Journal of Physics: Conference Series **219** (2010) 042008 doi:10.1088/1742-6596/219/4/042008

Visualization of the CMS Python Configuration System

M.Erdmann[1], R.Fischer[1], B.Hegner[2], A.Hinzmann[1*], T.Klimkovich[1], G.Müller[1], J.Steggemann[1]

[1]RWTH Aachen University, Physikalisches Institut 3A, 52062 Aachen, Germany
[2]CERN, CH-1211 Geneva 23, Switzerland
[*]Contact presenter andreas.hinzmann@cern.ch

Abstract. The job configuration system of the CMS experiment is based on the Python programming language. Software modules and their order of execution are both represented by Python objects. In order to investigate and verify configuration parameters and dependencies naturally appearing in modular software, CMS employs a graphical tool. This tool visualizes the configuration objects, their dependencies, and the information flow. Furthermore it can be used for documentation purposes. The underlying software concepts as well as the visualization are presented.

1. Introduction

The software framework of the CMS experiment is steered using a configuration system based on the Python [1] programming language. Due to the high number of the order of 6000 configuration files, visualization tools are essential. In this document the Configuration Browser, the visual tool employed by the CMS experiment for browsing the configuration system, is introduced.

2. The CMS software framework and the CMS Python configuration system

The CMS experiment uses a common software framework for simulation, high level trigger, reconstruction and analysis tasks. The CMS software CMSSW [2] is designed as a modular framework. All algorithms are encapsulated in modules with a common interface. Each module can read and write data to a data container, which represents a high energy physics event. Unidirectional communication between modules is done via the data container. The data container can be made persistent, allowing to save output from any executed module.

For the steering of the framework a Python configuration system [2, 3] is used. A processing job is executed using a Python configuration file as input. In the configuration file the parameters of the modules are defined. Also the setup, such as conditions and geometry data, is defined. The modules are organized in sequences which determine their order of execution. In the following example a sequence for the selection of reconstructed muons is given:

```
import FWCore.ParameterSet.Config as cms
from PhysicsTools.PatAlgos.producers.muonProducer_cfi import *
from PhysicsTools.PatAlgos.selection.muonSelector_cfi import *
from PhysicsTools.PatAlgos.selection.muonCountFilter_cff import *
Muons = cms.Sequence(allMuons * selectedMuons * countMuons)
```

Here, the sequence defines the execution of three modules *allMuons*, *selectedMuons* and *countMuons*. The modules are defined in separate files which are included using Python import statements. In this manner the configuration is split over several files. In principle, the configuration contains all the information on the processing flow except for the algorithms encapsulated in the modules.

3. The Configuration Browser

CMS has developed a visualization tool for browsing configuration files: the Configuration Browser [4] It allows users to inspect, debug and verify configuration files interactively. The Configuration Browser is designed to help users find or edit certain modules and their parameters in the configuration system consisting of a huge number of files. Further, it is designed to help users understand the dependencies between modules. In the configuration system, these are indicated only unidirectionally, whereas the Configuration Browser visualizes all dependencies. The parameters of a module indicate from which other modules input is used. Furthermore, the Configuration Browser identifies which other modules depend on the output of a certain module.

The development of the Configuration Browser focuses on the following features:

- Provide a quick and detailed view into any configuration file used in simulation, high level trigger, reconstruction and analysis;
- Allow debugging and verification of configuration files;
- Allow documentation of processing flows.

4. Functionality

4.1. Loading configuration files

The Configuration Browser can load any configuration file used in the CMS Python configuration system. When a configuration file is loaded, all the files imported by the configuration file are imported automatically. The complete configuration including all imported sequences and modules is then displayed in the browser.

4.2. Browsing

In Figure 1 a screenshot of the Configuration Browser is shown. The window is divided in three parts: the tree view, the center view and the property view (Figure 1a, b, c). The tree view displays the entire content of the configuration file organized by sequences. The center view shows a graphical representation of the configuration. The property view lists all properties of a selected sequence or module. This includes general information such as the name and the type of the selected sequence or module as well as all its parameters. For navigating through configuration files, the three views are connected. The selected sequence or module in the tree view is displayed in the center view. The module selected in the center view is displayed in the property view. To allow a quick access to any sequences or modules, the browser includes a search function to find sequences or modules by their label or certain properties.

4.3. Locating and editing configuration

When the configuration file is loaded, each sequence or module is associated with the file from where it was loaded. In this way the browser can provide the file name with the definition of a sequence or module in the property view (see Figure 1c). For quick editing of the configuration files, the user can select his favorite editor and edit any sequence or module selected in the property view via a shortcut.

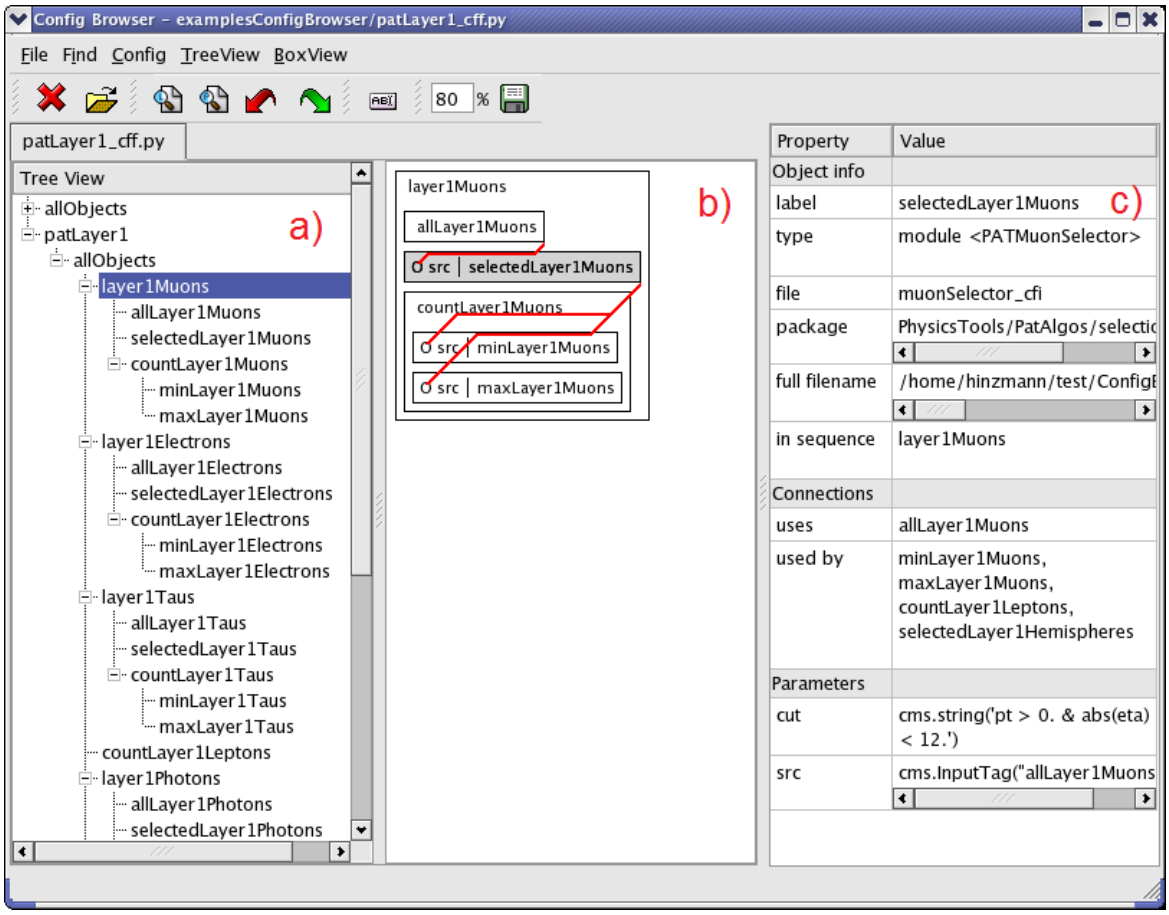

Figure 1. Screenshot of the Configuration Browser window: a) tree view, b) center view, c) property view.

4.4. Visualization of sequence structure and information flow

The Configuration Browser offers two options for the graphical representation of the configuration in the center view: the sequence structure view and the connection structure view. In both views sequences and modules are represented by boxes. The content of the boxes can be chosen by the user. By default the name of the sequence or module is displayed. In addition, for example, the filename of the file where the sequence or module was defined can be displayed.

In the sequence structure view the modules are organized by sequences, similarly to the tree view. All levels of nesting of sequences are displayed as a block diagram. This view gives the user an overview over the processing flow in a configuration file. In Figure 2 the representation of the following example is displayed:

```
countMuons = cms.Sequence(minMuons * maxMuons)
Muons = cms.Sequence(allMuons * selectedMuons * countMuons)
```

As it is seen from the example, the sequence structure is contained directly in the Python configuration and does not need advanced processing.

As mentioned in Section 3 the information flow between modules is contained in the configuration only unidirectionally. In the parameters of a module, input from another module is indicated by the name of the module whose output is used. In order to visualize the

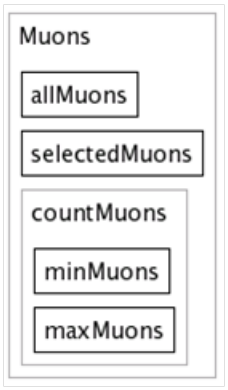

Figure 2. Sequence structure view.

Figure 3. Connection structure view.

dependencies between modules these names are mapped to the modules themselves. Also the entire configuration is scanned in order to calculate for each module which other modules use its output. The two dependencies "uses input from" and "output used by" are listed in the property view (see Figure 1c). In the connection structure view these dependencies are visualized by lines connecting the modules. In Figure 3 the connection structure view for the above-mentioned example is displayed. The modules are arranged according to the information flow between them. A more complex example is shown in Figure 4.

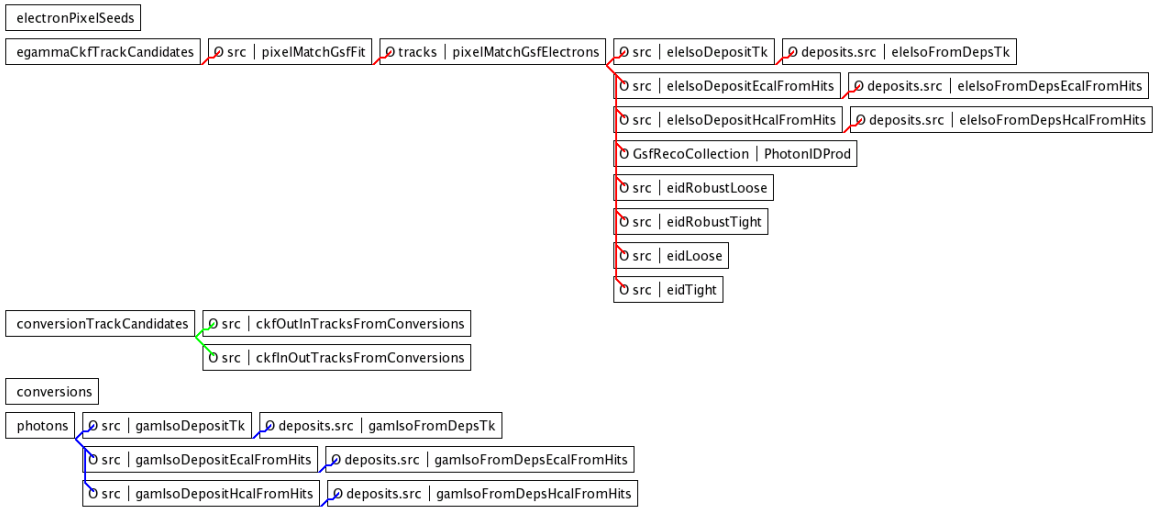

Figure 4. Connection structure view of the Electron and Photon reconstruction in CMS.

4.5. Documentation

For documentation purposes the browser allows to save images of the graphical representation displayed in the center view. Common image formats such as PNG (Portable network graphics) and BMP (Bitmap) are supported.

5. Implementation

5.1. Interface to the CMS Python configuration system

The Configuration Browser is implemented in Python to allow direct access to the information contained in the configuration files. Configuration files are loaded into the browser using Python "import". The configuration file is thus interpreted by Python rather than parsed. This has the advantage that the information read by the browser is exactly the same as the information used when the configuration file is run within the framework. For example, parameters which are modified more than once in the configuration file are seen with their latest and correct values. Also all the files imported by the configuration file are imported automatically. The analysis of the content of configuration files is performed using introspection provided by the Python programming language.

5.2. Framework of the graphical user interface

The Configuration Browser is a plug-in of an underlying software framework for the graphical user interface (GUI). This framework is part of the Vispa project [5], a framework for visual physics analysis. The GUI framework is designed for maximum reusability in applications for graphical browsing and editing. It is written in the Python programming language for fast development turnaround due to dynamic typing and automatic memory management. It is based on Qt [6], a widely spread cross-platform application and user interface development framework.

New browsers and editors can be added to the main application using a plug-in mechanism. An example for a further plug-in is the Edm Browser [7] - an event browser which allows to inspect the content of CMS data files, event by event, using introspection. The plug-ins share the same main application as well as graphical compounds such as the tree view, center view and properties view, etc. The degree of reusability is indicated by the lines of code of the two browsers. Both browsers share about 2000 lines of code of the GUI framework and have about 1000 lines each.

6. Outlook

The Configuration Browser was introduced in the CMS experiment in August 2008. Since then it has been intensively used by the people of the CMS collaboration. Recently, the GUI framework has undergone a transition from version 3 to 4 of the Qt development framework to include more advanced user interface components. The Configuration Browser itself has been extended to an editor. Users can now modify parameters in the browser and save their changes to a new configuration file, importing the original file and applying all changes.

7. Conclusions

Visualization is essential when dealing with huge configuration systems as in the CMS experiment. The Configuration Browser is a tool for inspecting, debugging and verifying CMS configuration files interactively. It has the following key features:

- Direct access to the information in configuration files due to its implementation in Python;
- Interactive browsing, editing and documentation of configuration files;
- Visualization of the structure and information flow in configuration files;
- A GUI framework designed for maximum reusability.

The configuration browser has become an essential tool within the CMS collaboration that is continuously maintained and extended.

References

[1] http://www.python.org.
[2] CMS Collaboration 2006 *CMS Physics Technical Design Report, Volume I* CERN-LHCC-2006-001 pp 31.
[3] R. Wilkinson, B. Hegner, C. D. Jones on behalf of the CMS Offline & Computing Project 2009 Using Python For Job Configuration in CMS *Proc. 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009), Prague, Czech Republic.*
[4] http://twiki.cern.ch/twiki/bin/view/CMS/SWGuideConfigBrowser.
[5] O.Actis et al. 2008 Visual Physics Analysis (VISPA): Concepts and First Applications *Proc. 34th Int. Conf. High Energy Physics (ICHEP 2008), Philadelphia, Pennsylvania* arXiv:0810.3609 http://vispa.sourceforge.net.
[6] http://www.qtsoftware.com.
[7] http://twiki.cern.ch/twiki/bin/view/CMS/SWGuideEdmBrowser.