Bizativiti

Grupo 04

Informe Final de Verificación
Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
24/11/2013	1.0	Comienzo de Documento	Diego Cossio
24/11/2013	1.0	Continuación	Martin Rubio
24/11/2013	1.0	Revisión SQA	Valeria Rocha

Índice

1. Resumen de la verificación

1. *Planificado vs. Realizado*

Prueba de Funcionalidad

Según lo planificado como solo existía un gran caso de uso que era parsear de xpdl a bpmn2.0 se crearon casos de prueba a partir de los elementos que estaban en el alcance. De esta manera, para cada liberación, se ejecutaban los casos de prueba relacionados con los elementos que estaban implementados.

Se ejecutaron casos de prueba por elementos y combinaciones con otros elementos. Se controló que el producto se parseara correctamente y que el manejo del logueo lo hiciera adecuadamente.

Prueba de Interfaz de Usuario

Como no era requerimiento al principio no se había definido como íbamos a desarrollar el producto, al final se optó por desarrollar en forma web.

La principal funcionalidad de la interfaz es poder elegir que elementos parsear y agregar atributos extendidos además de que tipo de formato parsear a bpmn2.0 si es un archivo txt o xpdl el de entrada.

Se detectaron defectos por ejemplo cuando uno quería filtrar algún elemento no lo hacía además de algunos arreglos en cuanto al diseño y la estética, luego se solucionaron.

También existieron problemas menores a la hora de la descarga del log generados en la transformación, estos fueron rápidamente detectados y corregidos.

Prueba de Configuración

Se probó el producto con la configuración definida no encontrándose problemas de configuración.

Prueba de Documentos

Se verificaron algunos documentos sin encontrar defectos de ambigüedades entre estos documentos están Especificación de Requerimientos, Manual de Soporte, Manual de Configuración y Documentación de Usuario. Cabe señalar que no se hizo un verificacion mantenida en este asunto y se optó por contar con ayuda de Valeria para asegurar la calidad de los mismos. (Resp. de SQA).

2. *Cantidad de errores encontrados*

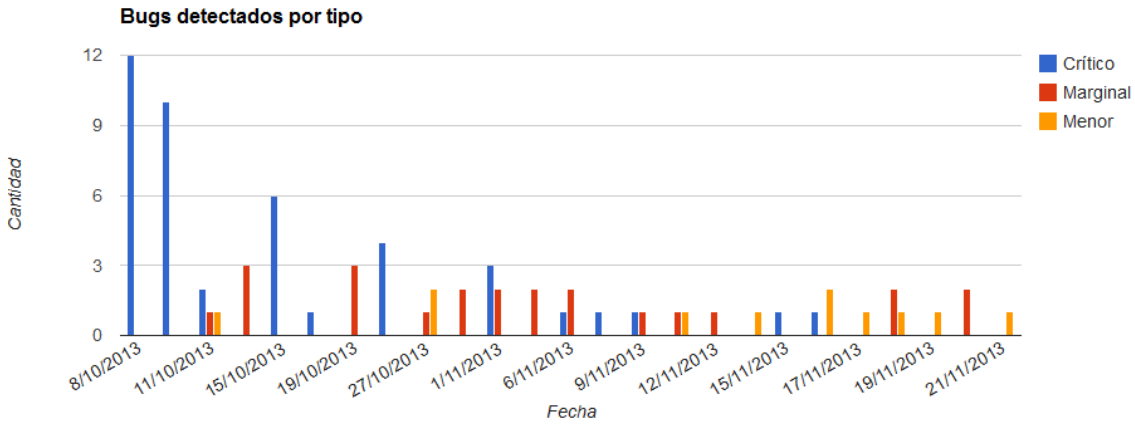
Bizativiti

Resumen de bugs por severidad:

	Detectados	Reparados
Cantidad Total Crítico	43	43
Cantidad Total Marginal	23	23
Cantidad Total Menor	11	11

Resumen de bugs por fase :

	FASE ELABORACIÓN (16/9 - 20/10)	FASE CONSTRUCCIÓN (21/10 - 17/11)	FASE TRANSICIÓN (18/11 - 24/11)
Cantidad Crítico	31	12	0
Cantidad Marginal	7	12	4
Cantidad Menor	1	7	3



Se observa de la gráfica la cantidad de errores críticos al principio, esto se dió por problemas de integración, con el correr de los días aprendiendo de lo ocurrido se mejoró a tal punto de no existir errores de integración.

Casos de Uso

Aunque consistía de un solo caso de uso se construyeron un total de 119 casos de pruebas relacionados con el alcance total del producto y todos ellos fueron testeados

2. Evaluación

Una vez que estaba determinado el problema, se estudió cual podría ser la mejor manera de realizar las pruebas para evitar el re-trabajo y garantizar la correctitud de las mismas. A partir de esto se diseñaron distintos mecanismos para realizar las pruebas de nuevas funcionalidades, de regresión, sobre la interfaz de usuario, etc.

Considerando que la realidad del proyecto contaba con innumerables parámetros, atributos, constantes,etc que debían ser controlados se creó una herramienta de software capaz de automatizar estos chequeos lo cual facilitó la tarea enormemente y finalmente el esfuerzo invertido en la implementación

de dicha herramienta fue considerado como muy fructuoso.

Cada vez que se realizaba una liberación que contenía nuevas funcionalidades o fixes a bugs detectados se realizaba una prueba de regresión. La misma consistió en comparar las salidas de la nueva versión con la salida de la versión anterior inmediata.

A modo de simplificación se pueden determinar 4 grandes áreas a la hora de la realización de los test:

1. ***Pruebas de conversión de elementos***

   Para llevar a cabo esta tarea se utilizó por un lado, la herramienta de software mencionada anteriormente y a su vez cada vez que existía una liberación con nuevos elementos se verificaron exhaustivamente cada uno de estos. Para ello se importaban los modelos generados en activiti y se corroboraba manualmente la correctitud del mismo.

   En este punto existieron algunos retrasos en la implementación lo cual obviamente llevó a algunos retrasos a la hora de realizar los test.

2. ***Pruebas de la funcionalidad de la Api-Rest***

   Para probar esto se creó un cliente en python el cual testeaba cada uno de los servicios expuestos por la rest. También se generaron sentencias para correr los casos de prueba utilizando dicha herramienta.

3. ***Pruebas de arquitectura de plugins***

   En este caso se creó un parser TXT simplificado que su función era comprobar que se podían generar distintos plugins y la aplicación podría soportarlos.

4. ***Pruebas sobre el log generado***

   En cuanto al log, se buscó desde un principio que el mismo fuera útil para el usuario, por esto se puso bastante énfasis a la hora de comprobar la claridad de los mismos tanto para casos de éxito como en casos de fallas.

Si bien en un principio no fue fácil organizar los casos de prueba y la realización de los tests, con el correr de las semanas se logró crear un método de testeo el cual resultaba práctico y eficaz, lo cual llevó a la detección de bugs de manera temprana y oportuna; esto tiene como consecuencia que la corrección de los mismos por parte de los implementadores fuera más fácil.

En aspectos generales consideramos que el producto realizado cuenta con un nivel de rigurosidad en las pruebas acorde a lo exigido por el cliente, el cual desde un principio, puso gran énfasis en que la herramienta desarrollada sea utilizable y que funcionará de manera correcta.

3. Informe final de las métricas de Pruebas Cubiertas

Se realizaron todas las pruebas planificadas según las funcionalidades se iban liberando.