

```
{  
  "Version": "2021-04-19",  
  "Statement": [  
    {  
      "Sid": "People:Ops",  
      "Action": "",  
      "Resource":  
        "arn:aws:Ibrahim-Cesar:sa-east-1"  
    }  
  ]  
}
```



@ibrahimcesar

<https://ibrahimcesar.cloud>

```
{  
  "Version": "2021-04-19",  
  "Statement": [  
    {  
      "Sid": "People:Ops",  
      "Action": "",  
      "Resource":  
        "arn:aws:Ibrahim-Cesar:sa-east-1"  
    }  
  ]  
}
```



@ibrahimcesar

<https://ibrahimcesar.cloud>

0 Intro

1 Qual o problema?

2 Visão estratégica

3 Táticas

4 Pessoas

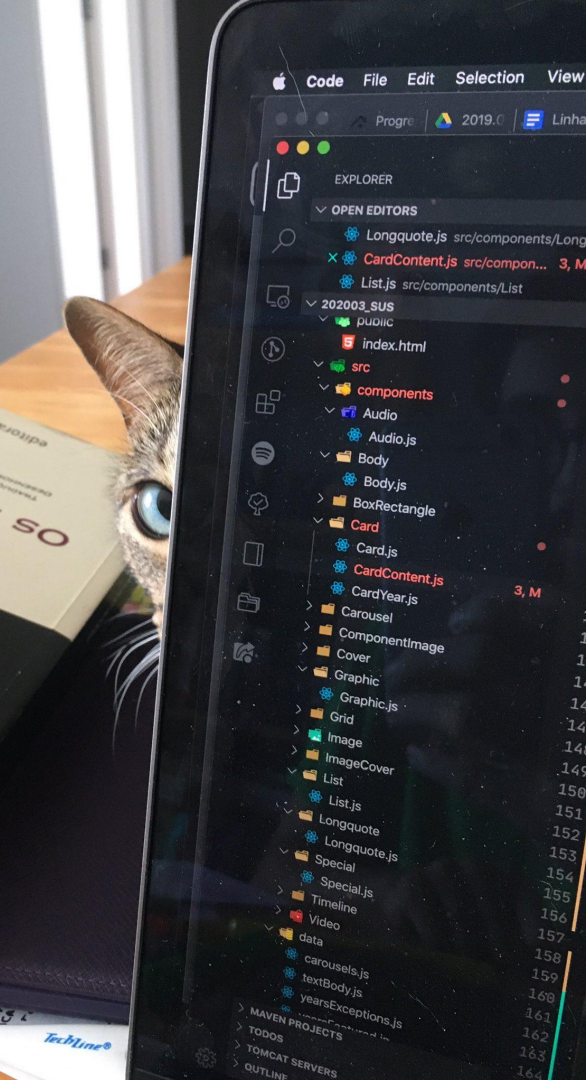
0 Intro

1 Qual o problema?

2 Visão estratégica

3 Tácticas

4 Pessoas



AWS community builder

Ibrahim Cesar Nogueira Bevilacqua

Ele/dele

Diretor de tecnologia **Nexo**

Nexo Jornal, Gama Revista e
Nexo Políticas Públicas

AWS Solutions Architect - Associate


AWS Certified Cloud Practitioner

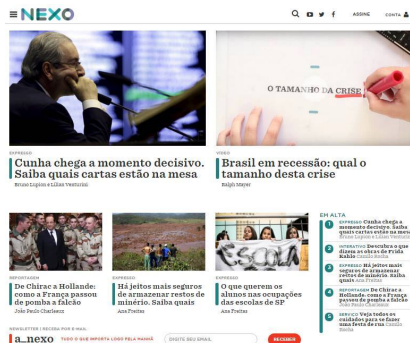
Desenvolvedor web / cloud / serverless

@ibrahimcesar

<https://ibrahimcesar.cloud>



 @ibrahimcesar



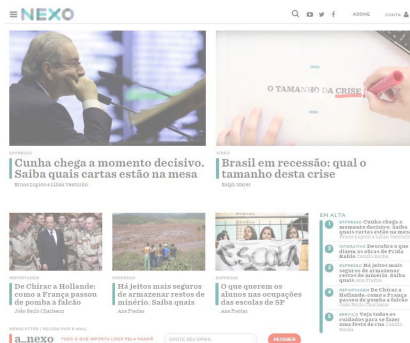
<NexoJornal />
<https://nexojornal.com.br>
 24 de novembro, 2015
 10h23



<GamaRevista />
<https://gamarevista.com.br>
 29 de março, 2020
 00h00



<NexoPolíticasPúblicas />
<https://pp.nexojornal.com.br>
 02 de julho, 2020
 12h15



<NexoJornal />

<https://nexojornal.com.br>

24 de novembro, 2015

10h23



<GamaRevista />

<https://gamarevista.uol.com.br>

29 de março, 2020

00h00

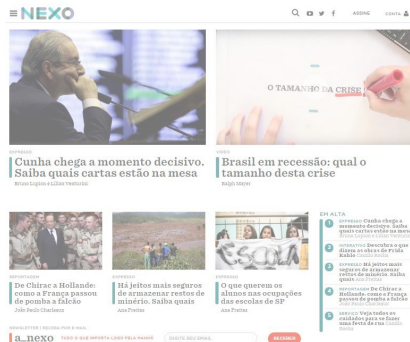


<NexoPolíticasPúblicas />

<https://pp.nexojornal.com.br>

02 de julho, 2020

12h15



<NexoJornal />

<https://nexojornal.com.br>

24 de novembro, 2015

10h23



<GamaRevista />

<https://gamarevista.com.br>

29 de março, 2020

00h00



<NexoPolíticasPúblicas />

<https://pp.nexojornal.com.br>

02 de julho, 2020

12h15

0 Intro

1 Qual o problema?

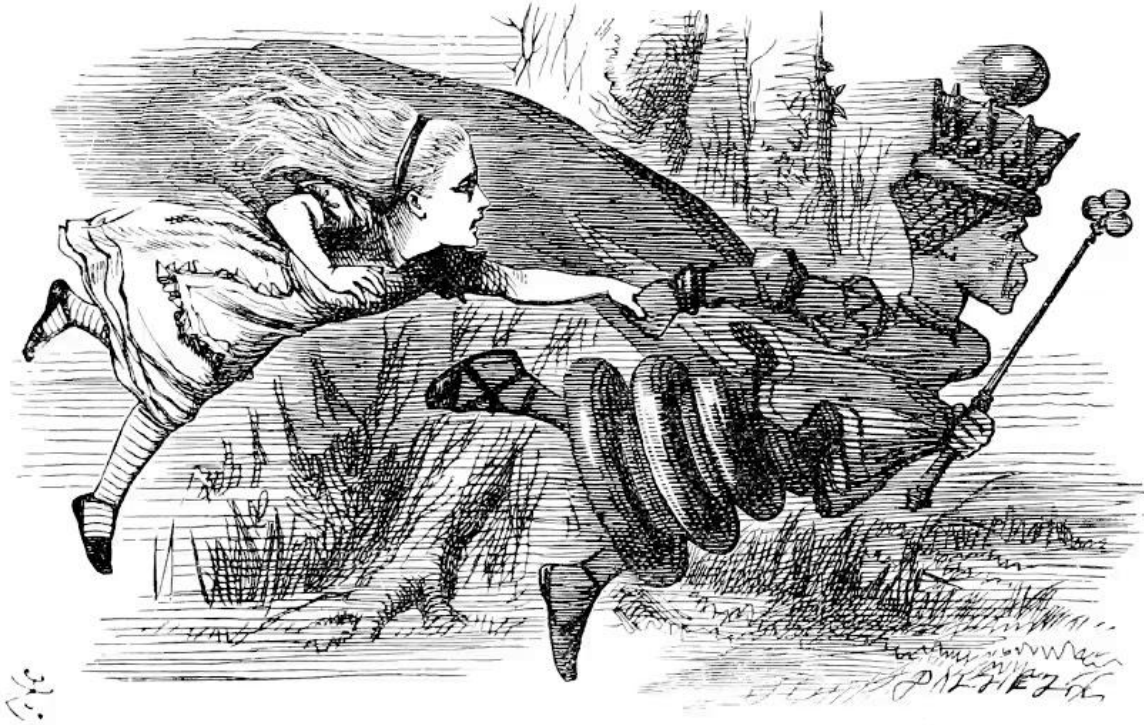
2 Visão estratégica

3 Tácticas

4 Pessoas

"Now, here, you see, it takes all the running you can do, to keep in the same place.
If you want to get somewhere else, you must run at least twice as fast as that!"

Lewis Carroll's Red Queen in Through the Looking-Glass



HOW DO COMMITTEES INVENT?

by MELVIN E. CONWAY

That kind of intellectual activity which creates a useful whole from its diverse parts may be called the *design* of a *system*. Whether the particular activity is the creation of specifications for a major weapon system, the formation of a recommendation to meet a social challenge, or the programming of a computer, the general activity is largely the same.

Typically, the objective of a design organization is the creation and assembly of a document containing a coherently structured body of information. We may name this information the *system design*. It is typically produced for a sponsor who usually desires to carry out some activity guided by the system design. For example, a public official may wish to propose legislation to avert a recurrence of a recent disaster, so he appoints a team to explain the catastrophe. Or a manufacturer needs a new product and designates a product planning activity to specify what should be introduced.

The design organization may or may not be involved in the construction of the system it designs. Frequently, in public affairs, there are policies which discourage a group's acting upon its own recommendations, whereas, in private industry, quite the opposite situation often prevails.

It seems reasonable to suppose that the knowledge that one will have to carry out one's own recommendations or that this task will fall to others, probably affects some design choices which the individual designer is called upon to make. Most design activity requires continually making choices. Many of these choices may be more than design decisions; they may also be personal decisions the designer makes about his own future. As we shall see later, the incentives which exist in a conventional management environment can motivate choices which subvert the intent of the sponsor.

design organization criteria

ing a design team means that certain design decisions have already been made, explicitly or otherwise. Given any design team organization, there is a class of design alternatives which cannot be effectively pursued by such an organization because the necessary communication paths do not exist. Therefore, there is no such thing as a design group which is both organized and unbiased.

Once the organization of the design team is chosen, it is possible to delegate activities to the subgroups of the organization. Every time a delegation is made and somebody's scope of inquiry is narrowed, the class of design alternatives which can be effectively pursued is also narrowed.

Once scopes of activity are defined, a coordination problem is created. Coordination among task groups, although it appears to lower the productivity of the individual in the small group, provides the only possibility that the separate task groups will be able to consolidate their efforts into a unified system design.

Thus the life cycle of a system design effort proceeds through the following general stages:

1. Drawing of boundaries according to the ground rules.
2. Choice of a preliminary system concept.
3. Organization of the design activity and delegation of tasks according to that concept.
4. Coordination among delegated tasks.
5. Consolidation of subdesigns into a single design.

It is possible that a given design activity will not proceed straight through this list. It might conceivably reorganize upon discovery of a new, and obviously superior, design concept; but such an appearance of uncertainty is unflattering, and the very act of voluntarily abandoning a creation is painful and expensive. Of course, from the



{ 1968-04 }

HOW DO COMMITTEES INVENT?

by MELVIN E. CONWAY

That kind of intellectual activity which creates a useful whole from its diverse parts may be called the *design* of a *system*. Whether the particular activity is the creation of specifications for a major weapon system, the formation of a recommendation to meet a social challenge, or the programming of a computer, the general activity is largely the same.

Typically, the objective of a design organization is the creation and assembly of a document containing a coherently structured body of information. We may name this information the *system design*. It is typically produced for a sponsor who usually desires to carry out some activity guided by the system design. For example, a public official may wish to propose legislation to avert a recurrence of a recent disaster, so he appoints a team to explain the catastrophe. Or a manufacturer needs a new product and designates a product planning activity to specify what should be introduced.

The design organization may or may not be involved in the construction of the system it designs. Frequently, in public affairs, there are policies which discourage a group's acting upon its own recommendations, whereas, in private industry, quite the opposite situation often prevails.

It seems reasonable to suppose that the knowledge that one will have to carry out one's own recommendations or that this task will fall to others, probably affects some design choices which the individual designer is called upon to make. Most design activity requires continually making choices. Many of these choices may be more than design decisions; they may also be personal decisions the designer makes about his own future. As we shall see later, the incentives which exist in a conventional management environment can motivate choices which subvert the intent of the sponsor.

design organization criteria

design group and the system are identical. In the case where some group designed more than one subsystem we find that the structure of the design organization is a collapsed version of the structure of the system, with the subsystems having the same design group collapsing into one node representing that group.

This kind of a structure-preserving relationship between two sets of things is called a *homomorphism*. Speaking as a mathematician might, we would say that there is a homomorphism from the linear graph of a system to the linear graph of its design organization.

systems image their design groups

It is an article of faith among experienced system designers that given any system design, someone someday will find a better one to do the same job. In other words, it is misleading and incorrect to speak of *the* design for a specific job, unless this is understood in the context of space, time, knowledge, and technology. The humility which this belief should impose on system designers is the only appropriate posture for those who read history or consult their memories.

The design progress of computer translators of programming languages such as FORTRAN and COBOL is a case in

* This claim may be viewed several ways. It may be trivial, hinging on the definition of meaningful negotiation. Or, it may be the result of the observation that one design group almost never will compromise its own design to meet the needs of another group unless absolutely imperative.

30

4. Coordination among delegated tasks.
5. Consolidation of subdesigns into a single design.

It is possible that a given design activity will not proceed straight through this list. It might conceivably reorganize upon discovery of a new, and obviously superior, design concept; but such an appearance of uncertainty is unflattering, and the very act of voluntarily abandoning a creation is painful and expensive. Of course, from the

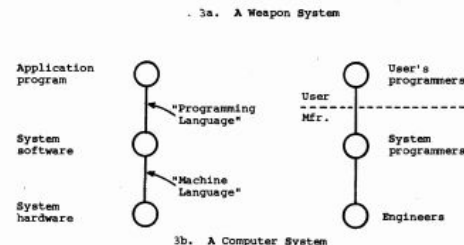


Figure 3 Two examples of identity of structure between a system and its design organization.

Figs. 3a and 3b

respective needs. After great effort they produced a copy of their organization chart. (See Fig. 3a.)

Consider the operating computer system in use solving a problem. At a high level of examination, it consists of three parts: the hardware, the system software, and the application program. (See Fig. 3b.) Corresponding to these subsystems are their respective designers: the computer manufacturer's engineers, his system programmers, and the user's application programmers. (Those rare instances where the system hardware and software tend to cooperate rather than merely tolerate each other are associated with

DATAMATION

SOONER

ANTIPATTERNS AND PATTERNS

SAFER

FOR BUSINESS AGILITY

HAPPIER

**Jonathan
Smart**

*with Zsolt Berend,
Myles Ogilvie
and Simon Rohrer*

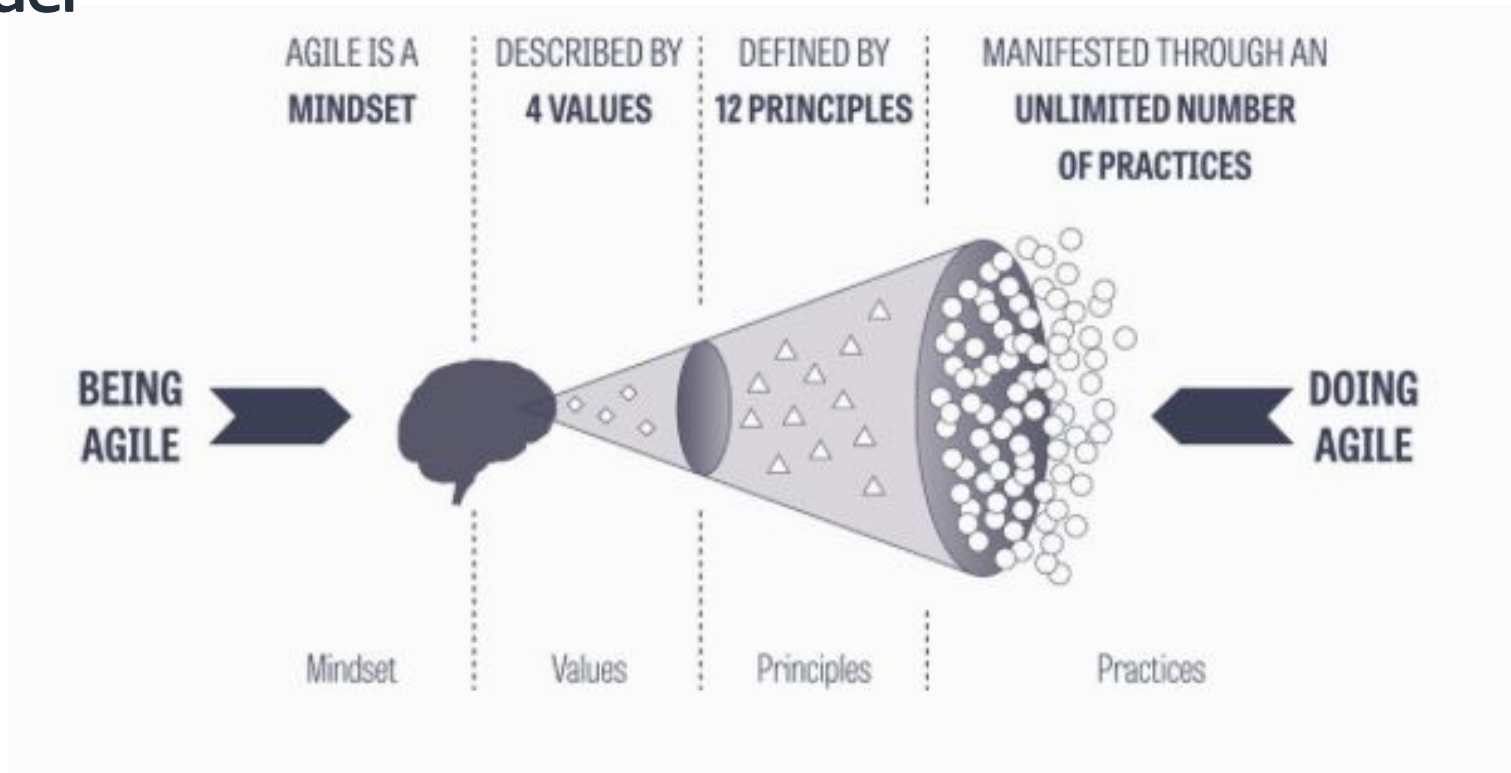
*"Sooner Safer Happier is a
must-read for any business
leader looking to succeed
in the digital age."*

*—David Silverman,
Co-Author of Team of Teams*

Tipos de trabalho



O espectro do Ágil



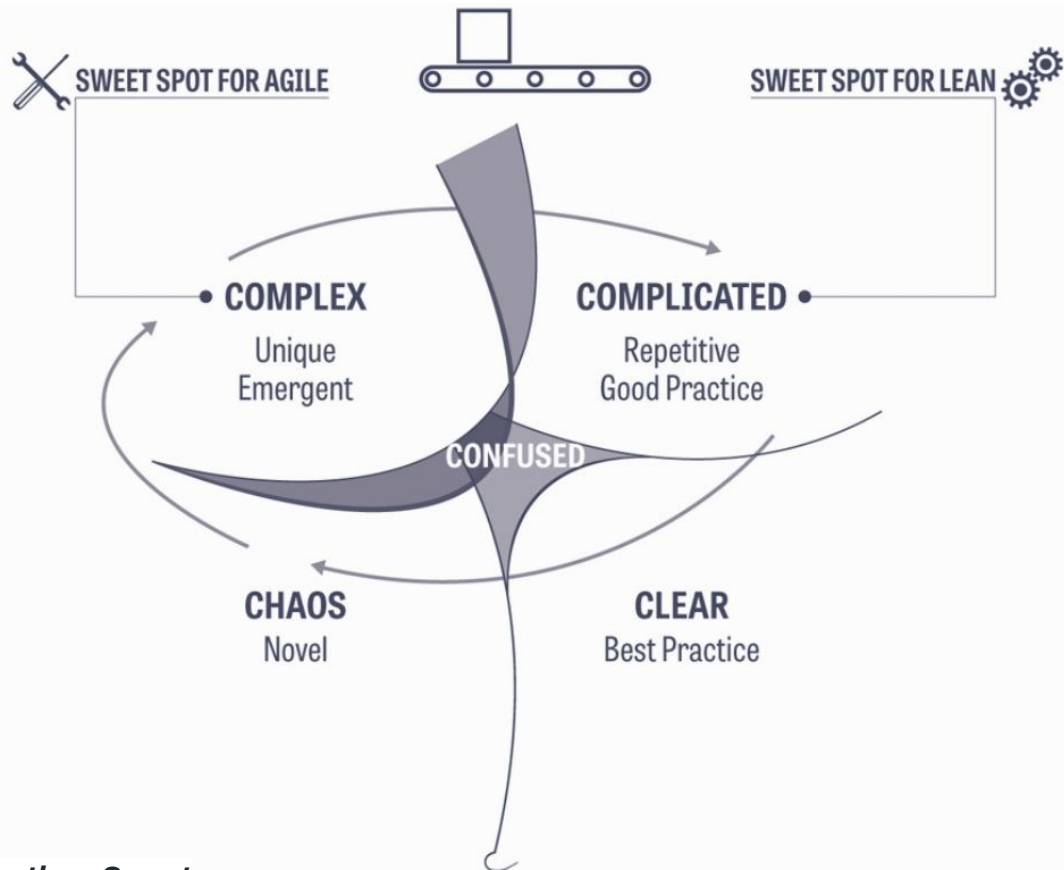
{ Em "The Unicorn Project", Gene Kim define cinco ideais de **DevOps**:

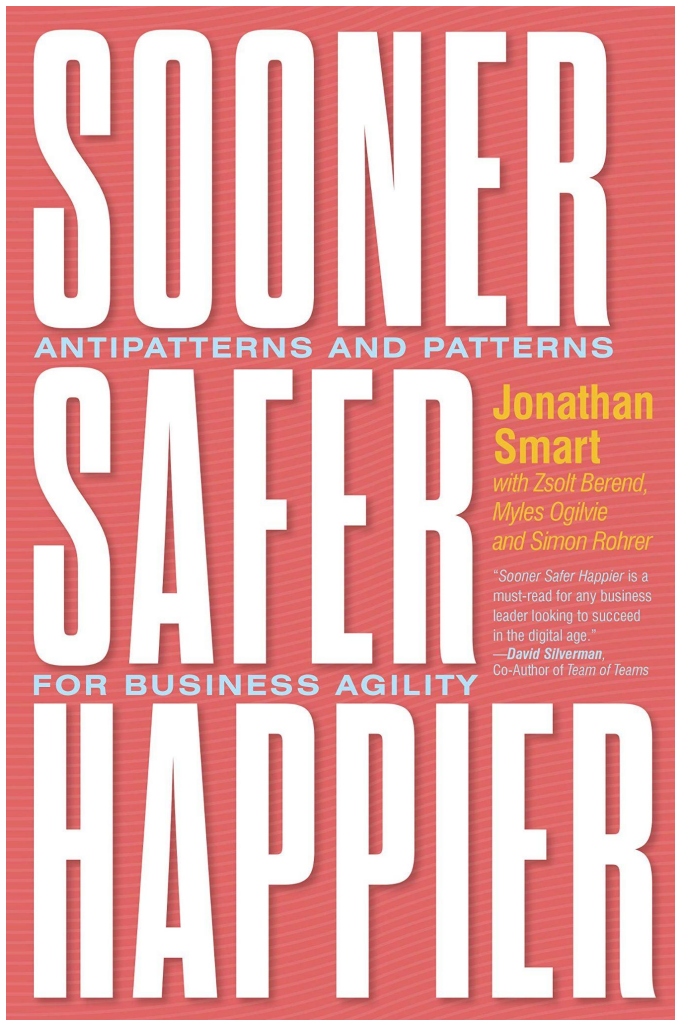
1. **Localidade e Simplicidade**: aliviar dependências entre times e sistemas.
2. **Foco, Flow, e Prazer**: o tranquilo flow de trabalho que permite foco e prazer.
3. **Melhoria do trabalho diário**: continuamente melhorar e eliminar débito técnico.
4. **Segurança psicológica**: um dos principais fatores na performance do time; possibilita a melhoria.
5. **Foco no cliente**: otimize o valor para o cliente, não para um papel específico em funções / verticais. }

{ Em "**Lean Thinking**", Daniel Jones e James Womack definem os cinco princípios lean:

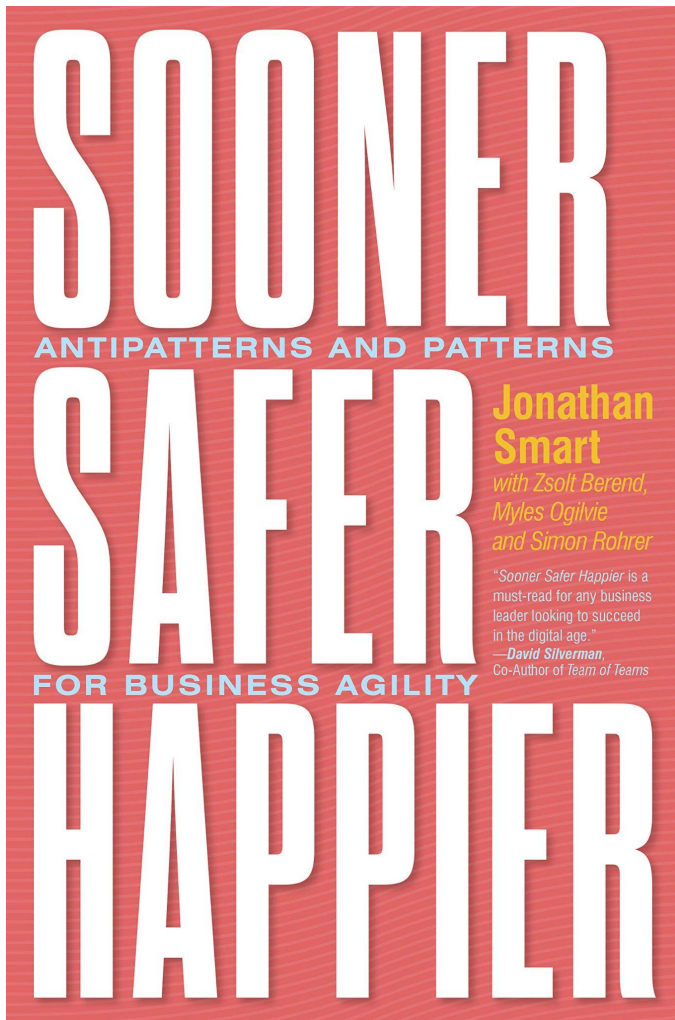
1. **Valor:** determinar o valor do ponto de vista do cliente.
 2. **Cadeia de valor:** identificar a cadeia de valor e todos os passos, do conceito ao dinheiro.
 3. **Fluxo:** limite trabalho em progresso (WIP); estabilize o fluxo; foque em *lead time*, eficiência de fluxo e entrega, eliminando impedimentos.
 4. **Pull:** mover de um sistema de trabalho baseado em empurrar (*push*) para um sistema de trabalho baseado em "puxar" (*pull*); vá até a capacidade do sistema de trabalho e não produza em excesso.
 5. **Perfeição:** a busca sem descanso por perfeição.
- }

Tipos de trabalho e abordagens





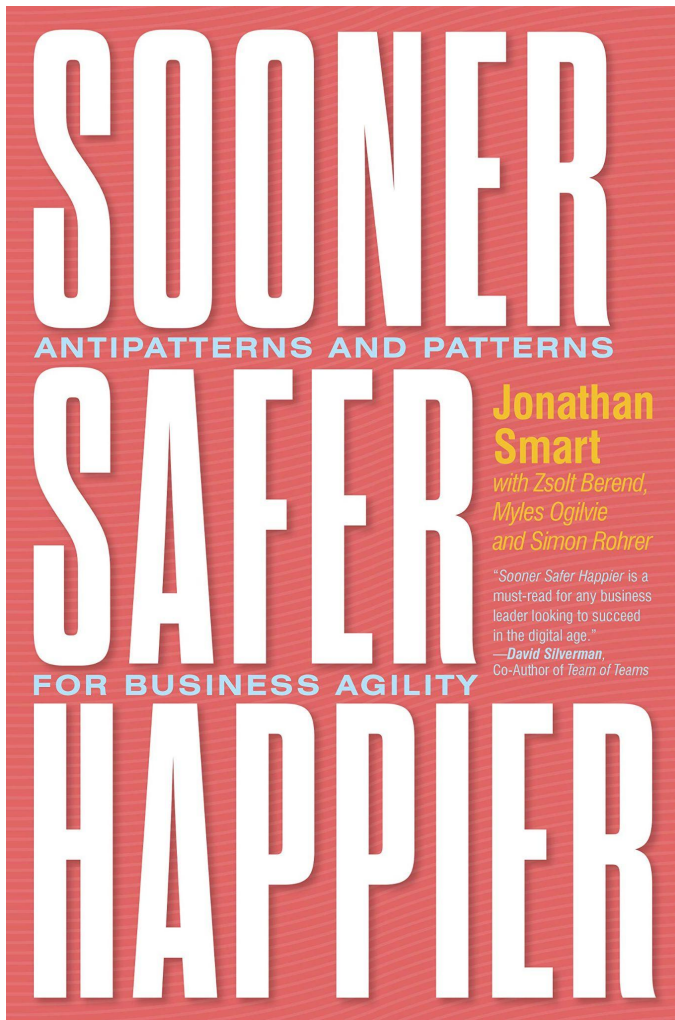
{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? }



{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

}

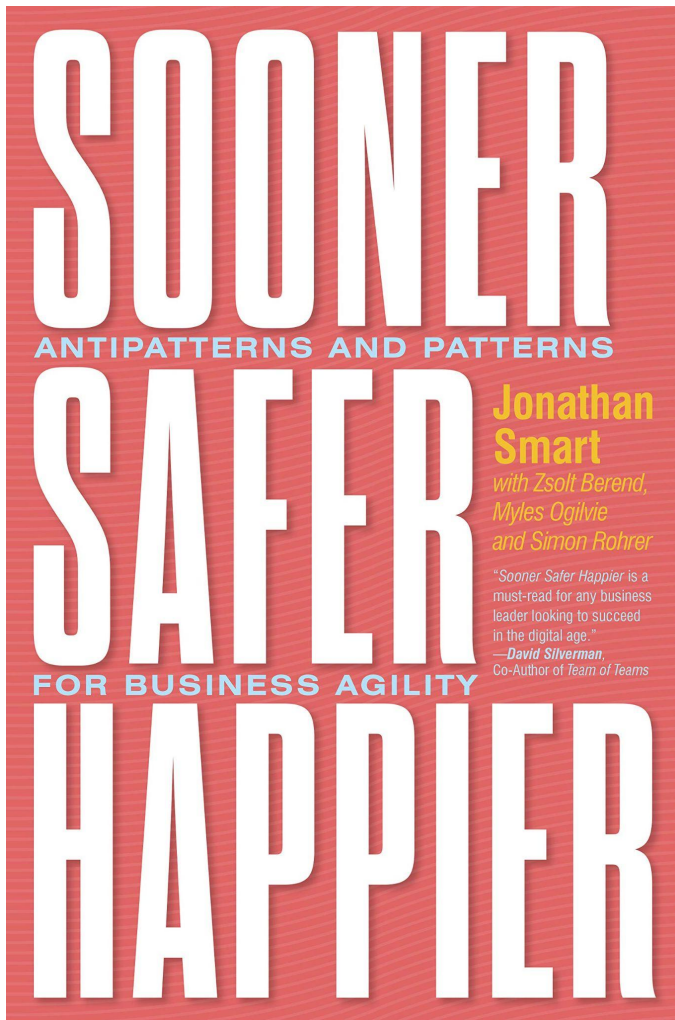


{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade. (E não necessariamente, **Ágil®**)

}



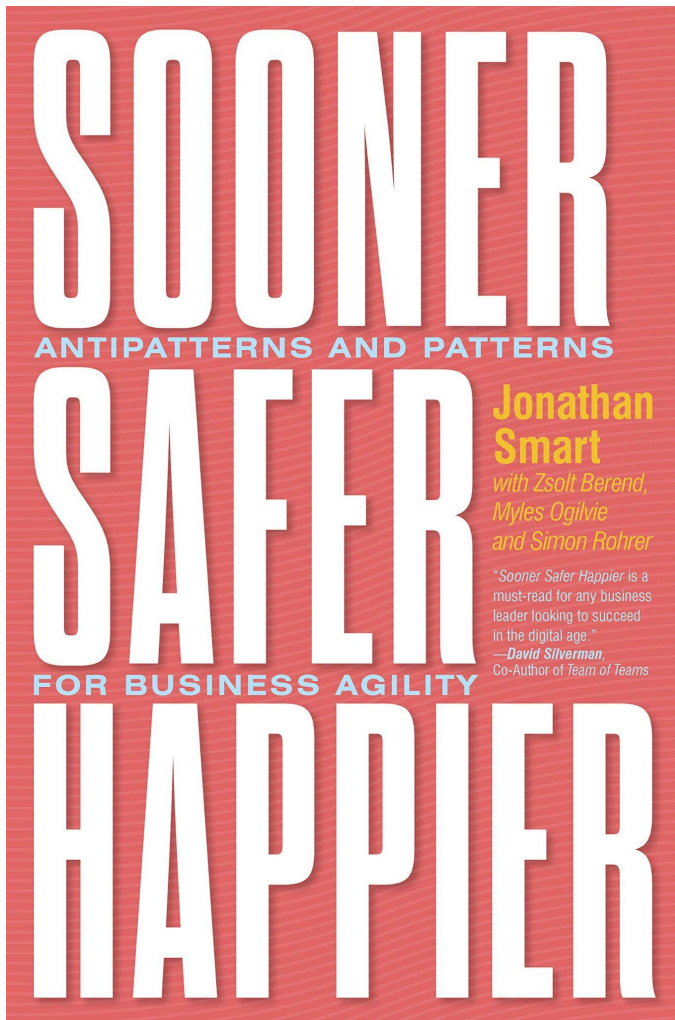
{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade.

Foque-se em:

Better Value Sooner Safer Happier
}



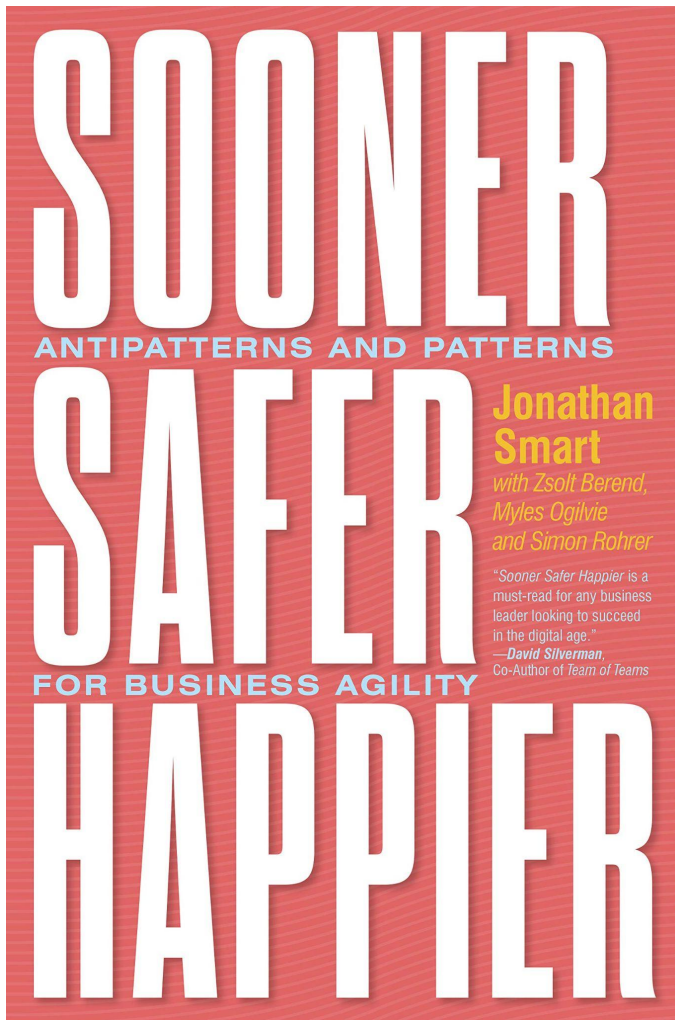
{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade.

Foque-se em:

Mais valor Mais cedo Mais seguro Mais feliz
}



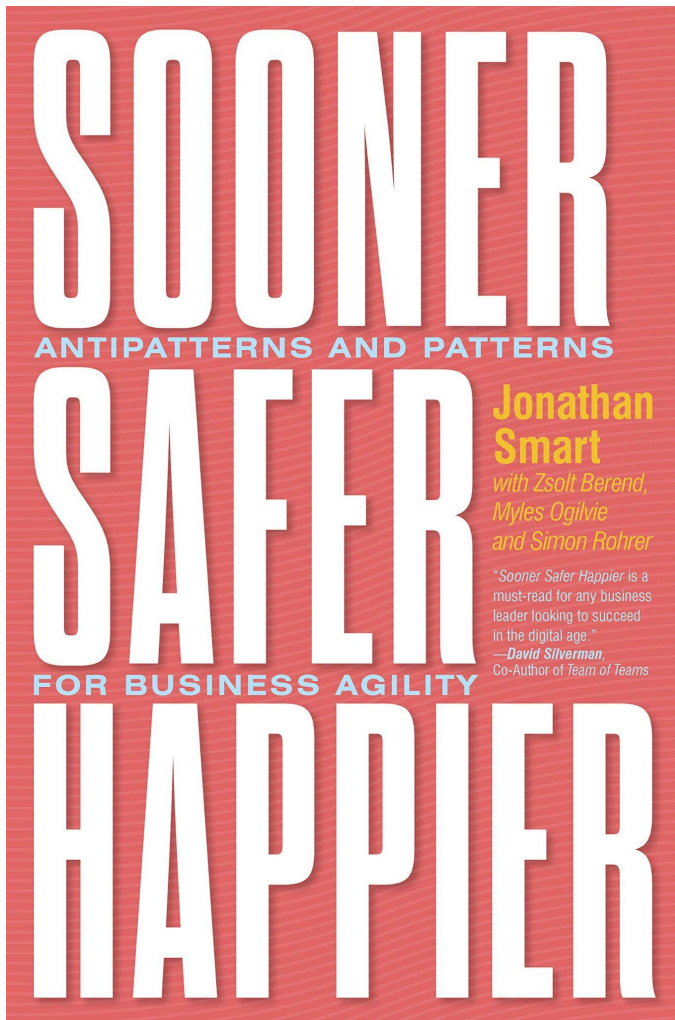
{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade.

Foque-se em:

Mais valor **Mais cedo** Mais seguro Mais feliz
}



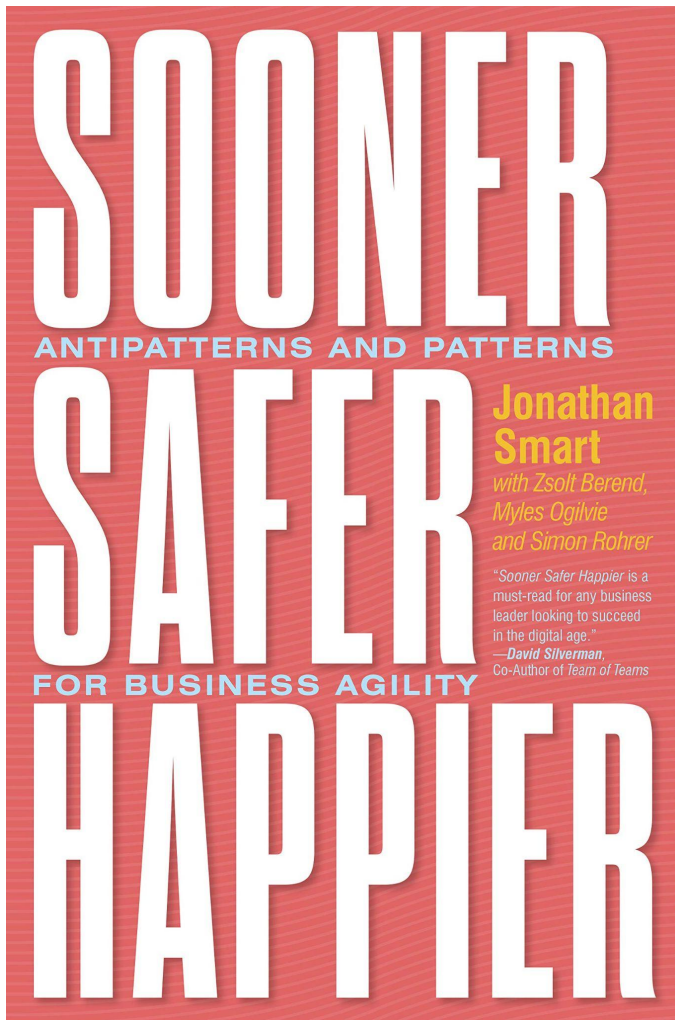
{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade.

Foque-se em:

Mais valor Mais cedo **Mais seguro** Mais feliz
}



{ Você quer fazer ou está atualmente conduzindo uma Transformação Ágil, Lean ou DevOps? Se sim, meu conselho é:

Pare.

Ao invés, foque-se nos resultados que deseja atingir. Então você atingirá agilidade.

Foque-se em:

Mais valor Mais cedo Mais seguro **Mais feliz**
}

0 Intro

1 Qual o problema?

2 Visão estratégica

3 Táticas

4 Pessoas

PERFORMANCE METRICS



SOFTWARE DEVELOPMENT

Lead Time



SOFTWARE DEPLOYMENT

Change Fail



SERVICE OPERATION

Availability

Deployment Frequency

Time to Restore

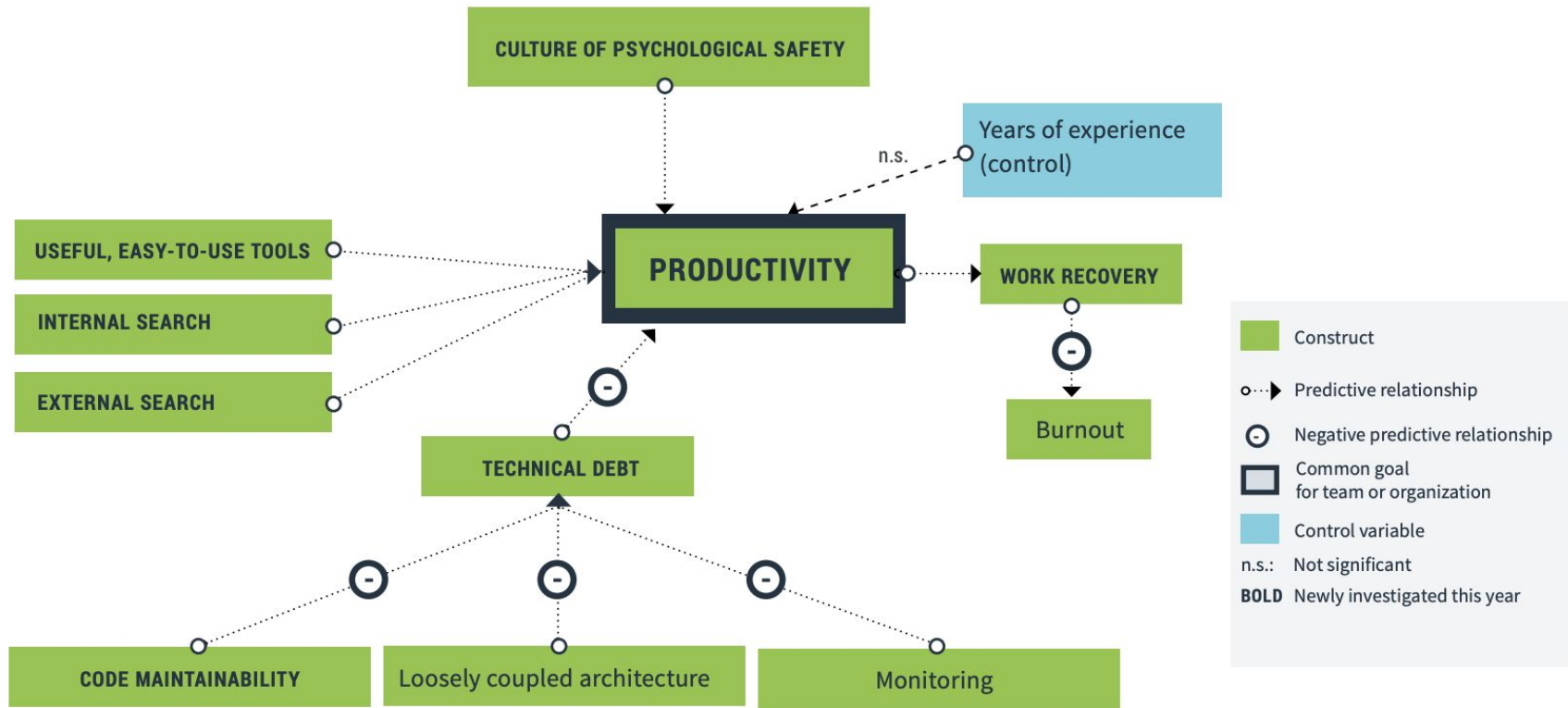
FOUR KEY METRICS

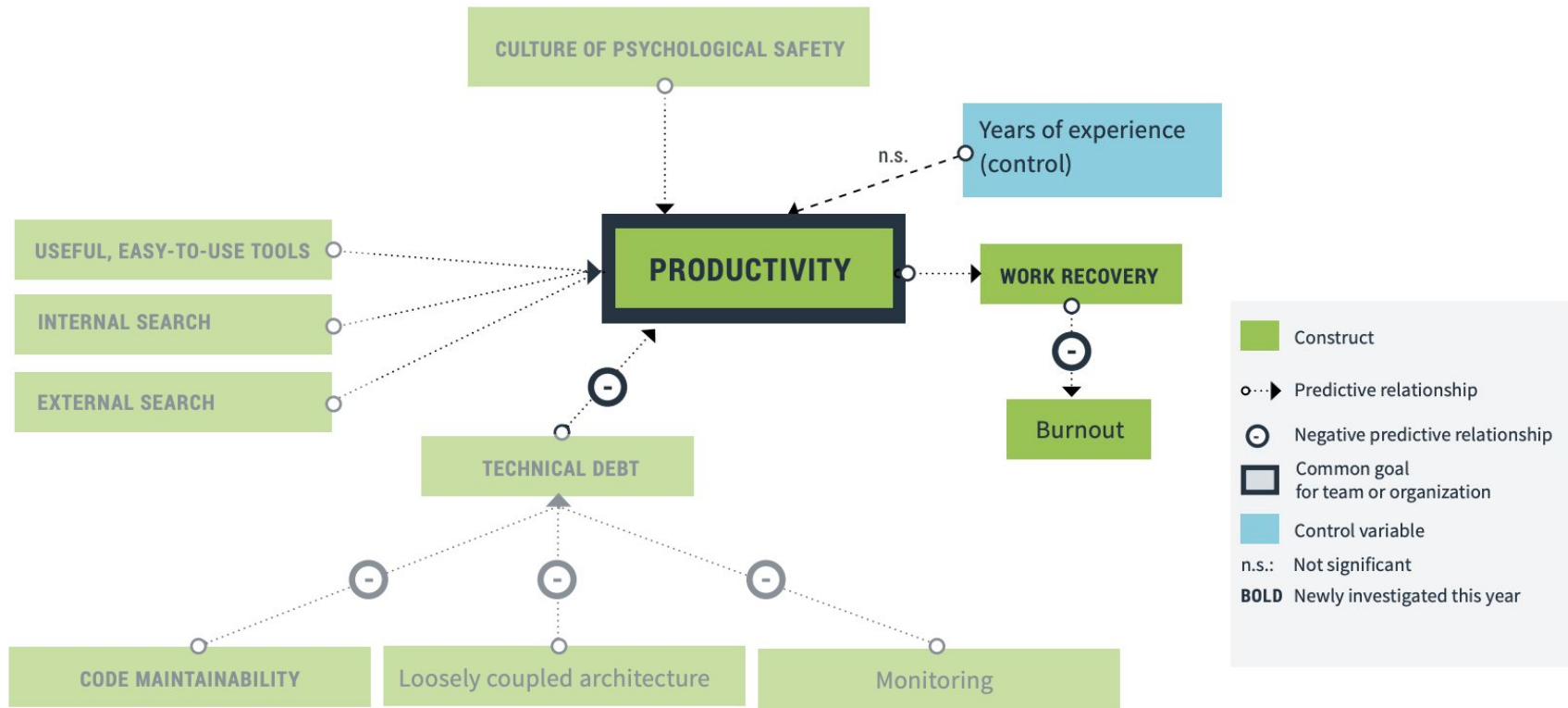
Start by identifying your goal...

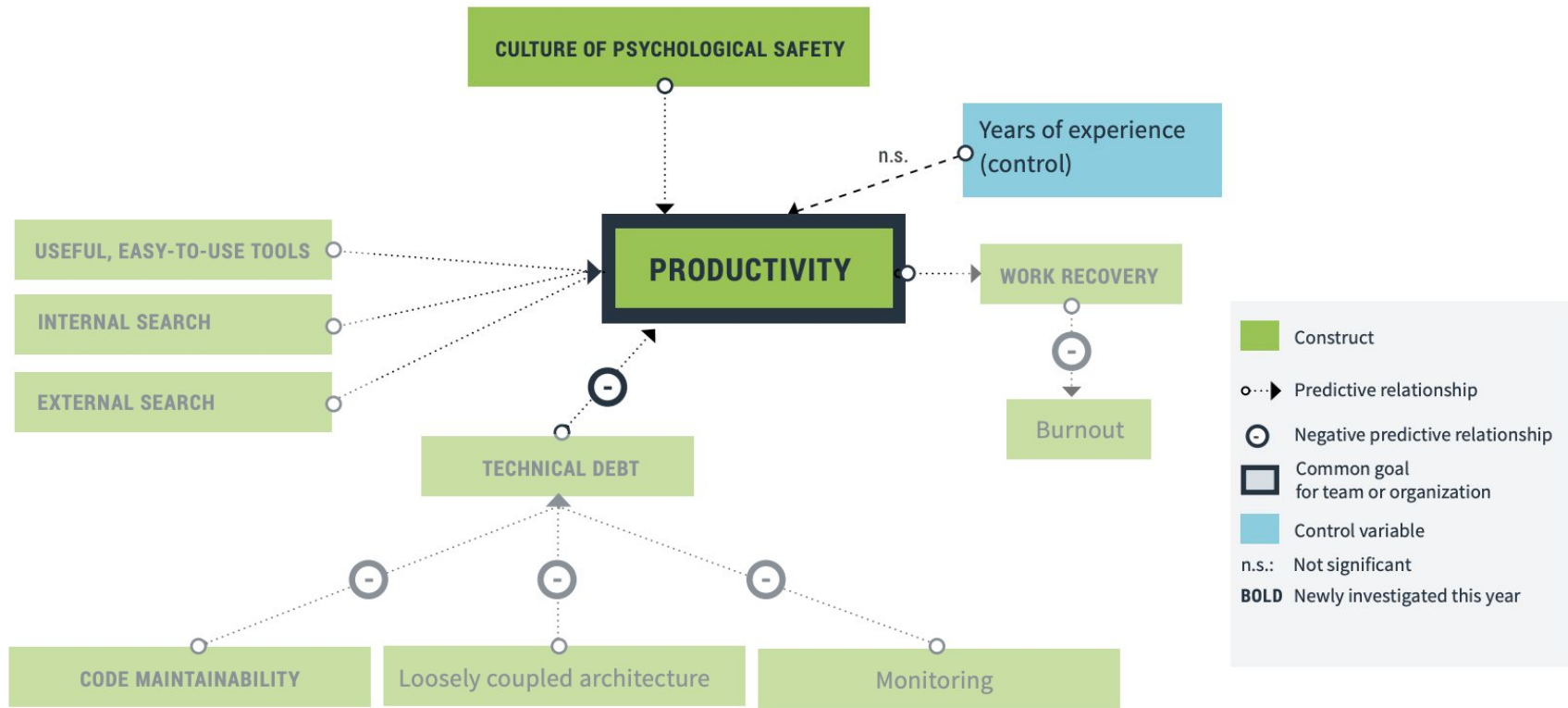


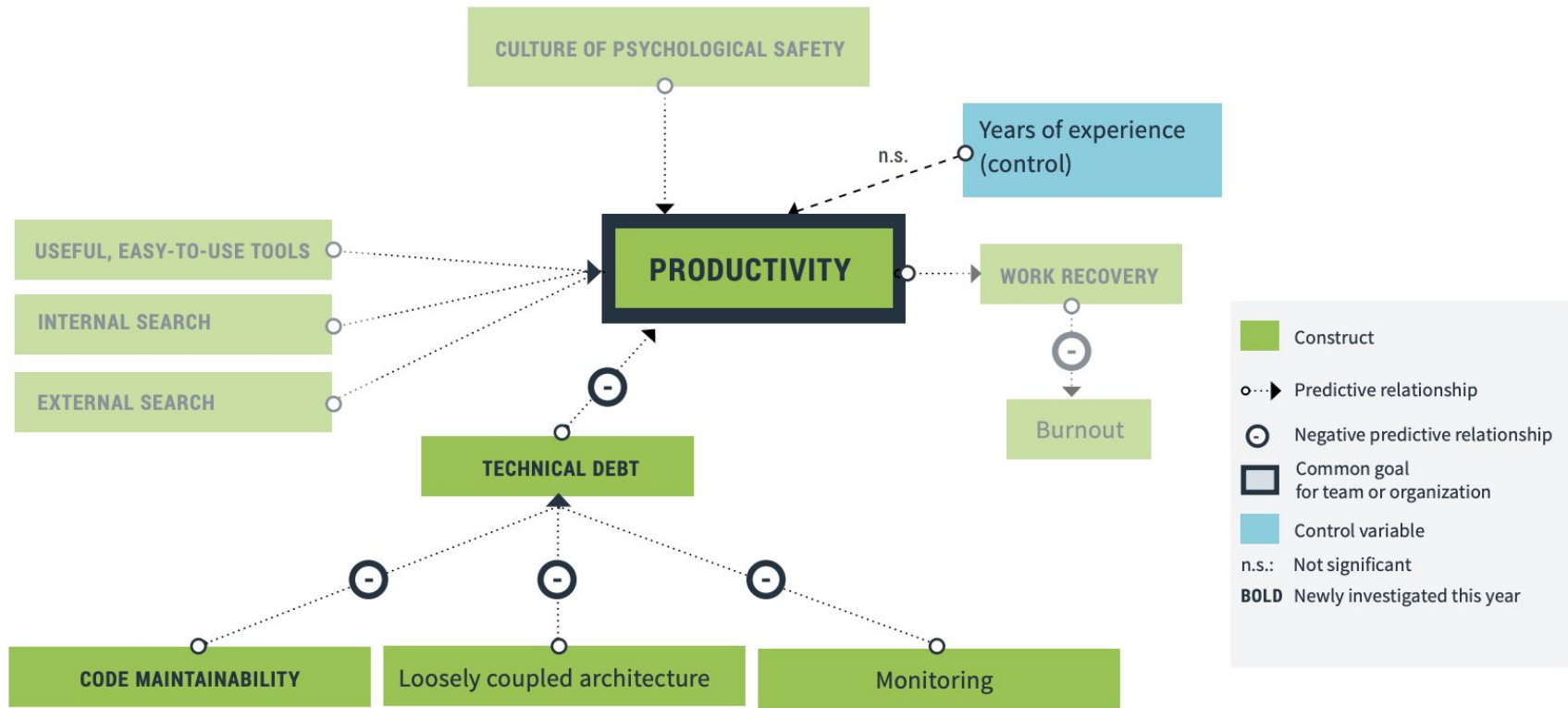
Start by identifying your goal...

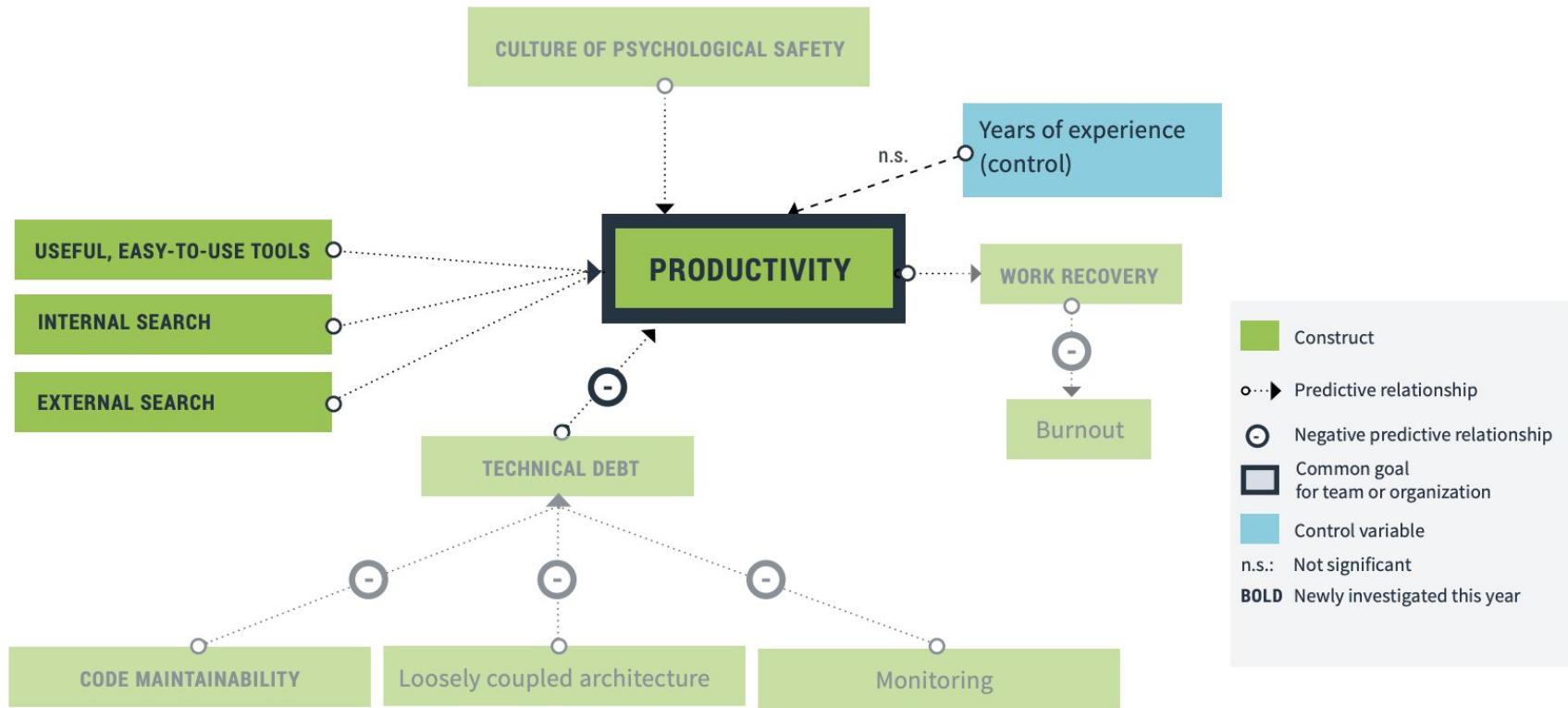












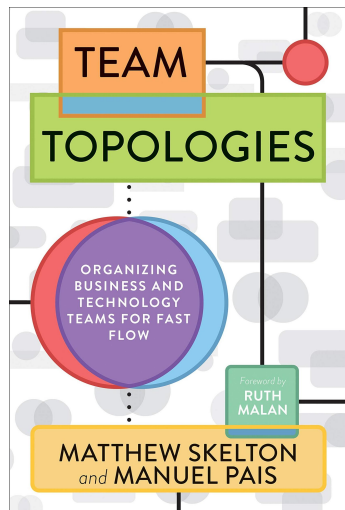
0 Intro

1 Qual o problema?

2 Visão estratégica

3 Tácticas

4 Pessoas

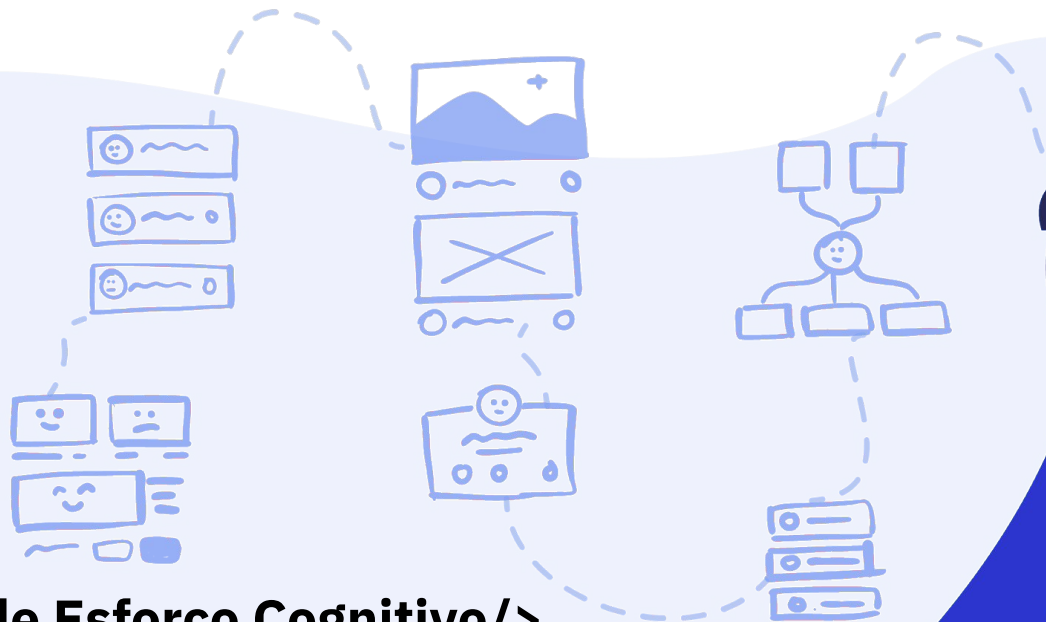


Combine sua arquitetura organizacional com sua arquitetura de software

{

*“Esqueça monolitos vs. microserviços.
Esforço cognitivo é o que importa”*

}



< Intro de Esforço Cognitivo/> **COGNITIVE LOAD 101**

Três tipos de esforço cognitivo

Esforço intrínseco < *Intrinsic cognitive load* />

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }

Esforço estranho

{ Se relaciona com aspectos do ambiente ou tarefas repetitivas ou com incertezas.

Exemplos: Como se faz mesmo o deploy?

Como eu inicializo um aplicação no Amplify? }

Esforço pertinente

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

Esforço intrínseco *< Intrinsic cognitive load />*

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }



< Deve ser **minimizado** – através de treinamento, escolhas acertadas de tecnologia para as habilidades do time, contratação, arquitetura, programação em par, *mob programming* >

s repetitivas ou com incertezas.

o Amplify? }

Esforço pertinente

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

Esforço intrínseco < Intrinsic cognitive load />

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }



< Deve ser **minimizado** – através de treinamento, escolhas acertadas de tecnologia para as habilidades do time, contratação, ferramentas externas, programação em par, *mob programming* >

USEFUL, EASY-TO-USE TOOLS

INTERNAL SEARCH

EXTERNAL SEARCH

certezas.

Esforço pertinente

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

Esforço intrínseco

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }

Esforço estranho *< Extraneous cognitive load />*

{ Se relaciona com aspectos do ambiente ou tarefas repetitivas ou com incertezas.

Exemplos: Como se faz mesmo o deploy?

Como eu inicializo um aplicação? }

Esforço pertinente

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

< Deve ser **eliminado** – através de uma cultura de DevOps, arquitetura, automatização de boas práticas de segurança, eliminar comunicações desnecessárias, processos, *runbooks*, *playbooks*, ferramentas externas, ***uso de serviços gerenciados*** >

ação do problema.
ava?

Esforço estranho

< *Extraneous cognitive load* />

- { Se relaciona com aspectos do ambiente ou tarefas repetitivas ou com incertezas.
Exemplos: Como se faz mesmo o deploy?
Como eu inicializo um aplicação? }

Esforço pertinente

- { Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance
Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

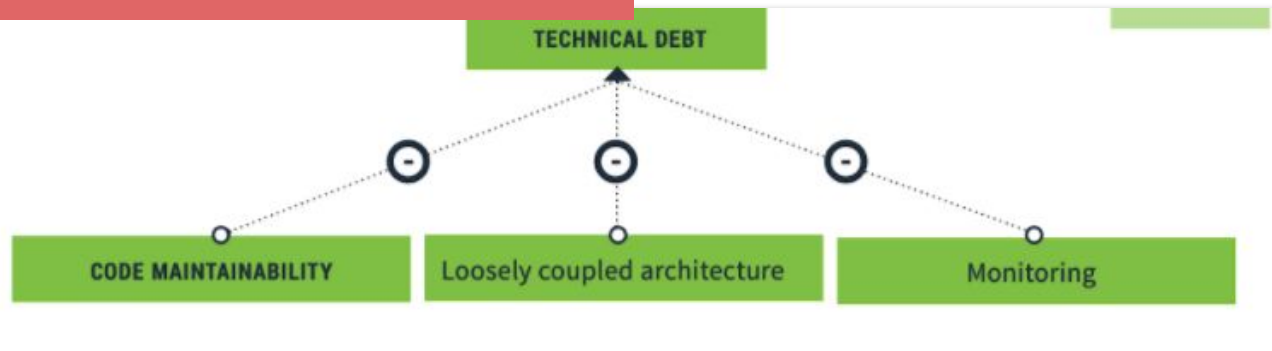
Três tipos de esforço cognitivo

< Deve ser **eliminado** – através de uma cultura de DevOps, arquitetura, automatização de boas práticas de segurança, eliminar comunicações desnecessárias, processos, *runbooks*, *playbooks*, ferramentas externas, **uso de serviços gerenciados** >

ação do problema.
ava?

Esforço

{ Se relaciona
Exemp



Esforço pertinente

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

Esforço intrínseco

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }

Esforço estranho

{ Se relaciona com aspectos do ambiente ou tarefas repetitivas ou com incertezas.

Exemplos: Como se faz mesmo o deploy?

Como eu inicializo um aplicação? }

Esforço pertinente *< Germane cognitive load />*

{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo

Esforço intrínseco

{ Se relaciona com aspectos fundamentais do espaço do problema.

Exemplos: Como se define uma classe em Java?

O que é o serviço X? }

Esforço estranho

{ < É aqui que ocorre a produtividade, escrita de código que adiciona valor ao negócio, concentração em tarefas desnecessárias >

E

s i

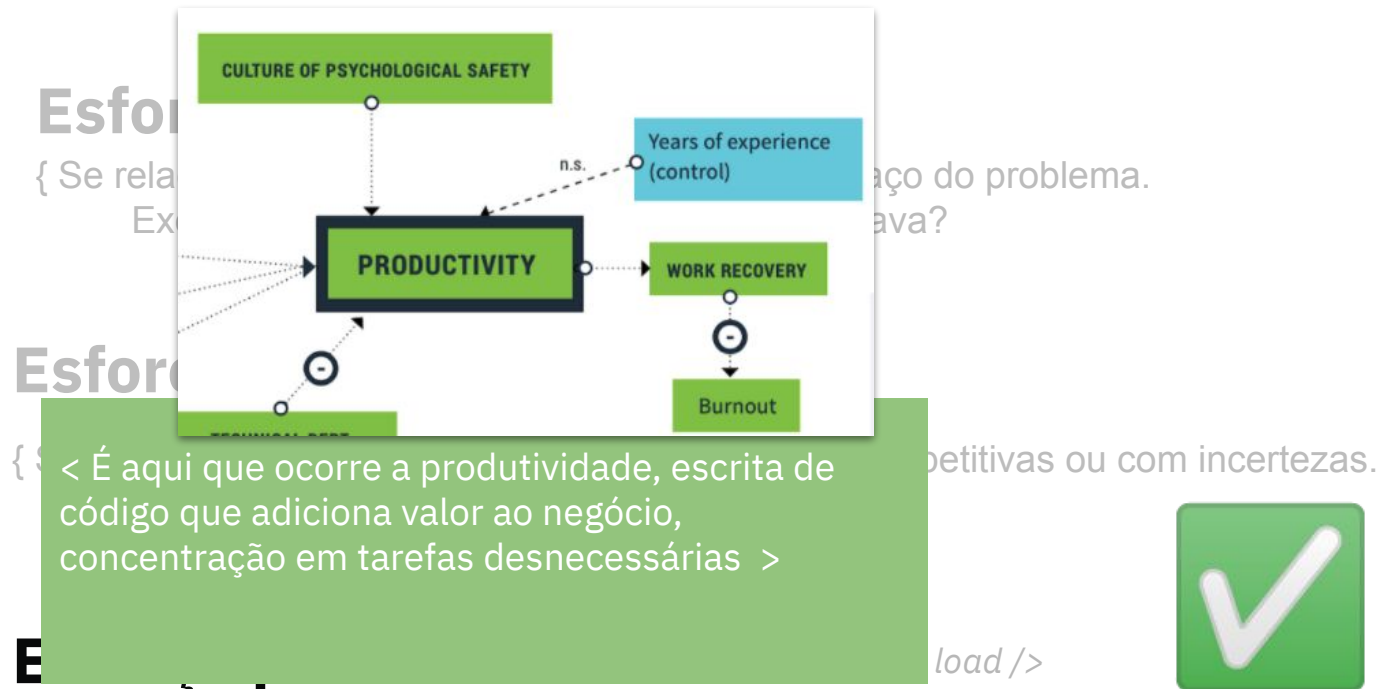
load />



{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

Três tipos de esforço cognitivo



{ Se relaciona com aspectos das tarefas que precisam de atenção especial, para aprendizado e alta performance

Exemplos: Como integrar DynamoDB streams com esta função Lambda? }

0 Intro

1 Qual o problema?

2 Visão estratégica

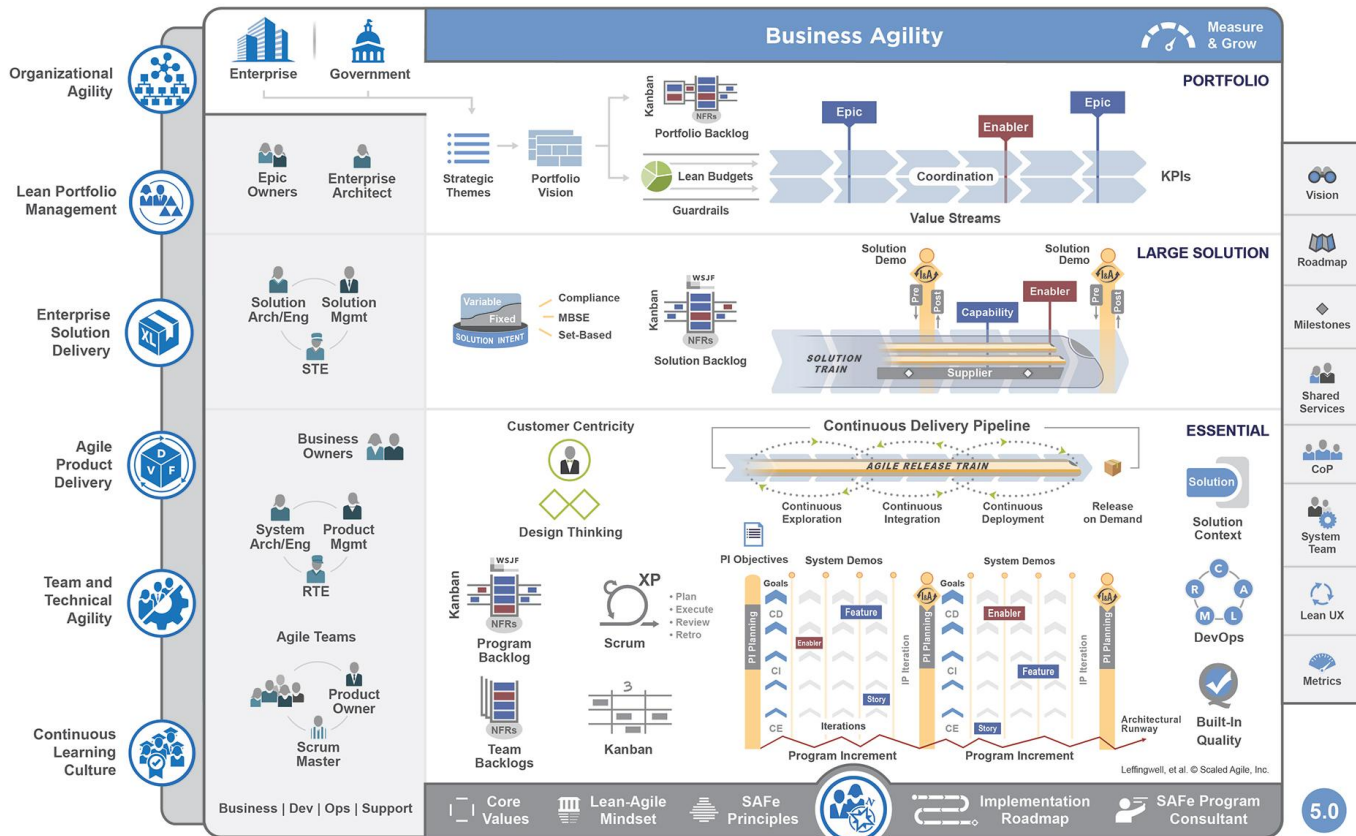
3 Tácticas

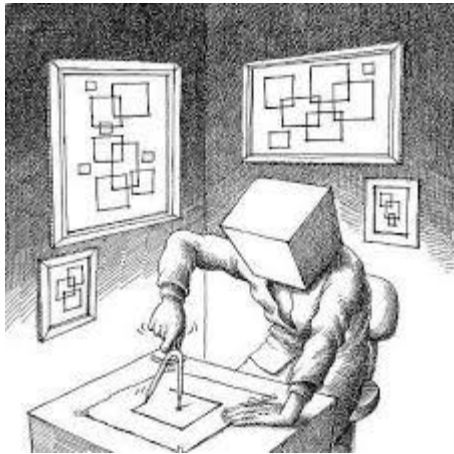
4 Pessoas



Soylent Green (1973)

Cuidado com as cachoeiras do Ágil®





**Nós moldamos nossas
ferramentas e nossas
ferramentas nos moldam**

*As relações humano-tecnológicas são uma dança sutil nas quais os objetos tecnológicos **empurram e puxam com vários degraus de insistência** enquanto os humanos navegam com **motivação, criatividade e habilidade em diferentes níveis**.*

Tecnologias são criadas, implementadas e utilizadas através de uma rede de escolhas.

– How Artifacts Afford, Jenny L. Davis

Sempre em movimento





Mês

Indicadores	Equipe					
Segurança emocional <i>Me sinto segura/o ao propor mudanças e dar feedback</i>	😐 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼
Valor final <i>Sinto orgulho daquilo que entrego</i>	😐 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼	😐 ▼
Processo <i>Sigo um processo que funciona no meu contexto</i>	😐 ▼	👍 ▼	👍 ▼	😐 ▼	👍 ▼	😐 ▼
Equipe <i>Colaboro ativamente com as outras pessoas</i>	😐 ▼	👍 ▼	👍 ▼	😐 ▼	👍 ▼	😐 ▼
Produtividade <i>Sinto que tiro bom proveito das horas de trabalho e consigo focar no que é prioridade</i>	😐 ▼	😐 ▼	😐 ▼	😐 ▼	👍 ▼	😐 ▼
Aprendizado <i>Estou aprendendo coisas novas regularmente</i>	😐 ▼	👍 ▼	😐 ▼	👍 ▼	👍 ▼	👍 ▼
Ambiente <i>Gosto de trabalhar e o tempo que passo aqui é agradável</i>	😐 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼
Senso de responsabilidade <i>Me sinto igualmente responsável pelas decisões dentro do meu time</i>	😐 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼
Cadência <i>Sinto que tenho um ritmo saudável de entregas</i>	😐 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼	👍 ▼
Missão <i>Sinto que meu trabalho é importante para a empresa</i>	😐 ▼	👍 ▼	😐 ▼	👍 ▼	👍 ▼	👍 ▼



< O que é um mestre? Me surpreendo a todo momento com Python />

– Guido van Rossum, criador da linguagem Python

<Obrigado />



@ibrahimcesar

<https://ibrahimcesar.cloud>

