Go4 v2.8 Analysis Design

J. Adamczewski, M. Al-Turany, D. Bertini, H.G.Essel, S.Linev

CHEP 2004

27.9.04 H.G.Essel: Go4 - http://go4.gsi.de
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Go4 tasks: GUI & analysis

Analysis task

User event loop
- Events
- Histograms
- Graphs

GUI task

Go4 GUI (Qt)
- control
- browsers
- editors
- graphics
- User GUI

Event IO:
- DAQ
- Server
- Files
- User

Commands
- Objects

ROOT files
The Go4 framework

- **Framework** for many kinds of experiments (Atomic & Nuclear Physics)
- The analysis is written by the user (**unlimited ROOT**)
- **Services and interfaces** for analysis
- **Batch mode** (CINT or compiled, on/off-line)
- **Interactive mode** (on/off-line):
  - A non blocking GUI controls and steers the analysis
  - Analysis runs independently and can update graphics asynchronously
  - ROOT object transport between analysis and GUI task
  - Qt based GUI interfaces **ROOT and Qt graphics**
  - **User defined GUI** supported (Qt designer)
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Analysis steps (objects)

TUserAnalysis

Step

Input

Process

Output

Factory1

Init.

TGo4Analysis

Step

Input

Process

Output

Factory2

Init.

TObjArray

Step

Input

Process

Output

FactoryN

Init.

Steps definition and control

Framework

User Code

User Code

GSI
Analysis steps (object IO)

Diagram showing the flow of data through processes. Each process has an input that is equal to the output. The diagram includes intermediate IO and coded processes.
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Analysis control

• Configuration of the steps via GUI or macro (initialization, analysis stand by)
  • Enable/disable
  • Set I/O

• Macros
  • Executed directly in analysis code (full access to framework)
  • Launched from GUI, executed in analysis synchronized with event loop (running event loop)

• User parameter objects
  • Generic editor for user objects
  • Update function executed in analysis can be used for anything (running event loop)

• Conditions (running event loop)
  • Editor can configure condition to return always TRUE or FALSE
  • Checked in analysis code (counters)

• Interactive histogramming (running event loop)
  • Create histograms and conditions on the fly
  • Fill with any members of events (event by event or from memory tree)
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Go4 analysis organization

So far

- Designed for linear flow of analysis, generation oriented
- Abstract Interfaces for IO, data structures, processing
- User defined factories for setting up the steps
- Fully controlled by framework
New requirements

So far

• Designed for linear flow of analysis, generation oriented
• Abstract Interfaces for IO, data structures, processing
• User defined factories for setting up the steps
• Fully controlled by framework

New requirements

• Event stacks
• Hierarchy of steps
• Concurrent steps
• Control of multiple I/O per step
• Logical mesh setup control
Event stacks

- Asynchronous DAQ branches (events are subevents with time stamps)
  - event building in analysis
- Decay measurements
- Mixing of events from different sources (simulation)

Event by event

Event stack
Hierarchy: like ROOT TTask mechanism

Enable/disable branches
Fixed execution order
Concurrent analysis steps

daq → raw event

unpack → built event → det 1 → calibrated event

Intermediary IO

analysis → physics event
Concurrent analysis steps

- **daq**: raw event
- **unpack**: raw event
- **built event**: det1, det2, det3
- **det 1**:
- **det 2**:
- **det 3**:
- **calibrated event**: det1, det2, det3
- **analysis**: physics event

Intermediate IO
Multiple IO

step 1

process

input event

output event

who instantiates the input events from file?

step 2

input event

output event

Files

Trees

Branches

event event
event event
event event
event event

event event event

event event event

event event event
Multiple IO and data generations

step 1

process

input event

output event

input event

output event

input event

output event

input event

output event

step 2

process

input event

output event

input event

output event

input event

output event

step 3

process

input event

output event

input event

output event
Analysis meshes, not hierarchy

Input: Multiple files, trees, branches

Execution order of steps evaluated at run time!
Avoid back reference!
TTask not suitable!
Intermediate store/retrieve

Output: Multiple files, trees, branches
Analysis meshes, not hierarchy

Input: Multiple files, trees, branches

Execution order of steps evaluated at run time!
Avoid back reference!
TTask not suitable!
Intermediate store/retrieve

Output: Multiple files, trees, branches
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Solution I: Use special steps

Provider steps (input from file or execution step, no output)
Execution steps (input from provider steps, optional output)
Solution I: Go4 v2.8 (v3.0)

- **Provider steps** are used when an event may come from file or another step (can be configured by GUI or macros)
- **Execution steps** do not use their “own“ input events but the ones from provider steps (coded, but can proofed)

To be done:
- Provider steps may get objects from branches of same tree (file)
- Eventually the file open/close handling must be moved to framework

Not easy to do:
- Providing event stacks of different depth
- No hierarchy, but *execution order* of steps could be evaluated by framework

→ Logical analysis mesh with multiple inputs can be provided with little effort!
Content

- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- Solution I
- Solution II
Solution II: New components, design

- **New Go4EventManager**
  - Control of *multiple IO* for each step
  - Control of object exchange in *analysis mesh*
  - Event stacks
  - *Execution order* of Go4Tasks at run time (initialization)

- **New Go4Task**
  - Combines steps and TTask (optional *hierarchical structure*)
  - Register to Event Manager
  - Subscribe for input objects
  - Register output objects
  - *Factory interface* between application and framework
  - Extend analysis configuration (GUI and macros)
Event Manager

Go4Task

Go4Task

Go4Task

Go4Task

Event manager

Multiple files, trees, branches

Multiple files, trees, branches

object references

input event

output event
The end

J. Adamczewski, M. Al-Turany, D. Bertini, H.G. Essel, S. Linev

27.9.04

H.G. Essel: Go4 - http://go4.gsi.de