

# FPA Depot – Web Application

Edwin M. Martinez Avila<sup>1</sup>, Ricardo Muñiz<sup>2</sup>, Jamie Szafran<sup>3</sup>, Adam Dalton<sup>4</sup>

University of Puerto Rico at Arecibo

<sup>1</sup>Edwin.m.Martinez2@upr.edu

<sup>2</sup>Ricardo.muniz2@upr.edu

\*National Aeronautics and Space Administration

<sup>3</sup>Jamie.szafran@nasa.gov

<sup>4</sup>Adam.s.dalton@nasa.gov

**Abstract**—Lines of code (LOC) analysis is one of the methods used to measure programmer productivity and estimate schedules of programming projects. The Launch Control System (LCS) had previously used this method to estimate the amount of work and to plan development efforts. The disadvantage of using LOC as a measure of effort is that one can only measure 30% to 35% of the total effort of software projects involves coding [8]. In the application, instead of using the LOC we are using function point for a better estimation of hours in each software to develop.

Because of these disadvantages, Jamie Szafran of the System Software Branch of Control And Data Systems (NE-C3) at Kennedy Space Center developed a web application called Function Point Analysis (FPA) Depot. The objective of this web application is that the LCS software architecture team can use the data to more accurately estimate the effort required to implement customer requirements. This paper describes the evolution of the domain model used for function point analysis as project managers continually strive to generate more accurate estimates.

## Nomenclature

**LCS:** Launch Control System

**NE-C:** NASA Engineering Control and Data Systems

**Ruby:** A dynamic, reflective, general-purpose object-oriented programming language

**Ruby on Rails:** Web application framework for the Ruby language.

**DA:** Development Activity

**CSCI:** Computer Software Configuration Item

**FPA:** Function Point Analyses

**FPA Version:** Different versions of an FPA.

**Keywords**— Java, Ruby-on-Rails, Web – Application, Server, Mid – Level, database, and design.

## INTRODUCTION

For a long time, “the software engineering field has counted Lines of Code (LOC), which is a hard and fast measurement of the amount of effort, time, and resources that a software project takes” [1]. LOC has the disadvantage that

it does not fully cover the development process, only the coding implementation; in addition it is not a consistent measure to compare productivity across different tools and platforms. Over time, engineers have come to conclusion that using the LOC metric to measure both programmer productivity and estimation of the programming timelines is not very effective. This stance is further solidified by the revelation that given two algorithms for implementing identical functionality, the method with fewer lines of code is usually preferable. On the other hand, Function Point Analyses offers improved project estimation because it is an independent methodology that can be used to compare products across differing toolsets, languages, and platforms.

In 2009, Jamie Szafran was a student employee of the Kennedy Space Center and assigned the task of creating the first version of the Function Point Analysis Depot. At the time, it was a little different from the spreadsheet (see Figure 1) that was being used by the Launch Control System (LCS) for the collection of function and was processed by hand. The problem with the use of spreadsheets is that the estimate of costs cannot be automated. In 2010, Szafran was hired fulltime and tasked with re-designing the Function Point Analysts Depot (FPA Depot) under the direction of software architect at the time. This implementation was then used to store Function Point Analysis results. These results were used by the architects of the LCS project to estimate the effort needed to implement the requirements of the customer. Furthermore as stated in *FPA Depot Orientation for New Developers* [1], “the application is expected to support data correlation between the entry point function of analysis of software engineers and requirements management tool currently in use by the LCS project”. The newest version of the FPA Depot promises to provide the LCS a precise technique to objectively estimate efforts required with more precision than previously possible.

On June 6, 2011, Edwin Martinez was selected by Space Grant Puerto Rico for work at NASA KSC as an intern to working in the Space Station Processing Facility (SSPF) with Engineering Support Software NE-C2 branch on the LCS project. The assignment included migrating the FPA Depot from version 4.1 to version 5.1. The project required the restructuring of a Java server mid-level (web service) and

creation of new methods for the schema 5.1. The work was coordinated with the efforts of Ricardo Muñiz, intern member of the NE-C3 branch, who restructured the interface side of this application.

Description	Simple	Average	Complex
<b>External Inputs (EI)</b>			
Input Screens			
Interactive Inputs			
Hardware Inputs			
Batch Input Streams			
<b>External Outputs (EO)</b>			
Screen Reports			
Printed Reports			
Media Outputs			
Software Outputs			
Hardware Outputs			
<b>External Inquiries (EQ)</b>			
Request/Response			
Menus			
Context Sensitive Help			
Embedded Computer Inquiries			
<b>External Interface Files (EIF)</b>			
Reference Data			
Fixed Messages			
Shared Data Files			
<b>Internal Logical Files (ILF)</b>			
Application Data Groups			
Data Tables			
Shared Data Files			
<b>Internal Functions</b>			

From FPA Depot Orientation for New Developers [1]

This paper is organized as follows. Section II discusses the tools, methods and approaches that were used in creating the Java mid-level of the application. Section III describes the design of the FPA Depot interface. Section IV is the results, section V is the conclusion, and section VI contains the references for this paper.

## II. IMPLEMENTATION OF THE JAVA MID – LEVEL

The Java mid-level is a web service, which is defined as “set of Hypertext Transfer Protocol (HTTP) request messages along with a definition of the structure of response messages, usually expressed in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format [4]”. This mid level was developed to be functional independently of GUI implementation. GUI development could then be performed in different languages and use it with the mid level without problem. The changes made to the web service were implemented in accordance to REST (Representational State Transfer) style, a that works by constraining the interface to a set of well-known, standard operations (specifically GET, POST, PUT, and DELETE).

### A. Tools for the Java Mid Level

The design of the new schema for the FPA Depot required configuration and development using a toolset consisting of the HTTP protocol, XML, the Apache Tomcat servlet container, and the PostgreSQL database. The Eclipse IDE

was used development and changing all the methods to the new version 5.1.

### B. New Classes and Methods

The new classes were developed for this application were: Developers, Developer Activities, CSCI, FPA Version, Actuals, and Granularity. The classes that existed before but were changed for this version were: FPA, HTTP Request Handler, Database Interface, Sax Event Handler, Xml Tags, and Xml Builder. Below, the most important and basic classes of mid level are explained.

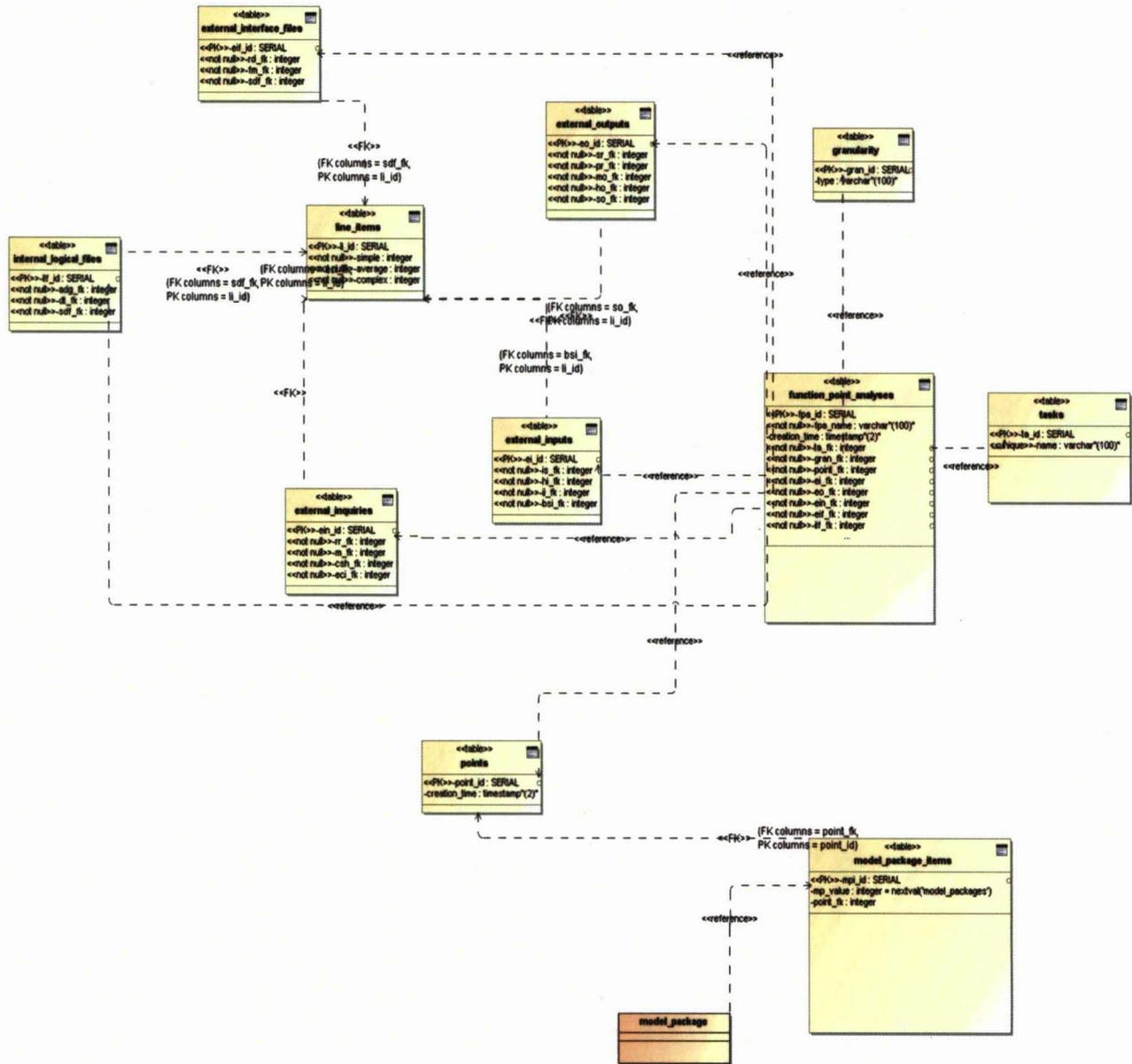
- 1) **Sax Event Handler:** SAX stands for “the Simple API for XML” [3]. This class contains methods that receive the info in XML (e.g. see Figure 3) from any GUI, and send them to the Http Request Handler to input the information into the database.

```
<developer>
  <developerName>Someone</developerName>
  <username>someUser</username>
</developer>
```

- 2) **Http Request Handler:** This class receives GET, POST PUT and DELETE methods and, the combined with the URL determines what function will be processed. PUT, DELETE or POST contain information in XML format and are parsed before being sent to the Sax Event Handler where the database is updated. On the other hand, if the class Http Request Handler receives a GET request, it goes directly to the Database Interface class to get the information from the database.
- 3) **Database Interface:** This class is responsible for obtaining, creating, deleting, and modifying the data from the tables in the PostgreSQL database. The data are obtained from the database through pre-specified queries that provide additional security and abstraction by preventing a user from interfacing with the database directly.
- 4) **Xml Builder:** In this class, the data is retrieved from database and converted to XML.
- 5) **Http Request Filter:** This class is responsible for receiving the HTTP request from the Apache server and filtering it using the REST protocol. In this way HTTP operations can be routed to the appropriated Http Request Handler method.

C. Old Schema 4.1 Version FPA Depot:

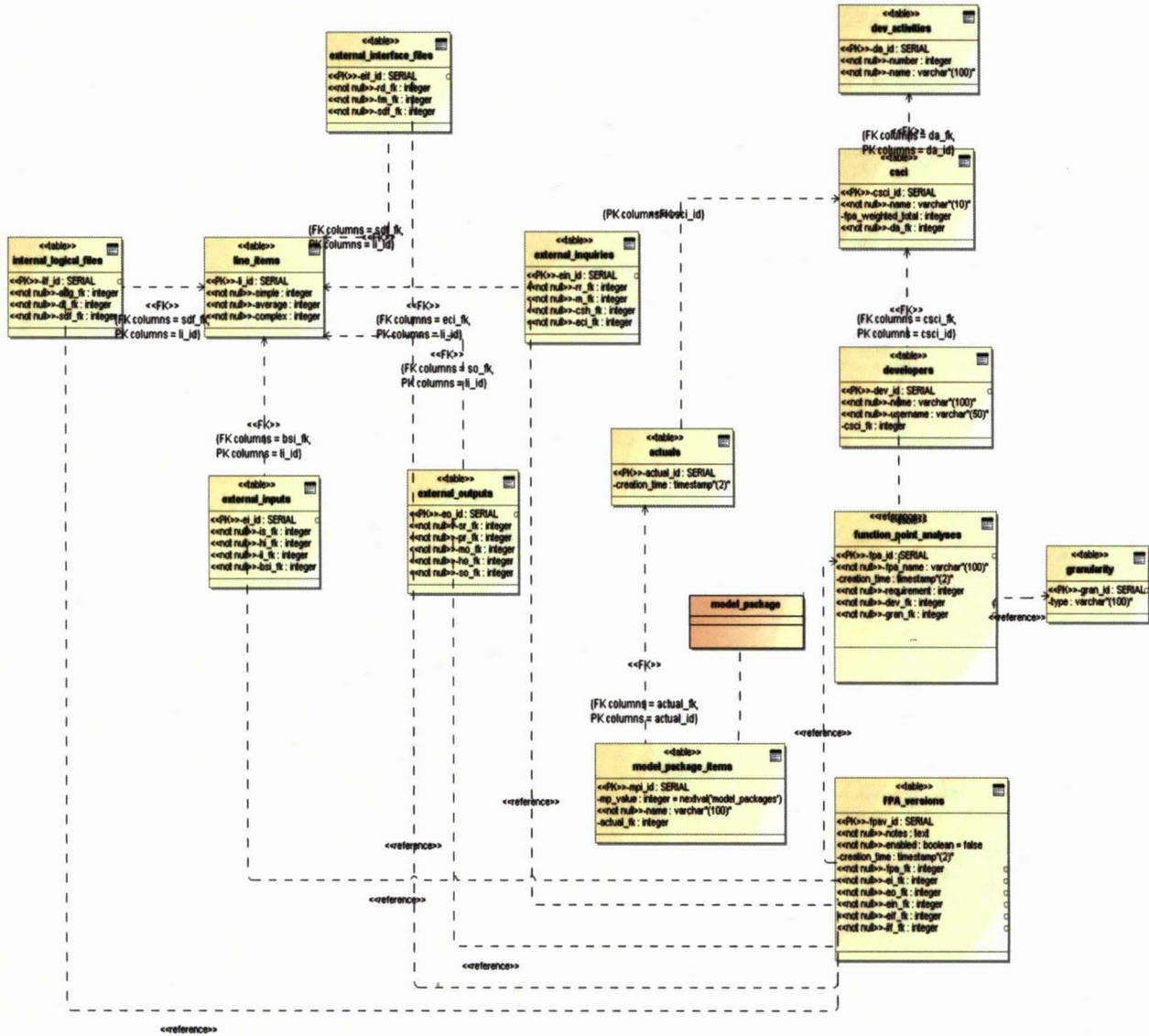
The schema of versions 4.1 of the FPA Depot was revised to include information that would be more useful to the Software Architecture team. The original schema (for this revision) was this:



[From FPA Depot Orientation for New Developers [1]]  
 Schema,IFPADepot 4.1

## D. New Schema 5.1 Version FPA Depot

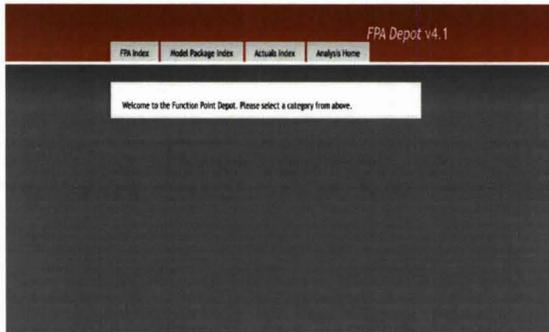
In order to utilize the change of this schema, the database had to be migrated and the Java mid-level and Ruby-on-Rails GUI had to be restructured.



(From FPA Depot Orientation for New Developers [1])  
Schema, IFPA Depot 5.1

### III. INTERFACE DESIGN

The GUI of the FPA Depot was originally written by Jamie Szafran in Flex, an open-source version of Flash that is a combination of ActionScript and MXML, which called back RESTfully to the mid-level. Flash proved to have some security issues, and the decision was restructure the interface design using Ruby-on-Rails. The initial refactoring concluded on June 25, 2011, the main screen of the GUI looked as follow:



#### A. Active Resource on Ruby:

The GUI for the application version 4.1 was coding on Ruby on Rails, but without using Active Resource. Adam Dalton (NE-C2 branch), had the idea of implementing Active Resource in this application. Active Resource is a class in Rails that "provides a framework for handling the connection between business objects and RESTful web services"[7]. We are using the XML methods that this class has to request and send data to the mid-level. Remember that the mid-level was created to receive and send the data using XML. The Active Resource model were created in such a way that the XML that was automatically generated by Rails would adhere to the API of the Java mid-level.

\*\*\* Example of the code without Active Resource. This function is for getting all CSCIs.

```
xml = Csci.get_all_cscis_xml
cscis_doc = REXML::Document.new(xml)
@cscis = Csci.get_list_cscis_from_xml_element(cscis_doc)
respond_to do |format|
  format.html & render :index.html.erb
```

\*\*\* Example of the code using Active Resource. This function is for getting all CSCIs.

```
#Build a list of CSCIs
@cscis = Csci.find(:all)
```

With Active Resource we can do the same thing, but with fewer lines of code.

#### B. New Interfaces Developed

In order to change FPA Depot to the new version 5.1, and with the approval of our mentors, Ricardo Muñiz and I decided to change the design of the GUI to a new one. In addition we worked on the design of new interfaces:

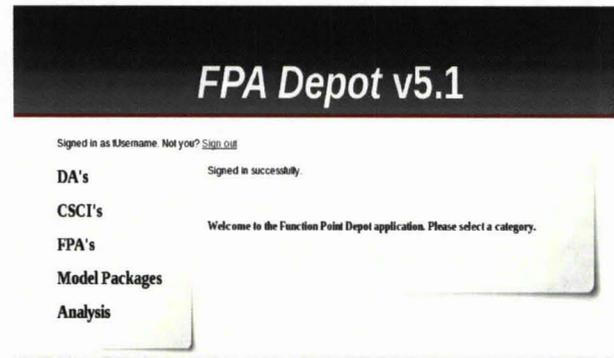
##### 1. Login Interface:

This interface has a requirement for the new FPA Depot 5.1. Here Ricardo Muñiz created the method of user authentication. Using this method the user has to have an account in order to login to the application.



##### 2. Welcome Interface:

In the main interface, we can see on the left of the page the new menu for the user that is different from version 4.1 (see figure 2). Here the user can select one option in order to edit, add or show the data that they need from the database.



##### 3. Development Activities Interface:

This interface is to display all the data of development activities. If the user needs to edit a development activity, they just have to select one activity and change the data. The

user can also create a new development activity, if they press the “new DA link” that is below the interface.

#### 4. Developer Interface:

This is the developer interface for showing developer information. In this interface the user can see the FPAs that belong to the developer and click it to see the FPA Versions.

#### 5. Function Point Analysis Interface:

In this interface the user can see all the information of the FPA that they selected, as well as see all of the FPA versions that exist inside of the FPA. In addition the user can create a new FPA version or edit an existing one.

#### 6. Analysis Interface:

This interface has different options for analysis and estimation of the function points and labour hours.

#### IV. Results

Using Java, Ruby-on-Rails, and other tools, we achieved good results for this application. Some results that we achieved were:

- Part of the Java mid-level and the Ruby-on-Rails GUI were changed to handle the new schema.
- Parts of the database interface, XML processing, and results displaying were changed.
- The XML schema in the accompanying release notes file was changed.
- The GUI was changed to accommodate the new schema.

## V. CONCLUSION

The purpose of this project was to edit an existing web application that will be used by developers in order to input and track function point analysis results from the Launch Control System (LCS) software development team. This web application will serve as a central point for storing all function analysis results. The LCS software architect team will also use the data from this app to estimate the effort required to implement customer requirements. In this internship my work was implementing schema changes in the Java server mid-level and changing XML format for FPA to conform to the new design.

## VI. ACKNOWLEDGEMENTS

I think that this internship at Kennedy Space Center has given me the experience to work hard in a real environment, and has also increased my chances of having success in finding a good job or even increased the chances of working permanently at NASA in the future. It has also helped to expand my knowledge and increased my self-esteem.

Finally, I would like to thank all of the people that helped me to learn new methods and new languages to improve my work. Also I would like to thank all of the reviewers for helping to improve the work. This work was carried out as part of the project of NE-C3 and NE-C2 branch on the LCS project.

## VII. REFERENCES

- [1] J. Szafran, "FPA Depot Orientation for New Developers," Last revised 7 June 2011.
- [2] D. Megginson, "About Sax," <http://www.saxproject.org>.
- [3] D. Megginson, "Megginson Technologies Ltd." "2000 – 05 – 05."
- [4] Package org.apache.http.protocol, "The Apache Software Foundation" <http://hc.apache.org/httpcomponents-core-ga/httpcore/apidocs/org/apache/http/protocol/HttpRequestHandler.html>, 2005 – 2011.
- [5] Benjamin, "*Diseño de Interfaces de Usuario Usables: Una Guía Rápida para Desarrolladores de Software Libre y de Código Abierto.*" Web. 8 December 2004. <http://mundogeek.net/traducciones/interfaces-usuario-usables/gui.html>
- [6] Sam Ruby, Dave Thomas, David Heinemeier Hansson, "Agile Web Development with Rails," The Facets of Ruby Series, 2011-03-31

[7] Ruby on Rails v3.0.9, "Class ActiveRecord::Base," <http://api.rubyonrails.org/classes/ActiveResource/Base.html>.

[8] Karmanadham V.V.G.B. Gollapudi, "Function Points or Lines of Code? - An Insight", Global Microsoft Business Unit, <http://sst.umt.edu.pk>

[9] "Source lines of code," [http://en.wikipedia.org/wiki/Source\\_lines\\_of\\_code](http://en.wikipedia.org/wiki/Source_lines_of_code)

[10] Andy Hertzfeld, "-2000 Lines of Codes", [http://www.folklore.org/StoryView.py?project=Macintosh&story=Negative\\_2000\\_Lines\\_Of\\_Code.txt](http://www.folklore.org/StoryView.py?project=Macintosh&story=Negative_2000_Lines_Of_Code.txt)

[11] "Lines of Codes", <http://c2.com/cgi/wiki?LinesOfCode>

[12] Roger Heller, "An Introduction to Function Point Analysis", 2000-2002, <http://www.qpmg.com/fp-intro.htm>