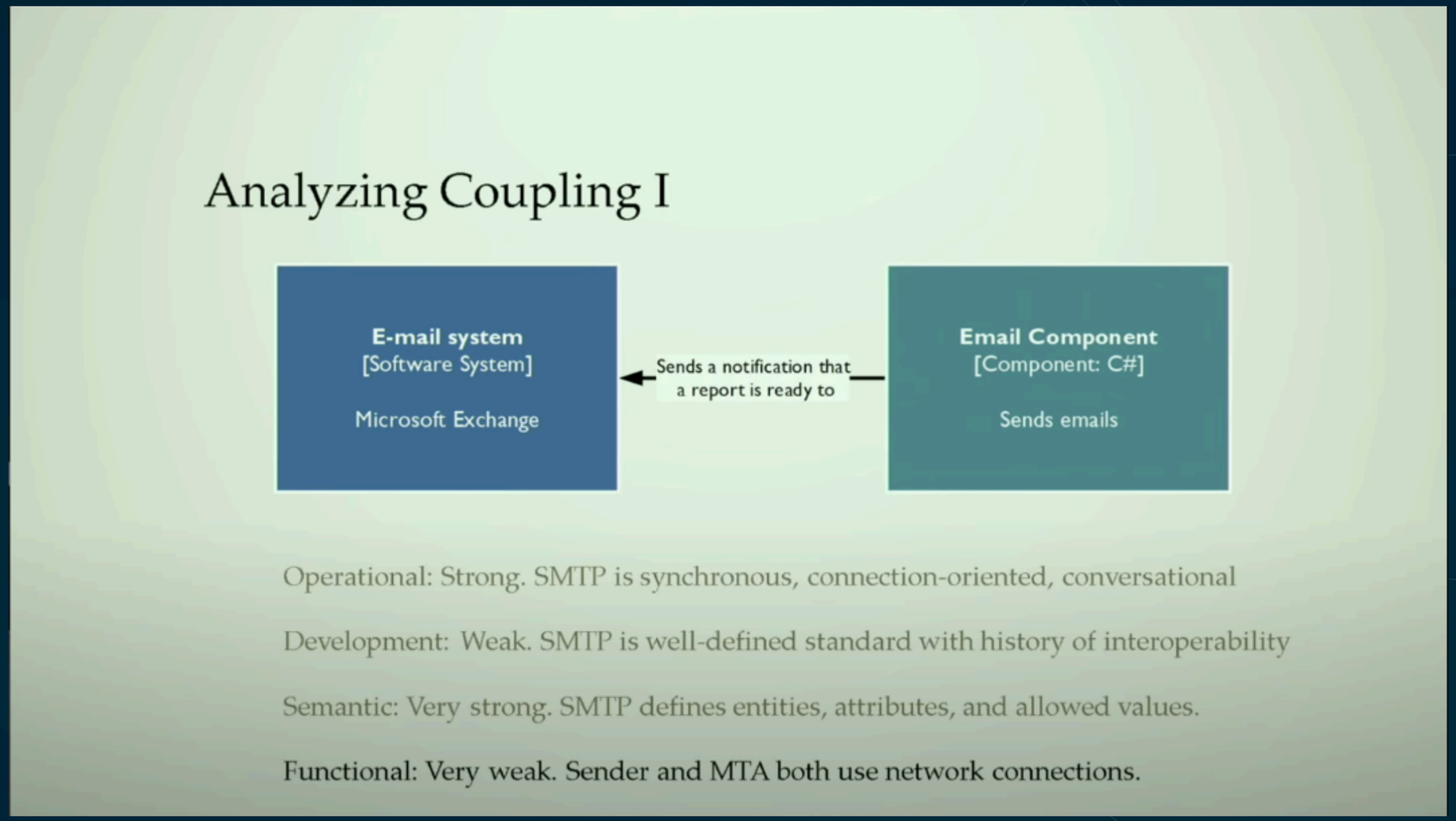
Kinds of Coupling

Type	Effect
Operational	Consumer cannot run without the provider
Development	Changes in producer and consumer must be coordinated
Semantic	Change together because of shared concepts
Functional	Change together because of shared responsibility
Incidental	Change together for no good reason. (E.g., breaking API changes.)

https://www.youtube.com/watch?v=esm-1QXtA2Q

FullCycle

Evolução vs Acoplamento



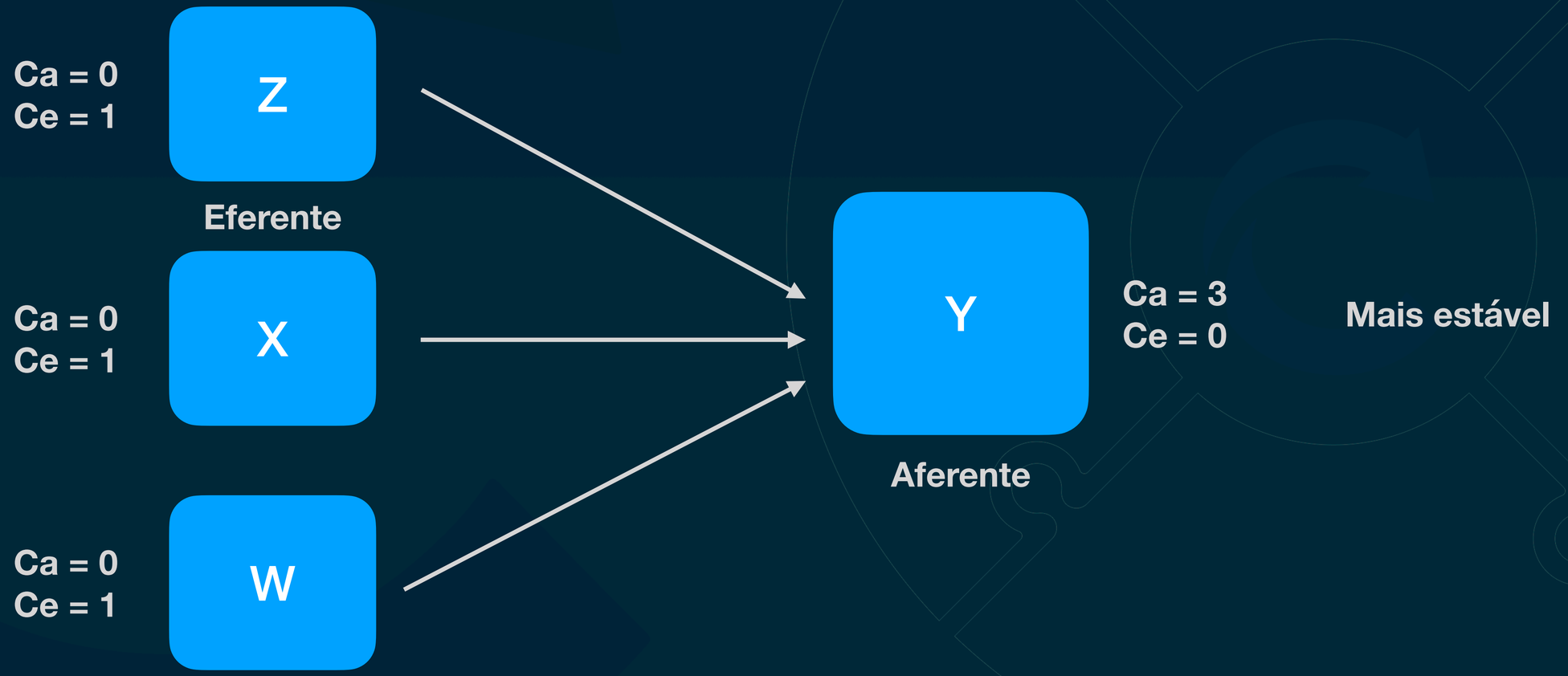
https://www.youtube.com/watch?v=esm-1QXtA2Q

FullCycle

Release it! Design and Deploy Production-Ready Software

FullCycle

Acoplamento aferente vs eferente
ou
Fan-in vs Fan-out



FullCycle

Acoplamento aferente vs eferente

- Componentes aferentes normalmente possuem um nível de risco crítico, pois afetam diretamente outros componentes. Logo, atenção dobrada ao criar e manter esses tipos de componentes.

- Componentes eferentes, por dependerem diretamente de outros componentes, normalmente estão mais propensos a falha, pois dependem diretamente do bom funcionamento de outros componentes

FullCycle

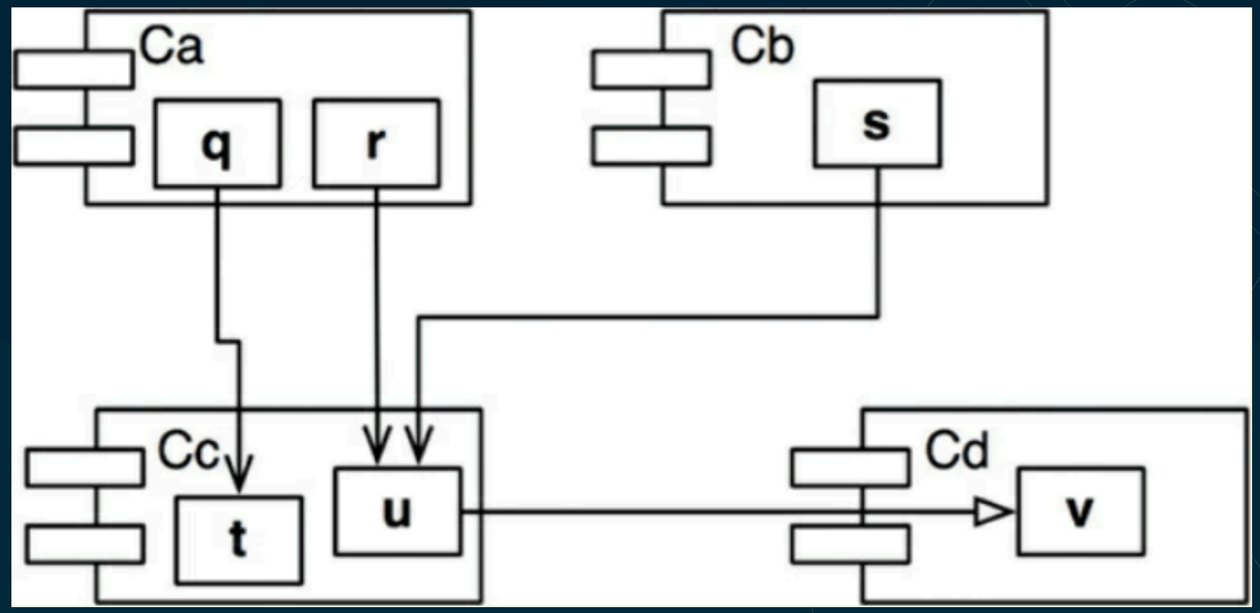
Metrificando Instabilidade

- Instabilidade = Fan out / (Fan-in + Fan-out)
- Range de 0-1
- Quanto menor o valor mais estável
- Quanto maior o valor menos estável

STABILITY METRICS
Martin, Robert C. . Clean Architecture (Robert C. Martin Series) (p. 122). Pearson Education. Kindle Edition.

FullCycle

Metrificando Instabilidade



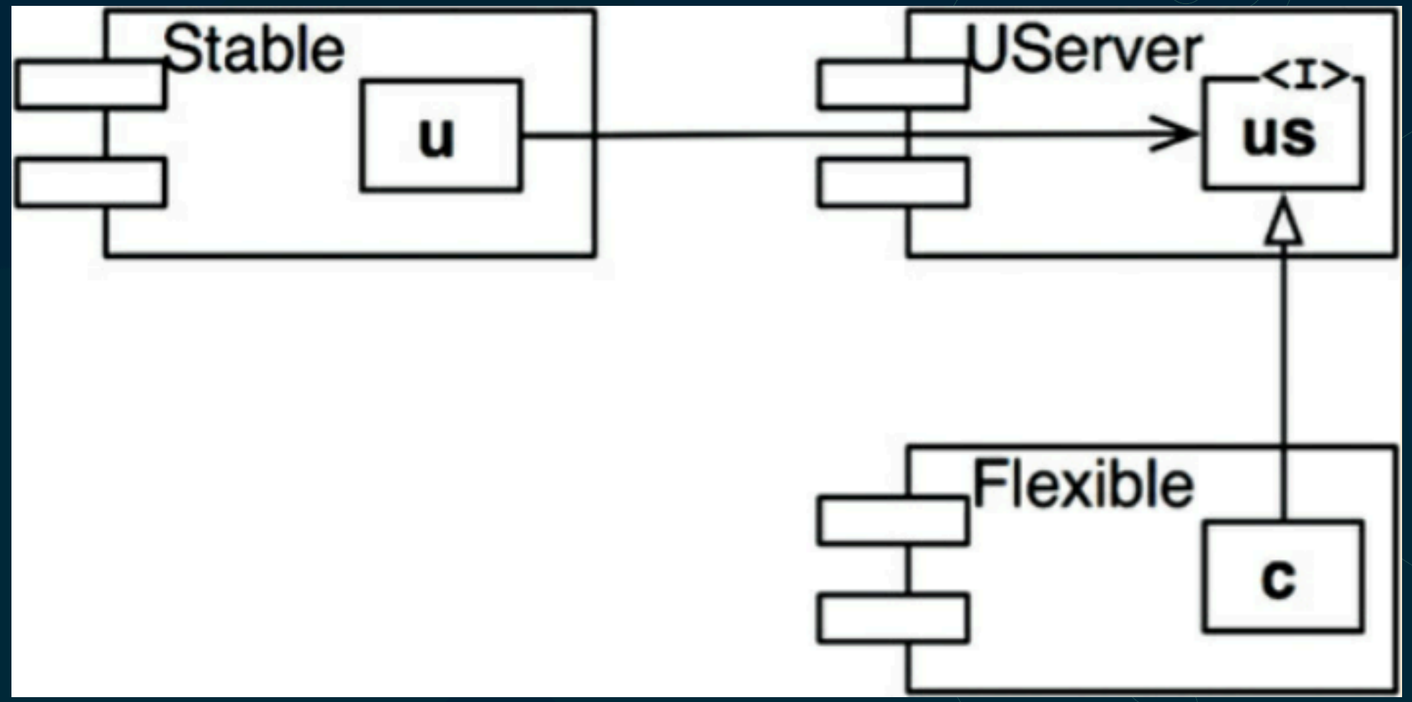
- Instabilidade = Fan out / (Fan-in + Fan-out)
- I = 1 / (3+1) = 1/4

STABILITY METRICS

Martin, Robert C. . Clean Architecture (Robert C. Martin Series) (p. 122). Pearson Education. Kindle Edition.

FullCycle

Componentes abstratos



Abstract Components
Martin, Robert C. . Clean Architecture (Robert C. Martin Series) (p. 125). Pearson Education. Kindle Edition.

FullCycle

Lei de Postel

"Seja conservador no que você faz, mas seja liberal no que você aceita dos outros”

FullCycle

Lei de Postel

- Crie sistemas que sigam padrões ao enviar informações
- Seja flexível e tolerante ao receber informações que possam ser ligeiramente diferentes do especificado

FullCycle

Leis de Lehman / Belady (1974)

- Lei da Mudança Contínua: Um sistema de software deve se adaptar às mudanças em seu ambiente, caso contrário, sua eficácia diminuirá ao longo do tempo.

Metrics and Laws of Software Evolution - The Nineties View
http://users.ece.utexas.edu/~perry/work/papers/feast1.pdf

FullCycle

Leis de Lehman

- Lei do Crescimento da Complexidade: À medida que um sistema evolui, sua complexidade tende a aumentar, a menos que haja um esforço explícito para reduzi-la.

Metrics and Laws of Software Evolution - The Nineties View
http://users.ece.utexas.edu/~perry/work/papers/feast1.pdf

FullCycle

Leis de Lehman

- Lei da Conservação da Familiaridade: O conteúdo global de um sistema de software deve ser mantido em um nível que seja familiar para os desenvolvedores envolvidos na sua evolução.

Metrics and Laws of Software Evolution - The Nineties View
http://users.ece.utexas.edu/~perry/work/papers/feast1.pdf

FullCycle

Leis de Lehman

- Lei da Conservação do Esforço: O esforço total para implementar e manter um sistema de software aumentará ao longo do tempo, mesmo que a quantidade de funcionalidade adicional permaneça constante.

Metrics and Laws of Software Evolution - The Nineties View
http://users.ece.utexas.edu/~perry/work/papers/feast1.pdf