Национальный исследовательский ядерный университет «МИФИ»

**Кафедра “Кибернетика”(22)**

“ Методология программной инженерии (технология разработки, верификация и сертификация программного обеспечения)”

**План верификации**

**Средство обработки аудиовизуальной информации**

*Руководитель:* Иващенко М.Г.

*Группа:* Девяткина А.

Карх Е.

Кузнецова К.

Аринчёхина Ю.

Бригинец А.

Буравенкова Я.

Пономарев Е.

Рогачев А.

Юдин Н.

Москва

2019 г.

# **История изменений**

|  |  |  |  |
| --- | --- | --- | --- |
| Дата изменений | Версия документа | Описание внесенных изменений | Автор изменений |
| 26.10.2019 | 1 | Определена структура документа | Аринчёхина |
| 21.12.2019 | 2 | Заполнены пункты 2.2, 2.4, 2.5, 3.4, 3.5, 4 | Кузнецова |
| 21.12.2019 | 3 | Определена структура и частично заполнен п. 3.2. | Карх |
| 21.12.2019 | 4 | Заполнен п. 3.2. | Аринчёхина |

[**Введение**](#_3acbnydsc1o7) **4**

[Цель](#_lak8jrw8c0ro) 4

[Области применения](#_vqpcv7wkl2fh) 4

[Ссылки](#_1ldedyfd1s1o) 4

[Термины, определения и соглашения](#_yvxg99i98dk3) 4

[**Среда верификации**](#_8lx2za6kab4j) **4**

[Связь с другими процессами жизненного цикла ПО](#_x4etbgagdfgh) 4

[Организация и ответственности](#_1q64e2rmwpe5) 4

[Независимость](#_x5lelywa383z) 4

[Инструменты верификации](#_qxyobw4yuiw5) 4

[Методы верификации](#_ikzfcnjobl16) 4

[**Мероприятия процесса верификации**](#_o4eyeuondmuy) **4**

[Формальная инспекция](#_qe6chdspriqn) 4

[Разработка тестовых примеров](#_gt3122sygc3c) 4

[Испытания ПО](#_9lc1drgg5403) 4

[Анализ трассировки](#_v9345dtbp5ba) 4

[Анализ структурного покрытия](#_8zhf8hsdg470) 4

[**Критерии перехода**](#_r1f4nxhz9c0o) **4**

[**Инструкции повторной верификации**](#_c3ybk74ixvfw) **4**

[**Дополнительные соображения**](#_tip50w4mokem) **4**

# **Введение**

## **Цель**

Данный документ определяет порядок и описание мероприятий, которые будут выполняться в рамках курса “Методология программной инженерии” (МПИ) для достижения целей процесса верификации ПО.

## **Области применения**

Требования и положения настоящего Плана распространяются на все работы по верификации, выполняемые в рамках курса МПИ, и обязательны для всех участников курса.

Положения, описанные в данном документе, применимы исключительно при выполнении проекта в рамках данного курса.

## **Ссылки**

[1] План управления конфигурацией ПО

[2] План контроля качества

[3] Техническое задание

[4] Технический регламент

## **Термины, определения и соглашения**

Таблица 1. Термины, определения и соглашения

|  |  |
| --- | --- |
| Термин | Определение, толкование |
| Сообщение о проблеме | Документ, содержащий описание несоответствия в данных ЖЦ ПО или в описании процессов ЖЦ ПО. |
| Запрос на изменение | Документ, созданный для внесения изменений в данные ЖЦ ПО с целью устранения несоответствия, описанного в СП. |
| Артефакт | Атомарное данное ЖЦ ПО. Артефактом может быть, например, требование или файл исходного кода |
| Агрегация | Составное данное ЖЦ ПО, которое включает в себя однотипные артефакты и/или агрегации. |
| Конфигурация | Составное данное ЖЦ ПО, которое может включать в себя любые артефакты, агрегации и конфигурации |
| Категория запроса на изменение | Атрибут запроса на изменение, идентифицирующий процесс ЖЦ ПО, данные которого будут меняться в рамках этого запроса на изменение. |
| Формальная инспекция | Способ верификации документов, основанный на экспертной оценке их правильности, выполняемой одним или несколькими инспекторами, как правило, с использованием проверочных перечней. |
| Первичная инспекция | Формальная инспекция называется первичной, если она не является повторной. |
| Повторная инспекция | Формальная инспекция называется повторной, если версия объекта инспекции создана с целью устранения замечаний предыдущей инспекции. |

# **Среда верификации**

## **Связь с другими процессами жизненного цикла ПО**

Процесс верификации взаимодействует с другими процессами жизненного цикла ПО посредством процесса управления конфигурацией ПО, получая из него все входные данные и возвращая в него все выходные данные жизненного цикла ПО.

Входными данными процесса верификации являются следующие выходные данные процессов ЖЦ ПО:

* Планы проекта;
* Требования к ПО;
* Описание проекта ПО;
* Исходный код;
* Исполняемый объектный код;

Выходными данными процесса верификации являются:

* Сообщения о проблемах;
* Тестовые примеры и процедуры;
* Результаты прогона тестовых примеров (результаты испытаний);
* Записи о ФИ;
* Результаты анализов.

Выходные данные процесса верификации ПО передаются как входные данные в соответствующие процессы ЖЦ ПО с целью устранения выявленных несоответствий.

## **Организация и ответственности**

Стадии верификации

1.Инициализация

* автор размещает созданные объекты в соответствии с планом конфигурации
* назначаются ведущий (не обязательно) и инспектор
* выбирается способ проведения экспертизы
* сообщение от ведущего, что можно начинать (если он не назначался, то сообщение от автора)

2.Проведение экспертизы

* поиск отклонения
* если объект верифицируется повторно, то явно отмечается, исправлено оно или нет

3.Обсуждение

* уточнение списков несоответствия

4.Привлечение к обсуждению автора (возможно, гарантии качества)

5.Регистрация результатов

* ведущий (или какой-либо участник верификации конкретной единицы) фиксирует договорённости
* создаются запросы на изменения

## **Независимость**

Независимость при выполнении мероприятий верификации обеспечивается на персональном уровне, то есть для каждого верифицируемого объекта из множества данных ЖЦ ПО сотрудник проекта, выполняющий мероприятие по верификации, не участвует в создании данного объекта.

## **Инструменты верификации**

Инструментальные средства, используемые для разработки ПО, применяются для проведения его испытаний. Для анализа структурного покрытия применяется библиотека coverage.py версии 5.0, с помощью которой установить процент задействованных при тестировании строк кода.

## **Методы верификации**

Методами верификации являются формальная инспекция, анализ и испытания ПО. Под формальной инспекцие подразумевается субъективное рассмотрение единицы данных ЖЦ ПО. Анализ подразумевает применение конкретных инструментов и алгоритмов, его результат не зависит от участника, проводящего анализ. Испытания ПО проводятся с целью продемонстрировать, что ПО удовлетворяет предъявляемым к нему

требованиям, а также то, что устранены ошибки, выявленные на каком-либо этапе верификации.

# **Мероприятия процесса верификации**

## **Формальная инспекция**

Цель применения формальной инспекции – предотвратить и исключить ошибки в требованиях, а также проекте и коде при выполнении процессов разработки ПО. Таким образом, она дополняет испытания.

Формальная инспекция в рамках проекта представляет собой субъективное рассмотрение единиц ЖЦ для оценки соответствия объекта требованиям, предъявленным к нему. Процедура начинается с сообщения автора объекта о его готовности к рассмотрению. Любой участник команды разработки (контроля качества, если требуется) может провести формальную инспекцию и выдвинуть замечания, которые подаются на вход процесса управления конфигурацией и обрабатываются согласно процедуре, описанной в плане управления конфигурацией.

## **Разработка тестовых примеров**

Тестовые примеры должны быть разработаны для каждого метода, представленного в разрабатываемой библиотеке.

Для каждого из методов необходимо предусмотреть хотя бы два примера: изображение маленького и большого размеров. Помимо этого необходимо предусмотреть примеры, исходя из особенностей используемых в методе алгоритмов.

3.2.1. Методы Read, Show

Методы для чтения и вывода изображения на экран предполагают, что при отсутствии последующей обработки метод Show возвращает исходное изображение, иначе - результат работы последнего вызванного метода. Поскольку эти два метода являются частью ядра библиотеки, и их вызовы предполагаются при проверке всех прочих методов, отдельного тестирования для них не требуется.

3.2.2. Метод Invert

Метод для получения инвертированного изображения не предполагает использования конкретно описанного алгоритма, поэтому в рамках разработки тестовых примеров следует ограничиться однократной проверкой на черно-белом изображении.

3.2.3. Метод Grayscale

Метод для получения изображения в оттенках серого не зависит от каких-либо параметров и одинаково работает на любых изображениях. Поэтому при тестировании достаточно провести однократную проверку корректности его работы на многоцветном изображении.

3.2.4. Метод Save

Метод сохранения должен учитывать возможность ввода некорректного имени файла с точки зрения файловой системы. Таким образом, при тестировании необходимо проверить его работу на корректном и некорректном имени.

3.2.5. Методы бинаризации (метод Эйквила, метод Кристиана)

Метод Эйквила в качестве параметров принимает размеры малого и большого окна. Предполагается, что при передаче одинакового значения в параметры, качество бинаризации ухудшится. Кроме того, ожидается что при размытом входном изображении результат бинаризации будет содержать больше шума, чем при четком.

Метод Кристиана был создан для преодоления проблемы при бинаризации изображений с тонкими, близко расположенными друг к другу линиями. Ожидается, что он будет лучше работать для изображений с текстом, поэтому предлагается проверка всех методов на изображениях с текстом, изображениях с фотографиями вывода электронных приборов и т.д..

3.2.6. Методы выделения контура (метод с использованием оператора Собеля, метод с использованием оператора Прюитт)

Для методов выделения контура на изображении необходимо предусмотреть проверку на изображениях с различной детализацией.

Для метода с использованием оператора Собеля одним из аргументов является порог бинаризации (целое число от 0 до 255). Для каждого из проверяемых изображений производится проверка с несколькими пороговыми значениями. При увеличении порога ожидается, что выделенный контур будет становиться четче, разрывы контура будут уменьшаться, но увеличатся помехи в виде выделения контуров текстур, помех и т.д..

Так как оператор Прюитт был разработан для применения в медицинских исследованиях, предлагается проверка обоих методов на наборе медицинских изображений. При этом ожидается, что метод с использованием оператора Прюитт должен работать более эффективно, чем метод с использованием оператора Собеля.

3.2.7. Методы фильтрации

В контексте разрабатываемой библиотеки под процедурой “фильтрации” подразумевается помеховая фильтрация или фильтрация изображений от “шума”. Задача помеховой фильтрации, таким образом, сводится к тому, чтобы путем некоторой обработки наблюдаемого реального изображения как можно лучше "очистить его от шума", то есть получить изображение, наиболее близкое по своим характеристикам к исходному "не зашумленному" изображению. В рамках разрабатываемой библиотеки реализованы следующие нелинейные методы помеховой фильтрации:

* Медианный фильтр. Эффективен для случаев, когда вероятности шума “соль и перец” удовлетворяют соотношению p=q<0,5. Что означает - соотношение шума “соль” и “перец” должно быть примерно равнозначно, но в сумме вероятность зашумления изображения не должна превышать 0,5. Кроме того, увеличение размера апертуры окна медианного фильтра позволяет увеличить эффективность шумоподавление, однако приводит к искажению границ и очертаний объектов.

В качестве тестовых примеров на вход медианного фильтра предлагается:

1. Подать на вход медианного фильтра изображение с вероятностью зашумления < 0,5 при условии равнозначности шумов “соль” и “перец”. Убедиться, что единичные пиксели “соли” и “перца” были отфильтрованы, получив разностную матрицу для исходного и отфильтрованного изображений.
2. Подать на вход медианного фильтра изображения сильно зашумленное изображение (>0,5). Провести тестирование с использованием разной апертуры окна фильтра: 3x3, 5х5, 7х7. Убедиться, что увеличение апертуры окна приводит к увеличению шумоподавления и искажению очертаний, получив разностные матрицы для каждого случая.

* Взвешенный ранговый фильтр. Данный алгоритм является модификацией стандартной ранговой фильтрации путем использования взвешенной апертуры. Придавая точкам окрестности фильтрации больший вес, достигается повышение устойчивости фильтрации. В качестве апертуры окна алгоритм использует апертуру типа “штраф за удаление” размерности 3х3.

В качестве тестовых примеров на вход взвешенного рангового фильтра предлагается:

1. Изображение с высокой степенью зашумления “перцем”. Провести тестирование, подавая на вход алгоритма различные показатели ранга - больше значения медианы и меньше значения медианы (медиана = 8). Для изображений с высокой степенью зашумления “перцем” ожидается получить лучшие результаты в случае использования ранга со значением, больше медианного.
2. Изображение с высокой степенью зашумления “солью”. Провести тестирование, подавая на вход алгоритма различные показатели ранга - больше значения медианы и меньше значения медианы (медиана = 8). Для изображений с высокой степенью зашумления “солью” ожидается получить лучшие результаты в случае использования ранга со значением, меньше медианного.

Для сравнения эффективности рангового фильтра и взвешенного рангового фильтра предлагается провести следующее испытание для одного и того же изображения с примерно равным соотношение шума “соль” и “перец”:

1. Подать на вход алгоритма в качестве апертуры окна единичную матрицу размерности 3х3 и значение ранга равное 5/9.
2. Подать на вход алгоритма в качестве апертуры окна взвешенную матрицу размерности 3х3 типа “штраф за удаление” и значение ранга равное 10/16.

Ожидаемым результатом является получение лучшего изображения с использованием взвешенного рангового фильтра.

3.2.8. Методы расчета профилей текста

В набор данных методов входят:

* Метод расчета вертикального и горизонтального профиля изображений с текстом. Предполагается, что рассчитанный горизонтальный профиль будет иметь соответствующий вид “расчески” с ярко выраженными максимумами и минимумами.
* Метод расчета нулей в профиле для нахождения минимумов профиля. Не предполагает особых проверок.
* Метод определения координат строк в тексте и букв в строке. Предполагается проверка на тексте с шрифтом без засечек и с шрифтом с засечками для проверки случаев, когда буквы в строке сливаются. Также предполагается проверка на тексте, выполненном курсивом и полужирным начертанием.
* Метод отрисовки сегментации текста на изображении. Не предполагает отдельной проверки, используется для тестирования метода сегментации текста.
* Метод сегментации текстовых изображений. В качестве параметра принимает порог, поэтому необходимо тестирование с различными значениями. Предполагается что при низком пороге символы будут “сливаться” - соседние символы будут выделяться как один. При тестировании с высоким значением порога допустимо разбиение символов на несколько частей.

3.2.9. Методы расчета символьных характеристик

Рассчитываются символьные характеристики для изображений разного размера с одинаковыми символами, выделенными с помощью методов сегментации. Ожидается получение близких значений.

3.2.10. Метод распознавания символов.

При распознавании символов должны быть получены лучшие гипотезы, допустимым является замена некоторых букв в распознаваемых словах без потери “формы” слова. Предполагается проверка на искусственно зашумленном изображении (с добавлением шума “соль и перец” либо размытием границ). Проверка на изображениях, полученных путем оцифровки документов, не производится.

## **Анализ трассировки**

Под трассируемостью понимается обеспечение доказательства взаимосвязи между опредлёнными единицами ЖЦ. Относительно трассируемости должны быть выполнены следующие аспекты:

* Обеспечение трассируемости между требованиями к системе и требованиями к ПО для того, чтобы дать возможность проверки полной реализации требований к системе и сделать обозримыми производные требования.
* Обеспечение трассируемости между требованиями низкого и высокого уровней для того, чтобы сделать обозримыми производные требования и архитектурные решения, принятые в процессе проектирования ПО, а также дать возможность проверки полной реализации требований высокого уровня.
* Следует обеспечить трассируемость между «Исходным кодом» и требованиями низкого уровня для того, чтобы сделать возможной проверку отсутствия незадокументированных частей в «Исходном коде» и проверку полной реализации требований низкого уровня.

Процесс верификации обеспечивает трассируемость между реализацией требований к ПО и верификацией этих требований:

• Трассируемость между требованиями к ПО и тестовыми примерами достигается анализом покрытия требований.

• Трассируемость между структурой программы и тестовыми примерами достигается анализом структурного покрытия.

На этапе анализа покрытия требований проводится анализ тестовых примеров, связанных с требованиями к ПО, с целью подтверждения, что выбранные тестовые примеры удовлетворяют установленным критериям, а именно подтверждают то, что ПО выполняет все предъявленные функциональные требования.

Способом анализа покрытия требований может служить составления таблицы выполненных тестовых примеров с сопоставлением с тем, какое требование, приведённое в ТЗ данный проверяет тестовый пример.

## **Анализ структурного покрытия**

Цель данного анализа – определить, какая структура кода (программы) не была

проверена тестовыми процедурами на основе требований. Тестовые примеры на основе требований могут не полностью проверять структуру программы, поэтому и выполняется анализ структурного покрытия и проводится дополнительная верификация для обеспечения структурного покрытия.

Инструкции следующие:

* Анализ должен подтвердить степень структурного покрытия
* Анализ структурного покрытия может быть выполнен по «Исходному коду»
* В результате данного анализа следует подтвердить связь между компонентами кода по данным и по управлению.

В результате анализа структурного покрытия может быть выявлена структура кода, не работавшая при проведении испытаний. В этом случае требуется выполнить дополнительные мероприятия процесса верификации. Такая неисполняемая структура может быть результатом:

* Недостатков в тестовых примерах на основе требований или процедур. Для обеспечения отсутствующего покрытия следует дополнить тестовые примеры или изменить тестовые процедуры. Может потребоваться пересмотр методов анализа покрытия на основе требований.
* Недостаточности требований к ПО. Такие требования следует уточнить, разработать дополнительные тестовые примеры и выполнить дополнительные тестовые процедуры.
* Наличия мертвого кода. Следует удалить такой код и оценить последствия и необходимость повторной верификации.
* Наличия отключенного кода. Если отключенный код не предназначен для выполнения ни при какой из конфигураций ПО, следует показать с помощью сочетания анализа и испытаний, что средства, позволяющие ненамеренно включить выполнение такого кода, защищены, изолированы или исключены. Если отключенный код выполняется только при определённых конфигурациях среды, целевого вычислителя, то следует установить конфигурацию, необходимую для нормального выполнения этого кода, и разработать дополнительные тестовые примеры и процедуры для обеспечения требуемого покрытия.

# **Критерии перехода**

Зафиксирована базовая версия единицы конфигурации, принято сообщение от автора объекта о том, что он готов быть верифицирован.

Процесс верификации завершается вместе с процессом разработки ПО.