

# Boca Analytics - Avance 2

Instituto Tecnológico de Costa Rica  
Escuela de computación  
Ingeniería en computación  
Aseguramiento de la calidad de software

Joshua Mata Araya  
Adrián López Quesada  
Josué Arrieta Salas

13 de octubre de 2016

## Índice

<b>1. Validación de diseño</b>	<b>1</b>
<b>2. Herramienta para verificar estándar de codificación</b>	<b>3</b>
2.1. Java . . . . .	3
2.2. Javascript . . . . .	4
2.3. AngularJS . . . . .	5
<b>3. Cómo obtener la versión actual del sistema</b>	<b>5</b>
3.1. Buscar la versión que deseamos descargar . . . . .	5
3.2. Descargar el repositorio de la versión que se quiere . . . . .	6
3.3. Seguir el readme . . . . .	6
<b>4. Cuantificación de dos métricas</b>	<b>7</b>
<b>5. Pruebas unitarias</b>	<b>7</b>
5.1. P01 - compareImage . . . . .	8
5.2. P02 - FieldDetectorTest . . . . .	8
5.3. PlayerDetectorTest . . . . .	8
5.4. CountFramesTest . . . . .	9
5.5. LargeVideoTest . . . . .	9

## 1. Validación de diseño

A continuación se presenta una tabla en la cual se valida la arquitectura del sistema contra los requerimientos. Para más detalles de los requerimientos y su prioridad, se puede consultar el [documento](#) en la sección 2.

ID	Requerimiento	Elemento de diseño
1	Se podrá cargar un video de fútbol para ser analizado	Implementado a través del api y junto con UI
2	Se podrá descargar un video de fútbol luego de ser analizado	Implementado a través del api y junto con UI
3	En el video descargado se visualiza los blobs correspondientes a jugadores	Para esto es necesario el 90 % del sistema, el único artefacto no usado para cumplir esta función es la clase CSV Handler
4	En el video descargado se visualiza junto a cada blob una etiqueta correspondiente a cuál equipo pertenece el jugador	Se usa el módulo del sistema KMeans y la función setTag en específico
5	Se muestra en el video la cantidad de jugadores detectados en cada cuadro	K means será el encargado de esto, en el método getCountPlayers
6	Se generan reportes de la cantidad de jugadores en cada equipo, por frame	Módulo CSV Handler
7	Se podrá visualizar el tiempo restante en procesar el video	Patrón observador implementado en el API
8	Se podrá cargar un archivo ground truth: el sistema debe analizarlo para cuantificar la cantidad de fallos encontrados	Métrica dice implementada en el método dice de la clase OpenCVImageProcessor en el módulo Imageprocessor
9	Se podrá cargar un video de futbol para ser analizado	Se carga a través del UI con el API y es analizado por los módulos ImageProcessor y
10	Se espera que la cantidad de requerimientos implementados sean un 90 % de la cantidad de requerimientos establecidos	La arquitectura presenta la oportunidad de implementar el 100 % de los requerimientos establecidos
11	La cantidad de errores en las pruebas unitarias debe ser menor o igual a un 10 %	Esto no se refleja en la arquitectura
12	Tiempo requerido por usuario en usar el programa por primera vez debe ser menos de 160 segundos	Esto se debe validar a través del módulo de UI
13	El programa debe mostrar mensajes significativos y resultados durante toda su ejecución para que el usuario entienda qué está pasando	Patrón de diseño observador entre el api y logic
14	El programa posee una interfaz de usuario consistente entre sí	Módulo de UI
15	Se espera que el procesamiento requerido por frame sea menos de 20ms	Test sobre el método getPlayers(), pero no se ve reflejado en el diagrama
16	Se espera que la utilización máxima de memoria del programa sea como máximo 5 %	Test sobre cada que herede de controller, pero no se puede ver en el diagrama
17	El índice de complejidad ciclomática se espera que este igual o por debajo de 10	Métrica obtenida sobre el módulo de logic
18	La cobertura de las pruebas de código debe ser de al menos un 80 %	Métrica obtenida sobre el módulo de logic
19	El índice de mantenibilidad debe ser de al menos un 80 %	Métrica obtenida sobre el módulo de logic

ID	Requerimiento	Elemento de diseño
20	La cantidad de elementos probados del software debe ser mayor a un 85 %	Métrica obtenida sobre el módulo de logic
21	El sistema debe por cada fallo sucedido, recuperarse en un 95 % de las veces	Manejo de excepciones sobre el paquete controller
22	El tiempo que el sistema logra estar sin caerse un 97 % del tiempo que este en ejecución	Metrica evaluada sobre todos los módulos del sistema
23	Se necesita un sistema con Tomcat 8.0	Requerimiento eliminado
24	Se necesita un sistema con OpenCV-3.1	Paquete de logic
25	Se necesita un sistema con JDK 8.1 o superior	Módulos Logic y API
26	Se espera que la página web sea compatible con al menos 3 navegadores diferentes con aprobación mayor a un 90 %	Módulo de UI
27	Se espera que el tiempo de instalación, usando documentación interna solamente, sea en un 75 %, menor a 45 minutos	No corresponde a la arquitectura
28	Se espera que el sistema tenga una coexistencia aprobada con otros sistemas al menos un 90 % del tiempo	Eficiencia de operación de los módulos API y Logic
29	Se espera que la corrupción de datos, solo se presente como máximo un 10 %	Módulo de Logic y de API
30	Se debe realizar una copia de seguridad de todo el servidor de manera diaria	Módulo de API
31	Todo tipo de video que reciba el servidor, debe ser def ormato .mp4, .webm, .ogg y debe pesar como máximo 200mb	Validación del módulo UI

## 2. Herramienta para verificar estándar de codificación

Según el documento de la [especificacion](#) (Sección 3, pag 15-16) se decidieron utilizar diferentes estándares. A continuación se explica como se evalúa cada uno para su cumplimiento.

### 2.1. Java

Para Java, se decidió utilizar el estándar de Google para el código. Para no tener que aprenderse de memoria el estándar y revisar que todos los miembros de equipo lo cumplieran, se utilizó un plugin de Eclipse llamado CheckStyle. Es posible ver la instalación de la herramienta por medio del [documento](#) en al carpeta de herramientas, el cual contiene todas las instalaciones de la herramientas utilizadas durante el proyecto, que no cuentan como dependencias.

Si se sigue el documento de instalación al pie de la letra se podrá hacer la verificación de estándar de codificación, sin embargo, a continuación se mostrara como se hace.

```

16 @Test
17 public void countFramesTest() {
18     OpenCVVideoProcessor vp = new OpenCVVi
19     int expected = 132;
20     int actual = vp.getFrameCount();
21     assertEquals(expected, actual);
22 }
23 @Test
24 public void createVideoTest(){
25     OpenCVVideoProcessor vp = new OpenCVVi
26     int frames = vp.getFrameCount();
27     while(frames>0){
28         Mat frame = (Mat) vp.readFrame();
29         vp.writeFrame(frame);
30         frames--;

```

En la imagen anterior se puede observar como todas las líneas están subrayadas con fondo amarillo, así es como CheckStyle avisa que una línea tiene algo malo según el estándar. Además, a la izquierda se puede observar una lupa que al presionarla sale una pequeña ayuda del error.

Como se aplicó esta herramienta es posible que no se encuentren errores, por lo que se recomienda que se modifique el espaciado de una línea y se guarde el código, para que CheckStyle vuelva a revisar y en caso de que esté bien instalado, marcará un error en la línea.

Al final de cuentas, se debe ir .java por .java para verificar que no haya ninguna línea amarilla que simbolice que no se siguió el estándar.

## 2.2. Javascript

Con respecto a Javascript, se utilizó el sitio web [JSLint](#). Respecto al uso de esta herramienta, se intentó acoplar lo más posible a el estándar asignado, sin embargo hay algunas excepciones. Como la herramienta está diseñada para evaluar javascript puro, al usar funcionalidades de AngularJS, esta dio algunos errores, los cuales se dan por seguir el estándar de AngularJS el cual tiene más importancia.

The screenshot shows the JSLint web interface. At the top, there's a text area with JavaScript code. Below it are buttons for 'JSLint' and 'clear'. A 'Warnings' section lists several issues:

- This function needs a 'use strict' pragma. (Line 1.25)
- Unused 'event'. (Line 23.37)
- Use spaces, not tabs. (Line 24.1)

En la imagen anterior, se observa en la parte superior el código que se está analizando, se debe hacer un copiar el código y pegar en esta área. Abajo se observan algunos errores, como en event no utilizado o mal uso de tabs.

Options			
<b>Assume...</b> <input type="checkbox"/> in development <input type="checkbox"/> ES6 <input checked="" type="checkbox"/> a browser <input type="checkbox"/> CouchDB <input type="checkbox"/> Node.js	<b>Tolerate...</b> <input type="checkbox"/> bitwise operators <input type="checkbox"/> eval <input type="checkbox"/> for statement <input type="checkbox"/> multiple vars <input type="checkbox"/> single quote strings <input type="checkbox"/> this <input type="checkbox"/> whitespace mess	<b>Number...</b> <input type="text"/> Maximum line length <input type="text"/> Maximum warnings	<b>Global variables...</b> <input type="text"/> angular <input type="text"/> WebSocket <input type="text"/> FileReader <input type="text"/> ArrayBuffer
<b>Fudge...</b> <input type="checkbox"/> First line number is 1			

La pagina tiene una sección inferior, en donde se puede configurar algunas secciones, se debe marcar en la columna de Assume: a browser y se deben poner la variables globales: angular WebSocket FileReader ArrayBuffer.



Esta ultima imagen muestra el resultado final, los warnings salen por cumplir el estándar de angular. Por ejemplo, los 'out of scope' se dan al definir las variables a algo que esta después de su definición, sin embargo así lo define el estándar de Angular. La directiva '\$digest' la toma como una propiedad incorrecta por su estándar de nombres para las propiedades y finalmente, al tener una función Control sin ninguna definición, pide que se utilice el 'use strict' sin embargo, tampoco se debe poner por el estándar de Angular.

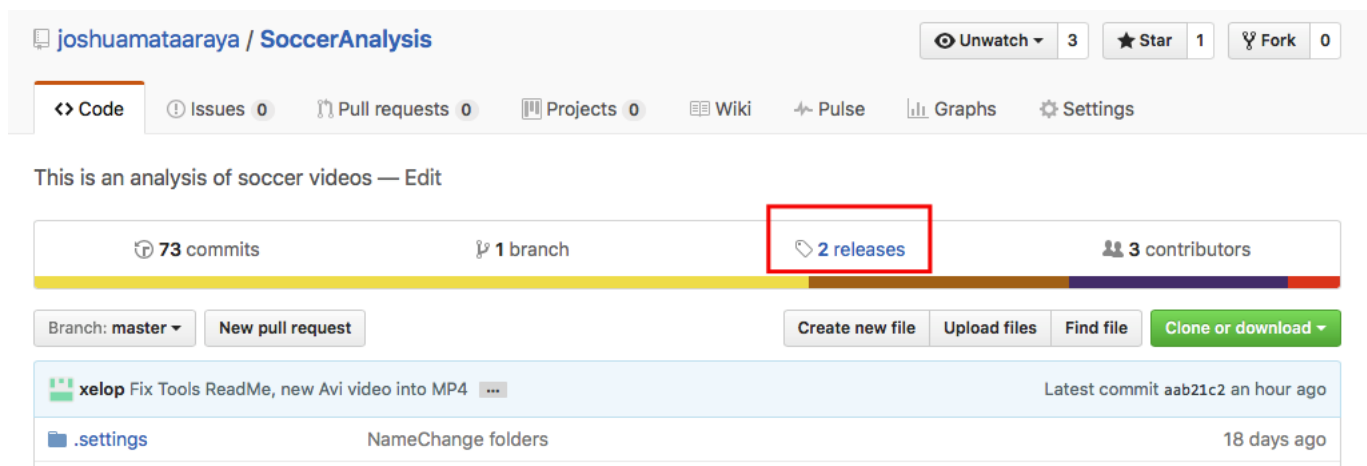
## 2.3. AngularJS

Para angular no se encontró una herramienta para su evaluación automática, sin embargo, se tomo en cuenta la extensión del estándar y como la mayoría de las especificaciones ayudan a la programación en Angular ademas de mejorar el estilo de código. Por lo tanto, seria bueno buscar una herramienta que automatice este proceso pero se considero oportuno que los miembros del proyecto leyeran con detenimiento el estándar y lo aplicaran de manera manual para su aprendizaje.

## 3. Cómo obtener la versión actual del sistema

Para obtener la versión más actual del sistema hay que seguir los siguientes pasos:

### 3.1. Buscar la versión que deseamos descargar



### 3.2. Descargar el repositorio de la versión que se quiere

Pre-release

v0.1.1-alpha

9adeac9

## Project Configuration Defined

joshuamataaraya released this 18 days ago · 29 commits to master since this release

### Características Agregadas

- Archivos necesarios nuevos agregados y eliminados para tener una correcta configuración del software en el proyecto.

### Downloads

Source code (zip)

Source code (tar.gz)

Pre-release

v0.1.0-alpha

65fad10

## Initial implemented features

joshuamataaraya released this 19 days ago · 40 commits to master since this release

### Características Agregadas

- Detección de jugadores (Lógica)
- Detección del campo de juego (Lógica)
- Subir videos (UI)

### 3.3. Seguir el readme

En cada versión del sistema se incluye un archivo Readme.md que contiene las instrucciones necesarias para establecer el ambiente de desarrollo de ese momento.

xelop Fix Tools ReadMe, new Avi video into MP4 ...	
.settings	NameChange folders
Architecture	Architecture diagrams
Dependencies	Merge remote-tracking branch 'origin/master'
ImagesReadme	ReadMe for webServer
Implementation	Merge remote-tracking branch 'origin/master'
Requirements	Merge remote-tracking branch 'origin/master'
Tools	Fix Tools ReadMe, new Avi video into MP4
.DS_Store	Tasks Report
.classpath	NameChange folders
.gitignore	NameChange folders
.project	NameChange folders
README.md	ffmpeg installation
Tasks Zoho.xls	Tasks Report
VideoAvi.mp4	Fix Tools ReadMe, new Avi video into MP4

## 4. Cuantificación de dos métricas

Según el [Documento](#) de métricas y requerimientos, se escogió analizar para este sprint la métrica de mantenibilidad de capacidad de ser analizado, pagina 6, sección 5.2.

Para esto se utilizo el plugin de Eclipse, Java Metrics 2, la cual muestra una gran cantidad de métricas en todo el proyecto. En el [documento](#) de herramientas del repositorio se puede ver como instalar y utilizar esta herramienta. Según la métrica, se evalúa el índice de complejidad ciclomática, la formula se puede ver en el [pagina](#) 5, sección 5. Es una formula compleja para calcular en muchos .java pero la herramienta lo calcula y muestra lo siguiente.

Metric	Total	Mean	Std. Dev.	Maximum
▼ java		1.22	0.746	6
▼ api		1.909	1.443	6
> WebSocketServlet.java		2	1.483	6
> LargeFileUploadTest.java		1	0	1
▼ controller		1.667	1.054	4
> GroundTruthController.java		2.5	1.5	4
> VideoAnalysisController.java		2	1	3
> Controller.java		1.25	0.433	2
> VideoAnalysisControllerTest.java		1	0	1
> GroundTruthControllerTest.java		0	0	
▼ logic.imageprocessor		1.13	0.579	4
> FieldDetector.java		1.5	1.118	4
> OpencvImageProcessor.java		1.267	0.772	4
> Detector.java		1	0	1
> PlayerDetectorTest.java		1	0	1
> PlayerDetector.java		1	0	1
> OpencvDetector.java		1	0	1
> ImageProcessor.java		1	0	1
> FieldDetectorTest.java		1	0	1
> ImageProcessorTest.java		1	0	1
▼ logic.videoprocessor		1.048	0.213	2
> OpenCvVideoProcessorTest.java		1.5	0.5	2
> OpenCvVideoProcessor.java		1	0	1
> VideoProcessor.java		1	0	1

En la imagen se puede ver como la mayoría de las clases están abajo de 10, el cual era el máximo establecido. Además podemos ver que la mayoría tiene un índice ciclomático de 1 y el máximo que se llega es a 6.

En el mismo [Documento](#) de métricas y requerimientos, se escogió analizar para este sprint la métrica de funcionalidad de exactitud, pagina 1, sección 1.2.

Para esto se utilizó la funcionalidad de JUnit, donde se puede poner a correr todas las pruebas y ver la cantidad de errores por cada una, esto dio el siguiente resultado:

## 5. Pruebas unitarias

Si se quiere ver la versión más actualizada del diseño, acceder al siguiente [link](#). Las salidas de cada prueba serán mostradas a tal vez de Criterios de Éxito y criterios de fallo. Al momento se tienen los siguientes planes de prueba:

### 5.1. P01 - compareImage

<b>Fecha</b>	12/10/2016
<b>Encargado</b>	Josué Arrieta Salas
<b>Objetivo</b>	Probar la funcionalidad del método compareImage() de la clase OpencvImageProcessor.
<b>Descripción</b>	Este método devuelve True si recibe 2 imágenes que son exactamente iguales; de lo contrario devuelve False.
<b>Tipo</b>	Prueba Unitaria - Caja negra
<b>Entradas</b>	Se tiene de entrada la imagen <a href="#">image1</a> .
<b>Criterio Éxito</b>	Se compara la imagen; con ella misma con el método. Devuelve True, pasa la prueba unitaria.
<b>Criterio Fallo</b>	Se compara la imagen; con ella misma con el método. Devuelve False, no pasa la prueba unitaria.

### 5.2. P02 - FieldDetectorTest

<b>Fecha</b>	12/10/2016
<b>Encargado</b>	Josué Arrieta Salas
<b>Objetivo</b>	Probar la funcionalidad del método detect() de la clase FieldDetectorTest.
<b>Descripción</b>	Este método debe detectar la cancha de una imagen de un partido de fútbol.
<b>Tipo</b>	Prueba Unitaria - Caja negra
<b>Entradas</b>	Se tiene de entrada la imagen <a href="#">image1</a> , y también se tiene de entrada el siguiente <a href="#">ground truth</a> . Se ha de mencionar que image1 tiene ciertas dependencias: la cancha debe ser verde, los jugadores no pueden tener la camisa de color "verdosa", si un jugador está chocando con la gradería no será detectado y se asume estático el cuadro a detectar como marcador en la esquina superior derecha.
<b>Criterio Éxito</b>	El método recibe image1, y le aplica detect(). Si la imagen se compara con el groundTruth, y son completamente iguales; pasa la prueba unitaria.
<b>Criterio Fallo</b>	El método recibe image1, y le aplica detect(). Si la imagen se compara con el groundTruth, y no son completamente iguales; no pasa la prueba unitaria.

### 5.3. PlayerDetectorTest

<b>Fecha</b>	12/10/2016
<b>Encargado</b>	Josué Arrieta Salas
<b>Objetivo</b>	Probar la funcionalidad del método detect() de la clase PlayerDetectorTest.
<b>Descripción</b>	Este método debe detectar los blobs correspondientes a los jugadores de una imagen de un partido de fútbol.
<b>Tipo</b>	Prueba Unitaria - Caja negra
<b>Entradas</b>	Se tiene de entrada la imagen <a href="#">image1</a> , y también se tiene de entrada el siguiente <a href="#">ground truth</a> . Se ha de mencionar que image1 tiene ciertas dependencias: la cancha debe ser verde, los jugadores no pueden tener la camisa de color "verdosa" los jugadores deben estar totalmente rodeados por píxeles "verdosos".
<b>Criterio Éxito</b>	El método recibe image1, y le aplica detect(). Si la imagen se compara con el groundTruth, y son completamente iguales; pasa la prueba unitaria.
<b>Criterio Fallo</b>	El método recibe image1, y le aplica detect(). Si la imagen se compara con el groundTruth, y no son completamente iguales; no pasa la prueba unitaria.



#### 5.4. CountFramesTest

<b>Fecha</b>	12/10/2016
<b>Encargado</b>	Joshua Mata Araya
<b>Objetivo</b>	Probar la funcionalidad del método <code>getFrameCount()</code> de la clase <code>OpenCVVideoProcessor</code> .
<b>Descripción</b>	Este método es un get de la cantidad de frames que contiene un video, y este valor se asigna en el constructor del método.
<b>Tipo</b>	Prueba Unitaria - Caja negra
<b>Entradas</b>	Se tiene un video de prueba en el cual se sabe que tiene 132 frames ( <a href="#">video</a> ).
<b>Criterio Éxito</b>	El método realiza la apertura del video para saber cuantos frames tiene, el resultado de esto se compara con el valor 132 que corresponde a la cantidad de frames y este valor es igual.
<b>Criterio Fallo</b>	La cantidad de frames que devuelve el método no corresponde a 132

#### 5.5. LargeVideoTest

<b>Fecha</b>	12/10/2016
<b>Encargado</b>	Adrián López Quesada
<b>Objetivo</b>	Probar la funcionalidad del procesamiento de video, en un archivo de gran tamaño.
<b>Descripción</b>	Debido a que subir un archivo dura mucho según la velocidad de internet, para generar el mejor resultado del análisis de video se requiere un archivo de alta calidad, por lo tanto se usa esta prueba para evitar el tiempo de espera de subida.
<b>Tipo</b>	Prueba Unitaria - Caja negra
<b>Entradas</b>	Se tiene un ( <a href="#">video</a> ) de prueba el cual tiene mas detallados los jugadores, viene en formato .avi con un peso de 104mb.
<b>Criterio Éxito</b>	Despliega un mensaje de éxito y se debe verificar el video generado en la carpeta de datos de prueba (Test Data).
<b>Criterio Fallo</b>	Por falta de memoria el proceso se cae o el video que genera se ve negro.