Тестирование программного обеспечения — 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

Определения тестирования

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

* процесс выполнения программы с целью нахождения ошибок[[2]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-2);
* интеллектуальная дисциплина, имеющая целью получение надежного программного обеспечения без излишних усилий на его проверку[[3]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-3);
* техническое исследование программы для получения информации о её качестве с точки зрения определённого круга заинтересованных лиц ([*С. Канер*](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D0%BD%D0%B5%D1%80,_%D0%A1%D1%8D%D0%BC&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Cem_Kaner) [[*уточнить*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%9F%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D1%8F%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C#%D0%91%D1%80%D0%B5%D0%BC%D1%8F_%D0%B4%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0)]);
* проверка соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выполненных определённым образом[[1]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-SWEBOOK-1);
* процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо аспектов её работы[[4]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-4);
* процесс, имеющий целью выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации[[5]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-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]](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F#cite_note-6).

## 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

### Тестирование программ

**Тестирование**можно рассматривать, как процесс семантической отладки (проверки) программы, заключающийся в исполнении последовательности различных наборов контрольных тестов, для которых заранее известен результат. Т.е. тестирование предполагает выполнение программы и получение конкретных результатов выполнения тестов [[7.1](https://intuit.ru/studies/courses/2190/237/literature#literature.7.1)-[7.5](https://intuit.ru/studies/courses/2190/237/literature#literature.7.5), [7.11](https://intuit.ru/studies/courses/2190/237/literature#literature.7.11), [7.12](https://intuit.ru/studies/courses/2190/237/literature#literature.6.20)].

Тесты подбираются так, чтобы они охватывали как можно больше типов ситуаций алгоритма программы. Менее жесткое требование - выполнение хотя бы один раз каждой ветви программы.

Исторически первым видом тестирования была *отладка*.

*Отладка*- это проверка описания программного объекта на ЯП с целью обнаружения в нем ошибок и последующее их устранение. Ошибки обнаруживаются компиляторами при их синтаксическом контроле. После этого проводится *верификация* *по* проверке правильности кода и *валидация* *по* проверке соответствия продукта заданным требованиям.

Целью тестирования - проверка работы реализованных функций в соответствии с их спецификацией. На основе внешних спецификаций функций и проектной информации на процессах ЖЦ создаются функциональные тесты, с помощью которых проводится тестирование с учетом требований, сформулированных на этапе анализа *предметной области*. Методы *функционального тестирования* подразделяются на статические и динамические.

#### Статические методы тестирования

*Статические методы*используются при проведении инспекций и рассмотрении спецификаций компонентов без их выполнения.Техника статического анализа заключается в методическом просмотре (или обзоре) и анализе структуры программ, а также в доказательстве их правильности. Статический анализ направлен на анализ документов, разработанных на всех этапах ЖЦ и заключается в инспекции исходного кода и сквозного контроля программы.

*Инспекция ПО*- это статическая проверка соответствия программы заданным спецификациями, проводится путем анализа различных представлений результатов проектирования (документации, требований, спецификаций, схем или исходного кода программ) на процессах ЖЦ. Просмотры и инспекции результатов проектирования и соответствия их требованиям заказчика обеспечивают более высокое качество создаваемых ПС.

При инспекции программ рассматриваются документы рабочего проектирования на этапах ЖЦ совместно с независимыми экспертами и участниками разработки ПС. На начальном этапе проектирования инспекция предполагает проверку полноты, целостности, однозначности, непротиворечивости и совместимости документов с исходными требованиями к программной системе. На этапе реализации системы под *инспекцией*понимается анализ текстов программ на соблюдение требований стандартов и принятых руководящих документов технологии программирования.

Эффективность такой проверки заключается в том, что привлекаемые эксперты пытаются взглянуть на проблему "со стороны" и подвергают ее всестороннему критическому анализу.

Эти приемы позволяют на более ранних этапах проектирования обнаружить ошибки или дефекты путем многократного просмотра исходных кодов. Символьное тестирование применяется для проверки отдельных участков программы на входных символьных значениях.

Кроме того, разрабатывается множество новых способов автоматизации символьного выполнения программ. Например, автоматизированное средство статического контроля для языков ориентированной разработки, инструменты автоматизации *доказательства корректности* и автоматизированный аппарат сетей Петри.

#### Динамические методы тестирования

*Динамические методы тестирования*используются в процессе выполнения программ. Они базируются на графе, связывающем причины ошибок с ожидаемыми реакциями на эти ошибки. В процессе тестирования накапливается информация об ошибках, которая используется при оценке надежности и качества ПС.

Динамическое тестирование ориентировано на проверку корректности ПС на множестве тестов, прогоняемых по ПС, в целях проверки и сбора данных на этапах ЖЦ и проведения измерения отдельных показателей (число отказов, сбоев) тестирования для оценки характеристик качества, указанных в требованиях, посредством выполнения системы на ЭВМ. Тестирование основывается на систематических, статистических, (вероятностных) и имитационных методах.

Дадим краткую их характеристику.

Систематические методы тестирования делятся на методы, в которых программы рассматриваются как "черный ящик" (используется информация о решаемой задаче), и методы, в которых программа рассматривается как "белый ящик" (используется структура программы). Этот вид называют тестированием с управлением по данным или управлением по входувыходу. Цель - выяснение обстоятельств, при которых поведение программы не соответствует ее спецификации. При этом количество обнаруженных ошибок в программе является критерием качества входного тестирования.

Цель динамического тестирования программ по принципу "черного ящика" - выявление одним тестом максимального числа ошибок с использованием небольшого подмножества возможных входных данных.

*Методы "черного ящика"*обеспечивают:

* эквивалентное разбиение;
* анализ граничных значений;
* применение функциональных диаграмм, которые в соединении с реверсивным анализом дают достаточно полную информацию о функционировании тестируемой программы.

Эквивалентное разбиение состоит в разбиении входной области данных программы на конечное число классов эквивалентности так, чтобы каждый тест, являющийся представителем некоторого класса, был эквивалентен любому другому тесту этого класса.

Классы эквивалентности выделяются путем перебора входных условий и разбиения их на две или более групп. При этом различают два типа классов эквивалентности: правильные, задающие входные данные для программы, и неправильные, основанные на задании ошибочных входных значений.Разработка тестов методом эквивалентного разбиения осуществляется в два этапа: выделение классов эквивалентности и построение тестов. При построении тестов, основанных на выборе входных данных, проводится символическое выполнение программы.

Итак, методы тестирования по принципу "черного ящика" используются для тестирования функций, реализованных в программе, путем проверки несоответствия между реальным поведением функций и ожидаемым поведением с учетом спецификаций требований. Во время подготовки к этому тестированию строятся таблицы условий, причинно-следственные графы и области разбивки. Кроме того, подготавливаются тестовые наборы, учитывающие параметры и условия среды, которые влияют на поведение функций. Для каждого условия определяется множество значений и ограничений предикатов, которые тестируются.

*Метод "белого ящика"*позволяет исследовать внутреннюю структуру программы, причем обнаружение всех ошибок в программе является критерием исчерпывающего тестирования маршрутов потоков (графа) передач управления, среди которых рассматриваются:

* (а) критерий покрытия операторов - набор тестов в совокупности должен обеспечить прохождение каждого оператора не менееодного раза;
* (б) критерий тестирования ветвей (известный как покрытие решений или покрытие переходов) - набор тестов в совокупности должен обеспечить прохождение каждой ветви и выхода, по крайней мере, один раз.

Критерий (б) соответствует простому структурному тесту и наиболее распространен на практике. Для удовлетворения этого критерия необходимо построить систему путей, содержащую все ветви программы. Нахождение такого оптимального покрытия в некоторых случаях осуществляется просто, а в других является более сложной задачей.

Тестирование по принципу "белого ящика" ориентировано на проверку прохождения всех путей программ посредством применения путевого и имитационного тестирования.

*Путевое тестирование*применяется на уровне модулей и графовой *модели программы* путем выбора тестовых ситуаций, подготовки данных и включает тестирование следующих элементов:

* операторов, которые должны быть выполнены хотя бы один раз, без учета ошибок, которые могут остаться в программе иззабольшого количества логических путей и необходимости прохождения подмножеств этих путей;
* путей по заданному графу потоков управления для выявления разных маршрутов передачи управления с помощью путевых предикатов, для вычисления которого создается набор тестовых данных, гарантирующих прохождение всех путей. Однако все пути протестировать бывает невозможно, поэтому остаются не выявленные ошибки, которые могут проявиться в процессе эксплуатации;
* блоков, разделяющих программы на отдельные частиблоки, которые выполняются один раз или многократно при нахождении путей в программе, включающих совокупность блоков реализации одной функции либо нахождения входного множества данных, которое будет использоваться для выполнения указанного пути.

"Белый ящик" базируется на структуре программы, в случае "черного ящика", о структуре программы ничего неизвестно. Для выполнения тестирования с помощью этих "ящиков" известными считаются выполняемые функции, входы (входные данные) и выходы (выходные данные), а также логика обработки, представленные в документации.

#### Функциональное тестирование

Цель *функционального тестирования* - обнаружение несоответствий между реальным поведением реализованных функций и ожидаемым поведением в соответствии со спецификацией и исходными требованиями. Функциональные тесты должны охватывать все реализованные функции с учетом наиболее вероятных типов ошибок. Тестовые сценарии, объединяющие отдельные тесты, ориентированы на проверку качества решения функциональных задач.

Функциональные тесты создаются по внешним спецификациям функций, проектной информации и по тексту на ЯП, относятся к функциональным его характеристикам и применяются на этапе комплексного тестирования и испытаний для определения полноты реализации функциональных задач и их соответствия исходным требованиям.

В задачи *функционального тестирования* входят:

* идентификация множества функциональных требований;
* идентификация внешних функций и построение последовательностей функций в соответствии с их использованием в ПС;- идентификация множества входных данных каждой функции и определение областей их изменения;
* построение тестовых наборов и сценариев тестирования функций;
* выявление и представление всех функциональных требований с помощью тестовых наборов и проведение тестирования ошибок в программе и при взаимодействии со средой.

Тесты, создаваемые по проектной информации, связаны со структурами данных, алгоритмами, интерфейсами между отдельными компонентами и применяются для тестирования компонентов и их интерфейсов. Основная цель - обеспечение полноты и согласованности реализованных функций и интерфейсов между ними.

Комбинированный метод "черного ящика" и "прозрачного ящика" основан на разбиении входной области функции на подобласти обнаружения ошибок. Подобласть содержит однородные элементы, которые все обрабатываются корректно либо некорректно. Для тестирования подобласти производится выполнение программы на одном из элементов этой области.

Предпосылки *функционального тестирования*:

* корректное оформление требований и ограничений к качеству ПО;
* корректное описание модели функционирования ПО в среде эксплуатации у заказчика;
* адекватность модели ПО заданному классу.

[https://ru.wikipedia.org/wiki/Тестирование\_программного\_обеспечения#Стандарты,\_ realtek-pcie-gbe-family-controllerотносящиеся\_к\_тестированию](https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения#Стандарты,_относящиеся_к_тестированию)

<https://intuit.ru/studies/courses/2190/237/lecture/6130?page=1>

<https://unetway.com/tutorial/testing-software-standards-iso>