Содержание

Список рисунков

Список таблиц

1 ВВЕДЕНИЕ

1.1 Назначение

Данный документ определяет мероприятия, которые будут выполняться в рамках курса “Методология программной инженерии” (“МПИ”) для достижения целей процессов разработки ПО.

1.2 Область применения

Данный документ применяется исключительно для демонстрации мероприятий, которые могут использоваться при достижении целей процессов, определенных рамками курса “МПИ”. Положения, описанные в данном документе, могут быть неприменимыми при выполнении реальных проектов и должны быть пересмотрены с учетом нужд реального проекта в каждом конкретном случае.

В данном документе использована спиральная схема организации ЖЦ ПО. Для использования более сложных схем организации ЖЦ, которые могут потребоваться в реальных проектах, потребуется переработка раздела 3 документа.

1.3 Ссылки

будут пополняться в процессе работы

1.4 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.
Методология программной инженерии	Название практикума, моделирующего процесс разработки реальной системы.

Таблица 2. Сокращения

Термин	Определение, толкование
ГК	Гарантия качества
ЖЦ	Жизненный цикл
ЗИ	Запрос на изменение
ПО	Программное обеспечение
СПР	Система поддержки разработки
УК	Управление конфигурацией
ФИ	Формальная инспекция

2 ОРГАНИЗАЦИЯ ЖИЗНЕННОГО ЦИКЛА ПО

2.1 Ответственность

Руководитель проекта ответствен за распределение задач сотрудникам посредством назначения ЗИ ответственным разработчикам. Назначение ЗИ ответственным разработчикам должно проводиться таким образом, чтобы исключить одновременную работу различных сотрудников над одними и теми же частями документов.

Ответственный разработчик, назначенный руководителем проекта, ответствен за выполнение соответствующих мероприятий, приведенных в разделе “Процессы разработки ПО”, в соответствии с положениями данного документа.

2.2 Взаимодействие процесса разработки с другими процессами ЖЦ ПО

Схема взаимодействия процесса разработки ПО с другими процессами ЖЦ ПО представлена на рисунке 1. Для облегчения восприятия рисунка на нем не показаны процессы УК ПО (на самом деле все входы и выходы взаимодействующих процессов проходят через процесс УК ПО) и ГК ПО. Полный перечень входов и выходов процесса верификации ПО также не показан, показаны лишь те, которые имеют непосредственное отношение к описанному в данном документе процессу разработки ПО. Процесс ГК ПО в рамках курса “МПИ” явно не выделяется (не рассматривается).

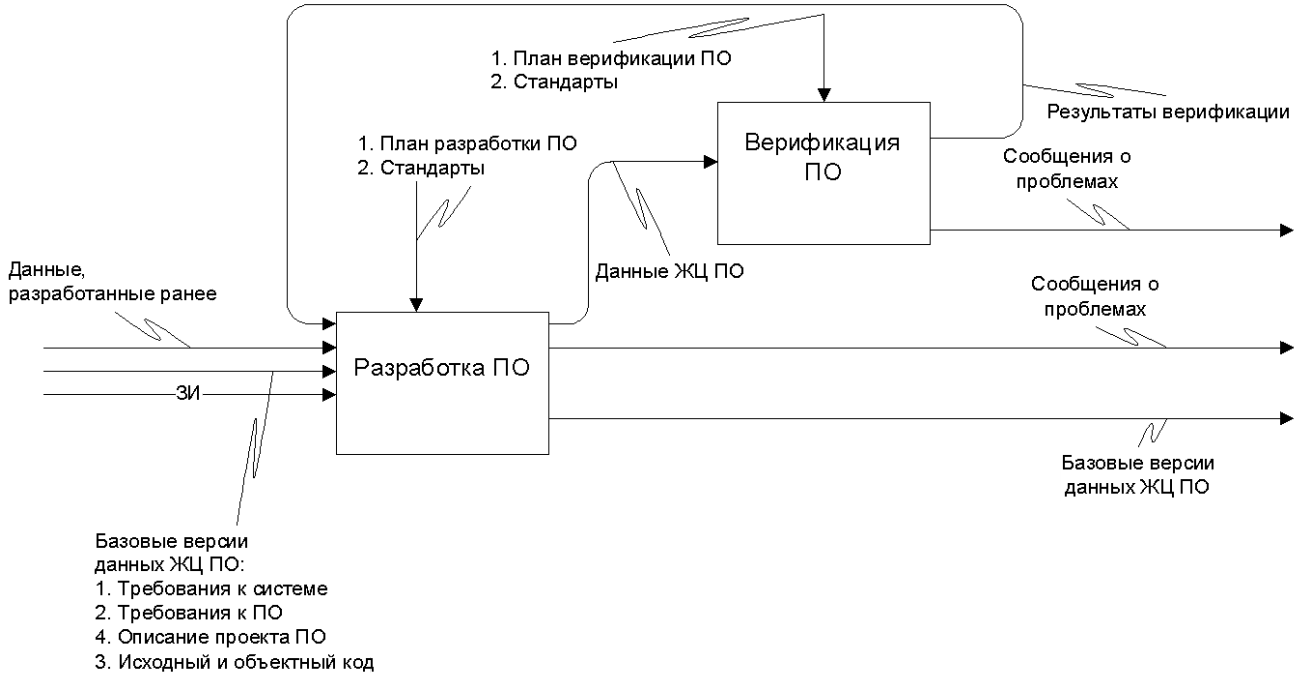

Рисунок 1. Схема взаимодействия процесса разработки ПО с другими процессами ЖЦ ПО

Как видно из рисунка, на вход процесса разработки ПО поступают (от процесса УК ПО) базовые версии данных ЖЦ ПО: требования к системе, требования к ПО, описание проекта ПО, исходный код ПО. В результате выполнения мероприятий процесса разработки появляются новые версии указанных данных ЖЦ ПО, которые передаются в процесс верификации ПО. По результатам выполнения процесса верификации в процесс разработки передается результат верификации, который может

потребовать дополнительного изменения данных ЖЦ ПО или же, если в процессе верификации не найдено несоответствий, устанавливается новая базовая версия для данных ЖЦ ПО.

Несоответствия, выявленные в результате выполнения мероприятий процесса разработки ПО, фиксируются посредством сообщений о проблемах, передаваемых в процесс УК ПО. Базовые версии данных поступают в процесс УК ПО.

3 ПРОЦЕССЫ РАЗРАБОТКИ ПО

Разработка ПО при спиральной модели ЖЦ представляет собой совокупность “спиралей” - ветвей разработки.

Процесс разработки ПО на каждой спирали состоит из процессов, показанных на рисунке 2. Процесс УК ПО, а также управляющие воздействия для процессов не показаны. Также не показан процесс верификации ПО, через который проходят все выходы процессов разработки ПО прежде, чем пойти на вход следующего процесса разработки ПО. Тот факт, что на вход следующего процесса идут верифицированные данные предшествующего процесса, отражен тем, что для каждого типа данных ЖЦ ПО, идущих на вход процессов, на рисунке сделана пометка “базовая версия”.

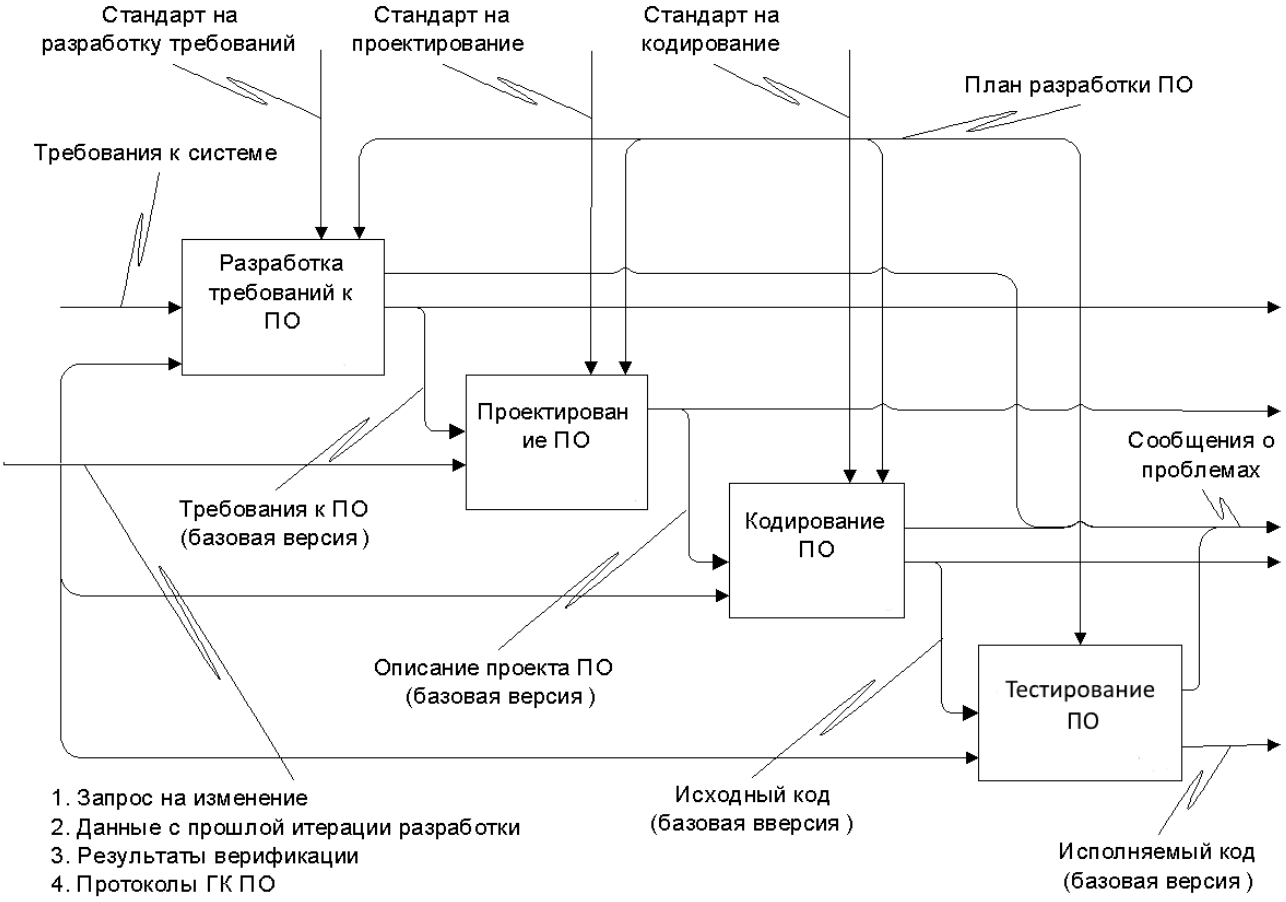

Рисунок 2. Процессы разработки ПО на каждой спирали

Детальное описание каждого из процессов разработки ПО дано в последующих разделах.

3.1 Разработка требований к ПО

Целями процесса разработки требований к ПО являются:
- Разработка требований к ПО высокого уровня;
- Разработка производных требований к ПО высокого уровня.

Постановка целей осуществляется в соответствии с методикой SMART. Поставленные цели должны соответствовать следующим принципам:

- S - Specific (Конкретный) - Цели должны быть обозначенными в виде четких результатов.
- M - Measurable (Измеримый) - Цели должны быть измеримыми в конкретных показателях.
- A - Achievable (Достижимый) - Цели должны быть достижимы, так как реалистичность выполнения задачи влияет на мотивацию исполнителя.
- R - Relevant (Реалистичный) - Цели должны быть реалистичными и значимыми, то есть достижимыми конкретными исполнителями.
- T - Time-bound (Ограниченный во времени) - Цель по SMART должна быть ограничена по выполнению во времени, а значит должен быть определен финальный срок, превышение которого говорит о невыполнении цели.

3.1.1 Входные данные процесса разработки требований к ПО

- План разработки ПО (настоящий документ);
- Требования к системе;
- Запрос на изменение;

3.1.2 Описание процесса разработки требований к ПО

При входе в процесс ответственный разработчик должен перевести задачу в раздел «В процессе» в веб-приложении для управления проектами “Trello”, тем самым извещая руководителя проекта о начале работы по ЗИ, после чего он должен выполнить действия, заданные в ЗИ.

На основании анализа описания ЗИ (при входе в процесс по условию А), записи о ФИ (при входе в процесс по условию Б) и требований к системе разработчик определяет те документы требований к ПО, которые необходимо изменить, создать или удалить.

Если документы требований к ПО уже существуют, то ответственный разработчик вносит изменения в указанные в ЗИ данные. Требования к ПО разрабатываются на основе анализа требований к системе.

Если в процессе разработки требований обнаружены несоответствия во входных данных процесса, то ответственный разработчик должен создать сообщение о проблеме.

После завершения работы над требованиями ответственный разработчик должен известить об этом руководителя проекта и переместить ЗИ в раздел “Готово” в приложении “Trello”.

3.1.3 Выходные данные процесса разработки требований к ПО

- Требования к ПО;
- Сообщения о проблемах.

3.2 Проектирование ПО

Целями процесса проектирования ПО являются:
- Определение архитектуры ПО;
- Разработка требований к ПО низкого уровня;
- Разработка производных требования к ПО низкого уровня.

3.2.1 Входные данные процесса проектирования ПО

- План разработки ПО (настоящий документ);
- Требования к ПО;
- Запрос на изменение;

3.2.2 Описание процесса проектирования ПО

С процедурной точки зрения процесс внесения изменений в файлы описания проекта ПО аналогичен процессу внесения изменений в требования к ПО, описанному в разделе 3.2.1. Отличие заключается в том, что разработчик вносит изменения в описание проекта ПО, руководствуясь стандартом на проектирование ПО и использует в качестве входных данных требования к ПО, а не требования к системе.

Основным методом при разработке описания проекта ПО будет являться метод структурного проектирования, представляющий собой процесс последовательного разбиения (декомпозиции) ПО на компоненты, а компонента на подкомпоненты и функции.

Инструментами для разработки описания проекта ПО будут являться встроенные инструменты в среду разработки, а также веб-приложение draw.io и Microsoft Visio 2010. Основная часть описания проекта будет выполнена в текстовом виде. Рисунки, схемы и другие графические элементы, являющиеся частью описания проекта ПО, будут оформляться в Visio 2010.

3.2.3 Выходные данные процесса проектирования ПО

- Описание проекта ПО;
- Сообщения о проблемах.

3.3 Кодирование ПО

Цель процесса кодирования ПО: разработать исходный код трассируемый, верифицируемый, непротиворечивый и правильно реализующий требования к ПО низкого уровня.

3.3.1 Входные данные процесса кодирования ПО

- План разработки ПО (настоящий документ);
- Описание проекта ПО;
- Требования к ПО;
- Запрос на изменение;

3.3.2 Описание процесса кодирования ПО

С процедурной точки зрения процесс внесения изменений в файлы модулей исходного кода аналогичен процессу внесения изменений в требования к ПО, описанному в разделе 3.2.1.

Отличия заключаются в следующем:

1) разработчики вносят изменения в модули исходного кода, и используют в качестве входных данных описание проекта ПО и требования к ПО, а не требования к системе;

2) при выполнении мероприятий ответственный разработчик использует инструментарий процесса кодирования (см. раздел 4.1).

3.3.3 Выходные данные процесса кодирования ПО

- Исходный код ПО;
- Сообщения о проблемах.

3.4 Тестирование ПО

Цель процесса тестирования ПО: определение соответствия ПО заявленным требованием, поиск ошибок и недоработок, определение необходимости проведения еще одной итерации спиральной модели ЖЦ.

Достижение поставленных целей осуществляется в соответствии с концепцией “пирамиды тестирования”. Руководствуясь данной концепцией, необходимо разработать и реализовать набор тестовых сценариев, который можно разделить на следующие слои:

- Юнит-тесты - тесты “белого ящика”, которые проверяют отдельные части кода, такие как функции, методы и классы. Должны быть написаны на том же языке, что и тестируемый продукт и храниться в том же репозитории. Прогоняются как часть сборки, чтобы сразу же увидеть успешно ли завершается тест или нет. Их доля среди тестов наибольшая, как самых низкоуровневых.
- Интеграционные тесты - тесты “черного ящика”, которые проверяют, что интеграционные точки между компонентами продукта работают корректно. Тестируемый продукт должен быть в активной фазе и развернут на тестовом окружении. Количество много меньшее, нежели более низкоуровневых тестов.
- End-to-end тесты - тесты “черного ящика”, которые проверяют пути выполнения внутри продукта. Здесь они рассматриваются как многошаговые интеграционные тесты и, как наиболее высокоуровневые, представлены в наименьшем количестве.

3.4.1 Входные данные процесса тестирования ПО

- План разработки ПО (настоящий документ);
- Описание проекта ПО;
- Исходный код ПО;
- Стандартные тестовые изображения.

3.4.2 Описание процесса тестирования ПО

Процесс юнит-тестирования является полностью автоматизированным. Юнит-тесты проводятся при каждом внесении изменений в уже разработанные функциональные модули, для первичных реализаций ранее не реализованных модулей набор тестов пополняется соответствующими новыми тестовыми случаями.

Процесс интеграционного тестирования также полностью автоматизирован. Интеграционные тесты начинают проводиться по мере появления новых реализаций модулей, поэтому процесс тестирования осуществляется сверху вниз: тестирование начинается с наиболее верхних модулей, отсутствующие модули более нижних уровней заменяются модулями-заглушками. Таким образом интеграционные тесты начинают производиться на ранних этапах разработки, когда реализация многих низкоуровневых модулей еще отсутствует.

End-to-end тестирование выполняется в полуавтоматическом режиме. Так как к появлению необходимости в тестах данного типа большая часть модулей имеет реализацию, возможно проведение тестирования снизу вверх, от низкоуровневых модулей к высокоуровневым. Тесты данного типа проводятся реже всего, на финальных этапах разработки продукта, и автоматизированы. Однако, результаты такого тестирования оцениваются в ручном режиме, экспертами, и их целью является получение представления о готовности продукта в целом.

3.4.3 Выходные данные процесса тестирования ПО

- Заключение о соответствии/несоответствии ПО установленным требованиям.
- Сообщения о проблемах.
- Заключение о необходимости проведения следующей спирали разработки ПО.

4 СРЕДА РАЗРАБОТКИ ПО

Совместная работа над проектом достигается при помощи распределенной системы управления версиями - Git с использованием репозитория, предоставленного веб-сервисом GitHub.

4.1 Инструменты разработки ПО

На компьютерах разработчиков ПО устанавливаются следующие инструменты.

Таблица 3. Программные средства, устанавливаемые на рабочих местах разработчиков ПО

Название	Разработчик	Назначение
Windows 10	Microsoft Corp.	Операционная система
Microsoft Visio 2010	Microsoft Corp.	Оформление схем, рисунков, диаграмм
Microsoft Office 2010	Microsoft Corp.	Подготовка документов, почтовая коммуникация, подготовка тест-планов
Microsoft Visual Studio 2015	Microsoft Corp.	Написание исходных кодов ПО
PyCharm	JetBrains	Написание исходных кодов ПО

4.2 Языки программирования

Основным языком программирования при разработке исходного кода должен являться язык Python.

Для написания и использования разрабатываемой библиотеки используются следующие пакеты языка программирования Python:

- NumPy
- Pillow