



Go4 v2.8 Analysis Design

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

CHEP 2004



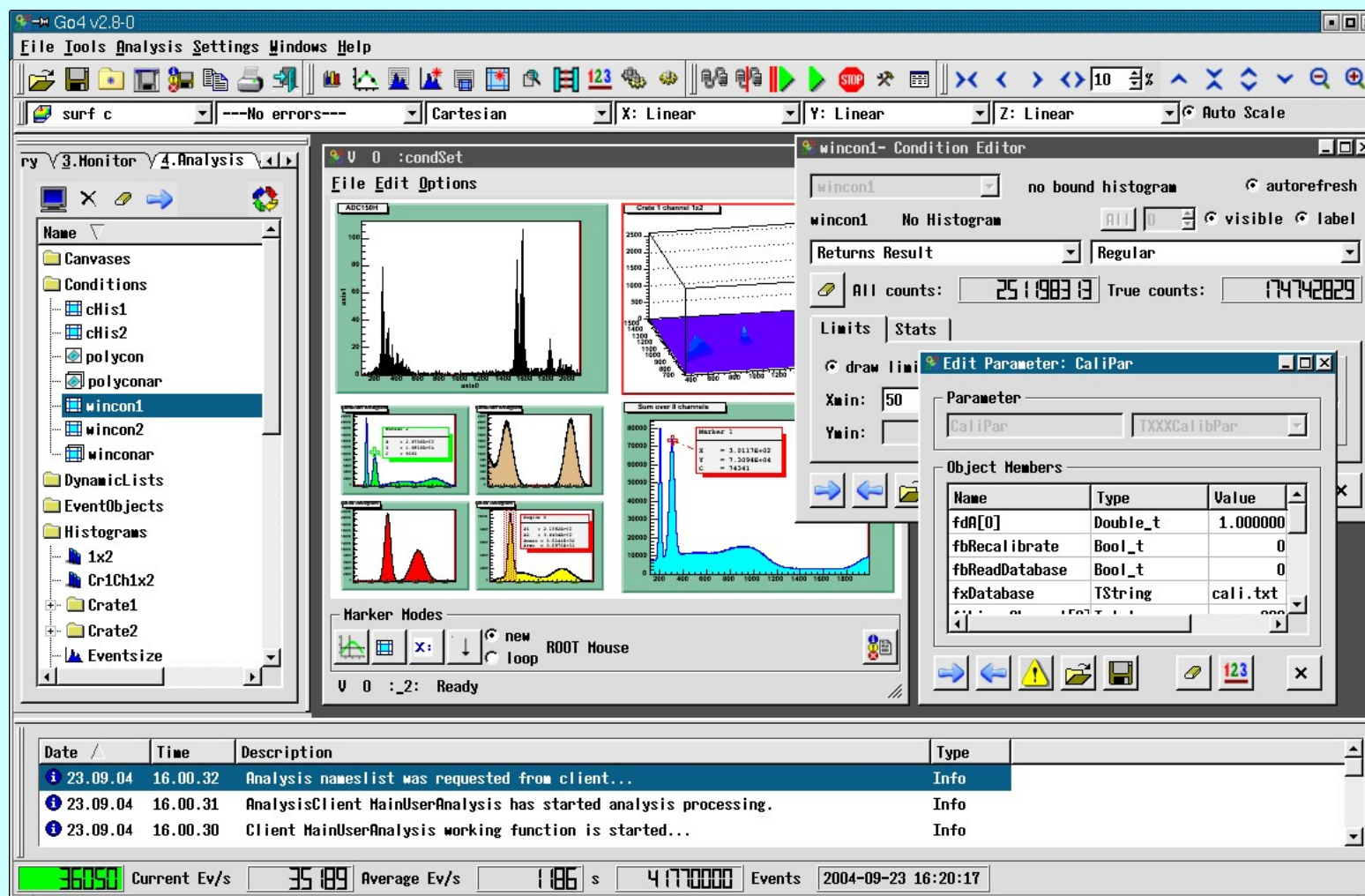
Content



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

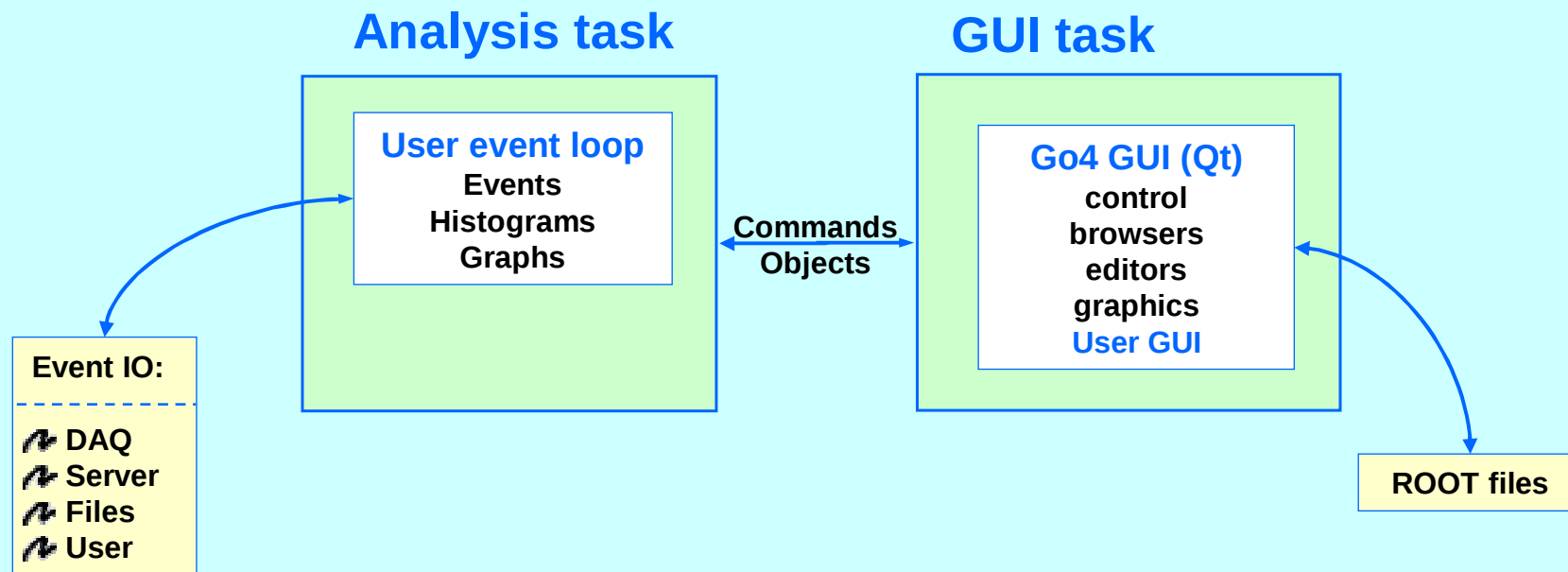


Screen shot





Go4 tasks: GUI & analysis





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)



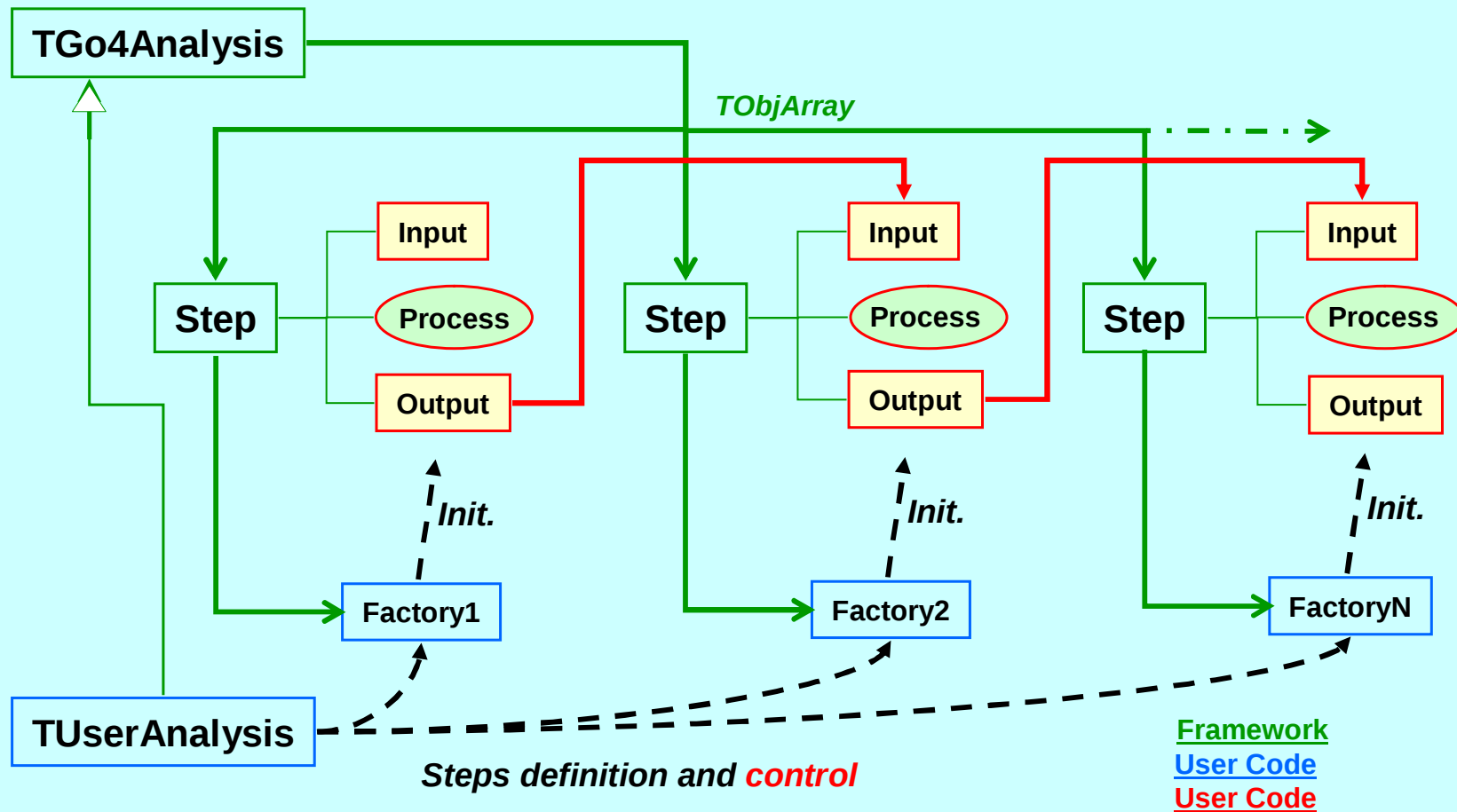
Content



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

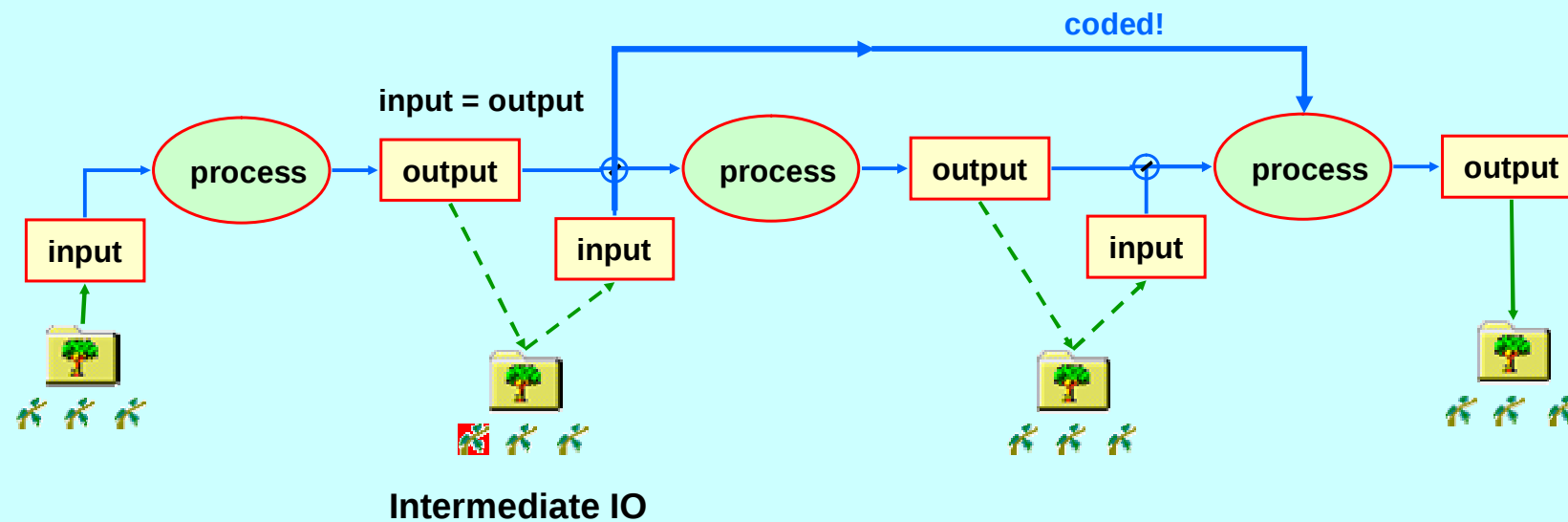


Analysis steps (objects)





Analysis steps (object IO)





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

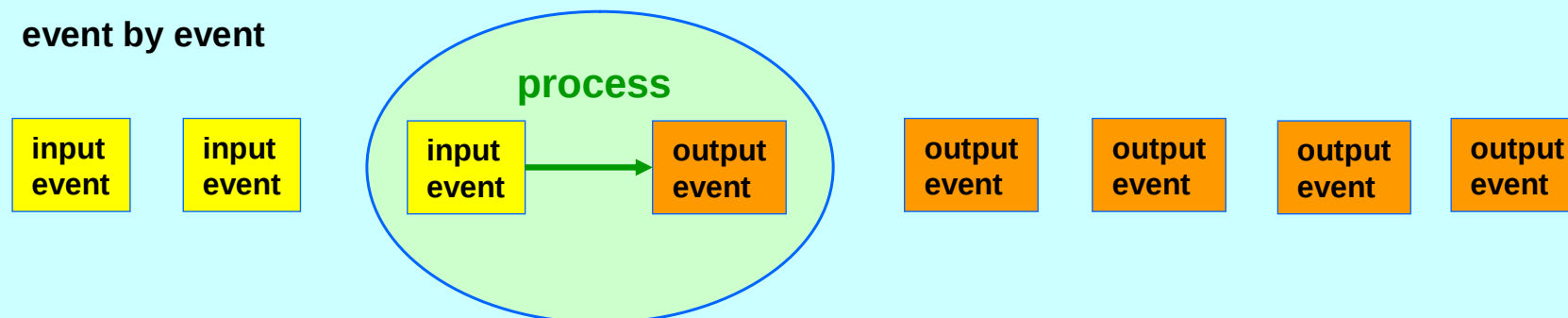
- Event stacks
- Hierarchy of steps
- Concurrent steps
- Control of multiple I/O per step
- Logical mesh setup control



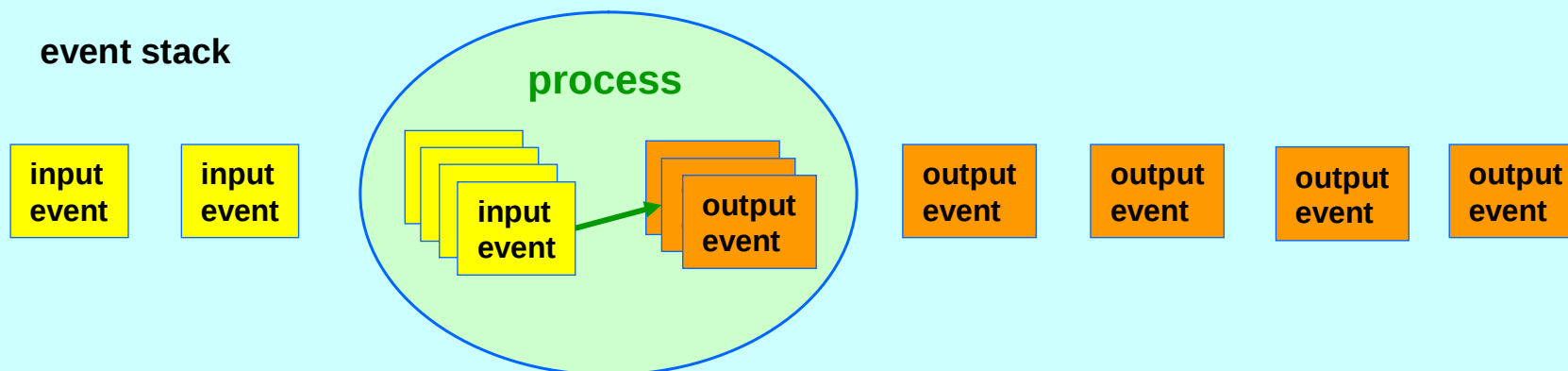
Event stacks

- 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

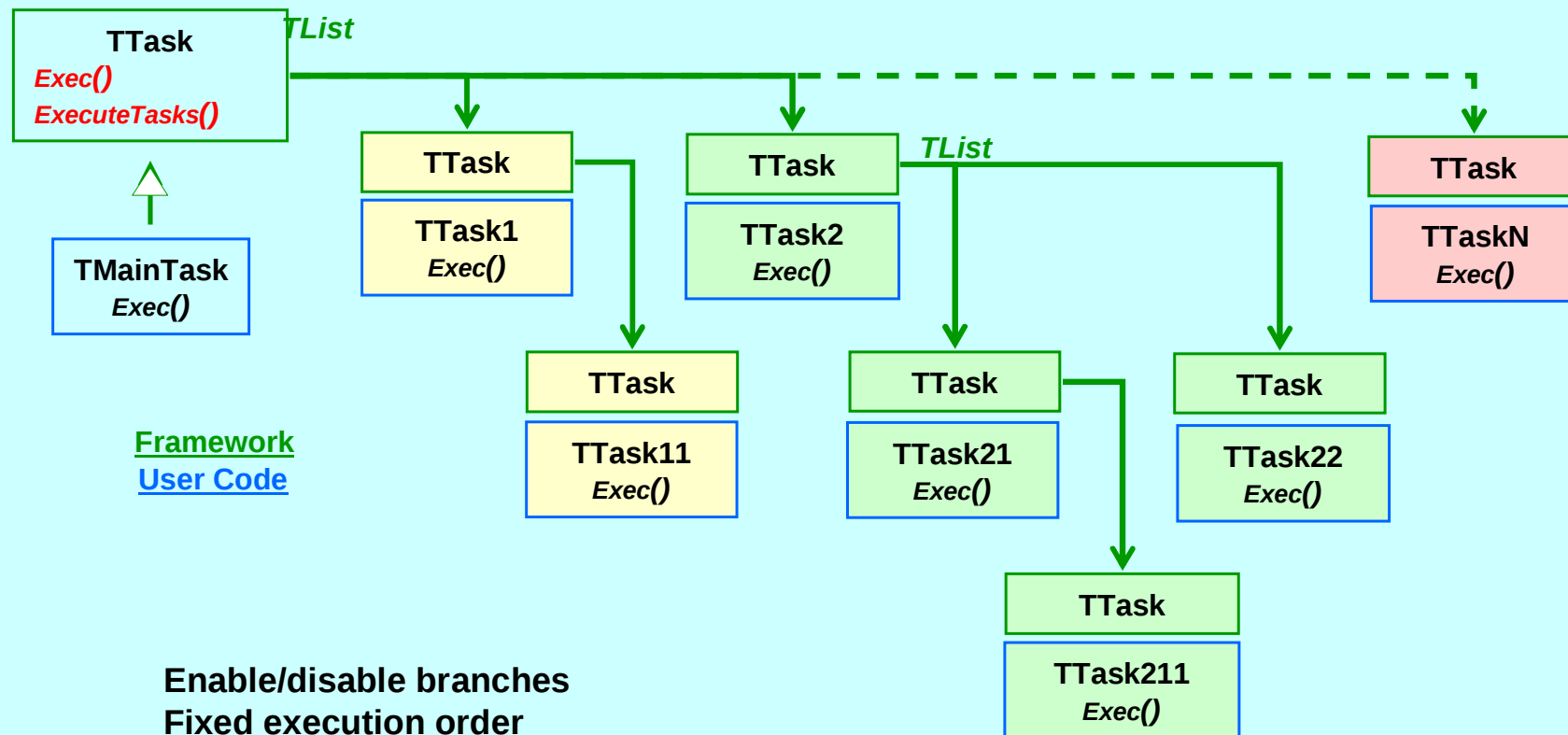


event stack



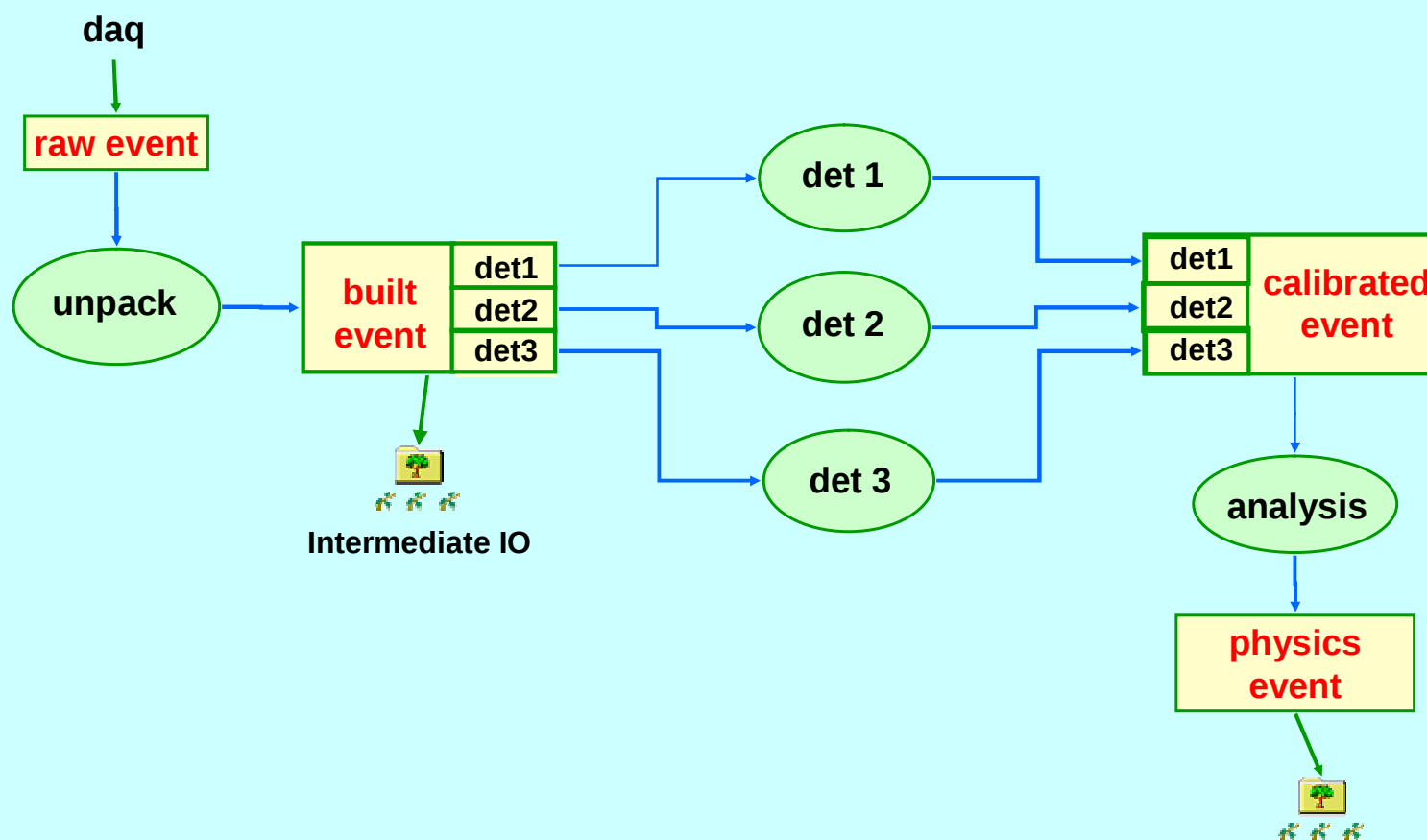


Hierarchy: like ROOT TTask mechanism



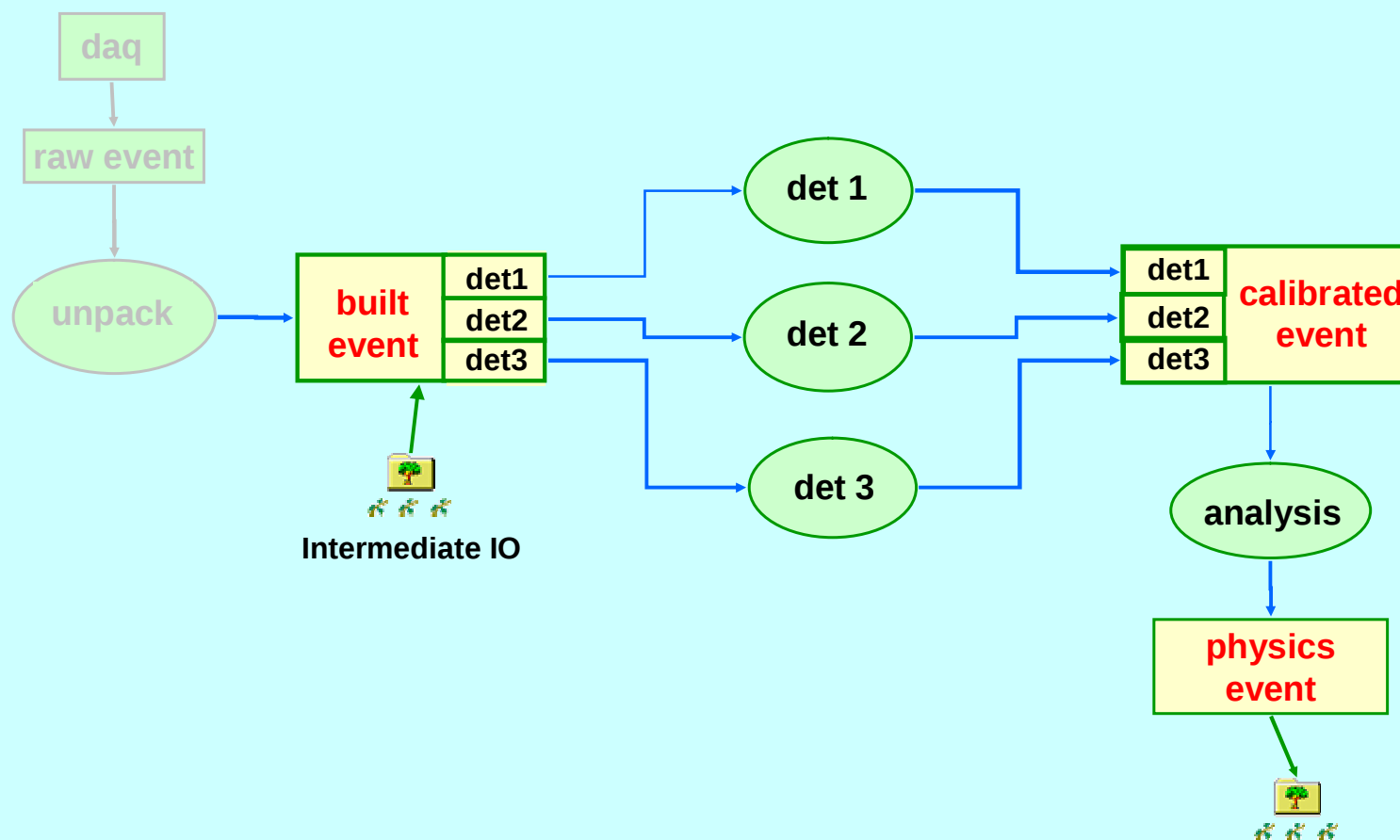


Concurrent analysis steps



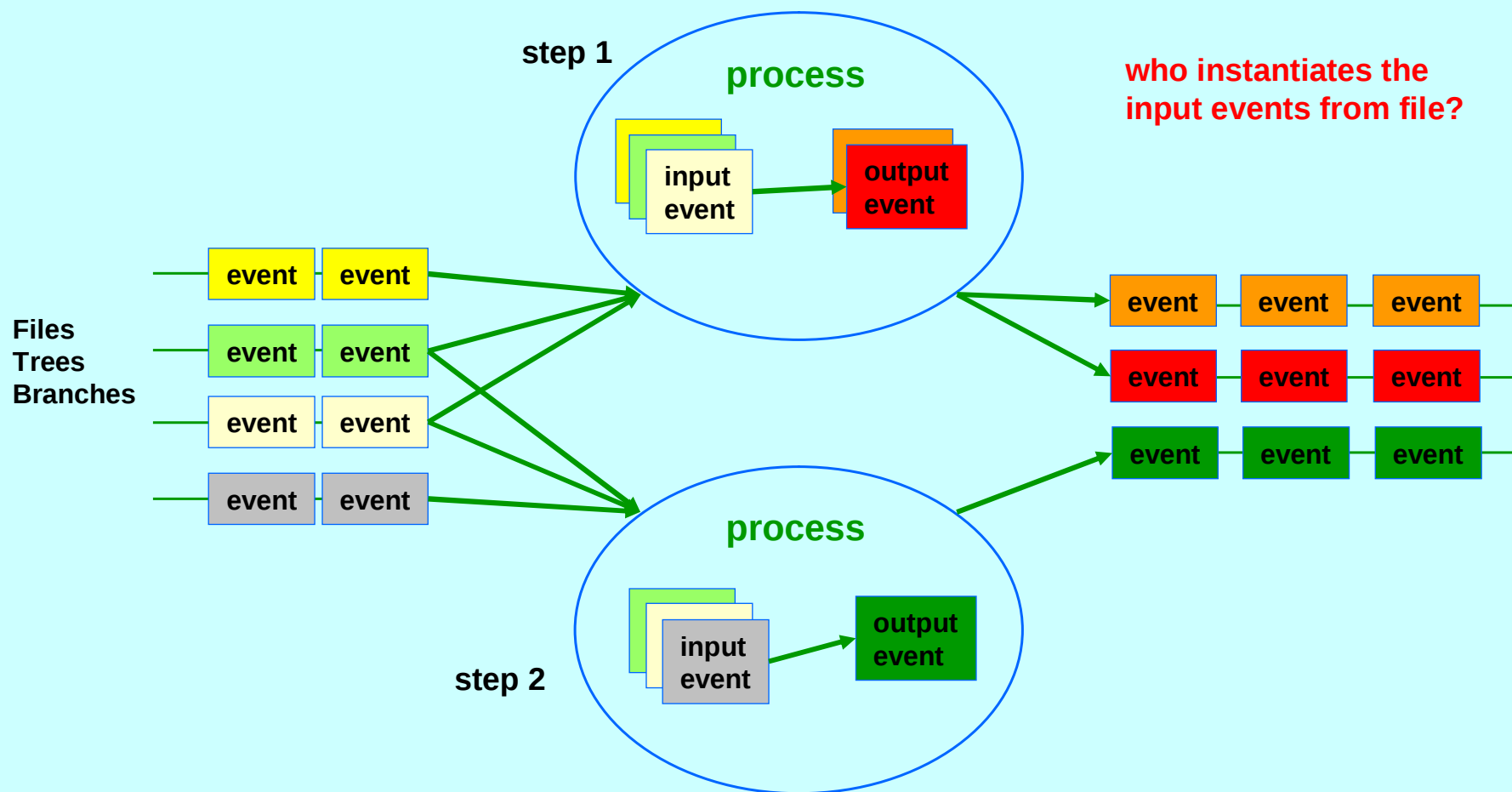


Concurrent analysis steps



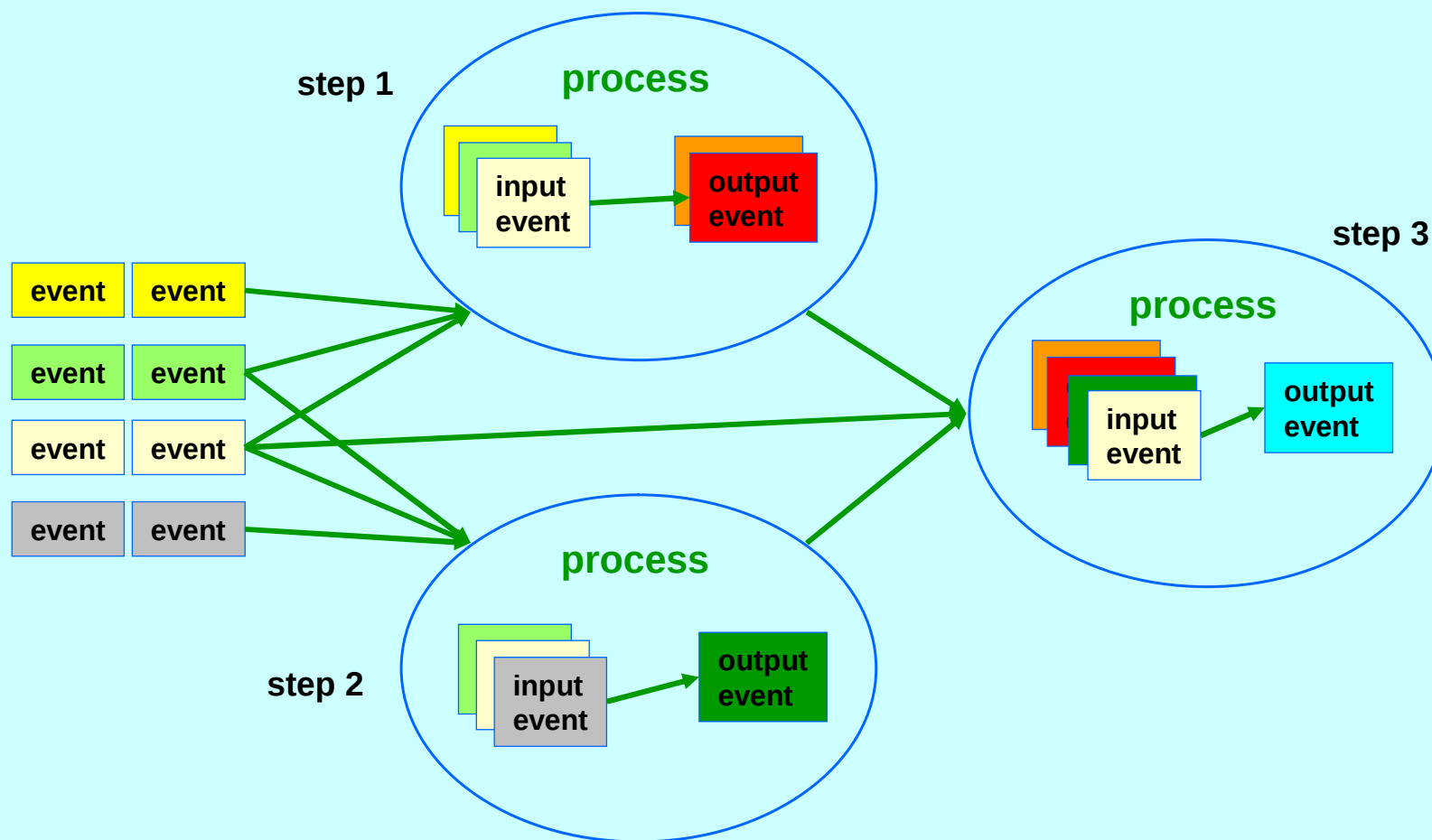


Multiple IO





Multiple IO and data generations

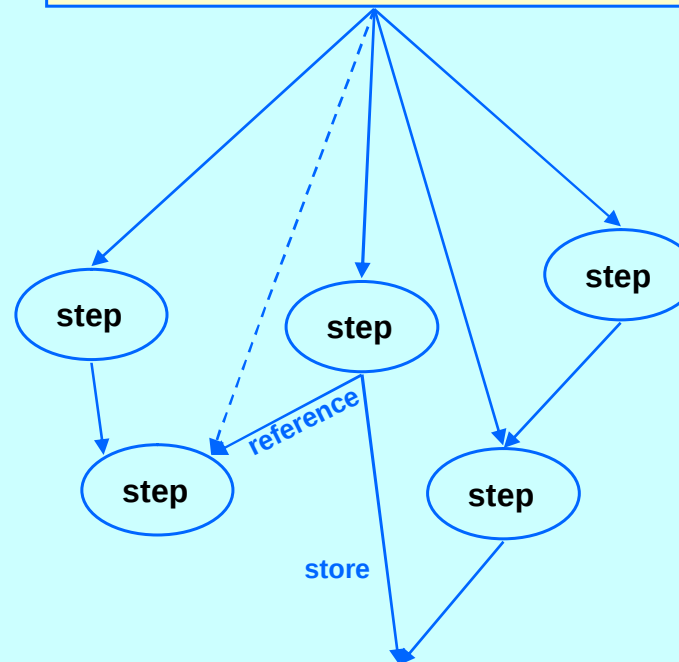




Analysis meshes, not hierarchy

Input:

Multiple files, trees, branches



Output:

Multiple files, trees, branches

Execution order of steps
evaluated at run time!

Avoid back reference!

TTask not suitable!

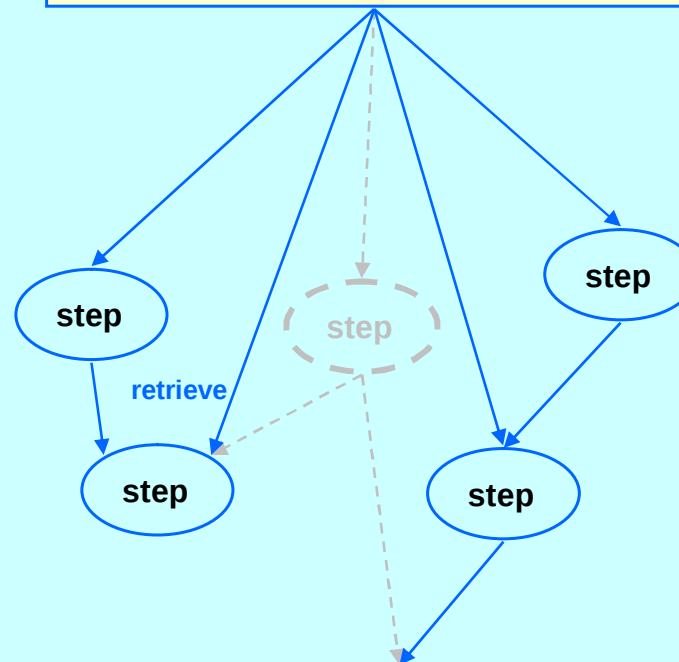
Intermediate store/retrieve



Analysis meshes, not hierarchy

Input:

Multiple files, trees, branches



Output:

Multiple files, trees, branches

Execution order of steps
evaluated at run time!

Avoid back reference!

TTask not suitable!

Intermediate store/retrieve



Content



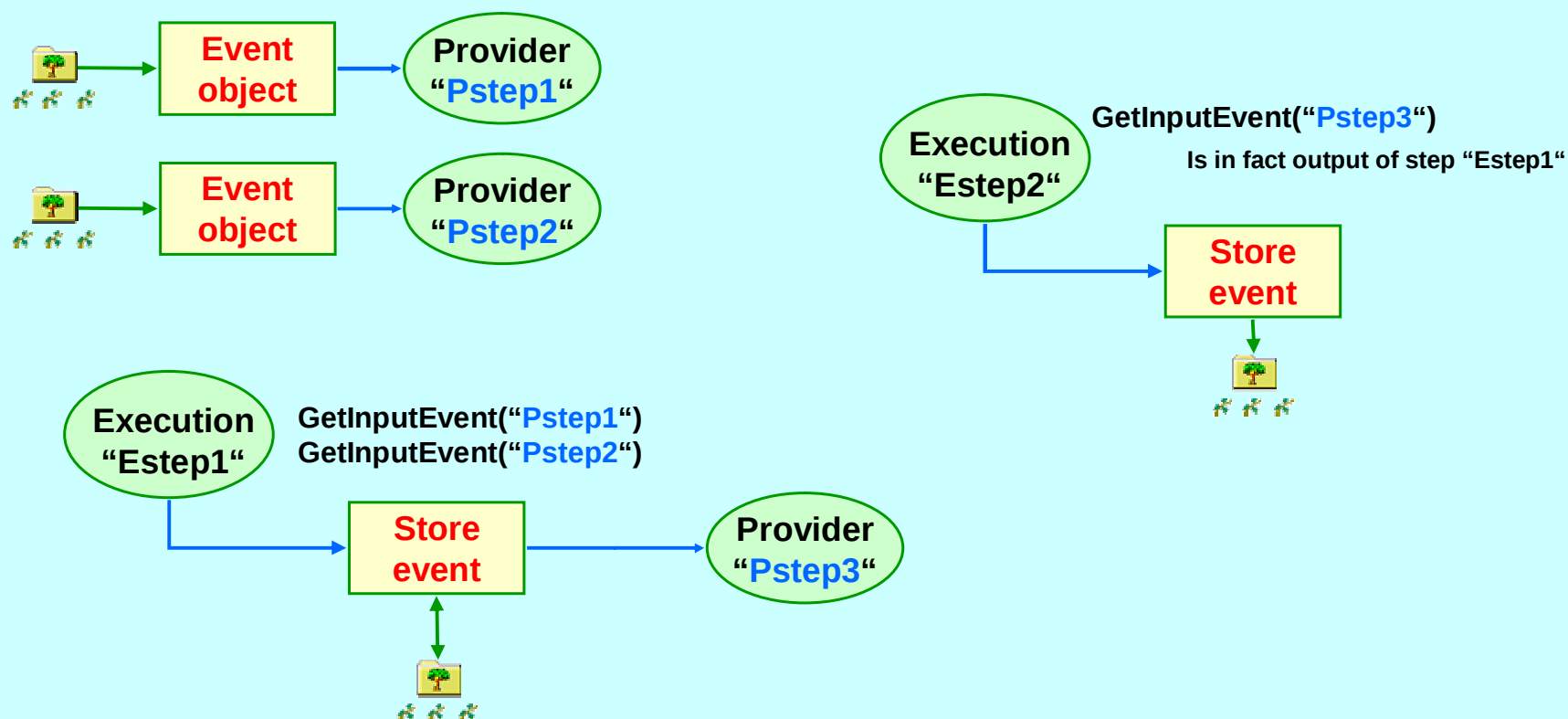
- Warm up
- Current analysis design
- Analysis control
- Advanced requirements
- **Solution I**
- Solution II



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 be 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**

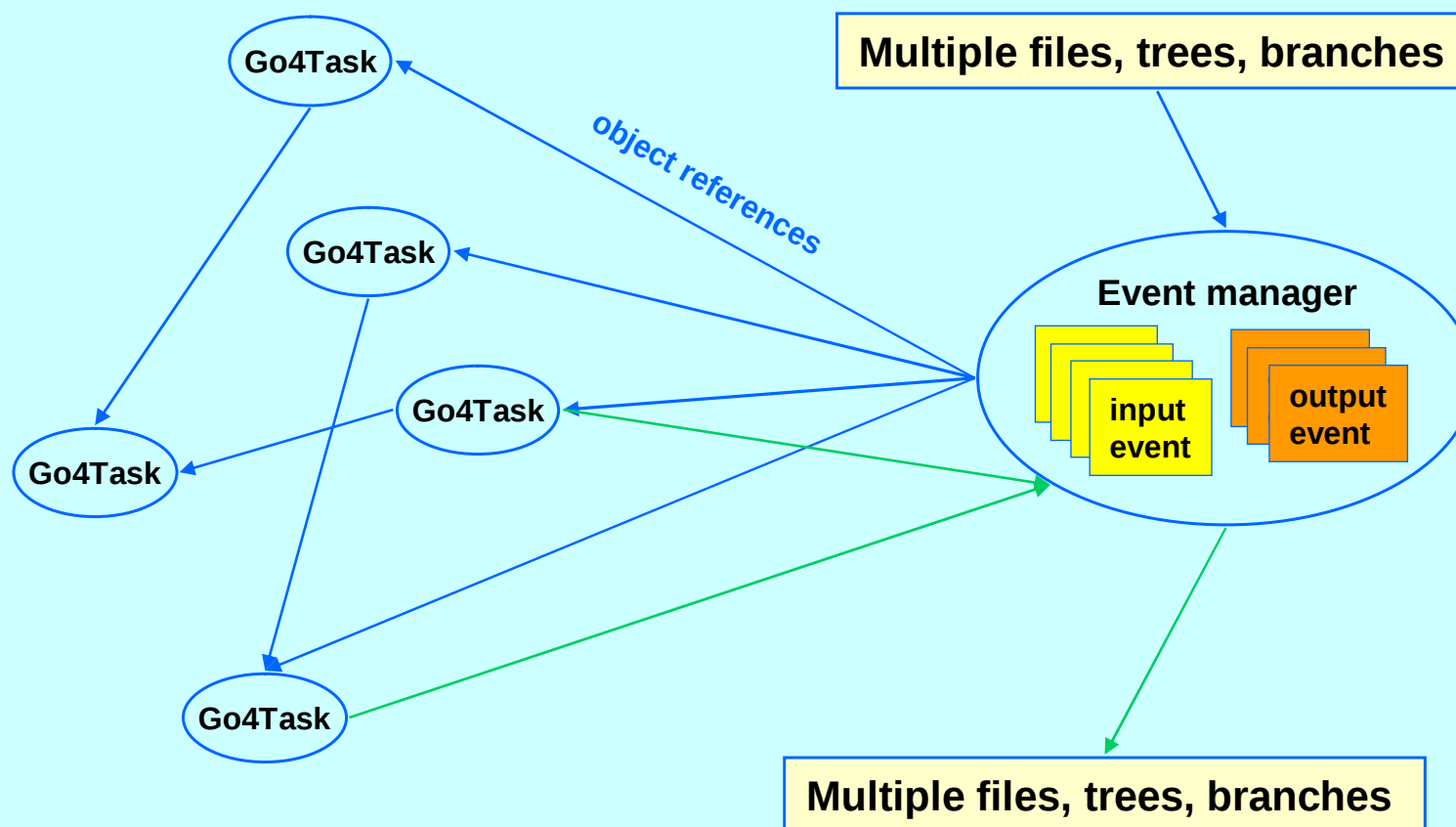


Solution II: New components, design

- 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)**

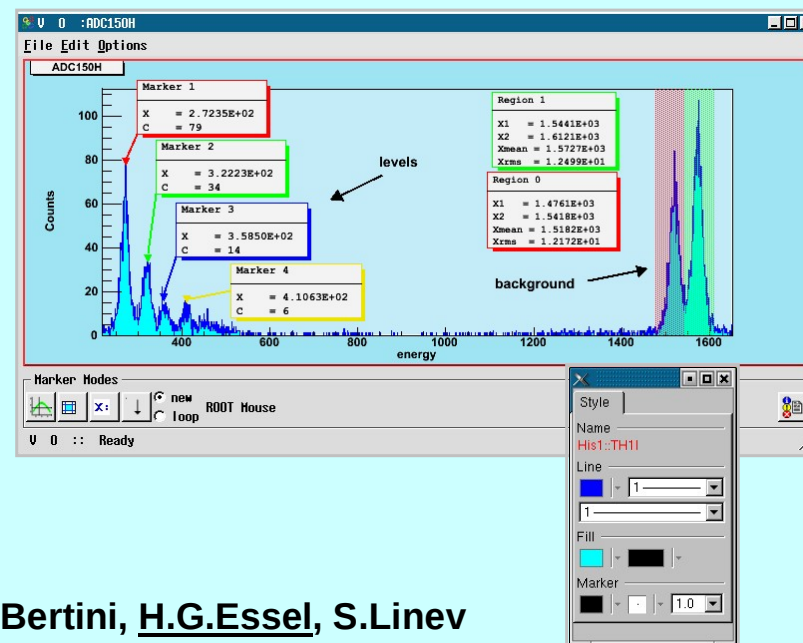
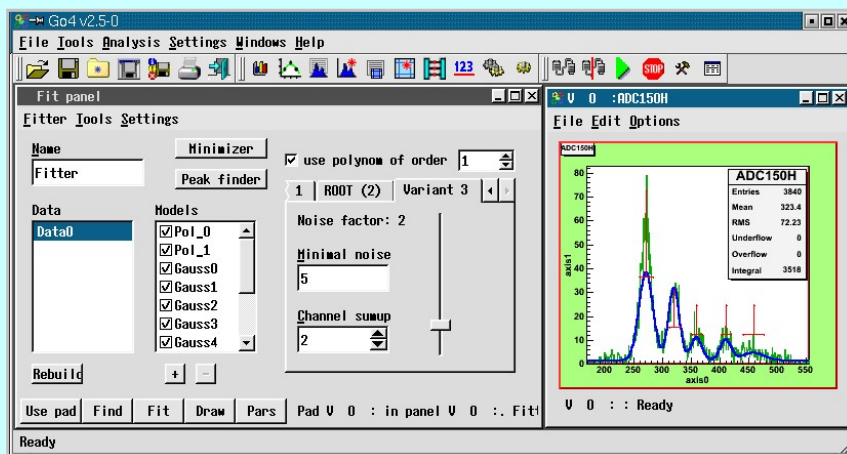
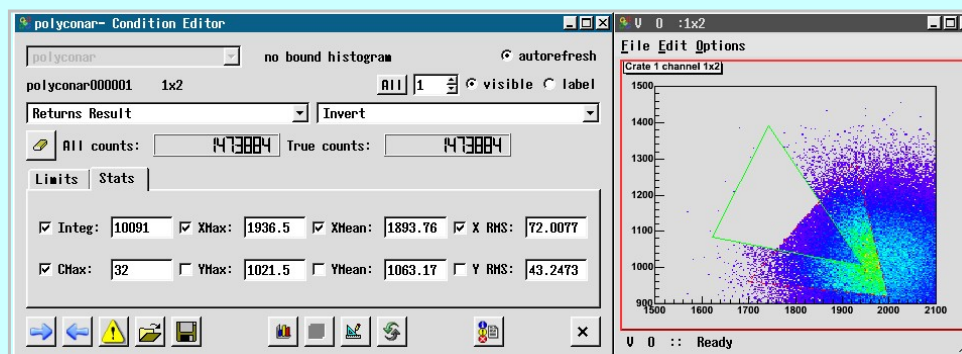


Event Manager





The end



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