TODO: insert a title here

Author:
Andrea G.B. DAMIOLI

Supervisor:
Dr. Germano BONOMI

Master Thesis

Abstract

Master degree in Computer Engineering

TODO: insert a title here

by Andrea G.B. DAMIOLI

The AEgIS Experiment at the CERN aims to verify the weak interaction principle for antimatter. This document talks about "gAnWeb", a web application designed to simplify the analysis of physical data under the AEgIS experiment. This analysis can be performed using Root Data Analysis Framework by the Linux Terminal using an application named "gAn", but a graphical interface can ensure a better user experience, ease the user training and improve the productivity. This document analyses from the Human Machine Interaction's point of view the production of this graphical interface.

Contents

Chapter 1

Introduction

1.1 User friendly Data analysis: gAn Web

GAn is a program that aims to analyse huge amount of data related to the AEgIS experiment at the CERN.

This program receive in input a folder containing several terabytes of files in ".root" format, and parameter named "run parameter", that identifies the information in which the users are interested. A file ".root" is a file produced by a variegated group of sensors in a complex machine that accelerates particles and lets them crash together. This sensors produce .root files continuously (8 hours per day). The time in this experiment is divided in "runs" (a run lasts about 140 seconds), so the user, by the run parameter can tell to gAn in which time slice he is interested.

The .root files can be analysed using a framework named ROOT Framework, that consists in a lot of libraries specialized in high-energy physics analysis, and an interpreter able to understand a C++ script. Actually, gAn is the sum of the Root Framework plus a lot of C++ scripts. The goal of gAn is reduce the huge amount of raw data in input in a little amount of scientifically interesting data in output. To do this it has to filter data, understand which of them are scientifically interesting, chose the parts that are related to the run selected by the user (by the run parameter), elaborate and compare them, and make advanced statistical analysis on them. gAn can be called using a common linux terminal, using a command with parameters.

The output of gAn consists of a single text file with computed, organized data, and a folder of images in png format. This structure (root files in input, data analysis using Root, images, organized and selected data in output) is very common in the CERN's experiments.

The output of gAn is quite understandable by an experienced physicist, but it is disorganized, complex for an untrained user, and and the terminal interface can be surely improved using some more user friendly technologies.

GAn Web is a web application, that aims to create a user friendly web interface, based on the most important human-machine interaction principles, between the users and gAn. A web interface can improve it in two ways:

1. gAn is a stand-alone program based on Root, installable on the user's machine; the user has to install the correct version of Root to avoid compatibility problems (Root is still not perfectly version independent: different versions can lead to different behaviours). Furthermore, this kind of program is continuously changing, the performed analysis is continuously improved, so the installed version of gAn is not final and unchangeable, and the user musts often update it. Instead, a centralized version installed on a server, with services accessible from a normal browser by the user can avoid (at least reduce) this kind of problems and be more usable.

2. a Linux terminal interface is practical for expert users, but a web based interface can be more attractive for new users, and, if well done, can be easier to use. It is important to notice that the users are physicists, not necessarily specialized in computer science, so, create a friendly and easily learnable interface can avoid them problems and time wasting.

The goal of gAn Web is to allow users to do analysis through a more friendly web interface, without install nothing on their machine. In the following image there is a schema that shows how this program is organized.

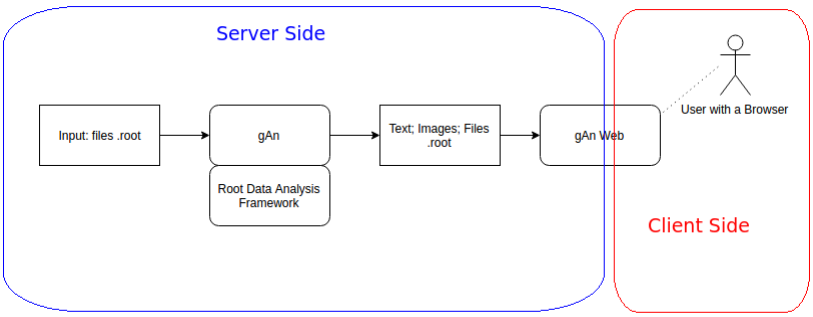

FIGURE 1.1: gAn - gAn Web simple scheme

Chapter 2

Requirements

First of all is important to analyse which are the requirements of gAn web (and how gAn web can improve the performances of gAn).

2.1 Functional requirements

The definition of functional requirements aims to specifying in detail what the web application can do, and in which way an user can use it. The development of gAn web is divided in two stages: an early stage, more simple, with basic functionalities, just to investigate what are the best ways to implement the functionalities and to test with a little group (2-3) of users if this software can really be useful and which functionalities are really important; A late stage, more complex, with some complex functionalities obtained listening the request of the first group of users; It is important to notice that the second stage (the late one) is a never-finished stage: the needs of the users are permanently changing and evolving, the application is designed to be adaptable, and to try to satisfy the unknown needs of future users. Following are exposed the requirements of the two versions:

2.1.1 Early version

The first version (from now "gAn web v1") is a very simple application: instead of access the program by a linux terminal like gAn,

in gAn web v1, the user can use the program through a graphical interface. The requirements of this version are the following:

1. The user, in the homepage, can chose the run (only one run, for the moment) in which he is interested, using a input field. This field has validator, able to understand if the run number is inserted, if it is effectively a number, and if it is in an acceptable range. The user receives an explaining and precise error message directly on the homepage if the input field is empty and if the inserted value is not acceptable. It is always possible validate the inserted run and ensure that the related Root file exists? No, because in some moments some sensors don't work properly and it is inevitable that some root files related to some runs are incomplete, or even inexistent.
2. The user can start the program with a single click, by a button (usable only if the inserted number is valid).
3. When the program is executed the user can see the text output on the screen. This text is clear for a physicist (it is not clear for a person who doesn't have a specific preparation).
4. When the program is executed the user can see the output images by clicking a button that link to a images-page. The images are ordered and organized by groups (the groups are related about which sensor takes the information necessary to create the image). The user can decide if he prefers to see the image in a little, medium or big format. The user can also decide if the images are distributed in the screen vertically or through a "carousel layout". The user can access the image in full-screen by clicking on it: he is redirected to a page with the image shown in full screen, and can return back to the all-images page by a return button.
5. The user can modify a configuration file (a .txt file on the server), by a web interface. In this files there are some values the need to be setted (otherwise it use default values), and the user can do it by radio buttons (in this way he is forced to chose valid values). This configuration file can modify the way in which gAn works and modify the resulting output (both the text and the images).

2.1.2 Late version

The late version is more complex. It was born from the tips and the observation of the first users. The modifications are not numerous, but there are a lot of additions of new features. All the new required characteristics are exposed following:

1. The user can insert multiple runs: separated by a semicolon (but in case of errors the system can automatically correct them replacing symbols like "-" or "," or "." with semicolons and giving a more robust service). These runs can be inserted by an input field of by a range select button: this button open a "modal" that allows the user to chose the first run and the last, and automatically insert the comprised runs (for example, if the user inserts 30000 and 30010 the system inserts automatically all the run numbers between 30000 and 30010). This modal has a validation system, that ensure the correctness of the inserted values.
2. The user can chose which branch of gAn use for the execution of the program. In this moments exist 2 complete branches, but in the future they can became more. They are externally very similar, the differences are the algorithms in the program, but they give a different output (different output but in the same format: text and images). Another difference is that the configuration file is not written mandatory in txt, like in the early version, but it can be written in xml, it depends which branch you chose (see next point).
3. The configuration file is not only in text format, but also can be in xml format (it depends on the selected version of gAn). This fact can ensure a stronger structure, and must be transparent to the user (he mustn't see differences).
4. The user can chose what version of Root he wants to use for the program. Theoretically different versions of Root are perfectly compatible, but in practice each version of gAn is designed to work with a particular version of Root and to avoid problems it was specifically requested to the gAn web designer to allows the user to chose freely which version of Root use among the installed versions on the server.
5. The user can save images on his hard disk: he can chose from the shown images in the images page an image to download

by clicking on a specific download button near the image. Furthermore there is another button "Download All" with whom the user can simple download all the output images.

6. The user can download a reduced version of the root file with informations about the images and the results: gAn produce this kind of files as "half-processed" during the computing, and it is not a problem to save this on the hard disk of the server in a specified folder. For an expert user can be scientifically interesting have this file (this root file contain more information than the output, the most of this information is useless (it is an "half-processed" file) , but sometimes an expert user can find something interesting), so the user must have the opportunity to download this.

7. The first little group of user prefers the dropdown menu to the radio button, so all the radio buttons in the program are replaced by dropdown menus.

8. The user has to access not only to a png image, but to a root-image. This kind of image is interactive: the user can with a left click of the mouse (a continued click, like the "dragging") select parts of the image and zoom them, and with a right click do dynamically some kind of image processing (set colors, chose what kind of chart to show, modify the chart legend, translate in a 3D space the image etcetera). All of this must be done by the user through a browser window. This requisite seems to be very complex, but Root provides libraries (these libraries work well but they are poorly documented) to interact with Javascript, and can in some way resolve the problem.

9. In the homepage the user can see the run number of the last root file produced by the machine, and its creation date and time (so, he can understand what is the maximum of the range of the insertable numbers)

   In the late version there was another functional requisite: ideally the user should have been able to select a gAn version also if not installed in the server machine: in this case the system should have been capable to automatically search on the AEgIS Gitlab repository the correct version (if existing), download it, unpack it in the server, and use it to execute the program. After some discussion this requirement has been cancelled, because it was considered complex, basically useless, and potentially harmful (on the branches of the repository there are untested

and incomplete versions, that can create if executed wrong outputs, so wrong scientific results). At this moment installing manually the stable versions of gAn on the server seems to be a more smart way to work.

10. There is a login system: the user must insert the password of the office to use the system. The authentication is based on the confrontation between the hash function of the inserted password and the hash function of the AEgIS password. If the password is correct the user receives a cookie, before each action in the site the server request and check this cookie to be sure about the identity of the user.

2.1.3 Ambiguities (and related solutions)

At least a point seems to be quite ambiguous:

The textual output of gAn needs to be formatted in some way to be more organized and clear? The answer is difficult: for a non-physicists this output seems to be disordered, too long, with too many groups of informations, and very difficult to understand, but on this question all the physicists questioned answered that the output is perfectly clear and doesn't need to be modified or improved in any way. The only requests of the users were about the font and the font-size. Anyway, in the second version, in case of multiple run selection, there is a "navbar" that allows the user to show only a run-result per time.

2.2 Non-functional requirements

Both versions, the first and the second, have three non-functional requisites:

1. The first is quite simple: gAn web has to ensure that in case of crash of the program the web server mustn't crash too. The point is that on this web server (Apache server, installed on Linux) there are some other important applications, so, if gAn web crashes it is not a big problem, but the crash cannot force

Apache, or worst the entire machine, to stop or restart. This requisite is quite easy to meet: a modern web application based on Html, Javascript, PHP and CSS is quite safe, a general crash of the server it is very unlikely to happen. If the C++ application or some Root libraries crashes (for example if the user asks for an inexistent run) the web application gracefully warn the user about the problem, but without uncontrolled behaviours.

2. The application must work without install nothing. Also this requirement is very easy to meet: gAn web is a web interface, it requires only a browser, nothing else.

3. The application must be easy to be modified and extended in the future by persons who aren't necessarily software engineers. The point is that the student who wrote this program is a "momentary collaborator" in the AEgIS experiment, and all the modifications to the program must be done by other people, in most cases physicists. So the best way is to comment in detail the code and keep the code simple (this is a basic good-programming requirement).

Chapter 3

Functional Analysis

3.1 Scenario-based requirements analysis

Following there are a list of scenarios in which a user achieves a goal by doing a list of steps. The goal of these scenarios is to show in detail how the interaction between the user and the system takes place.

1. The user wants to analyse the runs between 30000 and 30010, plus the run 31456, to make a confrontation, he is interested both in the text-output and in the images: the user goes to the homepage, he is redirected to the authentication page and he do the login. If successful he returns automatically in the homepage, and he can insert the runs between 30000 and 30010 manually separating them with a semicolon or better clicking the "add range of runs" button, that opens a modal, in which the user can insert the minimum and the maximum of the range, and confirm (confirmation redirects the user to the homepage). After the user can add manually the run 31456 separating it from the others with a semicolon. If the inserted run numbers make doesn't make sense the system show on the page an error message. If the inserted run numbers are valid, the user can click the "send" button (before the button was red and un-clickable, now it is green and clickable) and start gAn. A progress bar is shown, a waiting message appears, and the user waits for some seconds (at this stage of development it is very hard to predict how much time gAn requests to execute). After some seconds the user is redirected in a page that shows the textual results: on the top of the page there is a navbar that shows the computed run numbers: the user uses this navbar to chose what part of the results to show in the screen. This bar is draggable, the user can freely move it. From this page the user can, through the button "show all images", access to

another page dedicated to the images. In this page he can configure through dropdown menus the dimension, the layout, the group of the images (each image belongs to a group, the group depends on the sensor that generates the data from that the image is generated) to show. He can also, by clicking on a single image, access to another page, with a single image (the clicked image) that is completely accessible: the user can rotate the image, move it in a 3d space, zoom in, zoom out, select part, do some basic digital image processing and chose the kind of chart to show.

2. The user wants to use the version 5.34 of Root (an old but stable version) to execute gAn: he complete the login, in the home page there is a button name "Chose Root version", clicking on this button the user is sent to a page where, he is informed about the current version of Root, and through a dropdown menu can chose among some version of Root already installed on the server (all the acceptable Root versions are installed on the server) . If the 5.34 version is one of the installed version he can select it and confirm, the goal is achieved. If the 5.34 version is not installed, the user cannot use this version.

3. The user wants to select the "Rug-dev" branch of gAn: the process is very similar to the process that allows the user to chose a Root version. The user complete the login, in the home page there is a button name "Chose gAn version", clicking on this button the user is sent to a page where, he, through a dropdown menu, can chose among some branches of gAN existing on the machine. "Rug-dev" is one of these, the user can confirm and the task is completed.

4. The user wants to make the computation using only the data taken from the sensor named "Mimito": The user, after the login, in the homepage can use the button "Edit Config" the reach a page in which, through some dropdown menus, he can change the configuration file of gAn. Each of the dropdown menus is related to a sensor (often they are 5-6, it depends on the branch), and the option of the dropdown are "yes" or "no": if "yes" is selected the sensor's data are used in the computation, if "no" they aren't. One of the dropdown menus is named "Mimito", the user select "yes" for this sensor, "no" for all the others.

5.

6. The user want to download all the images related to the runs 40001 and 40002: He can, after the login, insert the runs in the

homepage (it is possible both by input field and by range selector), run gAn, wait the end of the execution, click "Show all images", and from here click "download all images". All the images will be downloaded in png format.

7. The user wants to download the semi-processed root file related to the runs 31111 and 31112: The steps are the same as the steps used to download the images, but instead of the button "Show all images", the user has to use the button "download root files".

Chapter 4

Prototypation

4.1 Scenario-based requirements analysis