Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"

Факультет компьютерных наук
Департамент программной инженерии

Рабочая программа дисциплины
Верификация программ

для образовательной программы «Программная инженерия»
направления подготовки 09.03.04 «Программная инженерия»
уровень - бакалавр

Разработчик программы
Петренко А.К, проф., д.ф.-м.н., petrenko@ispras.ru

Одобрена на заседании департамента программной инженерии «___»_____ 2016 г.
Руководитель департамента Авдошин С.М. _____

Утверждена Академическим советом образовательной программы «___»_____ 2016 г.,
№ протокола_____

Академический руководитель образовательной программы Шилов В.В. _____

Москва, 2016

1 Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает минимальные требования к знаниям и умениям студента и определяет содержание и виды учебных занятий и отчетности. Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов образовательной программы «Программная инженерия» направления подготовки 09.03.04 «Программная инженерия», изучающих дисциплину "Верификация программ". Программа разработана в соответствии с образовательным стандартом Национального исследовательского университета «Высшая школа экономики» по направлению 09.03.04 «Программная инженерия»

2 Цели освоения дисциплины

Цель курса – Целью курса является изучение основ построения оптимизирующих статических и динамических компиляторов современных языков программирования, учитывающих особенности архитектур современных компьютеров; ознакомление студентов с предметом и основными подходами к формальной спецификации и верификации программ.

Задачами данного курса являются:

- освоение студентами базовых современных достижений формальных методов в разработке программного обеспечения для критических по безопасности систем;
- формирование практических навыков применения формальных методов при проектировании и разработке программного обеспечения.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины студент должен:

1. Знать:

– фундаментальные понятия, теории современного системного программирования;
– основные виды формальных спецификаций, виды моделей программных систем, формализация требований к программному обеспечению;
– основные подходы к анализу свойств программ;
– методы верификации и тестирования ПО на основе формальных моделей;
– методы автоматизированного доказательства теорем.

2. Уметь:

– формализовывать требования к программному обеспечению в виде алгебраических, имплицитных и эксплицитных спецификаций;
– осуществлять построение моделей программ, аналитическую верификацию последовательных программ;
– осуществлять формальное доказательство свойств программ в инструментах автоматизированного доказательства теорем.

3. Иметь навыки (приобрести опыт):

– освоения большого объема информации;
– самостоятельной работы в Интернете;
– культурой разработки и реализации системного программного обеспечения современных компьютеров;
– использования формальных методов для доказательства корректности программ;

В результате освоения дисциплины студент должен обладать следующими компетенциями:

Универсальные компетенции:

- Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза (УК-3)
- Способен оценивать потребность в ресурсах и планировать их использование при решении задач в профессиональной деятельности (УК-4)
- Способен работать с информацией: находить, оценивать и использовать информацию из различных источников, необходимую для решения научных и профессиональных задач (в том числе на основе системного подхода) (УК-5)
- Способен вести исследовательскую деятельность, включая анализ проблем, постановку целей и задач, выделение объекта и предмета исследования, выбор способа и методов исследования, а также оценку его качества (УК-6)

Профессиональные компетенции:

- Способен использовать методы и инструментальные средства исследования объектов профессиональной деятельности (ПК-3)
- Способен обосновать принимаемые проектные решения, осуществлять постановку и выполнение экспериментов по проверке их корректности и эффективности (ПК-4)
- Способен проектировать, конструировать и тестировать программные продукты (ПК-10)
- Способен читать, понимать и выделять главную идею прочитанного исходного кода, документации (ПК-11)
- Способен использовать различные технологии разработки программного обеспечения (ПК-16)
- Способен применять основные методы и инструменты разработки программного обеспечения (ПК-17)

4 Место дисциплины в структуре образовательной программы

Изучение данной дисциплины базируется на знаниях, полученных студентами при освоении учебных дисциплин:

- «Дискретная математика»,
- «Программирование»,
- «Алгоритмы и структуры данных»,
- «Архитектура вычислительных систем»,
- «Операционные системы».

5 Тематический план учебной дисциплины

№	Название раздела	Всего часов	Аудиторные часы			Самостоятельная работа
			Лекции	Семинары	Практические занятия	
1	Понятие модели и спецификации программы. Парадигмы моделирования поведения программ	24	4		4	16
2	Дедуктивная верификация. Метод Флойда	36	8		8	20
3	Язык моделирования Event-B	46	8		8	30
4	Дедуктивная верификация Си-программ	46	8		8	30
5	Тестирование соответствия программ и их моделей	38	4		4	30
Итого:		**190**	**32**		**32**	**126**

6 Формы контроля знаний студентов

Тип контроля	Форма контроля	3 год	
		3 модуль	4 модуль
Текущий			1-6 неделя
	Домашнее задание		*
Итоговый	Экзамен		*

7 Критерии оценки знаний, навыков

Оценки по всем формам текущего контроля выставляются по 10-ти балльной шкале.

8 Порядок формирования оценок по дисциплине

Оценка по курсу состоит из оценки за выполнение домашнего задания $O_{дз}$ (10 баллов) и оценки за итоговый устный экзамен (10 баллов). В диплом выставляется результирующая оценка по учебной дисциплине, которая формируется по следующей формуле:

$$O_{результ} = 0{,}4*O_{дз} + 0{,}6*O_{экз}$$

9 Содержание дисциплины

№№	Разделы и темы лекционных занятий	Содержание
1	Понятие модели и спецификации программы. Парадигмы моделирования поведения программ	Методологии разработки программ, основанные на использовании моделей. Формальные методы анализа и разработки программ. Понятие спецификации программы и спецификации языка программирования. Примеры методологий и языков моделирования: VDM, UML, Design-by-contract, JML, Eiffel, Spec#, AADL
		Явные/императивные модели поведения Неявные, аппликативные модели поведения Алгебраические/аксиоматические спецификации свойств программ. Задачи построение и трансформацию моделей поведения.
2	Дедуктивная верификация. Метод Флойда	Аналитическая верификация программ. Общая схема формальной верификации. Формальная модель требований. Формальная модель программы. Метод индуктивных утверждений Флойда. Метод фундированных множеств. Логика Хоара. Частичная и полная корректность программы.
3	Язык моделирования Event-B	Основы разработки и верификации моделей на Event-B с использованием инструмента Rodin/
		Изучение примеров использования языка Event-B: 1) Модель системы управления трафиком автомобилей на мосту с реверсивным движением 2) Модель контроллера механического пресса 3) Модель простого протокола передачи данных
		Генерация кода программы из Event-B-моделей.
		Верификация Event-B-моделей. Уточнение моделей (refinement).

4	Дедуктивная верификация Си-программ	Построение моделей последовательных Си программ. Введение в язык ACSL (ANSI C Specification Language). Формальная спецификация свойств Си программ на языке ACSL. Архитектура платформы Frama-C. Плагин Jessie для дедуктивного анализа C-программ (платформа Why). Использование ACSL/Frama-C/Jessie. Автоматизация верификации программ. Синтез инвариантов циклов. Генерация условий верификации. Доказательство условий верификации.
5	Тестирование соответствия программ и их моделей	Основы тестирования на основе формальных моделей (Model Based Testing - MBT). Виды моделей. Архитектура систем тестирования на основе моделей.

10 Оценочные средства для текущего контроля и аттестации студента

Перечень контрольных вопросов для экзамена

1. Практическое использование формальных методов. Место формальных методов в процессе проектирования.
2. Общая схема формальной верификации. Формальная модель требований. Формальная модель программы. Формальное соответствие модели программы модели требований.
3. Методы Флойда. Моделирование программ с помощью блок-схем. Синтаксическая структура блок-схем. Операционная семантика блок-схем.
4. Методы Флойда. Формальная верификация программ. Частичная и полная корректность программы.
5. Метод Флойда доказательства частичной корректности блок-схем.
6. Метод Флойда доказательства полной корректности блок-схем.
7. Язык ACSL. Формальная спецификация C-программ на языке ACSL.
8. Автоматическая генерация условий корректности. Методы прямого и обратного прослеживания.
9. Автоматическая проверка условий корректности. Примеры разрешимых теорий.
10. Язык Event-B.
11. Тестирование на основе формальных моделей. Виды формальных моделей.
12. Архитектура систем тестирования на основе формальных моделей.

11 Учебно-методическое и информационное обеспечение дисциплины

11.1 Базовый учебник

1. В. В. Кулямин. Технологии программирования. Компонентный подход // М.: ИНТУИТ-Бином, 2007. 463 с.
2. Р. Андерсон, "Доказательство правильности программ", Москва, Мир, 1982.
3. Б. Мейер. Основы объектно-ориентированного программирования. "Русская редакция" 2005.

11.2 Основная литература

1. R. W. Floyd, "Assigning meanings to programs", Proc. Symp. Appl.
2. N. Francez, "Verification of programs", Addison-Wesley Publishers Ltd., 1992
3. C.A.R. Hoare, An axiomatic basis for computer programming, Comm. ACM 12 (1969) 576-580

4. ANSI/ISO C Specification Language. http://frama-c.com/acsl.html
5. Jean-Raymond Abrial - Modeling in Event-B: System and Software Engineering - основная книга.
6. A Concise Summary of the Event B mathematical toolkit - полное описание нотации Event-B.
7. Rodin User’s Handbook - руководство по использованию Rodin и Event-B.

11.3 **Дополнительная литература**

1. Proving with computer assistance: practical part (Course - 2IF40, Department of Mathematics and Computer Science, Technical University of Eindhoven) http://www.win.tue.nl/~fdechesn/2IF48/
2. Virgile Prevosto. ACSL Mini-Tutorial. http://frama-c.com/download/acsl-tutorial.pdf
3. Jochen Burghardt, Jens Gerlach, Kerstin Hartig, Hans Pohl, Juan Soto. ACSL By Example. Fraunhofer FIRST, May 2010 http://www.first.fraunhofer.de/owx_download/acsl-by-example-5_1_0.pdf.