Tips and Rules to Control your Application Development and Performances

Jean-Philippe Bacher

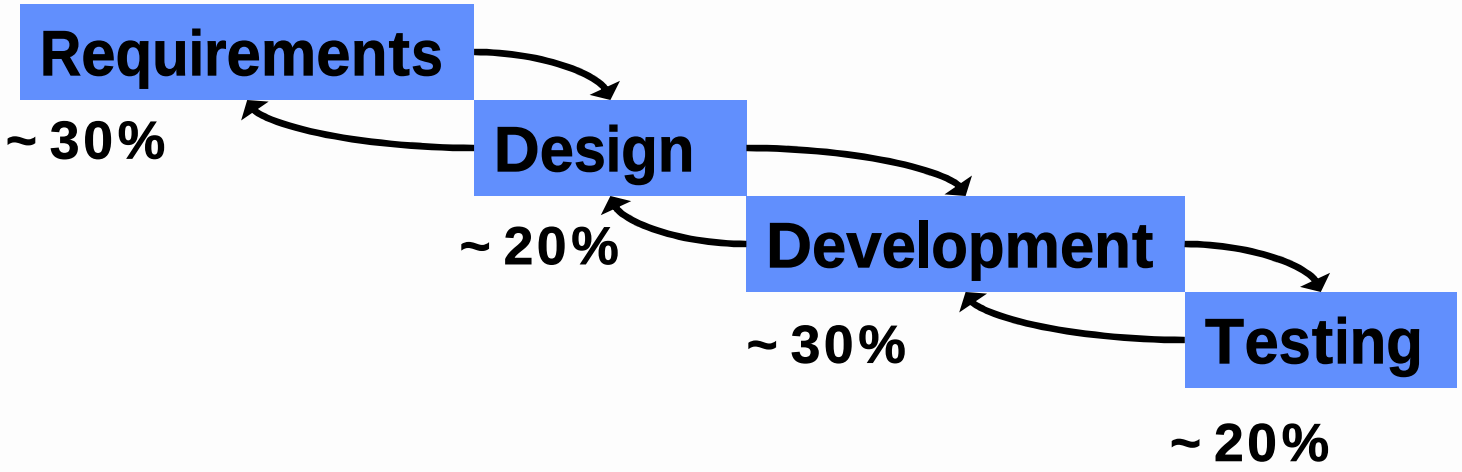
qualimatest sa *(Head Office)*
Chemin des Aulx 18
1228 Geneva – Switzerland
Tel. +41-22 884 00 30 • Fax +41-22 884 00 40

(Branch Office)
Brunnmattstrasse 9
3174 Thörishaus – Switzerland
Tel. +41-31 888 88 00 • Fax +41-31 888 88 01

Qualimatest SA in 4 lines

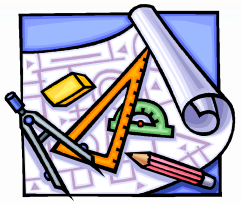
- Integrator in Vision & Automation
- Using LabVIEW since 1991
- National Instruments Select Alliance Partner
- Markets: medical devices, watch industry, automotive, …

Reminder: Software development process



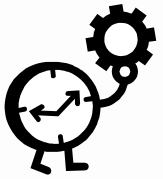
Is everything under control ?

- Is SW architecture and data management clear for me, did I document it ?

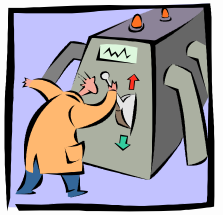


- Today, do I know which specifications are fulfilled and which are not ?

- Do I have clearly identified critical functions from the performances point of view ?



- Do I have any idea about the SW test protocol ?

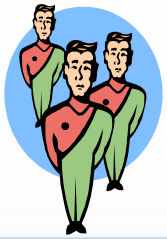


- What if somebody else should continue the development ?

How to increase control ?

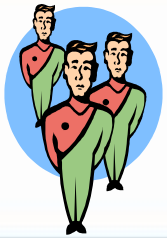
- Specifications management (see next talk)
- Early definition of test protocol
- Standardize, Re-use
- Follow development rules
- Use modular architecture
- Identify critical tasks
- Module test and code review

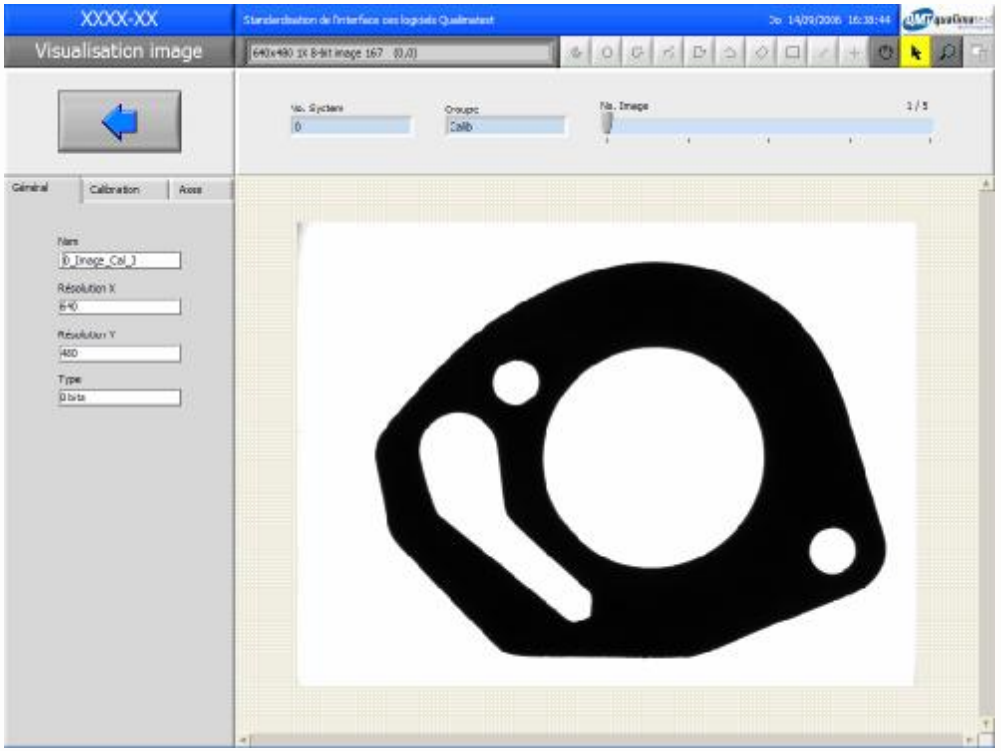
Standardization, where should I start ?



- Controls & Indicators
- UI interface (menu)
- Configuration
- Modules related to often used HW
- Your standard data processing
- Your standard reports or data logging
- Things that you are coding for the 3rd time

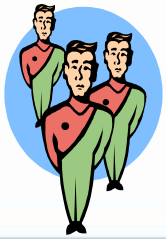
Standardization, example



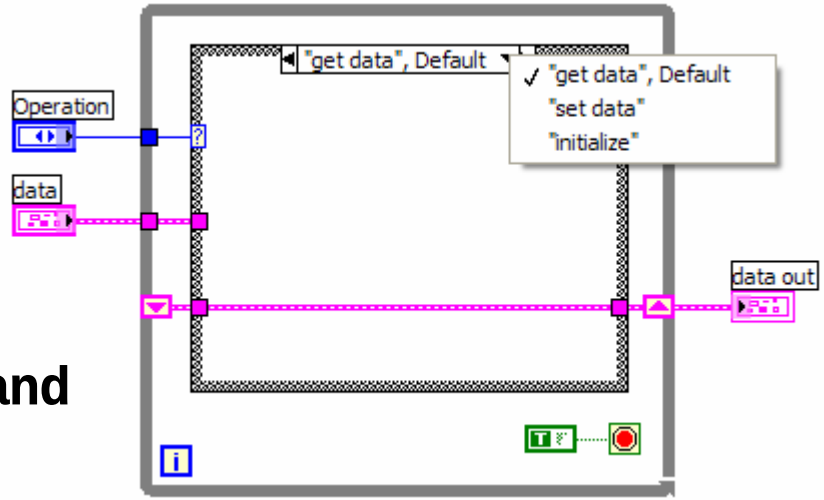


- GUI
- Image acquisition configuration
- Calibration
- Image processing configuration
- Standard production mode

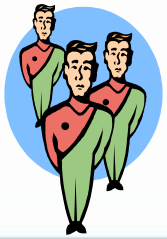
Examples and templates

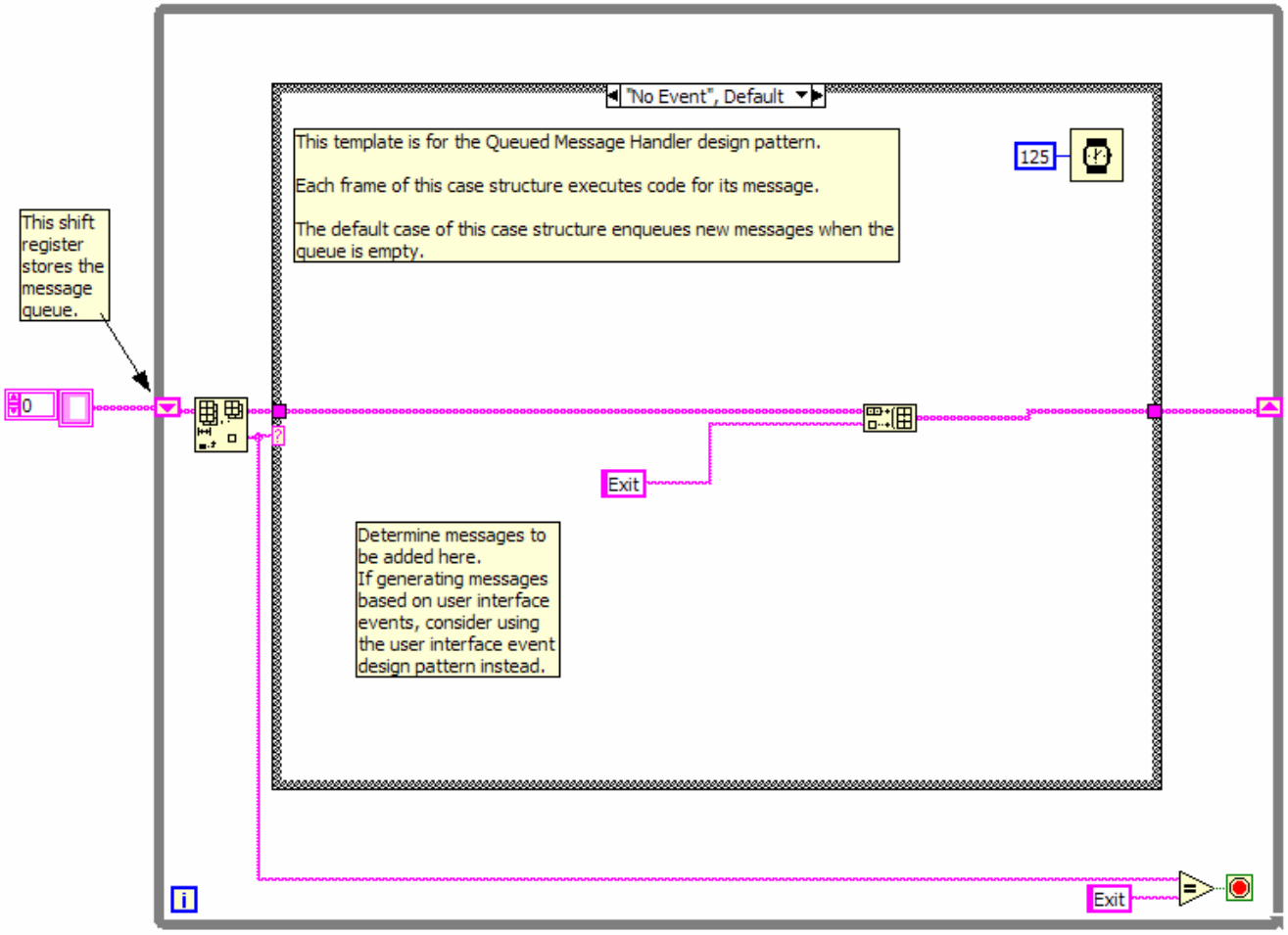


- Often a good start
- A good way to increase your knowledge
- Among them, some popular ones :
  - Functional Global
  - State Machine
  - Queued Message Handler
  - Master-Slave Design Pattern
- Inspiration for your own templates and standards

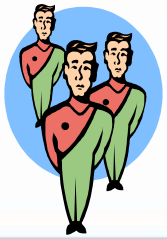


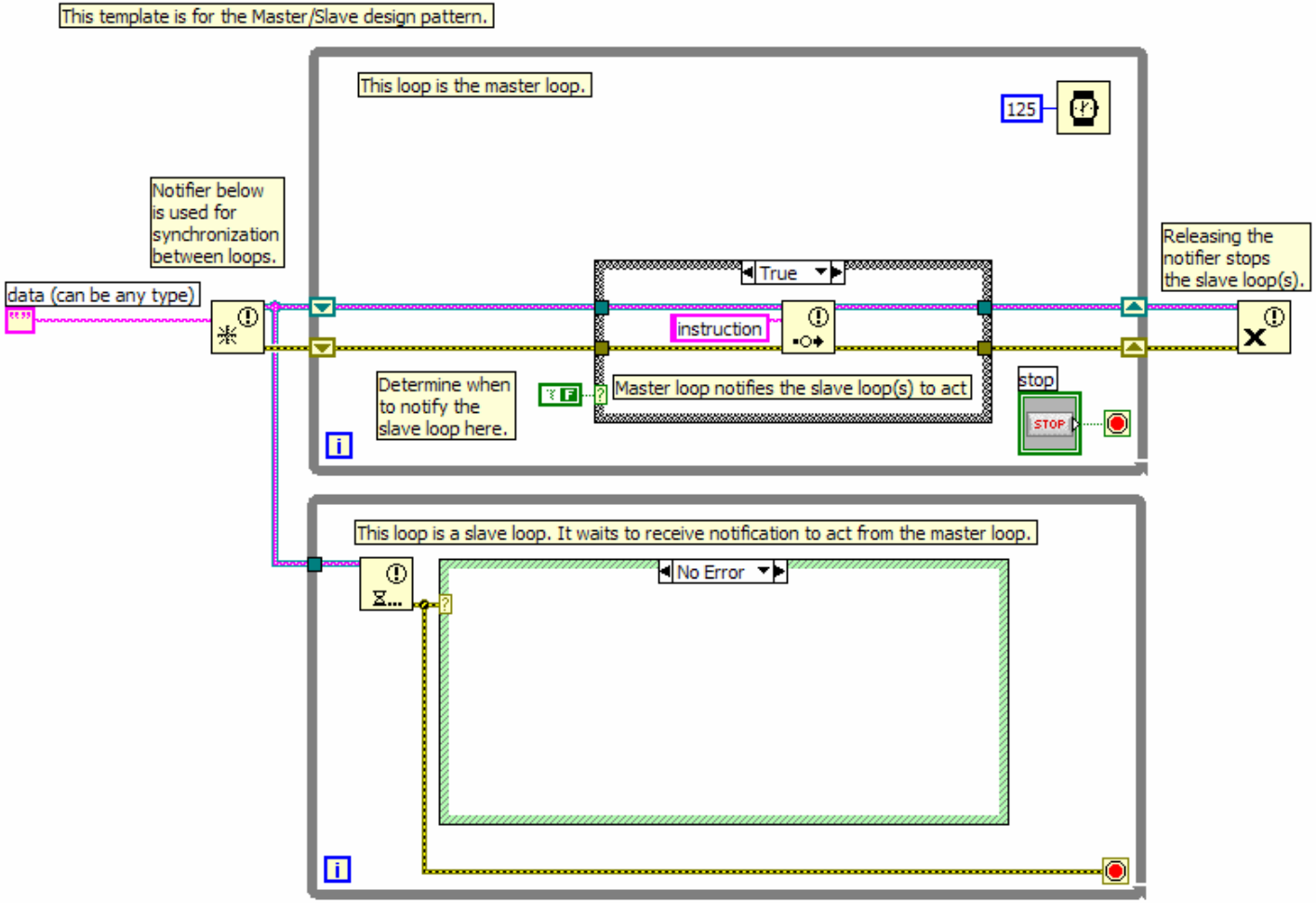
Examples and templates



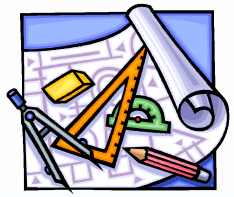


Examples and templates



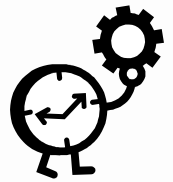


Modular architecture



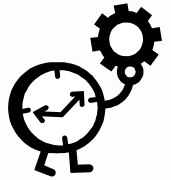
- 1 module per HW type
- 1 module per group of functions
- 1 data set per module (data is where it is needed)
- Separation between critical and non-critical tasks
- Benefits:
  - Increased Reusability
  - Increased Testability
  - Increased Maintainability

Critical tasks identification

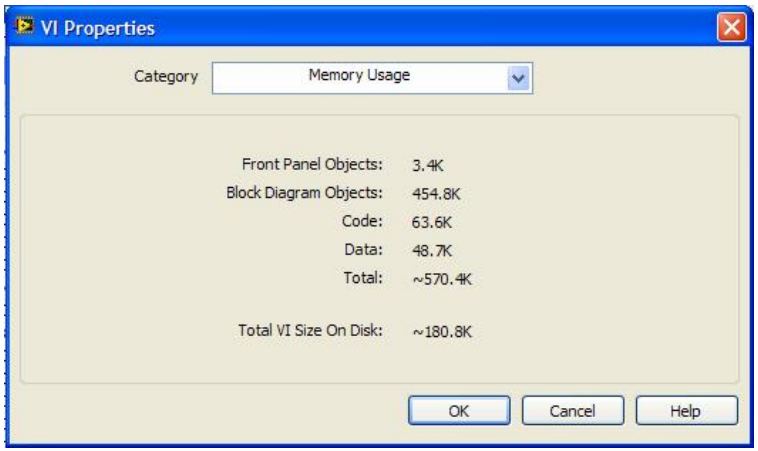


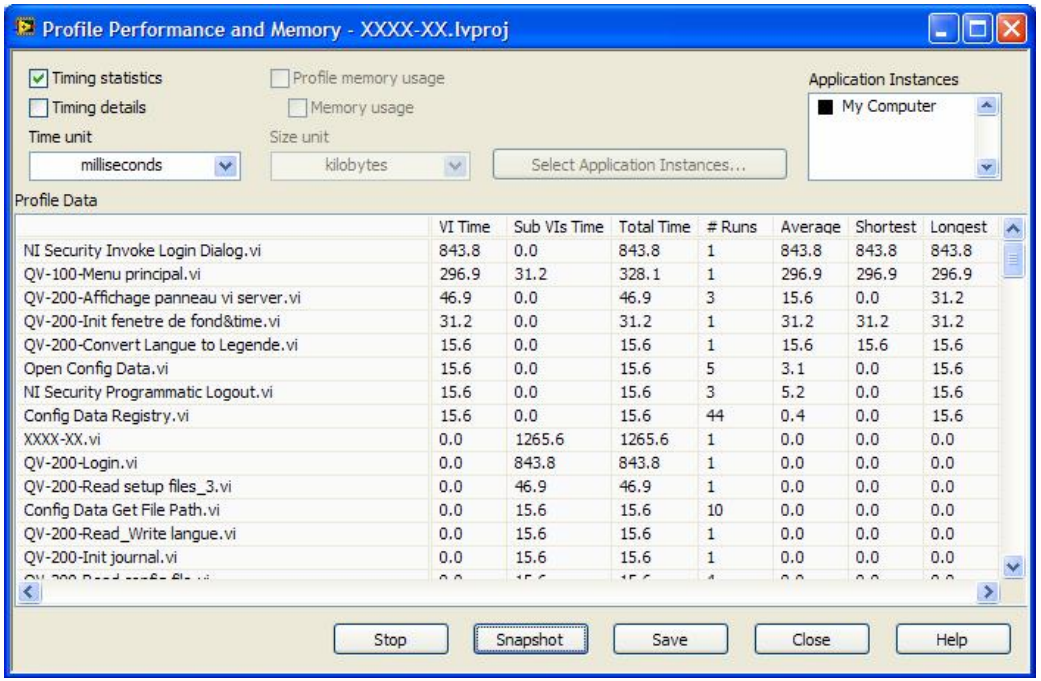
- Acquisition tasks
- Parallel tasks
- Triggered tasks
- Synchronizations
- Big-size data storage
- Big-size data manipulation
- Complex signal processing
- Communication

Performance measurement tools

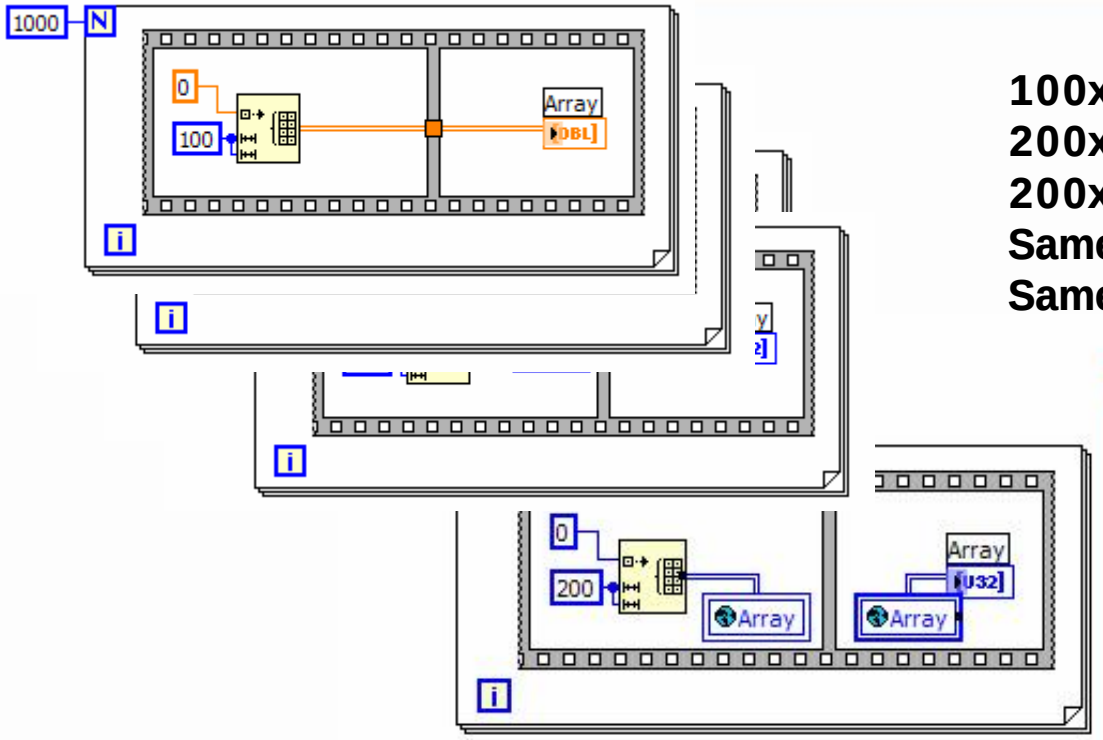


- VI Properties
- Profile window
- Show Buffer Allocation
- Custom tool

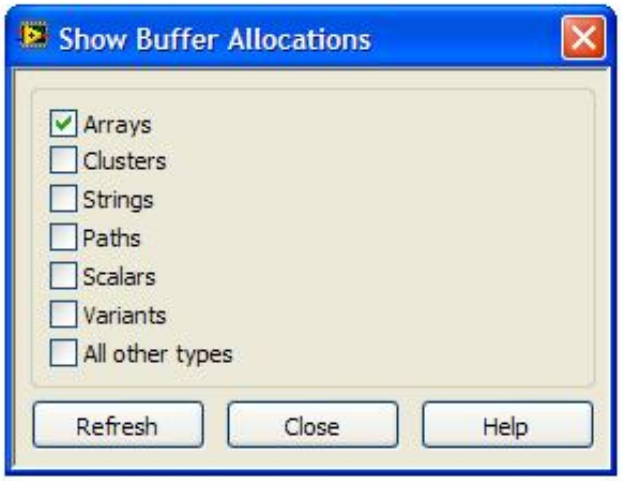




Example

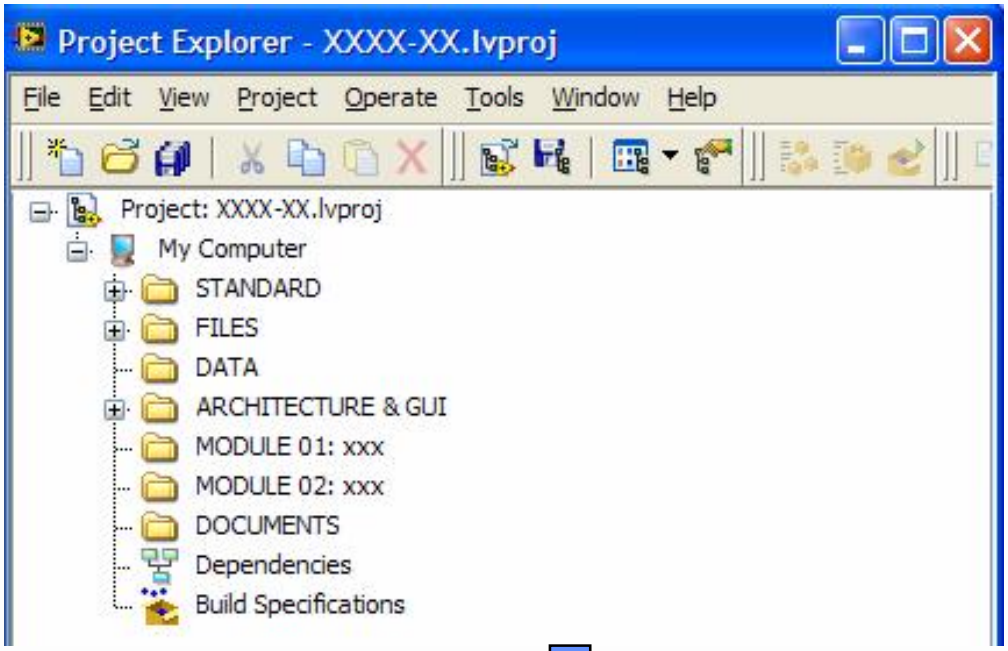


100x100 DBL -> 15ms 250kB
200x200 DBL -> 1500ms 1300kB
200x200 U32 -> 120ms 640kB
Same via LOC -> 550ms 800kB
Same via GLO -> 640ms 800kB



See "LabVIEW Performance and Memory Management", AN-168

Project manager



Gives you the overview,
helps you structure your work.

10 tips summary

1. Define and manage requirements
2. Take your time on design, architecture & data structure
3. Think modular
4. Define simulation modes
5. Use project explorer to structure your work
6. Use your standard
7. Use templates & examples
8. Use STD, FGLO & QMH
9. Do code review
10. Add to Standard

Tutorials and application notes

- **LabVIEW Performance and Memory Management, AN-168**
- **Best Practices for Developing Large Applications using a Structured Development Approach, Tutorial**
- **Optimizing VI Performance, Tutorial**
- **Using LabVIEW to Create Multithreaded Applications for Maximum Performance and Reliability, AN-114**
- **LabVIEW Unit Validation Test Procedure, AN-137**

Webcasts

- Best Practices for Managing Application Development with the LabVIEW Project
- How to Perform Validation on a LabVIEW Application
- Managing Requirements and Developing Large LabVIEW Applications
- Optimize Your VI Performance (Part I & II)
- Performing Technical Code Reviews to Improve LabVIEW Code Quality
- Software Design Architectures in NI LabVIEW
- Using NI LabVIEW for Object-Oriented Programming
- Using the Event Structure for More Than Just the User Interface

Contact us for your projects…

- On the web
  - Web Site: http://www.qmt.ch
  - E mail : info@qmt.ch
- Geneva, headquarters (Plan-les-Ouates)
  - Tél. : + 41 22 - 884 00 30
  - Fax : + 41 22 - 884 00 40
- Thörishaus (Bern area)
  - Tél. : + 41 31 - 888 88 00
  - Fax : + 41 31 - 888 88 01