TECNOLÓGICO DE COSTA RICA

Escuela de Ingeniería en Computación

Proyecto de Ingeniería de Software

Profesora:

María Estrada Sánchez

Entrega 1:

Analizador contextual de Triángulo:
Plan de pruebas de software

Estudiantes:

Christian León Guevara - 2013371982

Gabriel Ramírez Ramírez - 201020244

Fecha de entrega:

22-12-2018

Período Verano

Cartago

Información del proyecto 3
Resumen ejecutivo 4
Alcance de las pruebas 4
Elementos de pruebas 4
Pruebas de regresión 5
Funcionalidades que no se van probar 5
Enfoque de pruebas (estrategia) 5
Pruebas funcionales 6
Pruebas no funcionales 6
Criterios de aceptación o rechazo 6
Entregables 7
Recursos 7
Requerimientos de entornos – Hardware 7
Requerimientos de entornos – Software 7
Personal 8
Herramientas de pruebas requeridas 8
Entrenamiento 8
Planificación y organización 9
Matriz de responsabilidades 9
Dependencias y Riesgos 9

Información del proyecto

Proyecto	Analizador Contextual del Lenguaje Triangulo
Fecha de preparación	
Cliente	Ignacio Trejos
Gerente / Líder de proyecto	Gabriel Ramírez Ramírez
Gerente / Líder de pruebas de software	Christian León Guevara

Resumen ejecutivo

Este documento define el proceso de aplicación de pruebas del proyecto. El documento no define completamente las pruebas pero da una visión general del proceso por seguir. Este proyecto tiene un fin educativo, por lo que no requerirá hardware potente o específico; el compilador completo debe poder ser ejecutado en una computadora no muy potente.

Alcance de las pruebas

Elementos de pruebas

Si tomamos en consideración el diagrama hecho por otro equipo que implementó la totalidad del compilador Triángulo en el pasado, podemos determinar tres componentes a los cuales se le realizarán las pruebas en ese proyecto.

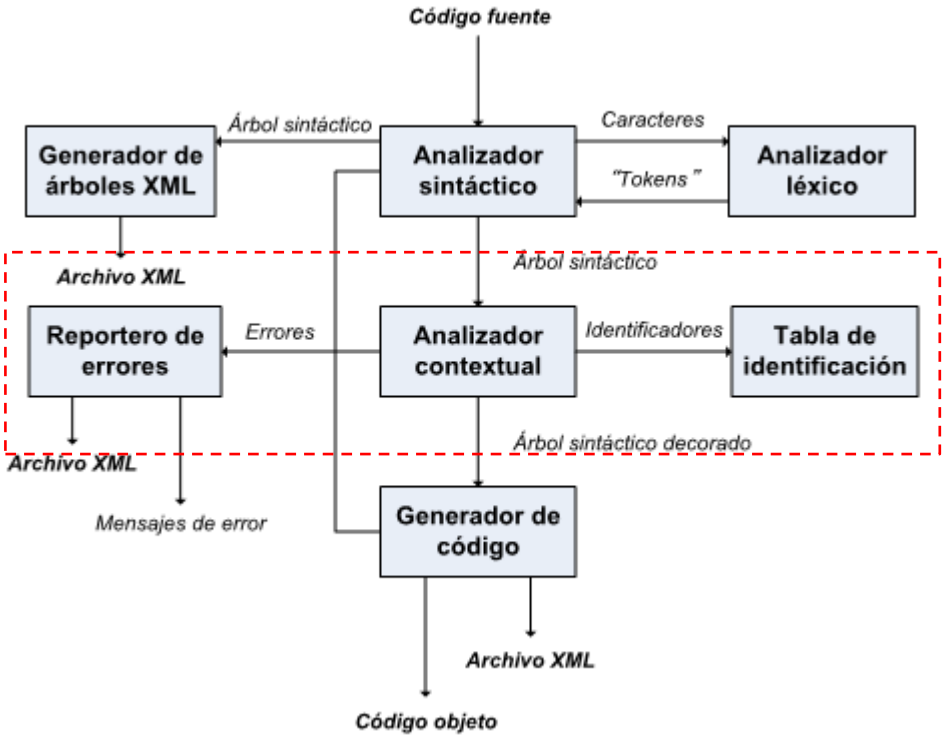

Diagrama 1

A continuación, se presentan los componentes que se probarán en este proyecto:

- Analizador contextual
- Tabla de Identificación
- Reportero de errores

Se validará que el componente Analizador contextual recorra el árbol sintáctico generado por el Analizador sintáctico y que interactúe con la tabla de identificación para validar posibles errores de tipo o declaración.

Se corroborará que el Analizador contextual notifique correctamente al Reportero de errores cuando detecte un error de identificación, de tipo, o de modo de uso de un identificador.

Pruebas de regresión

Este proyecto se está realizando en paralelo con uno cuyo alcance es construir y validar analizadores léxicos y sintáctico para procesar el lenguaje Triángulo. El fin del cliente es poder integrar el analizador contextual con el proyecto del otro grupo, para cual se definió una interfaz del árbol sintáctico (Ast.mli) que es el elemento que une a estos dos módulos. Vía la interfaz compartida, debe ser posible integrar los analizadores sintácticos creados por el otro proyecto con el analizador contextual creado en este proyecto.

Funcionalidades que no se van probar

A continuación, se listan las funciones que no serán probadas durante el proyecto:

- La correcta generación y consistencia del árbol sintáctico; para efectos de nuestro proyecto bastará con recibir actualmente una implementación correcta de una ‘interfaz’ del árbol sintáctico en OCaml.

Enfoque de pruebas (estrategia)

Pruebas funcionales

Debido a las limitaciones con las herramientas del lenguaje Ocaml, se optará por realizar las pruebas usando OUnit para las pruebas unitarias. Estas comprenderán las pruebas de elementos como el manejo correcto de la tabla de Identificación.

En general, se procederá a generar casos de prueba conforme a esta estrategia:

- Aplicar las pruebas para validar el correcto análisis sintáctico. Esta es una regresión de las pruebas realizadas por el grupo encargado de hacer los analizadores léxicos y sintácticos.
- Crear casos de prueba 'positivos' para probar que los programas que respetan las restricciones contextuales (de identificación, de tipo, o de modo de uso de un identificador) son aceptados por el Analizador contextual.
- Crear casos de prueba 'negativos' para probar todo posible error contextual (de identificación, de tipo, o de modo de uso de un identificador) es detectado por el Analizador contextual.

Pruebas no funcionales

El código al ser usado con fines educativos es de alta importancia que siga los criterios de Clean Code y algún estándar de programación.

Criterios de aceptación o de rechazo

El código deberá cumplir con los siguientes estándares para ser aceptado:

- El 100% de las pruebas unitarias deben de salir exitosas
- Deberá cumplir con el estándar de codificación de OCaml (definido por INRIA)
- Deberá hacer cumplir el 100% de las reglas de declaración de variables.
- Deberá hacer cumplir el 100% de las reglas de comprobación de tipos.
- Deberá hacer cumplir el 100% de las reglas de modo de uso de identificadores.
- El código deberá poder correr un código de 100 líneas en menos de 3 min.

Entregables

Los entregables de las pruebas serán:

- El inventario completo de los casos de prueba diseñados, con documentación de su objetivo y resultados esperados.
- Un reporte de pruebas especificando las pruebas realizadas, su resultado y descripción en caso de que sea fallida.
- Un reporte sobre la herramienta de pruebas OUnit, describiendo cuál es su funcionamiento (sitio web de descarga, instalación y uso) y cuáles son los resultados que genera su aplicación.

Recursos

Requerimientos de entornos – Hardware

El entorno del proyecto va ser el de una computadora que tenga como mínimo las siguientes características:

- 512 MB de memoria RAM
- 500 MB de espacio de disco duro

Requerimientos de entornos – Software

La computadora donde se realizarán las pruebas debe tener lo siguiente instalado:

- El lenguaje Ocaml
- Algún editor de archivos (Sublime Text, Notepad++, ...) que se ejecute sobre la plataforma de Windows.

Personal

Se requerirá solamente de 2 personas para realizar las pruebas:

- 1 Tester: El que realizará la aplicación las pruebas en el software.
- 1 Analista: El que analizará los datos obtenidos de las pruebas y determinará si la prueba fue exitosa o fallida.

El diseño de los casos de prueba estará a cargo de ambos miembros del equipo los cuales harán cada uno casos independientes. Luego la validación de dichos casos se someterá a revisión intercambiando lo elaborado por cada miembro.

De esta forma ambos miembros del equipo tendremos roles de tester y de analista a lo largo del diseño, validación, ejecución y análisis de las pruebas.

Herramientas de pruebas requeridas

Se busca realizar pruebas de manera automática que permitan un ahorro del tiempo y esfuerzo empleado para estas, para esto se investigó y se decidió utilizar las siguientes herramientas:

- Ounit
- Ocamllint

Entrenamiento

Para poder realizar las pruebas se necesitará de entrenamiento en el uso de:

- Ounit
- Ocamllint

La adquisición de este conocimiento se logrará mediante autoestudio.

Planificación y organización

Matriz de responsabilidades

Actividad / Recurso	Christian León	Gabriel Ramírez
Ejecutar pruebas unitarias	I	R
Ejecutar pruebas de Ocamllint	I	R
Realizar reporte de los resultados	R	C

- R: Responsable
- A: Aprobador
- C: Consultado,
- I: Informado

Dependencias y Riesgos

El tiempo de prueba en cada iteración puede llegar a ser menor a 1 día debido a que la cantidad de tiempo para estudiar e implementar el código es de 1 semana aproximadamente.