

A decorative pattern of hexagons in various colors (yellow, orange, green, blue, purple) arranged in a honeycomb-like structure, primarily on the left side of the slide.

NIDays09

CONFERÊNCIA TECNOLÓGICA SOBRE
PROJETO GRÁFICO DE SISTEMAS



Engenharia de software para desenvolvimento com LabVIEW: Orientação a Objetos, Statechart e Validação

André Pereira – Engenheiro de Vendas (Grande São Paulo)

Alexsander Loula – Coordenador Suporte Técnico



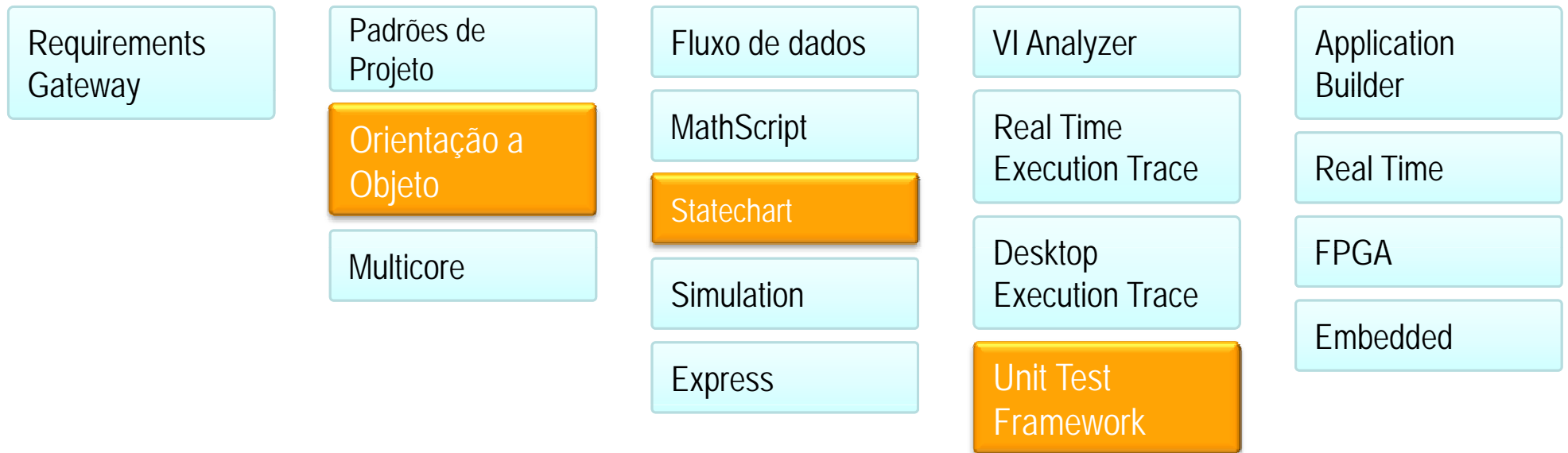
Agenda

- Orientação a Objetos
- Implementação de OO em LabVIEW
- Statechart em LabVIEW
- Validação de Software – Unit Test Framework

Processo de Engenharia de Software



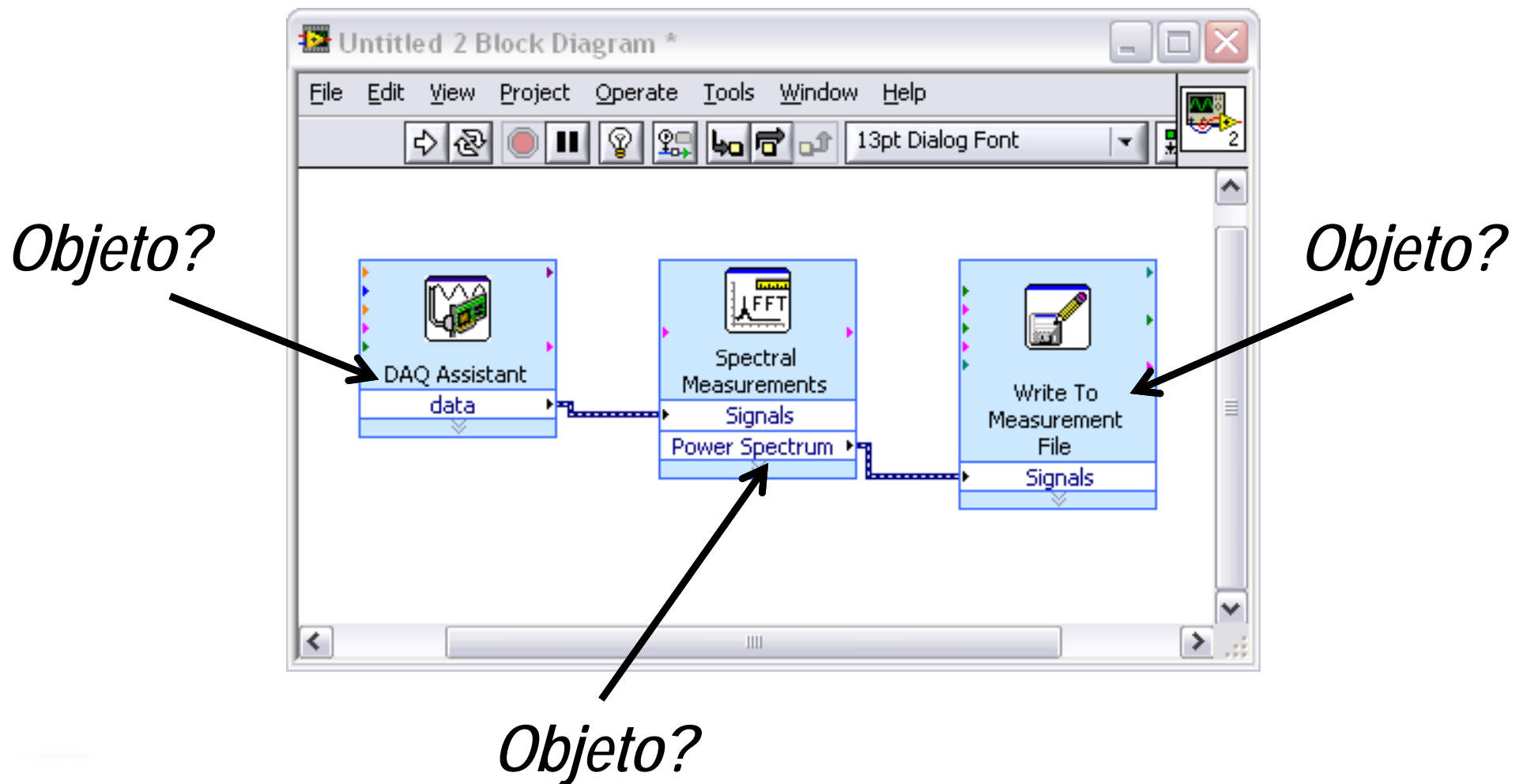
Engenharia de Software e Boas Práticas de Desenvolvimento



..."It's very hard to program extremely badly in
LabVIEW."

(John Conway - A Software Engineering Approach to LabVIEW)

O LabVIEW não foi sempre “Orientado a Objeto”?



O que é Projeto Orientado a Objeto?

Projeto Orientado a Objeto requer que programadores pensem em um programa em termos de objetos, ao invés de processos.

Um objeto é composto de dado encapsulado e métodos para acesso a tal dado.

Programação Orientada a Objeto (POO) utiliza objetos e suas interações para desenvolver aplicações.

POO é baseado em técnicas de programação tais como encapsulamento, herança e polimorfismo.

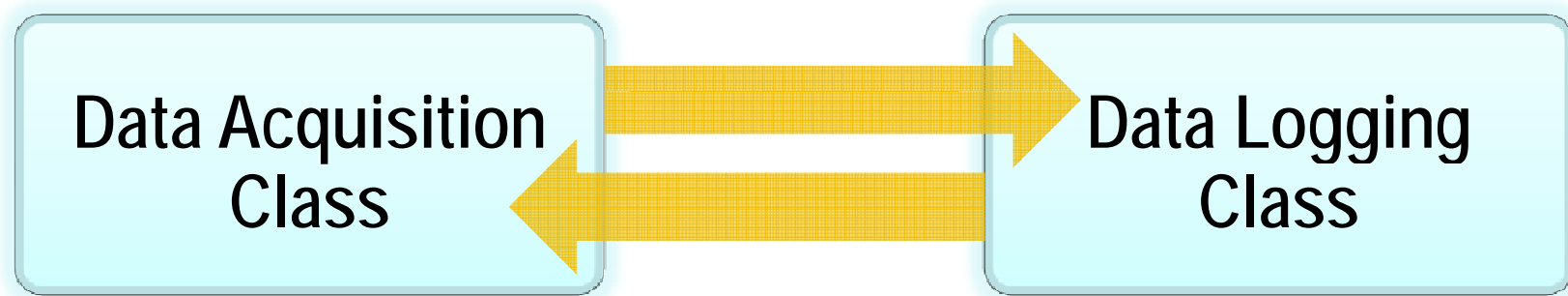
Quando utilizar POO?

- Utilize POO quando precisar de:
 - Encapsulamento
 - Herança
 - *Dynamic/Virtual dispatching* (polimorfismo)
- Benefícios de POO:
 - Manutenção de código reduzida
 - Reuso de código maximizado
 - Extensão de software/Adição de funcionalidades mais simples

Exemplo: Grande Aplicação de Teste

Um objeto pode se comunicar com outro sem o conhecimento de sua organização interna

- Estrutura interna pode mudar com o tempo
- Interfaces (métodos públicos) permanecem os mesmos



Linguagens orientadas a objeto

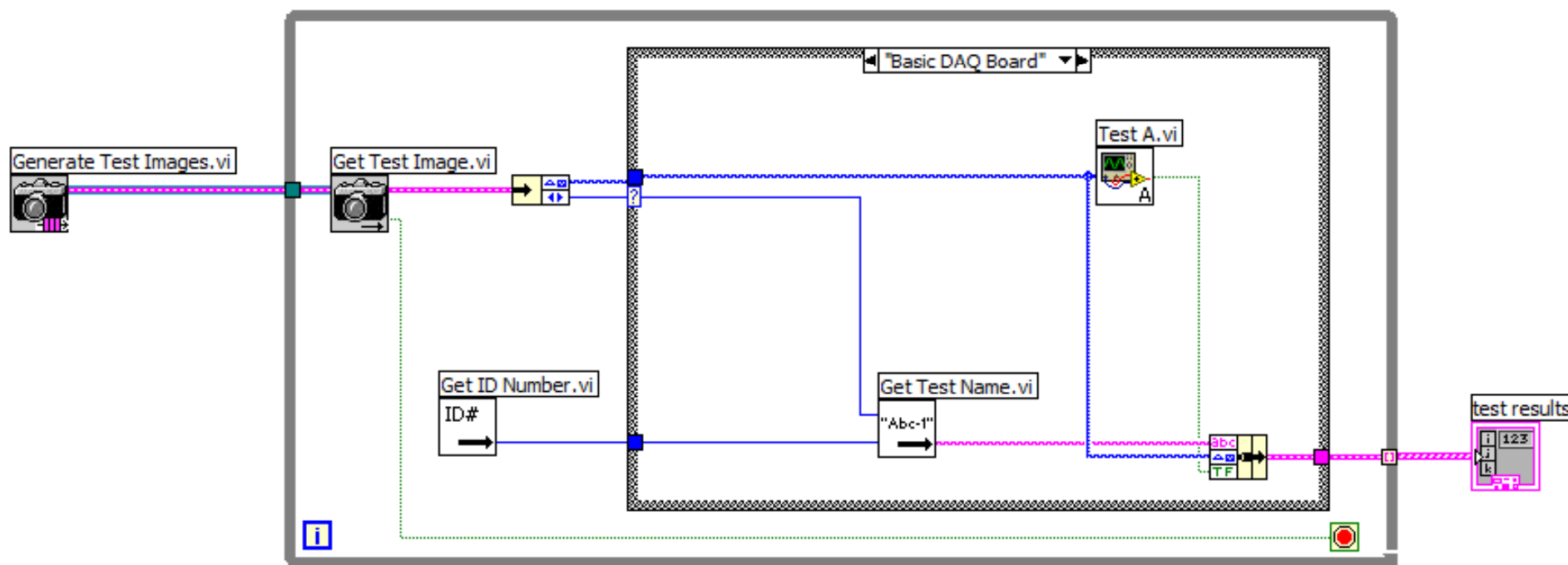
- C++
- C#
- Java
- Objective-C
- Perl
- Python
- LabVIEW 8.20 e superior

Exemplo: Teste em placa eletrônica



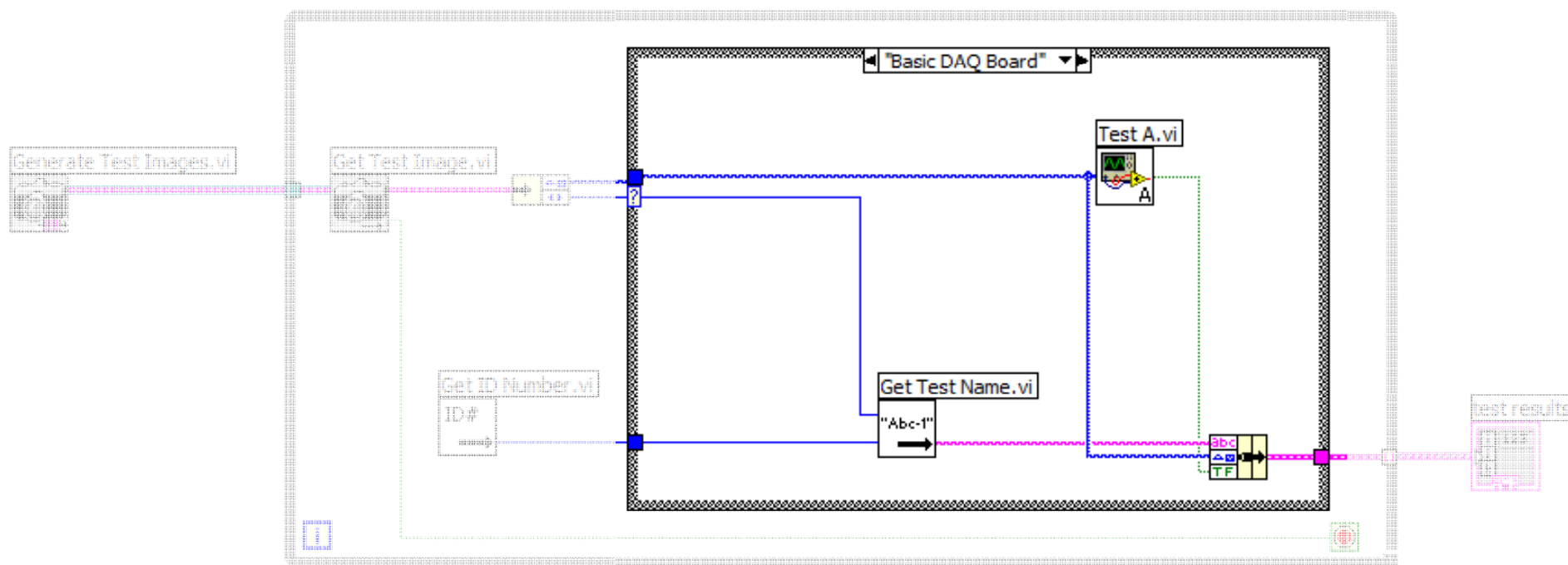
- Cenário
 - Sistema de teste para placa eletrônica baseado em LabVIEW
- Requisitos
 - Diferentes tipos de placas precisam ser testadas
 - Novos modelos serão adicionados no futuro
- Objetivos
 - Maximizar o reuso de código e escalabilidade do sistema

Solução tradicional baseada em processos



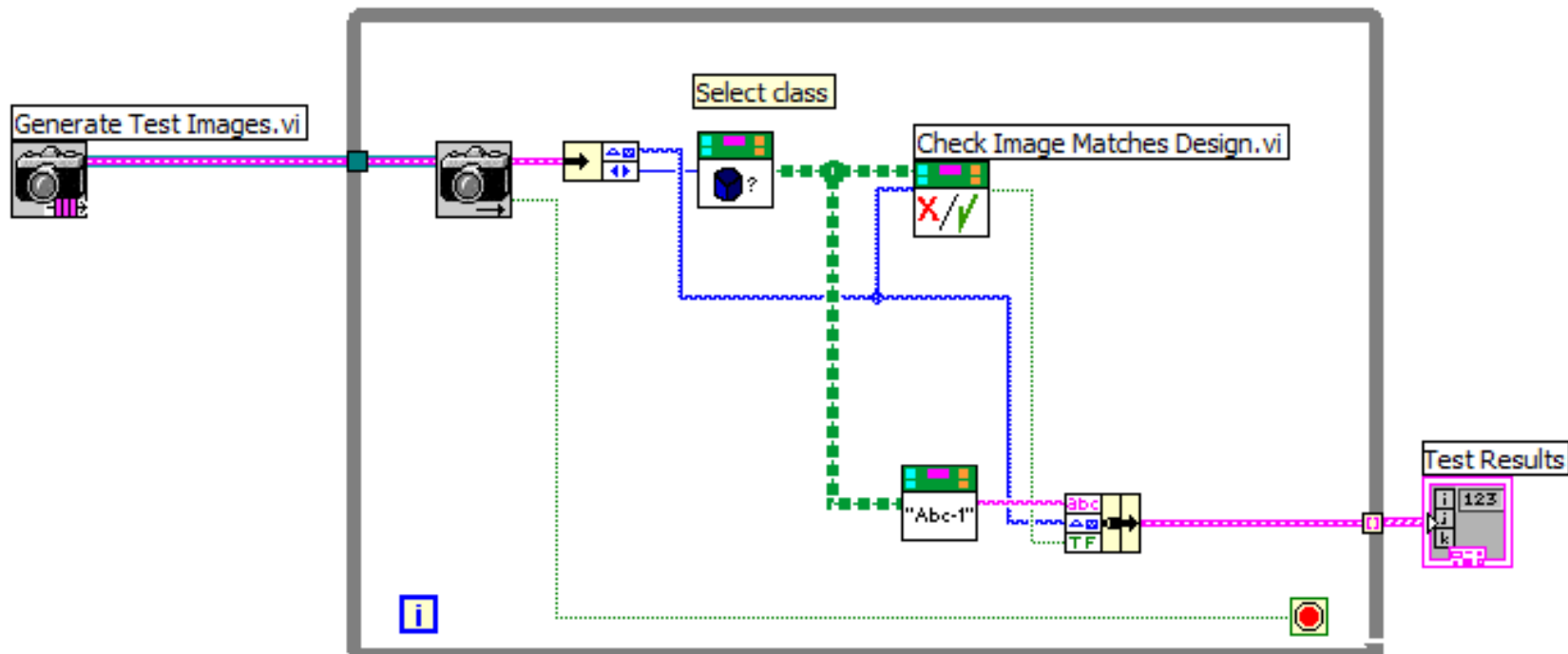
- Estrutura Case dentro de laço While (um caso por placa)
- Não é facilmente escalável; novas placas significariam novos cases

Solução tradicional baseada em processos



- Estrutura Case dentro de laço While (um caso por placa)
- Não é facilmente escalável; novas placas significariam novos cases

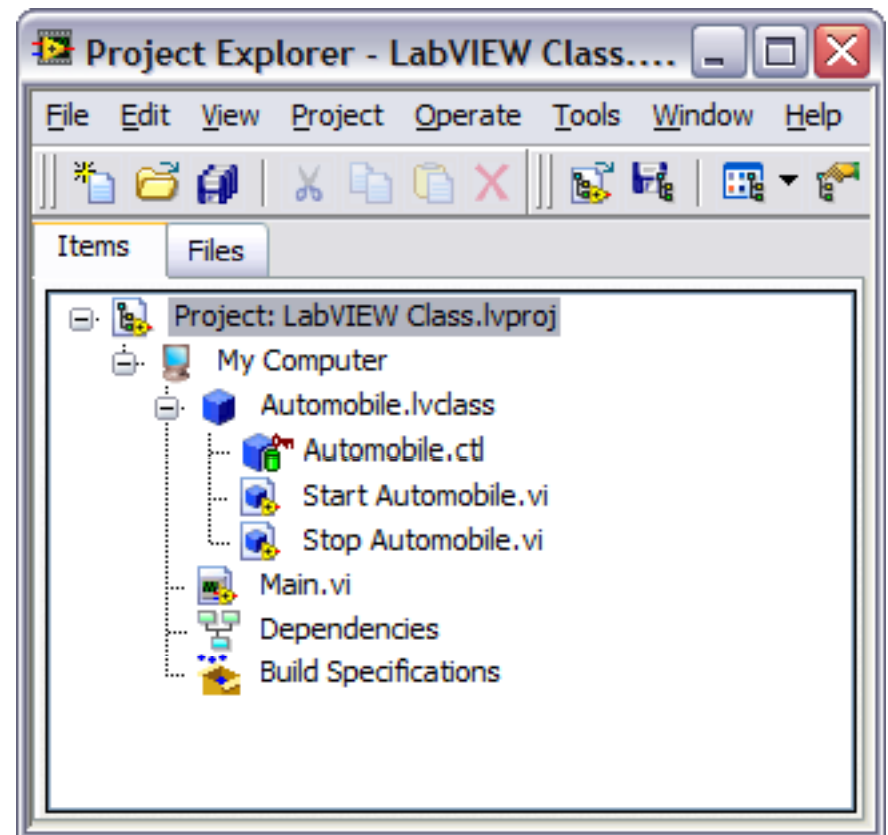
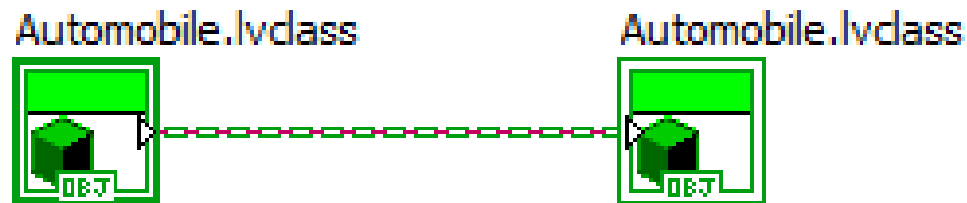
Solução baseada em Objetos



- Baseado em um laço While, mas sem a estrutura Case
- Novas placas são adicionadas sem alteração no programa principal

O que é uma classe em LabVIEW?

- Um tipo especial de cluster
- Um tipo de dado definido pelo usuário
- Um tipo de biblioteca de projeto em LabVIEW



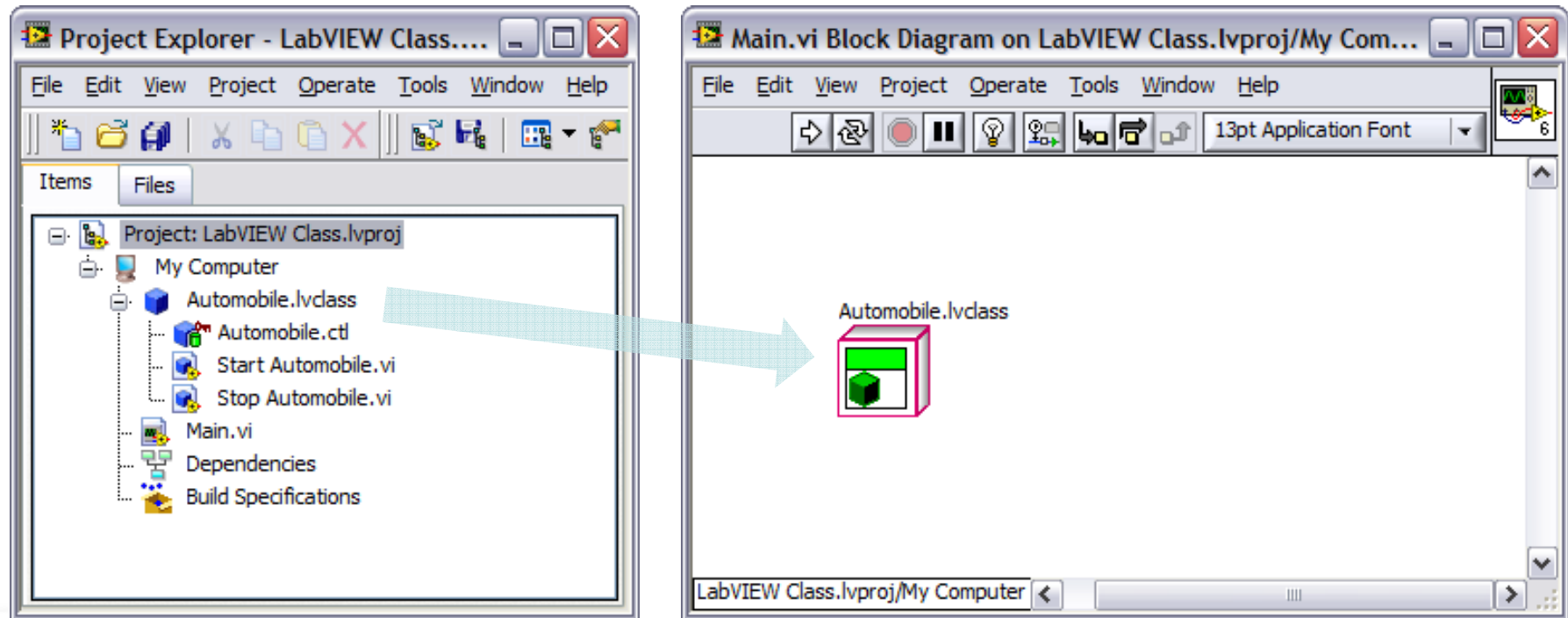
Anatomia de uma classe em LabVIEW

- Cada classe em LabVIEW possui:
 - Um controle contendo dados (cluster)
 - VIs Membro para acessar tais dados
- O arquivo Class (.lvclass) armazena informação da classe
 - Definição dos dados internos à classe
 - Lista de VIs membro
 - Propriedades do VI membro



O que é um Objeto?

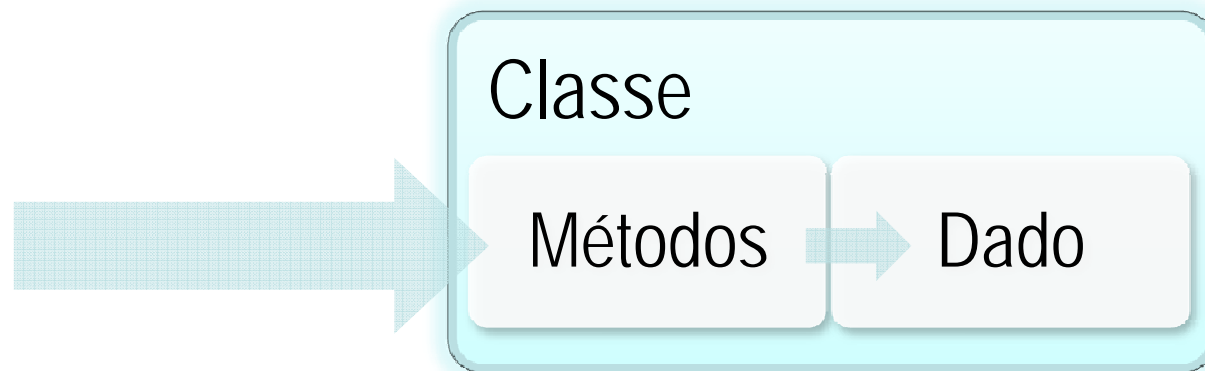
- Um objeto é uma instância específica de uma classe
- Os dados e métodos de acesso são definidos pela classe



O que é Encapsulamento?

Consolidação de dados e métodos em uma classe, com acesso restrito aos dados

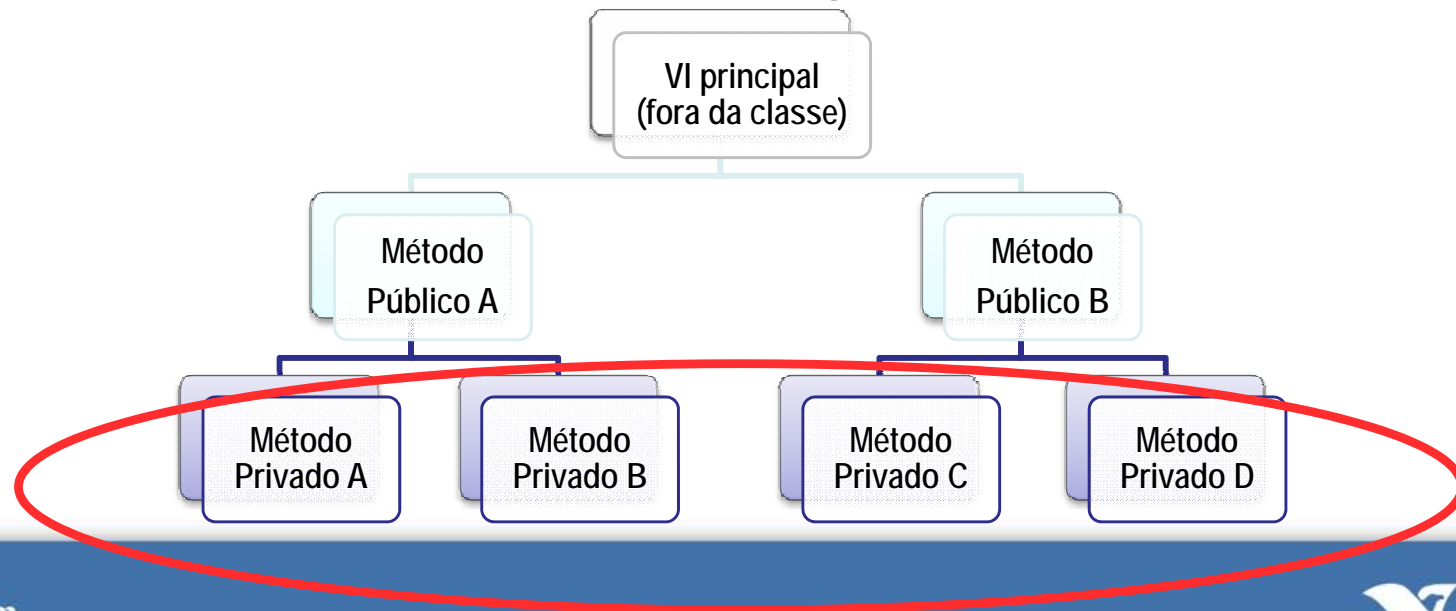
- Oculta informação sobre como um objeto realiza suas tarefas; muda o foco para quais tarefas ele executa
- Difícil implementação sem suporte da linguagem



Encapsulamento em LabVIEW

Classes em LabVIEW podem ser de três tipos de VIs

- Públicos: Chamados por qualquer VI como um subVI
- Privados: Chamados apenas por VIs pertencentes à mesma classe
- Protegidos: Chamados apenas por membros de subclasses da mesma hierarquia



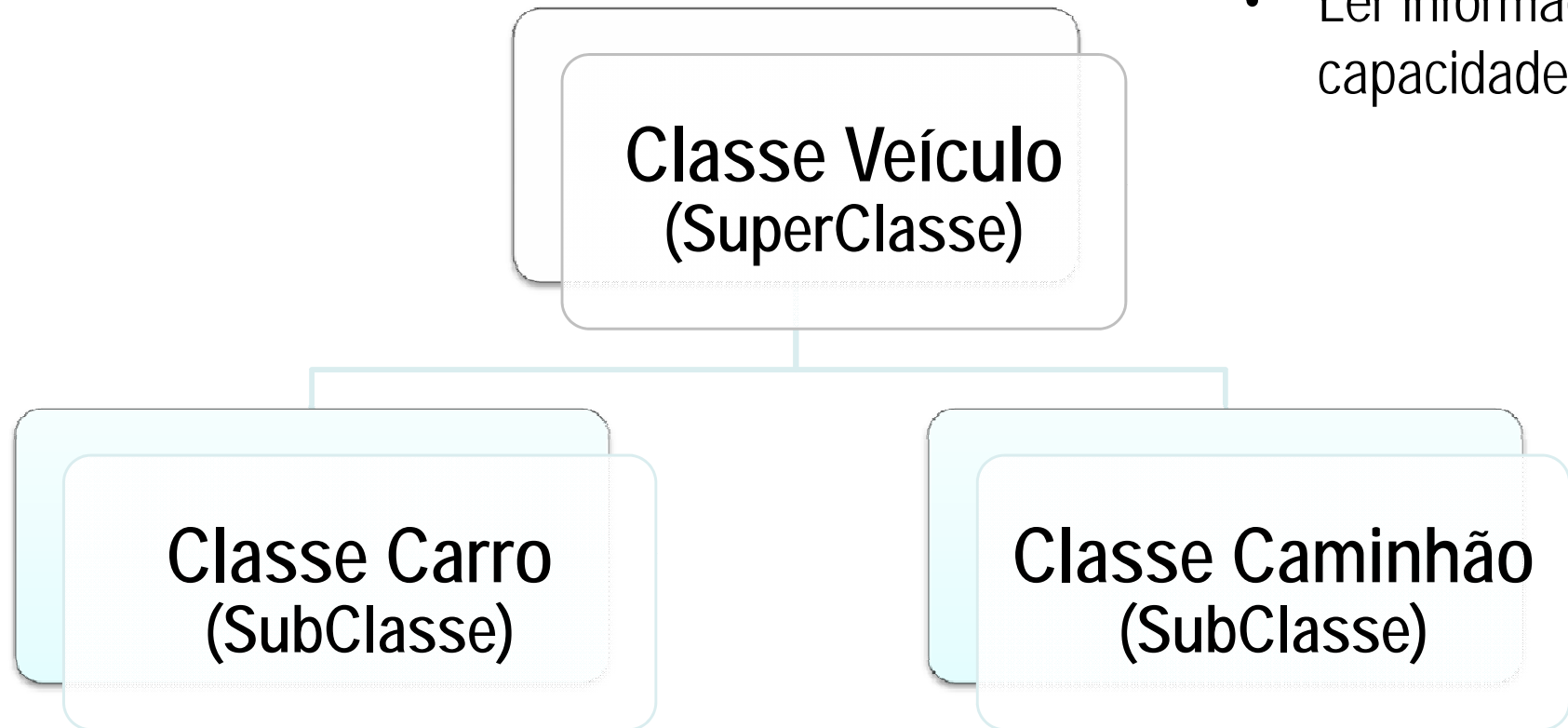
Revisão: Clusters x Classes

- Clusters
 - *Unbundle/Bundle* e *Unbundle/Bundle By Name* oferece acesso irrestrito aos dados
- Classes LabVIEW
 - Apenas VIs na mesma classe podem ler e escrever dados (o dado é encapsulado)

O que é herança?

Exemplos de métodos:

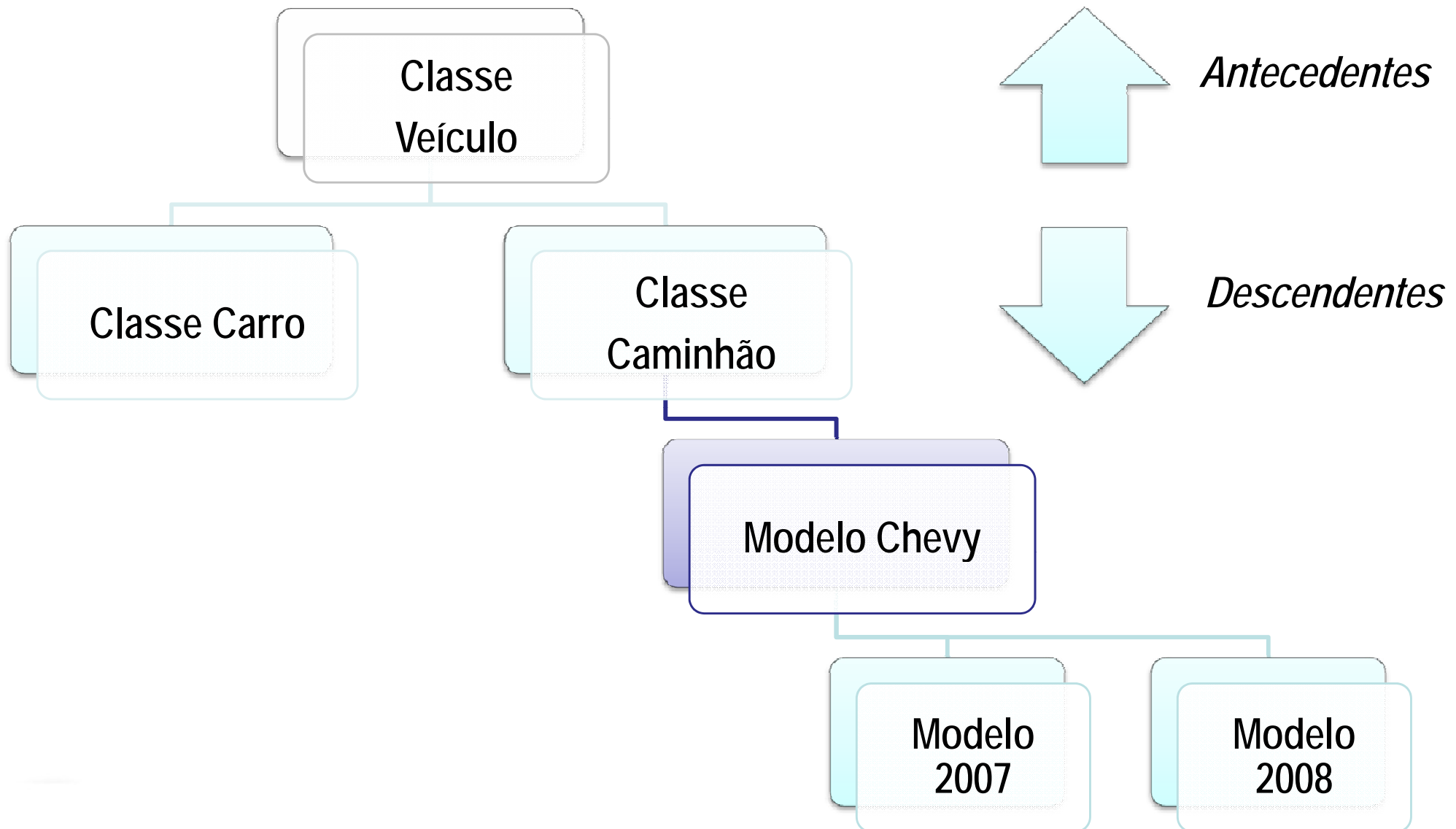
- Inicializar
- Ler informação de capacidade



Um carro é um tipo de veículo.

Um caminhão é um tipo de veículo.

Exemplo de Herança



Herança em LabVIEW

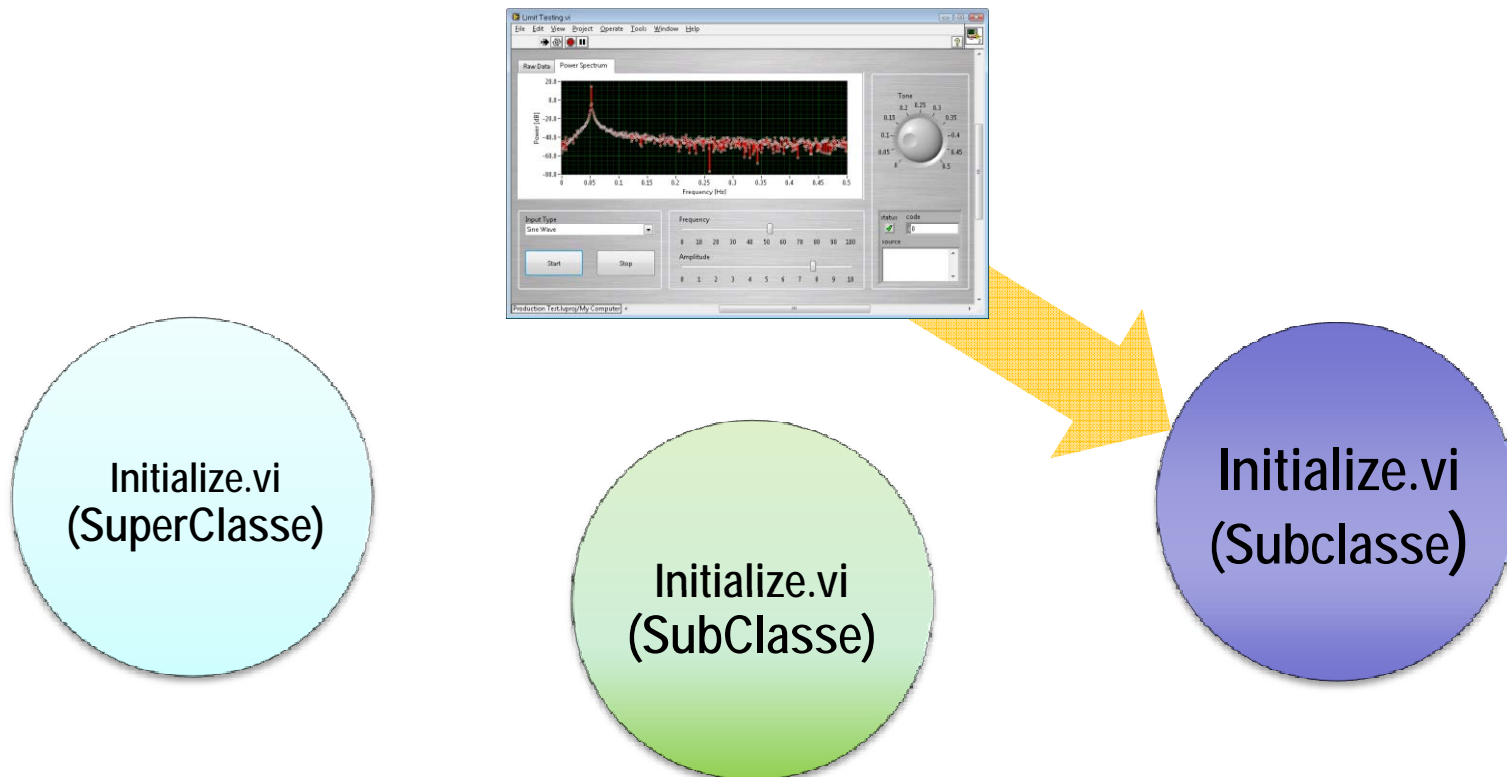
- Classe existente é o ponto inicial para uma nova classe
 - Classe existente é a SuperClasse
 - Nova classe é SubClasse
 - SubClasse reusa ou estende a funcionalidade da SuperClasse
- Benefícios
 - O reuso de código é combinado com especialização
 - Alterações na SuperClasse são propagadas para as SubClasses

Chamada Estática x Dinâmica (*Dispatching*)

- **Método Estático** – Definido por um único VI através da hierarquia de classes
- **Método Dinâmico** – Definido por múltiplos VIs com o mesmo nome através da hierarquia de classes
 - Público
 - Protegido

Métodos de chamada dinâmica

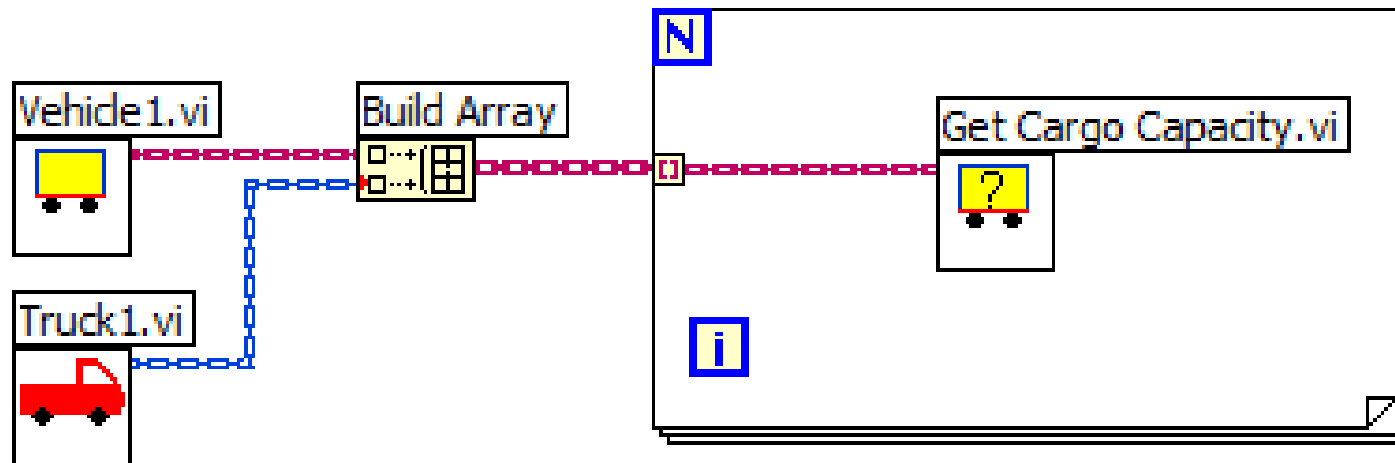
O VI determina qual versão da subVI será utilizada em tempo de execução.



Chamada Dinâmica

VI de SubClasse sobrepõe a funcionalidade do VI da SuperClasse

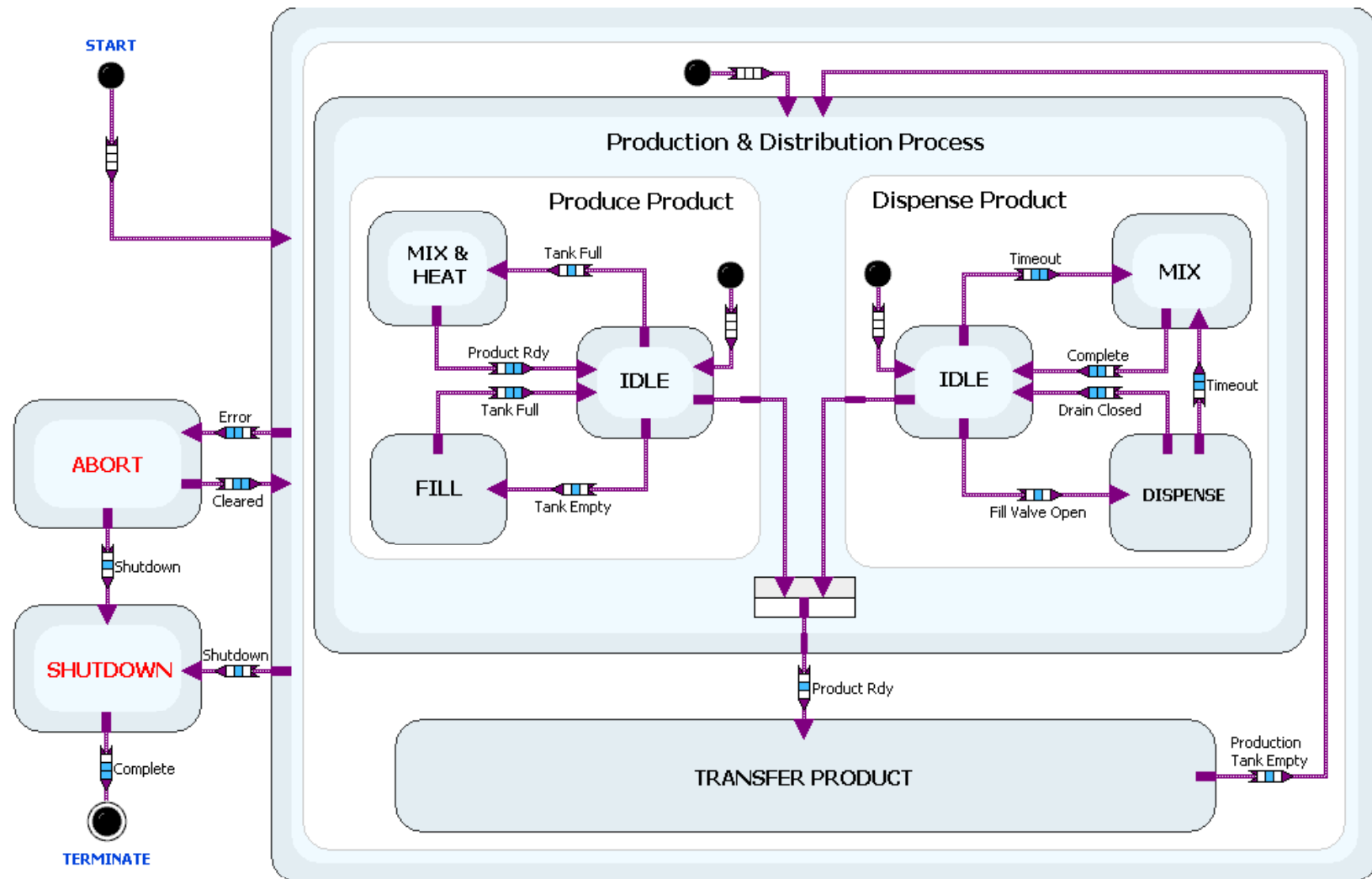
- São definidos vários VIs com o mesmo nome (um em cada nível dentro da hierarquia de classes)
- Valor do fio em tempo de execução determina qual versão do VI será chamada



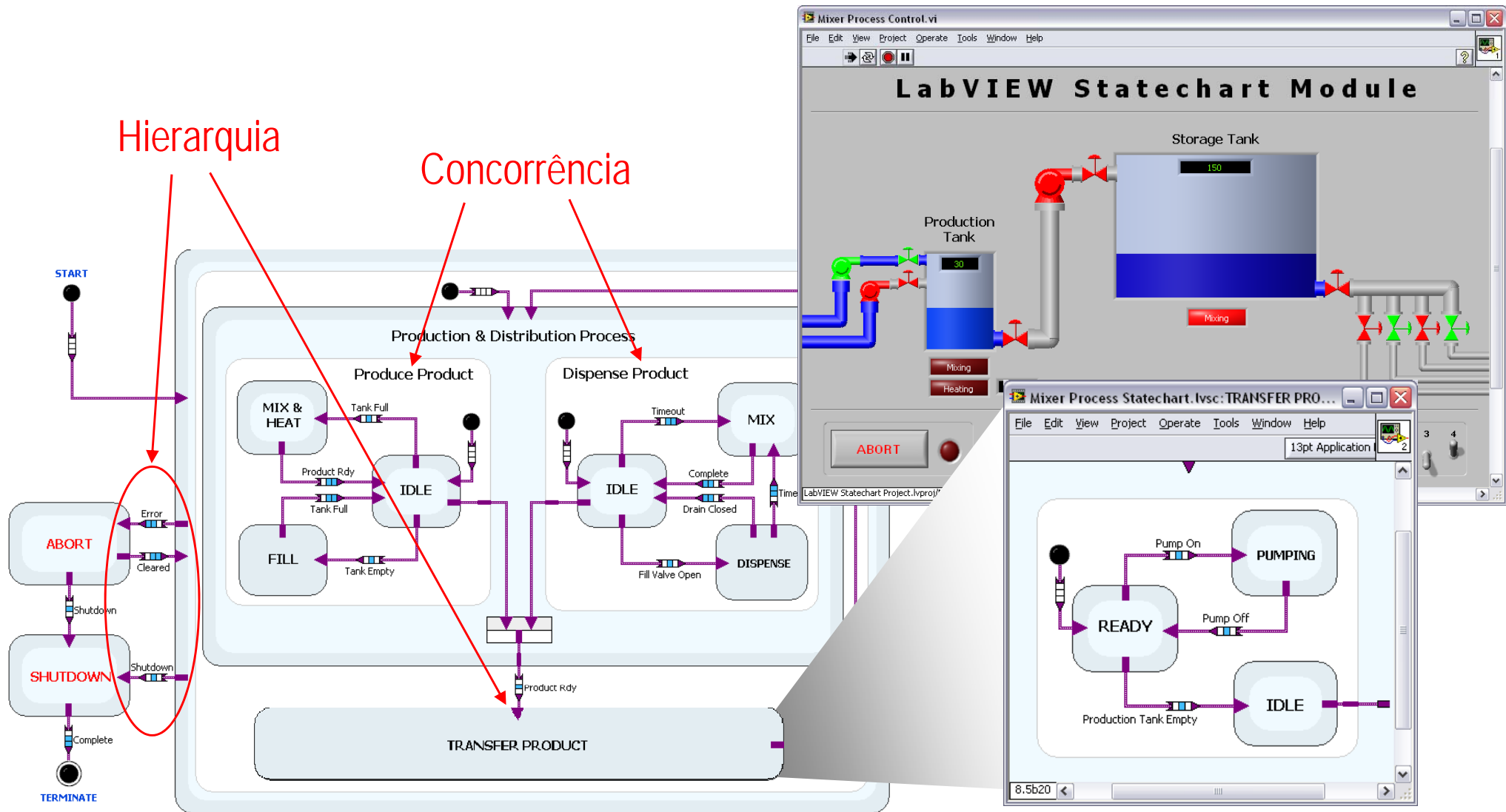
Demo

- Classes em LabVIEW
- Encapsulamento
- Herança

Desenvolvendo Aplicações com o Novo Módulo LabVIEW Statechart



Controle de Máquinas e Processos

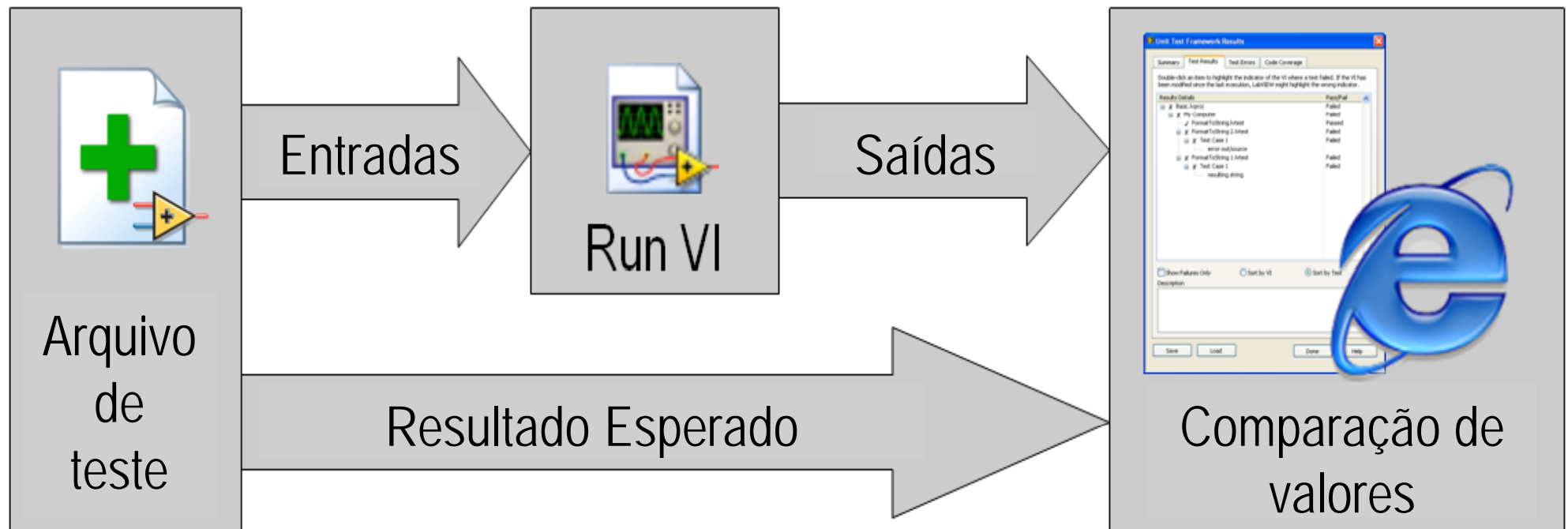


Benefícios do Statechart

- Abstração
 - Semântica simples para representar sistemas complexos
 - Visão de sistema
 - Documentação
- Escalabilidade
 - Fácil ampliação de aplicações
 - Plataforma de software aberta
- Geração de Código Automático
 - Tecnologia LabVIEW embedded

Demo: Ventilador de Teto com Iluminação

LabVIEW Unit Test Framework



Unit Test Framework

- Gerar e executar testes para VIs rapida e facilmente
- Garantir que o código é seguro, confiável e tolerante a falha
- Rastrear a porcentagem do código que foi executada
- Definir testes fora do ambiente de desenvolvimento
- Verificar cobertura dos requisitos de testes



DEMO: Criar Vetores de Testes com o LabVIEW Unit Test Framework

Resumo

- Os conceitos de Orientação a Objetos implementados em LabVIEW podem auxiliar e muito em:
 - Arquitetura de aplicações de médio e grande porte
 - Reuso de código
 - Manutenção de aplicações
- O módulo Statechart é uma poderosa ferramenta para arquitetura e modularização de código em LabVIEW
- Prove de forma eficiente que seu código FUNCIONA utilizando o Unit Test Framework

Obrigado!

Não esqueça de preencher a avaliação.

Para mais informações acesse ni.com ou
ligue para (11) 3149-3149