Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий
Кафедра технологий программирования

Методические указания
к выполнению лабораторной работы №11

по дисциплине «**Надёжность программного обеспечения**»
для специальности 1-40 01 01 Программное обеспечение информационных технологий

на тему «**Системные испытания. Прогон тестов**»

Новополоцк, 2017 г.

Название: «Системные испытания. Прогон тестов».

Цель работы: ознакомиться с основами обнаружения и отслеживания дефектов, прогоном тестов. Научиться составлять сводный отчёт по системным испытаниям.

Теоретическая часть

Для профессиональных тестировщиков программного обеспечения стадия системных испытаний – это то же, что игровой день для футболиста или день спектакля для театрального актера. Проделана большая работа по планированию, все подготовительные работы завершены, наступило время действовать. Как показано на рисунке 1, первое, что нужно сделать, это проверить, все ли на месте и все ли готово. Все эти мероприятия могут быть дополнены применением критериев входа в системные испытания, определения которых включаются в план проведения испытаний. Критерий входа в испытания принимает форму опросного листа: составлен ли, пересмотрен и утвержден план проведения испытаний? Завершила ли группа разработчиков запланированное тестирование модулей и проверку взаимодействия и функционирования компонентов программного продукта? Проведены ли испытания «на герметичность» с целью удостовериться в том, что тестируемая программа может быть установлена и, по меньшей мере, способна продемонстрировать открытый экран?

Если достигнуто соответствие критериям вхождения в испытания, тестирование системы можно начинать. Начало тестирования обычно представляет собой важный этап графика проектных работ. В идеальном случае задержка системных испытаний преобразуется в задержку поставки программного продукта, хотя группу тестирования часто просят отыскать возможности ускорить испытания, чтобы программный продукт был поставлен вовремя. Несмотря на риск использования слишком многих аналогий, добавим еще одну: если сравнить разработку программного продукта с эстафетой, то группа тестирования пробегает последний этап гонки. Если бегуны на предыдущих этапах проигрывают своим соперникам, то нереально надеяться на то, что спортсмен, бегущий на последнем этапе, – это супермен, способный наверстать ранее упущенное. Реально это или нереально, но на практике ситуация, складывающаяся в бизнесе, часто приводит к необходимости ускоренного выполнения стадии тестирования, поэтому всегда нужно быть готовым применить тестирование по приоритетам и другие планы на случай непредвиденных обстоятельств с целью смягчение последствий от овеществления рисков нарушения графика работ.

Рисунок 1 – Обзор системных испытаний

Основу тестирования составляет прогон тестов, включенных в план проведения испытаний. В результате прогона тестов обнаруживаются различные дефекты и выдаются сообщения об обнаруженных дефектах. При этом данные, обеспечивающие возможность отслеживания дефектов, вводятся в специальную базу данных, в которой хранится информация, предназначенная для отслеживания дефектов.

Во время прогона тестов результаты тестирования должны передаваться другим подразделениям организации. Все, кто принимает участие в разработке, маркетинге или в руководстве проектом, должны знать, как продвигается тестирование, сколько и какие типы дефектов были обнаружены. По окончании тестирования должен быть подготовлен отчет, в котором подводятся итоги тестирования. В этом отчете должен использоваться набор критериев для оценки готовности программного продукта к выпуску. Например, если были выполнены все запланированные тесты, но еще остается какое-то допустимое число неустраненных дефектов, группа тестирования может утверждать, что программный продукт соответствует требованиям, предъявляемым к нему заказчиком, благодаря чему можно начинать поставки.

Обнаружение и отслеживание дефектов

Цель тестирования заключается в нахождении дефектов. Однако недостаточно только отыскать дефекты. Об их наличии должным образом должны быть оповещены разработчики, чтобы они смогли эти дефекты устранить, а сведения о результатах выполненных ими исправлений доводятся до заинтересованных служб с тем, чтобы график разработки программного продукта не нарушался. Любые неэффективные действия при оповещении о дефектах, об их отслеживании, устранении и контроле приводят к потере времени.

В связи со сказанным выше, следующие два вида деятельности считаются основными при выявлении дефектов. Первым из них является прогон тестов в соответствии с планом проведения испытаний для выяснения, имеются ли какие-либо расхождения между ожидаемыми и фактическими результатами тестирования. Вторым видом деятельности является качественное документирование в системе отслеживания дефектов всех отклонений, обнаруженных во время тестирования. Хорошее качество документации важно по двум причинам – во-первых, разработчикам нужна достоверная и надежная информация для того, чтобы снять все порождаемые неисправностями проблемы, и, во-вторых, тестировщики должны иметь возможность воспроизвести проблемы, чтобы убедиться, что внесенные изменения устранили дефект. Система, применяемая для документирования и отслеживания дефектов, представляет собой важное инструментальное средство тестирования программного обеспечения.

Определение состояний дефектов

Дефектом является ошибка в рабочем документе, таком как технические требования, план проведения испытаний или программный код. Дефект остается необнаруженным до тех пор, пока рабочий продукт не будет подвергнут пересмотру или не будет выполнен прогон программы, во время которого программа выдаст ожидаемые результаты.

После нахождения дефект подвергается анализу с целью определения, представляет ли он неисправность теста, ошибку в программном коде или дефект в проектных спецификациях. Если источником проблемы является тест, а не программный код, то тест должен быть исправлен, а обнаруженный дефект «проигнорирован». Проверяется также, насколько серьезен дефект: принадлежит ли рассматриваемый дефект к категории катастрофических, подлежащих обязательному устранению перед поставкой программного продукта? Является ли возникшая проблема настолько серьезной, что она существенно снижает функциональные возможности программного продукта, но при этом все еще остаются открытыми обходные пути, позволяющие продолжать процессы тестирования и разработки? Либо же эта проблема не настолько критична, что ее решение нельзя отложить до будущих выпусков программного продукта?

Программный код, содержащий обнаруженный дефект, должен быть направлен разработчикам для исправлений. Нужно внимательно следить за стадиями исправления, чтобы не потерять из виду сам дефект. Один из эффективных способов отслеживания дефекта состоит в четком определении явных состояний, которые может принимать дефект на пути от его обнаружения до завершающего тестирования. Каждое состояние должно быть недвусмысленным и обладать набором определений критериев входа и выхода из испытания.

В таблице 1 показан пример определения состояний дефекта. Названия состояний в каждой компании могут быть свои, возможны также дополнительные состояния, которые не указаны в этой таблице. В ней показан обобщенный набор названий и состояний, который, возможно, стоит

использовать как отправную точку при создании системы отслеживания дефектов «с нуля».

Таблица 1 – Определение категории дефекта

Категория дефекта	**Ответственность**	**Описание**
Новый (new)	Группа тестирования	Дефект был обнаружен во время тестирования, этот факт был отражен в отчете, однако разработчик не получил распоряжения устранить проблему.
Исправить (fix)	Группа разработки	Разработчик получил распоряжение устранить проблему, и соответствующие работы ведутся.
Отложить (defer)	Группа анализа дефектов	Принято решение о том, чтобы отложить работы по устранению дефекта до последующих выпусков программного продукта.
Игнорировать (trash)	Группа анализа дефектов	Дефект был получен по ошибке, программный продукт работает в соответствии с проектным замыслом, возможны и другие причины того, что решено ничего не делать по устранению дефекта.
Исправлено (repair)	Группа разработки	Разработчик выявил и устранил основную причину дефекта и выполнил предварительное тестирование с целью убедиться в том, что проблема решена.
Исправлено и проверено (fix verified)	Группа тестирования	Специалист по тестированию проверил исправление, используя для этой цели ту же методику тестирования и конфигурацию средств тестирования, которые позволили первоначально обнаружить дефект.

Отслеживание дефекта на протяжении его жизненного цикла, от его обнаружения до устранения и проверки правильности исправления, показывает, что он меняет свое состояние в соответствии с некоторым набором правил. На рисунке 2 приведен пример изменения состояний дефекта. Дефект обнаруживается во время пересмотра или в процессе системных испытаний и попадает в систему отслеживания дефектов, будучи в состоянии «новый». Ему присваивается уровень серьезности, после чего он передается на рассмотрение в группу анализа дефектов. Группа анализа

дефектов (известная в некоторых кругах как группа контроля над внесением изменений в техническую документацию) определяет, нужно ли исправлять дефект в рамках текущего выпуска или отложить это до будущих версий. Если группы разработчиков и тестировщиков провели дальнейший анализ уже после того, как дефект был зафиксирован, и пришли к заключению, что в программный код не содержит каких-либо проблем, требующих его исправления, то группа анализа дефектов может принять решение не обращать на него внимания. Действие, предпринятое группой анализа дефектов, документируется, а сам дефект может находиться в состоянии «исправить», «отложить» или «игнорировать».

Если принято решение исправлять дефект, то эта работа поручается разработчику, который вносит соответствующие изменения в программный код и проводит тестирование, дабы удостовериться, что исправленный код работает правильно. Как только разработчик придет к выводу, что дефект устранен, он меняет состояние дефекта на «исправлено» и включает исправленный код в сборку, направляемую на системные испытания. Если тестировщик убеждается в том, что дефект устранен, то последний переходит в окончательное состояние «исправлено и проверено».

Если вы не обладаете опытом тестирования или работаете в компании, которая не использует мощных промышленный средств отслеживания дефектов, возможно, возникнет впечатление, что методология отслеживания дефектов, в основе которой лежит состояние дефекта, чересчур избыточна и громоздка. Если вы занимаетесь разработкой программного продукта, выпуски которого редко когда содержат более 40 или 50 дефектов, то вы, скорее всего, правы. Однако в условиях разработки множества различных проектов количество дефектов, требующих отслеживания, достигает многих сотен и даже тысяч. Если система отслеживания неэффективна, или если хотя бы всего лишь несколько заметных серьезных дефектов просочатся в поставляемый программный продукт, руководству вашей испытательной лаборатории предстоят довольно-таки неприятные объяснения с заказчиком.

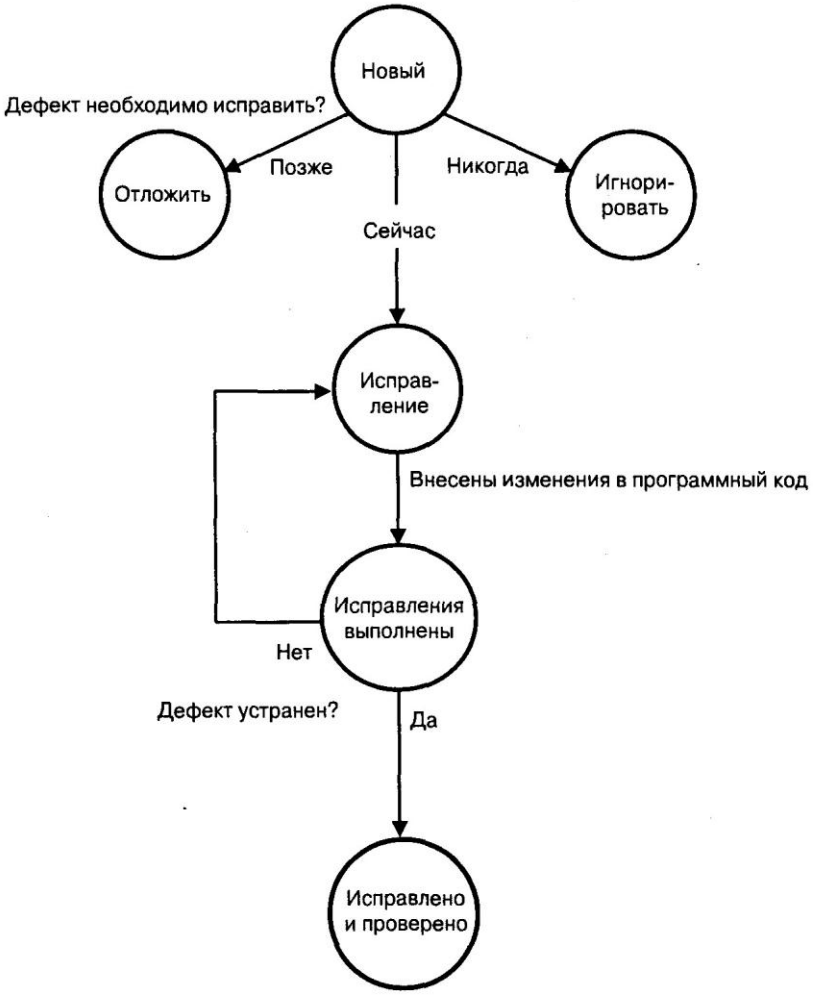

Рисунок 2 – Изменения состояний дефекта

Базовые характеристики системы отслеживания дефектов

В момент самого первого столкновения с дефектом необходимо собрать о нем большую порцию информации, чтобы облегчить задачу его последующего устранения Пример информации о вновь обнаруженном дефекте, которую необходимо записать в системе отслеживания дефектов, приводится в таблице 2. В этой таблице приводится список параметров с кратким описанием каждого параметра и причины необходимости его регистрации. Если для отслеживания дефектов применяются инструментальные средства управления базами данных, параметры, представленные в первом столбце, соответствуют полям базы данных.

Таблица 2 – Данные о дефекте, пребывающем в состоянии «Новый»

Характеристика	**Описание**	**Обоснование**
Идентификатор дефекта	Уникальный идентификатор дефекта, обычно это строка алфавитно-цифровых символов.	Позволяет отслеживать дефект в процессе изменения его состояний и логически связать с дефектом всю его обработку.
Состояние	Одно из допустимых состояний; в рассматриваемом случае	Присвоить дефекту состояние «новый» в момент его обнаружения.

Характеристика	Описание	Обоснование
дефекта	таким состоянием является «новый».	
Кем обнаружен	Имя исполнителя, обнаружившего дефект.	Обеспечивает возможность задавать вопросы относительно дефекта.
Дата обнаружения дефекта	По меньшей мере, день/месяц/год; в этом поле может также быть указано время в часах и минутах.	Позволяет отслеживать хронологию событий, связанных с дефектом. Время в часах и минутах может оказаться полезным при отладке, если возникнет необходимость восстановить последовательность тестовых событий.
Обнаружен в сборке	Уникальный идентификатор сборки программного обеспечения, проходящего тестирование.	Этот показатель позволяет разработчикам выявить источник проблемы и дает возможность разработчику или тестировщику воспроизвести проблему.
Идентификатор	Уникальный идентификатор теста, во время прогона которого возникла проблема. Если это проблема была обнаружена во время специализированного тестирования, это поле может быть помечено как NA (not applicable – не применимое).	Снабжает разработчика и тестировщика сведениями о том, прогон какого теста выполнять, чтобы воспроизвести проблему. Тест, который обнаруживает проблему, должен стать частью регрессивного тестирования, предназначенного для того, чтобы убедиться, что дальнейшие программные сборки не нужно подвергать регрессивному тестированию.
Краткое описание проблемы	Описание проблемы на концептуальном уровне. Обычно такое описание дефекта умещается в одной строке.	Это описание используется в сообщениях о статусе дефекта и в отчетных докладах и адресовано руководству проекта и членам других групп, стремящихся к пониманию проблемы во всех деталях.
Описание проблемы	Детальное описание симптомов проблемы и того, как она ограничивает функциональные возможности или производительность системы.	Это описание предназначено для тех, кому требуется детальное понимание проблемы, например, разработчику, который работает над устранением дефекта, или руководителю, распределяющему ресурсы.

Характеристика	Описание	Обоснование
Степень серьезности дефекта	Упорядочивание дефектов по степени серьезности последствий. Примеры: **Катастрофический** – вызывает разрушение системы. **Крупный** – программный продукт не подлежит использованию. **Умеренный** – продукт можно использовать, однако имеют место некоторые неудобства для пользователя. **Незначительный** – пользователь не ощущает последствий. **Помеха** – можно устранить, когда позволит время.	Серьезность последствий часто определяет приоритеты усилий разработчиков при устранении дефектов и тестировщиков при проверке результатов этих исправлений.
Как воспроизвести проблему	Используется детально проработанная методика воспроизведения проблемы, включая выбор используемой конфигурации средств тестирования.	Детально проработанная методика позволяет разработчикам воспроизвести дефект и предоставляет возможность разработчикам и тестировщикам проверять результаты устранения проблемы.

Каждый дефект получает в базе данных отслеживания дефектов уникальный идентификатор, так называемый идентификатор дефекта. Идентификатор дефектов представляет собой ключ, который поддерживает запросы, поступающие в базу данных с таким расчетом, чтобы можно было получить информацию о его состоянии, которое можно рассматривать как показатель продвижения работ по его устранению. В базе данных отслеживания дефектов должны храниться сведения о том, кто обнаружил дефект, когда он был обнаружен, уровень серьезность дефекта, а также подробное описание проблем, которые он вызвал. Если дефект был обнаружен во время системных испытаний, должна быть записана информация, касающаяся теста, который выявил этот дефект. Для хорошо документированного тестового случая вполне достаточно запомнить только идентификатор теста. Если тест не задокументирован должным образом или если дефект был обнаружен в результате *специализированного* тестирования, необходимо записать информацию о конфигурации средств тестирования и о специальных действиях, выполненных для обнаружения этого дефекта.

Данные о дефекте, попадающие в базу данных отслеживания дефектов, должны быть изучены группой анализа обнаруженных дефектов (или группой контроля над внесением изменений) с целью определения

степени серьезности дефекта. На самом ли деле это дефект? Правильная ли степень серьезности была присвоена обнаруженному дефекту? Должен ли он быть устранен в текущем выпуске программного продукта? Информация, которая должна быть внесена в базу данных отслеживания дефектов по результатам работы группы анализа дефектов, показана в таблице 3. В результате анализа дефект переводится в одно из следующих трех состояний:

– **исправить (fix)** – дефект переходит в это состояние, если группа анализа решит, что данных дефект должен быть исправлен в текущем выпуске;

– **отложить (defer)** – дефект переводится в это состояние, если группа анализа решит отложить устранение этого дефекта до последующих выпусков программного продукта;

– **игнорировать (trash)** – дефект переходит в это состояние, если группа анализа решила, что дефект был внесен по ошибке.

В зависимости от итогового состояния дефекта, в базу данных отслеживания дефектов вводятся различные данные. Например, если дефект двигается в направлении состояния «исправить», следует ввести имя исполнителя, ответственного за устранение дефекта. Если информация относительно дефекта готовится для передачи заказчикам, должен быть введен текст пояснительной записки по выпуску.

Таблица 3 – Данные отслеживания дефекта в состояниях «исправить» (fix), «отложить» (defer) и «игнорировать» (trash)

Характеристика	Описание	Обоснование
Идентификатор дефекта (не изменяется)	Уникальный идентификатор дефекта	Позволяет отслеживать дефект в процессе изменения его состояний.
Состояние дефекта	Состояния «исправить», «отложить» и «игнорировать»	Присвоить дефекту состояние «исправить» в момент его отправки разработчикам на исправление; присвоить дефекту состояние «отложить», если он будет устранен в следующем выпуске программного продукта; присвоить дефекту состояние «игнорировать», если его нельзя воспроизвести или если проблема заключается не в дефекте, а в чем-то другом; дефект игнорируется также в случае, когда принято решение не исправлять его.
Кто исправил	Имя исполнителя, которому поручено устранить дефект.	Обеспечивает возможность задавать вопросы относительно исправления дефекта.
Кем отложен срок исправления	Имя ответственного лица, которое своим решением отодвинуло исправления	Обеспечивает возможность задавать вопросы относительно решения отложить исправление

Характеристика	Описание	Обоснование
	дефекта на более поздний срок (оставить поле пустым, если исправление дефекта не отложено).	дефекта на более поздний срок.
Кто проигнорировал дефект	Имя ответственного лица, которое приняло решение не исправлять дефект (оставить поле пустым, если исправление дефекта не откладывалось).	Обеспечивает возможность задавать вопросы относительно решения не исправлять дефект.
Дата начала исправления	День, месяц и год, когда устранение дефекта было поручено разработчику, ответственному за исправления.	Позволяет отслеживать хронологию событий, связанных с дефектом.
Дата, когда исправление было отложено	День, месяц и год, когда исправление было отложено.	Позволяет отслеживать хронологию событий, связанных с дефектом.
Дата, когда было принято решение не делать исправления.	День, месяц и год, когда было принято решение не делать исправления.	Позволяет отслеживать хронологию событий, связанных с дефектом.
Нужна пояснительная записка к выпуску (Y/N)?	Введите «Y», если нужна пояснительная записка к выпуску, и «N», если такая пояснительная записка не нужна.	Помечает дефект флагами таким образом, чтобы можно было составить список дефектов, для которых требуются пояснительные записки к выпуску.
Информация о выпуске программного продукта	Описание дефекта, которое будет передано заказчику как составная часть пояснительных записок, прилагаемых к выпуску (необходимы только в тех случаях, когда дефекты не исправлены в конкретном выпуске).	

Как составлять сообщения о дефектах

Каждый раз, когда обнаруживается новый дефект, должно составляться письменное сообщение об этом дефекте. Если информация о дефекте не будет получена достаточно оперативно, могут возникнуть трудности при восстановлении обстоятельств, которые привели к

возникновению проблемы. Это тем более справедливо, если применяются специализированные виды тестирования. Если выполняется прогон документированного тестового случая, то его исход может быть обозначен как неудачный, а отличия фактических результатов тестов от ожидаемых могут быть зарегистрированы как составная часть прогона теста. Как только специализированный вид тестирования обнаруживает проблему или, как только прогон теста завершен, должно быть составлено сообщение о дефекте.

Пример шаблона сообщения о дефекте показан на рисунке 3. Этот шаблон может быть представлен в форме печатного бланка, который вручную заполняется тестировщиком, в виде шаблона ввода данных в Web-формате или как часть базы данных системы отслеживания дефектов. Такой шаблон сообщения о дефекте представляет собой средство сбора тестовых данных, которые отображаются в таблице 2.

От умения специалистов по тестированию составить толковое сообщение о дефекте в значительной мере зависит успех их деятельности. Часто время, затрачиваемое тестировщиками на изучение сообщений о дефектах, которые были составлены их организациями во время разработки аналогичных проектов, приносит большую пользу. В идеальном случае имеется набор «золотых примеров», который дает возможность тестировщикам, не имеющим достаточного опыта, получить понимание того, что такое высококачественное сообщение о дефекте и какие факторы делают такое сообщение неэффективным.

Вообще говоря, сообщение о дефекте считается плохо составленным или неэффективным, если:

– оно составлено по ошибке – дефект, описанный в сообщении, не существует;

– возникшая проблема представлена в описании нечетко или неоднозначно – проблема, возможно, и существует, но в то же время она описана настолько плохо, что поведение системы не ясно;

– оно не содержит информации, необходимой разработчику для воссоздания проблемы – признаки проблемы описаны, но при этом отсутствуют пояснения к действиям, которые послужили причиной возникновения проблемы.

Сообщение о дефекте
Идентификатор дефекта:
Кем обнаружен:
Дата обнаружения:
Обнаружен в сборке:
Степень серьезности дефекта (катастрофический, крупный, незначительный):
Идентификатор теста:
Краткое описание проблемы:
Подробное описание проблемы:
Как воспроизвести проблему: (Если был использован недокументированный тест, подробно опишите методику тестирования. При необходимости дайте схему конфигурации средств тестирования.)

Рисунок 3 – Шаблон сообщения об ошибке

Правильно составленное сообщение о дефекте обладает совершенно другими свойствами:

– оно описывает фактический дефект программного продукта;

– в нем четко описаны признаки проблемы через отклонения поведения системы от нормы;

– оно содержит процедуру пошагового воспроизведения проблемы.

Анализ обнаруженных дефектов

Если бы процесс обнаружения и исправления дефектов был совершенным, то необходимость в анализе дефектов отпала бы сама по себе. К сожалению, человеческий фактор вступает в действие уже на стадии системных испытаний, впрочем, как и на других этапах цикла разработки. Ошибки могут неожиданно обнаружиться в тестовых случаях, в конфигурациях средств тестирования или в интерпретации данных тестирования. Когда обнаружен дефект, необходимо проследить за тем, чтобы своевременно был назначен исполнитель, ответственный за его устранение, что обеспечит исправление дефекта и проверку результатов этого исправления.

Один из способов проверки результатов тестирования и предотвращения возникновения ошибок при отслеживании дефектов предусматривает еженедельный анализ дефектов, обнаруженных во время системных испытаний. Назначение анализа дефектов состоит в том, чтобы:

– выявить все очень серьезные дефекты, требующие немедленного внимания;

– выявить все дефекты, которые требуют более глубокого исследования или не могут быть воспроизведены;

– определить изменения состояний дефектов.

Анализ дефектов проводится не так, как пересмотр программных кодов или других рабочих продуктов. В большинстве случаев экономически нецелесообразно вести учет времени, затраченного исполнителями на подготовку к анализу, к тому же роли участников не расписаны так же четко, как в случае перепроверки программных кодов и тестовых случаев. Тем не менее, полезно выработать подходящую процедуру проведения анализа дефектов. Часто в испытательных лабораториях совещания, посвященные анализу дефектов, проводят ведущие инженеры, при этом дефекты, требующие анализа, представлены в рамках отчетного доклада, который можно рассматривать как высокоуровневую информацию о дефектах. Пример формата отчетного доклада, посвященного дефектам, показан на рисунке 4.

Анализ дефектов проекта

Дата анализа: день/месяц/год

Участники:

Идентификатор дефекта	**Состояние**	**Серьезность дефекта**	**Дата обнаружения**	**Краткое описание дефекта**	**Действие**
B1233	Новый	Катастрофический	День/месяц/год	Нарушение функций резервирования/восстановления - невозможен поиск данных после резервирования.	Исправить
B1234	Новый	Крупный	День/месяц/год	Нельзя открыть резервный экран из меню; обходной путь - использование командной строки.	Исправить
B1235	Новый	Незначительный	День/месяц/год	Проблема удобства использования - не работает вызов функции поиска через клавишу ускоренного доступа.	Отложить

Примечания:

B1235 — эта проблема требует изменений в архитектуре программного продукта, в силу чего ее решение откладывается до следующей версии.

Рисунок 4 – Пример отчета по результатам анализа дефектов

В примере сообщения на рисунке 4 приводится список всех дефектов с указанием их идентификаторов, а также содержится информация об их состоянии, степени серьезности и дате обнаружения. Дается краткое описание проблем, но если потребуется подробное описание конкретного дефекта, возникает потребность в первоначальном сообщении о дефекте. Одним из ключевых полей в отчете является столбец «Действие». В поле

предполагаемого действия можно определить какое-то значение еще до того, как будет выполнен анализ, однако основной результат совещания заключается в определении конкретных состояний, которые должны быть указаны в этом столбце.

В сообщении, приведенном на рисунке 4, резервируется место для ввода данных анализа, для перечисления участников, а также поле «Примечания», которое может использоваться для документирования обоснования решений, принятых на совещании. Если по результатам анализа все поля заполняются корректно, итоговый отчет послужит источником для того, чтобы расписать совещание по минутам.

Прогон тестов

До сих пор основное внимание уделялось дефектам, ибо обнаружение дефектов является основной причиной для прогона тестов. Далее сосредоточимся на изучении самого процесса тестирования, следуя обзорной диаграмме, которая показана на рисунке 1.

Вход в системные испытания

При передаче группе тестирования новой сборки применяется некоторый набор критериев входа в испытания. Перед началом системных испытаний должен быть разработан и утвержден план проведения испытаний и тестовые случаи. Группа разработки должна закончить тестирование модулей и проверку взаимодействия и функционирования компонентов этой сборки. Они должны исправить все катастрофические дефекты, обнаруженные во время испытаний, и только затем передавать сборку группе тестирования.

Первый тест, прогоняемый на новой сборке, – это обычно тест «на герметичность», который служит для проверки возможности установки программы на целевой платформе, возможности успешного обновления и определения факта работоспособности, по меньшей мере, базовых функций. Причина проведения теста на герметичность довольно проста: если программа не работает на целевой платформе, ее просто невозможно тестировать.

К сожалению, нередки случаи, когда новая сборка не проходит испытания на герметичность, особенно если перед ее передачей тестировщикам проверка взаимодействия и функционирования компонентов была выполнена в недостаточных объемах. Разработчики могли настолько сосредоточиться на том, чтобы заданные функции успешно работали в среде разработки, что до проблем системного уровня, таким как, скажем, установка программы и ее работа в среде заказчика, просто не доходили руки.

Циклы тестирования

Вход в системное тестирование должно означать начало совместных работ при участии групп разработчиков и тестировщиков. Тестировщики выполняют прогон запланированных тестов, выявляют дефекты и пишут сообщения о дефектах. Разработчики читают сообщения о дефектах,

воспроизводят проблемы и исправляют программный код. Вопрос заключается вот в чем: как исправления передаются в группу тестирования?

Ответ на этот вопрос зависит от цикла создания программного обеспечения. Разработчики регистрируют проделанные исправления в системе управления конфигурациями в рамках подготовки к периодическим построениям сборок. Построение сборки обычно выполняется на базе ежедневного или еженедельного цикла, после чего блок передается группе тестирования в соответствии с их потребностями.

Возникают различные вопросы относительно того, как часто группа тестирования может принимать новые сборки. Если сборки приходят слишком часто, устойчивость среды тестирования может оказаться нарушенной – потребуется время на установку новых сборок, прогон тестов на герметичность и проверку того, что дефекты, которые были обнаружены в предыдущей версии сборки, исправлены. Если «смесь сборок» будет излишне «пестрой», то попросту не хватит времени на прогон достаточного количества тестов, дабы обеспечить эффективное обнаружение дефектов в соответствии с выбранной методологией тестирования.

С другой стороны, если сборки поступают слишком медленно, то и в этом случае нельзя достичь высокой эффективности обнаружения дефектов. Довольно часто изменения, вносимые в программный код, приводят к тому, что обнаруживаются новые дефекты, либо выявляются дефекты, которые и раньше были в сборке, но были замаскированы теперь уже устраненными дефектами. Это значит, что включать новые сборки в цикл тестирования нужно достаточно часто, чтобы можно было обнаружить следующую партию дефектов.

То, как часто доводится принимать новые сборки от разработчиков, зависит от нескольких факторов, в том числе и от сложности программного обеспечения, от численности штата тестировщиков, от процентного отношения автоматизированных тестов к общему числу тестов. Например, если можно прогнать все тесты в течение трех дней, вероятно, имеет смысл принимать новые сборки каждые четыре-пять дней. В этом случае три дня можно отвести на прогон тестов, а день или около того – на проверку исправлений дефектов, на прогон специализированных тестов в некоторых многообещающих областях, на подготовку отчетов об устранении дефектов и на регулярный анализ дефектов.

В идеальном случае *цикл тестирования (test cycle)* состоит из прогона полного набора тестов на некоторой сборке программного обеспечения. План проведения испытаний обычно предусматривает выполнение некоторого количества циклов тестирования, причем на каждый цикл затрачивается определенное время. Например, этот план может предусматривать выполнение трех или четырех циклов длительностью в одну или две недели каждый.

На практике тестирование не всегда проводится с соблюдением плана проведения испытаний. Первый цикл тестирования может продолжаться одну-две недели, однако может потребоваться большое число сборок, чтобы

тестирование оказалось достаточно устойчивым, и можно было прогнать большую часть тестов. Когда очередная новая сборка поступает на тестирование, принимается решение, нужно ли выполнить повторный прогон всех тестов с самого начали или продолжать тестирование в прежней последовательности. Обычно более эффективной оказывается выборочная проверка новой сборки, которая дает возможность убедиться в том, что ее можно остановить и запустить в работу, а обнаруженные ранее дефекты уже устранены. После этого тестирование продолжается без повторного прогона всех тестов. Подходящим моментом для очередного запуска тестирования может служить начало нового тестового цикла. Это означает, что может произойти так, что на промежуточной сборке не будет выполнен полный набор тестов. Исключением из этой стратегии является завершающая сборка программного обеспечения, так называемая «золотая сборка», на которой прогоняются все тесты.

Один из способов отслеживания тестов, прогон которых выполнен на заданной сборке, предусматривает ведение списка прогона тестов на сборке для каждой сборки тестируемого программного обеспечения. Пример списка прогона тестов на сборке показан на рисунке 5. В этом списке в качестве заголовка указан идентификатор сборки и дата прогона теста; список прогона тестов, по существу, представляет собой список того, «что нужно сделать» каждому специалисту по тестированию. Если планируется применение испытательных комплексов или рабочих станций коллективного пользования, этот список позволит избежать конфликтов, если в нем для каждого специалиста по тестированию будет указываться испытательный комплекс, на котором должны запускаться тесты. Если же возникнут конфликты требований на использование испытательного комплекса, то в таких случаях может потребоваться разработка рабочего графика для комплексов. В списке прогона тестов на сборке один столбец резервируется под краткое изложение результатов тестирования.

Список прогона тестов на сборке				
Идентификатор сборки:				
Дата начала тестирования:				
Номер теста	**Идентификатор теста**	**Имя тести-ровщика**	**Конфигурация средств тестирования**	**Результат (P/F/NR)**
1.	**A10**	JimD.	1A на системе 1	Pass (тест прошел)
2.	**A11**	Jim D.	1A на системе 1	Pass (тест прошел)
3.	**B11**	BobZ.	2B на системе 2	Fail (тест не прошел)
4.	**B11**	BobZ.	2B на системе 2	Not run (тест не выполнялся)

Рисунок 5 – Пример списка прогона тестов на сборке

Список прогона тестов на сборке может оказаться исключительно полезным при частой передаче сборок в группу тестирования. На основе

информации о тестах, прогон которых выполнялся на предыдущих сборках, список можно составить так, чтобы обеспечить эффективное покрытие тестами свойств программного продукта на некоторой последовательности сборок. Список прогона тестов на сборке помогает также приспосабливаться к ситуации, когда в заданной сборке присутствуют дефекты, которые не позволяют обеспечить покрытие конкретной области программного кода. Как только блокирующий дефект будет исправлен, можно сосредоточить свое внимание на этой области и обеспечить высокую степень покрытия.

Регистрация результатов тестирования

Когда тестировщик получает задание прогнать некоторую последовательность тестов на конкретной сборке, он должен следовать пошаговой методике тестирования, оговоренной в тестовом случае, и протоколировать результаты тестирования. Как видно из примера тестового случая, приведенного на рисунке 2, в нем отводится специальное место, куда тестировщик заносит результаты выполнения каждого шага, равно как и исход испытания (прошел, не прошел). Кроме того, выделяется также и место, куда заносится идентификатор любого дефекта, обнаруженного во время испытаний. Один из способов регистрации результатов прогона тестов предусматривает использование формата тестового случая, который показан на рисунке 2.

Другой способ заключается в применении журнала испытаний, изображенного на рисунке 6. Этот журнал испытаний удобен для сведения воедино результатов всех тестов, выполненных одним специалистом по тестированию на одной сборке. В нем также есть место для регистрации общего результата «прошел/не прошел/тест не выполнялся», а также для идентификатора любого дефекта, обнаруженного во время прогона теста. Предусматривается также место для комментариев – обычно в нем протоколируется информация, которая может впоследствии оказаться полезной при воспроизведении проблемы или для понимания условий, в которых проблема наблюдалась.

Независимо от того, какой метод будет использоваться для регистрации результатов тестирования, очень важно, чтобы результат каждого теста был зарегистрирован и сохранен в безопасном архиве. Заархивированные результаты тестирования могут позже понадобиться в силу различных причин: подтверждение факта проведения тестирования, поддержка воспроизведения проблемы или хронологические записи, обосновывающие трудозатраты на разработку программного продукта.

Шаблоны списка прогона тестов на сборке, тестовых случаев и журналов испытаний приводятся в данной книге в форме текстовых документов, тем не менее, их можно реализовать и как часть системы ввода данных через Web либо инструментальных средств организации испытаний. Использование автоматизированных средств для сохранения тестовых случаев, прогона тестов и генерации отчетов по испытаниям может существенно повысить качество построения эффективной стратегии системных испытаний.

Журнал испытаний				
Дата: день/месяц/год				
Цикл тестирования: 1				
Идентификатор сборки: B020202				
Конфигурация средств тестирования: 1A				
Тестирование выполнял: Jim D.				
Тест №	**Идентификатор теста**	**Исход Pass/Fail/Not run**	**Идентификатор дефекта**	**Комментарий**
1.	**A10**	**P**		
2.	A11	**P**		
3.	**A12**	**P**		
4.	**A13**	**F**	XY1233	Эта проблема характерна для конфигурации средств тестирования.
5.	**A14**	**P**		
6.	**A15**	**F**	XY1234	
7.	**B1**	**NR**		Тестовый набор заблокирован дефектом XY1234.
8.	**B2**	**NR**		Заблокирован
9.	**B3**	**NR**		Заблокирован
10.	**B4**	**NR**		Заблокирован

Рисунок 6 – Журнал испытаний

Составление отчетов по результатам тестирования

Обычно требуются два вида отчетов по результатам тестирования. Первый из них – это регулярный доклад о ходе тестирования, который часто делается на еженедельных совещаниях. На таких совещаниях представители всех коллективов, ответственных за успех проекта, отчитываются о проделанной работе. Доклад о ходе тестирования обычно состоит из отчета о ходе работ по тестированию и отчетного доклада об обнаруженных дефектах. Доклад о ходе работ должен дать ответ на вопрос «насколько далеко мы продвинулись в тестировании?». Отчетный доклад об обнаруженных дефектах предназначен для выяснения степени успешности тестовых мероприятий в контексте обнаружения дефектов. Одна из его целей состоит в получении текущей оценки качества тестируемого программного продукта.

Второй тип доклада по результатам тестирования представляет собой формальную сводку результатов тестирования, которая составляется по окончании системных испытаний. Доклад второго типа предназначен для документирования результатов тестирования с тем, чтобы в будущем можно было получить ответ на вопросы о том, что подвергалось тестированию,

какие были получены результаты, и какого мнения придерживалась группа тестирования относительно готовности продукта к выпуску.

Отчет о ходе работ по тестированию

Отчет, показанный на рисунке 7, дает представление о ходе работ по тестированию. Из информации о дате, идентификаторе сборки и цикле тестирования видно, какой программный продукт проходит тестирование и приблизительно насколько проводимое тестирование отклоняется от плана. Последний столбец отчета, % Run («процент прогонов»), есть мера того, сколько прогонов тестов было выполнено относительно общего количества запланированных тестов. Число тестов, прогон которых завершился неудачно, указывается в столбце «# Fail» (тест не прошел) – этот показатель дает приближенные данные о том, сколько дефектов будет зафиксировано в проверяемом программном продукте. Столбец «# Not Run» (тест не выполнялся) показывает, сколько тестов заблокировано по причине неустраненных дефектов или других проблем. Общее количество тестов приводится в столбце «# Tests», а количество успешно пройденных тестов – в столбце «# Pass».

Отчет о ходе работ по тестированию					
Дата составления отчета: день/месяц/год					
Идентификатор сборки: B123					
Цикл тестирования: 2/3					
Дата начала тестовых работ: месяц / день / год					
Тестовый набор	**# Tests**	**#Pass**	**# Fail**	**# Not Run**	**% Run**
Ввод данных	20	12	2	6	70%
Редактирование	15	10	4	1	93%
Резервирование и восстановление	25	12	6	7	72%
Обмен данными	18	8	8	2	89%
Итого	78	42	20	16	79%

Рисунок 7 – Пример отчета о ходе работ по тестированию

Отчет об устранении дефектов

Другим видом анализа положения дел в тестировании является отчет об устранении дефектов, пример которого представлен на рисунке 8. В этом отчете показано общее число неустраненных дефектов как функция от времени в неделях. Временная ось часто градуируется в датах, а не в «неделя первая», «неделя вторая» и т.д. Количество дефектов заданной серьезности (катастрофические, крупные и незначительные) в этом отчете выглядит как столбик соответствующей высоты в выбранном масштабе, а также указывается явно в числовой форме. Например, на третьей и четвертой

неделе тестирования в программном продукте было зафиксировано четыре катастрофических дефекта, а на второй неделе – 32 незначительных дефекта.

Общее количество неустраненных дефектов есть показатель качества программного продукта. Вполне понятно, что большое количество дефектов высокого уровня серьезности не говорит в пользу качества программного продукта. Окончательным определителем качества программного продукта является то, насколько он устраивает заказчика. Разумеется, заказчик вряд ли будет доволен, если в предоставленной ему версии продукта присутствует слишком много дефектов. Иногда имеются смягчающие вину обстоятельства, когда наиболее важной является скорость поставки продукта, а заказчик согласен даже на версию с дефектами, лишь бы что-то работало.

По мере того как работы по тестированию приближаются к завершению, есть все основания ожидать, что число неустраненных дефектов будет уменьшаться, поскольку разработчики исправляют их, а тестировщики проверяют, насколько правильно были выполнены исправления. Степень серьезности дефектов по мере продвижения тестирования также имеет тенденцию к снижению. Пример, представленный на рисунке 8, необычен тем, что количество неустраненных катастрофических и крупных дефектов остается практически на одном уровне, в то время как число незначительных дефектов существенно снижается. Более типичным случаем является быстрое устранение дефектов с высоким уровнем серьезности, в то время как менее серьезные дефекты остаются не устраненными в течение более длительного времени; возможно, даже, что их устранение откладывается до будущих выпусков.

Рисунок 8 – Пример отчета об устранении дефектов

Другой отчет, который часто используется при анализе состояния проекта, – отчет об анализе дефектов, представленный в таблице 5. Диаграмма на рисунке 4 достаточно точно характеризует ход тестовых работ и служит показателем качества программного продукта, однако все решения относительно того, исправления каких дефектов будут включаться в выпуск, и в каком порядке дефекты будут устраняться, требуют более подробной информации о не устранённых дефектах. В некоторых компаниях функции

анализа проекта и анализа дефектов объединены, причем оба вида анализа регулярно ставятся на повестку дня заседаний группы контроля над внесением изменений CCB (change control board). Как правило, в состав группы CCB входят представители руководства, отдела маркетинга, организаций, которые занимаются разработкой и тестированием.

Отчетный доклад

В отчетном докладе должны быть отражены следующие моменты:

– какой программный продукт тестировался;

– насколько фактические работы по тестированию отклонились от плана проведения испытаний;

– насколько график работ и трудозатраты отличаются от соответствующих показателей в плане проведения испытаний;

– какие дефекты были обнаружены;

– какие дефекты остались не устранёнными по завершении тестовых работ и что с ними делать дальше.

Отчетный доклад по результатам тестирования не обязательно должен содержать результаты прогона каждого теста, в то же время он может ссылаться на такие результаты, если они где-то накапливаются и архивируются. Сбор и архивирование всех тестовых результатов — это одно из преимуществ, которое дает использование серийных инструментальных средств управления тестированием. Если это инструментальное средство сводит результаты тестирования в единый документ, то на этот документ можно сослаться в отчетном докладе.

Один из способов указать, какой продукт подвергается тестированию, предусматривает использование набора окончательных вариантов журналов испытаний (рисунок 6) на каждом цикле испытаний. Этот способ позволяет показывать результаты прогона каждого теста без использования пошагового отчета. Другой полезный отчет можно получить за счет компиляции полного набора данных о ходе работ по тестированию (рисунок 7).

Важно понять, насколько фактический процесс тестирования отклоняется от плана проведения испытаний, поскольку план проведения испытаний представляет собой утвержденное определение объемов тестирования и соглашение о том, сколько сотрудников принимают участие в этих работах, и сколько времени будет затрачено на выполнение работ. Особенно важно отметить, какие области не были охвачены тестированием в полном объеме, поскольку эти области являются источниками рисков снижения качества программного продукта и источниками распространения дефектов в различные части продукта.

Несмотря на то, что база данных отслеживания дефектов должна содержать полный набор данных о дефектах, обнаруженных во время тестирования программного продукта, сводку дефектов, обнаруженных во время системных испытаний, целесообразно включить в отчетный доклад. В таком случае всем сторонам, заинтересованным в успехе разработки, не потребуется обращаться в базу данных дефектов за сведениями о ходе работ

по тестированию и о качестве программного продукта. Окончательная версия отчета по результатам анализа дефектов (рисунок 4), в котором показаны все обнаруженные дефекты и их окончательные состояния, а также окончательная редакция отчета о категориях не устранённых дефектов (рисунок 8) являются полезными дополнениями отчетного доклада о тестировании.

Отчетный доклад о результатах тестирования должен содержать список всех не устранённых дефектов с указанием того, будут ли они исправлены в более поздних выпусках или же их исправление откладывается на неопределенное время. Дальнейшие выпуски программного продукта все еще будут содержать не устранённые дефекты до тех пор, пока они не будут исправлены. В силу этого обстоятельства желательно вести за такими дефектами наблюдение, чтобы не тратить дополнительные усилия на их выявление в будущих выпусках продукта. В плане проведения испытаний будущих выпусков должны присутствовать ссылки на список не устранённых дефектов; это позволит тестировщикам понять, что их ожидает во время тестирования нового выпуска программного продукта.

Критерий выхода из испытаний и готовность выпуска программного продукта

Существует множество способов определить подходящий момент для останова испытаний, одни из них простые, другие же – очень сложные. Вот некоторые из условий, которые используются для принятия решения о прекращении испытаний:

– время, отведенное на тестирование, истекло. Если зафиксирован некоторый предельный срок, неизбежно наступает день, когда вы просто прекращаете все работы, связанные с тестированием, и задаете себе вопрос: «Насколько плох программный продукт?». Любой другой метод останова придает большую уверенность в качестве поставляемого программного продукта;

– завершены все запланированные циклы тестирования. Если план проведения испытаний предусматривает три цикла, то в конце третьего цикла вы, по-видимому, зададите тот же вопрос: «Насколько плох программный продукт?» Если план проведения испытаний выполнен, то тестовое покрытие, скорее всего, будет лучше, чем в случае, когда просто не хватило времени для доведения тестирования до конца;

– **профиль дефектов соответствует критерию выхода из испытаний.** Если в результате выполнения алгоритма SWEEP (Software Error Estimation Program – программа оценки ошибочности программного продукта) становится ясно, что предел, т.е. количество неустраненных ошибок на тысячу строк программного кода, достигнут, появляются все основания прекратить тестирование. В подобной ситуации опять-таки придется задаться вопросом: «Насколько плох программный продукт?».

Принимая решение прекратить работы по тестированию, следует учесть несколько факторов, которые играют важную роль при оценке

готовности программного продукта к поставкам. Обычным набором факторов, которые определяют готовность программного продукта к поставкам, являются:

– количество катастрофических дефектов, обнаруженных в процессе тестирования, которые остаются неисправленными;

– общее количество дефектов, которые не были исправлены;

– процентное отношение количества тестов, которые завершились успешным исходом, к числу всех запланированных тестов;

– количество тестов, прогон которых не может быть выполнен, поскольку они заблокированы дефектами.

С целью выработки оценки готовности выпуска отлаженного программного продукта проводятся специальные совещания. В некоторых организациях оценку готовности определяет группа тестирования; в других организациях совещание проводит руководитель проекта либо руководитель разработки. В любом случае в задачу группы тестирования входит представление результатов тестирования и выработка рекомендаций относительно готовности продукта к поставкам. Хотя оценка готовности может проводиться формально, подобно тому, как выполняется инспекция программного кода, в любом случае потребуется зафиксировать имена участников и принятое ими решение, касающееся поставок программного продукта. Чтобы оценка готовности содержала результаты и рекомендации организации, проводившей тестирование, эта оценка должна содержать ссылку на отчетный доклад по результатам тестирования.

Практическая часть

Содержание задания

Составить сводный отчет по системным испытаниям на основании разработанных тестов для программного продукта из предыдущей лабораторной работы. Пример сводного отчета по системным испытаниям приведен в книге «Быстрое тестирование», страницы 368 – 376 (книга прикреплена к заданию вместе с методическими указаниями).

Порядок выполнения работы

1 Ознакомиться с теоретической частью.

2 Провести испытания программного продукта (используя тесты из предыдущей лабораторной работы).

3 Составить сводный отчет по системным испытаниям (**обязательными являются ВСЕ пункты указанные ниже, в содержании документа**).

4 Показать работу преподавателю.

5 После защиты, выполненного задания, скинуть отчёт преподавателю (в **названии** указать **фамилию и номер лабораторной**, например, Ivanov11).

Содержание отчета

1 Титульный лист (**на титульном листе вместо варианта указать название тестируемого программного продукта**).

2 Введение.

3 Сводка по результатам тестирования.

4 Свойства, которые должны тестироваться.

5 Свойства, которые не должны тестироваться.

6 Отчет по качеству программного продукта.

7 Вывод о проделанной работе.

Защита работ проводится индивидуально.

Контрольные вопросы

1 Какие виды деятельности считаются основными для выявления дефектов?

2 Какие данные о дефекте в состоянии «Новый» должны храниться?

3 Какие данные о дефекте в состоянии «Исправить» должны храниться?

4 В каком случае дефекту можно присвоить состояние «Игнорировать»?

5 Какое сообщение о дефекте считается плохо составленным?

6 Какого назначение анализа дефектов?