МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)
09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль))

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Разработка методики облачного тестирования программного обеспечения»

Студент И.Д. Сафиуллов
(И.О. Фамилия) (личная подпись)

Научный руководитель канд. пед. наук, доцент Е.В. Панюкова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Важнейшей задачей современного вендора ИТ-компании, занимающейся разработкой и продажей программного обеспечения, является сокращение непроизводственных затрат, в том числе на содержание и обслуживание собственной ИТ-инфраструктуры.

Одним из наиболее эффективных способов решения данной проблемы является переход на применение облачных сервисов.

Облачные вычисления - это новейший вычислительный стандарт, который широко применяется для поддержки задач проектирования программного обеспечения (ПО), и в том числе, для его тестирования.

Как показывает практика, такое ИТ-направление, как облачное тестирование, помогает обеспечить потребность вендоров ПО в аппаратных и программных ресурсах за счет применения гибкой и относительно недорогой облачной платформы для тестирования.

Однако важно отметить, что для реализации широких возможностей облачного тестирования необходимо использовать эффективную методику тестирования.

Методика тестирования - концептуальная основа, применимая к организационным процессам тестирования, процессам менеджмента тестирования и/или процессам динамического тестирования, чтобы упростить тестирование [2].

Таким образом, разработка эффективной методики облачного тестирования ПО представляет **актуальность** и научно-практический интерес.

**Объектом исследования** магистерской диссертации является облачное тестирование ПО.

**Предметом исследования** является методика облачного тестирования ПО.

**Цель исследования** - сокращение затрат на проектирование ПО за счет применения предлагаемой в работе эффективной методики облачного тестирования.

Для достижения поставленной в работе цели необходимо решить следующие задачи:

1. Проанализировать современные технологии облачного тестирования ПО.
2. Проанализировать методологические подходы к разработке методик облачного тестирования ПО.
3. Разработать методику облачного тестирования ПО.
4. Оценить эффективность разработанной методики облачного тестирования ПО.

**Гипотеза исследования:** применение разработанной в рамках диссертационного исследования методики облачного тестирования обеспечит сокращение затрат на проектирование ПО и повысит эффективность деятельности вендоров ПО.

**Методы исследования**. В процессе исследования использованы следующие методы и подходы: облачные вычисления, методы и средства облачного тестирования, объектно-ориентированный подход.

**Новизна исследования** заключается в разработке методики облачного тестирования, обеспечивающей сокращение затрат на проектирование ПО.

**Практическая значимость** исследования заключается в возможности практического применения предлагаемой методики облачного тестирования вендорами ПО в процессе проектирования последнего.

**Теоретической основой** диссертационного исследования являются научные труды российских и зарубежных ученых и специалистов, занимающихся проблемами облачного тестирования ПО.

**Основные этапы исследования**: исследование проводилось с 2017 по 2019 года в несколько этапов:

На первом этапе (констатирующем этапе) – формулировалась тема исследования, выполнялся сбор информации по теме исследования из различных источников, проводилась формулировка гипотезы, определялись

постановка цели, задач, предмета исследования, объекта исследования и выполнялось определение проблематики данного исследования.

Второй этап (поисковый этап) – в ходе проведения данного этапа осуществлялся анализ существующих методик и методологий облачного тестирования ПО, были разработаны методики облачного тестирования для приложений 1С, проводилось написание и публикация научной статьи по теме исследования в сборнике научных статей.

Третий этап (оценка эффективности) – осуществлялась оценка эффективности предлагаемых методик облачного тестирования на примере приложений 1С8 и 1С:Битрикс, были сформулированы выводы о полученных результатах по проведенному исследованию.

**На защиту выносятся**:

1. Методика облачного тестирования ПО.
2. Результаты оценки эффективности предлагаемой методики облачного тестирования ПО.

По теме исследования опубликована 1 статья [4].

Диссертация состоит из введения, трех глав, заключения и списка литературы.

В первой главе даны обзор и анализ технологий облачного тестирования ПО. Описаны основные положения, задачи, методы и виды облачного тестирования ПО. Произведен обзор и анализ существующих методик облачного тестирования ПО.

Во второй главе рассмотрены современные подходы к разработке методик облачного тестирования ПО. Проанализированы методологии облачного тестирования ПО. Дан обзор и анализ популярных облачных платформ.

Третья глава посвящена разработке методик облачного тестирования приложений 1С. Разработаны методики облачного тестирования приложений 1С8 и 1С:Битрикс.

В четвертой главе произведена оценка эффективности методик облачного тестирования 1С. Реализованы и проверены методики облачного тестирования приложений 1С8 и 1С:Битрикс.

В заключении приводятся результаты исследования.

Работа изложена на 76 страницах и содержит 25 рисунков, 9 таблиц, 42 источника.

# Глава 1 ОБЗОР И АНАЛИЗ ТЕХНОЛОГИЙ ОБЛАЧНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 1.1 Основные положения и задачи облачного тестирования программного обеспечения

Тестирование программного обеспечения - это вид деятельности в области разработки программного обеспечения. Это исследование, проводимое в отношении программного обеспечения для предоставления заинтересованным сторонам информации о его качестве.

ИТ-компании, проводящие тестирование в целом и тестирование на нагрузку, тестирование производительности и мониторинг производства, в частности, сталкиваются с рядом таких проблем, как ограниченный бюджет, соблюдение сроков, высокие затраты на тесты, большое количество тест-кейсов, ограничения на повторное использование тестов и т.д. [26].

Решить данные проблемы можно с помощью облачного тестирования.

Облачные вычисления стали новой вычислительной парадигмой, в которой облако может предоставлять удаленно размещенные виртуальные аппаратные и программные ресурсы и предлагать пользователям модель обслуживания по требованию.

Именно облачное тестирование стало новым подходом к тестированию, благодаря которому среды облачных вычислений используются для применения современных видов тестирования, обеспечивающих высокую эффективность данного процесса при минимизации затрат организаций-пользователей.

Облачное тестирование - это форма тестирования программного обеспечения, при которой тестирование проводится с использованием ресурсов через облачные приложения в среде облачного тестирования.

Как показывает успешный опыт известных вендоров ПО, эффективное неограниченное хранилище, быстрая доступность инфраструктуры с масштабируемостью, гибкость и доступность распределенной среды

тестирования сокращают время выполнения тестирования больших приложений и приводят к экономически эффективным решениям.

Основными задачами облачного тестирования являются:

− подтверждение качества облачных приложений, включая их функциональность, бизнес-процедуры, производительностью и масштабируемость, с учетом набора требований, предъявляемых к приложениям;

− проверка облачной совместимости в облачной инфраструктуре.

Для запуска набора текст-кейсов в облачном приложении необходимо выполнить следующие действия:

− создание и настройка облачных вычислений;

− запуск облачных вычисления;

− загрузка тестируемых приложений и тестовых данных в облако;

− запуск тестов пользователя;

− получение результатов теста.

Весь процесс занимает много времени и подвержен ошибкам, но его можно упростить путем выполнения некоторых тестов на облачных машинах с автоматизацией.

Ведущие облачные провайдеры и предприятия внедряют центры обработки данных, специально предназначенные для облачных вычислений.

Как показывает практика, операторы центров обработки данных, поставщики сетевого оборудования, производители и поставщики высокопроизводительных маршрутизаторов и коммутаторов, устройств хранения данных, вычислительных платформ и средств безопасности сталкиваются с новыми проблемами в облачном тестировании при решении задач оценки и оптимизации.

Выделено сеть категорий облачного тестирования:

1. Виртуальная инфраструктура.

Виртуальная коммутация позволяет создавать такие топологии сети, как виртуальные машины, и связывать их на программном уровне. Поскольку

производительность сильно варьируется в зависимости от общей нагрузки на сервер, очень важно протестировать масштабируемость виртуального коммутатора, чтобы удовлетворить огромные требования к пропускной способности на уровне коммутации доступа.

Многопользовательские облака должны быть проверены, чтобы гарантировать, что соглашения об уровнях обслуживания могут быть выполнены при стрессовых событиях, таких как повышенная мобильность виртуальных машин.

2. Инфраструктура хранения данных.

Трафик хранилища использует большую доступную полосу пропускания в конвергентной сети LAN/ SAN и требует преференциального режима. Ключевые показатели, включая пропускную способность ввода-вывода, чтение / запись, ошибки и задержки, могут использоваться для выбора поставщиков или настройки хранилища. Необходимо проверить целевые локальные и сетевые хранилища, конвергентные сетевые адаптеры и коммутаторы.

3. Коммутация уровней.

Огромный объем трафика проходит между приложениями в центре обработки данных на отдельных физических серверах (с востока на запад). Поскольку тысячи пользователей одновременно получают доступ к контенту / приложениям из Интернета (с севера на юг), вам необходимо переключать уровни, которые можно масштабировать. Тестирование должно проверить масштабируемость и отказоустойчивость сетевых доменов.

4. Переход на протокол IPv6.

Поскольку операторы центров обработки данных сталкиваются с проблемой поддержки IPv6 в центре обработки данных, необходимо выработать оптимальные решения.

5. Безопасность.

Приложения виртуальной безопасности - это виртуальные реализации функций безопасности, распределенные между компонентами облачных

приложений. Они служат для защиты каждого компонента от другого трафика в общих сетях и других виртуальных машин на виртуализированных серверах.

Чтобы проверить эффективность общих мер безопасности, используется тестовая система для отправки набора протоколов приложений между виртуальными машинами. При этом следует убедиться, что трафик был заблокирован или разрешен на основе настроенных политик безопасности.

Устройства защиты периметра, такие как брандмауэры, системы предотвращения вторжений, унифицированные системы управления угрозами и шлюзы VPN, могут быть протестированы с использованием сочетания реального трафика приложений и вредоносных программ для определения эффективности, точности и производительности безопасности. Пиковые емкости и скорости соединения / транзакции / туннеля (IPsec) можно оценить для выбора поставщика и настройки сети.

6. Сквозное качество обслуживания (End-to-end QoS).

Любое публичное или частное облако требует соблюдения качества обслуживания от порта входа глобальной сети на всем пути до отдельных виртуальных машин. Многоуровневая архитектура обеспечивает качество обслуживания для клиентов и типов данных. Тестирование и оценка являются важными компонентами оптимизации всего центра обработки данных для конкретных приложений и клиентов, обеспечивая наилучшее общее качество обслуживания.

7. Доставка приложений.

Постоянно растущее использование приложений, насыщенных мультимедийными данными, приводит к тому, что все большее число пользователей чаще сталкиваются с низким качеством обслуживания.

Облачный дата-центр имеет наилучшую возможность решить данную проблемы благодаря эластичности полосы пропускания.

Чтобы обеспечить качественную доставку приложений, необходимо протестировать архитектуру вычислительной платформы и приложений, которые могут охватывать множество физических и виртуализированных

серверов, используя огромный объем трафика между центрами обработки данных, а также огромный объем встречных видео-сессий.

Проблематике облачного тестирования ПО посвящены труды таких российских и зарубежных ученых и специалистов, как Б.А. Бурняшов, А. Крупин, Chan W.K., R. Malhotra, Xiaoying Bai, специалисты компаний 1С, Google, Microsoft и др.

Вместе с тем, необходимо констатировать недостаточность работ, посвященных разработке методического обеспечения процесса облачного тестирования ПО.

### 1.2 Методы облачного тестирования программного обеспечения

В облачном тестировании применяются классические методы тестирования [11].

1. Метод черного ящика, также известный как поведенческое тестирование является методом тестирования ПО, в котором внутренняя структура, дизайн и реализация тестируемого ПО не известны тестировщику.

Эти тесты могут быть как функциональными (предпочтительно), так не нефункциональными [13].

Методика тестирования по методу черного ящика представляет собой описание процедуры получения и/или выбора тест-кейсов на основе анализа спецификации, функционального или нефункционального описания компонента или системы без ссылки на его внутреннюю структуру.

2. Тестирование по методу белого ящика (также известное как прозрачное тестирование, тестирование на основе кода или структурное тестирование) - это метод тестирования ПО, в котором внутренняя структура, дизайн и реализация тестируемого элемента известны тестировщику.

Тестирование белого ящика - это тестирование, выходящее за рамки пользовательского интерфейса и в мельчайших подробностях системы.

3. Тестирование по методу серого ящика - это метод тестирования ПО, который представляет собой комбинацию методов черного и белого ящиков.

Если в методе черного ящика внутренняя структура тестируемого элемента неизвестна тестировщику, в методе белого ящика полностью известна, то в методе серого ящика внутренняя структура ПО известна тестировщику частично.

Эти знания включают в себя доступ к внутренним структурам данных и алгоритмам для целей разработки тестовых случаев, хотя тестирование проводится на уровне пользователя или черного ящика.

Выбор того или иного метода тестирования определяется особенностями используемой методики облачного тестирования.

### 1.3 Обзор видов облачного тестирования

Выделяются следующие группы видов облачных тестирований:

1. Тестирование архитектуры облачных вычислений.

Архитектура облачных вычислений, как и любое другое приложение или программное обеспечение, состоит из двух основных разделов: Front-End и Back-End.

Front end - это клиент или любое приложение, использующее облачные сервисы.

Back-end - это сеть клиентских машин с серверами, имеющими компьютерную программу и систему хранения данных.

Облако имеет централизованный сервер для администрирования системных клиентов, запросов и т. д.

После разработки пользовательских сценариев, разработки и выполнения теста.

После завершения теста поставщик облачных услуг предоставляет результаты и аналитику корпоративным ИТ-специалистам через информационные панели в реальном времени для полного анализа работы их приложений в периоды пиковых нагрузок.

2. Основные виды облачного тестирования [34] (рисунок 1.1).


Рисунок 1.1 – Основные виды облачного тестирования

Рассмотрим особенности основных видов облачного тестирования.

### 1.3.1 Функциональное облачное тестирование программного обеспечения

Функциональное тестирование ПО проверяет все функции программного обеспечения и его взаимодействие с оборудованием. Для проведения функциональных испытаний тестировщики могут использовать такие инструменты, как Rapise, Sauce Labs и TimeShiftX.

Эти облачные инструменты тестирования программного обеспечения используют следующие методы:

− тестирование системы. Этот метод тестирования оценивает соответствие системы функциональным и системным требованиям.

− приемочные испытания. Это доказывает, что приложение удовлетворяет определенные потребности пользователей.

− интеграционное тестирование. Это тестирование гарантирует, что приложение совместимо с различными платформами и хорошо работает при переходе от одной облачной инфраструктуры к другой.

Функциональное веб-тестирование может выполняться как вручную тестировщиком, так и автоматически с использованием программного обеспечения.

Инструменты функционально тестирования программного обеспечения, которые используются для тестирования обычных приложений, должны быть переоценены применительно к приложению тестирования, размещенному в облаке. Это важно, поскольку необходимы инструменты, позволяющие инженерам-тестировщикам анализировать сеть, рабочий стол и последствия изменений в облаке.

### 1.3.2 Нефункциональное облачное тестирование программного обеспечения

Нефункциональное тестирование также известное как тестирование производительности, так как оно позволяет проверять нефункциональные аспекты программного обеспечения, такие как его производительность, удобство использования и надежность. Для проведения этого типа тестирования можно использовать облачные инструменты, такие как CloudTest, AppPerfect, CloudTestGo и AppLoader.

Эти инструменты тестирования предлагают следующие типы нефункционального тестирования:

− тестирование бизнес-требований. Этот метод тестирования проверяет, насколько точно приложение соответствует указанным бизнес-требованиям. Этот метод также включает тестирование облачной доступности, которое гарантирует отсутствие времени простоя приложения.

− тестирование безопасности. Этот тип тестирования необходим для обеспечения безопасного хранения и передачи данных. Механизмы безопасности приложений тестируются в соответствии с тремя критериями: эффективность, точность и производительность. Наиболее популярными инструментами для тестирования безопасности в облаке являются Nmap, Nessus и Wireshark.

− масштабируемость и тестирование производительности. Хотя облачные решения должны быть масштабируемыми по требованию, этот тип тестирования гарантирует, что приложение будет работать правильно с различным количеством пользователей. Во время нагрузочных испытаний тестеры измеряют время отклика программного обеспечения, в то время как система подвергается увеличению нагрузки. Также необходимо проверить, как приложение будет работать при чрезмерном стрессе, поэтому также необходимо выполнить стресс-тестирование.

Если необходимо измерить задержку ответа приложения после его развертывания в облаке, можно провести тестирование латентности (тестирование на задержку).

### 1.3.3 Тестирование работоспособности

Тестирование работоспособности необходимо для проверки того, действительно ли пользователи действительно получают прикладные услуги по требованию.

Чтобы проверить это, команда тестирования может проводить следующие типы тестирования:

− тестирование на совместимость. Это тестирование оценивает совместимость приложения с различными средами и платформами.

− аварийное восстановление. Тестирование аварийного восстановления позволяет оценить время аварийного восстановления и убедиться, что приложение снова становится доступным пользователям с минимальной потерей данных.

− многоуровневое тестирование. Это тестирование проверяет, может ли приложение обеспечить достаточный уровень безопасности и контроля доступа, когда несколько пользователей обращаются к нему в облаке.

Вышеперечисленные тесты облачных приложений могут иметь разные цели и результаты тестирования.

Применение того или иного вида тестирования определяется особенностями конкретного облачного приложения и/или используемой методикой.

## 1.4 Обзор и анализ существующих методик облачного тестирования программного обеспечения

Рассмотрим известные методики облачного тестирования ПО.

### 1.4.1 Пирамида автоматизированного тестирования

Автоматизированное тестирование предложено М. Коном, который представлял процесс автоматизации системы и проверки ПО в форме пирамиды.

Структура пирамиды тестирования изображена на рисунке 1.2 [35].


Рисунок 1.2 – Пирамида автоматизированного тестирования

Пирамида автоматизированного тестирования состоит из следующих основных слоев (снизу-вверх):

1) автоматизированное тестирование модулей;

2) автоматизированное интеграционное тестирование;

3) автоматизированное сквозное (end-to-end) тестирование.

Как следует из рисунка, сквозное тестирование находится на самой вершине пирамиды тестирования, а модульное тестирование образует ее основание.

Верхний компонент пирамиды также включает тесты пользовательского интерфейса.

Элементы этой пирамиды отличаются друг от друга размером, поскольку они содержат различное количество тестовых случаев, необходимых для выполнения того или иного типа тестирования.

Кроме того, на вершине пирамиды есть облако. Именно ручное тестирование представляется в виде облака, поскольку оно не является неотъемлемой и обязательной частью пирамиды.

Представленная пирамида типична для тестирования программного обеспечения. Но есть много его модификаций.

Все зависит от типа тестируемого приложения.

Составные элементы пирамиды, а также их расположение могут быть изменены.

Иногда пирамида может быть изменена полностью.

### 1.4.2 Облачный тест SOASTA

Облачный тест SOASTA развертывается как сервис по требованию, используя облако для создания нагрузки веб-сайтов [31].

Он включает услуги, предоставляемые тестировщиками нагрузки, и платформы Global Cloud Test Platform, которая обеспечивает кросс-облачную инфраструктуру для генерирования нагрузки.

Внутри приложения библиотеки с открытым исходным кодом являются фундаментальной частью предложения, используемого во всем продукте для обеспечения различных функций.

SOASTA предоставляет программное обеспечение как часть сервиса [42].

Как следует из рисунка 1.3, облачный тест развернут с использованием распределенной архитектуры в облаке, дополненной устройством для тестирования за брандмауэром.

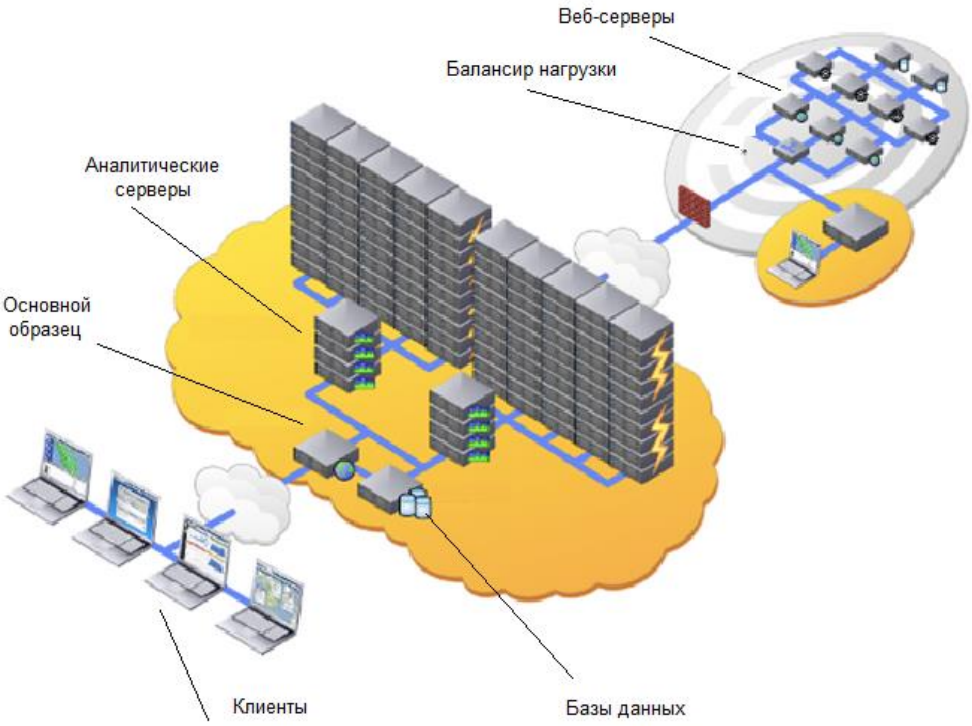



Рисунок 1.3 – Архитектура платформы SOASTA

В то время как клиенты могут использовать приложение SOASTA для создания и выполнения тестов, платформа Global Cloud Test Platform создана для поддержки дополнительных инструментов, включая Apache JMeter, самый популярный инструмент для нагрузочного тестирования с открытым исходным кодом.

Платформа SOASTA уменьшает сложность и время развертывания сценариев JMeter в облаке, что значительно облегчает сообществу JMeter создание, развертывание, выполнение и анализ тестов нагрузки и производительности в масштабах сети.

Скрипты JMeter работают без изменений.

После формирования теста SOASTA необходимо обеспечить управление и подготовку серверов и проведении теста.

Технология обеспечения в реализации SOASTA является одной из ключевых особенностей платформы. По мере появления новых API, в том числе альтернатив с открытым исходным кодом, таких как libCloud, SOASTA будет использовать их для расширения охвата Global Test Cloud.

Блок-схема алгоритма тестирования веб-сайтов и приложений по методике SOASTA представлена на рисунке 1.4.

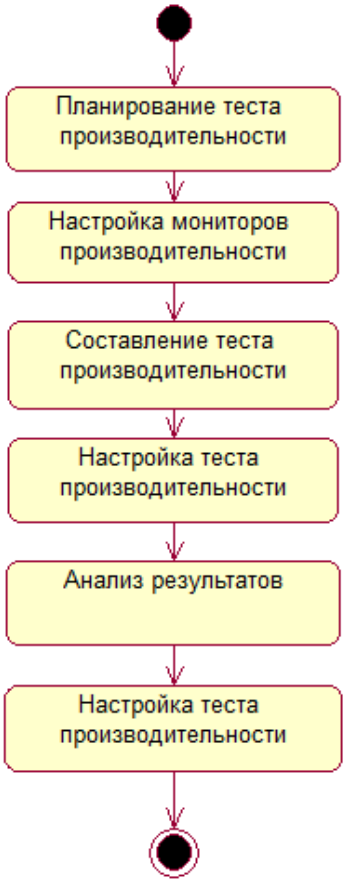



Рисунок 1.4 – Блок-схема алгоритма тестирования веб-сайтов и приложений по методике SOASTA

Другая ключевая возможность - это аналитический механизм в реальном времени, созданный исключительно для тестирования веб-приложений и мобильных приложений, что позволяет командам по обеспечению качества и разработке тестировать и контролировать свои веб-сайты как в типичных, так и в экстремальных условиях трафика. Учитывая огромные объемы данных, генерируемых в веб-тестах, в том числе мониторинг ресурса во время

выполнения теста, облачный высоко масштабируемый движок необходим для предоставления полезной информации в режиме реального времени.

Примером теста, построенного по представленной методике, является SOASTA TouchTest, который обеспечивает полную автоматизацию функционального тестирования мобильных приложений для сенсорных телефонов [12].

### 1.4.3 Методика нагрузочного тестирования Visual Studio

В качестве примера реализации методики облачного тестирования рассмотрим сервис Visual Studio/Team Foundation Services (TFS) на платформе Windows Azure.

TFS можно использовать в качестве бэк-энда для многочисленных интегрированных сред разработки (IDE), но он предназначен для Microsoft Visual Studio и Eclipse на всех платформах.

На рисунке 1.5 представлена структурная схема методики нагрузочного тестирования в Visual Studio [22].

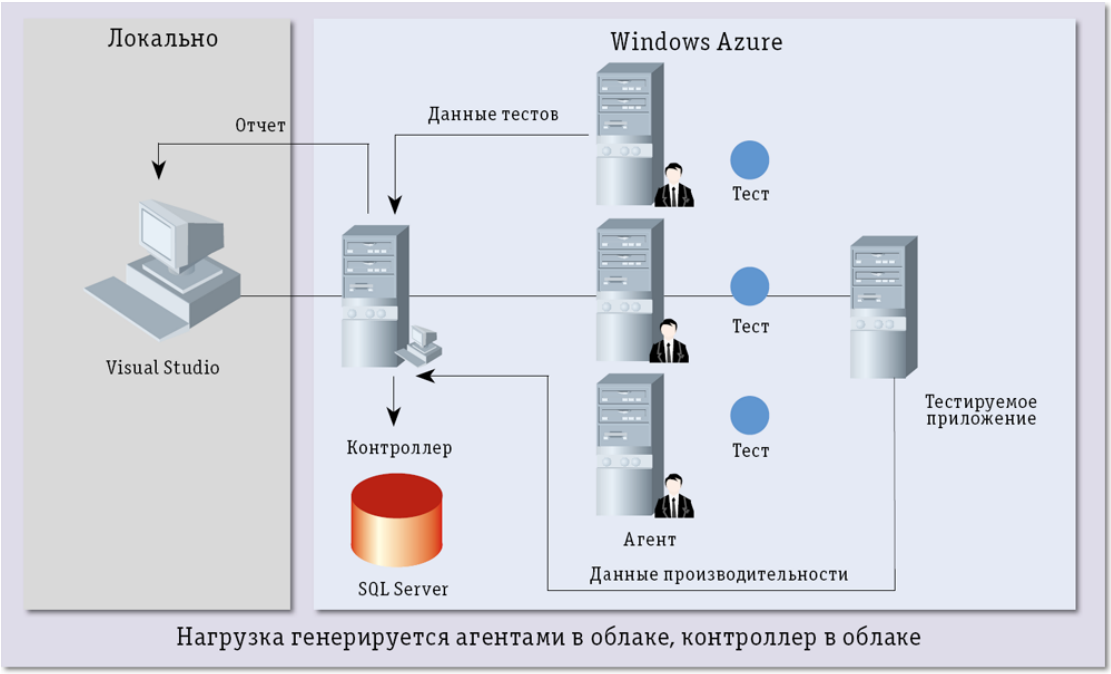



Рисунок 1.5 – Структурная схема методики нагрузочного облачного тестирования в Visual Studio/TFS

Автоматизация процесса тестирования выполняется с помощью компоненты Lab Management.

### 1.4.4 Методика Testing as a Service (TaaS)

Тестирование как услуга (TaaS) - это модель аутсорсинга, в которой действия по тестированию, связанные с некоторыми из бизнес-операций организации, выполняются поставщиком услуг, а не сотрудниками [33, 39].

TaaS может привлекать консультантов для помощи и консультирования сотрудников или просто передавать сторонним поставщикам услуг область тестирования.

TaaS наиболее подходит для специализированных испытаний, которые не требуют глубоких знаний дизайна или системы.

Услуги, которые хорошо подходят для модели TaaS, включают автоматическое регрессионное тестирование, тестирование производительности, тестирование безопасности, тестирование основного программного обеспечения ERP (планирование ресурсов предприятия) и мониторинг / тестирование облачных приложений.

TaaS также иногда называют «тестированием по требованию».

Структурная схема методики TaaS представлена на рисунке 1.6 [32].


Рисунок 1.6 - Структурная схема методики TaaS

Блок-схема алгоритм облачного тестирования ПО на основе TaaS представлена на рисунке 1.7.

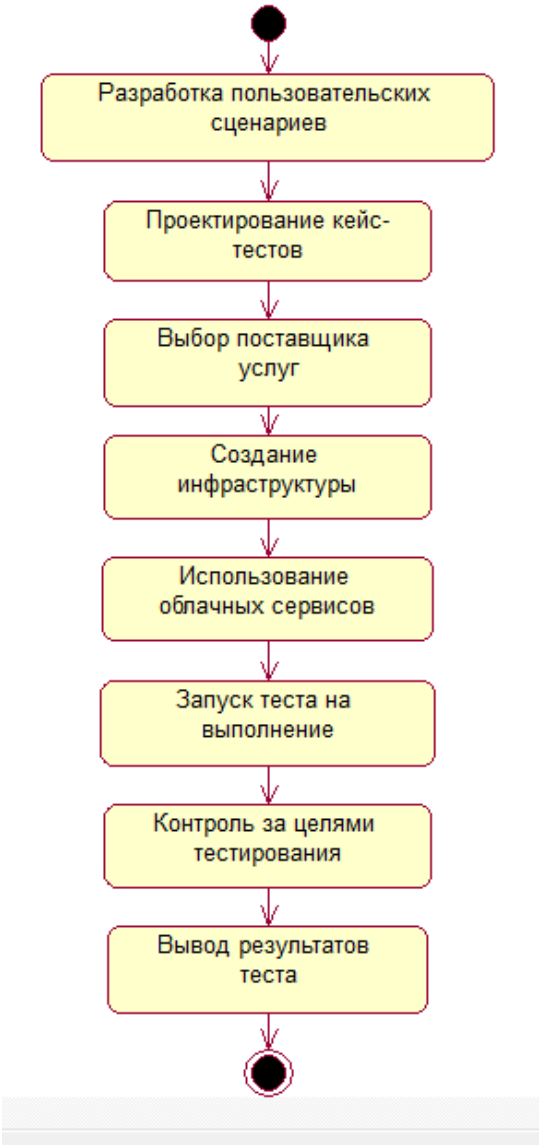



Рисунок 1.7 – Блок-схема алгоритма TaaS

TaaS рекомендуется использовать и проводить тестирование при возникновении подобных проблем, указанных ниже:

− тестирование приложения имеет очень короткий цикл выполнения;

− тестирование приложений требует обширной автоматизации;

− задача тестирования не требует глубоких знаний дизайна системы;

− команда тестирования должна провести специальное тестирование;

− тестирование приложений требует обширных ресурсов для достижения производительности или функционального тестирования.

Виды TaaS:

− функциональное тестирование как услуга. Это поведенческое тестирование, поддерживаемое TaaS Functional Testing. Он включает в себя GUI-тестирование, интеграционное тестирование (SIT), регрессионное тестирование и Acceptance Testing или UAT-тестирование;

− тестирование производительности как услуга. Нагрузочное тестирование и стресс-тестирование являются частью тестирования производительности, которое может проводиться с помощью теста производительности TaaS. TaaS позволяет создавать среду реальных пользователей, которые фактически являются виртуальными пользователями, которые вносят вклад в выполнение нагрузочного и стресс-тестирования для тестируемого приложения;

− тестирование безопасности как услуга. TaaS также может выполнять тестирование безопасности, просто выполняя сканирование уязвимостей в тестируемых программных приложениях и веб-сайтах.

Методика TaaS является лучшим выбором для организаций, которые хотят снизить затраты на тестирование.

### 1.4.5 Анализ известных методик облачного тестирования

Результаты сравнительного анализа методик облачного тестирования представлены в таблице 1.1.

Таблица 1.1 – Сравнительный анализ методик облачного тестирования

| Методика | Достоинства | Недостатки |
|---|---|---|
| Пирамида автоматизированного тестирования | − оперативность;<br>− экономия времени;<br>− повторное использование;<br>− отсутствие «человеческого фактора»;<br>− автоматическая | − большие затраты на средства автоматизации;<br>− недостаточная гибкость;<br>− возможны ошибки тестирования. |

| Методика | Достоинства | Недостатки |
|---|---|---|
| | отчетность. | |
| SOASTA TouchTest | − возможность быстрого проектирования, исполнения, редактирования и анализа тест кейсов;<br>− возможность тестирования мультитач-жестов в нативных, гибридных и веб-приложениях на iOS и Android;<br>− непрерывное тестирование производительности мобильного приложения<br>− TouchTest лучше всего подходит для автоматизации критических тест кейсов, которые будут многократно использоваться, поскольку каждый девайс должен быть протестирован отдельно. | − малая точность масштабирования;<br>− малая точность синхронизации;<br>− нет возможности управления оборудованием.<br>− отсутствует опция симуляции входящих звонков;<br>− используется только для веб- и мобильных приложений. |

| Методика | Достоинства | Недостатки |
|---|---|---|
| Нагрузочное тестирование MS Visual Studio | − генерирование нагрузки непосредственно в облаке;<br>− поддерживает все типы рабочих процессов тестирования Visual Studio. | − требует дополнительные ресурсы для создания нагрузки;<br>− ограниченная область применения. |
| TaaS | − гибкость выполнения теста;<br>− экономия общей стоимости тестирования;<br>− быстрые результаты тестирования;<br>− целостность данных. | − организация должна отдать на аутсорсинг свою интеллектуальную собственность;<br>− может потребоваться работа с группой поддержки облака, которая может быть внешней по отношению к организации;<br>− организация может стать зависимой от внешнего тестировщика. |

Анализ известных методик облачного тестирования показал, что главным их недостатком является недостаточная универсальность и как следствие, низкая низкая эффективность для некоторых видов ПО.

Пирамида тестирования представляет собой методику облачного автоматизированного тестирования.

Так, технология SOASTA предназначена для тестирования вэб-сайтов.

Сервис TFS рекомендуется прежде всего для Microsoft Azure.

Методика TaaS предназначена для аутсорсингового тестирования.

Кроме того, в описаниях методик нет сведений об особенностях применения в условиях конкретной модели облачных вычислений.

Таким образом, представляет актуальность разработка облачной методики тестирования, обладающей большей универсальностью, чем вышеописанные.

## Выводы к первой главе

Облачное тестирование стало новым подходом к тестированию, благодаря которому среды облачных вычислений используются для применения современных видов тестирования, обеспечивающих высокую эффективность данного процесса при минимизации затрат организаций-пользователей.

В облачном тестировании применяются классические методы тестирования. Выбор того или иного метода тестирования определяется особенностями используемой методики облачного тестирования.

Применение того или иного вида тестирования определяется особенностями конкретного облачного приложения и/или используемой методикой.

Анализ известных методик облачного тестирования показал, что главным их недостатком является недостаточная универсальность. Поэтому представляет актуальность разработка методики облачного тестирования, обладающей большей универсальностью, чем известные.

# Глава 2 СОВРЕМЕННЫЕ ПОДХОДЫ К РАЗРАБОТКЕ МЕТОДИК ОБЛАЧНОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 2.1 Методологии облачного тестирования программного обеспечения

Для понимания особенности методики облачного тестирования, рассмотрим сервисы тестирования, основанные на моделях облачных вычислений: SaaS, PaaS и IaaS [37].

### 2.1.1 Методология SaaS-тестирования

Модель облачных вычислений SaaS (программное обеспечение как услуга) доступна клиентам через интернет и помогает организациям обходить сложности, сопряженные с установкой ПО на компьютеры [17].

Платформа для тестирования SaaS - метод обеспечения качества ПО за счет всевозможных проверок, включая тестирование производительности , безопасности, интеграции данных, масштабируемости, надежности и прочего.

Среди преимуществ методологии тестирования SaaS можно выделить следующие:

- большая надежность, масштабируемость и доступность;
- сокращение расходов на установку и обслуживание ПО;
- быстрое устранение ошибок;
- быстрое размещение ПО;
- невысокая плата за пользование;
- минимизация взаимозависимости системы на нескольких уровнях;
- простота обновления SaaS-системы.

К недостаткам SaaS относят:

- сложность проведения тестирования;
- программные продукты разрабатываются намного быстрее, поэтому на первый план выходит обеспечение качества;
- для работы с компонентами SaaS-приложений требуется соответствующая квалификация ИТ-персонала;

− среда тестирования должна поддерживать автоматические размещение и валидацию приложения.

При SaaS-тестировании используются методы, которые обеспечивают надежную работу приложения, созданного для этой модели.

Приложения, инфраструктура и сеть считаются ключевыми компонентами SaaS-тестирования.

Методология SaaS-тестирования применяется для решения следующих задач:

− тестирование по стратегии белого и черного ящика как часть модульного тестирования;

− функциональное тестирование для тщательной проверки соответствия приложения требованиям;

− интеграционное тестирование проводится для проверки интеграции SaaS-системы с другими;

− исследовательское тестирование для новых тест-кейсов;

− тестирование безопасности сети, различных угроз, интеграции и доступности;

− обеспечение качества соединения SaaS, а также тестирование пользовательского интерфейса, с акцентом на портативность и компактность;

− регрессионное тестирование для любого обновления, релиза или переноса данных в приложении;

− тестирование надежности производится с целью снизить вероятность ошибок;

− проверка безопасности сети;

− тестирование работоспособности и масштабируемости для проверки, как ведет себя приложение при пиковой нагрузке в разных условиях

− проверка совместимости приложения, с доступом на разных браузерах;

− тестирование обновлений при добавлении новых функциональных особенностей или модернизации старых;

− тестирование API для проверки функциональности и безопасности;

− клиентские запросы, платежи и биллинг — составляющие этапы тестирования в реальных условиях.

Сложности, связанные с тестированием SaaS-приложений:

− частые обновления и релизы оставляют мало времени на проверку валидности и безопасности приложений;

− иногда компоненты бэкэнда, связанные с пользовательским интерфейсом приложения, остаются без проверки;

− обеспечение безопасности данных и предотвращение утечек данных;

− важно определить самые доступные зоны и провести пользовательское тестирование с участием как можно большего числа людей из разных мест;

− во время интеграции и переноса SaaS-приложений нередко возникают сложности с сохранением данных тестирования;

− для новых релизов проверяются все аспекты, относящиеся к лицензированию, в том числе количество пользователей и функциональность приложения;

− отсутствие стандартизации приложения.

Средства SaaS-тестирования:

1) PractiTest.

Этот инструмент тестирования разработан для того, чтобы обеспечить клиентов комплексными тестовыми решениями и предоставить рычаги контроля для процесса разработки приложений и тестирования.

Функциональные особенности:

− обеспечение коммуникации на разных уровнях;

− средства управления проектом, общим процессом тестирования;

− в любое время можно узнать статус проекта.

2) qTest.

Этот облачный инструмент для управления тестированием, который упрощает коммуникации.

Функциональные особенности:

− простота взаимодействия с участниками команды;

− предусмотрена функция создания заметок, детализированные отчеты о дефектах;

− доступна пробная версия;

− инструмент позволяет планировать и составлять график проекта, документацию по тест-кейсам, отчетам и результатам;

− есть удобная приборная доска, на которой отображается достигнутый прогресс, запросы и полезные отчеты.

3) QMetry.

Инструмент позволяет связать требования проекта с его тест-кейсами и дефектами.

Функциональные особенности:

− QMetry удобен в условиях быстро меняющихся требований, позволяет использовать старые тест-кейсы;

− результаты и статус тест-кейсов могут быть зафиксированы во время выполнения тест-кейсов;

− страница выполнения доступна для редактуры тест-кейсов в режиме реального времени;

− управление дефектами с помощью ссылки;

− все дефекты, зафиксированные во время прошлых тестов, можно легко обнаружить, так один и тот же дефект не будет записан дважды.

Cisco Web Ex, Google Apps, Consur — известные примеры SaaS-приложений, которые доступны посредством сети и не требуют дополнительной установки.

### 2.1.2 Методология PaaS-тестирования

В модели облачных вычислений PaaS (платформа как услуга) разработчики, по сути, арендуют все, что им нужно для создания приложения, полагаясь на облачного провайдера для средств разработки, инфраструктуры и операционных систем. Это одна из трех сервисных моделей облачных

вычислений [30]. PaaS значительно упрощает разработку веб-приложений; с точки зрения разработчика, все управление бэкэндом происходит за кулисами.

Доступ к PaaS можно получить через любое интернет-соединение, что позволяет создать целое приложение в веб-браузере.

Поскольку среда разработки не размещается локально, разработчики могут работать с приложением из любой точки мира. Это позволяет группам, которые распределены по географическим местам, сотрудничать. Это также означает, что разработчики имеют меньший контроль над средой разработки, хотя это требует гораздо меньших затрат. PaaS предоставляет быстрые, простые и экономичные способы разработки, тестирования и развертывания приложений. С помощью стороннего поставщика мы можем управлять операционными системами, виртуализацией, хранилищем, сетью, серверами и самим PaaS [27].

PaaS предоставляет платформу с инструментами для тестирования, разработки и размещения приложений в одной среде. Поставщики услуг PaaS предлагают в частности облачные решения для приложений .Net и Java [24].

Популярные платформы PaaS: AWS Elastic Beanstalk, Windows Azure, Google App Engine, Apache Stratos и др.

### 2.1.3 Методология IaaS-тестирования

Модель облачных сервисов IaaS, известная как «инфраструктура как услуга», представляет собой модель самообслуживания для доступа, мониторинга и управления инфраструктурами удаленных центров обработки данных, такими как вычислительные (виртуальные или физические), службы хранения, сети и сети (например, брандмауэры).

Вместо того чтобы приобретать аппаратное обеспечение напрямую, пользователи могут приобретать услуги IaaS на основе аутсорсинга.

По сравнению с SaaS и PaaS пользователи IaaS отвечают за управление приложениями, данными, средой выполнения, промежуточным ПО и операционными системами. Поставщики по-прежнему управляют виртуализацией, серверами, жесткими дисками, хранилищами и сетями.

Многие провайдеры IaaS теперь предлагают базы данных, очереди сообщений и другие услуги выше уровня виртуализации. Иными словами, пользователи IaaS получают инфраструктуру, поверх которой они могут устанавливать любую необходимую платформу.

Пользователи несут ответственность за их обновление в случае выпуска новых версий. Вычислительные и сетевые возможности делают IaaS идеальным местом для запуска и управления циклами тестирования и разработки.

Популярные платформы IaaS: Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE) и др.

### 2.1.4 Сравнительный анализ методологий облачного тестирования

Результаты сравнительного анализа характеристик методологий облачного тестирования, построенных на вышеописанных моделях, представлены в таблице 2.1.

Таблица 2.1 - Сравнительный анализ характеристик методологий облачного тестирования

| Модель | Характеристики среды | Область применения |
|---|---|---|
| SaaS | − централизованное управление;<br>− размещено на удаленном сервере;<br>− доступно через интернет;<br>− пользователи не несут ответственности за обновления оборудования или ПО. | − стартапы или небольшие компании, которым нужно быстро запустить электронную коммерцию при минимальных затратах;<br>− краткосрочные проекты, которые требуют быстрого, простого и доступного сотрудничества;<br>− приложения, которые используются нечасто;<br>− приложения, которым нужен как веб, так и |

| Модель | Характеристики среды | Область применения |
|---|---|---|
| | | мобильный доступ. |
| PaaS | − основана на технологии виртуализации;<br>− предоставляет различные услуги, помогающие в разработке, тестировании и развертывании приложений;<br>− обеспечивает многопользовательский режим через одно и то же приложение для разработки и тестирования;<br>− интегрирует веб-сервисы и базы данных | −оптимизация рабочих процессов, когда несколько разработчиков работают над одним проектом разработки;<br>−создание заказных приложений;<br>− быстрая разработка или развертывании приложения. |
| IaaS | −ресурсы доступны как услуга;<br>−стоимость зависит от потребления;<br>−сервисы хорошо масштабируются;<br>−многопользовательский режим;<br>−организация сохраняет полный контроль над ИТ-инфраструктурой;<br>−динамичность и гибкость. | −компании, которые хотят сохранить контроль над своей ИТ-инфраструктурой при минимальных затратах на ИТ-ресурсы;<br>−быстрорастущие компании. |

Таким образом, выбор методологии тестирования определяется особенностями модели облачных вычислений, используемой конкретным вендором ПО.

Как показывает практика, наиболее востребованными для решения задач облачного тестирования являются методологии SaaS и IaaS.

## 2.2 Принципы построения методики облачного тестирования программного обеспечения

Универсальная методика облачного тестирования (рисунок 2.1) состоит из следующих этапов [15].

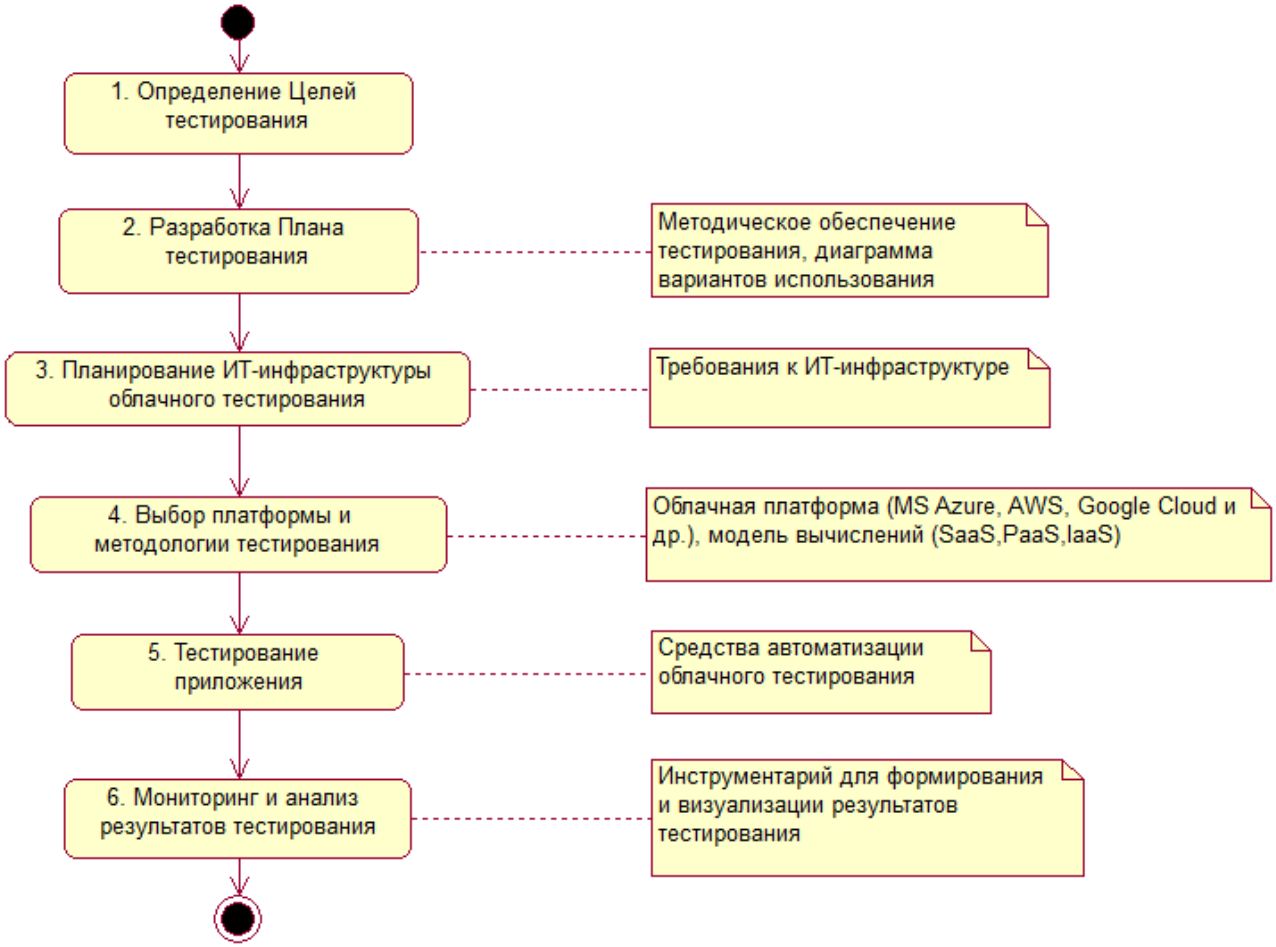



Рисунок 2.1 – Этапы методики облачного тестирования ПО

1. Формулировка цели тестирования.

Облачное тестирование имеет как преимущества, так и недостатки, поэтому извлечь из него выгоду можно только тогда, когда у компании-вендора

ПО есть четкое понимание потребностей собственного бизнеса. Тестирование в облаке требует более тесного сотрудничества между разработчиками и тестировщиками для проведения всех необходимых тестов на протяжении всего жизненного цикла разработки программного обеспечения.

2. Разработка плана тестирования.

План тестирования представляет собой описание области и деятельности облачного тестирования.

Прежде чем переместить свой проект в облако, необходимо определить, какие тесты нужно выполнить, сколько времени они будут принимать и какие риски при этом возможны. Эта стратегия тестирования позволяет лучше оценить бюджет компании и избежать непредвиденных затрат.

Структурная схема типового плана облачного тестирования ПО представлена на рисунке 2.2 [25, 40].


Рисунок 2.2 – Структурная схема типового плана облачного тестирования ПО

Методика тестирования должна развиваться во всех этих сценариях и должна учитывать виртуализированную инфраструктуру, сеть, бизнес-логику приложений, данные и взаимодействие с конечным пользователем.

Тестирование облачных приложений требует тестирования бизнес-процессов, механизмов исключений, имитации сценариев отказов и сценариев аварийного восстановления.

3.Планирование инфраструктуры.

При создании стратегии тестирования следует также подумать о требованиях к инфраструктуре, необходимых для создания тестовой среды. Необходимо убедиться, что облачные службы предоставляют необходимые средства автоматизации тестирования, программное обеспечение, аппаратное обеспечение и пропускную способность. Также важно определить, как долго понадобится тестовая среда в этой конфигурации, и потребуются ли какие-либо изменения в ней.

4. Выбор платформы и методологии облачного тестирования.

При выборе облачной платформы необходимо учитывать особенности разработки конкретного ПО и область его использования. Выбор методологии тестирования определяется используемой моделью облачных вычислений.

5. Тестирования приложения.

На данном этапе определяются средства автоматизации тестирования.

6. Мониторинг и анализ результатов тестирования.

Хотя тестирование в облаке позволяет обеспечить постоянную доступность услуг, лучше контролировать результаты тестирования в режиме реального времени. Анализируя результаты в реальном времени, тестировщики могут быстро реагировать на проблемы, связанные с эффективностью или производительностью облачного тестирования.

Рассмотрим отдельные элементы данной методики.

## 2.3 Методика облачного тестирования программного обеспечения

### 2.3.1 Методика функционального тестирования

Функциональное облачное тестирование выполняется как для удаленных, так и для локальных приложений.

Функциональное облачное тестирование - это тестирование всех функций и свойств системы, включая аппаратное и программное обеспечение.

Оно проводится на полной, интегрированной программной платформе, чтобы проверить ее соответствие требованиям.

При функциональном облачном тестировании процесс проверки выполняется в соответствии со спецификациями системы или требованиями в облаке вместо локального тестирования ПО.

Функциональное тестирование не является достаточно полным, чтобы определить все комбинации сайта и его работоспособность в стрессовых условиях.

К функциональному тестированию относятся:

– системное тестирование.

Методы системного тестирования используются для проверки поведения системы в пределах ее собственных границ. Крайне важно доказать, что система функционирует так, как она была спроектирована, когда компоненты системы работают вместе, входы и выходы соответствуют ожидаемым значениям, и в результате общая система представляет собой облачную систему высокого качества;

– интеграционное тестирование.

Интеграционное облачное тестирование позволяет компании проверить, что облачное решение будет работать в рамках текущей инфраструктуры и сред. Это в конечном итоге доказывает, что внедрение облачного решения не оказывает вредного воздействия на любые существующие системы.

Наконец, бизнес-требования должны быть проверены и подтверждены, чтобы доказать, что конечный результат облачного решения будет соответствовать документированным потребностям бизнеса;

– пользовательское приемочное тестирование.

В ходе приемочного тестирования проверяется, что поставляемое облачное решение соответствует бизнес-требованиям, что является основанием для пользователя для приемки разработанного облачного решения.

Пользовательское приемочное тестирование проводится как в режиме онлайн, так и в офф-лайн. Вместе с тем локальное тестирование позволяет немедленно проконтролировать и выполнить мониторинг хода тестирования.

### 2.3.2 Методика нефункционального тестирования

Нефункциональное тестирование проводится, чтобы убедиться в том, что веб-приложение соответствует заданным требованиям к производительности.

Нефункциональное тестирование также известно, как тестирование производительности.

В облаке область масштабируемости приложений намного шире, чем в традиционных методиках тестирования производительности ПО.

К нефункциональному тестированию относятся:

– тестирование бизнес-требований.

Перед переводом своего бизнеса на решение для облачных вычислений организации и партнеры должны тщательно проанализировать и документировать свои бизнес-требования четко, точно и однозначно.

Бизнес-требования являются основой для создания решения облачных вычислений. Эти бизнес-требования могут быть достигнуты с помощью обзоров, периодических встреч с клиентами и семинаров. Позже это, в свою очередь, позволяет создать идеальную систему, способную удовлетворить требования бизнеса.

Данный вид тестирования включает в себя тестирование доступности: облачные сервисы должны быть доступны постоянно. Должна также быть уверенность в том, что нет резких простоев, приводящих к неблагоприятному воздействию на бизнес клиента;

– тестирование безопасности.

Тестирование безопасности является неотъемлемой частью тестирования приложений из-за увеличения проблем с безопасностью в бизнесе. Это может обеспечить уверенность в том, что важные для бизнеса данные хранятся и транспортируются безопасно.

Определение методов получения доступа к системе с использованием общих инструментов и методов, используемых хакерами, вполне может гарантировать безопасность облачных решений.

– тестирование масштабируемости и производительности.

Облачная масштабируемость является еще одной серьезной проблемой, требующей длительного тестирования.

Решения облачных вычислений всегда претендуют на масштабируемость по требованию. Нагрузочное или стресс-тестирование можно использовать, чтобы доказать, что разработанное облачное решение можно масштабировать в соответствии с требованиями с помощью программных инструментов.

Следовательно, облачное решение может быть точно оценено, а его мощность проверена. Методики тестирования производительности облака позволяют измерять производительность облачных систем достаточно точно.

Тестирование производительности с использованием методов нагрузочного тестирования позволяет получить точное представление о возможностях решения в облаке.

Производительность обычно связана с возможностями приложения в облачной инфраструктуре.

Выявление порогов, узких мест и ограничений является частью тестирования производительности.

Для этого необходимо тестировать производительность при определенной рабочей нагрузке и варьировать характер трафика по требованию.

Данный вид тестирования включает в себя:

– облачное нагрузочное и стрессовое тестирование.

Стабильность приложения является основным фактором, поскольку ожидается увеличение количества пользователей.

Нагрузочное тестирование приложения предполагает создание большого пользовательского трафика и измерение его отклика.

Также необходимо настроить производительность любого приложения в соответствии с определенными стандартами.

Необходимо измерять время отклика и выявлять проблемы, связанные с конкретными действиями, в то время как система подвергается возрастающей нагрузке из разных мест и многопользовательским операциям.

Следует обязательно выявлять проблемы, поскольку система тестируется до предела максимальной ожидаемой емкости или часто за пределами ожидаемого использования.

Стресс-тест используется для определения способности приложения поддерживать определенный уровень эффективности за пределами критической точки или максимальной ожидаемой емкости или сверх ожидаемого использования.

Для любого приложения важно работать даже при чрезмерном стрессе и поддерживать стабильность.

Стресс-тестирование гарантирует это путем создания пиковых нагрузок с использованием тренажеров. Но стоимость создания таких сценариев огромна.

– латентное тестирование.

Облачное тестирование используется для измерения задержки между действием и соответствующим ответом для любого приложения после его развертывания в облаке.

### 2.3.3 Методика тестирования работоспособности

Тестирование работоспособности проводится для того, чтобы облачная среда могла предоставлять свои услуги пользователям по запросу.

В этой категории проверяется совместимость, функциональная совместимость и многопользовательская способность среды облачных вычислений.

К данному виду тестированию относятся:

– тестирование на совместимость.

В облачной среде различные программы и операционные системы используются и создаются по требованию, что делает обязательным тестирование на совместимость.

Облачное приложение должно работать в разных средах и на разных облачных платформах.

Следовательно, очень легко осуществить миграцию облачных приложений и платформ из одной инфраструктуры в другую.

– тестирование аварийного восстановления.

Поставщик облачных услуг всегда предпочитает, чтобы его облачные услуги были постоянно доступны конечным пользователям, но на самом деле это невозможно.

Может быть некоторая вероятность сбоя, поэтому время аварийного восстановления должно быть низким.

Облачная проверка призвана гарантировать, что сервис вернулся онлайн с минимальным неблагоприятным воздействием на бизнес.

– тестирование ПО, используемое несколькими арендаторами.

Данное тестирование гарантирует, что несколько клиентов и организаций, использующих услуги по требованию, активируются в заданное время.

Облачный сервис должен быть настраиваемым для каждого клиента и обеспечивать уровень данных и безопасности, чтобы избежать проблем, связанных с доступом.

## 2.4 Обзор и анализ платформ облачного тестирования

Облачные вычисления усложняют процесс разработки ввиду огромного количества компонентов, которые должны взаимодействовать между собой.

Поэтому чтобы гарантировать качество и надежность, важно иметь четкое представление о платформах и услугах облачных вычислений.

В то же время эластичность и автоматизированное выделение ресурсов, которые обеспечивают облачные технологии, радикально улучшают жизненный цикл разработки ПО [5].

Возможность выполнения по требованию тестов на массовую нагрузку или на проникновение позволяет группам программистов легко и дешево разрабатывать и тестировать масштабируемые приложения.

Другими словами, существует тесная взаимосвязь между средствами разработки и развертывания с одной стороны и возможностями для тестирования с другой.

Поэтому для разработки качественных методик облачного тестирования необходимо учитывать особенности облачной платформы, которая будет использована в качестве среды облачного тестирования.

Рассмотрим характеристики популярных промышленных платформ облачного тестирования.

### 2.4.1 Облачная платформа Google Cloud

Google Cloud Platform (GCP), предлагаемая Google, представляет собой набор служб облачных вычислений, которые работают на той же инфраструктуре, которую Google использует для своих конечных пользователей, таких как Google Search и YouTube [28].

GCP написан на языках Java, C++, Python, Go, Ruby.

Наряду с набором инструментов управления GCP предоставляет ряд модульных облачных сервисов, включая вычисления, хранение данных, анализ данных и машинное обучение.

Регистрация требует кредитной карты или реквизиты банковского счета.

Облачная платформа Google предоставляет сервисы IaaS, PaaS и безсерверные вычислительные среды.

GCP является частью Google Cloud, которая включает в себя инфраструктуру общедоступного облака GCP, а также G Suite, корпоративные версии Android и Chrome OS, а также интерфейсы прикладного

программирования (API) для машинного обучения и сервисов корпоративного мапинга.

Доступ к сервисам Google Cloud Platform могут получить разработчики программного обеспечения, администраторы облака и другие ИТ-специалисты предприятия через общедоступный Интернет или через выделенное сетевое соединение.

Google Cloud Platform предлагает сервисы для вычислений, хранения, сетей, больших данных, машинного обучения и Интернета вещей (IoT), а также инструменты управления облаком, безопасности и разработки (рисунок 2.3).

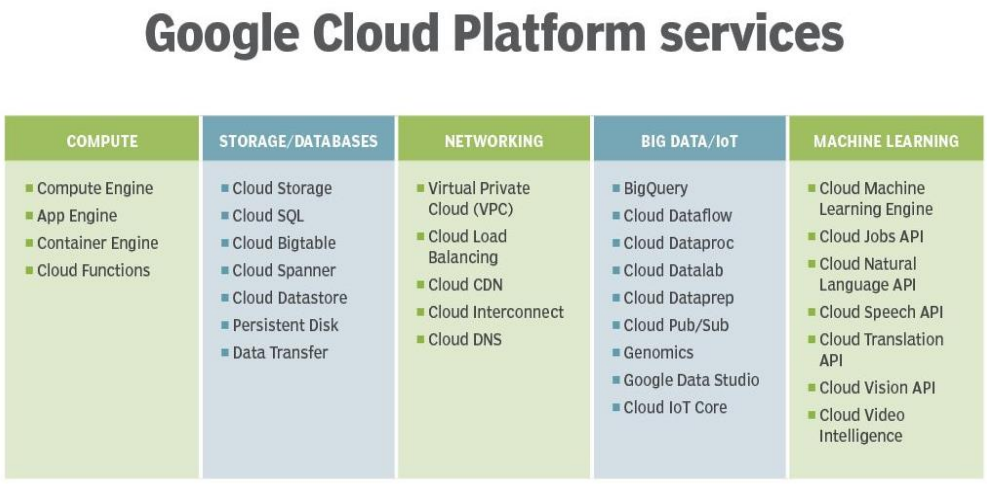



Рисунок 2.3 – Сервисы платформы Google Cloud

Основные продукты облачных вычислений в Google Cloud Platform включают в себя:

− Google Compute Engine, представляющий собой инфраструктуру как услугу (IaaS), которая предоставляет пользователям экземпляры виртуальных машин для размещения рабочей нагрузки;

− Google App Engine, представляющий собой платформу как услугу (PaaS), которая предоставляет разработчикам программного обеспечения доступ к масштабируемому хостингу Google. Разработчики также могут использовать комплект разработчика программного обеспечения (SDK) для разработки программных продуктов, работающих на App Engine;

− Google Cloud Storage - платформа облачного хранения, предназначенная для хранения больших неструктурированных наборов данных. Google также предлагает варианты хранения базы данных, в том числе Cloud Datastore для нереляционного хранилища NoSQL, Cloud SQL для полностью реляционного хранилища MySQL и нативную базу данных Cloud Bigtable от Google;

Google Container Engine - система управления и оркестровки контейнеров Docker, которая работает в общедоступном облаке Google. Google Container Engine основан на движке оркестровки контейнеров Google Kubernetes.

Google Cloud Platform предлагает услуги по разработке приложений и интеграции.

Например, Google Cloud Pub / Sub - это управляемая служба обмена сообщениями в режиме реального времени, которая позволяет обмениваться сообщениями между приложениями. Кроме того, конечные точки Google Cloud позволяют разработчикам создавать сервисы на основе API RESTful, а затем делать эти сервисы доступными для клиентов Apple iOS, Android и JavaScript. Другие предложения включают DNS-серверы Anycast, прямые сетевые соединения, балансировку нагрузки, услуги мониторинга и ведения журналов.

### 2.4.2 Облачная платформа Amazon Web Services

Amazon Web Services (AWS) – облачная платформа с широкими возможностями, которая предоставляет 165 полнофункциональных сервисов для центров обработки данных по всей планете.

Миллионы клиентов, в том числе стартапы, ставшие лидерами по скорости роста, крупнейшие корпорации и передовые правительственные учреждения, доверяют AWS в вопросах размещения инфраструктуры, повышения гибкости и снижения затрат (рисунок 2.4) [23].

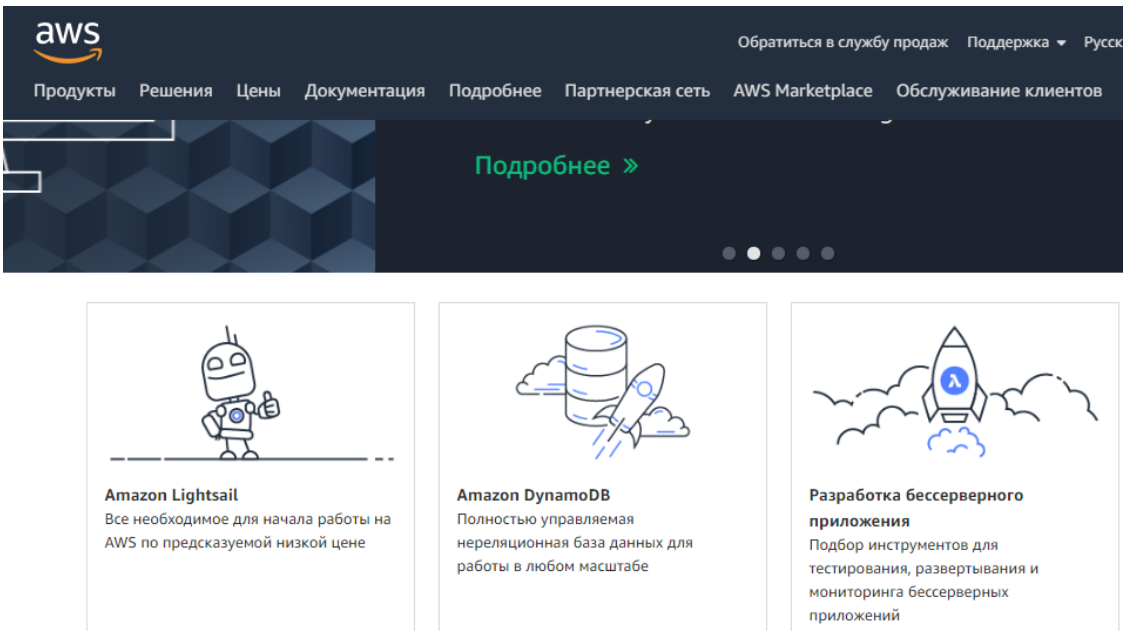



Рисунок 2.4 – Сервисы платформы Amazon Web Services

Достоинства платформы:

1. Широкие функциональные возможности: AWS предлагает намного больше сервисов и возможностей в их рамках, чем любой другой поставщик облачных решений. Облачная платформа AWS предлагает более 165 полнофункциональных сервисов, в том числе более 40 уникальных.

2. Широкая сеть клиентов: на платформе AWS создана крупнейшая и самая динамичная экосистема сервисов с миллионами активных клиентов и десятками тысяч партнеров по всему миру. Клиенты с разным масштабом деятельности, в том числе стартапы, корпорации и организации государственного сектора, практически в любой отрасли используют AWS в самых разнообразных целях.

3. Безопасность: AWS - гибкая и защищенная среда для облачных вычислений. Базовая инфраструктура спроектирована так, чтобы удовлетворить требования к безопасности международных банков, учреждений в сфере обороны и других организаций с высокими требованиями к безопасности.

4. Надежность.

AWS – это тщательно продуманные решения, надежность, безопасность и производительность. Более 12 лет AWS поставляет облачные сервисы миллионам клиентов по всему миру для использования в разнообразных целях,

а также предлагает самые лучшие на рынке решения для работы в любом масштабе.

В совокупности эти веб-сервисы облачных вычислений предоставляют набор примитивной абстрактной технической инфраструктуры и строительных блоков, и инструментов распределенных вычислений.

Одним из таких сервисов является Amazon Elastic Compute Cloud, который позволяет пользователям иметь в своем распоряжении виртуальный кластер компьютеров, доступный постоянно через Интернет. Версия виртуальных компьютеров AWS эмулирует большинство атрибутов реального компьютера, включая аппаратные центральные процессоры (CPU) и графические процессоры (GPU) для обработки, локальную / оперативную память, жесткий диск / твердотельный накопитель; выбор операционных систем; сетей; и предварительно загруженное прикладное программное обеспечение, такое как веб-серверы, базы данных, управление взаимоотношениями с клиентами (CRM) и т. д.

### 2.4.3 Облачная платформа Microsoft Azure

Microsoft Azure - это служба облачных вычислений, созданная Microsoft для создания, тестирования, развертывания и управления приложениями и службами через центры обработки данных, управляемые Microsoft (рисунок 2.5).

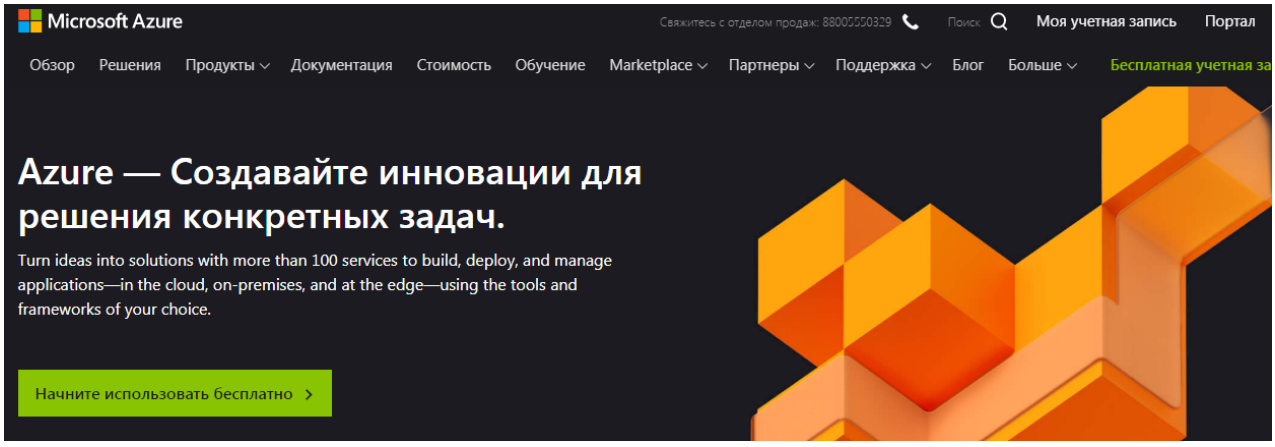



Рисунок 2.5 – Экран главной страницы Microsoft Azure

Azure предоставляет сервисы SaaS, PaaS и IaaS и поддерживает множество различных языков программирования, инструментов и сред, включая как программное обеспечение и системы Microsoft, так и сторонних производителей [29].

Azure использует специализированную операционную систему, называемую MS Azure, для запуска своего «фабричного слоя».

Кластер, размещенный в центрах обработки данных Microsoft, который управляет вычислительными ресурсами и ресурсами хранения компьютеров и предоставляет ресурсы (или их подмножество) приложениям, работающим поверх Azure.

Azure описана как «облачный слой» поверх ряда систем Windows Server, которые используют Windows Server 2008 и настроенную версию Hyper-V, известную как гипервизор Microsoft Azure, для обеспечения виртуализации служб.

Масштабирование и надежность контролируются Microsoft Azure Fabric Controller, который обеспечивает отказ служб и среды при сбое одного или нескольких серверов в центре обработки данных Microsoft, а также обеспечивает управление веб-приложением пользователя, таким как выделение памяти. и балансировка нагрузки.

Azure предоставляет API, основанный на REST, HTTP и XML, который позволяет разработчику взаимодействовать со службами, предоставляемыми Microsoft Azure. Microsoft также предоставляет клиентскую библиотеку управляемых классов, которая включает в себя функции взаимодействия со службами. Он также интегрируется с Microsoft Visual Studio, Git и Eclipse.

В дополнение к взаимодействию со службами через API пользователи могут управлять службами Azure с помощью веб-портала Azure, который достиг общей доступности в декабре 2015 года.

Портал позволяет пользователям просматривать активные ресурсы, изменять настройки, запускать новые ресурсы и просматривать базовый мониторинг. данные с активных виртуальных машин и сервисов.

Azure - это постоянно расширяющийся набор служб облачных вычислений, который помогает вашей организации решать бизнес-задачи.

По мнению специалистов Microsoft, платформа Azure предоставляет разработчикам свободу создания, тестирования и развертывания приложений, а также управления ими в обширной глобальной сети с использованием различных инструментов и платформ.

Необходимо отметить, что выбор конкретной платформы для применения в качестве среды облачного тестирования зависит от многих критериев: модели облачных вычислений, аналитики, хранения, сети, ценообразования [21].

Важно также учитывать, какую среду разработки ПО использует компания-вендор.

Так, для разработчиков, использующих продукты корпорации Microsoft более предпочтительной представляется платформа MS Azure.

Немаловажным фактором являлся и опыт работы облачного провайдера на рынке по ИТ-проектам, которыми занимается вендор ПО.

## Выводы ко второй главе

Как показал анализ, выбор методологии облачного тестирования определяется особенностями модели облачных вычислений, которую использует конкретный вендор ПО. Вместе с тем как показывает практика, наиболее востребованными для решения задач облачного тестирования являются методологии SaaS и IaaS.

Типовая методика облачного тестирования состоит из следующих этапов: формулировка цели тестирования, разработка плана тестирования, планирование инфраструктуры, выбор платформы и поставщика облачного сервиса, мониторинг и анализ результатов тестирования.

Типовой план облачного тестирования ПО включает в себя план функционального тестирования, план нефункционального тестирования и план тестирования работоспособности.

Для разработки качественных методик облачного тестирования необходимо учитывать особенности облачной платформы, которая будет использована в качестве среды облачного тестирования.

# Глава 3 РАЗРАБОТКА МЕТОДИК ОБЛАЧНОГО ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЙ 1С

Рассмотрим процесс разработки методики на примере облачного тестирования приложений 1С8 и 1С:Битрикс.

## 3.1 Методика облачного тестирования приложений 1С8

На основе анализа известных подходов к облачному тестированию ПО, с учетом их функциональных и архитектурных особенностей разработана методика облачного тестирования приложений, реализованных на базе технологической платформы «1С: Предприятие 8» [7].

Диаграмма деятельности методики облачного тестирования приложения 1С8 представлена на рисунке 3.1.

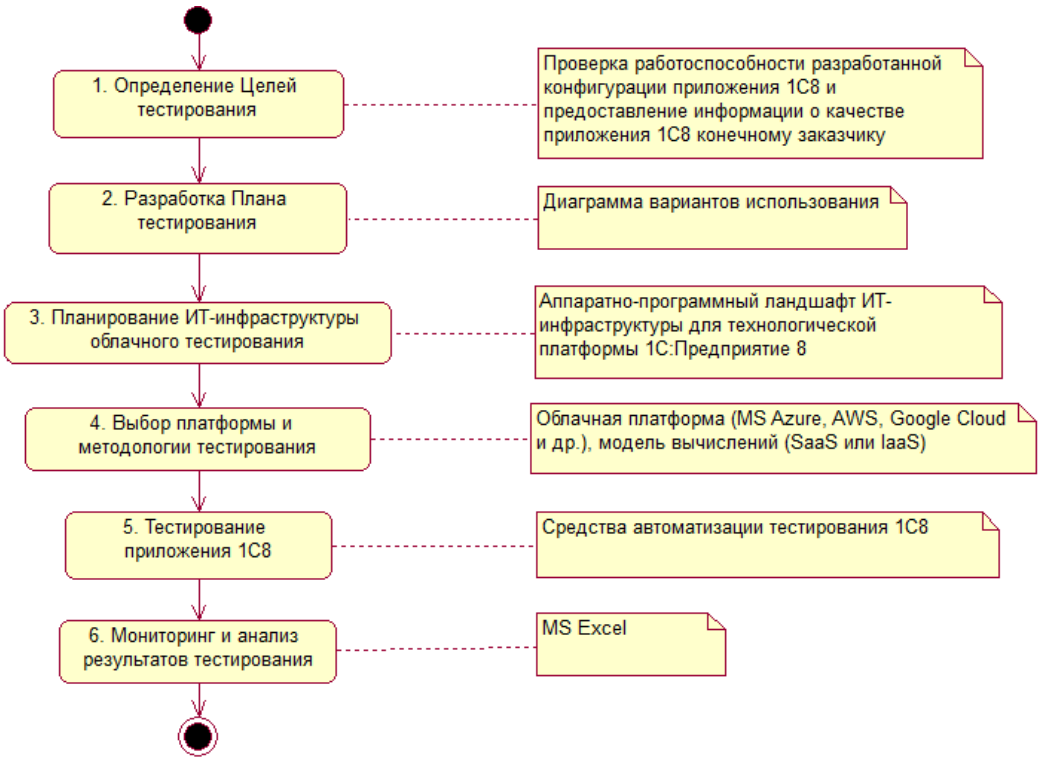



Рисунок 3.1 - Диаграмма деятельности методики облачного тестирования приложения 1С8

Опишем каждый этап данной методики.

1. Цели тестирования: проверка работоспособности разработанной конфигурации приложения 1С8 и предоставление информации о качестве приложения 1С8 конечному заказчику.

2. Разработка плана тестирования.

Тестирование приложения 1С8 включает в себя:

- тестирование конфигурации приложения 1С8;
- тестирование серверной части приложения 1С8.

Для разработки плана облачного тестирования используем диаграмму вариантов использования UML [20].

Для разработки диаграммы вариантов использования применим методологию RUP (Rational Unified Process) [10].

RUP - это итерационная структура процесса разработки программного обеспечения, созданная корпорацией Rational Software. Представляет собой адаптируемую структуру процесса, предназначенную для адаптации вендорами

ПО, которые будут выбирать элементы процесса, соответствующие их потребностям.

Акторами в рассматриваемом случае являются: Тестировщик, Средства автоматизации облачного тестирования.

Варианты использования (прецеденты) представлены в таблицах 3.1, 3.2.

Таблица 3.1 - Описание прецедента: Тестирование конфигурации приложения 1С8

| Прецедент: Тестирование конфигурации приложения 1С8 |
|---|
| ID: 1 |
| Краткое описание: выполнение функционального тестирования и тестирования производительности приложения 1С8 |
| Главный актор: Тестировщик |
| Второстепенные акторы: Средства автоматизации тестирования 1С8 |
| Предусловие: прецедент начинается по инициативе Тестировщика |
| Основной поток: Тестировщик запускает процесс тестирования конфигурации приложения 1С8 |
| Постусловие: средствами автоматизации тестирования 1С8 формируются результаты тестирования. |
| Альтернативные потоки: нет |

Таблица 3.2 - Описание прецедента: Тестирование серверной части приложения 1С8

| Прецедент: Тестирование серверной части приложения 1С8 |
|---|
| ID: 2 |
| Краткое описание: проведение нагрузочного тестирования серверной части приложения 1С8. |
| Главный актор: Тестировщик |
| Второстепенные акторы: Средства автоматизации тестирования 1С8 |
| Предусловие: прецедент начинается по инициативе Тестировщика |
| Основной поток: Тестировщик запускает процесс нагрузочного тестирования серверной части приложения 1С8. |
| Постусловие: средствами автоматизации тестирования 1С8 формируются результаты тестирования. |

| Альтернативные потоки: нет |
|---|

Диаграмма вариантов использования плана облачного тестирования приложения 1С8, разработанная на основе вышеописанных рекомендаций, представлена на рисунке 3.2.

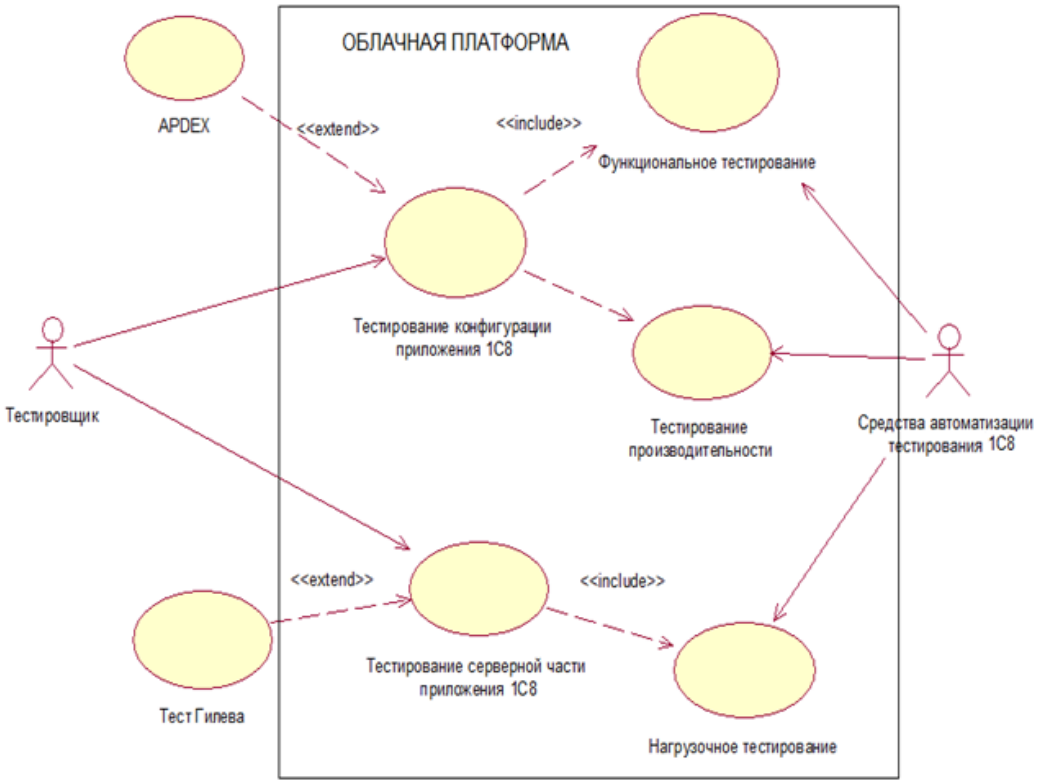



Рисунок 3.2 – Диаграмма вариантов использования плана облачного тестирования приложения 1С8

Как следует из диаграммы, в качестве методической основы тестирования конфигурации приложения 1С8 используется методика Apdex.

Apdex - открытый международный стандарт, разработанный с целью формирования объективной оценки показателей производительности корпоративных информационных систем [19].

Такая методика позволяет:

– привести к простому значению разнородные факторы и множество статистических данных о производительности. Главное преимущество методики - в простом результате, для быстрой оценки состояния производительности информационной системы;

– ранжировать отслеживаемые операции по приоритетности с точки зрения бизнеса, что позволяет правильно акцентировать внимание при мониторинге и оптимизации большого количества операций;

– построить индекс на основании фактических данных, полученных при работе всех пользователей приложения. Результирующая оценка производительности по методике Apdex является общей, фактической и объективной.

Apdex является числовой мерой удовлетворенности пользователей производительностью приложений.

Для расчета Apdex собирается множество статистических данных о времени исполнения операций приложением.

В качестве методической основы тестирования серверной части приложения 1С8 используется тест Гилева (ТРС-1С) [14].

Тест Гилева в условных единицах также условно показывает возможную производительность системы 1С – чем выше значение, тем выше потенциальная производительность.

В основу данного теста положен подход к оценке производительности сервера, основанный на том, что последняя определяется не загруженностью и очередями к процессору, а способностью выполнить количество операций в единицу времени.

3. Планирование инфраструктуры.

Аппаратно-программный ландшафт ИТ-инфраструктуры среды тестирования определяется соответствующими требованиями технологической платформы 1С: Предприятие 8.

4. Выбор платформы и поставщика облачного сервиса.

Основные критерии выбора таких сервисов:

- надежность партнера, принадлежность к брендовым вендорам;
- возможность создания нужной архитектуры работы;
- размещение за рубежом или в России;
- производительность сервисов;

– стоимость арендуемых ресурсов.

При выборе облачной платформы следует руководствоваться рекомендациями, представленными выше. Принимая во внимание, что платформа 1С8 ориентирована на работу в ОС Windows, рекомендуется отдать предпочтение MS Azure. В качестве методологий выбираем SaaS или IaaS.

5. Тестирование приложения 1С8. Для автоматизации используются встроенные средства автоматизации тестирования технологической платформы 1С: Предприятие 8.

6. Мониторинг и анализ результатов тестирования.

Для сбора, обработки и визуализации результатов облачного тестирования рекомендуется использовать табличный процессор Excel.

## 3.2 Методика облачного тестирования приложений 1С:Битрикс: «Управление сайтом»

«Управление сайтом» — это программный продукт, предназначенный для создания и управления интернет-магазинов, сайтов-визиток и т.д. Существует огромное количество редакций этого программного продукта, но все они предназначены для одной цели – создания сайта и дальнейшего его обслуживания (наполнения, редактирования и пр.) [6].

Диаграмма деятельности методики облачного тестирования приложения 1С:Битрикс представлена на рисунке 3.3.

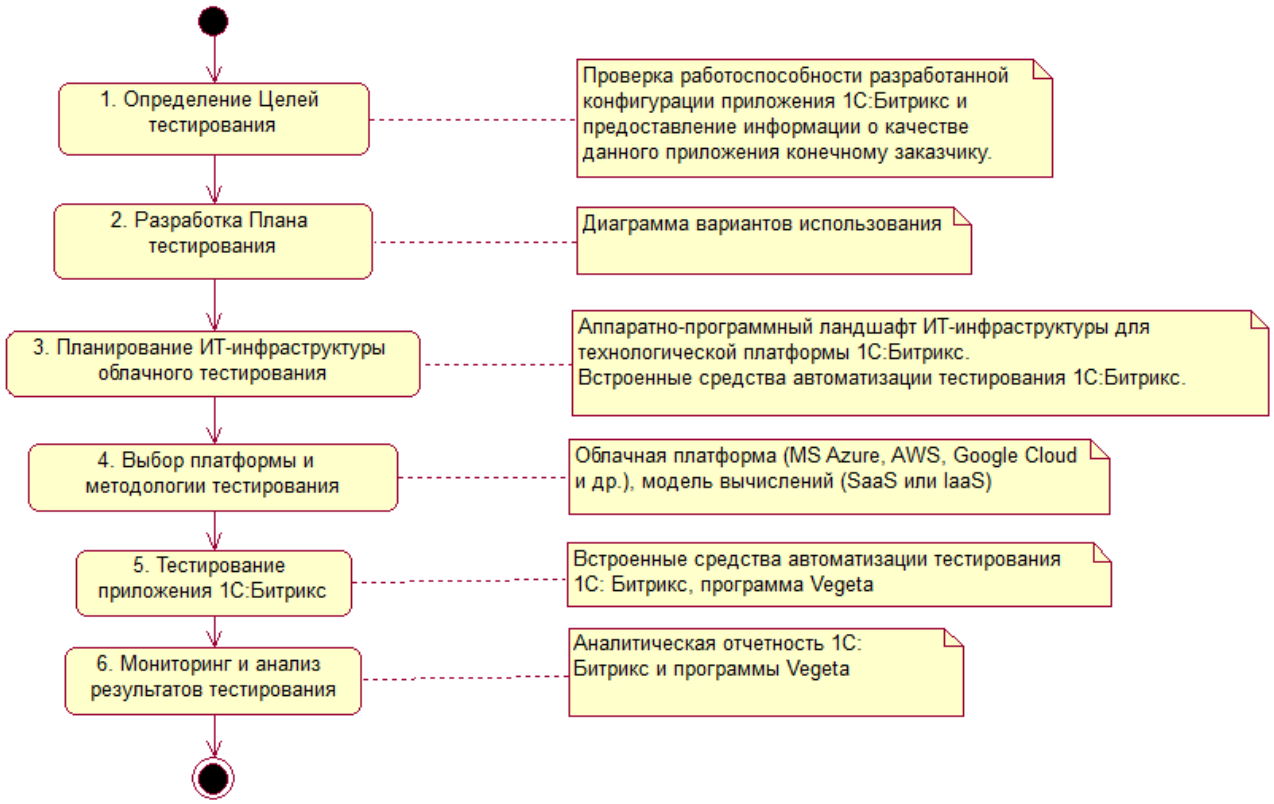



Рисунок 3.3 - Диаграмма деятельности методики облачного тестирования приложения 1С: Битрикс

Рассмотрим этапы данной методики.

1. Цели тестирования: проверка работоспособности разработанной конфигурации приложения 1С:Битрикс и предоставление информации о качестве данного приложения конечному заказчику.

2. Разработка плана тестирования.

Тестирование приложения 1С:Битрикс включает в себя следующие виды тестирования:

- тестирование веб-приложения;
- тестирование серверной части.

Акторами в рассматриваемом случае являются: Тестировщик, Средства автоматизации тестирования 1С:Битрикс, программа Vegeta.

Варианты использования (прецеденты) представлены в таблицах 3.3, 3.4.

Таблица 3.3 - Описание прецедента: Тестирование веб-приложения 1С:Битрикс

| Прецедент: Тестирование веб-приложения 1С:Битрикс |
|---|
| ID: 1 |
| Краткое описание: выполнение тестирования скорости сайта, тестирование конфигурации веб-приложения, тестирование производительности страниц сайта и тестирования на масштабирование. |
| Главный актор: Тестировщик |
| Второстепенные акторы: Средства автоматизации тестирования 1С:Битрикс |
| Предусловие: прецедент начинается по инициативе Тестировщика |
| Основной поток: Тестировщик запускает процесс тестирования веб-приложения 1С:Битрикс |
| Постусловие: средствами автоматизации тестирования 1С:Битрикс формируются результаты тестирования. |
| Альтернативные потоки: нет |

Таблица 3.4 - Описание прецедента: Тестирование серверной части приложения 1С:Битрикс

| Прецедент: Тестирование серверной части приложения 1С:Битрикс |
|---|
| ID: 2 |
| Краткое описание: проведение нагрузочного тестирования серверной части приложения 1С8. |
| Главный актор: Тестировщик |
| Второстепенный актор: программа Vegeta |
| Предусловие: прецедент начинается по инициативе Тестировщика |
| Основной поток: Тестировщик запускает процесс нагрузочного тестирования серверной части приложения 1С:Битрикс. |
| Постусловие: средствами программы Vegeta формируются результаты тестирования. |
| Альтернативные потоки: нет |

Диаграмма вариантов использования плана облачного тестирования приложения 1С:Битрикс, разработанная на основе вышеописанных рекомендаций, представлена на рисунке 3.4.

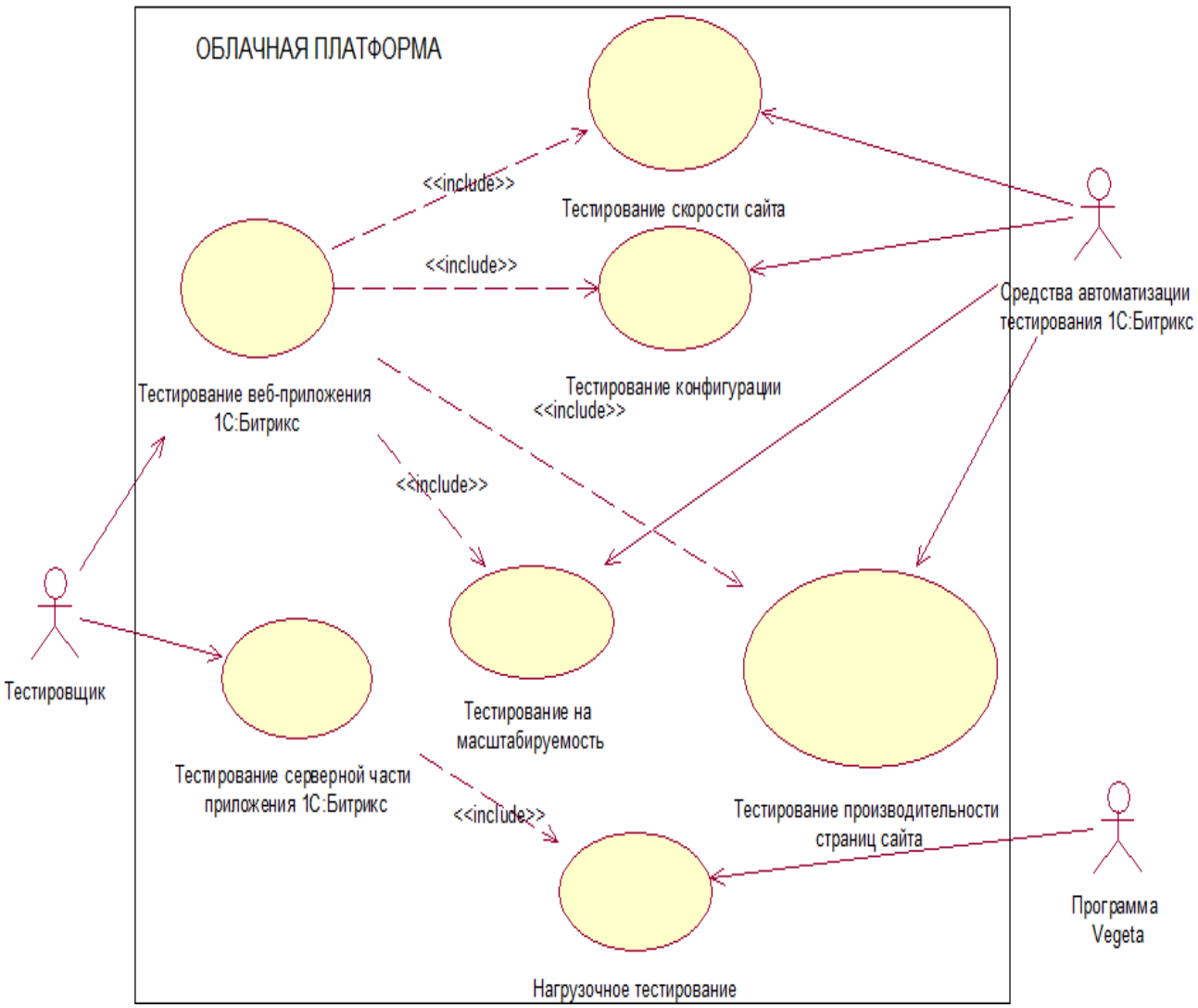



Рисунок 3.4 – Диаграмма вариантов использования плана облачного тестирования приложения 1С:Битрикс

Как следует из диаграммы, помимо встроенных средств тестирования 1С:Битрикс, для автоматизации тестирования серверной части используется программа-бенчмарк Vegeta [18].

Vegeta – это бесплатная утилита командной строки для тестирования HTTP-сервисов, написанная на языке Go. Её можно подключить как библиотеку для создания своих инструментов нагрузочного тестирования.

3. Планирование инфраструктуры.

Аппаратно-программный ландшафт ИТ-инфраструктуры среды тестирования определяется соответствующими требованиями платформы 1С: Битрикс, программа Vegeta.

4. Выбор платформы и поставщика облачного сервиса осуществляется в соответствии с рекомендациями, приведенными выше. В качестве методологий выбираем SaaS или IaaS.

5. Тестирование приложения 1С:Битрикс. Для автоматизации используются встроенные средства автоматизации тестирования 1С: Битрикс и программа Vegeta.

6. Мониторинг и анализ результатов тестирования.

Для сбора, обработки и визуализации результатов облачного тестирования используется встроенные средства 1С: Битрикс и программы Vegeta, которая позволяет создавать отчеты в текстовом и графическом формате.

## Выводы к третьей главе

Для автоматизации облачного тестирования приложения 1С8 используются встроенные средства автоматизации тестирования технологической платформы «1С: Предприятие 8». Для сбора, обработки и визуализации результатов облачного тестирования рекомендуется использовать табличный процессор Excel.

В процессе облачного тестирования приложения 1С:Битрикс помимо встроенных средств тестирования данной платформы для автоматизации тестирования серверной части используется программа-бенчмарк Vegeta. Для сбора, обработки и визуализации результатов облачного тестирования используется встроенные средства системы управления контентом 1С: Битрикс и программы Vegeta, которая позволяет создавать отчеты в текстовом и графическом формате.

# Глава 4 ОЦЕНКА ЭФФЕКТИВНОСТИ МЕТОДИК ОБЛАЧНОГО ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЙ 1С

## 4.1 Оценка эффективности методики облачного тестирования приложений 1С8

С помощью предлагаемой методики определим показатели работы с учетной программой 1С8 в облачной среде MS Azure, представляющей IaaS-сервис.

Для проведения теста на облачной платформе была создана виртуальная ИТ-инфраструктура, имеющая конфигурацию, представленную в таблице 4.1 [16].

Таблица 4.1 – Характеристики виртуальной ИТ-инфраструктуры

| Ресурс | Роль |
|---|---|
| Виртуальная машина F8 v2 (8 vCPU, 16 ГБ ОЗУ) | Сервер 1С + SQL Server |
| Хранилище Premium SSD 512 ГБ | Диск для баз данных 1С |
| Виртуальная машина E4 v3 (4 vCPU, 32 ГБ ОЗУ) | Сервер RDS (служба удаленного стола) |
| Хранилище Standard HDD 512 ГБ | Диск для профилей пользователей |
| Шлюз VPN | VPN Site-to-Sit |

Конфигурация сервера была выбрана исходя из калькуляции необходимых ресурсов для работы 10 пользователей с конфигурациями 1С: Управление Торговлей 11.3.

На тестовый сервер был установлен сервер 1С предприятия, СУБД SQL Server Standard 2017 и толстый клиент 1С. Протокол взаимодействия сервера 1С и SQL-сервера - Shared Memory, по умолчанию использующийся при размещении сервера 1С и SQL-сервера на одной платформе.

На тестовом сервере размещено две базы: база для теста Гилева, Управление Торговлей 11.3 - реальная база, в которых могут работать пользователи [9].

В конфигурацию «Управление Торговлей» была встроена система Apdex и сценарий тестирования по выполнению стандартных операций, которые делают пользователи ежедневно в подобных конфигурациях.

Для конфигурации 1С:Управление торговлей 11.3 это:

1) Проведение возврат товаров от клиента.
2) Проведение возврат товаров поставщику.
3) Проведение заказа клиента.
4) Проведение пересчет товаров.
5) Проведение поступление ТУ.
6) Проведение реализации ТУ.
7) Проведение РКО.

Разработан сценарий, по которому запускалось 10 виртуальных рабочих мест, которые одновременно выполняли ряд операций по сценарию, итогом которого было измерение время выполнения каждой из операции.

Каждый тест на платформе проводился 3 раза в разный промежуток времени, чтобы исключить фактор “нагрузки” платформы, на которой тестировалось приложение 1С8.

Результаты тестирования по тесту Гилева представлены в таблице 4.2

Таблица 4.2 - Итоги теста Гилева

| Перечень тестов (среднее значение по итогам серии из 3 тестов) | MS Azure |
|---|---|
| Проведение тестов Гилева, в условных единицах | 25,5 |

Данный результат относится к «зеленой» зоне результатов теста Гилева и считается хорошим для данного типа задач.

Следует учесть, что относительный показатель и, исходя из практики, очень чувствительный к частоте процессора, который используется на платформе, как собственно и система 1С8.

Следующим этапом было проведено тестирование по методике Apdex на конфигурации Управлении Торговлей 11.3.

На рисунке 4.2 представлен пример кода сценария тестирования.

```
#Если Сервер ИЛИ ТолстыйКлиентОбычноеПриложение ИЛИ ВнешнееСоединение Тогда
#Область Обработчики
// Выполняет проверку конфигурации в соответствии с переданным списком требований.
//
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
	Проверка.ОбработкаПроведенияДокументаПроверки(ЭтотОбъект);
КонецПроцедуры

// Обработчик перед записью документа. Синхронизирует информацию конфигурации по
версии.
//
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
	Если ОбменДанными.Загрузка Тогда
		Возврат;
	КонецЕсли;
	Если Версия.Владелец <> Конфигурация Тогда
		Конфигурация = Версия.Владелец;
	КонецЕсли;

КонецПроцедуры
#КонецОбласти
#КонецЕсли
```

Рисунок 4.1 – Код сценария тестирования

Результаты представлены в таблице 4.3 и рисунке 4.2, соответственно.

Таблица 4.3 - Итоги тестов Apdex, 1С:Управление торговлей 11.3

| Перечень тестов | MS Azure |
|---|---|
| Проведение и возврат от клиента, секунды | 5,15 |
| Проведение и возврат товаров поставщику, секунды | 3,69 |
| Проведение заказа клиента, секунды | 0,96 |
| Проведение пересчета товаров, | 0,39 |

| Перечень тестов | MS Azure |
|---|---|
| секунды | |
| Проведение поступления ТУ, секунды | 3,90 |
| Проведение реализации ТУ, секунды | 3,35 |
| Проведение РКО, секунды | 1,86 |

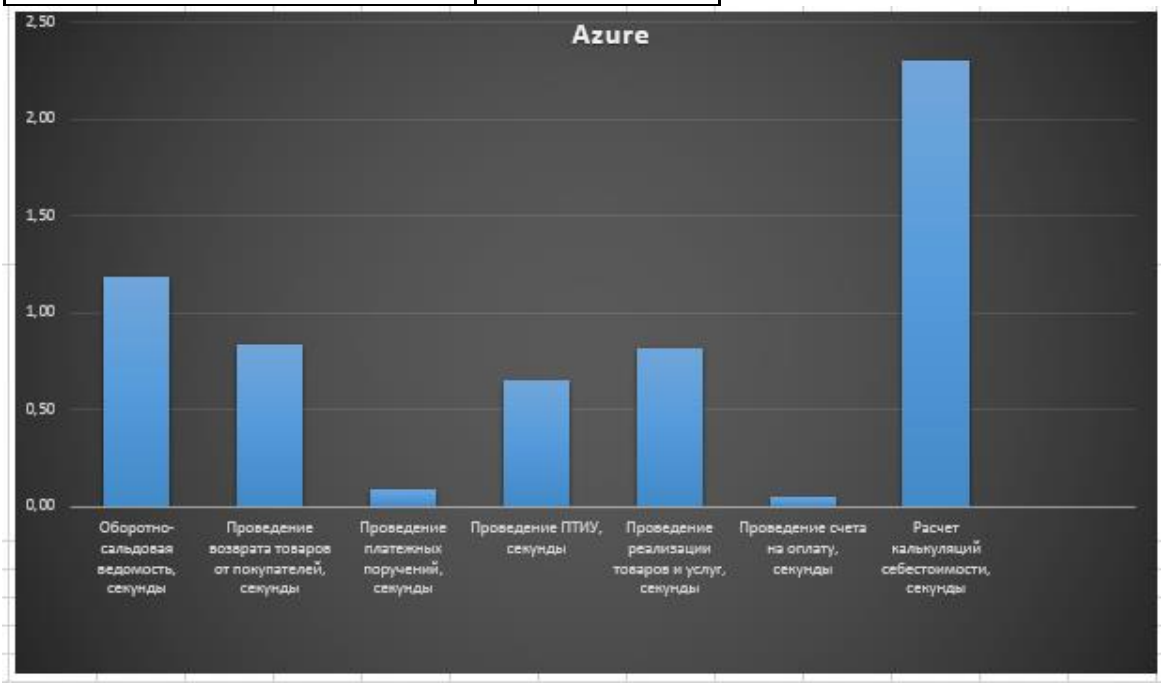



Рисунок 4.2 - Результаты тестирования по методике Apdex конфигурации 1С: Управление Торговлей

Следует отметить, что система 1С является очень сложным программным продуктом, конструктором, из элементов которого собирается информационная система предприятия.

Очень важно уметь правильно подобрать под эту информационную систему аппаратную платформу, которая бы отвечала всем требованиям программного обеспечения.

Именно поэтому необходимо использовать специализированная платформы для достижения максимальной производительности в рассматриваемых тестах.

В платформе для 1С важны и система виртуализации, и протокол работы серверов 1С и SQL, и даже настройки BIOS хост-машин, на которых располагаются виртуальные машины, поэтому при выборе облака для 1С рекомендуется выбирать специализированного хостинг-провайдера, который сможет предоставить максимально производительную платформу для пользовательской информационной системы.

## 4.2 Оценка эффективности методики облачного тестирования приложений 1С:Битрикс

Рассмотрим результаты облачного тестирования ИТ-решения на основе приложения 1С:Битрикс.

В качестве облачной платформы выбрана платформа MS Azure, предоставляющая сервисы IaaS и SaaS.

1) Тестирование скорости сайта (рисунок 4.3).

Эта страница отображает статистику по загрузку сайта, собираемую во время работы пользователей с сайта.

Функции:

− отображает скорость загрузки для пользователей с разных регионов;

− позволяет узнать минимальную, среднюю и наибольшую скорость загрузки страниц;

− позволяет увидеть, какие подсистемы влияют на скорость отрисовки страницы: скорость отдачи от сервера, dns-сервер, обработка html, прочая обработка.

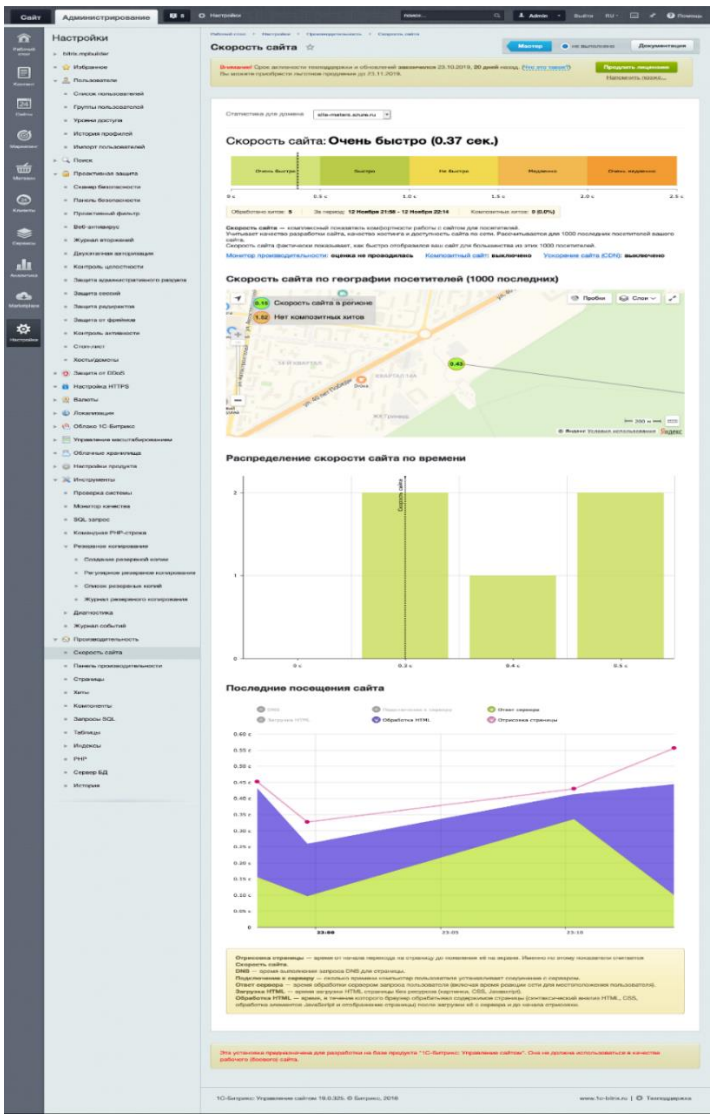



Рисунок 4.3 – Страница тестирования скорости сайта 1С:Битрикс

2) Тестирование конфигурации.

Позволяет определить, насколько хватает хостинга для оптимальной работы системы.

Проверяются процессор, память, файловая подсистема.

Также проверяется работа php, почты, MySQL (рисунок 4.4)

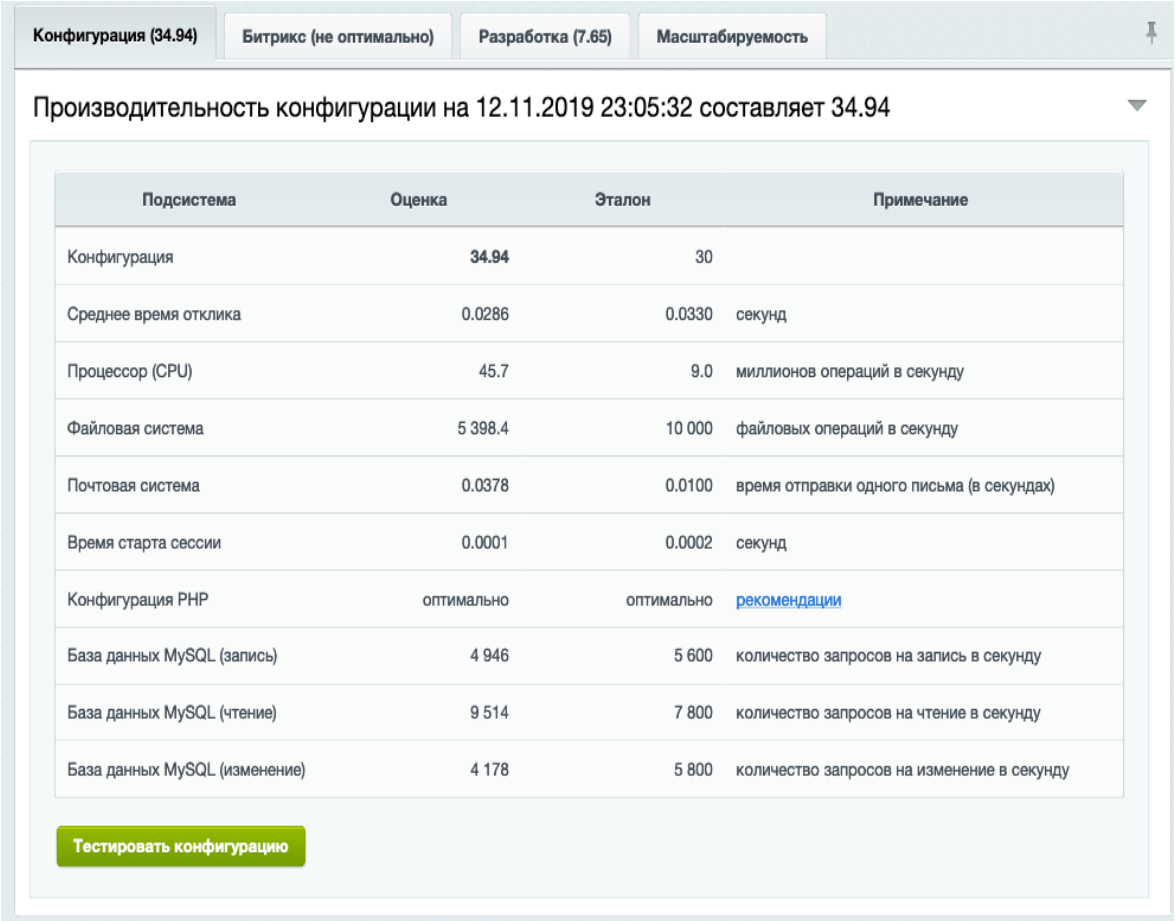



Рисунок 4.4 – Фрагмент страница тестирования конфигурации веб-приложения

3) Тестирование производительности страниц сайта (рисунок 4.5).

Запускается на панели разработки.

Позволяет найти медленные страницы сайта, выявить как элементы на сайте замедляют ее работу, найти отключённый кеш для «тяжелых» страниц и др.

4) Тест на масштабируемость.

Позволяет получить для одно или много серверных решений базовые метрики по отдаваемых страниц в секунду и времени генерации страницы (рисунок 4.6).

5) Нагрузочное тестирование серверов.

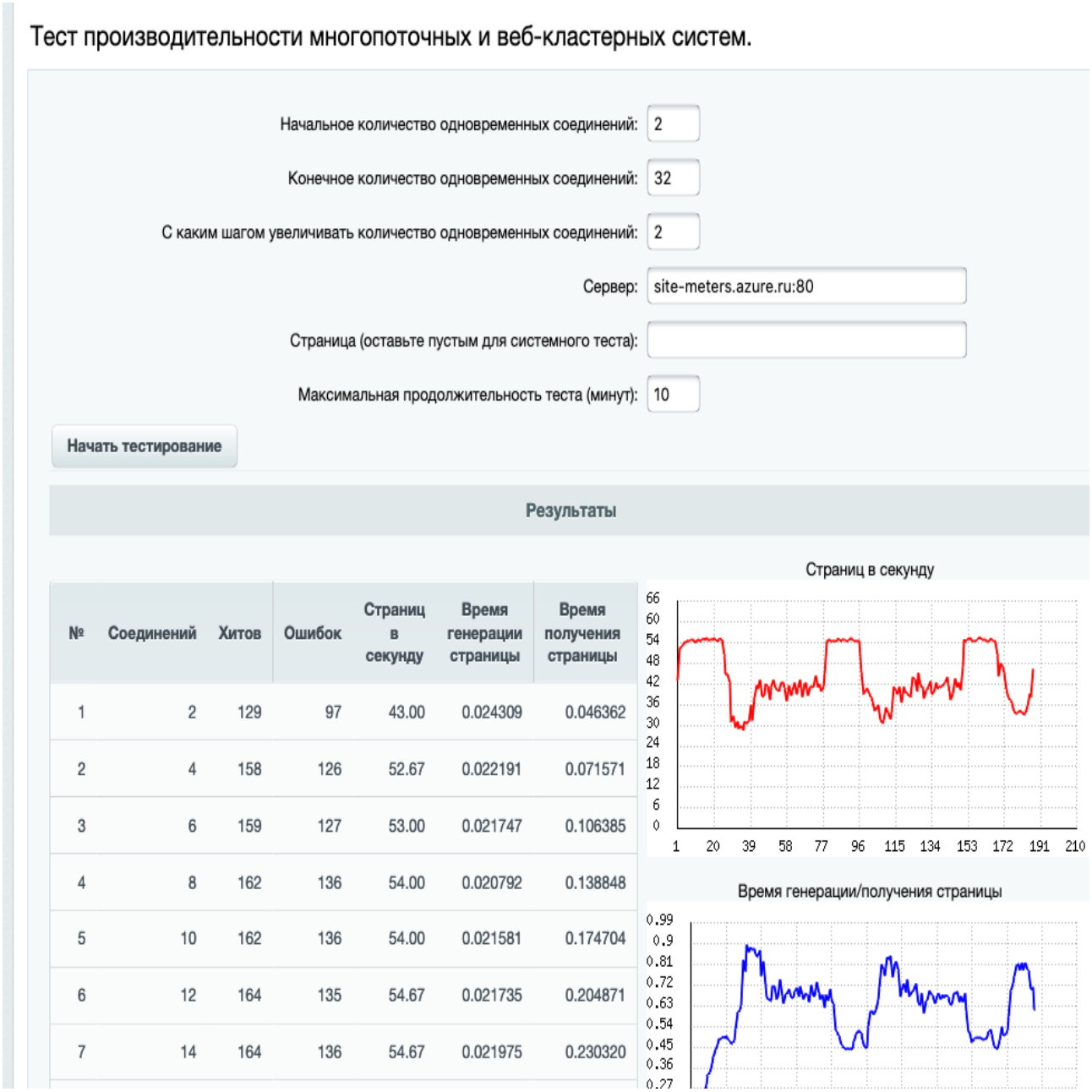



Рисунок 4.5 – Фрагмент страницы тестирования производительности страниц сайта

Для тестирования нагрузки на сервер лучше использовать внешние приложения для тестирования.

Поскольку они лучше позволяют вычислить максимальные возможности сервиса, лишние ресурсы сервера не тратятся на систему мониторинга внутри сервиса.

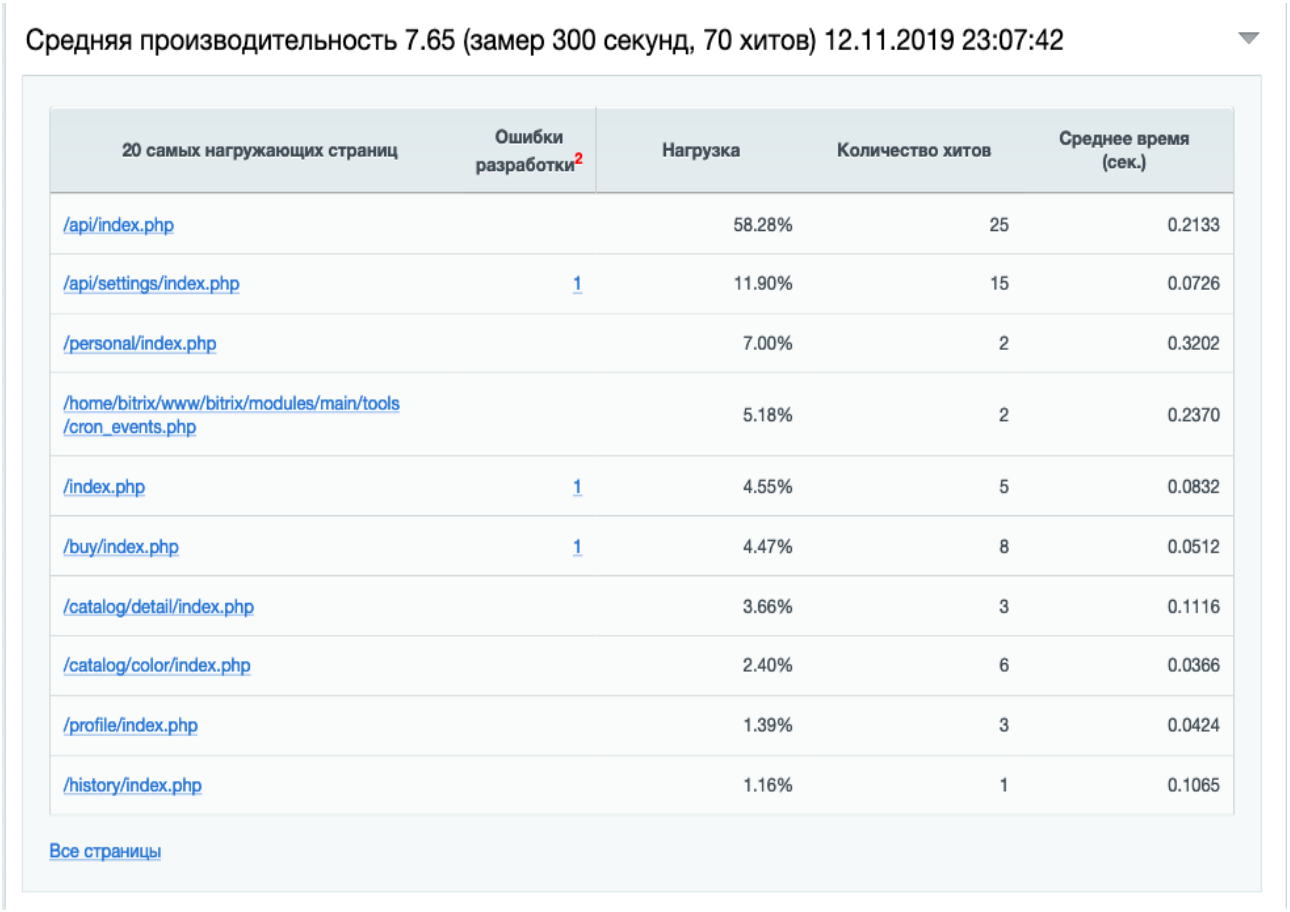


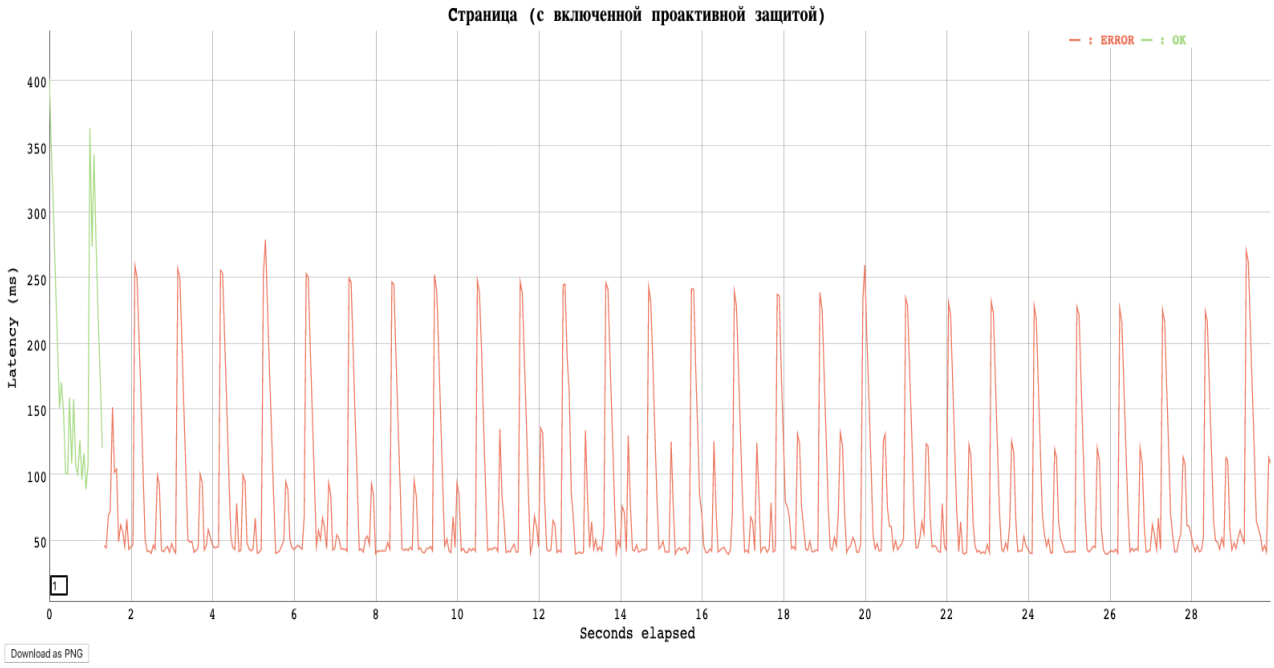

Рисунок 4.6 – Фрагмент страницы тестирования на масштабируемость

Для этого этой цели используем приложение Vegeta.

Следует отметить, что в продукте «1С-Битрикс: Управление сайтом» реализован механизм, который позволяет подключать к сайту любые «облака» и легко управлять ими - вплоть до обмена данными между хранилищами.

Рассмотрим отчеты результатов тестирования с помощью программы Vegeta.

Графики результатов нагрузочного тестирования для страниц с включенной проактивной защитой (позволяет отсекать слишком частое обращение к серверу от пользователя) приведены на рисунке 4.7.



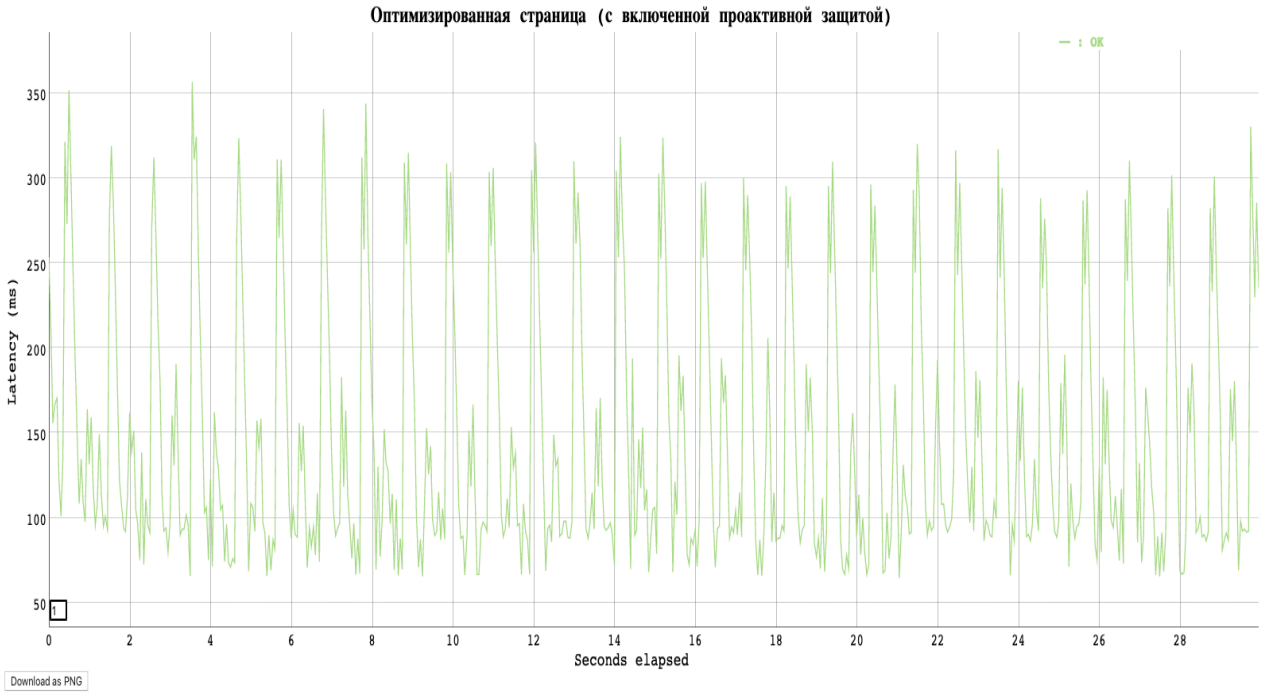




Рисунок 4.7 – Графики результатов нагрузочного тестирования для страниц с включенной проактивной защитой

Графики результатов нагрузочного тестирования для страниц с отключенной проактивной защитой приведены на рисунке 4.8.

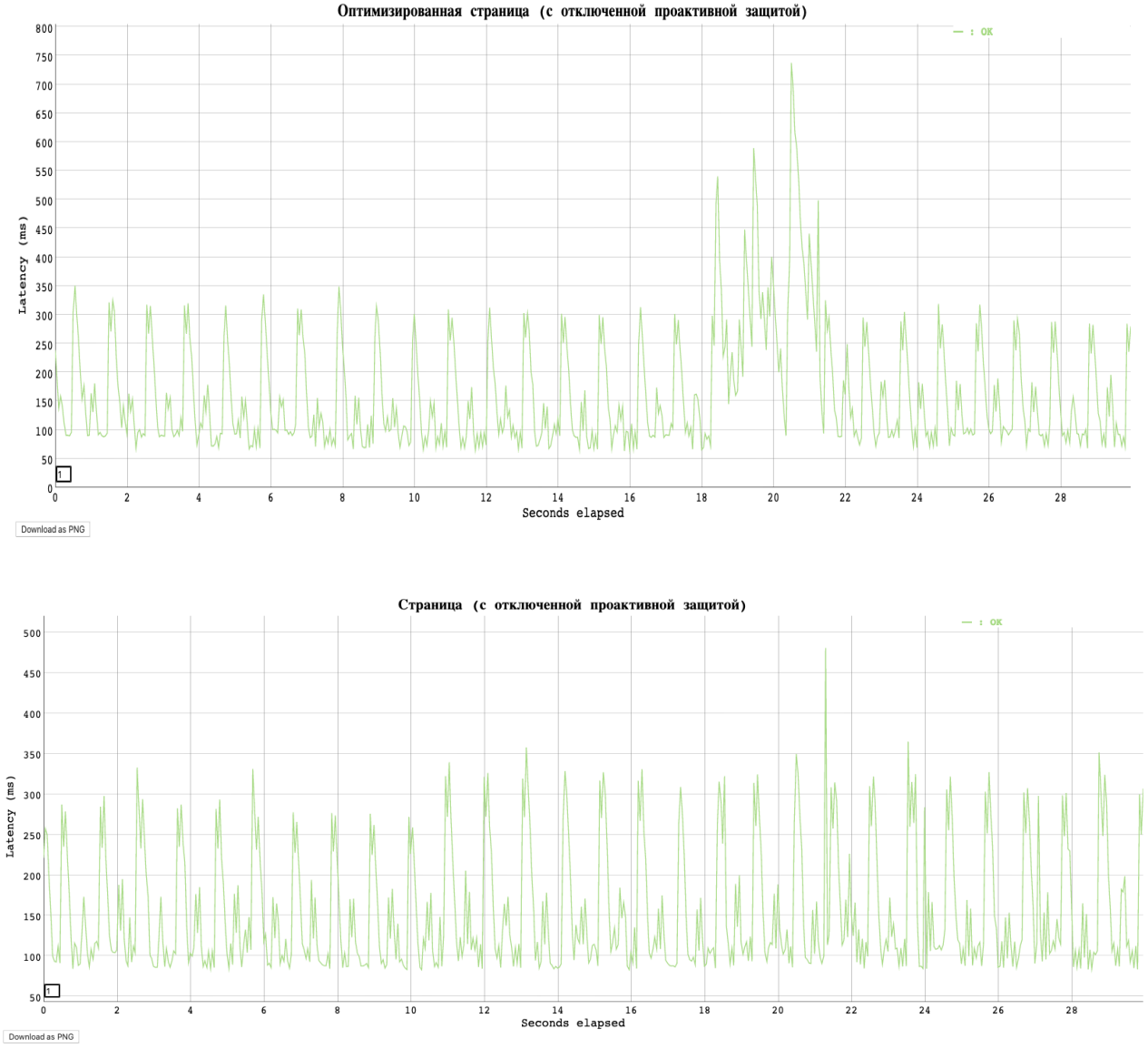



Рисунок 4.8– Графики результатов нагрузочного тестирования для страниц с включенной проактивной защитой

На рисунке 4.9 представлен команды программы Vegeta для вывода результатов тестирования.

```
echo "GET http://site-meters.azure.ru/" | vegeta attack -duration=30s -
rate=20 | tee results.bin | vegeta report
vegeta plot -title=Results results.bin > results-plot1.html
echo "GET http://site-meters.azure.ru/optimaze_page/" | vegeta attack -
duration=30s -rate=20 | tee results.bin | vegeta report
vegeta plot -title=Results results.bin > results-plot2.html
echo "GET http://site-meters.azure.ru/" | vegeta attack -duration=30s -
rate=20 | tee results.bin | vegeta report
vegeta plot -title=Results results.bin > results-plot3.html
echo "GET http://site-meters.azure.ru/optimaze_page/" | vegeta attack -
duration=30s -rate=20 | tee results.bin | vegeta report
vegeta plot -title=Results results.bin > results-plot4.html
```

Рисунок 4.9– Команды программы Vegeta для вывода результатов тестирования

Таким образом, предлагаемая методика обеспечивает тестирование различных видов ПО, что подтверждает ее универсальность и эффективность.

## Выводы к четвертой главе

При выборе облака для 1С8 рекомендуется выбирать специализированного хостинг-провайдера, который сможет предоставить максимально производительную платформу для пользовательской информационной системы.

Предлагаемая методика обеспечивает тестирование различных видов ПО, что подтверждает ее универсальность и эффективность.

# ЗАКЛЮЧЕНИЕ

Целью магистерской диссертации является сокращение затрат на проектирование ПО за счет применения предлагаемой в работе эффективной методики облачного тестирования.

Выполненные в работе научные исследования представлены следующими основными результатами:

1. Произведен анализ современных технологии облачного тестирования ПО, который показал, что главным недостатком известных методик облачного тестирования является недостаточная универсальность. Поэтому представляет актуальность разработка методики облачного тестирования, обладающей большей универсальностью.

2. Произведен анализ методологических подходов к разработке методик облачного тестирования ПО. Для разработки качественных методик облачного тестирования необходимо учитывать особенности облачной платформы, которая будет использована в качестве среды облачного тестирования.

3. Разработаны методики облачного тестирования приложений 1С8 и 1С:Битрикс.

4. Выполнена апробация разработанной методики, которая подтвердила, что предлагаемая методика обеспечивает тестирование различных видов ПО, что подтверждает ее универсальность и эффективность.

Таким образом, в работе решена актуальная научно-исследовательская задача разработки методики облачного тестирования ПО, обеспечивающей повышение эффективности данного процесса.

Значение диссертационной работы определяется тем, что в ее рамках исследованы возможности повышения эффективности процесса облачного тестирования ПО и предложена универсальная методика, обеспечивающая решение данной задачи.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

*Нормативно-правовые акты*

1. ГОСТ Р 53622-2009 Информационные технологии. Информационно-вычислительные системы. Стадии и этапы жизненного цикла, виды и комплектность документов.

2. ГОСТ Р 56922-2016. Системная и программная инженерия. Тестирование программного обеспечения.

*Научная и методическая литература*

3. Ашихмин Н.А. Разработка облачного сервиса для тестирования микросервисных приложений / Н.А. Ашихмин, Д.И. Савченко, Г.И. Радченко, PaVT206, -№ 1576. – С. 411-424.

4. Сафиуллов И.Д. Среда облачного тестирования программного обеспечения / И.Д. Сафиуллов // Вестник научных конференций. -№ 9-3(49). – С.74.

5. Роутон Д. Тестирование программного обеспечения в облаке: как разные облачные платформы обеспечивают тестирование приложений до и после развертывания / Д. Роутон. - IBM Corporation, 2013.

*Электронные ресурсы*

6. 1С:Битрикс [Электронный ресурс]. — Режим доступа: https://www.1c-bitrix.ru/ (дата обращения: 25.09.2019).

7. 1С: Предприятие 8 [Электронный ресурс]. — Режим доступа: https://v8.1c.ru/ (дата обращения: 25.09.2019).

8. Бурняшов Б. А. Информационные технологии в менеджменте. Облачные вычисления [Электронный ресурс] : учебное пособие / Б. А. Бурняшов. — 2-е изд. — Саратов : Вузовское образование, 2019. — 87 с. — Режим доступа: http://www.iprbookshop.ru/79630.html (дата обращения: 25.09.2019).

9. Выбор облачного сервиса для работы в 1С [Электронный ресурс]. — Режим доступа: https://efsol.ru/articles/cloud-services.html (дата обращения: 25.09.2019).

10. Классификация и специфицирование требований (RUP) [Электронный ресурс]. - Режим доступа: https://www.intuit.ru/studies/courses/2188/174/lecture/4726?page=2 (дата обращения: 25.09.2019).

11. Котляров В. П. Основы тестирования программного обеспечения [Электронный ресурс] / В. П. Котляров. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 334 с. — Режим доступа: http://www.iprbookshop.ru/62820.html (дата обращения: 25.09.2019).

12. Лучшие инструменты для тестирования мобильных приложений [Электронный ресурс]. — Режим доступа: https://geteasyqa.com/ru/blog/best-mobile-testing-tools/ (дата обращения: 25.09.2019).

13. Лучшие методики тестирования в JavaScript и Node.js [Электронный ресурс]. — Режим доступа: https://habr.com/ru/company/mailru/blog/466879/ (дата обращения: 25.09.2019).

14. Нагрузочный тест TPC-1C [Электронный ресурс]. — Режим доступа: http://www.gilev.ru/tpc1cgilv/ (дата обращения: 25.09.2019).

15. Облачное тестирование: преимущества, проблемы, типы и советы [Электронный ресурс]. — Режим доступа: https://www.facebook.com/notes/start-it-training-center/%D0%BE%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D0%BE%D0%B5-%D1%82%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%B5%D0%B8%D0%BC%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B0-%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B-%D1%82%D0%B8%D0%BF%D1%8B-%D0%B8-%D1%81%D0%BE%D0%B2%D0%B5%D1%82%D1%8B/2299938326743993/

16. Организация виртуальной инфраструктуры 1С в Microsoft Azure [Электронный ресурс]. — Режим доступа: https://infostart.ru/public/897231/ (дата обращения: 25.09.2019).

17. Особенности тестирования облачных сервисов [Электронный ресурс]. — Режим доступа: http://getbug.ru/osobennosti-testirovaniya-oblachnyih-servisov/#_SaaS (дата обращения: 25.09.2019).

18. Открытые бенчмарки для нагрузочного тестирования серверов и веб-приложений [Электронный ресурс]. — Режим доступа: https://habr.com/ru/company/1cloud/blog/474474/ (дата обращения: 25.09.2019).

19. Оценка интегральной производительности системы по методике APDEX [Электронный ресурс]. — Режим доступа: https://its.1c.ru/db/metod8dev#content:5807:hdoc (дата обращения: 25.09.2019).

20. Самуйлов С.В. Объектно-ориентированное моделирование на основе UML [Электронный ресурс]: учебное пособие/ Самуйлов С.В.— Саратов: Вузовское образование, 2016.— 37 с.— Режим доступа: http://www.iprbookshop.ru/47277.html.— ЭБС «IPRbooks» (дата обращения: 25.09.2019).

21. Сравнение услуг облачных провайдеров [Электронный ресурс]. — Режим доступа: http://la.by/blog/sravnenie-uslug-oblachnyh-provayderov-microsoft-azure-aws-ili-google-cloud (дата обращения: 25.09.2019).

22. Тестирование облачных сервисов [Электронный ресурс]. — Режим доступа: https://www.osp.ru/os/2012/05/13016242 (дата обращения: 25.09.2019).

23. Amazon Web Services [Электронный ресурс]. — Режим доступа: 2

24. Apprenda [Электронный ресурс]. — Режим доступа: https://apprenda.com/ (дата обращения: 25.09.2019).

25. Cloud Computing and Testing Cloud based Applications [Электронный ресурс]. — Режим доступа: https://www.360logica.com/blog/cloud-computing-and-testing-cloud-based-applications/ (дата обращения: 25.09.2019).

26. Cloud testing [Электронный ресурс]. — Режим доступа: https://en.wikipedia.org/wiki/Cloud_testing (дата обращения: 25.09.2019).

27. Cloud Testing Tutorial with SaaS and PaaS [Электронный ресурс]. — Режим доступа: http://www.professionalqa.com/cloud-testing-tutorial-with-saas-and-paas (дата обращения: 25.09.2019).

28. Google Cloud [Электронный ресурс]. — Режим доступа: https://cloud.google.com/ (дата обращения: 25.09.2019).

29. Microsoft Azure [Электронный ресурс]. — Режим доступа: https://azure.microsoft.com/ru-ru/ (дата обращения: 25.09.2019).

30. SaaS vs PaaS vs IaaS: What's The Difference and How To Choose [Электронный ресурс]. — Режим доступа: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/ (дата обращения: 25.09.2019).

31. SOASTA [Электронный ресурс]. — Режим доступа: https://en.wikipedia.org/wiki/SOASTA (дата обращения: 25.09.2019).

32. Software Testing as a Service (TaaS) [Электронный ресурс]. — Режим доступа: https://www.softwaretestingclass.com/software-testing-as-a-service-taas/ (дата обращения: 25.09.2019).

33. Testing as a Service (TaaS) [Электронный ресурс]. — Режим доступа: https://searchcloudcomputing.techtarget.com/definition/Testing-as-a-Service-TaaS (дата обращения: 25.09.2019).

34. Types of Cloud Testing [Электронный ресурс]. — Режим доступа: https://etutorialsworld.com/2016/04/types-of-cloud-testing/ (дата обращения: 25.09.2019).

35. What Is Automation Testing Pyramid? [Электронный ресурс]. — Режим доступа: https://ru.qatestlab.com/resources/knowledge-center/test-automated-pyramid/ (дата обращения: 25.09.2019).

36. What Is Platform-as-a-Service [Электронный ресурс]. — Режим доступа: https://www.cloudflare.com/learning/serverless/glossary/platform-as-a-service-paas/ (дата обращения: 25.09.2019).

*Литература на иностранном языке*

37. Blokland and others. Testing Cloud Services. How to test Saas, Paas and Iaas, EuroSTAR Software Testing Community (2013).

38. Chan and others. Modeling and Testing of Cloud Applications, IEEE APSCC 2009 (Singapore; Dec 7-11, 2009).

39. Jain P. and others. How to Assured Quality in a Cloud and its Verification Process, Council for Innovative Research International Journal of Computers & Technology, Vol. 4, No. 1, Pp. 33 (2013).

40. Malhotra R. and Jain P. Testing Techniques and its Challenges in a Cloud Computing Environment, The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), 1(3), (2013).

41. Mohammad A., Mcheick H. Cloud Services Testing: An Understanding, Procedia Computer Science 5, 513–520 (2011).

42. Mohsenzadeh A. Software Trustworthy Testing Based on Cloud Testing, Journal of mathematics and computer science 14, 284-294, (2015).