Министерство науки и высшего образования Российской Федерации
Рубцовский индустриальный институт (филиал)
ФГБОУ ВО «Алтайский государственный технический
университет им. И.И. Ползунова»

Кафедра прикладной математики

Л.А. ПОПОВА

ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания для студентов направления
«Информатика и вычислительная техника»
очной и заочной форм обучения

Рубцовск 2021

ББК 32.973.2

Попова, Л.А. Тестирование и верификация программного обеспечения: Методические указания для студентов направления Информатика и вычислительная техника» очной и заочной форм обучения / Л.А. Попова; Рубцовский индустриальный институт. – Рубцовск: РИИ, 2021. – 18 с. [ЭР].

Методические указания предназначены для организации самостоятельной работы по курсу «Тестирование и верификация программного обеспечения» у студентов направления 09.03.01 «Информатика и вычислительная техника» очной и заочной форм обучения.

Рассмотрены и одобрены на заседании кафедры прикладной математики Рубцовского индустриального института.
Протокол № 9 от 18.03.2021 г.

© Рубцовский индустриальный институт, 2021

Содержание

Введение

Методические указания написаны в соответствии с программой дисциплины «Тестирование и верификация программного обеспечения» для студентов направления «Информатика и вычислительная техника» очной и заочной форм обучения, предназначены для самостоятельной работы по данному курсу.

Указания содержат комплект тестовых заданий для подготовки к промежуточной аттестации.

Перечень планируемых результатов обучения по дисциплине, соотнесенных с индикаторами достижения компетенций

Компе-тенция	Содержание компетенции	Индика-тор	Содержание индикатора
ПК-12	Способен разрабатывать стратегии тестирования и управление процессом тестирования, разрабатывать документы для тестирования и анализировать качество покрытия	ПК-12.1	Применяет методы тестирования для оценки работоспособности и эффективности программного обеспечения
		ПК-12.2	Анализирует результаты тестирования

Объем дисциплины в зачетных единицах с указанием количества академических часов, выделенных на контактную работу обучающегося с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающегося

Общий объем дисциплины в з.е. /час: 3 / 108
Форма промежуточной аттестации: Зачет

Форма обучения	Виды занятий, их трудоемкость (час.)			Объем контактной работы обучающегося с преподавателем (час)
	Лекции	Лабораторные работы	Самостоятельная работа	
очная	16	16	76	43
заочная	6	8	94	18

Содержание дисциплины, структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

Содержание дисциплины для очной формы обучения

Лекционные занятия (16ч.)

1. Документация, сопровождающая процесс верификации и тестирования {беседа} (2ч.)[1,2,4,5] Роль тестирования в разработке программного обеспечения, сопровождении и функционировании программного обеспечения. Роль тестировщика в команде, его задачи и зона ответственности. Технологические процессы верификации в проекте, документация, создаваемая в ходе жизненного цикла проекта, ее назначение. Стратегия и планы верификации. Отчеты о прохождении тестов.

2. Тестирование программного кода и анализ результатов тестирования {беседа} (2ч.)[1,2,4,5] Задачи и цели тестирования программного кода. Методы тестирования («Черный ящик», «Белый ящик», инспекции). Тестовое окружение. Типы тестовых примеров.

3. Модульное и системное тестирование. Документирование проекта {лекция с разбором конкретных ситуаций} (2ч.)[1,2,4,5] Задачи и цели модульного тестирования. Понятие модуля и его границ. Тестирование классов. Подходы к проектированию тестового окружения. Организация модульного тестирования. Виды системного тестирования. Системное тестирование, приемо-сдаточные и сертификационные испытания при разработке сертифицируемого программного обеспечения.

4. Тестирование пользовательского интерфейса {лекция с разбором конкретных ситуаций} (2ч.)[1,2,4,5,10] Задачи и цели тестирования пользовательского интерфейса. Функциональное тестирование пользовательских интерфейсов. Проверка требований к пользовательскому интерфейсу. Полнота покрытия пользовательского интерфейса. Тестирование удобства использования пользовательских интерфейсов.

5. Методы тестирования для оценки работоспособности и эффективности программного обеспечения(2ч.)[1,2,4,5,10] Анализ алгоритмической сложности программ. Оценка алгоритмической сложности программ. Построение графа сложности программы. Метрика Мак-Кейба и другие характеристики графа потокауправления программы. Примеры расчета метрики алгоритмической сложности.

6. Функциональное автоматизированное тестирование(2ч.)[1,2,5,6,8,10] Системы автоматизированного функционального тестирования. Подход Apple к автоматизации. Тестирование Web-приложений с помощью Selenium. Использование модуля unittest для Python.

7. Тестирование сайтов и баз данных {лекция с разбором конкретных ситуаций} (2ч.)[1,2,6,7,8,9,10] Тестирование домашней страницы при помощи

модульных тестов. Взаимодействие с сайтом. Скрапинг с помощью удаленных серверов. Подключение формы для отправки POST-запроса. Сохранение POST-запроса в базу данных.

8. Поддержка процесса тестирования при промышленной разработке программного обеспечения(2ч.)[1,2,3,4,5] Задачи и цели управления качеством. Аудит процессов разработки и верификации. Корректирующие действия и коррекция процессов. Задачи и процедуры процесса конфигурационного управления. Уровни управления данными. Управление качеством и конфигурационное управление при разработке сертифицируемого программного обеспечения.

Лабораторные работы (16ч.)

1. Разработка и составление сопроводительной документации для проведения тестирования {тренинг} (2ч.)[1,2,3,9,10]

2. Методы, основанные на спецификациях (методы «черного ящика») {тренинг} (2ч.)[1,2,3,7,9,10]

3. Тестирование на основе структуры (методы «белого ящика») {тренинг} (2ч.)[1,2,3,7,9,10]

4. Планирование и оценка тестирования. Анализ результатов тестирования {тренинг} (2ч.)[1,2,3,9,10]

5. Тестирование пользовательского интерфейса {тренинг} (2ч.)[1,2,3,7,9,10]

6. Тестирование с использованием модуля unittest на Python {тренинг} (2ч.)[1,3,6,8,10]

7. Тестирование сайтов {тренинг} (2ч.)[1,3,6,7,8,10]

8. Тестирование баз данных(2ч.)[1,3,6,7,8,10]

Самостоятельная работа (76ч.)

1. Изучение теоретического материала с целью формирования знаний о методах тестирования для оценки работоспособности и эффективности программного обеспечения(24ч.)[1,2,3,4,5,6,7,8] Изучение теоретического материала (работа с конспектом лекций, первоисточниками основной и дополнительной литературы, учебными пособиями)

2. Выполнение курсовой работы для формирования умения применять методы тестирования для оценки работоспособности и эффективности программного обеспечения и анализировать результаты тестирования(20ч.)[1,2,3,4,5,6,7,8,9,10] Проведение теоретического исследования, выполнение практических заданий. Составление отчета о работе

3. Подготовка к лабораторным работам для формирования умения применять методы тестирования для оценки работоспособности и эффективности программного обеспечения и анализировать результаты тестирования(24ч.)[2,3,4,5,6,7,8] Изучение теоретического материала, выполнение практических заданий, связанных с разработкой и реализацией тестов для оценки работоспособности и эффективности программного обеспечения. Составление отчетов о выполнении лабораторных работ

4. Подготовка к зачету(8ч.)[1,2,3,4,5,6,7,8] Повторение теоретического и практического материала

Содержание дисциплины для заочной формы обучения

Лекционные занятия (6ч.)

1. Документация, сопровождающая процесс верификации и тестирования {беседа} (2ч.)[1,2,4,5] Роль тестирования в разработке программного обеспечения, сопровождении и функционировании программного обеспечения. Роль тестировщика в команде, его задачи и зона ответственности. Технологические процессы верификации в проекте, документация, создаваемая в ходе жизненного цикла проекта, ее назначение. Стратегия и планы верификации. Отчеты о прохождении тестов.

2. Модульное и системное тестирование. Документирование проекта {лекция с разбором конкретных ситуаций} (2ч.)[1,2,4,5] Задачи и цели модульного тестирования. Понятие модуля и его границ. Тестирование классов. Подходы к проектированию тестового окружения. Организация модульного тестирования. Виды системного тестирования. Системное тестирование, приемо-сдаточные и сертификационные испытания при разработке сертифицируемого программного обеспечения.

3. Функциональное автоматизированное тестирование. Тестирование сайтов и баз данных {лекция с разбором конкретных ситуаций} (2ч.)[1,2,5,6,8,10] Системы автоматизированного функционального тестирования. Подход Apple к автоматизации. Тестирование Web-приложений с помощью Selenium. Использование модуля unittest для Python. Тестирование домашней страницы при помощи модульных тестов. Взаимодействие с сайтом. Скрапинг с помощью удаленных серверов. Подключение формы для отправки POST-запроса. Сохранение POST-запроса в базу данных.

Лабораторные работы (8ч.)

1. Разработка и составление сопроводительной документации для проведения тестирования {тренинг} (2ч.)[1,2,3,9,10]

2. Методы тестирования ("черного ящика" и "белого ящика") {тренинг} (2ч.)[1,2,3,7,9,10]

3. Планирование и оценка тестирования. Анализ результатов тестирования {тренинг} (2ч.)[1,2,3,9,10]

4. Тестирование с использованием модуля unittest на Python для тестирования сайтов {тренинг} (2ч.)[1,3,6,8,10]

Самостоятельная работа (94ч.)

1. Изучение теоретического материала с целью формирования знаний о методах тестирования для оценки работоспособности и эффективности программного обеспечения(50ч.)[1,2,3,4,5,6,7,8] Изучение теоретического материала (работа с конспектом лекций, первоисточниками основной и дополнительной литературы, учебными пособиями)

2. Выполнение курсовой работы для формирования умения применять методы тестирования для оценки работоспособности и эффективности программного обеспечения и анализировать результаты тестирования(24ч.)[1,2,3,4,5,6,7,8,9,10] Проведение теоретического исследования, выполнение практических заданий. Составление отчета о работе

3. Подготовка к лабораторным работам для формирования умения применять методы тестирования для оценки работоспособности и эффективности программного обеспечения и анализировать результаты тестирования(16ч.)[2,3,4,5,6,7,8] Изучение теоретического материала, выполнение практических заданий, связанных с разработкой и реализацией тестов для оценки работоспособности и эффективности программного обеспечения. Составление отчетов о выполнении лабораторных работ

4. Подготовка к зачету(4ч.)[1,2,3,4,5,6,7,8] Повторение теоретического и практического материала

Тесты для самоподготовки к экзамену по дисциплине

1. Ошибка (определение)
   - ✓ Неправильное поведение программы
   - ✓ Неожидаемое поведение программы
   - ✓ Несоответствие поведения программы спецификации
   - ✓ Несоответствие поведения программы ее исходному коду на языке программирования
   - ✓ Поведение программы, при котором возникают неожиданные окна с исключениями
2. Спецификация
   - ✓ Определяет все возможные ошибки в программе
   - ✓ Определяет требования заказчика к языку программирования, на котором будет реализована система
   - ✓ Определяет требования программиста к разрабатываемой системе и возможные ошибки
   - ✓ Специфицирует только поведение программы по BDD
   - ✓ Определяет требования к разрабатываемой системе от заказчика
3. Тестирование ПО
   - ✓ Часть процесса разработки, который выполняется на последнем шаге разработки и в котором выявляются ошибки как недочеты работы программистов
   - ✓ Часть процесса разработки, в котором проверяются соответствие программы ее спецификации и выявляются ошибки
   - ✓ Процесс доказательства правильности программы
   - ✓ Вместе с верификацией служит для доказательства правильности программы на всех возможных тестах математическими и программными методами
4. Проблема тестирования
   - ✓ Необходимо доказать наличие тестировщиков при тестировании продукта и их оптимальное количество

- ✓ Нельзя доказать правильность программы тестированием на всех случая
- ✓ Нельзя доказать правильность программы с верификацией на всех случаях
- ✓ Другое название проблемы остановки, но это разрешимая проблема с помощью тестируемой верификации

5. Как доказать проблему тестирования?
   - ✓ Сведением другой известной неразрешимой проблемы в матлогике к этой
   - ✓ Проверифицировать данную проблему
   - ✓ Сведением к другой известной неразрешимой проблеме в матлогике
   - ✓ Создать спецификацию на BDD и проверифицировать, далее свести контрпример к ошибке
   - ✓ С использованием методов темпоральной логики
6. Знаменитая фраза Дейкстры про тестирование
   - ✓ Тестирование программ может использоваться для демонстрации наличия ошибок, но оно никогда не покажет их отсутствие
   - ✓ Тестирование программ может использоваться для доказательства отсутствия ошибок, но оно никогда не покажет их наличие
   - ✓ Тестирование программ может использоваться для демонстрации ключевых ошибок, оно всегда покажет их наличие
   - ✓ Верификация программ может использоваться для демонстрации наличия ошибок, но она никогда не покажет их отсутствие
   - ✓ Верификация программ может использоваться для доказательства отсутствия ошибок, но она никогда не покажет их наличие
7. Верификация программного обеспечения
   - ✓ Процесс доказательства корректности любой программы
   - ✓ Процесс доказательства корректности формальной программы
   - ✓ Процесс опровержения корректности специально созданной модели программы с помощью темпоральной логики
   - ✓ Процесс доказательства корректности специально любой программы с помощью динамических и статических методов
   - ✓ Процесс доказательства корректности специально созданной модели программы
8. Тестировщик программного обеспечения
   - ✓ Проверяет программы, ищет ошибки, вносит ошибки, верифицирует модели самостоятельно с целью получения прибыли
   - ✓ Проверяет программы, ищет ошибки, осуществляет диалог с разработчиками и другими членами команды с целью их устранения и выпуска программного обеспечения хорошего качества
   - ✓ Ставит задачи программистам на разработку, рисует спецификации, проверяет программы, ищет ошибки, осуществляет диалог с разработчиками и другими членами команды с целью их устранения и выпуска программного обеспечения любого качества
   - ✓ Проверяет программы, ищет ошибки, далее, по мере карьерного роста, доказывает правильность программ методами математической логики
9. Тестирование черного ящика

- ✓ Когда для тестирования системы доступен ее исходный код частично, например, некоторых компонентов или интерфейсы классов
- ✓ Когда исходный код системы недоступен и производится проверка соответствия системы формальной спецификации или ожидаемому поведению
- ✓ Когда для тестирования системы доступен ее исходный код

10. Два критерия для спецификации
    - ✓ Полнота и понятность
    - ✓ Простота и понятность
    - ✓ Полнота и сложность
    - ✓ Тестирование и верификация
    - ✓ Доброта и полнота
11. Предусловие в спецификации
    - ✓ Результат выполнения действия, т.е. что получается в системе после того как действие сработает, и чего нам надо ожидать от действия
    - ✓ Те условия, при которых возможно возникновение описываемого действия в системе
    - ✓ Глобальные условия в системе, которые постоянно выполняются на протяжении либо всей программы, либо заданного момента времени
12. Постусловие в спецификации
    - ✓ те условия, при которых возможно возникновение описываемого действия в системе
    - ✓ результат выполнения действия, т. е. что получается в системе после того, как действие сработает, и чего нам надо ожидать от действия
    - ✓ глобальные условия в системе, которые постоянно выполняются на протяжении либо всей программы, либо заданного момента времени
13. Инвариант в спецификации
    - ✓ Глобальные условия в системе, которые постоянно выполняются на протяжении либо всей программы, либо заданного момента времени
    - ✓ Те условия, при которых возможно возникновение описываемого действия в системе
    - ✓ Результат выполнения действия, т. е. что получается в системе после того, как действие сработает, и чего нам надо ожидать от действия
14. Ошибка в спецификации
    - ✓ Полное или непротиворечивое описание работы программного обеспечения
    - ✓ Несоответствие требованиям заказчика
    - ✓ Неполное или противоречивое описание работы программного обеспечения
    - ✓ Соответствие требованиям заказчика
15. Тройка Хоара
    - ✓ {Q} C {I}, где I – инвариант, C – команда, Q – постусловие
    - ✓ {P} C {Q}, где P – предусловие, C – команда, Q – постусловие
    - ✓ C {P} {Q}, где P – предусловие, C – команда, Q – постусловие
    - ✓ {P} I {Q}, где P – предусловие, I – инвариант, Q – постусловие
16. Правило 4W+1H работы хорошего тестировщика

- ✓ When (когда найден баг или при каких условиях он происходит)
  Where (где найден баг)
  Who (кто виновен или ответственен)
  Why (почему случается данный баг – предположение)
  и How (что, по мнению тестировщика, нужно сделать, чтобы исправить данный баг)
- ✓ When (когда найден баг или при каких условиях он происходит)
  Where (где найден баг)
  what (что теперь с ним делать)
  Why (почему случается данный баг – предположение)
  и How (что, по мнению тестировщика, нужно сделать, чтобы найти данный баг)

17. Багтрекинговая система
    - ✓ Предназначена для поиска информации об ошибках в проекте и хранения их в едином месте
    - ✓ Предназначена для занесения информации об ошибках в выбранном проекте и хранения их в едином месте, а также для изменения статусов ошибок и проведения автоматический верификации проектов
    - ✓ Предназначена для занесения информации об ошибках в выбранном проекте и хранения их в едином месте, а также для изменения статусов ошибок, ревью кода, коммитов и проект-менеджмента
    - ✓ Предназначена для занесения информации об ошибках в выбранном проекте и хранения их в едином месте, а также для изменения статусов ошибок
18. Модульное тестирование
    - ✓ Процесс разработки
    - ✓ Процесс тестирования
    - ✓ Процесс верификации
    - ✓ Процесс развертывания
19. Unit testing
    - ✓ Это процесс тестирования, при котором тестировщики программного обеспечения одновременно с написанием кода структурных единиц проекта (модулей, классов, функций и т. д.) осуществляют написание тестов для проверки их работы как соответствие предопределенным предположениям о работе кода этих структурных единиц
    - ✓ Это процесс разработки, при котором разработчики программного обеспечения одновременно с написанием кода структурных единиц проекта (модулей, классов, функций и т. д.) осуществляют написание тестов для проверки их работы как соответствие предопределенным предположениям о работе кода этих структурных единиц
    - ✓ Это процесс верификации, при котором тестировщики программного обеспечения одновременно с написанием кода структурных единиц проекта (модулей, классов, функций и т. д.) осуществляют написание тестов и верифицирующих моделей для проверки их работы как соответствие предопределенным предположениям о работе кода этих структурных единиц

- ✓ Это процесс поиска багов в юнит-тестах, который осуществляется одновременно с разработкой модулей

20. Кто пишет модульные тесты
    - ✓ Технический писатель
    - ✓ Разработчик
    - ✓ Тестировщик
    - ✓ Инженер по качеству
21. Для улучшения качества, кто должен писать модульные тесты
    - ✓ Тот же самый разработчик, что и автор кода
    - ✓ Другой разработчик
    - ✓ Тестировщик
    - ✓ Тестировщик другого продукта
22. Doxygen
    - ✓ Генерирует тесты по описаниям в коде
    - ✓ Генерирует документацию и тесты по описаниям в коде
    - ✓ Генерирует документацию автоматически путем анализа программ без необходимости описания
    - ✓ Генерирует документацию по описаниям в коде
23. Один модульный тест
    - ✓ Покрывает один метод хорошо
    - ✓ Одного теста для метода явно недостаточно
    - ✓ Покрывает весь модуль
    - ✓ Покрывает класс
24. xUnit
    - ✓ Фреймворки модульного тестирования
    - ✓ Фреймворки модульного документирования
    - ✓ Фреймворки юнит-верификации
    - ✓ Фреймворки разработки через верификацию юнитов
25. MDD
    - ✓ Module driven development
    - ✓ Model driven development
    - ✓ Magnifique driven debugging
    - ✓ Model driven donate
26. TDD
    - ✓ Test driven development
    - ✓ Test driven debugging
    - ✓ Turbo driven development
    - ✓ Tradition driven debugging
27. BDD
    - ✓ Bug driven development
    - ✓ Behavior driven development
    - ✓ Behavior driven debugging
    - ✓ Brother debugging development
28. Scenario:

Given I have my software calculator
When I have entered 5 as first operand
And I have entered 10 as second operand
And I press 'Add'
Then The result should be

- ✓ BDD
- ✓ TDD
- ✓ MDD
- ✓ ADD

29. public double add(double a, double b) throws IllegalArgumentException {
//todo реализовать метод позднее до конца
if (a == 2 && b == 2) return 4;
throw new IllegalArgumentException("add() works only for 2+2");}

- ✓ Unittest
- ✓ BDD
- ✓ TDD
- ✓ Код с ошибкой

30. Системы автоматизированного функционального тестирования

- ✓ Они записывают, как работает тестировщик с тестируемой системой, генерируют тесты на разных наборах исходных данных
- ✓ Они автоматически записывают, как работает программист с тестируемой системой, генерируют тест на некотором языке программирования, далее этот записанный тест можно юнит-тестировать автоматически и на всех наборах исходных данных
- ✓ Они записывают, как работает тестировщик с тестируемой системой, генерируют тест на некотором языке, далее этот записанный тест можно воспроизвести автоматически и на разных наборах исходных данных
- ✓ Служат для замены тестировщиков и программистов

31. Selenium IDE

- ✓ Плагин для браузера для записи и воспроизведения юнит-тестов
- ✓ Среда разработки для записи и воспроизведения функциональных тестов на Java SE
- ✓ Среда разработки для записи и воспроизведения функциональных тестов для Eclipse IDE
- ✓ Плагин для браузера для записи и воспроизведения функциональных тестов

32. IBM Rational Tester

- ✓ Средство модульного тестирования
- ✓ Средство разработки по TDD
- ✓ Средство стресс-тестирования
- ✓ Средство функционального тестирования

33. Selenium WebDriver

- ✓ Плагин для браузера для записи и воспроизведения функциональных тестов
- ✓ Плагин для браузера для записи и воспроизведения юнит-тестов
- ✓ Среда разработки для записи и воспроизведения функциональных тестов на

Java
- ✓ Библиотека для управления браузерами для проведения функциональных тестов

34. Spin лучше всего подходит
- ✓ Для верификации протоколов
- ✓ Для доказательства алгоритмов
- ✓ Для генерации всех тестов
- ✓ Для статической проверки
- ✓ Для динамической проверки

35. Code contracts лучше всего подходит
- ✓ Для верификации протоколов
- ✓ Для доказательства алгоритмов
- ✓ Для генерации всех тестов
- ✓ Для статической проверки
- ✓ Для динамической проверки

36. Spec Explorer лучше всего подходит
- ✓ Для верификации протоколов
- ✓ Для доказательства алгоритмов
- ✓ Для генерации всех тестов
- ✓ Для статической проверки
- ✓ Для динамической проверки

37. Valgrind лучше всего подходит
- ✓ Для верификации протоколов
- ✓ Для доказательства алгоритмов
- ✓ Для генерации всех тестов
- ✓ Для статической проверки
- ✓ Для динамической проверки

38. Cppcheck лучше всего подходит
- ✓ Для верификации протоколов
- ✓ Для доказательства алгоритмов
- ✓ Для генерации всех тестов
- ✓ Для статической проверки
- ✓ Для динамической проверки

39. LTL
- ✓ Линейная логика тестирования
- ✓ Локальная тестовая логика
- ✓ Линейная логика темпорального времени
- ✓ Локальная логика темпорального времени
- ✓ Темпоральная логика линейного времени

40. Какую логику использует SPIN?
- ✓ LTL
- ✓ Математическую S-логику
- ✓ CTL
- ✓ Предикатную логику высшего порядка

41. Оператор “когда-нибудь” на LTL в Promela
- ✓ U
- ✓ <>
- ✓ []
- ✓ ->

42. LTL для “Всегда, если A, то когда-то всегда B” в Promela
- ✓ “[] (A -> <>[] B)”
- ✓ “<> (A -> []<> B)”
- ✓ “W (A -> W U B)”
- ✓ “<> (A U []<> B)”

43. Инвариант цикла на переменную цикла i, пробегающую [0, size).
- ✓ 0<=i<size
- ✓ 0<=i<=size
- ✓ size-i

44. A || B в Cord-скрипте в Spec Explorer означает
- ✓ Либо правило A, либо правило B
- ✓ Последовательность A, за ним из всех заключительных состояний B
- ✓ Параллельная композиция автоматов, заданных A и B
- ✓ Инструкция создания тестов либо для A, либо для B

45. Решение какой задачи может построить алгоритм эффективной генерации тестов?
- ✓ Задача кенинбергских мостов
- ✓ Задача китайского почтальона
- ✓ Задача нью-йоркского чистильщика улиц
- ✓ Задача спящего парикмахера

46. Преимущества тестирования в Eiffel
- ✓ Модульное тестирование, TDD
- ✓ Генерация тестов по модели
- ✓ Autotest, тесты из отладчика

47. ACSL
- ✓ Стандартизированный язык описания спецификаций на код для статических проверок
- ✓ Стандартизированный язык описания спецификаций на код для динамических проверок
- ✓ Стандартизированный язык описания спецификаций LTL-требований для C программ
- ✓ Стандартизированные средства для преобразования C-кода в Promela с целью верификации

Задания для курсовой работы (разработка через тестирование)

1. Реализовать приложение с нуля (согласно заданию) с использованием методологии TDD или BDD по выбору студента.
2. Каждый промежуточный шаг разработки (рабочий код+тесты) коммитить в систему контроля версий.

3. Сложный графический интерфейс необязателен. Вся функциональность должна быть реализована в классах, допустимо использование консольного интерфейса, где явно не нужен графический.
4. При сдаче преподавателю показывать функционал приложения, далее код, так, чтобы были видны diff (различия по строкам) между коммитами, начиная с первого и можно было отследить последовательность разработки (в средстве git или на GitHub).
Задания разной сложности, выбираются индивидуально преподавателем, исходя из уровня студентов.
Варианты
1. Игра “Поле чудес” (угадать загаданное слово).
2. Игра ”Города” (база городов, игроки по очереди вводят город за отведенное время, проверяется на правильность).
3. Игра ”Крестики-нолики”.
4. Научный калькулятор (решение уравнений, матричные операции, интегрирование и т. д.).
5. Матричный калькулятор (операции с матрицами, в т. ч. с разряженными).
6. Программа-будильник (заданная мелодия в заданное время).
7. Конвертер различных величин (американская система, старорусская, СИ).
8. Бесследный уничтожитель файлов и папок (забивать файл несколько раз заданными символами, потом уничтожать).
9. Электронный терапевт (есть некая база, пользователю задаются вопросы и ставится диагноз).
10. Генератор паттернов (генерация исходного кода по заданному паттерну проектирования с предпросмотром).
11. Игра “Как стать милиционером”.
12. Игра “Угадай мелодию” (3 игрока, по очереди проигрывается мелодия и спрашивается).
13. Конвертер валют с возможностью предсказания курсов (несколько валют и история курса, можно узнать курс на след. дни экстраполяцией).
14. Приложение – хранитель паролей (сайты, кредитные карты), доступ в виде игры.
15. Шифратор и дешифратор файлов в заданной папке (шифруем все файлы и расшифровываем несколькими методами, в т.ч. несколькими).
16. Игра “Морской бой” .
17. Игра “Пятнашки” .
18. Игра “512” (аналог “2048”).
19. Игра в тестирование программ (2 игрока). Дается код с ошибками и необходимо все найти.
20. Программа для конвертации графических файлов (размеры + разные форматы).
21. Игра в дурака (2 игрока).
22. Приложение для создания миксов мелодий из нескольких файлов (выбираем файлы, начало и конец для каждого и получаем итоговый микс).
23. Игра “Угадай паттерн” (игрокам предоставляется код или UML-схема и надо

угадать паттерн в виде игры).
24. Мини-социальная сеть (логинка, добавление в друзья с заданным аккаунтом, посылка сообщения другу).
25. Генератор мелодий по заданным аккордам.
26. Программа для напоминания о днях рождения друзей.
27. Игра “Упрощенный футбольный симулятор”.
28. Программа – консольный друг (диалог с компьютером на разные темы).

Список использованной литературы

1. Старолетов С.М. Основы тестирования и верификации программного обеспечения. Учебное пособие. Барнаул: АлтГТУ, 2020. – 336с. Прямая ссылка: http://elib.altstu.ru/eum/download/pm/Staroletov_OsnTestVerifPO_up.pdf

2. Синицын, С. В. Верификация программного обеспечения : учебное пособие / С. В. Синицын, Н. Ю. Налютин. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 367 c. — ISBN 978-5-4497-0653-9. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/97540.html (дата обращения: 10.01.2021). — Режим доступа: для авторизир. пользователей

3. Котляров, В. П. Основы тестирования программного обеспечения / В. П. Котляров. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 334 c. — ISBN 5-94774-406-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/62820.html (дата обращения: 10.01.2021). — Режим доступа: для авторизир. пользователей

4. Широков, А. И. Стандартизация, сертификация и оценка качества программного обеспечения : учебное пособие / А. И. Широков, Е. П. Потоцкий. — Москва : Издательский Дом МИСиС, 2013. — 208 c. — ISBN 978-5-87623-722-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/98891.html (дата обращения: 10.01.2021). — Режим доступа: для авторизир. пользователей

5. Липаев, В. В. Тестирование компонентов и комплексов программ : учебник / В. В. Липаев. — Москва : СИНТЕГ, 2010. — 393 c. — ISBN 978-5-89638-115-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/27301.html (дата обращения: 10.01.2021). — Режим доступа: для авторизир. пользователей

6. Митчелл, Р. Скрапинг веб-сайтов с помощью Python : руководство / Р. Митчелл ; перевод с английского А. В. Груздев. — Москва : ДМК Пресс, 2016. — 280 с. — ISBN 978-5-97060-223-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100903 (дата обращения: 10.01.2021). — Режим доступа: для авториз. пользователей.

7. Ошероув, Р. Искусство автономного тестирования с примерами на C# / Р. Ошероув. — 2-е изд. — Москва : ДМК Пресс, 2014. — 360 с. — ISBN 978-5-94074-945-5. — Текст : электронный // Лань : электронно-библиотечная

система. — URL: https://e.lanbook.com/book/90106 (дата обращения: 10.01.2021). — Режим доступа: для авториз. пользователей.

8. Персиваль, Г. Python. Разработка на основе тестирования. Повинуйся Билли-тестировщику, используя Django, Selenium и JavaScript / Г. Персиваль ; перевод с английского А. В. Логунов. — Москва : ДМК Пресс, 2018. — 622 с. — ISBN 978-5-97060-594-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/111440 (дата обращения: 10.01.2021). — Режим доступа: для авториз. пользователей.

9. Электронно-библиотечная система «Университетская библиотека Online» [Электронный ресурс]. – М.: Издательство «Директ-Медиа». – Режим доступа: http://www.biblioclub.ru

10. Институт информационных технологий [режим доступа] www.intuit.ru