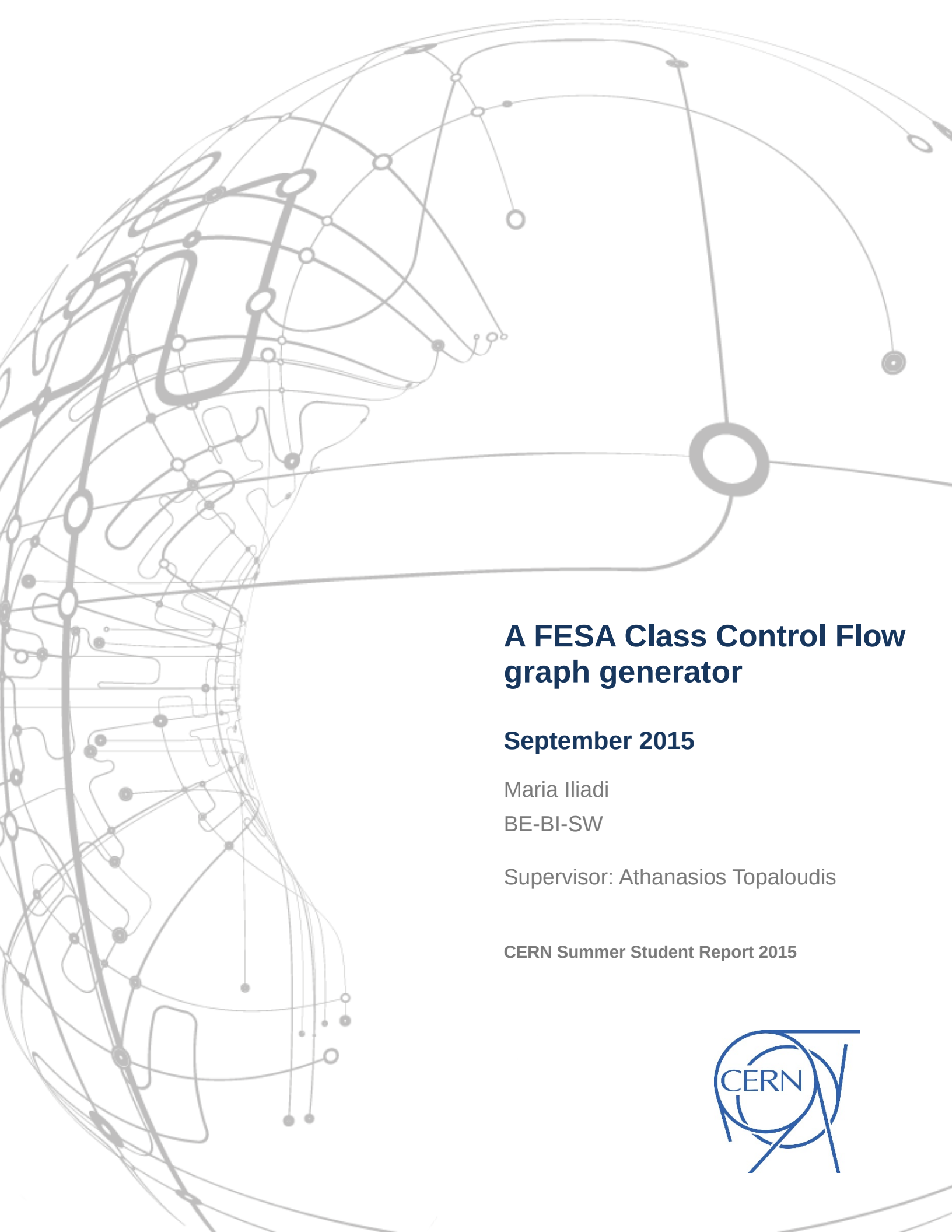


A FESA Class Control Flow graph generator

September 2015

Maria Iliadi

BE-BI-SW

Supervisor: Athanasios Topaloudis

CERN Summer Student Report 2015

Acknowledgements

I would like to thank the Athens University of Economics and Business for providing me with the incredible opportunity to participate in CERN's summer student program. I am also grateful to the entire BE-BI-SW section for their warm welcome and constant support throughout the term. In my section I had the pleasure to experience what a team truly is, since all my colleagues were volunteering their expertise and helped each other. I am particularly obliged to Athanasios Topaloudis for his help, guidance and supervision. Athanasios not only contributed to my self-awareness of the problematic at cause in my projects but also stood as an example of organization, problem facing and time / space management.

Abstract

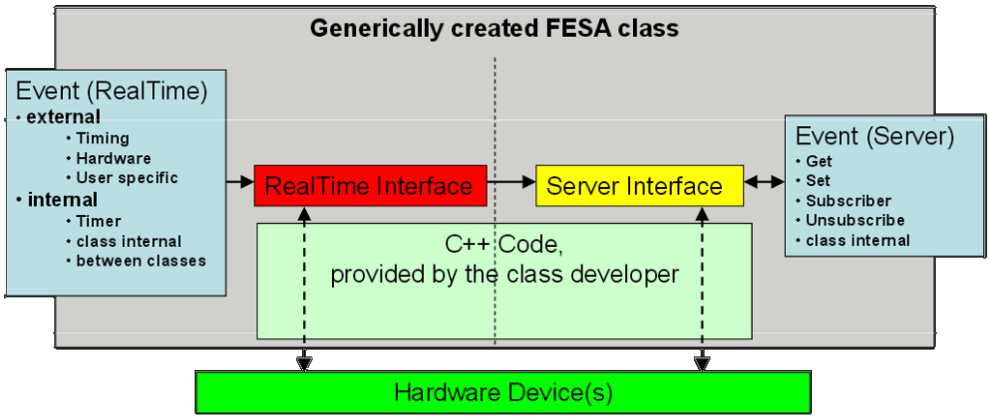
This report documents the work that was done during a summer student internship in the CERN BE-BI-SW group in the summer of 2015. The project proposal was to improve an existing tool for generating flowcharts from the design of a class and then create a GUI for the tool. The end result of the project is the improvement of the tool, so that the developer can have an overall image of the class's design. Also, the GUI is functional at its current state and it can be extended with further work in order to be more user-friendly and offer more options to the user.

Table of Contents

1 Introduction

This document is a report for the project “FESA Class Control Flow Graph Generator”. The project is an attempt to improve the existing tool for generating flowcharts from the design of FESA classes and create a GUI (Graphical User Interface) so that the programmers can generate the flowcharts easily, without specific knowledge of the capabilities of the tool.

Equipment specialists at CERN use the Front-End Software Architecture Framework, known as FESA. That is a complete environment for designing, developing, testing and deploying real-time SW for the control of the instruments. The framework is useful for designing RT SW with a standard structure and generate C++ code easily, with all the constrains needed.



In FESA there are have two main parts: the Real-Time Interface and the Server Interface. The Real-Time Interface is the core interface and is driven by internal and/or external triggers organized in events. This interface is constantly running and is responsible for the communication with the hardware. The Server Interface is organized in Properties and it is the actual interface of the SW with the outside world (other SW or humans). The communication of these two parts is achieved via a shared memory.

The design of a FESA class is in XML format. In addition to the xml view, there is a graphical representation of a FESA class from FESA Graph. This tool is a class control flow graph generator, written in Python from B.Bielawski (BE-CO-FE). It allows the declarative specification and drawing of a FESA class’s design.

Creating a diagram for the representation of a FESA class offers numerous of advantages:

- Easing the documentation of the SW.
- Easier identification of errors or possible improvements in the SW design. Distinguish differences between the model that we have in mind and the actual implementation.

- A nice overall global view of the SW.
- Use macro processors for generating different diagrams (ex. Add extra items, increase spacing between nodes, choose the output format).

The aim of this project is to improve the existing tool by changing the process of generating flowcharts, retain only the useful information and remove the trivial attributes of the diagram. Then, the developer will be able to have an overall image of the class's design with a quick look. Another aim is to create a user-friendly GUI for this tool, so as to be flexible enough to use it, without knowing how to use the FESA Graph.

2 FESA Class Control Flow Graph generator

FESA Graph is a class control flow graph generator. It allows the declarative specification and drawing of a FESA class's design. The specification is done in a DOT file from the .design file of the class. Then, using Graphviz, which is an open source graph visualization software, the tool transforms the DOT file into a flowchart with a graphical representation in PNG or PDF format.

Output of using FESA Graph:

- className-Version.png or .pdf – file with the graph,
- className-Version.dot – file with the description of the graph in dot language. This file is generated from the Python script.
  (both files are stored in the current directory)

2.1 Background Research

The project came about in the response to the need of having a flowchart that represents the implementation of a FESA class. First, I had to study the process of designing SW using FESA and understand which the important constraints of the classes are. I also had to study about the DOT language and the Graphviz documentation, so that I can understand the existing Python script that produces the flowchart in order to be able to improve it.

2.2 Definition and outputs of the previous and the new FESA Graph

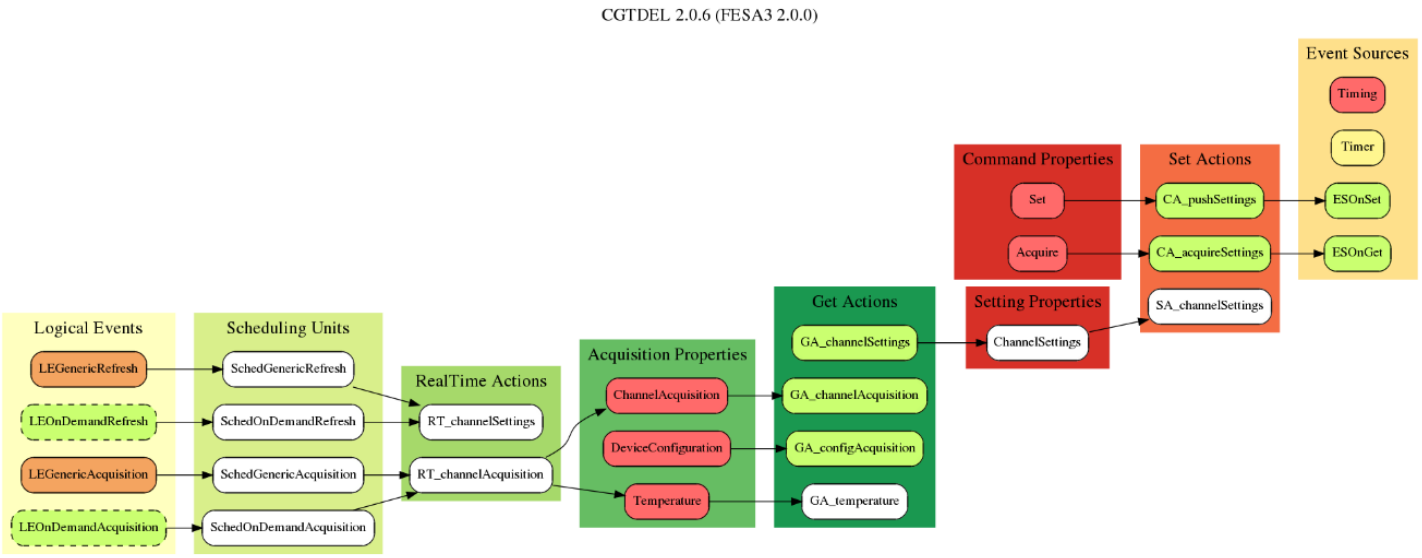


Figure 1. Flowchart of a FESA class from the previous tool.

In Fig.1 there is a simple example of a flowchart from the previous tool. With this flow, the Logical Events are attached to Scheduling Units. A Scheduling Unit triggers a Real-Time Action, which notifies an Acquisition Property. A Setting Property triggers an Event Source and is notified by an Acquisition Property. This example is not complicated, but as FESA classes grow in size and complexity, it becomes difficult to read and understand the control flow.

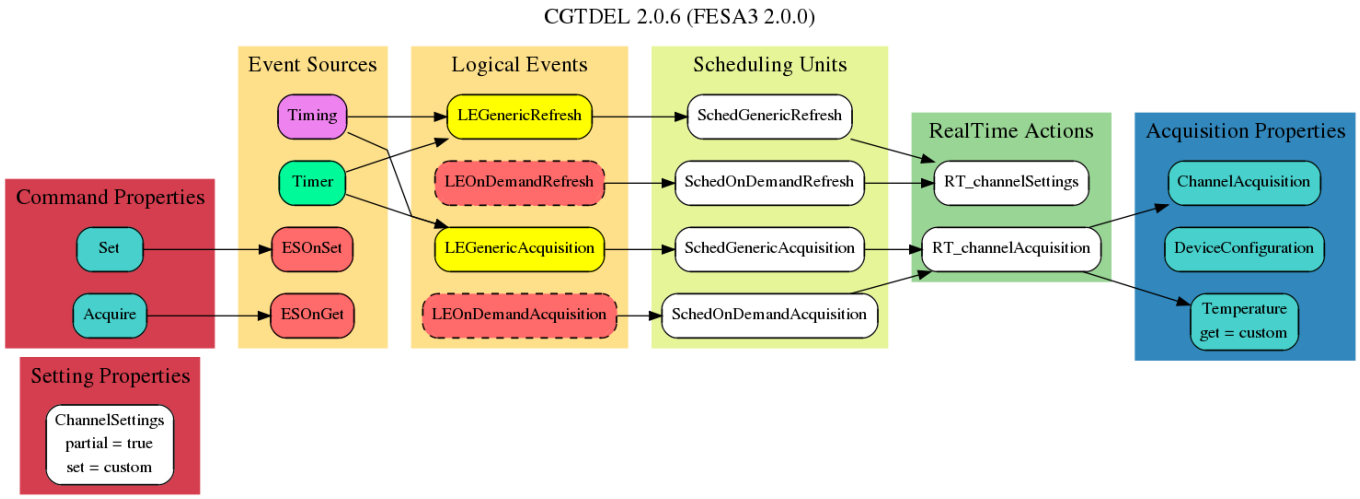


Figure 2. Flowchart of a FESA class from the new tool.

In Fig. 2 there is the same class's flowchart from Fig.1, generated from the new FESA Graph. The changes that were applied are:

- Flow of the node sets: the new flow is more logical, since the Setting/Command Properties trigger the Event Sources, which are connected to the Logical Events. These Events are attached to Scheduling Units, which trigger Real-Time Actions. Finally, the RTA notify the Acquisition Properties.
  Also, the Real-Time Interface in the middle (from Event Sources to Real-Time Actions) and the Properties on the right and the left, because it's more logical to have the core process in the middle and the actual interface at the edges. A typical work flow starts with the user setting the settings (left side), the core picking up the new settings (middle) and publishing the acquired data (right side).
- Removed the Set and Get Actions node sets: these node sets are not really important since they are compulsory and they follow the same pattern hence, they are no longer part of the flowchart in order to reduce the number of node sets and thus simplify the flowchart. The attributes of the Server actions are the important part and that is why these attributes were added to the according properties.
- Colors: changed the colors of the node sets and the nodes. Each node that has a special color represents an information (see the LEGEND of FESA Graph in Fig.3).

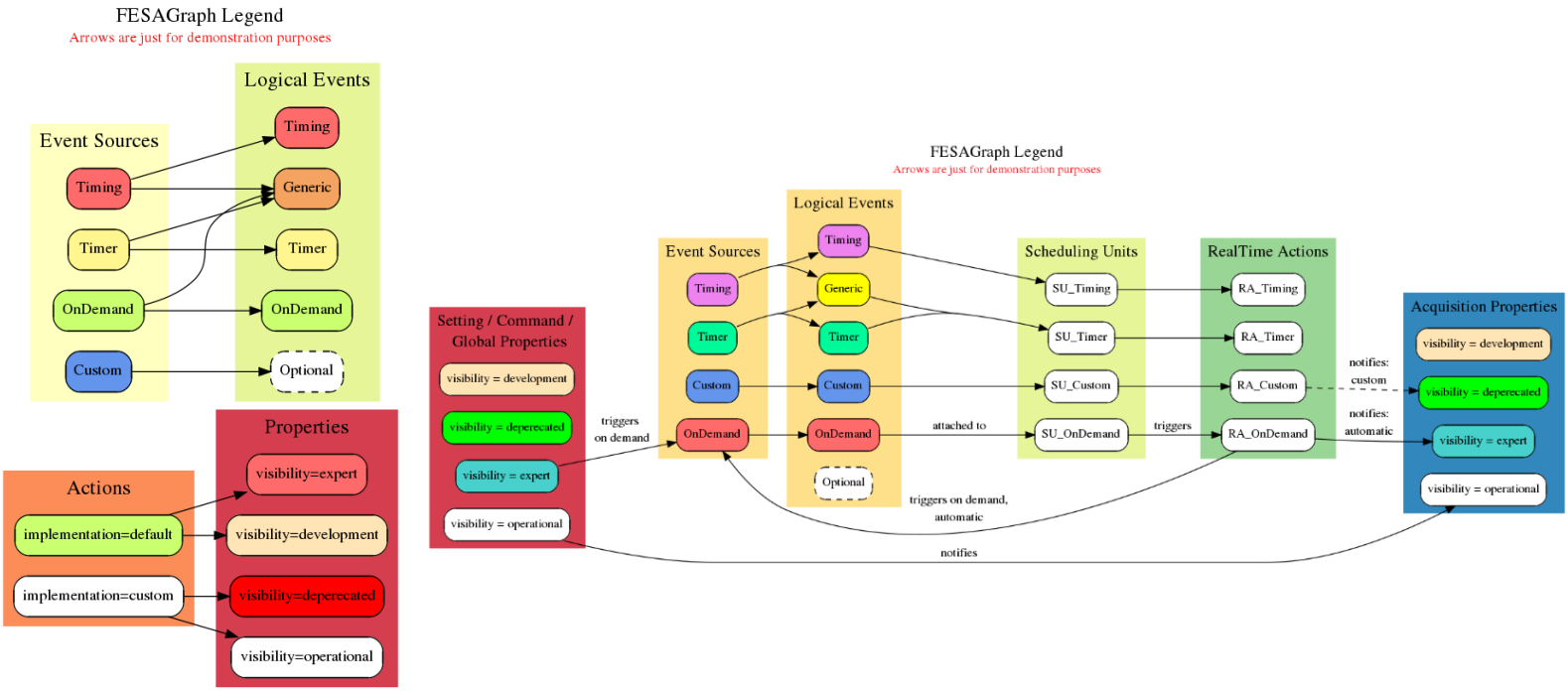


Figure 3. The previous (left) and the new (right) legend of FESA Graph

In Fig.3 there is the previous legend for FESA Graph. It's easy to read but we can't see the connection between the node sets and some attributes are missing that the developer needs to know. With the new legend of the classes the user can see the connection between all the node sets through the colors and the arrows. There are also some new connections (Scheduling Units with Event Sources) and descriptions on the arrows.

2.3 Results

What has been achieved so far:

- There are less node sets, so the flowchart is clearer and easier to read.
- The control flow is more logical
- The trivial attributes were removed and added new important information inside the nodes and through new connections.

2.4 Future work for FESA Graph

The current FESA Graph is working properly. Although, there are still some improvements and new options that can be implemented in the future:

- Possibility to remove or change the color of the nodes and add new connections
- Possibility to change the font or the shape of the nodes
- Improve the arrow connections for a clearer result
- Use the .deploy file of the class instead of the .design file to include more information
- Improve the result in svg format.

3 FESA Graph GUI

Since the improvement of the FESA Graph, the next step was the development of a GUI (Graphical User Interface) in Python, using PyQt. The aim was for the developers to be able to generate the flowcharts easily, without specific knowledge of the capabilities of the tool. Also, a GUI is always a more attractive way to use a tool, and the user can easily generate his flowchart, check the result and apply some changes, such as expand the diagram, add new items and choose the format.

3.1 Background Research

One thing most people need at some point when writing Python for desktop applications is a way to make GUIs. Qt is a GUI framework, available as free and open source software. PyQt is the Python wrapper for Qt. Our two main criteria for this project for GUI framework are:

- Easy to use, particularly for beginners
- Cross-Platform (Windows, Linux, Mac)

PyQt meets the first criteria pretty well, and it definitely meets the second criteria. Although there is a lot of power and depth in PyQt, it's easy to do simple things with it. In this respect, it follows the Python philosophy of making simple things simple and complex things possible. I had to study on how I can build and connect the GUI with the FESA Graph.

3.2 Design of the GUI and implementation

In order the GUI to be useful, the user should be able to choose a .design file from a directory then display the default flowchart and have the options that are available from the tool (ex. expand the diagram, choose a different format). He should also be able to display the output and the legend of the tool. Finally, he needs to handle different files in the GUI.

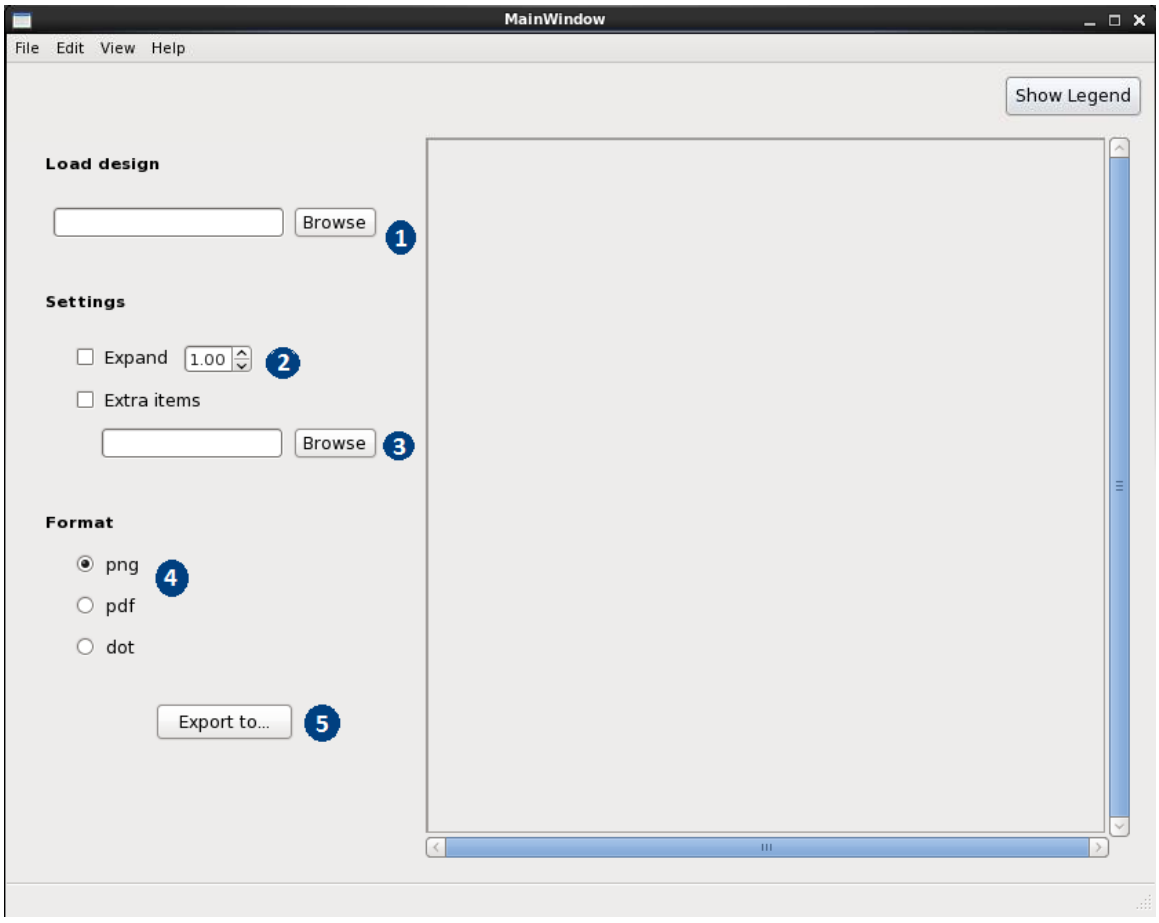


Figure 4. The main frame of the FESA Graph GUI

Basic process:

1. First the user needs to load a .design file. Using the “Browse” button in the “Load design”, he iterates through the directory and he opens a file. In the panel on the right of Fig. 4 it will appear the flowchart of the class with the default options, in .png format.
2. The default value of the expanding is 1.00. The user can handle the distance between your node sets using the spin box. If he unclicks the checkbox, the expanding returns to 1.00.
3. The user can also add a .design or a .dot file to provide extra items to his flowchart, using the “Extra Items” check box. If he unclicks the checkbox, the extra items will not be included in the flowchart.
4. In the “Format” group box he chooses the output of your flowchart. The default option is the .png format.
5. Finally, using the “Export to” button, the user can save his file to a directory. The .dot file and the .png file are automatically saved in the current directory.
6. In Fig. 5 is the output result.
7. If the user clicks on the “Show legend” button, it pops up another frame for a simultaneous view of the legend and a graph.

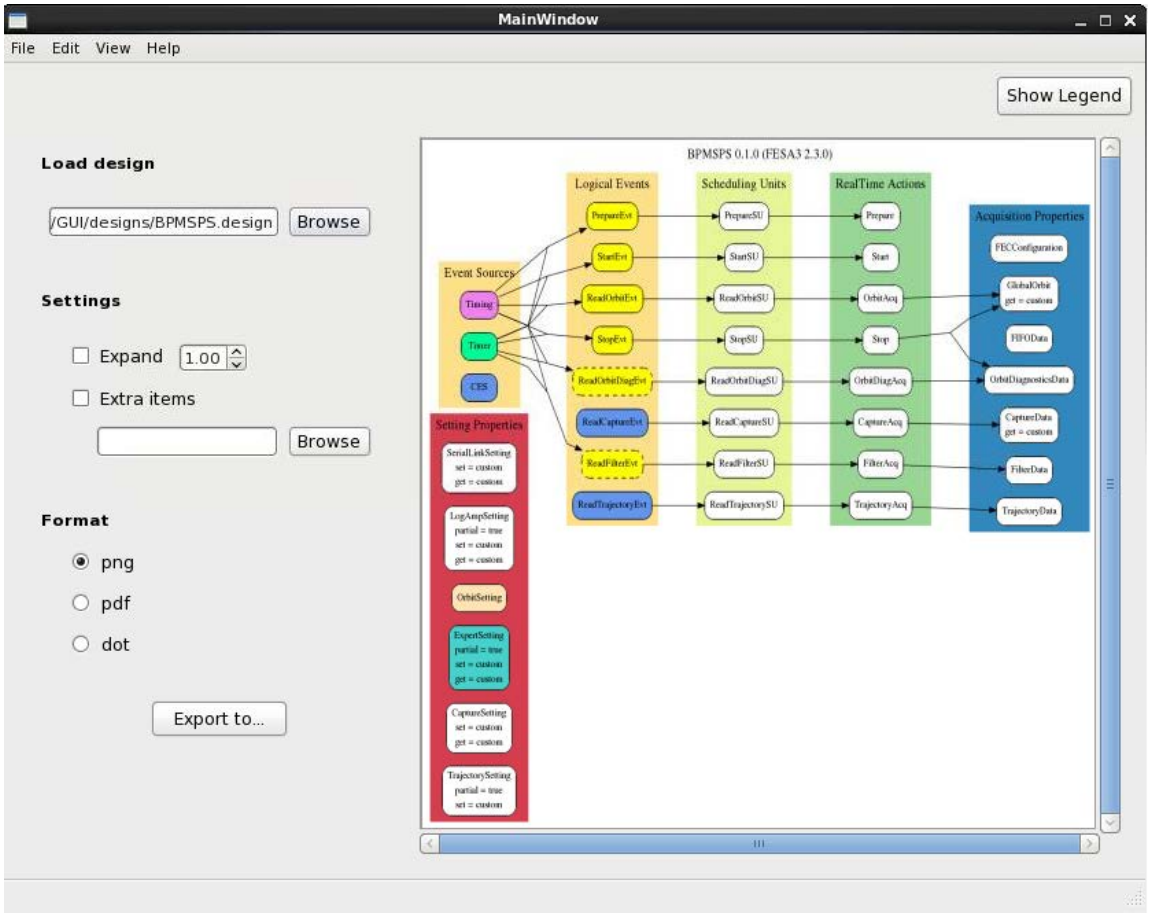


Figure 5. The output result of the GUI

3.3 Future work for FESA Graph GUI

There are some improvements that need to be done:

- The scroll area should be displayed only when the user needs to use it.
- Add the tab option so that the user can have more than one file at the same time.
- Fix the problem with the Zoom option.

4 Conclusion

In this report I presented the work done during my summer student internship at CERN. My aims were to improve the existing FESA Graph and then create a GUI for the tool, so as to be flexible enough to use it. The improvement of FESA Graph is pretty much done and functional in its current state, apart from a few possibilities. However, some features are currently missing from the GUI that the end users will require. I hope that this project will be extended with further work. Overall, the internship at CERN was an inspiring and special experience that provided a well-organized and fascinating task to work with throughout the summer.

5 References

- Athanasios Topaloudis, “Design and implementation of a client-server system for acquiring beam intensity data from high energy accelerators at CERN”, CERN Thesis, 2012
- CER, Beams Department – Beam Instrumentation – Software Section (BE-BI-SW), http://project-beam-instr-sw.web.cern.ch/project-beam-instr-sw/Welcome.php