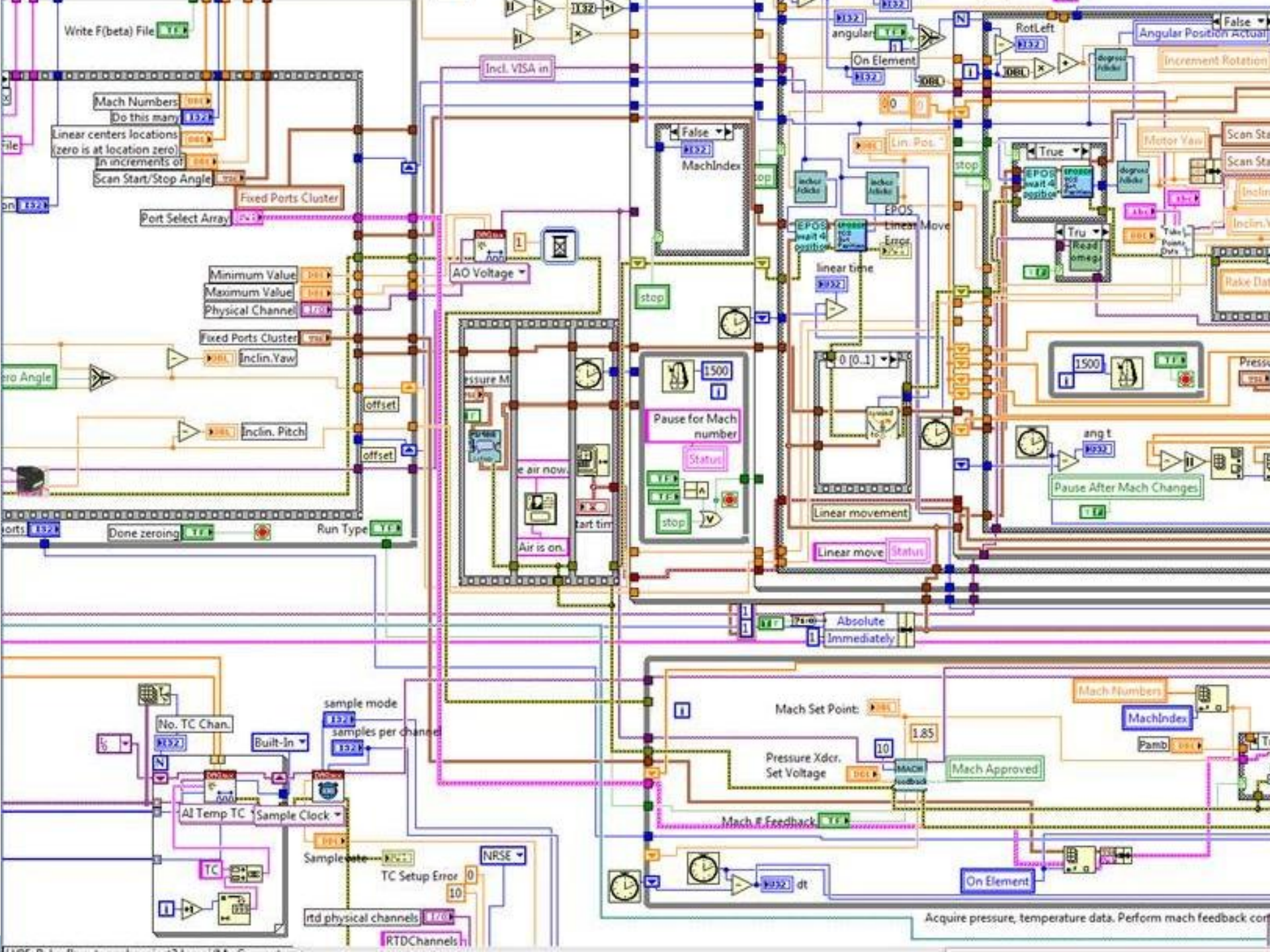


# Padrões e exemplos de projetos no LabVIEW 2012

Desenvolvendo aplicações escaláveis com arquiteturas orientadas a Objeto

Ilton Pereira – Gerente da Engenharia de Aplicações

Felipe Flores – Engenheiro de Aplicações



# Você já ouviu isso alguma vez?

Adicionar ou alterar suporte à hardware requer tempo e esforço significativos.

Para adicionar uma funcionalidade similar a uma já existente, você constantemente copia trechos do código.

Ao adicionar uma nova funcionalidade, o código começa a apresentar falhas devido a um mau projeto.



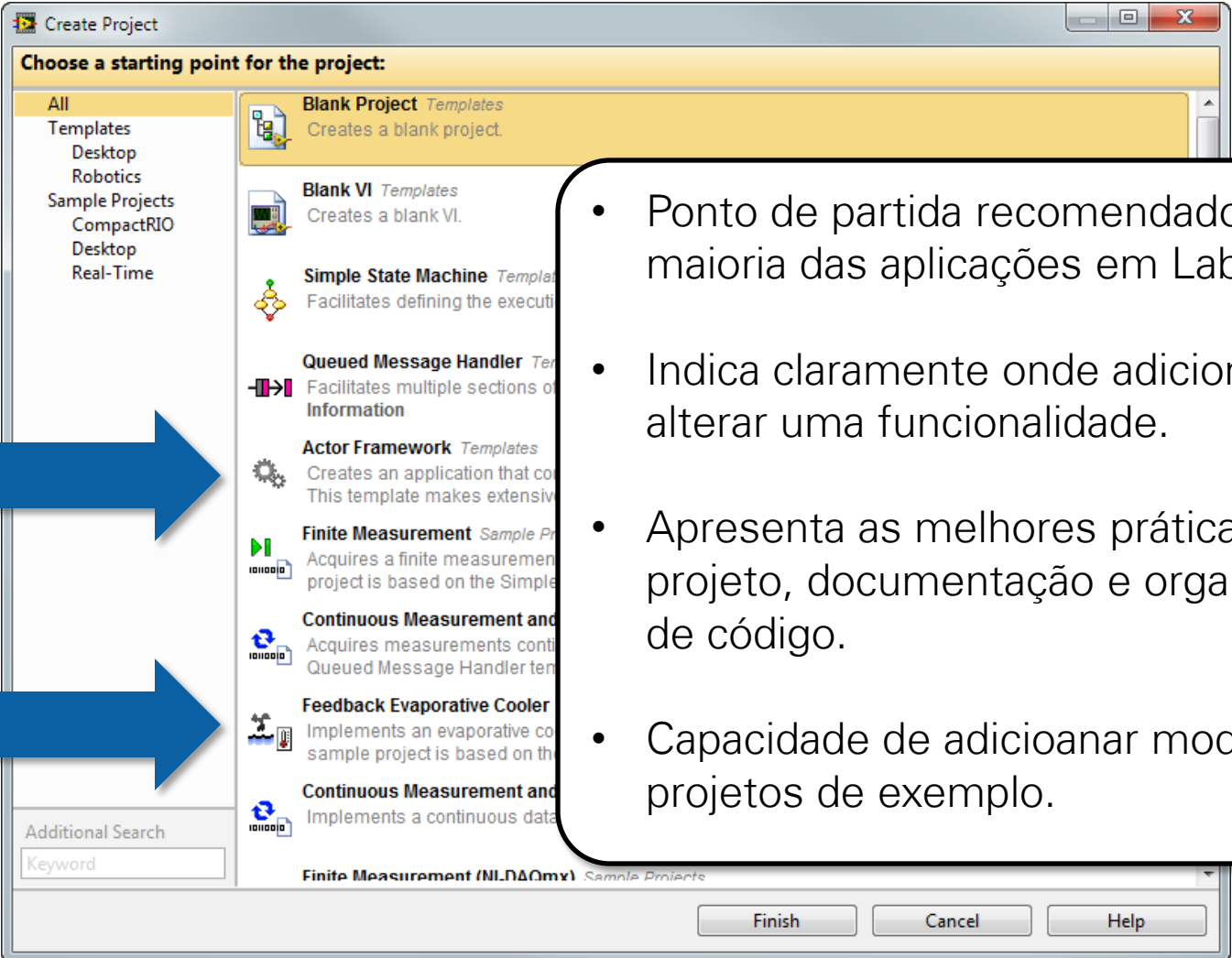
Simples Escalável Reutilizável

# Agenda

- Problemas clássicos em desenvolvimento de software
- Abordagem tradicional de programação no LabVIEW
- Limitações da abordagem tradicional
- Orientação a objeto no LabVIEW
- Actor Framework – Modelo de arquitetura orientada a objeto.
- Feedback Evaporative Cooler - Demonstração



# Modelos e projetos de exemplo no LabVIEW



**Create Project**

Choose a starting point for the project:

**All**

- Templates
  - Desktop
  - Robotics
- Sample Projects
  - CompactRIO
  - Desktop
  - Real-Time

**Blank Project** *Templates*  
Creates a blank project.

**Blank VI** *Templates*  
Creates a blank VI.

**Simple State Machine** *Templates*  
Facilitates defining the execution flow.

**Queued Message Handler** *Templates*  
Facilitates multiple sections of code.

**Actor Framework** *Templates*  
Creates an application that can be extended. This template makes extensive use of the Actor Framework.

**Finite Measurement** *Sample Projects*  
Acquires a finite measurement. The project is based on the Simple State Machine template.

**Continuous Measurement and Control** *Sample Projects*  
Acquires measurements continuously. The project is based on the Queued Message Handler template.

**Feedback Evaporative Cooler** *Sample Projects*  
Implements an evaporative cooler. The project is based on the Simple State Machine template.

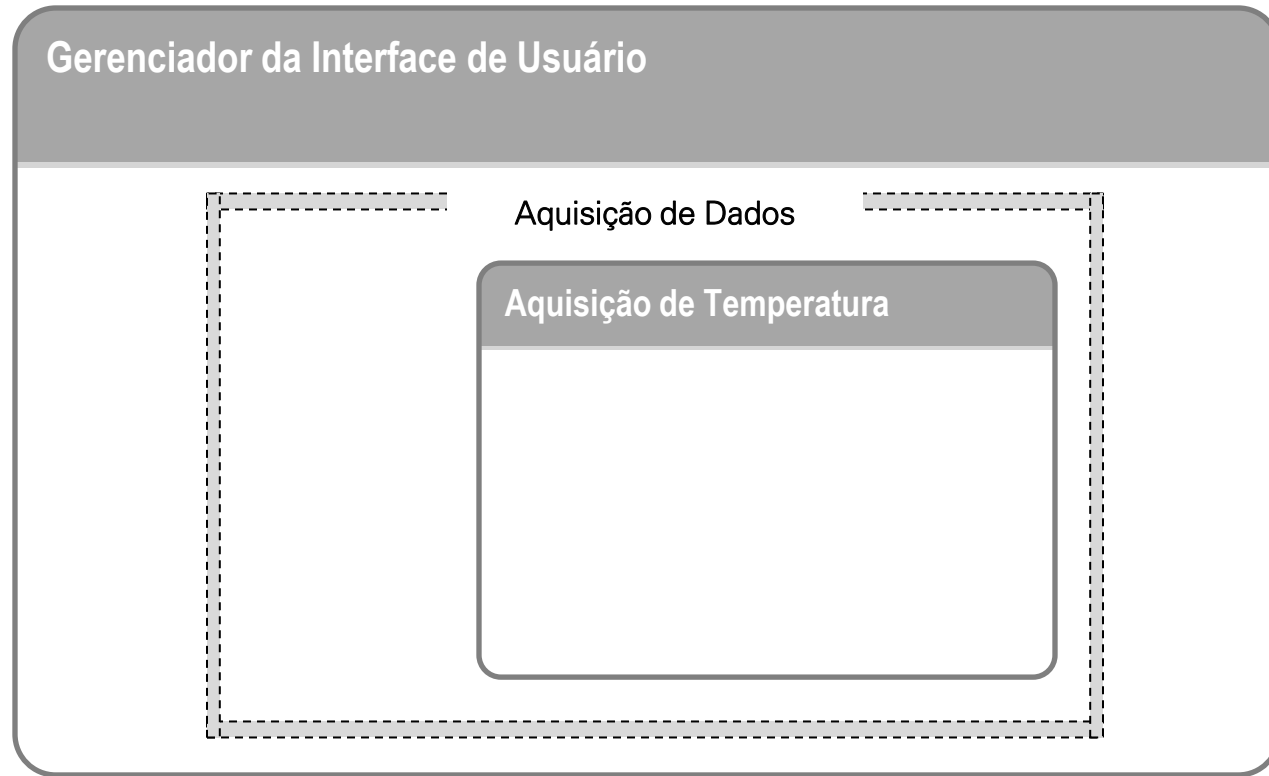
**Continuous Measurement and Control** *Sample Projects*  
Implements a continuous data acquisition system. The project is based on the Queued Message Handler template.

Additional Search  
Keyword

Finish Cancel Help

- Ponto de partida recomendado para a maioria das aplicações em LabVIEW
- Indica claramente onde adicionar ou alterar uma funcionalidade.
- Apresenta as melhores práticas para projeto, documentação e organização de código.
- Capacidade de adicionar modelos e projetos de exemplo.

# Por que o acoplamento (forte) é ruim?

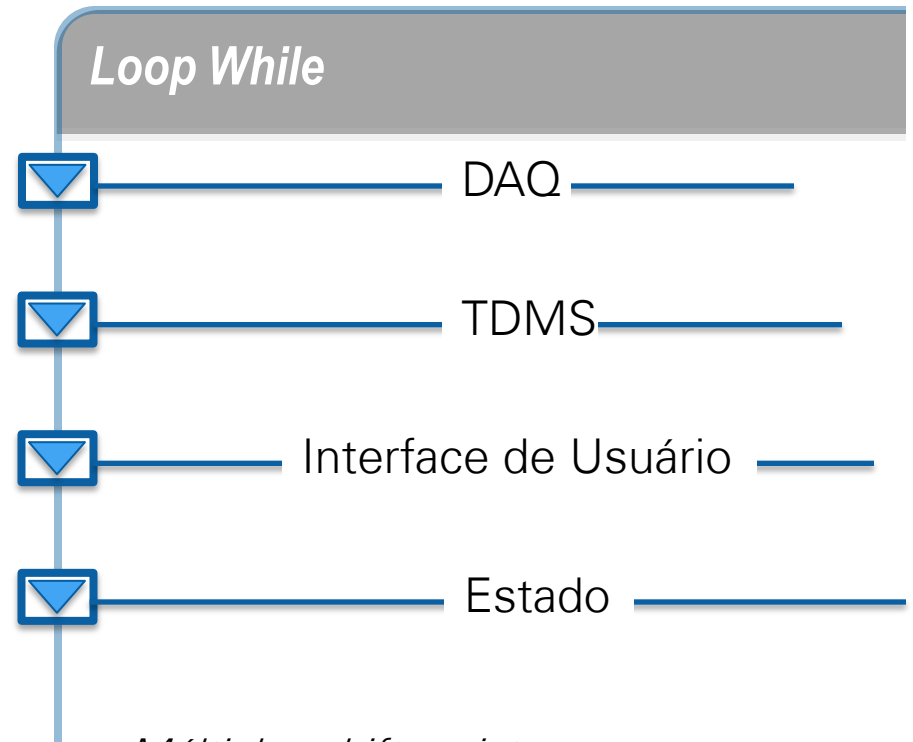


# Coesão: entendendo o escopo de dados

*O shift register define o escopo do processo.*

Os processos deveriam ser bem coesos.

Processos independentes deveriam estar em *loops* separados.



*Múltiplos shift registers para um processo indica um superacoplamento.*

# Desacoplando processos independentes

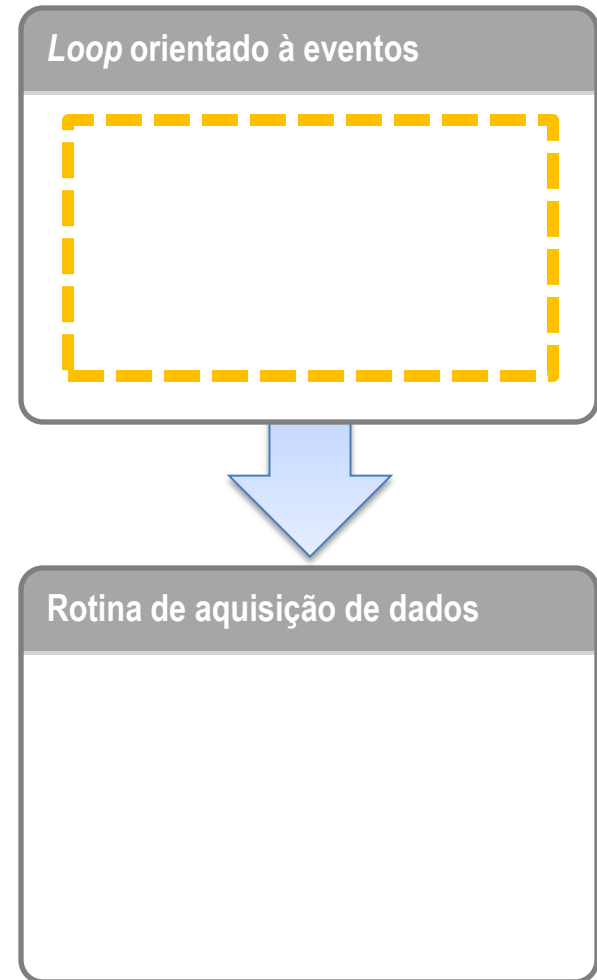
## Melhores práticas

1. Identificar o escopo de dados;
2. Delegar ações ao processo apropriado;
3. Não faça *polling* ou use *timeouts*\*

*\*exceto para códigos que se comunicam com "Hardware."*

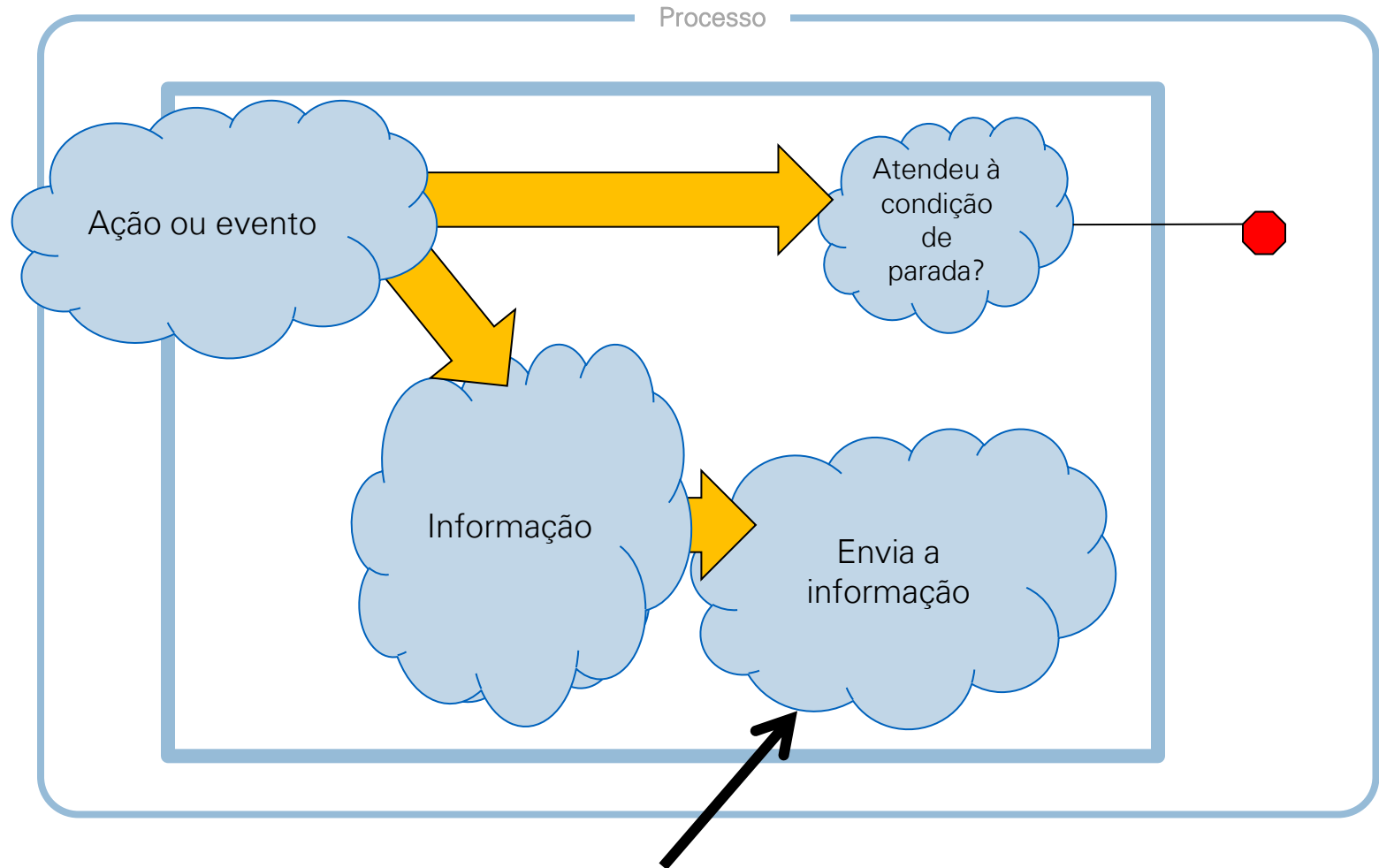
## Considerações

1. Como enviar comandos?
2. Como enviar dados?
3. Que processos podem se comunicar entre si?
4. Como atualizar a interface de usuário?



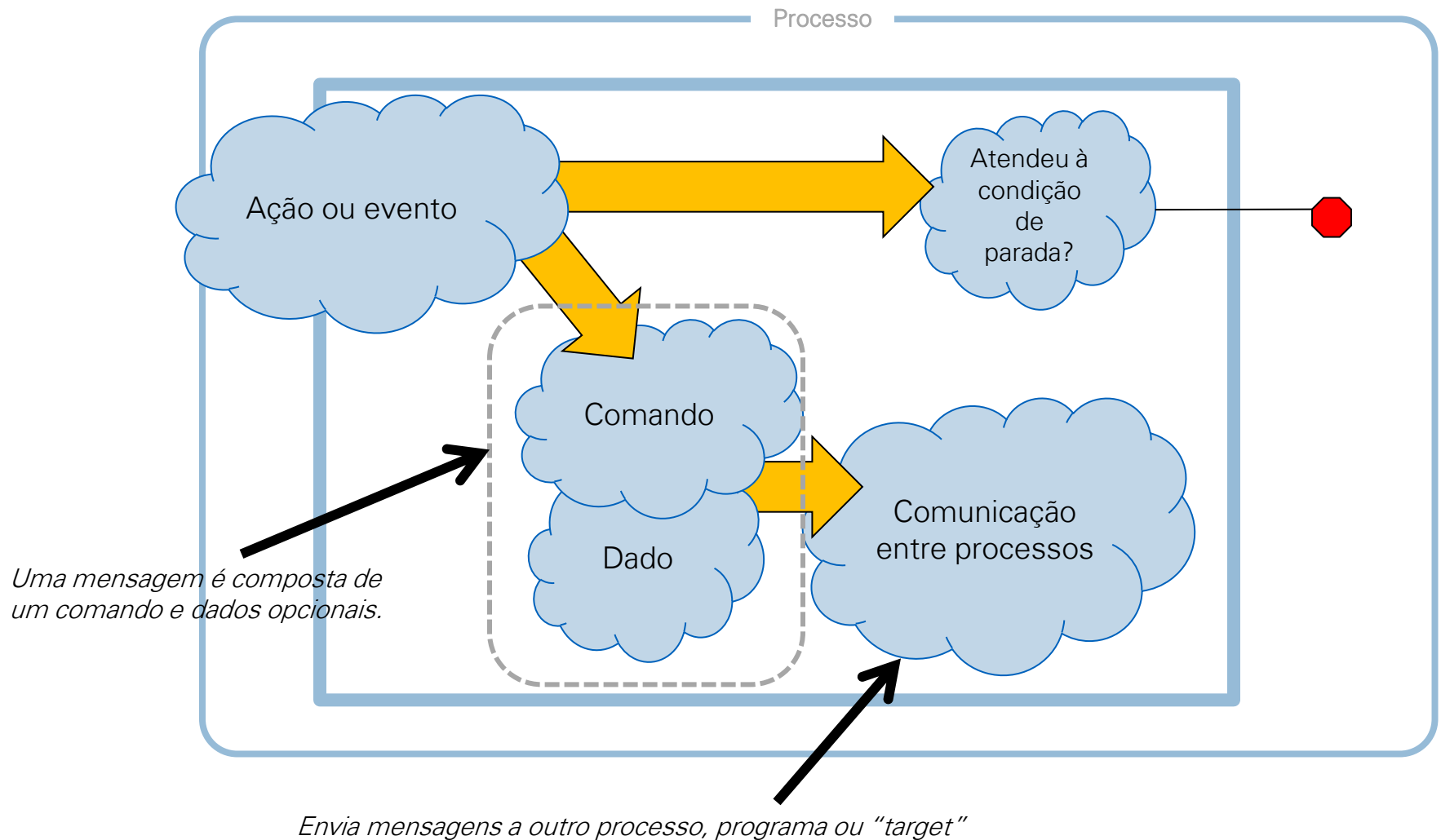


# Análise de um *loop* produtor



*Envia a mensagem a outro processo, programa ou "target".*

# Análise de um *loop* produtor de mensagens

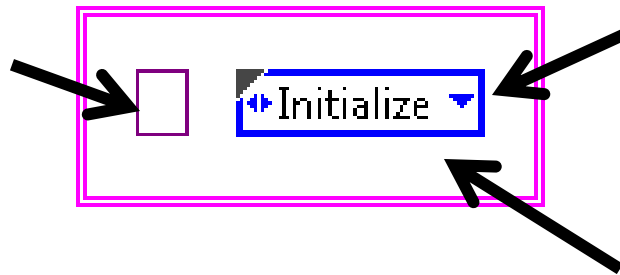


# Construindo uma mensagem

## Dados

O dado do tipo "Variant" pode ser convertido em qualquer outro tipo de dado.

Diferentes mensagens podem exigir diferentes dados.



## Comando

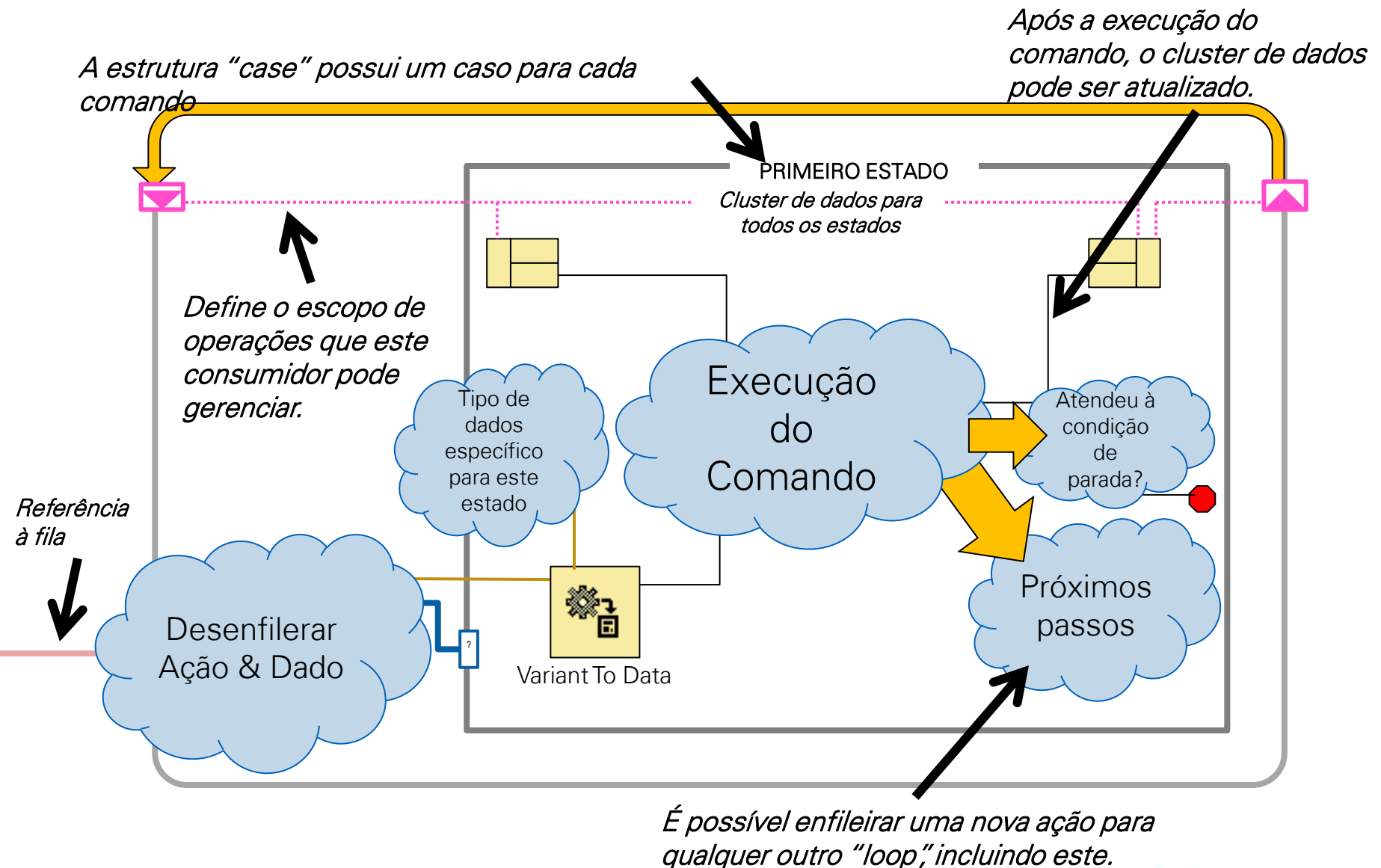
Constantes enumeradas (Enums) listam todas as opções

Type Def.

## Exemplos

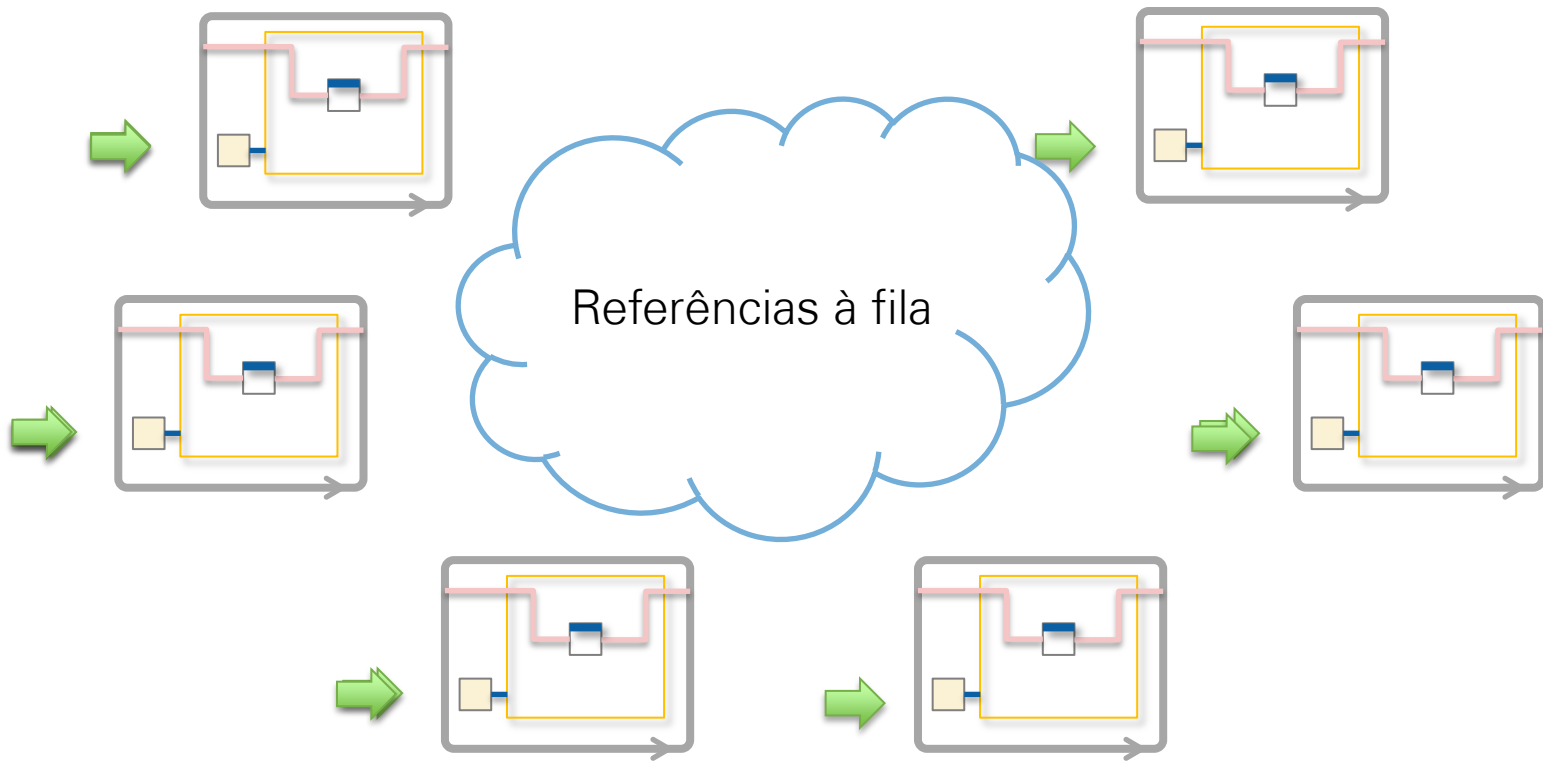
Comando	Dado
Inicializar Interface de Usuário	<i>Cluster</i> contendo os dados de configuração
Popular <i>Menu</i>	<i>Array de strings</i> para mostrar no <i>menu</i>
Redimensionar Janela	<i>Array</i> de inteiros [Largura, Altura]
Carregar <i>Subpanel</i>	Referência do VI a ser carregado
Inserir Cabeçalho	<i>String</i>
Parar	-

# Processo de gerenciamento de mensagens por fila



# Criando múltiplos processos consumidores

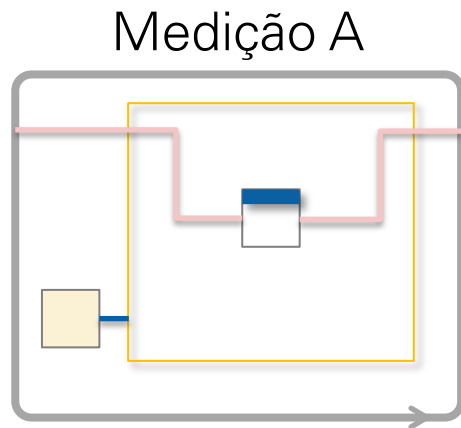
**Cenário um:** Múltiplas instâncias do mesmo processo



**Solução Tradicional:** Carregar o VI dinamicamente ou duplicar código.

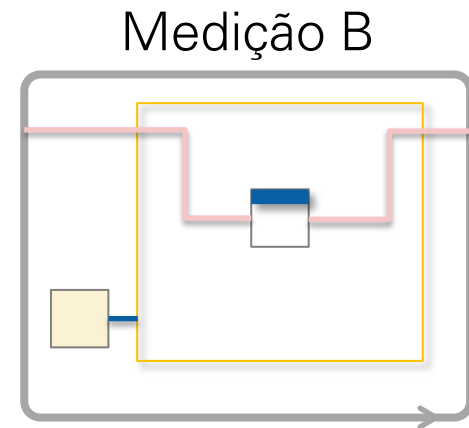
# Criando múltiplos processos consumidores

**Cenário dois:** Personalizar o comportamento existente



Mensagens

- Configurar
- Adquirir
- Medir
- Fechar
- Sair



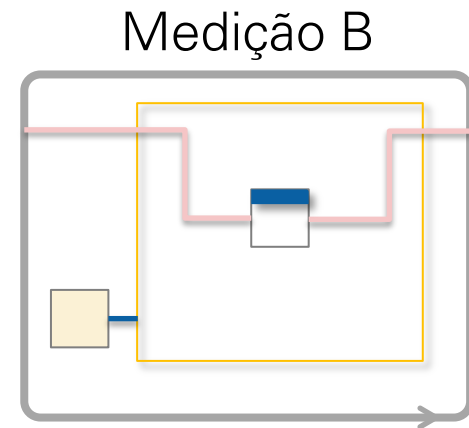
Mensagens

- Configurar
- Adquirir
- **Medir**
- Fechar
- Sair

**Solução Tradicional:** Manter duas cópias separadas



## Cenário três: Adicionar nova funcionalidade



# Mensagens

- Configurar
- Adquirir
- Registrar
- Medir
- Sacar
- Sair

## Solução Tradicional: Manter duas cópias separadas

Qual seria então uma outra abordagem para o desenvolvimento de aplicações escaláveis, reutilizáveis e simples?

# Classe: Carro

Exemplos de objetos dessa classe



Propriedades desses itens:

- Consumo (km/l)
- Índice de segurança
- Câmbio manual/automático
- Motor
- Potência
- Combustível

Dados

Esses itens podem...

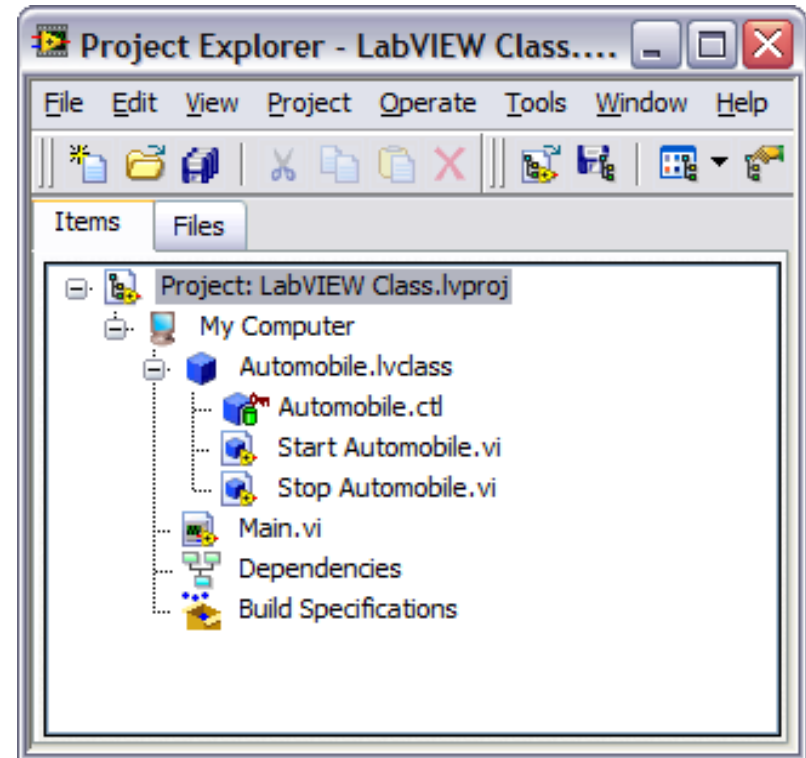
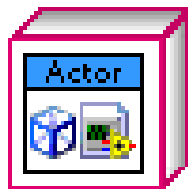
- Ligar
- Dirigir
- Trocar marcha
- Acionar os *Air Bags*
- **Viajar no tempo**

Métodos

# O que é uma Classe do LabVIEW?

**Classe:** Um conjunto de dados e métodos que interagem com esses dados.

**Objeto:** Uma instância de uma classe.



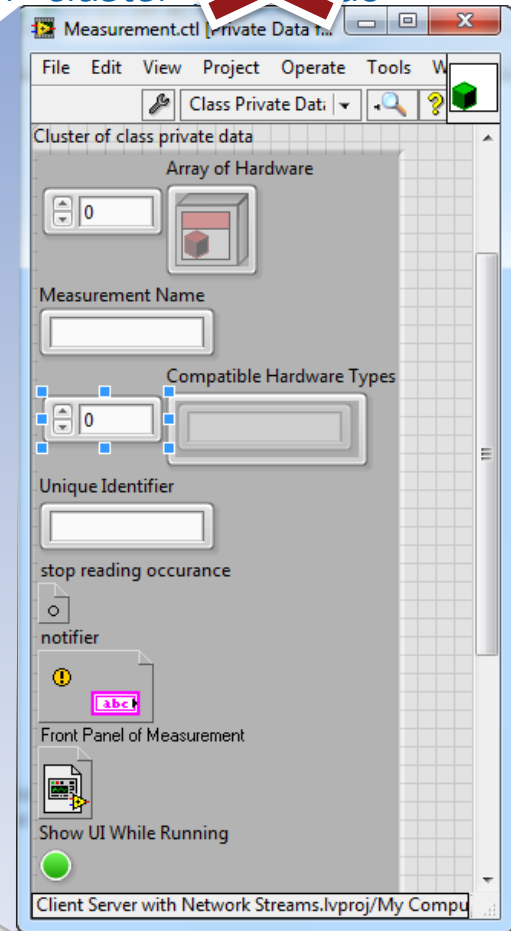
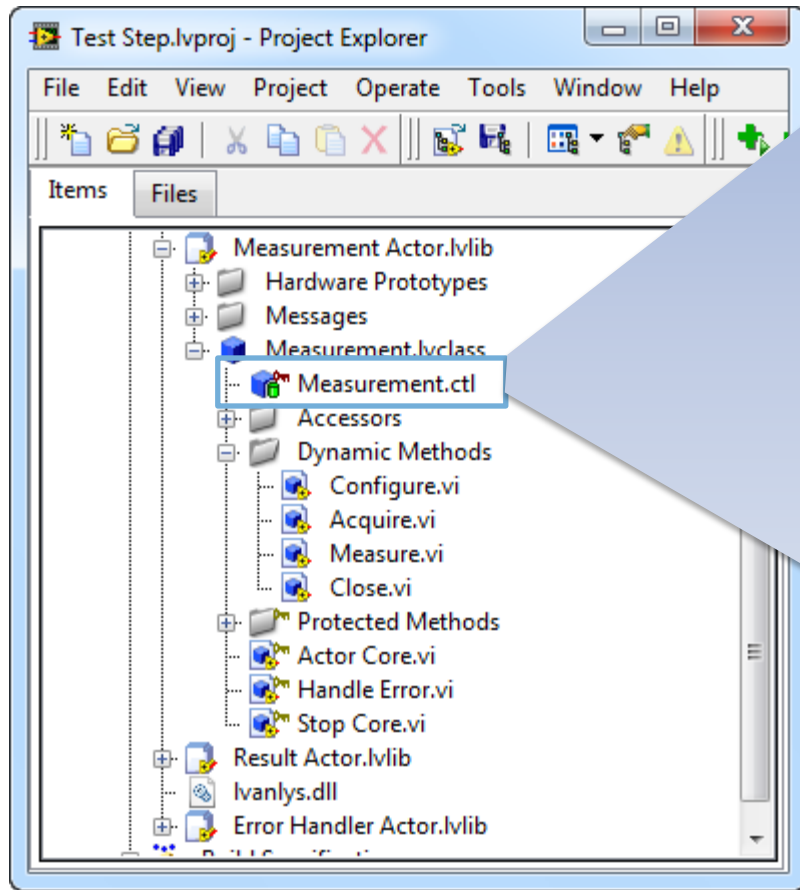
Uma classe do LabVIEW é...

- **Cluster "Chuck Norris"**
- Um tipo de dado definido pelo usuário
- Um tipo de biblioteca de projeto

# Controle de dados privados: encapsulamento!

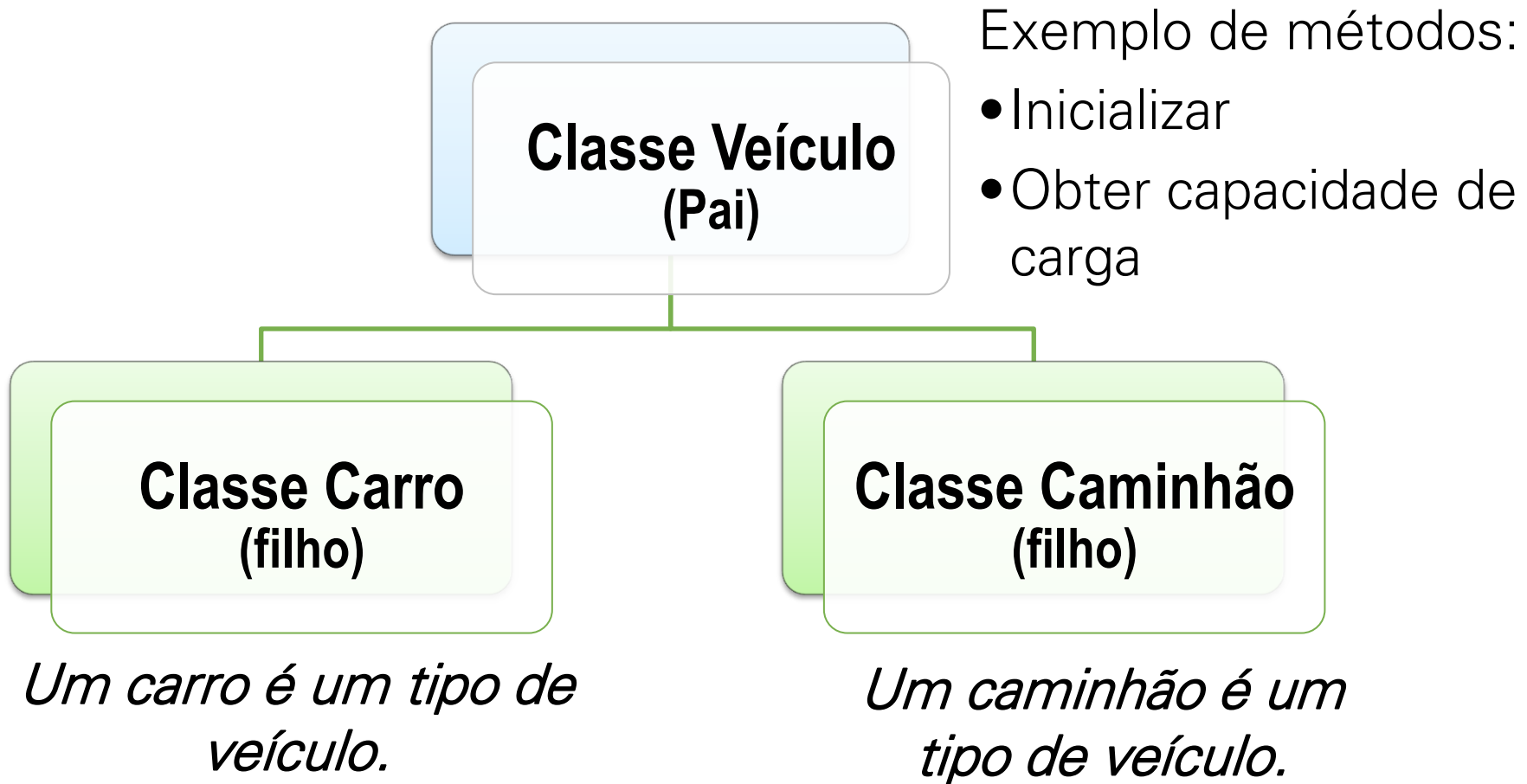
O motivo pelo qual algumas pessoas chamam classes de um "*cluster* ~~privado~~"

## Cluster "Chuck Norris"



Somente VIs pertencentes à classe podem acessar ou modificar estes dados.

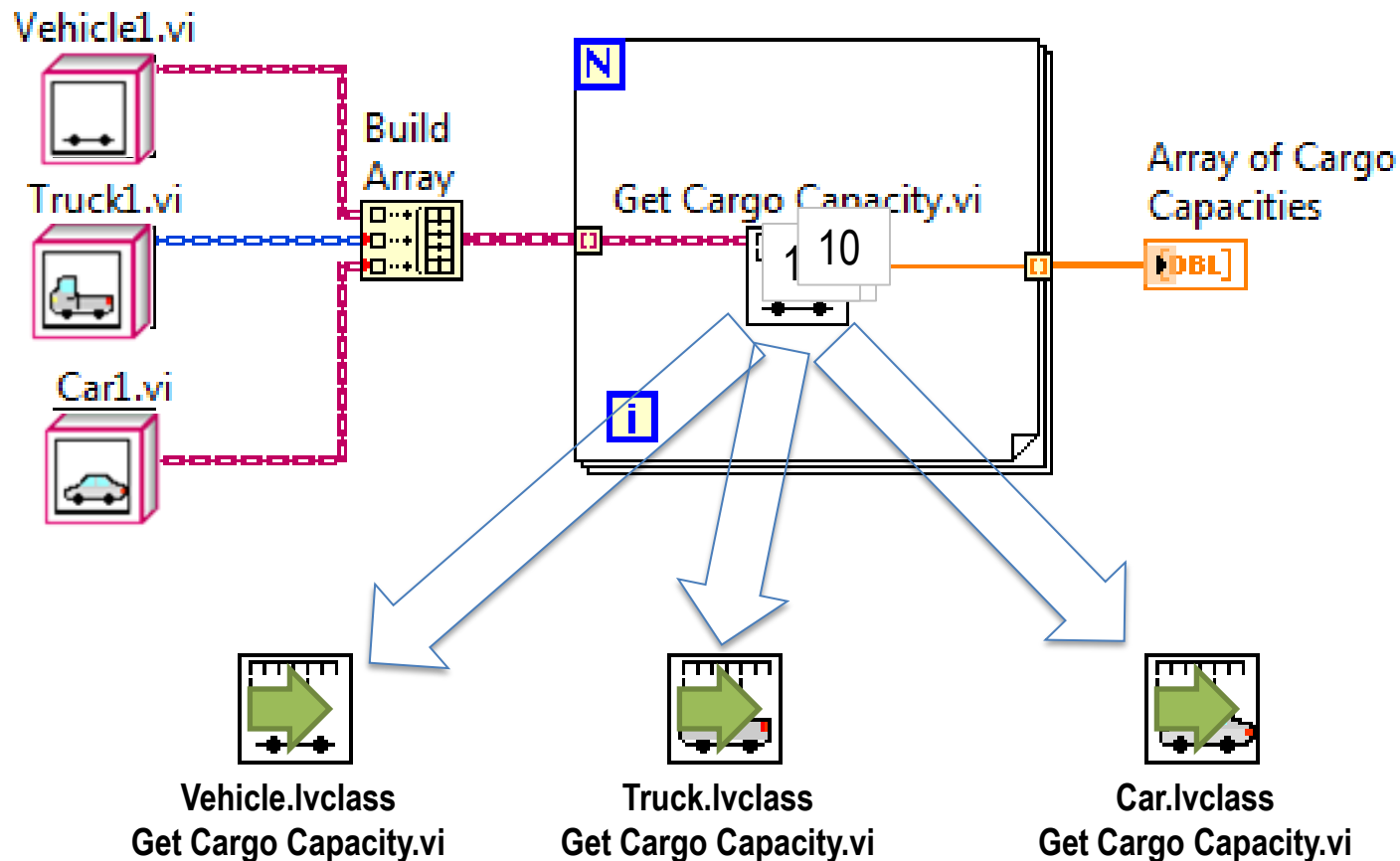
# O que é herança?



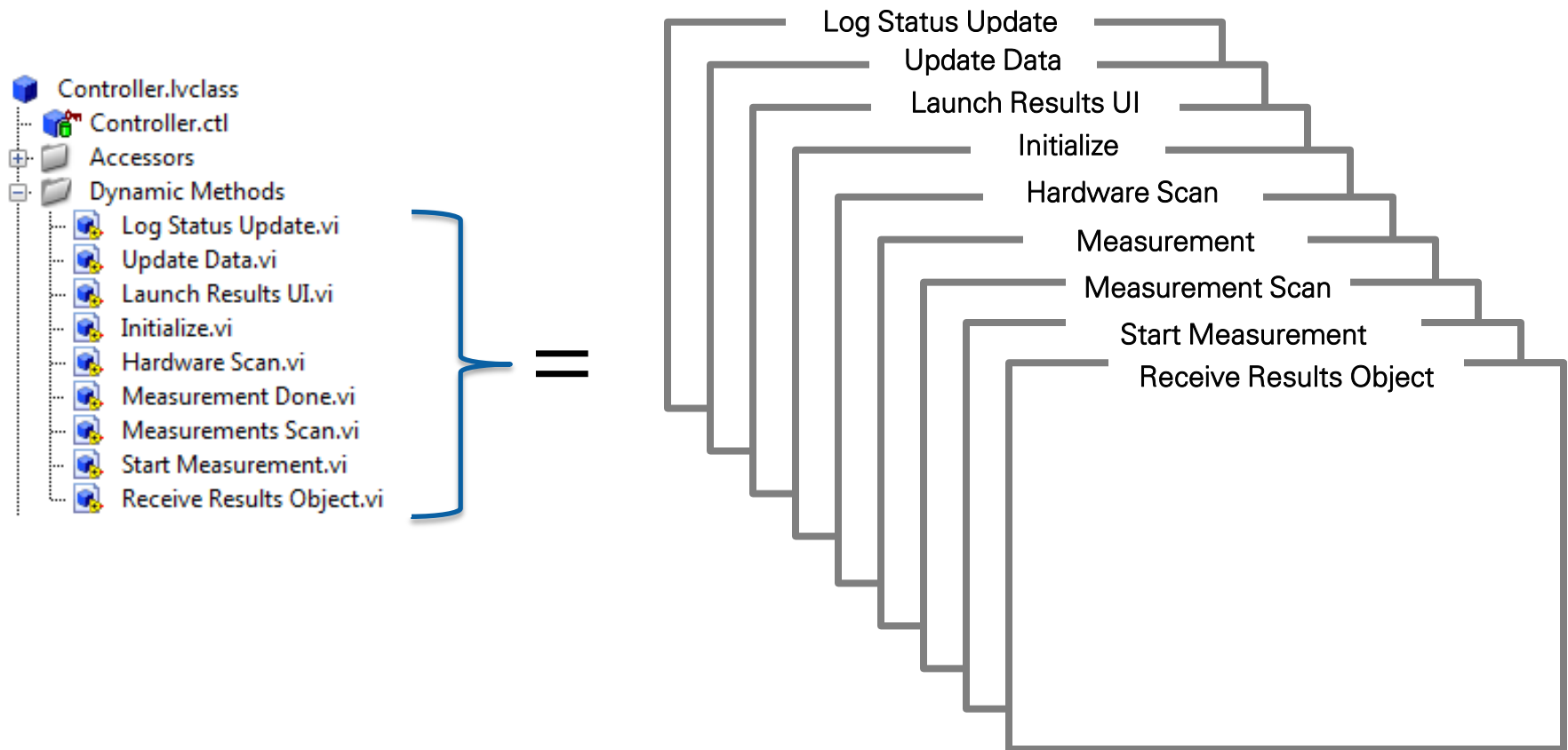


# Dynamic Dispatch (Polimorfismo)

A ligação dinâmica pode ser imaginada como um “polimorfismo no tempo de execução”. O objeto que você passa em um terminal de ligação dinâmica determina o método específico a ser invocado no tempo de execução.

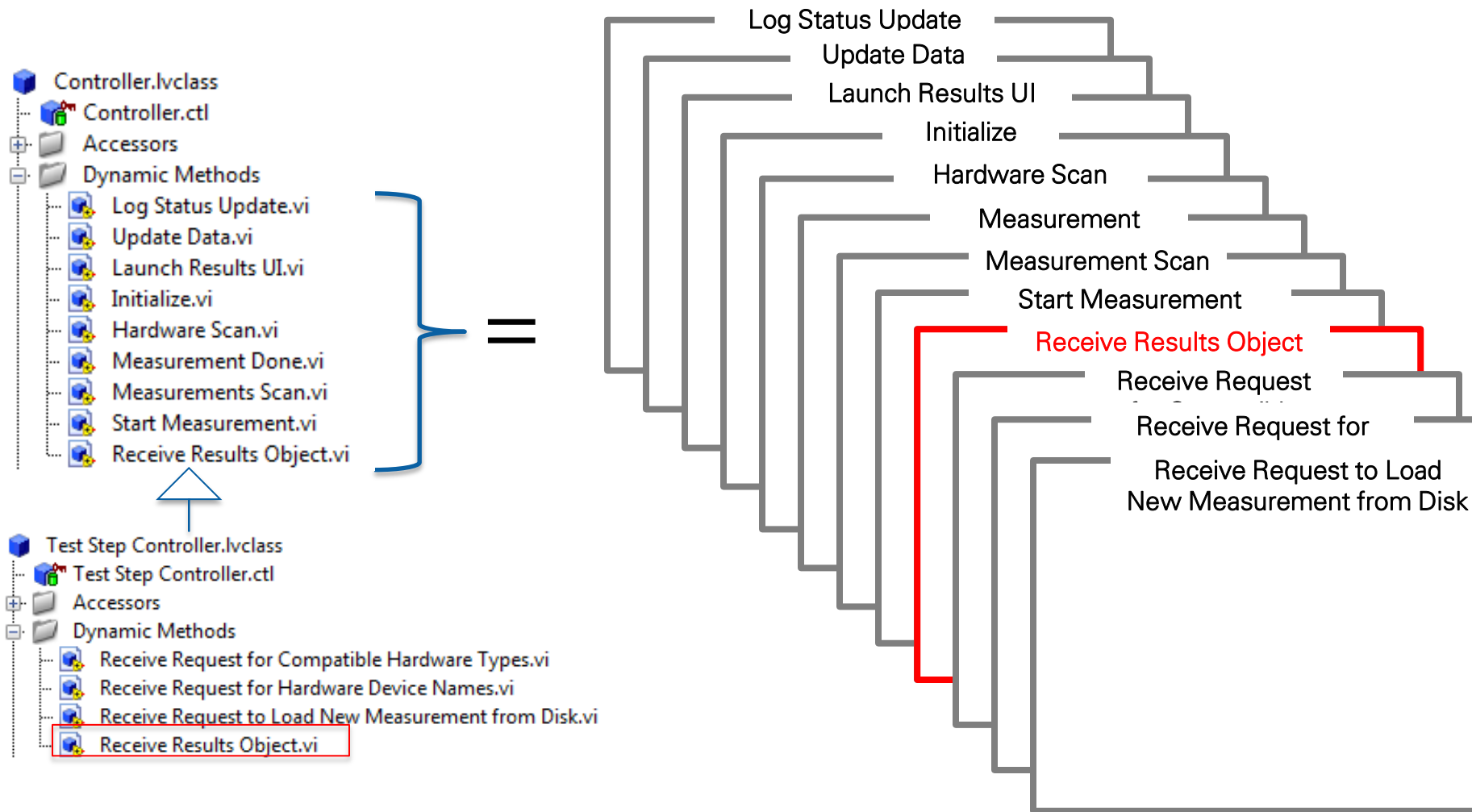


# Os métodos de um ator são executados quando uma mensagem é recebida



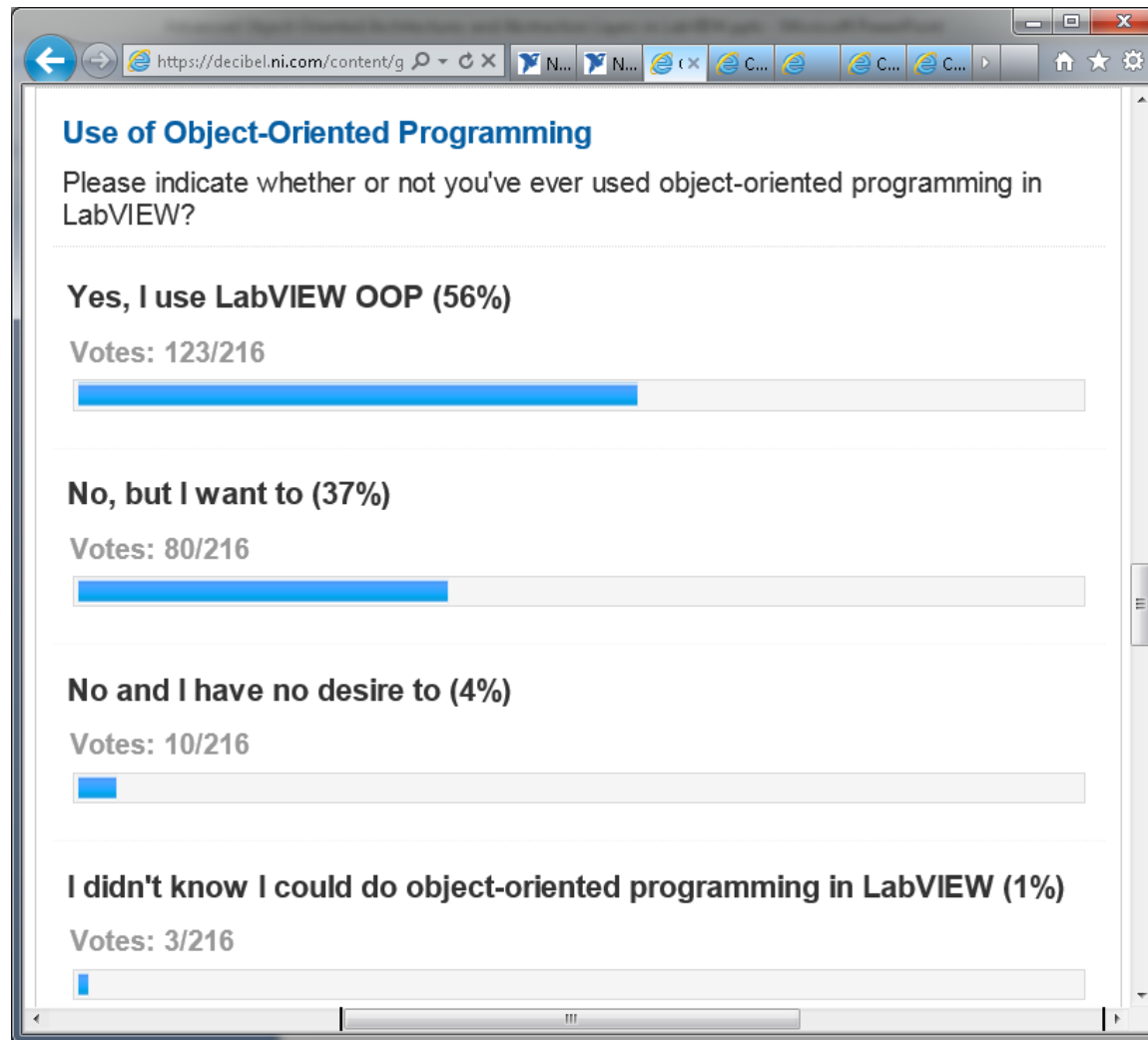
Métodos do Ator definem as mensagens que podem ser consumidas por ele mesmo

# Mensagens podem ser alteradas ou adicionadas pela classe filho



# Demonstração – Feedback Evaporative Cooler

# Comunidade: LabVIEW Development Best Practices



<https://decibel.ni.com/content/groups/large-labview-application-development>  
ni.com

Novo Usuário

Usuário Experiente

Usuário Avançado

LabVIEW Core 1

LabVIEW Core 2

LabVIEW Core 3

LabVIEW Connectivity

Object-Oriented Design  
and Programming in LabVIEW

LabVIEW Performance

Managing Software  
Engineering in LabVIEW

Advanced Architectures  
in LabVIEW

Certificações

Certified LV Associate  
Developer

Certified LabVIEW  
Developer

Certified LabVIEW  
Architect

Outros Treinamentos

LabVIEW Real-Time 1  
LabVIEW Real-Time 2

DAQ & Signal Conditioning  
LabVIEW FPGA

LabVIEW Machine Vision



# Baixe exemplos e apresentações

[ni.com/largeapps](http://ni.com/largeapps)



Ferramentas de engenharia de software



Práticas de desenvolvimento



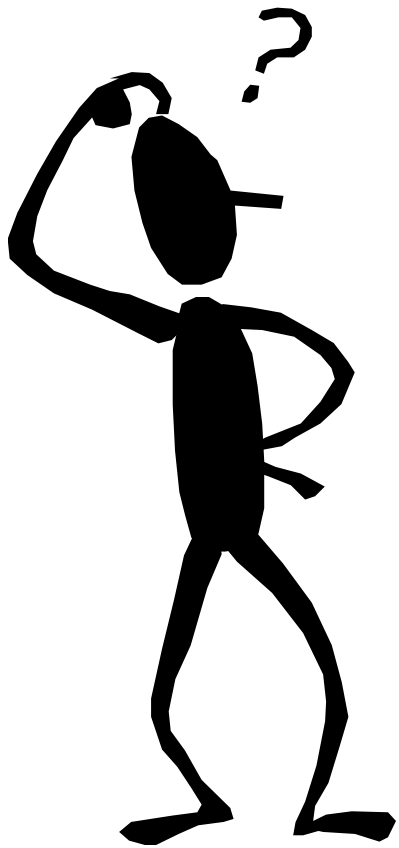
Comunidade para grandes aplicações



Comunidade para grandes aplicações



Práticas de desenvolvimento



Perguntas?



Muito obrigado!