



Microservices - Hands-on com Kafka, Api Gateway & Service Mesh



O SUCESSO É CONSTRUÍDO À “NOITE”, DURANTE O DIA VOCÊ FAZ O QUE TODOS FAZEM. PARA SER BEM SUCEDIDO VOCÊ TEM QUE SER ESPECIAL EM TEMPO INTEGRAL. NÃO SE COMPARE À MAIORIA, POIS ELA NÃO É MODELO DE SUCESSO.

Roberto Shinyashiki

Estrutura, Regras e Wang Fuman Effect



9 - Anos.
9 - Graus negativos
9 - km dia



Regras

- Todos temos algo para fazer lá fora, mas vamos dedicar este tempo a nossa educação e fazer o melhor possível para participar.
 - Perguntas são bem vindas a qualquer momento, fique a vontade.
 - Coloque seu celular no silencioso.
 - Chamada é feita em dois momentos, antes e depois do intervalo.
- Portanto: Chegue no horário, fique na sala e aproveite a aula.

Cristiano Uniga Bajdiuk

- Mestrando em Gestão para Competitividade - Tecnologia da Informação na FGV EAESP.
 - Mestrado Enterprise Architecture na HTWG Konstanz na Alemanha.
- MBA Business Innovation Management na Universidade Federal do Paraná.
 - Bacharel em Engenharia de Software PUCPR.
- Graduação Técnica em Desenvolvimento de Sistemas OPET-PR.

Business Layer

Enterprise Architect em empresas como HSBC, CGI e GVT, trabalho na Monsanto desde 11/2014 como South América Enterprise Architect & AI Head.

Tasks:

“Suportar todo o processo de negócio na região. Com foco principal em arquitetura de aplicações e arquitetura de integração de aplicações, construindo o AS-IS e TO-BE, suportar a inclusão de tecnologias inovadoras alinhando a capacidade de Arquitetura a necessidade e estratégia de Business. E desenvolver a prática de AI”

Education Layer

Coordenador e Professor nos seguintes cursos na FIAP:

Coordenador do MBA de Engenharia de Software, DevOps e BPM.

MBA de Gestão de Tecnologia de Informação. Presencial e OnLine

MBA de Artificial Intelligence and Machine Learning.

MBA de Babook.

Shift Togaf.

Shift Microservices and Kafka.

Contatos:

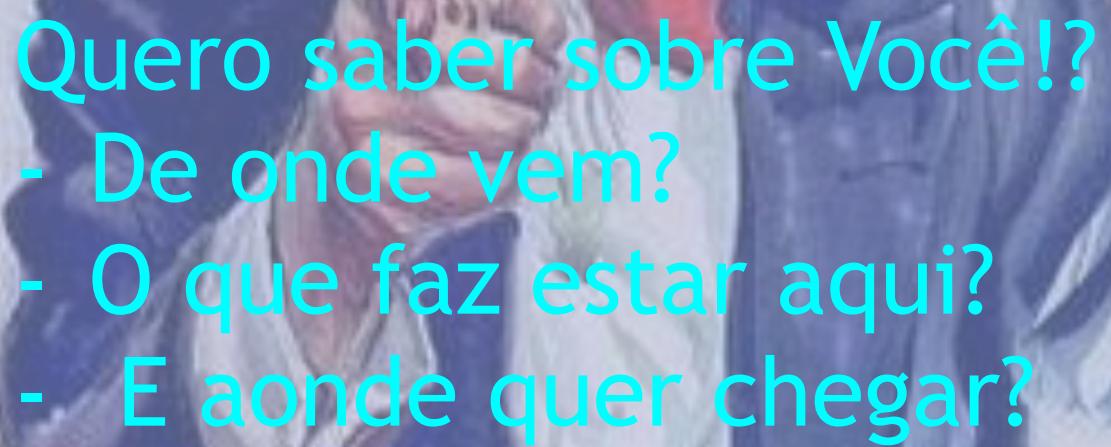
@cristianoub

profchristiano.uniga@fiap.com.br

(11)970580385

linkedin.com/in/cristianoub





Quero saber sobre Você!?

- De onde vem?
- O que faz estar aqui?
- E aonde quer chegar?

Estrutura do Programa

Estrutura do Programa

Day by Day dos nossos encontros

Modulo 1 - Microservices

- Definindo Microservices.
- Arquitetura Evolutiva.
- Arquitetura de Microservices.
- Domain-Driven Design

Modulo 2 - Desenvolvendo Microservices

- Construindo Microservices da Camada de Apresentação ao Backend.
- Containers e Microservices.
- Microservices e Databases.

Modulo 3 - Data Stream com Kafka

- Conhecendo e Instalando o Kafka.
- Entendendo e Desenvolvendo Kafka Producers e Consumers.
- Construindo Pipelines de Data.
- Administrando e Monitorando o Kafka.

Modulo 4 - Integração

- Conhecendo os padrões de integração síncrono e Assíncrono.
- Integrando Microservices com API Gateway e service Mesh.
- Integrando Microservices com Kafka.

Modulo 5 - Microservices no Dia a Dia

- Dividindo o Monolítico.
- Deploy, teste e Monitoramento de microservices.
- Executando o deploy na Cloud.

Foundations for Execution Overview

Sand Castle



Para a estratégia Digital, Construa a Fundação

- ✓ Segue os passos, mas não consegue destaque.
- ✓ Alguns dos seus competidores sempre estão a frente.
- ✓ O que eles fazem de diferente?

Eles executam melhor, pois tem melhor **fundação para execução**. Eles inserem tecnologia nos seus processos. Tornando o “Core Operations” muito eficiente e sustentável.

Enterprise Architecture the Digital Agent

Estratégia de Negócios:

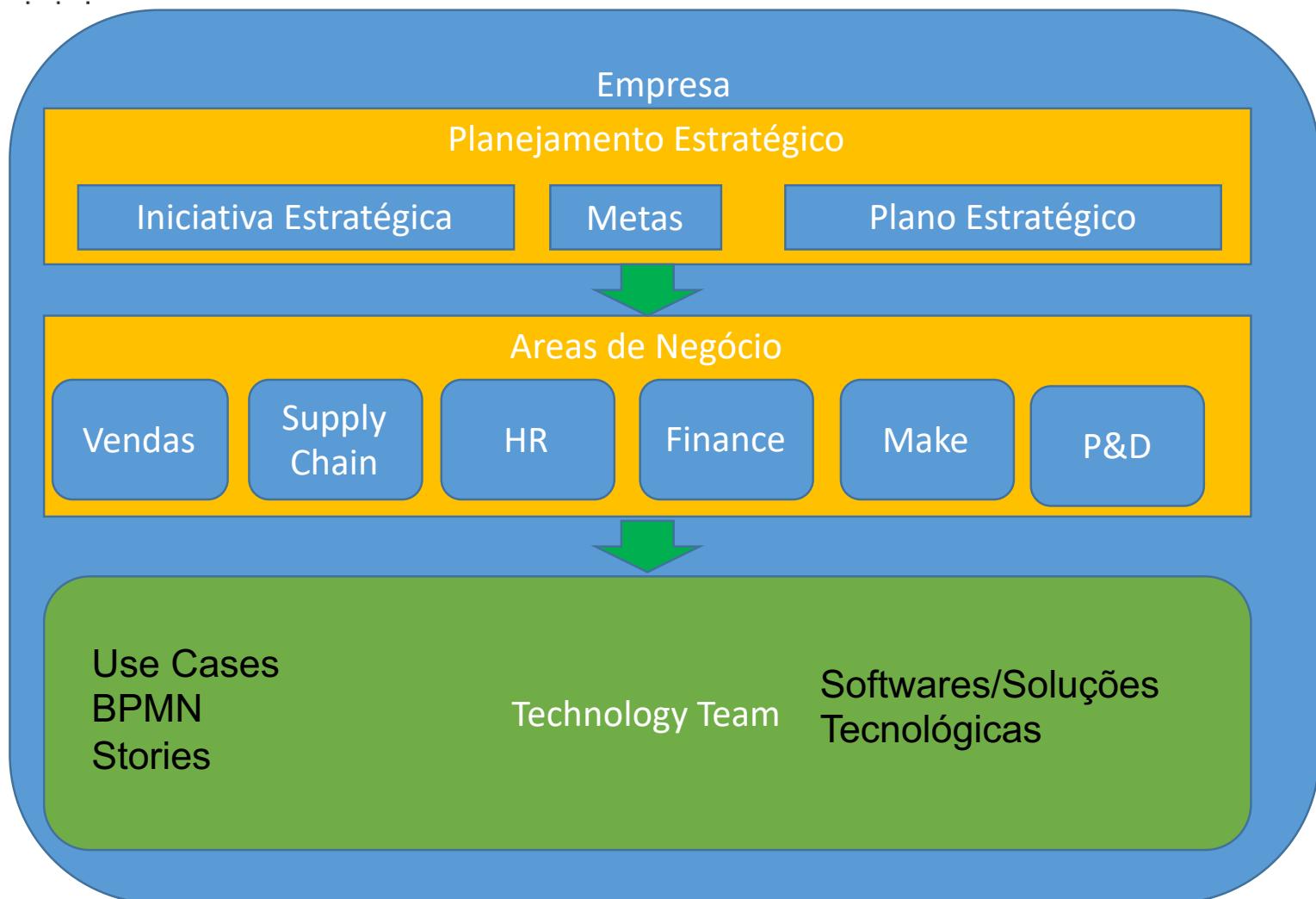
“Conceber um plano que garanta que a empresa atinge os seus objetivos. No caso dos negócios, é ser lucrativo e bater a concorrência.”

Planejamento Estratégico:

“Se trata de uma metodologia gerencial que permite estabelecer a direção a ser seguida pela organização”

Fontes: <https://www.portal-gestao.com/artigos/6316-o-que-%C3%A9-a-estrat%C3%A9gia-de-neg%C3%B3cios.html>
<http://www.portal-administracao.com/2014/06/planejamento-gestao-estrategica-o-que-e.html>

Enterprise Architecture

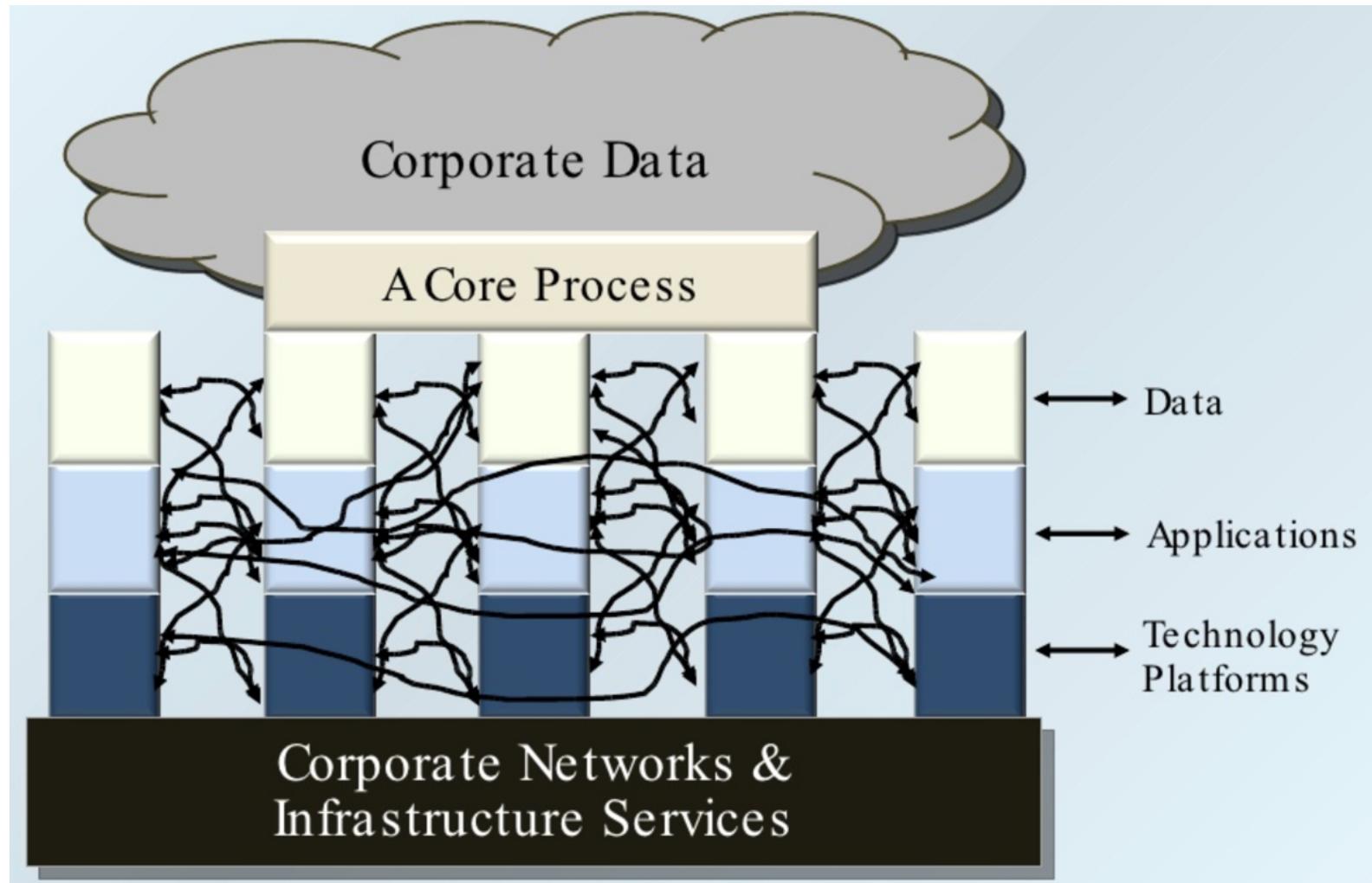


Enterprise Architecture

- A quanto tempo a empresa que você trabalha existe?
- A quanto tempo a estratégia de Negócio da empresa existe?
- A quanto tempo o principal sistema da empresa existe?
- Quanto tempo demora para uma alteração no sistema ser efetivada?

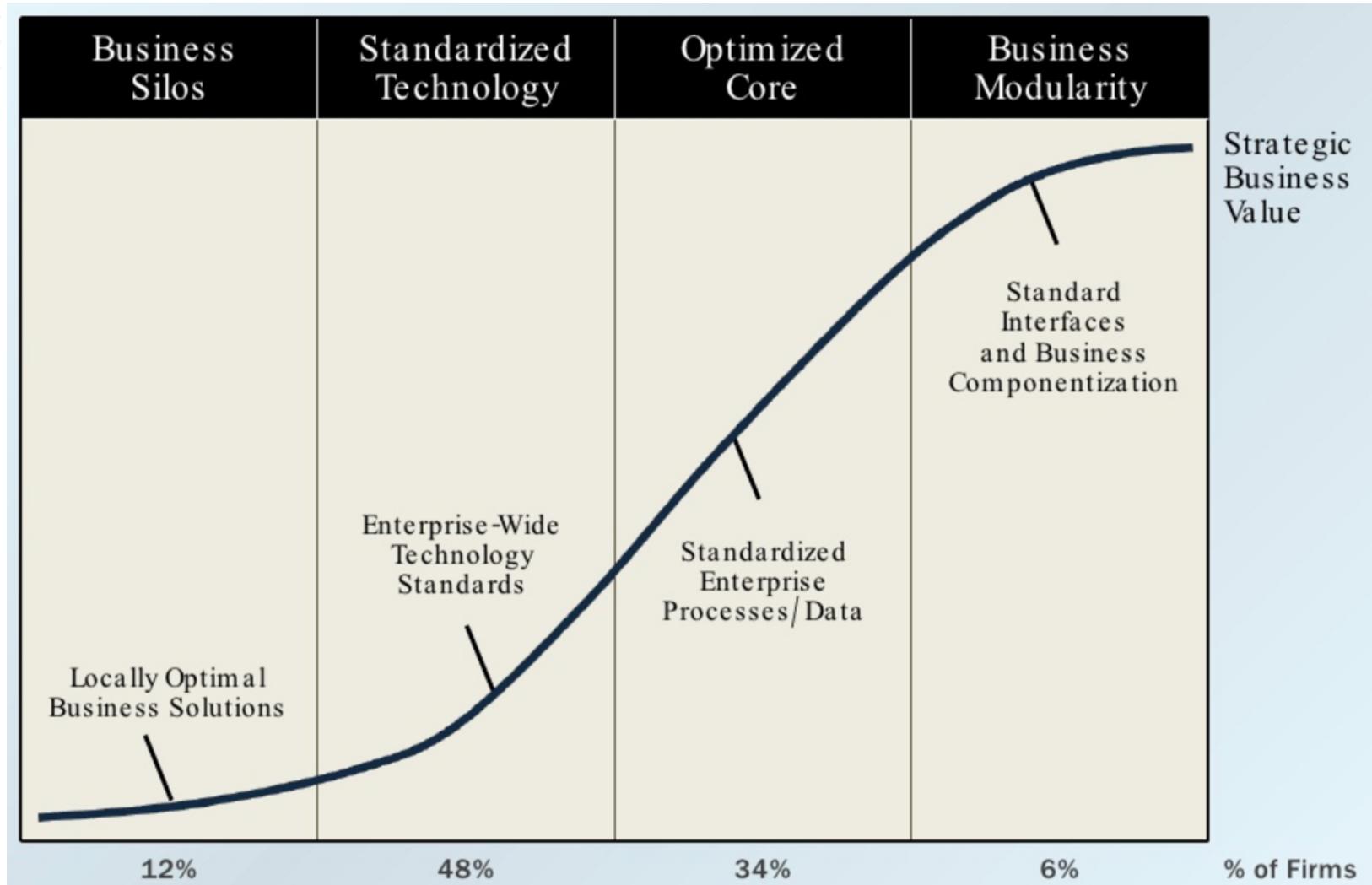
Enterprise Architecture

Economia Digital



Fontes: Architecture as Strategy: Creating a Foundation for Business Execution, J. Ross, P. Weill, D. Robertson, HBS Press, June 2006.

Enterprise Architecture



Fontes: Architecture as Strategy: Creating a Foundation for Business Execution, J. Ross, P. Weill, D. Robertson, HBS Press, June 2006.

Enterprise Architecture

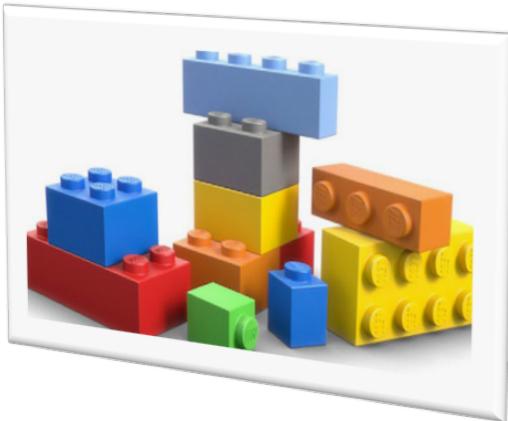
Em Resumo:

As empresas, esperam da área de Tecnologia:

- Agilidade nas definições de negócio.
- Time to Market.
- Alta e rápida capacidade de adaptação as mudanças necessidades.

Foundations Building Blocks Overview

Building Blocks



NATIONAL GEOGRAPHIC | PHOTO OF THE DAY | TV | PERPETUAL PLANET | LATEST STORIES | LEARN MORE

SEE FOR YOURSELF, WHAT WILD EDENS IS ALL ABOUT

| EXPLORING MARS | NEWS |

Building Blocks of Life Found on Mars

Two landmark discoveries reveal organic carbon on the red planet, shaping the future hunt for life on Mars.



NEWS | news | sport | weather | obituaries | cars | travel | capital | culture | photos | future | TV | books | sign in

Home | Video | World | UK | Business | Tech | Science | Stories | Entertainment & Arts | Health | World News TV | In Pictures | Reality Check | Newsbeat | More ▾

NOW PLAYING

- ▶ 1:05 Life on Mars?
- ▶ 1:19 'Marsquake' monitor set for lift-off
- ▶ 2:13 Science stories coming up in 2018
- ▶ 0:59 Nasa rocket takes off for Mars
- ▶ 1:06 Nasa to send helicopter to Mars

08 Jun 04 May 01 Jan 05 May 12 May

Nasa: Building blocks of life on Mars found

Nasa says organic matter found at the bottom of a crater on the Red Planet suggests there might have been life there once. Samples of the matter were analysed by a team of scientists, including Prof Sanjeev Gupta from Imperial College London. He told **Today** how significant the discovery is.

08 Jun 2018

f t e m Share

Formada há cerca de 4,5 bilhões de anos a partir do remanesciente de poeira e gás em espiral da explosão de uma supernova de uma velha estrela.



Em 150 milhões de anos a massa fundida se estabilizou e resfriou, uma crosta sólida logo se formou, junto com uma atmosfera rudimentar composta principalmente de dióxido de carbono, vapor de água e nitrogênio.

Cerca de 3,8 bilhões de anos atrás, após uma colisão quase catastrófica com outro planeta logo após a formação da Terra (que criou a Lua no processo), acredita-se que oceanos quentes gradualmente se formaram, do vapor escapando da crosta e da atividade vulcânica e meteoritos gelados

Embora em um ambiente perigoso para a vida (incluindo bombardeamento por asteróides e atividade vulcânica) os ingredientes necessários estavam todos presentes de alguma forma ou de outra: água líquida, e building blocks químicos (geralmente considerados seis elementos: oxigênio, hidrogênio, carbono, nitrogênio, enxofre e fósforo) e algum tipo de fonte de energia.

Building Blocks

O que é um building block?

Building Block é um pacote de funcionalidades definido para atender uma necessidade de business ao longo da empresa.

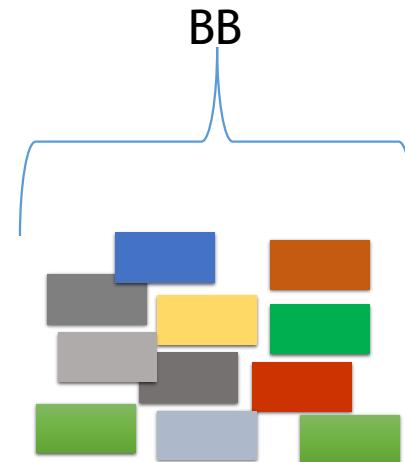
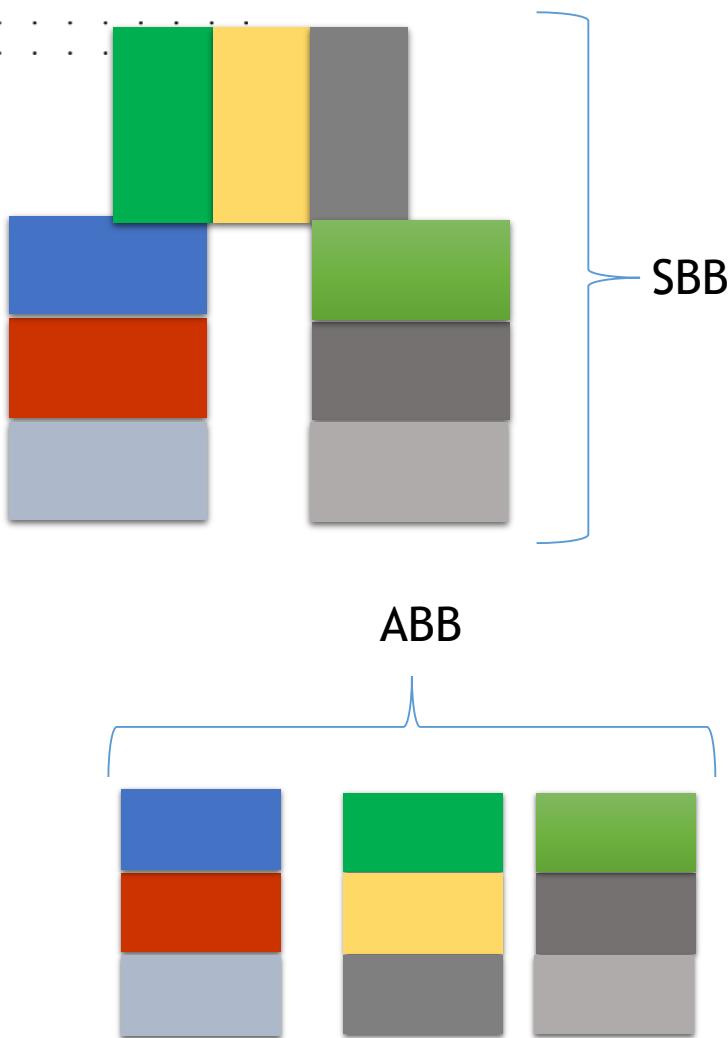
Um building block como característica fornece interfaces para permitir acesso a suas funcionalidades.

O Building block pode interoperar com outros BB, criando uma interdependência entre eles.

Características de um “bom” Building Block:

- ❑ Considera a implementação e o uso, e evoluí para explorar tecnologia e padrões.
- ❑ Pode ser montado a partir de outros building block.
- ❑ Pode ser uma submontagem de outro building block.
- ❑ Idealmente, um building block é reutilizável e substituível, e bem especificado com interfaces estáveis.
- ❑ Sua especificação deve ser ligeiramente acoplada à sua implementação, de modo que possa ser realizada de várias maneiras sem impactar a especificação do building block.
- ❑ Uma arquitetura é a composição de um grupo de building Block representados em um modelo arquitetônico junto a uma especificação de como esses building Block estão conectados para atender aos requisitos gerais de um sistema de informação.

Building Blocks {BB, ABB, SBB}



Modulo 1 - Microservices

Origens

Domain-Driven Design

Continuous Delivery

Microservices

Small autonomous Teams

On-Demand Virtualization

Infrastructure Automation

Microservices

Microservices são serviços **pequenos** e **autônomos** que trabalham juntos. Newman, Sam.

Pequenos e focados em fazer uma coisa muito bem!

- Manutenção X Crescimento.
- Repetição de Funcionalidades.
- Coesão (pedaços de código relacionados agrupados)

Single Responsibility Principle “Junte as coisas que mudam pela mesma razão, e separe as coisas que mudam por diferentes razões”.

- Services boundaries on business boundaries.
- How small is small?
 - 2 W
 - Small enough and no smaller.
 - System to big that you'd like to break down? Nós temos o senso.
 - Complexidade X Facilidade.

Microservices

Microservices são serviços **pequenos** e **autônomos** que trabalham juntos. Newman, Sam.

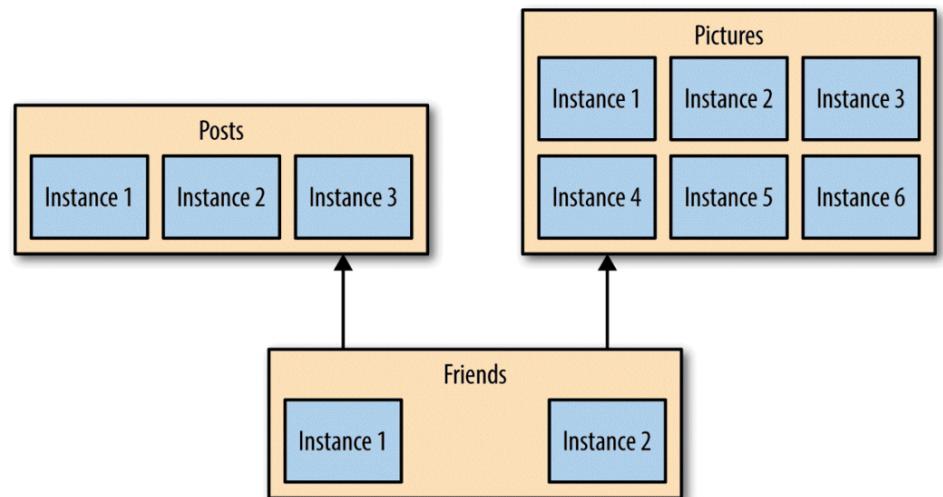
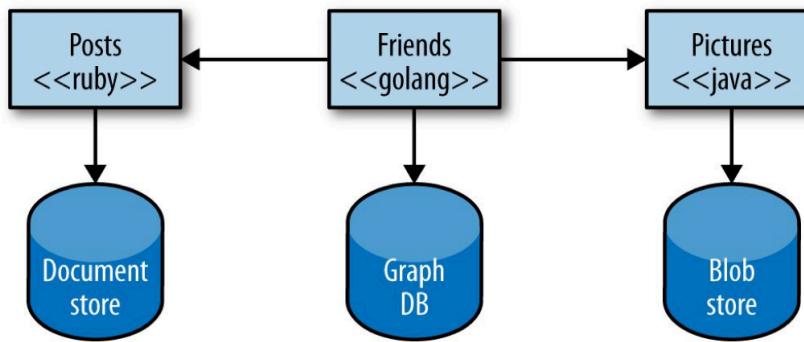
Autônomos

- Microservices são Building Blocks separados. (Deploy, PaaS ou S.O)
- Comunicação via chamadas de network. (+ Separação e Alto Acoplamento)
- Habilidade de mudar com independência. (Consumers salvos)
- O service deve expor uma API. (Colaboração através de API)
- Technology-agnostic API. (Reduz Impactos, Ambiente heterogêneo)
- Golden Rule:
 - Você pode fazer uma mudança no service e fazer o deploy dele sozinho sem mudar qualquer outra coisa?
- Ações para o sucesso:
 - Modelar o service corretamente.
 - Fazer a API da maneira correta.

Microservices

Benefícios (Como vender para o chefe)

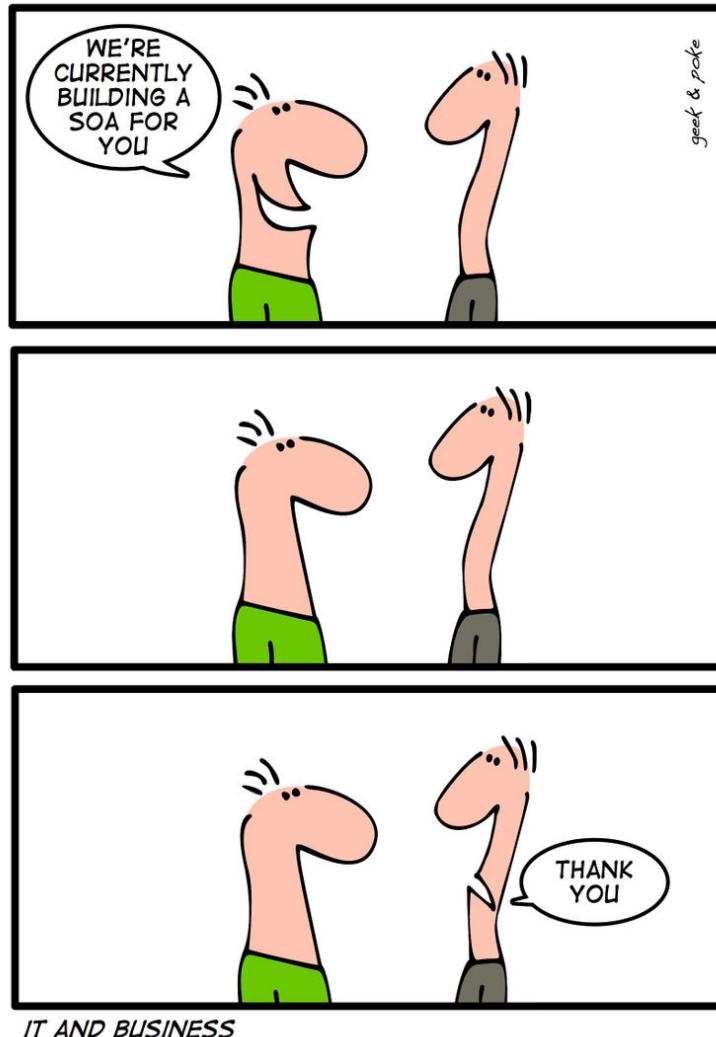
- Heterogênea em Tecnologia. (all u can)*
- Resiliência. (O mundo não acaba, isolamos o problema)
- Scaling. (be strong)**
- Fácil de realizar o deploy. (+ / -)
- Alinhamento Organizacional. (Times grandes problemas grandes)
- Composability (Verdadeiro re-uso).
- Otimizado para substituição.





E o SOA??? X Microservices

Microservices, e o SOA?



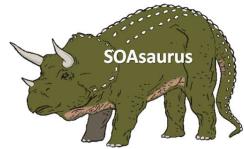
Microservices, e o SOA? O Problema

- Monster Services.
- Produziu “Driven Services” e não Business Services.
- Orquestração.
- Ferramenta era tudo na vida (O cara da oracle, o cara da IBM)
- Muito difícil de mudar (Porquê fizemos isto??)
- Muito difícil de testar.

Microservices, e o SOA? Is Dead



SOA met its demise on January 1, 2009, when it was wiped out by the catastrophic impact of the economic recession. SOA is survived by its offspring: mashups, BPM, SaaS, Cloud Computing, and all other architectural approaches that depend on "services"



Once thought to be the savior of IT, SOA instead turned into a great failed experiment—at least for most organizations. SOA was supposed to reduce costs and increase agility on a massive scale. Except in rare situations, SOA has failed to deliver its promised benefits. After investing millions, IT systems are no better than before. In many organizations, things are worse: costs are higher, projects take longer, and systems are more fragile than ever. The people holding the purse strings have had enough. With the tight budgets of 2009, most organizations have cut funding for their SOA initiatives.

People forgot what SOA stands for. They were too wrapped up in silly technology debates (e.g., "what's the best ESB?" or "WS-* vs. REST"), and they missed the important stuff: architecture.

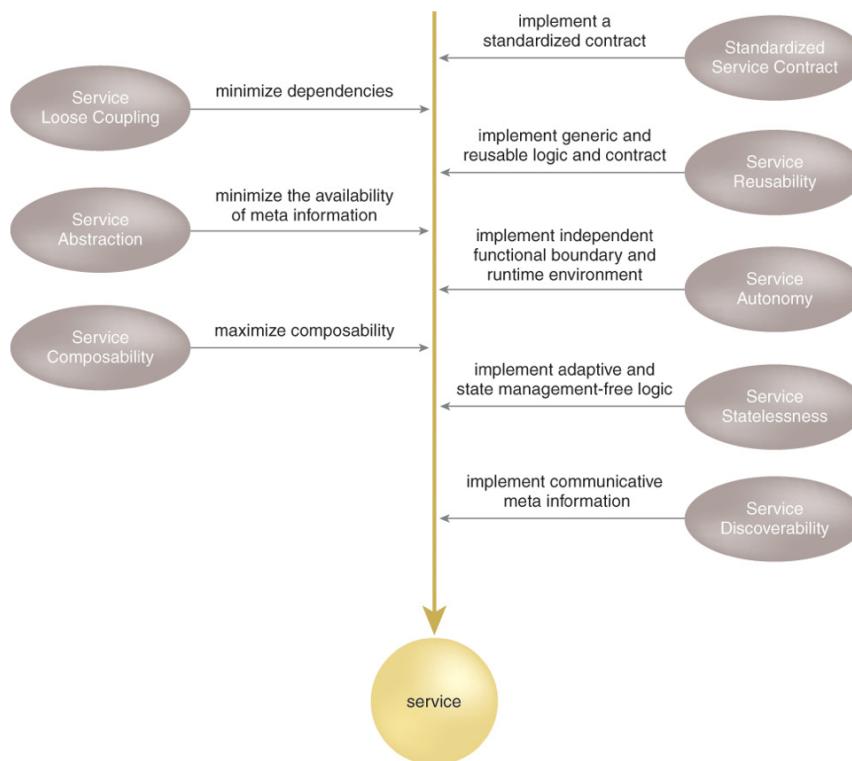
The demise of SOA is tragic for the IT industry. Organizations desperately need to make architectural improvements to their application portfolios. Service-orientation is a prerequisite for rapid integration of data and business processes; it enables situational development models, such as mashups; and it's the foundational architecture for SaaS and cloud computing.

SOA fatigue has turned into SOA disillusionment. Business people no longer believe that SOA will deliver spectacular benefits. "SOA" has become a bad word. It must be removed from our vocabulary.

Although the word "SOA" is dead, the requirement for service-oriented architecture is stronger than ever.

Microservices, e o SOA? O Lado Bom!

- Quebrar o monolítico em serviços independentes.
- Integração Invés de acoplamento interno.
- Encorajou maior aceitação de eventual consistência. (DBA the old guy)
- Forçou “Pensarmos diferente” sobre a forma antiga de trabalhar.



Fonte: <http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html>

Microservices

Microservices são serviços **pequenos** e **autônomos** que trabalham juntos. Newman, Sam.

Pequenos e focados em fazer uma coisa muito bem!

- Manutenção X Crescimento.
- Repetição de Funcionalidades.
- Coesão (pedaços de código relacionados agrupados)

Single Responsibility Principle “Junte as coisas que mudam pela mesma razão, e separe as coisas que mudam por diferentes razões”.

- Services boundaries on business boundaries.
- How small is small?
 - 2 W
 - Small enough and no smaller.
 - System to big that you'd like to break down? Nós temos o senso.
 - Complexidade X Facilidade.

Microservices

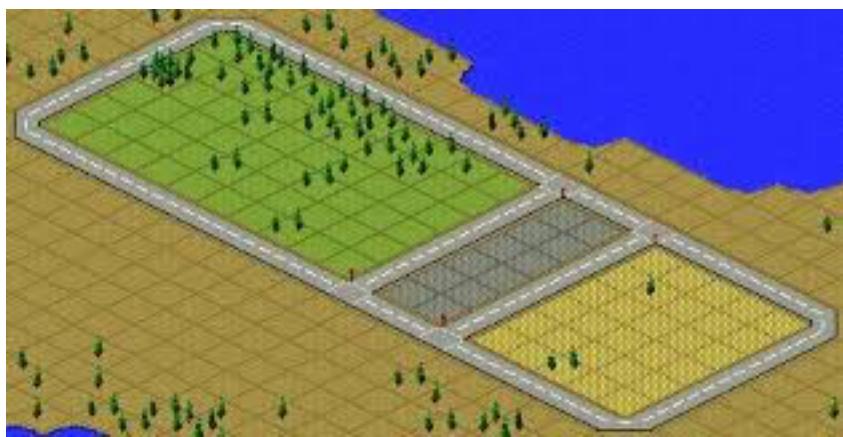
Pontos de Atenção quando optamos por Microservices Microservices não é solução para tudo!

- Granularidade é crucial. (Balanceamento entre coesão e acoplamento)
- Scale independente.
- Monitoramento é crucial.
- Design para services fail.
- Automação de infraestrutura e deploy é essencial.
- Flexibilidade de plataforma deve ser gerenciada.
- Mudanças de Interface devem ser gerenciadas.
- Esteja preparado para combinar ou dividir services.

Modulo 1 - Evolutionary Architecture

Evolutionary Architecture

Arquitetos e Engenheiros de Software precisam mudar a sua forma de pensar. Invés de criarmos o produto final perfeito, para criar um framework a partir do qual os sistemas corretos possam surgir e que neste framework possam crescer conforme crescemos o nosso conhecimento de business.



Evolutionary Architecture

Evolutionary Architecture suporta mudanças continuas e incrementais ocorrendo em múltiplas dimensões como princípio básico.

Nó devemos planejar os nossos sistemas para permitir mudanças e não pensar em especificar tudo até o mínimo detalhe.

Devemos nos preocupar muito menos com o que ocorre dentro das services boundaries e muito mais com o que acontece entre as services boundaries.

Estrutura:

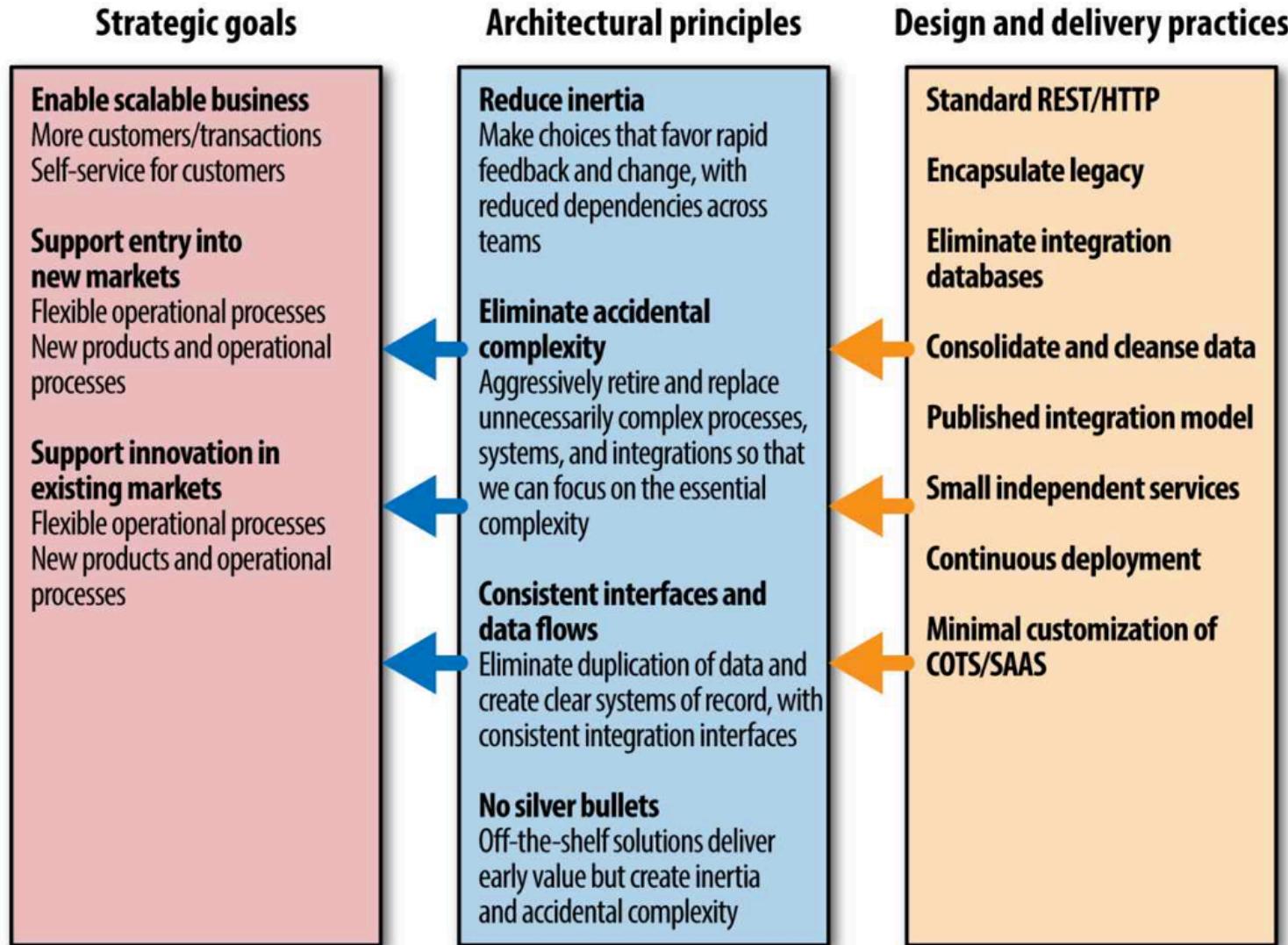
Metas Estratégicas. (Como ajudar o business)

Princípios (Regras para alinhar com uma meta maior - 10 to 15)

Práticas (Como garantir o princípio)

Fitness Functions (Teste e Monitoramento)

Evolutionary Architecture

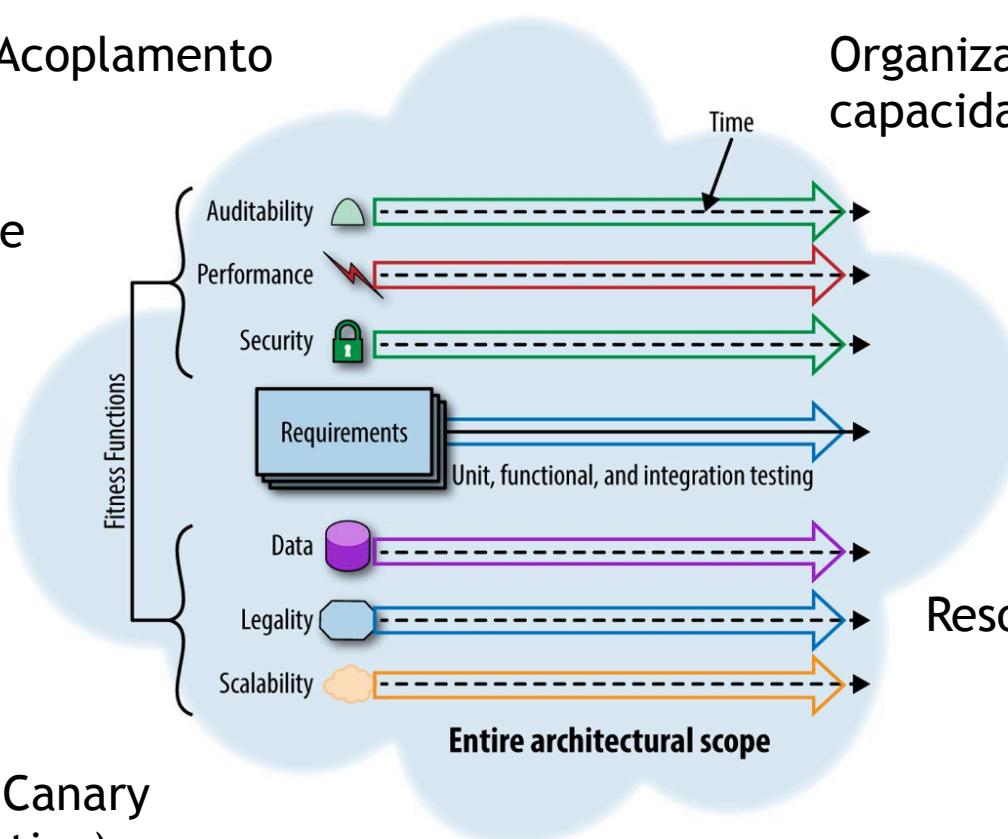


Evolutionary Architecture - Princípios

Modularidade & Acoplamento

Organizado “em volta” das capacidades de business

Último momento de decisão



Experimentação (Canary Release & A/B Testing)

Resolva as dores primeiro

Fitness Function

Modulo 1 - Arquitetura de Microservices

Arquitetura de Microservices

Para a arquitetura de microservices damos o nome de MSA.

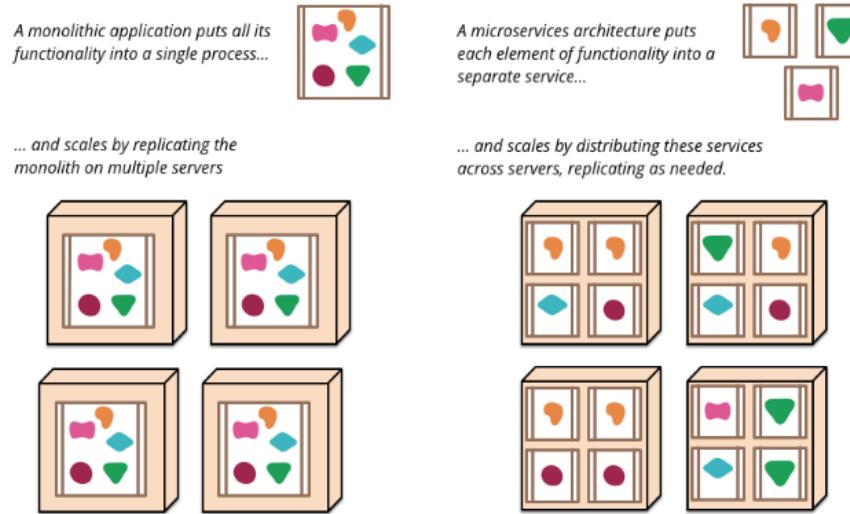
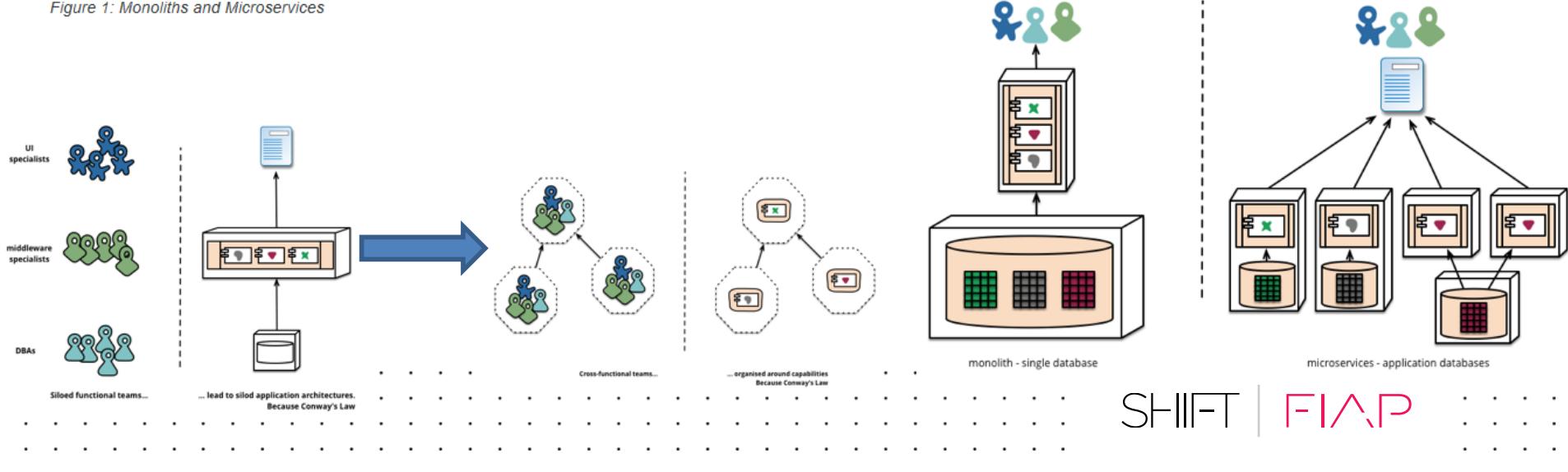


Figure 1: Monoliths and Microservices

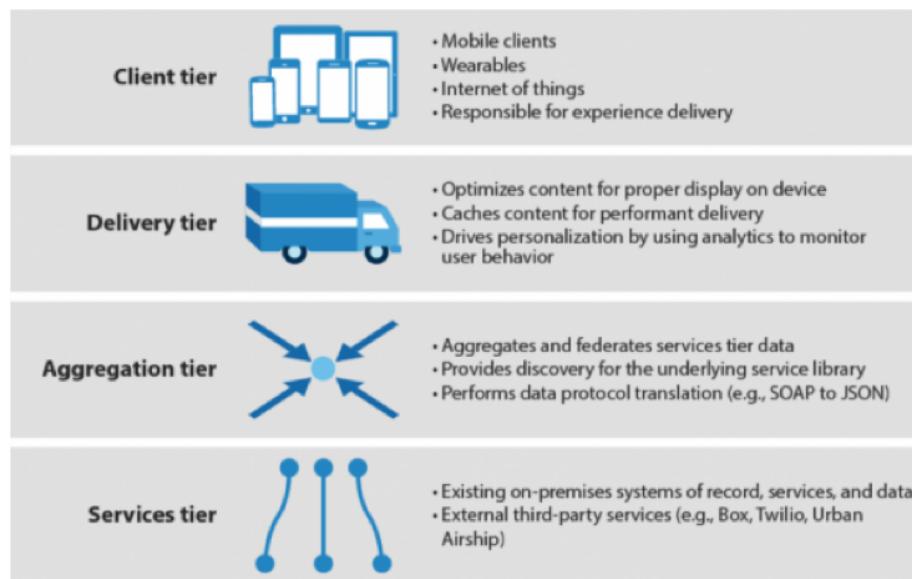
Características:

- Componentização via Serviços.
- Organizado através da capacidade de Business.
- Produtos não Projetos.
- Smart endpoints and dumb pipes.
- Governança Descentralizada.
- Gerenciamento de Data descentralizado.
- Automação de Infraestrutura.
- Design for Failure.
- Design Evolucionário.



Arquitetura de Microservices

Camada	Responsabilidades
Cliente	Totalmente independente das demais camadas e preparada para suportar todos os devices.
Entrega	Responsável pela otimização da entrega, podendo rodar isoladamente nas CDN. Incluindo in memory DB`s.
Agregação	API layer para demanda interna e externa.
Serviços	Camada composta pelos serviços S.A. e microservices que compõem a aplicação.



MSA e BaaS

Modulo 1 - Domain-Driven Design

Domain-Driven Design

O Desafio da Complexidade:

Muitas coisas podem colocar projetos fora do rumo:

- Burocracia.
- Perda do controle da complexidade.
- Alguns fatores de design são técnicos, mas a complexidade na maioria dos projetos não é técnica.

A Complexidade está no Domínio.

Quando a complexidade do domínio não é resolvida no design ... não interessa o quanto bom somos tecnicamente.

Domain-Driven Design

O que é:

- Domain-Driven design é uma ferramenta para Engenheiros de Software e Arquitetos para necessidades complexas, conectando a implementação a um modelo em evolução.
- Domain: É a área de domain para a qual os usuários aplicam um programa, este é o domínio do software.
- Domain-Driven Design (DDD) é sobre o mapeamento de conceitos de domínio de negócios em artefatos de software.

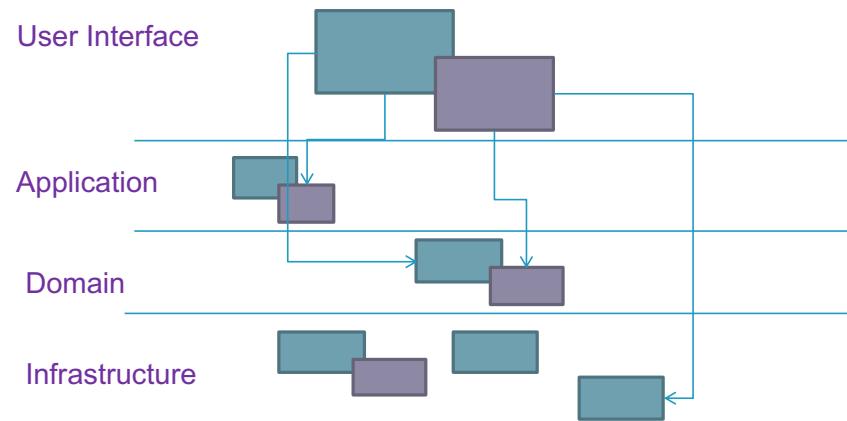
Benefícios:

- Concentre-se na lógica do domínio e no domínio.
- Pensando e um conjunto de prioridades no desenvolvimento de software.
- Modelo comum, entre as partes interessadas de negócios e de TI na empresa.
- O modelo reflete o modelo de negócios.
- Melhora a capacidade de reutilização e capacidade de teste dos objetos do domínio de negócios.

Domain-Driven Design

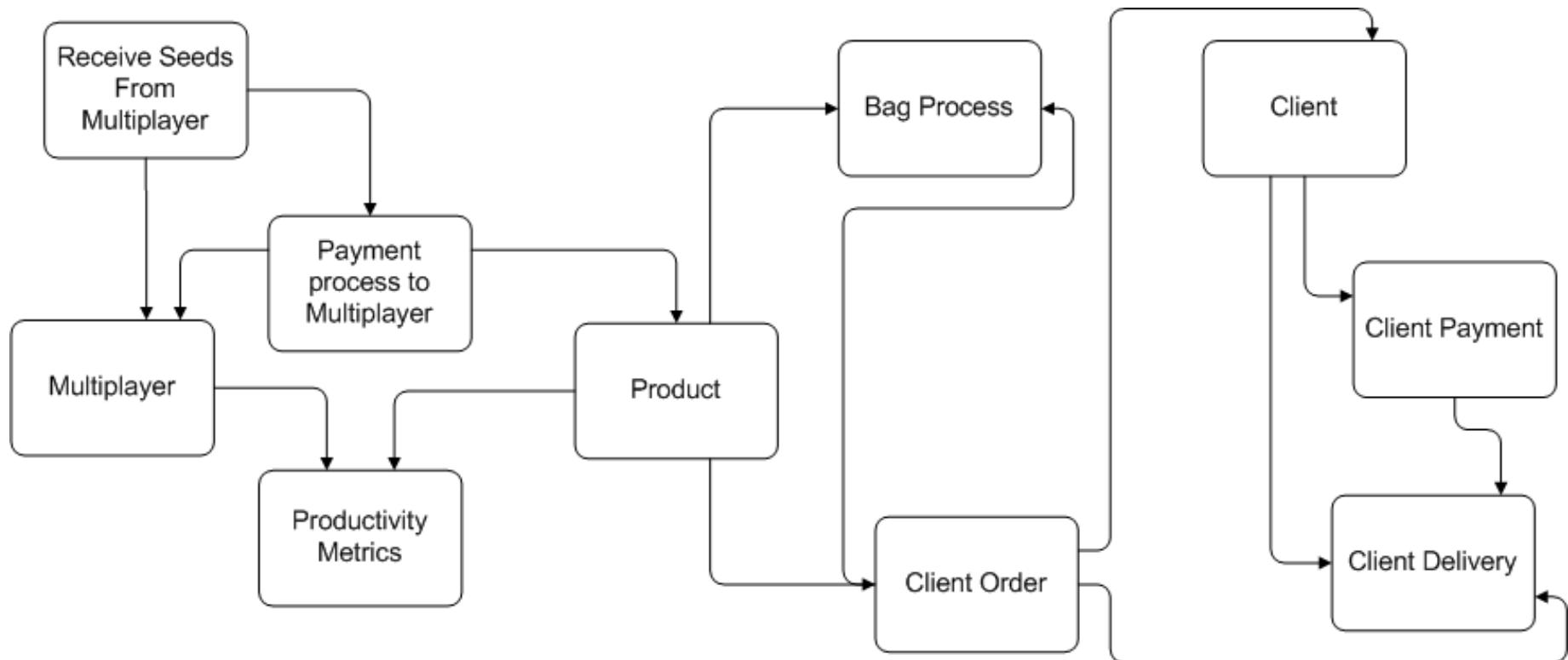
Componentes:

- Strategic Design (Business Process)
 - Bounded Context
 - Define o limite (borda ou tamanho) de cada modelo.
 - Context Map
 - Provê a visão global do projeto através de contexto e relacionamento destes contextos.
- Arquitetura em Camadas (Layered Architecture)



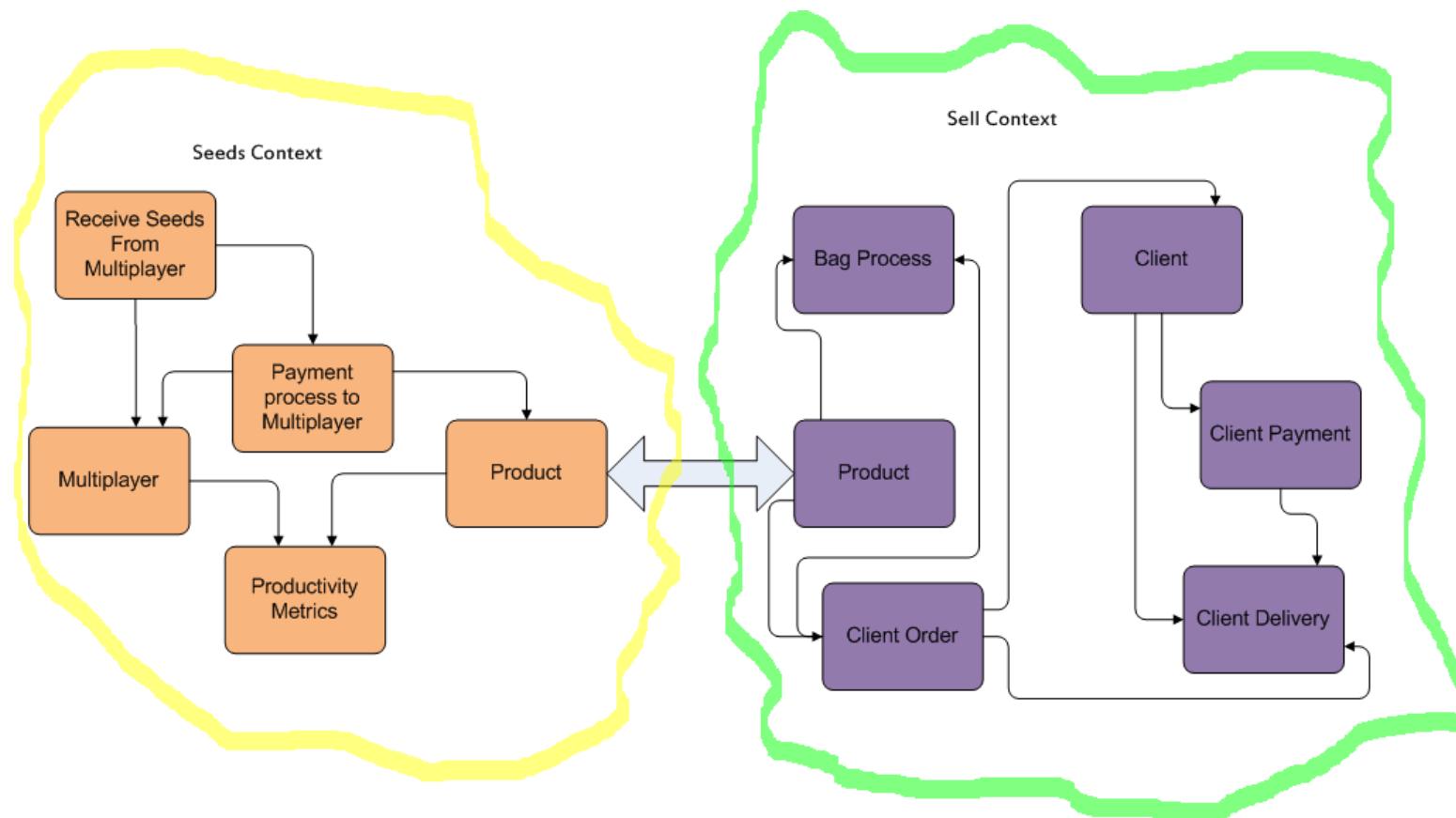
Domain-Driven Design

Strategic Design (Business Process Map)



Domain-Driven Design

Context Map - (Define a Aplicação de um modelo em particular e o bounded context do Microservices)



Domain-Driven Design

Layers:

User Interface/API:

Responsável por mostrar a informação para o usuário e interpretar os seus comandos. os atores podem ser usuários ou outros sistemas.

Application Layer:

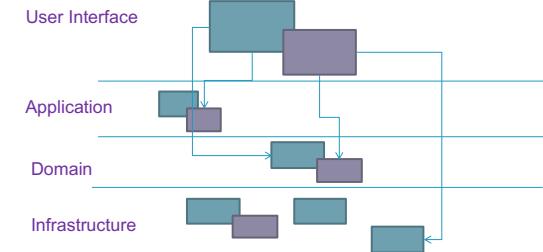
Camada que deve ser mantida muito pequena, sem regras de negócio ou conhecimento. Somente coordena tarefas e delega trabalhos para as demais camadas.

Domain Layer:

Responsável por demonstrar os conceitos de business e regras.

Infrastructure Layer:

Provê capacidades técnicas para suportar as demais camadas.



Domain-Driven Design

"Depois de encontrar seus Bounded Context em seu domínio, eles se tornam excelentes candidatos a Microservices."

“Portanto, se nossos limites de serviço se alinharem aos Bounded Context em nosso domínio e nossos microservices representarem esses bounded Context, teremos um excelente começo para garantir que nossos microservices tenham baixo acoplamento e fortemente coesos.”

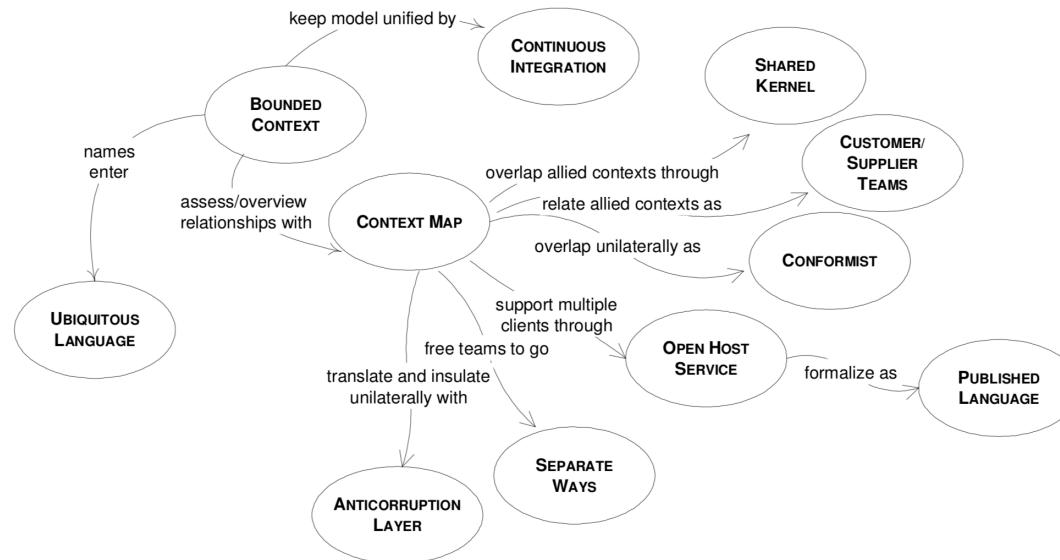
Model Integrity

Bounded Context: Context maps bounded em um model, com os seus relacionamentos.

Patterns para Context Maps: (Contextos com papéis definidos e seus relacionamentos.)

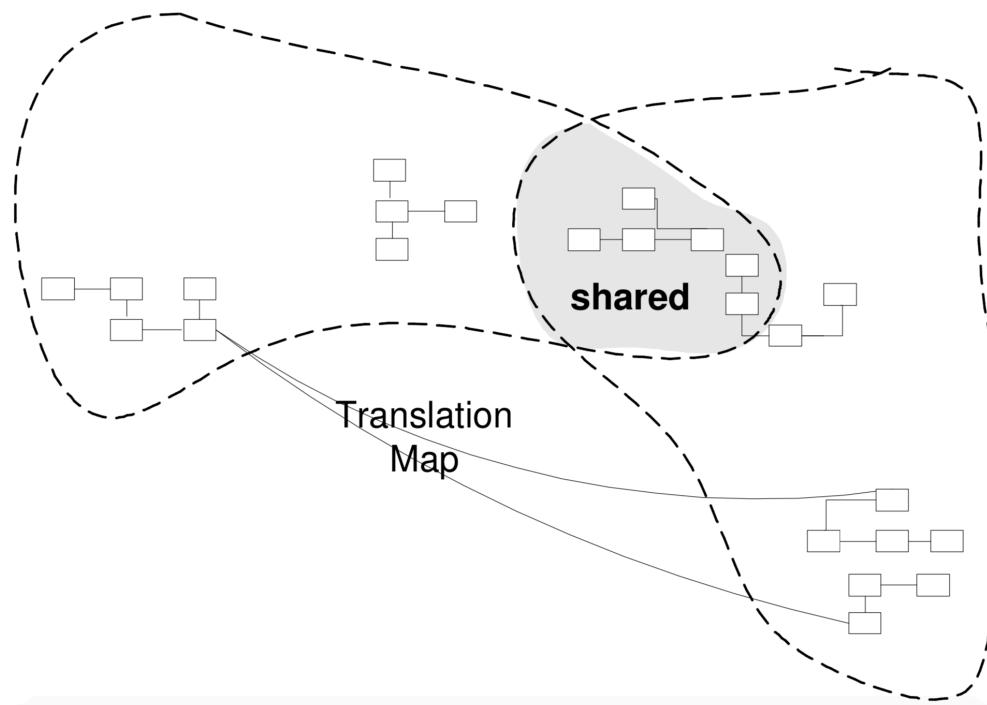
Upstream context Contexto que influênciia outro contexto.

Downstream context é influenciado por outro context.



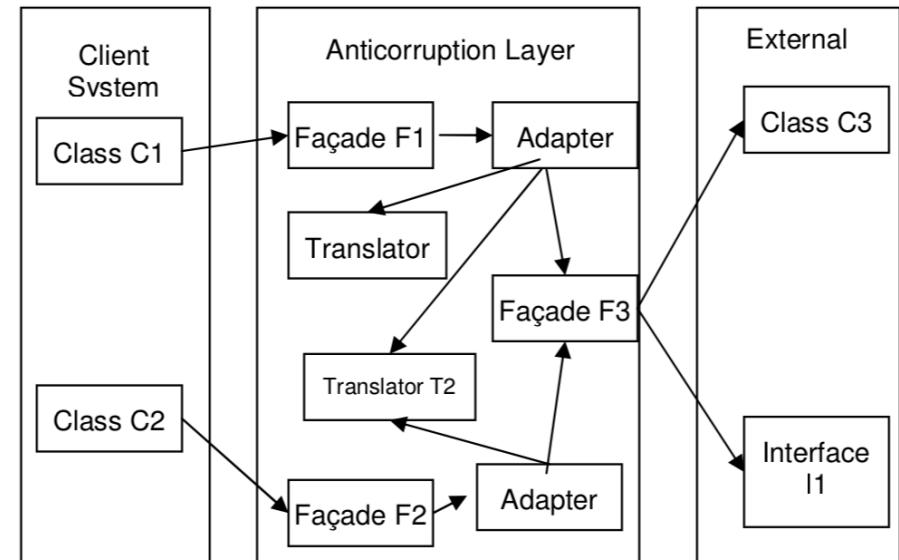
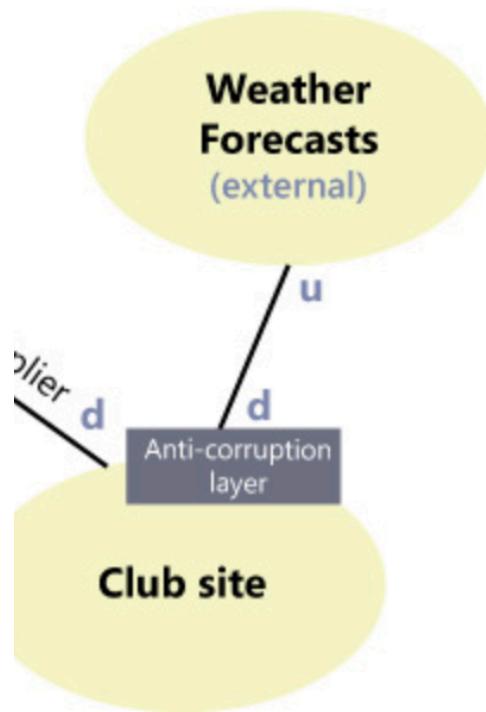
Model Integrity - Shared Kernel

Shared Kernel: Objetivo é reduzir a duplicação mas de responsabilidade de múltiplos times.



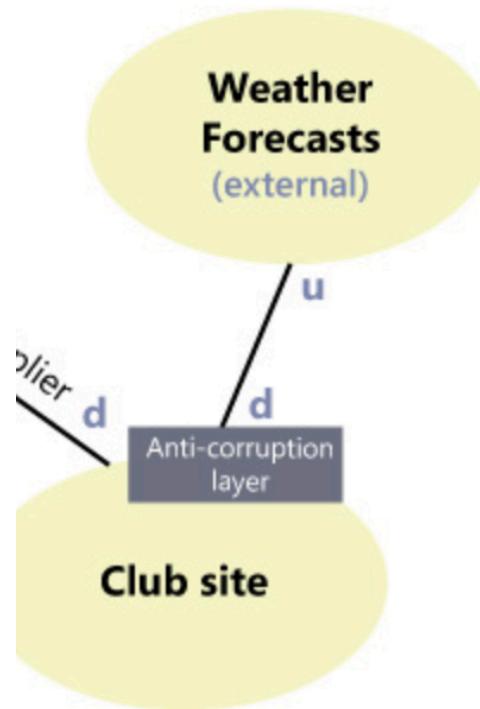
Model Integrity - Anticorruption Layer (ACL)

No Anticorruption Layer uma camada extra de código é implementada que esconde do downstream qualquer mudança implementada no upstream.



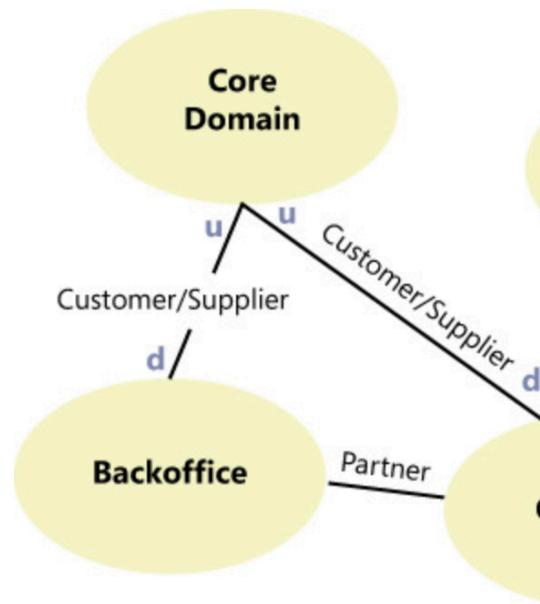
Model Integrity - Conformist

No conformist o downstream simplesmente aceita qualquer mudança no upstream. Recebendo normalmente dados não necessários. Implementação light do ACL.



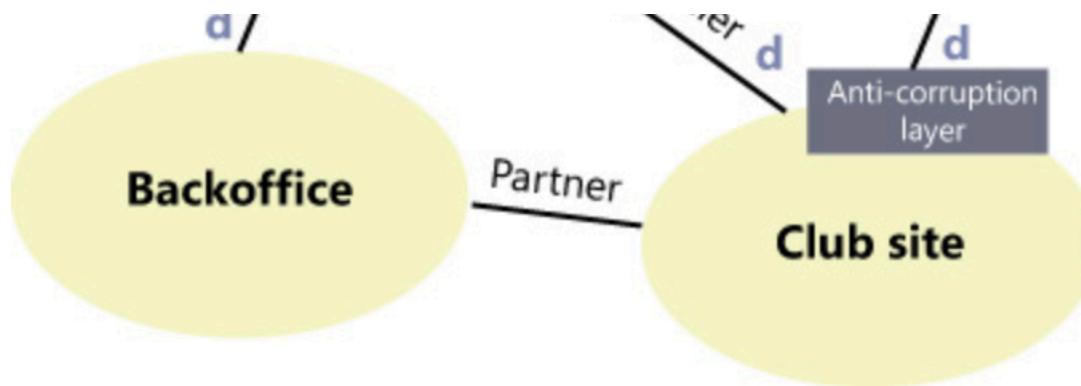
Model Integrity - Customer Supplier

Customer Supplier: Relacionamento clássico entre upstream e downstream e o supplier é o upstream mas os times trabalham juntos para que dados evitando que dados desnecessários sejam enviados.



Model Integrity - Partnership

No Partnership os dois contextos são desenvolvidos de forma independente, sem código compartilhado, os dois são upstream e downstream context ao mesmo tempo. Portanto um não pode ignorar a existência do outro nas alterações.



Nosso Case

Case para o Curso

Frankiling é um cara inovador, sempre a frente no seu tempo de tanto estudar resolveu ter uma escola. A escola de Frankiling não poderia ser uma escola normal era uma escola digital. Chamada Fkiling School.

Após testar vários modelos de ensino web e trabalhar junto ao sebrae Frankiling chegou ao modelo ideal. Atualmente conta com + 4000 alunos mês que pagam R\$ 20,00 mensal e tem acesso a uma educação básica complementar como (Curso de excel, auto-ajuda, exoterismo, cultura religiosa, histórias da bíblia, Elétrica básica e etc) as aulas são previamente gravadas e disponibilizadas para o aluno. Cada professor ganha 1 real por aula. Frankiling mantinha todo o modelo através de infinitas planilhas de excel e chegou a hora de colocar uma espinha dorsal digital na empresa.

A EDD será responsável por gerenciar os alunos, os professores, pagamentos, e cursos.

Usando todo o conhecimento adquirido, desenvolva:

- Business Process Map.
- Destaque as Fitness Functions.
- Context Map.
- Selecionando os bounded Context e destacando os Microservices que vamos desenvolver.

Estórias

Alunos:

Efetuam o cadastro via web, com poucas informações: {Nome, nascimento, cidade, estado, nível educacional, profissão, Cartão de Crédito}.

Alunos podem realizar o CRUD das suas informações e dispõem de um painel com os cursos que querem executar, dar nota e gerar diploma e a possibilidade de enviar uma mensagem.

Cursos:

Cursos são cadastrados com informações básicas: {Nome, Descrição e link para o video e professor}

O CRUD de cursos também pode ser realizado e cada curso tem uma nota aplicada pelo aluno.

Professor:

Dispõe de uma lista de cursos que é responsável por resolver as mensagens e uma conta com o resumo financeiro que mostra o quanto ganhou por curso executado.

Pagamentos-Financeiro:

Professores e alunos acessam esta funcionalidade para pagar e receber pelos cursos.