МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ярославский государственный университет им. П.Г. Демидова

Кафедра теоретической информатики

УТВЕРЖДАЮ
Декан факультета ИВТ
_____ Д.Ю. Чалый
«_24_» ___мая_____ 2022 г.

Рабочая программа дисциплины
«Верификация программного обеспечения»

Научная специальность
1.1.5 Математическая логика, алгебра, теория чисел и дискретная математика

Форма обучения
очная

Программа рассмотрена
на заседании кафедры
от 15 марта 2022 г.,
протокол № 8

Программа одобрена НМК
факультета ИВТ
протокол № 6 от
18 апреля 2022 г.

Ярославль

1. Цели освоения дисциплины

Дисциплина «Верификация программного обеспечения» обеспечивает приобретение знаний и умений в соответствии с ФГОС ВПО, содействует формированию мировоззрения и развитию способности понимать и применять в исследовательской и прикладной деятельности современный аппарат разработки и анализа корректности алгоритмов, умению моделировать программы, специфицировать и исследовать их свойства. Кроме того, дисциплина должна обеспечивать развитие логического, эвристического и алгоритмического мышления и давать представление о месте и роли алгоритмов в современном мире, мировой культуре и истории, должна содействовать целевой направленности образования, умению разрабатывать и анализировать корректность алгоритмов, изучать их свойства.

Цель дисциплины «Верификация программного обеспечения» – изучение общих основ моделирования программ, способов спецификации свойств программ, методов и приемов исследования свойств программ, анализа и доказательства корректности программ и их моделей.

2. Место дисциплины в структуре ОП аспирантуры

Дисциплина «Верификация программного обеспечения» относится к вариативной части (дисциплина по выбору) ОП аспирантуры.

Дисциплина необходима для решения задач построения корректных программ, анализа корректности и исследования различных свойств программ и их моделей. Данная дисциплина направлена на подготовку к сдаче кандидатского экзамена по соответствующей научной специальности.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения ОП аспирантуры

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Знать:

- современные методы спецификации, построения и анализа корректности программ;
- способы моделирования программ;
- способы спецификации и анализа свойств программ;
- способы дедуктивного доказательства корректности программ;
- стратегию спецификации и доказательства корректности программ, написанных на процедурном языке высокого уровня;
- методы автоматической проверки корректности программной модели.

Уметь:

- проводить тестирование программ ПЛК;
- применять инструментальные средства верификации для анализа корректности программ;
- проводить спецификацию программ на языке предикатов;
- применять метод доказательства теорем для доказательства корректности программ, написанных на языках высокого уровня;
- строить программные модели и проводить спецификацию и верификацию программных свойств на языке темпоральной логики;
- строить модели систем реального времени с помощью формализма временных автоматов и проводить спецификацию свойств таких систем на языках временных темпоральных логик.

Владеть:

навыками проверки корректности программ;

- формальными методами моделирования и спецификации программ;

- формальными методами анализа корректности программ (алгоритмов): дедуктивным анализом (метод доказательства теорем) и методом проверки модели (model checking).

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 3 зач. ед., 108 акад. час.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
			Контактная работа						
			лекции	практические	лабораторные	консультации	аттестационные испытания	самостоятельная работа	
1.	Дедуктивный анализ корректности программ на примере «простого» языка программирования	2	8	5		1		40	Индивидуальное задание
2.	Построение моделей параллельных и распределенных систем	2	8	5		1		40	Индивидуальное задание
	Всего за 2 семестр		**16**	**10**		**2**		**80**	**Зачет**
	Всего		**16**	**10**		**2**		**80**	

Содержание разделов дисциплины:

1. Дедуктивный анализ корректности программ на примере «простого» языка программирования. Спецификация программ с помощью пред- и постусловий. Доказательство корректности программ относительно спецификации, инвариантов и ограничивающей функции. Построение инвариантов и ограничивающих функций.
2. Построение моделей параллельных и распределенных систем .Асинхронные и синхронные процессы. Взаимодействие процессов. Структура Крипке. Метод проверки модели. Верификация моделей и теория автоматов. Автоматы над бесконечными словами. Структура Крипке как автомат Бюхи. Темпоральная логика линейного времени LTL. Формула LTL как обобщенный автомат Бюхи. Редукция автомата Бюхи для формулы LTL. Пересечение языков структуры Крипке и автомата Бюхи. Проверка пустоты автомата Бюхи. Проверка модели «на лету».

5. Образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует аспиранта в системе изучения данной дисциплины. Аспиранты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция (или лекция общего курса) – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Требования к академической лекции: современный научный уровень и насыщенная информативность, убедительная аргументация, доступная и понятная речь, четкая структура и логика, наличие ярких примеров, научных доказательств, обоснований, фактов.

В основу образовательной технологии по дисциплине «Верификация программного обеспечения» помимо традиционных форм лекций положена также форма, состоящая в выполнении аспирантом индивидуальных заданий по темам дисциплины. Имеются четыре индивидуальных здания по дисциплине. Задания должны быть решены письменно или в эл. виде с последующей устной защитой. Первое задание закрывает тематику дедуктивного анализа корректности последовательных программ. Представляет собой небольшую программу, написанную на «простом» языке высокого уровня, корректность которой необходимо доказать относительно спецификации, составляемой аспирантом по словесному описанию требований к программе. При безуспешных со стороны аспиранта попытках построения инварианта, ограничивающей функции и постусловия эти необходимые для выполнения задания компоненты могут быть предоставлены аспиранту. Выполнение второго, третьего и четвертого заданий предполагает моделирование (представление в виде структуры Крипке), спецификацию (запись свойств на языке темпоральной логики) и автоматическую верификацию некоторой параллельной и распределенной программы с помощью свободно распространяемых для учебных целей средств верификации SPIN, SMV и Uppaal. Ошибки, допущенные при выполнении задания, отмечаются подробно преподавателем, ведущим лабораторные занятия. После исправления ошибок задание сдается вновь преподавателю на проверку. Аспиранты, сдавшие все индивидуальные в установленные сроки, после успешного ответа на ряд дополнительных вопросов, закрывающих оставшиеся темы, получают отметку о сдаче экзамена досрочно. Такой подход стимулирует постоянную работу аспирантов в течение семестра и активизирует усвоение материала. Эта технология позволяет проводить индивидуальное обучение аспирантов и дает хорошие результаты для приобретения аспирантами заявленных компетенций. Она дополняется обсуждением общих (типичных) ошибок на лабораторных и лекционных занятиях.

6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем (при необходимости)

В процессе осуществления образовательного процесса используются:

– для формирования текстов материалов для промежуточной и текущей аттестации – программы Microsoft Office, издательская система LaTex;

– для поиска учебной литературы библиотеки ЯрГУ – Автоматизированная библиотечная информационная система "БУКИ-NEXT" (АБИС "Буки-Next");

– свободно распространяемые для учебных целей средства верификации SPIN, SMV и Uppaal.

7. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

а) основная:

1. Кузьмин, Е. В., Верификация моделей программ : учеб. пособие для вузов / Е. В. Кузьмин ; под ред. В. А. Соколова, Ярославль, ЯрГУ, 2008, 174с

2. Кузьмин, Е. В., Верификация моделей программ [Электронный ресурс] : учеб. пособие для вузов / Е. В. Кузьмин ; под ред. В. А. Соколова, Ярославль, ЯрГУ, 2008, 174с

3. Синицын, С. В., Основы разработки программного обеспечения на примере языка C [Электронный ресурс] / С. В. Синицын, О. И. Хлытчиев. - 2-е изд., испр., М., Национальный Открытый Университет ИНТУИТ, 2016, 212с

4. Кузьмин Е.В. Введение в теорию вычислительных процессов и структур. – Учебное пособие, Ярославль, ЯрГУ, 2006. – 140 с.

5. Грис Д. Наука программирования. – М.: Мир, 1984. – 416 с.

6. Карпов Ю. Г. Model Checking. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

7. Кларк, Э. М., Верификация моделей программ : Model Checking : пер. с англ. / Э. М. Кларк мл., О. Грамберг, Д. Пелед ; под ред. Р. Смелянского, М., МЦНМО, 2002, 416с

б) дополнительная:

1. Синицын, С. В., Верификация программного обеспечения [Электронный ресурс] : курс / С. В. Синицын, Н. Ю. Налютин, М., Интернет-Университет Информационных Технологий, 2007, 367с

2. Кузьмин, Е. В., Структурированные системы переходов : учеб. пособие для вузов / Е. В. Кузьмин, В. А. Соколов, М., ФИЗМАТЛИТ, 2006, 174с

3. Карпов Ю. Г. Теория автоматов. – СПб.: Питер, 2003. – 208 с.

4. Математическая логика в программировании: сб. статей 1980—1988 гг.: Пер. с англ. – М.: Мир, 1991. – 408 с.

5. Минский М. Вычисления и автоматы – М.: Мир, 1971. – 268 с.

6. Непомнящий В. А., Рякин М. О. Прикладные методы верификации программ – М.: Радио и связь, 1988. – 256 с.

7. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984. 263с.

8. Хоар Ч. Взаимодействующие последовательные процессы. М.: Мир, 1989. 264 с.

9. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. – М.: Вильямс, 2002. – 528 с.

в) ресурсы сети «Интернет»

1. SMV. Symbolic Model Verifier. Carnegie Mellon University. http://www.cs.cmu.edu/~modelcheck/smv.html
2. SPIN. http://spinroot.com/spin/whatispin.html
3. UPPAAL. http://www.uppaal.com

8. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Аудитории, оборудованные для проведения лекций, практических занятий и консультаций, фонд библиотеки, компьютерная техника.

Автор(ы) : профессор кафедры теоретической информатики, д. ф.-м. н. Кузьмин Е.В.

Приложение №1 к рабочей программе дисциплины
«Верификация программного обеспечения**»**

Фонд оценочных средств
для проведения текущей и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

1.1. Контрольные задания и иные материалы, используемые в процессе текущей аттестации

Варианты контрольной работы.

Текущий контроль успеваемости студентов организован в виде домашних заданий и четырех индивидуальных заданий, которые должен выполнить каждый студент. В предыдущем разделе описана технология индивидуального обучения студентов при помощи таких заданий. Требования к задачам индивидуальных заданий и примеры задач следующие:

I. Докажите формально, что следующий алгоритм предназначен для записи в переменную z произведения чисел a и b при $b \geq 0$ без использования операции умножения.

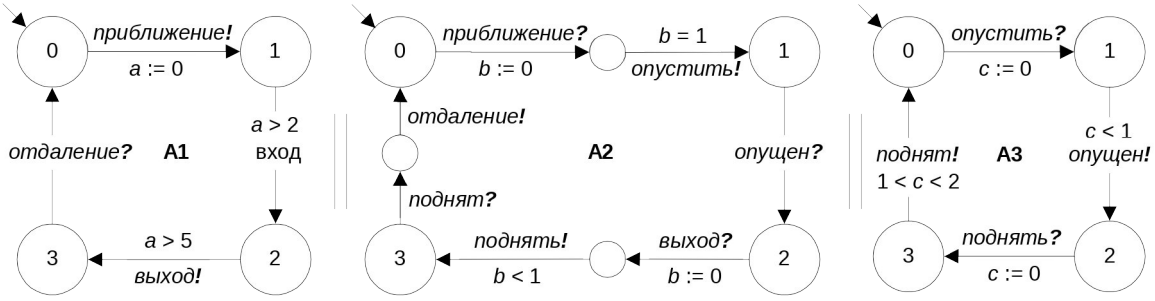
$x, y, z := a, b, 0;$
do $y > 0$ & even(y) -> $y, x := y/2, x+x$ []
odd(y) -> $y, z := y-1, z+x$
od

II. Используя определения синтаксиса и семантики формул темпоральной логики LTL (или CTL) произведите спецификацию с последующей верификацией с помощью средства верификации SPIN и SMV указанных ниже свойств для системы асинхронных параллельных процессов со взаимным исключением, представленной на рис. 1.1. в пособии [4]:

1. «Взаимное исключение». Процессы никогда не окажутся в своих критических участках одновременно. Другими словами, система ни при каких обстоятельствах не попадет в состояние, в котором процессы Пр1 и Пр2 будут находиться в своих локальных состояниях с номерами 7.
2. «Отсутствие взаимной блокировки». Не существует ситуации, при которой ни процесс Пр1, ни процесс Пр2 не могут перейти в другое локальное состояние.
3. «Отсутствие бесконечного откладывания процессов». Если один из процессов пожелает войти в свою критическую секцию, он обязательно в нее войдет. Другими словами, исключается возможность, при которой один из процессов бесконечно часто заходит в свой критический участок, а второй процесс вынужден постоянно откладывать свой вход в этот участок, бесконечно долго, таким образом, ожидая своей очереди.
4. «Справедливость». Если оба процесса одновременно вышли из своих некритических участков, т. е. процессы Пр1 и Пр2 находятся в локальных состояниях с номерами 1, то в свой критический участок первым обязательно войдет тот процесс, приоритет которого в данным момент выше (приоритет определяется исходя из значения переменной *trn*).
5. «Отсутствие чередования». Если первый процесс только что посетил свой критический участок и хочет вновь в него войти, а второй процесс не выражает

такого желания, то первому нет необходимости дожидаться, пока второй процесс проявит себя, войдет в критическую область, а затем покинет ее. Другими словами, посещения процессами своих критических участков не обязательно должны чередоваться. Если один из процессов навсегда остается в своем некритическом участке (например, закончил работу), то это не оказывает никакого влияния на возможность входа другого процесса в свой критический участок.

III. Проведите проверку свойств модели (реального времени) железнодорожного переезда с помощью программного средства верификации Uppaal, базирующемся на теории временных автоматов.



Модель автоматной системы в виде параллельной
композиции синхронизирующихся временных автоматов

Свойства:

1) шлагбаум не может быть опущен (переезд не может быть закрыт) более чем на 10 мин., т.е. состояние автомата A3 «0. шлагбаум поднят» достигается как максимум через 10 мин. после попадания им в состояние «2. Шлагбаум опущен».

2) когда поезд войдёт в переезд, шлагбаум уже должен быть опущен.

Кроме того, при сдаче задания могут быть заданы вопросы по теории. Материал по теме задания должен быть подробно и полно освещен и проиллюстрирован на примере решения этого задания.

1.2. Список вопросов и (или) заданий для проведения промежуточной аттестации

Вопросы на Экзамен.

Теория семантики и верификации программ

1. Корректность программ. Спецификация и верификация. Верификация и тестирование.

2. Спецификация программ. Предусловие. Постусловие. Примеры спецификаций программ. Представление начальных и конечных значений переменных. Наброски доказательств.

3. Семантика простого языка программирования. Преобразователь предикатов wp. Спецификация программ через преобразователь предикатов wp. Свойства wp.

4. Семантика простого языка программирования. Команды skip, abort и композиция команд. Команда присваивания. Кратное присваивание. Присваивание элементу массива.

5. Семантика простого языка программирования. Команда выбора. Примеры. Теорема о команде выбора. Доказательство корректности программ, не содержащих команд повторения. Примеры доказательств.

6. Семантика простого языка программирования. Команда повторения. Инвариант. Ограничивающая функция. Теорема о цикле, инварианте и ограничивающей функции. Доказательство программ, содержащих циклы. Список условий для проверки цикла. Примеры доказательств корректности цикла.

7. Построение программ. Стратегия построения команд выбора.

8. Построение программ. Построение циклов исходя из инвариантов и ограничений.

9. Построение инвариантов цикла. Теория воздушного шарика. Основная идея и стратегии построения инвариантов.

10. Построение инвариантов цикла методом устранения конъюнктивного члена. Примеры.

11. Построение инвариантов цикла методом замены константы переменной. Примеры.

12. Построение инвариантов цикла методом расширения области значений переменной. Примеры.

13. Построение инвариантов цикла методом комбинирования пред- и постусловий. Примеры.

Модели вычислительных процессов

1.Модели вычислительных процессов: сети Петри, взаимодействующие процессы и т.д.

2.Взаимодействие процессов. Асинхронные и синхронные процессы. Синхронизация параллельных процессов. Проблема критических участков. Анализ подходов к решению проблемы. Алгоритм Деккера. Программная реализация взаимоисключений.

3.Структура Крипке и метод автоматической верификации моделей.

Верификация моделей и теория автоматов

1. Автоматы над бесконечными словами.

2. Структура Крипке как автомат Бюхи.

3. Темпоральная логика линейного времени LTL.

4. Формула LTL как обобщенный автомат Бюхи. Замыкание формулы. Правила разметки последовательностей. Построение обобщенного автомата Бюхи по формуле LTL.

5. Редукция обобщенного автомата Бюхи для формулы логики LTL. Исключение избыточных переходов. Построение автомата исходя из необходимости. Определение эквивалентных состояний.

6. Пересечение языков структуры Крипке и автомата Бюхи.

7. Проверка пустоты автомата Бюхи.

8. Проверка модели «на лету»

Верификация моделей для логики CTL

1. Темпоральная логика CTL.

2. Верификация моделей для CTL.

3. Верификация моделей и неподвижные точки.

4. Символьная верификация моделей для CTL.

Двоичные диаграммы решений

1. Двоичные диаграммы решений ROBDD.

2. Построение и манипуляция ROBDD. Процедура Mk.

3. Построение и манипуляция ROBDD. Процедура Build.

4. Построение и манипуляция ROBDD. Процедура Apply.

5. Построение и манипуляция ROBDD. Процедура Restrict.

6. Построение и манипуляция ROBDD. Кванторы существования и всеобщности.

Теория временных автоматов

1. Детерминированные и недетерминированные временные автоматы Бюхи и Мюллера.

2. Разрешимые свойства временных автоматов.

3. Верификация систем реального времени с помощью временных автоматов.

2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкалы оценивания

2.1. Шкала оценивания сформированности компетенций и ее описание

Оценивание уровня сформированности компетенций в процессе освоения дисциплины осуществляется по следующей трехуровневой шкале:

Пороговый уровень - предполагает отражение тех ожидаемых результатов, которые определяют минимальный набор знаний и (или) умений и (или) навыков, полученных студентом в результате освоения дисциплины. Пороговый уровень является обязательным уровнем для студента к моменту завершения им освоения данной дисциплины.

Продвинутый уровень - предполагает способность студента использовать знания, умения, навыки и (или) опыт деятельности, полученные при освоении дисциплины, для решения профессиональных задач. Продвинутый уровень превосходит пороговый уровень по нескольким существенным признакам.

Высокий уровень - предполагает способность студента использовать потенциал интегрированных знаний, умений, навыков и (или) опыта деятельности, полученных при освоении дисциплины, для творческого решения профессиональных задач и самостоятельного поиска новых подходов в их решении путем комбинирования и использования известных способов решения применительно к конкретным условиям. Высокий уровень превосходит пороговый уровень по всем существенным признакам.

2.2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования

Форма контроля	Этапы формирования (№ темы (раздела)	Показатели оценивания	Шкала и критерии оценивания компетенций на различных этапах их формирования		
			Пороговый уровень	Продвинутый уровень	Высокий уровень
Индивидуальная работа. Зачет.	2 сем.: 1, 2 3 сем.: 1, 2	**Знать:** - современные методы спецификации, построения и анализа корректности программ. **Уметь:** - проводить тестирование программ ПЛК; - применять инструментальные средства верификации для анализа корректности программ. **Владеть:** навыками проверки корректности программ.	**Знать:** - современные методы спецификации, построения и анализа корректности программ. **Уметь:** - проводить тестирование программ ПЛК; - применять инструментальные средства верификации для анализа корректности программ. **Владеть:** навыками проверки корректности программ.	**Знать:** - современные методы спецификации, построения и анализа корректности программ. **Уметь:** - проводить тестирование программ ПЛК; - применять инструментальные средства верификации для анализа корректности программ. **Владеть:** навыками проверки корректности программ.	**Знать:** - современные методы спецификации, построения и анализа корректности программ. **Уметь:** - проводить тестирование программ ПЛК; - применять инструментальные средства верификации для анализа корректности программ. **Владеть:** навыками проверки корректности программ.

Индиви-дуальная работа. Зачет.	2 сем.: 1, 2 3 сем.: 1, 2	**Знать**: - способы моделирования программ; - способы спецификации и анализа свойств программ; - способы дедуктивного доказательства корректности программ; - стратегию спецификации и доказательства корректности программ, написанных на процедурном языке высокого уровня; - методы автоматической проверки корректности программной модели. **Уметь**: - проводить спецификацию программ на языке предикатов; - применять метод доказательства теорем для доказательства корректности программ, написанных на языках высокого уровня; - строить программные модели и проводить спецификацию и верификацию программных свойств на языке темпоральной логики; - строить модели систем реального времени с помощью формализма	**Знать**: - способы моделирования программ; - способы спецификации и анализа свойств программ; - способы дедуктивного доказательства корректности программ; - стратегию спецификации и доказательства корректности программ, написанных на процедурном языке высокого уровня; - методы автоматической проверки корректности программной модели. **Уметь**: - проводить спецификацию программ на языке предикатов; - применять метод доказательства теорем для доказательства корректности программ, написанных на языках высокого уровня; - строить программные модели и проводить спецификацию и верификацию программных свойств на языке темпоральной логики; - строить модели систем реального времени с	**Знать**: - способы моделирования программ; - способы спецификации и анализа свойств программ; - способы дедуктивного доказательства корректности программ; - стратегию спецификации и доказательства корректности программ, написанных на процедурном языке высокого уровня; - методы автоматической проверки корректности программной модели. **Уметь**: - проводить спецификацию программ на языке предикатов; - применять метод доказательства теорем для доказательства корректности программ, написанных на языках высокого уровня; - строить программные модели и проводить спецификацию и верификацию программных свойств на языке темпоральной логики; - строить модели систем реального времени с помощью формализма	**Знать**: - способы моделирования программ; - способы спецификации и анализа свойств программ; - способы дедуктивного доказательства корректности программ; - стратегию спецификации и доказательства корректности программ, написанных на процедурном языке высокого уровня; - методы автоматической проверки корректности программной модели. **Уметь**: - проводить спецификацию программ на языке предикатов; - применять метод доказательства теорем для доказательства корректности программ, написанных на языках высокого уровня; - строить программные модели и проводить спецификацию и верификацию программных свойств на языке темпоральной логики; - строить модели систем реального времени с помощью формализма временных автоматов и проводить спецификацию свойств таких систем на языках временных темпоральных логик.

		временных автоматов и проводить спецификацию свойств таких систем на языках временных темпоральных логик. **Владеть**: - формальными методами моделирования и спецификации программ; – - формальными методами анализа корректности программ (алгоритмов): дедуктивным анализом (метод доказательства теорем) и методом проверки модели (model checking).	помощью формализма временных автоматов и проводить спецификацию свойств таких систем на языках временных темпоральных логик. **Владеть**: - формальными методами моделирования и спецификации программ; 1. - формальными методами анализа корректности программ (алгоритмов): дедуктивным анализом (метод доказательства теорем) и методом проверки модели (model checking).	временных автоматов и проводить спецификацию свойств таких систем на языках временных темпоральных логик. **Владеть**: - формальными методами моделирования и спецификации программ; 1. - формальными методами анализа корректности программ (алгоритмов): дедуктивным анализом (метод доказательства теорем) и методом проверки модели (model checking).	**Владеть**: - формальными методами моделирования и спецификации программ; 1. - формальными методами анализа корректности программ (алгоритмов): дедуктивным анализом (метод доказательства теорем) и методом проверки модели (model checking).

3. Методические рекомендации преподавателю по процедуре оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Целью процедуры оценивания является определение степени овладения студентом ожидаемыми результатами обучения (знаниями, умениями, навыками и (или) опытом деятельности).

Процедура оценивания степени овладения студентом ожидаемыми результатами обучения осуществляется с помощью методических материалов, представленных в разделе «Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций»

3.1 Критерии оценивания степени овладения знаниями¸ умениями, навыками и (или) опытом деятельности, определяющие уровни сформированности компетенций

Пороговый уровень (общие характеристики):

- владение основным объемом знаний по программе дисциплины;
- знание основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы без существенных ошибок;
- владение инструментарием дисциплины, умение его использовать в решении стандартных (типовых) задач;
- способность самостоятельно применять типовые решения в рамках рабочей программы дисциплины;
- усвоение основной литературы, рекомендованной рабочей программой дисциплины;
- знание базовых теорий, концепций и направлений по изучаемой дисциплине;
- самостоятельная работа на практических и лабораторных занятиях, периодическое участие в групповых обсуждениях, достаточный уровень культуры исполнения заданий.

Продвинутый уровень (общие характеристики):

- достаточно полные и систематизированные знания в объёме программы дисциплины;
- использование основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать выводы;
- владение инструментарием дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку;
- самостоятельная работа на практических и лабораторных занятиях, участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

Высокий уровень (общие характеристики):

- систематизированные, глубокие и полные знания по всем разделам дисциплины;
- точное использование терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;

- безупречное владение инструментарием дисциплины, умение его использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно и творчески решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- полное и глубокое усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку;
- активная самостоятельная работа на практических и лабораторных занятиях, творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

3.2 Описание процедуры выставления оценки

В зависимости от уровня сформированности каждой компетенции по окончании освоения дисциплины студенту выставляется оценка. Для дисциплин, изучаемых в течение нескольких семестров, оценка может выставляться не только по окончании ее освоения, но и в промежуточных семестрах. Вид оценки («отлично», «хорошо», «удовлетворительно», «неудовлетворительно», «зачтено», «незачтено») определяется рабочей программой дисциплины в соответствии с учебным планом.

Оценка «отлично» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована на высоком уровне.

Оценка «хорошо» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на продвинутом уровне.

Оценка «удовлетворительно» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «неудовлетворительно» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Оценка «зачет» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «незачтено» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Приложение №2 к рабочей программе дисциплины
«Верификация программного обеспечения»

Методические указания для студентов по освоению дисциплины

Верификация представляет собой анализ корректности программных систем относительно спецификации, в которой задаются исследуемые программные свойства. Современные методы анализа корректности включают в себя тестирование, метод доказательства теорем (theorem proving) и метод проверки модели (model checking).

Тестирование применяется после окончательного написания программы. Но, как известно (Э. Дейкстра), если при тестировании ошибки найдены не были, это ещё не означает, что их нет вовсе. Отсюда следует необходимость рассмотрения формальных методов доказательства корректности программ. Поскольку тестирование – это способ проверки правильности программ, с которым аспиранты так или иначе (обычно в бессистемной форме) сталкивались при выполнении различных лабораторных работ, связанных с программированием, это направление предлагается аспирантам на самостоятельное изучение. Аудитории ставится задача подготовить ряд докладов и рефератов, отражающих современное положение дел в науке по тестированию. Основная цель – обратить внимание аспирантов на то, что тестирование представляет собой серьёзную проблематику, по которой в настоящее время проводится активная научно-исследовательская работа, а также определить круг проблем и ограничений, существующих в тестировании.

Метод доказательства теорем является очень трудоёмким методом с сильной привязкой к семантике языка программирования. Поэтому в рамках курса (на лекциях) вводится «простой» язык программирования, включающий привычные конструкции условного перехода и цикла, но имеющий ограничения на типы данных: допускаются переменные только целочисленного типа (или булевого типа). После определения семантики операторов введённого языка программирования, разъяснения основных принципов доказательства программ, написанных на этом языке, и рассмотрения нескольких примеров каждому аспиранту предлагается (по вариантам) доказать корректность небольшой программы. При этом допускаются «мягкая» и «жёсткая» формы задания. В первом случае формальная спецификация даётся вместе с текстом программы, во втором случае аспиранту необходимо самостоятельно произвести спецификацию формальным образом по сопровождению на естественном (русском) языке, т.е. построить ряд предикатов, отражающих начальные условия и ожидаемый результат выполнения программы. Важно отметить, что доказательство корректности программы проходит в терминах предикатов (что позволяет на практике применять полученные ранее знания по теории исчисления предикатов), занимает от трёх до пяти страниц и требует от аспиранта предельной точности рассуждений. Более того, особенностью метода доказательства теорем является то, что в ходе рассуждений можно установить корректность программы относительно спецификации, однако если этого не удаётся сделать, то это, вообще говоря, не означает наличия ошибок в программе, а требует более тонкого разбора. Перефразируя известное изречение Э. Дейкстры (о тестировании) можно сказать, что метод доказательства теорем способен доказать отсутствие ошибок в программе (относительно спецификации), но не всегда доказывает их присутствие. С позиции технологии программирования метод доказательства теорем следует рассматривать как дополнение тестирования, позволяющее получать более совершенную стратегию отладки программ.

Метод проверки модели (model checking) – это метод автоматической верификации программных систем. При этом подходе для программы строится модель с конечным числом состояний, а свойства модели (спецификация) выражаются на языке темпоральной логики. Конечная модель и формулы темпоральной логики подаются на вход программе-верификатору, и далее проверка истинности формул для модели осуществляется автоматически (в основном с применением переборных алгоритмов). Поэтому в рамках

обсуждаемой дисциплины по методу проверки модели важно рассмотреть следующие разделы: построение конечной модели программы, спецификация свойств на языках темпоральных логик и программы-верификаторы с описанием методов и алгоритмов, применяемых в этих программных средствах.

Необходимо заметить, что построение конечной адекватной (исходной программе) модели является весьма серьёзной задачей, поскольку модель может не учитывать ряд программных свойств или порождать несуществующие. Несмотря на то, что проверяются свойства модели, объектом исследования всё же по-прежнему остаётся программа.

Для спецификации свойств программ используются наиболее распространённые и широко применяемые темпоральные логики: логика линейного времени LTL (linear-time temporal logic) и логика ветвящегося времени CTL (branching-time temporal logic или computation tree logic). Логики имеют довольно простые синтаксис и семантику и не встречают трудностей в понимании у аудитории. В то же время прикладной аспект, касающихся применения этих логик, аспирантам не кажется столь прозрачным и полезным без достаточно простых, но семантически нагруженных примеров. Важно отметить, что в имеющейся литературе (статьи, учебники и монографии) таких примеров либо нет, либо они имеют вырожденный характер (применение логик в этом случае становится бессмысленным). В публикациях основной упор сделан на алгоритмы и методики. В работах, касающихся практического применения метода проверки модели, объектами рассмотрения являются или (довольно нетривиальные) параллельные алгоритмы, или протоколы передачи данных, перегруженные деталями реализации и требующие специальной подготовки и дополнительных знаний.

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

Для самостоятельной работы особенно рекомендуется использовать учебную литературу.

Также для подбора учебной литературы рекомендуется использовать широкий спектр интернет-ресурсов:

1. Электронно-библиотечная система «Университетская библиотека online» (www.biblioclub.ru) - электронная библиотека, обеспечивающая доступ к наиболее востребованным материалам-первоисточникам, учебной, научной и художественной литературе ведущих издательств (*регистрация в электронной библиотеке – только в сети университета. После регистрации работа с системой возможна с любой точки доступа в Internet.).

2. Информационная система "Единое окно доступа к образовательным ресурсам" (http://window.edu.ru/library).

Целью создания информационной системы "Единое окно доступа к образовательным ресурсам" (ИС "Единое окно ") является обеспечение свободного доступа к интегральному каталогу образовательных интернет-ресурсов и к электронной библиотеке учебно-методических материалов для общего и профессионального образования.

Информационная система "Единое окно доступа к образовательным ресурсам" создана по заказу Федерального агентства по образованию в 2005-2008 гг. Головной разработчик проекта - Федеральное государственное автономное учреждение Государственный научно-исследовательский институт информационных технологий и телекоммуникаций (ФГАУ ГНИИ ИТТ "Информика") www.informika.ru.

ИС "Единое окно" объединяет в единое информационное пространство электронные ресурсы свободного доступа для всех уровней образования в России. Разделы этой системы:

- Электронная библиотека – является крупнейшим в российском сегменте Интернета хранилищем полнотекстовых версий учебных, учебно-методических и научных материалов с открытым доступом. Библиотека содержит более 30 000 материалов, источниками которых являются более трехсот российских вузов и других образовательных и научных учреждений. Основу наполнения библиотеки составляют электронные версии учебно-

методических материалов, подготовленные в вузах, прошедшие рецензирование и рекомендованные к использованию советами факультетов, учебно-методическими комиссиями и другими вузовскими структурами, осуществляющими контроль учебно-методической деятельности.

- Интегральный каталог образовательных интернет-ресурсов содержит представленные в стандартизованной форме метаданные внешних ресурсов, а также содержит описания полнотекстовых публикаций электронной библиотеки. Общий объем каталога превышает 56 000 метаописаний (из них около 25 000 - внешние ресурсы). Расширенный поиск в "Каталоге" осуществляется по названию, автору, аннотации, ключевым словам с возможной фильтрацией по тематике, предмету, типу материала, уровню образования и аудитории.

- Избранное. В разделе представлены подборки наиболее содержательных и полезных, по мнению редакции, интернет-ресурсов для общего и профессионального образования.

- Библиотеки вузов. Раздел содержит подборки сайтов вузовских библиотек, электронных каталогов библиотек вузов и полнотекстовых электронных библиотек вузов.

Для самостоятельного подбора литературы в библиотеке ЯрГУ рекомендуется использовать:

1. Личный кабинет (http://lib.uniyar.ac.ru/opac/bk_login.php) дает возможность получения on-line доступа к списку выданной в автоматизированном режиме литературы, просмотра и копирования электронных версий изданий сотрудников университета (учеб. и метод. пособия, тексты лекций и т.д.) Для работы в «Личном кабинете» необходимо зайти на сайт Научной библиотеки ЯрГУ с любой точки, имеющей доступ в Internet, в пункт меню «Электронный каталог»; пройти процедуру авторизации, выбрав вкладку «Авторизация», и заполнить представленные поля информации.

2. Электронная библиотека учебных материалов ЯрГУ

(http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php) содержит более 2500 полных текстов учебных и учебно-методических материалов по основным изучаемым дисциплинам, изданных в университете. Доступ в сети университета, либо по логину/паролю.

3. Электронная картотека «Книгообеспеченность»

(http://www.lib.uniyar.ac.ru/opac/bk_bookreq_find.php) раскрывает учебный фонд научной библиотеки ЯрГУ, предоставляет оперативную информацию о состоянии книгообеспеченности дисциплин основной и дополнительной литературой, а также цикла дисциплин и специальностей. Электронная картотека «Книгообеспеченность» доступна в сети университета и через Личный кабинет.