МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

СРС:

По Введению в ПИ на тему:

«Классификация тестирования»

Работа выполнена:

Атабекова А.Н. (ПИ-1-21)

Работа проверена:

Досболова Ф.К.

Бишкек 2022

Оглавление

**Введение**……………………………………………………………………… 4

**Глава 1.1** **По знанию системы**

Глава 1.1.2 Тестирование черного ящика (black box)

Глава 1.1.3 Тестирование белого ящика (white box)

Глава 1.1.4 Тестирование серого ящика (grey box)

**Глава 1.2** **По степени автоматизации**

Глава 1.2.2 Ручное тестирование (manual testing)

Глава 1.2.3 Автоматизированное тестирование (automated testing)

Глава 1.2.4 Полуавтоматизированное тестирование (semiautomated testing)

**Глава 2** **По объекту тестирования**

Глава 2.1 Тестирование интерфейса пользователя (UI testing)

Глава 2.2 Тестирование локализации (localization testing)

Глава 2.3 Тестирование производительности (Performance testing)

Глава 2.3.2 Нагрузочное тестирование (load testing)

Глава 2.3.3 Стресс-тестирование (stress testing)

Глава 2.3.4 Тестирование стабильности (stability /endurance / soak testing)

Глава 2.4 Юзабилити-тестирование (usability testing)

Глава 2.5 Тестирование безопасности (security testing)

Глава 2.6 Тестирование совместности (compatibility testing)

Глава 2.7 Функциональное тестирование (functional testing)

**Глава 3** **По степени изолированности компонентов и времени проведения тестирования**

Глава 3.2 Компонентное(модульное) тестирование (component / unit testing)

Глава 3.3 Интеграционное тестирование (integration testing)

Глава 3.4 Системное тестирование (system / end-to-end testing)

Глава 3.5 Альфа-тестирование (alpha testing)

Глава 3.5.2 Тестирование новой функциональности (new feature testing)

Глава 3.5.3 Регрессивное тестирование (regression testing)

Глава 3.5.4 Дымовое тестирование (smoke testing)

Глава 3.5.5 Подтверждающее тестирование (Confirmation testing)

Глава 3.5.6 Приемочное тестирование (**Acceptance testing**)

Глава 3.6 Бета-тестирование (beta testing)

**Заключение**

**Список литературы**

**Введение**

Тeстирoвaниe (testing) прoгрaммнoгo oбeспeчeния – это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле тестирование ПО – это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение:

Тестирование – важный этап процесса разработки ПО, потому что с его помощью в той или иной мере обеспечивается безопасность, надёжность и удобство создаваемого продукта. В настоящее время существует множество подходов и методик к решению задачи тестирования ПО, но эффективное тестирование сложных программных систем — процесс творческий, не сводящийся к следованию строгим и чётким правилам.

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

**Глава 1.1** **По знанию системы**

Глава 1.1.2 Тестирования черного ящика (black box testing)

Тестирование методом «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Согласно ISTQB, тестирование черного ящика – это:

* тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
* тест-дизайн, основанный на технике черного ящика – процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Тестируемая программа для тестировщика – как черный непрозрачный ящик, содержания которого он не видит. Целью этой техники является поиск ошибок в таких категориях:

* неправильно реализованные или недостающие функции;
* ошибки интерфейса;
* ошибки в структурах данных или организации доступа к внешним базам данных;
* ошибки поведения или недостаточная производительности системы;

Таким образом, мы не имеем представления о структуре и внутреннем устройстве системы. Нужно концентрироваться на том, что программа делает, а не на том, как она это делает.

**Преимущества:**

1. тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;
2. тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;
3. тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;
4. можно начинать писать тест-кейсы, как только готова спецификация.

**Недостатки:**

1. тестируется только очень ограниченное количество путей выполнения программы;
2. без четкой спецификации (а это скорее реальность на многих проектах) достаточно трудно составить эффективные тест-кейсы;
3. некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования.

Глава 1.1.3 Тестирование белого ящика (white box testing)

Тестирование методом белого ящика (также прозрачного, открытого, стеклянного ящика или же основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки. Знание всех особенностей тестируемой программы и ее реализации обязательны для этой техники. Тестирование белого ящика – углубление во внутреннее устройство системы за пределы ее внешних интерфейсов.

Согласно ISTQB: тестирование белого ящика – это:

* тестирование, основанное на анализе внутренней структуры компонента или системы;
* тест-дизайн, основанный на технике белого ящика – процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.

Тестируемая программа для тестировщика – прозрачный ящик, содержимое которого он прекрасно видит.

**Преимущества:**

1. тестирование может производиться на ранних этапах: нет необходимости ждать создания пользовательского интерфейса;
2. можно провести более тщательное тестирование с покрытием большого количества путей выполнения программы.

**Недостатки:**

1. для выполнения тестирования белого ящика необходимо большое количество специальных знаний;
2. при использовании автоматизации тестирования на этом уровне поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

Сравнение Black Box и White Box

|  |  |  |
| --- | --- | --- |
| Критерий | Черный ящик | Белый ящик |
| Уровни, к которым применима техника | * Приемочное тестирование * Системное тестирование | * Юнис-тестирование * Интеграционное тестирование |
| Кто выполняет | Тестировщик | Разработчик |
| Знание программирования | Не нужно | Необходимо |
| Знание реализации | Не нужно | Необходимо |
| Основа для тест-кейсов | Спецификация, требования | Проектная документация |

**Глава 1.1.4 Тестирование серого ящика (grey box testing)**

Тестирование методом серого ящика – метод тестирования программного обеспечения, который предполагает комбинацию White Box и Black Box подходов. То есть внутреннее устройство программы нам известно лишь частично. Предполагается, например, доступ ко внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники черного ящика, то есть с позиции пользователя.

Эту технику тестирования также называют методом полупрозрачного ящика: что-то мы видим, а что-то – нет.

**Глава 1.2** **По степени автоматизации**

**Глава 1.2.2 Ручное тестирование (manual testing)**

**Ручное тестирование (manual testing)**  — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя. В роли тестировщиков могут выступать и обычные пользователи, сообщая разработчикам о найденных ошибках.

При**ручном тестировании (manual testing)**тестировщики вручную выполняют тесты, не используя никаких средств автоматизации. Ручное тестирование – самый низкоуровневый и простой тип тестирования, не требующий большого количества дополнительных знаний.

Тем не менее, перед тем, как автоматизировать тестирование любого приложения, необходимо сначала выполнить серию тестов вручную. Мануальное тестирование требует значительных усилий, но без него мы не сможем убедиться в том, возможна ли автоматизация в принципе. Один из фундаментальных принципов тестирования гласит: 100% автоматизация невозможна. Поэтому, ручное тестирование – необходимость.

Тестирование может быть очень непростым занятием. Проведение тестирования для проверки максимально возможного количества путей выполнения, с использованием минимального числа тест-кейсов, требует серьезных аналитических навыков.

**Глава 1.2.3 Автоматизированное тестирование (automated testing)**

**Автоматизированное тестирование программного обеспечения** — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

Существует несколько основных видов автоматизированного тестирования:

* **Автоматизация тестирования кода** (Code-driven testing) – тестирование на уровне программных модулей, классов и библиотек (фактически, автоматические юнит-тесты).
* **Автоматизация тестирования графического пользовательского интерфейса** (Graphical user interface testing) – специальная программа (фреймворк автоматизации тестирования) позволяет генерировать пользовательские события– нажатия клавиш, клики мышкой, и отслеживание реакции программы на эти действия: соответствует ли она спецификации.
* **Автоматизация тестирования API** (Application Programming Interface) – тестирование программного интерфейса программы. Тестируются интерфейсы, предназначенные для взаимодействия, например, с другими программами или с пользователем. Здесь, опять же, как правило, используются специальные фреймворки.

**Глава 1.2.4 Полуавтоматизированное тестирование (semiautomated testing)**

Полуавтоматизированное тестирование – ручное тестирование с частичным использованием средств автоматизации - программного обеспечения (например, средств захвата/воспроизведения) для контроля выполнения тестов, сравнения полученных результатов с эталонными, установки предусловий тестов и других функций контроля тестирования и организации отчетов.

**Глава 2** **По объекту тестирования**

**Глава 2.1 Тестирование интерфейса пользователя (UI testing)**

**UI-тестирование — этап комплексного тестирования программного обеспечения, на этом этапе QA-специалисты проверяют качество всех компонентов интерфейса.**UI-тестирование помогает убедиться, что функции приложения отрабатывают на достижение целей и удобны для пользователей. UI-тестирование улучшает качество программного обеспечения и обеспечивает удобство приложения для пользователей.

**5 аспектов, которые изучает UI-тестирование:**

* Визуальный дизайн
* Функциональность
* Юзабилити
* Производительность
* Согласованность

UI-тестирование выполняют вручную и с помощью автоматизированных методов. Цель UI-тестирования — обеспечить соответствие спецификации.

**Глава 2.2 Тестирование локализации (localization testing)**

Тестирование локализации – это процесс тестирования локализованной версии программного продукта. Проверка правильности перевода элементов интерфейса пользователя, системных сообщений и ошибок, проверка перевода раздела «Помощь/Справка», сопроводительной документации и основного контента.

Задачами локализационного тестирования являются:

* устранение ошибок, которые нельзя выявить в процессе перевода и адаптации;
* проверка целостности перевода;
* финальный анализ качества перевода с точки зрения пользователя.

**Глава 2.3 Тестирование производительности (Performance testing)**

**Тестирование производительности (performance testing)** – тестирование ПО, позволяющее осуществлять оценку быстродействия программного продукта при определённой нагрузке. Задачей тестирования производительности является определение масштабируемости приложения под нагрузкой, при этом происходит:

* Измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций.
* Определение количества пользователей, одновременно работающих с приложением.
* Определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций).
* Исследование производительности на высоких, предельных, стрессовых нагрузках.

**Глава 2.3.2 Нагрузочное тестирование (load testing)**

**Нагрузочное тестирование (load testing)** – тестирование ПО, позволяющее осуществлять оценку быстродействия программного продукта при плановых, повышенных и пиковых нагрузках. Осуществление нагрузочного тестирования перед вводом системы в промышленную эксплуатацию позволяет избегать неожиданных потерь в производительности через полгода—год, когда система будет заполнена данными.

**Глава 2.3.3 Стресс-тестирование (stress testing)**

**Стресс-тестирование (stress testing)** – тестирование ПО, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования. Это проверка программы в таких стрессовых ситуациях как наличие большого объёма входных параметров, нехватка дискового пространства или маломощный процессор.

**Глава 2.3.4 Тестирование стабильности (stability /endurance / soak testing)**

**Тестирование стабильности (stability/endurance/soak testing)** – тестирование ПО, при котором проверяется работоспособность системы при длительном тестировании со средним уровнем нагрузки. Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом на первое место выходит отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты влияющие именно на стабильность работы.

**Глава 2.4 Юзабилити-тестирование (usability testing)**

**Тестирование удобства пользования -**это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

* **Производительность, эффективность**( **efficiency**) - сколько времени и шагов понадобится пользователю для завершения основных задач приложения, например, размещение новости, регистрации, покупка и т.д. (меньше - лучше).
* **Правильность**( **accuracy**) - сколько ошибок сделал пользователь во время работы с приложением (меньше - лучше).
* **Активизация в памяти**( **recall**) – как много пользователь помнит о работе приложения после приостановки работы с ним на длительный период времени (повторное выполнение операций после перерыва должно проходить быстрее, чем у нового пользователя).
* **Эмоциональная реакция**( **emotional response**) – как пользователь себя чувствует после завершения задачи - растерян, испытал стресс? Порекомендует ли пользователь систему своим друзьям? (положительная реакция - лучше).

**Глава 2.5 Тестирование безопасности (security testing)**

**Тестирование безопасности (security testing)** – тестирование ПО, которое проверяет фактическую реакцию встроенных в систему защитных механизмов на проникновение злоумышленников.

**Принципы безопасности программного обеспечения**

Общая стратегия безопасности основывается на трех основных принципах:

1. Конфиденциальность.
2. Целостность.
3. Доступность.

##### Конфиденциальность

Конфиденциальность - это сокрытие определенных ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей или, другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.

##### Целостность

Существует два основных критерия при определении понятия целостности:

1. **Доверие.** Ожидается, что ресурс будет изменен только соответствующим способом определенной группой пользователей.
2. **Повреждение и восстановление.**В случае, когда данные повреждаются или неправильно меняются авторизованным или не авторизованным пользователем, Вы должны определить, на сколько важной является процедура восстановления данных.

##### Доступность

Доступность представляет собой требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс, тем выше уровень доступности должен быть.

**Глава 2.6 Тестирование совместности (compatibility testing)**

Тестирование совместимости используется, чтобы убедиться, что ваше приложение совместимо с другими версиями ОС, различными оболочками и сторонними сервисами, а также аппаратным обеспечением устройства.

**Глава 2.7 Функциональное тестирование (functional testing)**

**Функциональное тестирование (functional testing)** – тестирование ПО, направленное на проверку реализуемости функциональных требований. При функциональном тестировании проверяется способность ПО правильно решать задачи, необходимые пользователям.

Тестирование функциональности может проводиться в двух аспектах:

* Требования.
* Бизнес-процессы.

**Преимущества функционального тестирования**:

* имитирует фактическое использование системы.

**Недостатки функционального тестирования**:

* возможность упущения логических ошибок в программном обеспечении;
* вероятность избыточного тестирования.

**Нефункциональное тестирование** описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, как система работает.

**Глава 3 По степени изолированности компонентов и времени проведения тестирования**

**Глава 3.2 Компонентное(модульное) тестирование (component / unit testing)**

**Модульное тестирование** – это процесс исследования ПО, при котором тестируется минимально возможный компонент, например отдельный класс или функция. Часто модульное тестирование осуществляется разработчиками ПО. **Компонентное (модульное) тестирование**проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.). Обычно компонентное (модульное) тестирование проводится вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки. Все найденные дефекты, как правило исправляются в коде без формального их описания в системе менеджмента багов (Bug Tracking System).

**Глава 3.3 Интеграционное тестирование (integration testing)**

**Интеграционное тестирование** – это процесс исследования ПО, при котором тестируются интерфейсы между компонентами или подсистемами.

Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

**Уровни интеграционного тестирования:**

* **Компонентный интеграционный уровень**( Component Integration testing) проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.
* **Системный интеграционный уровень**(System Integration Testing) - проверяется взаимодействие между разными системами после проведения системного тестирования.

**Глава 3.4 Системное тестирование (system / end-to-end testing)**

**Системное тестирование** – это процесс исследования ПО, при котором тестируется интегрированная система на её соответствие требованиям заказчика. Альфа- и бета-тестирование относятся к подкатегориям системного тестирования.

Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований ,дефекты в системе в целом. При этом выявляется неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д

**Глава 3.5 Альфа-тестирование (alpha testing)**

**Альфа-тестирование** – это процесс имитации реальной работы разработчиков с программным продуктом или реальная работа потенциальных пользователей с системой. 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.

**Глава 3.5.2 Тестирование новой функциональности (new feature testing)**

Тестирование новой функциональности должно происходить по мере её разработки, чтобы как можно быстрее дать обратную связь команде. Это очень важно, потому как уменьшает риск появления серьезных ошибок на этапе “причесывания” нового релиза. Возможно, что и ошибки в новой функциональности будут исправляться быстрее по “горячим следам” разработки.

**Глава 3.5.3 Регрессивное тестирование (regression testing)**

**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. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

Сам по себе термин "регрессионное тестирование", в зависимости от контекста использования, может иметь разный смысл. Сэм Канер, к примеру, описал 3 основных типа регрессионного тестирования:

* **Регрессия багов**(**Bug regression**) - попытка доказать, что исправленная ошибка на самом деле не исправлена.
* **Регрессия старых багов**(**Old bugs regression**) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.
* **Регрессия побочного эффекта**(**Side effect regression**) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

**Глава 3.5.4 Дымовое тестирование (smoke testing)**

**Дымовое тестирование (smoke testing)** — тестирование ПО, при котором выполняется набор тестов, после которого можно сказать, что программный продукт запускается. Если ошибок при запуске не происходит, то дымовой тест считается пройденным. Если программа не прошла дымовой тест, то её отправляют на доработку. Дело в том, что разработчики пишут различные компоненты приложения отдельно. После объединения компонентов может случиться так, что совместно они работать не cмогут. В таком случае нет смысла тестировать продукт в целом.

**Глава 3.5.5 Подтверждающее тестирование (Confirmation Testing)**

Подтверждающее тестирование направлено на проверку исправления бага. Суть его в том, что после исправление дефекта программное обеспечение может быть протестировано с использованием тестовых сценариев, которые завершились с ошибкой из-за найденного дефекта. То есть на новой версии программного обеспечения должны быть повторно выполнены шаги по воспроизведению сбоев, вызванных дефектом.

Целью подтверждающего тестирования является удостоверение в том, что найденный дефект был исправлен.

**Глава 3.5.6 Приемочное тестирование (Acceptance Testing)**

Приeмoчнoe тeстирoвaниe выпoлняeтся нa ocнoвaнии нaбopa типичныx тecтoвыx cлyчaeв cцeнaриeв, рaзрaбoтaнных нa oснoвaнии трeбований к данному приложению.  
Решение о проведении приемочного тестирования принимается, когда:

* Продукт достиг необходимого уровня качества.
* Заказчик ознакомлен с Планом Приемочных Работ (Product Acceptance Plan) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные и т.д.

Фaзa приeмoчнoгo тeстирoвaния длится дo тeх пoр, пoкa зaкaзчик нe вынoсит рeшeниe oб oтпрaвлeнии прилoжeния нa дoрaбoткy или выдaчe прилoжeния.

**Глава 3.6 Бета-тестирование (beta testing)**

**Бета-тестирование** – это распространение версий с ограничениями для некоторой группы лиц с целью проверки содержания допустимо минимального количества ошибок в программном продукте. В отличие от альфа-тестирования, проводимого силами штатных разработчиков или тестировщиков, бета-тестирование предполагает привлечение добровольцев из числа обычных будущих пользователей продукта, которым доступна предварительная версия продукта (так называемая **бета-версия**).

Тaкими дoбрoвoльцaми (их нaзывaют **бета-тестерами**) чaстo движeт любoпытствo к нoвoму прoдyктy — любoпытствo, рaди удoвлeтвoрeния кoтoрoгo oни вполне согласны мириться с возможностью испытать последствия ещё не найденных (а потому и не исправленных) ошибок. Кроме любопытства, мотивация может быть обусловлена желанием повлиять на процесс разработки и в итоге получать более удовлетворяющий их нужды продукт и многим другим.

**Заключение**

Тестирование (testing) прoгрaммнoгo oбeспeчeния – этo прoцeсс исслeдoвaния ПO c цeлью выявлeния oшибoк и oпрeдeлeния сooтвeтствия мeжду рeaльным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле тестирование ПО – это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение тестирования и анализ полученных результатов. AТестирование – важный этап процесса разработки ПО, потому что с его помощью в той или иной мере A обеспечивается безопасность, надёжность и удобство создаваемого продукта. В настоящее время существует множество подходов и методик к решению задачи тестирования ПО, но эффективное тестирование сложных программных систем — процесс творческий, не сводящийся к следованию строгим и чётким правилам.

**Список литературы:**

Курс лекций «Тестирование программного обеспечения» <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-2-ch4-vidi-i-napravleniya-testirovaniya.html>

Википедия <https://ru.wikipedia.org>

Лекция «тестирование на основе моделей» В.В. Кулямин http://panda.ispras.ru/~kuliamin/lectures-mbt/Lecture02.pdf