Лабораторная работа №7

Регрессионное тестирование

*Цель работы:* научиться производить регрессионное тестирование.

*Отчет по лабораторной работе:* тестируемая программа, отчет о проведении регрессионного тестирования.

## Теоретическая часть

Одним из важных моментов качественного тестирования ПО является проведение так называемого регрессионного тестирования (тестов регрессии). Часто эта группа тестов проводится не в полном объеме или не проводится вообще.

**Регрессионное тестирование** - это вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде.

Под регрессионным тестированием понимают те виды тестов, которые проводятся с каждой новой версией программы. Иными словами, тесты регрессии - это своего рода "старые песни о главном". Цель проведения этих тестов проста - убедиться, что новая версия программы не содержит ошибок в уже протестированных участках кода. По данным зарубежных авторов количество ошибок, возникающих в процессе изменения кода (исправления багов, внедрения новой функциональности и т.п.) колеблется от 50% до 80%. Выявить эти ошибки и помогают тесты регрессии.

Как правило, для регрессионного тестирования используются тест-кейсы, написанные на ранних стадиях разработки и тестирования. Это дает гарантию того, что изменения в новой версии приложения не повредили уже существующую функциональность. Рекомендуется делать автоматизацию регрессионных тестов, для ускорения последующего процесса тестирования и обнаружения дефектов на ранних стадиях разработки программного обеспечения.

#### Виды верификационного тестирования

Поскольку регрессионное тестирование представляет собой повторное проведение цикла обычного тестирования, виды регрессионного тестирования совпадают с видами обычного тестирования. Можно говорить, например, о модульном регрессионном тестировании или о функциональном регрессионном тестировании.

Другой способ классификации видов регрессионного тестирования связывает их с типами сопровождения, которые, в свою очередь, определяются типами модификаций. Выделяют три типа сопровождения:

* Корректирующее сопровождение, называемое обычно исправлением ошибок, выполняется в ответ на обнаружение ошибки, не требующей изменения спецификации требований. При корректирующем сопровождении производится диагностика и корректировка дефектов в программном обеспечении с целью поддержания системы в работоспособном состоянии.
* Адаптивное сопровождение осуществляется в ответ на требования изменения данных или среды исполнения. Оно применяется, когда существующая система улучшается или расширяется, а спецификация требований изменяется с целью реализации новых функций.
* Усовершенствующее (прогрессивное) сопровождение включает любую обработку с целью повышения эффективности работы системы или эффективности ее сопровождения.

В процессе адаптивного или усовершенствующего сопровождения обычно вводятся новые модули. Чтобы отобразить то или иное усовершенствование или адаптацию, изменяется спецификация системы. При корректирующем сопровождении, как правило, спецификация не изменяется, и новые модули не вводятся. Модификация программы на фазе разработки подобна модификации при корректирующем сопровождении, так как из-за обнаружения ошибки вряд ли требуется менять спецификацию программы. За исключением редких моментов крупных изменений, на фазе сопровождения изменения системы обычно невелики и производятся с целью устранения проблем или постепенного расширения функциональных возможностей

Соответственно, определяют два типа регрессионного тестирования: прогрессивное и корректирующее.

* Прогрессивное регрессионное тестирование предполагает модификацию технического задания. В большинстве случаев при этом к системе программного обеспечения добавляются новые модули.
* При корректирующем регрессионном тестировании техническое задание не изменяется. Модифицируются только некоторые операторы программы и, возможно, конструкторские решения.

Прогрессивное регрессионное тестирование обычно выполняется после адаптивного или усовершенствующего сопровождения, тогда как корректирующее регрессионное тестирование выполняется во время тестирования в цикле разработки и после корректирующего сопровождения, то есть после того, как над программным обеспечением были выполнены некоторые корректирующие действия. Вообще говоря, корректирующее регрессионное тестирование должно быть более простым, чем прогрессивное регрессионное тестирование, поскольку допускает повторное использование большего количества тестов.

Подход к отбору регрессионных тестов может быть активным или консервативным. Активный подход во главу угла ставит уменьшение объема регрессионного тестирования и пренебрегает риском пропустить дефекты. Активный подход применяется для тестирования систем с высокой исходной надежностью, а также в случаях, когда эффект изменений невелик. Консервативный подход требует отбора всех тестов, которые с ненулевой вероятностью могут обнаруживать дефекты. Этот подход позволяет обнаруживать большее количество ошибок, но приводит к созданию более обширных наборов регрессионных тестов.

##### Верификационные тесты (Verification Test)

**Тесты верификации багов** (Bug Verification Test). Представляют собой тесты проверки исправления багов. Приведем пример. Допустим, что тест с номером 3 выявил баг, что было зафикс***и***ровано и передано разработчику для исправления. Через определенное время Вы получили от разработчика новую версию программы, с информацией о том, что опис***а***нны***й баг испра***влен. Ваша задача - провести тест с номером 3 повторно - для того, чтобы убедиться, что баг действительно больше не проявляется. В случае успешного прохождения теста такой баг помечается как Verified, в противном случае - как re-do, о чем сообщается разработчику и передается на доработку. Проведение таких тестов является обязательным. Так как причин, из-за которых исправленный баг может сохраниться в программе - множество (от ошибочного описания, а, возможно, и понимания проблемы, до ошибочного утверждения о том, что исправление имело место).

**Тесты верификации версии** (Build Verification Test; Build Acceptance Test, smoke test, quick check). Представляют собой набор тестов для проверки сохранности основной функциональности в каждой новой версии программы. Иными словами - это краткое тестирование всех основных функций разрабатываемого ПО, цель которого - убедится, что программа "работает нормально", что основная функциональность программы не нарушена. Если хотя бы один из тестов верификации версии выявляет баг - то тестер возвращается к предыдущей (последней "рабочей"), дальнейшей тестирование новой версии не проводится, а информация об ошибке вносится в базу и отправляется разработчику. Т.о. тесты верификации версии представляют собой краткий набор основных тестов функциональности.

##### Тесты Регрессии (или Regression Test Pass)

Под этим понятием объединяют те тесты, которые уже проводились с предыдущими версиями программы, притом успешно, т.е. не выявили багов и были отмечены (например в TCM) как pass (passed). Необходимость проведения таких тестов очевидна. Приведем пример. Допустим, что ранее проведенный тест № 2, который обеспечивал проверку в программе участка кода (назовем его условно кодом-А) не выявил ошибок в программе, и был отмечен как pass. В ходе разработки возникла необходимость изменить участок кода-А (например, при исправлении какого либо иного бага или же придания программе новой функциональности). В результате этот участок кода требует дополнительной проверки, что и будет сделано при повторном проведении теста № 2. Среди Собственно Тестов Регрессии можно выделить две группы. Первая - тесты, входящие в набор (т.н. Regression Test Pass with Regression Test Suit), другие - тесты не входящие в набор (т.н. Regression Test Pass without Regression Test Suit). Существенные отличия между ними в следующем: первые - вносятся в базу и описываются, для них могут и должны быть созданы скрипты, которые позволяют автоматизировать процесс тестирования; вторые - существуют только "в голове" тестировщика и проводятся в ручную, причин этого может быть много - от малых сроков тестирования, до отсутствия необходимого ПО, для автоматизации процесса.

##### Тесты регрессии на "закрытых" багах

Рассмотрим пример. Допустим, что тест № 3, выявивший баг, после исправления этого бага разработчиком был проведен повторно, при том успешно. Тест был отмечен как pass, а баг - как Verified. Такой баг откладывается "на полочку", "дело" закрыто. Такой баг и будет "закрытым". Допустим теперь, что в ходе разработки, участок кода, где был исправлен этот баг был изменен, или сменился разработчик, который случайно удалил патч в коде исправлявшую этот баг и показавшийся ему лишним и т.п. В этом случае баг проявится снова. Что бы не допустить подобного, бета-тестеру время от времени необходимо проводить тесты, выявлявшие ранее баги в измененном участке кода, исправление которых уже было проверено ранее и зафиксировано в базе. Это и есть тесты регрессии на "закрытых" багах.

##### Управляемое регрессионное тестирование

Для обеспечения управляемости регрессионного тестирования необходимо выполнение ряда условий:

* Как при модульном, так и при интеграционном регрессионном тестировании в качестве модулей, вызываемых тестируемым модулем непосредственно или косвенно, должны использоваться реальные модули системы. Это легко осуществить, поскольку на этапе регрессионного тестирования все модули доступны в завершенном виде.
* Информация об изменениях корректна. Информация об изменениях указывает на измененные модули и разделы спецификации требований, не подразумевая при этом корректность самих изменений. Кроме того, при изменении спецификации требований необходимо усиленное регрессионное тестирование изменившихся функций этой спецификации, а также всех функций, которые могли быть затронуты по неосторожности. Единственным случаем когда мы вынуждены положиться на правильность измененного технического задания, является изменение технического задания для всей системы или для модуля верхнего (в графе вызовов) уровня, при условии, что кроме технического задания, не существует никакой дополнительной документации и/или какой-либо другой информации, по которой можно было бы судить об ошибке в техническом задании.
* В программе нет ошибок, кроме тех, которые могли возникнуть из-за ее изменения.
* Тесты, применявшиеся для тестирования предыдущих версий программного продукта, доступны, при этом протокол прогона тестов состоит из входных данных, выходных данных и траектории. Траектория представляет собой путь в управляющем графе программы, прохождение которого вызывается использованием некоторого набора входных данных. Ее можно применять для оценки структурного покрытия, обеспечиваемого набором тестов.
* Для проведения регрессионного тестирования с использованием существующего набора тестов необходимо хранить информацию о результатах выполнения тестов на предыдущих этапах тестирования.

#### Обоснование корректности метода отбора тестов

* Некоторые участки кода программы не получают управление при выполнении некоторых тестов.
* Если участок кода реализует требование, но измененный фрагмент кода не получает управления при выполнении теста, то он и не может воздействовать на значения выходных данных программы при выполнении данного теста.
* Даже если участок кода, реализующий требование, получает управление при выполнении теста, это далеко не всегда отражается на выходных данных программы при выполнении данного теста. Действительно, если изменяется первый блок программы, например, путем добавления инициализации переменной, все пути в программе также изменяются, и, как следствие, требуют повторного тестирования. Однако может так случиться, что только на небольшом подмножестве путей действительно используется эта инициализированная переменная.
* Не каждый тест, проверяющий код, находящийся на одном пути с измененным кодом, обязательно покрывает этот измененный код.
* Код, находящийся на одном пути с измененным кодом, может не воздействовать на значения выходных данных измененных модулей программы.
* Не всегда каждый оператор программы воздействует на каждый элемент ее выходных данных

#### Классификация тестов при отборе

Создание наборов регрессионных тестов рекомендуется начинать с множества исходных тестов. При заданном критерии регрессионного тестирования все исходные тесты подразделяются на четыре подмножества:

1. Множество тестов, пригодных для повторного использования. Это тесты, которые уже запускались и пригодны к использованию, но затрагивают только покрываемые элементы программы, не претерпевшие изменений. При повторном выполнении выходные данные таких тестов совпадут с выходными данными, полученными на исходной программе. Следовательно, такие тесты не требуют перезапуска.
2. Множество тестов, требующих повторного запуска. К ним относятся тесты, которые уже запускались, но требуют перезапуска, поскольку затрагивают, по крайней мере, один измененный покрываемый элемент, подлежащий повторному тестированию. При повторном выполнении такие тесты могут давать результат, отличный от результата, показанного на исходной программе. Множество тестов, требующих повторного запуска, обеспечивает хорошее покрытие структурных элементов даже при наличии новых функциональных возможностей.
3. Множество устаревших тестов. Это тесты, более не применимые к измененной программе и непригодные для дальнейшего тестирования, поскольку они затрагивают только покрываемые элементы, которые были удалены при изменении программы. Их можно удалить из набора регрессионных тестов.
4. Новые тесты, которые еще не запускались и могут быть использованы для тестирования.

Классификация тестов по отношению к изменениям в коде требует анализа последствий изменений. Тесты, активирующие код, затронутый изменениями, могут требовать повторного запуска или оказаться устаревшими. Чтобы тест был включен в класс тестов, требующих повторного запуска, он должен быть затронут изменениями в коде, а также должен способствовать увеличению степени покрытия измененного кода по используемому критерию. Затронутым элементом теста может быть траектория, выходные значения, или и то, и другое. Чтобы тест был включен в класс тестов, пригодных для повторного использования, он должен вносить вклад в увеличение степени покрытия кода и не требовать повторного запуска.

Степень покрытия кода определяется для тестов, пригодных для повторного использования, поскольку к этому классу относятся тесты, не требующие повторного запуска и способствующие увеличению степени покрытия до желаемой величины. Если имеется компонент программы, не задействованный пригодными для повторного использования тестами, то вместо них выбираются и выполняются с целью увеличения степени покрытия тесты, требующие повторного запуска. После запуска такой тест становится пригодным для повторного использования или устаревшим. Если тестов, требующих повторного запуска, больше не осталось, а необходимая степень покрытия кода еще не достигнута, порождаются дополнительные тесты и тестирование повторяется.

Окончательный набор тестов собирается из тестов, пригодных для повторного использования, тестов, требующих повторного запуска, и новых тестов. Наконец, устаревшие и избыточные тесты удаляются из набора тестов, поскольку избыточные тесты не проверяют новые функциональные возможности и не увеличивают покрытие.

## Задание

Разработать регрессионные тесты для программы, разрабатывавшейся и тестировавшейся в лабораторных работах №3,4,5. Внести изменения в программу (переработать структуру классов, добавить новую функциональность, исправить ошибки). Провести регрессионное тестирование. Составить отчет о результате проведенного регрессионного тестирования.