Andrei Zagorodni

Stockholm

From Interfaces to Full-Scale Multiple Inheritance

Interfaces introduced new dimension in LabVIEW OOP architectures. Full-scale multiple inheritance could offer even more possibilities. The presentation describes solution and toolkit for implementing multiple parent classes and class branches. The basic idea is new thus feedback and brainstorming following the presentation would contribute to further concept development.

Andrei Zagorodni

- CLA, PhD
- Works with LabVIEW from 1994 or 1995
- Works only with LabVIEW from 2009

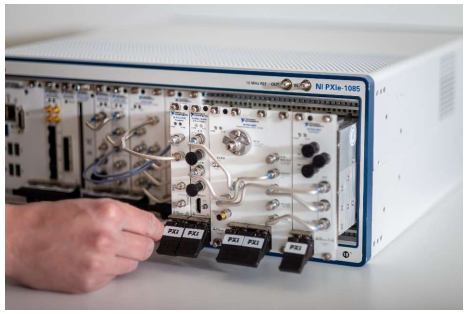
Main area or interests:

- SW architecture

NOVATOR SOLUTIONS

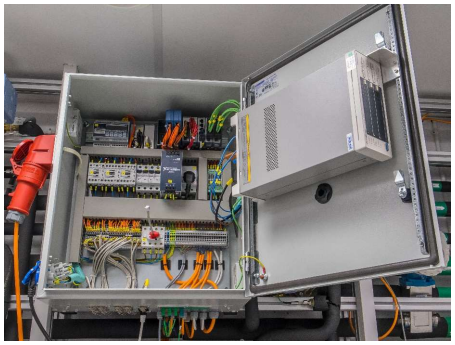
Gold Partner
RF AND WIRELESS

- ✓ Products, Services & Turn-Key Solutions provider
- ✓ ISO 9001 & ISO 14001 certified
- ✓ National Instruments Gold Alliance Partner with RF & Wireless specialty



Spectral Data Analysis (SDA)

- Wideband Recorders
- Multichannel Receivers
- Customized RF/SDR Solutions



Control Automate Test (CAT)

- Control System Development
- Test System Development
- LabVIEW & TestStand Consulting

Remote Measurement (RM)

- Remote Sensors
- IoT
- Telemetry

Content

- Why do we need multiple inheritance?
- Concept of Interface-based classes step-by-step
- AZM Toolkit
- Non-canonical OOP behavior
- Questions and brainstorming

DAQ workstation

Device 1
- Voltmeter

Device 2
- Digital I/O

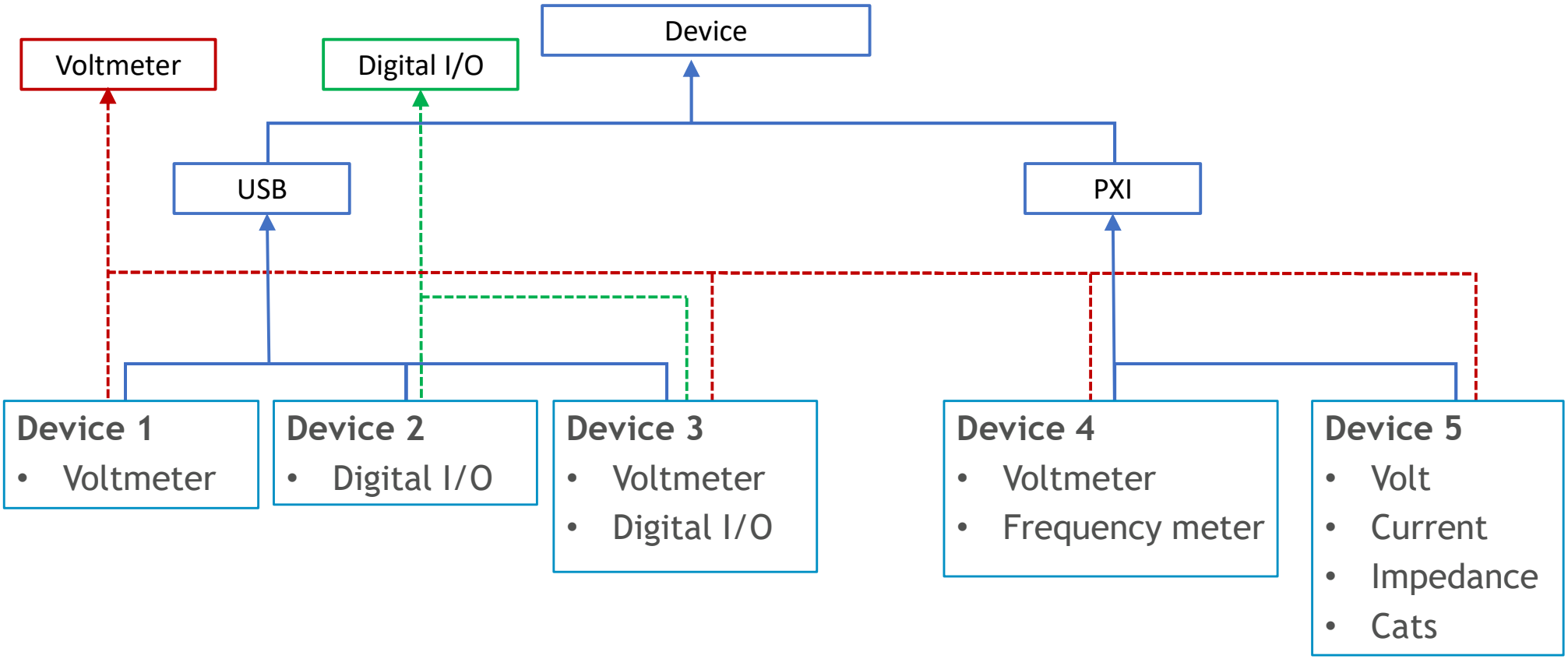
We have problem here!

Device 3
USB-6009
- Voltmeter
- Digital I/O

Device 4
- Voltmeter
- Frequency meter

Device 5
- Voltage
- Current
- Impedance
- Number of cats in GDevCon presentations

Solution? Interfaces! Interfaces?



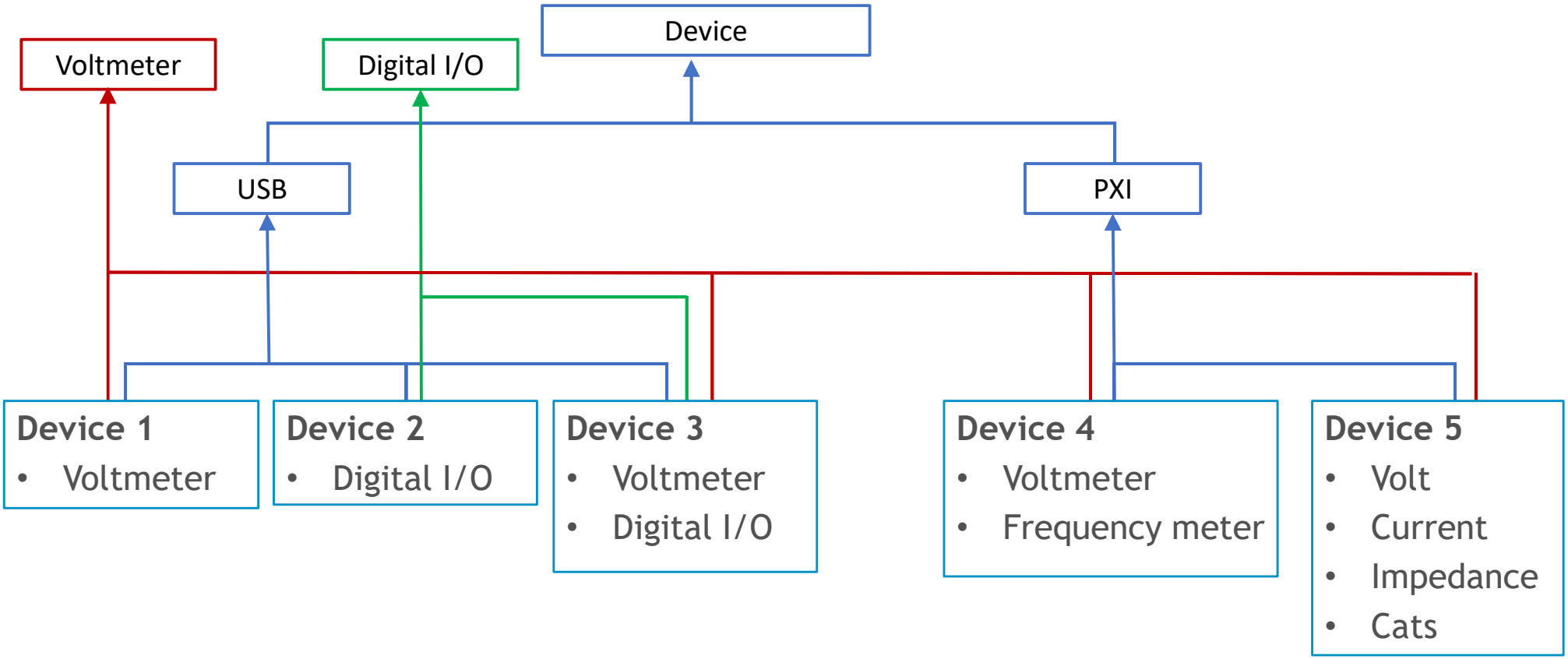
Interfaces

- Class can inherit from many different interfaces
- Even more: interface can have own non-abstract methods;

  i.e., own code
- Unfortunately: interface cannot have attributes
  - Indeed, an interface with attributes is a fully-capable class

Can we solve this single limitation?

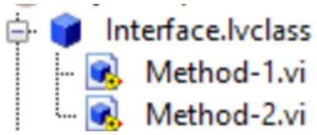
Multiple inheritance

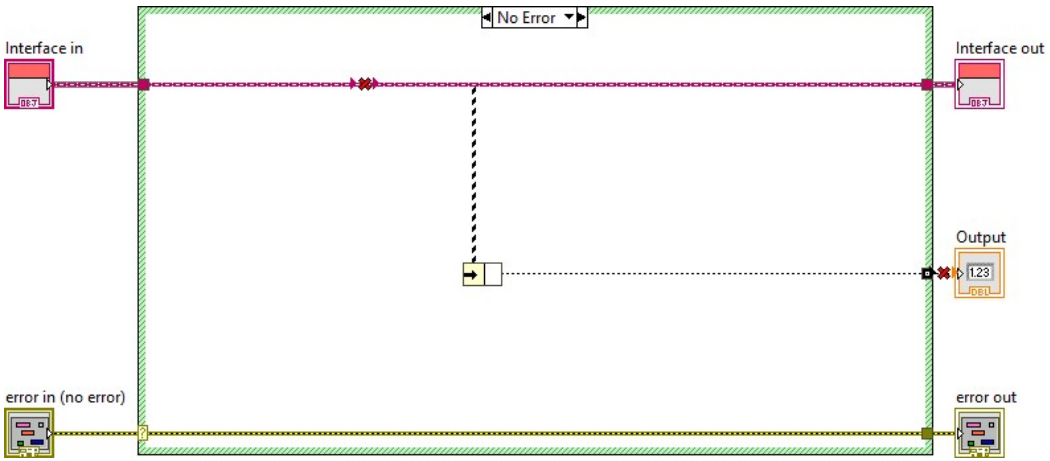


Can we supply an Interface with attributes?

- Using available tools
  - I.e. using conventional LabVIEW code
- Preferably without calls to dll-s or vi.lib
  - Yes, let's do it

LabVIEW Interface

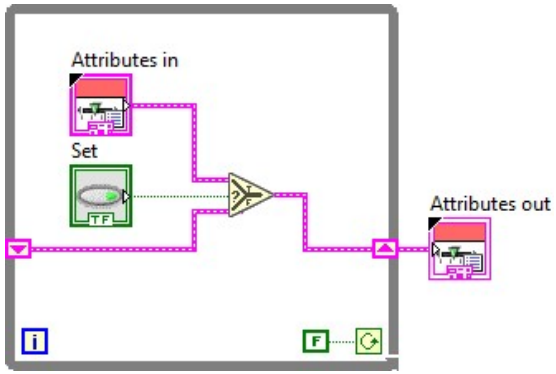




- Descriptor
  - Data type definition
  - Method definitions: names + terminal patterns
- Not only
  - Code

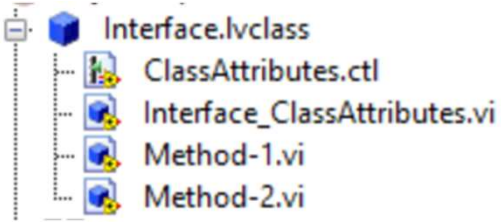
- Is interface a class?
  - **No**
- What is missing?
  - Attributes

Let’s start from FGV (Functional Global Variable)



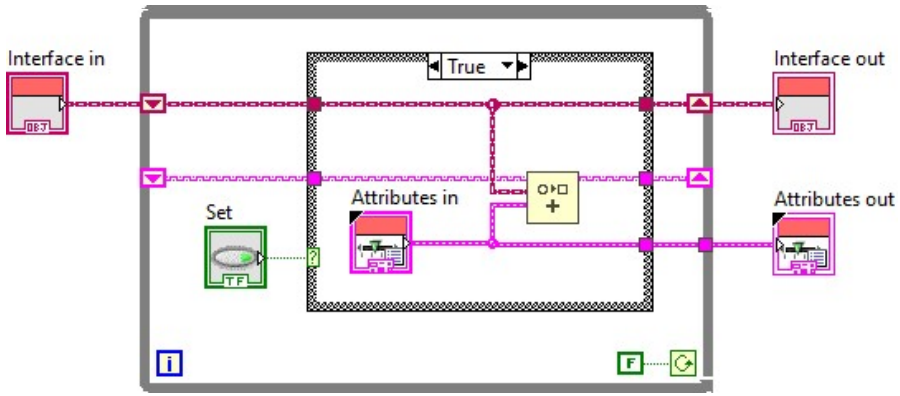
- One set of values for all instances of the class
  - Class attributes

Interface is not interface now but
a Singleton class



- Good achievement but not good enough
- We need own set of attributes for each instance

We need object attributes

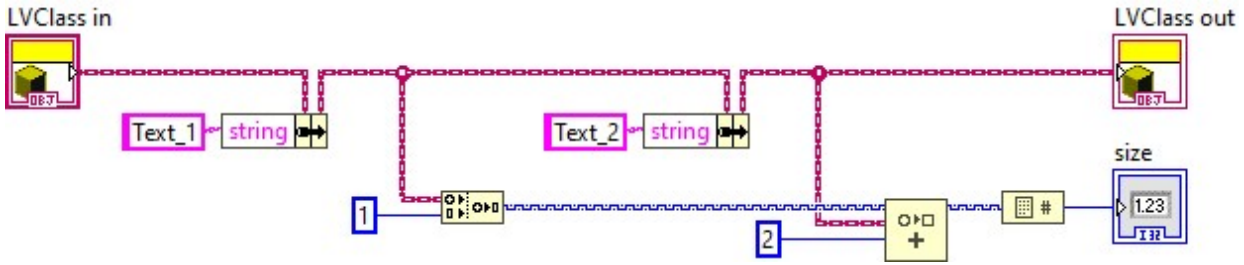


- Map of values
- Type of the key?

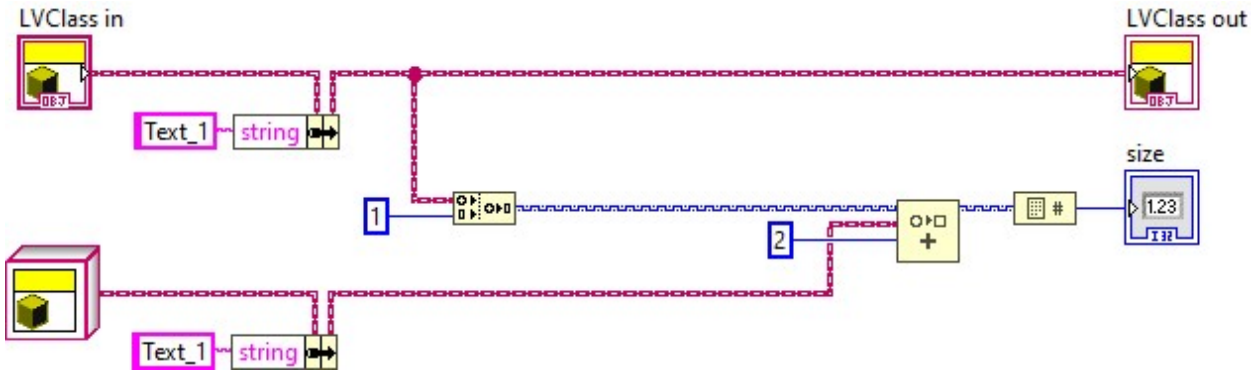
- Interface data type would be perfect
- Is it possible?

- No! Because:
  - The wire ”secretly” carries object data cluster
  - Changing attribute value would alter the key

Indeed: Simple test



size = 2



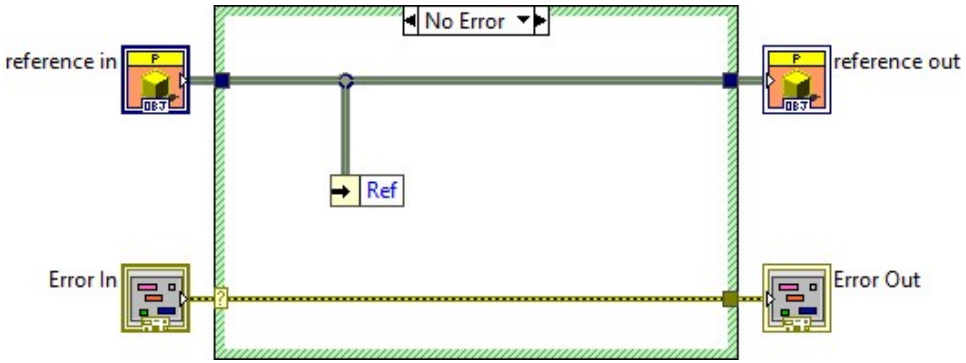
size = 1

Requirements to the key

- Only two requirements
  - Key must be unique for each object
  - Key must not change during whole lifetime of the object

Can LabVIEW class wire satisfy these requirements?

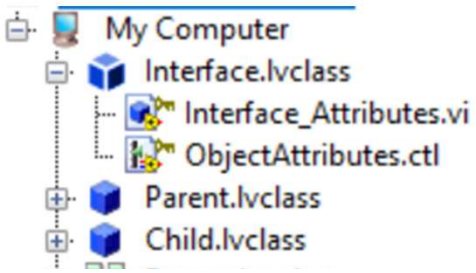
Yes! GOOP classes

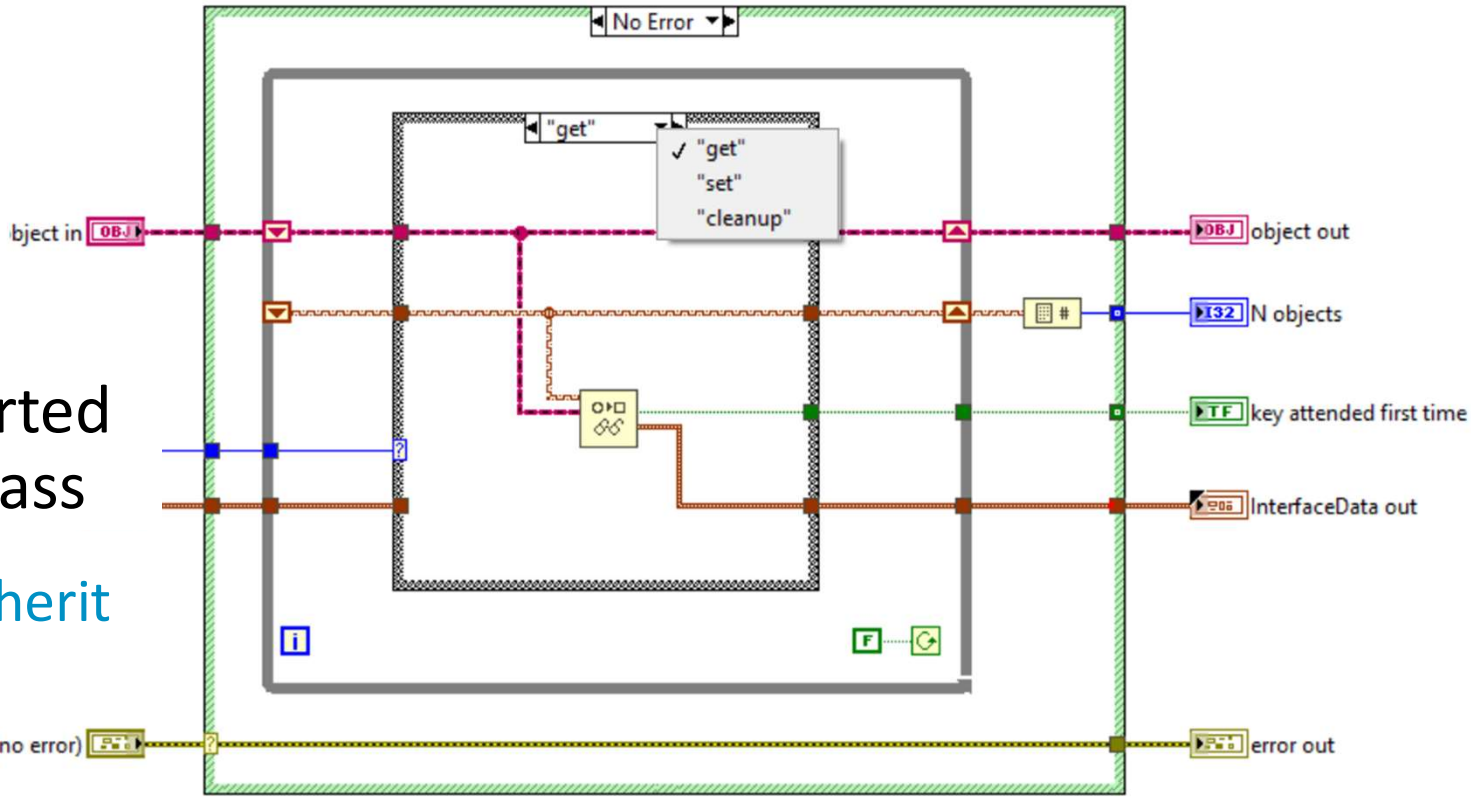


GOOP class wire

- Carries only unique refnum
- The refnum is assigned when the object is created and disposed by object destructor

Interface-based class: FGV+Map-based attribute accessor

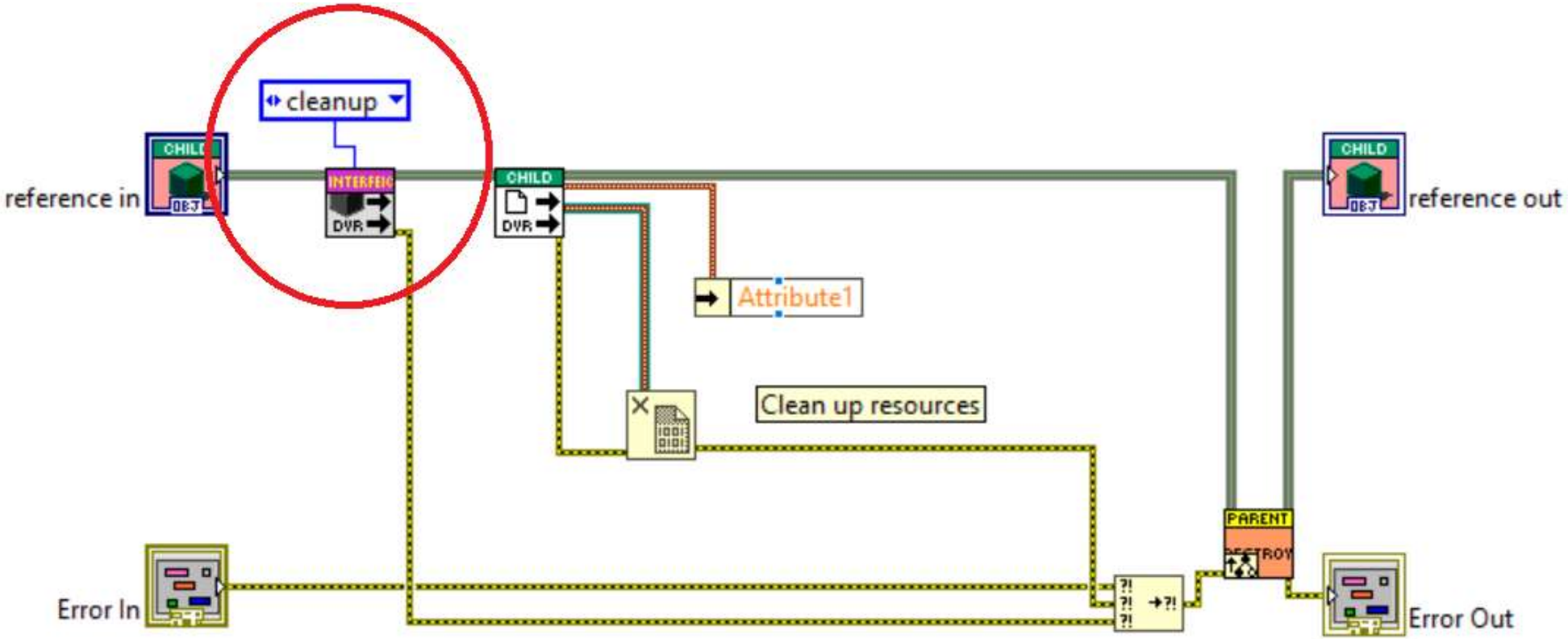




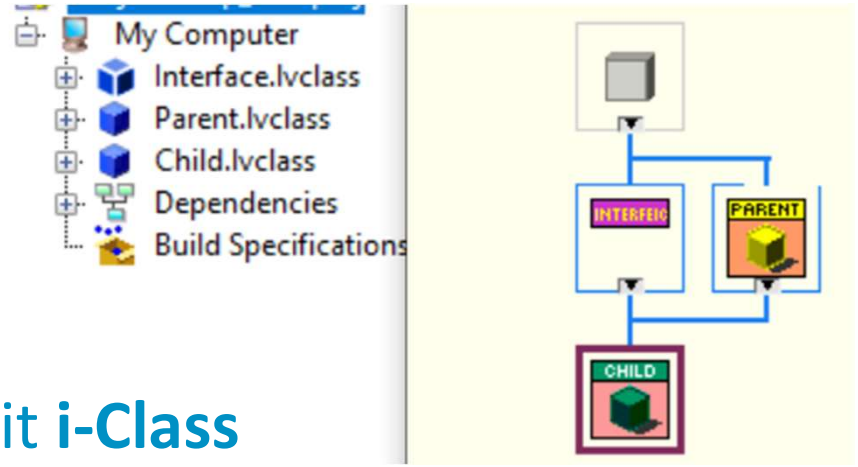
- Interface is converted to fully-capable class

If **only GOOP** classes inherit from such i-Classes

Destructor of GOOP4 class inheriting from i-Class



Inheriting from i-Class

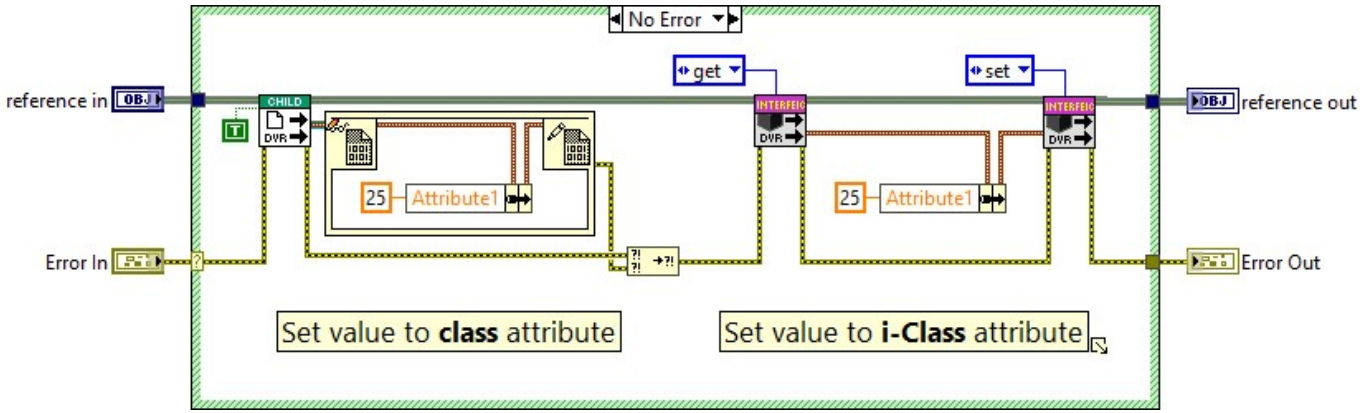


Let's call it **i-Class**

- Is this solution good enough?

Limitation

- There is no attribute-locking
  - Race conditions cannot be prevented but only debugged in the same way as with any other FGV.
- Different approaches for accessing attributes are confusing.

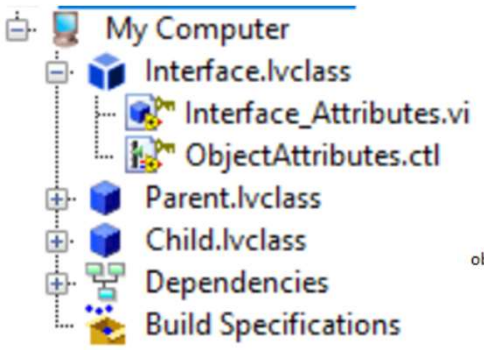


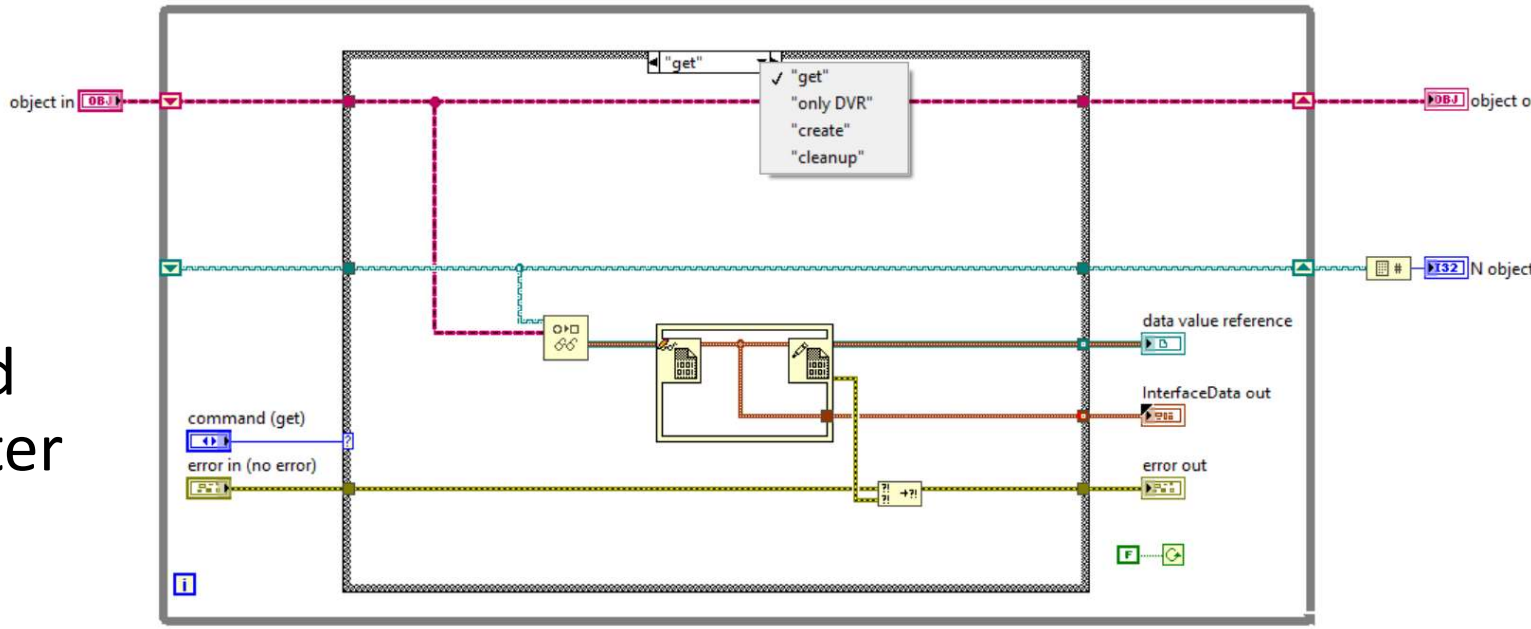
Solution?

Data value references (DVR) in both

- GOOP4 classes
- i-Classes

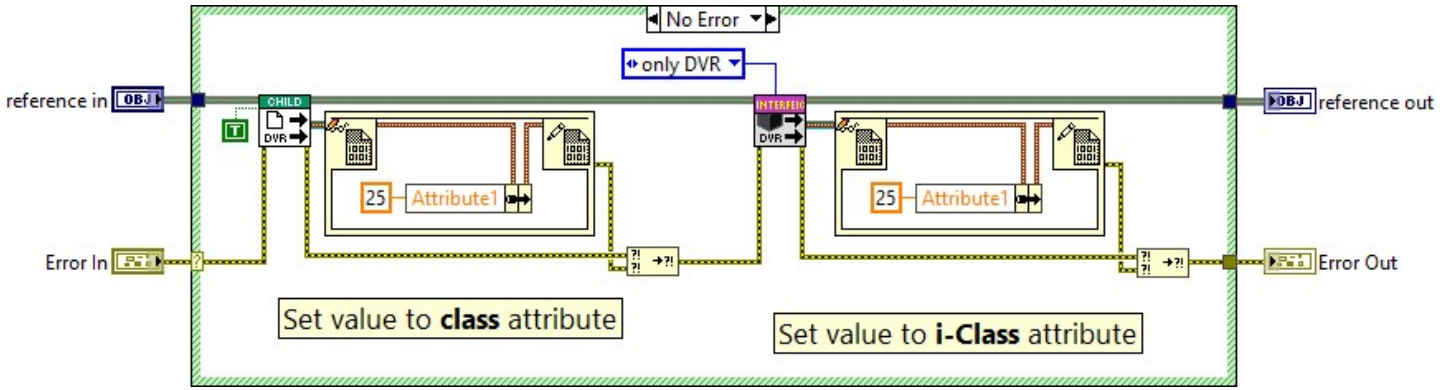
Interface-based class: FGV+Map+DVR-based attribute accessor



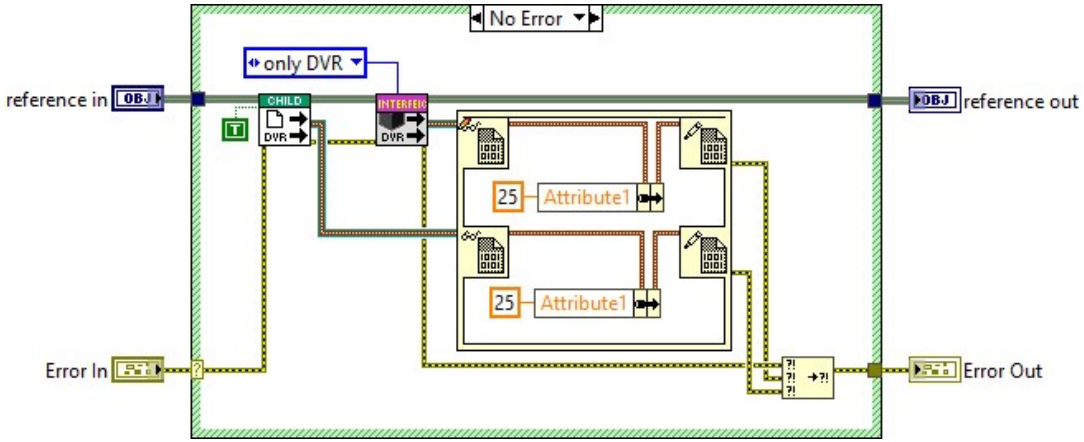


- DVR map instead of attribute cluster map

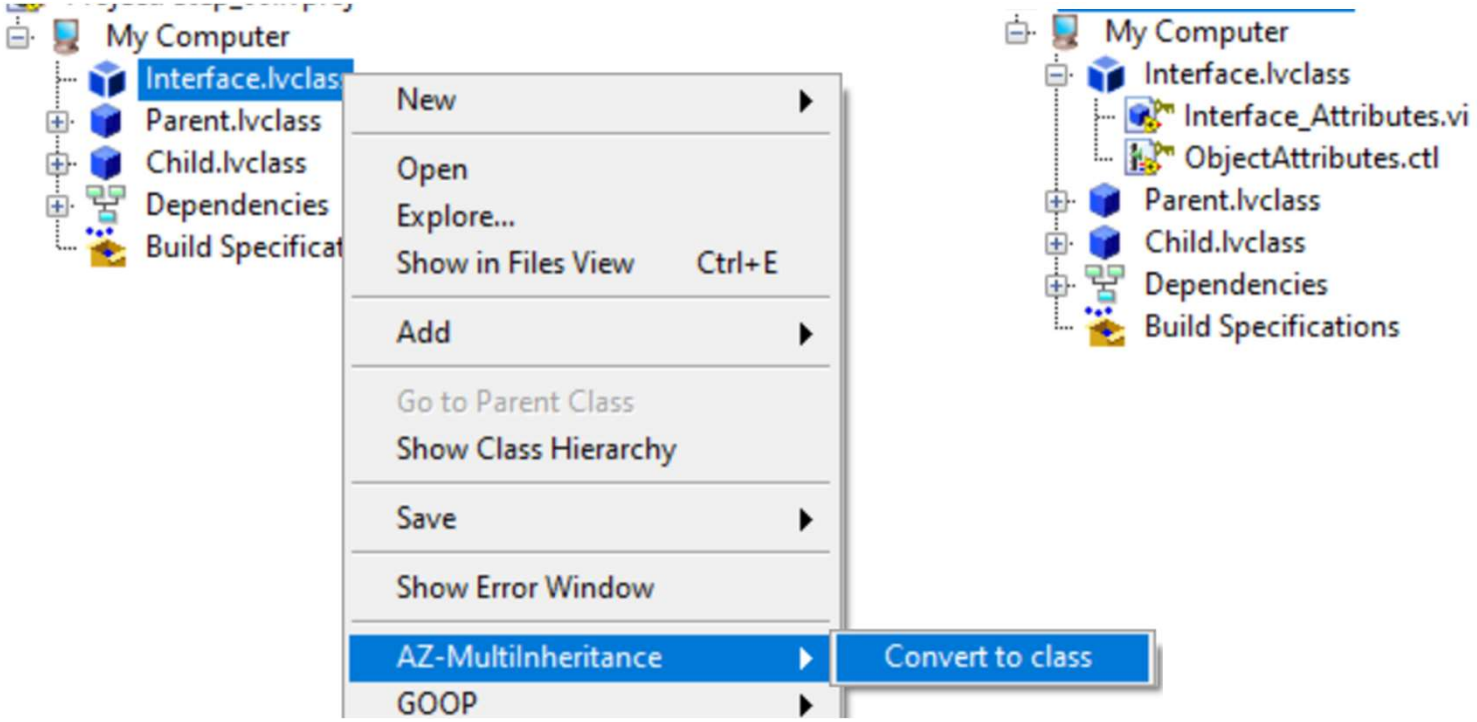
Same approach in GOOP4 and i-Classes



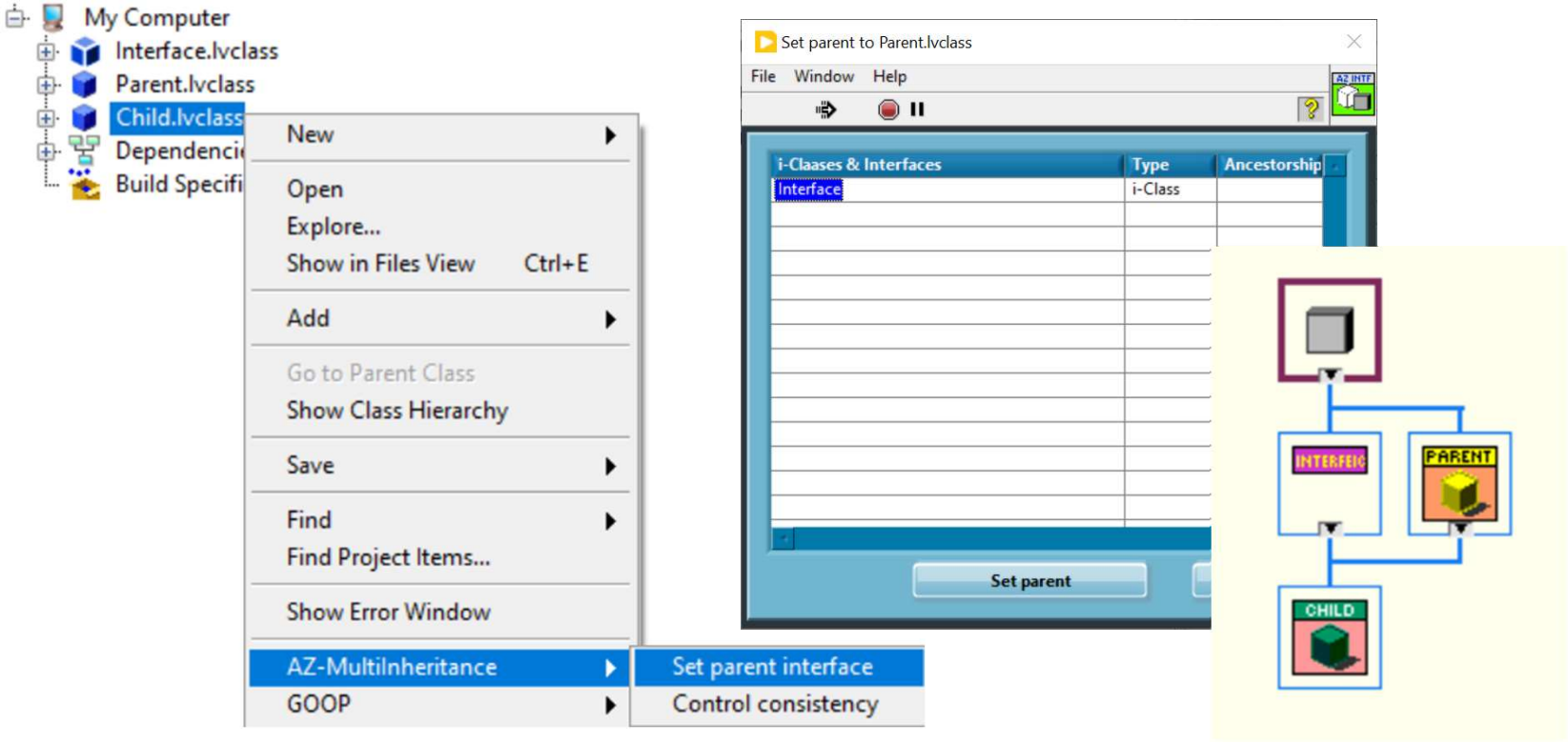
- Race conditions are prevented with In Place Element Structure.
- Same approach for accessing attributes.



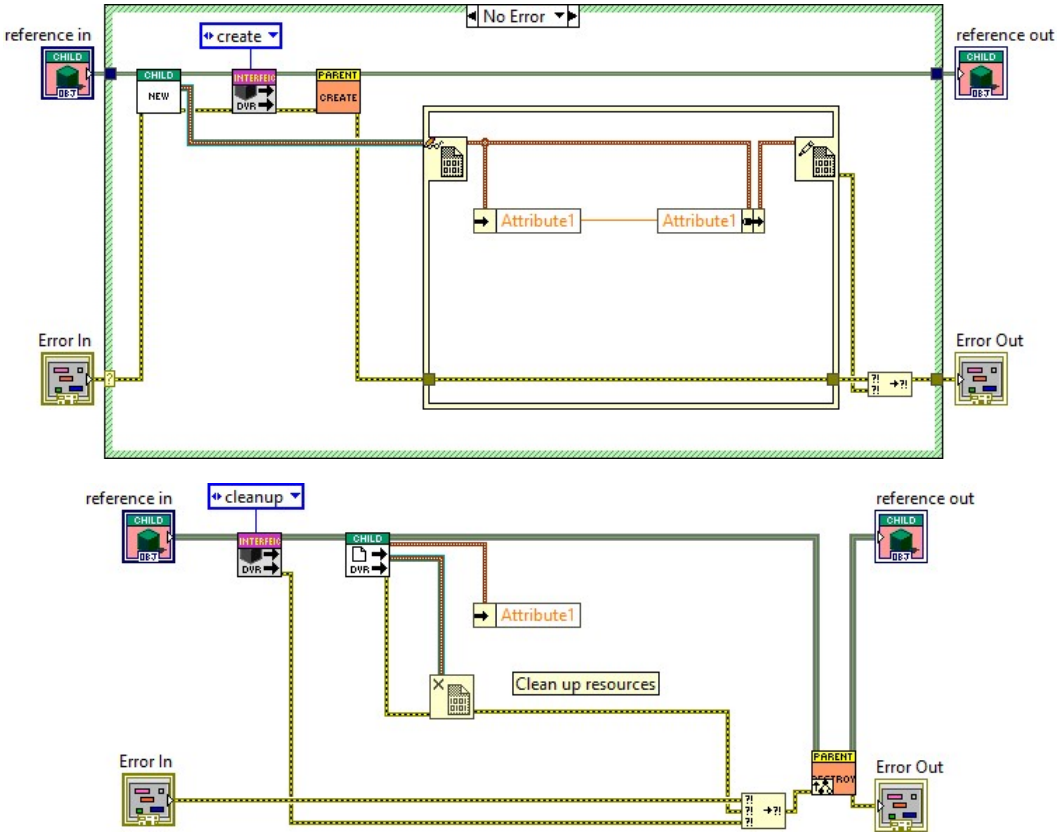
Toolkit


My Computer
Interface.lvclas
Parent.lvclass
Child.lvclass
Dependencies
Build Specificat
New
Open
Explore...
Show in Files View Ctrl+E
Add
Go to Parent Class
Show Class Hierarchy
Save
Show Error Window
AZ-MultiInheritance
GOOP
Convert to class
My Computer
Interface.lvclass
Interface_Attributes.vi
ObjectAttributes.ctl
Parent.lvclass
Child.lvclass
Dependencies
Build Specifications

Add i-Class as parent



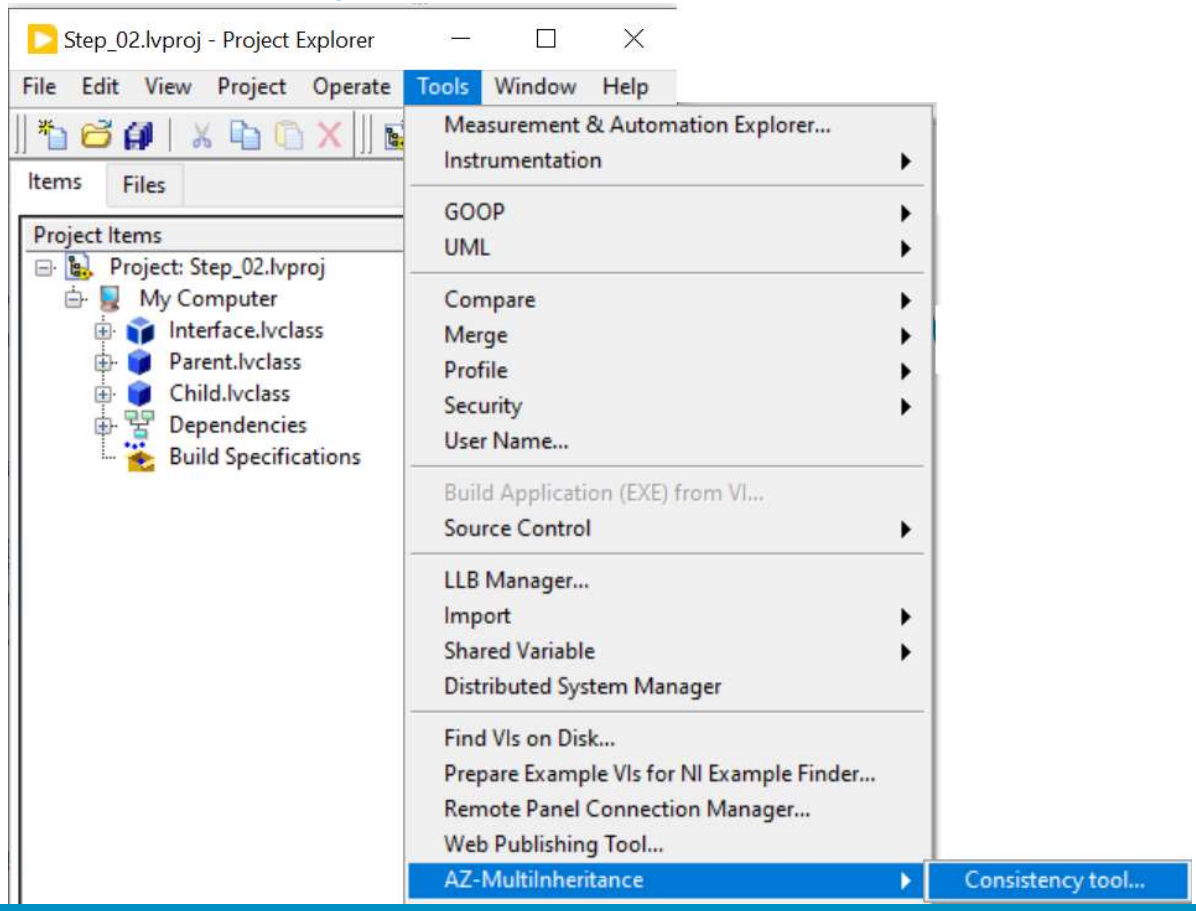
Inheriting from i-Class: constructor and destructor



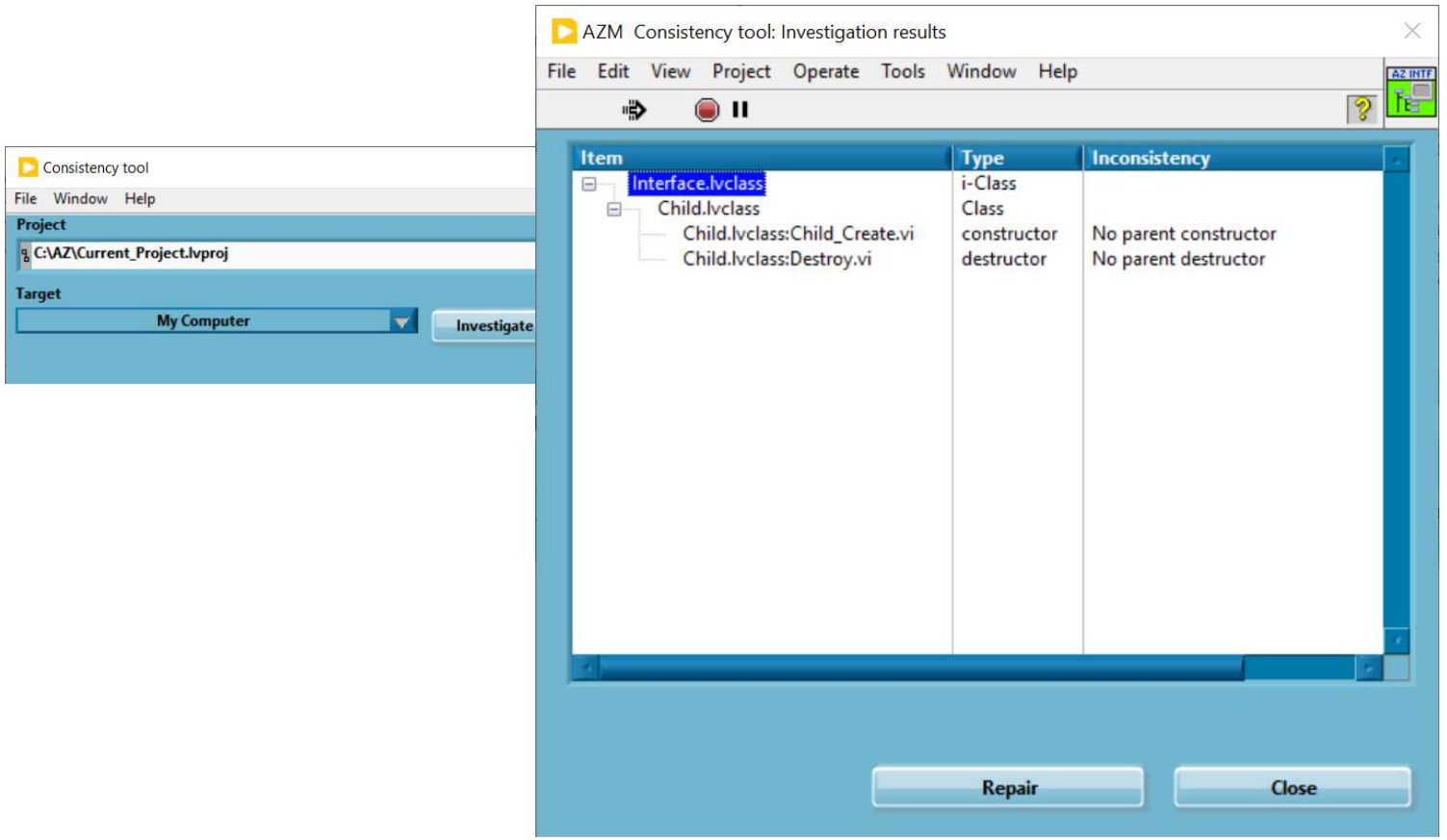
Important:

- i-Class accessor is added in constructor and destructor of the class
  - Corresponding element of the i-Class map has the same lifetime as object of the GOOP class

What to do? Interface was a parent before conversion to i-Class



Consistency tool



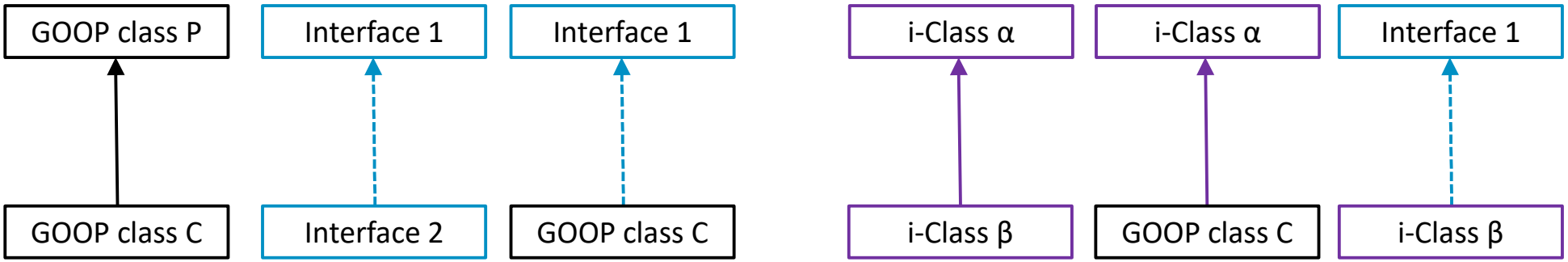
Limitations

- Conventional class can inherit from “main” parent (GOOP-class) and secondary parents (i-Classes).
  - There is no way to convert main to secondary and vice versa.
  - GOOP-class can inherit from i-Class, but i-Class cannot inherit from GOOP-class.
- i-Class is always abstract.
  - It cannot be instantiated.
- i-Classes can be used only with GOOP4
  - Probably they can be used with GOOP3 and G#.
  - They can never be used with Native LabVIEW classes.
- Non-canonical OOP behavior.

Canonical OOP behavior

Subclass is superclass with added or overridden members:

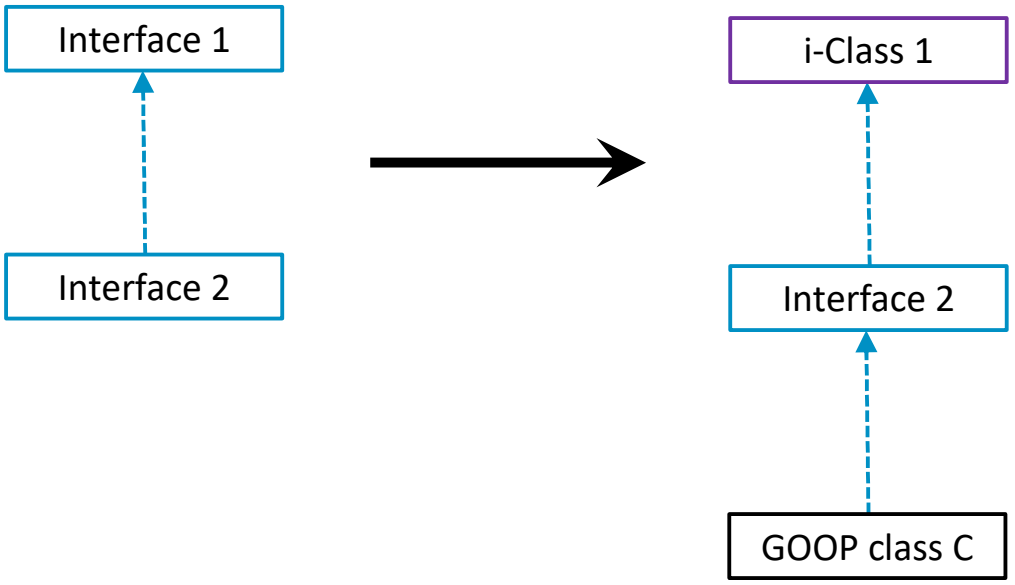
- methods,
- attributes.



Single limitation: Interface cannot inherit from a class

Non-canonical OOP behavior

But for LabVIEW… i-Class is still an Interface!

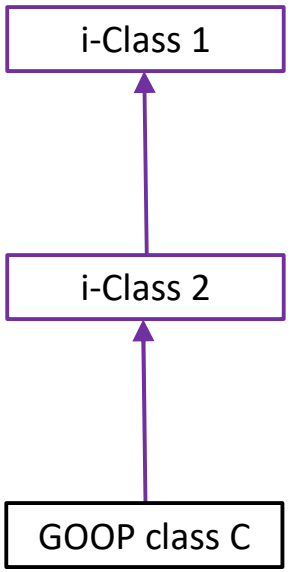


- It is absolutely legal from LabVIEW point of view!

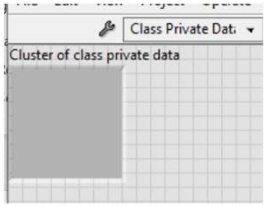
Non-canonical OOP behavior

We solve the problem if we can agree on:

Interface inheriting from i-Class is i-Class



- We have an excuse



- It is native LabVIEW class, but... there is an analogy

AZInterface.net

andrei.zagorodni@novatorsolutions.se

AZ Interface Toolkit

AZ interface is a solution for implementing Java-like interface architecture in LabVIEW projects.

Contrary to other solutions providing Java-like interface architecture, **AZ interface** is simple while fulfilling basic programming demands.

Downloads and Notes

Version 2.2.0-beta/alpha - last version

2019-04-22

- AZ interface v.2.2.0.0-beta/alpha (4.2MB ZIP)
- Only manual (0.6MB PDF)
- Improvements

Questions for brainstorming

- What is forgotten?
- What is wrong?
- What is ok but can be improved?
- Are there better ideas or solutions?

NOVATOR SOLUTIONS

Andrei Zagorodni

Stockholm

Thank you!