UNIVERSIDAD
REY JUAN CARLOS

INGENIERO DE TELECOMUNICACIÓN

Curso Académico 2016/2017

Proyecto Fin de Carrera

ANÁLISIS ESTADÍSTICO DE PROYECTOS SCRATCH CON PANDAS

Autor : José Manuel Durán Vivancos

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

Análisis Estadístico de Proyectos Scratch con Pandas

Autor : José Manuel Durán Vivancos

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2017

Dedicado a

mi familia

II

Agradecimientos

A quienes lo hicieron posible. Muchas gracias.

Resumen

Este proyecto tiene como objetivo medir y puntuar el grado de pensamiento computacional de cientos de programas Scratch almacenados en el repositorio web del Instituto Tecnológico de Massachussets (MIT). Para llevar a cabo este objetivo se hará un estudio estadístico que se dividirá en los siguientes pasos:

- Obtención de código fuente Scratch
- Análisis del código fuente
- Modelado estadístico

Summary

This project targets to measure the level of computational thinking of hundreds of Scratch programs stored in the database of the Massachussets Institute of Tecnology (MIT). To reach such goal an statistical analysis will be peformed by acomplishing the following steps:

- Obtainining Scratch source code
- Source code analysis
- Statistical modelling

Índice general

Índice de figuras

Capítulo 1

Introducción

Este proyecto se basa en el análisis estadístico de cientos de programas realizados con el lenguaje de programación Scratch, un lenguaje pensado para introducir en la programación a personas de corta edad o poca formación técnica. Este proyecto utiliza la librería de análisis común a la utilizada por Doctor Scratch para obtener una puntuación de cada uno de los programas Scratch analizados, reflejando esta la calidad de la programación a través de diferentes parámetros. El resultado final es un análisis estadístico de cientos de programas Scratch que permite visualizar las capacidades más y menos desarrolladas en materia de programación dentro de la comunidad de usuarios de Scratch.

1.1. Motivación

Scratch es un lenguaje de programación visual utilizado por estudiantes, profesores y padres para crear fácilmente animaciones y juegos. Proporciona ademas una puerta de entrada en el mundo de la programación para los estudiantes de cursos mas tempranos, siendo una estupenda inicialización en esta disciplina técnica que es la programación.

Dado que este lenguaje de programación fue creado para los mas novatos, parece una buena idea crear una herramienta de benchmarking para el análisis de las cualidades de los mas principiantes en materia de programación. Doctor Scratch fue creado para satisfacer esa necesidad, proporcionando una interfaz web de fácil acceso donde el código de Scratch puede ser subido para su análisis. El correspondiente servidor corriendo doctor Scratch generará una salida que representa la puntuación en los diferentes parámetros de programación.

Pero todo lo mencionado en estos primeros párrafos ya existe, lo que realmente es nuevo e introduce este proyecto, es proveer de una herramienta para el análisis masivo de múltiples programas de Scratch. Esta nueva aplicación se aproxima a la programación en Scratch de una forma industrial, bajándose en un primer paso 1332 programas escritos en Scratch residentes en la base de datos del MIT, para luego en un segundo paso proceder al escaneo de todos ellos aplicando la librería Hairball en la que se basa Doctor Scratch. El resultado es la creación de una base de datos con todos los resultados de cada uno de estos programas, pudiéndose de este modo analizar tendencias y hábitos comunes en materia de programación.

1.2. Estructura de la Memoria

La memoria es un documento donde se describe el seguimiento del proyecto de forma detallada, desde el momento en que surge la idea hasta la obtención de unos resultados tangibles. Los capítulos que aparecerán a lo largo de esta memoria se detallan a continuación.

La introducción, es el primero de sus capítulos, explica la idea general del proyecto, los motivos de su elección y los objetivos planteados.

El siguiente capitulo es el estado del arte, en este capítulo se explica cuales son las fuentes de conocimiento y tecnologías utilizadas para la consecución y entendimiento de este proyecto. Estas tecnologías se pueden entender como un conjunto de capas superpuestas que dan lugar a un resultado final. La capa mas inferior sería el lenguaje de programación Python, este lenguaje son los ladrillos en con los que se construye este proyecto. Las librerías de Hairball y Doctor Scratch son las capas intermedias, unos módulos de los cuales podemos aprovecharnos para obtener resultados específicos necesarios. La capa final sería la programación realizada en este proyecto, aprovechando las capas inferiores moldearemos una aplicación con la cual obtendremos los datos estadísticos necesarios para nuestro análisis. El siguiente capítulo describe la solución e implementación del proyecto, en el se recogen las características de la aplicación según los requisitos indicados y las librerías utilizadas. En el capítulo de resultados se explica de forma detallada las pruebas realizadas durante el proceso de codificación de la aplicación. Una vez finalizado el proyecto, en el capítulo de conclusiones, se analizan las aportaciones de éste, los objetivos conseguidos y los no conseguidos, así como las posibles mejoras y ampliaciones futuras.

En el anexo 1 describimos el entorno y equipos con los cuales se ha realizado el desarrollo. En el anexo 2 obtendremos un pequeño manual de usuario, con capturas de pantallas y explicaciones que permitirán gestionar, e instalar de una forma correcta nuestra entorno de desarrollo. En el anexo 3 se muestra un cálculo de presupuesto donde se describen los requisitos económicos derivados de los recursos humanos y materiales necesarios para la ejecución del proyecto. Aquí hablaremos del plan de proyecto y planificación, sección en la que se recoge el conjunto de actividades que permiten desarrollar, gestionar, ejecutar y controlar el proyecto. A partir de esta información se decide si el proyecto es viable.

En el capítulo de referencias se detalla la bibliografía, la cual engloba el conjunto de libros

y páginas Web consultadas.

Capítulo 2

Objetivos

El objetivo principal del proyecto consiste en obtener un conjunto de histogramas que nos aporten información estadística acerca del nivel de pensamiento computacional dentro de un espacio muestral determinado. Para lograr este fin deberemos descomponer el proyecto en un conjunto de tareas o subobjetivos.

2.1. Tareas

De las múltiples tareas necesarias para la elaboración de este proyecto, se detallan a continuación los grupos de tareas mas representativos.

1. Diseño del plan de proyecto
2. Comprensión y dominio del estado del arte
    a) Estudio del lenguaje de programación Python
    b) Estudio del lenguaje de programación Scratch
    c) Estudio de la aplicación Doctor Scratch
    d) Estudio del la librería de Python Pandas
3. Implementación
    a) Preparación del entorno de desarrollo
    b) Programación del script de download
    c) Programación del script de análisis
    d) Programación del script de dibujado de gráficos
    e) Programación del script de almacenamiento de resultados en fichero
4. Análisis de resultados
5. Cálculo de presupuesto
6. Documentación y memoria del proyecto

2.2. Planificación Temporal

A continuación se detalla la duración de cada uno de los grupos de tareas, de esta forma se hace fácilmente visualizable el tiempo necesario para la elaboración de este proyecto fin de carrera.

1. Diseño del plan de proyecto. 1 semana.
2. Comprensión y dominio del estado del arte. 1 semana.
3. Implementación. 3 semanas.
4. Análisis de resultados. 1 semanas.
5. Cálculo de presupuesto. 0.5 semanas.
6. Documentación y memoria del proyecto. 2 semanas.

Todas estas tareas dan lugar a una duración completa del proyecto de 6 meses.

Capítulo 3

Estado del Arte

3.1. Pensamiento Computacional

Uno de los pilares de este proyecto es el pensamiento computacional. El objetivo final de este proyecto es evaluar como de sofisticado es dentro de nuestro espacio muestral, por ello es de vital importancia conocer la materia que estamos evaluando.

El pensamiento computacional es una disciplina que nos permite partir de un un problema complejo, entenderlo y desarrollar posibles soluciones. Podemos entonces presentar estas soluciones de una manera que un ordenador, un humano, o ambos, puedan entenderlo.

Hay cuatro técnicas clave para el pensamiento computacional: Descomposición, basada en descomponer un problema o sistema complejo en partes más pequeñas y más manejables. Reconocimiento de patrones, busca similitudes entre los problemas. Abstracción, se centra sólo en la información importante, ignorando detalles irrelevantes. Algoritmos, desarrollan de una solución paso a paso para el problema, o las reglas a seguir para resolver el problema.

Cada técnica es tan importante como las otras. Son como las patas de una mesa, si falta una pata, la mesa se derrumbará, la correcta aplicación de las cuatro técnicas ayudará a programar un ordenador, el cual puede usarse para ayudarnos a resolver problemas. Sin embargo, antes de que se pueda abordar un problema, es necesario entenderlo en sí mismo y las maneras en que puede resolverse.

3.2. Lenguaje de Programación Scratch

Scratch permite a los usuarios utilizar una programación basada en eventos con múltiples objetos activos llamados Sprites. Estos Sprites pueden ser dibujados como mapas de bits o vectores generados desde un editor de Scratch o simplemente importados desde otra fuente externa, como por ejemplo una cámara web.

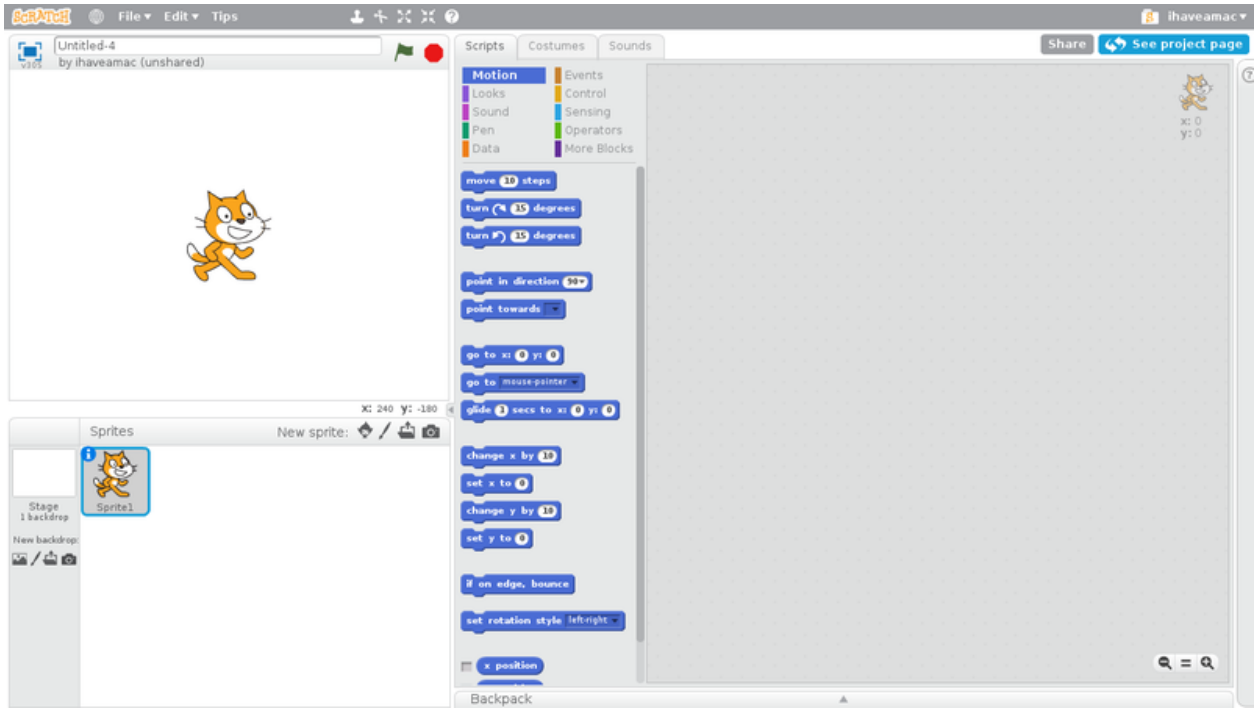

Figura 3.1: Interfaz de programación Scratch

Scratching en el ámbito informático significa reutilizar código para satisfacer nuevos escenarios, lo cual es una de las características mas importantes de Scratch. Los usuarios pueden bajarse y editar programas subidos y creados anteriormente por otros usuarios. Es también una forma de reconocer el mérito de los usuarios que crearon los programas originales. El nombre Scratch deriva de la técnica de mezclar sonidos utilizada por los disc jockey, utilizándose esta equivalencia para el desarrollo de programas Scratch basados en diferentes programas iniciales.

Scratch se empezó a hacer famoso en Reino Unido a través de los diferentes clubs de código, siendo utilizado este lenguaje por su relativa facilidad para hacer programas interesantes que desarrollan habilidades que despues pueden ser fácilmente transladadas a lenguajes de programación secuencial tipo Python o Java.

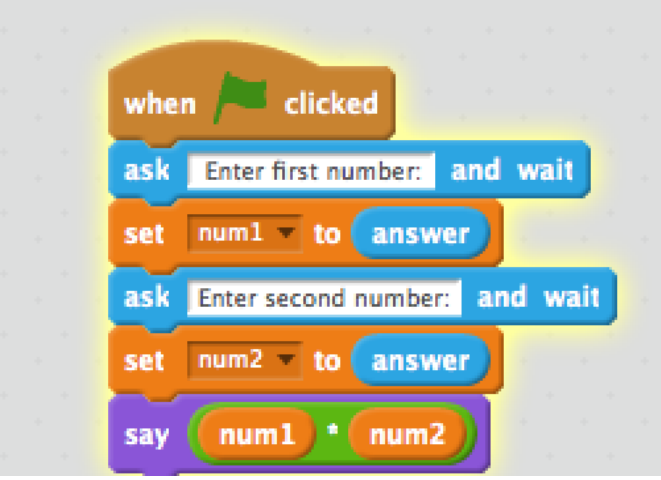

Figura 3.2: Ejemplo de script Scratch

Scratch no se utiliza exclusivamente para la creación de programas, dado que proporciona muchos elementos visuales, los programadores pueden crear historias animadas. Para estudiantes de mayor edad, se puede usar Scratch para crear relatos foto periodísticos, para crear lecciones animadas de matemáticas o cualquier otro tipo de contenido didáctico o audiovisual de una forma sencilla muy adecuada para entornos educativos. Utilizar Scratch permite a los mas jóvenes entender la lógica de programación además de como crear y cooperar.

Para mas información sobre lenguaje de programación Scratch referirse a [2] y [3].

3.3. Python

Para el análisis de los programas Scratch se utilizarán herramientas codificadas utilizando el lenguaje de programación Python. Es un lenguaje de programación que está disponible gratuitamente y hace que programar sea casi tan fácil como escribir los propios pensamientos sobre la solución. El código se puede escribir una vez y correr en casi cualquier computadora sin necesidad de cambiar el programa.

Python es un lenguaje de programación de uso general que se puede utilizar en cualquier sistema operativo moderno. Puede usarse para procesar texto, números, imágenes, datos científicos y casi cualquier otro dato que pueda almacenar en un ordenador. Se utiliza diariamente en las operaciones del motor de búsqueda de Google, la web de vídeo YouTube, la NASA y la Bolsa de Nueva York. Estos son sólo algunos de los lugares donde Python juega un papel importante

en el éxito de las empresas, gobierno u otras organizaciones.

Python es un lenguaje interpretado. Esto significa que no se convierte en código legible por computadora antes de ejecutar el programa sino en tiempo de ejecución. En el pasado, este tipo de lenguaje se llamaba lenguaje de scripting, indicando que su uso era para tareas triviales. Sin embargo, los lenguajes de programación tipo Python han forzado un cambio en esa nomenclatura. Cada vez más, las grandes aplicaciones se escriben casi exclusivamente en Python. Algunos ámbitos de aplicación de Python incluyen programación de CGI para aplicaciones Web, creación de un lector RSS, leer y escribir en bases de datos, creación de webs en HTML, trabajo con archivos... y un largo etcétera.

Python admite módulos y paquetes, lo que fomenta la modularidad de la programación y la reutilización del código. El intérprete de Python y la amplia librería estándar están disponibles en forma de código fuente o binario, sin coste alguno para las plataformas principales, y se pueden distribuir libremente.

Para mas información sobre el lenguaje de programación Python referirse a [5].

3.4. Pandas

Uno de los módulos mas importantes de Python que utilizaremos en este proyecto se llama Pandas, este nombre se deriva de la fusión de las palabras ”panel data”. Pandas es una biblioteca de software escrita para el lenguaje de programación Python para la manipulación y análisis de datos. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. La librería está altamente optimizada para dar un alto rendimiento, con caminos de código crítico escritos en Cython o C. El desarrollador Wes McKinney comenzó a trabajar en pandas en 2008 para una empresa privada (AQR), creando una herramienta flexible para realizar análisis cuantitativos sobre datos financieros. Antes de abandonar la empresa, Wes McKinney fue capaz de convencer a la gerencia para que le permitiera abrir el código fuente de la librería.

Otro empleado de AQR, Chang She, se unió al esfuerzo en 2012 como el segundo mayor contribuyente a la librería.

Pandas es un software libre que en el caso de este proyecto utilizaremos para almacenar estructuras de datos matriciales que almacenarán nuestros resultados en forma de filas o entradas.

Para mas información sobre la librería Pandas referirse a [6].

3.5. Doctor Scratch y Hairball

Doctor Scratch es una aplicación web que permite tanto a profesores como estudiantes analizar de forma automatizada si los programas Scratch son realizados de una forma correcta en lo que a pensamiento computacional se refiere. Además permite también aprender de errores, recibir consejos para una mejora de la programación y recibir incentivos con los que premiar el esfuerzo y la superación de los programadores novatos.

La filosofía de esta aplicación queda definida en el paper científico ”Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional”[4]. En el se resume con total claridad cuales son los objetivos de Doctor Scratch:

Una de las barreras de entrada de la programación informática en las escuelas es la falta de herramientas que ayuden al profesorado en la evaluación de los proyectos del alumnado. Con el objetivo de resolver esta situación, este artículo presenta Dr. Scratch, una aplicación web que permite a educadores y alumnos analizar automáticamente proyectos Scratch, el lenguaje de programación más utilizado globalmente en educación primaria y secundaria, para comprobar si se han programado correctamente, aprender de sus errores y recibir retroalimentación para mejorar su código y desarrollar el Pensamiento Computacional (PC). Uno de los objetivos de Dr. Scratch, además de ayudar al docente en las tareas de evaluación, es ser un estímulo para animar a los aprendices a seguir mejorando sus habilidades de programación. Para comprobar la efectividad de la herramienta en este sentido, se organizaron talleres en 8 colegios con alumnos de entre 10 y 14 años en los que los estudiantes analizaron uno de sus proyectos Scratch con Dr. Scratch, leyeron la información del informe de resultados e intentaron mejorar sus proyectos usando los consejos ofrecidos por la herramienta. Al finalizar el taller los alumnos mejoraron su puntuación de PC así como sus habilidades como programadores.

También en [4] queda definido el sistema de puntuación, mostrado en la tabla 3.3.

Concepto}	Null (0 puntos)	Basic (1 puntos)	Developing (2 puntos)	Proficiency (3 puntos)
Abstraction	-	Más de un script o sprite	Definición de bloques	Uso de clones
Parallelism	-	Dos scripts en bandera verde	Dos scripts tras pulsación de tecla o click en sprite	Creación de clones al recibir mensaje
Logical thinking	-	if	if else	Operaciones lógicas
Synchronization	-	Esperar	Broadcast, cuando recibo mensaje, parar el programa, parar los sprite	Esperar hasta, broadcast y espera
Flow control	-	Secuencia de bloques	Repetir, siempre	Repite hasta
User Interactivity	-	Bandera Verde	Pulsación de tecla, click en sprite, mouse block	When A is > B, video, audio
Data representation	-	Modificadores de propiedades de sprites	Operaciones y variables	Operaciones en listas

Figura 3.3: Sistema de puntuación de Doctor Scratch

Estas reglas de calificación son las que dan un resultado final en forma de puntuación numérica, permitiendo un análisis automatizado del nivel de pensamiento computacional detectado en un proyecto de Scratch concreto.

Doctor Scratch utiliza librerías públicas para el análisis de programas Scratch (Hairball, Kurt...), añadiendo una interfaz web que simplifica enormemente el uso de estas librerias de análisis a través de un navegador web, eliminando la necesidad de un entorno Linux-Python con el cual ejecutar los comandos de Hairball y Kurt.

Para cada proyecto, Doctor Scratch comprueba lo siguiente...

- Algún fragmento de código que nunca se ejecuta.
- Si los atributos han sido inicializados correctamente.
- Si hay código repetido.
- Si los nombres de las variables siguen una semántica adecuada.

Con el fin de medir la lógica computacional de cada proyecto analizado, Doctor Scratch puntúa en función de la correctitud del pensamiento lógico, paralelización, control de flujo, interacción de usuario y representación de la información.

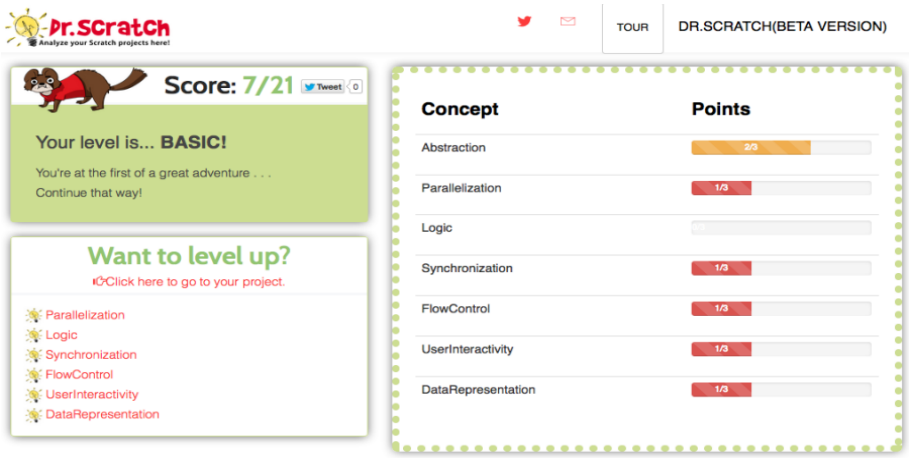

Figura 3.4: Resultado de Dr. Scratch

Los proyectos Scratch son almacenados en ficheros tar ball con una extensión especial *.sb2

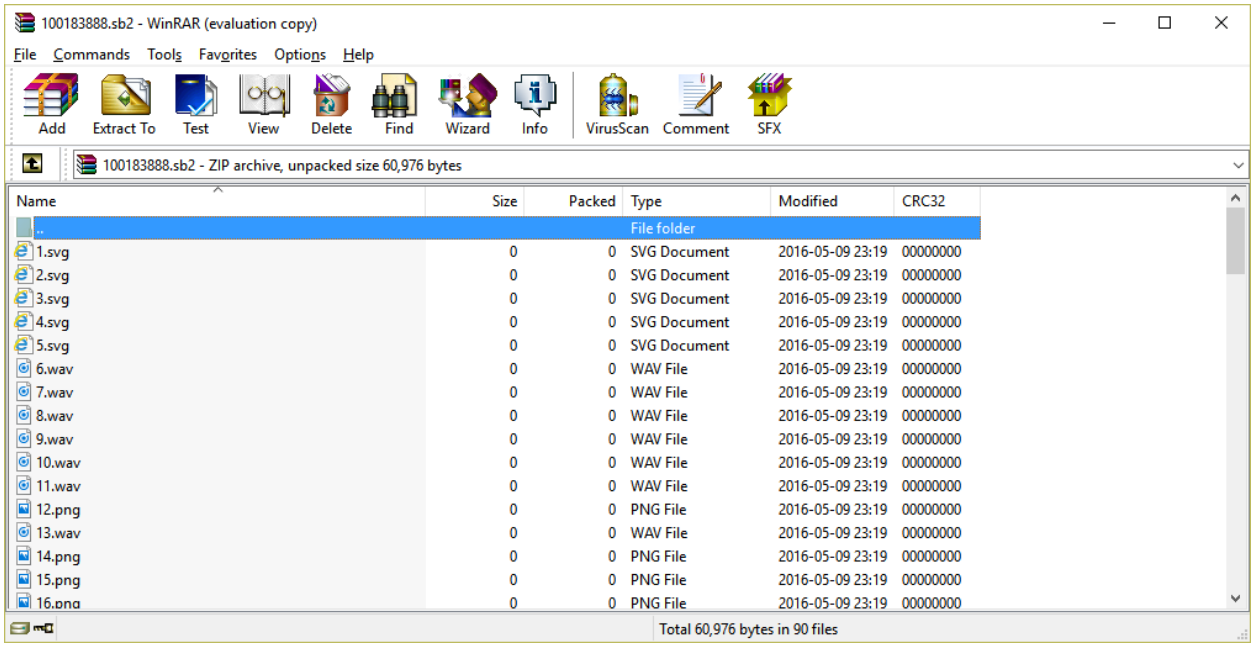

Figura 3.5: Ejemplo de fichero *.sb2

Estos proyectos contienen todos los recursos audiovisuales y de programación necesarios para ejecutar un programa Scratch.

Para mas información sobre Doctor Scratch y Hairball referirse a [1].

Capítulo 4

Solución

De cara a hacer un análisis masivo de múltiples programas Scratch, será necesario seguir los siguientes pasos:

- Obtención de código fuente Scratch
- Análisis del código fuente
- Modelado estadístico

Veamos detalladamente como se han llevado a cabo cada uno de estos pasos.

4.1. Obtención de Código Fuente Scratch

El lugar del cual obtener un número casi ilimitado de proyectos Scratch es el repositorio del Instituto de Tecnología de Massachusetts (MIT). Este repositorio es accesible a través de un navegador web, pues partiendo de una URL determinada es posible obtener los programas Scratch simplemente introduciendo aleatoriamente uno de los strings de la URL que vendría a representar el ID de cada proyecto.

El diagrama de flujo de la figura 4.1 explica la técnica de obtención de cada uno de los archivos *.sb2 que almacenan el código fuente de un proyecto de Scratch.

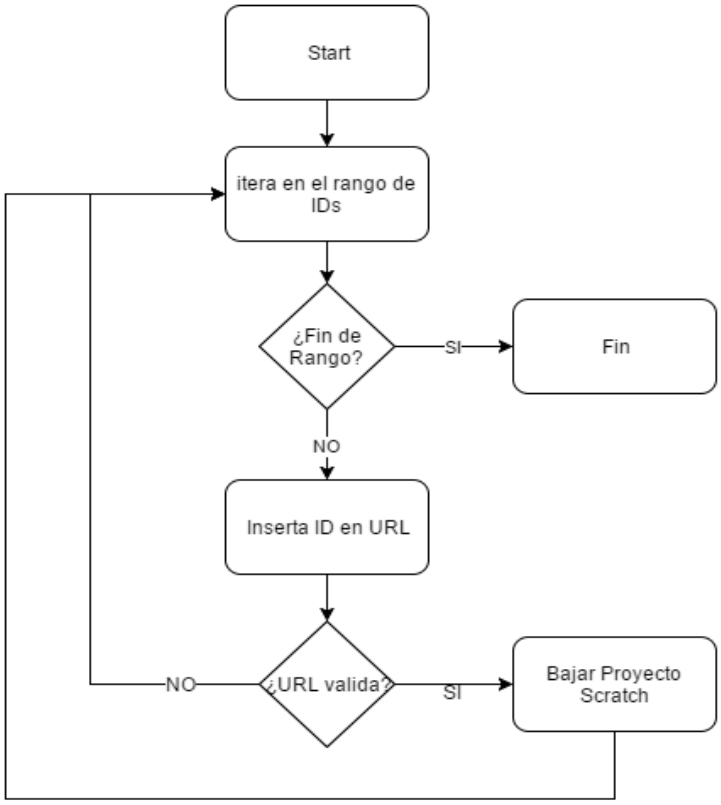

Figura 4.1: Diagrama de flujo del script de download

El método resumidamente es hacer GET HTTP probando IDs de forma aleatoria a partir de un rango numérico establecido.

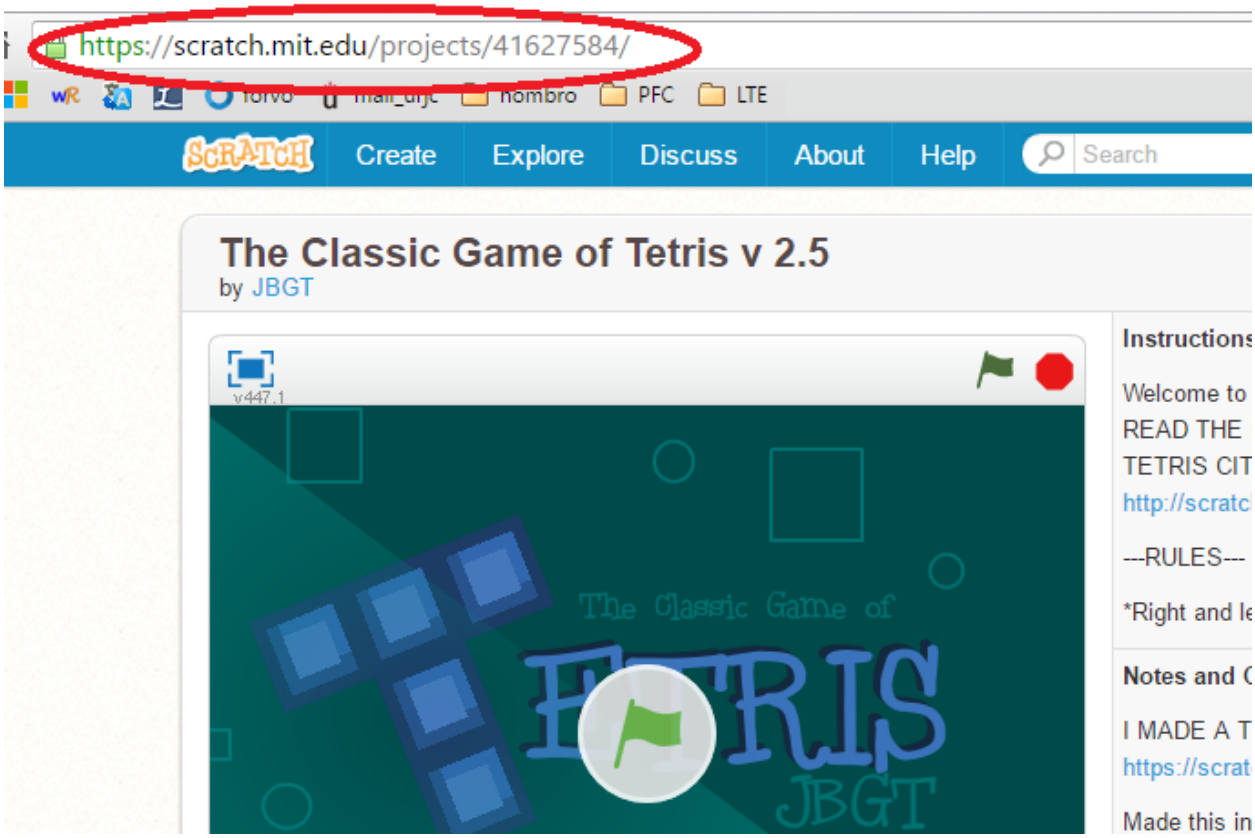

Figura 4.2: Ejemplo de URL de un proyecto Scratch del MIT

El script de la figura 4.3 muestra como obtener un programa Scratch a partir de un id (41627584 en este caso). La clave es enviar una petición con el id correspondiente a la url http://getsb2.herokuapp.com/. El resultado será la obtención de un fichero *.sb2, el cual hemos comentado que es un tar ball que contiene tanto el código fuente como los recursos audiovisuales del programa.

```
def sendRequestgetSB2(idProject):
        """First request to getSB2"""
        getRequestSb2 = "http://getsb2.herokuapp.com/" + idProject
        nameFile = idProject + '.sb2'
        outputFile = 'downloads/' + nameFile
        sb2File = urllib2.urlopen(getRequestSb2)
        output = open(outputFile, 'wb')
        output.write(sb2File.read())
        output.close()
        return nameFile
```

Figura 4.3: Script SendRequestGetSB2

Utilizando el script anterior como punto de partida, es posible añadir complejidad para poder bajarnos todos los proyectos Scratch dentro de un rango de identificadores determinado:

```
def main():

        for idProject in range(41627584, 59999999):

                idProjectSTR = str(idProject)
                print "processing %s ..." % idProjectSTR

                try:
                        fileName = sendRequestgetSB2(idProjectSTR)
                except:
                        print "id %s not found" % idProjectSTR

                time.sleep(5)
```

Figura 4.4: Script download main

De esta forma seleccionaríamos todos los proyectos Scratch dentro del rango 41627584 - 59999999. En el caso de este proyecto se ha bajado un total de 1332 proyectos Scratch para el análisis.

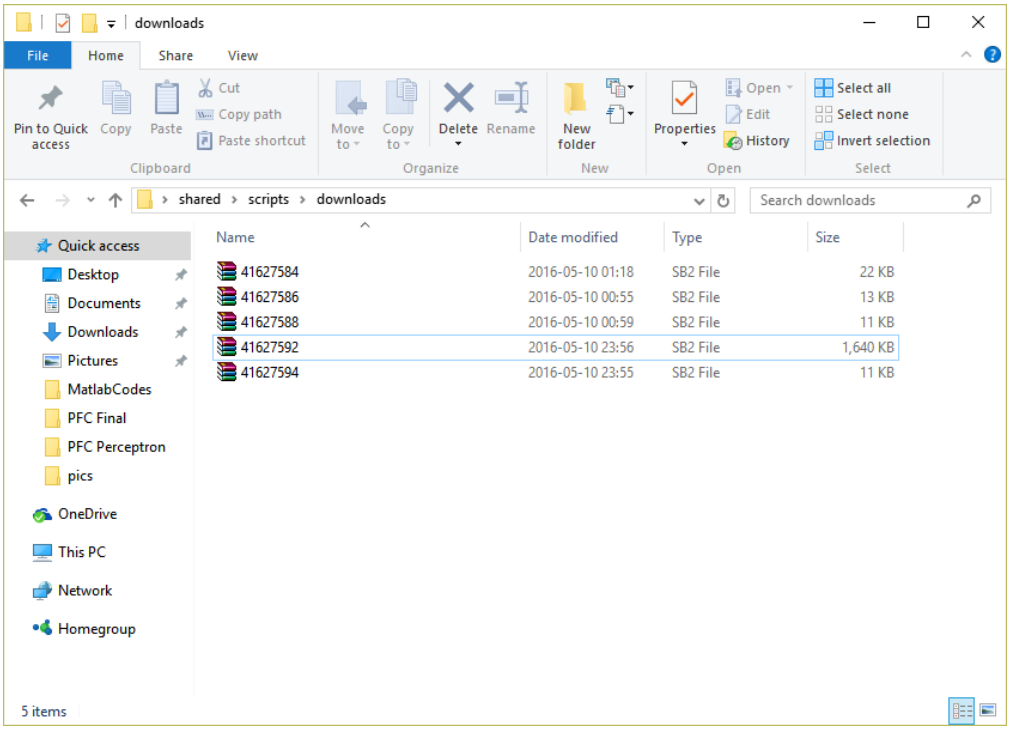

Figura 4.5: Proyectos .sb2 almacenados en carpeta

4.2. Análisis del Código Fuente

Para el análisis de los ficheros *.sb2 que nos hemos bajado utilizamos el script de la figura 4.6, el cual nos a partir de un path de fichero encontrará el *.sb2 que será analizado. Este análisis se hace con el comando Hairball, el cual obtiene una salida de texto con las correspondientes puntuaciones en pensamiento computacional. Esta salida de texto es parseada por nuestra función getDictio.

```
def getDictio(filename,directory):
#{'Abstraction': 0, 'Parallelization': 0, 'Logic': 0, 'Synchronization': 0
#, 'FlowControl': 0, 'UserInteractivity': 0, 'DataRepresentation': 0}
#Total mastery points: 0/21
#Average mastery points: 0.00/3
#Overall programming competence: Basic

        metricMastery = "hairball -p mastery.Mastery " + directory + "/" + filename
        resultMastery = os.popen(metricMastery).read()
        print "%s\n" % resultMastery
        lineone = resultMastery.splitlines()[1]
        #print "%s\n" % lineone
        dictio = ast.literal_eval(lineone)
        #print "%s\n" % dictio['Logic']
        line_total = resultMastery.splitlines()[2]
        line_total = line_total.split(": ",1)[1]
        line_total = line_total.split("/",1)[0]
        dictio['total_mastery'] = int(line_total)
        line_average = resultMastery.splitlines()[3]
        line_average = line_average.split(": ",1)[1]
        line_average = line_average.split("/",1)[0]
        dictio['average_mastery'] = float(line_average)
        line_competence = resultMastery.splitlines()[4]
        line_competence = line_competence.split(": ",1)[1]
        dictio['programming_competence'] = line_competence
        return dictio
```

Figura 4.6: Script analyze getDictio

El resultado del análisis es una puntuación para cada uno de los parámetros devueltos por el comando Hairball, véase el siguiente ejemplo.

{’Abstraction’: 1, ’Parallelization’: 1, ’Logic’: 3, ’Synchronization’: 1, ’FlowControl’: 2, ’UserInteractivity’: 2, ’DataRepresentation’: 2}

Total mastery points: 12/21

Average mastery points: 1.71/3

Overall programming competence: Developing

4.3. Análisis Estadístico

Para cada proyecto analizado, se ha almacenado el resultado en una fila de una matriz. Esta matriz ha sido creada con la librería de Python mas adecuada para el manejo de grandes estructuras de datos, la librería Pandas. Pandas crea una estructura de datos, la matriz de resultados, la cual es exportada a fichero para facilitar su almacenamiento y consulta. Por ser el formato mas común y además muy potente se ha elegido excel como formato del fichero final. Excel aparte de almacenar datos permite elaborar multitud de gráficos con mucha flexibilidad.

En la figura 4.7 se detalla el método main a través del cual se realiza este proceso.

```
def main():
        df_whole = pd.DataFrame()
        directory = "downloads"
        i = 0
        for filename in os.listdir(directory):
                if filename.endswith(".sb2"):
                        try:
                                dictio = getDictio(filename,directory)
                                print "%s\n" % dictio
                        except:
                                continue
                        df = pd.DataFrame(dictio,index=[i])
                        pieces = [df_whole, df]
                        df_whole = pd.concat(pieces)
                        #print "%s\n" % df_whole
                        print i
                        i=i+1
                        continue
                else:
                        continue

        dfToExcel(df_whole)
```

Figura 4.7: Script analyze main

El diagrama de flujo de la figura 4.8 explica este script y su secuencia de análisis para cada uno de los archivos *.sb2 dentro del directorio que almacena el código fuente de todos los proyectos de Scratch, bajados anteriormente con el script de download. En resumen se trata de un algoritmo que recorre un directorio en busca de ficheros *.sb2, los cuales son analizados uno a uno hasta escanear el directorio completo.

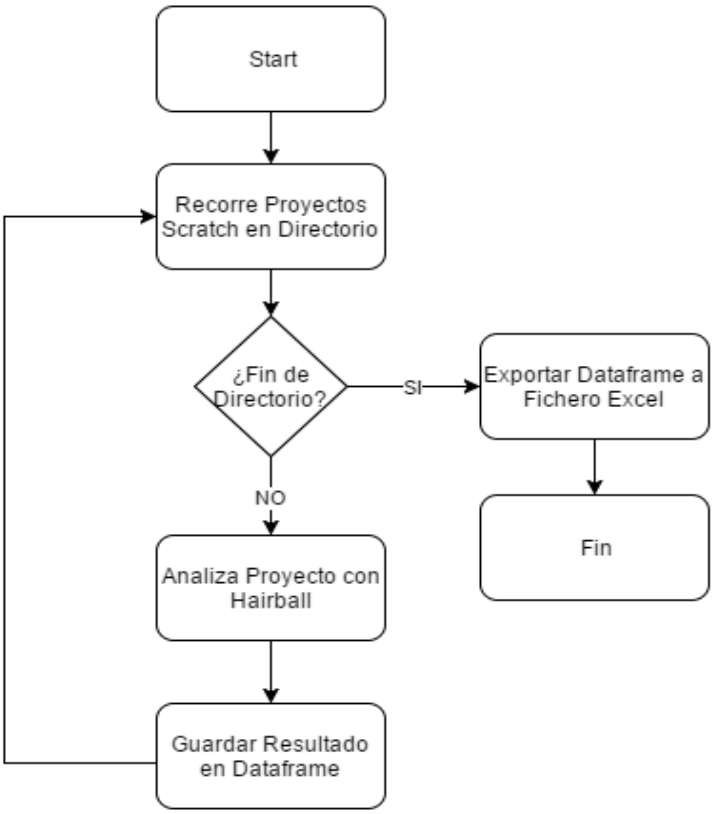

Figura 4.8: Diagrama de flujo del script de análisis

El script de la figura 4.9 obtiene como resultado un fichero excel donde se representan todos los parámetros que resultan del comando Hairball, quedando almacenado cada programa en las diferentes líneas de una matriz.

```
def dfToExcel(df):
        writer = pd.ExcelWriter('output.xlsx')
        df.to_excel(writer,'Sheet1')
        writer.save()
```

Figura 4.9: Script analyze dfToExcel

Además el script de la ficura 4.10 llamado myplot que será utilizado para dibujar los histogramas mediante la librería Pandas. Este script como primer paso importa los resultados desde un fichero excel para posteriormente dibujar los correspondientes gráficos.

```
import pandas as pd
import matplotlib.pyplot as plt
def main():
        xls_file = pd.ExcelFile('results.xlsx')
        df = xls_file.parse('Sheet1')
        df1 = df[['Abstraction']]
        df1.hist('Abstraction')
        plt.savefig('Abstraction.png')
        df2 = df[['DataRepresentation']]
        df2.hist('DataRepresentation')
        plt.savefig('DataRepresentation.png')
        df3 = df[['FlowControl']]
        df3.hist('FlowControl')
        plt.savefig('FlowControl.png')
        df4 = df[['Logic']]
        df4.hist('Logic')
        plt.savefig('Logic.png')
        df5 = df[['Parallelization']]
        df5.hist('Parallelization')
        plt.savefig('Parallelization.png')
        df6 = df[['Synchronization']]
        df6.hist('Synchronization')
        plt.savefig('Synchronization.png')
        df7 = df[['UserInteractivity']]
        df7.hist('UserInteractivity')
        plt.savefig('UserInteractivity.png')
        df8 = df[['average_mastery']]
        df8.hist('average_mastery')
        plt.savefig('average_mastery.png')
        writer = pd.ExcelWriter('outputtest.xlsx')
        df1.to_excel(writer,'Sheet1')
        writer.save()

if __name__ == "__main__":
    main()
```

Figura 4.10: Script myplot

El diagrama de flujo de la figura 4.11 explica la secuencia de acciones necesaria para el dibujado de cada uno de los histogramas. Resumidamente el script consiste en separar cada una de las columnas de la matriz en Dataframe diferentes, de modo que estas columnas pueden ser representadas en histogramas diferentes, viniendo a representar los distintos parámetros del pensamiento computacional analizados en este proyecto.

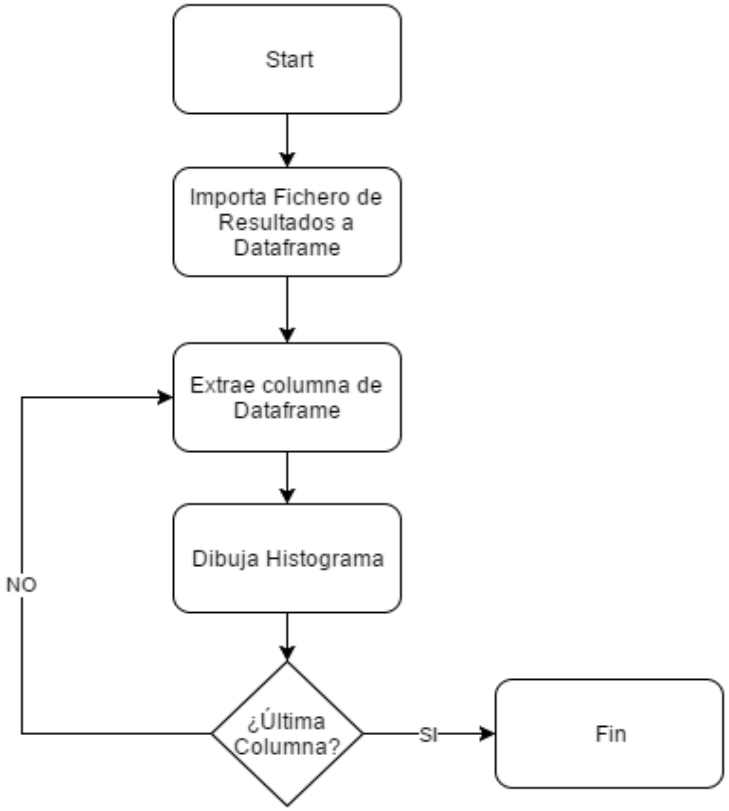

Figura 4.11: Diagrama de flujo del script de dibujo de gráficos

Capítulo 5

Resultados

5.1. Matriz de Resultados

El script para bajarse proyectos de Scratch funcionó correctamente y 1332 proyectos pudieron ser bajados para su posterior análisis.

El resultado del análisis es una matriz de 1332 filas creada con la librería Pandas, esta matriz almacena en distintas columnas los parámetros que miden la calidad de la programación de cada proyecto Scratch.

	Abstraction	DataRepresentation	FlowControl	Logic	Parallelization	Synchronization	UserInteractivity	average_mastery	programming_competence	total_mastery
0	0	0	0	0	0	0	0	0	Basic	0
1	1	2	2	1	3	2	2	1,86	Developing	13
2	0	0	0	0	0	0	0	0	Basic	0
3	1	2	2	1	2	0	2	1,43	Developing	10
4	1	2	1	3	1	2	2	1,71	Developing	12
5	1	2	2	2	3	2	1	1,86	Developing	13
6	1	1	2	1	3	2	2	1,71	Developing	12
7	1	2	2	1	1	1	2	1,43	Developing	10
8	1	0	1	0	1	0	1	0,57	Basic	4
9	0	1	2	1	0	0	2	0,86	Basic	6
10	1	1	1	0	0	0	2	0,71	Basic	5
11	3	2	3	3	3	3	2	2,71	Proficiency	19
12	0	1	1	2	0	2	2	1,14	Developing	8
13	1	2	3	3	2	2	2	2,14	Proficiency	15
14	0	0	0	0	0	0	0	0	Basic	0
15	0	1	0	0	0	0	0	0,14	Basic	1
16	0	1	2	0	0	1	1	0,71	Basic	5
17	0	0	2	0	0	0	1	0,43	Basic	3
18	1	1	2	0	1	0	1	0,86	Basic	6

Figura 5.1: Matriz de resultados

La figura 5.1 nos muestra la apariencia de la matriz de resultados. Podemos visualizar que cada columna representa cada uno de los parámetros o dimensiones a analizar de cara a medir el pensamiento computacional de un proyecto Scratch. Las puntuaciones obtenidas para cada parámetro están comprendida entre los valores 0 y 3 (0: Basic, 1: Developing, 2: Proficiency, 3: Expert), existiendo unas columnas llamadas average mastery y total mastery que muestran la media de todas las demás columnas y el valor total respectivamente. Cada una de las filas de la matriz representa una ocurrencia o entrada, dándose para cada una de ellas un valor en cada columna.

El siguiente paso es construir un gráfico a partir de estas columnas, de este modo podremos hacer un análisis conjunto y visual de los 1332 programas analizados en función de cada parámetro. A continuación se analizan los gráficos obtenidos.

5.2. Análisis de Gráficos

La abstracción en programación orientada a objetos se refiere a la creación de clases con una funcionalidad específica, de modo que las funciones de esta clase no están duplicadas cada vez que son necesarias dentro de nuestro código fuente. La figura 5.2 representa los 1332 valores obtenidos en el parámetro de abstracción, indica una puntuación mayoritaria dentro de los rangos básico y desarrollo, menos de 100 programas obtuvieron puntuaciones superiores a 2, lo cual indicaría una ausencia de dominio de la técnica de abstracción en la mayoría de los programas Scratch.

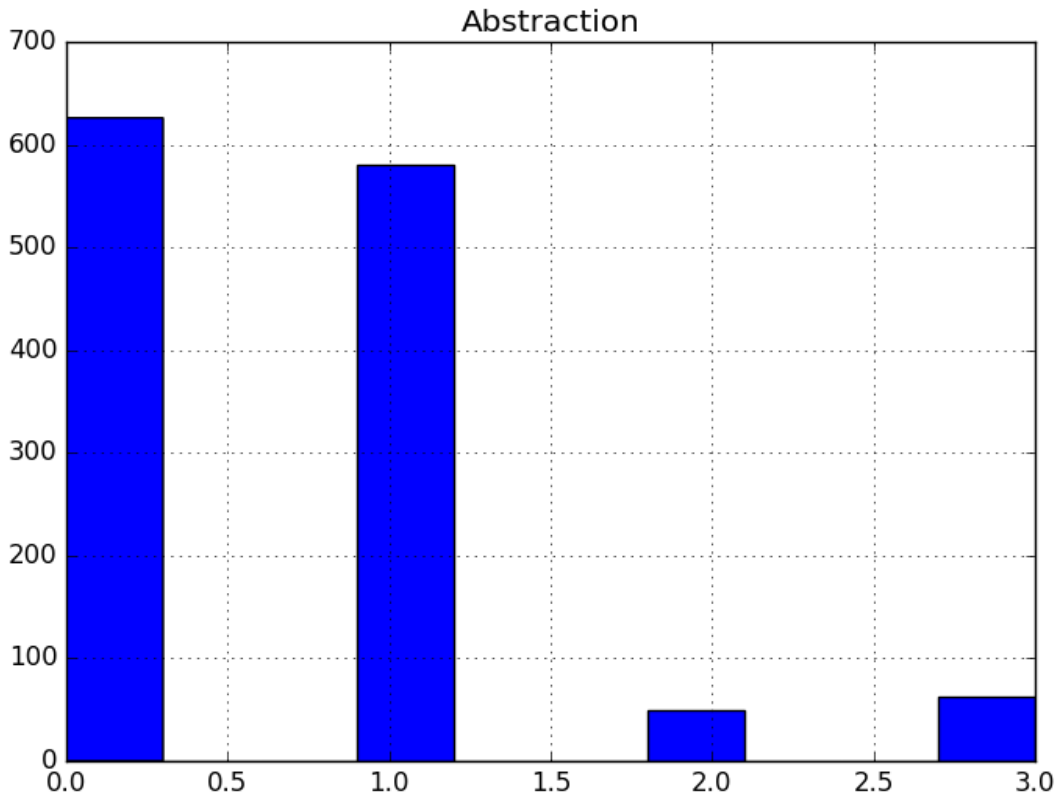

Figura 5.2: Abstracción

Data representation o representación de datos es la capacidad de dar el tipo de dato mas adecuado (string, integer, float...) a cada atributo o variable. La figura 5.3 representa los 1332 valores obtenidos en representación de datos, indica una puntuación mayoritaria dentro de los rangos básico y desarrollo, menos de 200 programas obtuvieron puntuaciones superiores a 2, lo

cual indicaría una ausencia de dominio de la técnica de representación de datos en la mayoría de los programas Scratch.

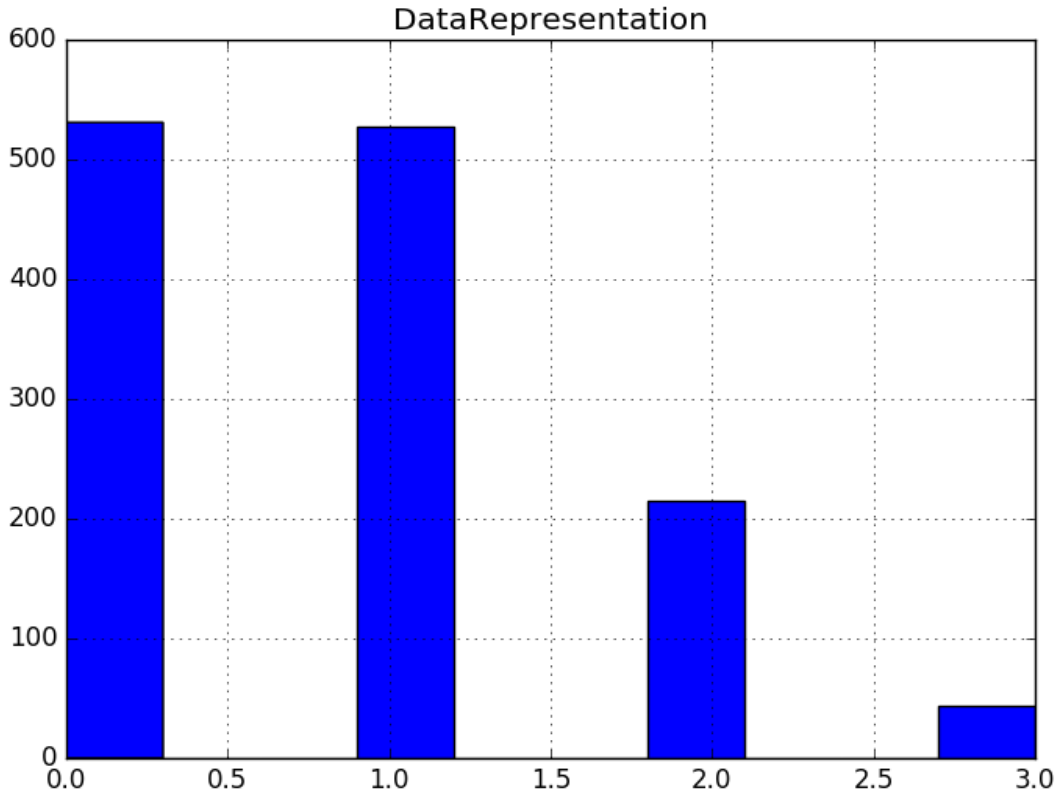

Figura 5.3: Representación de datos

Flow control o control de flujo hace referencia a la capacidad de un programa de tomar diferentes caminos en función del escenario concreto. La figura 5.4 representa los 1332 valores obtenidos en representación de datos, indica una puntuación levemente superior en los rangos básico y desarrollo, pero sin embargo prácticamente la mitad de los programas obtuvieron puntuaciones superiores a 2, lo cual indicaría un buen dominio de la técnica de control de flujo en aproximadamente la mitad de los programas Scratch.

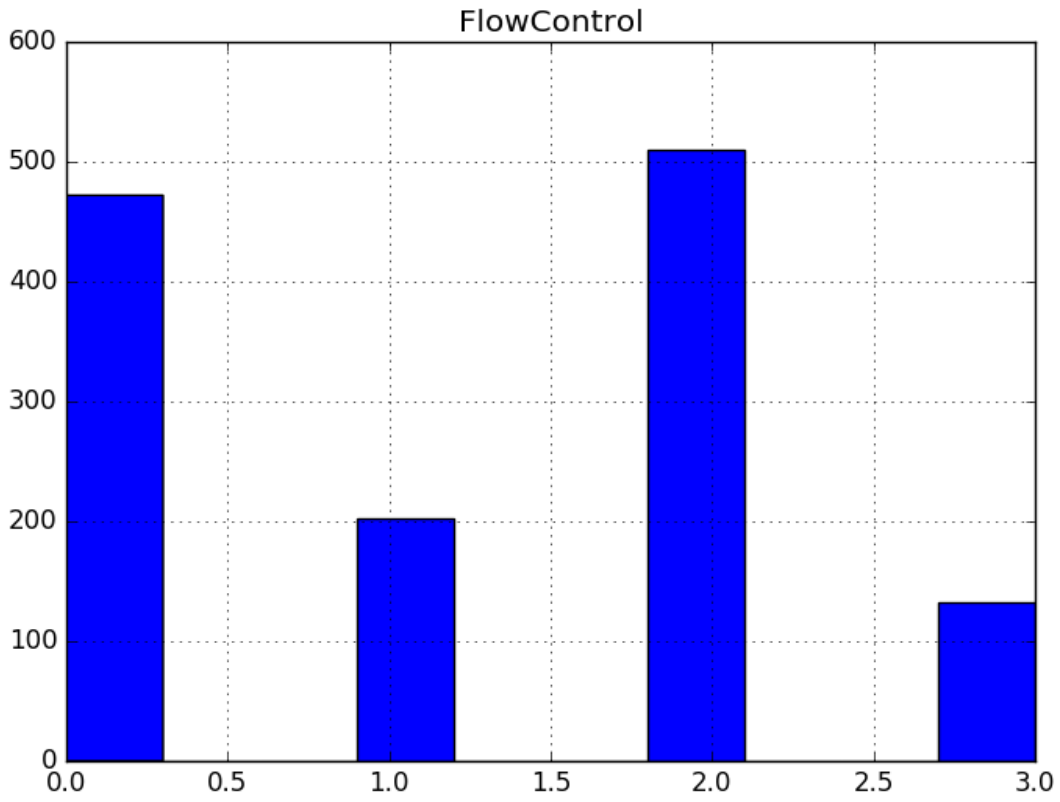

Figura 5.4: Control de flujo

Programación lógica es la resolución de un problema a través de un conjunto de hechos y reglas genéricas que cubren todos los escenarios y casuística de ese problema. La figura 5.5 representa los 1332 valores obtenidos en programación lógica, indica una puntuación mayoritaria dentro de los rangos básico y desarrollo, menos de 300 programas obtuvieron puntuaciones superiores a 2, lo cual indicaría una ausencia de dominio de la técnica de programación lógica en la mayoría de los programas Scratch.

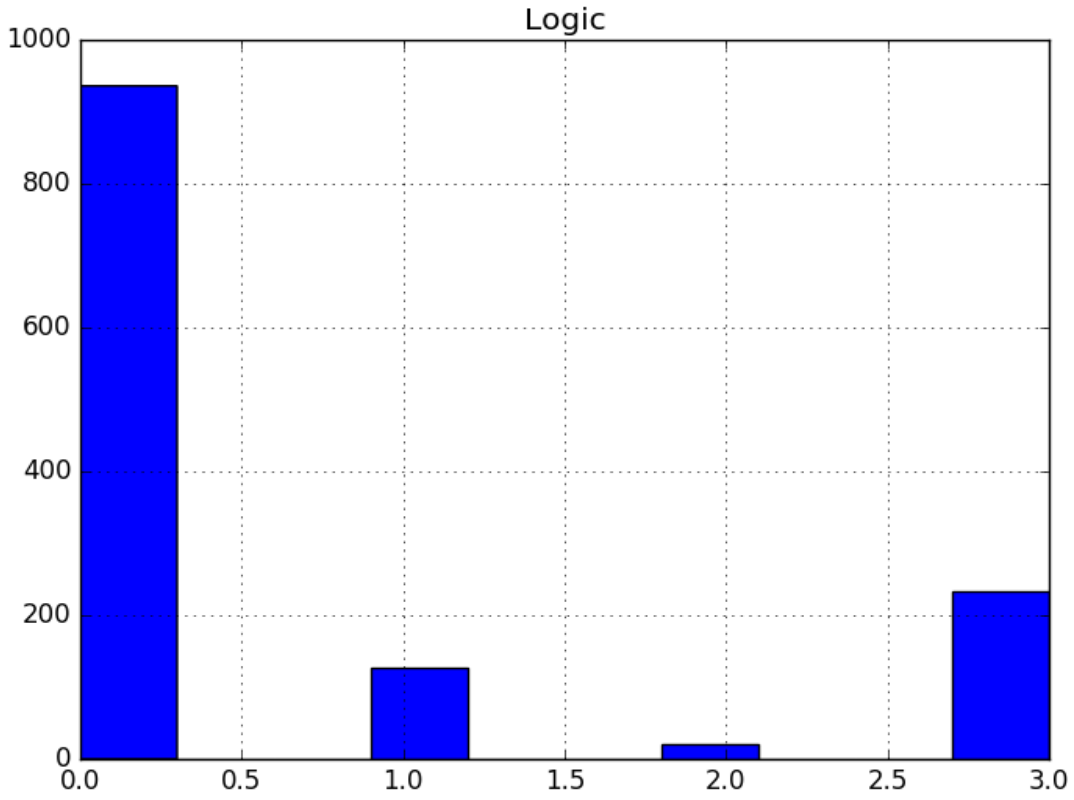

Figura 5.5: Lógica

Paralelización en programación es la reducción de un problema en varios subproblemas mas peque?os que pueden ser realizados al mismo tiempo. La figura 5.6 representa los 1332 valores obtenidos en Paralelización, indica una puntuación mayoritaria dentro de los rangos básico y desarrollo, menos de 300 programas obtuvieron puntuaciones superiores a 2, lo cual indicaría una ausencia de dominio de la técnica de Paralelización en la mayoría de los programas Scratch.

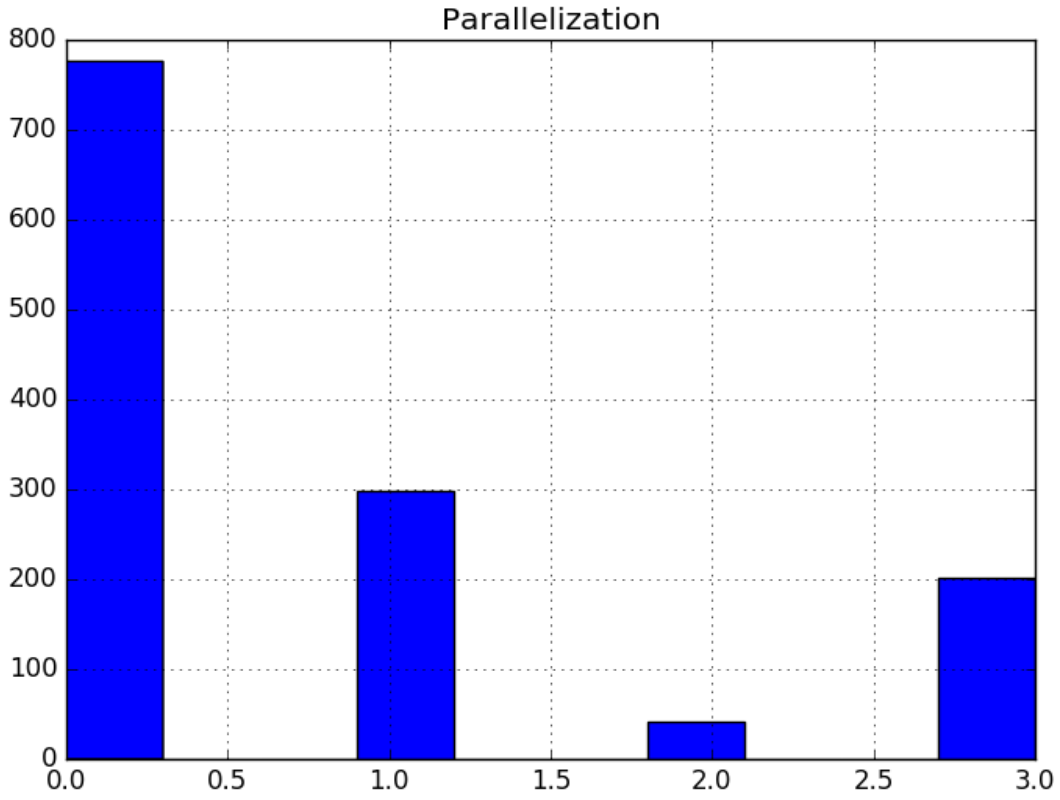

Figura 5.6: Paralelización

Sincronización hace referencia a la capacidad de evitar que múltiples hilos de un programa accedan simultáneamente a secciones de código críticas. La figura 5.7 representa los 1332 valores obtenidos en Sincronización, indica una puntuación mayoritaria dentro de los rangos básico y desarrollo, menos de 500 programas obtuvieron puntuaciones superiores a 2, lo cual indicaría una considerable ausencia de dominio de la técnica de Sincronización en la mayoría de los programas Scratch.

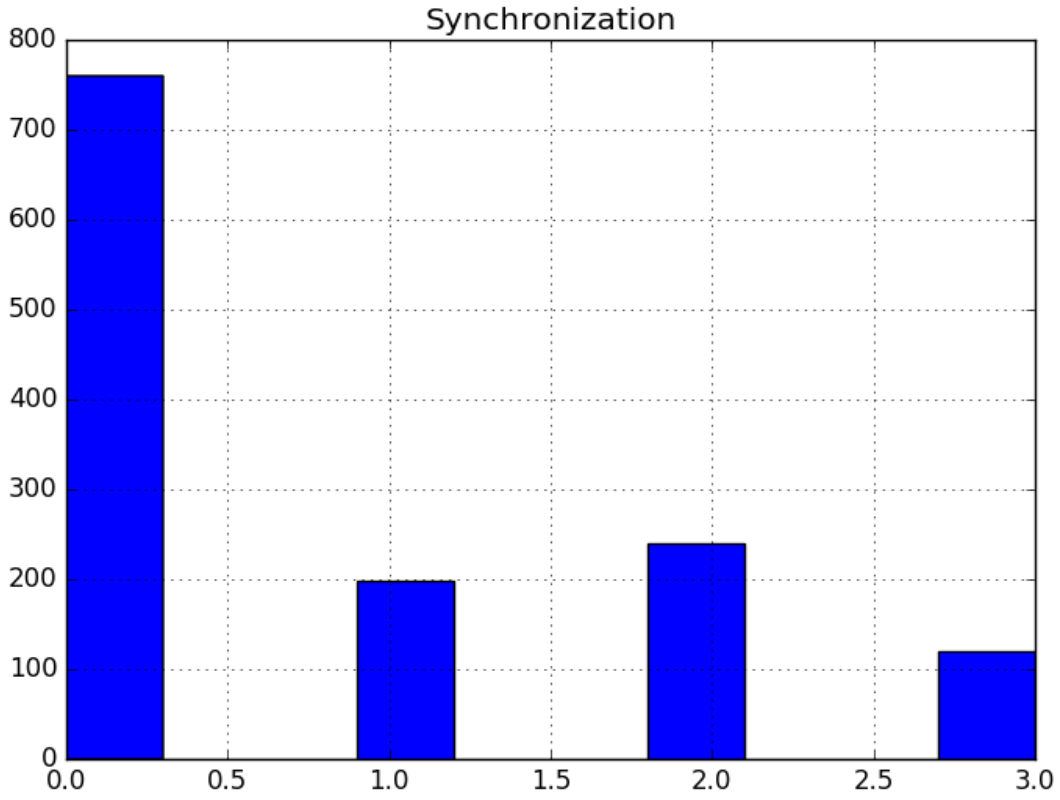

Figura 5.7: Sincronización

User interactivity o interactividad de usuario es la capacidad de un programa de leer entradas de un operador humano. La figura 5.8 representa los 1332 valores obtenidos en interactividad, indica una puntuación mayoritaria dentro de los rangos básico, desarrollo, únicamente un tercio de los programas analizados obtuvieron una puntuación superior que denota un mayor uso de la técnica de interactividad.

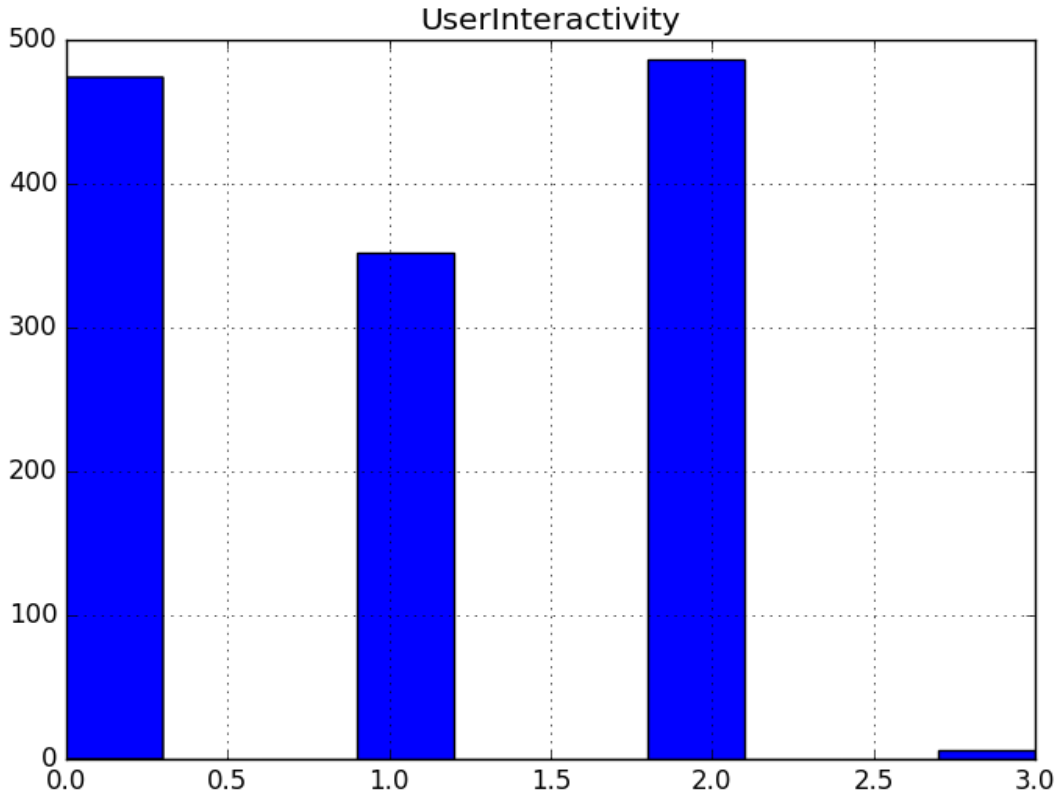

Figura 5.8: Interactividad de usuario

A continuación en la figura 5.9 se muestra el parámetro conocido como Average mastery o maestría media, tal como su nombre indica, mide el valor medio del resto de columnas. El gráfico nos muestra el revelador dato de que prácticamente la mitad de los programas analizados tiene una puntuación media por debajo de 0.5, dato que indica un nivel de programación muy bajo en gran parte de los programas Scratch.

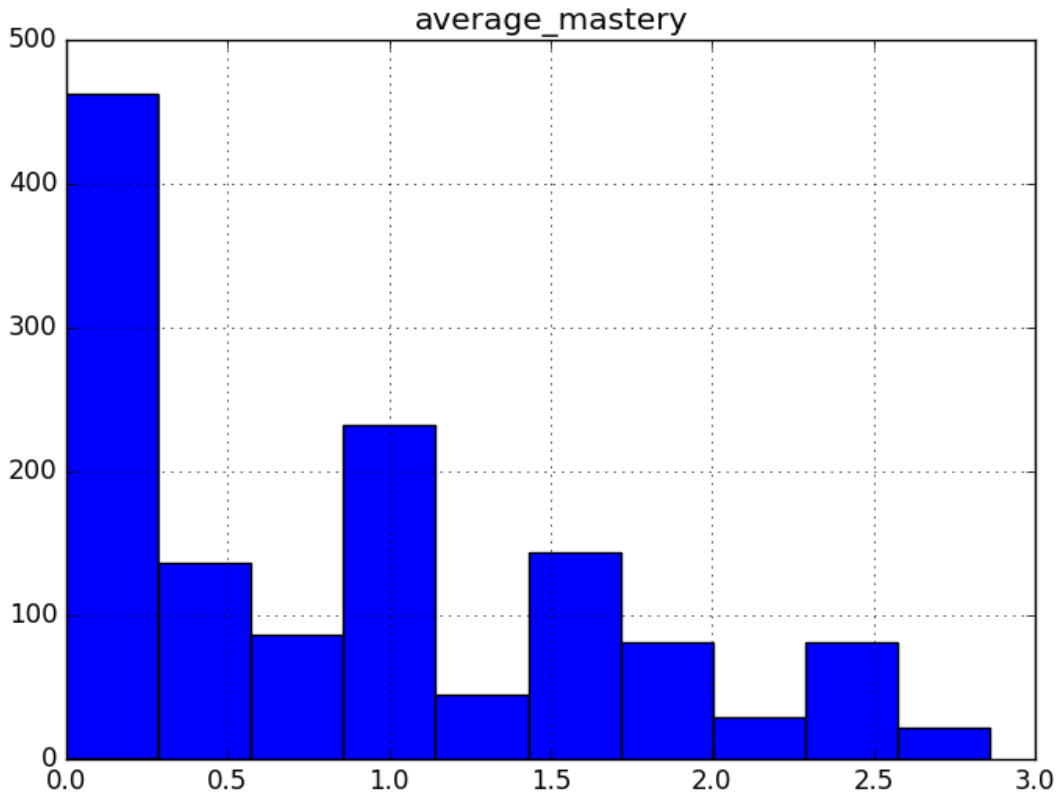

Figura 5.9: Average mastery

Capítulo 6

Conclusión y Futuras Líneas

Scratch es usado por estudiantes, profesores y padres para crear animaciones de forma sencilla y servir de trampolín hacia el más avanzado mundo de la programación. Puede usarse para un gran número de propósitos educativos y de entretenimiento, tales como proyectos de ciencias, matemáticas, simulación y visualización de experimentos, presentaciones animadas, arte interactivo, música, etc. Precisamente por su polivalencia, se intuye que muchos de los programas realizados en Scratch no tienen como finalidad didáctica una mejora de la calidad de la programación.

Nuestro análisis estadístico precisamente viene a constatar ese hecho, que una gran mayoría de los programas de Scratch han sido programados con poca calidad en lo que a materia de programación se refiere. Esto es debido principalmente a dos factores: 1. Los programas realizados en Scratch son realizados por personas muy inexpertas, por lo que la calidad de la programación es baja, 2. Un gran número de programas Scratch no tienen como fin didáctico la mejora de la programación, esta es simplemente un medio a través del cual conseguir otros objetivos didácticos (matemáticas, ciencias, etc.), hecho que viene a explicar que muchos programas Scratch no hagan ningún hincapié en los paradigmas de la programación analizados por este proyecto.

Como línea futura de investigación sería interesante añadir mas parámetros a nuestra base de datos, tales cómo la edad o el ámbito académico de la persona que realizo el programa. Esto abriría líneas de trabajo que permitirían un estudio del desarrollo de las capacidades de programación en función con la edad y maduración de la persona, o bien en relación al grupo social o entorno. Esto permitiría comprobar si se cumpliría el tópico que establece que las personas que estudian matemáticas o ciencias desarrollan mas cualidades para programar.

Capítulo 7

APÉNDICE: ENTORNO DE PRUEBAS

El proyecto fue desarrollado con un ordenador portátil Windows x86 de gama media. Para la ejecución del entorno Linux se han utilizado máquinas virtuales de typo Linux Mint 18 y Ubuntu 14.04. Ambas máquinas virtuales cuentan con las librerías Python 2.7.X preinstaladas. Para correr las máquinas virtuales en el ordenador, fue instalado el software hipervisor de VMware y Oracle Virtual Box.

Capítulo 8

Apéndice: Manual de Usuario

Es necesario seguir las instrucciones de instalación de VMware o de Oracle Virtual Box, en función del hipervisor elegido. Una vez hecha la instalación ya es posible editar y ejecutar los scripts de Python con la shell preinstalada de Linux (Konsole típicamente). Los scripts a ejecutar serían los de la figura 8.1:

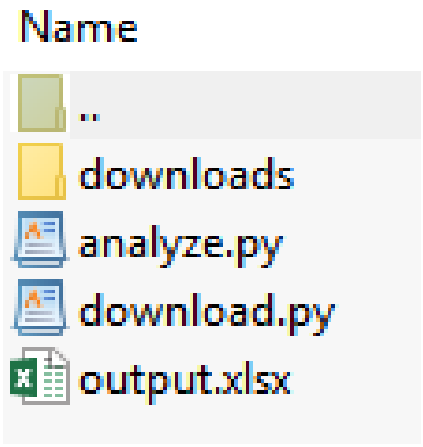

Figura 8.1: Scripts de ejecución

Siendo necesario que el ordenador tenga acceso a internet para poder acceder al repositorio web del MIT.

Capítulo 9

APÉNDICE: CÁLCULO DE PRESUPUESTO

9.1. Planificación

Fase 1: Análisis y definición de requisitos	1 semanas
Fase 2: Adaptación al entorno y aprendizaje del estado del arte	1 semanas
Fase 3: Diseño de la solución	1 semanas
Fase 4: Implementación	2 semanas
Fase 5: Testeo	1 semanas
Fase 6: Registro de conclusiones	2 semanas

9.2. Recursos

Cargo	**Horas Hombre**
Consultor	160
Técnico Informático	2
Desarrollador de SW	80
Quality Assurance	40

Listado de Material	**Unidades**
PC	3
Aplicación SW	8

9.3. Precios Unitarios

Cargo	**Euro/hora**
Consultor	25
Técnico Informático	10
Desarrollador de SW	20
Quality Assurance	15

Listado de Material	**EURO**
PC	700
Aplicación de SW	90

9.4. Precio Total

Item	**P*Q**
Consultor	25 eur *160 hora = 4000
Técnico Informático	10 eur *2 hora = 20
Desarrollador de SW	20 eur *80 hora = 1600
Quality Assurance	15 eur *40 hora = 600
PC	700 eur *3 unidades = 2100
Aplicación de SW	90 eur *8 unidades = 720
Total	**9.040 euros**

Bibliografía

[1] Doctor Scratch. http://drscratch.programamos.es/ , https://github.com/jemole/drScratch

[2] Lenguaje Scratch, Wikipedia. https://es.wikipedia.org/wiki/Scratch

[3] Lenguaje Scratch, MIT. http://scratch.mit.edu/projects/

[4] Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional http://www.um.es/ead/red/46/morenorobles.pdf

[5] Python página oficial. https://www.python.org

[6] Librería Pandas http://pandas.pydata.org/