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 — 7 de mayo de 2017
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

Sprint 0 (1/2/2017 - 8/2/2017)

En este primer «Sprint» determinamos que posibles herramientas de traducción pueden ser útiles para este proyecto sin profundizar demasiado. Se hace también la primera toma de contacto con Thoth para ver su funcionamiento básico.

Por otro lado, analizamos las posibilidades del proyecto, determinando los caminos en los que puede derivar dicho proyecto. Es decir, en estos primeros pasos no sabemos como funcionan estas herramientas, es por ello que el resultado sea más simple del esperado o por el contrario resulten complicaciones que lleven un tiempo mayor al esperado.

Además de esto, hacemos una introducción a las herramientas de documentación y gestión. Por ello aclaramos que:

- Para la gestión de versiones haremos uso de GitHub, asociando un gestor de tareas llamado Zenhub que funciona como un «plugin» en el buscador.
- Realizo un primer contacto con LaTeX

En esta primeras semanas no me aclaro mucho con el funcionamiento del gestor de versiones, y es por ello que no hago un buen uso de los «commits» ni de los «issues» que proporciona Github.

Sprint 1 (8/2/2017 - 15/2/2017)

Ya en la segunda semana realizo una evaluación más exhaustiva de las herramientas de traducción, analizando los pros y los contras de ellas. Por lo tanto tomamos la decisión de centrarnos en GWT como la principal y con la que vamos a llevar a cabo el proyecto.

Definimos como tareas semanales:

- Evaluar los pros y contras de las diferentes herramientas de traducción de código.
- Documentar esa evaluación con LaTeX, familiarizándome con la manera de documentar.
- Realizar las primeras pruebas con GWT, de una forma simple.

En esta semana hago alguna prueba simple con GWT, gracias a los ejemplos que proporciona la página oficial a modo de tutorial. También realizo alguna prueba simple con JSweet para ver su funcionamiento real y si es, de verdad, útil para poder llevar a cabo el proyecto. En el último ejemplo que hago me ocurre un problema con GWT que no termino de solucionar y que me obliga a posponer la prueba de ese ejemplo para el siguiente sprint.

También profundizo algo más en la documentación y en el uso de las herramientas para documentar. Hasta este punto no he asociado las tareas o «issues» con los «milestones» y por lo tanto no queda registrado el tiempo del sprint.

Sprint 2 (15/2/2017 - 22/2/2017)

En este tercer sprint, lo primero que hago es solucionar el anterior problema que tuve con GWT. Consistía en configurar bien el entorno de Eclipse y el «plugin» para poder hacer la ejecución de GWT con «Super Dev Mode», alternativa implantada por los desarrolladores para evitar la necesidad de instalar una extension de GWT en el buscador en el cual se lanza la aplicación.

También en esta semana descubrimos una nueva herramienta relacionada con el tema, que se llama Vaadin y que nos puede servir, por lo menos para hacer una comparativa más completa de las herramientas de traducción.

La parte más importante de esta semana es la prueba de traducción del «core» de Thoth, que aunque no sale como esperamos, ha resultado útil para conocer con mayor profundidad tanto la aplicación como la herramienta.

Por lo tanto como tareas para este sprint:

- Solucionar el error surgido con GWT.
- Realizar pruebas de traducción con el «core» de Thoth.
- Incluir la nueva herramienta en la comparativa.

Muy a mi pesar, en el «milestone» de esta semana, aunque he pasado cada tarea al estado de realizada o «done» no las he cerrado hasta darnos cuenta al final del sprint, es por ello que el «burndown» queda de esta manera.

Sprint 3 (22/2/2017 - 1/3/2017)

Ya en la cuarta semana se hace un intento más completo para traducir la aplicación. Como pudimos comprobar en la semana pasada, GWT no traducía las librerías de la parte visual de Thoth. Es por ello decidimos hacer un ejemplo de forma manual que consiste en programar parte de la vista que esta asociada a las partes más relevantes del núcleo

Por ejemplo, nos centramos en hacer una prueba con la gramática, que esta dentro del núcleo, creando una pantalla con un «text label» para comprobar si funcionaba la traducción de esa parte del núcleo. De esta forma podríamos ver como hacía la traducción de todo el núcleo, ya que las otras partes de las que se compone son similares en el uso de librerías y bibliotecas.

Quedan así asignadas las tareas para del sprint número tres:

- Transformar la Gramática del núcleo de Thoth.
- Transformar el Autómata del núcleo.
- Transformar la simulación del núcleo de Thoth.
- Documentar toda esta parte.

Al final solo se pudo llevar a cabo la tares de la Gramática y la documentación porque no se pudo avanzar a las demás. Nuestra idea era probar a tratar de traducir todo, núcleo incluido, ejecutando esas partes en el cliente de GWT pero vimos que esto no es posible. Intentamos hacerlo de varias formas eliminando partes no esenciales de la aplicación para reducir errores de compilación hasta darnos por vencidos y ver que esa no era la solución.

Sprint 4 (1/3/2017 - 8/3/2017)

La cuarta es la semana en la cual, hemos intentado pasar la aplicación en la parte del servidor y probar por nuestra cuenta. Sólo hemos metido el núcleo en el paquete servidor para ver que surgía. Como vimos que no había una

comunicación entre el cliente y el servidor hicimos varios intentos, probando con el paquete de «shared» o compartido, pero GWT también traduce ese paquete a JavaScript por lo que seguía dando los mismo errores que en el cliente.

Por lo tanto en este sprint tenemos esta tareas:

- Solucionar un error en la traducción.
- Realizar pruebas cliente-servidor con el núcleo de la aplicación.
- Cambios y mejoras en la documentación.

Una vez nos dimos cuenta de que el fallo de tratar de hacer la aplicación en la parte del servidor era que GWT no reconocía algunas de las librerías claves, tanto en la parte visual como en el núcleo de la aplicación, decidimos buscar otros caminos alternativos.

El funcionamiento de GWT consiste en traducir a «JavaScript» la parte del cliente y la compartida. En consecuencia decidimos hacer pruebas en las que las partes mas fundamentales del núcleo se encontrasen en el lado del servidor. De esta forma cuando el cliente necesitase hacer algún uso de métodos con librerías no reconocidas por GWT, simplemente llamase al servidor ya que este podría soportar dichos métodos.

En los primeros intentos nos dimos cuenta de que estas llamadas no se podían hacer de una forma simple, ya que la comunicación entre cliente y servidor no funcionaba y no obteníamos los resultados que esperábamos. Aún así seguimos haciendo pruebas para asegurarnos, metiendo dentro del paquete «compartido» las partes del núcleo mas cercanas a lo que nosotros consideramos la vista. El problema seguía siendo esa comunicación. Interpretaba como del lado del cliente lo que nosotros queríamos que formara parte del servidor, dando errores debido a que GWT no trabaja con esas librerías.

Sprint 5 (8/3/2017 - 15/3/2017)

Principalmente, en este quinto sprint, se llevan a cabo las pruebas para entender y poder evaluar la comunicación cliente-servidor, por medio de unos ejemplos. Además de eso, planteamos la idea de realizar un «login» y validación de usuarios, pero solo como idea, ya que no es de gran importancia.

Así que en este sprint tenemos esta tareas:

- Ejemplo cliente-servidor con GWT.
- Login y validación en GWT.

La comunicación entre el cliente y el servidor se lleva a cabo mediante la comunicación RPC (Remote Procedure Call). Es por ello que se hace necesario entender y practicar el funcionamiento de esta práctica. Los ejemplos realizados han sido dos: el primero es un ejemplo o tutorial ofrecido por la página oficial de GWT, que consiste en hacer un visor del «stock» que cambia de forma aleatoria sus valores. Y el segundo ejemplo consistió en hacer un pequeño ejemplo de llamada de funciones con más clases que en el anterior.

Sprint 6 (15/3/2017 - 22/3/2017)

En esta semana nos metemos ya en serio con la aplicación propiamente dicha. Lo primero que hacemos es conseguir que la comunicación entre en núcleo (ya hecho) y su uso sea fluido. Para ello lo que hacemos es incluir algunas partes en el cliente y otras en el servidor. En el cliente sólo podemos incluir las clases más simples, más primitivas de la aplicación porque su contenido es entendido por GWT y puede hacer la traducción a JavaScript sin problemas de librerías.

Por lo tanto las tareas son básicamente dos:

- Llevar a cabo el primer prototipo o sección de la aplicación.
- Documentar y corregir errores anteriores en la documentación.

La realización del prototipo nos lleva tiempo ya que se necesita comprender muy bien el funcionamiento interno de Thoth y así poder definir un diseño del software adecuado según ese funcionamiento. Una vez hecho eso parece simplificarse los problemas que al principio se tenían.

Sprint 7 (22/3/2017 - 29/3/2017)

Parece ser que en esta octava semana ya podemos empezar a trabajar más en la programación del diseño e implementar funcionalidades. Lo primero que tenemos que hacer es que esa comunicación entre métodos de resultados «más» visibles e integrarlos en una «GUI» denominada en español como interfaz gráfica de usuario.

Así es que definimos como tareas las siguientes:

- Reestructurar la aplicación y limpiar el código.
- Mejorar la GUI con Vaadin, ya que la anterior es muy básica.
- Implementar el algoritmo .ElIminar símbolos no terminales"

Primeramente hay que organizar el código haciéndolo más legible y limpiarlo de comentarios, y pruebas para ver su funcionamiento. Queremos que los resultados queden de una forma similar al Thoth original, para conservar su esencia, usabilidad, y buen diseño. Para ello hacemos uso de Vaadin, una herramienta que se puede utilizar como un «plugin» en eclipse y que se integra perfectamente con GWT.

Sprint 8 (29/3/2017 - 5/4/2017)

Al principio de este octavo sprint o 9 semana, estuve pendiente de la respuesta por parte de Vaadin sobre si me podía conceder o no una licencia gratuita, así que me centré en incluir el algoritmo de eliminación de símbolos no terminales, mejorando lo que tenía hasta el momento, que era solo una pequeña interfaz con una funcionalidad que no era la exacta. Por ello me centré en corregirla. Así lo hicimos. Posteriormente llegó la respuesta de Vaadin explicando que no me podía conceder la licencia y que buscase otras opciones dentro de las que ellos mismos me ofrecían. Al principió lo intenté pero resulto que no supe como hacerlo. Lograba hacer un proyecto de Vaadin pero no conseguía incluirlo en el mio propio de GWT.

Así que comencé a hacer la interfaz por mi mismo, con las posibilidades de GWT.

Las tareas para esa semana fueron:

- La inclusión del algoritmo de Eliminacion de símbolos no terminales, de una forma mejorada.
- Recabar información sobre internacionalización en GWT.

Logramos hacer que funcionase el algoritmo como queríamos y estuvimos mejorando el diseño y la funcionalidad de la interfaz de dicho algoritmo.

Sprint 9 (05/4/2017 - 19/4/2017)

Es el sprint más largo hasta la fecha ya que incluye dos semanas de trabajo por coincidir con las vacaciones de Semana Santa. Engloba fundamentalmente el hacer los demás algoritmos, con sus respectivas vistas.

- Implementar los algoritmos restantes.
- Mejorar el código, haciéndolo más comprensible y organizar la parte visual.
- Hacer la documentación sobre la parte del diseño y sobre las herramientas que tiene la UBU relacionadas con este proyecto.

En estas dos semanas no pude realizar todos los algoritmos como en un principio pretendía porque la parte visual se alejaba de lo que tenia hecho hasta el momento, es decir, necesitaba hacer nuevas vistas que llevaban más tiempo del pensado y no me dio tiempo, pero si que incluí la mayor parte de ellos. Estuvimos estudiando como aplicar el resaltado de las producciones. En el Thoth original utilizaba una librería, java swing, que ya comprobamos anteriormente que no podía ser incluida en GWT en este proyecto así que buscamos otras alternativas como hacer un resaltado a mano con HTML. De momento lo dejamos ahí.

Sprint 10 (19/4/2017 - 26/4/2017)

Una vez pasadas las vacaciones de Semana Santa, el proyecto tiene una forma más madura y podemos ir haciendo añadidos más funcionales a la aplicación. Es por ello que decidimos empezar con la internacionalización, viendo como funcionaba para poder entenderla y una vez entendida incluirla en el proyecto. Además de esto, después de comprobar en la semana pasada que el resaltado lo podíamos hacer con HTML nos dedicamos de lleno a ello. La verdad es que nos llevó muchos quebraderos de cabeza. El porqué no era otra cosa que teníamos que tener en cuenta todo el funcionamiento interno, «destripar» que contenía cada variable, como funcionaban cada método de GWT etc. Todo este trabajo nos llevo mucho tiempo, ya que estuvimos todo el sprint a base de prueba y error con los diferentes algoritmos hasta que funcionó en todos de la forma que deseamos.

Estas fueron las tareas correspondientes a esta semana.

- Aplicar internacionalización.
- Cambiar la visualización de los paneles.
- Resaltado de producciones en los algoritmos con HTML.
- Incluir los demás algoritmos.
- Incluir pestañas y elementos de Vaadin.

La parte visual, es decir, la inclusión de pestañas en realidad no lo llegamos a aplicar bien, de la forma deseada y se trató en el siguiente sprint. Además los elementos de Vaadin no se pudieron incluir ya que la única forma de añadirlos a un proyecto GWT son con licencias de pago. La visualización de los paneles simplemente fue una recolocación para que quedase más agradable a la vista.

Sprint 11 (26/4/2017 - 03/5/2017)

En la semana 13, seguimos acumulando un pequeño «bug» en el resaltado de las producciones, y es que la interpretación de las comillas dobles «“”» no las interpretaba como nosotros queríamos y por ello no hacía un buen borrado del resaltado, manteniéndose en cada paso. La solución fue simplemente sustituirlas por las comillas dobles de java «/“». Pero las dos grandes tareas de esta semana consistieron en aplicar el algoritmos FirstFollow y un TabLayoutPanel para hacer un panel con pestañas como los que se ven en los navegadores web.

Las tareas quedaron asi.

- Solucionar bug en el resaltado.
- Implementar algoritmo FirstFollow.
- Incluir TabLayoutPanel para hacer diferentes pestañas.

El algoritmo FirstFollow fue fácil hasta el momento de mostrar los resultados en las tablas. Resulta que al tratar de imprimir los resultados de tipo Object pasándolos como un «string» no los reconocía bien. Este tipo de errores no son fáciles de detectar en GWT, ya que la forma para verlos claramente es imprimiendo los valores por pantalla y tratando de analizarlos, en que formato están etc. Al final descubrimos que el problema era al tratar de pasara al formato string valores que GWT interpretaba como «undefined» y que no eran más que valores en blanco, espacios en blanco dentro de la tabla.

El otro gran quebradero de cabeza fue el tratar de incluir pestañas en la aplicación. No es nada fácil añadir una pestaña con una nueva vista sin que esta remplace otra, se superponga o cualquier otra cosa. A día de hoy no sabemos bien el porqué de esto al cien por cien. Lo que si es cierto es que no es completamente necesaria y se puede hacer un reemplazamiento de otras vistas.

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

C.2. Diseño de datos

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. Instalación y configuración de GWT:

Para poder trabajar con GWT debo descargar el SDK de GWT proporcionado en la página web oficial. Los requisitos previos para crear un aplicación web con Google Web Toolkit son básicamente dos: tener instalada la SDK de Java en su versión 1.6 o cualquiera superior a esta y tener instalado también Apache Ant o en su defecto Apache Maven.

Es fácil saber si cumplo ambos requisitos. Una vez descargado el SDK de GWT desde la página oficial [1], accedo a la carpeta desde la consola de comandos, y ahí intento ejecutar el comando «webAppCreator». En caso de que la consola me devuelva un error en el que indica que Java no ha podido reconocerlo como un comando interno, quiere decir que no cumplo esos requisitos previos.

Posteriormente puedo proceder de varias formas y con diferentes plataformas. En mi caso he elegido la plataforma Eclipse sobre la que trabajaré en su versión 4.4 que es también denominada con el nombre de Luna.

GWT se instala en Eclipse como un «plugin» y para ello debo ir, dentro de Eclipse a «añadir un nuevo software» donde encontraremos una ventana

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html

donde escribir la dirección del software a instalar. Para mi versión (llamada Luna) la URL es «https://developers.google.com/eclipse/docs/install-eclipse-4.4» que una vez añadida mostrará los paquetes a instalar. Estos deben ser Google Plugin for Eclipse y GWT Designed for GPE, y una vez seleccionados se procederá a su instalación normal. Después de reiniciar procedo a referenciar el SDK de GWT yendo a «preferencias», «Google» y dentro de «Web Toolkit» añadir el SDK que descargué anteriormente.

Para añadir un nuevo proyecto a Eclipse tengo que importarlo desde Maven, puedo hacerlo añadiendo un proyecto de Maven existente. Una vez seleccionado el proyecto, hay que configurar su ejecución. Debemos ir a «Run Configuration» y seleccionar el constructor de Maven. Una vez ahí, puedo añadirle un nombre específico y seleccionando el directorio base el proyecto a ejecutar y por último el «Goal», o meta debe ser «gwt:run», ya que en caso contrario no se producirá una ejecución correcta.

Antes de la ejecución se debe instalar el constructor de Maven desde el debuguer, sino puede dar algunos errores. Una vez finalizada la instalación, podremos ejecutar un proyecto con GWT apareciendo así el modo desarrollador y ahí se podrá lanzar la aplicación en el buscador por defecto.

El «plugin» de GWT es sencillo de utilizar y solo con hacer click en «New Web Application Project» aparecerá un «wizard» en el cual hay que introducir el nombre del proyecto y el paquete principal. Se recomienda desmarcar «Use Google App Engine» ya que además de que no nos interesa, puede que nos dé algún error. Hecho esto se creará un proyecto automáticamente en forma de ejemplo, con un paquete cliente, otro servidor y uno compartido.

(05/03/2017) Los proyectos en GWT se componen de una parte cliente y otra servidor. He creado un proyecto con sólo la parte del cliente, la cual es la más importante. En el paquete «src/client» he incluido la clase principal que llamará a las otras. En realidad no se si esto es correcto al cien por cien. Solución del error del «prefuse» es descargando de la página oficial [2] el proyecto, incluyo el paquete prefuse y añado, desde «Java Build Path» la librería «lucene» porque es necesaria para algunas clases del paquete. Inicialmente muestra unos errores del tipo «did you forget to inherit a required module?» referente a que dentro del proyecto, la clase con extensión .gwt.xml no incluye algunas de las rutas de los recursos que utilizo. No estiendo muy bien el error. (continuar con la explicación)

[2] http://prefuse.org/

D.5. Compilación, instalación y ejecución del proyecto

D.6. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía