ПЛАН ТЕСТИРОВАНИЯ
КЛИЕНТ-СЕРВЕРНОЙ СИСТЕМЫ

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

В настоящем плане тестирования описаны и определены стратегия и принципы тестирования, применяемые при тестировании системы удаленного доступа. План будут использовать исполнители работ для получения представления о тестировании на проекте. Документ определяет распределение обязанностей при тестировании и описывает тесты, намеченные к выполнению.

План тестирования разработан для решения следующих задач.

- Спланировать управление тестированием и техническую поддержку тестирования в ходе всего жизненного цикла разработки системы.
- Определить исчерпывающий план тестирования, который описывает природу и рамки тестирования, достаточные для достижения целей и решения задач тестирования в проекте.

1.1. Цели тестирования

Основными целями тестирования являются:

- обеспечение выполнения всех системных требований и критериев, установленных к программному продукту;
- повышение вероятности того, что приложение при любых обстоятельствах будет функционировать надлежащим образом и соответствовать установленным требованиям за счет обнаружения максимально возможного числа дефектов;
- обеспечение работоспособности каждого разрабатываемого модуля согласно спецификации требований к данному модулю;
- обеспечение работоспособности всей системы в целом согласно спецификации требований к системе;
- обеспечение отказоустойчивости системы и каждого отдельного модуля;
- обеспечение установленных параметров производительности;
- обеспечение нормального качества исходных материалов и исходных кодов;
- оперативное информирование заинтересованных лиц об уровне качества регулярных сборок;
- обеспечение пользователя наиболее удобным графическим интерфейсом.

1.2. Стратегии тестирования

Основными задачами тестирования являются:

- проведение функционального тестирования каждого модуля и компонента системы для обеспечения его соответствия функциональным требованиям;
- проведение комплексного тестирования для обеспечения взаимодействия модулей и компонентов друг с другом согласно требованиям к системе;
- определение и максимальное увеличение производительности системы и каждого отдельного модуля;
- проведение нагрузочного тестирования для обеспечения отказоустойчивости системы и каждого отдельного модуля;
- максимальная автоматизация процесса тестирования;
- разработка достаточного набора контрольных примеров для тестирования новых модулей и компонентов;
- своевременная разработка контрольных примеров для покрытия устраняемых ошибок;
- увеличение покрытия кода тестовыми примерами;
- тестирование удобства применения модулей, имеющих графический интерфейс.

1.3. Виды тестирования

Для решения указанных выше задач тестирования будут использоваться следующие виды тестирования.

Тестирование — процесс многогранный, и указанные ниже виды могут пересекаться. Конкретный список видов тестирования для каждого модуля приводится в задании на тестирование.

- *Анализ спецификаций требований к каждому модулю и компоненту* — подготовка и определение параметров тестирования каждого отдельного компонента системы.
- *Анализ спецификаций требований к системе* — подготовка и определение параметров тестирования всей системы в целом.
- *Ручное тестирование* — выполнение тестировщиком прохода тестового цикла вручную, с последующей

ручной фиксацией результатов по каждому тесту в отчете.

• *Автоматизированное тестирование* — автоматический проход тестового цикла, с последующим автоматическим уведомлением заинтересованных лиц о результатах.

• *Смешанное тестирование* — вариант объединения ручного и автоматизированного тестирования. Наиболее часто используется в практике.

• *Дымовое тестирование* — простейший вид тестирования, основанный на определении успешности сборки системы из ветви исходного кода, находящейся в разработке. Обычно проводится один раз в день.

• *Ежедневное тестирование* — часть тестового цикла, обязательная к проходу каждый день.

• *Модульное тестирование* — **самый важный вид тестирования,** основанный на проверке работоспособности функций, методов и свойств в условиях их нормального и ошибочного исполнения. Это тестирование проводится на уровне исходного кода каждого существующего класса. Что нужно тестировать на данном этапе:

- класс правильно объявлен;
- структура класса соответствует спецификации требований;
- класс имеет достаточную функциональность;
- класс совместим со средствами автоматической обработки кода (построение автодокументации, анализ покрытия, качества кода и т.п.);
- некорректное функционирование и ошибочные ситуации корректно обрабатываются;
- класс совместим со связанными классами в рамках используемого наследования, полиморфизма, процедур вызова и т.п.;
- время выполнения, частота выполнения, нагрузка на ресурсы соответствуют требованиям;
- класс не содержит утечек памяти и других ресурсов.

В идеале необходимо покрыть тестами каждую строку исходного кода. Когда продукт находится на ранней стадии разработки — исправление ошибок обходится гораздо дешевле. По мере продвижения разработки выявление и исправление ошибок становится все более и более затратным. В большинстве случаев предоставление модульных тестов является ответственностью разработчика, их создание производится в момент разработки класса.

• *Интеграционное тестирование* — после разработки тестов на отдельные классы необходимо проверить, как они будут работать вместе в рамках одного исполняемого процесса. Необходимо проверить, как соотносятся классы, разработанные по-разному разными разработчиками. Данный вид тестирования базируется на предыдущем и также производится на уровне исходного кода. Обычно тестовые примеры строятся на основе вызова одного компонента из другого.

• *Сквозное тестирование* — проверяет работоспособность компонентов системы на уровне взаимодействия нескольких отдельных исполняемых процессов. На данном этапе тестируется функционирование клиент-серверных систем, их взаимодействие внутри и с внешними компонентами. Обычно сквозные тесты содержатся во внешнем по отношению к системе процессе, который делает тестирующие запросы. Так проверяются функции макроуровня, надежность, производительность, координация.

• *Функциональное тестирование* — рассматривает продукт, состоящий из множества классов, процессов, компонентов, данных как единое целое. На этом этапе проверяется в целом его работоспособность, функциональные и технические характеристики, а также бизнес-логика. Такая проверка может осуществляться в нескольких конфигурациях окружения оборудования и наборов данных.

• *Тестирование интерфейса* — проверка клиентских и административных интерфейсов пользователя на возможность выполнения с их помощью сценариев использования. Сценарий использования представляет собой последовательность действий пользователя, которые имитируют его активность при работе с интерфейсами системы. Сценарий использования должен покрывать спецификацию требований к пользовательскому интерфейсу. Такое тестирование производится в автоматическом режиме с помощью специализированных утилит. Тестирование должно проверять корректность работы интерфейсной части приложения при любых возможных настройках экрана (различное разрешение, масштаб, шрифт), при изменениях фокуса, при работе с мышью и клавиатурой.

• *Нагрузочное тестирование* — определение и проверка характеристик производительности системы в заданной конфигурации оборудования и набора данных.

• *Тестирование базы данных* — проверка функционирования внешней базы данных и хранимых процедур в соответствии со спецификацией требований. Проверка политики безопасности доступа к базе в соответствии с ролями системы. Определение и проверка характеристик базы данных, таких как производительность, среднее время доступа, максимальное количество обслуживаемых клиентов, минимальная и максимальная длительность обработки запроса и т.п.

• *Тестирование безопасности* — определение ролей и проверка списка функций системы, доступных для каждой роли. Может осуществляться на уровне интерфейса, на уровне компонента, на уровне базы данных, на уровне модуля и на сетевом уровне. Проводится в соответствии с принятым документом «Политика безопасности». Включает проверку методов шифрования данных при хранении и передаче, отказа доступа к запрещенным функциям, перехвата данных, подделки удостоверения личности, отказа в обслуживании и других атак.

• *Тестирование удобства использования интерфейса* — разработка отчета об удобстве использования, быстроте освоения, наглядности пользовательских интерфейсов системы. Данный отчет может содержать предложения по улучшению этих характеристик.

• *Тестирование конфигурации* — проверка работоспособности системы в заданном окружении конфигурации оборудования и набора данных.

• *Верификация* — проверка успешности исправления разработчиком ошибки, проведенная тестировщиком в тестовом окружении.

• *Регрессивное тестирование* — повторное выборочное тестирование продукта с модифицированными частями после исправления ошибки или добавления новой функции. Внесение изменений в исходный код может повлечь цепочку зависимостей и получение новых ошибок во взаимозависимых функциях. Данный вид тестирования минимизирует риск подобного события.

• *Инспекции и критические просмотры* — регулярное или по требованию исследование системы и отдельных ее компонентов с целью:

- выявления слабых мест;
- определения степени соответствия стандартам и требованиям;
- определения тенденций развития;
- разработки новых архитектурных решений;
- выработки предложений по рефакторингу кода;
- улучшения качественных и функциональных характеристик.

• *Анализ исходного кода* — регулярное исследование исходного кода с целью определения степени его соответствия документу «Требования к оформлению исходного кода». Разработка предложений по рефакторингу.

• *Анализ покрытия тестами кода* — автоматическое определение областей кода, не затронутых контрольными тестами, с целью разработки новых тестов для максимизации покрытия.

• *Тестирование инсталляции* — проверка корректной работы инсталляционного пакета, инсталляционных сценариев для копирования, обновления и последующей автоматической настройки работоспособности системы.

• *Тестирование документации* — проверка документации на полноту описания инструкций пользования в соответствии с «Требованиями по разработке пакета рабочей документации пользователя и администратора системы».

• *Финальное тестирование релиза* — тестирование, проводимое как последняя стадия разработки релиза, которое обычно состоит из двух частей:

- *alpha-тестирование*, проводимое в соответствии с формальными требованиями на тестовой площадке компании разработчика;
- *beta-тестирование*, завершающая стадия тестирования, возникающая при использовании релиза в «реальном мире» на площадке компании заказчика.

1.4. Документирование

В процессе разработки и проведения тестовых примеров обязательным является их документирование. Документация должна в любой момент времени давать следующую информацию:

- полный список тестов;
- задание на тестирование каждого компонента;
- список тестов по каждому компоненту;
- каждый пример должен быть хорошо комментирован;
- хронологический архив результатов тестирования.

2. ЦИКЛ ТЕСТИРОВАНИЯ

В данной главе описан процесс тестирования, который состоит из следующих видов активности в порядке приоритета: срочная внеплановая активность, тестирование релиза, ежедневная плановая активность, разработка новых тестов, полугодичная активность.

2.1. Срочная активность

Заключается в выполнении тестировщиком срочных поручений менеджера проекта, большая часть из которых является критическими и блокирует ход разработки.

В данный вид активности также включается проведение верификации критических ошибок и доработок. Тестировщику следует внимательно подходить к верификации, внесение изменений оказывает влияние на другие участки кода и может вносить дополнительные ошибки. Необходимо тщательно проверять предполагаемые зависимые участки кода таким образом, чтобы верификация приближалась к регрессивному тестированию по каждому инциденту.

Менеджерам следует понимать, что большое количество срочности срывает выполнение плановых работ и ведет к уменьшению тестирования. Для этого необходимо планировать загрузку и увеличивать ритмичность работы тестирования.

2.2. Тестирование релиза

Данная процедура носит временный характер, стартуя в момент начала финального тестирования и заканчиваясь после принятия релиза. Обычно целью тестирования релиза является сдача продукта с определенными характеристиками к определенному сроку.

Обязательным атрибутом данного вида тестирования является проверка обеспечения функциональных характеристик продукта в соответствии со спецификацией требований. Для этого проводится тщательный анализ спецификаций требований к системе в целом и к каждому компоненту.

Важным является определение отличий текущего релиза от предыдущего и проведение по этим отличиям регрессивного тестирования.

Тестировщик разрабатывает инсталляционный пакет.

В процессе тестирования релиза удостоверяется успешное прохождение предыдущего ежедневного автоматического теста, выполняется функциональное alpha- и beta-тестирование, производится тестирование инсталляции и документации, проводятся дополнительные ручные тесты. Также осуществляется консультирование заказчика по вопросам переноса и внедрения системы.

В процессе тестирования релиза тестировщик открывает новые срочные инциденты.

2.3. Ежедневная активность

Ежедневная активность тестирования заключается в проведении запланированных автоматических тестов на тестовом стенде.

Каждую ночь производится попытка осуществить сборку системы из исходного кода. Таким образом выполняется операция «дымового» тестирования. В случае успеха осуществляется автоматический проход запланированных тестов во всех запланированных конфигурациях. В результате формируется отчет, который рассылается всех заинтересованным лицам. Сбой дымового тестирования является критической ошибкой и подлежит немедленному анализу.

В данный вид активности также включается проведение верификации некритичных ошибок и доработок в соответствии. На каждый открытый инцидент желательно написать тесты, проверяющие его. Повторно возникающие инциденты должны повлечь за собой разработку дополнительных тестов.

Тестировщик контролирует ход тестирования и получает ежедневный отчет о ходе тестирования. Он управляет включением тестов, настраивает тестовый стенд, конфигурирует необходимые окружения для проведения тестов.

Тестировщик постоянно анализирует спецификацию требований к системе, технические задания и регулярно

выдает рекомендации по улучшению тестирования, а также сообщает разработчикам о расхождениях функциональности описанной в задании и реальных характеристик системы, если таковые появляются.
Тестировщик осуществляет также дополнительные ручные тесты, которые необходимы. Действия, повторенные более чем три раза, нуждаются в последующей автоматизации.
Задачей ежедневной активности тестирования является также постоянная автоматизация тестирования.
В процессе усовершенствования системы некоторые контрольные примеры могут перестать выполняться успешно, например, при изменении пользовательского интерфейса. Поэтому ежедневной задачей тестирования является анализ причин неудач, обновление и исправление контрольных примеров.
В процессе тестирования тестировщик открывает новые инциденты по результатам тестирования, назначая их разработчикам.

2.4. Разработка новых тестов

Разработку новых тестовых блоков выполняет разработчик в процессе создания и доработки системы. По завершении разработки блока тестировщик включает его проход в планировщик задач автоматического тестирования.
Тестировщик, при наличии времени, также может принимать участие в разработке тестов как разработчик. Тестировщик может принимать участие в первоначальной разработке базового набора модульных тестов, а также дополнять список тестов в процессе анализа и улучшения тестирования.
Обычно ответственность за разработку автоматических тестов распределяется следующим образом:

Планировщик автоматического тестирования	Тестировщик, разработчик
Дымовое тестирование	Тестировщик
Модульное тестирование	Разработчик
Интеграционное тестирование	Разработчик
Сквозное тестирование	Разработчик
Нагрузочное тестирование	Разработчик, тестировщик
Тестирование конфигурации	Тестировщик
Тестирование базы данных	Разработчик, тестировщик
Тестирование безопасности	Разработчик, тестировщик
Тестирование интерфейса	Тестировщик, разработчик

2.5. Полугодичная активность

Производится один раз в полгода по запросу менеджера или после сдачи очередного релиза. В результате формируется отчет о текущем положении дел с предложениями по улучшению. Данная активность включает в себя проведение:

Инспекция и критический просмотр кода	Руководитель, проектировщик, главный программист
Тестирование удобства использования интерфейса	Тестировщик
Анализ качества исходного кода	Тестировщик
Анализ покрытия тестами кода	Тестировщик

3. ТЕСТОВЫЙ СТЕНД

3.1. Планировщик задач автоматизированного тестирования

Необходимо разработать программное обеспечение на любом из скриптовых языков, которое позволит тестировщику конструировать и проигрывать необходимые тестовые последовательности в заданное время.
Данное программное обеспечение осуществляет ежедневную сборку, дымовое тестирование и автоматически запускает установленные блоки тестирования. Необходимо обеспечить поддержку управления несколькими компьютерами и различными виртуальными машинами из данного планировщика.
Планировщик обеспечивает интерфейс для формирования отчетов по результатам тестирования. Данные отчета представляют собой пронумерованный и разделенный список результатов пройденных контрольных примеров. В

случае ошибки теста ее текст также сохраняется в отчет. Результатом является успех или код ошибки.
Минимальным заданием является блок тестов. Блок тестов реализуется как отдельный исполняемый файл, который может содержать модульные, интеграционные, нагрузочные и т.п. тесты.
Необходимо разработать шаблон такого файла. Данный шаблон должен предоставлять функции записи результатов тестирования для планировщика. Необходимо разработать механизм интеграции средства автоматизированного тестирования GUI для проведения тестирования интерфейса пользовательских приложений.
После завершения всего тестирования производится автоматическая экспертная оценка по проведенному тестированию. Отчет анализируется, и формулируется общее заключение. Помимо этого планировщик предоставляет возможность проведения экспертного анализа. Производится расчет общего процента успехов при выполнении тестов.
Планировщик осуществляет слежение за запусками, формирует лог запусков и сохраняет результаты каждого прохода в отдельной папке.
Обработанный отчет высылается по электронной почте списку заинтересованных лиц. Этот список можно конфигурировать.

3.2. Конфигурации оборудования

Для работы системы требуется следующие тестовые конфигурации серверов и клиентских ПК. Для исполнения тестовых процедур необходим тестовый стенд, состоящий из двух серверов конфигурации №1, одного сервера конфигурации №2, по одному ПК конфигураций №2 и №3.

Конфигурация №1 (Сервер):

Конфигурация №2 (Клиент):

Конфигурация №3 (Удаленный клиент — минимальная):

3.3. Конфигурации и расположение данных

Система тестируется в единственном окружении, которое используется у заказчика:

- Сервер SRV1 с установленным массивом данных. IP = x. В тестовом массиве документов 30 шт. Конфигурация сервера следующая: x.
- Сервер SRV2 с установленным сервером приложений. IP = x. PDFServer установлен в папку x. Конфигурация следующая: x.
- Клиентский ПК CLI3 с установленными клиентским и административным интерфейсами в папках x.
- Сервер SRV1 подключен к серверу SRV2 напрямую через сетевой интерфейс 1 Gbps.
- ПК CLI3 подключен к SRV2 через свитч 100 Mbps и другой сетевой интерфейс 100 Mbps.
- На сервере SRV2 и клиентском ПК CLI3 запускаются клиентские и административные интерфейсы.
- На сервере SRV2 установлена база данных SQL Server в папке x, имя instance x. Настройки ODBC: x.
- Встроенные брандмауэры отключены.

3.4. Тестируемые компоненты

На основе данного плана формулируются задания на тестирование конкретных модулей в качестве Приложений к Плану тестирования. Далее идет общий список компонентов:

Все тестовые процедуры проводятся на тестовом стенде, компоненты системы тестируются на следующих конфигурациях:

1		№1, №2
2		№1, №2
3		№1, №2
4		№1, №2
5		№1, №2, №3
6		№1, №2, №3
7		№1, №2
8		№1, №2
9		№1, №2
10		№1
11		№1, №2, №3
12		№1, №2, №3
13		№1, №2, №3
15		№1, №2, №3