Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий
Кафедра технологий программирования

Методические указания
к выполнению лабораторной работы №2

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

на тему «**Дефекты. Описание дефектов. Определение критичности и приоритетности дефекта**»

Новополоцк, 2017 г.

Название: «Дефекты. Описание дефектов. Определение критичности и приоритетности дефекта».

Цель работы: Изучить понятие «дефект», изучить и научиться различать понятия «критичность» и «приоритетность» дефектов, научиться описывать дефекты, найденные в приложении.

Теоретическая часть

Дефект (жарг. – баг) – ни что иное, как изъян в разработке программного продукта, который вызывает несоответствие ожидаемых результатов выполнения программного продукта и фактически полученных результатов. В целом, разработчики различают дефекты программного обеспечения и сбои. В случае сбоя программа ведёт себя не так, как ожидает пользователь. Дефект – это ошибка/неточность, которая может быть (а может и не быть) следствием сбоя.

Если подводить итог, то дефект – это поведение программы, затрудняющее или делающее невозможным достижение целей пользователя или удовлетворение интересов участников. Подразумевает возможность исправления. При невозможности исправления переходит в разряд ограничения технологии.

При обнаружении дефекта у тестировщика ПО (программного обеспечения) встает задача **правильного описания дефекта** для того, чтобы разрабатывающий данное ПО, программист мог также его обнаружить и исправить.

Описание дефектов

В различных IT-компаниях, где есть отдел тестировщиков, описание дефектов происходит фактически одинаково, иногда формы описания дефектов отличаются парой мелочей. Для примера рассмотрим форму описания дефектов, которая используется в компании A1QA.

Описание дефекта состоит из нескольких пунктов:

1 Суть проблемы (**Headline**).

2 Степень критичности дефекта (**Severity**).

3 Алгоритм воспроизведения дефекта (**Description**).

4 Ожидаемое поведение в описанной ситуации (**Expected result**).

5 Вспомогательное средство передачи информации о проблеме (**Attachment**).

6 Степень критичности дефекта (**Priority**).

Подробное разъяснение пунктов описания дефектов.

1 Суть проблемы (Headline)

Цель составления **Headline** состоит в том, чтобы предоставить проектному менеджеру и разработчику максимально понятную информацию о том, что произошло, где, в результате чего, и насколько эта проблема критична для проекта. Должен отображать общую суть дефекта.

У хорошо оформленного **Headline** есть определенные характеристики:

– краткость (не длиннее поле заголовка для удобства чтения);

– информативность;
– точная идентификация проблемы;

Способы написания эффективного Headline:

Способ №1:

1 Составьте описание дефекта.

2 Посмотрите на него внимательно и выделите из него ключевые моменты в тексте.

3 Попробуйте сложить эти ключевые моменты вместе.

4 То, что получится при складывании ключевых моментов, и есть заголовок дефекта.

Способ №2:

Второй способ является более грамотным и рекомендуется к применению всеми тестировщиками. Этот подход называется «Где? Что? Когда?». Согласно нему, заголовок дефекта представляет собой предложение или высказывание, в котором факты дефекта изложены в следующей последовательности:

– где?: в каком месте интерфейса пользователя или архитектуры программного продукта находится проблема. Причем начинать предложение нужно с существительного, а не предлога;

– что?: что происходит или не происходит согласно спецификации или вашему представлению о нормальной работе программного продукта. При этом необходимо указывать на наличие или отсутствие объекта проблемы, а не на его содержание (его указывают в описании). Если содержание проблемы варьируется, все известные варианты указываются в описании;

– когда?: в какой момент работы программного продукта или по наступлению, какого события проблема проявляется.

2 Description - алгоритм воспроизведения дефекта

Цель составления - описать шаги для повторения дефекта.

Грамотный Description должен описывать алгоритм воспроизведения дефекта, по определенной структуре. В общем виде структура выглядит следующим образом:

1 Step #1
2 Step #2
3 ...

Result:

В случае, если для воспроизведения дефекта требуется ряд начальных условий – они должны быть вынесены в самое начало описания, что изменит структуру до следующего шаблона:

Preconditions:
1 Step #1
2 Step #2

Steps to reproduce:
1 Step #3
2 Step #4
3 ...

Result:

Правила оформления description

При описании алгоритма повторения дефекта необходимо:

– начинать с того, под каким пользователем вы тестируете (если это важно);

– добавлять логи ошибок, если это возможно (только формат txt);

– использование фразы «See attachment» должно сопровождаться краткими дополнительными комментариями;

– описание предложений по улучшению должно быть максимально полным и аргументированным, должно предлагать адекватное решение по реализации, желательно одно без выбора вариантов;

– можно использовать специальные символы «+», «=», *«o»*, «->» которые помогут сделать подобие навигации: File -> Open, DOC + XLS. Однако при описании шагов воспроизведения дефекта не рекомендуется писать шаги в строку через символы перехода – это затрудняет восприятие дефекта;

– без необходимости не рекомендуется перегружать описание дефекта постоянно повторяемыми общими действиями как то запуск проекта, в деталях описать создание необходимых для воспроизведения дефекта данных и т.п.

Важно: Description не должен выглядеть как рассказ. Это четкий последовательный алгоритм, в котором приветствуются короткие, понятные фразы и нумерация.

3 Expected result – ожидаемое поведение

Цель написания **Expected result** – показать разработчикам, как именно должен быть исправлен дефект с точки зрения тестировщика. Грамотно составленный Expected result исключает любые двойные толкования в способах исправления дефекта, что облегчает процедуру проверки ошибки после ее исправления.

Правила оформления

Параметр Expected result аналогично другим параметрам дефекта имеет ряд требований, которые должен соблюдать тестировщик при описании дефекта:

В Expected result должно быть четкое обоснование, почему именно так, а не иначе должен быть исправлен дефект. Лучше всего, если в нем приведена ссылка на конкретный пункт спецификации. Если функция работает, но работает некорректно, то в Expected Result обязательно должно быть описание того, как она должна работать корректно. Если сделана ошибка в надписи или интерфейсе проекта, лучше грамотно и полностью указать, как она должна быть исправлена. Текст Expected result рекомендуется писать законченными полными безличными предложениями.

Expected result допускается не писать только в том случае, если очевидно, что ожидаемый результат - отсутствие того результата, который сейчас наблюдается.

4 Attachment – вспомогательное средство передачи информации о проблеме

Attachment – это любой прикрепленный к дефекту файл, облегчающий его понимание либо дополняющий описание дефекта: снимок экрана (скриншот), тестовые файлы, файлы лога проекта, видеозаписи ошибки и т.п.

Если к дефекту прикрепляется файл, об этом обязательно должно быть указано в описании дефекта. Прикрепленный файл не должен быть слишком большим по размеру, а также должен относиться именно к описанной ошибке.

5 Классификация уровней критичности дефекта

Взаимосвязь между критичностью и приоритетом дефектов.

В поле Severity тестировщики выставляют критичность дефекта с точки зрения конечного пользователя. На основании уровня критичности, указанного в поле Severity, а также информации из других источников, менеджеры проектов определяют срочность исправления дефекта, т.е. назначают приоритет.

Тестировщик заполняет поле Severity при внесении дефекта в систему. Затем менеджер проекта читает описание дефекта и заполняет поле Priority.

Критичность не влияет напрямую на порядок исправления дефектов. Очередность исправления в первую очередь зависит от приоритета. Критичность – это не единственный фактор, который влияет на приоритет. Описание уровней критичности дефектов представлено в таблице 1. Менеджер проекта может выставить приоритет Resolve immediately (исправить немедленно) незначительному дефекту, а для серьезного дефекта выбрать значение Low priority (низкий приоритет), в зависимости от требований заказчика.

Таблица 1 – Уровни критичности дефектов

Уровень критичности	**Описание дефекта**
Blocker (Bugzilla)	Дефект полностью блокирует работу приложения. Продолжать тестирование при наличии такого дефекта невозможно.
Critical (CQ, Bugzilla, Jira)	Дефект полностью (CQ, Jira) или частично (Bugzilla) блокирует работу приложения. Продолжать тестирование при наличии такого дефекта невозможно.

Уровень критичности	Описание дефекта
Major (CQ, Bugzilla, Jira)	Дефект нарушает нормальную работу одной или нескольких функций приложения, но не препятствует дальнейшему проведению тестов.
Average (CQ, реализовано на корпоративной JIRA) Normal (Bugzilla, Jira)	Дефект частично влияет на основные функции приложения, но выполнение сценария в ходе тестирования возможно при минимальных изменениях. Графический дефект, значительно влияющий на восприятие проекта пользователями.
Minor (CQ, Bugzilla, Jira)	Несущественная функциональная ошибка (CQ, Bugzilla, Jira) или дефект графического интерфейса (CQ, Jira). Исправление незначительно улучшит поведение или выполнение сценария. Тестирование проводится согласно сценарию без каких-либо изменений.
Trivial (Bugzilla)	Дефект графического интерфейса, не вызывающий значительного ухудшения восприятия проекта пользователями.
Enhancement (CQ, Bugzilla, Jira)	Мелкий дефект, не требующий обязательного исправления, или рекомендация, не предполагающая обязательного внесения изменений. Тестирование проводится согласно сценарию без каких-либо изменений.

Критерии определения критичности

В данном разделе описываются критерии, на которые ориентируется тестировщик при определении критичности дефекта. Критерии являются частью стандарта, которого тестировщики должны придерживаться. Наличие критериев гарантирует, что процесс определения критичности дефектов соответствует требованиям компании.

Помимо указанных критериев, при определении уровня критичности дефектов тестировщику следует ориентироваться на целевую аудиторию проекта и сценарии его использования. Достаточно представить себя конечным пользователем проекта и определение уровня критичности облегчится – тестировщик может сам оценить, насколько важна ему та ИЛИ иная ошибка проекта. В таблице 2 описаны критерии определения уровней критичности дефектов.

Таблица 2 – Критерии определения уровня критичности дефекта

Уровень критичности	**Критерии дефекта**
Blocker (блокирующий)	Тестирование не может быть продолжено, проект неработоспособен.
Critical (критический)	Функции приложения недоступны или заблокированы (CQ, Jira). Важные части системы в нерабочем состоянии. Функция работает с серьезными нарушениями функциональных требований. Основная задача приложения или его основной части не может быть выполнена. Дефект нейтрализовать невозможно. В результате возникновения дефекта произошла потеря данных. Дефект существенно влияет на работу пользователя. Нарушение нормальной работы серьезно влияет на системы заказчика, наносит урон его репутации, или нарушает положение о конфиденциальности информации. Приложение невозможно установить. Приложение после обновления не работает. Механизм лицензирования продукта не работает.
Major (серьезный)	Нормальная работа части приложения нарушена. Существует способ нейтрализации дефекта, но он недостаточно проверен и надежен. Дефект вызвал серьезные функциональные или другие проблемы. Причины отказа в работе точно не определены, но отказ имеет место быть, блокирует часть функциональности и воспроизводится всегда. Функция выполняется некорректно, и это влияет на работу пользователя. Ввод определенных значений в поле блокирует функцию. Сохраняется только часть данных, и это влияет на работу других модулей. Система игнорирует некоторые данные, и это влияет на основные функции. Одна или несколько функций выполняются не полностью, что влияет на работу пользователя. Результаты подсчетов некорректны (значительно отличаются).

Уровень критичности	Критерии дефекта
Average (средний), Normal	Дефект не блокирует работу приложения. Наличие дефекта не вызывает значительного ухудшения производительности, функционирования или удобства использования. Существуют проверенные способы нейтрализации дефекта. Дефект нарушает нормальную работу функции, но его можно быстро исправить проверенным способом. Поведение функции частично соответствует сценарию. Проверка правильности ввода данных не проводится или выдает неверные результаты либо тип данных, вводимых в поле, не соответствует требованиям. Механизм навигации не работает должным образом. Дефекты удобства использования серьезно влияют на работу пользователя. Количество полей на форме не соответствует требованиям. Функция работает недостаточно четко или не во всех случаях, но это серьезно не влияет на работу пользователя. Одна или несколько функций выполняются частично (на работу пользователя это влияет незначительно).
Minor (незначительный)	Дефект не влияет в значительной степени на точность или удобство выпускаемой версии. Функция имеет несущественное значение или вообще не влияет на работу. Опечатки, нечеткие формулировки или ограниченная область видимости сообщений об ошибках. Названия полей, форм, таблиц и окон отличаются от указанных в сценарии. Длина поля не ограничена или не соответствует требованиям. Сообщение об ошибке в этом случае не выдается. Формат данных, отображаемых на форме, отличается от указанного в требованиях, но это не влияет на понимание смысла. Не выдаются подтверждения, сообщения об

Уровень критичности	Критерии дефекта
	ошибках или информационные сообщения. Неверный порядок расположения столбцов на форме. В аналогичных случаях отображаются разные сообщения. В аналогичных случаях названия кнопок, форм, полей, окон и таблиц имеют отличия. Текст сообщений не соответствуют стандартам графического интерфейса или сценарию использования. Табуляция не соответствует требованиям.
Trivial (несущественный)	Опечатки, нечеткие формулировки или ограниченная область видимости сообщений об ошибках. Названия полей, форм, таблиц и окон отличаются от указанных в сценарии. В аналогичных случаях названия кнопок, форм, полей, окон и таблиц имеют отличия. Текст сообщений не соответствуют стандартам графического интерфейса или сценарию использования.
Enhancement (рекомендация)	Рекомендации по улучшению качества. Предложения по незначительному улучшению функциональности.

Правила изменения уровней критичности дефектов

Необходимо отметить, что определение уровня критичности дефекта является субъективным процессом, при этом оно зависит не только от восприятия критичности ошибок самим тестировщиком, но и от проекта, на котором найдена ошибка. По согласованию с командой разработки либо с Заказчиком, процесс определения уровней критичности может быть изменен.

Помимо всего, на определение уровней критичности дефектов может влиять фаза развития проекта - на начальных этапах любые дефекты внешнего вида проекта могут вноситься исключительно с незначительным уровнем важности, зато на этапе тестирования финальной версии продукта любая ошибка в слове на странице может оцениваться как серьезная.

Тестировщик не должен изменять уровень критичности дефектов, но возможны исключения. Ниже перечислены правила изменения критичности.

Понижать уровень критичности дефектов можно в следующих случаях:

– дефект обнаружен при тестировании на редко используемой операционной системе (Win2000);

– дефект трудно воспроизвести либо он был воспроизведен только единожды и не имеет четкого пути воспроизведения;

– дефект обнаружен в части приложения, редко используемой пользователями.

Повышать уровень критичности дефектов можно в следующих случаях:

– дефект графического интерфейса очевиден, и его легко обнаружить (например, орфографическая ошибка в хорошо обозримой части интерфейса), или обнаруженный дефект имеет негативные ассоциации и может восприниматься пользователем как оскорбление;

– дефект обнаружен в часто используемой и очень важной для пользователя части приложения;

– потенциально дефект может серьезно повлиять на работу пользователя.

Практическая часть

Содержание задания

В рамках индивидуального задания требуется найти всевозможные дефекты функциональности и интерфейса кнопок программы «Калькулятор». Описать найденные дефекты, как это показано в теоретической части.

Порядок выполнения работы

1 Ознакомиться с теоретической частью.

2 Получить индивидуальное задание у преподавателя.

3 Открыть программу calc.exe, которая находится в папке с заданием на лабораторную работу.

4 Найти минимум четыре дефекта, согласно варианту задания, полученного у преподавателя.

5 Описать найденные дефекты.

6 Составить отчёт о проделанной работе.

7 Показать отчёт преподавателю.

8 После защиты, выполненного задания, скинуть отчёт преподавателю (в **названии** указать **фамилию и номер лабораторной**, например, Иванов2).

Варианты заданий

1 Кнопки «1», «2», «3», «<», «C».

2 Кнопки «4», «5», «6», «+», «-».

3 Кнопки «7», «8», «9», «/», «*».

4 Кнопки «0», «,», «+/-», «=».

5 Меню и поле ввода.

Содержание отчета

1 Титульный лист.

2 Цель работы.

3 Краткие теоретические сведения.

4 Анализ задания.

5 Найденные дефекты, с описанием их нахождения (в виде скриншота программы, на котором выделен дефект).

6 Уровни критичности дефектов с обоснованием.

7 Описание найденных дефектов.

8 Вывод о проделанной работе.

Защита работ проводится индивидуально.

Контрольные вопросы

1 Что такое дефект?

2 Какие уровни критичности дефекта вы знаете.

3 Дефект, связанный с масштабированием приложения, какой уровень критичности имеет?

4 Всегда ли нужны мультимедийные файлы для описания дефектов?

5 Правила определения уровней критичности.

6 Как правильно описать HeadLine?

7 Что такое Expected Result?

8 Опишите что такое Attachment.

9 Зачем нужен Description?

10 Опишите, что характерно для дефектов, имеющих уровни критичности Major и Average.