METODOS FORMALES INGENIERÍA DEL SOFTWARE

PRÁCTICA 1: ANALIZADORES ESTÁTICOS AUTOMÁTICOS DE CÓDIGO

Introducción

Los analizadores estáticos son herramientas software que escanean el código fuente de un programa y detectan posibles defectos y anomalías. Analizan el código del programa y así reconocen los tipos de sentencias en el programa. Pueden detectar si las sentencias están bien formadas, hacer inferencias sobre el flujo de control del programa y, en muchos casos, calcular el conjunto de todos los posibles valores para los datos del programa. Complementan las facilidades de detección de errores proporcionadas por el compilador del lenguaje. Pueden utilizarse como parte del proceso de inspección o como una actividad separada del proceso V & V.

¿Qué hay que hacer?

Selecciona una herramientas de análisis estático para cualquier lenguaje que conozcas, pruébala y coméntala.

Algunos cualidades o criterios que podéis analizar o comentar son:

1. *Tipo de licencia*.
2. *Usabilidad.* El desarrollador tiene que encontrar en la herramienta facilidad de instalación, de uso y de obtención de resultados. Posibles categorías:
   - Ayuda de la herramienta y la documentación,
   - Tiempo de instalación y el tiempo de configuración del entorno
   - Actualizaciones
   - Facilidad para interpretar los resultados que aborda la herramienta.
3. *Eficiencia*. Tiempo de ejecución y consumo de recursos de cada herramienta.
4. *Extensibilidad*. En el sentido de que permita añadir reglas fácilmente. Si la herramienta puede extenderse podremos adaptarla a nuestro proyecto y en consecuencia no habrá límite en la cantidad de *bugs* que pueden ser detectados.
5. *Técnica de análisis usado*. Según Nicole et al[1]
   - Análisis basado en AST. Ciertas herramientas no analizan el código fuente tal y como está escrito, en su lugar transforman el código en una representación en árbol (AST

1 Christopher, C. N. (2006). Evaluating Static Analysis Frameworks. *Analysis*, *pág*, 1-17.

Abstract Syntax Tree, Árbol de Sintaxis Abstracta) que refleja la estructura del fichero.

- Análisis *dataflow*. Es la técnica más utilizada en el análisis estático. Dentro de este tipo de análisis podemos encontrar varias categorías:
- Análisis *style-checkers*. Se centran exclusivamente en la estructura léxica y sintáctica. Con este tipo de analizadores se detectan espacios en blanco, código demasiado extenso, nombres de variables que no siguen las convenciones indicadas, líneas de código mal indentadas.
- Análisis *bug-checker*. Método basado en búsqueda de patrones o reglas (*pattern matchin*g) predefinidos por la herramienta o diseñados por el usuario de la misma. Se utilizan más que los *style-checkers*. Ejemplos: FindBugs, PMD y JLint.
- *Theorem proving*. Demostración del teorema. Construye una prueba de los requerimientos mediante inducción lógica sobre la estructura del programa. ESC/Java2 es un ejemplo de analizador de este tipo.

6. *Frameworks* disponibles e integración en IDEs. A mayor número de plugins más posibilidades de utilizarla en distintas plataformas.

¿Qué hay que entregar?

Documento pdf en el que se indique la herramienta elegida y el análisis de la misma. Si se prueba con algún código, indica el tamaño de éste (nº aproximado de líneas).