UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 18 de octubre de 2016
Tutor: nombre tutor

Índice general

Índice de figuras

Apéndice A

Manuales

A.1. Introducción

A.2. Planificación temporal

Sprint 0 (9/9/2016 - 16/9/2016)

Se ha hablado del problema a resolver.

Se va a hacer un mini prototipo para evaluar las herramientas, librerías y algoritmos necesarios. Posteriormente a la reunión con el cliente (Rebeca) se decidirá el lenguaje y librerías a utilizar.

En esta primera iteración se ha hablado de evaluar las distintas herramientas de gestión, documentación y de programación y las tareas son:

- probar LaTex
- probar gestores de tarea:
  - Trello
  - Zenhub
  - Version One
- gestores de versiones
  - GitHub
  - Bitbucket
- examinar el problema, evaluar el notebook y las posibilidades de las librerías
- echar un ojo al artículo

Cumplido:

Esta semana como aun no sabía muy bien cómo usar el repositorio pues las gráfica no nos dicen nada porque tuvimos que cambiar el uso de los milestones inicial a seguir ya que no era posible con GitHub usarlo como deseábamos por lo que hasta la semana 1 no pondré burndown porque no refleja nada del trabajo hecho.

Todos los puntos han sido realizados y destacar que la implementación para el algoritmo ha sido amena y ha funcionado aunque aun tiene cosas que otras semanas mejoraremos.

Sprint 1 (16/9/2016 - 22/9/2016)

En esta semana vamos a tener tareas de interfaz gráfica , de documentación y de codificación y las tareas son:

- Mejora de la detección de las lineas que quedas solapadas.
- Analizar herramientas de interfaces gráficas y comparativa.
  - PyQt4.
  - WxWidget.
- Prototipado inicial de la herramienta y documentar el prototipo.

Cumplido:

Esta semana hemos hecho algunos de los puntos mas relevantes del proyecto ya que la interfaz ha sido realizada correctamente con uso de layouts para facilitar el re escalado de las pantallas sin que se oculten o descoloquen botones.
hemos ampliado el rango de frameworks de interfaces con Tkinter y WxPython porque al investigar vimos que también eran muy relevantes en este campo.
En cuanto a la mejora de la detección de lineas también mejoramos el algoritmo ajustando los parámetros y cambiando algunas propiedades.
También al aveces fallar y como aun no sabemos si es posible dejar pasar el fallo hemos implementado por recomendación de los tutores un modo manual para encontrar las lineas que no encontraba el algoritmo, a su vez también valdrá para pintar una imagen vacía manualmente.

Gráfico del sprint 1:**??**

Sprint 2 (22/9/2016 - 30/9/2016)

En la semana dos vamos a abarcar puntos de la interfaz y puntos de la documentación del proyecto y las tareas son:

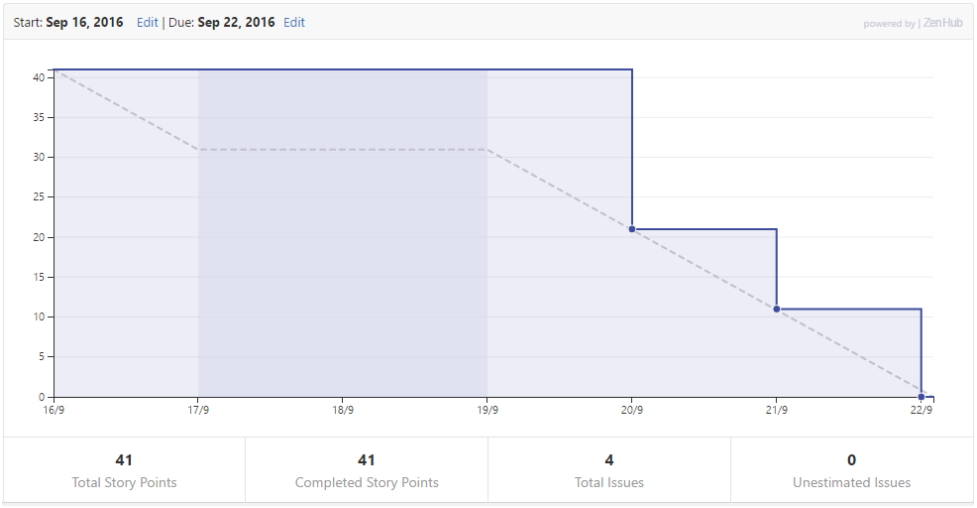

Figura A.1: Burndown de la semana 1

- Documentación.
  - aspectos relevantes.
  - técnicas y herramientas.
  - planificación temporal.
- Listas en la interfaz gráfica (Usar tablas para mostrar las rectas que hemos añadido manualmente).
- Cargar imágenes con el file chooser.

Cumplido:

Hemos cumplido los objetivos aunque mas adelante y después de una revisión seguramente tengamos que modificar algunas cosas y añadir mas ya que es la primera semana de documentación del proyecto.
Respecto al punto de la tabla donde aparezcan las listas de lineas que vamos añadiendo queda preguntar si vendría bien añadir tres botones mas al modo manual cosa que en la reunión con el Cliente (Rebeca) voy a exponer y posteriormente si parece bien desarrollar.
En cuanto al punto de cargar las imágenes con un file chooser de paso como me sobro algo de tiempo añadí una pantalla de inicio con un mensaje para así la primera vez que abramos la herramienta no se muestren tablas vacías ni una imagen predefinida opte por hacer usar una pagina de inicia a modo de fachada y cuando se cargue la imagen ya iniciar todas las funcionalidades de la aplicación.

Gráfico del sprint 2:**??**

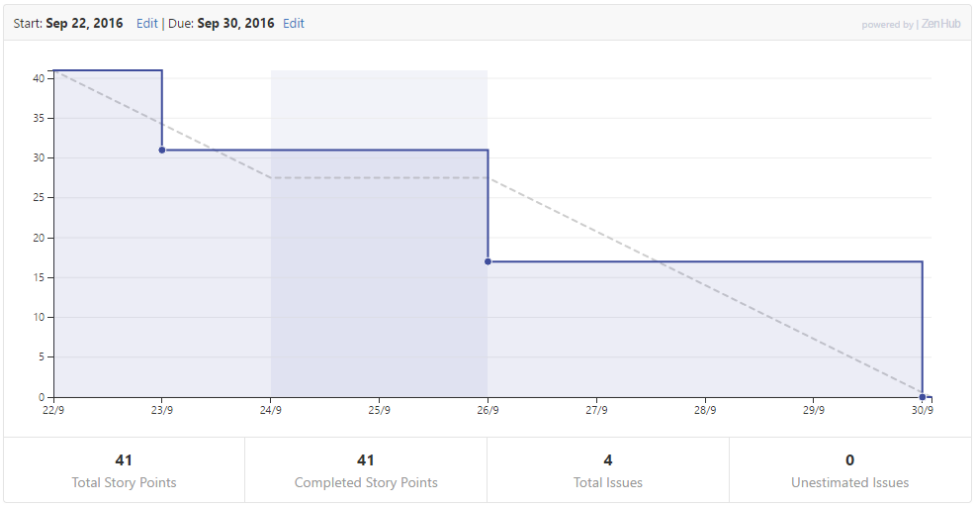

Figura A.2: Burndown de la semana 2

Sprint 3 (30/9/2016 - 7/10/2016)

En la semana tres vamos a abarcar puntos de la interfaz GUI , generar el informe, y pasar actualizar el código de PyQt4 a PyQt5 y las tareas son:

- Acabar la GUI.
  - Mostrar todas las lineas en la tabla.
  - Poder borrar la linea seleccionada.
- Informe.
  - Calcular estadísticas.
  - Mirar documentación Python.
- Pasar código de PyQt4 a PyQt5.

Cumplido:

Hemos cumplido los objetivos de esta semana y hemos generado funcionalidad a la tabla para agregar las lineas, también añadido que se ilumine la linea seleccionada dentro de la imagen en color amarillo. Hemos añadido funcionalidad para borrar la linea que tenemos seleccionada y también para poder limpiar la tabla completa.

Respecto a la tarea del informe, hemos Calculado los estadísticos de todas las lineas, también su clasificación y escritura dentro de un fichero CSV, también la generación de una tabla latex que se actualiza a cada ejecución con los datos que han salido para poder pegarla fácilmente a un informe.
Como nos dimos cuenta que la versión de PyQt se había actualizado de la cuatro a la cinco pues hemos pasado el código a la nueva version y no ha sido muy difícil.

Gráfico del sprint 3:**??**

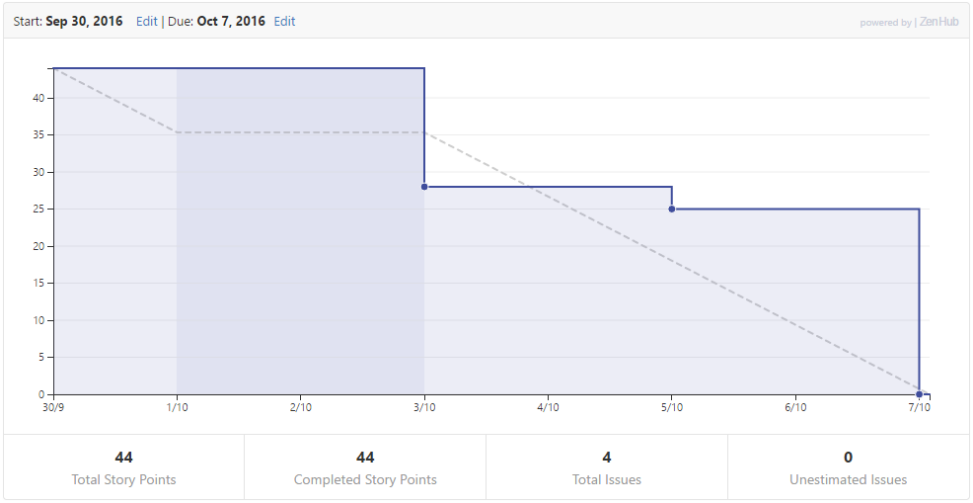

Figura A.3: Burndown de la semana 3

Sprint 4 (7/10/2016 - 14/10/2016)

En esta semana vamos a abarcar el diseño software de la aplicación así como sacarlo de los NoteBooks y pasarlo a un IDE en condiciones con su subdivisión en clases y paquetes y las tareas son:

- Diseño de la aplicación.
  - Diagrama de clases.
  - Diagrama de paquetes.
- Implementación del código.
- Corrección de las memorias.
- Herramienta SonarQube.
  - Ejecutar la aplicación.
  - Corregir las horas de débito.

Cumplido:

Hemos cumplido los objetivos de la semana. En paralelo hemos implementado el diseño y el código a medida que teníamos una parte del diseño, de forma incremental, por lo que no se queda del todo reflejado cuando cerramos las tareas.
Hemos elegido Eclipse como IDE junto con su plugin PyDev para Python. La división en clases hemos conseguido tener una primera estimación de como estaba la aplicación. Respecto a la herramienta SonarQube hemos corregido todos los errores y defectos que salían, a mencionar que no había código repetido ni errores graves.
También detectamos un Bug y fue corregido.

Gráfico del sprint 4:**??**

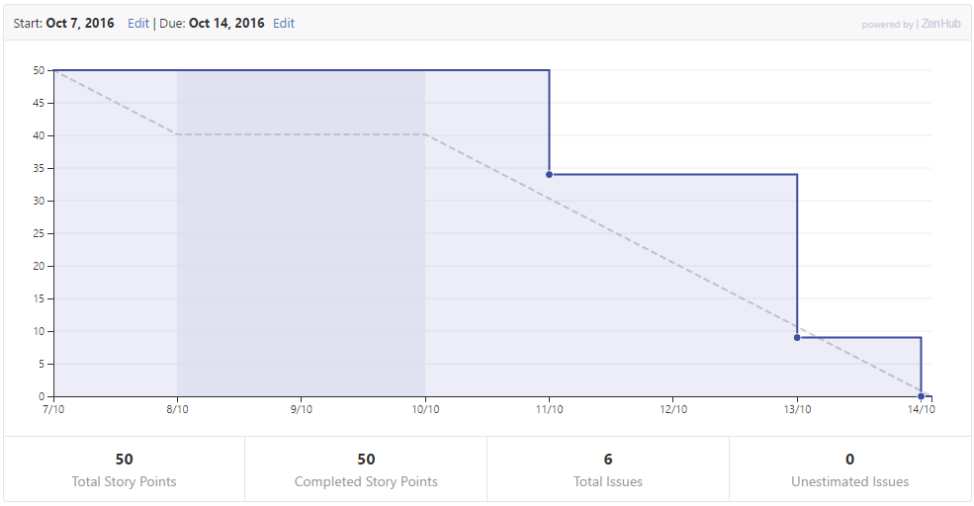

Figura A.4: Burndown de la semana 4

Sprint 5 (14/10/2016 - 21/10/2016)

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

En este punto vamos a hacer el diseño inicial de las clases de la aplicación y del diseño de paquetes en el que estarán contenidas.

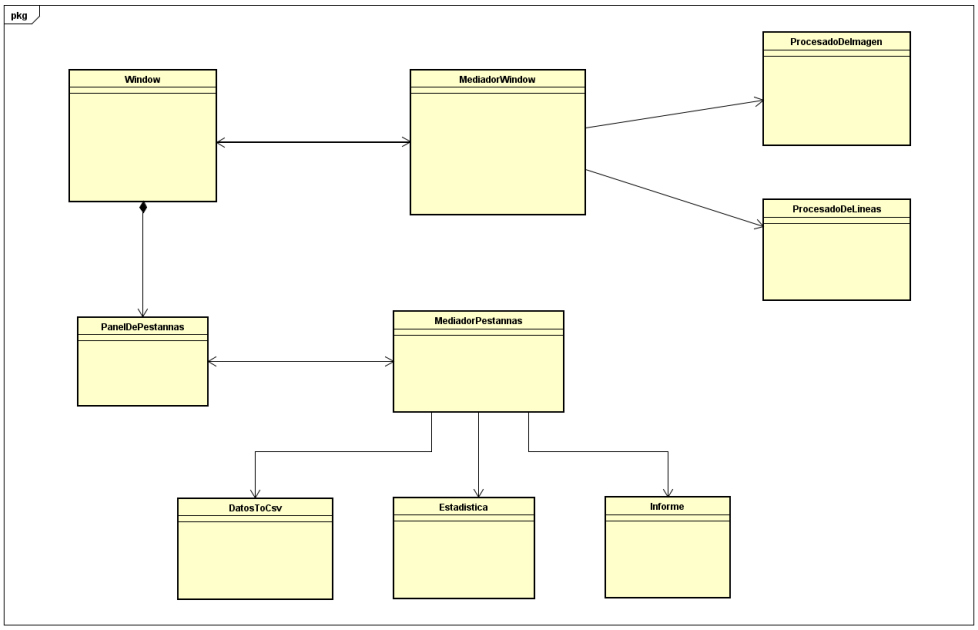

Figura C.1: Diagrama de clases inicial

Clases

En esta parte vamos a hacer una breve descripción del contenido de cada clase y su función.

- VentanaInicio: Esta clase simplemente hará de fachada entre la ejecución y el acceso a la aplicación real cuando hayamos elegido una imagen que procesar
- Window: Esta clase contendrá la ventana principal, menús, el panel de pestañas y la imagen sobre la que dibujar.
- PanelDePestannas: Esta clase contendrá la botonería y la interacción con la imagen a evaluar.
- MediadorPestannas: Esta clase va a ser un mediador para las pestañas para deslocalizar código y complejidad.
- MediadorVentana: En esta clase va ser un mediador entre la ventana principal y sus métodos para así reducir complejidad en la clase de ventana.
- ProcesadoDeImagen: Esta clase contendrá el tratamiento que tiene que llevar la imagen para la extracción de características a evaluar.
- ProcesadoDeLineas: Esta clase contendrá el procesado de las lineas obtenidas por el procesado de la imagen y el resultado final que obtendremos.
- Informe: Esta clase contendrá la generación del informe (Tabla) de estadísticas en formato LaTex para poder exportar fácilmente a un documento PDF.
- DatosToCsv: Esta clase contendrá el paso de los datos que contiene la tabla a un documento en formato csv del que desde Excel podemos extraer fácilmente los datos.
- Estadistica: Esta clase contendrá el cálculo de los datos estadísticos y la clasificación de las lineas según su orientación.

Paquetes

En este punto vamos a hacer una breve descripción de cada paquete y de su contenido dentro de la aplicación.

- Gui: Este paquete contendrá todos los elementos gráficos de la aplicación y todos lo que se corresponde con interacción con el usuario.
  - Mediadores: Este paquete va a contener los mediadores de la interfaz para deslocalizar código y complejidad en esta.
- Código: Este paquete contendrá todas las clases de cálculo que necesitara la aplicación:

- Estadísticas: Este paquete contendrá las clases del cálculo de estadísticas.
- Informes: Este paquete contendrá la generación del informe LaTex
- Procesado: Este paquete se va a encargar tanto del procesado de la imagen como del procesado de los segmentos extraídos hasta conseguir los segmentos finales.

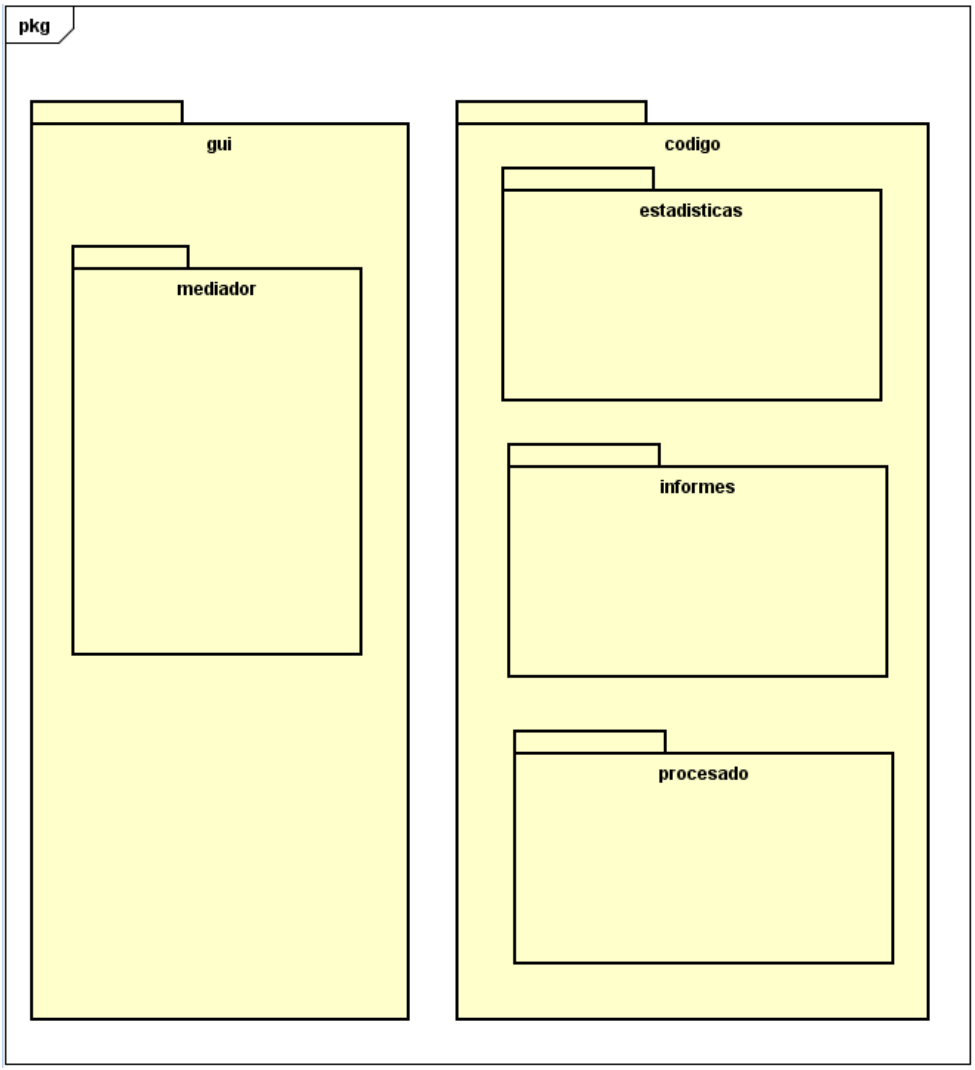

Figura C.2: Diagrama de paquetes inicial

C.2. Diseño de datos

Introducción

Patrones de diseño

En este apartado vamos a usar de los patrones de diseño que hemos utilizado en nuestra aplicaron.
El patrón más importante y con el que vamos a empezar es el Medidor que consigue hacer que la interfaz este limpia simplemente con las declaraciones de las variables que más adelante va a utilizar. La base de los patrones de diseño su libro más significativo es: [**?**]

- Mediador [**?**]: Se utiliza para encapsulas las interacciones entre los objetos de la interfaz por lo que se clasifica como patrón de comportamiento. Como dato curioso, es de los pocos patrones que permiten la bidireccionalidad de comunicación entre las clases, por eso es el patrón idóneo para la interacción de las clases de la interfaz gráfica. Aplicando este patrón reducimos la dependencia del código por lo que de esta forma las interfaces no ven lo que tienen por debajo y reduce la complejidad.

  Obtenemos:

  - Conseguir desacoplamiento.
  - Simplificar y aclarar la comunicación.
  - Centralizar el control.
  - Interfaz simple para un sistema complejo.

- Comando [**?**]: Se utiliza para encapsular el contenido de una función, así ni el emisor de la operación ni el receptor saber que hay por debajo, pero si poder acceder a las funciones que hay.

  Obtenemos:

  - Permitir la parametrización.
  - Visualizar que ordenes se ejecutan en cada paso.
  - Construir operaciones complicadas a partir de otras más sencillas.
  - Permitir saber que habría que hacer para deshacer una operación.

- Fachada [**?**]: Es un patrón de diseño estructural, sirve para reducir la complejidad con la división en clases más pequeñas y reduciendo sus dependencias.

  Obtenemos:

- Separar en niveles la aplicación.
- Reducir su complejidad.
- Desacoplar el código.
- Interfaz simple para un sistema complejo.
- Potable y reutilizable.

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario