Дисциплина "Принципы статического анализа исходного кода"

3 курс

Язык преподавания:

Читается на русском, материалы на английском.

Аннотация	Статический анализ позволяет найти слабые места в исходном коде, не запуская его. Современные компании-разработчики ПО широко применяют инструменты статического анализа для обеспечения качества исходного кода, а также создают собственные решения для специфических задач. Цель данного курса – научить студентов основам статического анализа исходного кода. Лекции позволят приобрести необходимые теоретические знания. Семинары позволят получить практические навыки использования и создания инструментов статического анализа. Курс будет полезен тем, кому интересно: глубже понимать современные языки программирования и компиляторы; создавать собственные языки; научиться использовать инструменты статического анализа; разрабатывать собственные инструменты анализа исходного кода.
Цели освоения курса	1. Понимать, как устроены языки программирования и компиляторы. 2. Изучить принципы статического анализа исходного кода. 3. Научиться использовать и создавать инструменты статического анализа.
Планируемые результаты обучения	1. Понимать цели и задачи, а также основные принципы статического анализа исходного кода. 2. Знать основные структуры данных, используемые для представления исходного кода при статическом анализе. 3. Понимать принципы работы основных видов статического анализа. 4. Уметь использовать и создавать инструменты статического анализа.
Технические требования	1. Для лекций – лекторий с экраном на 20 человек. 2. Для семинаров – компьтерный класс с компьютерами со следующим ПО: Linux Ubuntu, Java, IntelliJ IDEA, CLion.

Содержание учебной дисциплины

№	Тема лекции	Тема семинара

1	Введение в статический анализ.	Знакомство с инструментами статического анализа: Checkstyle, PMD, SpotBugs, JaCoCo, ErrorProne, Infer, Clang-Tidy.
2	Метрики ПО.	Цикломатическая сложность, связанность (cohesion и coupling). Инструмент JaCoCo.
3	Лексический анализ.	Регулярные выражения и грамматика лексического разбора на примере ANTLR.
4	Синтаксический анализ.	Грамматики синтаксичесого разбора, деревья разбора, и средства анализа деревьев разбора в ANTLR.
5	Внутреннее представление.	Абстрактные синтаксические деревья, таблицы символов и граф потока выполнения.
6	Анализ абстрактных синтаксических деревьев.	Построение абстактных синтаксических деревьев. Вывод типов.
7	Анализ потока выполнения.	Анализ графов потока выполнения.
8	Анализ потока данных.	Форма статического единственного присваивания (SSA-форма).
9	Межпроцедурный анализ.	Межпроцедурный анализ.
10	Символическое исполнение и SMT решатели.	Использование языка SMT-LIB и решателя ограничений Z3.
11	Дедуктивкая верификация.	Инструменты дедуктивной верификации: JML, Why3, Coq.
12	Анализ на основе майнинга данных. Увтоматическое исправление ошибок.	Паттерны изменений в исходном коде. Майнинг частых подграфов.
13	Анализ Java байткода.	Анализ Java байткода при помощи ObjectWeb ASM.
14	LLVM и Clang Static Analyzer.	Статические анализаторы для C++: Clang-Tidy и Clang Static Analyzer.

15	EOLANG и Polystat.	Язык EOLANG и инструмент статического анализа Polystat.

Текущий контроль

Форма контроля	Критерии оценивания	Возможность пересдачи	Условия и формат пересдачи	Примеры заданий
Экзамен.	Устный экзамен. Теоретический вопрос и практическая задачка со временем на подготовку. Оценка от 0 до 10 баллов.	Да	При оценке < 4. Формат пересдачи какой же, как у экзамена.	
Бонусные баллы за работу на семинарах.	Доклад на семинаре дает 1 (по 10-бальной шкале) бонусный балл.			

Промежуточная аттестация

Формула.

Итоговая оценка = Минимум (Экзамен + Бонус, 10)

Правила округления.

Оценка округляется в большую сторону, если ее дробная часть >= 0.5. В противном случае – меньшую.

Есть ли блокирующие оценки, и если есть, то какую часть оценки они блокируют.

Нет.

Предусмотрены ли автоматы.

Нет.

Условия для выставления автомата.

Нет.

Особенности пересдачи.

Пересдача не отличается от экзамена.

Литература

Рекомендуемая основная литература:

1. Ульман Джеффри Д., Сети Рави. “Компиляторы. Принципы, технологии и инструментарий”. 2016.
2. Jonathan Aldrich, Claire Le Goues, and Rohan Padhye. "Program Analysis". Carnegie Mellon University. 2022. https://cmu-program-analysis.github.io/2022/resources/program-analysis.pdf
3. Steven Muchnick. “Advanced Compiler Design and Implementation”. 1st edition. 1997.
4. Terence Parr. “Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages”. 1st Edition. 2010.

Рекомендуемая дополнительная литература:

1. Аулер Рафаэль, Лопес Бруно Кардос. “LLVM. Инфраструктура для разработки компиляторов”. 2015.
2. Flemming Nielson, Hanne Riis Nielson, Chris Hankin. “Principles of Program Analysis”. 1999.
3. Anders Møller and Michael I. Schwartzbach. “Static Program Analysis”. Aarhus University. 2021. https://cs.au.dk/~amoeller/spa/