THE DEVELOPER'S CONFERENCE

Generalização Prematura e Complexidade Acidental: A raiz do **mal** de todo software

Letticia Nicoli
Software Engineer @Nubank
@letticianicoli

Lucas Teles
Software Engineer @Nubank
@lucasteles42

THE DEVELOPER'S CONFERENCE

Generalizaçã… …rd e Com… …cidental: A … de todo software

Palestra Motivacional

Letticia Nicoli
Software Engineer @Nubank
@letticianicoli

Lucas Teles
Software Engineer @Nubank
@lucasteles42

Somos ótimos em
gerar **complexidade**.

```
if (condition)
{
    return null;
}
```

```
if (condition == true)
{
    return null;
}
```

```
if (Convert.ToString(condition).Equals("true"))
{
    return null;
}
```

```
bool isTrue = true;
string isTrueStr = Boolean.TrueString;
int count = 0;

var chars = Convert.ToString(condition).ToCharArray();

foreach (var item in chars)
{
    if (item != chars[count])
    {
        isTrue = false;
    }
    count = count + 1;
}

if (Convert.ToString(isTrue).Equals(Boolean.TrueString))
{
    return null;
}
```

```
public static bool IsEmailValid(string email)
{
    string emailValidPattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\\])*""|"
        + @"([-a-z0-9!#$%&'*+/=?^_`{|}~]|(?<!\.)\.)*)(?<!\.)"
        + @"@[a-z0-9][\w\.-]*[a-z0-9]\.[a-z][a-z\.]*[a-z]$";

    var regex = new Regex(emailValidPattern, RegexOptions.IgnoreCase);

    return regex.IsMatch(email);
}
```

```
public bool IsEmailValid(string email)
{
    try
    {
        new MailAddress(email);
        return true;
    }
    catch (FormatException)
    {
        return false;
    }
}
```

Nick
@Zorchenhimer

Follow

THE DEVELOPER'S CONFERENCE

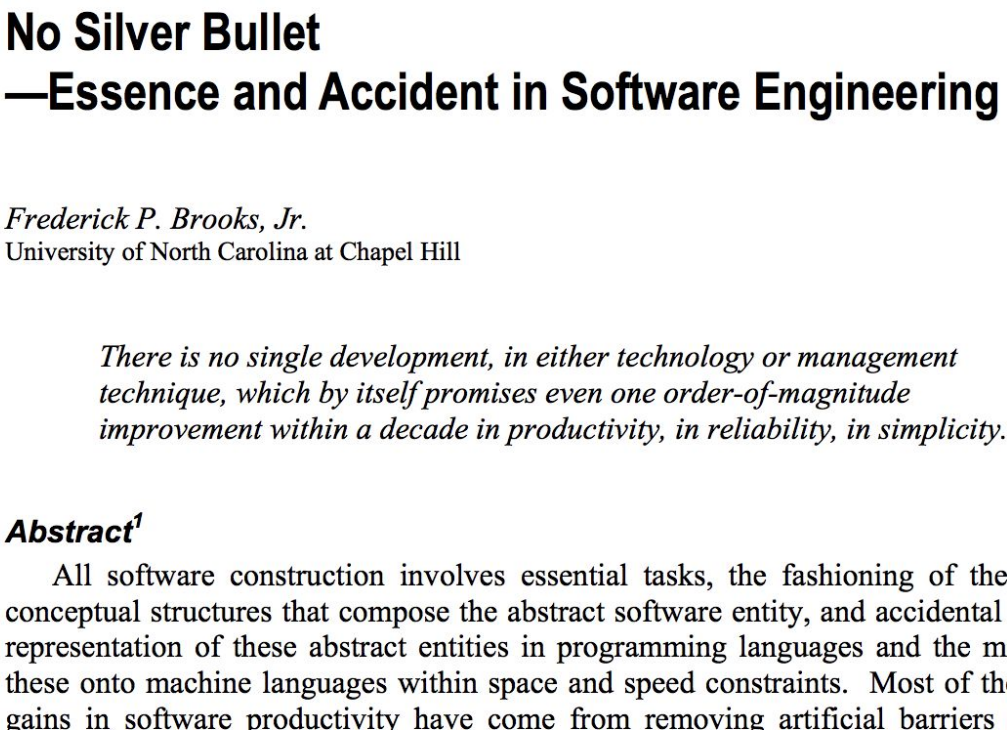
Found this in production today. I need a drink.

```
public static bool CompareBooleans(bool orig, bool val)
{
    return AreBooleansEqual(orig, val);
}

internal static bool AreBooleansEqual(bool orig, bool val)
{
    if(orig == val)
        return false;
    return true;
}
```

12:24 PM - 30 May 2019

3,272 Retweets **9,020** Likes

No Silver Bullet



No Silver Bullet
—Essence and Accident in Software Engineering

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill

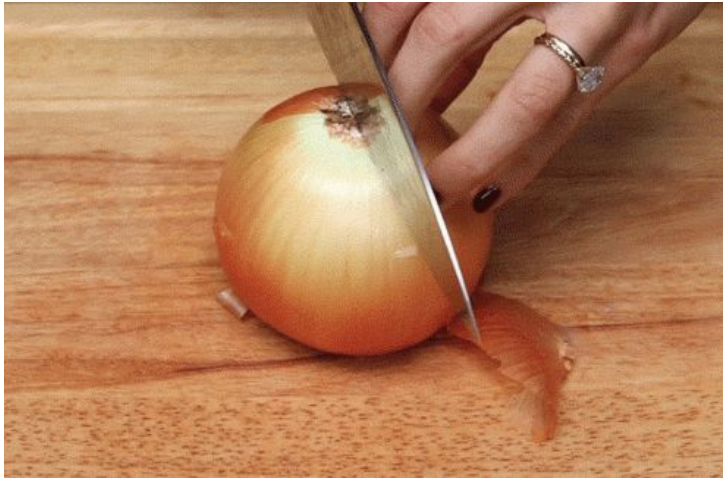
There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

Abstract[1]

All software construction involves essential tasks, the fashioning of the complex conceptual structures that compose the abstract software entity, and accidental tasks, the representation of these abstract entities in programming languages and the mapping of these onto machine languages within space and speed constraints. Most of the big past gains in software productivity have come from removing artificial barriers that have

Complexidade Essencial

- Totalmente ligada ao **domínio/negócio**.
- Aquilo que **não pode ser evitado**, **independentemente** da sua experiência, de quais ferramentas você utiliza ou qual padrão de arquitetura novo e chamativo você usou para resolver o problema.



Complexidade Acidental

THE DEVELOPER'S CONFERENCE

- Totalmente ligada às **decisões** que tomamos.

- Causada pela **abordagem escolhida** para resolver o problema:
  - Todas as ***buzzwords*** que você não precisava mas achou que era necessário.
  - Técnicas e tecnologias (frameworks, patterns, ambientes operacionais etc)

Não esqueça!

THE DEVELOPER'S CONFERENCE

Como identificar essa complexidade?

Boat Anchor

Cargo Cult

Lasagna Code

THE
DEVELOPER'S
CONFERENCE

Keep It Stupid Simple

KISS

KISS

- Complicações geram **confusão**, pense na manutenção e fácil entendimento no futuro.
- **Lembre-se:** a funcionalidade pode ser complexa mas o seu código não!

"Não é a linguagem que faz os programas parecerem simples.
É o programador que faz a linguagem parecer simples!"
Uncle Bob

Otimização Prematura

"O verdadeiro problema é que programadores gastam muito tempo se preocupando com eficiência em lugares e momentos errados; otimização prematura é a *raiz de todo o mal* (ou pelo menos a maior parte) na programação."

Donald Knuth

Otimização Prematura

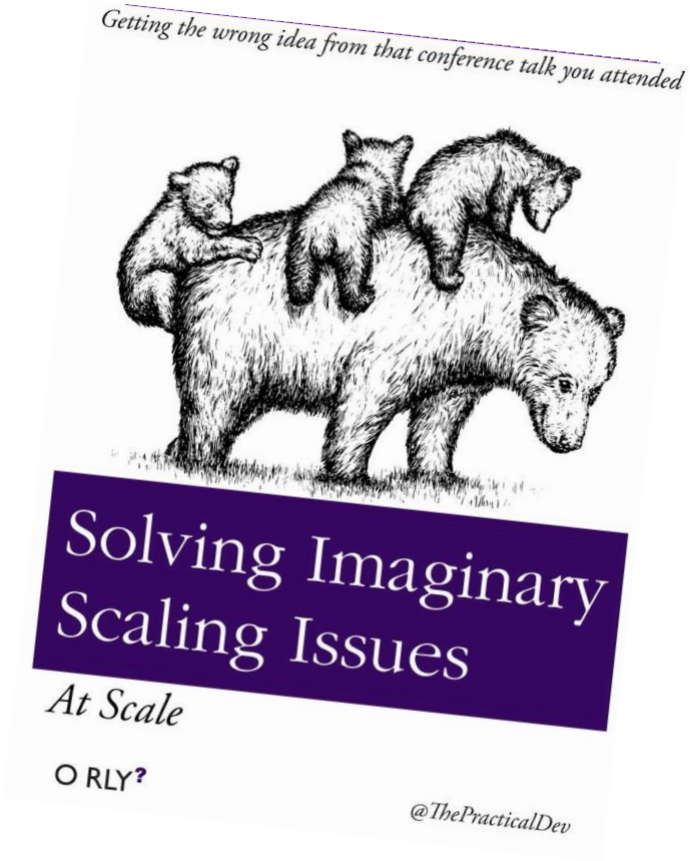
- Você precisa de **Event Sourcing**?
- Você precisa de **Micro-serviços**?
- Qual valor **CQRS** vai te trazer agora?
- Vamos usar **Dapper** por que é mais rápido?

Tom Maiaroto
@tmaiaroto

Seguir

Used to pay $5/mo on a small instance for my personal site. Then I discovered Kubernetes and realized my site didn't scale! No canary deployments! So I upgraded and pay $200/mo now. Took weeks to configure. Millions of people can now read my resume. Damn, it's never looked better

07:53 - 8 de jul de 2019


Getting the wrong idea from that conference talk you attended
Solving Imaginary Scaling Issues
At Scale
O RLY?
@ThePracticalDev

Otimização Prematura

Moral da história: Medir antes de otimizar

Generalização Prematura

- Remover código duplicado é sempre uma boa ideia?
- Prefira código duplicado a uma abstração errada!

> "O propósito da abstração não é sobre ser vago, mas sobre criar um novo nível de semântica na qual podemos ser absolutamente precisos."
>
> *Edsger Dijkstra*

Generalização Prematura

THE DEVELOPER'S CONFERENCE

1. O Lucas vê uma **duplicação de código**.

```
public void AprovarCompra(Compra compra)
{
  compra.Aprovado = true;
  salvar(compra);
}

public void AprovarSolicitacao(Solicitacao solicitacao)
{
  solicitacao.Aprovado = true;
  salvar(solicitacao);
}
```

Generalização Prematura

2. O Lucas **extrai a duplicação** e dá um nome.

*Com isso cria uma **nova abstração**.*
Um novo método ou talvez até uma nova classe.

```
public interface IAprovavel {
  public bool Aprovado { get; set; }
}

public class Solicitacao : IAprovavel { ... }
public class Compra : IAprovavel { ... }

public void Aprovar(IAprovavel aprovavel)
{
  aprovavel.Aprovado = true;
  salvar(aprovavel);
}
```

Generalização Prematura

3. O Lucas **substitui a duplicação** pela nova abstração.

Ah, o código é perfeito.
O Lucas está feliz.

Generalização Prematura

4. O **tempo** passa...

Generalização Prematura

5. Um **novo requisito** aparece para o qual a abstração atual é quase perfeita.

```
{
    compra.Aprovado = true;
    compra.DataAprovacao = DateTime.Now;
    salvar(compra);
}
```

Generalização Prematura

6. Agora a Letticia é encarregada de **implementar** esse **novo requisito**.

Generalização Prematura

THE DEVELOPER'S CONFERENCE

7. A Letticia se sente obrigada a **manter** a **abstração existente**, é igualzinho, ela altera o código para receber um parâmetro, e adiciona a lógica para fazer a coisa de acordo com parâmetro.

*O que antes era uma abstração universal agora se **comporta** de maneira **diferente** para casos diferentes.*

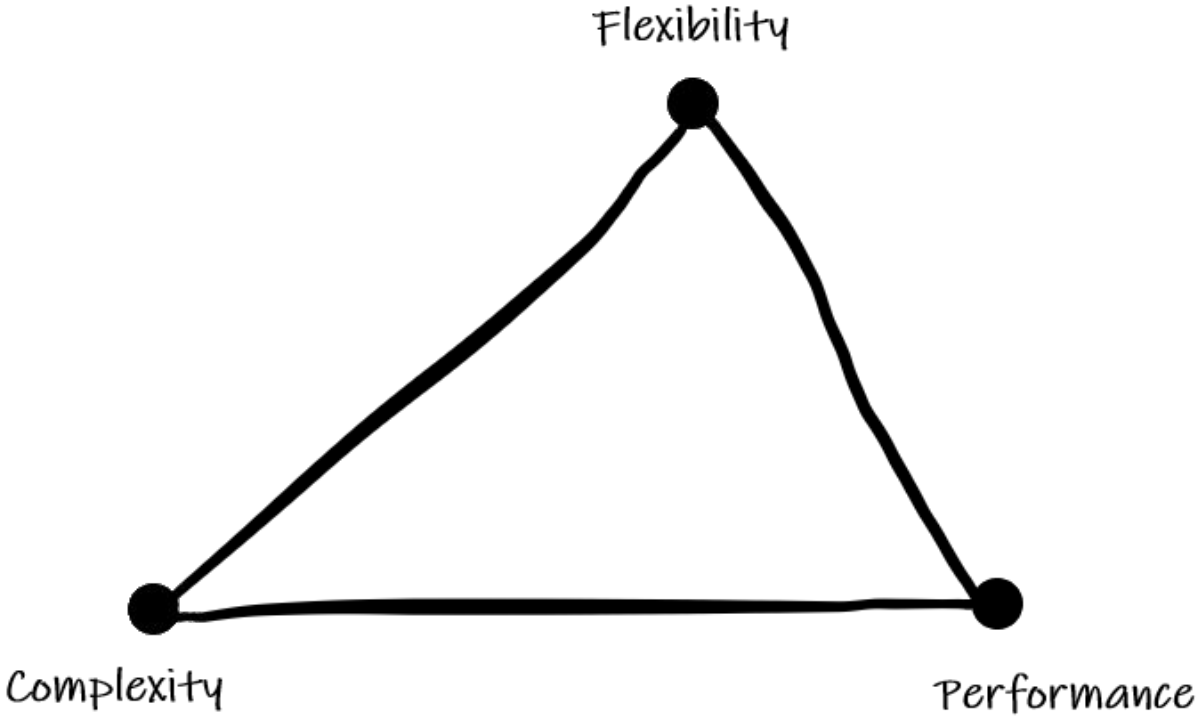
```
public void Aprovar(IAprovavel aprovavel, bool temData = false)
{
    aprovavel.Aprovado = true;

    if (temData)
        aprovavel.DataAprovacao = DateTime.Now;

    salvar(aprovavel);
}
```

Generalização Prematura

8. Outra **nova exigência** chega.

- *Chegou outra pessoa no time.*
- *Outro parâmetro adicional.*
- *Outra nova condicional.*
- *Faça um loop até que o* ***código*** *se torne* ***incompreensível****.*

Tudo não terás



THE
DEVELOPER'S
CONFERENCE

You Ain't Gonna Need It

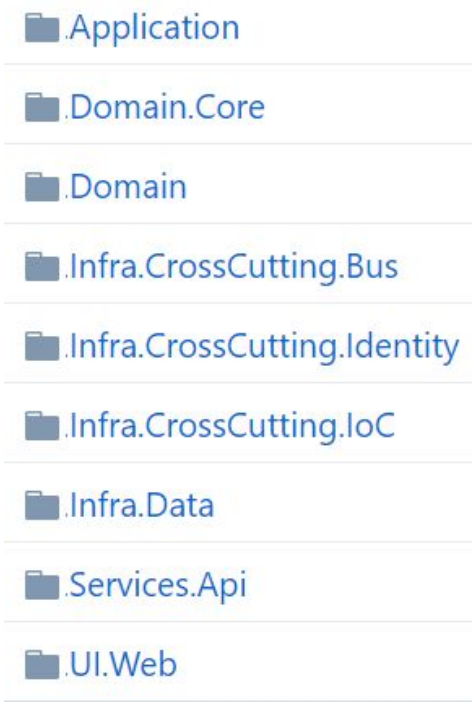
YAGNI

YAGNI

- Somos péssimos em prever o futuro!
- Implemente apenas o que você **realmente vai precisar** naquele momento.
- **Reflita:** Eu tenho 100% de certeza de que isso é necessário agora?

YAGNI

- Isso não é **DDD**



Manifesto Ágil

1. **Indivíduos e interações** mais que processos e ferramentas
2. **Software em funcionamento** mais que documentação abrangente
3. **Colaboração com o cliente** mais que negociação de contratos
4. **Responder a mudanças** mais que seguir um plano

Software funcional com entrega de valor é a medida primária de **progresso**.

Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.

~~Débito Técnico~~

Dívida Técnica

Refatoração

- Um software não é apenas mais um pedaço de código finalizado e entregue, se trata de uma solução em **constante evolução**.

Testes

- Antecipação de problemas
- Faz parte do nosso dia a dia de desenvolvimento.

Premissa

THE DEVELOPER'S CONFERENCE

Não julgar, acreditar que fizeram o melhor que podia com tempo que tinham

@letticianicoli

@lucasteles42