МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ – МСХА имени К.А. ТИМИРЯЗЕВА»
(ФГБОУ ВО РГАУ - МСХА имени К.А. Тимирязева)

Институт экономики и управления АПК
Кафедра Прикладной информатики

УТВЕРЖДАЮ:
Директор института
экономики и управления АПК
_____ Л.И. Хоружий
“__28__”_____08_____2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.В.08 «Математические основы верификации программного обеспечения»

для подготовки магистров

ФГОС ВО

Направление: 09.04.03 «Прикладная информатика»
Направленность: «Архитектура систем искусственного интеллекта»

Курс 2
Семестр 4

Форма обучения: очная

Год начала подготовки: 2025

Москва, 2025

Разработчик: _Греченева А.В._____
(ФИО, ученая степень, ученое звание)

«28» августа 2025г.

Рецензент: ___Ашмарина Т.И._____
(ФИО, ученая степень, ученое звание)

(подпись)

«28» августа 2025г.

Программа составлена в соответствии с требованиями ФГОС ВО, профессионального стандарта и учебного плана по направлению подготовки 09.04.03 «Прикладная информатика»

Программа обсуждена на заседании кафедры прикладной информатики протокол № 1 от «28»августа 2025г.

И.о. зав. кафедрой прикладной информатики
д.э.н., профессор Худякова Е.В._____
(подпись)

«28» августа 202_г.

Согласовано:

Председатель учебно-методической
комиссии института экономики и управления АПК
к.э.н., доцент Гупалова Т.Н. _____
(подпись)

«28» августа 2025г.

И.о. заведующий выпускающей кафедрой
прикладной информатики
д.э.н., профессор Худякова Е.В._____
(подпись)

«28» августа 2025г.

Заведующий отделом комплектования ЦНБ _____
(подпись)

СОДЕРЖАНИЕ

АННОТАЦИЯ

рабочей программы учебной дисциплины Б1.В.08 «Математические основы верификации программного обеспечения» для подготовки магистра по направлению 09.04.03 «Прикладная информатика», направленности «Архитектура систем искусственного интеллекта»

Цель освоения дисциплины: формирование теоретических и методических основ формальной верификации и доказательства корректности программных систем на базе математического аппарата логики, теории множеств, теории отношений, формальных спецификаций и моделей вычислений, а также развитие навыков применения формальных методов для анализа требований, построения спецификаций, выявления ошибок проектирования и обоснования надежности программного обеспечения на этапах жизненного цикла разработки.

Задачи дисциплины:

1. Формирование понятийного аппарата формальных методов верификации: корректность, спецификация, инварианты, свойства безопасности и живучести, выполнимость и доказуемость.
2. Освоение математической базы формальной верификации: логика высказываний и предикатов, теории множеств и отношений, элементы теории доказательств.
3. Изучение формальных моделей программ и систем: переходные системы, автоматы, системы с состояниями, модели конкурентных/распределённых вычислений.
4. Овладение методами формальной спецификации поведения программных систем (пред- и постусловия, контракты, инварианты, абстракции состояния).
5. Освоение базовых подходов к доказательству корректности программ (Hoare logic, индукция, доказательство по инварианту, рассуждения о завершимости).
6. Изучение принципов и алгоритмов проверки моделей (model checking), формулирования свойств во временных логиках и интерпретации результатов проверки.
7. Формирование навыков верификации требований: непротиворечивость, полнота, трассируемость, формализация и проверка свойств на моделях.
8. Развитие компетенций применения инструментальных средств формальной верификации (теоремные доказчики/SMT/модельные проверяющие) в учебных кейсах.
9. Формирование навыков подготовки доказательной документации по результатам верификации (спецификации, доказательства, отчёты о проверке свойств) для использования в проектной и исследовательской деятельности.

Место дисциплины в учебном плане:

Дисциплина Б1.В.08 «Математические основы верификации программного обеспечения» относится к части, формируемой участниками образовательных отношений учебного плана основной профессиональной образовательной программы подготовки магистров по направлению 09.04.03 «Прикладная информатика» (направленности «Архитектура систем искусственного интеллекта» и/или «ИТ-инновации и цифровые решения для бизнеса»).

Освоение дисциплины опирается на результаты изучения следующих курсов:Математические методы и модели поддержки принятия решений (логика, формальные модели, методы анализа);Современные технологии разработки программного обеспечения (жизненный цикл ПО, требования, архитектура, тестирование, CI/CD);дисциплины, формирующие фундамент по ИС и ИИ (при наличии в учебном плане): архитектура ИС, базы данных, алгоритмы и структуры данных, парадигмы программирования, основы кибербезопасности.

Дисциплина является методологической и инструментальной основой для последующего изучения и/или проектной работы в рамках дисциплин по архитектуре и качеству ИС/ИИ-систем (архитектурный анализ, атрибуты качества, надежность); дисциплин по безопасности и защите информации, включая анализ уязвимостей и безопасность разработки (Secure SDLC);дисциплин по эксплуатации/развертыванию ИИ-систем, DevOps/MLOps (контроль корректности, мониторинг свойств, compliance); подготовки и выполнения курсовых проектов/ВКР, где требуется доказательная база корректности, формальная спецификация и/или применение инструментов model checking, SMT/SAT, theorem proving, статического анализа.

Требования к результатам освоения дисциплины: в результате освоения дисциплины формируются следующие компетенции (индикаторы) их достижения: ПКос-1.1, ПКос-1.2, ПКос-1.3, ПКос-2.1, ПКос-2.2, ПКос-2.3, ПКос-4.1, ПКос-4.2, ПКос-4.3

Краткое содержание дисциплины: Математическая логика и формализация требований. Исчисление высказываний и предикатов. Булева алгебра и преобразование логических формул. Формальные спецификации требований и моделей (инварианты, предусловия, постусловия). Логика Хоара и доказательство корректности программ. Инварианты циклов и частичная/полная корректность. Теория автоматов и системы переходов. Временные логики LTL/CTL и спецификация свойств поведения. Model checking и проверка достижимости/безопасности/живучести. Абстракция, уточнение и compositional verification для больших систем. SAT/SMT-подходы и решатели (напр., Z3) для задач верификации. Theorem proving и интерактивные доказательства (напр., Coq/Isabelle/Lean). Символьное выполнение и поиск дефектов. Статический анализ кода и анализ потоков данных/управления. Формальные методы анализа надежности и рисков (FMEA/FTA/HAZOP, Марковские модели). Верификация требований безопасности и соответствия (compliance). Интеграция верификации в процессы разработки и CI/CD, документирование V&V-артефактов.

Общая трудоемкость дисциплины: 144/4 (часы/зач. ед.).

Промежуточный контроль: экзамен.

1. Цель освоения дисциплины

Сформировать теоретические и методических основ формальной верификации и доказательства корректности программных систем на базе математического аппарата логики, теории множеств, теории отношений, формальных спецификаций и моделей вычислений, а также развитие навыков применения формальных методов для анализа требований, построения спецификаций, выявления ошибок проектирования и обоснования надежности программного обеспечения на этапах жизненного цикла разработки.

2. Место дисциплины в учебном процессе

Дисциплина Б1.В.08 «Математические основы верификации программного обеспечения» относится к части, формируемой участниками образовательных отношений учебного плана основной профессиональной образовательной программы подготовки магистров по направлению 09.04.03 «Прикладная информатика» (направленности «Архитектура систем искусственного интеллекта» и/или «ИТ-инновации и цифровые решения для бизнеса»).

Дисциплина реализуется в соответствии с требованиями ФГОС ВО, ОПОП, локальных нормативных актов образовательной организации, а также с учетом актуальных требований к обеспечению качества, надежности и безопасности программного обеспечения в критичных и высоконагруженных информационных системах.

Место дисциплины в структуре и логике освоения ОПОП определяется ее ориентированностью на формирование у магистрантов математического и инструментального базиса формальных методов верификации, применяемых в жизненном цикле разработки ПО (спецификация → проектирование → реализация → тестирование → V&V → сопровождение), а также на развитие навыков выбора и обоснования верификационной стратегии с учетом ограничений проекта.

Освоение дисциплины опирается на результаты изучения следующих курсов Математические методы и модели поддержки принятия решений (логика, формальные модели, методы анализа);Современные технологии разработки программного обеспечения (жизненный цикл ПО, требования, архитектура, тестирование, CI/CD);дисциплины, формирующие фундамент по ИС и ИИ (при наличии в учебном плане): архитектура ИС, базы данных, алгоритмы и структуры данных, парадигмы программирования, основы кибербезопасности.

Дисциплина является методологической и инструментальной основой для последующего изучения и/или проектной работы в рамках дисциплин по архитектуре и качеству ИС/ИИ-систем (архитектурный анализ, атрибуты качества, надежность);дисциплин по безопасности и защите информации, включая анализ уязвимостей и безопасность разработки (Secure SDLC); дисциплин по эксплуатации/развертыванию ИИ-систем, DevOps/MLOps (контроль корректности, мониторинг свойств, compliance);подготовки и выполнения курсовых проектов/ВКР, где требуется доказательная база корректности, формальная спецификация и/или применение инструментов model checking, SMT/SAT, theorem proving, статического анализа.

Дисциплина обеспечивает формирование готовности магистрантов: применять логические и автоматные модели для описания поведения программ и систем; использовать инструменты верификации и статического анализа (model checking, SMT-решатели, theorem provers) для доказательства свойств корректности и безопасности;обосновывать выбор стратегии V&V (включая risk-based verification) и оформлять результаты верификации в виде отчетов и артефактов качества.

Рабочая программа дисциплины для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается (при необходимости) с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья обучающихся и предусматривает адаптацию форм представления учебного материала и процедур контроля.

3. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Образовательные результаты освоения дисциплины обучающимся, представлены в таблице 1.

4. Структура и содержание дисциплины

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам

Общая трудоёмкость дисциплины составляет 4 зач. единиц (144 часов), их распределение по видам работ представлено в табл. 2.

Таблица 1

Требования к результатам освоения учебной дисциплины

№ п/п	Индекс компетенции	Содержание компетенции (или её части)	Индикатор компетенций	В результате выполнения курсового проекта по учебной дисциплине обучающиеся должны:		
				знать	уметь	владеть
	ПКос-1	Способность применять современные методы и инструментальные средства прикладной информатики для автоматизации и информатизации решения прикладных задач различных классов и создания ИС	ПКос-1.1 Знает методы прикладной информатики	-основы математической логики: исчисление высказываний, предикатов, булева алгебра; - формальные методы верификации: model checking, theorem proving, abstract interpretation; -темпоральные логики: LTL (Linear Temporal Logic), CTL (Computation Tree Logic), CTL; -методы статического анализа кода: data flow analysis, control flow analysis, taint analysis; - основы теории автоматов и формальных языков для моделирования систем; - принципы формальной спецификации требований: Z-notation, VDM, Alloy; - математические основы корректности программ: предусловия, постусловия, инварианты циклов.	-	-
			ПКос-1.2 Умеет применять современные методы и инструментальные	-	- применять инструменты model checking: SPIN, NuSMV, UPPAAL для верификации систем;	-

			средства прикладной информатики для автоматизации и информатизации решения прикладных задач различных классов и создания ИС		- использовать theorem provers: Coq, Isabelle/HOL, Lean для доказательства корректности; - проводить статический анализ с помощью инструментов: SonarQube, PMD, Pylint, Coverity; - формализовать требования к ПО с использованием темпоральных логик; - строить модели конечных автоматов и систем переходов для анализа поведения; - применять методы символьного выполнения: KLEE, angr для поиска ошибок; - использовать SAT/SMT-солверы: Z3, CVC4 для решения задач верификации.	

			ПКос-1.3 Владеет инструментальными средствами прикладной информатики для автоматизации и информатизации решения прикладных задач различных классов и создания ИС	-	-	- методами формальной верификации критических систем (авиация, медицина, АСУ ТП); - инструментами автоматического поиска ошибок: fuzzing (AFL, LibFuzzer), symbolic execution; - техниками доказательства корректности алгоритмов с использованием логики Хоара; - методами верификации concurrent и distributed систем; - навыками интеграции верификационных инструментов в CI/CD pipeline; - практиками формальной спецификации API и протоколов; - инструментами runtime verification и мониторинга соответствия спецификациям.
	ПКос-2	Способность проектировать архитектуру ИС предприятий и организаций в прикладной области	ПКос-2.1 Знает способы проектирования архитектуры ИС	- архитектурные паттерны для верифицируемых систем: layered architecture, microservices; - принципы design by contract: pre/post-conditions, invariants; - методы формального моделирования архитектуры: AADL, SysML; - подходы к проектированию fault-tolerant и safety-critical систем;	-	-

				- стандарты безопасности ПО: DO-178C (авиация), IEC 61508 (промышленность), ISO 26262 (автомобили); - методы архитектурного анализа: ATAM (Architecture Tradeoff Analysis Method); - принципы defensive programming и fail-safe design.		
			ПКос-2.2 Умеет проектировать архитектуру ИС предприятий и организаций АПК	-	- проектировать архитектуру ИС с учетом требований верифицируемости; - применять формальные методы на этапе проектирования: refinement, abstraction; - разрабатывать спецификации компонентов с использованием формальных нотаций; - использовать CASE-средства для моделирования и верификации архитектуры; - проводить архитектурный risk assessment с использованием формальных методов; - интегрировать верификационные инструменты в процесс проектирования;	-

					- применять методы compositional verification для больших систем.	
			ПКос-2.3 Владеет методикой проектирования архитектуры ИС предприятий			- методологией формального проектирования ИС: B-method, Event-B, TLA+; - инструментами архитектурного моделирования: AADL Toolset, SysML/UML с OCL; - техниками верификации архитектурных решений: architecture compliance checking; - практиками проектирования safety-critical систем для АПК (SCADA, IoT, автоматизация); - методами формального анализа безопасности архитектуры: threat modeling, formal security analysis; - навыками документирования формальных спецификаций архитектуры; - инструментами для contract-based design: ACSL (ANSI/ISO C Specification Language), JML (Java Modeling Language).
	ПКос-4	Способность принимать эффективные проектные решения в условиях	ПКос-4.1 Знает методы принятия управленческих решений	- методы формального анализа рисков в ПО: FMEA, FTA (Fault Tree Analysis), HAZOP; - вероятностные методы верификации: probabilistic	-	-

		неопределенности и риска		model checking, Markov chains; - подходы к количественной оценке надежности ПО; - методы анализа trade-off между cost, time, quality в верификации; - техники формального анализа безопасности: security verification, vulnerability analysis; - стандарты оценки качества и надежности ПО: ISO/IEC 25010, CMMI; - методы принятия решений в условиях неполной формальной спецификации.		
			ПКос-4.2 Умеет принимать эффективные проектные решения в условиях неопределенности и риска	-	- применять probabilistic model checking: PRISM, STORM для анализа надежности; - проводить формальный анализ рисков безопасности ПО; - выбирать оптимальные методы верификации с учетом ограничений проекта; - применять cost-benefit analysis для выбора верификационной стратегии; - использовать инструменты для	-

					автоматической генерации тестов из формальных спецификаций; - проводить формальный анализ compliance с регуляторными требованиями; - применять методы bounded model checking для больших систем.	
			ПКос-4.3 Владеет инструментами обоснования эффективных проектных решений в условиях неопределенности и риска			- методами обоснования выбора верификационной стратегии для проекта; - техниками risk-based testing и verification; - инструментами для формального анализа надежности: PRISM, STORM, MRMC; - практиками формального обоснования решений перед stakeholders; - навыками документирования verification & validation (V&V) процесса; - методами формального анализа trade-off: performance vs. safety, cost vs. completeness; - инструментами для certification evidence generation в регулируемых доменах.

Таблица 2

Распределение трудоёмкости дисциплины по видам работ по семестрам

Вид учебной работы	**Трудоёмкость**	
	час. **всего/***	**в т.ч. по семестрам**
		№4
Общая трудоёмкость дисциплины по учебному плану	**144/4**	**144/4**
1. **Контактная работа:**		
Аудиторная работа:	**52,4/4**	**52,4/4**
в том числе:		
лекции (Л)	20	20
практические занятия (ПЗ)	30/4	30/4
курсовой проект (консультация, защита)		
консультация перед экзаменом	2	2
контактная работа на промежуточном контроле (КРА)	0,4	0,4
2. **Самостоятельная работа** (СРС)	**67**	**67**
в том числе:		
курсовой проект (подготовка)		
самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к практическим занятиям, и т.д.)	58	58
подготовка к экзамену	9	9
Вид промежуточного контроля:	**Экзамен**	

* в том числе практическая подготовка

4.2 Содержание дисциплины

Таблица 3

Тематический план учебной дисциплины

Наименование разделов и тем дисциплин (укрупнённо)	**Всего**	**Аудиторная работа**			**Внеаудиторная работа СР**
		Л	**ПЗ/С** **всего/***	**ПКР**	
Тема 1. Логические основы формальной спецификации требований к ПО и ИИ-компонентам	20,00	4	6	-	10,00
Тема 2. Доказательство корректности программ и контрактная верификация ML/LLM-пайплайнов	20,00	4	6/2	-	10,00

Наименование разделов и тем дисциплин (укрупнённо)	**Всего**	**Аудиторная работа**			**Внеаудиторная работа СР**
		Л	**ПЗ/С** **всего/***	**ПКР**	
Тема 3. Моделирование поведения программных и ИИ-систем: переходные системы и проверка свойств	20,00	4	6/2	-	10,00
Тема 4. Временные логики, SAT/SMT и формальная верификация свойств ИИ-моделей	20,00	4	6	-	10
Тема 5. Инструменты и инженерная практика V&V для ПО и ИИ: доказательства, анализ, CI/CD	37,00	4	6	-	27
Сдача экзамена	2	-	-	-	-
Подготовка к экзамену	24,6	-	-	-	-
Контактная работа на промежуточном контроле (КРА)	0,4	-	-	0,4	-
Всего за 1 семестр	144/4	20	30/4	0,4	67
Итого по дисциплине	**144/4**	**20**	**30/4**	**0,4**	67

* в том числе практическая подготовка

Тема 1. Логические основы спецификации и корректности ПО

Формализация требований к ML/AI-компонентам: корректность входных данных, допустимые диапазоны, инварианты признаков, требования к качеству/устойчивости.Спецификация нефункциональных свойств ИИ: устойчивость (robustness), справедливость (fairness), интерпретируемость, приватность, безопасность.Контракты для data/ML pipeline: data contracts, schema constraints, drift constraints.

Тема 2. Доказательство корректности программ и логика Хоара

Контракты и инварианты для preprocessing/feature engineering, контроля утечек данных (data leakage), воспроизводимости.Design-by-contract для сервисов inference: требования к латентности, отказоустойчивости, обработке OOD (out-of-distribution).Корректность интеграции ИИ-модуля в бизнес-логику (guardrails, проверки на входе/выходе).

Тема 3. Моделирование систем и проверка свойств

Моделирование ИИ-систем как компонентных систем: data → model → decision → actuation (особенно для АПК/IoT/SCADA).Переходные системы для политик/правил принятия решений, гибридных систем (алгоритмы + ML).
Проверка safety-свойств для автономных/полуавтономных контуров (например, «не выдавать команды вне допустимого диапазона»).

Тема 4. Временные логики и автоматизированная верификация

LTL/CTL-спецификации для поведения ИИ-сервиса во времени: деградация качества, реакция на дрейф, требования к мониторингу.Верификация policy (RL) и decision logic: свойства достижимости/безопасности.SMT/SAT-подходы для верификации нейросетей (классы задач: локальная робастность, ограничение выходов, монотонность, отсутствие «опасных» решений в области входов).

Тема 5. Инструменты и инженерная практика V&V

Формальная/полуформальная верификация нейросетевых моделей (SMT/abstract interpretation/linear relaxation).Runtime verification для ИИ: мониторы, алерты, policy enforcement, human-in-the-loop.Threat modeling и security verification для ML: data poisoning, model stealing, prompt injection (для LLM), adversarial examples.ИИ как инструмент верификации: автоматизация поиска дефектов, генерация тестов, анализ требований, ассистенты для доказательств/инвариантов (с обязательной валидацией результатов).

4.3 Лекции/ практические занятия

Таблица 4

Содержание лекций/ практических занятий и контрольные мероприятия

№ п/п	Название раздела, темы	№ и название лекций/практических занятий	Формируемые компетенции (индикаторы)	Вид контрольного мероприятия	Кол-во часов / из них практическая подготовка
1	Тема 1. Логические основы формальной спецификации требований к ПО и ИИ-компонентам	**Лекция №1.** Формализация требований к ПО и ИИ-системам: логика высказываний/предикатов, модели, интерпретации	ПКос-1.1	—	2
		Лекция №2. Непротиворечивость требований и ограничений данных: булева алгебра, выполнимость, введение в SAT/SMT	ПКос-1.1	—	2
		Практическая работа №1. Формализация функциональных/нефункциональных требований к ИИ-сервису в виде логических ограничений (CNF/DNF)	ПКос-1.2	Защита работы	2
		Практическая работа №2. Data contracts для ML/NLP-пайплайна: схемы, домены,	ПКос-1.2	Защита работы	2

		проверки качества и валидности данных			
		Практическая работа №3. Поиск контрпримеров к требованиям: построение модели-свидетеля и разбор конфликтов спецификации	ПКос-1.2	Защита работы	2
2	Тема 2. Доказательство корректности программ и контрактная верификация ML/LLM-пайплайнов	**Лекция №3.** Логика Хоара и design by contract: предусловия/постусловия, корректность модулей ИИ-пайплайна	ПКос-1.1	—	2
		Лекция №4. Инварианты циклов и завершимость: корректность preprocessing/обучающих процедур, типовые схемы доказательств	ПКос-1.1	—	2
		Практическая работа №4. Доказательство корректности процедуры предобработки данных/текста (NLP): pre/post условия и частичная корректность	ПКос-1.2	Защита работы	2
		Практическая работа №5. Инварианты и завершимость для циклов обработки данных/обучения (итеративные алгоритмы, mini-batch)	ПКос-1.2	Защита работы	2/2
		Практическая работа №6. Контракты inference-API: граничные случаи, обработка ошибок, OOD-входы, отказоустойчивое поведение	ПКос-1.2, ПКос-1.3	Защита работы	2
3	Тема 3. Моделирование поведения программных и ИИ-систем: переходные системы и проверка свойств	**Лекция №5.** Переходные системы и автоматы для ИИ-сервисов: состояния, события, недетерминизм, композиция компонентов	ПКос-1.1, ПКос-2.1	—	2
		Лекция №6. Model checking: safety/liveness свойства, контрпримеры; применение к жизненному циклу ML-систем	ПКос-1.1, ПКос-2.1	—	2
		Практическая работа №7. Модель жизненного цикла ML-системы (train–validate–deploy–monitor–rollback) как переходная система	ПКос-2.2	Защита работы	2/2
		Практическая работа №8. Проверка safety-свойств ML-	ПКос-1.2, ПКос-2.2	Защита работы	2

		контура: контроль версий данных/моделей, запрет невалидированных релизов			
		Практическая работа №9. Композиционное моделирование “ИИ-сервис + бизнес-правила”: выявление конфликтов требований и сценариев	ПКос-2.2, ПКос-2.3	Защита работы	2
4	Тема 4. Временные логики, SAT/SMT и формальная верификация свойств ИИ-моделей	**Лекция №7.** Временные логики LTL/CTL для требований к ИИ-системам: мониторинг, дрейф, триггеры, откаты	ПКос-1.1	—	2
		Лекция №8. SAT/SMT и bounded model checking; типовые свойства ML-моделей (ограничения, монотонность, робастность)	ПКос-1.1	—	2
		Практическая работа №10. Спецификация эксплуатационных требований ML/LLM-сервиса в LTL/CTL (SLO, дрейф, условия остановки/отката)	ПКос-1.2	Защита работы	2
		Практическая работа №11. Bounded model checking сценариев релиза модели: approve–deploy–monitor–rollback, анализ контрпримеров	ПКос-1.2	Защита работы	2
		Практическая работа №12. SMT-верификация свойства упрощённой модели (ограничение выходов/монотонность/локальная робастность)	ПКос-1.2, ПКос-1.3	Защита работы	2
5	Тема 5. Инструменты и инженерная практика V&V для ПО и ИИ: доказательства, анализ, CI/CD	**Лекция №9.** Доказательные артефакты и теоремные доказчики: спецификация, леммы, доказательства корректности компонентов	ПКос-1.1, ПКос-1.3	—	2
		Лекция №10. Интеграция верификации в SDLC/MLops: статический анализ, символьное выполнение, runtime verification, evidence, риск-ориентированный V&V	ПКос-4.1	—	2
		Практическая работа №13. Символьное выполнение и/или property-based подход для проверки свойств	ПКос-1.2, ПКос-1.3	Защита работы	2

		компонентов NLP/ML-пайплайна			
		Практическая работа №14. Статический анализ и формальные требования безопасности для ML/LLM-сервиса (валидаторы, аудит, контроль утечек)	ПКос-1.2, ПКос-4.2	Защита работы	2
		Практическая работа №15. Риск-ориентированный план V&V для ИИ-системы: трассируемость требований, выбор методов, критерии приемки и evidence	ПКос-4.2, ПКос-4.3	Защита работы	2

Таблица 5

Перечень вопросов для самостоятельного изучения дисциплины

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
1	**Тема 1. Логические основы формальной спецификации требований к ПО и ИИ-компонентам**	Семантика и выразительная мощность логики высказываний и исчисления предикатов (soundness/completeness, выполнимость, выводимость). Разрешимость и вычислительная сложность задач SAT/SMT (NP-completeness, PSPACE, границы применимости). Теории SMT и их роль в инженерной верификации: линейная арифметика, массивы, бит-векторы, uninterpreted functions. Интерполяция (Craig interpolation) и применение в CEGAR/доказательствах. Паттерны формализации требований и ограничений для ИИ-пайплайнов: data/model contracts, инварианты качества данных, ограничения на домены и допустимые трансформации. Формализация требований к этичности и надежности ИИ (fairness constraints, robustness requirements) как логических ограничений. Компетенции: **ПКос-1.1, ПКос-1.2**.
2	**Тема 2. Доказательство корректности программ и контрактная верификация ML/LLM-пайплайнов**	Полная корректность vs частичная корректность; доказательства завершимости (ranking functions, well-founded orders). WP/SP-исчисления (weakest precondition/strongest postcondition) и автоматизация доказательств. Логика разделения (Separation Logic) для памяти, алиасинга и структур данных. Дедуктивная верификация программ с побочными эффектами и исключениями; спецификация ошибок и fail-safe поведения. Верификация численных вычислений и устойчивости (floating-point pitfalls, переполнения, NaN/Inf) в коде обучения/инференса. Контрактная спецификация интерфейсов ML/LLM-сервисов (API-level contracts): ограничения входов, деградация качества, обработка OOD-входов. Связь спецификаций с

		тестированием: property-based testing как “исполняемые свойства”. Компетенции: **ПКос-1.2, ПКос-1.3**.
3	**Тема 3. Моделирование поведения программных и ИИ-систем: переходные системы и проверка свойств**	Автоматы (в т.ч. Бюхи) и их связь с LTL; построение автоматов по спецификации. Символический model checking (BDD/SAT-based), частичный порядок (partial order reduction) и борьба со взрывом состояний. Композиционная верификация (assume–guarantee reasoning), интерфейсные автоматы, контрактные модели компонентов. Абстракция и уточнение (abstraction/refinement) для больших систем; типовые абстракции окружения ИИ-сервиса. Параметрическая и модульная верификация: семейства конфигураций и политика управления версиями (data/model versioning). Интерпретация контрпримеров как диагностических сценариев (counterexample analysis) и трассировка до требований. Компетенции: **ПКос-2.1, ПКос-2.2, ПКос-2.3**.
4	**Тема 4. Временные логики, SAT/SMT и формальная верификация свойств ИИ-моделей**	Темпоральные свойства LTL/CTL/CTL*: safety, liveness, fairness; спецификация эксплуатационных ограничений (SLO/SLA, условия остановки, rollback policies). Bounded model checking (BMC), k-индукция, IC3/PDR и интерполяционные методы. SMT-кодирование ограничений для компонентов ИИ: проверка монотонности, инвариантов, ограничений выходов, согласованности правил. Формальная верификация нейросетей и моделей: encodings для ReLU/conv, методы абстрактной интерпретации, релаксации (convex relaxations), оценки робастности к adversarial attacks. Гиперсвойства (Hyperproperties/HyperLTL) для приватности и утечек: membership inference как формализуемое требование, связь с differential privacy. Верификация корректности пост-обработки и guardrails (policy constraints, rule-based filters) в LLM-оркестрации. Компетенции: **ПКос-1.2, ПКос-1.3, ПКос-2.2**.
5	**Тема 5. Инструменты и инженерная практика V&V для ПО и ИИ: доказательства, анализ, CI/CD**	Вероятностная верификация и оценка надежности: Марковские цепи/MDP, probabilistic model checking, метрики риска и отказов. Risk-based V&V: выбор стратегии верификации по критичности и ограничениям ресурсов (cost–benefit, coverage vs assurance). Runtime verification и мониторинг соответствия спецификациям (monitors, traces, drift detection как формализуемое условие). Комбинация формальных методов с тестированием: fuzzing (coverage-guided), concolic/symbolic execution, генерация тестов из спецификаций. Интеграция V&V в CI/CD и MLOps: quality gates, автоматическое доказательство/проверка свойств, воспроизводимость и артефакты доказательств (evidence). Документирование и аргументация безопасности/качества: assurance cases (GSN), трассируемость требований, datasheets/model cards, управление изменениями (change impact analysis). Компетенции: **ПКос-4.1, ПКос-4.2, ПКос-4.3, ПКос-1.3**.

5. Образовательные технологии

Таблица 6

Применение активных и интерактивных образовательных технологий

№ п/п	Тема и форма занятия		Наименование используемых активных и интерактивных образовательных технологий
1	Тема 1. Киберугрозы и специфика кибербезопасности цифровых систем АПК	**Л**	Лекция-визуализация, разбор отраслевых кейсов и сценариев атак, обсуждение моделей угроз и рисков.
		ПЗ	Решение задач профессиональной направленности, проблемно-поисковое занятие, групповое обсуждение (модель угроз, риск-матрица, требования безопасности).
2	Тема 2. Данные безопасности и подготовка датасетов для ИИ-аналитики	**Л**	Лекция-визуализация, демонстрация источников телеметрии и схем данных, разбор примеров нормализации и корреляции событий.
		ПЗ	Решение задач профессиональной направленности, практикум по подготовке данных, проблемно-поисковое занятие, групповое обсуждение (признаки, качество данных).
3	Тема 3. Методы ИИ для детектирования атак и аномалий	**Л**	Лекция-визуализация, разбор архитектур ML-детекторов, обсуждение метрик качества и ошибок детектирования.
		ПЗ	Решение задач профессиональной направленности, практикум по построению и оценке моделей, проблемно-поисковое занятие, групповое обсуждение (ложные срабатывания, интерпретация).
4	Тема 4. Архитектура AI-for-Security и эксплуатация (SOC/SIEM/SOAR, надежность и доверие)	**Л**	Лекция-визуализация, анализ архитектурных схем SOC/SIEM/SOAR и AI-контура, обсуждение эксплуатационных метрик и требований аудита.
		ПЗ	Решение задач профессиональной направленности, проектно-ориентированное занятие, проблемно-поисковое занятие, групповое обсуждение (архитектура, playbooks, мониторинг дрейфа/качества).

6. Текущий контроль успеваемости и промежуточная аттестация по итогам освоения дисциплины

6.1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

1) Примеры заданий практических работ

Практическая работа №2 SAT/SMT-проверка непротиворечивости требований и инвариантов к данным ИИ-пайплайна.

Цель: формализовать требования к данным/фичам в виде ограничений и проверить их выполнимость (SAT/SMT), выявить противоречия и получить unsat core.

Инструменты: Python 3.x, z3-solver (Z3Py), Jupyter.

Источник данных : OpenML (пример: датасет wine-quality): https://www.openml.org/ (выберите dataset id и используйте openml-API).

Задание:

Сформировать набор data contracts для табличного датасета (диапазоны, допустимые категории, допустимые пропуски, взаимные зависимости признаков).

Перевести контракты в SMT-ограничения (теории: арифметика, равенства, множества категорий через EnumSort/Int).

Проверить выполнимость: SAT/UNSAT.

Для UNSAT — извлечь и интерпретировать unsat core: какие именно требования конфликтуют.

Предложить «исправление»: ослабление/уточнение одного из ограничений и повторная проверка.

Порядок выполнения:

1. Загрузите датасет, выпишите 10–15 требований к данным (включая минимум 3 межпризнаковые зависимости).
2. Опишите требования в виде логических формул/предикатов и реализуйте их в Z3Py.
3. Запустите solver.check() и зафиксируйте результат.
4. Если UNSAT: включите assert_and_track, получите unsat_core(), сопоставьте элементы ядра исходным требованиям.
5. Выполните «ремонт спецификации» (например, смягчите порог/исправьте домен) и повторите шаги 3–4.
6. Оформите итоговый набор контрактов (как технические требования к датасету и фичам).

Метрики/критерии:

- Количество требований: ≥ 10, из них межпризнаковых ≥ 3.
- Доля требований, покрытых SMT-моделью: 100%.
- Время решения (solve time), размер unsat core, количество итераций ремонта.
- Качество интерпретации: корректное объяснение причин противоречия и осмысленное исправление.

Форма отчёта: ноутбук + краткая записка (1–2 стр.): требования, формулы, результаты SAT/UNSAT, unsat core, исправления.

Компетенции: ПКос-1.1, ПКос-1.2.

Практическая работа №5 Контрактная верификация функции предобработки данных для ML (pre/post-conditions, инварианты).

Цель: описать корректность функции предобработки через контракты и автоматически находить контрпримеры (symbolic checking через SMT).
Инструменты: Python 3.x, CrossHair (symbolic execution для Python) + icontract (или аналоги).

Источники:

CrossHair: https://github.com/pschanely/CrossHair
icontract: https://github.com/Parquery/icontract

Задание:

1. Реализовать функцию prepare_features(df) (или sanitize_record(x)), которая делает: обработку пропусков, нормализацию, контроль типов/диапазонов.
2. Задать предусловия (типы, допустимые пропуски, минимальные диапазоны) и постусловия (диапазоны после нормализации, отсутствие NaN, монотонность отдельных преобразований и т.п.).
3. Запустить CrossHair для поиска контрпримеров к контрактам.
4. Исправить реализацию или контракты так, чтобы контрпримеры исчезли (или были осмысленно объяснены как невозможные входы).
5. Добавить 3–5 property-based тестов, которые дублируют ключевые свойства (на уровне тестирования).

Порядок выполнения:

1. Согласуйте формат входа (DataFrame/словарь/класс записи) и перечислите свойства корректности.
2. Запишите контракты: pre/post + инварианты.
3. Запустите CrossHair в режиме анализа выбранного модуля/функции.
4. Для каждого найденного контрпримера: классифицируйте проблему (ошибка кода / неверное требование / неполная спецификация).
5. Выполните исправление и повторите анализ до получения «зелёного» результата.
6. Подготовьте краткий «traceability»: какое требование проверялось каким контрактом и каким кейсом контрпримера.

Метрики/критерии:

- Количество формализованных свойств: ≥ 8 (из них ≥ 2 про устойчивость к NaN/Inf/границам).
- Количество контрпримеров «до/после» и качество их интерпретации.
- Доля свойств, перенесённых в автоматические тесты: ≥ 50%.

Форма отчёта: код, журнал контрпримеров (таблица: свойство, контрпример, исправление).
Компетенции: ПКос-1.2, ПКос-1.3.

Практическая работа №9 Model checking оркестрации сервиса ИИ (конечная модель + LTL/CTL-свойства).

Цель: построить переходную систему (FSM) для ИИ-сервиса (запрос → предобработка → инференс → post-check → ответ) и верифицировать безопасность/живучесть.

Инструменты: NuSMV (CTL/LTL) или SPIN (Promela).
Источники:
NuSMV: https://nusmv.fbk.eu/
SPIN: https://spinroot.com/

Сценарий (пример): сервис агроаналитики с ML-моделью и guardrails: состояния: Idle, Auth, Validate, Infer, PolicyCheck, Respond, Reject, FailSafe; события: ok_auth, bad_auth, valid, invalid, policy_ok, policy_bad, timeout, model_error.

Задание:

1. Задать FSM с параметрами (например, лимит попыток, таймаут, fallback).
2. Сформулировать минимум 6 свойств:
    a. safety: «нельзя отвечать без успешной авторизации»,
    b. safety: «при нарушении политики всегда Reject»,
    c. liveness: «при валидном запросе и отсутствии ошибок система в конце концов отвечает»,
    d. fail-safe: «при ошибке модели система переходит в FailSafe и не выдает “успешный” ответ»,
    e. fairness/ограничения по таймауту.
3. Проверить свойства, получить контрпримеры для нарушенных.
4. Исправить модель/условия переходов (или уточнить свойства) и добиться выполнения согласованного набора требований.

Порядок выполнения:

1. Опишите модель (переменные состояний, переходы, ограничения).
2. Запишите свойства в CTL/LTL (не менее 6).
3. Запустите проверку; сохраните результаты и контрпримеры.
4. Проанализируйте контрпример: какой переход/условие приводит к нарушению.
5. Исправьте модель (или уточните предпосылки свойства) и повторите проверку.
6. Оформите итоговую спецификацию требований сервиса как набор темпоральных свойств.

Метрики/критерии:

- Количество свойств: ≥ 6 (safety ≥ 3, liveness ≥ 2).
- Количество состояний/переходов; наличие/отсутствие deadlocks.
- Количество итераций исправления до прохождения набора свойств.

Форма отчёта: файл модели, список свойств, протокол проверки и разбор контрпримеров.
Компетенции: ПКос-2.1, ПКос-2.2, ПКос-2.3.

Практическая работа №14 Вероятностная верификация (probabilistic model checking) стратегии V&V для ИИ-компонента.

Цель: выбрать верификационную стратегию на основе формальной оценки риска/надежности (probability of failure, expected cost).
Инструменты: PRISM.
Источники:
PRISM model checker: https://www.prismmodelchecker.org/

Набор примеров PRISM (страница Examples):
https://www.prismmodelchecker.org/manual/Examples/Introduction
Сценарий (пример): MDP-модель жизненного цикла релиза ML-модуля:
действия: ship_now, run_formal_check, run_fuzzing, run_runtime_monitor, rollback;
события: «дефект проявился в проде», «дефект пойман до релиза», «ложноположительное блокирование релиза».

Задание:

1. Построить MDP/DTMC модель релиза, параметризовав вероятности обнаружения дефекта разными методами (формальная проверка/фаззинг/мониторинг).
2. Посчитать:
    a. вероятность попадания дефекта в прод: P=? [F prod_failure],
    b. ожидаемые затраты: R{"cost"}=? [F done],
    c. ожидаемое время: R{"time"}=? [F done].
3. Провести сравнение 3 стратегий (например: только тесты; тесты+fuzzing; формальная проверка+мониторинг).
4. Сформировать рекомендацию стратегии под заданные ограничения (лимит времени релиза, порог риска).

Порядок выполнения:

1. Задайте состояния и переходы PRISM-модели (с наградами cost/time).
2. Определите минимум 3 альтернативные политики (через разные доступные действия/вероятности).
3. Запустите вычисление вероятностей и наград.
4. Выполните чувствительный анализ: измените ключевой параметр (например, вероятность выявления дефекта формальным методом) и оцените изменение риска/стоимости.
5. Подготовьте вывод: какую стратегию выбрать при заданных ограничениях и почему.

Метрики/критерии:

- Наличие формально рассчитанных показателей: P(failure), E(cost), E(time) для ≥ 3 стратегий.
- Чувствительный анализ по ≥ 1 параметру.
- Обоснованная рекомендация (risk–cost trade-off).

Форма отчёта: PRISM-модель + таблица метрик по стратегиям + вывод (0,5–1 стр.).
Компетенции: ПКос-4.1, ПКос-4.2, ПКос-4.3.

2) Примерный перечень вопросов, выносимых на промежуточную аттестацию (Экзамен)

1. Понятия верификации и валидации (V&V). Место формальных методов в жизненном цикле ПО и в процессах CI/CD.
2. Корректность программ: спецификация, частичная и полная корректность. Предусловия, постусловия, инварианты.
3. Исчисление высказываний и предикатов: синтаксис, семантика, выводимость. Роль логики в формализации требований.
4. Булева алгебра и нормальные формы (CNF/DNF). Сведение задач проверки к SAT.
5. SAT: постановка задачи, понятие выполнимости, типовые подходы SAT-решателей (DPLL/CDCL) и их применение в верификации.
6. SMT: отличие от SAT, теории (линейная арифметика, массивы, битовые векторы, неинтерпретируемые функции). Типовой SMT-пайплайн.
7. Практика использования SMT для проверки контрактов и инвариантов данных (data contracts). Unsat core и его интерпретация.
8. Логика Хоара: тройки Хоара, правила вывода для присваивания, ветвлений, циклов.
9. Инварианты циклов: назначение, методы построения, типовые ошибки при выборе инварианта.
10. Символьное выполнение: идея, ограничения, связь с SMT. Контрпримеры и их роль в уточнении спецификаций.
11. Абстрактная интерпретация: домены абстракций, аппроксимации, soundness/precision trade-off, типовые классы ошибок (переполнение, null, диапазоны).
12. Анализ потоков управления и данных (CFG/DFG): определения, задачи достижимости, живости, доступности выражений.
13. Taint analysis: постановка, источники/приёмники, правила распространения, примеры уязвимостей.
14. Конечные автоматы и системы переходов: формализация поведения программ/протоколов/оркестраций сервисов.
15. Model checking: идея исчерпывающего перебора состояний, проблема взрыва состояний и способы её смягчения.

16. Темпоральные логики LTL и CTL: основные операторы, различия выразительности и типовые классы свойств.
17. Формулирование требований безопасности (safety) и живучести (liveness) в LTL/CTL на примере сервисов и ИИ-пайплайнов.
18. Контрпример в model checking: структура, интерпретация, исправление модели или уточнение требования.
19. Теоремные доказатели (Coq/Isabelle/HOL/Lean): роль интерактивных доказательств, преимущества и ограничения по сравнению с model checking.
20. Refinement и abstraction в верификации: смысл уточнения спецификации и доказательство корректности уточнения.
21. Формальные спецификации требований: подходы и нотации (например, Z/VDM/Alloy) и их применимость в промышленной разработке.
22. Проверка архитектурных решений и контрактов компонентов: верифицируемая архитектура, traceability требований.
23. Runtime verification: мониторинг свойств во времени, спецификации мониторинга, ограничения и области применения.
24. Fuzzing: базовая идея, coverage-guided fuzzing, метрики покрытия, ограничения и риски ложных выводов.
25. Связка fuzzing + symbolic execution: назначение гибридных подходов, типовые сценарии использования.
26. Вероятностная верификация: DTMC/MDP, интуиция вероятностного model checking. Примеры метрик риска/надёжности.
27. Risk-based verification: как выбирать методы V&V при ограничениях проекта (стоимость/время/риск).
28. Метрики качества и надёжности результатов верификации: soundness vs completeness, ложноположительные/ложноотрицательные срабатывания.
29. Верификация ML/ИИ-компонентов: специфика требований (robustness, безопасность данных, устойчивость к сдвигу распределений), ограничения формальных методов.
30. Типовые классы атак на ИИ/данные и их отражение в спецификациях (data poisoning, model extraction, membership inference) на уровне требований и контроля.

6.2. Описание показателей и критериев контроля успеваемости, описание шкал оценивания

Оценочные средства текущего контроля успеваемости и сформированности компетенций основана на подсчете баллов, «заработанных» студентом в течение семестра.

Успеваемость студента по дисциплине оценивается в баллах от 0 до 100.

Оценка знаний проводится по следующим критериям:

- посещение занятий – 30 баллов;
- выполнение практических заданий – 30 баллов;

− промежуточный контроль (зачет) – 40 баллов;

Соответствие балльной оценки общепринятой 4-х балльной шкале оценок приведено в таблице 7.

Таблица 7

Соответствие балльных оценок по 4-х балльной шкале

Балльная оценка	Оценка по 4хбалльной шкале	Оценка по шкале «Зачтено» / «Не зачтено»
0-59	Неудовлетворительно - 2	Не зачтено
60-69	Удовлетворительно - 3	Зачтено
70-89	Хорошо – 4	Зачтено
90-100	Отлично - 5	Зачтено

Критерии оценивания результатов обучения показаны в таблице 8.

Таблица 8

Критерии оценивания результатов обучения (Экзамен)

Оценка	**Критерии оценивания**
Высокий уровень «5» (отлично)	оценку **«отлично»** заслуживает студент, освоивший знания, умения, компетенции и теоретический материал без пробелов; выполнивший все задания, предусмотренные учебным планом на высоком качественном уровне; практические навыки профессионального применения освоенных знаний сформированы. **Компетенции**, закреплённые за дисциплиной, **сформированы на уровне – высокий**.
Средний уровень «4» (хорошо)	оценку **«хорошо»** заслуживает студент, практически полностью освоивший знания, умения, компетенции и теоретический материал, учебные задания не оценены максимальным числом баллов, в основном сформировал практические навыки. **Компетенции**, закреплённые за дисциплиной, **сформированы на уровне – хороший (средний)**.
Пороговый уровень «3» (удовлетворительно)	оценку **«удовлетворительно»** заслуживает студент, частично с пробелами освоивший знания, умения, компетенции и теоретический материал, многие учебные задания либо не выполнил, либо они оценены числом баллов близким к минимальному, некоторые практические навыки не сформированы. **Компетенции**, закреплённые за дисциплиной, **сформированы на уровне – достаточный**.
Минимальный уровень «2» (неудовлетворительно)	оценку «**неудовлетворительно»** заслуживает студент, не освоивший знания, умения, компетенции и теоретический материал, учебные задания не выполнил, практические навыки не сформированы.

	Компетенции, закреплённые за дисциплиной, **не сформированы**.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1. Основная литература

1. Старолетов С. М. Основы тестирования и верификации программного обеспечения: учебное пособие для вузов. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023. – 344 с. – URL: https://e.lanbook.com/book/319445. – ISBN 978-5-507-46773-0.
2. Казарин О. В., Шубинский И. Б. Надежность и безопасность программного обеспечения: учебник для вузов. – Электрон. дан. – Москва: Юрайт, 2025. – 342 с. – (Высшее образование). – URL: https://urait.ru/bcode/563862. – ISBN 978-5-534-05142-1.

7.2. Дополнительная литература

1. Казарин О. В., Забабурин А. С. Программно-аппаратные средства защиты информации. Защита программного обеспечения: учебник и практикум для вузов. – Электрон. дан. – Москва: Юрайт, 2024. – 312 с. – (Высшее образование). – URL: https://urait.ru/bcode/538066. – ISBN 978-5-9916-9043-0.
2. Миронов А. И., Трушин С. М., Петренко А. А. Тестирование и верификация программного обеспечения: Практикум: практикум. – Москва: РТУ МИРЭА, 2022. – 65 с. – URL: https://e.lanbook.com/book/240095.
3. Вендров А. М. Практикум по проектированию программного обеспечения экономических информационных систем: учебное пособие для студентов высших учебных заведений, обучающихся по специальностям "Прикладная информатика в экономике", "Математическое обеспечение и администрирование информационных систем". – Москва: Финансы и статистика, 2006. – 191 с.: ил. – Библиогр.: с. 189 (7 назв.). – ISBN 5-279-03106-2.

7.3. Нормативные правовые акты

1. Гост 19.001-77. Единая система программной документации: Общие положения. – М.: Изд.-во стандартов, 1994.
2. Гост 19.101-77. Единая система программной документации: Виды программ и программных документов. – М.: Изд.-во стандартов, 1994.
3. Гост 19.102-77. Единая система программной документации: Стадии разработки. – М.: Изд.-во стандартов, 1994.
4. Гост 19.105-78. Единая система программной документации: Общие требования к программным документам. – М.: Изд.-во стандартов, 1994.

5. Гост 19.201-78. Единая система программной документации: Техническое задание. Требования к содержанию и оформлению. – М.: Изд.-во стандартов, 1994.

6. Гост 19.202-78. Единая система программной документации: Спецификация. Требования к содержанию и оформлению. – М.: Изд.-во стандартов, 1994.

7. Гост 19.502-78. Единая система программной документации: Описание применения. Требования к содержанию и оформлению. – М.: Изд.-во стандартов, 1994.

8. Гост 19.404-79. Единая система программной документации: Пояснительная записка. Требования к содержанию и оформлению. – М.: Изд.-во стандартов, 1994.

9. Гост 3.11.09-82. Система технологической документации: Термины и определения основных понятий. – М.: Изд.-во стандартов, 1994.

10. Гост 34.201-89. Виды, комплектность и обозначение документов при создании автоматизированных систем. – М.: Изд.-во стандартов, 1991.

11. ГОСТ 34.601-90. Автоматизированные Системы Стадии создания. Комплекс стандартов на автоматизированные системы. - М.: Изд-во стандартов, 1997

12. ISO/IEC 12207:1995

13. Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации».
14. Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации».
15. Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».
16. Федеральный закон от 26.07.2017 № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации».
17. Федеральный закон от 29.07.2004 № 98-ФЗ «О коммерческой тайне».
18. Федеральный закон от 06.04.2011 № 63-ФЗ «Об электронной подписи».
19. Постановление Правительства РФ от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных».
20. Постановление Правительства РФ от 15.09.2008 № 687 «Об утверждении Положения об особенностях обработки персональных данных, осуществляемой без использования средств автоматизации».
21. Постановление Правительства РФ от 16.11.2015 № 1236 «Об установлении запрета на допуск программного обеспечения, происходящего из иностранных государств, для целей осуществления закупок для обеспечения государственных и муниципальных нужд».
22. Приказ ФСТЭК России от 18.02.2013 № 21 «Об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных».

23. Приказ ФСТЭК России от 25.12.2017 № 239 «Об утверждении Требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации».
24. Указ Президента РФ от 10.10.2019 № 490 «О развитии искусственного интеллекта в Российской Федерации» (в части общих рамок развития и применения ИИ, актуальных для задач верификации/оценки корректности ИИ-компонентов).

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. NuSMV (Symbolic Model Checker), документация и загрузки — https://nusmv.fbk.eu/
2. Документация SPIN Model Checker (Promela) — https://spinroot.com/spin/whatispin.html
3. UPPAAL (моделирование и верификация временных автоматов), официальный сайт — https://uppaal.org/
4. PRISM (probabilistic model checking), официальный сайт и руководство — https://www.prismmodelchecker.org/
5. TLA+ (спецификация и модельная проверка), официальный сайт — https://lamport.azurewebsites.net/tla/tla.html
6. TLC (модельная проверка TLA+), репозиторий/инструменты — https://github.com/tlaplus/tlaplus
7. Event-B и платформа Rodin (формальная разработка), официальный сайт — https://www.event-b.org/
8. Alloy Analyzer (реляционная логика и анализ моделей), официальный сайт — https://alloytools.org/
9. Coq (интерактивное доказательство теорем), официальный сайт — https://coq.inria.fr/
10. Isabelle/HOL (theorem prover), официальный сайт — https://isabelle.in.tum.de/
11. Lean (theorem prover), официальный сайт — https://leanprover-community.github.io/
12. Z3 SMT Solver, официальный сайт — https://github.com/Z3Prover/z3
13. cvc5 SMT Solver, официальный сайт — https://cvc5.github.io/
14. KLEE (symbolic execution для C/LLVM), официальный сайт — https://klee.github.io/
15. angr (symbolic execution/анализ бинарей), документация — https://docs.angr.io/
16. AFL++ (coverage-guided fuzzing), официальный репозиторий — https://github.com/AFLplusplus/AFLplusplus
17. LLVM LibFuzzer (fuzzing), документация — https://llvm.org/docs/LibFuzzer.html

18. NIST SATE / Software Assurance (материалы по анализу/верификации), портал — https://samate.nist.gov/
19. SEI CERT (безопасное программирование и свойства корректности), рекомендации — https://wiki.sei.cmu.edu/
20. OWASP ASVS (требования к верификации безопасности приложений, как источник проверяемых требований) — https://owasp.org/www-project-application-security-verification-standard/
21. Microsoft SDL (жизненный цикл разработки с встраиванием верификации/контролей) — https://www.microsoft.com/en-us/securityengineering/sdl/
22. Документация SonarQube (статический анализ и правила качества) — https://docs.sonarsource.com/sonarqube/latest/
23. Pylint (статический анализ Python; правила/конфигурации) — https://pylint.pycqa.org/
24. PMD (статический анализ; правила/плагины) — https://pmd.github.io/

9. Перечень программного обеспечения и информационных справочных систем

9.1. Программное обеспечение

1. Python 3.x (реализация примеров, автоматизация проверок, интеграция SMT) — https://www.python.org/
2. Anaconda / Miniconda (управление окружениями Python) — https://www.anaconda.com/ / https://docs.conda.io/en/latest/miniconda.html
3. Jupyter Notebook / JupyterLab (оформление вычислительных экспериментов и отчётов) — https://jupyter.org/
4. Visual Studio Code (редактор/IDE) — https://code.visualstudio.com/
5. PyCharm Community Edition (альтернативная IDE) — https://www.jetbrains.com/pycharm/
6. Git (контроль версий, воспроизводимость артефактов V&V) — https://git-scm.com/
7. Docker (воспроизводимые окружения, запуск инструментов в контейнерах) — https://www.docker.com/
8. Z3 SMT Solver (решение SMT-задач, доказательство/контрпримеры) — https://github.com/Z3Prover/z3
9. cvc5 SMT Solver (альтернативный SMT-решатель) — https://cvc5.github.io/
10. Alloy Analyzer (моделирование/поиск контрпримеров в реляционных спецификациях) — https://alloytools.org/
11. SPIN Model Checker (проверка моделей, Promela) — https://spinroot.com/
12. NuSMV (model checking, CTL/LTL) — https://nusmv.fbk.eu/
13. UPPAAL (верификация временных автоматов, timed systems) — https://uppaal.org/
14. PRISM (вероятностная верификация, DTMC/MDP) — https://www.prismmodelchecker.org/

15. TLA+ / TLC (спецификация и модельная проверка распределённых алгоритмов) — https://lamport.azurewebsites.net/tla/tla.html
16. Coq (интерактивное доказательство теорем) — https://coq.inria.fr/
17. Isabelle/HOL (theorem prover) — https://isabelle.in.tum.de/
18. Lean (theorem prover; формализация доказательств) — https://leanprover-community.github.io/
19. KLEE (symbolic execution для C/LLVM) — https://klee.github.io/
20. angr (символьное выполнение/анализ бинарей) — https://docs.angr.io/
21. AFL++ (coverage-guided fuzzing) — https://github.com/AFLplusplus/AFLplusplus
22. LLVM LibFuzzer (фаззинг) — https://llvm.org/docs/LibFuzzer.html
23. SonarQube Community (статический анализ, правила качества) — https://www.sonarsource.com/products/sonarqube/
24. Pylint (статический анализ Python) — https://pylint.pycqa.org/

9.2. Информационные справочные системы и ресурсы

1. NIST SATE / SAMATE (материалы по software assurance, тестовые наборы и методики) — https://samate.nist.gov/
2. SEI CERT Coding Standards (рекомендации безопасного программирования, типовые дефекты) — https://wiki.sei.cmu.edu/
3. OWASP ASVS (каталог требований безопасности, пригодных для формализации/трассировки) — https://owasp.org/www-project-application-security-verification-standard/
4. Microsoft Security Development Lifecycle (SDL) (организация процессов разработки и контроля качества/безопасности) — https://www.microsoft.com/en-us/securityengineering/sdl/
5. MITRE CWE (классификация слабостей ПО как база для формулирования проверяемых требований) — https://cwe.mitre.org/
6. MITRE CVE (каталог уязвимостей; примеры для кейсов анализа) — https://www.cve.org/
7. NVD (National Vulnerability Database) (метаданные по уязвимостям, CVSS, CPE) — https://nvd.nist.gov/

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Таблица 10

Перечень программного обеспечения

№ п/п	Наименование раздела учебной дисциплины	Наименование программы	Тип программы	Автор (разработчик)	Год разработки

1	Темы 1–5	Microsoft Office (Word, PowerPoint, Excel)	Офисный пакет (подготовка текста, таблиц, презентаций)	Microsoft	1989
2	Темы 1–5	Visual Studio Code	Редактор исходного кода / IDE	Microsoft	2015
3	Темы 1–5	Git	Система контроля версий	Linus Torvalds / сообщество	2005
4	Темы 2–5	Python	Язык программирования / среда вычислений	Python Software Foundation	1991
5	Темы 2–5	Anaconda / Miniconda	Дистрибутив и менеджер окружений Python	Anaconda Inc. / сообщество	2012
6	Темы 2–5	Jupyter Notebook / JupyterLab	Интерактивная среда для вычислений и отчётности	Project Jupyter	2014
7	Темы 4–5	Docker	Платформа контейнеризации и воспроизводимых окружений	Docker Inc. / сообщество	2013
8	Темы 1–5	Z3	SMT-решатель (satisfiability modulo theories)	Microsoft Research	2007
9	Темы 1–5	cvc5	SMT-решатель	cvc5 project / сообщество	2020
10	Темы 2–3	SPIN	Model checking (LTL), язык Promela	Gerard J. Holzmann / сообщество	1991
11	Темы 2–3	NuSMV	Символьный model checker (CTL/LTL)	NuSMV project / FBK (IRST) / сообщество	1999
12	Темы 2–3	UPPAAL	Верификация временных автоматов (timed automata)	Uppsala Univ. / Aalborg Univ.	1995
13	Темы 2–3	PRISM	Вероятностный model checker (DTMC/CTMC/MDP)	PRISM team / академическое сообщество	2001
14	Темы 1–3	Coq	Интерактивное доказательство теорем	INRIA	1989
15	Темы 1–3	Isabelle/HOL	Интерактивное доказательство теорем	Isabelle community	1986
16	Темы 1–3	Lean	Интерактивное доказательство теорем	Microsoft Research / сообщество	2013

17	Темы 1–2	Alloy Analyzer	Анализ формальных спецификаций, поиск контрпримеров	MIT / Alloy community	2000
18	Темы 4–5	KLEE	Символьное выполнение (LLVM)	KLEE project / академическое сообщество	2008
19	Темы 4–5	angr	Анализ бинарного кода / символическое выполнение	Trail of Bits / сообщество	2014
20	Темы 4–5	AFL++ (или AFL)	Coverage-guided fuzzing	AFL/AFL++ community	2007/2019
21	Темы 4–5	LLVM LibFuzzer	Фаззинг (in-process)	LLVM community	2014
22	Темы 4–5	SonarQube Community	Платформа статического анализа и качества кода	SonarSource	2007
23	Темы 4–5	Pylint	Статический анализ кода Python	PyCQA / сообщество	1998

Сведения об обеспеченности специализированными аудиториями, кабинетами, лабораториями

Наименование специальных* помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	**Оснащенность специальных помещений и помещений для самостоятельной работы****
1	2
Корпус 1, Аудитория 201 Количество рабочих мест: 24	Встроенные сетевые адаптеры (Intel I219-V или Realtek RTL8111H), интерфейс RJ-45, скорость 10/100/1000 Мбит/с. Точки доступа: Ubiquiti UniFi AP AC Pro, стандарты IEEE 802.11a/b/g/n/ac, частоты 2.4 ГГц (450 Мбит/с) и 5 ГГц (1300 Мбит/с), поддержка MU-MIMO, питание PoE.
Корпус 1, Аудитория 203 Количество рабочих мест: 18	Встроенные сетевые адаптеры (Intel I219-V или Realtek RTL8111H), интерфейс RJ-45, скорость 10/100/1000 Мбит/с. Точки доступа: Ubiquiti UniFi AP AC Pro, стандарты IEEE 802.11a/b/g/n/ac, частоты 2.4 ГГц (450 Мбит/с) и 5 ГГц (1300 Мбит/с), поддержка MU-MIMO, питание PoE.Структурное подразделение: Кафедра Цифровая кафедра
Корпус 1, Аудитория 206 Количество рабочих мест: 24	Встроенные сетевые адаптеры (Intel I219-V или Realtek RTL8111H), интерфейс RJ-45, скорость 10/100/1000 Мбит/с. Точки доступа: Ubiquiti UniFi AP AC Pro, стандарты IEEE 802.11a/b/g/n/ac, частоты 2.4 ГГц (450 Мбит/с) и 5 ГГц (1300 Мбит/с), поддержка MU-MIMO, питание PoE.
Центральная научная библиотека имени Н.И. Железнова	Читальные залы библиотеки
Студенческое общежитие	Комната для самоподготовки

11. Методические рекомендации обучающимся по освоению дисциплины

Образовательный процесс по дисциплине организован в форме учеб-ных занятий (контактная работа (аудиторной и внеаудиторной) обучающихся с

преподавателем и самостоятельная работа обучающихся). Учебные занятия(в том числе по реализации практической подготовки) представлены следу-ющими видами, включая учебные занятия, направленные на практическую подготовку обучающихся и проведение текущего контроля успеваемости:

лекции (занятия лекционного типа);

семинары, практические занятия, лабораторные работы (занятия семинарского типа);

индивидуальные консультации и иные учебные занятия, предусматривающие индивидуальную работу преподавателя с обучающимся;

самостоятельная работа обучающихся;

занятия иных видов.

На учебных занятиях обучающиеся выполняют запланированные настоящей программой отдельные виды учебных работ, в том числе отдельных элементов работ, связанных с будущей профессиональной деятельностью.

Виды и формы отработки пропущенных занятий

Студент, пропустивший занятия обязан отработать:

Пропущенные лекции – предоставив преподавателю конспект лекции, ответив на вопросы устно, пройдя собеседование по пропущенной теме, пройти тестирование.

Пропущенные практические занятия – в форме выполненных заданий, устного опроса, посещения дополнительных занятий.

Защита индивидуальных заданий проводятся в часы в дни и часы, устанавливаемые преподавателем.

Пропуск занятия по документально подтвержденной дирекцией уважительной причине не является основанием для снижения оценки выполненной практической работы.

Методические рекомендации преподавателям по организации обучения по дисциплине

Преподавание курса должно носить контекстный характер. В процессе обучения должна четко прослеживаться целевая установка на развитие личности; интеграционное единство форм, методов и средств обучения; взаимодействие обучаемых и педагогов; индивидуальный стиль педагогической деятельности.

Реализация технологий контекстного обучения в профессионально-образовательном процессе обеспечивается соблюдением следующих условий:

– мотивационное обеспечение субъектов педагогической деятельности и учение, основанное на реализации их личностных функций в этом процессе;

– наличие четкой и диагностически заданной цели образования, т.е. измеримого представления об ожидаемом результате;

– представление учебного материала в виде системы познавательных и практических задач, ситуаций, заданий, проектов, упражнений и др.;

– указание способов взаимодействия субъектов профессионально-

образовательного процесса;

– обозначение границ правилосообразной (алгоритмической) и творческой деятельности педагогов, допустимого отклонения от правил;

– обеспечение открытости обучения профессиональному будущему, направленность на его предвосхищение.

В результате изучения дисциплины студенты получают знания о распространении программного обеспечения, а также методологии и стандартах на основе лицензии и договоров, а также применять достижения отечественной и зарубежной науки и практики.

Методика преподавания дисциплины строится на сочетании лекций с практическими занятиями; групповыми и индивидуальными консультациями по отдельным разделам программы; внеаудиторной самостоятельной работой студентов (работа с учебниками, учебными пособиями, методическими указаниями, заданиями, специальной литературой, поиск необходимой информации в сети Интернет).

Лекционный курс, как одна из составляющей дисциплины, должен быть логическим и последовательным. Каждая лекция должна, согласно правилам дидактики, начинаться с актуализации знаний. Чтение лекций должно происходить на основе проблемного метода обучения, что будет стимулировать деятельность студентов к самостоятельному поиску знаний. Интерес к изучению материала преподаватель должен стимулировать, используя наглядные методы обучения (мультимедийные презентации, иллюстрации, стенды и т.д.). Помимо традиционной лекции необходимо использовать проблемные лекции, лекции-визуализации, бинарные лекции, дискуссии и т.д.

В начале каждой лекции следует четко формулировать цель, которую необходимо достигнуть посредством решения ряда задач. При этом сами задачи должны быть четко оговорены. Важная роль на лекции должна быть отведена дискуссии. Преподаватель заранее должен продумать траекторию изучения материала с вовлечением студентов в дискуссии. Это позволит на смену авторитарному методу обучению, укоренившемуся в современной системе образования, быть студентам собеседниками преподавателя. Эта особенность лекции важна для более глубокого понимания изучаемого материала.

Как и любое занятие, лекция должна заканчиваться подведением итогов и формулировкой выводов.

Что касается практических занятий, то для них должны соблюдаться такая же структура, как и для лекционных занятий: актуализация знаний, постановка цели и задач и т.д. Практическая работа также должна соответствовать принципам контекстного подхода, с использованием решения исследовательских задач профессиональной направленности. На практических занятиях должны быть использованы технологии дифференцированного обучения студентов, уделяя большее внимание «слабым» студентам.

Практические занятия проводятся под руководством преподавателя. В рамках этих занятий производится анализ типовых ошибок, допущенных при выполнении заданий, рассматриваются наиболее удачные варианты. Студенты привлекаются к разбору и сравнительному анализу предлагаемых вариантов

решений. Происходит коллективное обсуждение, в результате которого приобретаются навыки ведения дискуссии по обсуждаемым вопросам.

Успех закрепления знаний и умений определяется стройной системой подобранных вопросов для текущего контроля.

В процессе самостоятельной работы студенты отрабатывают теоретические положения, изложенные на лекциях, и изучают примеры, рассмотренные на практических занятиях.

Конкретная тема обсуждается с каждым студентом и учитывает направление научных интересов студента или тему выпускной квалификационной работы.

Большое значение в ходе самостоятельной работы студентов имеет работа над литературой и другими источниками информации (периодические издания, Интернет и т.д.).

Особенности методики преподавания данной дисциплины состоят в интенсификации теоретической, практической и самостоятельной работы студентов и широком применении активных и интерактивных форм и методов обучения.

Программу разработал:

Греченева А.В. к.т.н.,доцент _____

РЕЦЕНЗИЯ

на рабочую программу дисциплины Б1.В.08 «Математические основы верификации программного обеспечения» ОПОП ВО по направлению подготовки 09.04.03 «Прикладная информатика», направленности «Архитектура систем искусственного интеллекта» (квалификация выпускника – магистр)

Ашмариной Татьяной Игоревной, кандидатом экономических наук, доцентом кафедры экономики и организации производства ФГБОУ ВО «Российский государственный аграрный университет – МСХА им. К.А. Тимирязева» (далее по тексту рецензент), проведено рецензирование рабочей программы дисциплины Б1.В.08 «Математические основы верификации программного обеспечения» ОПОП ВО по направлению подготовки 09.04.03 «Прикладная информатика», направленности «Архитектура систем искусственного интеллекта» (магистратура) разработанной в ФГБОУ ВО «Российский государственный аграрный университет – МСХА имени К.А. Тимирязева», на кафедре прикладной информатики (разработчик – Греченева Анастасия Владимировна, доцент кафедры прикладной информатики, кандидат технических наук).

Рассмотрев представленные на рецензирование материалы, рецензент пришел к следующим выводам:

1. Предъявленная рабочая программа дисциплины «Математические основы верификации программного обеспечения» (далее по тексту Программа) *соответствует* требованиям ФГОС ВО по направлению *подготовки* 09.04.03 «Прикладная информатика». Программа *содержит* все основные разделы, *соответствует* требованиям к нормативно-методическим документам.

2. Представленная в Программе *актуальность* учебной дисциплины в рамках реализации ОПОП ВО *не подлежит сомнению* – дисциплина относится к дисциплинам по выбору части формируемой участниками образовательного процесса учебного цикла – Б1.О.

3. Представленные в Программе *цели* дисциплины *соответствуют* требованиям ФГОС ВО направления подготовки 09.04.03 «Прикладная информатика».

4. В соответствии с Программой за дисциплиной «Математические основы верификации программного обеспечения» закреплено 3 компетенции (9 индикаторов). Дисциплина «Математические основы верификации программного обеспечения» и представленная Программа *способна реализовать* ее в объявленных требованиях. Результаты обучения, представленные в Программе в категориях знать, уметь, владеть *соответствуют* специфике и содержанию дисциплины и *демонстрируют возможность* получения заявленных результатов.

5. Общая трудоёмкость дисциплины «Математические основы верификации программного обеспечения» составляет 4 зачётных единицы (144 часа).

6. Информация о взаимосвязи изучаемых дисциплин и вопросам исключения дублирования в содержании дисциплин *соответствует* действительности. Дисциплина «Математические основы верификации программного обеспечения» взаимосвязана с другими дисциплинами ОПОП ВО и Учебного плана по направлению 09.04.03 «Прикладная информатика».

7. Представленная Программа предполагает использование современных образовательных технологий, используемые при реализации различных видов учебной работы. Формы образовательных технологий *соответствуют* специфике дисциплины.

8. Программа дисциплины «Математические основы верификации программного обеспечения» предполагает проведение занятий в интерактивной форме.

9. Виды, содержание и трудоёмкость самостоятельной работы студентов, представленные в Программе, *соответствуют* требованиям к подготовке выпускников, содержащимся во ФГОС ВО направления 09.04.03 «Прикладная информатика».

10. Представленные и описанные в Программе формы *текущей* оценки знаний (защита практических работ, групповое обсуждение) *соответствуют* специфике дисциплины и

требованиям к выпускникам. Форма промежуточного контроля знаний студентов, предусмотренная Программой, осуществляется в форме зачета в 4 семестре, что *соответствует* статусу дисциплины, как дисциплины, включенной в дисциплины по выбору части формируемой участниками образовательного процесса учебного цикла – Б1.В.ФТД. ФГОС ВО направления 09.04.03 «Прикладная информатика».

11. Формы оценки знаний, представленные в Программе, *соответствуют* специфике дисциплины и требованиям к выпускникам.

12. Учебно-методическое обеспечение дисциплины представлено: основной литературой – 2 источника, дополнительной литературой – 3 наименования, Интернет-ресурсы – 6 источников и *соответствует* требованиям ФГОС ВО направления 09.04.03 «Прикладная информатика».

13. Материально-техническое обеспечение дисциплины соответствует специфике дисциплины «Математические основы верификации программного обеспечения» и обеспечивает использование современных образовательных, в том числе интерактивных методов обучения.

14. Методические рекомендации студентам и методические рекомендации преподавателям по организации обучения по дисциплине дают представление о специфике обучения по дисциплине «Технологии искусственного интеллекта в кибербезопасности АПК».

ОБЩИЕ ВЫВОДЫ

На основании проведенного рецензирования можно сделать заключение, что характер, структура и содержание рабочей программы дисциплины «Математические основы верификации программного обеспечения» ОПОП ВО по направлению 09.04.03 «Прикладная информатика», направленности «Архитектура систем искусственного интеллекта» (квалификация выпускника – магистр), разработанная Греченевой А.В., соответствует требованиям ФГОС ВО, современным требованиям экономики, рынка труда и позволит при её реализации успешно обеспечить формирование заявленных компетенций.

Рецензент:
Ашмарина Т.И., кандидат экономических наук, доцент, доцент кафедры экономики и организации производства ФГБОУ ВО РГАУ-МСХА имени К.А. Тимирязева

«28» августа 2025 г.