# Отладка и тестирования ПО

Тестирование как часть процесса верификация программного обеспечения.

# Цель валидации и верификации в тестировании ПО

Тестирование программного обеспечения позволяет оценить новое приложение, чтобы убедиться в том, что после запуска оно работает так, как задумано. Составление плана тестирования помогает предотвратить ошибки, снизить затраты на разработку и повысить производительность приложения. В современной цифровой среде люди сильно зависят от программного обеспечения и приложений, поэтому надежность - одно из самых важных качеств.

Верификация и валидация при тестировании жизненно важны для обеспечения того, чтобы разработчики использовали передовые методы создания программного обеспечения. Цель состоит в том, чтобы избежать сбоев в работе приложений на критическом этапе и гарантировать, что они продолжают работать на благо пользователя. Давайте сравним верификацию и валидацию и то, как они влияют на конечный продукт.

Что такое верификация?

При верификации команда разработчиков изучает документы для создания программного обеспечения или приложения. Цель состоит в том, чтобы убедиться, что разработчик, которому поручен проект, соблюдает все изложенные требования. Логика кода должна соответствовать проектной документации независимо от языка программирования.

Процесс верификации

Он включает следующие этапы:

Проверка требований: команда проекта подтверждает правильность всех пунктов в требованиях на основе данных, полученных от бизнеса. Заказчики должны иметь возможность вносить любые изменения или удалять ранее запрошенные. Команда должна получить окончательное одобрение от всех соответствующих заинтересованных сторон, чтобы убедиться, что конечный результат отвечает их потребностям.

Проверка дизайна: команда тестировщиков должна проанализировать дизайн программного обеспечения, включая макеты и прототипы. Охватывают ли логические модели баз данных все соответствующие поля для обработки данных, передаваемых туда и обратно между приложениями? Доволен ли заказчик удобством предлагаемого интерфейса? Прежде чем приступить к работе, необходимо прийти к единому мнению о достоверности проектной документации.

Проверка логики кода: команда проверяет код, чтобы убедиться в его полноте и согласованности. Они также должны проверить исходный код и связанные с ним артефакты, например, дизайн базы данных, и убедиться, что они соответствуют требованиям.

После того как команда подтверждает, что все вышеперечисленное соответствует требуемым стандартам, начинается процесс верификации:

Планирование: команда изучает взаимосвязи между различными требованиями, затем определяет все задачи, которые должны быть выполнены в ходе верификации. Назначаются ответственные за другие задачи, включая демонстрацию того, насколько хорошо функционируют различные компоненты, проверку обеспечения качества или выявление и составление отчетов о дефектах.

Выполнение: команда выполняет поставленные перед ними задачи и документирует результаты верификации.

Отчет: все результаты, полученные в ходе верификации, собираются в отчеты. Команда проверяет, функционирует ли приложение так, как задумано.

Что такое валидация?

Валидация включает в себя проверку приложения на разных этапах разработки, чтобы убедиться, что оно соответствует требованиям. Если в документе требуется веб-страница с функцией живого чата, то разработчик должен создать именно ее. Если что-то отсутствует или не соответствует запросу клиента, это следует выявить и исправить, чтобы получить ожидаемый продукт.

Процесс валидации важен, поскольку он не дает команде разработчиков отклониться от намеченного пути. Необходимо часто проверять документы, включая требования и диаграммы таблиц базы данных. Постоянная проверка документов помогает разработчикам найти упущенные функции, которые необходимо добавить, или противоречивые запросы, которые необходимо решить с клиентом.

Процесс валидации

Он включает следующие этапы:

Проверка дизайна

На этом этапе команда описывает все бизнес-требования конечного пользователя. Затем составляется план валидации для каждого пункта, прежде чем подтвердить готовность программного обеспечения к разработке. После этого команда может получить одобрение от высшего руководства перед началом тестирования.

Проверка установки

Команда пытается установить приложение в соответствии с планом валидации. Цель состоит в том, чтобы убедиться, что процесс установки и все необходимое системное оборудование соответствуют требованиям проекта. Кроме того, тестировщики подтверждают, что тестовая среда функционирует аналогично производственной среде.

Проверка функциональности

Тестировщики подвергают продукт различным сценариям тестирования, чтобы убедиться, что он соответствует заданным пользовательским требованиям. Цель - проверить все доступные функциональные возможности, чтобы определить, работает ли программное обеспечение в соответствии с требованиями заказчика. Команда полагается на план валидации, чтобы убедиться, что тестовая среда подходит для реализации сценариев, описанных в требованиях. Таким образом, они могут гарантировать, что конечный результат соответствует ожидаемому и способен удовлетворить потребности клиента.

Проверка производительности

Тестирование производительности показывает, что программное обеспечение может функционировать так, как это необходимо бизнесу в реальных условиях. Клиенты могут сами проводить бета-тестирование, чтобы получить представление о продукте и понять, был ли он разработан в соответствии с их требованиями. Взгляд со стороны помогает выявить ошибки и дефекты, которые команда разработчиков могла пропустить.

Проверка готовности продукта

После завершения всех других этапов процесса валидации продукт считается готовым к релизу. Это означает, что команда разработчиков может двигаться вперед, выпуская программное обеспечение в производственную среду. Любые дефекты, обнаруженные после выпуска продукции, устраняются с помощью обновлений программного обеспечения.

Зачем проводится верификация и валидация?

Процесс верификации и валидации помогает убедиться, что конечный программный продукт соответствует потребностям клиента, изложенным в его требованиях. Многие компании используют автоматизацию, чтобы справиться с более рутинными задачами тестирования. Ниже перечислены некоторые из основных причин, по которым необходимо использовать верификацию и валидацию.

Уменьшить количество дефектов в реальном продукте

Если новая часть программного обеспечения при выпуске не работает так, как предполагалось, качество продукта может пострадать. Заказчики и команда будут чувствовать себя более уверенно в том, что приложение будет работать без ошибок, если они тщательно его протестировали, используя тестовые сценарии, описанные в процессе верификации и валидации.

Убедиться, что дизайн соответствует техническим условиям

Последнее, чего хочет команда разработчиков – это вызвать недовольство клиента тем, что полученный продукт не соответствует его запросам. Регулярная проверка требований во время верификации и валидации помогает разработчикам не упустить критически важные функциональные и проектные требования, отмеченные в документации.

Заметные различия между верификацией и валидацией

Проверки верификации и валидации происходят на разных этапах цикла разработки программного обеспечения. Например, верификация происходит до того, как разработчик завершает создание программного обеспечения. Это помогает проектным группам выявить ошибки до того, как они попадут в конечный продукт, где их исправление становится более дорогостоящим.

Валидация обычно происходит после того, как программное обеспечение создано и ожидает интеграционного тестирования и производственного релиза. Процесс валидации определяет удобство использования приложения в его текущем состоянии. Тестировщики смотрят на продукт глазами пользователя и пытаются выявить проблемы с функционированием программного обеспечения и недостающие функции. Как правило, валидационные проверки не могут проводиться до тех пор, пока продукт не пройдет процесс верификации.

Когда следует выбрать верификацию?

Верификация должна проводиться до и во время фазы сборки билда. Разработчики должны иметь всю документацию, необходимую для начала создания приложения. Они должны основывать код на требованиях и подтверждать, что они используют логику, соответствующую потребностям пользователя. Это включает в себя частые проверки любого завершенного кода для получения обратной связи от коллег.

Когда следует выбрать валидацию?

Процесс валидации должен проводиться после того, как команда разработчиков завершит разработку продукта. Цель состоит в том, чтобы предоставить клиенту продукт, который он ожидал. Тестировщики работают над тем, чтобы убедиться в правильности работы приложения, используя такие методы проверки, как:

Тестирование методом белого ящика: при тестировании методом белого ящика изучается внутренний код программного обеспечения. Тестировщики должны понимать логику кода и искать такие проблемы, как дыры в безопасности и неработающие функции. Некоторые используют алгоритмы анализа для поиска проблем. Тестирование методом белого ящика можно проводить на разных уровнях разработки программного обеспечения для проверки работы кода приложения.

Тестирование методом черного ящика: тестирование методом черного ящика рассматривает только внешнее функционирование программного обеспечения. О внутреннем устройстве приложения ничего не известно. Тестировщики ищут несоответствия в том, как функционирует программное обеспечение по сравнению с тем, что запросил клиент.

Преимущества верификации

Верификация – это контрольная точка для различных этапов процесса разработки. Команда следит за тем, чтобы разрабатываемый продукт продолжал соответствовать потребностям клиента на основе проектной и технической документации. Проще и дешевле выявить проблемы до того, как они достигнут более поздних стадий жизненного цикла программного обеспечения. Это означает, что команде приходится устранять меньше дефектов во время системного или модульного тестирования.

Преимущества валидации

Валидация помогает команде убедиться, что работа соответствует ожиданиям заинтересованных сторон. Она также дает им последний шанс устранить любые дефекты или несоответствия между приложением и требованиями. Программное обеспечение становится более надежным и менее склонным к сбоям, потому что ошибка не была обнаружена до выпуска продукта.

# Верификация и тестирования программных средств

1. Основные понятия верификации

Под верификацией понимается процесс определения, в какой степени программные средства выполняют наложенные на них требования. В соответствии с требованиями стандарта “ISO 12207” процессы верификации программного средства включают в себя следующие задачи:

* 1. Первая задача верификации «Проверка контракта». При проверке контракта необходимо удостовериться в следующем:
* Во-первых, в том, что поставщик имеет возможность удовлетворить требования контракта;
* Во-вторых, в том, что требования непротиворечивы и покрывают все нужды пользователя;
* В-третьих, в том, что предусмотрены адекватные процедуры для регулирования изменений требований и возрастающих проблем;
* В-четвертых, в том, что предусмотрены процедуры и их приложение на сотрудничество между сторонами, включая право собственности, гарантию, авторское право и конфиденциальность;
* В-пятых, в том, что предусмотрены критерии и процедуры приемки результатов разработки согласно требованиям.
  1. Вторая задача верификации «Проверка процесса проектирования». При проверке процесса проектирования необходимо удостовериться в следующем:
* Во-первых, в том, что требования проектного планирования адекватны и скоординированы во времени;
* Во-вторых, в том, что процессы, выбранные для проекта, адекватны, реализуемы, выполняются, как запланировано и соответствуют контракту;
* В-третьих, в том, что стандарты, процедуры и область функционирования адекватны для проектных процедур;
* В-четвертых, в том, что проект укомплектован ресурсами, и персонал обучен, как требуется по контракту.
  1. Третья задача верификации «Проверка требований». При проверке требований необходимо удостовериться в следующем:
* Во-первых, в том, что требования системы непротиворечивы, выполнимы и проверяемы;
* Во-вторых, в том, что требования системы распределены между компонентами аппаратных средств, компонентами программного средства и ручными операциями согласно критериям проекта;
* В-третьих, в том, что требования к программному средству последовательны, выполнимы, тестируемы и точно отражают требования системы;
* В-четвертых, в том, что требования к программному средству, связанные с безопасностью, защитой и критичностью правильны, как показано соответствующими точными методами анализа.
  1. Четвертая задача верификации «Проверка проекта». При проверке проекта необходимо удостовериться в следующем:
* Во-первых, в том, что проект корректен и не противоречит исходным требованиям контракта;
* Во-вторых, в том, что проект реализует правильный ход событий, вводов, выводов, интерфейсов, логического потока, распределения временных и вычислительных ресурсов, определения ошибок, их обнаружения и восстановления работоспособности программного средства;
* В-третьих, в том, что выбранный проект полностью исходит от требований;
* В-четвертых, в том, что проект реализует безопасность, защищенность и другие критические требования, правильно, как показано соответствующими точными методами анализа.
  1. Пятая задача верификации «Проверка программы». При проверке программы необходимо удостовериться в следующем:
* Во-первых, в том, что текст программы удовлетворяет проекту и требованиям, тестируем, корректен и соответствует стандартам программирования;
* Во-вторых, в том, что программа отражает истинный ход событий, последовательные интерфейсы, правильные данные и управляющий поток, завершенность, размещение временных и вычислительных ресурсов, определение ошибок, их обнаружение и восстановление работоспособности программного средства;
* В-третьих, в том, что программа исходит из проекта и требований контракта;
* В-четвертых, в том, что программа обеспечивает безопасность, защищенность и другие критические требования корректно, как показано соответствующими точными методами анализа.
  1. Шестая задача верификации «Проверка интеграции». При проверке интеграции необходимо удостовериться в следующем:
* Во-первых, в том, что компоненты программного средства и элементы каждого компонента программного средства полностью и правильно интегрированы в комплекс программ;
* Во-вторых, в том, что компоненты аппаратных средств, компоненты программного средства и ручные операции полностью и правильно интегрированы в информационную систему;
* В-третьих, в том, что интеграционные задачи выполнены согласно интеграционному плану.
  1. Седьмая задача верификации «Проверка документации». При проверке документации необходимо удостовериться в следующем:
* Во-первых, в том, что документация адекватна, полна и последовательна;
* Во-вторых, в том, что подготовка документации выполнена своевременно;
* В-третьих, в том, что схема управления документами следует определенным плановым процедурам.

Для повышения эффективности затрат при реализации проекта, верификация должна быть выполнена как можно раньше.

1. Задачи тестирования

Тестирование является одним из видов верификации и направлено на определение того, соответствует ли программное обеспечение требованиям спецификации. Тестированием называется процесс выполнения программного изделия с целью определения содержащихся в нём ошибок. Определение процесса тестирования исходит из того, что всякое программное изделие содержит ошибки. Задача тестирования сводится к обнаружению, как можно большего числа ошибок. В соответствии с определением тестирования, хорошим считается тест, который имеет высокую вероятность обнаружения еще не выявленных ошибок. Тестовый прогон, приведший к обнаружению ошибки, является удачным. Прогон теста, в результате которого ошибка не была обнаружена, является не удачным прогоном. Тест, который с самого начала демонстрирует отсутствие ошибок, как правило, недостаточно хорошо разработан. Осуществляя тестирование необходимо исходить из следующей закономерности. Вероятность наличия необнаруженных ошибок на конкретном участке программы пропорциональна числу ошибок, уже обнаруженных на данном участке. Указанная закономерность отражает неравномерность распределения ошибок на различных участках программы. Поэтому, если первые результаты тестирования позволяют выявить наиболее уязвимые участки программы, то на эти участки при дальнейшем тестировании необходимо обратить особое внимание. Выполняя тестирование необходимо, во-первых, проверить, делает ли данный программный продукт то, для чего он предназначен; во-вторых, проверить, не делает ли данный программный продукт, то, что он не должен делать. При тестировании необходимо проверять возможность наличия неправильных и непредусмотренных входных данных. Тесты, представляющие некорректные входные данные, обладают большой обнаруживающей способностью. При разработке тестов необходимо исходить из того, что исчерпывающее тестирование для всех входных данных невозможно. Поэтому каждый тест должен давать максимальную отдачу по сравнению с произведенными затратами. Отдача теста измеряется вероятностью того, что тест выявит не обнаруженную прежде ошибку. Затраты измеряются стоимостью подготовки теста, стоимостью его выполнения, и стоимостью анализа результатов теста. При рассмотрении эффективности теста, учитывается в какой степени, данный тест обнаруживает ошибки, наибольшим образом снижающие надежность программы. Разработка тестов должна соответствовать международным и российским стандартам. Можно выделить следующие стандарты: Стандарт ISO 12119 (“ Пакеты программ. Требования к качеству и тестирование”). Данный стандарт содержит раздел, который кратко определяет порядок тестирования программного продукта на соответствие данного программного продукта требованиям к качеству. Данные указания охватывают как тестирование для характеристик, присущих всем аналогичным продуктам, так и тестирование для особых характеристик, декларированных в описании данного программного продукта. ГОСТ Р ИСО/МЭК 15408 («Общие критерии»). Данный стандарт в отношении тестирования вводит понятия «Покрытие» и «Глубина». Покрытие показывает полноту охвата тестами (т.е. покрытие тестами) функциональных возможностей объекта оценки. Глубина характеризует уровень детализации тестирования. В стандарте ANSI/IEEE 1008 (“Тестирование программных модулей и компонентов ПС”) рассмотрена методика отладки отдельных модулей и небольших групп программ. Целями данного стандарта являются создание единого подхода к тестированию компонентов программного средства, который может быть использован на практике в качестве основы для получения эффективных ПС, а также описание концепций и допущений, принимаемых при тестировании.

1. Принципы организации тестирования

Эффективная организация тестирования должна основываться на соблюдении следующей совокупности принципов.

Принцип сегментации основан на разделении сложной задачи на ряд небольших подзадач, которые в последующем рассматриваются как самостоятельные.

Принцип проектирования тестирования. Данный принцип требует рассмотрения вопросов тестирования на всех этапах проектирования программного обеспечения.

Принцип моделирования. Данный принцип предусматривает предварительное построение моделей. Моделирование позволяет обеспечить разработку и проверку тестов до разработки основных элементов программного обеспечения.

Принцип выборочного контроля. Тестирование не может быть исчерпывающим, поэтому для контроля выбирают определенную совокупность данных.

Принцип стандартизации. Стандартизация важна как средство для упрощения проблем тестирования и развития методологии получения надежных средств программного обеспечения.

1. Метод тестирования

При разработке тестов исходят из невозможности создать тест, обеспечивающий всестороннее тестирование.

Задача сводится к созданию максимально эффективного теста, обеспечивающего обнаружение ошибок наибольшим образом снижающих надежность программ.

Можно выделить два вида тестирования:

* Во-первых, структурное тестирование;
* Во-вторых, функциональное тестирование.

Структурное тестирование означает тестирование, управляемое логикой программы.

В этом случае, осуществляется тщательный анализ алгоритма программы и предусматривается разработка теста, обеспечивающего наиболее эффективным образом проверку различных участков программы.

Структурное тестирование характеризуется степенью покрытия логики программы. Полное покрытие всей логики программы, т.е. исчерпывающее тестирование, предполагает выполнение программы по всем маршрутам передач управления.

Если программа содержит цикл, то полное тестирование предполагает необходимость расчета по каждому из вариантов цикла.

Например, если циклически повторяется обработка записей трех наборов данных, каждый из которых может содержать до 10 000 записей, то полное тестирование предполагает создание, по меньшей мере 10000\*10000\*10000 тестов.

Если алгоритм будет содержать логическую обработку, то количество необходимых тестов при полном тестировании многократно возрастает. Поэтому при тестировании путем покрытия логики программы при создании теста ориентируются не на полный тест, а на тест, отвечающий требуемому критерию.

Для создания тестов при структурном тестировании могут быть использованы следующие методы:

* Во-первых, метод «покрытия операторов» (Statement Coverage – SC). Данный метод, основан на критерии, который предполагает выполнение каждого оператора программы, хотя бы один раз.
* Во-вторых, метод «покрытия условий» (Decision Coverage – DC). Данный метод, основан на критерии, который предполагает запись числа тестов, достаточного для того, чтобы результаты каждого условия в решении выполнялись по крайней мере один раз.
* В-третьих, метод «комбинаторного покрытия условий» (Modified Condition/Decision Coverage – MC/DC). Данный метод, основан на критерии, который требует создание такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз.

Функциональное тестирование предусматривает тестирование с управлением по данным.

При использовании этой стратегии не учитываются знания о внутренней структуре программы.

При данном подходе определяется поведение программы, которое не соответствует спецификации данной программы.

Функциональное тестирование предполагает задание различных вариантов наборов входных данных в соответствии со спецификацией программы.

Полное тестирование с использованием управления по данным предполагает:

* во-первых, задание всех возможных корректных сочетаний входных данных;
* во-вторых, задание всех возможных не корректных сочетаний входных данных.

Данное число тестов будет являться бесконечно большим.

Реальное тестирование с применением управления по данным ограничивается использованием небольшого подмножества всех возможных данных.

Тесты, которые позволяют обнаружить ошибки одного типа, объединяются в классы эквивалентности.

Можно выделить следующие методы функционального тестирования:

* во-первых, метод «эквивалентного разбиения»;

Разработка тестов методом эквивалентного разбиения осуществляется в два этапа.

На первом этапе выполняется выделение классов эквивалентности.

На втором этапе выполняется построение тестов.

Классы эквивалентности выделяются путем анализа каждого входного условия. За входное условие, как правило, принимается конкретное предложение спецификации.

При выделении классов различают правильные классы эквивалентности и неправильные классы эквивалентности.

Правильные классы эквивалентности представляют правильные входные данные программы.

Неправильные классы эквивалентности представляют ошибочные входные данные.

* во-вторых, метод «анализа граничных значений»;

Данный метод предполагает использование граничных условий. Под граничными условиями понимаются условия, характеризующие ситуации, возникающие на границах входных и выходных классов эквивалентности, а также непосредственно выше или ниже этих границ.

Отличительные особенности анализа граничных значений заключаются в следующем:

* Во-первых, в том, что элементы в класс эквивалентности выбираются таким образом, чтобы проверить тестом каждую границу класса эквивалентности;
* Во-вторых, при разработке тестов рассматриваются не только входные, но и выходные классы эквивалентности.

Следует отметить следующие особенности функционального тестирования:

* во-первых, функциональное тестирование не заканчивается после передачи прикладного программного обеспечения;

Каждый пользователь выполняет функциональное тестирование для своей конкретной ситуации. Иногда, производители прикладного программного обеспечения обговаривают, что предлагаемая версия программного продукта предполагает тестирование пользователями (бета версия).

* во-вторых, основные подходы функционального тестирования, в той или иной мере, могут эффективно использоваться не только при разработке прикладного программного обеспечения;

Основные идеи функционального тестирования изложены в библии. В библии сказано, что отличать пророков от лжепророков следует по результатам их деятельности.

Очевидно, что принципы функционального тестирования выполняет любой покупатель бытовой техники.

В развитых странах по результатам деятельности политических партий осуществляется голосование избирателей. Данный подход также можно рассматривать, как функциональное тестирование.

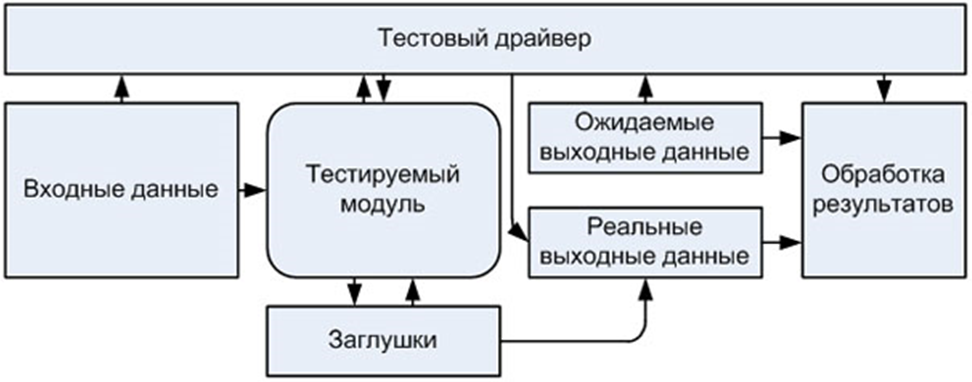
Рассмотренные методы тестирования прикладного программного обеспечения. отличаются целевыми задачами тестирования, проверяемыми компонентами программы и методами оценки результатов. В связи с этим, только совместное применение различных методов тестирования позволяет достигать высокое качество функционирования программных средств.

1. Инструментальные средства тестирования

Тестирование представляет собой дорогой и трудоемкий этап разработки программных систем. Поэтому создан широкий спектр инструментальных средств для поддержки процесса тестирования, которые значительно сокращают расходы на него.

Для выполнения тестирования создается программная среда, которая обеспечивает запуск и выполнение тестируемого модуля, передаёт ему входные данные, и собирает реальные выходные данные, полученные в результате работы системы на заданных входных данных. После этого, программная среда тестирования должна сравнить реальные выходные данные, с ожидаемыми данными, и на основании выполненного сравнения сделать вывод о готовности модуля.

Программная среда тестирования может иметь следующий вид:



Можно выделить следующие инструментальные средства программной среды тестирования:

* во-первых, «организатор тестов», который управляет выполнением тестов;
* во-вторых, «генератор тестовых данных», который генерирует тестовые данные для тестируемой программы, а именно, выбирает тестовые данные из базы данных или использует шаблоны для генерации случайных данных;
* в-третьих, «программа оракул», которая генерирует ожидаемые результаты тестов;
* в-четвертых, «компаратор файлов», который сравнивает результаты текущего тестирования с результатами предыдущего тестирования и составляет отчет об обнаруженных различиях;
* в-пятых, «генератор отчетов тестирования», который формирует отчеты по тестам;
* в-шестых, «динамический анализатор», который добавляет в программу код подсчета количества выполнения каждого оператора;
* в-седьмых, «имитатор», который моделирует выполнение программы.

1. Валидация программных средств

Под валидацией программного средства понимается доказательство того, что в результате разработки системы были достигнуты цели, которые планировали достичь благодаря применению данной системы.

Таким образом, валидация представляет собой проверку соответствия программного средства ожиданиям заказчика.

В стандарте ANSI/IEEE 1012 («Планирование проверки (оценки) и подтверждения достоверности (валидация – аттестация) программных средств») понятие валидации определено, как систематическое, поэтапное тестирование компонентов разного уровня интеграции в течение всего жизненного цикла программных средств.

В стандарте оговорены единые требования, предъявляемые к формату и содержимому методик проверки программного средства и аттестации результатов тестирования. Данный стандарт обеспечивает всестороннюю оценку каждой фазы проекта программного средства и дает возможность гарантировать соблюдение следующих условий:

* во-первых, обнаружение и устранение ошибок на как можно более ранней стадии жизненного цикла ПС;
* во-вторых, снижение риска, связанного с проектом, затрат и действий, связанных с планированием разработки;
* в-третьих, повышение качества и надежности ПС;
* в-четвертых, улучшение обозримости организации проведения работ в процессе создания ПС;
* в-пятых, возможность быстро подвергнуть оценке предлагаемые изменения и их значение.

# Верификация и валидация ПО: технологии и инструменты

Сейчас как никогда актуальна задача обеспечения качества ПО, для решения которой сегодня предлагается множество инструментов верификации и валидации кода. При этом важно не только внедрить сами инструменты, развивать соответствующие компетенции и выстроить стратегию тестирования. Для устранения рисков, связанных с человеческим фактором, нужны развитые возможности автоматического обнаружения критических точек и дефектов.

Программное обеспечение лежит в основе почти любой инфраструктуры, и задача обеспечения его качества сегодня актуальна как никогда. Практически все компании уже используют в своей деятельности Интернет вещей, бизнес-аналитику, искусственный интеллект, облака, социальные сети и т. п. Традиционные ИТ и встроенные системы уступают место повсеместному ПО, а успех цифровой трансформации предприятий зависит от работы программных систем, отвечающих всем отраслевым требованиям к надежности и доступности сервисов.

Сегодня организации вкладывают около 30% бюджета, выделяемого на ИТ, в обеспечение качества и тестирование, что неудивительно – более половины всех систем являются критически важными для бизнеса. При этом компаниям и организациям необходимо максимально гибко реагировать на изменения и внедрять формализованные процессы и методы контролируемого выпуска безопасного ПО. Активно внедряются механизмы автоматического обновления ПО по Сети, методы DevOps и «непрерывная инженерия» (continuous software engineering), что повышает потребность в процессах непрерывной верификации и валидации, требующих намного более тщательного, чем еще совсем недавно, выполнения процедур испытания на общую работоспособность.

Методы Agile и непрерывной разработки ПО, направленные на повышение качества и удовлетворение требований пользователей, должны опираться на эффективные, удобные в применении инструменты, которые могут войти в арсенал как разработчиков ПО, так и пользователей. Кроме того, оценка качества может проводиться третьей стороной – аккредитованной лабораторией или сертификационным центром.

Выбор методов верификации и валидации ПО зависит от модели разработки (V-модель, каскадная, спиральная и т. д.) и стандарта (ISO/IEC 25000 SQUARE, ISO/IEC 12207:2017) (рис. 1)

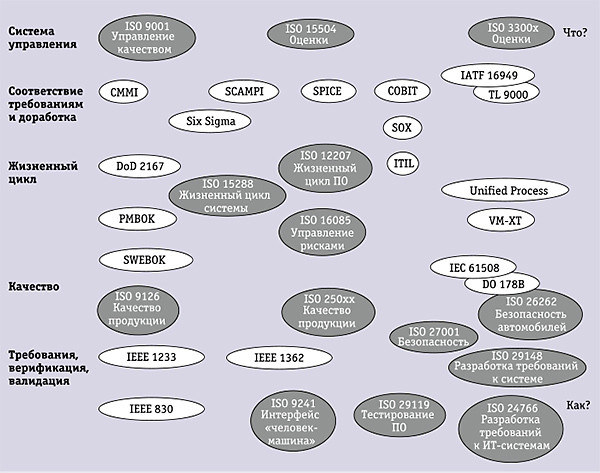


Рисунок 1 - Стандарты обеспечения качества процесса разработки ПО и конечного продукта

За качество собственно процесса разработки ПО отвечают стандарт ISO/МЭК 12207, регламентирующий процессы верификации и валидации, а также V-модель, в рамках которой каждой задаче разработки ставится в соответствие процесс тестирования. Например, модульный тест проверяет соответствие исходного кода низкоуровневой архитектуре, интеграционные тесты проверяют совместимость (интеграцию) ранее протестированных компонентов, системные тесты позволяют выяснить, соответствует ли полностью интегрированный продукт спецификациям, а приемочные тесты – отвечает ли продукт ожиданиям пользователя.

В стандартах серии ISO 25000, относящихся к качеству программного продукта, выделены следующие характеристики ПО:

* функциональная пригодность – степень соответствия продукта или системы заявленным функциям, объявленным и подразумеваемым потребностям при эксплуатации ПО в указанных условиях;
* сопровождаемость – удобство и гибкость модификации продукта или системы, ее корректировки или адаптации к изменениям среды или требований;
* удобство использования – простота работы с продуктом или системой для эффективного достижения пользователями указанных целей в заданном контексте применения;
* безопасность – степень, до которой продукт или система защищают данные при условии, что люди либо системы имеют доступ к данным в соответствии со своими привилегиями;
* относительная производительность – соотношение производительности и объема использованных ресурсов при заданных условиях.

На рис. 2 изображены элементы обеспечения качества процессов и продуктов и соответствующие инструменты. Данная схема может помочь в выборе средств верификации и валидации, которые можно задействовать на каждом из этапов V-модели и для каждого параметра качества ПО.

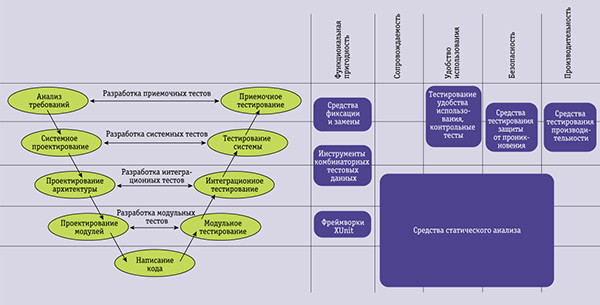


Рисунок 2 - Инструменты для проверки основных характеристик качества ПО

На этапе написания кода чаще применяются средства статического анализа, позволяющие непосредственно в интегрированной среде разработки контролировать соответствие кода стандартам. В то же время в рамках статического анализа проверяются особенности структуры модулей и архитектуры ПО, поэтому на рис. 2 поле средств статического анализа размещено на одном уровне с написанием кода, но захватывает тесты, касающиеся верификации архитектуры. Средства статического анализа имеют отношение ко всем четырем характеристикам качества.

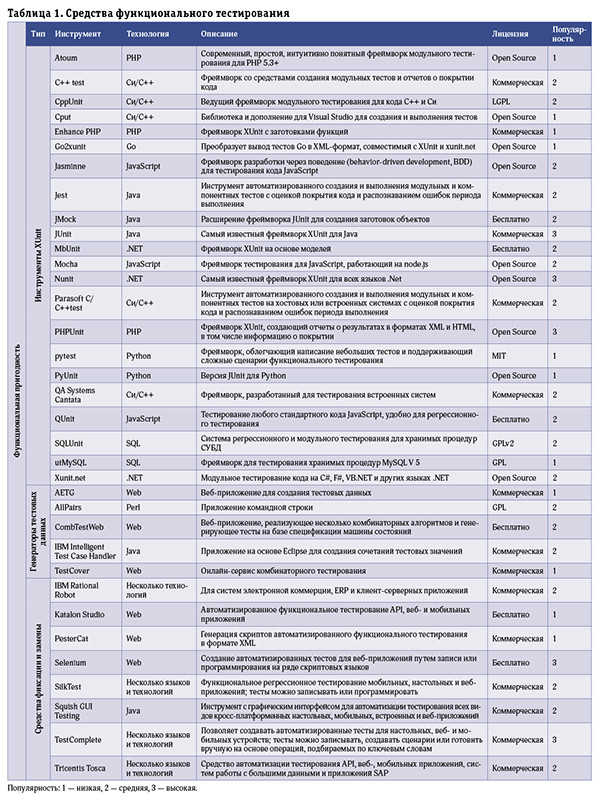
## Технологии и инструменты

Средства XUnit применяются для проверки правильности работы каждого разработанного модуля, при этом необходимо обеспечить максимальное покрытие кода. Фреймворки XUnit получили наиболее широкое распространение среди технологий автоматизации тестирования, позволяя на специальных языках описывать тестовые ситуации и автоматически их выполнять.

Средства комбинаторного тестирования генерируют тестовые данные, что дает возможность проверять корректность функционирования различных интегрированных модулей.

Инструменты фиксации и замены используются для проверки корректности и полноты функционирования системы, в том числе при проведении приемочных тестов. Эти средства регистрируют взаимодействия тестировщиков с приложением, генерируя тестовые сценарии, которые затем можно выполнять автоматически.

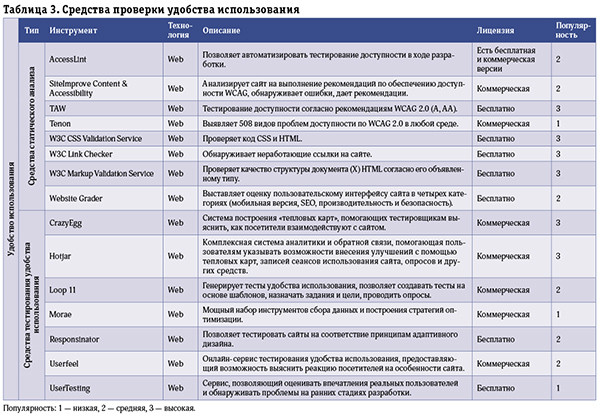
Инструментов тестирования так же много, как и языков программирования. В табл. 1 приведены основные средства тестирования для первой десятки наиболее популярных языков программирования (согласно рейтингу, Tiobe).



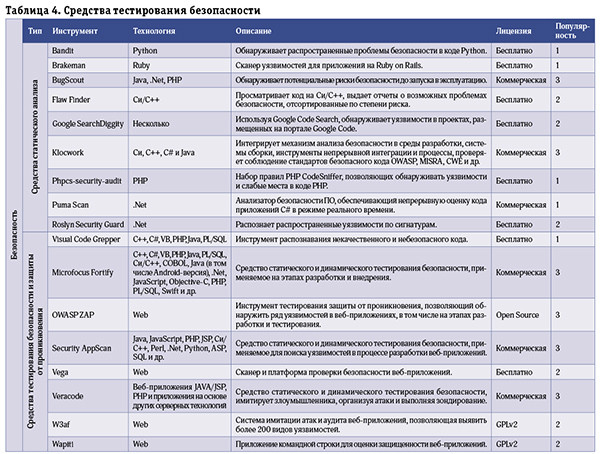
Инструменты контроля сопровождаемости. Позволяют анализировать исходный код и проверять его на соответствие правилам модульности, читаемости и др. (табл. 2).



Инструменты контроля удобства использования. Применяются для оценки программного продукта в процессе работы с ним реальных пользователей. В то же время такие инструменты позволяют проводить валидацию пользовательского интерфейса без участия самих пользователей (табл. 3).



Средства контроля безопасности. Позволяют обнаруживать уязвимости в системе и определять, защищает ли она данные при сохранении необходимой функциональности. В частности, средства испытания на защиту от проникновения имитируют атаки на программную систему или сеть с помощью сканирования и других действий, направленных на поиск и использование слабых мест. Их также называют инструментами этичного, или белого, хакерства (табл. 4).



Средства проверки производительности. Позволяют выяснить, насколько быстро система работает под нагрузкой (табл. 5). Эти средства не предназначены для поиска дефектов в приложении, а применяются для оценки измеримых характеристик: времени отклика, пропускной способности и т. д.



Средства непрерывной верификации и валидации. Принцип непрерывного обеспечения качества предполагает тесную интеграцию процессов верификации и валидации на всех этапах разработки. Типичный пример – разработка с ориентацией на тестирование: плановые показатели качества задаются еще до разработки ПО. С учетом преимуществ непрерывной интеграции, одной из основ DevOps, все большее значение придается инструментам, помогающим не только автоматизировать верификацию и валидацию, но и интегрировать эти процессы в цикл разработки. Перечисленные в табл. 6 инструменты позволяют проверять характеристики качества в рамках сред непрерывной интеграции Jenkins, Travis CI, Bamboo, GoCD, Ansible и т. п. Требуется встроить валидацию и верификацию в жизненный цикл ПО, обеспечив автоматическое, прозрачное для разработчиков выполнение соответствующих процессов. Представленные в табл. 6 инструменты проверяют несколько характеристик качества и позволяют получить данные для глобальной оценки и визуализации. В рамках процессов непрерывной верификации и валидации также могут применяться инструменты модульного тестирования, фиксации и замены и статического анализа.



Для управления действиями, выполняемыми при помощи всех перечисленных видов инструментов, и контроля над процессом верификации и валидации в целом, применяются средства управления тестовыми случаями: Test Link, Test Rail, Microfocus Quality Center, VSTS, IBM Rational Quality Manager, XStudio и др.

## Перспективы

Современное общество все сильнее зависит от программного обеспечения, которое становится двигателем всех отраслей экономики, в связи с чем растут и требования к качеству ПО. Применение методов и технологий автоматизации соответствующих процессов становится обязательным элементом защиты инфраструктур.

В современном мире становится все больше рисков, связанных с ростом количества угроз кибербезопасности и проблемами, обусловленными неудобством использования приложений. Это означает, что нужно ускорить развитие технологий верификации и валидации, а также удвоить усилия, направленные на повышение надежности систем. В частности, необходимы сценарии мягкой деградации возможностей ПО и его работы в условиях отказа.

Избежать обнаружения злоумышленниками критических уязвимостей и появления новых атак после выпуска продукта помогут непрерывная верификация и валидация, которые должны стать основой стратегии обеспечения качества ПО. Необходима возможность гибко вносить исправления и изменения с использованием беспроводных сетевых соединений. Понадобятся механизмы, препятствующие запуску систем в случае, если на них не установлены самые свежие обновления программного обеспечения. В число таких систем входят автомобили, производственные линии и иное оборудование, требующее особо высокого уровня безопасности. Еще в большей степени это касается медицинской техники — для нее нужна иерархическая система контроля качества.

Выбор систем верификации и валидации определяется многими факторами. В зависимости от применяемой среды разработки, в разных организациях соответствующие процессы имеют свои особенности и строятся на базе различных сочетаний инструментов. При этом важно не только внедрить сами инструменты, развивать соответствующие компетенции и выстроить стратегию тестирования. Для устранения рисков, связанных с человеческим фактором, нужны развитые возможности автоматического обнаружения критических точек и дефектов.

\*\*\*

С внедрением технологий искусственного интеллекта появляется потребность в обеспечении прозрачности работы соответствующих систем – необходимо понимание того, по каким правилам нейронная сеть определяет, можно ли предоставить клиенту кредит или как автомобиль-робот отреагирует на стечение нескольких опасных обстоятельств. Классические регрессионные тесты и средства отслеживания в подобных случаях не помогут. В средствах верификации и валидации новых поколений будет все больше интеллектуальных механизмов на основе больших данных, способных к анализу, самообучению и автоматическому улучшению качества ПО.

Аристотель говорил: «Совершенство – это не действие, а привычка». Широкий выбор инструментов – это важно, но главное – создание культуры обеспечения качества ПО и приобретение соответствующих навыков.

31.03.2023 г.

# Методы верификации ПО

В файле.

06/07.04.2023г.

# Виды тестирования

<https://habr.com/ru/post/549054/>

<https://testengineer.ru/chto-takoe-funkcionalnoe-testirovanie-mini-gajd/> <https://testengineer.ru/sanity-testirovaniye/>

<https://testengineer.ru/chto-takoe-obezjane-testirovanie-chem-otlichaetsja-ot-ad-hoc-testirovanija-chto-takoe-gorilla-test/>

Горилла тестирование – это многократное тестирование.

<https://testengineer.ru/nonfunctional-testing-guide/>

<https://testirovanieprob.blogspot.com/p/12.html>

<https://habr.com/ru/companies/redmadrobot/articles/280618/>

10.04.2023

# Ручное тестирование

<https://testengineer.ru/otlichiya-ruchnogo-i-avtomatizirovannogo-testirovaniya/>

<http://getbug.ru/ruchnoe-testirovanie-programmnogo-obespecheniya/>

<https://habr.com/ru/companies/skillbox/articles/418889/>

11.04.2023

# Отладка ПО и виды ошибок

<https://skarlupka.ru/articles.php?id=33>

<https://studfile.net/preview/9604807/page:8/#13>

13.04.2023

# Тестирование пользовательского интерфейса

<https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-7-ch1-testirovanie-ui-i-verstki.html>

<https://studfile.net/preview/3640848/page:3/>