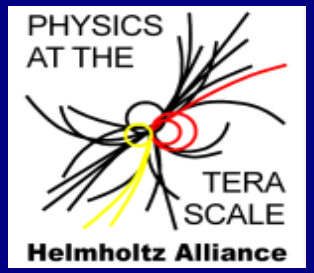
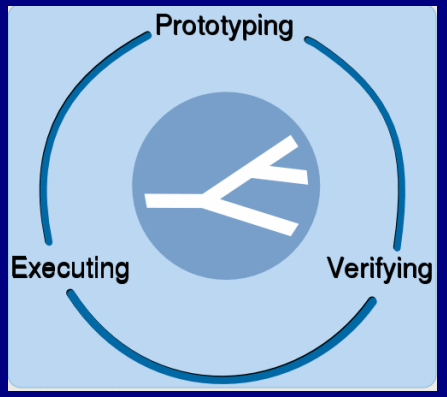


CMS and ILC Applications within the Visual Physics Analysis Project

Tatsiana Klimkovich for the **VISPA** group

M.Brodski, M.Erdmann, R.Fischer, A.Hinzmann, T.Klimkovich, D.Klingebiel, M.Komm, G.Müller, T.Münzer, J.Steggemann, T.Winchen

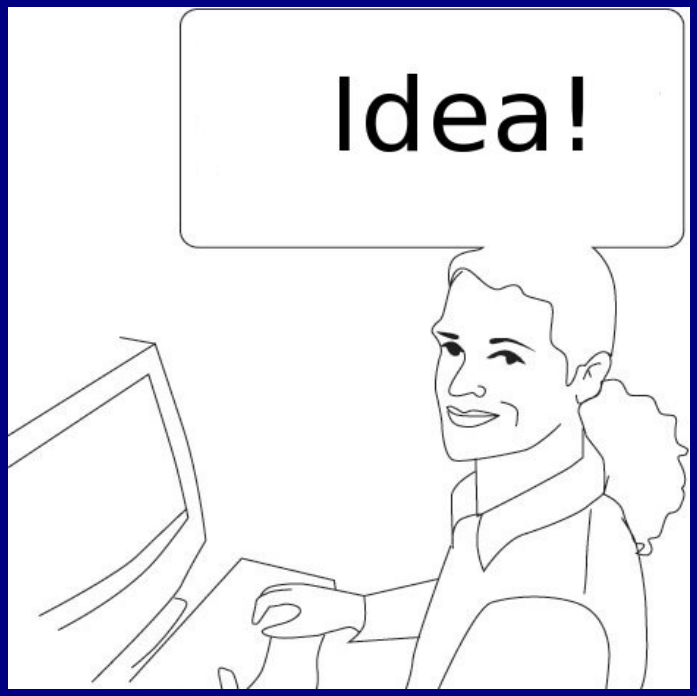
Workshop of the Helmholtz Alliance “Physics at the Terascale”
Hamburg, November 2009

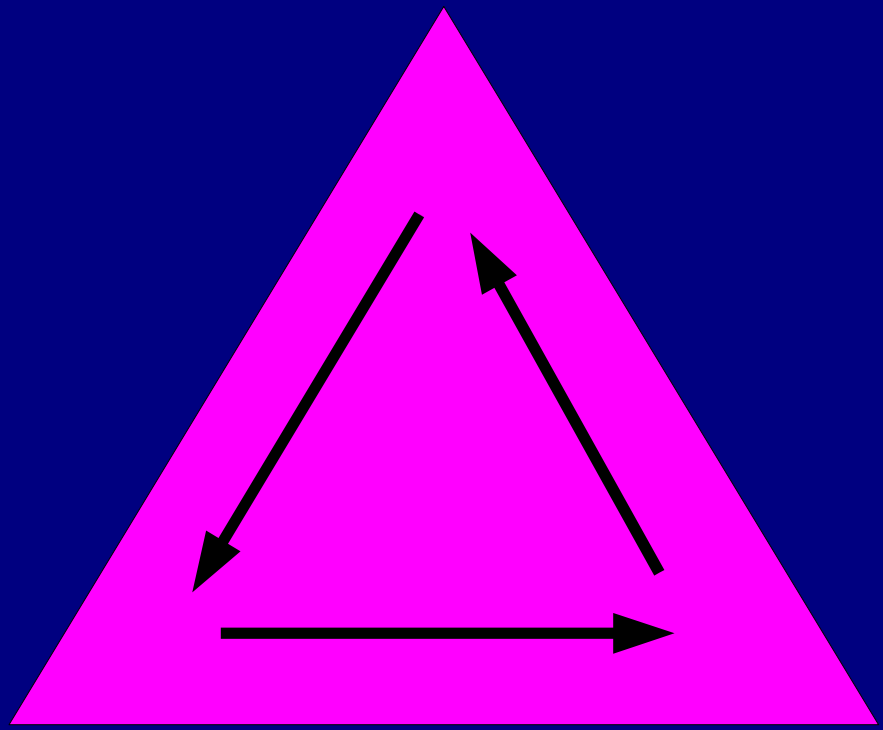


Contents

- **Physics Analysis in High Energy Physics experiment**
- **VISPA Applications at CMS and ILC**
- **Look & feel with analysis example from CMS**

High Energy Physics Analysis

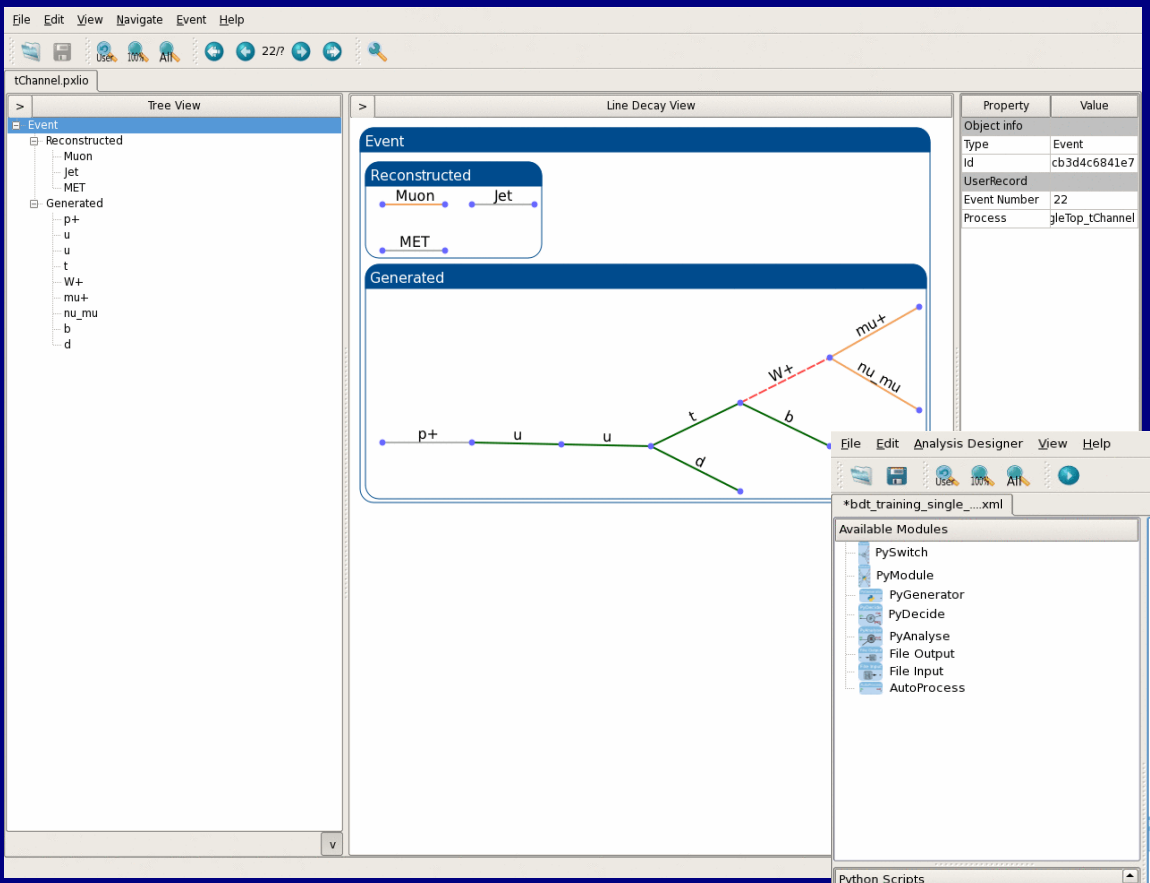


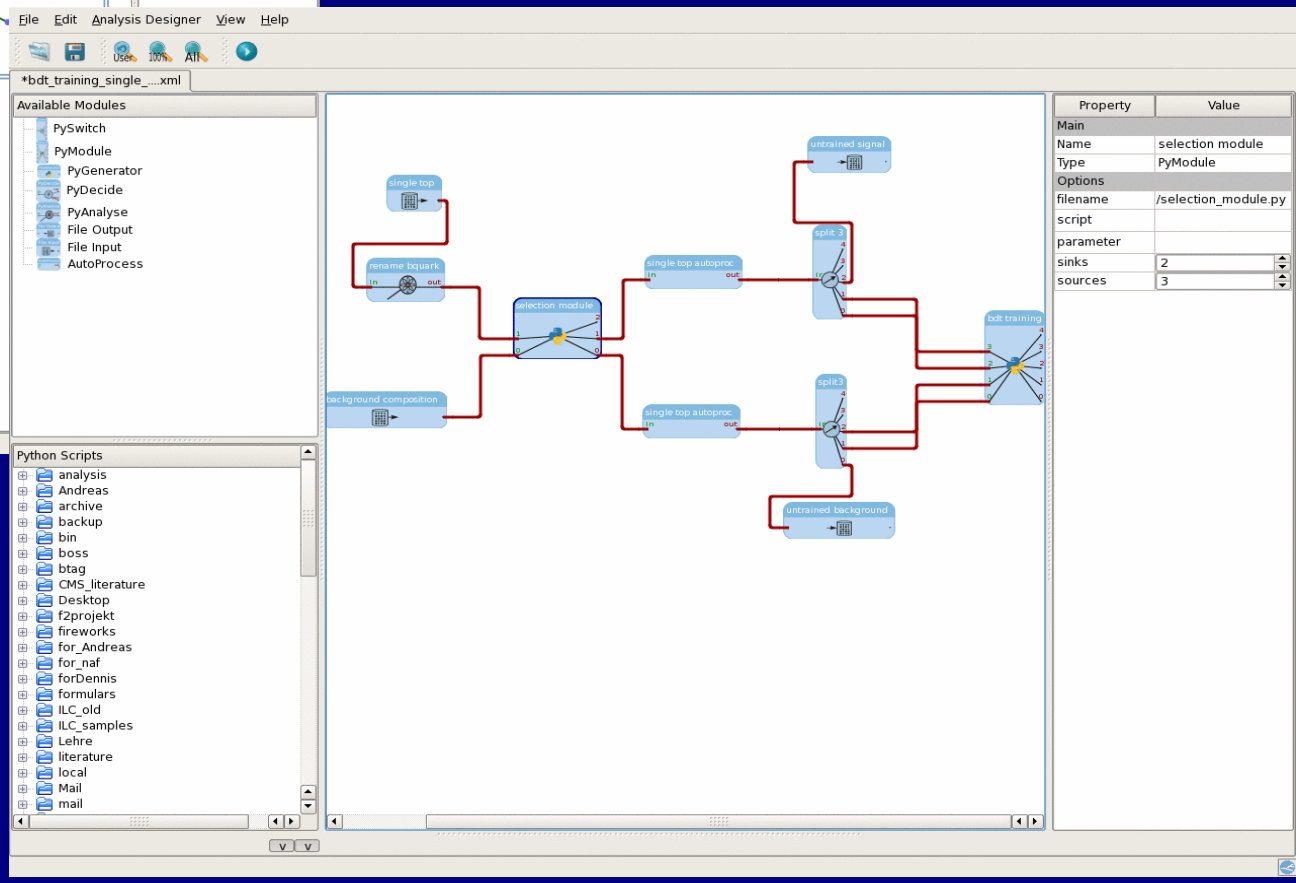


Wish List of the Analyser

- To **start fast**
- Small summary **data sets** (ntuples)
- **Modular structure** of the analysis
- Many **reusable components**
- Reduced time for **analysis cycle**
- Perform analysis on the **laptop** (including MAC)

VISPA: Visual Physics Analysis





Mixture of graphical and textual programming (like LabView)

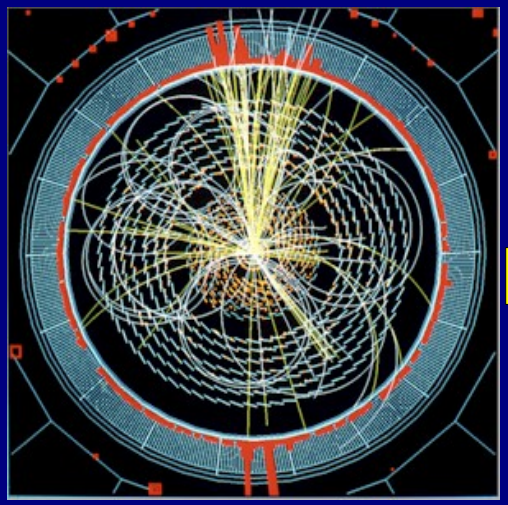
VISPA: Development environment for HEP analyses

Graphical part

- Multi-purpose window
- Visualisation of analysis data and analysis flow in one Graphical User Interface (GUI)
- Module steering

Underlying Software PXL

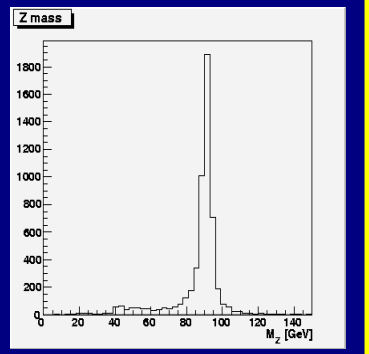
- PXL is a C++ toolkit for high-level physics analysis
- Version 2.5.2 (October 2009)
- Successor of PAX (Physics Analysis Expert) (2002-2007)



Experiment Software Framework

VISPA

PXL

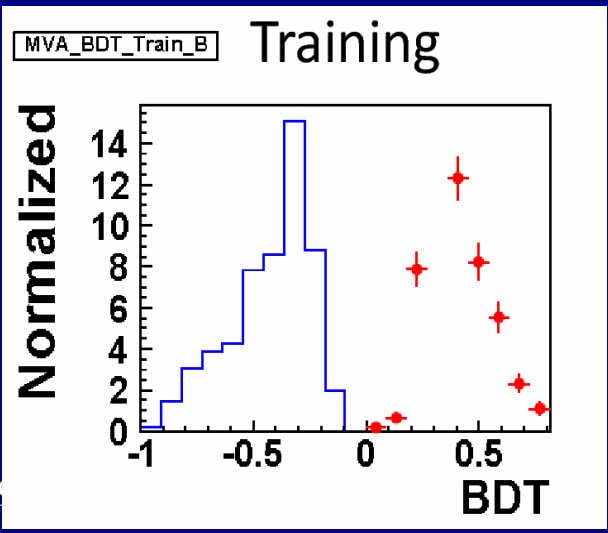


Python Interface to PXL

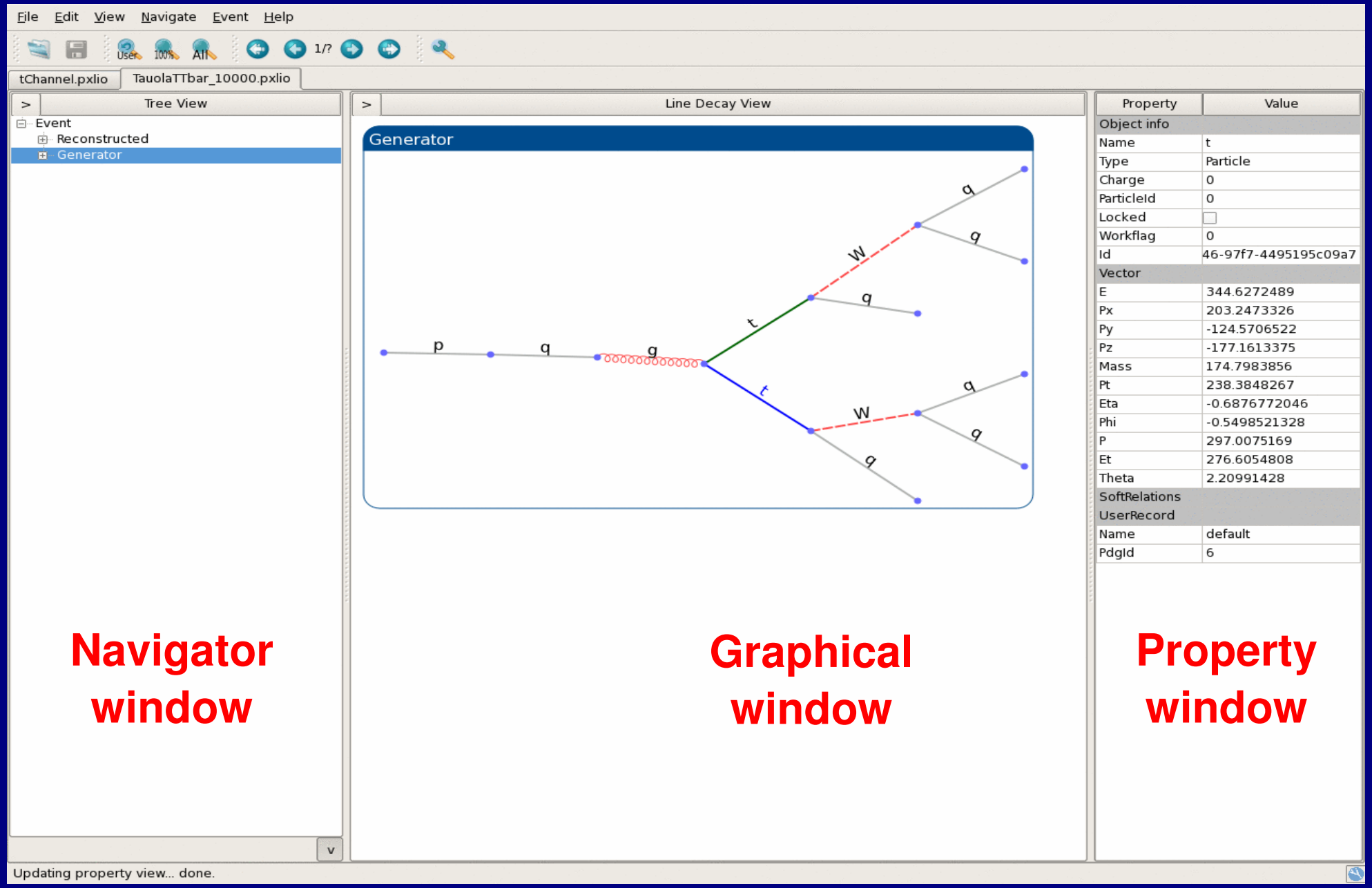
Python extension PyPXL: enables the usage of all PXL objects and their methods within Python programs:

- Python code is easy to read
- Less code compared to C++
- Dynamic typing
- Automatic memory management

- Use of SWIG for automatic interface of C++ to Python
- Histogramming: PyROOT

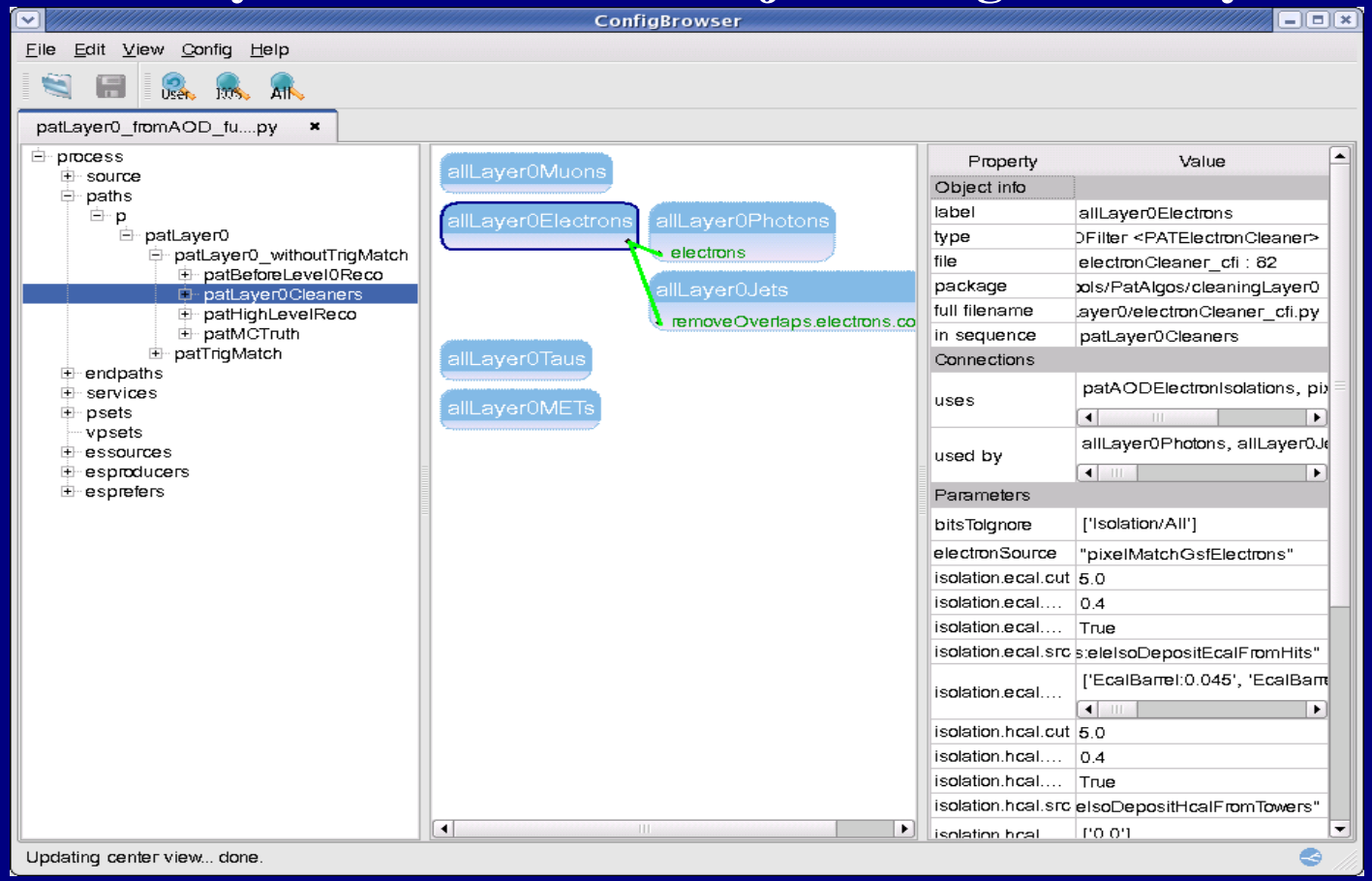


VISPA graphical part: Common Layout of VISPA Window



VISPA graphical part: applications at CMS

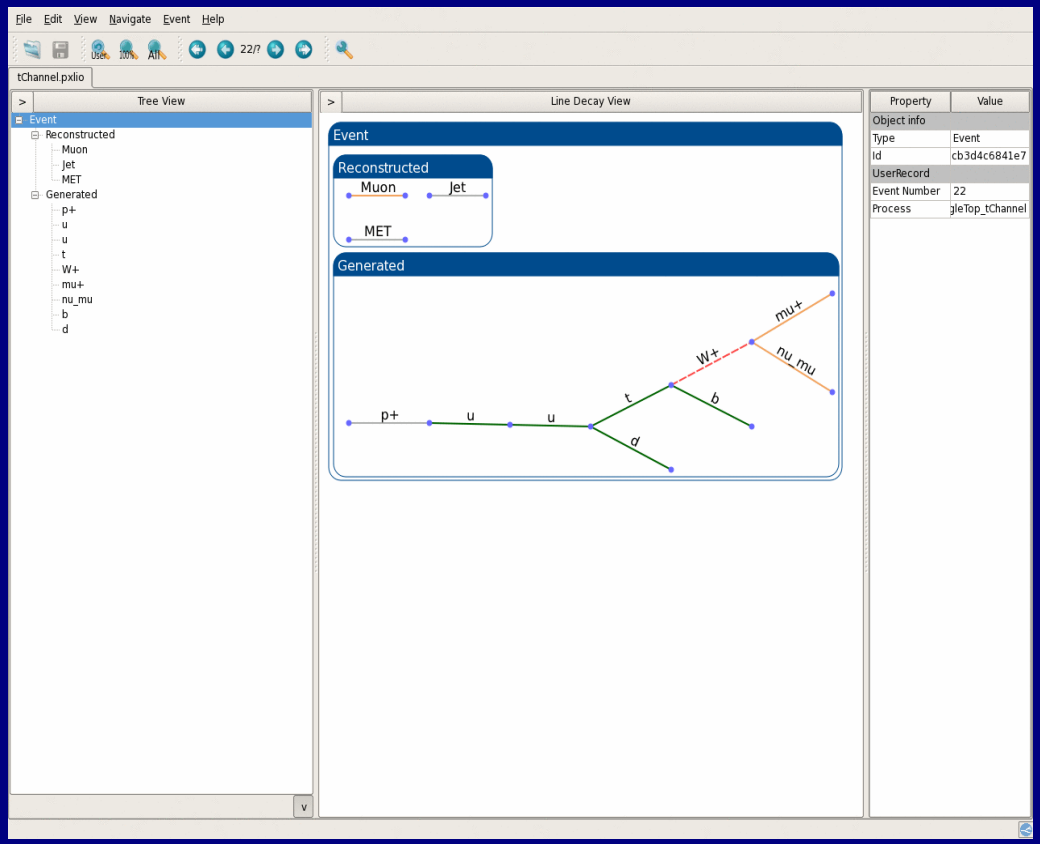
- **CMS Configuration Browser** uses VISPA GUI as a platform
  - Allows visually browse and edit the job configuration system of CMS

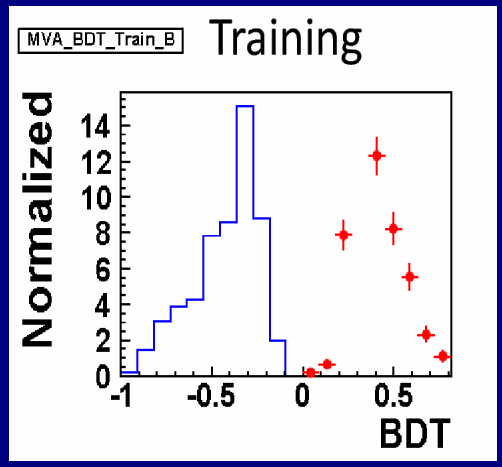


- **EDM Browser:** an event data browser at CMS, developed as VISPA GUI plugin
  - Allows to inspect the content of CMS data files, event by event

Structure of Physics Analysis

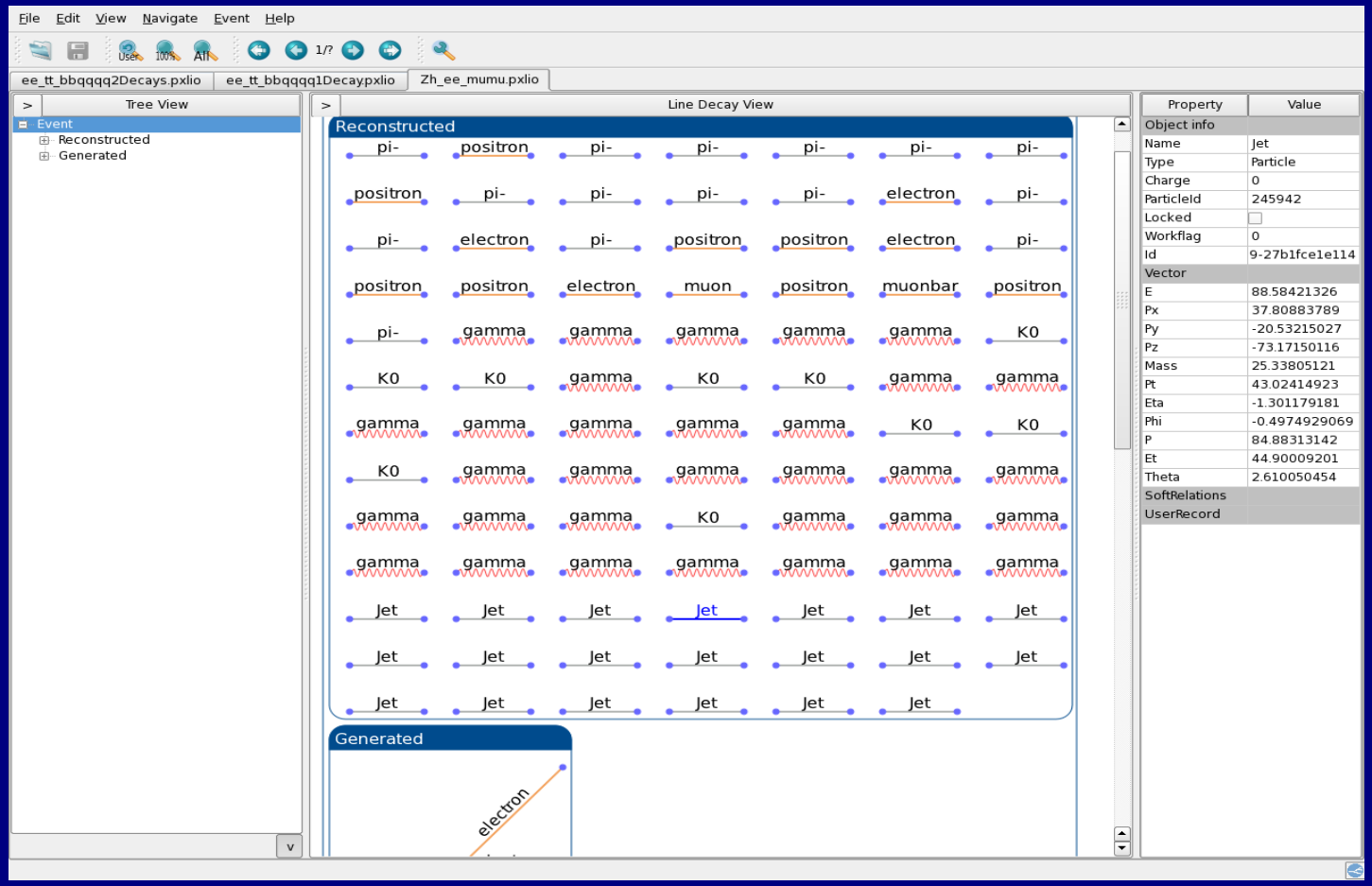
Data input → **Data selection** → **Advanced analysis** → **Histograms**



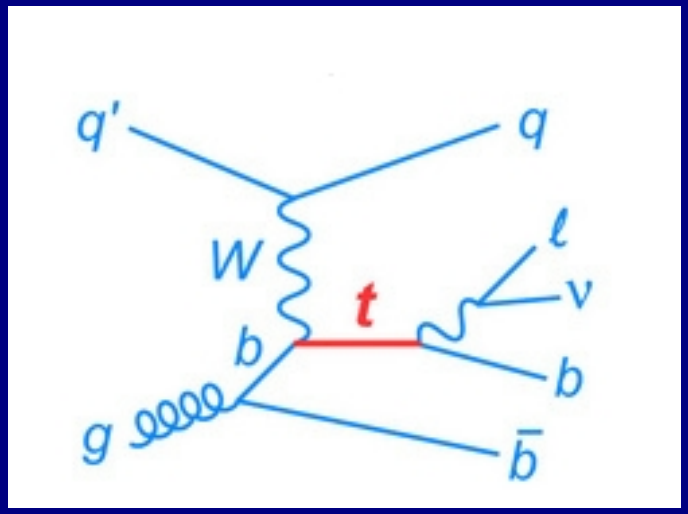


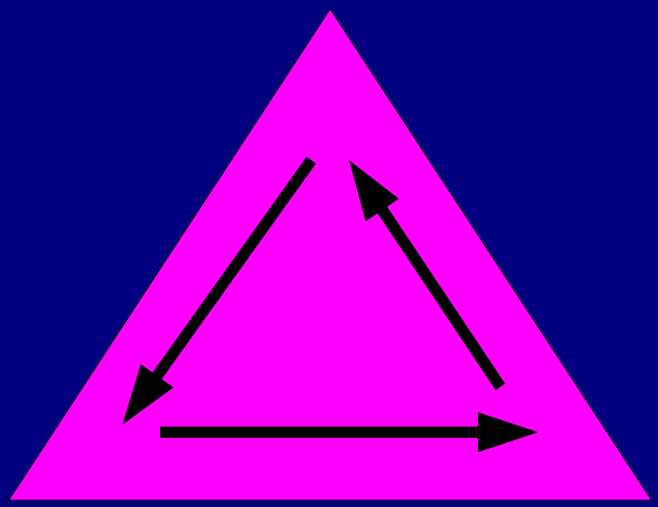
VISPA Application for the ILC Analysis

- VISPA can be used for ILC analyses
- LCIO format files can be converted into pxlio format
- Converter implemented as a Marlin processor

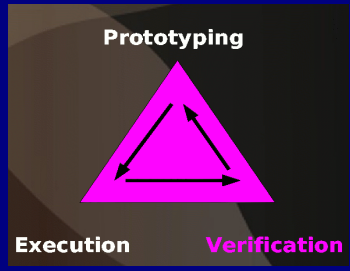


Look closer: Single Top analysis at CMS





Single Top Analysis
First step: inspect an input file



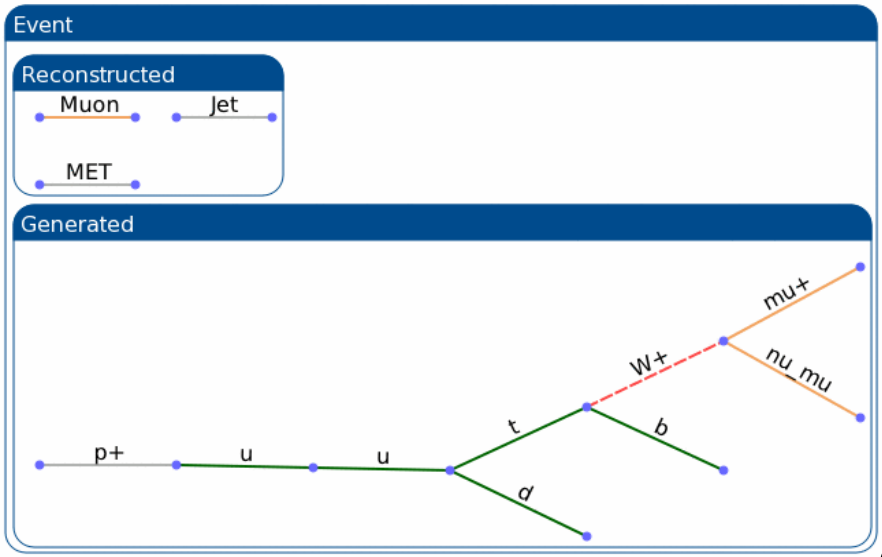
File Edit View Navigate Event Help

tChannel.pxlio

Tree View

- Event
  - Reconstructed
    - Muon
    - Jet
    - MET
  - Generated
    - p+
    - u
    - u
    - t
    - W+
    - mu+
    - nu_mu
    - b
    - d

Line Decay View

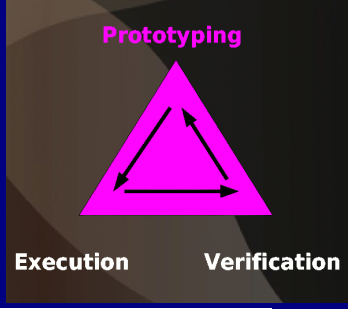


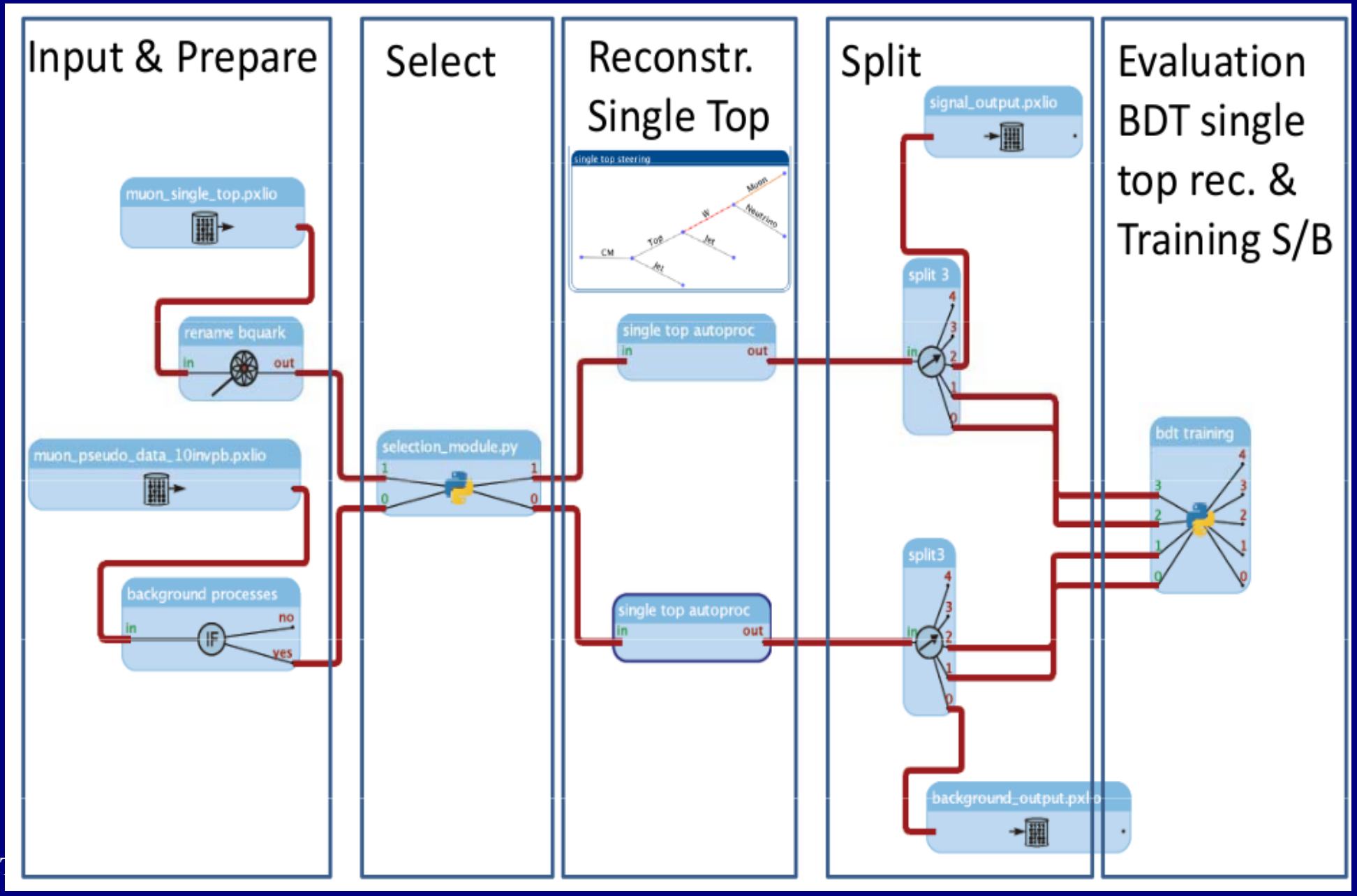
Property	Value
Object info	
Type	Event
Id	cb3d4c6841e7
UserRecord	
Event Number	22
Process	gleTop_tChannel

Reconstructed and generated levels

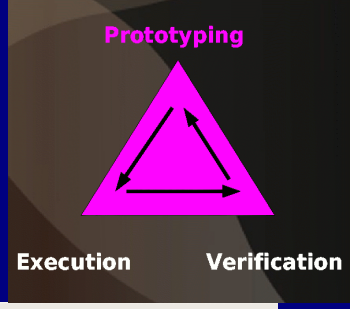
Properties

Single Top Modular Analysis
BDT Training: Single Top vs Background





Use GUI to design analysis



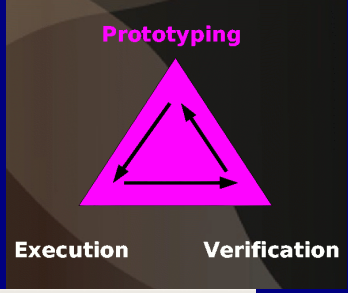
Input file

Choose module

Insert and connect modules

Configure module

Create Python Analysis Module



File Edit Analysis Designer View Help

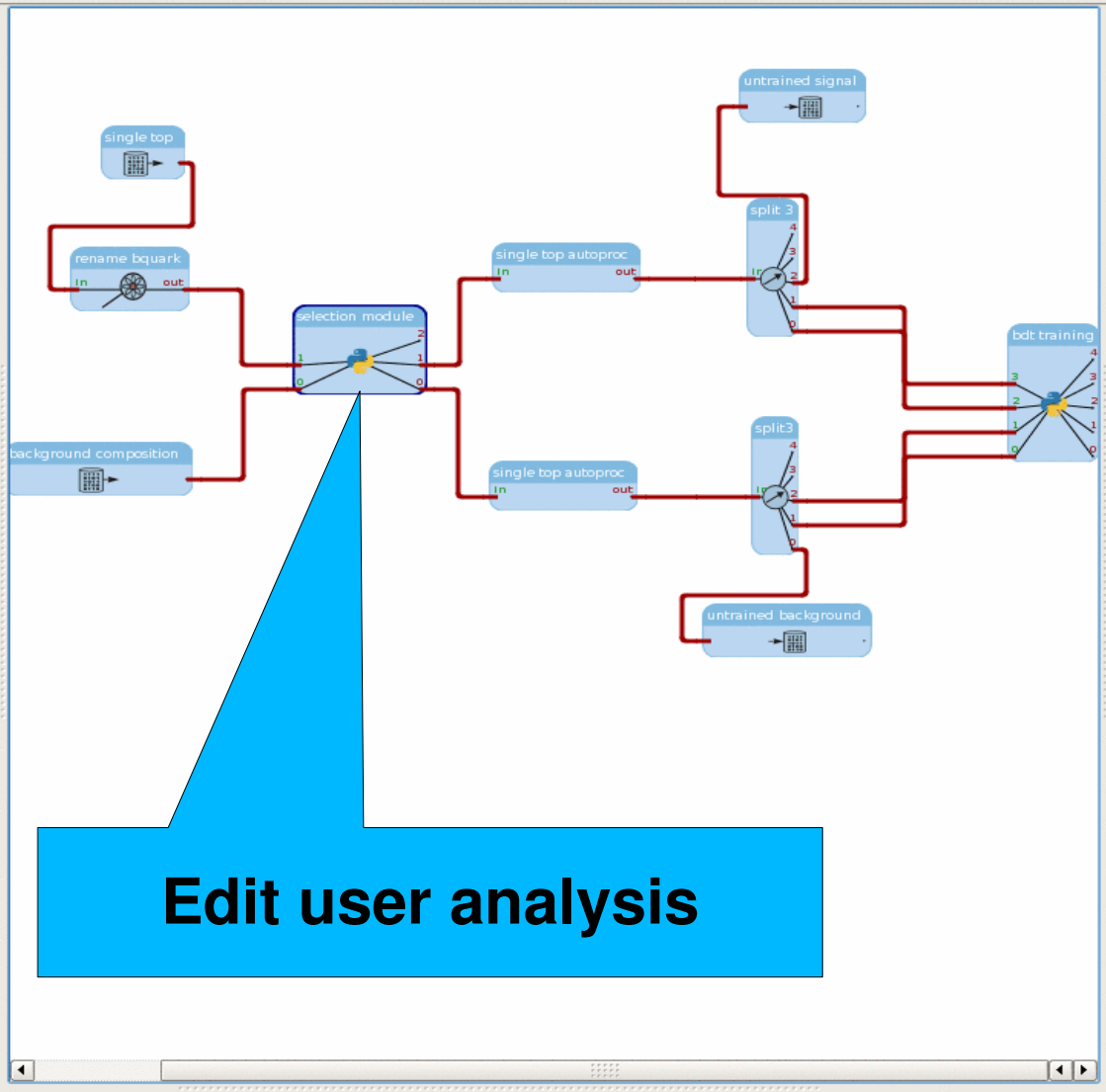
*bdt_training_single_....xml

Available Modules

- PySwitch
- PyModule
- PyGenerator
- PyDecide
- PyAnalyse
- File Output
- File Input
- AutoProcess

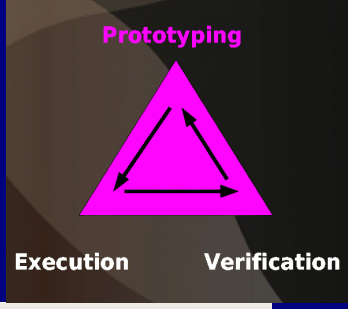
Python Scripts

- analysis
- Andreas
- archive
- backup
- bin
- boss
- btag
- CMS_literature
- Desktop
- f2projekt
- fireworks
- for_Andreas
- for_naf
- forDennis
- formulars
- ILC_old
- ILC_samples
- Lehre
- literature
- local
- Mail
- mail



Property	Value
Main	
Name	selection module
Type	PyModule
Options	
filename	/selection_module.py
script	
parameter	
sinks	2
sources	3

Create Python Analysis Module



File Edit Analysis Designer View Help

*bdt_training_single_....xml

Available Modules
- PySwitch
- PyModule
- PyGenerator
- PyDecide
- PyAnalyse
- File Output
- File Input
- AutoProcess

Python Scripts
- analysis
- Andreas
- archive
- backup
- bin
- boss
- btag
- CMS_literature
- Desktop
- f2projekt
- fireworks
- for_Andreas
- for_naf
- forDennis
- formulars
- ILC_old
- ILC_samples
- Lehre
- literature
- local
- Mail
- mail

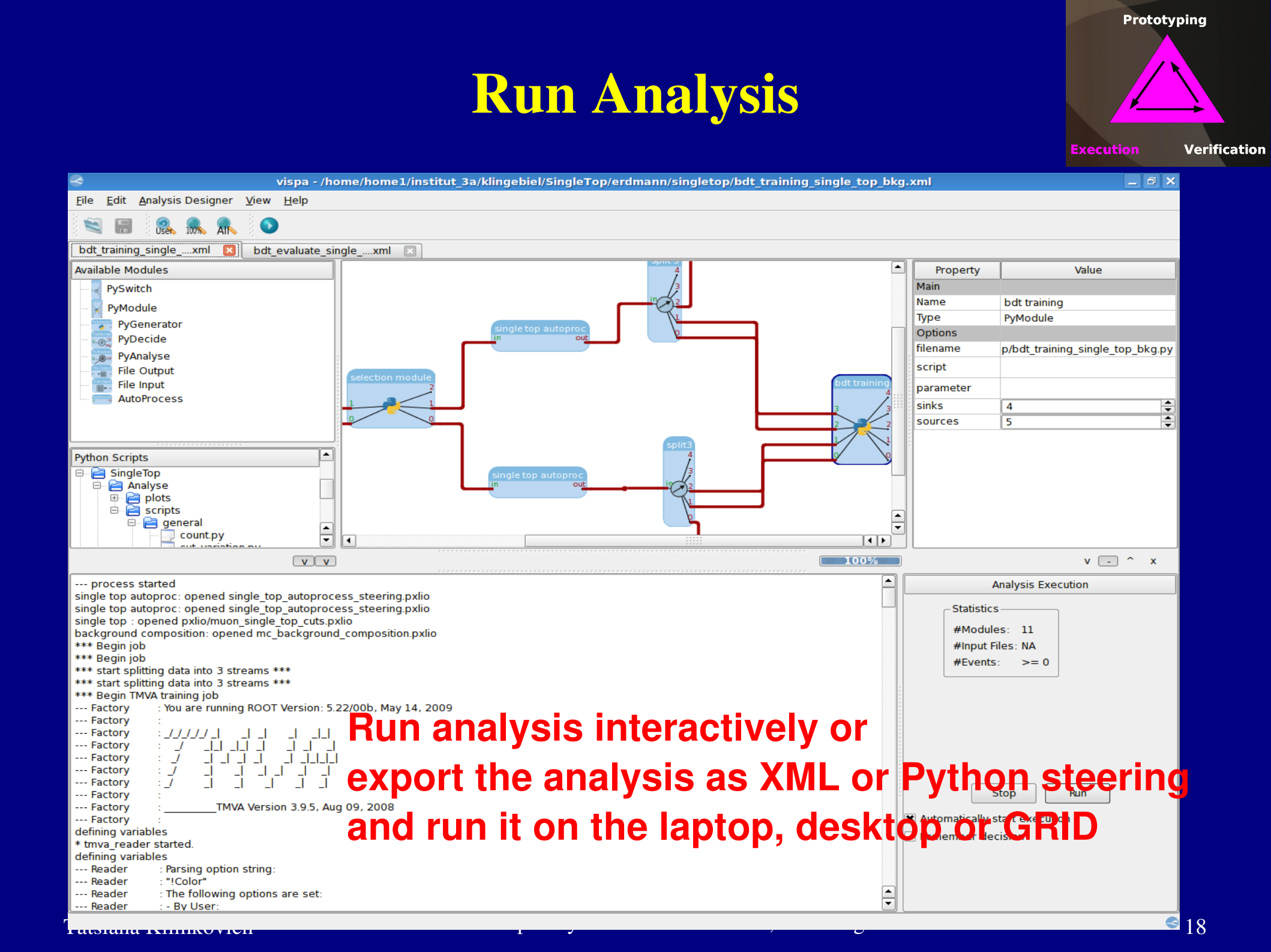
single top
rename bquark
background composition
selection module
single top autoproc
single top autoproc
untrained signal
split 3
split3
bdt training
untrained background

Property	Value
Main	
Name	selection module
Type	PyModule
Options	
filename	/selection_module.py
script	
parameter	
sinks	2
sources	3

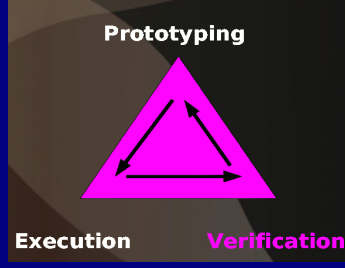
```
for particle in eventview.getParticles():
    if (particle.getName() == "Jet" and particle.getPt() > 30)
        selected.setObject(particle)
```

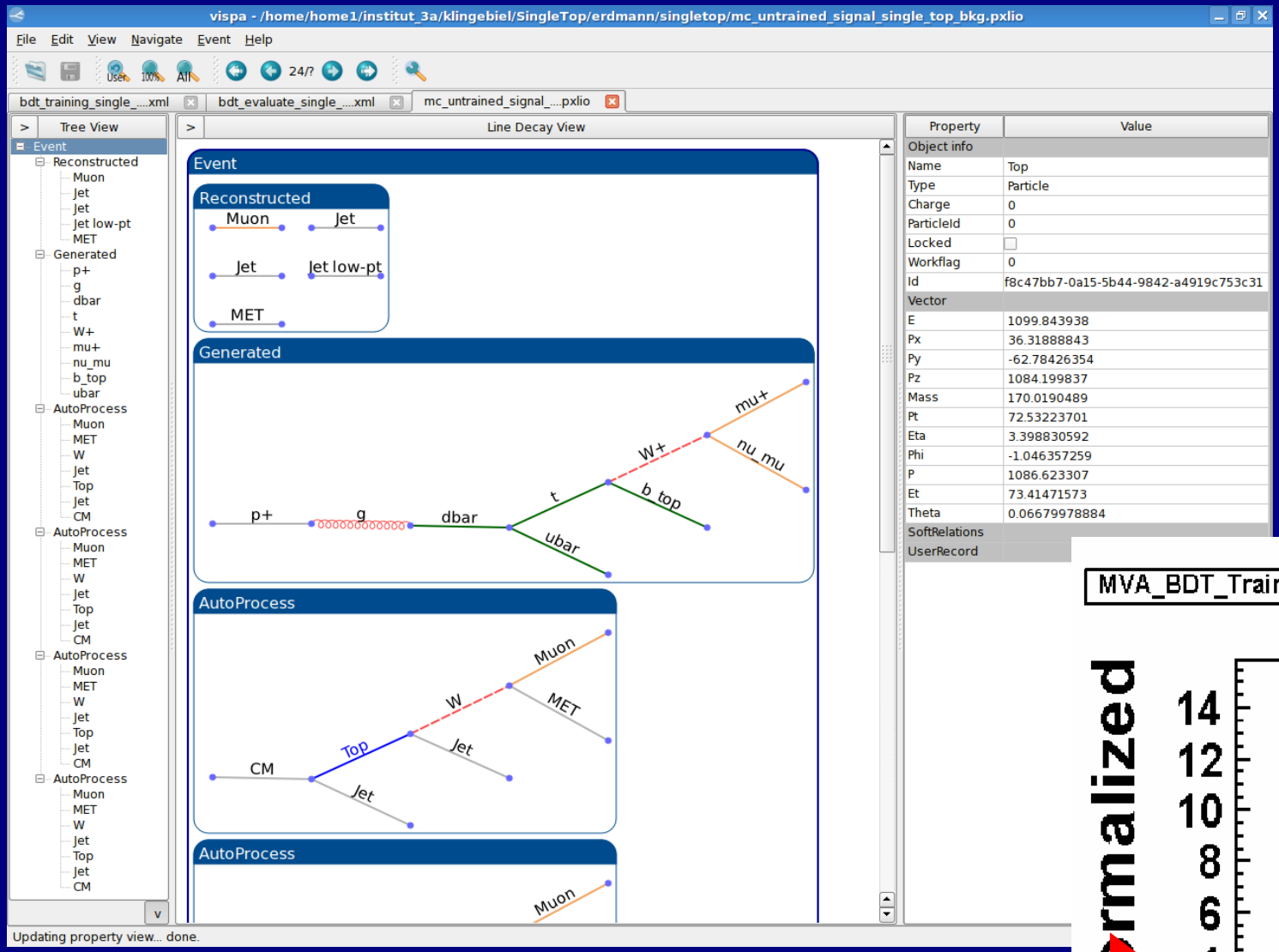
Rapid prototyping of the analysis

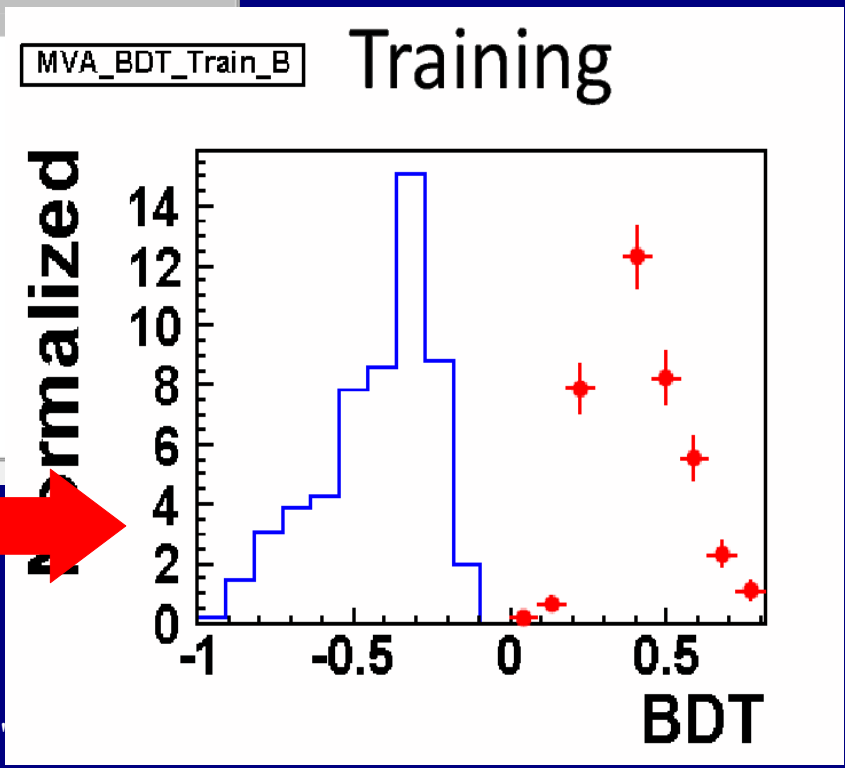
Run Analysis



Verify Analysis Output



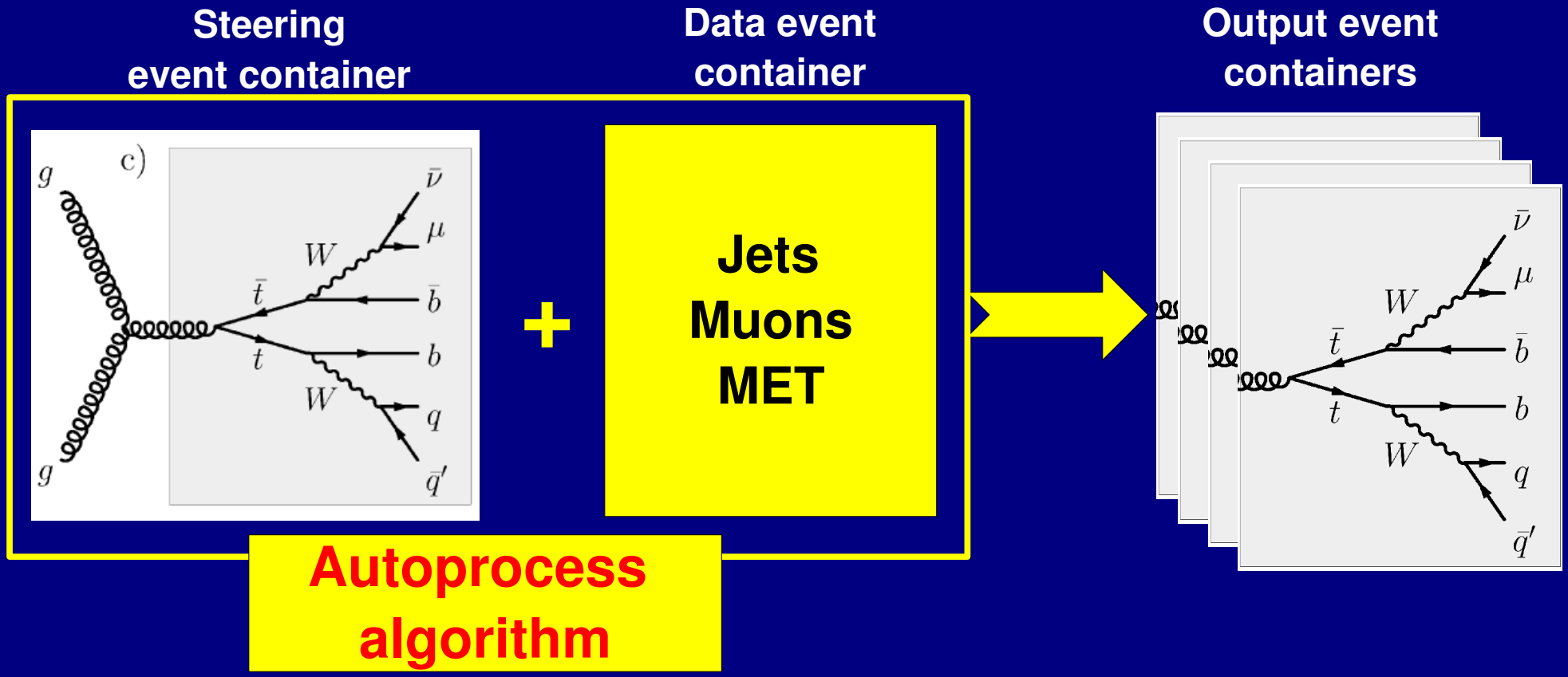




Create histogram using PyROOT

If needed repeat prototyping, execution, verification

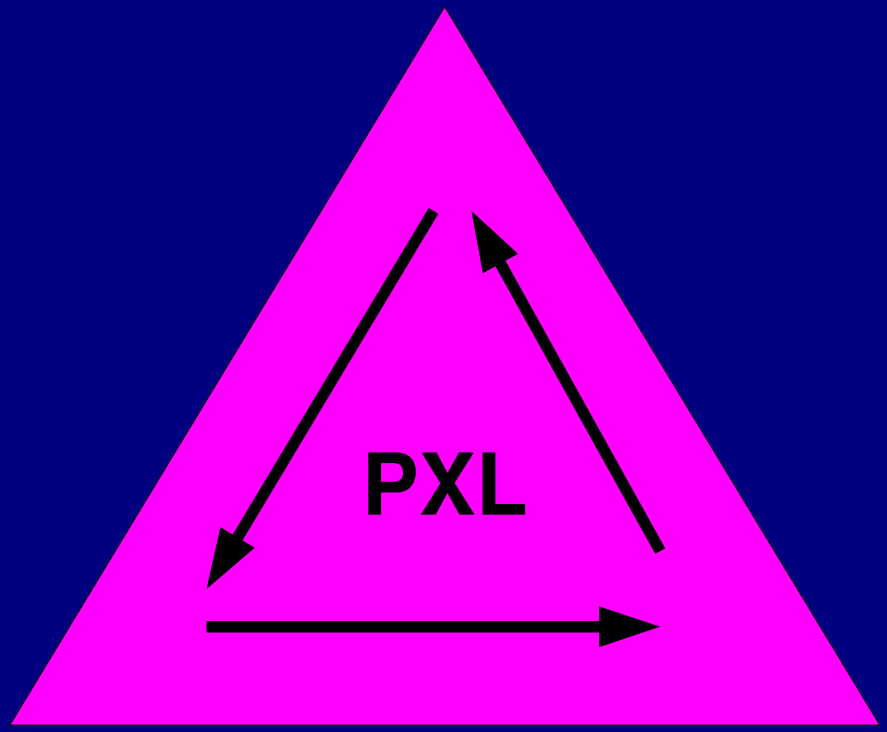
VISPA Module: Autoprocess



- In various physics analyses (Top, Higgs, SUSY) a reconstruction of the whole decay chain is needed
- Several possible configurations need to be built
- **Autoprocess** is a module for automated reconstruction of particle cascades

Summary: Analysis flow with VISPA

Prototyping: Development Environment
Analysis modules: Python or C++



Execution
XML or Python steering
interactive or batch

Verification
Event Browser
ROOT histograms

Summary

- **VISPA is a development environment for high energy physics analyses**
  - **Combines visual and textual programming**
  - **Allows fast prototyping, execution, and verification of an analysis**
  - **For application in any HEP and astroparticle experiment**
  - **First applications for CMS analyses**
  - **First steps for applications in ILC analyses**

Summary II

- **All software is continuously maintained and fully documented:**

  http://vispa.sourceforge.net

  http://pxl.sourceforge.net

- **Installers for MS Windows, MAC OS X, Debian and Ubuntu Linux are provided**

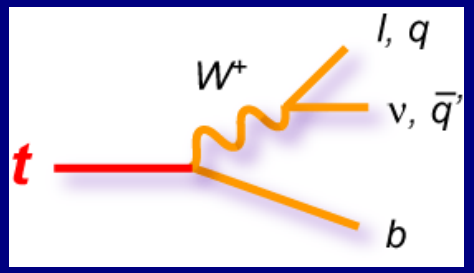
- **Publications:**

  http://arxiv.org/abs/0810.3609

  http://arxiv.org/abs/0801.1302v2

Backup slides

PXL key components: Event Container



Event container *pxl::Event*

- Particles (*pxl::Particle*)
- Vertices (*pxl::Vertex*)
- Collisions (*pxl::Collision*)
- User data (*pxl::UserRecord*)
- Their **relations** and **roles**

Physics objects

Event Interpretation

pxl::EventView

pxl::Event **represents an entire physics event**

pxl::Event **can hold several EventViews**

pxl::EventView **is a special view of this event**

Copies of these classes preserve all contained information (e.g. particle relations)

PXL key components: *pxl::UserRecord*

- All major PXL objects provide UserRecord for storage of arbitrary user data
- Deploys Copy-On-Write mechanism

PXL key components: I/O System *pxl::Serializable*

- Fast, flexible
- Small file size (use ZLIB library for data compression)

Module Steering System

- **Data flow**

  ➔ each module has a number of sources and sinks

  ➔ interface between modules: PXL event container

- **Modules**

  ➔ plug-in mechanism

  ➔ interactive creation of PYTHON modules

- **XML configuration**

  ➔ exchange format

  ➔ save and restore any state of the analysis

- **PYTHON configuration**

  ➔ high flexibility

  ➔ easy-to-read

Interface: Event Container