ВикипедиЯ

Тестирование программного обеспечения

Материал из Википедии — свободной энциклопедии

Тести́рование програ́ммного обеспе́че́ния — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом (ISO/IEC TR 19759:2005)[1].

Содержание

Определения тестирования

В разное время и в различных источниках тестированию давались различные определения, в том числе:

- процесс выполнения программы с целью нахождения ошибок[2];
- интеллектуальная дисциплина, имеющая целью получение надежного программного обеспечения без излишних усилий на его проверку[3];
- техническое исследование программы для получения информации о её качестве с точки зрения определённого круга заинтересованных лиц (*С. Канер* [*уточнить*]);
- проверка соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выполненных определённым образом[1];
- процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо аспектов её работы[4];

- процесс, имеющий целью выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации[5];
- процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов[6].

История

Первые программные системы разрабатывались в рамках программ научных исследований или программ для нужд министерств обороны. Тестирование таких продуктов проводилось строго формализованно с записью всех тестовых процедур, тестовых данных, полученных результатов. Тестирование выделялось в отдельный процесс, который начинался после завершения кодирования, но при этом, как правило, выполнялось тем же персоналом.

В 1960-х много внимания уделялось «исчерпывающему» тестированию, которое должно проводиться с использованием всех путей в коде или всех возможных входных данных. Было отмечено, что в этих условиях полное тестирование программного обеспечения невозможно, потому что, во-первых, количество возможных входных данных очень велико, во-вторых, существует множество путей, в-третьих, сложно найти проблемы в архитектуре и спецификациях. По этим причинам «исчерпывающее» тестирование было отклонено и признано теоретически невозможным.

В начале 1970-х годов тестирование программного обеспечения обозначалось как «процесс, направленный на демонстрацию корректности продукта» или как «деятельность по подтверждению правильности работы программного обеспечения». В зарождавшейся программной инженерии верификация ПО значилась как «доказательство правильности». Хотя концепция была теоретически перспективной, на практике она требовала много времени и была недостаточно всеобъемлющей. Было решено, что доказательство правильности — неэффективный метод тестирования программного обеспечения. Однако, в некоторых случаях демонстрация правильной работы используется и в наши дни, например, приёмо-сдаточные испытания. Во второй половине 1970-х тестирование представлялось как выполнение программы с намерением найти ошибки, а не доказать, что она работает. Успешный тест — это тест, который обнаруживает ранее неизвестные проблемы. Данный подход прямо противоположен предыдущему. Указанные два определения представляют собой «парадокс тестирования», в основе которого лежат два противоположных утверждения: с одной стороны, тестирование позволяет убедиться, что продукт работает хорошо, а с другой — выявляет ошибки в программах, показывая, что продукт не работает. Вторая цель тестирования является более продуктивной с точки зрения улучшения качества, так как не позволяет игнорировать недостатки программного обеспечения.

В 1980-е годы тестирование расширилось таким понятием, как предупреждение дефектов. Проектирование тестов — наиболее эффективный из известных методов предупреждения ошибок. В это же время стали высказываться мысли, что необходима методология тестирования, в частности, что тестирование должно включать проверки на всем протяжении цикла разработки, и это должен быть управляемый процесс. В ходе тестирования надо проверить не только собранную программу, но и требования, код, архитектуру, сами тесты. «Традиционное» тестирование, существовавшее до начала 1980-х, относилось только к скомпилированной, готовой системе (сейчас это обычно называется системное тестирование), но в дальнейшем тестировщики стали вовлекаться во все аспекты жизненного цикла разработки. Это позволяло раньше находить проблемы в требованиях и архитектуре и тем самым сокращать сроки и бюджет разработки. В середине 1980-х появились первые инструменты для автоматизированного тестирования. Предполагалось, что компьютер

сможет выполнить больше тестов, чем человек, и сделает это более надёжно. Поначалу эти инструменты были крайне простыми и не имели возможности написания сценариев на скриптовых языках.

В начале 1990-х годов в понятие «тестирование» стали включать планирование, проектирование, создание, поддержку и выполнение тестов и тестовых окружений, и это означало переход от тестирования к обеспечению качества, охватывающего весь цикл разработки программного обеспечения. В это время начинают появляться различные программные инструменты для поддержки процесса тестирования: более продвинутые среды для автоматизации с возможностью создания скриптов и генерации отчетов, системы управления тестами, ПО для проведения нагрузочного тестирования. В середине 1990-х годов с развитием Интернета и разработкой большого количества веб-приложений особую популярность стало получать «гибкое тестирование» (по аналогии с гибкими методологиями программирования).

Стандарты, относящиеся к тестированию

- IEEE 829—2008 IEEE Standard for Software and System Test Documentation
- ANSI/IEEE Std 1008—1987 — IEEE Standard for Software Unit Testing
- ISO/IEC/IEEE 29119-1:2013 Software and systems engineering — Software testing — Part 1: Concepts and definitions
- ISO/IEC/IEEE 29119-2:2013 Software and systems engineering — Software testing — Part 2: Test processes
- ISO/IEC/IEEE 29119-3:2013 Software and systems engineering — Software testing — Part 3: Test documentation

Классификации видов и методов тестирования

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

По объекту тестирования

- Функциональное тестирование
- Тестирование производительности
  - Нагрузочное тестирование
  - Стресс-тестирование
  - Тестирование стабильности
- Конфигурационное тестирование
- Юзабилити-тестирование
- Тестирование безопасности
- Тестирование локализации
- Тестирование совместимости

По знанию внутреннего строения системы

- Тестирование чёрного ящика
- Тестирование белого ящика
- Тестирование серого ящика

По степени автоматизации

- Ручное тестирование
- Автоматизированное тестирование
- Полуавтоматизированное тестирование

По степени изолированности[7][8]

- Тестирование компонентов
- Интеграционное тестирование
- Системное тестирование

По времени проведения тестирования

- Альфа-тестирование
  - Дымовое тестирование (англ. *smoke testing*)
  - Тестирование новой функции (*new feature testing*)
  - Подтверждающее тестирование
  - Регрессионное тестирование
  - Приёмочное тестирование
- Бета-тестирование

По признаку позитивности сценариев

- Позитивное тестирование
- Негативное тестирование

По степени подготовленности к тестированию

- Тестирование по документации (формальное тестирование)
- Интуитивное тестирование (англ. *ad hoc testing*)

Уровни тестирования

- Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.
- Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
- Системное тестирование — тестируется интегрированная система на её соответствие требованиям.
  - Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут

быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.
- Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

Статическое и динамическое тестирование

Описанные ниже техники — тестирование белого ящика и тестирование чёрного ящика — предполагают, что код исполняется, и разница состоит лишь в той информации, которой владеет тестировщик. В обоих случаях это *динамическое тестирование*.

При статическом тестировании программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. В некоторых случаях анализируется не исходный, а промежуточный код (такой как байт-код или код на MSIL).

Также к статическому тестированию относят тестирование требований, спецификаций, документации.

Регрессионное тестирование

После внесения изменений в очередную версию программы, регрессионные тесты подтверждают, что сделанные изменения не повлияли на работоспособность остальной функциональности приложения. Регрессионное тестирование может выполняться как вручную, так и средствами автоматизации тестирования.

Тестовые сценарии

Тестировщики используют тестовые сценарии на разных уровнях: как в компонентном, так и в интеграционном и системном тестировании. Тестовые сценарии, как правило, пишутся для проверки компонентов, в которых наиболее высока вероятность появления отказов или вовремя не найденная ошибка может быть дорогостоящей.

Тестирование «белого ящика», «чёрного ящика» и «серого ящика»

В зависимости от доступа разработчика тестов к исходному коду тестируемой программы различают «тестирование (по стратегии) белого ящика» и «тестирование (по стратегии) чёрного ящика».

При тестировании белого ящика (также говорят — *прозрачного ящика*), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения. Это типично для компонентного тестирования, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода или мутационное тестирование.

При тестировании чёрного ящика тестировщик имеет доступ к программе только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования. Например, тестирующий компонент может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе. Обычно в данном виде тестирования критерий покрытия складывается из покрытия структуры входных данных, покрытия требований и покрытия модели (в тестировании на основе моделей).

При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Если «альфа-» и «бета-тестирование» относятся к стадиям до выпуска продукта (а также, неявно, к объёму тестирующего сообщества и ограничениям на методы тестирования), тестирование «белого ящика» и «чёрного ящика» имеет отношение к способам, которыми тестировщик достигает цели.

Бета-тестирование в целом ограничено техникой чёрного ящика (хотя постоянная часть тестировщиков обычно продолжает тестирование белого ящика параллельно бета-тестированию). Таким образом, термин «бета-тестирование» может указывать на состояние программы (ближе к выпуску, чем «альфа»), или может указывать на некоторую группу тестировщиков и процесс, выполняемый этой группой. То есть, тестировщик может продолжать работу по тестированию белого ящика, хотя программа уже «бета-стадии», но в этом случае он не является частью «бета-тестирования».

Покрытие кода

Покрытие кода показывает процент исходного кода программы, который был выполнен («покрыт») в процессе тестирования. По способам измерения выделяют покрытие операторов, покрытие условий, покрытие путей, покрытие функций и др.

См. также

- Формальная верификация
- Разработка через тестирование
- Система отслеживания ошибок
- Аутсорсинг тестирования программного обеспечения
- Внесение неисправностей

Примечания

1. ISO/IEC TR 19759:2005 (SWEBOOK)
2. *Майерс Г.* Надежность программного обеспечения. М: Мир, 1980

3. *Бейзер Б.* Software Testing Techniques, Second Edition. — NY:van Nostrand Reinhold, 1990
4. ANSI/IEEE standart 610.12-1990 Glossary of SE technology NY:IEEE, 1987
5. *Sommerville I.* Software Engineering, 8th ed. Harlow, England: Pearson Education, 2007
6. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения, версия 2.3, под ред. Erik van Veenendaal // International Software Testing Qualifications Board (ISTQB), 2014
7. ГОСТ Р 56920-2016 | НАЦИОНАЛЬНЫЕ СТАНДАРТЫ (http://protect.gost.ru/document1.aspx?control=31&baseC=6&page=4&month=6&year=2016&search=&id=203397). protect.gost.ru. Дата обращения: 12 марта 2017.
8. Software and systems engineering Software testing Part 1:Concepts and definitions (http://ieeexplore.ieee.org/document/6588537/) // ISO/IEC/IEEE 29119-1:2013(E). — 2013-09-01. — С. 1–64. — doi:10.1109/IEEESTD.2013.6588537 (https://dx.doi.org/10.1109%2FIEEESTD.2013.6588537).

Литература

- *Гленфорд Майерс, Том Баджетт, Кори Сандлер.* Искусство тестирования программ, 3-е издание (http://www.dialektika.com/books/978-5-8459-1796-6.html) = The Art of Software Testing, 3rd Edition. — М.: «Диалектика», 2012. — 272 с. — ISBN 978-5-8459-1796-6. Архивная копия (http://web.archive.org/web/20120719122632/http://www.dialektika.com/books/978-5-8459-1796-6.html) от 19 июля 2012 на Wayback Machine
- *Лайза Криспин, Джанет Грегори.* Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — М.: «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series). — 1000 экз. — ISBN 978-5-8459-1625-9.
- *Канер Кем, Фолк Джек, Нгуен Енг Кек.* Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев: ДиаСофт, 2001. — 544 с. — ISBN 9667393879.
- *Калбертсон Роберт, Браун Крис, Кобб Гэри.* Быстрое тестирование. — М.: «Вильямс», 2002. — 374 с. — ISBN 5-8459-0336-X.
- *Синицын С. В., Налютин Н. Ю.* Верификация программного обеспечения. — М.: БИНОМ, 2008. — 368 с. — ISBN 978-5-94774-825-3.
- *Бейзер Б.* Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб.: Питер, 2004. — 320 с. — ISBN 5-94723-698-2.

Ссылки

- Семь принципов тестирования программ (http://www.osp.ru/os/2008/07/5478839/) (рус.)
- Улучшая, не навреди (http://www.osp.ru/os/2014/08/13043490/) (рус.)
- Портал специалистов по тестированию и обеспечению качества ПО (http://software-testing.ru/) (рус.)
- Портал об автоматизированном тестировании ПО (http://automated-testing.info/) (рус.)
