

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**“МИРЭА – Российский Технологический Университет”**

**РТУ МИРЭА**

**Практическая работа по**

**Тех. разработки ПО АСОИУ**

**Вариант 11**

Выполнил студент группы

ИВБО-07-16 Попов К.С.

Принял Сувальский А.А.

Москва 2019

Содержание

[1 Вопрос №2 3](#_Toc24024997)

[2 Вопрос №4 8](#_Toc24024998)

[3 Вопрос №8 10](#_Toc24024999)

[4 Список использованной литературы 12](#_Toc24025000)

**Билет №11**

1. Вопрос №2
   1. Перечислите основные приемы экстремального программирования. Последовательно распишите для каждого приема, как скажется на проект (Разработки ПО) и его реализацию исключение выбранного приема из методологии?
      1. Осуществление обратной связи:

* Взаимодействие с заказчиком;
* Планирование этапов разработки;
* Парное программирование;
* Разработка через тестирование.
  + - 1. Взаимодействие с заказчиком – подразумевает наличие обратной связи команды разработчиков с представителем заказчика. Такая связь улучшает взаимопонимание между двумя сторонами и способствует повышению качества конечного продукта.

В противном случае, итоговый продукт не будет удовлетворять потребностям заказчика и, в лучшем случае, будет неоднократно перерабатываться.

* + - 1. Планирование этапов разработки – формирование начального плана разработки и постоянное его обновление по мере продвижения прогресса.

Иначе процесс разработки раздробляется на части и нарушается последовательность процесса разработки, что влечет за собой увеличение времени на разработку проекта и усложняет сам процесс.

* + - 1. Парное программирование – предполагается, что код пишется парами программистов, каждый из которых выполняет свои задачи, не исключена и смена ролей. Данный прием позволяет улучшить взаимопонимание в команде, отлавливать ошибки и оптимизировать написанный код.

Отсутствие данного приема может сказаться на качестве кода и на его понимании остальными разработчиками.

* + - 1. Разработка через тестирование – способствует написанию программного кода под заготовленные тесты. В ходе данного приема понимается логика и функционал частей написанного кода.

Отсутствие тестирования кода приводит к неверной работе кода программы или неправильной логике реализации функций.

* + 1. Непрерывный процесс разработки
* Непрерывная интеграция;
* Рефакторинг;
* Частые небольшие релизы.
  + - 1. Непрерывная интеграция – каждая новая написанная часть кода интегрируется в общую систему. Такой подход позволяет отслеживать влияния добавленных частей кода на общую интегрированную сборку.

Если же проводить интеграцию на заключительных частях разработки, то можно столкнуться в результате с неправильно интегрированной системой, исправление которой займет очень много времени.

* + - 1. Рефакторинг – процесс переработки и улучшения состояния системы, дабы соответствовать предъявленным требованиям. Каждый этап рефакторинга должен сопровождаться тестированием.

В противном случае, результатом разработки может стать громоздкий и большой код, снабженный излишками.

* + - 1. Частые небольшие релизы – выпускаемые версии продукта должны поступать в эксплуатацию как можно чаще. Каждая версия должна снабжаться ценным, для предметной области заказчика, функционалом и с наименьшими временными затратами.

Отсутствие данного приема сводит всю разработку к одному большому релизу. Данный подход может быть очень рискованным и породить множество недостатков системы.

* + 1. Простота процесса
* Коллективное владение кодом;
* Метафора;
* Стандарты кодирования.
  + - 1. Коллективное владение кодом – каждый участник команды разработчиков в праве вносить изменения в код программы. Данный подход улучшает понимание кода. Таким образом увеличивается скорость разработки.

Как и наличие, так и отсутствие данного приема может повлечь за собой риск появления ошибки. Например, инициатива участника команды внести в часть кода, или что-то исправить, может быть наказана системой в бедующем.

* + - 1. Метафора – анализ структуры системы позволяет понять каким образом связана система и как она работает, где можно добавить новый компонент в систему и как они должны быть реализованы.
      2. Стандарты кодирования – все участники команды разработчиков должны соблюдать общие требования работы с кодом. Таким образом разработчикам не приходится тратить время на понимание того, или иного куска кода.

Игнорирование данного приема отрицательно сказывается на взаимодействие в команде, а также отрицательно влияет на качество рефакторинга.

Все основные приемы данной методологии отображены на рисунке 1.1

* 1. Какие основные преимущества и недостатки есть у данной методологии?

Преимущества:

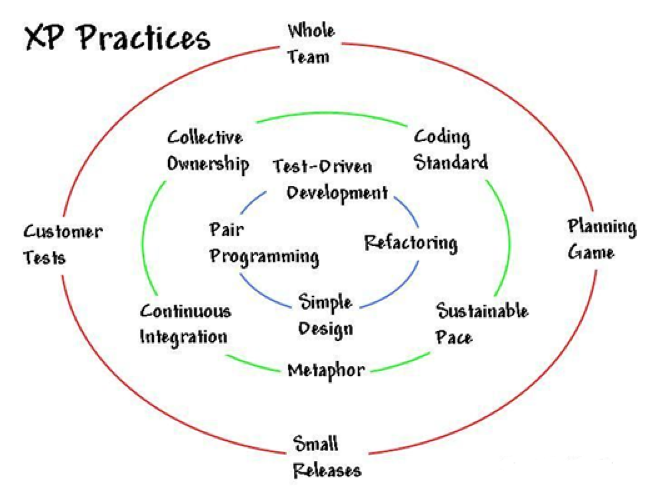


Рисунок 1.1 – Основные практики методологии экстремального программирования

* Соблюдение всех правил методологии позволяет на выходе получить продукт высокого качества, с минимальными затратами различных ресурсов;
* Качественный контроль работоспособности системы;
* Быстрое реагирование на возникшие проблемы;
* Высокое качество кода, за счет осуществления поддержки и использования рефакторинга;
* Владения кодом всей командой
* Быстрый темп разработки продукта.

Недостатки:

* Требует высокой квалификации;
* Постоянное необходимое взаимодействие с заказчиком;
* Низкое сопровождение проекта документацией.

# Вопрос №4

* 1. Зачем нужно тестировать программы?

Процесс тестирования заключается в проверке соответствие программы выдвинутым требованиям, путем создания нетривиальных ситуаций, выбранных определенным образом. Тестировка позволяет сравнить насколько отличается фактический результат программы от ожидаемого.

* 1. Какие способы/методологии тестирования программ вы знаете?

В большинстве проектов предполагается написание автоматических тестов. Особое внимание уделяется двум видам тестирования:

* Юнит тестирование модулей;
* Функциональное тестирование.

Юнит тестирование модулей – позволяет осуществить проверку корректности отдельных модулей кода программы. Основная идея состоит в том, чтобы написать тест для каждой функции программы. При этом программа разбивается на отдельные части, с целью проверки работоспособности каждой из них. Данный метод позволяет упростить интеграцию кода и повысить отказоустойчивость программы, из-за меньшего количества ошибок.

Функциональное тестирование – глобальное тестирование ПО с целью проверки соответствия результата предъявленным требованиям. Данный вид тестирования оценивает актуальность, точность, защищенность и другие параметры системы.

* 1. Когда можно обойтись без тестирования программы?

При использовании в программе сторонних библиотек и модулей, разработанных ранее и прошедших проверку функциональности, их тестирование может и не потребоваться.

# Вопрос №8

* 1. Исходя из знаний, полученных в данном курсе, проведите

первичный анализ и подробно опишите:

* + 1. Как вы будете согласовывать, координировать и планировать работу с партнером?

Использование определенных видов методологий разработки и определенных сервисов для согласования и координации, а именно:

* При выборе методологии разработки очень важно учесть возможность подстраиваться под меняющиеся требования и иметь возможность вернуться к предыдущим итерациям проекта. Таким образом наш выбор падет на методологии RUP или XP.
* В выборе таск-менеджера большое значение будет иметь удобство командной работы, платформа и цена сервиса. Оптимальным выбором окажутся такие сервисы как: Todoist, Microsoft To-Do и Wrike.
  + 1. Все этапы вашей работы.
* Составление спецификации - осуществляется описание входных данных и формулировка требований к результату. Выбор языка программирования для написания программы. Декомпозиция поставленной задачи с подробным описанием функционала и состава каждого модуля.
* Разработка алгоритма программы - определение последовательности работы программы. В случае, если имеется несколько доступных алгоритмов, выбрать из них оптимальный.
* Написание тестов – необходимо проанализировать данные и составить тесты, основываясь на имеющихся данных физического процесса;
* Кодирование – реализация написанных алгоритмов в коде программы. Данную процедуру следует разделить на итерации, для удобства тестирования бедующих частей программы и их интеграции в единое целое. Такой способ разработки кода позволит обнаружить ошибки в кусках кода, эффективно производить тестировку и постепенно наращивать общий код программы с проверкой общего функционала всей программы;
* Подготовка документации и презентации для демонстрации достигнутых результатов.
  + 1. С какими рисками вы можете столкнуться в ходе реализации проекта?
* Вероятность не успеть к дедлайну;
* Из-за неправильной постановки задачи и выдвинутых требований, можно прийти к неверному результату;
* Ошибки, возникшие в результате отсутствия практического опыта, или неверное понимание предметной области;
* Возникновение непредвиденных ситуаций.

# Список использованной литературы

* Гагарина, Л. Г. Технология разработки программного обеспечения / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. - М.: Форум, Инфра-М, **2013**.
* Котляров, В. П. Основы тестирования программного обеспечения / В.П. Котляров, Т.В. Коликова. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, **2011**.
* Макаровских, Т. А. Документирование программного обеспечения. В помощь техническому писателю. Учебное пособие / Т.А. Макаровских. - М.: Ленанд, 2015.
* Лукин, В. В. Технология разработки программного обеспечения. Учебное пособие / В.В. Лукин, В.Н. Лукин, Т.В. Лукин. - М.: Вузовская книга, 2015.