УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные, дайте ответы на контрольные вопросы.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 06.03.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 36 (продолжение)

Тема: «Описание процесса тестирования как этапа разработки программного продукта»

План лекции:

1. Задача тестирования

2. Эскизный проект

3. Модульное тестирование

4. Вспомогательный код

5. Достаточные тестовые случаи (наборы тестов)

6. Нецелевая аудитория

Тестирование – вид деятельности испытательной лаборатории, целью которого является проверка соответствия программного продукта возможностям, заданным в тактико-техническом задании и описанным разработчиком. В жизненном цикле программного обеспечения (ПО) процесс тестирования в основном стоит за этапом разработки, перед эксплуатацией продукта пользователями. Психология разработчика, а также финансирование создания ПО, как правило, не позволяют начать процесс тестирования пока

разработка программного продукта не подойдет к завершению. С точки зрения данных исследований тестирование может быть осуществлено в различные стадии разработки, а не только на завершающем этапе и применено не только к программному коду.

Задача тестирования может быть поставлена следующим образом:

Рассчитать моменты для проведения подтестов так, чтобы сумма времен, затраченных на каждый подтест, была меньше чем время, затраченное на тестирование продукта целиком (рисунок 1). Испытательная лаборатория, имеющая соответствующие лицензии и подготовленный штат экспертов, выступает в роли тестирующей организации. Причем задачей испытательной лаборатории должно быть обнаружение отказов, но не ошибки или неисправности, которые явились причиной отказов. Выявление источников отказов вменяется в обязанности разработчиков.

На рисунке 1а приведено описание процесса тестирования, использующегося на сегодняшний день. В данном случае разработка ПО находится на стадии завершения и передано на проведение тестирования. Тестирование является циклическим процессом и заканчивается, когда результаты тестирования оцениваются как положительные. Однако процесс доработки ПО при данном подходе может затянуться на длительное время. На рисунке 1б приведен подход, при котором разработчику нет необходимости ждать стадии завершения разработки, а отправлять отдельные части ПО на тестирование.

Безусловно, программное обеспечение отличается между собой по различным параметрам и моменты для проведения подтестов для каждого программного продукта будут отличаться, поэтому в статье приведен методологический аппарат для проведения тестирования базового продукта в рамках поставленной выше задачи.

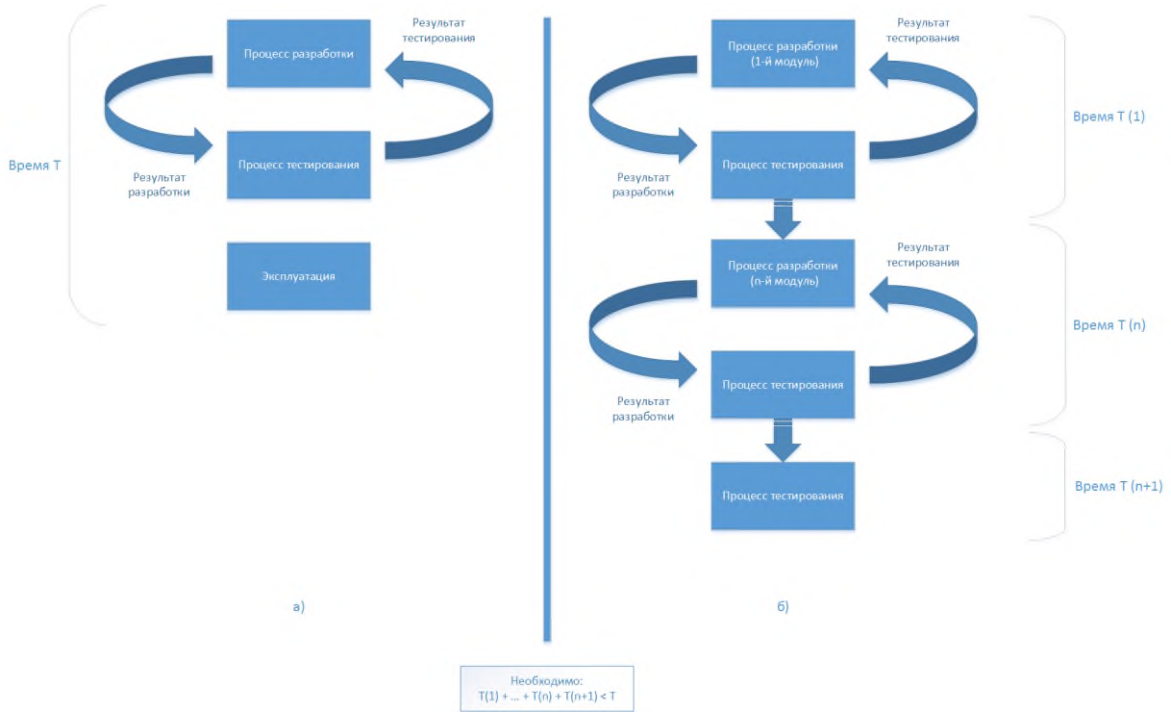

Рисунок 1 – Постановки задачи тестирования

Описанные ниже методы в совокупности и по отдельности позволят определить моменты времени для проведения подтестов.

Эскизный проект

Первым этапом в жизненном цикле продукта (с точки зрения заказчика), в том числе и программного, является эскизное проектирование (ЭП). Разработчики, выбирая методы и инструментальные средства на этапе ЭП, тем самым устанавливают ограничения на процесс тестирования. Так что одной из точек приостановления процесса разработки и обращения к тестировщикам, является момент выбора схемы программного проекта (обычно эскизный проект). Лучшим решением на данном этапе является привлечение на защиту эскизного проекта испытательной лаборатории.

Модульное тестирование

Модульность в программировании является не только одним из основных принципов программирования, но также требованием, задаваемым в тактико-технических заданиях на разработку ПО. Модульные тесты выявляют такие виды ошибок, как бесконечный цикл, невыход из рекурсии,

присвоение неинициализированной переменной и многие другие. Программисты стараются строить свою работу по принципу модульности. И, в основном, если модуль готов, то он может быть представлен на тестирование. Однако это является рутиной в работе как испытательной лаборатории, так и разработчиков, так как проект состоит из большого числа модулей.

Поэтому целесообразнее на тестирование представлять модули, выполняющие основное функциональное назначение программного продукта. В данном случае необходимо применять метод покрытия кода, который, по своей сути, является тестированием методом белого ящика. Тестируемое ПО собирается со специальными настройками или библиотеками и/ или запускается в особом окружении, в результате чего для каждой используемой (выполняемой) функции программы определяется местонахождение этой функции в исходном коде ПО. Этот процесс позволяет разработчикам и специалистам по обеспечению качества определить части системы, которые, при нормальной работе, используются очень редко или никогда не используются (такие как код обработки ошибок и т.п.). Это позволяет сориентировать экспертов на тестирование наиболее важных режимов.

На примере разработки расчетной задачи (рисунок 2) существует необходимость сориентировать экспертов на тестирование модуля, отвечающего за основной функционал ПО – модуля расчета.

Рисунок 2 – Расчет времени для проведения подтестов

Вспомогательный код

Как было описано выше, задачей тестирования является обнаружение отказов. Поиском неисправного кода, приводящего к отказам ПО, занимается разработчик. Время, затраченное на поиск неисправного кода, может быть очень большим. В данном случае необходимо перед тестированием в исходный код поместить вспомогательные переменные, сигнализирующие об изменениях основных переменных, о связи модулей по информации и о другой информации, делающей процесс тестирования более прозрачным. При данном подходе снижается не только количество циклов между процессами разработки и тестированием, но и время на поиск сбоев.

Достаточные тестовые случаи (наборы тестов)

Когда программный продукт наиболее пригоден для использования (выполняет декларированный функционал), он может быть отдан в испытательную лабораторию. Понятие «тестовые наборы» не ново и широко используется, но, сколько этих наборов может быть и каким образом они могут быть построены, зависит от квалификации экспертов. Одним из способов сокращения количества «тестовых наборов» является использование метода минимального покрытия графа программы. Пример применения такого подхода представлен на рисунке 3. Используя метод минимального покрытия графа программы, количество маршрутов выполнения программы сокращается (рисунок 3), что позволяет снизить количество «тестовых наборов».

Нецелевая аудитория

В используемой терминологии в области тестирования существует понятие альфа-тестирование – имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком на стороне разработчика. Часто альфа-тестирование применяется для законченного продукта в качестве внутреннего

приёмочного тестирования. Иногда альфа-тестирование выполняется с использованием отладчика или окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться ПО.

Однако, имея облик потенциального пользователя, разработчик ПО не защитит продукт от рук пользователей, которые не владеют компьютерной терминологией. Поэтому перед использованием программного продукта, считается целесообразным отправить его на использование нецелевой аудитории. Итак, четкие границы для начала подтестов достаточно трудно определить, но описанные выше подходы позволят ускорить процесс тестирования ПО в различные моменты его жизненного цикла.

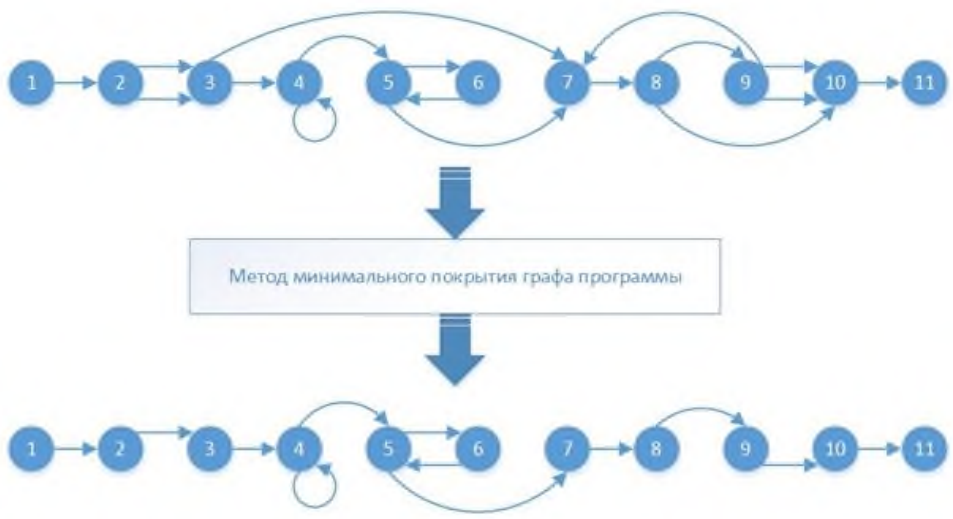

Рисунок 3 – Расчет времени для проведения подтестов

Контрольные вопросы:

1. Задача тестирования

2. Эскизный проект

3. Модульное тестирование

4. Вспомогательный код

5. Достаточные тестовые случаи (наборы тестов)

6. ***Нецелевая аудитория***