**Регрессионное тестирование.**

*Введение.*

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. Цель данной статьи - дать краткую характеристику тестам регрессии, выделить ключевые положения темы.

*Определение.*

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.

*Виды тестов регрессии.*

Таким образом регрессионное тестирование - понятие комплексное. Рассмотрим основные виды тестов регрессии:

1. Верификационные тесты (Verification Test).
   * Тесты верификация багов (Bug Verification Test). Представляют собой тесты проверки исправления багов. Приведем пример. Допустим, что тест с номером 3 выявил баг, что было зафиксровано и передано разработчику для исправления. Через определенное время Вы получили от разработчика новую версию программы, с информацией о том, что описнный баг исправлен. 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.
2. Собственно 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.
3. 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 "нашлепку" в коде исправлявшую этот глюк и показавшуюся ему лишней 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.

*Когда и как?*

На вопрос когда и как проводить регрессионное тестирование, и какие тесты ставить в первую очередь ответить не просто. Все определяется видом разрабатываемого ПО, продолжительностью жизненного цикла, сроками тестирования, количеством членов команды.

Далее описаны лишь общие положения:

1. Регрессионное тестирование проводится в каждой новой версии.
2. Начинают регрессионное тестирование с Тестов верификации версии. Если программа приходит от разработчика в виде полноценной инсталляции, то Тесты верификации начинаются с проверки инсталляции, после чего проводится краткий набор тестов функциональности. Если хотя бы один из тестов failed, версия передается на доработку, регрессионное тестирование прекращается, а тестер возвращается к тестированию последней "рабочей" версии.
3. После успешного прохождения тестов верификации версии, проводят серию Тестов верификации багов.
4. Из Собственно тестов регрессии проводят лишь те, которые сопряжены с измененным в новой версии участком кода.
5. Аналогичным образом (см. пункт 4 ) отбираются тесты в группу регрессии на "закрытых" багах.
6. Тесты регрессии, выполненные успешно (pass) дважды считаются "закрытыми". Дальнейшее их использование производится так как описано в пункте 4.
7. Для тестов регрессии, которые предполагается проводить более 3-5 раз рекомендуется писать скрипты для автоматизации процесса. Это относится ко всемгруппам тестов регрессии.
8. Отбор тестов для Финального регрессионного тестирования осуществляется по следующим принципам:

* В первую очередь отбирают тесты забракованные (failed) два и более раз. В том числе и те, которые выявляли баги, требующие доработки (re-do).
* Во вторую очередь отбираются тесты забракованные один раз, и успешно пройденные повторно.
* Далее отбираются все тесты, которые были пройдены успешно (pass), но проводились только один раз.
* Затем проводятся все остальные тесты, в зависимости от поставленной задачи.

Для наглядности при проведении Регрессионного тестирования можно использовать следущую таблицу:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № теста | № версии | № бага | № версии | № бага |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Количество столбцов соответствует количеству версий.

Например,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № теста | Версия 1 | № бага | Версия 2 | № бага | Версия 3 | № бага |
| 1 | Pass |  |  |  |  |  |
| 2 | Pass |  |  |  |  |  |
| 3 | Fail | 1 | Pass | 1 - verified |  |  |
| 4 | Pass |  |  |  |  |  |
| 5 | Pass |  |  |  |  |  |
| 6 | Fail | 2 | Pass | 2 - verified |  |  |
| 7 | Fail | 3 | Fail | 3 - re-do | Pass | 3 - verified |
| 8 |  |  | Pass |  |  |  |
| 9 |  |  | Fail | 4 |  |  |
| 10 |  |  |  |  | Pass |  |

**Комментарий.** В ходе тестирования среди первых тестов №№ 1, 2, 4, 5 были проведены успешно и отмечены как pass. Тесты № 3, 6, 7 выявили баги (fail) соответственно №№ 1, 2 и 3. В версии №2 разработчик сообщил, что баги №№ 1,2 и 3 исправлены. В ходе Тестов верификации багов (предполагается, что Тесты верификации версии прошли успешно) выяснилось, что тесты №№ 3 (выявил баг № 1) и 6 (баг № 2) прошли успешно (баги помечены как verified), а тест № 7 (баг № 3 ) - вновь выявил тот же баг, о чем сообщено разработчику (re-do). Кроме того во второй версии было продолжено тестирование и проведены тесты №№ 8 и 9. Тест № 8 прошел успешно, а тест № 9 выявил баг № 4. В третьей версии (тесты верификации версии также прошли успешно) разработчик повторно сообщил, что баг № 3 исправлен, что и подтвердило повторное проведение этого теста (тест - pass, баг - verified). Информации об исправлении бага № 4 в третьей версии от разработчика не поступало, поэтому этот тест верификации не проводился. Очередной тест № 10 багов не выявил (pass).