Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3745 - Testing

– Programa de curso –

Profesores : Juan Pablo Sandoval (`juanpablo.sandoval@uc.cl`)
: Maximiliano Narea Carvajal (`manarea@uc.cl`)
Requisitos : IIC2143
Sitio Web : Canvas (`http://cursos.canvas.uc.cl/`)
Clases : Martes y Jueves, módulo 6, (16:00 - 17:20)

1. Presentación del curso

Este curso ofrece una visión integral sobre las pruebas de software, abarcando desde los fundamentos y técnicas básicas hasta las metodologías más avanzadas de análisis estático y dinámico. Los estudiantes explorarán diversas prácticas y herramientas que les permitirán identificar y corregir defectos en el software de manera eficiente. Además, se abordarán técnicas modernas de generación automática de pruebas, incluyendo el uso de algoritmos genéticos y modelos de lenguaje, para preparar a los estudiantes a enfrentar los desafíos actuales en el campo de la ingeniería de software.

2. Objetivos de aprendizaje

Al finalizar el curso, los alumnos serán capaces de:

- Implementar técnicas avanzadas de fuzzing sintáctico para identificar defectos en el software.
- Utilizar técnicas de auto-reparación de programas para mejorar la calidad y la mantenibilidad del software.
- Aplicar métodos de análisis dinámico para monitorear y mejorar la calidad del código durante su ejecución.
- Emplear técnicas de fuzzing semántico para probar la lógica del software y detectar errores difíciles de encontrar.
- Aplicar técnicas de pruebas para requisitos no funcionales.
- Diseñar y ejecutar pruebas de API y pruebas de sistema, asegurando la integridad y el rendimiento de las aplicaciones.

3. Contenido (Content)

A continuación se presenta un desglose detallado de los contenidos del curso:

1. **Introducción**
2. **Fundamentos**
   - Atributos de la calidad (Quality Attributes)
   - Defectos (Bugs)
   - Pruebas de Software y principios fundamentales de las pruebas de software
   - Pruebas y el ciclo de desarrollo de software
   - Verificación y Validación
3. **Pruebas de Unidad (Unit Testing)**
   - Creación de Pruebas de Unidad (Creating Unit Tests)
   - Ventajas y Dificultades (Advantages and Challenges)
   - Dobles de Prueba (Test Doubles: Stubs and mocks)
   - Cobertura de Código (Code Coverage)
   - Pruebas de Caja Blanca y Negra (White Box and Black Box Testing)
4. **Syntactic and Semantic Fuzzing**
   - Random Fuzzing
   - Mutation-Based Fuzzing
   - Search-Based Fuzzing
   - Taint Analysis
   - Concolic Fuzzing
   - Symbolic Fuzzing
   - Mutation Testing
5. **Técnicas de Detección de Errores (Fault-Localization Techniques)**
   - Spectrum-Based Fault Localization
   - Mutation-Based Fault Localization
   - Predicate Switching
   - Stack Trace Fault Localization
6. **Análisis Estático (Static Analysis)**
   - Árbol de Sintaxis Abstracta (Abstract Syntax Tree)
   - El patrón Visitor (The Visitor Pattern)
   - Code Smells (Code Smells)
   - Detección Automática de Code Smells (Automatic Detection of Code Smells)
   - Transformaciones de Código (Code Transformations)
   - Auto-reparación de Programas (Auto-repair of Programs)

7. **Análisis Dinámico (Dynamic Analysis)**
    - Muestreo de Ejecución (Execution Sampling)
    - Instrumentación para Monitoreo (Instrumentation Profiling)
    - Code Smells (Code Smells)
    - Detección Automática de Code Smells (Linters)
8. **Diseño de Pruebas Manuales**
    - Análisis de Límites (Boundary Analysis)
    - Particiones de Equivalencia (Equivalence Partitioning)
    - Tablas de Decisión (Decision Tables)
    - Transición de Estados (State Transition)
9. **Pruebas de Integración**
    - API Testign
    - Testeando Controladores
10. **Pruebas de Systema**
    - End to End Testing
11. **Pruebas de Rendimiento (Performance Testing)**
    - Tiempo de Ejecución y Memoria (Execution Time and Memory)
    - Load and Stress Testing (Load and Stress Testing)
12. **Generación Automática de Pruebas de Unidad (Automatic Unit Test Generation)**
    - Generación basada en Algoritmos Genéticos (Generation Based on Genetic Algorithms)
    - Generación basada en Large Language Models (Generation Based on Large Language Models)
    - Generación Automática de Oráculos (Automatic Generation of Oracles)

4. Metodología

El curso se desarrollá en clases expositivas, tutoriales, actividades y tareas. El curso contara con dos canales de información. Los anuncios y enlaces relevantes seran publicados a traves de la plataforma canvas. Todo el material del curso, incluyendo los apuntes de clases, enunciados de tareas, pautas de corrección de interrogaciones, y ayudantias, estarán disponibles en el sitio web del curso.

El curso contará con un examen (**E**), tareas (**T**), y actividades (**A**) sobre los contenidos del curso, presentaciones, y lecturas complementarias. A lo largo del semestre los estudiantes trabajarán de forma individual para desarrollar cuatro tareas practicas y actividades. Durante el desarrollo tendrán que aplicar diferentes aplicar las técnicas vistas en clase a problemas de complejidad media-alta.

5. Evaluaciones

El curso se evalúa mediantes los siguientes tipos de actividades.

- Examen (**E**) - 40 %: El mismo contendra preguntas relacionadas a las tareas, actividades, lecturas complementarias y conceptos vistos en clase.
- Actividades (**A**) - 20 %: En el curso definiremos como actividad, a los ejercicios que el profesor dejara al finalizar algunas clases. Los mismos son pequeños que se pudieran resolver durante clase. Se realizarán alrededor de 7 actividades pequeñas personales de igual valor y de carácter sumativo, sobre tópicos visto en cátedra. El objetivo de cada actividad es aplicar lo recien aprendido en clases anteriores. La nota final de las actividades (**A**) está dada por el promedio de estas.
- Tareas (**T**) - 40 %: Se realizarán 4 tareas (**T**) individuales. Las tareas permitiran al estudiante aprender en profundidad 4 de los topicos del curso: fuzzing, analisis dinamico, analisis estatico, integración continua, y end two end testing.

La nota final se calcula de la siguiente manera:

$$\mathbf{NF} = 0{,}2 * \mathbf{A} + 0{,}4 * \mathbf{T} + 0{,}4 * \mathbf{E}$$

Fechas. Las interrogaciones se realizaran en las fechas que aparecen en busca cursos.

- Examen (**E**): 12 de Noviembre - 17:30
- Recuperativo del Examen (**ER**): 13 de Diciembre - horario por confirmar.
- Las semanas que contaran con actividades y tareas esta en la planificación del curso en canvas.

Inasistencia Justificada.

- Las personas que no puedan asistir al examen deben presentar un justificativo de la DIPRE para poder rendir el examen recuperativo(ER). La nota del examen sera la nota que saquen en el recuperativo.
- Como ultima oportunidad, aquellos que reprueben el examen pueden asistir al examen recuperativo(ER) sin necesidad de justificación. Si sacan una nota >= 4,95 su nota del examen sera 4 caso contrario, sera la nota que saquen menos uno ($\mathbf{E} = \mathbf{ER} - 1$).
- Si no asisten al examen y no presentan justificativo, no podrán rendir el examen recuperativo y se les asignará la nota mínima.

Requisitos de Aprobación. Para aprobar el curso se deben cumplir con los siguiente requisitos. Las notas **T** y **E** deben ser mayores o iguales a 3,95. En caso de cumplir los criterios anteriores, la nota final de reprobación será:

$$\mathbf{F} = min(3{,}9, \mathbf{NF})$$

.

Retrasos en Tareas o Actividades. Se podran entregar los actividades hasta con 2 dias de retraso. Si se entrega al dia siguiente de la hora/fecha de entrega se descontara 1 punto. Si se entrega durante el segundo dia de la hora/fecha de entrega se descontara 2 puntos. Actividades con mas de 2 dias de retraso no seran consideradas y tendran la nota minima. Para las tareas cada persona tendra 1 cupon, el cupón le permitira entregar una tarea hasta con 2 dias de retraso, sin penalización. Se recomienda utilizar sabiamente el cupon. En el caso que una tarea sea entregada con retraso y no contar con un cupon. Se asignará la nota minima en esa tarea.

Solicitudes de Recorrección. Cada tarea y interrogación tendrá un periodo para recibir solicitudes de re corrección. Despues de ese plazo no es posible solicitar una recorrección.

6. Bibliografía

- Dorothy Graham & Erik van Veenendaal & Isabel Evans & Rex Black - *FOUNDATIONS OF SOFTWARE TESTING*. ISTQB CERTIFICATION.
- P. Ammann & Offutt *Introduction to Software Testing (2nd ed.)*. Cambridge University Press, 2016.
- J.C Huang *Software Error Detection Through Testing and Analysis (1st ed.)*. Wiley, 2009.
- A. Whittaker *Exploratory Software Testing (1st ed.)*, Addison-Wesley Professional, 2009.
- Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler The Fuzzing Book – *Tools and Techniques for Generating Software Tests*
- Stéphane Ducasse, Guillermo Potilo, Juan Pablo Sandoval – *Testing in Pharo*

7. Política de Integridad Académica

Los/as estudiantes de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los/as estudiantes que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada estudiante conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un/a estudiante para los efectos de la evaluación de un curso debe ser hecho individualmente por el/la estudiante, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros.

En particular, si un/a estudiante copia un trabajo, o si a un/a estudiante se le prueba que compró o intentó comprar un trabajo, obtendrá nota final 1.1 en el curso y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral.

Por "copia" se entiende incluir en el trabajo presentado como propio, partes hechas por otra persona. En caso que corresponda a "copia" a otros estudiantes, la sanción anterior se aplicará a todos los involucrados. En todos los casos, se informará a la Dirección de Pregrado de la Escuela de Ingeniería para que tome sanciones adicionales si lo estima conveniente.

También se entiende por copia extraer contenido sin modificarlo sustancialmente desde fuentes digitales como Wikipedia o mediante el uso de asistentes inteligentes como ChatGPT o Copilot. Se entiende que una modificación sustancial involucra el análisis crítico de la información extraída y en consecuencia todas las modificaciones y mejoras que de este análisis se desprendan. Cualquiera sea el caso, el uso de fuentes bibliográficas, digitales o asistentes debe declararse de forma explícita, y debe indicarse cómo el/la estudiante mejoró la información extraída para cumplir con los objetivos de la actividad evaluativa.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Estudiante de la Pontificia Universidad Católica de Chile (`https://registrosacademicos.uc.cl/reglamentos/estudiantiles/`). Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.

Compromiso del Código de Honor. Este curso suscribe el Código de Honor establecido por la Universidad, el que es vinculante. Todo trabajo evaluado en este curso debe ser propio. En caso que exista colaboración permitida con otros/as estudiantes, el trabajo deberá referenciar y atribuir correctamente dicha contribución a quien corresponda. Como estudiante es un debe conocer el Código de Honor (`https://www.uc.cl/codigo-de-honor/`)