# 1. Введение

## 1.1 Цель

Данный документ определяет мероприятия, которые будут выполняться в рамках проекта «Студенческий портал»

## 1.2 Область применения

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

Положения, описанные в данном документе, могут быть неприменимы к некоторым процессам, возникающим при выполнении проекта и должны быть пересмотрены с учётом требований.

## 1.3. Термины, определения и соглашения

*Таблица 1. Термины, определения и соглашения*

|  |  |
| --- | --- |
| Термин | Определение, толкование |
| Сообщение о проблеме | Документ процесса УК ПО, содержащий описание несоответствия в данных ЖЦ ПО или в описании процессов ЖЦ ПО. |
| Запрос на изменение | Документ процесса УК ПО, 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. |
| Спецификация программных требований | Совокупность всех требований, предъявляемых к программному обеспечению, и дополнительных сведений, необходимых для правильной интерпретации этих требований. Спецификация требований организована как агрегация в. |
| Формальная инспекция | Способ верификации документов, основанный на экспертной оценке их правильности, выполняемой одним или несколькими инспекторами, как правило, с использованием проверочных перечней. |
| Базовая версия | Утверждённая зарегистрированная конфигурация одной или более единиц конфигурации, которая в дальнейшем служит основой для последующей разработки и изменяется только в соответствии с процедурами управления изменениями. |
| Проблема | Тип сообщений в системе GitHub «Issue»с возможностью присвоения тегов, связки с версиями данных ЖЦ ПО, назначения ответственных разработчиков и комментирования изменений. |

# 2. Организация и ответственности

В мероприятиях процесса верификации ПО принимают участие члены группы, имеющие различные роли в процессах ЖЦ ПО. В соответствии с методологией ведения проекта никому из участников команды не присваиваются какие-либо четкие роли. Поэтому в верификации может участвовать любой из участвующих в процессе создания ПО, но не участвующих в процессе создания верифицируемой единицы.

# 3. Независимость

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

Меры для достижения независимости следующие:

* автор объекта формальной инспекции не может быть инспектором в данной инспекции;
* при проведении инспекции требований к ПО как целого инспектором не может быть ни один из авторов, участвовавших в разработке частей требований;
* автор кода, реализующего требования, не могут быть авторами тестовых примеров для верификации этих требований;
* автор кода не может быть авторами отчета об анализе структурного покрытия.

# 4. Методы верификации

Для проведения мероприятий верификации будут использоваться методы, описанные ниже. Порядок применения этих методов детально описан в разделе 3.

* Испытания – метод верификации, при котором на вход ПО подаются определенные в тестовых примерах входные воздействия, а результат работы испытуемого ПО, полученный для данных входных воздействий, сравнивается со значениями, ожидаемыми на основании требований. Тестовые примеры разрабатываются на основе требований к ПО и описания проекта ПО таким образом, чтобы обеспечить полное покрытие требований тестовыми примерами. В зависимости от совпадения результатов работы ПО с ожидаемыми значениями, для каждого отдельного тестового примера определяется результат “Прошёл/Не прошёл” (“Pass/Fail”). Если существуют тестовые примеры с результатом “Не прошел” (“Fail”), открываются запросы на изменения о проблемах.
* Испытания интеграции ПО и аппаратных средств - целью этого вида испытаний является гарантия того, что ПО в среде целевого вычислителя будет удовлетворять требованиям высокого уровня. Испытания проводятся путем прогона тестовых примеров на стенде отладки и испытаний.
* Испытания низкого уровня - целью этого вида испытаний является гарантия соответствия компонентов ПО предъявляемым к ним требованиям низкого уровня. Испытания проводятся на компьютерах разработчиков путем выполнения тестов, посредством использования внутреннего инструментария среды разработки Eclipse PDT. Метод испытаний заключается в создании различных входных условий и вызове функций тестируемого ПО с различными параметрами, и сравнении полученной реакции с реакцией, ожидаемой на основании требований низкого уровня.
* Анализ – метод верификации, при котором проводится документальное подтверждение корректности анализируемых данных жизненного цикла ПО. Отличительным свойством анализа является воспроизводимость, т.е. результаты анализа, проведенного по одной и той же процедуре для одних и тех же данных жизненного цикла ПО, но разными людьми, будут идентичными. Несоответствия, выявленные при проведении анализа, документируются в сообщениях о проблемах и таким образом передаются в процессы, результаты которых подвергаются анализу.
* Формальная инспекция – форма проведения рассмотрений и некоторых видов анализов. «Формальная» означает, что результат инспекции является основанием для изменения статуса документа.
* В ходе формальной инспекции одним или несколькими участниками проекта, назначенными инспекторами, осуществляется независимая проверка соответствия инспектируемых документов исходным документам. Для самоконтроля инспектора при проведении формальных инспекций, как правило, применяются заранее подготовленные и утвержденные проверочные перечни, представленные в виде чек-листов соответствующего issue на GitHub, различные для разных типов объектов инспекции. Ответы на вопросы проверочного перечня даются инспектором на основании экспертной оценки объектов формальной инспекции независимо от других инспекторов, если таковые имеются. Проверочные перечни хранятся в соответствующем Pull Request на GitHub и утверждаются первоначально менеджером изменений.

# 5. Среда верификации

Инструменты, использующиеся в процессе верификации ПО, и их предназначение описаны в следующей таблице.

*Таблица 2. Программные средства, устанавливаемые на рабочих местах разработчиков ПО*

|  |  |  |
| --- | --- | --- |
| Название | Разработчик | Назначение |
| Windows 7 или более поздние версии | MicrosoftCorp. | Операционная система |
| Eclipse PDT | Eclipse Foundation | Интегрированная среда разработки |
| PHP | PHP Project | Язык программирования, используемый для создания серверной части приложения |
| Apache HTTP Server | Apache Foundation | Веб-сервер |
| MySQL | Oracle Corp. | Система управления базами данных |
| phpMyAdmin | phpMyAdmin Developer Team | Веб-интерфейс для администрирования СУБД MySQL |
| Bootstrap | Bootstrap Core Team | Набор инструментов для создания клиентской части приложения на языках HTML и CSS |
| GitHub | Microsoft Corp. | Система поддержки разработки и контроля версий |

# 6. Критерии перехода

Все тестовые примеры разработаны, прогнаны с использованием соответствующей среды испытаний – критерий начала процесса верификации.

Отчет об верификации ПО создан и идентифицирован, все найденные несоответствия зафиксированы в соответствующем Pull Request на GitHub – критерий окончания процесса верификации.

# 7. Соображения, связанные с обособлением

Обособление ПО в практикуме не применяется.

# 8. Допущения относительно компилятора

Предполагается, что единожды успешно выполнив испытания ПО, можно сделать допущение, что результаты работы интерпретатора не внесёт ошибок в последующие сборки исполняемого и объектного кода. Это допущение будет сохраняться, при условии неизменности версии интерпретатора, его программно-аппаратного окружения и способа использования (неизменности параметров интерпретатора и т.д.)

# 9. Инструкции повторной верификации

Процесс управления конфигурацией ПО гарантирует, что для внесения любого изменения в данные жизненного цикла ПО должен быть создан запрос на изменение. При рассмотрении сообщений о проблемах (см. План УК ПО [3]), группа управления изменениями определяет самый ранний этап жизненного цикла ПО, где требующееся изменение начинает влиять на результаты выполняемых процессов жизненного цикла ПО и создает запросы на изменения для этого этапа и всех последующих этапов жизненного цикла ПО, затрагиваемых изменениями. Список тестовых примеров, которые следует изменить и/или перепрогнать в связи с изменениями, определяется при рассмотрении сообщений о проблемах (issue в github). Этот процесс гарантирует, что необходимые изменения будут внесены и верифицированы для каждого из этапов жизненного цикла, начиная с этапа, где изменение начинает влиять на результаты выполнения процесса.

# # 10. Ранее разработанное ПО

Ранее разработанное ПО в практикуме не используется.

# # 11. Многоверсионное разнородное ПО

Многоверсионное разнородное ПО в рамках настоящего проекта не используется.