Analysis Organization: Go4 Analysis Steps and TTask

ROOT workshop February 2004
Contents

• The problem: organization of analysis
• ROOT TTask mechanism
• The Go4 analysis framework
• The Go4 analysis steps
• Upgrade of Go4 framework
The problem

- Modular analysis
- Modules must be set up and controlled (IO, processing, interactive, GUI)
- General purpose framework for different experiments
The problem: General analysis organization

```
<table>
<thead>
<tr>
<th>Components</th>
<th>Det1</th>
<th>Det2</th>
<th>Det3</th>
<th>DetN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsystem2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SubsystemN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stores</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generations:
- branch1
- branch2
- branchN

Operations:
- (de-) activate
- calibrate
- partial IO!
```

"Raw Event"
General analysis organization: use cases

• **Components** analysis may run as **independent branches** each branch may be switched off

• **Store and retrieve** events / data structures of **each generation/each component**

  *partially IO for components*

• **Set-up** (active components, range of generations, „calibration“ data) should be easy *config script, GUI*

=> **Framework with basic definitions required!**
ROOT TTask mechanism

TTask

- Exec()
- ExecuteTasks()

TMainTask

- Exec()

TTask

- Exec()

TList

TTask

- Exec()

TTask

- Exec()

TList

TTask

- Exec()

TList

TTask

- Exec()

TTask

- Exec()

TList

TTask

- Exec()
Logical mesh of TTasks

TMainTask

TTask2

TTask21

TTask211

TTask1

TTask3

TTask31

TTask4

TTask41

DetA 1
input
output

DetA 2
input
output

DetB 1
input
output

DetB 2
input
output

DetC 1
input
output

DetC 2
input
output

input
output

input
output

input
output

input
output

input
output

input
output

input
output

input
output

input
output

input
output

Generations

Components

Feb-2004

J.Adamczewski - Go4 -
http://go4.gsi.de
TTask control with TBrowser
ROOT TTask mechanism

- Lightweight and flexible framework: Inheritance of one class TTask
- independent branches and subbranches of tasks
- GUI control via TBrowser (branch view, activation, breakpoints)

- data interface between dependend tasks not given
  user resonsibility! subframework?

- IO interface of intermediate results not given
  user TTree, GUI controlled enable/disable? subframework?
The **Go4** (GSI Object Oriented Online Offline)

- **Framework** for many kinds of experiments (Atomic & Nuclear Physics)
- The analysis is written by the user (*unlimited ROOT*)
- **Services** (GSI DAQ, *analysis organization*, IO, ...) are provided
- **Batch mode** (CINT or compiled, off-line)
- **Interactive mode** (on-line or off-line):
  - A non blocking GUI controls and steers the analysis
  - Analysis may run permanently and can update graphics asynchronously
  - ROOT object transport between *analysis and GUI process*, multithreaded, sockets
  - GUI interfaces **ROOT** and Qt graphics
  - User defined GUI possible (Qt designer)
Go4 Processes: GUI & Analysis

**GUI**
- Preferences
  - QApplication
  - TApplication
- User GUI
- Go4 GUI
- Histogram servers:
  - Go4
  - LeA
  - MBS DAQ
  - GOOSY

**Analysis**
- User event loop
- QApplication
- TApplication
- socket threads
- Commands
  - Objects

**Event IO:**
- DAQ
- Server
- Files
- User

**Histogram server**

**Histogram clients:**
- Go4
- Origin
- LeA

**Auto-save file**

**ROOT files**

**GSI histogram API**

**Preferences**
Eventloop Actions

UserPreLoop: init objects

ProcessAnalysisSteps

Step1: get event
        fill event
        histograming

StepN: store event

UserEventFunc: histograming

ProcessDynamicList: dynamic histograming

UserPostLoop: reset objects

? other Analysis

HYDRA

TTree registry

online TTree::Draw()
Go4 GUI: Qt interface to ROOT

Qt Eventloop
QApplication::Exec()
Go4 Releases

- Go4 v1.x GSI internal release (May 2002)
- Go4 v2.0 public release (Nov 2002)
- Go4 v2.1 public release (Jan 2003)
- Go4 v2.2 public release (Mar 2003)
- Go4 v2.3 public release (May 2003)
- Go4 v2.4 public release (Aug 2003)
- Go4 v2.5 public release (Dec 2003)

Linux: Debian 3.0, Suse 8.1, Suse 8.2, RedHat 7.3, RedHat 9.0
Compiler: gcc 2.95, gcc 3.2, gcc 3.3

Users: GSI-FRS, SHIP, Euroball/RISING, HADES (online monitor), Atomic physics,...
Standard GUI
User GUI (Qt)  Parameter Setup for SHIP
User GUI (Qt) On-line monitoring of HADES

Courtesy HADES coll.

Feb-2004 J.Adamczewski - Go4 - http://go4.gsi.de
Go4 Analysis steps

TGo4Analysis

TUserAnalysis

Steps definition and control

Framework
User Code
GUI control of steps
Go4 Analysis steps

Chain of analysis steps processed **sequentially**
Each step can be **en/disabled** (framework)
Input/output can be switched (framework)
**Partial IO** (steered by application)

Each processor has access to all inputs!
Each processor has access to all parameters
Logical mesh of steps

Components

Generations

Feb-2004  J.Adamczewski - Go4 -
http://go4.gsi.de
Logical mesh of steps

Setup by framework
Setup by application (macros)

Generations

Components

DetA 1
input
output

DetB 1
input
output

DetC 1
input
output

DetA 2
input
output

DetB 2
input
output

DetC 2
input
output

final1
input
output

final2
input
output
Logical mesh of steps

Setup by framework
Setup by application (macros)

Generations

DetA 1 → output
DetB 1 → output
DetC 1 → output
DetA 2 → output
DetB 2 → output
DetC 2 → output
final1 → output
final2 → output
Go4 analysis organisation

- Designed for linear flow of analysis, generation oriented
- Abstract Interfaces for IO, data structures, processing
- User defined factory for each step
- Fully controlled by framework (GUI, macros)

- No hierarchy of substeps (no execution branches)
- Control of multiple inputs for one step not supported by framework
- Logical mesh setup possible, but not yet controlled by framework
Further Go4 developments

- Advanced framework interfaces:
  - Control of multiple IO for each step
  - Control of data flow in analysis mesh
- Analysis steps redesign for hierarchical structure (TTask subclass!)
- Extend standard analysis configuration GUI for above cases
Further Go4 framework developments

Stepmanager

StepA1
- ProcA1
  - inputABC
  - outputA1

StepB1
- ProcB1
  - inputB
  - outputB1

StepC1
- ProcC1
  - inputC
  - outputC1

Step2
- Proc2
  - inputA
  - output2
  - partial IO
  - multiple inputs

Ownership
Data flow
Generations

hierarchy of substeps
The End.