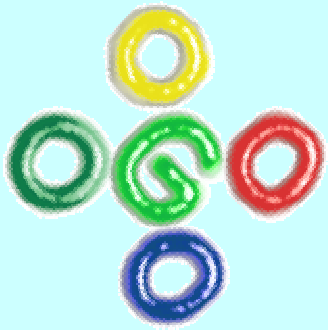


GO4 Version 3

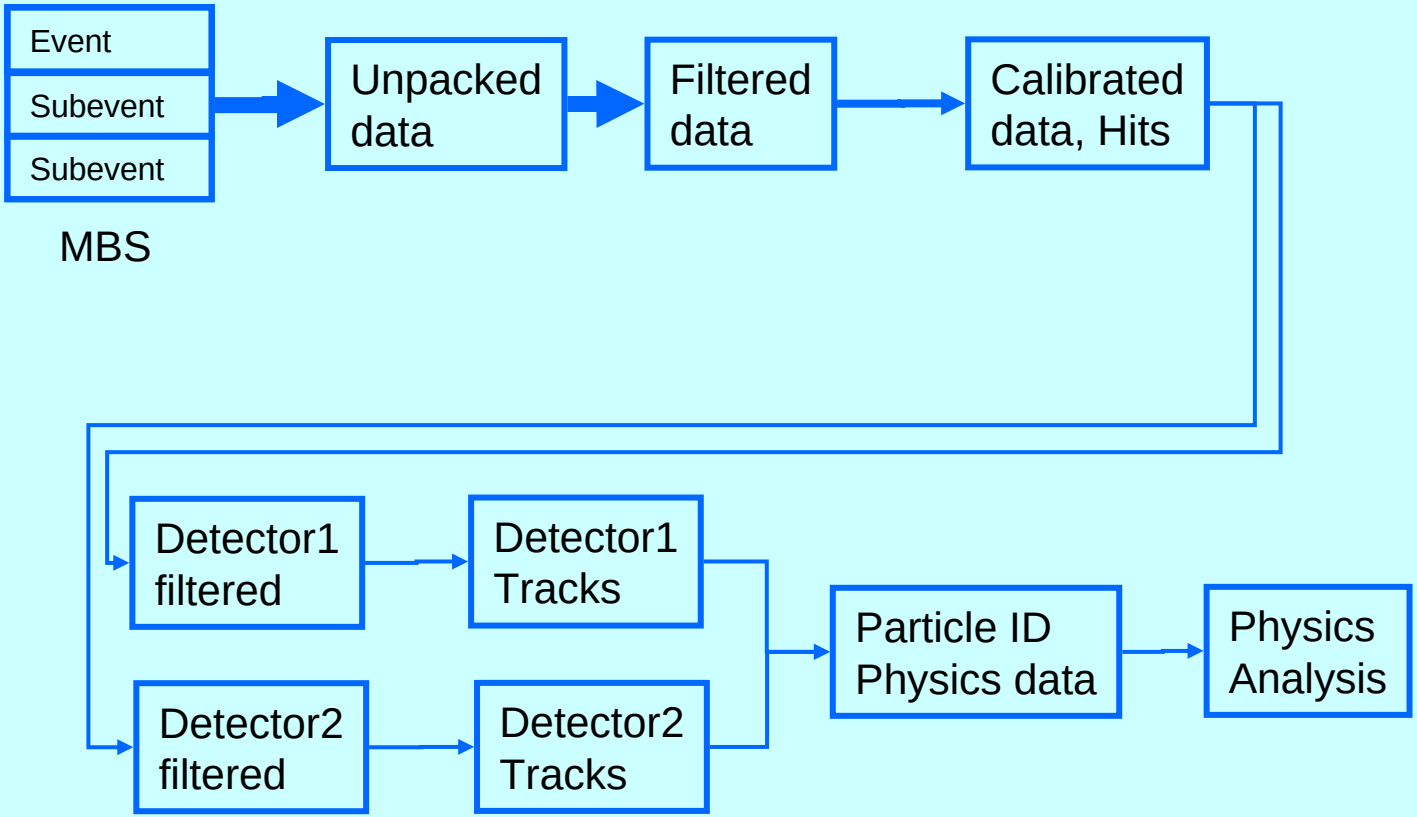
Analysis concepts

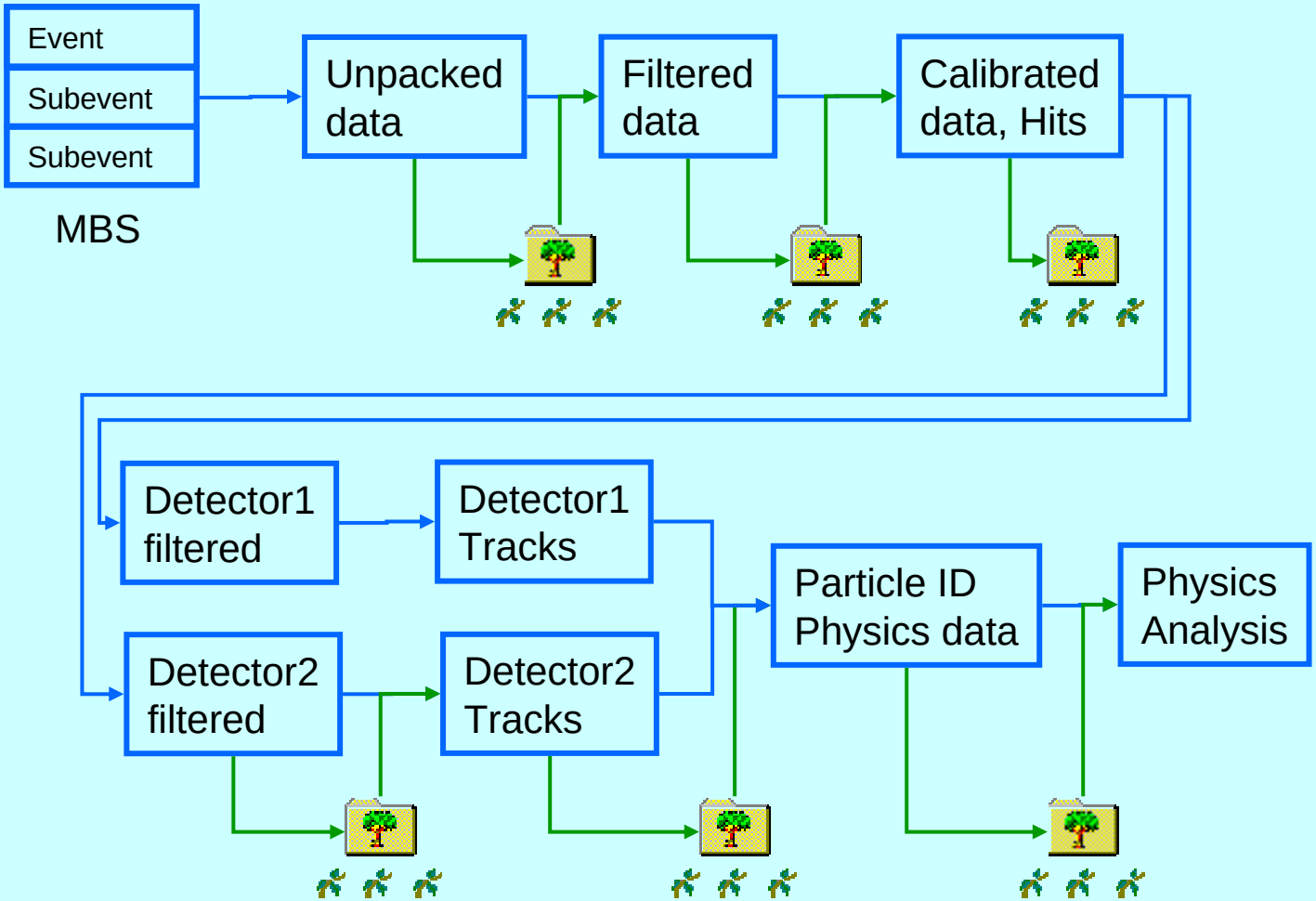
J.Adamczewski, H.G.Essel, S.Linev

Lectures

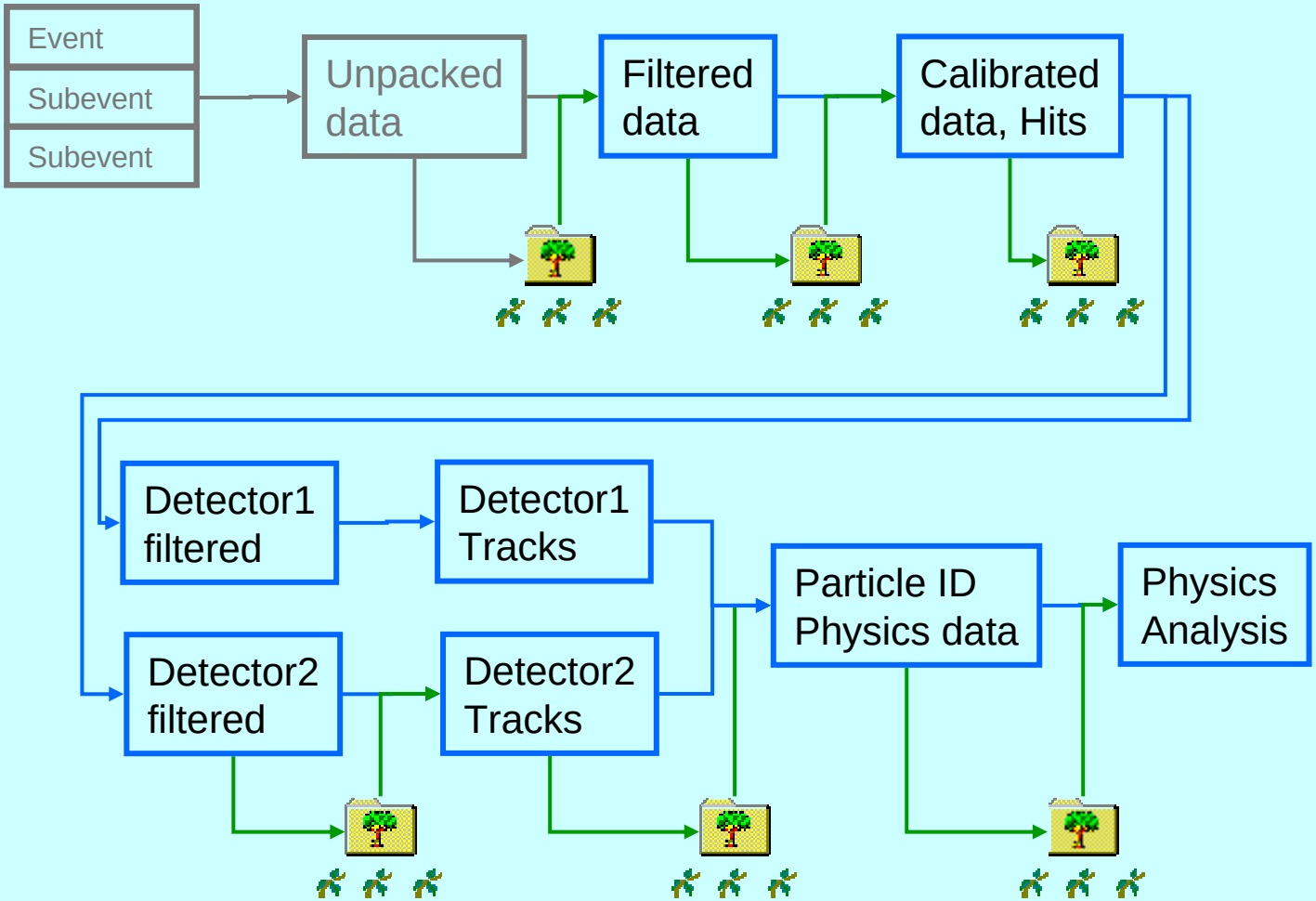
- Go4 is a framework for many kinds of experiments (Atomic & Nuclear Physics)
- The analysis is written by the user (C++, unlimited ROOT)
- Go4 provides services and interfaces for analysis
- It runs in batch mode (CINT or compiled, on/off-line)
- or interactive mode (on/off-line):
  - A non blocking GUI controls and steers the analysis
  - The analysis runs independently and can update graphics asynchronously
  - ROOT objects are transported between analysis and GUI task
  - ROOT and Qt graphics are interfaced
  - User may create specific GUIs (Qt designer)

Analysis of event data:
The event loop is executed by framework
User code is plugged in

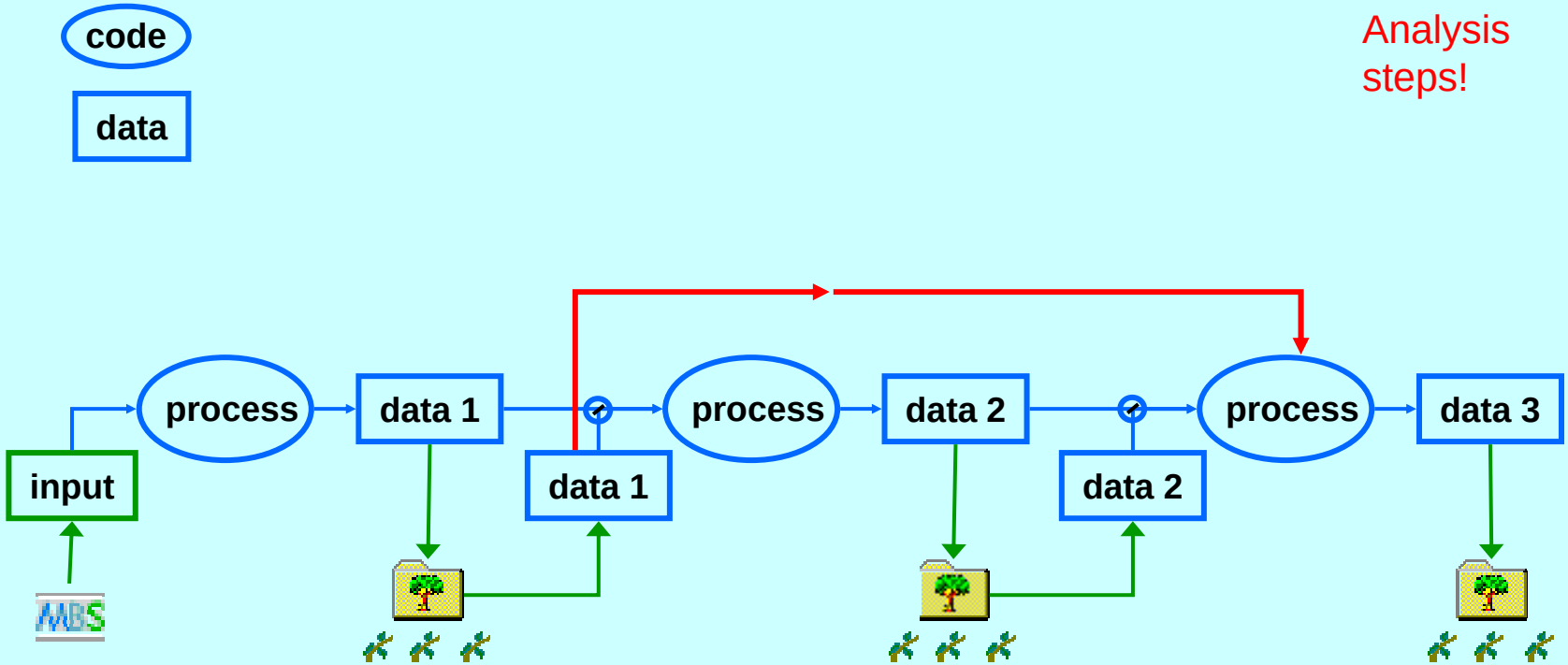

Event
Subevent
Subevent
MBS
Unpacked data
Filtered data
Calibrated data, Hits
Analysis steps!
Detector1 filtered
Detector1 Tracks
Detector2 filtered
Detector2 Tracks
Particle ID Physics data
Physics Analysis



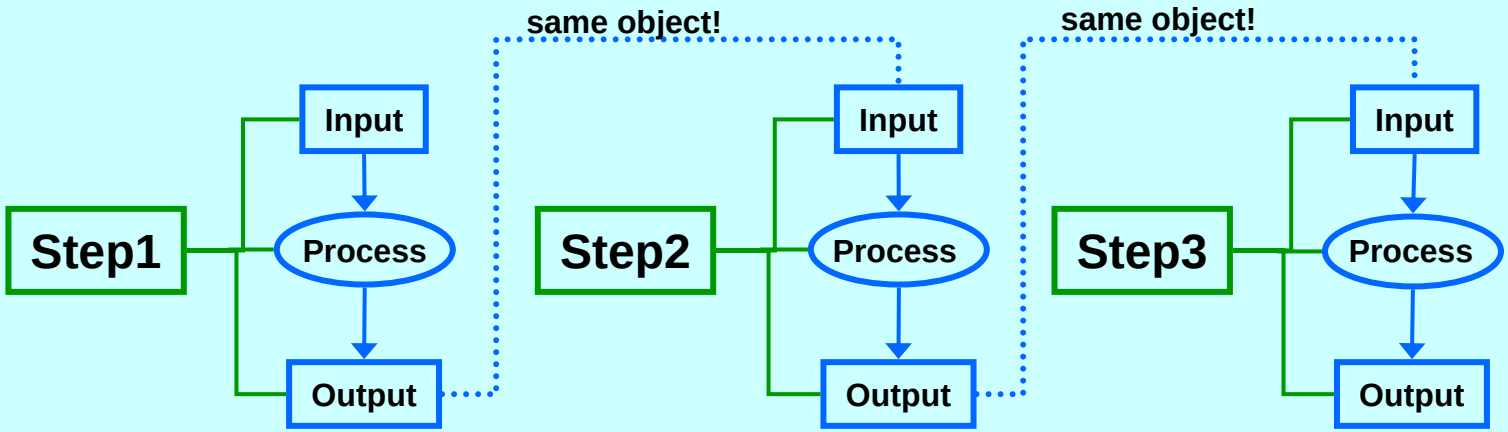
Analysis steps!

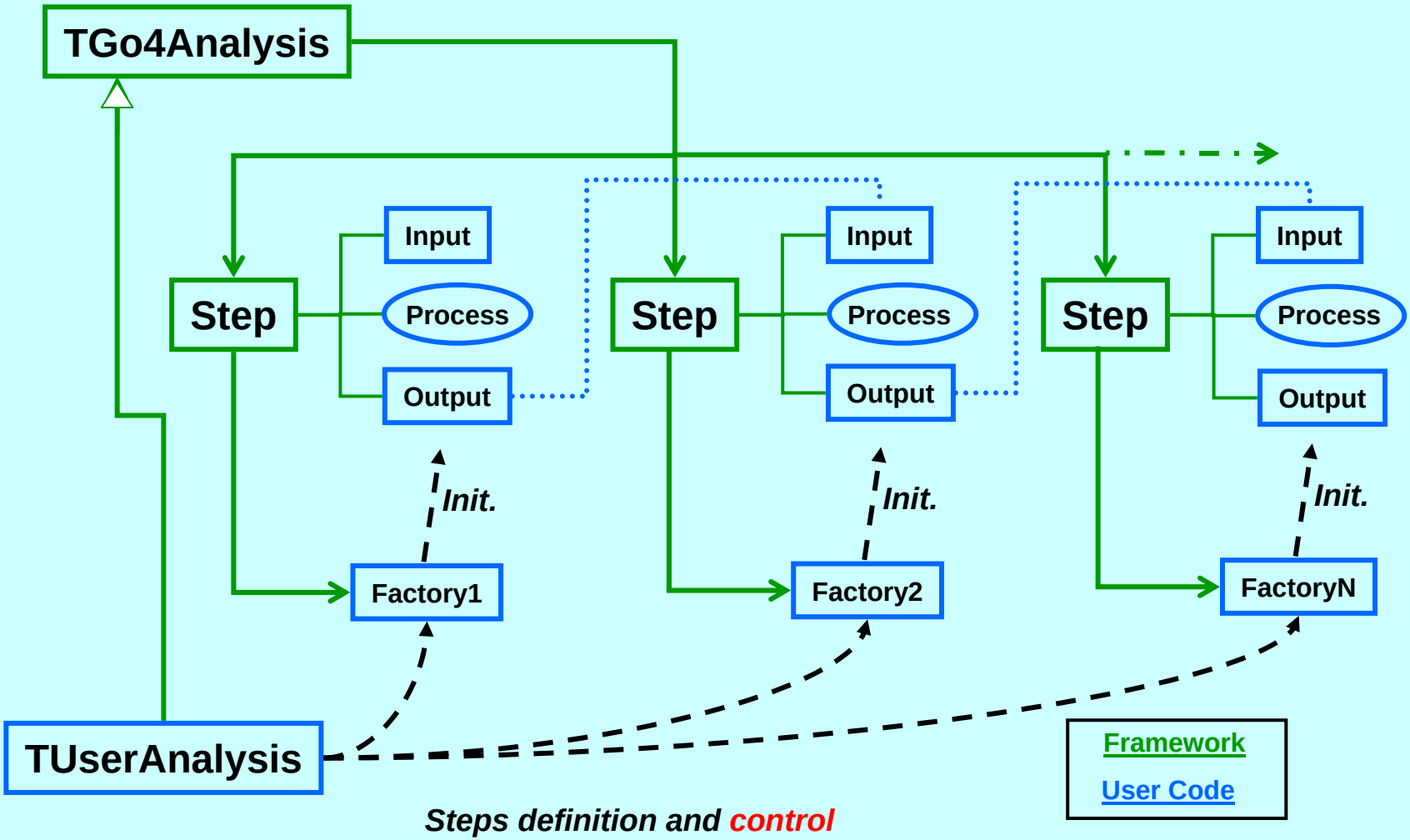


Analysis steps!

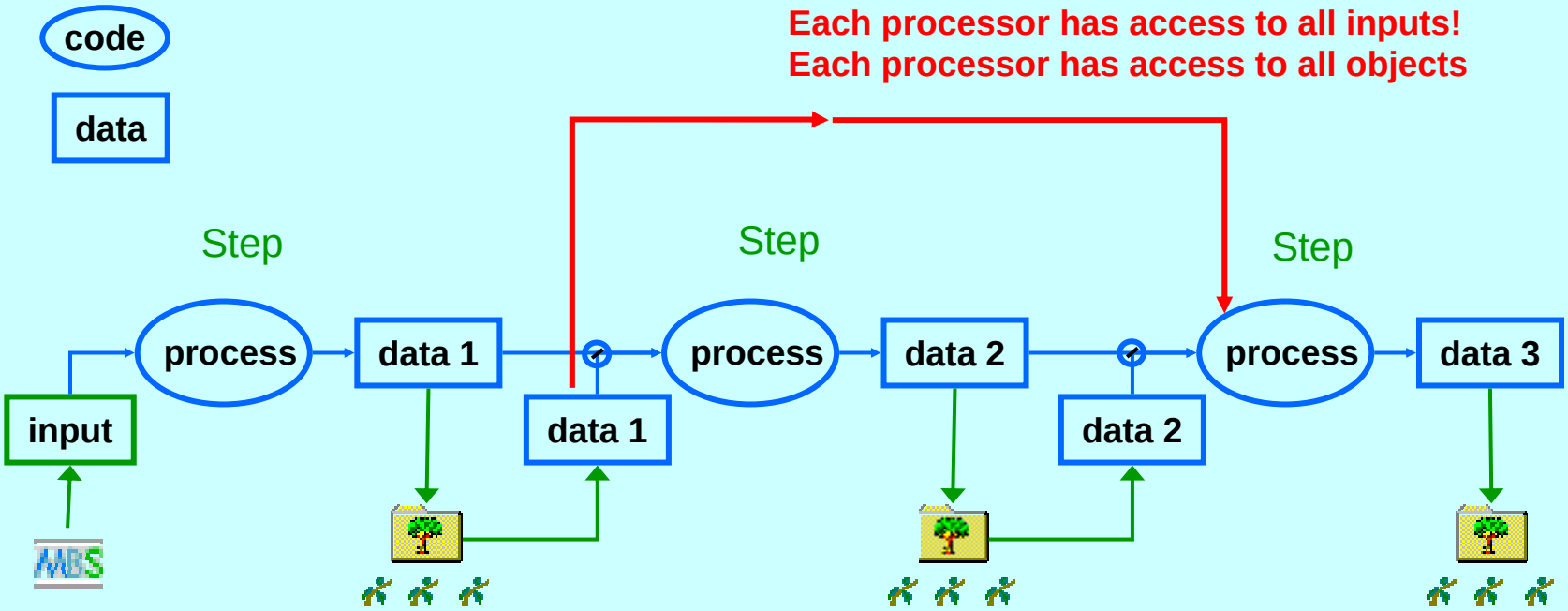

code
data
Analysis steps!
process
data 1
process
data 2
process
data 3
input
data 1
data 2
MBS

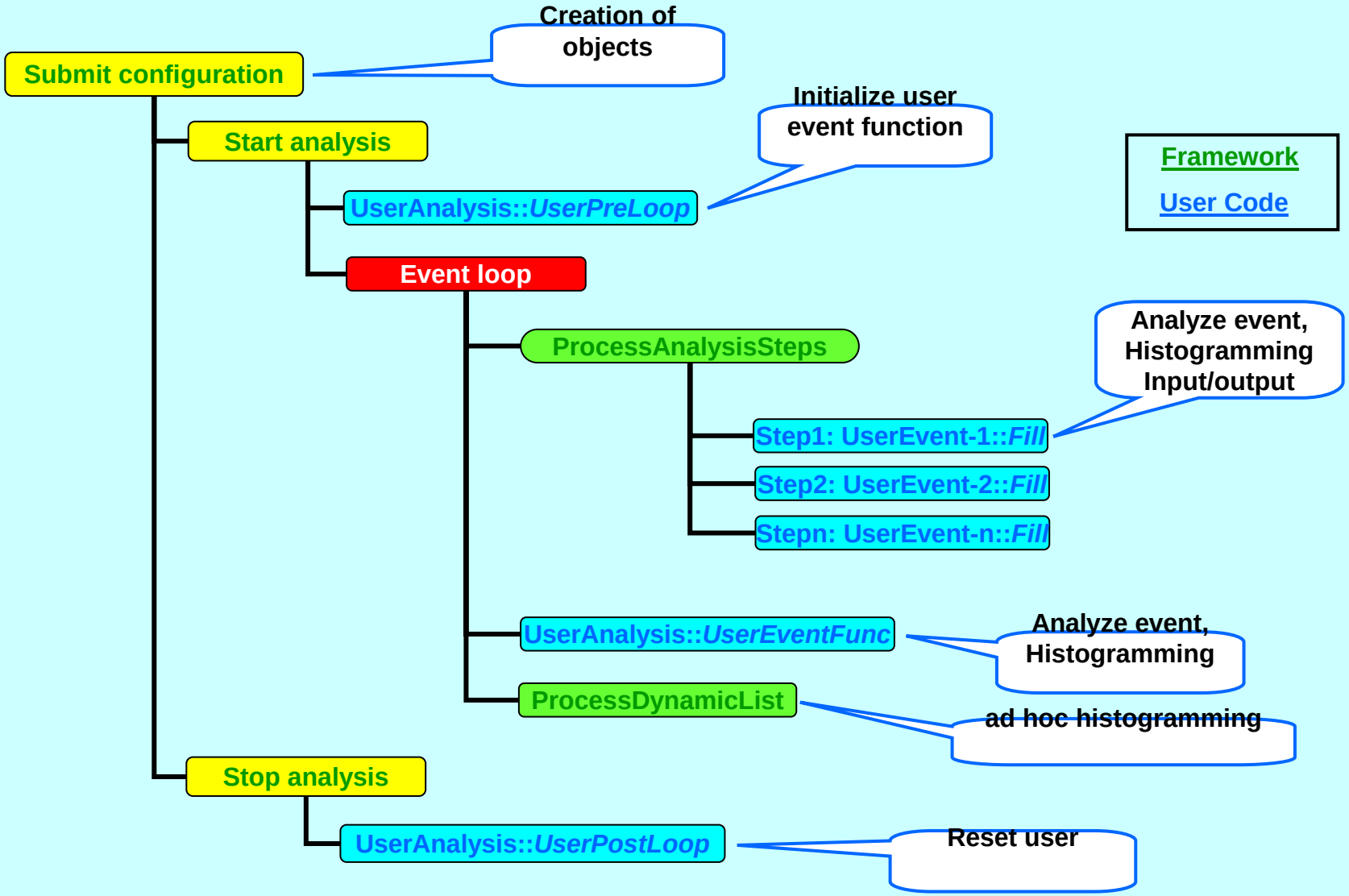
Analysis steps!

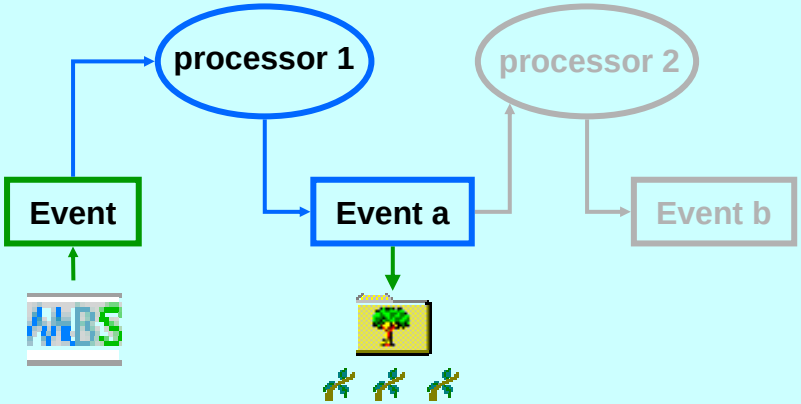



TGo4Analysis
Step
Input
Process
Output
Step
Input
Process
Output
Step
Input
Process
Output
Init.
Init.
Init.
Factory1
Factory2
FactoryN
TUserAnalysis
Framework
User Code
Steps definition and control

Chain of analysis steps processed sequentially
Each step can be en/disabled (framework)
Input/output can be switched (framework)



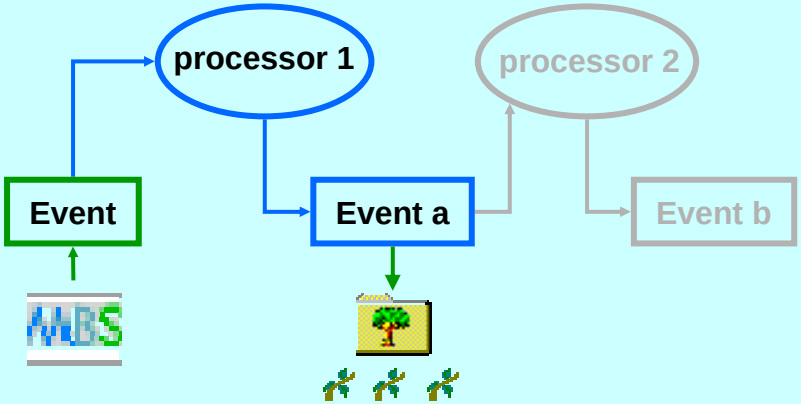

Creation of objects
Submit configuration
Initialize user event function
Start analysis
Framework
User Code
UserAnalysis::UserPreLoop
Event loop
ProcessAnalysisSteps
Analyze event, Histogramming Input/output
Step1: UserEvent-1::Fill
Step2: UserEvent-2::Fill
Stepn: UserEvent-n::Fill
UserAnalysis::UserEventFunc
Analyze event, Histogramming
ProcessDynamicList
ad hoc histogramming
Stop analysis
UserAnalysis::UserPostLoop
Reset user



Event::*Init* and *Fill* called by framework (step)

If no steps follow:

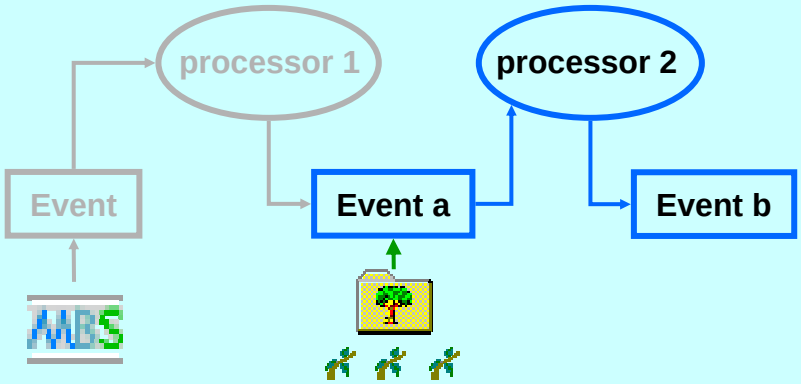
Event a->*Init*:
get processor 1 (from framework)
Event a->*Fill*:
call user event function of processor 1
optionally store Event a (by framework)



Event ::*Init* and *Fill* called by framework (step)

If no steps follow:

Event a->*Init*:
get processor 1 (from framework)
Event a->*Fill*:
call user event function of processor 1
optionally store Event a (by framework)



Event a->*Init*:
1. is event source = processor 1?
YES: get processor 1 (from framework)
2. is event source a file (*TGo4FileSource*)?
YES: get file source

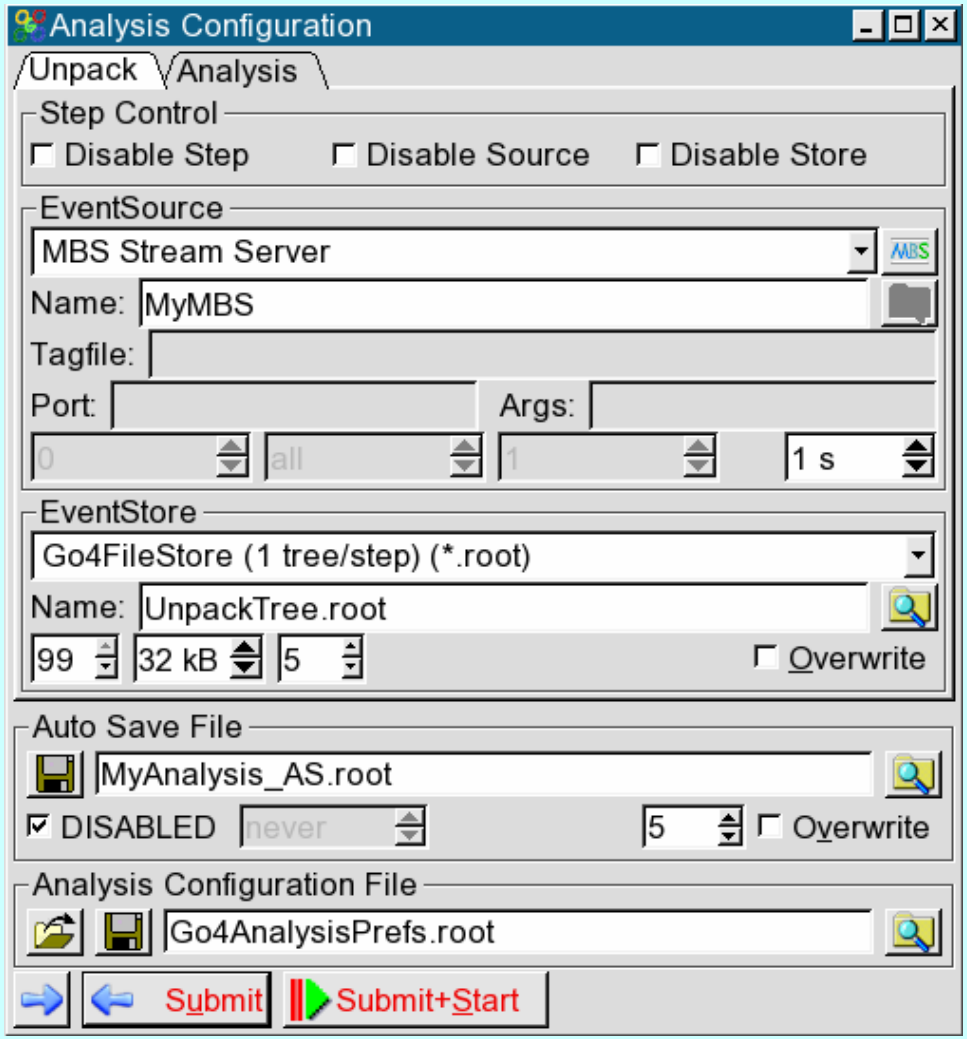
Event a->*Fill*:
1. processor 1?
YES: call user event function of processor 1
2. file source?
YES: call *BuildEvent* function of file source

Go4 class	Derived user class	*implements*	calls
TGo4Analysis	MyAnalysis (optional)	*PreLoop* *UserEventFunc* *PostLoop*	Create steps (factories) Configure analysis Central management
TGo4EventServerFactory	MyFactoryStep_N (optional)	*CreateEventProcessor* *CreateOutputEvent* *CreateInputEvent*	
TGo4AnalysisStep	none		Calls from MyFactoryStep-N: *CreateEventProcessor* *CreateOutputEvent* *CreateInputEvent*
TGo4EventElement	MyEvent_N (optional)	*Init* *Clear* *Fill*	*Fill* calls the event function of processor (below), e.g.: *BuildEvent* (*MyEvent_N) Argument is event object self
TGo4EventProcessor	MyProcessor_N (required)	*BuildEvent* or as called in MyEvent_N	Calls *GetInputEvent* Gets output event as argument
TGo4Parameter	MyParameter1	*UpdateFrom*	Update data as wanted

IO classes	Event objects	Functionality
	TGo4MbsEvent, *TGo4MbsSubEvent*	MBS format 10,1
	TGo4CompositeEvent	Complex event structures „toolbox“
TGo4MbsFile *TGo4MbsEventServer* *TGo4MbsStream* *TGo4MbsTransport* *TGo4RevServ*		read from MBS *.lmd connect to Mbs connect to remote event server mrevserv
TGo4FileSource *TGo4FileStore* *TGo4BackStore*		ROOT TTree in TFile ROOT TTree in memory online TTree::Draw()
TGo4EventElement		Default event class no data, no output calls BuildEvent
TGo4EventProcessor		Default processor class hook for input event
Go4ExampleUserSource		User event source

Tabs for steps

2Step example



Event input

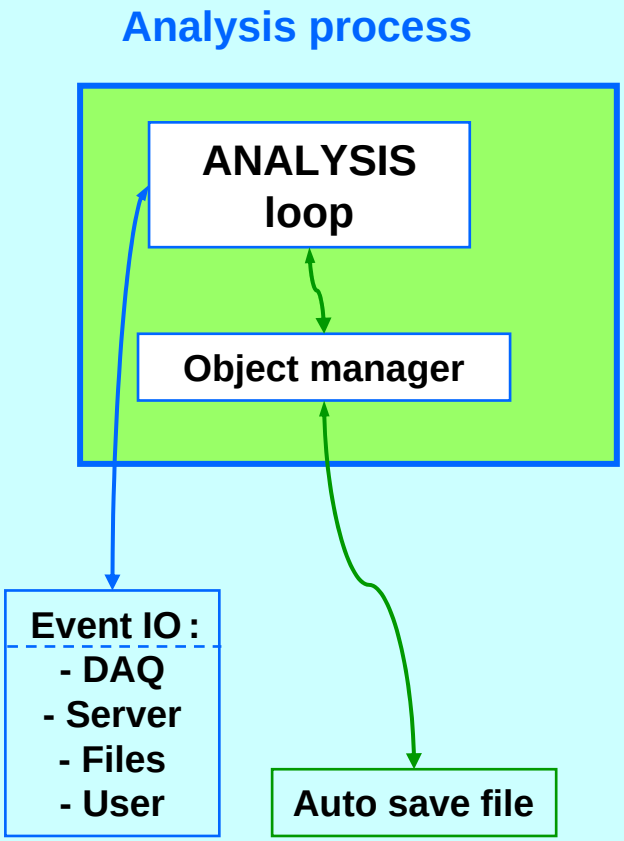
Event output

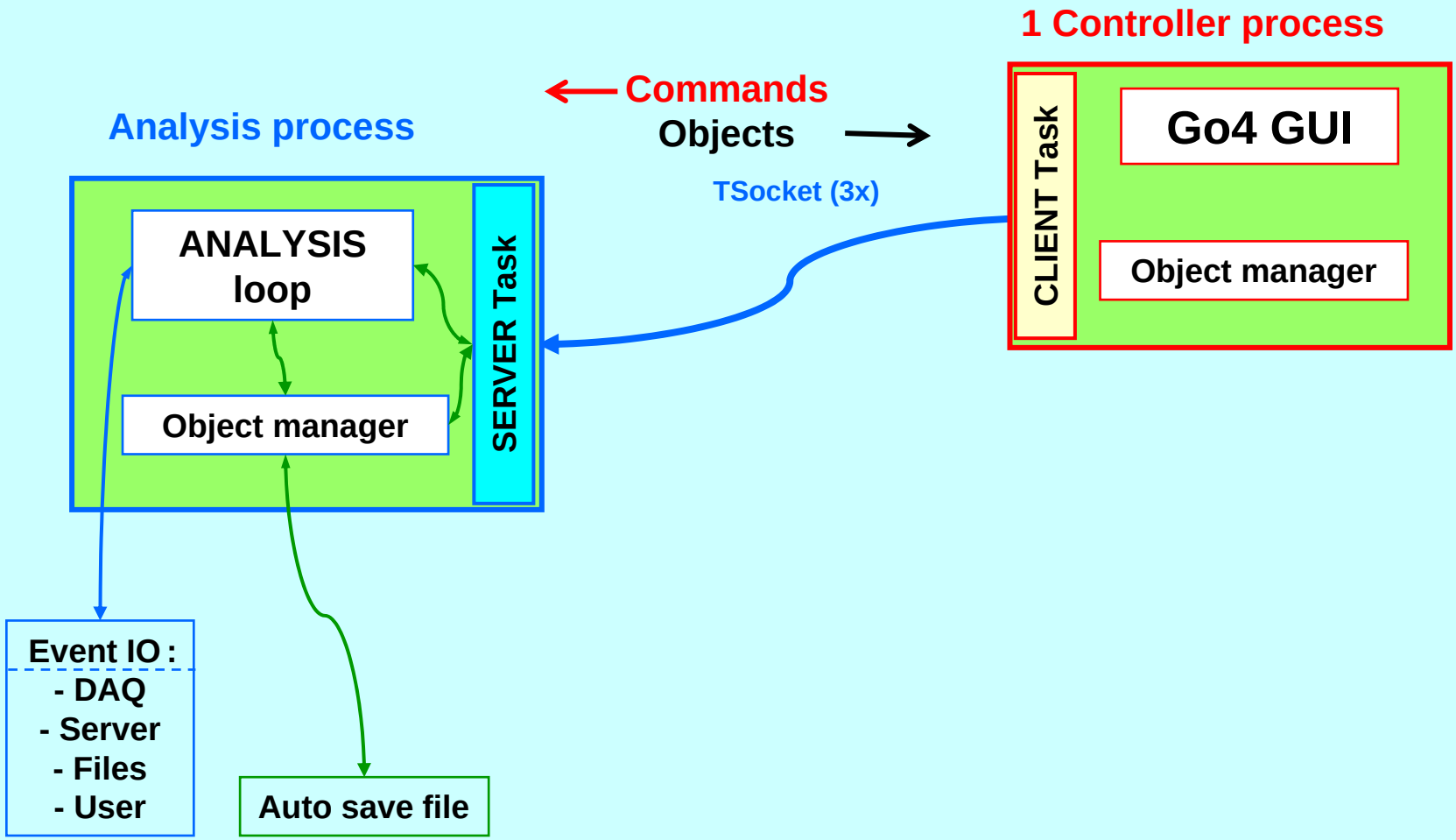
Object persistency

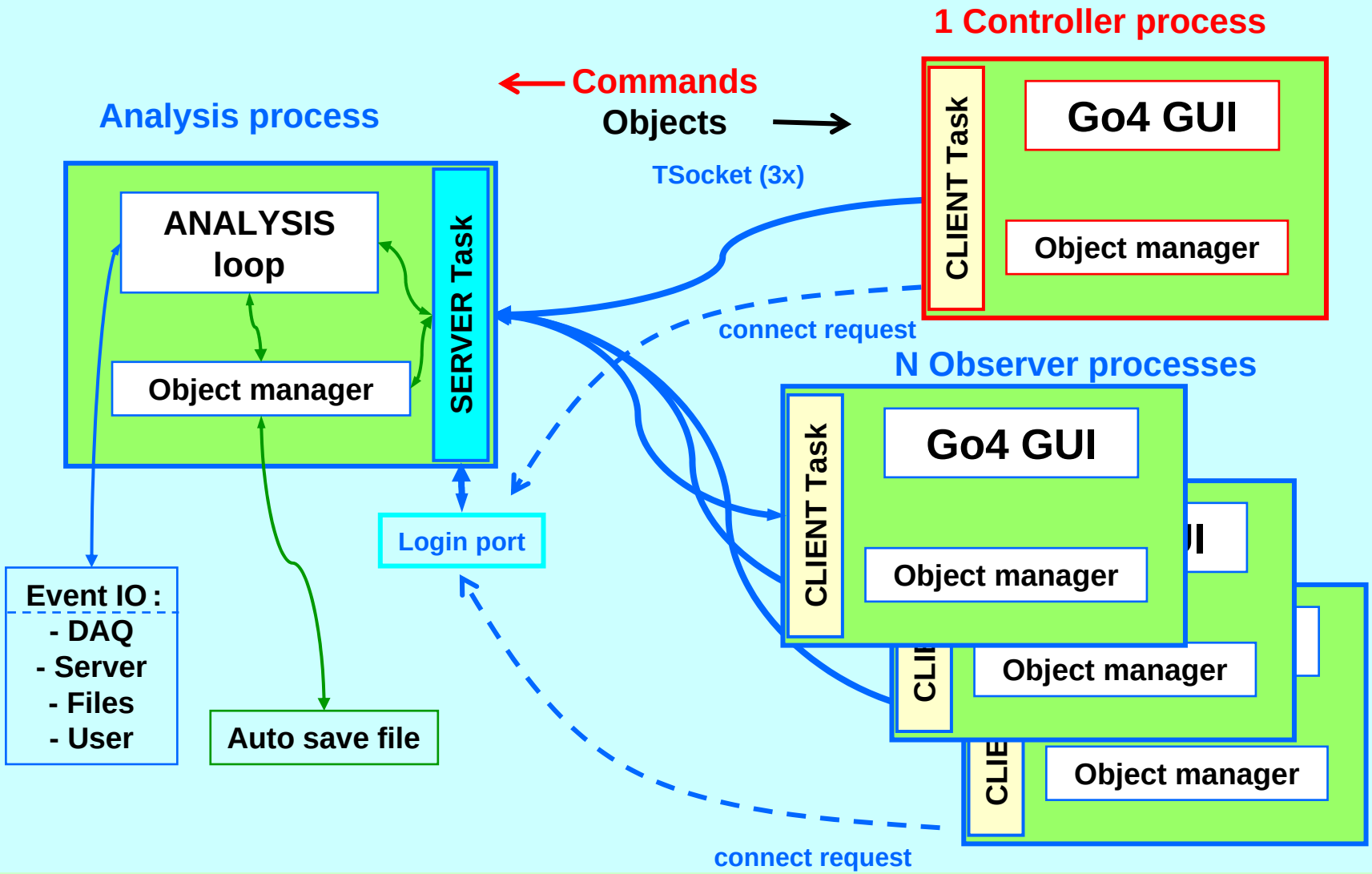
Load/save config

- **Conditions**
  - inspired from GOOSY
  - window condition : check 1 (2) value(s) against 2 limits (pairs of limits)
  - polygon condition : check if point (x,y) is inside/outside polygon
  - indexable arrays of conditions
  - allows for analysis flow control
  - statistics (true/false counters)
  - interactive control (GUI editor)
- **Parameters**
  - User classes keeping parameter variables
  - interactive control (generic GUI editor)
  - value protection (update controlled by user function)
  - allows for specific analysis control
  - "cheap" commands (executed through editor) easy to implement
  - supports besides atomic data types also fit objects

Set up in code or macros




1 Controller process
Commands
Objects
Analysis process
TSocket (3x)
CLIENT Task
Go4 GUI
Object manager
ANALYSIS loop
SERVER Task
Object manager
Event IO :
- DAQ
- Server
- Files
- User
Auto save file


1 Controller process
Commands
Objects
Analysis process
TSocket (3x)
CLIENT Task
Go4 GUI
Object manager
ANALYSIS loop
SERVER Task
Object manager
connect request
N Observer processes
CLIENT Task
Go4 GUI
Object manager
Login port
Event IO :
- DAQ
- Server
- Files
- User
Auto save file
Object manager
Object manager
connect request