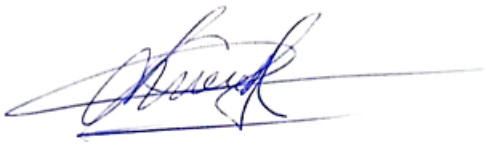
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

На правах рукописи



АМОА Куадио-кан Армел Жеафруа

УПРАВЛЕНИЕ ПРОЦЕССАМИ РАЗРАБОТКИ СПЕЦИАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ В ОБЛАЧНЫХ АРХИТЕКТУРАХ

Специальность: 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель:
д.т.н., доцент Рындин Никита Александрович

Воронеж – 2025

Содержание

ВВЕДЕНИЕ

Актуальность темы. Инфокоммуникационные системы являются базовыми в большинстве значимых аспектов функционирования современного общества. В этой связи особенно важной является задача эффективно управления моделированием и разработкой программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных архитектурах. В рамках жизненного цикла такого программного обеспечения одна из подзадач связана с необходимостью ситуационного моделирования и анализа организации-разработчика специального программного обеспечения с открытым программным кодом,. В конечном счете табличный учет каждого варианта открытости и учет специфики предметной области создаваемого продукта на основе горизонтальных или диагональных связей позволяет разрабатывать и распространять программное обеспечение максимально эффективно. Большой вклад в развитие методов моделирования и разработки программного обеспечения инфокоммуникационных систем с открытым исходным кодом воблачных архитектурах внесли Ковалев И.В., Кравец О.Я., Лукьянчиков О.И., Никульчев Е.В., Ciardo G., Gribaudo M.I., Jakobik A. Разумной идеей кажется создание ситуационной модели организации-разра-ботчика программного обеспечения с открытым программным кодом, обеспечивающей определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

Исследование облачных архитектур вследствие практической безграничности объекта исследования невозможно без моделирования, особенно с учетом высокой стохастичности происходящих процессов. Методы аналитического исследования, к сожалению, еще не развиты в достаточной степени, поэтому на первый план выступают имитационные исследования. Как следствие, особый интерес представляют технологии стохас-

тического моделирования облачной архитектуры, позволяющую автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры.

Важным этапом управления инфокоммуникационными системами в облачных средах является идентификация состояния инфокоммуникационной системы на основе облачных вычислений, обеспечивающая расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки. Разумеется, при этом важен анализ производительности и прогнозирование доступности многоуровневой облачной среды, обеспечивающую учет времени жизни основных компонентов и оценку доступности облачной среды.

Таким образом, актуальность темы диссертационного исследования продиктована необходимостью развития моделей анализа и алгоритмов разработки специального программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных средах на основе ситуационной обработки каждого варианта открытости и учетом специфики предметной области продукта на основе горизонтальных или диагональных связей.

Тематика диссертационной работы соответствует научному направлению ФГБОУ ВО «Воронежский государственный технический университет» «Вычислительные комплексы и проблемно-ориентированные системы управления».

Целью работы является создание моделей анализа и алгоритмов разработки специального программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных средах.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать проблематику сквозного моделирования и алгоритмизации процессов разработки специального программного обеспе-

чения с открытым кодом в облачных средах.

2. Разработать ситуационную модель анализа организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающую определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

3. Предложить модификацию технологии стохастического моделирования облачной архитектуры, позволяющую автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры

4. Разработать алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений, обеспечивающий расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки

5. Разработать модель анализа производительности и прогнозирования доступности многоуровневой облачной среды, обеспечивающую учет времени жизни основных компонентов и оценку доступности облачной среды.

6. Разработать структуру специального программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов, обеспечивающую реконфигурацию инфокоммуникационной системы в зависимости от параметров инфраструктуры и качества обслуживания.

Объект исследования: процессы построения моделей анализа и алгоритмов разработки специального программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных средах.

Предмет исследования: средства математического и программного управления процессами анализа облачной архитектуры и производительности инфокоммуникационных систем на основе алгоритмов генерации имитационных моделей и архитектуры системы распознавании состояний

блокировок.

Методы исследования. При решении поставленных в диссертации задач использовались методы теории графов, теории вероятностей, теории принятия решений, теории моделирования, а также методы объектно-ориентированного программирования.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»: п.3 «Модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем»; п.9 «Модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных»; п.10 «Оценка качества, стандартизация и сопровождение программных систем».

Научная новизна работы. В диссертации получены следующие результаты, характеризующиеся научной новизной:

1. Ситуационная модель анализа организации-разработчика программного обеспечения с открытым программным кодом, отличающаяся табличным учетом каждого варианта открытости и учетом специфики предметной области продукта на основе горизонтальных или диагональных связей и обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

2. Модификация технологии стохастического моделирования облачной архитектуры, отличающаяся использованием мультиформального подхода для определения классов рабочей нагрузки, конфигурации облака и взаимосвязей между рабочей нагрузкой и облаком в сценариях мульти-облачных/гибридных облачных/пограничных вычислений, позволяющая автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры.

3. Алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений, отличающийся событийным мониторингом ситуаций блокировки системы и обеспечивающий расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки

4. Модель анализа производительности и прогнозирования доступности многоуровневой облачной среды, отличающаяся периодическим расчетом времени жизни облачной системы и обеспечивающая учет времени жизни основных компонентов и оценку доступности облачной среды.

5. Структура комплекса программного обеспечения процесса распознавания блокировок и оптимизации доступности облачных сервисов, отличающаяся реализацией механизмов интеграции системы мониторинга и облачных сервисов и обеспечивающая реконфигурацию инфокоммуникационной системы в зависимости от параметров инфраструктуры и качества обслуживания.

Теоретическая и практическая значимость исследования заключается в развитии специальных средств разработки специального программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных средах на основе ситуационной обработки каждого варианта открытости и учетом специфики предметной области продукта на основе горизонтальных или диагональных связей.

Теоретические результаты работы могут быть использованы в проектных и научно-исследовательских организациях, занимающихся разработкой программного обеспечения инфокоммуникационных систем с открытым исходным кодом в облачных средах.

Положения, выносимые на защиту

1. Ситуационная модель анализа организации-разработчика программного обеспечения с открытым программным кодом обеспечивает определение степени открытости программного кода и уровень стратегиче-

ской открытости организации в экосистеме жизненного цикла.

2. Модификация технологии стохастического моделирования облачной архитектуры позволяет автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры

3. Алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений обеспечивает расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки.

4. Модель анализа производительности и прогнозирования доступности многоуровневой облачной среды обеспечивает учет времени жизни основных компонентов и оценку доступности облачной среды.

5. Структура программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов позволяет осуществить реконфигурацию инфокоммуникационной системы в зависимости от параметров инфраструктуры и качества обслуживания.

Результаты внедрения. Основные результаты работы внедрены в ООО «Центр информационных технологий» (г. Воронеж) при проектировании доступной облачной системы управления инфокоммуникационными сервисами, в учебный процесс Воронежского государственного технического университета в рамках дисциплин: «Вычислительные машины, системы и сети», «Информационные сети и телекоммуникационные технологии», а также в рамках курсового и дипломного проектирования.

Апробация работы. Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях: VII Международной НПК «Наука и технологии: перспективы развития и применения» (Петрозаводск, 2024); VI Всероссийской НПК «Информационные технологии в экономике и управлении» (Махачкала, 2024); XXX International Open Science Conference «Modern informatization problems in

the technological and telecommunication systems analysis and synthesis» (Yelm, WA,, USA, 2025), а также на научных семинарах кафедр САПРИС и искусственного интеллекта и цифровых технологий ВГТУ (2019-2025 гг.).

Обоснованность и достоверность полученных результатов обусловлена корректным использованием теоретических методов исследования и подтверждена результатами сравнительного анализа данных вычислительных и натурных экспериментов.

Публикации. По результатам диссертационного исследования опубликовано 12 научных работ (5 – без соавторов), в том числе 5 – в изданиях, рекомендованных ВАК РФ (из них 1 – в издании Wos и одно свидетельство о регистрации программы для ЭВМ). В работах, опубликованных в соавторстве и приведенных в конце автореферата, лично автором получены следующие результаты: [2, 12] - технология стохастического моделирования облачной архитектуры на основе мультиформального подхода; [3] - алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений; [8] - ситуационная модель организации-разработчика программного обеспечения с открытым программным кодом; [4] - структура программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов; [1, 5] - информационное и программное обеспечение для экспериментальной оценки качества разработанных методов и алгоритмов.

Структура и объем работы. Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы из 169 наименований. Работа изложена на 167 страницах основного текста.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе исследуются особенности математического и программного управления процессами исследования облачных архитектур и

производительности инфокоммуникационных систем на основе алгоритмов генерации имитационных моделей и архитектуры системы распознавании состояний блокировок. Отмечено, что повысить эффективность управления можно путем разработки ситуационной модели организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

Исследована проблема автоматической генерации стохастических имитационных моделей с высоким уровнем согласованности с поведением облачной архитектуры. Определен мультиформальный метод, позволяющий интегрировать ситуационный подход к высокоуровневому моделированию с известным симулятором облачных архитектур для получения более реалистичных результатов в процессе анализа производительности. Этот метод призван заполнить пробел между предварительными (аналитическими) моделями анализа производительности и подробными имитационными моделями на основе поэтапного моделирования.

Далее необходимо провести исследование и разработку алгоритма идентификации состояния инфокоммуникационной системы на основе облачных вычислений, обеспечивающего расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки.

Затем нужно решить задачу построения модели анализа производительности и прогнозирования доступности многоуровневой облачной среды, обеспечивающей учет времени жизни основных компонентов и оценку доступности облачной среды.

Наконец, решение задачи проектирования структуры программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов, обеспечивающей реконфигурацию инфокоммуникаци-

онной системы в зависимости от параметров инфраструктуры и качества обслуживания, также является важным этапом исследования. Сформулирована цель и задачи исследования.

Вторая глава посвящена разработке ситуационной модели анализа организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающей определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

Программные экосистемы определяются как совокупность субъектов, функционирующих как единое целое и взаимодействующих на общем рынке программного обеспечения и услуг, а также отношений между ними. Эти отношения поддерживаются общей технологической платформой или рынком и осуществляются посредством обмена информацией, ресурсами и артефактами. Существует несколько факторов, побуждающих к созданию экосистемы программного обеспечения, таких как способность более гибко реагировать на меняющиеся требования к различные специализированные решения, основанные на платформе, или более быстрое внедрение, чем у конкурентов. Выживание и эффективность организаций в экосистеме взаимозависимы.

Оперирующая система определяется как организация, производящая программное обеспечение и предоставляющая доступ к своим процессам по крайней мере одному из типов участников, которые присутствуют в ее программной экосистеме. Оперирующие системы могут быть любого типа, от независимых поставщиков программного обеспечения до организаций с открытым исходным кодом.

Ситуационная модель направлена на установление того, насколько открытым или закрытым является SPO. Смысл существования модели заключается в трех аспектах. Во-первых, необъективные решения о снабжении принимаются на основе предвзятой оценки открытости, что не позво-

ляет выиграть лучшему продукту или поставщику. Во-вторых, SPO не знают о различных существующих вариантах открытости. В третьих, эта модель направлена на то, чтобы сместить акцент с “открытого исходного кода” и показать, что открытость - это нечто большее, чем просто лицензия на открытое программное обеспечение.

Модель оперирующей системы создана в двух измерениях, одним из которых является измерение практик SPO и измерение уровня управления. В измерении уровня управления есть три уровня: стратегический, тактический и оперативный (от долгосрочного до краткосрочного).

Представлена ситуационная модель организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации.

Третья глава посвящена разработке модификации технологии стохастического моделирования облачной архитектуры, позволяющей автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры.

Представлена эволюция модели облачной архитектуры на протяжении нескольких итераций. Технология направлена на разработку облачных приложений с учетом проблем с производительностью с самого начала цикла проектирования.

Подход основан на процессе моделирования, при котором разрабатываются модели, состоящие из двух (или двух наборов) подмоделей: первая подмодель представляет приложение, вторая - используемую облачную инфраструктуру. В то время как первое представляет собой нечто, что полностью контролируется разработчиком, о втором ничего не известно с ранних этапов процесса проектирования, и оно становится частично известным, когда выбирается окончательная облачная платформа, поскольку физические детали реализации в любом случае недоступны. На первом

этапе обе подмодели могут быть смоделированы с помощью аналитических методов, таких как сети Петри или более специфичный язык, адаптированный к конкретной предметной области, для получения первого приближения оценки производительности и поддержки выбора дизайна. Впоследствии подмодели могут быть доработаны путем добавления деталей: об окончательной реализации приложения, используя полные знания о нем; о гипотетической целевой архитектуре для облачной инфраструктуры, чтобы экспериментировать с различными конфигурациями.

Следовательно, вся модель разрабатывается до тех пор, пока приложение не будет закодировано и запущено в имитационной облачной среде для окончательной оценки.

Определены два формализмы и их состав, включая все необходимые элементы для создания базовой модели. Формализмы описываются элементами и ребрами, каждое из которых обозначается набором свойств (учитывающих параметры, внутреннее состояние и переменные, необходимые для оценки производительности) и поведение (описывающих, как элементы и ребра взаимодействуют для определения общей семантики модели), описывающих формализм и которые могут быть использованы чтобы построить график, представляющий модель. Ребра связывают два элемента и ведут себя соответственно определенной семантике.

Вышеупомянутые формализмы называются, соответственно, GSPN (Обобщенные стохастические сети Петри) и новый DSL, названный CDL (Cloud(Sim) Description Language).

В итоге предложена модификация технологии стохастического моделирования облачной архитектуры, отличающаяся использованием мультиформального подхода для определения классов рабочей нагрузки, конфигурации облака и взаимосвязей между рабочей нагрузкой и облаком в сценариях мультиоблачных/гибридных облачных/пограничных вычислений, позволяющая автоматически генерировать стохастические имитаци-

онные модели с высоким уровнем согласованности с поведением облачной архитектуры.

В главе 4 представлены модели и инструменты облачного управления доступностью и производительностью инфокоммуникационной подсистемы.

Для решения проблем, связанных с тем, что традиционный метод имеет длительное время отклика на мониторинг перегрузки сети связи, а эффект обнаружения не идеален, предлагается метод мониторинга в реальном времени, основанный на облачных вычислениях для блокировки сети связи. Во-первых, устанавливается точка мониторинга сети связи, и приемник завершает процесс сбора коммуникационных данных. На основе собранных данных выполняется постоянный расчет трафика для определения наличия состояния блокировки в канале сети связи и определения точного местоположения точки блокировки. Таким образом, информация генерирует тревожное сообщение для получения результатов мониторинга. Экспериментально проанализированы время работы в режиме реального времени и точность метода мониторинга. Установлено, что метод мониторинга позволяет контролировать время задержки в пределах 0,2 с, а частота ошибок мониторинга является низкой.

Представлен алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений, отличающийся событийным мониторингом ситуаций блокировки системы и обеспечивающий расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки.

Также рассматривается проблема традиционного моделирования производительности, а также предлагается решение для анализа производительности в многоуровневом облаке. Доступность оценивается путем объединения набора важных параметров, чтобы приблизительное предсказание доступности было более точным в этой модели по сравнению с из-

вестными моделями доступности. Предлагаемое моделирование оценивается путем реализации модели в инструменте SHARPE. Результатом этой работы является разработка модели производительности, которая позволяет измерить любую облачную систему на основе ее доступности и проанализировать качество предоставляемых ею услуг.

По результатам имитационного моделирования предложенная модель является более точной по сравнению со стандартными моделями. Она предсказывает значение доступности с точностью приблизительно 99%, на 8.3% точнее стандартных.

Далее разработана структура программного прототипа системы распознавания блокировок и оптимизации доступности облачных сервисов. Элементы программной реализации прошли государственную регистрацию в Роспатенте.

Представлена структура программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов, отличающаяся интеграцией системы мониторинга и облачных сервисов и обеспечивающая реконфигурацию инфокоммуникационной системы в зависимости от параметров инфраструктуры и качества обслуживания.

1. ПРОБЛЕМЫ И ОСОБЕННОСТИ ЖИЗНЕННОГО ЦИКЛА И МОДЕЛИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ В ОБЛАЧНЫХ СРЕДАХ

1.1. Проблемы организации-разработчика программного обеспечения как результат реализации модели «Open software enterprise» в экосистеме

1.1.1. Оперирующая система и производитель программного обеспечения

Организации, производящие программное обеспечение (SPO), преследуют одну общую цель: обеспечить как можно более частое внедрение разрабатываемого программного обеспечения. Для достижения этой цели существуют различные бизнес-модели: от моделей с открытым исходным кодом, когда исходный код предоставляется бесплатно, до моделей обслуживания, когда исходный код выполняется на защищенном сервере. Поставщики услуг обнаружили, что могут быстрее достичь своих целей, создав вокруг продукта экосистему партнеров, таких как разработчики или реселлеры. Эти разработчики и реселлеры однако будут участвовать только в том случае, если откроется SPO: бизнес-процессы, которые традиционно являются закрытыми, необходимо разделить с участниками экосистемы, чтобы помочь им создавать ценность. Такая ценность создается за счет массовой разработки (с открытым исходным кодом), массовой настройки (с открытой конфигурацией) и многих других стратегий, которые включают в себя открытость бизнес-процессов. Традиционно монолитные и закрытые организации, которые открывают свои бизнес-процессы, называются открытыми предприятиями, занимающимися разработкой программного обеспечения (OSE).

Оперирующая система создает свою собственную сеть сотрудников,

называемую программной экосистемой. Программные экосистемы определяются как совокупность субъектов, функционирующих как единое целое и взаимодействующих на общем рынке программного обеспечения и услуг, а также отношений между ними [2.28]. Эти отношения часто поддерживаются общей технологической платформой или рынком и осуществляются посредством обмена информацией, ресурсами и артефактами. Существует несколько факторов, побуждающих к созданию экосистемы программного обеспечения, таких как способность более гибко реагировать на меняющиеся требования к различные специализированные решения, основанные на платформе, или более быстрое внедрение, чем у конкурентов, благодаря крупным сетям партнеров и разработчиков. В [2.25] отмечено, что выживание и эффективность организаций в экосистеме зависят друг от друга. Таким образом, работоспособность отдельных организаций зависит не только от них самих, но и от работоспособности всей сети фирм.

Оперирующая система определяется как организация, производящая программное обеспечение и предоставляющая доступ к своим процессам по крайней мере одному из типов участников, которые присутствуют в ее программной экосистеме. Оперирующие системы могут быть любого типа, от независимых поставщиков программного обеспечения до организаций с открытым исходным кодом.

Субъектом может быть любой участник программной экосистемы. В статье выделено четыре типа участников: разработчиков, реселлеров с добавленной стоимостью (VAR), сервисных партнеров и клиентов. У каждого из этих участников есть возможность повысить ценность SPO, добавляя исходный код, осуществляя продажи для организации или просто платя напрямую поставщику программного обеспечения.

Когда SPO намеревается стать OSE, он должен пересмотреть свой бизнес модель. Существует несколько различных методов оценки бизнес-моделей, таких как применение методов моделирования экосистемы про-

граммного обеспечения [2.8] или метод оценки [2.36], которые определяют бизнес-модель программного обеспечения по четырем направлениям: модель распространения, услуги и внедрение предоставляемые поставщиком программного обеспечения, применяемая им логика получения дохода и его продуктовая стратегия. Показано, что бизнес-модель играет ключевую роль в определении того, как SPO открывает свои процессы, и что различные дополнительные преимущества заинтересованные стороны, такие как сторонние разработчики и VAR, заинтересованы в открытии различных областей процессов.

Несколько инициатив [2.33] призывают использовать "открытое" программное обеспечение и стандарты вместо закрытых альтернатив. Этим общественным организациям трудно точно определить, насколько открытым является SPO, особенно потому, что в настоящее время не существует объективных инструментов. Это наносит ущерб индустрии программного обеспечения, поскольку обычно SPO несправедливо считаются открытыми или закрытыми, что приводит к ошибочным решениям о покупке.

Модель, представленная далее, направлена на установление того, насколько открытым или закрытым является SPO. Смысл существования модели заключается в трех аспектах.

Во-первых, несправедливые решения о снабжении принимаются на основе предвзятой оценки открытости, что не позволяет выиграть лучшему продукту или поставщику. Во-вторых, SPO не знают о различных существующих вариантах открытости, и они не знают, как они могут безопасно изменят свою бизнес-модель и откроют свой бизнес, чтобы соответствовать требованиям местных органов власти и стимулировать экосистему программного обеспечения. В-третьих, эта модель направлена на то, чтобы сместить акцент с “открытого исходного кода” и показать, что открытость - это не просто лицензия на открытое программное обеспечение.

1.1.2. Исследовательский подход

Основным результатом исследования является модель OSE. Модель OSE была разработана с применением исследовательского метода [2.24]. В нем есть набор из семи рекомендаций, которые помогают исследователю проводить, оценивать и презентовать научные исследования в области дизайна.

Семь рекомендаций касаются актуальности проблемы, дизайна как артефакта, строгости исследования, дизайна как процесса поиска, оценки дизайна, вклада в исследование и научной коммуникации. В рекомендациях обсуждаются проблемы исследования, проектирование как артефакт, строгость исследования и оценка дизайна.

Проблема исследования, которая была выявлена, заключается в том, что термин “открытость” часто используется в контексте рекламных роликов, обычно как синоним полезного, без фактического определения открытости в этом контексте. Принимаются необоснованные решения об открытии части объектов предприятия, а неподтвержденные решения о снабжении принимаются на основе предвзятого восприятия открытости в контексте SPO. Эти необоснованные решения могут привести к неправильным стратегическим решениям и могут потенциально это может сделать хорошие продукты менее привлекательными, основываясь на спекулятивном подходе требование открытости. Были сформулированы две подзадачи:

1. в настоящее время не существует модели, которая предоставляла бы руководителям обзор мер открытости, которые могут быть приняты SPO;

2. нет способа измерить, насколько открытой является организация.

Результатом этих проблем является то, что компаниям-разработчикам программного обеспечения приходится думать, каким образом им следует расширить свои процессы, чтобы использовать экосистемы программного обеспечения, и что компании-разработчики про-

граммного обеспечения преждевременно оцениваются как слишком закрытые. Оба результата пагубны для индустрии программного обеспечения. Разрабатываемый продукт – это модель OSE, которая решает две выявленные проблемы. Модель OSE может использоваться SPO в качестве справочной системы для определения того, какие варианты следует использовать, если SPO решит стать более дружественным к экосистеме. Кроме того, эта модель может быть использована закупающими организациями для определения того, какие из продуктов, включенных в короткий список, являются более “открытыми”, путем определения того, какой из вариантов открытости был выбран поставщиком этих продуктов.

Что касается тщательности исследования, то исследование было проведено с использованием двух циклов проектирования, в каждом из которых использовался свой метод оценки. В ходе первого цикла артефакт оценивался в ходе пяти интервью с экспертами в области экосистем программного обеспечения. Во втором цикле были проведены три тематических исследования, чтобы дополнительно установить, полезна ли модель и может ли она быть применена к примерам оперирующих систем. Оба цикла исследований проводились разными исследователями.

1.1.3. Оценочные интервью

Было организовано пять интервью с экспертами в области экосистем программного обеспечения. Этими экспертами были два менеджера по продуктам, технический директор и генеральный директор CEO компании-разработчика программного обеспечения и, наконец, менеджер экосистемы. Каждый из опрошенных имел как минимум 10-летний опыт работы в индустрии программного обеспечения. Интервью обычно длились около 90 минут и состояли из 10 открытых вопросов об открытости (табл. 1.1), оценки модели и окончательного определения конкретного случая. Открытые вопросы касались определения оперирующей системы, различных

способов восприятия открытости респондентом и того, почему выглядят некоторые компании быть более открытым, чем другие. Вопросы разделены на категории "актуальность проблемы", "осуществимость модели" и "оценка модели". Эти вопросы были сформулированы с использованием концептуальной карты и руководства по проектным исследованиям, представленного в [24], в которых четко указаны актуальность, осуществимость и оценка для разработки артефакта (в нашем случае модели OSE). Ответы на вопросы были записаны во время собеседований, обработаны и занесены в электронную таблицу.

Таблица 1.1

Вводные вопросы для экспертных оценок (до демонстрации модели)

Вопрос	**Категория**
Как вы воспринимаете открытость организации в контексте SPO?	Актуальность проблемы
Каковы, на ваш взгляд, основные причины создания SPO?	Актуальность проблемы
Считаете ли вы, что SPO иногда ошибочно считают открытыми или закрытыми?	Актуальность проблемы
Как вы считаете, можно ли создать модель, которая измеряла бы открытость SPO? Как бы это выглядело?	Осуществимость модели
Какие компании вы считаете открытыми? Какие из них закрылись? Что убедило вас в том, что они открыты или закрыты?	Осуществимость модели
Считаете ли вы, что открытость SPO зависит от конкретной предметной области? Может ли производитель банковского программного обеспечения вообще быть открытым или это выбор, связанный с бизнес-моделью?	Осуществимость модели
Считаете ли вы, что инициативы по обеспечению открытости действительно могут стимулировать открытость экономики?	Оценка модели
Считаете ли вы, что более открытое SPO стимулирует свою экосистему лучше, чем закрытая организация?	Оценка модели
Считаете ли вы, что у SPO достаточно информации, чтобы открыть свой бизнес? Как вы считаете, необходим ли обзор с вариантами открытости?	Оценка модели
Считаете ли вы, что чрезмерная открытость может угрожать бизнес-модели SPO?	Оценка модели

После обсуждения 10 вопросов об открытости модели, оценка началась с представления графического представления модели в табл. 1.1 и подробного обсуждения каждого элемента модели. Во время собеседований было записано множество возможных дополнений. Всего было предложено 15 изменений, из которых 6 были реализованы в модели. Примером варианта открытости, который не вошел в модель, является “политизация платформы”, поскольку, хотя он и имеет отношение к "открытой" бизнес-модели, он конкретно не касается открытости разработки программного обеспечения или управления им [2.6]. В последних двух интервью модель OSE была представлена следующим образом: опрошенные сочли модель OSE завершенной, и никаких дальнейших изменений предложено не было.

1.1.4. Проектирование тематических исследований

Три тематических исследования были отобраны путем просмотра списка из примерно 200 контактов с университетами. Для проведения оценки были привлечены четыре компании [2.27]. Третье тематическое исследование Eclipse Foundation было добавлено, чтобы дополнить кейс с закрытым исходным кодом и кейс с исходным кодом сообщества кейсом, который полностью предоставляет свой исходный код в виде открытого исходного кода.

Тематические исследования проводились в соответствии с рекомендациями [2.28, 2.51]. В соответствии с этими рекомендациями было проведено тематическое исследование, был создан отчет и база данных тематических исследований. Следует отметить, что тематическое исследование не включало никаких интервью, а проводилось на основе изучения документов. Результаты тематического исследования были представлены участникам, чтобы выяснить, согласны ли они с тем, что варианты открытости правильно отражают реальность как в самих вариантах открытости, так и в

общем обзоре открытости.

Тематические исследования состояли из трех этапов. Сначала было подготовлено общее описание кейса, в котором описывалась деятельность организации, владелец, структура, программный продукт(ы) и некоторые сведения о его недавней истории. Второй шаг состоял в проверке вариантов открытости модели OSE с участием респондентов, чтобы определить, какие из вариантов считаются открытыми. Третий шаг состоял в обсуждении того, как и почему некоторые варианты открытости на самом деле были открытыми или закрытыми, и как эти варианты открытости соотносились с бизнес-моделью организации.

Эта модель была применена к тематическим исследованиям, в ходе которых для каждого варианта открытости был задан вопрос о том, как организация оценивает себя с точки зрения открытости. Для каждого варианта, в котором респондент считал организацию открытой или закрытой, его просили привести мотивирующие причины, по которым он называл этот вариант открытым или закрытым, чтобы в ходе обсуждения можно было составить качественную оценку.

1.1.5. Альянс открытого дизайна (ODA)

Альянс открытого дизайна (ODA) - это организация, которая стремится к внедрению открытых стандартов в отношении форматов CAD, и, в частности, формата файлов dwg. В настоящее время в альянсе насчитывается 2000 членов.

ODA разрабатывает платформу ODA, которая включает библиотеки ODA. Платформы ODA могут быть включены в любую САПР, позволяющие чтение и запись в файлы в формате DWG. У ODA есть четыре различных типа участников: ассоциированный, коммерческий, поддерживающий, учредительный и образовательный. Каждый из этих типов участников обладает различными правами, привилегиями и стоимостью, вплоть до полного доступа к исходному коду. Членство предоставляется после подписа-

ния соглашения о членстве.

В 1998 году Visio основала альянс OpenDWG, или, как его еще называют, ODA, и открыла свое членство для других. Еще одним интересным событием является то, что ODA начиналась как организация, которая сосредоточилась на том, чтобы предоставить доступ к стандарту dwg поставщикам программного обеспечения. Однако за время своего существования центр расширился и включил в библиотеки несколько других дополняющих и конкурирующих стандартов. По мере увеличения количества и разнообразия форматов, включенных в библиотеки, структура программного обеспечения менялась платформа ODA также претерпела изменения. В настоящее время платформа ODA состоит из средств просмотра, считывания, API и документации, которые позволяют разработчикам получать доступ к форматам хранения, связанным с САПР, и изменять их. Вместе с этими разработками произошло изменение названия: библиотеки ODA стали платформой ODA.

Многие из существующих утилит для платформы ODA были поставлены или приобретены у третьих сторон, что, таким образом, обогатило платформу ODA. Поскольку эти утилиты все чаще включались в платформу, в ODA поняли, что платформа должна поощрять такое повторное использование. Недавно началась программа сторонних поставщиков, которая позволяет поддерживающим участникам и учредителям вносить свои собственные компоненты в платформу. В настоящее время платформа ODA содержит компоненты для рынка геоинформации, рынка механического проектирования и рынка архитектуры ODA изучает другие области, представляющие интерес для ее членов.

Открытость программной экосистемы ODA представляет угрозу для бизнес-модели AutoDesk в формате dwg, что привело к ряду судебных исков со стороны AutoDesk в отношении ODA. Эти судебные иски являются относительно успешной стратегией в конкурентной борьбе, учитывая, что

они, как правило, обходятся ODA в большие деньги, которые обычно используются для найма разработчиков платформ. Программную экосистему ODA можно рассматривать как находящуюся в прямой конкуренции с программной экосистемой AutoDesk, поскольку AutoDesk предоставляет API-интерфейсы для доступа к формату dwg своим собственным участникам.

Успех ODA определяется количеством его участников. Участники являются наиболее ценным активом ODA, поскольку благодаря своим членским взносам они позволяют ODA привлекать больше разработчиков.

ODA активно объединяет своих членов для создания более активного сообщества, поскольку члены организации разбросаны по всему миру.

1.1.6. Фонд Eclipse Foundation

Eclipse Foundation [2.20] – некоммерческая корпорация, поддерживаемая членами Eclipse Projects, которая предоставляет ряд продуктов для поддержки разработчиков в процессе разработки с помощью компиляторов, сред разработки, инструментов визуализации и т.д. Фонд предоставляет сообществу Eclipse управленческие услуги, такие как управление ИТ-инфраструктурой, проведение комплексной проверки интеллектуальной собственности и наставничество в проектах с открытым исходным кодом. EF можно рассматривать как одно из ведущих сообществ с открытым исходным кодом.

Индустрия щедро поддерживает его. IBM, которая в то время разрабатывала первые версии Eclipse, выпустила ее с открытым исходным кодом в 2001 году. В том же году консорциум Eclipse был создан девятью компаниями, которые сочли, что существующих инструментов разработки недостаточно и что их не следует разрабатывать одной организации в одиночку.

Подобно Альянсу открытого дизайна, история EF является отправной точкой для создания успешной экосистемы. Когда платформа Eclipse

была выпущена в открытый доступ, члены консорциума могли свободно разрабатывать свои собственные решения для Eclipse, ориентированные на конкретную предметную область, что немедленно создало оживленное сообщество вокруг платформы. Всего через девять лет после создания первого консорциума, EF и разработчики, окружающие экосистему, состоят из десятков тысяч разработчиков, четырнадцати стратегических участников и нескольких других уровней участников.

Eclipse и степень его открытости

EF придерживается большинства параметров модели OSE. Начнем с того, что в отношении управления EF полностью раскрыл свою стратегию управления. Члены EF могут, в зависимости от их уровня членства и общего влияния в экосистеме, изменять Политику EF в соответствии с уставом EF и соглашением о членстве.

Кроме того, EF создала расширенную модель членства, учитывающую различные типы участников (пользователи, разработчики, VARs). Кроме того, у EF есть четкая стратегия в области интеллектуальной собственности, которая полностью описана на его сайте. EF также координирует вклады в другие экосистемы, такие как экосистема Apache, по решению участников.

EF не придерживается четкой стратегии приобретения, хотя несколько проектов с открытым исходным кодом присоединились к EF. Кроме того, не было разработано четкой стратегии борьбы с конкуренцией со стороны EF. Для EF существует четко определенная стратегия управления знаниями EF, в результате чего появилось несколько порталов, форумов и документов. Одна часть - это стандарты процесса разработки, которые полностью описаны на веб-сайте EF и могут быть приняты как проекты EF, так и внешние проекты. EF может выступать посредником, когда участникам требуется арбитраж в конфликте внутри проекта или между

проектами. Наконец, EF явно создала группы пользователей с соответствующим типом членства, экосистема была четко обозначена, и на вебсайте EF существует большой каталог партнеров.

Что касается исследований и разработок EF, то EF активно работает над продвижением своих технологических и стратегических планов. Кроме того, EF открыто рассказывает о том, как разрабатывается ее программное обеспечение. EF еще больше поощряет открытые стандарты, открывая различные форматы файлов и повторно используя открытые стандарты там, где это возможно.

Кроме того, исходный код EFs доступен на портале CVS [2.16]. EF делится своим IP-адресом, включая любые нововведения, которые не соответствуют стратегии EF. Кроме того, в EF четко указано, какой код можно, а какой нельзя повторно использовать в проектах. EF не сертифицирует ни один из доступных сторонних плагинов, хотя компонент сертификации для Eclipse [2.42] доступен. Проект Eclipse создал легко расширяемую архитектуру, которая пытается способствовать повсеместному повторному использованию. Результаты модульного тестирования проекта Eclipse публикуются, хотя и неофициально. Кроме того, открыто хранилище ошибок проекта Eclipse [2.9]. Наконец, Eclipse позволяет своим участникам помогать в разработке, проводится обучение разработчиков и создаются многоразовые лицензии на программное обеспечение.

Что касается управления программными продуктами EF, то EF разделяет жизненный цикл продукта для различных продуктов (или проектов). Кроме того, EF открыто разделяет стратегию и видение платформы EF.

EF передает разработку требований на аутсорсинг участникам, делится и корректирует дорожные карты продуктов на основе мнений участников, а процесс разработки требований в рамках проекта Eclipse полностью прозрачен. Наконец, планирование выпуска осуществляется четко, и

кандидаты на выпуск обычно публикуются заранее, чтобы удовлетворить потребности пользователей, будущее которых зависит от продукта Eclipse.

Что касается маркетинга и продаж EF, то EF делится своим видением рынка и разрабатывает инновационные бизнес-модели, в том смысле, что подключаемые модули платформы могут быть разных форм и размеров, с различными бизнес-моделями, при условии соблюдения соглашения о членстве в EF. В проекте Eclipse нет партнерской программы по продажам, но EF предоставляет всю маркетинговую информацию, которую может найти на платформе Eclipse, а также потенциальным участникам, партнерам и разработчикам. EF не сертифицирует партнеров. Однако EF привлекает партнеров к рекламной деятельности. Кроме того, EF создала несколько рынков компонентов, которые могут использоваться сторонними разработчиками для продвижения своих плагинов и продуктов.

Что касается консультационных услуг и поддержки EF, то у EF нет конкретных проектов по внедрению, в которых проект Eclipse должен быть развернут, сконфигурирован и внедрен у заказчика. Таким образом, EF не имеет стратегии управления предоставлением услуг и не передает какие-либо подобные проекты на аутсорсинг. EF не располагает специальной базой данных о проблемах внедрения и не делится знаниями о процессе реализации проекта или показателях качества.

И, наконец, EF не проводит обучение консультантов.

Согласование бизнес-модели EF

Рассматривая бизнес-модель EF, мы видим, что разработчики вносят существенный вклад в развитие организации благодаря своему ежедневному вкладу в EF. Кроме того, значительная часть доходов поступает от заказчиков и конечных пользователей (членов предприятия) из-за своей зависимости от платформы Eclipse. Кроме того, вторая часть дохода поступает от поставщиков услуг, которые используют платформу для созда-

ния новых преимуществ для своих клиентов.

Последнее замечание в отношении платформы Eclipse - это способ ее распространения: Eclipse можно загрузить с веб-сайта EF и довольно легко развернуть. Принимая во внимание дизайн бизнес-модели EF, легко объяснить степень открытости EF. Открытость в сферах управления, исследований и разработок и SPM необходима для удобства разработчиков и VAR. Открытость в области маркетинга и продаж требуется VARs. Наконец, тот факт, что в области CSS практически нет открытости, заключается в том, что для развертывания и внедрения Eclipse в организации практически не требуются услуги.

Случай с EF отличается от предыдущих тем, что основной деятельностью EF является управление экосистемой Eclipse. Команда Eclipse смоделирована в структуре управления на рис. 1.1.

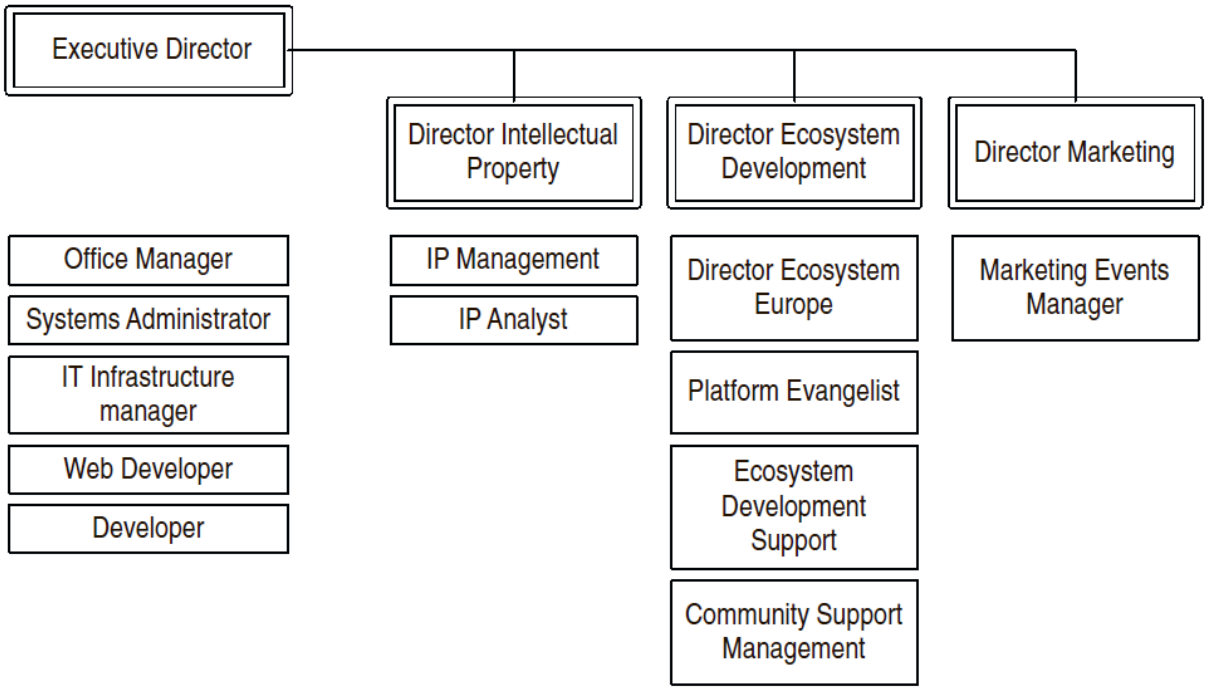


Рис. 1.1. Структура Eclipse Foundation Management

Основными функциями, которые можно выделить, являются управление интеллектуальной собственностью, развитие экосистемы и маркетинг, что представляет собой совершенно иную структуру, чем, например,

в случае с ODA, где 30 из 34 человек занимаются разработкой программного обеспечения и обеспечением контроля качества.

Экосистема Eclipse успешна по ряду причин. Начнем с того, что EF удалось мобилизовать своих пользователей, разработчики, ориентированные на конкретную предметную область, и "вертикали", то есть третьи стороны, которые разрабатывают плагины для конкретной технологии разработки, такие как среда Eclipse для PHP (PDT) или Ruby on Rails (Aptana).

Вторая причина успеха заключается в том, что Eclipse - это платформа для разработчиков, созданная самими разработчиками. Разработчикам и исследователям в области разработки программного обеспечения, как и ожидалось, требуется больше возможностей от такой платформы, как Eclipse, и эти разработчики хорошо подходят для разработки расширений и функций. Это отличается от обычного случая, когда конечные пользователи редко знают, что они используют компонент, когда они это делают, они получают финансовую поддержку от ODA. Третьей причиной успеха EF является мощная поддержка со стороны промышленности. Члены-основатели EF являются одними из самых уважаемых SPO в мире.

1.1.7. Компания GX Software

GX Software - голландская компания, насчитывающая около 120 сотрудников, работающих как в Нидерландах, так и в Соединенных Штатах. Программный продукт GX WebManager - это CMS, которая используется многими крупными организациями для создания и поддержки сложных веб-сайтов и веб-приложений. Архитектура GX Software WM была изучена и смоделирована в [2.44].

Бизнес-модель GX Software значительно изменилась за последнее десятилетие, как можно видеть на рис. 1.2. Традиционно GX Software состояла из одной основной команды разработчиков и одного основного команда сервисных инженеров, которые обеспечивали развертывание оборудования в соответствии с требованиями заказчика. Эти две группы интен-

сивно сотрудничали, что привело к созданию сложных структур управления: иногда функции требовались для всех клиентов, иногда для одного конкретного клиента.

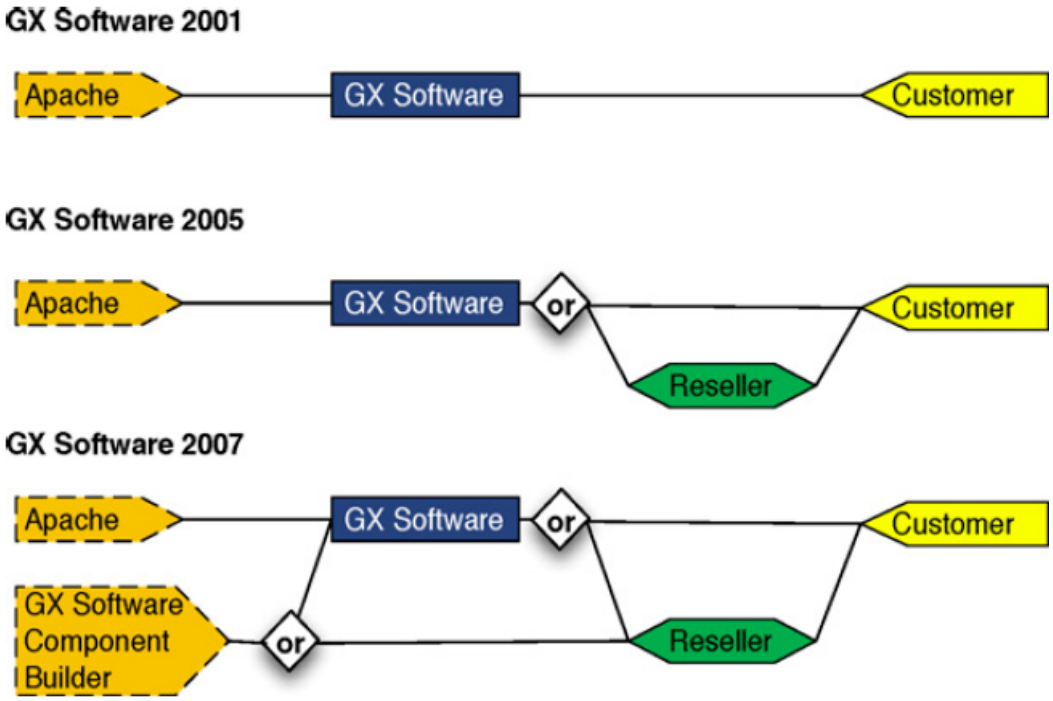


Рис. 1.2. Эволюция бизнес-модели для компании GX Software

В настоящее время GX Software уделяет особое внимание издательствам, средствам массовой информации и финансам (банкам и страхованию).

Однако за прошедшие годы у них сформировалась разнообразная клиентская база: разные вертикальные отрасли требуют разных индивидуальных решений. Кроме того, требуются решения, специфичные для конкретной предметной области, которых так много что одной компании будет сложно разрабатывать и поддерживать каждый подключаемый модуль, API и интерфейс. Кроме того, партнеры проявляли все больший интерес к более совершенным API и наборам для разработки, чтобы в дальнейшем создавать расширения поверх существующей платформы.

Поэтому, начиная с 2007 года, платформа GX Software была пере-

строена, что позволило партнерам разрабатывать свои пользовательские компоненты, а GX Software представила онлайн-платформу обмена компонентами, где компоненты могут публиковаться под различными лицензиями (открытой, закрытой бесплатной и закрытой коммерческой). В настоящее время этот маркетплейс предлагает более 1000 компонентов для программного обеспечения GX Software.

AppStore. Одной из наиболее успешных мер, предпринятых компанией GX Software, является создание инструмента сертификации компонентов, который автоматически проверяет и сертифицирует подключаемые модули сторонних производителей. Инструмент сертификации позволяет GX Software быстро определить, не нарушает ли компонент какие-либо стандарты взаимодействия и соответствует ли он требованиям критериев качества программного обеспечения GX.

Программное обеспечение GX и степень его открытости

GX Software традиционно является сервисной компанией, в основе услуг которой лежит собственный программный продукт, в частности что она всегда предоставляла большое количество услуг, помимо продажи лицензий. Кроме того, все основатели, ученые, имеют открытый взгляд на бизнес программного обеспечения и готовы экспериментировать с новыми бизнес-моделями. Поэтому неудивительно, что GX Software уже на ранней стадии приступила к разработке хранилища компонентов. Кроме того, GX Software активно участвовала в нескольких проектах Apache с открытым исходным кодом, что свидетельствует о том, что GX Software готова приносить пользу сообществу. Однако бизнес-модель всегда заключалась в продаже как лицензий, так и услуг.

Что касается управления GX Software, то компания не придерживается стратегии открытого управления или интеллектуальной собственности, не делится своей стратегией приобретения и не делится с партнерами

своей политикой в отношении борьбы с конкуренцией. GX Software разработала модель простого партнерства. Программное обеспечение GX не сообщает об этом сторонним разработчикам как бороться с конкуренцией и стоит ли взаимодействовать с другими экосистемами. GX Software четко сформулировала свою экосистему и имеет четкую стратегию управления знаниями в отношении обмена знаниями о разработке и внедрении с партнерами.

Если быть более точным, GX Software создала различные веб-порталы, такие как GX Software AppStore.com портал для обмена и продажи компонентов и веб-сайт GX Software Developer, на котором разработчики могут делиться знаниями. Кроме того, у GXSoftware есть каталог партнеров, в котором перечислены партнеры по внедрению и разработке программного продукта GX Software.

Что касается исследований и разработок программного обеспечения GX, исходный код компонентов-примеров предоставляется через WCMExchange, а разработчики продуктов ведут блоги на конкретные темы, обычно с исходный код прилагается. Основной программный продукт, однако, имеет закрытый исходный код. Дорожная карта и новые инновации (разработка нового продукта) доступны избранной группе отраслевых экспертов, сертифицированных партнеров и клиентов. Также доступны новые инновации, которые не являются существенными для продукта. Примерами этого являются компоненты, которые разработчики программного обеспечения GX размещают на WCMExchange. Группа экспертов обеспечивает обратную связь и получает доступ к возможным версиям программного обеспечения. GX Software делится в форме научных статей, блога по разработке и руководств тем, как они разрабатывают программное обеспечение. GX Software также делится своим технологическим и исследовательским видением в презентациях и блогах. У GX Software есть внутренняя политика в отношении повторного использования исходного кода, по-

скольку она повторно использует несколько компонентов с открытым исходным кодом и делится ими с сертифицированными партнерами и заказчиками. GX Software разработала пакет SDK для разработчиков, который позволяет им создавать новые приложения, и GX Software сертифицирует эти сторонние компоненты с помощью автоматизированного инструмента. Инструмент сертификации был хорошо принят (и используется) этими сторонними разработчиками. Программное обеспечение GX также использует архитектуру, позволяющую повторное использование, хотя компоненты в основном используются в основном продукте GX Software. В настоящее время процесс тестирования полностью закрыт.

Хранилище ошибок открыто для сторонних разработчиков. GX Software публикует версии-кандидаты и тестирует их на пилотных заказчиках и партнерах. Кроме того, GX Software в настоящее время занимается совместной разработкой. Компания проводит всевозможные тренинги, такие как обучение пользователей, разработчиков и внедрение. Наконец, GXSoftware применяет несколько открытых стандартов в своих продуктах.

Что касается управления программным обеспечением, то организация делится своим видением платформы в блогах и презентациях.

Кроме того, для партнеров и заказчиков регулярно публикуются дорожные карты продуктов и планы их выпуска. Они основаны на видении компании, тенденциях рынка и обобщении мнений клиентов и пользователей. Однако дорожная карта не адаптирована к конкретным пожеланиям отдельных заказчиков. Процесс управления требованиями частично определяется требованиями партнеров и заказчиков, которые предоставляются через портал. GX Software не передает процесс разработки требований на аутсорсинг партнерам.

Маркетинг и продажи GX Software в GX Software в основном направлены на стимулирование партнеров к продаже программного продукта GX Software, для чего была создана партнерская сеть.

Проекты, ориентированные на конкретные домены, распространяются среди конкретных партнеров, специализирующихся в этой области. GX Software также создала новые бизнес-модели для своих партнеров, создав магазин компонентов.

Что касается консалтинга и технической поддержки ПО для Xsoftware, компания GX Software быстро добилась прогресса в оказании поддержки своим партнерам.

Начнем с того, что GX Software распределяет поступающие проекты между своими партнерами. Во-вторых, что касается знаний о внедрении, всем партнерам доступна база данных заявок, в которой обсуждаются и решаются конкретные вопросы внедрения. В-третьих, GX Software разработала несколько мер по повышению качества, которые неофициально обсуждаются с каждым партнером. Кроме того, GX Software проводит тренинги для консультантов и разработчиков третьих сторон. Наконец, у GX Software есть отдел профессиональных услуг (PS), который по запросу оказывает поддержку партнерам в процессе внедрения.

Согласованная бизнес-модель GX Software

Основная деятельность GX Software заключается в разработке программного обеспечения и оказании помощи VAR и сервисным партнерам в обеспечении успешных проектов внедрения для клиентов. Продукт GX Software можно загрузить как общедоступную версию, но чаще всего он распространяется через партнеров и реселлеров, что является основным источником дохода. В настоящее время GX Software не зависит от внешних разработчиков, которые предоставляют код, поэтому в отношении управления все относительно закрыто. Что касается исследований и разработок, GX Software является все еще открывает для себя множество возможностей открытости, но поскольку GX-экосистема программного обеспечения относительно молода, она еще не воспользовалась возможностью

полностью раскрыть сферу исследований и разработок и SPM. В областях слияний и поглощений и CSS программное обеспечение GX открыто во многих отношениях, как и предсказывает модель OSE.

Область применения подходит для реализации стратегии экосистемы программного обеспечения [2.7]: есть рынок, на котором программное обеспечение GX требует как вертикальных решений (правительства, отрасли и т.д.), так и горизонтальных (плагины, средства просмотра, контент функции импорта и экспорта и т.д.), которые не могут быть созданы одной организацией эффективным образом. Однако производительность сторонних разработчиков по-прежнему остается проблемой в экосистеме программного обеспечения GX. Программное обеспечение GX страдает от проблемы начальной загрузки [2.28], поскольку в экосистеме недостаточно участников, чтобы быстро набрать критическую массу. Однако было обнаружено быстрое увеличение количества компонентов в хранилище компонентов, что позволяет полагать, что стратегия экосистемы программного обеспечения GX работает.

1.1.8. Программные экосистемы и открытость

Программным экосистемам и открытости уделяется много внимания. Тема программных экосистем находится на стыке множества различных тем, связанных с взаимодействием между программными продуктами и компаниями. Такие взаимодействия обнаруживаются, например, когда продукты повторно используют другие продукты и требуют специальных лицензий [2.1]. Взаимодействие также обнаруживается при рассмотрении повторного использования компонентов в открытых сообществах разработчиков, и в этом контексте также использовался подход экосистемы программного обеспечения [2.17, 2.31]. Конкретный подход в области экосистемы программного обеспечения рассматривает роль ключевого элемента, то есть игрока, который имеет фундаментальное значение для экосистемы.

В работах [2.7, 2.15, 2.25] эта роль конкретно рассматривается. Разработкой в области программных экосистем является моделирование экосистем с целью понимания динамики внутри такой экосистемы [2.3]. Интересно исследование в области партнерских соглашений - как они влияют на открытость организации и как эти партнерские соглашения влияют на бизнес-модель [2.46]. Для углубленного изучения литературы по экосистемам полезно систематическое картографическое исследование [2.5].

Что касается открытости, то разные работы играют определенную роль в продвижении открытого программного обеспечения и открытых организаций, преследуя разные цели.

Под концепцией закрытости понимаем [2.45] и еще раз подчеркиваем разницу между открытостью и закрытостью и определяем модель, которая оценивает открытость в непрерывном масштабе. В основе этой модели лежат такие понятия, как доступность, прозрачность, взаимовыгодность и лицензирование, которые необходимо оценить, чтобы определить, является ли тот или иной вариант открытым или нет. Модель применяется в контексте компании, занимающейся разработкой программного обеспечения. Она отличается тем, что в ней делается попытка более подробно раскрыть характеристики каждого варианта открытости и учитывается специфика предметной области продукта. Некоторые вопросы, заданные в [2.32], непосредственно повлияли на некоторые варианты открытости в модели, такие как, например, “сертификация партнеров” и “делиться информацией о рынке”. Еще одна оценка открытости, которая учтена, - это оценка архитектур платформ мобильных операционных систем, в ходе которой установлено, обнаружили, что бизнес-модель оказывает сильное влияние на параметры открытости в архитектуре программного обеспечения [2.2]. Что касается продвижения открытого исходного кода, то большая работа ведется в области открытых сообществ [2.38] и бизнес-моделей с открытым исходным кодом [2.6, 2.39, 2.41]. Также актуальна работа над

открытостью интерфейсов (в программных экосистемах), таких как работа [2.10]. Открытость в контексте программных экосистем не была четко обозначена, и в этой статье предпринята попытка заполнить эту нишу. Однако имеется некоторая литература об открытых стандартах и их роли в экосистемах программного обеспечения [2.4, 2.49].

1.2. Особенности мультиформального подхода к облачной автоматической генерации кода

1.2.1. Мультиформализмы

Облачные архитектуры создают множество проблем, начиная от некачественного оборудования и заканчивая длительным временем ожидания и проблемами с балансировкой ресурсов. Большинство из этих проблем можно устранить, используя надлежащие методы оценки производительности, в зависимости от точности модели, абстрагирующей реальную систему. Предпочтение одного формализма другому при описании рассматриваемой инфраструктуры играет решающую роль в достижении этой цели. В этом смысле мультиформализм зарекомендовал себя как мощный подход к моделированию, описывающий каждый компонент системы в соответствии с наиболее подходящим представлением. Представлен новый метод моделирования, ориентированный на прогнозирование производительности облачных архитектур, подходящий для объединения преимуществ высокоуровневых абстракций моделирования и детализации специализированного симулятора. Для описания рабочей нагрузки и поведения пользователей и приложений используются обобщенные стохастические сети Петри, а для облачной части используется Cloudsim, хорошо известный симулятор облачной инфраструктуры. Для демонстрации эффективности предложенного подхода приведен пример упрощенного приложения для пограничных вычислений.

Аналитические решения дают эффективные результаты в тех случаях, когда рассмотренная модель правильно абстрагирует реальную систему. В большинстве случаев этого достаточно для приближения к процессу моделирования на ранних этапах анализа. Поскольку сложные архитектуры демонстрируют специфическое поведение, более точные результаты можно получить, полагаясь на имитатор, специально разработанный для конкретной подсистемы [3.7].

Такой подход помогает анализу, позволяя разработчикам моделей постепенно переходить от полностью поведенческой аналитической модели к имитационной модели, которая точно имитирует реальную архитектуру и механизмы системы. Однако опыт, необходимый для освоения аналитических моделей, и специализированные модели, основанные на имитационном моделировании, как правило, различаются и часто дополняют друг друга [3.45].

Специалисты в первой области, как правило, знакомые с формализмами моделирования, такими как сети Петри или сети массового обслуживания, и специалисты во второй области, которые используются для построения имитационных моделей путем разработки программ, основанных на таких инструментах, как CloudSim или NS3 (https://www.nsnam.org/), имеют разный опыт работы. Частичное повторное использование моделей может быть достигнуто за счет использования подходов с несколькими решениями, таких как [3.17]. Эта методология требует надлежащей поддержки для сохранения, по крайней мере частично, описания модели. В то же время необходимо изменить подход к анализу, например с помощью преобразования между форматами описания, между доменами или автоматической генерации кода. Сокрытие последнего с помощью языков, специфичных для домена (DSL) [3.32, 3.38, 3.46], особенно если оно представлено графическими обозначениями, может облегчить переход. Разработчику моделей предлагается описать на высоком уровне, возможно, с

помощью графических метафор, параметры и конфигурацию, необходимые для создания сценария моделирования для конкретного инструмента. Мультиформальний подход [3.35] добавляет возможность комбинирования различных формализмов моделирования в одной и той же модели, что еще больше снижает уровень обучения для разработчиков моделей, имеющих опыт работы в области аналитического моделирования.

Представлен подход к поддержке моделирования облачных архитектур с помощью автоматической генерации кода на основе мультиформальной модели. Решение использует сети Петри для определения рабочих нагрузок модели, как это часто делается на предварительных этапах моделирования производительности проекта, и DSL для описания облачной архитектуры, указывая, что необходимо для создания стохастической модели на основе CloudSim. Основным вкладом является введение технологии стохастического моделирования, позволяющей:

1. представлять на высоком уровне сценарий, основанный на облачных вычислениях, с использованием абстрактных примитивов моделирования, подходящих для неспециалистов;

2. автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры, используя CloudSim.

1.2.2. Обзор проблем и подходов к мультиформальному моделированию

В этом разделе дается общее представление о проблемах и подходах, связанных с мультиформальным моделированием и производительностью облачных вычислений, архитектурами DSL и симуляторов облачных вычислений.

Моделирование производительности

В [3.42] авторы представляют концепцию моделирования произво-

дительности как специфический подход к прогнозированию производительности системы. Преимущества этого метода заключаются во многих аспектах, в частности, в недорогих прогнозах производительности системы и более глубоком понимании тонкостей системы как таковой. Первый момент довольно очевиден, поскольку с большинством проблем приходится сталкиваться заранее (например, правильно подобрать размеры и закупить оборудование). Второй аспект вытекает из количественных знаний, полученных в ходе вышеупомянутых прогнозов. Подходы к моделированию производительности различны и могут быть классифицированы как:

1) аналитические (т.е. с использованием математической модели),

2) имитационные [3.23],

3) измерительные (т.е. приложение выполняется с учетом ряда различных настроек),

4) сравнительный анализ.

Хотя прогнозирующее моделирование программных систем представляет интересные аспекты (см. [3.13], где авторы рассматривают набор алгоритмов, преобразующих расширенные диаграммы UML в инструмент Armani, и [3.18], в котором обсуждается PANORAMA, метод моделирования и прогнозирования производительности сложных научных рабочих процессов во время выполнения), особый интерес представляет недавно компания была посвящена облачным вычислениям, то есть определенному типу архитектуры, предназначенной для предоставления размещенных сервисов через Интернет и развертывания высокой степени виртуализации и вычислений, работающих распределенным образом.

Мультиформальное моделирование

Мультиформальное моделирование - это хорошо известный метод абстрагирования, используемый для моделирования частей системы с помощью набора разнородных формализмов. Этот подход особенно интере-

сен, поскольку он обеспечивает свободу выбора наилучшего языка формального описания для каждого компонента системы. Оценка конкретных показателей модели эффективности, полученных путем одновременной интеграции различных методов оценки, обозначается термином "мультирешение". Однако усилия, затрачиваемые на объединение различных формализмов, должны быть должным образом скоординированы, чтобы сохранить общую согласованность модели (обсуждение см. в [3.41]). Кроме того, различные используемые формализмы могут быть статическими или динамическими в зависимости от типа рассматриваемого решателя. Типичные инструменты мультиформализма, использующие фиксированный набор формализмов, можно найти в DEDS [3.6], SMART [3.11, 3.12] и SHARPE [3.39, 3.44]. С другой стороны, динамические и модульные инструменты обеспечивают большую гибкость при разработке моделей за счет внедрения новых решателей. Инструменты, относящиеся к этой категории, исторически определены как AToM3 [3.33] и Mobius [3.14, 3.17]. Интерес к инструментам открытого мультиформализма проявился в разработке OsMoSys [3.47] и SIMTHESys [3.2, 3.5, 3.26]. В Mobius и OsMoSys мультирешение рассматривается по-разному. В первом случае решатель получается в виде оптимизированной исполняемой модели, основанной на описании, предоставленном пользователем.

Последний решает модели путем (полуавтоматического) генерирования бизнес-процесса, выполняемого его механизмом документооборота, описывающего решение в терминах активаций внешних решателей. С этой точки зрения можно сказать, что выполняется модель Мебиуса и организуется модель OsMoSys. Новизна SIMTHESys состоит в следующих аспектах:

1) он отделяет компонент, определяющий формализм и реализующий для него решатель, от компонента, решающего модель;

2) он обеспечивает открытую архитектуру, в которой потенциально

могут использоваться любые виды формализмов (т.е. не только стохастический) может быть добавлен при условии создания надлежащего механизма решения;

3) он обеспечивает высокую степень гибкости, поскольку новые формализмы (или расширение существующих) могут быть легко добавлены в фреймворк путем создания нового формализма;

4) он сокращает объем кода, необходимого для обозначения новый формализм по сравнению с другими инструментами (например, SMART, DEDS tool box, Mobius и OsMoSys) и, наконец, v) он четко отделяет синтаксические компоненты формализма от семантических.

Сравнение наиболее важных функций инструментов приведено в табл. 1.2.

Таблица 1.2

Сравнение инструментов мультиформального моделирования

Свойство	**DEDS**	**SMART**	**SHARPE**	**AToM3**	**Mobius**	**OsMoSys**	**SIM-THESys**
Программируемость	-	+	-	-	+	-	-
Расширяемость	-	-	-	+	-	+	+
Метамоделирование	-	-	-	+	-	+	+
Генерация решателя	-	-	-	-	+	-	+
Мультирешение	-	-	+	-	+	+	+
Оптимизированные решатели	+	+	-	-	+	-	-
Поддержка DSL	-	-	-	-	-	-	+
Поддержка развития формализма	-	-	-	+	-	+	+

1.2.3. Системы имитационного моделирования для архитектур облачных вычислений

В литературе приводится множество примеров инструментов, моделирующих процессы облачных вычислений, наиболее известными из которых являются GridSim, CloudSim, SIMGREEN, SCORE и GreenCloud. Ниже кратко описаны и сравнены архитектуры вышеупомянутых инструментов.

Универсальный фреймворк для моделирования GridSim [3.8]. Он основан на библиотеке SimJava [3.25]. Он позволяет моделировать системы обработки больших объемов данных, такие как сетки или облако. Конечные пользователи, виртуальные машины и запущенные приложения могут быть абстрагированы в облако. GridSim развертывает агенты, представляющие уровень абстракции, позволяющий выполнять пользовательский код на данном хосте. Прикладной уровень моделируется в виде задач, обменивающихся данными с помощью MPI-тегов. Начиная с версии v3.12, был добавлен дополнительный модуль для учета энергопотребления.

SIMGREEN It - это инструмент для измерения расхода энергии на основе загрузки процессора. Динамическое масштабирование частоты напряжения и изменение частоты процессора - это задачи, реализуемые вместе с возможностями включения и выключения выбранных хостов. Дополнительная возможность измерения энергопотребления основана на энергопотреблении сети и дисков, принадлежащих виртуальным машинам.

Планировщики основаны на простой циклической политике.

CloudSim

Это симулятор, который позволяет создавать тестовые стенды на языке Java, используя

1) концепцию рабочей нагрузки выполняемых задач,

2) вычислительную мощность виртуальных машин (более подробную информацию см. в [3.9]).

Реализованы различные политики распределения виртуальных ма-

шин и миграции. Кроме того, эта среда моделирования позволяет оценивать различные виды цен на услуги. Функциональные возможности CloudSim были выбраны для поддержки моделирования:

1) крупномасштабных центров обработки данных облачных вычислений и объединенных облаков;

2) виртуализированных серверных узлов и различных видов политик для предоставления ресурсов хоста виртуальным машинам;

3) контейнеров;

4) вычислительных ресурсов, учитывающих энергопотребление;

5) элементов динамического моделирования.

Green Cloud [3.31] специально моделирует энергоэффективные центры обработки данных облачных вычислений с коммуникацией между элементами.

Выбранными функциями симулятора являются:

1) моделирование облачных сетей и осведомленности об энергопотреблении;

2) моделирование ресурсов процессора, памяти, хранилища и сети;

3) независимая абстракция сущности «энергия» для каждого типа ресурсов;

4) миграция виртуальных машин в облако;

5) протокол TCP/IP.

SCORE [3.19] - это инструмент, ориентированный на моделирование крупных центров обработки данных. Он предлагает несколько различных моделей планирования, распределения, выключения и включения питания для виртуальных машин. SCORE также поддерживает низкоуровневые функции (такие как сведения о физических машинах) и моделирование сетевых элементов.

Основной рабочий процесс каждого моделирования состоит из:

1) определения и инициализации облачной системы;

2) постановки целей моделирования;

3) определения входных данных модели;

4) выбора уровня детализации для моделирования.

Наиболее важные характеристики вышеуказанных симуляторов сравниваются в [3.1, 3.29, 3.43]. С точки зрения облачного моделирования, наиболее важными характеристиками являются:

• модели архитектуры планировщика (т.е. параллельные, распределенные или централизованные);

• масштабируемость (т.е. возможность расширения симулятора за счет введения новых классов с минимальными усилиями);

• измерение энергопотребления;

• автоматический сбор статистики.

В табл. 1.3 представлены различные возможности инструмента в отношении вышеупомянутых свойств.

Таблица 1.3

Сравнение систем имитационного моделирования

Система	**Модели планирования**	**Учет энергии**	**Автоматическая статистика**	**Открытый код**	**Учебные пособия**
GridSim	-	-	-	+	+
CloudSim	-	+	-	+	+
GreenCloud	-	+	+	+	+
SCORE	+	+	+	+	-

1.2.4. Прогнозирование производительности облачных вычислений

Во многих аспектах облачные вычисления отличаются от архитектуры больших данных. Хотя они разделяют бремя управления большими объемами данных и проблемы масштабирования, парадигма облачных вычислений ориентирована на мелкозернистые, взаимодействующие и распределенные приложения. Обзор по моделированию производительности больших данных и облачной архитектуры представлен в [3.22].

Проблемы, связанные с прогнозированием производительности архитектуры такого типа, рассмотрены в [3.30], где используются методы аналитического и имитационного моделирования. В [3.34] представлен второй обзор с точки зрения методов, основанных на измерениях.

Прогнозирование производительности на основе моделирования рассмотрено в [3.40].

Ранее исследовалось моделирование производительности облачных архитектур на обоих этапах поэтапного процесса. В [3.4, 3.10] был предложен аналитический подход, соответствующий масштабу облачных архитектур, основанный на анализе среднего поля или марковских агентах, который будет использоваться на предварительном этапе процесса. В [3.10] описывается пример анализа производительности крупного мультиоблачного объекта, выполняющего сложные параллельные приложения с большими данными, в то время как в [3.4] представлен анализ влияния хранилища в аналогичном глобальном распределенном сценарии. Подробное моделирование планирования в облачных средах можно найти в [3.27, 3.28], в которых обсуждаются недетерминированные методы для создания временных моментов после запуска процесса планирования. Авторы в [3.27, 3.28] сосредоточились на моделировании потока задач в облаках с использованием концепции рабочей нагрузки и вычислительной мощности, которые учитывают операции по обеспечению безопасности. Кроме того, был представлен планировщик на основе генетического алгоритма (учитывающий ограничения безопасности в облачных организациях). В [3.21] моделирование используется для оценки влияния на надежность и производительность крупномасштабных распределенных приложений или методов управления хранилищем, основанных на репликации и стирающем кодировании.

В работе предлагается решение, которое может быть применено на промежуточном этапе процесса моделирования. Цель состоит в том, чтобы

разрешить, по крайней мере, частичное повторное использование аналитических моделей и использовать DSL в качестве связующего звена между логикой абстрактного моделирования и потребностями специализированного имитационного подхода.

Представленное решение использует мультиформальный подход, позволяющий разработчику модели абстрактно определять классы рабочей нагрузки, конфигурацию облака и взаимосвязи между рабочей нагрузкой и облаком в сценарии мультиоблачных/гибридных облачных/пограничных вычислений.

В то время как формализмы основаны на подходе SIMTHESys, а механизм множественного решения частично основан на подходе OsMoSys, поддержка моделирования была выбрана из наиболее известных литературных предложений. Выбор пал на CloudSim из-за, во-первых, его точности и гибкости в описании внутренних деталей, во-вторых, его популярности, что приводит к широко признанной точности и широкому использованию в реальных условиях эксплуатации, и в-третьих, его широкой конфигурируемости, что в принципе позволяет моделировать конкретные коммерческие стандарты и продукты, такие как OpenStack. Поскольку чистый CloudSim изначально не поддерживает стохастическое моделирование, была введена соответствующая поддержка, использующая внутренние механизмы управления событиями CloudSim. Хотя рассматривался CloudSimPlus [3.20] и его полезное дополнение к CloudSim, было решено полагаться на CloudSim из-за его более широкой аудитории.

Далее предполагается, что предварительная модель облачного сценария показала многообещающие результаты после анализа, основанного на чисто аналитических инструментах. Поскольку модель синтетически описывает рабочую нагрузку и архитектуру, сохранение такого представления рабочих нагрузок на протяжении всего процесса до последнего этапа является разумным решением. Наконец, необходимо изменить представле-

ние архитектуры на формализм, совместимый с созданием модели CloudSim. В результате стохастические рабочие нагрузки могут генерироваться в соответствии с высокоуровневым описанием, предоставляемым частью модели GSPN, и путем внедрения поддержки генерации облачных блоков CloudSim произвольной длины. Описание архитектуры заменено на описание, соответствующее характеристикам CloudSim, в дополнение к возможностям планирования времени случайного использования ресурсов CloudSim. Аналогичный сценарий, касающийся приложений с большими данными, можно найти в [3.3].

1.3. Высокоуровневое моделирование производительности и прогнозирования доступности многоуровневой облачной среды

Моделирование производительности является важным процессом для оценки качества облака. Процесс и методы оценки производительности облака значительно отличаются от других проверенных методологий, связанных с производительностью, которые используются в таких областях, как компьютерные сети, распределенные вычисления и операционные системы. Многоуровневое облако - это масштабируемая система, в которой можно создать множество сервисов или уровней для всех типов приложений. Качество обслуживания в многоуровневой облачной среде тесно связано с несколькими факторами, такими как надежность, доступность, безотказность, безопасность, производительность, и каждый из элементов производительности прямо или косвенно влияет на общее функционирование облака. Существует множество моделей для оценки производительности и качества облака, но эти традиционные модели недостаточно эффективны и учитывают при оценке только определенные первичные параметры. В этой статье предлагается высокоуровневая модель анализа производительности, которая может прогнозировать доступность многоуровневой

облачной среды. Поскольку многоуровневое облако работает с несколькими сервисами в зависимости от их применения, прогнозирование доступности является важным для моделирования производительности. В предлагаемой модели также учитываются различные важные параметры, влияющие на производительность облачной системы. Предложенный алгоритм экспериментально проверен с помощью инструмента SHARPE.

Облачные вычисления - это модель, обеспечивающая повсеместный доступ к вычислительным ресурсам в рамках общего пула сервисов. Облачные вычисления предоставляют новый метод потребления, модели доставки и дополнительные методы для ИТ-услуг с использованием Интернета. Клиент может получить доступ к приложению через Интернет и использовать его динамически с оплатой по мере поступления. Приложения, разрабатываемые в облаке, виртуализируются в зависимости от потребностей клиента. Облачные вычисления стали похожи на услугу. В настоящее время многие организации переходят на облачные технологии, основной целью которых является снижение стоимости инфраструктуры и ее динамичное приобретение с помощью облачной инфраструктуры, размещаемой и обслуживаемой поставщиками облачных услуг. Многие важные облачные приложения имеют многоуровневую архитектуру [4.20]. Традиционные модели, такие как 1-уровневая, 2-уровневая и 3-уровневая, могут предоставлять от 1 до 3 видов услуг в комбинированной форме, с другой стороны, в многоуровневой модели N услуг предоставляются в независимой форме. На одном уровне можно предоставлять несколько сервисов, но это имеет некоторые ограничения и негибкость. На одном уровне производительность снижается в большей степени. Чтобы избежать этих недостатков, выбрана многоуровневая облачная среда. Для каждой отдельной службы в архитектурный проект добавляется уровень, позволяющий распределенно использовать службы повторно. Например, Amazon Elastic compute cloud (EC2), как описано в [4.26], и облачные сервисы Microsoft

Azure, как описано в [4.28], помогают в процессе подготовки многоуровневого веб-приложения. Кроме того, традиционная архитектура нацелена на обеспечение соглашения об уровне обслуживания (SLA) путем реализации различных сервисов на отдельном физическом компьютере. Многоуровневая архитектура имеет ряд преимуществ перед традиционной архитектурой. Многоуровневая архитектура может обслуживать большее количество пользователей с высокой рабочей нагрузкой и пропускной способностью на каждом уровне. Чтобы сократить эти затраты, пространство и мощность, на каждом уровне внедряется виртуализация для достижения бизнес-целей с высокой производительностью. Виртуализация позволяет снизить энергопотребление и обслуживать большее число пользователей. В работе [4.30] был сделан вывод о том, что технология виртуализации хорошо подходит для многоуровневых приложений, она потребляет меньше энергии, требует меньших затрат на внедрение и обеспечивает большую доступность. Таким образом, наша статья посвящена повышению доступности многоуровневого облачного приложения, технология виртуализации которого развернута в нашей облачной системе.

Доступность является важным качеством облачной системы. Доступность может гарантировать качество обслуживания (QoS) в облачных системах. Высокая доступность может быть достигнута с помощью аналитической модели, приведенной в [4.17]. В [4.19] была разработана модель доступности для облака с запуском репликации, когда система становится недоступной для служб. Доступность системы определяет возможности и характер системы. Многие облачные машины оцениваются на основе доступности и быстродействия, предлагаемых клиенту. Amazon EC2 - это веб-сервис-провайдер, который работает в течение 750 часов на сервере Microsoft, Linux и на плате Elastic board. Ему требуется 15 ГБ для обработки данных и 15 ГБ для выполнения всех сервисов AWS. Статистика доступности Amazon EC2 составляет 99,95%. Эти показатели используются

для определения качества предоставляемых услуг. В cloud machine часто происходят сбои в работе, что влияет на доступность системы. Перебои в работе, произошедшие за последние годы, перечислены в таблице перебоев в работе поставщиков услуг [4.13]. Сбои в работе подразделяются на человеческие ошибки, аппаратные сбои, внешние причины и программные ошибки. В [4.12] было представлено развертывание частного облака на базе Eucalyptus для обеспечения высокой доступности. Во всех предыдущих исследованиях доступность облачной системы рассчитывалась по-разному. И они также учли, что качество обслуживания (QoS) может быть достигнуто за счет доступности. В [4.14] проведено исследование доступности многоуровневых облачных сервисов и получен вывод, что доступность гарантирует Соглашение об уровне обслуживания (SLA). SLA - это сервисные контракты, которые обеспечивают объем и качество услуг для конечного пользователя. В облачных вычислениях SLA необходим для того, чтобы конечный пользователь удовлетворял свои потребности. Таким образом, в многоуровневых приложениях SLA должен гарантироваться за счет повышения доступности услуг.

В работе рассматривается проблема традиционного моделирования производительности, а также предлагается решение для анализа производительности в многоуровневом облаке. В этом моделировании среднее время жизни отдельных компонентов объединяется в целостную многоуровневую систему для расчета доступности. Моделирование выполняется с двух разных точек зрения, например, на основе инфраструктуры и приложений. При моделировании на основе инфраструктуры рассматривается набор компонентов, таких как виртуальная машина (VM) и хост-машина (HM). Основные компоненты VM и HM включают в себя такие их подкомпоненты, как загрузка процессора, памяти, диска, передача данных и сетевых байтов. В предлагаемой модели рассматриваются эти конечные компоненты, которые могут повлиять на производительность многоуровневой

облачной среды. В модели, основанной на приложении, учитывается набор показателей, таких как метрика хоста, пропускная способность, использование ресурсов и виртуальные показатели. После оценки срока службы этих компонентов вычисляется среднее время наработки на отказ (MTTF). Доступность многоуровневой облачной системы определяется по MTTF виртуальной машины и HM. На основе этой доступности выполняется анализ производительности. Доступность оценивается путем объединения набора важных параметров, чтобы приблизительное предсказание доступности было точным в этой модели по сравнению с моделью доступности [4.25]. Предлагаемое моделирование оценивается путем реализации модели в инструменте SHARP. Результатом этой работы является разработка модели производительности, которая позволяет измерить любую облачную систему на основе ее доступности и проанализировать качество предоставляемых ею услуг.

Доступность и надежность являются важными требованиями к облачным сервисам, и их необходимо решать с помощью надлежащего планирования и математического моделирования [4.18]. Модель доступности и надежности сосредоточена в облачной виртуальной машине для достижения высокой производительности в облаке [4.11]. В [4.29] охарактеризован параметр надежности облачного оборудования для поддержания согласованности системы. Анализ производительности включает эти показатели для измерения качества обслуживания. В [4.8] утверждается, что доступность и надежность тесно связаны при анализе производительности облачных вычислений. Для достижения доступности разрабатывается множество новых моделей и параметров. Ранжирование и выбор CSP рассматривается как процесс оценки и сертификации [4.27]. Для достижения высокой производительности в облачной системе используется показатель надежности, который заменяет традиционную доступность, используя теоретический алгоритм и архитектуру [4.19]. Во многих статьях и рекомен-

дациях по безопасности обсуждается доступность услуг и данных в облачных вычислениях [4.24], они также посвящены аппаратному обеспечению и избыточности данных в CSP при обсуждении перебоев в работе Интернета. Некоторые исследователи провели исследование доступности для CSP. Здесь для CSP разработана модель сквозной доступности [4.16].

Определение качества обслуживания (QoS) является важной проблемой для поставщиков облачных услуг (CSP) и потребителей облачных услуг (CC). Авторы [4.1] проводят систематический обзор QoS. Различные показатели и методологии тщательно анализируются с точки зрения QoS. Также перечислены все работы, связанные с обеспечением качества обслуживания. Они внедрили множество показателей, которые могут улучшить качество обслуживания. Важным параметром среди них является доступность и надежность. Таким образом, для улучшения качества обслуживания в облаке нам необходимо поддерживать доступность и надежность в среде облачных вычислений. В [4.15] было проведено моделирование производительности и доступности облачных вычислений. Они разработали определенные показатели и алгоритмы для обеспечения доступности облачной системы баз данных. Показателями производительности для обеспечения доступности являются доступность файлового сервера, частота отклонений и доля запросов пользователей. Для оценки производительности облачного центра обработки данных и расчета QoS разработана стохастическая система вознаграждений (SRN) [4.9]. Для целей проверки используются такие параметры, как скорость отклика, доступность, время ожидания и коэффициент использования. Разработана аналитическая модель [4.17] для анализа производительности облачных вычислений с использованием модели массового обслуживания M/G/m/m+r. В результате достигается высокое время отклика и меньшая интенсивность трафика. Для доставки сообщений без потерь и обеспечения высокой доступности в облачных средах используется новый метод обработки запросов под на-

званием Silver Dove Queuing System (SDQS). Для преодоления накладных расходов на уровне виртуализации, недостаточного количества журналов трассировки и сложной рабочей нагрузки при использовании ресурсов облачных вычислений предлагается статистическое распределение с расширением CloudSim simulation [4.21], создающее аналитический подход к закрытой сетевой очереди в многоуровневых приложениях с однопоточным параллелизмом. Тип сервера - это виртуальная машина с уровнем детализации транзакций. Эта модель повышает производительность многоуровневого приложения. В работе [4.10] предложен сетевой подход к организации массового обслуживания с несколькими станциями для многоуровневых приложений с многопоточным параллелизмом, уровнем детализации запросов и типом сервера виртуальной машины. В работе [4.8] разработан гибридный подход к организации массового обслуживания для повышения производительности многоуровневых приложений с однопоточным параллелизмом, уровнем детализации запросов и типом сервера виртуальной машины.

Облачные вычисления используются в различных областях применения с различным качеством обслуживания (QoS), таким как производительность, полезность, доступность и надежность. Эти аспекты определены в соглашении об уровне обслуживания, заключаемом между потребителем облачных услуг и поставщиком облачных услуг. Сбой в QoS приводит к снижению производительности и требует соглашения SLA. Чтобы гарантировать SLA для облачного сервиса, в работе [4.23] предлагается сервис с поддержкой SLA (SlaaS). Этот SlaaS предоставляет специфический язык, теоретический подход к управлению и различные сервисы. Тщательно изучаются такие аспекты QoS, как частота отказов, MTTF, время отклика, пропускная способность, надежность и финансовые затраты. Из приведенных выше работ мы узнали о различных методах и показателях, используемых для достижения QoS, таких как доступность и надеж-

ность для обеспечения SLA.

1.4. Постановка задач работы

Целью исследования является создание моделей и алгоритмов разработки специального программного обеспечения инфокоммуникационных систем в облачных средах.

Для достижения цели необходимо решить следующие **задачи**:

1. Провести анализ проблем сквозного моделирования и алгоритмизации разработки специального программного обеспечения с открытым кодом в облачных средах.

2. Создать ситуационную модель организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающую определение степени открытости программного кода и уровень стратегической открытости организации.

3. Предложить технологию стохастического моделирования облачной архитектуры, позволяющая автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры

4. Разработать алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений, обеспечивающий расчет трафика для определения наличия аварийного состояния блокировки в системе и определения точного местоположения точки блокировки

5. Предложить модель анализа производительности и прогнозирования доступности многоуровневой облачной среды, обеспечивающую учет времени жизни основных компонентов и оценку доступности облачной среды.

6. Разработать структуру программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов, обеспечивающую реконфигурацию инфокоммуникационной системы в зависимости

от параметров инфраструктуры и качества обслуживания.

Анализ задач привел к дизайну исследования, представленному на рис. 1.3.

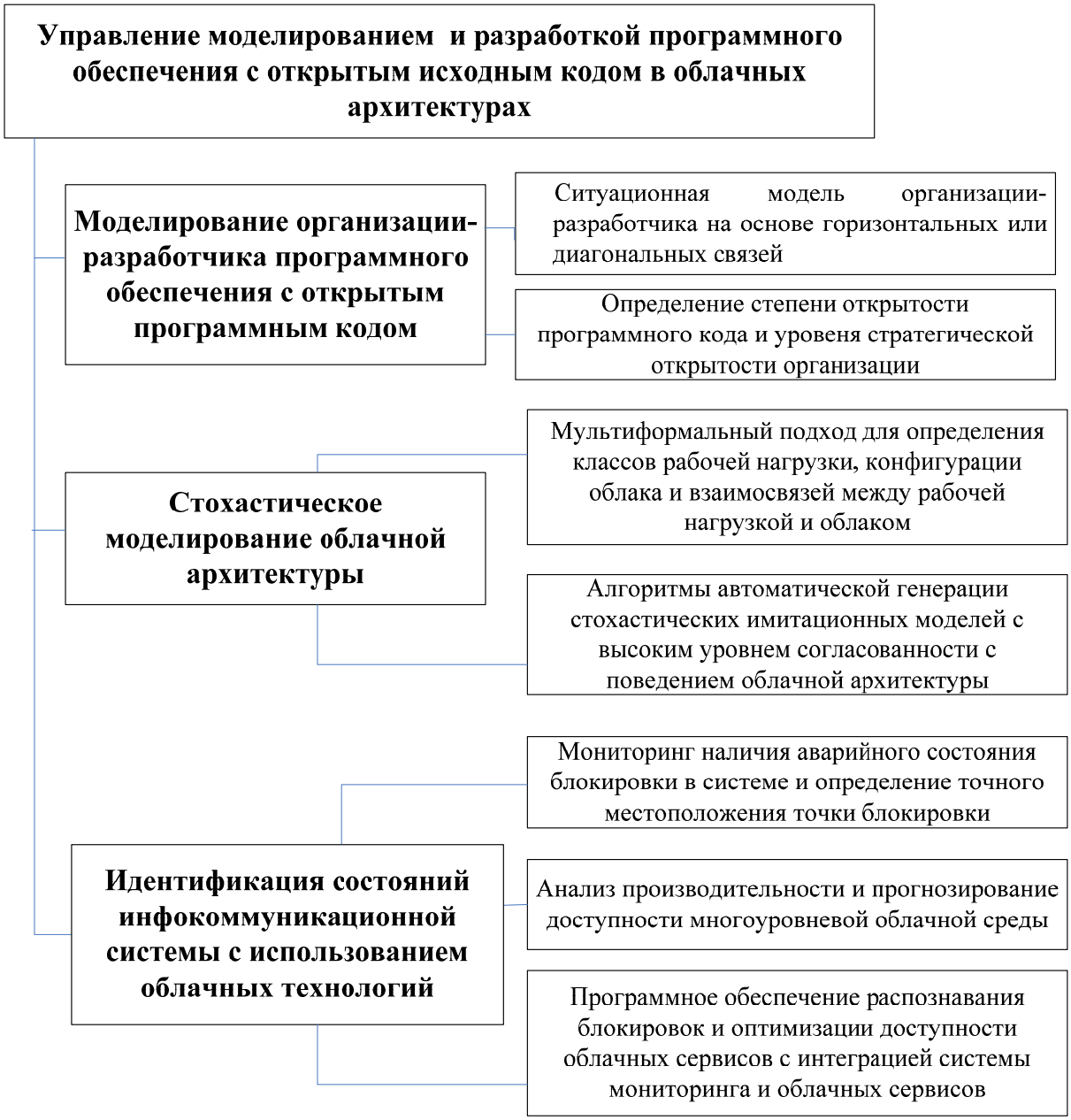


Рис. 1.3. Дизайн исследования

2. СИТУАЦИОННОЕ МОДЕЛИРОВАНИЕ ОРГАНИЗАЦИИ-РАЗРАБОТЧИКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ОТКРЫТЫМ КОДОМ В ЭКОСИСТЕМЕ ЖИЗНЕННОГО ЦИКЛА

2.1. Модель предприятия с открытым программным обеспечением

Определены три фактора, побуждающих к открытию бизнеса по разработке программного обеспечения. Основная цель, стоящая за каждым из этих факторов, заключается в более широком внедрении программного продукта.

Первый мотиватор - это идеализм. Как только бизнес, занимающийся разработкой программного обеспечения, полностью раскрывается, организация безоговорочно заявляет, что верит в свои продукты и их внедрение. Благодаря этим открытым аспектам больше участников экосистемы могут оказывать свое влияние на окружающую среду.

Вторым мотивирующим фактором является выживание. Если организация конкурирует с другой, открытость может способствовать более быстрому внедрению и стандартизации ее продуктов, тем самым опережая несвободного конкурента. Примерами этого являются Eclipse от IBM и Netscape от Mozilla, которые были созданы для того, чтобы конкурировать с де-факто (коммерческим) решением.

Третий мотиватор обычно носит коммерческий характер: открытость может, например, заключаться в бесплатном предоставлении клиентам (части) функционала, но когда они хотят использовать программное обеспечение на коммерческой основе, им необходимо перейти на коммерческую лицензию.

Модель предприятия с открытым программным обеспечением, представленная далее, определяется как перечень различных вариантов открытости, которыми располагает SPO, для обеспечения открытости (частей)

организационного управления, процессов разработки программного обеспечения, процессов управления программными продуктами, процессов маркетинга и продаж, а также процессов консультирования и поддержки. Эти варианты открытости могут оказывать как положительное, так и отрицательное влияние на бизнес-модель. Каждый из вариантов открытости связан с решением поделиться чем-то, что не было бы доступно другим пользователям.

‘Традиционное’ закрытое SPO. Одним из убедительных аргументов в пользу модели OSE является интеграция партнеров по цепочке поставок, таких как VAR и внешние разработчики программного обеспечения. Концепция OSE и ее интеграции с цепочкой поставок основана на концепции расширенного корпоративного подхода, представленного в [2.14]. Модель OSE основана на предпосылке, что открытость - это не вопрос полной открытости или полной закрытости: организации могут выбирать степень открытости.

2.1.1. Модель оперирующей системы

Модель оперирующей системы представлена в табл. 2.1. Модель оперирующей системы была создана в двух измерениях, одним из которых является измерение практик SPO и измерение уровня управления. В измерении уровня управления есть три уровня: стратегический, тактический и оперативный (от долгосрочного до краткосрочного). Как показывает эмпирическое правило, каждый более низкий уровень управления открытостью предполагает более короткий управленческий временной интервал: в то время как стратегическое управление обычно охватывает период в 5-10 лет, оперативное управление связано с управлением организацией. Эти три уровня управления открытостью (стратегический, тактический и оперативный) взяты из традиционной литературы по менеджменту [2.18]. Что касается практик SPO, то пять областей открытости - управление, разра-

ботка программного обеспечения, управление продуктами, маркетинг и продажи, а также консультационная поддержка и услуги - основаны на исследовательской модели [2.50] и пяти основных бизнес-процессах SPO [2.39] Возможности открытости для исследований и разработок в рамках модели были определены в значительной степени с помощью CMM-i [2.12]. Программное обеспечение элементов управления продуктом было проверено путем сравнения и противопоставления с системой управления продуктом [2.47].

Таблица 2.1

Модель предприятия с открытым программным обеспечением

Задачи	**Управление**	**Исследование и разработка**	**Управление программными продуктами**	**Маркетинг и сбыт**	**Служба поддержки**
стратегические	Запуск управления Создание модели партнерства Открытая стратегия в области ИС Координация вклада в другие экосистемы Политика совместного участия в конкуренции Стратегия приобретения доли Стратегия управления, основанная на знаниях об экосистеме	Делиться технологиями и планами исследований Делиться знаниями о процессе разработки Стимулировать открытые стандарты Делиться исходным кодом Подавать заявки на совместное финансирование исследований и разработок	Делиться планами жизненного цикла продукта по продуктам Делиться стратегией и видением платформы	Делиться видением рынка Разрабатывать инновационные бизнес-модели Создавать партнерскую программу по продажам	Делиться стратегией управления предоставлением услуг
тактические	Внедрять стандарты процесса разработки Создавать	Делиться инновациями Поддержка взаимозаменяемых фор-	Передача разработки требований на аутсорсинг партнерам	Делиться информацией о рынке Делиться информацией о	Передача проектов внедрения на аутсорсинг партнерам

Задачи	**Управление**	**Исследование и разработка**	**Управление программными продуктами**	**Маркетинг и сбыт**	**Служба поддержки**
	модели партнерства Согласовывать жалобы Помогать партнерам в конфликтах интеллектуальной собственности	матов данных Политика совместного использования исходного кода Создание политики повторного использования кода Передача задач на аутсорсинг Сертификация компонентов сторонних производителей	Совместное использование и корректировка плана (ов) разработки продукта (линейки) Управление интеллектуальной собственностью сторонних производителей в продукте	клиентах и поставщиках Разработка каналов сбыта	Общий доступ к базе данных прецедентов Обмен знаниями о процессах проекта Разработка и распространение показателей качества
оперативные	Наглядное представление экосистемы Создание каталога партнеров Создание групп пользователей Использование и создание многоразовых лицензий на программное обеспечение	Создание и публикация (контента) API и SDK Создание архитектуры, позволяющей повторно использовать программное обеспечение Запуск процесса тестирования Совместное использование хранилища ошибок Совместная разработка Обучение разработчиков Распространение знаний об эксплуатации про-	Запуск процесса управления требованиями Запуск планирования процесса выпуска Делиться кандидатами на выпуск	Сертифицировать партнеров Создание внутреннего и внешнего рынков комплектующих Вовлекать партнеров в маркетинг и продаж	Делиться знаниями о внедрении Делиться (клиент) конфигурацией знаний Использовать рабочие пространства для совместной работы с клиентами Обеспечить обучение консультантов

Задачи	Управление	Исследование и разработка	Управление программными продуктами	Маркетинг и сбыт	Служба поддержки
		граммного обеспечения			

Итоговая ситуационная модель организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации, представлена на рис. 2.1.

Ключевой игрок - это участник экосистемы, чей вклад в экосистему стимулирует здоровье всей экосистемы. Особый тип ключевых игроков - это поставщики технологий, которые предоставляют платформу, программное обеспечение и/или стандарты, которые используются большим количеством участников экосистемы. Ключевой игрок определяется не по его размеру, а по его вкладу в общее состояние «здоровья» игрока экосистемы.

Модель OSE ориентирована на участников экосистемы программного обеспечения, которые напрямую контактируют с Ключевым поставщиком технологий [2.21]. Партнеры определяются как предприятия, работающие в качестве партнеров в какой-либо деятельности, начинании или сфере общих интересов.

При открытии клиенты и партнеры, окружающие компанию, занимающуюся разработкой программного обеспечения, привлекаются в организацию и могут стать активными участниками ранее закрытых процессов. Когда рассматриваются перспективы экосистемы программного обеспечения [2.28] (роль субъекта, сети поставок программного обеспечения и уровень экосистемы программного обеспечения), модель OSE сосредоточена на уровне сети поставок программного обеспечения. Необходимо отметить, что для успешной работы открытой организации существуют крити-

ческие требования. Наиболее важным требованием является то, чтобы число партнеров и клиентов, способных и желающих участвовать в процессах открытой организации, было достаточно большим.

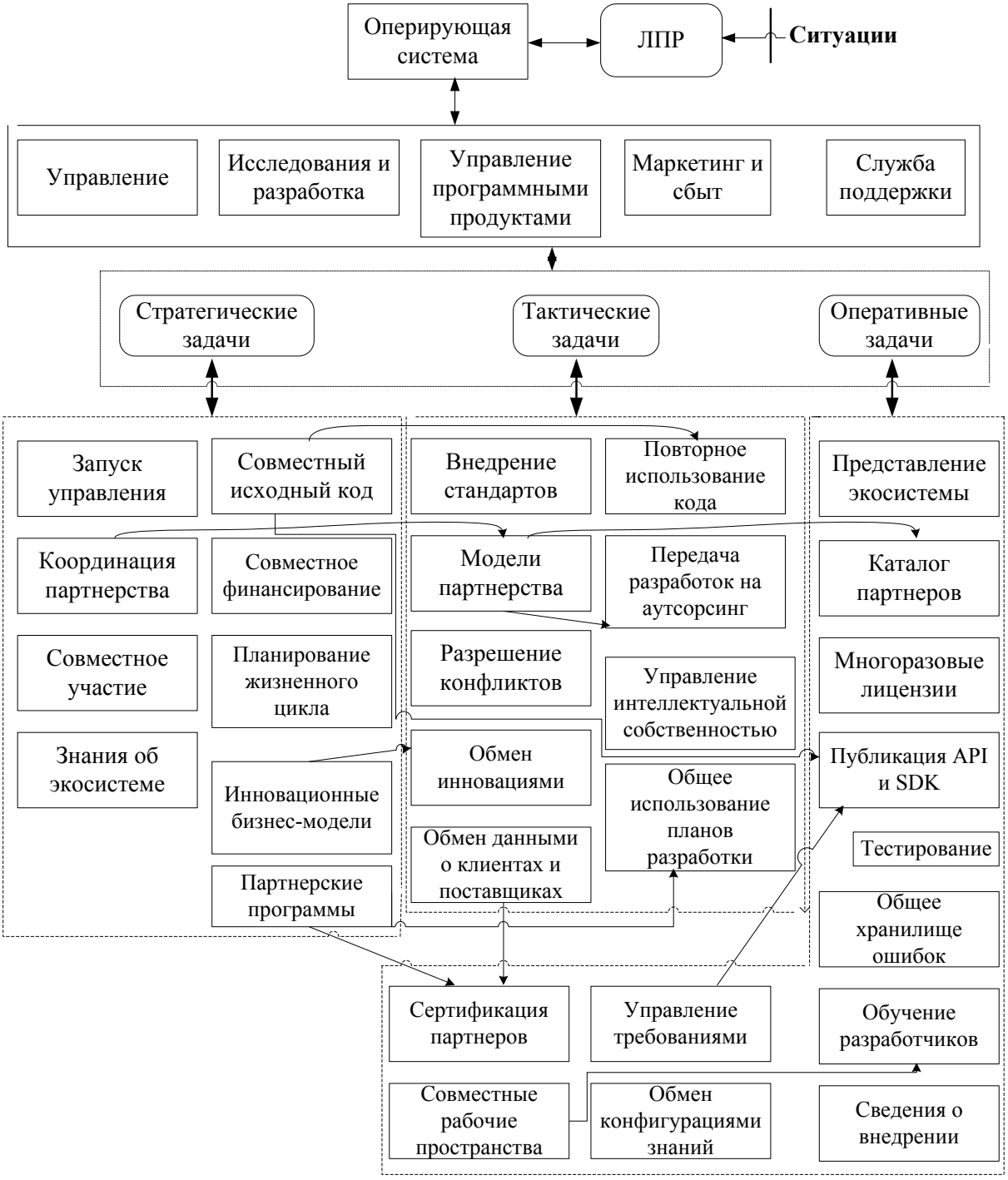


Рис. 2.1. Ситуационная модель организации-разработчика программного обеспечения с открытым программным кодом

2.1.2. Оперирующая система: управление

Управление определяется как способ управления организацией, включая ее полномочия, ответственность и процессы принятия решений [2.19]. Как правило, принципы управления организацией закреплены в ее уставе или юридическом статусе. Управление разработкой программного обеспечения включает в себя распределение ролей и прав на принятие решений, а также меры и политику, которые позволяют проводить непрерывную оценку. В контексте статьи также определяется, сколько полномочий остается у сообщества и сколько программного обеспечения организация-разработчик "создает сама по себе", поскольку координация разработки программного обеспечения в цепочке поставок отличается от разработки в рамках одной организационной единицы [2.39].

На стратегическом уровне (управление) организация может принять решение о полной открытости своей политики управления (в форме чтения или даже записи), тем самым предоставляя сообществу вокруг организации понимание или даже власть в организации. Примером открытости политики управления является назначение открытого комитета, который решает, что будет содержаться в новых версиях продукта. Предварительное условие заключается в том, что экосистема, окружающая организацию, четко обозначена и существует своего рода модель партнерства или членства. Вторая часть политики открытости управления – это открытость интеллектуальной собственности (ИС), созданной организацией, где ИС относится не только к исходному коду, но и к любым другим артефактам, разработанным в организации, вплоть до описаний процессов и планов жизненного цикла продукта. Стратегический уровень открытости управления тесно связан с вопросами конкуренции и другими экосистемами. Начнем с того, что организация могут принять решение внести свой вклад или даже интегрироваться в другие экосистемы.

Кроме того, экосистемы могут иметь явные или неявные стратегии

взаимодействия с другими (конкурирующими) экосистемами. Кроме того, организация может открыто заявить о своей политике приобретения, т.е. о том, каким образом организация выбирает предприятия для приобретения и патенты для приобретения. Наконец, организация должна разработать стратегию управления знаниями, которая позволит партнерам по экосистеме программного обеспечения расширить свои возможности, не теряя при этом слишком много информации и ценности. Примером может служить обнаружение ошибки системы отслеживания, которая информирует сообщество о том, какие ошибки все еще остаются открытыми, а также информирует конкурентов о слабых местах в продукте.

На тактическом уровне (управление) организация может раскрыться, помогая партнерам несколькими способами: во-первых, она может обеспечить или даже навязать своим партнерам процесс разработки. Во-вторых, общие процессы управления могут быть общими и распространены среди партнеров, разрабатывающих компоненты или подключаемые модули для платформы, предоставляемой организацией. В-третьих, организация может помогать партнерам в разрешении споров между собой и даже оказывать содействие в разработке новых продуктов.

На операционном уровне (управление) у организации есть три варианта стать более открытой: организация может четко обозначить экосистему, указав, какие типы партнеров являются частью экосистемы и как они связаны. Кроме того, организация может создать каталог партнеров, в котором партнеры и заказчики смогут найти (других) партнеров и заказчиков.

Кроме того, создание активных групп пользователей может помочь этим клиентам объединить свои усилия и запросы и, таким образом, обеспечить более эффективную обратную связь с SPO. Наконец, организация может выбрать способ создания лицензий, которые могут быть повторно использованы другими участниками экосистемы.

2.1.3. Оперирующая система: исследования и разработки (R&D)

Расходы на исследования и разработки (R&D) в бизнесе программного обеспечения в среднем составляют 25% от общей выручки [2.40]. Отдел исследований и разработок играет самую важную роль в открытии бизнеса в области программного обеспечения, поскольку именно там создаются наиболее ценные знания. Кроме того, отдел исследований и разработок осведомлен о потенциальных партнерах: в какой-то момент разработки эти партнеры начали обращаться с запросами информации, вопросами по API или даже запросами исходного кода. Из-за этого, как правило, первые запросы на открытость поступают из отдела исследований и разработок.

На стратегическом уровне (НИОКР) отдел исследований и разработок может принять решение поделиться технологической картой, касающейся предстоящих задач, имеющих отношение к области, на которой сосредоточен бизнес программного обеспечения. Сразу за технологическим видением следует исследовательское видение, касающееся задач, на решении которых отдел исследований и разработок собирается сосредоточиться в последующие годы. Есть несколько организаций, которые преуспели в представлении этих идей, например, Apple на своих мероприятиях, за которыми пристально следят журналисты, и Microsoft на своих всемирных днях разработки. Еще один вариант исследований и разработок - департаменты должны публиковать процедуры разработки с различными целями. Одной из целей может быть обеспечение прозрачности, но чаще эти процедуры публикуются для того, чтобы побудить участников экосистемы следовать этим же процедурам. Еще одним стратегическим вариантом для отдела исследований и разработок является поддержка открытых стандартов, в отличие от закрытых. Эти открытые стандарты оказываются полезными для экосистемы [2.4], поскольку запатентованные стандарты, как правило, сопровождаются несколькими дорогостоящими лицензионными

продуктами. Примерами таких закрытых стандартов являются, например например, формат DWG [2.34] от AutoCAD и формат изображений PSD Photoshop от Adobe.

Одним из наиболее важных вариантов является предоставление общего доступа к исходному коду продукта. Открытие исходного кода может осуществляться на разных уровнях. Эти уровни варьируются от обеспечения возможности чтения исходного кода партнерами по консорциуму до обеспечения возможности записи любым пользователем. Степень открытости полностью зависит от используемой бизнес-модели.

Если партнеры платят "арендную плату" за участие в экосистеме, первый вариант может оказаться выгодным. Если организация-поставщик только продает услуги, с таким же успехом исходный код может быть доступен для записи любому, кто хочет улучшить продукт. У компаний, занимающихся разработкой программного обеспечения, также есть возможность подать заявку на финансирование исследований для дальнейшего стимулирования открытых стандартов и знаний. Поставщик может подать заявку на финансирование исследований, поскольку его технология способствует достижению целей не только бизнеса в области программного обеспечения, но и его партнеров.

На тактическом уровне (НИОКР) отдел исследований и разработок может использовать инновации, которые напрямую не связаны с основной ценностью продукта. Если организация не планирует проводить дальнейшие исследования учитывая жизнеспособность и бизнес-модель, стоящие за инновацией, возможно, стоит поделиться ею с партнерами, заинтересованными в этой конкретной области. Другой вариант – поддерживать взаимозаменяемые форматы данных, которые позволяют обмениваться данными между различными продуктами (т.е. различными экосистемами).

Еще один вариант - предоставить общий доступ к исходному коду и политикам повторного использования.

Эти политики в отношении исходного кода определяют, как следует повторно использовать другой исходный код (например, из Интернета) и как следует публиковать исходный код. Кроме того, политика повторного использования определяет, какие лицензии могут быть рассмотрены для включения в основной продукт.

К сожалению, компании, занимающиеся разработкой программного обеспечения, по-прежнему часто обнаруживают, что компоненты сторонних производителей, используемые в некоторых продуктах, имеют несовместимую лицензию [2.1]. Другим вариантом на тактическом уровне является передача задач на аутсорсинг для достижения масштабируемости.

Поскольку эти задачи передаются на аутсорсинг, другие партнеры включаются в производственный процесс, что еще больше приближает их к нему. Поставщик, который добился успеха в этом в прошлом - Cisco [2.22]. Наконец, организация может сертифицировать компоненты других участников экосистемы, чтобы присвоить им рейтинг качества, который эти участники могут использовать для продаж и продвижения.

На операционном уровне (НИОКР) компания, занимающаяся разработкой программного обеспечения, может задействовать несколько операционных частей процесса разработки, таких как система отслеживания ошибок и процесс тестирования. Кроме того, что касается архитектуры программного обеспечения, для развертывания программного продукта может быть предпринято несколько шагов. Прежде всего, продукты могут быть открыты с использованием Интерфейсов прикладных программ (API), позволяющих партнерам использовать функциональность и повторно использовать контент. Во-вторых, по мере роста продукта может оказаться целесообразным модулировать и создавать архитектуру, позволяющую повторно использовать разработку. Такая архитектура стимулирует повторное использование как в отделе исследований и разработок, так и для других лиц, желающих повторно использовать компоненты продукта.

Что касается операционной деятельности, то R&D может ускорить процесс тестирования, предоставляя доступ как к результатам тестирования, так и к возможным версиям продукта.

Такая прозрачность позволяет зависимым сторонам предварительно протестировать и потенциально соответствующим образом корректировать свои продукты. Кроме того, совместное использование хранилища ошибок может быть полезным для третьих сторон, поскольку позволяет получить представление о потенциально неожиданном поведении продукта.

Кроме того, совместная разработка продукта с другими разработчиками и обучение сторонних разработчиков могут принести пользу экосистеме. Наконец, знания об эксплуатации программного обеспечения, то есть знания, которые могут быть получены из программного обеспечения, работающего в полевых условиях, такие как данные о производительности и отчеты о сбоях, могут быть переданы участникам экосистемы для дальнейшего стимулирования более качественного программного обеспечения в полевых условиях [2.48].

2.1.4. Оперирующая система: Управление программным продуктом (SPM)

Управление программным продуктом (SPM) определяется в [2.23] как процесс управления программным обеспечением, которое создается и внедряется как продукт, с учетом особенностей жизненного цикла. Это дисциплина и бизнес-процесс, которые управляют продуктом с момента его создания до выхода на рынок или предоставления услуг клиентам с целью создания максимально возможной ценности для организации [2.23]. SPM отличается от управления тем, что оно больше фокусируется на внутренних особенностях продукта, таких как его требования, качество, планы развития и рыночные планы, в то время как управление в большей степени сосредоточено на владении организационной и интеллектуальной собст-

венностью, процессах разработки, организационной стратегии и управлении сообществом.

На стратегическом уровне (SPM) организации могут поделиться как стратегией, так и видением своей платформы, представляя их на ежегодных собраниях. Кроме того, менеджеры уровня SPM могут публиковать свои процессы и жизненные циклы в отношении продуктов, чтобы другие могли получить представление о них и, возможно, повторно использовать жизненный цикл в своих собственных целях. Поступая таким образом, другие компании могут использовать аналогичные методы разработки, создавая более однородную экосистему.

На тактическом уровне (SPM) организации могут передать разработку требований на аутсорсинг, чтобы другие стороны участвовали в сборе требований для будущих версий. Кроме того, организации могут по своему усмотрению открывать дорожные карты продуктов.

Они могут быть открыты таким образом, чтобы другие могли видеть, в каком направлении движется продукт, и основывать свои решения на этих дорожных картах.

Дорожные карты продуктов также могут быть открыты в том смысле, что другие могут скорректировать план действий, предоставив больше возможностей. Типичный способ сделать это - организовать дни клиентов, на которых клиенты могут высказать свое мнение и внести свой вклад в разработку продукта, например, посредством голосования [2.29].

На операционном уровне (SPM) организации могут начать процесс разработки требований, например, путем предоставление пользователям возможности голосовать за новые предлагаемые функции. Кроме того, организации могут начать планирование выпуска, где они будут согласовывать с заинтересованными сторонами публикацию следующего выпуска. Такая координация, как правило, выгодна для заинтересованных сторон, поскольку в этом случае заинтересованные стороны могут привести свое

планирование в соответствие с планами организации.

Кроме того, организация может публиковать версии-кандидаты, чтобы заинтересованные стороны в экосистеме могли протестировать их перед выпуском новой версии.

2.1.5. Корпоративный маркетинг и продажи (M&S)

На стратегическом уровне (M&S) организация может раскрыть видение рынка и ожидания, чтобы побудить партнеров заниматься определенными областями и отбить у них охоту заниматься другими. Кроме того, организация может выбрать новые инновационные бизнес-модели для своих партнеров. Примером такой инновационной бизнес-модели является модель оплаты внутри приложения на iPhone, которая позволяет разработчикам приложений для платформы iPhone зарабатывать деньги с помощью самого приложения, а не только за счет продаж из приложения. Эти инновационные бизнес-модели открывают новые возможности для партнеров. Еще одним важным вариантом является разработка прогрессивной партнерской модели, при которой наиболее ценные и высокопрофессиональные партнеры также получают контроль над организацией.

На тактическом уровне (слияния и поглощения) организация может принять решение открыться и делиться своей рыночной информацией с целью повышения конкурентоспособности всей экосистемы. Еще одним источником информации, стимулирующим развитие экосистемы, является публикация (частично) информации о текущих и потенциальных клиентах и поставщиках в рамках экосистемы.

Наконец, организация может разработать новые каналы распространения, позволяющие партнерам продавать программное обеспечение по каналам, которые ранее были недоступны. Примером новых каналов распространения являются магазины приложений, такие как Android Market, в которых разработчики могут продавать приложения по всему миру всем

пользователям Android.

На операционном уровне (слияния и поглощения) организация может принять решение о создании внутреннего и внешнего рынков компонентов, чтобы стимулировать использование этих каналов для продаж приложений и решений, относящихся к конкретной предметной области. Кроме того, организация может поощрять партнеров организации выдают сертификаты, основанные на критериях качества.

Эти сертификаты информируют клиентов этих партнеров об уровне качества, которого они могут ожидать от этих партнеров. Это может касаться продаж и обслуживания, а также сертификации компонентов. Наконец, организация может привлекать партнеров к продаже программного обеспечения. Примером может служить визит клиента к партнеру организации, чтобы получить информацию об организации. Другой пример - предоставление партнерам стимулов для продажи основной платформы и ее компонентов.

2.1.6. Услуги по консультированию и поддержке оперирующей системы (CSS)

Почти каждый SPO в какой-то момент будет предоставлять консультации службы технической поддержки (CSS) [2.13]. Эти сервисы необходимы для того, чтобы помочь клиентам в настройке, партнерам - в развертывании продукта в первые пару раз, а разработчикам подключаемых модулей - в запуске их первой версии. Поскольку эти услуги предоставляются, организация извлекает выгоду из того, что раскрывает свои знания и делится ими с партнерами, поскольку эти знания не нужно раскрывать, когда организация предоставляет эти услуги снова. В закрытых экосистемах ключевые игроки часто ограничивают передачу знаний о внедрении, используя политику или процедуры для обеспечения ключевых компетенций [2.43]. Как показано в контексте внедрения ERP, передача знаний между

техническими консультантами и бизнес-пользователями способствует успеху внедрения [2.30].

На стратегическом уровне (CSS) организация может делиться всеми накопленными знаниями, предоставляя CSS партнерам.

Знания, собранные организацией и ее партнерами, могут принести пользу будущим проектам CSS, а централизованное хранение данных и предоставление их партнерам приносят пользу экосистеме. Более того, разработка четкой стратегии для этих процессов, например, делегирование небольших проектов партнерам, приносит пользу экосистеме программного обеспечения, поскольку партнеры получают от организации новые возможности для бизнеса.

На тактическом уровне (CSS) организация может делегировать проекты CSS партнерам. Экосистема получает от этого выгоду двумя способами: партнер может продавать CSS, а также узнавать что-то новое; платформа, предоставляемая организацией, создает ценные навыки.

Кроме того, на тактическом уровне знания CSS должны передаваться с использованием знаний о процессах проекта и баз данных заявок, предпочтительно, чтобы партнеры могли помогать друг другу и обмениваться знаниями друг с другом. Наконец, организация может принять решение о разработке показателей качества для клиентов и партнеров, позволяющих определить, на каком уровне качества обслуживания работают партнеры и что они могут сделать для повышения этого уровня.

Наконец, на операционном уровне (CSS) организации могут делиться подробными описаниями деталей проекта CSS. Одной из таких деталей может быть совместное использование пользовательской конфигурации, хотя это будет редкостью. Кроме того, организация может использовать рабочие пространства для совместной работы, в которых различные проектные организации работают вместе и обмениваются знаниями на платформе. Кроме того, организации могут проводить тренинги для консуль-

тантов, чтобы аккредитовать консультантов и заверить клиентов.

Бизнес-модель фокусируется на том, как повысить ценность организации.

Различают четыре типа внешних организаций, которые могут повысить ценность SPO. Организация открывает свое предприятие в зависимости от того, от какой из этих внешних сущностей организация хочет получить выгоду. Выделяются четыре добавляющие ценность сущности: разработчики, клиенты, VAR и сервисные партнеры.

Разработчики - это особый случай, поскольку их ценность увеличивается за счет исходного кода, а не за счет денег.

Если программное обеспечение разрабатывается в среде с открытым исходным кодом, вопросы управления требуют гораздо большего внимания, чем в закрытой организации-разработчике. Вопросы управления, которые заслуживают внимания, - это процесс разработки, роль разработчиков программного обеспечения, и владелец IP-адреса. Кроме того, это влияет на открытость области исследований и разработок: технологические и исследовательские планы и видение должны быть четко представлены сообществу, IP-адрес должен четко управляться и проверяться в исходном коде, а хранилища ошибок и отчеты о тестировании должны быть открыты.

Наконец, это влияет на управление программными продуктами, поскольку теперь требуется, чтобы все разработчики могли вносить свой вклад в планирование выпуска и требования, которые будут реализованы в будущем.

Можно выделить три различных канала распространения, косвенные - VAR, непрямой – через организацию обслуживания и прямой – к клиенту [2.35]. Следует отметить, что возможно сочетание первых двух, но это лишь незначительно относится к вариантам открытости. Первый канал распространения, косвенный – через VAR, требует открытости в области

исследований и разработок, особенно при рассмотрении API и SDK, необходимых для расширения платформы. Кроме того, в зависимости от того, какую часть дохода приносят эти торговые посредники, требуется открытость как в сфере маркетинга и продаж, так и в сфере консалтинга и поддержки.

То же самое относится и ко второму каналу дистрибуции, косвенному - через организацию обслуживания, который требует открытости как в области маркетинга и продаж, так и в области консалтинга и поддержки.

Третий канал дистрибуции, направленный непосредственно заказчику, требует открытости практически во всех областях, поскольку заказчик в значительной степени заинтересован в будущем SPO. Однако основное внимание уделяется покупке, а затем развертыванию и внедрению программного решения в организации заказчика.

2.2. Применение модели

Модель должна использоваться двумя разными сторонами, являющимися внешними и внутренними оценщиками. Внутренние эксперты, имеющие доступ ко всем заинтересованным сторонам в организации, оценивают открытость организации с целью предоставления информации о доступных и реализуемых вариантах открытости организации.

Внешний эксперт по оценке - это тот, кому необходимо знать, насколько организация открыта по сравнению с другими, например, для принятия решений о закупках. Эти внешние эксперты могут быть не в состоянии раскрыть подробную информацию о каждом из вариантов открытости, но все равно можно провести объективное сравнение организаций, если оценивать их по видимым характеристикам.

В связи с исследовательским характером работы трудно предложить строгий метод определения того, является ли опцион открытым для проведения SPO или нет. Тем не менее, мы хотели бы обратить внимание на

следующие рекомендации. Начнем с того, что открытость опции должна учитываться для продукта или платформы, которые отвечают за основную часть доходов, выручки и деятельности компании (в случае организации без дохода или с доходом, который легко поддается измерению). Во-вторых, все варианты основаны на наличии некоторой критической концепции, такой как наличие другого канала, отличного от основного канала сбыта, для “Развития каналов сбыта”. Эта важнейшая концепция обычно доступна на веб-сайте SPO, но также может быть установлена путем опроса партнеров. В третьих, за вариантом открытости должна стоять критическая масса: т.е. это не может быть простым “хвастовством” организации, но другие игроки должны быть готовы к этому. Это сотрудничество должно быть заметным и предпочтительно измеримым. Наконец, открытость предполагает, что любая организация может участвовать и сотрудничать в экосистеме. В случае, если некоторые объекты по умолчанию исключены из совместной работы, несмотря на доказанные попытки совместной работы, опция просто не открыта.

Модель применяется путем оценки для каждого варианта открытости, реализован ли он в рамках SPO. Для каждого из вариантов открытости необходимо качественно оценить, является ли он открытым или нет.

Некоторые из них легко установить, например, опцию “Поделиться технологией и планом исследований”, поскольку ее можно найти с помощью веб-поиска или связавшись с компанией. Если дорожная карта доступна только ограниченному кругу третьих лиц, таких как партнерская сеть, она не считается открытой. По словам одного из экспертов: “Платформа Apple iPhone открыта в том смысле, что вы можете разрабатывать для нее, но по их правилам, так что на самом деле она не настолько открыта”. Другие, такие как “Разработка каналов распространения”, сложнее создать, поскольку может показаться неясным, что считается каналом распространения. В рамках тематических исследований “Разработка каналов

распространения” считалась открытой, когда существовал магазин приложений или когда партнеры были вовлечены в распространение основного программного продукта. Критерии открытости или закрытости были установлены на основе описаний, представленных в этом разделе. В случае тематических исследований всегда было установлено, что мы говорим об основном продукте SPO, а не о продукте, который не обязательно отражает стратегию компании.

Например, выпуск Microsoft Wix deployment tool set с открытым исходным кодом не делает Microsoft компанией с возможностью “поделиться открытым исходным кодом”.

2.3. Взаимосвязи модели

Модель не так легко разделить, т.е. большинство параметров напрямую влияют на другие параметры. Мы смогли определить несколько типов взаимосвязей в модели, основываясь на совокупности всех зависимостей между параметрами открытости. Матрица не была представлена здесь по трем причинам: трудно реализовать зависимости, в настоящее время матрица не подверглась оценке, и это напрямую не добавляет ясности модели оперирующей системы. Однако некоторые примеры и категории зависимостей описаны ниже.

Первый обнаруженный тип связей - это горизонтальные связи "сверху вниз", в которых один из вариантов открытости на уровне руководства зависит от другого варианта открытости на более высоком уровне управления.

В этой категории содержится наибольшее количество связей. Примером таких взаимоотношений является оперативный вариант “Создание внутреннего и внешнего рынков компонентов”, на который влияет тактический вариант “Развитие каналов сбыта”, на который, в свою очередь, влияет стратегический вариант “Разработка инновационных бизнес-

моделей”. Эти связи многочисленны, что свидетельствует о сильной внутренней сплоченности модели.

Второй тип отношений называется диагональными отношениями. Эти диагональные отношения, которые, как правило, идут сверху вниз слева направо, являются отношениями, которые показывают, что варианты стратегической открытости влияют не только на их собственную практику SPO, но и на других.

Тривиальным примером таких отношений является “Вовлечение партнеров в маркетинг и продажи”, на который напрямую влияет “Создание модели партнерства”. Интересно, что некоторые из вариантов, особенно в практике управления SPO, влияют на многие другие варианты. Это варианты, на которые влияют 10 или более факторов, - “Создать модель партнерства”, “Внедрить стратегию управления знаниями экосистемы”, “Поделиться исходным кодом” и “Поделиться технологией и планом исследований”.

Некоторые из других выявленных взаимосвязей относятся к той же категории практики на том же уровне управления (в пределах одного блока на рис. 2.1). Кроме того, отсутствует влияние более низких уровней управления на более высокие уровни управления.

На одном и том же управленческом уровне существует ряд факторов, влияющих от практики к практике, таких как “Обмен знаниями о процессе разработки”, полученными в ходе исследований и практики разработки, на которые оказывает влияние “Стратегия управления знаниями об экосистемах” из "Практики управления". Наконец, все практики на двух более низких уровнях управления зависят от вариантов более высоких уровней или связаны с ними.

2.4. Анализ тематических исследований

В этом разделе представлены выводы, полученные на основе темати-

ческих исследований. Один из основных выводов этой статьи заключается в том, что бизнес-модель и стратегия определяют степень открытости SPO. В табл. 2.2 предполагается, что наиболее открытые домены связаны с типами участников (сторонними разработчиками, реселлерами, сервисными партнерами и клиентами), которых организация хочет стимулировать. Например, можно видеть, что когда SPO хочет, чтобы разработчики добавляли ценность организации, оно должно открыть раздел "управление", "исследования и разработки" и "программный продукт". Распространение продукта определяет, требуются ли знания в области обслуживания и внедрения. Например, в компании SAP, одном из крупнейших в мире поставщиков услуг по планированию ресурсов предприятия, существует широкая сеть партнеров, которые занимаются внедрением своих сложных конфигурируемых продуктов. С другой стороны, для такого продукта, как Eclipse, который, как правило, работает "из коробки", требуется меньшая открытость в области развертывания и обслуживания.

Таблица 2.2

Оценка открытости в SPO (GX = GX Software)

Группа	**S/T/O**	**Открытость**	**ODA**	**ECL**	**GX**
Управление	S	Приоритет – задачи управления	+	+	
	S	Создание модели партнерства	+	+	+
	S	Разработка стратегии IP	+	+	
	S	Координирует вклад в другие экосистемы		+	
	S	Проводит политику в области конкуренции	+		
	S	Проводит стратегию приобретения доли			
	S	Внедряет стратегию управления знаниями об экосистемах	+	+	+
	T	Обеспечивает соблюдение стандартов процесса разработки X		+	
	T	Предоставляет партнерам процедуры управления		+	
	T	Согласовывает жалобы	+	+	

Группа	S/T/O	Открытость	ODA	ECL	GX
	T	Помогает партнерам в конфликтах интеллектуальной собственности	+		
	O	Четко определяет экосистему	+	+	+
	O	Создает каталог партнеров		+	+
	O	Создает группы пользователей		+	+
	O	Использует и создает многоразовые лицензии на программное обеспечение		+	
Исследования и разработки	S	Делится технологиями и планами исследований	+	+	+
	S	Делится знаниями о процессе разработки		+	+
	S	Поощряет открытые стандарты	+	+	+
	S	Делится исходным кодом	+	+	
	S	Заявки на совместное финансирование исследований и разработок		+	
	T	Делится инновациями		+	+
	T	Поддерживает взаимозаменяемые форматы данных	+	+	+
	T	Политика совместного использования исходного кода	+	+	+
	T	Создает политику повторного использования	+	+	+
	T	Задачи аутсорсинга			
	T	Сертифицирует компоненты сторонних производителей			+
	O	Создание и публикация (контента). API и SDK-пакеты	+	+	+
	O	Создание архитектуры, обеспечивающей повторное использование	+	+	+
	O	Запуск процесса тестирования		+	
	O	Делится репозиторием ошибок	+	+	+
	O	Занимается совместным развитием		+	+
	O	Обеспечивает обучение разработчиков		+	+
	O	Распространяет знания о работе программного обеспечения			
Управление программными продуктами	S	Делится планами жизненного цикла продуктов	+		

Группа	S/T/O	Открытость	ODA	ECL	GX
	S	Делится стратегией и видением платформы	+	+	+
	T	Передает разработку требований на аутсорсинг партнерам	+	+	
	T	Совместно использует и корректирует планы разработки продукта (линейки продуктов)		+	+
	T	Управляет интеллектуальной собственностью	+	+	+
	O	Открытый процесс управления требованиями	+	+	+
	O	Открытый процесс планирования выпуска		+	+
	O	Информирование о кандидатах на выпуск		+	
Консультационные и вспомогательные службы	S	Совместно использует стратегию управления предоставлением услуг			
	T	Передача проектов внедрения на аутсорсинг партнерам			+
	T	Общий доступ к базе прецедентов			+
	T	Общий доступ к знаниям о проектном процессе	+		+
	T	Разрабатывает и распространяет показатели качества			+
	O	Общий доступ к знаниям о внедрении			+
	O	Общий доступ к знаниям о конфигурации (с клиентами)			
	O	Использует рабочие пространства для совместной работы для общения с клиентами			
	O	Проводит обучение консультантов			+

Пример EF заставляет предположить, что в разработке с открытым исходным кодом сообществом управление играет самую большую роль. Если программное обеспечение если бы управление не находилось в руках самих членов EF, эти члены не захотели бы вкладывать свои деньги или

свое время в экосистему Eclipse. Кроме того, тот факт, что платформа Eclipse выпущена с открытым исходным кодом, побуждает разработчиков новых компонентов также открывать свои компоненты, создавая живую экосистему, в которую разработчики с удовольствием вносят свой вклад из-за прибыли, которую они получают от этой экосистемы.

Таблица 2.3

Бизнес-модель и участники, создающие добавленную стоимость

	Управление	**R&D**	**SPM**	**M&S**	**CSS**
Разработчики	x	x	x		
Управляющие организации	x	x	x	x	x
Сервисные организации				x	x
Клиенты	x	x	x		x

Этот пример показывает, что в закрытом консорциуме разработка является основным видом деятельности. Консорциум ожидает повышения эффективности своей деятельности, ценность должна заключаться в основном продукте консорциума (программном обеспечении). Управление находится в руках небольшого числа участников консорциума с открытым исходным кодом, а не какой-либо другой стороны, поэтому эти участники ожидают получить максимальную отдачу от консорциума. Это приводит к снижению накладных расходов на организацию.

При изучении трех организаций было замечено, что программные экосистемы создаются путем открытости.

В случае с GX Software, компания приступила к внедрению, настройке и кастомизации своего основного продукта. Благодаря ODA было открыто управление разработкой важного компонента для многих разработчиков программного обеспечения САПР, что позволило им контролировать дальнейшее развитие экосистемы.

В случае с Eclipse исходный код был открыт для поддержания критической массы разработчиков и постепенного усиления влияния на более

дорогие коммерческие альтернативы. Одно из наиболее показательных наблюдений заключается в том, что некоторые из наиболее коммерческих организаций, таких как, например, Apple и Microsoft, хотя обычно считаются крайне закрытыми организациями, были вынуждены открыть большую часть своей деятельности из их доменов M&S и CSS. Не совсем верно, что эти организации являются "закрытыми" по сути, но они пытались сохранить как можно больший контроль, не раскрывая свою модель управления. Таким образом, для индустрии программного обеспечения вредна не "закрытость", а уровень контроля.

Хотя открытость организации, как правило, считается преимуществом для разработчиков программного обеспечения, открытость не всегда приносит пользу организации. В примере с GX Software было потрачено много усилий на то, чтобы сделать организацию более открытой. Предполагалось, что отказ от части интеллектуальной собственности, такой как знания консультантов, увеличит критическую массу платформы GX Software, тем самым увеличив оборот. Однако, несмотря на то, что эти знания были доступны, GX Software не сразу начала приносить больше денег. Казалось, что сначала необходимо набрать критическую массу экосистемы программного обеспечения. Этот период роста до критической массы сопряжен с рисками, которые должны учитываться организациями, пытающимися следовать одной и той же стратегии.

Менеджер экосистемы отвечает за поддержание различных участников экосистемы, чтобы они были удовлетворены и могли разрабатывать новые политики и стратегии, основанные на стремлении организации к открытости.

Роль менеджера экосистемы актуальна во всех областях SPO, как показывает модель OSE. Роль партнерства и менеджера экосистемы в SPO возрастает. Во всех трех объектах исследования были четко определены роль и обязанности партнеров. Управляющий партнерством или экосисте-

мой, как правило, является сотрудником отдела исследований и разработок или слияний и поглощений.

Ниже приводятся два других вывода. Приведем два примера - RedHat Linux и MySQL.

Red Hat - это публично зарегистрированная компания, которая добилась успеха в получении прибыли от использования открытого исходного кода и наиболее известна своей операционной системой Red Hat Linux. Red Hat получает прибыль, продавая услуги, а не лицензии, такие как поддержка клиентов на основе подписки, тренинги и обеспечение качества программного обеспечения, для (частично) открытых программных продуктов. Интересно, что Red Hat использует разные лицензии для разных продуктов. Некоторые продукты имеют полностью открытый исходный код, в то время как другие продукты (и даже версии) имеют более закрытые лицензии. В этих более закрытых лицензиях обычно указывается, что исходный код доступен только подписчикам определенных служб поддержки.

Компания постоянно экспериментирует с открытостью и вносит коррективы в бизнес-модель. Red Hat является примером компании, которая по-прежнему в значительной степени полагается на третьи стороны из сообщества разработчиков с открытым исходным кодом (хотя все чаще и на своих собственных клиентов) в поставках новых компонентов и продуктов для решения узкоспециализированных задач.

Red Hat никогда не смогла бы решить проблему самостоятельно. Red Hat служит главным примером SPO, который основан на мире открытого исходного кода, но успешно приносит прибыль в закрытой отрасли.

Другим примером может служить продукт MySQL, который может обеспечить уровень базы данных в любом программном продукте. MySQL, недавно Oracle, продается с использованием мультилицензирования, т.е. один и тот же исходный код распространяется с разными лицензиями на разных условиях. Например, продукт с открытым исходным кодом может

свободно использовать MySQL, в то время как коммерческий продукт с закрытым исходным кодом может повторно использовать и перепродавать MySQL только по коммерческой лицензии. Концепция, при которой несколько лицензий используются для разных групп клиентов, также известна как сегрегация рынка. Часть дохода MySQL поступает от сервисов, как в примере с Red Hat.

Лицензии не определяют само по себе членство

В таблице 2.4 перечислены различные типы участников, которые были явно определены в трех случаях. Например, GX Software в этом контексте выделяет четыре различных типа участников: сервисных партнеров, разработчиков программного обеспечения, разработчиков компонентов и клиентов.

Таблица 2.4

Идентификация участников экосистемы и их лицензии

Организации	**Выявленная добавленная стоимость**	**Тип участника**	**Взносы**	**Использование**	**Перепродажа**	**Разработчик**	**Доступ к коду**
GX	Сервисные партнеры	Сервисные партнеры	+		+		
GX	Посредники с высокой добавленной стоимостью.	Посредники с высокой добавленной стоимостью.	+		+	+	
GX	Разработчики компо-	Разработчики	-	+		+	

Организации	**Выявленная добавленная стоимость**	**Тип участника**	**Взносы**	**Использование**	**Перепродажа**	**Разработчик**	**Доступ к коду**
	нентов						
GX	Клиенты	Клиенты	+	+			
EF	Ассоциированные члены	Клиенты	--	+			+
EF	Члены группы "Решения"	Посредники с высокой добавленной стоимостью.	-	+	+		+
EF	Предприятия	Клиенты	+	+			+
EF	Стратегические партнеры	Клиенты	++	+			+
EF	Члены комитета	Разработчики		+			+
ODA	Образовательные участники	Посредники с высокой добавленной стоимостью, клиенты	--			+	
ODA	Ассоциированные члены	Клиенты	-	+		+	
ODA	Коммерческие	Посредники с высокой	+-	+	+	+	

Организации	**Выявленная добавленная стоимость**	**Тип участника**	**Взносы**	**Использование**	**Перепродажа**	**Разработчик**	**Доступ к коду**
	участники	добавленной стоимостью					
ODA	Поддерживающие участники	Посредники с высокой добавленной стоимостью	+	+	+	+	
ODA	Основатели	Посредники с высокой добавленной стоимостью	++	+	+	+	+

Клиенты, поставщики услуг и сервисные партнеры повышают ценность программного обеспечения GX, оплачивая лицензии. Разработчики повышают ценность GX Software, создавая новые компоненты для программной платформы GX.

Еще одно интересное наблюдение заключается в том, что разные участники экосистемы, такие как EF enterprise и стратегические участники, платят разные суммы денег за одни и те же права на исходный код [2.46]. Однако следует отметить, что разница в членстве заключается не только в лицензиях, но и в других стимулах: корпоративный участник просто платит взносы, в то время как стратегический участник может заплатить еще большие взносы, предоставив разработчику Eclipse часы работы вместо денег. Кроме того, стратегические участники размещаются на специальной

веб-странице стратегических участников, среди которых такие компании, как SAP и IBM.

2.5. Результаты и выводы

2.5.1. Результаты

Модель OSE может играть роль справочной системы при определении степени открытости SPO.

Необходимо сделать некоторые оговорки в отношении различных уровней открытости для каждого варианта открытости, сферы охвата модели и факторов, способствующих открытию SPO. Кроме того, необходимо четко обозначить некоторые угрозы для законности.

При попытке определить, готова ли организация к использованию определенных вариантов, иногда бывает трудно провести грань между чисто открытыми и закрытыми. Например, в случае ODA исходный код доступен только членам альянса, в то время как в случае EF исходный код полностью открыт. С похожим примером, когда было трудно провести грань между открытым и закрытым, столкнулись, пытаясь установить, является ли программное обеспечение GX с его вкладом в один компонент Apache с открытым исходным кодом таковым

GX Software “координирует вклад в другие экосистемы”. Требуется больше исследований для определения степени открытости конкретных опций и их влияния на бизнес-модель. Это важный шаг.

Результатом циклов проектирования стала модель оперирующей системы. Были завершены два цикла проектирования, в ходе которых в первом цикле модель была проверена экспертами экосистемы программного обеспечения в ходе интервью, а во втором цикле модель была проверена путем проведения трех тематических исследований.

Одним из самых значительных изменений, внесенных в модель, стало добавление области управления после обсуждений с экспертами.

Для получения достоверных результатов в ходе тематических исследований были применены руководящие принципы исследования конкретных примеров [2.40].

Стратегическое управление SPO - это сложная задача, требующая большого опыта, дальновидности и предпринимательской смелости. Когда речь заходит об открытости, это, безусловно, так. Непосредственные выгоды от открытия бизнеса можно получить только экспериментальным путем, а оценить фактические выгоды после открытия определенных вариантов непросто. В конечном счете, состояние экосистемы определяет, готова ли она к дальнейшей открытости и, как следствие, к дальнейшему росту. В области исследований экосистем программного обеспечения были бы использованы дополнительные методы оценки [2.15], которые определяют зрелость и готовность к открытости экосистемы.

Предметом обсуждения является вопрос о том, можно ли обобщить эти результаты в виде какой-либо оценки или степени. Можно представить себе внедрение программного обеспечения, определяющего степень открытости (SPOOD). Такой анализ может состоять из пяти компонентов, представляющих собой пять областей в модели OSE: управление, исследования и разработки, SPM, M&S и CSS, где для каждого из компонентов рассчитывается взвешенный балл OSE.

Для каждого из доменов D={g, r, s, m, c} можно рассчитать показатель открытости для компании, где каждый из доменов может в равной степени увеличить итоговый показатель открытости, т.е.

$|O_g|=|O_r|=|O_s|=|O_m|=|O_c|=20$

Однако для создания и валидации SPOOD требуется больше данных, чтобы создать модель с более высокой точностью.

2.5.2. Выводы к главе 2

1. Предложена модель организации с открытым программным ко-

дом (OSE), отличающаяся табличным учетом каждого варианта открытости и учетом специфики предметной области продукта на основе горизонтальных или диагональных связей и обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации. Первый обнаруженный тип связей - это горизонтальные связи "сверху вниз", в которых один из вариантов открытости на уровне руководства зависит от другого варианта открытости на более высоком уровне управления. Второй тип отношений называется диагональными отношениями. Эти диагональные отношения, которые, как правило, идут сверху вниз слева направо, являются отношениями, которые показывают, что варианты стратегической открытости влияют не только на их собственную практику SPO, но и на других.

2. Установлено, что модель OSE дает представление о вариантах открытости и может помочь поставщикам услуг в определении их стратегий открытости. Модель OSE также показывает, что организация может выбрать открытость как со стороны предложения, так и со стороны спроса в цепочке поставок.

3. Показано, что степень открытости полностью определяется бизнес-моделью, разработанной лицами, принимающими решения в рамках SPO.

4. Открытость может быстро создать критическую массу разработчиков или партнеров вокруг продукта, если и только если окружающие партнеры готовы войти в экосистему и занять одну из четырех ролей: разработчика, VAR, партнера по обслуживанию или заказчика. Наконец, необходимым условием для динамично развивающейся экосистемы является открытая платформа.

5. Показано, что решение об открытии или закрытии многогранно и не может быть принято без тщательного изучения. С дальнейшей разработкой шкалы открытости покупатели программного обеспечения смогут

честно и недвусмысленно оценивать открытость продуктов-кандидатов, тем самым предоставляя им инструмент (модель) для объективной оценки степени открытости организации. С другой стороны, модель жизненного цикла, которая определяет, готово ли сообщество, окружающее платформу или продукт, к переходу к экосистеме, выгодна для SPO: она позволит им определить, когда открываться и какие варианты использовать.

3. МУЛЬТИФОРМАЛЬНЫЙ ПОДХОД К МОДЕЛИРОВАНИЮ СЦЕНАРИЕВ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ CLOUDSIM

3.1. Подход к моделированию сценариев облачных вычислений

Реализация процесса моделирования производительности сложной системы, такой как сценарий облачных вычислений, является довольно сложной задачей. Например, поставщик облачных услуг должен управлять общими политиками для большего числа центров обработки данных и множества сервисов на разных уровнях абстракции.

Во-первых, необходимы экспертные знания в области оценки эффективности, чтобы обеспечить предсказуемость последствий выбора конструкции.

Во-вторых, знание облачных архитектур необходимо для того, чтобы внести свой вклад в развитие предметной области. И, наконец, опыт в управлении облаками и администрировании обязателен для рассмотрения организационной и макроскопической перспективы, а также связанных с этим системно-ориентированных соображений.

Учитывая структурную и динамическую сложность системы, состоящей потенциально из тысяч и тысяч многоядерных процессоров, вычислительных узлов, высоконагруженной сети и, возможно, миллионов и миллионов запланированных вычислительных задач, жизнеспособный подход к управлению сложностью заключается в использовании различных моделей в процессе моделирования. Скрывая детали реализации, можно постепенно перейти к более реалистичным моделям, чтобы определить ту, которая наиболее близка к реальной системе, подлежащей оценке. Такой процесс также позволяет изменить метод оценки (например, инструменты, основанные на анализе или моделировании) и перекрестной проверке результатов между этапами. В то время как модель первого прибли-

жения реализуется общими методами (например, чисто математическими инструментами, такими как автоматы или дифференциальные уравнения, или общими инструментами моделирования, такими как сети Петри или сети массового обслуживания), более конкретные модели должны содержать больше деталей, чтобы включить в цикл специализированные симуляторы. Пока процесс моделирования приближается к завершающей стадии, для управления моделью и ее внедрения требуются различные навыки, которые редко можно найти у одного разработчика моделей, с возможными последующими несоответствиями, приводящими к аномалиям оценки. В середине процесса должна быть доступна система бережливого управления, позволяющая беспрепятственно осуществлять передачу полномочий между разработчиками моделей с различными навыками, обеспечивая средства для постепенной передачи обязанностей в соответствии с процессом. Автоматическая трансформация модели [3.12, 3.13, 3.15, 3.16, 3.35, 3.37], включая автоматическую генерацию, является технологией, отвечающей этой потребности.

Чтобы дать представление об общей методологии, рассмотрим рис. 3.1. Методология направлена на разработку облачных приложений с учетом проблем с производительностью с самого начала цикла проектирования.

Этот подход основан на процессе моделирования, при котором разрабатываются модели, состоящие из двух (или двух наборов) подмоделей: первая подмодель представляет приложение, вторая - используемую облачную инфраструктуру. В то время как первое представляет собой нечто, что полностью контролируется разработчиком, о втором ничего не известно с ранних этапов процесса проектирования, и оно становится частично известным, когда выбирается окончательная облачная платформа, поскольку физические детали реализации в любом случае недоступны. На первом этапе обе подмодели могут быть смоделированы с помощью ана-

литических методов, таких как сети Петри или более специфичный язык, адаптированный к конкретной предметной области, для получения первого приближения оценки производительности и поддержки выбора дизайна. Впоследствии подмодели могут быть доработаны путем добавления деталей: об окончательной реализации приложения, используя полные знания о нем; о гипотетической целевой архитектуре для облачной инфраструктуры, чтобы экспериментировать с различными конфигурациями.

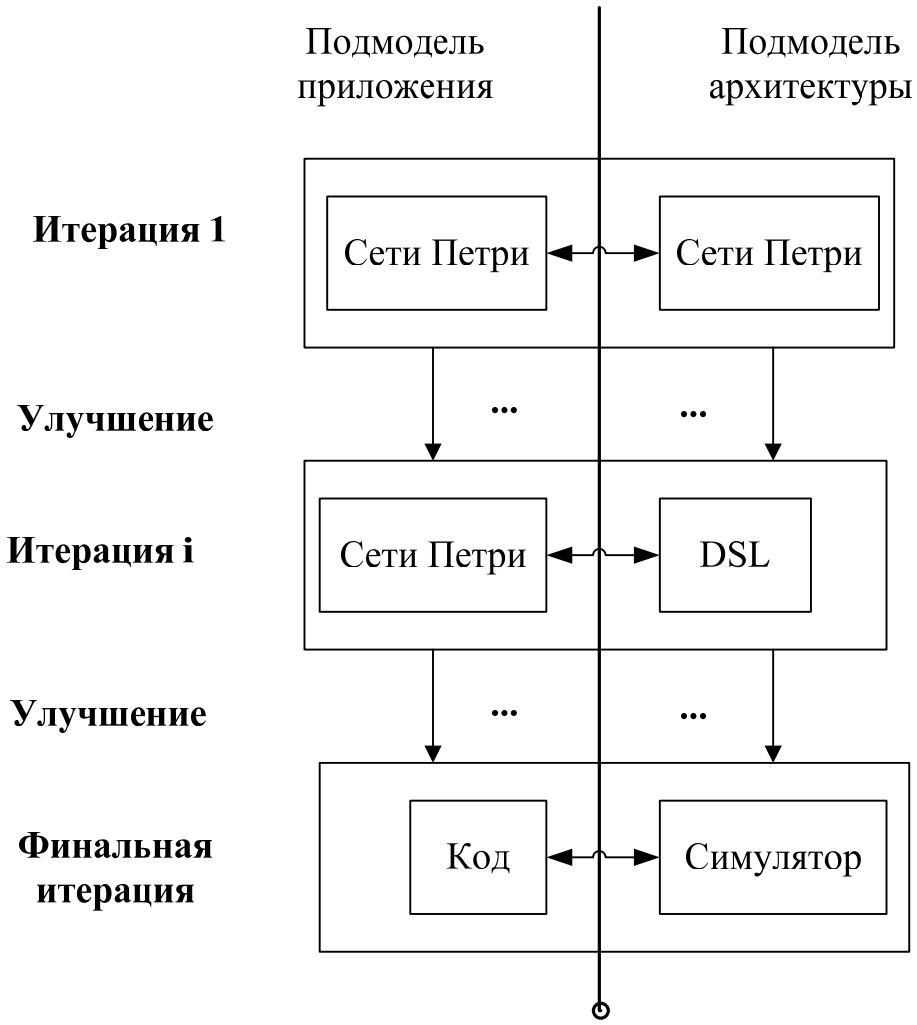


Рис. 3.1. Эволюция модели на протяжении нескольких итераций применения методологии

Следовательно, вся модель разрабатывается до тех пор, пока приложение не будет закодировано и запущено в имитационной облачной среде

для окончательной оценки. Промежуточный этап - это этап, на котором приложение представлено подмоделью сети Петри, имитирующей конечный алгоритм, а подмодель архитектуры представлена языком моделирования [3.14], который точно имитирует структуры облачного симулятора, прежде чем будет написан и доработан фактический код приложения, чтобы получить более точное представление о производительности и сравнить влияние различных архитектур и параметров. На этой промежуточной итерации процесса моделирования оценка производительности выполняется не аналитически, а с помощью моделирования, которое использует существующий облачный симулятор и сеть Петри для моделирования общей модели.

3.2. Формализмы

Определены два формализмы и их состав, включая все необходимые элементы для создания базовой модели CloudSim. Формализмы были разработаны в соответствии с методологией синтеза [3.26]. В этой структуре формализмы описываются элементами и ребрами, каждое из которых обозначается набором свойств (учитывающих параметры, внутреннее состояние и переменные, необходимые для оценки производительности) и поведение (описывающих, как элементы и ребра взаимодействуют для определения общей семантики модели), описывающих формализм и которые могут быть использованы чтобы построить график, представляющий модель. Ребра связывают два элемента и ведут себя соответственно определенной семантике.

Вышеупомянутые формализмы называются, соответственно, GSPN (Обобщенные стохастические сети Петри [3.36]) и новый DSL, названный CDL (Cloud(Sim) Description Language).

3.2.1. Определение формализма GSPN

Формализм SIMTHESys GSPN является простым расширением формализма SPN, представленного в [3.2], с техническими отличиями. Это простая реализация формализма, представленного в [3.36]. На рис. 3.2 все элементы и ребра, описывающие GSPN, показаны в правом поле: позиция, временной переход и немедленный переход - это элементы, в то время как ребро и задержка (не используемые в данном примере) представляют ребра. Позиция обладает свойством маркировки, описывающим количество токенов, включенных в данный момент.

Рис. 3.2. Элементы GSPN, CDL и формализмы композиции: 1 – центр обработки данных; 2 – брокер; 3 – хост; 4 – виртуальная машина; 5 – память; 6 – cloudlet; 7 - ребро планирования пуска; 8 – ребро соединения; 9 – ребро рабочей нагрузки

Временной переход имеет свойство rate, определяющее скорость его срабатывания; мгновенный переход имеет свойство веса, описывающее его относительный вес (в случае, если два мгновенных перехода активированы с помощью отметки общего исходного места, используемой для стохастического определения того, какой из них выстрелит из двух). Ребро обладает свойством множественности, определяющим, сколько токенов требуется в исходном месте для обеспечения перехода в пункт назначения или

создается в месте назначения при запуске перехода. Наконец, задержка обладает свойством множественности, определяющим, сколько токенов в исходном месте необходимо для предотвращения запуска целевого перехода.

Все элементы всех формализмов имеют свойство Name, которое в дальнейшем будет опущено.

3.2.2. Определение формализма CDL

Элементы формализма CDL показаны слева на рис. 3.2. Элементы были выбраны таким образом, чтобы абстрактно представлять все основные концепции, необходимые для создания модели CloudSim.

Общая идея состоит в том, чтобы описать:

1) как организован центр обработки данных с точки зрения его физических и виртуальных машин с помощью подхода, основанного на композиции,

2) визуально распределить рабочие нагрузки с желаемыми характеристиками для брокера, работающего в центре обработки данных. Элементы формализма описаны ниже.

Центр обработки данных

Он представляет объект центра обработки данных CloudSim. Основными свойствами являются:

1) пользователи, количество пользователей, предназначенных для использования;

2) набор параметров вычислительных средств, в частности номенклатура локальных средств обработки данных в составе ЦОД;

3) os, платформа операционной среды;

4) vmm, механизм виртуализации;

5) географическое расположение ЦОД в части часового пояса;

6) затраты на обслуживание – стоимость обработки в ЦОД;

7) стоимость памяти, удельные затраты на использование памяти;

8) стоимость хранения, определяющая удельные затраты на использование хранилища;

9) стоимость полосы пропускания - удельная стоимость использования полосы пропускания сети;

10) политика распределения виртуальных машин, определяющая, как виртуальные машины распределяются между узлами в ЦОД.

CloudSim определяет как стандартные параметры вышеуказанные свойства. Далее, Bind arc - это ребро, которое соединяет ЦОД с одним или несколькими узлами или элементами хранилища, или брокера с одним или несколькими элементами виртуальной машины. У него есть свойство instances, которое определяет, сколько экземпляров узла или элемента хранилища составляют физическую архитектуру центра обработки данных или сколько элементов виртуальной машины запрашивается элементом-посредником для запроса выполнения рабочей нагрузки. Семантика дуги привязки состоит в предоставлении сведений и измерений, необходимых для создания кода, используемого для создания внутренних структур данных моделирования CloudSim, поэтому она никоим образом не изменяет состояние связанных элементов после инициализации модели.

Хост

Он обозначает тип физической машины, которая будет создана в центре обработки данных. Свойства хоста. Основными свойствами являются mips, описывающий, сколько миллионов команд в секунду может выполнить ядро процессора; core – степень ядерности исполнительного устройства; core provisioner как описатель модельного интерфейса CloudSim; cpus – степень процессорности хоста; ram – объем ОЗУ, ram provisioner как описатель модельного интерфейса CloudSim; storage - определяет объем доступного хранилища в МБ; bandwidth - пропускную способность сетевого соединения в Мбит/с; bandwidth provisioner - определяет, какая модель подготовки CloudSim должна использоваться для обеспече-

ния пропускной способности; vms cheduler - определяет, какой планировщик CloudSim выбран для решения задачи управления запасами между ВМ. Гетерогенный ЦОД задает топологию связей хостов с компонентами ЦОД.

Память

Память задает один из вариантов способа и физического механизма управления данными для интеграции в ЦОД. Основными свойствами хранилища являются тип, который описывает его технологию, емкость, пропускную способность и задержку в сети (применимо только для сетевых дисков, таких как SAN).

Брокер

Это касается процесса выполнения рабочей нагрузки в облаке. Рабочая нагрузка задается в виде элементов Cloudlet, соединенных ребрами рабочей нагрузки, которые должны выполняться на виртуальных машинах, указанных в элементах виртуальной машины, соединенных ребрами привязки. Брокер запускается в ЦООД в зависимости от параметров расписания исполнения arc edge. Computesuccessful есть состояние брокера.

Представлена упрощенная версия формализма, представляющего моделирование CloudSim на основе стандартных классов. CDL изначально разработан для расширения, как и все формализмы SIMTHESys, и включает в себя новые функции, которые специалист по CloudSim может реализовать с помощью новых классов.

Все ядра считаются равными, даже если CloudSim допускает разные значения. Это считается реалистичным упрощением, поскольку оно велико. Небольшие архитектуры нечасто встречаются в обычных облаках, и единственным заметным контрпримером является описание хоста, в значительной степени использующего графические процессоры для обеспечения вычислительной мощности.

Опять же, все ядра считаются идентичными и используют одну и ту

же схему подготовки; значение по умолчанию - ”simple”, соответствующее классу PeProvisionerSimple в CloudSim.

В настоящее время CloudSim имеет два класса, представленных в формализме значениями ”HD” или ”SAN”..

Cloudlet

Он представляет собой тип рабочей нагрузки, которую должен выполнять брокер. Свойства Cloudlet. Основными атрибутами являются length, указывающий количество инструкций, необходимых для его выполнения; inputfilesize, обозначающий размер файла в байтах перед выполнением; outputfilesize, отображающий размер файла в байтах после выполнения; cpunumber, указывающий количество вычислительных блоков, необходимых для выполнения; cpuutilizationmodel, ramutilizationmodel и bandwidthutilizationmodel, обозначает, какая модель использования CloudSim принята для использования виртуальных процессоров, оперативной памяти и пропускной способности сети.

Ребро рабочей нагрузки

Назначает Cloudlet Брокеру в соответствии с его свойством множественности. Cloudlet может быть подключен к нескольким элементам Брокера, каждый из которых выполняет различную рабочую нагрузку с параметрами, определенными Cloudlet; несколько элементов Cloudlet подключаются к одному брокеру для определения составной рабочей нагрузки.

Виртуальная машина

Обозначает тип виртуальной машины, доступной для обработки в центре обработки данных. Свойства виртуальной машины. Основными свойствами являются imagesize, описывающий объем МБ, занимаемый изображением, ram, указывающий объем виртуальной оперативной памяти в МБ, cpus, указывающий количество доступных виртуальных процессоров; mips, определяющий мощность виртуального процессора; bandwidth, обозначающий ограничение пропускной способности; vmm, описывающий

используемую технологию виртуализации; cloudletscheduler, определяющий политику планирования для cloudlet, выполняемого этой виртуальной машиной; пользователь, представляющий его владельца.

Планировщик запуска ребра

Определяет, в каком центре обработки данных выполняется облачный сервер. Каждый облачный сервер подключен к одному центру обработки данных.

3.2.3. Определение формализма композиции

Формализм композиции очень прост и определяет только три граничных элемента, соединяющих элементы, принадлежащие двум другим формализмам. Таким образом, они называются гибридными ребрами, или, в более общем смысле, гибридными элементами в терминологии синтеза. Запрос на выполнение arc может подключать только переход GSPN к брокеру, и его семантика соответствует разрешению создания экземпляра рабочей нагрузки и попытке ее выполнения в центре обработки данных. Успешное выполнение arc и неудачное выполнение arc подключают брокера к месту GSPN и выдают токен в этом месте, если выполнение было завершено без проблем или если выполнение было прекращено из-за проблемы.

3.3. Генерация симулятора

На рис. 3.3 показан процесс, необходимый для создания симулятора для сценария, описанного в модели. Мультиформальная модель анализируется соответственно с точки зрения

1) описания PN,

2) структуры CDL,

3) гибридных элементов, соединяющих элементы из первых двух компонентов.

Генератор рабочей нагрузки/стохастических компонентов анализи-

рует рабочую нагрузку, генерирует код для соответствующих облачных наборов CloudSim и соответствующие им распределения вероятностей. Облачные наборы - это самые элементарные единицы рабочей нагрузки, которые можно запланировать в CloudSim, отправив их в облачный планировщик. Поскольку облачный кластер характеризуется параметром, обозначающим его детерминированный общий объем вычислительных требований в терминах ”линий”, каждый облачный кластер многократно создается во время выполнения моделирования с различными значениями линий в соответствии с распределением вероятностей, как описано в следующем подразделе.

Генератор архитектурного кода CloudSim анализирует CDL-описание архитектуры и генерирует экземпляр полной модели CloudSim для каждого центра обработки данных, включенного в модель, используя библиотеку-заглушку. Поскольку CloudSim легко настраивается, для этой работы было использовано подмножество только значимых параметров без потери общности. Такой подход позволяет легко включать расширения в модели, предназначенные для реализации специальных характеристик или дополнительных элементов (например, компоненты, имитирующие конкретную облачную реализацию, такие как OpenStack, или дополнительные облачные планировщики), просто расширяя коллекцию заглушек и привязку между параметрами модели и новыми заглушками.

Генератор симуляторов анализирует гибридные границы, которые связывают описания рабочей нагрузки и облачных элементов, создавая код, необходимый для правильного создания экземпляра и планирования рабочей нагрузки и архитектуры. Поскольку созданный симулятор полностью основан на стандартных примитивах CloudSim, он выдает те же выходные результаты, которые затем собираются во время выполнения запрошенных запусков для получения статистики, и те же события, используемые для управления симуляцией.

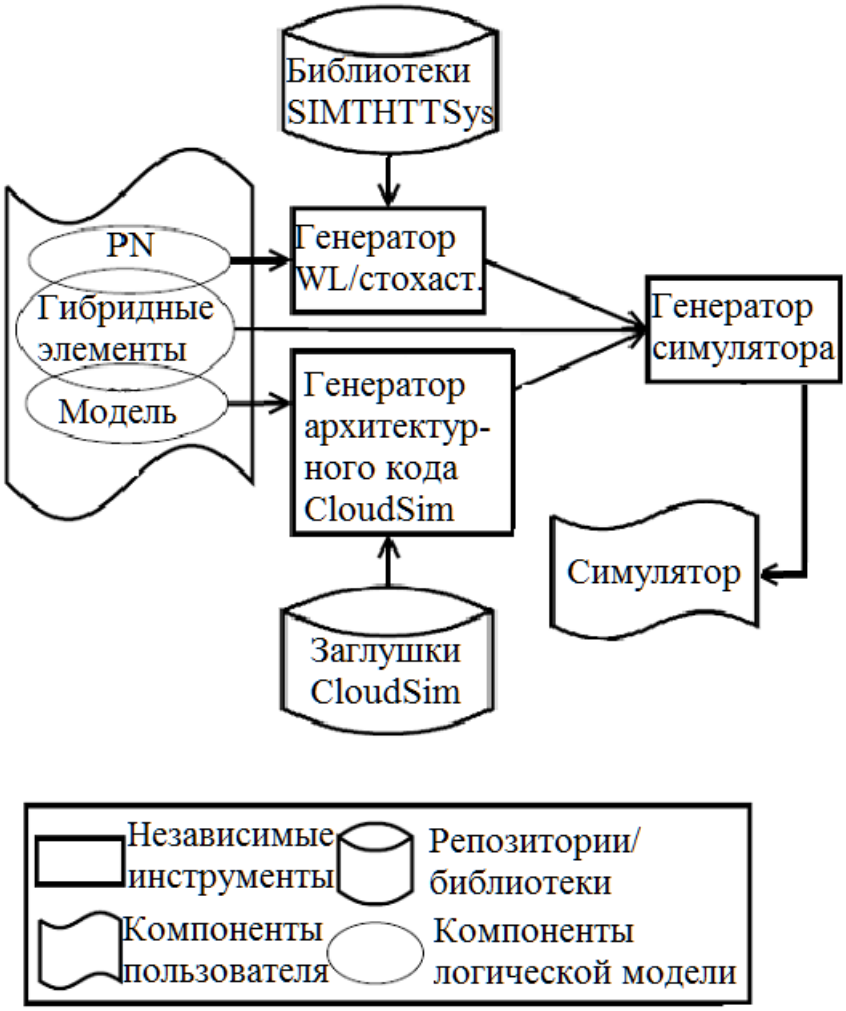


Рис. 3.3. Процесс генерации симулятора

3.3. Управление имитацией

Созданный симулятор основан на разработке пользовательского брокера CloudSim, а именно GlobalBroker, который отвечает за обработку всей симуляции. GlobalBroker содержит структуры данных, определяющие подмодель GSPN, и логику обработки событий GSPN и CloudSim, что позволяет моделированию развиваться.

После настройки начального состояния GSPN, разрешив все немедленные разрешенные переходы, начиная с начальной отметки и до тех пор, пока немедленный переход больше не будет разрешен, моделирование начинает обработку событий. Эволюция получается путем обработки событий CloudSim. Некоторые из этих событий особенно важны для понимания

того, как управляется планирование в созданном симуляторе.

Когда происходит событие SIMULATION_EVENT, GSPN эволюционирует, запуская все разрешенные переходы. Если переход является отправной точкой arc-запроса на выполнение, соответствующий брокер CloudSim генерируется и планируется в соответствии со свойствами соответствующего брокера с его виртуальной машиной и Cloudlet. Это событие также явно генерируется при обработке включенного временного перехода, чтобы учесть его влияние на моделируемое время и оценить переходные процессы из-за наличия исчезающих состояний. Поскольку время запуска для такого рода переходов генерируется случайным образом, это добавляет случайности в эволюцию.

Когда возникает ошибка BROKER_FAIL, это означает, что брокер CloudSim не был запущен должным образом. Следовательно, если связанный брокер является источником ошибки исполнения, то маркировка места назначения обновляется надлежащим образом.

Когда происходит BROKER_SHUTDOWN, он сообщает о завершении работы брокера CloudSim, так что, если связанный брокер является отправной точкой для успешного выполнения, обновляется отметка места назначения.

CLOUDLET_START и CLOUDLET_END выполняют внутренние операции CloudSim, необходимые для обработки запуска и завершения Cloudlet. Генерация облачков происходит в соответствии с параметрами задействованного облачка, что расширяет стандартное поведение CloudSim, предоставляя возможность создания облачков со случайным распределением длин в зависимости от свойства length.

На рис. 3.4 показан общий процесс моделирования с помощью диаграммы последовательности UML.

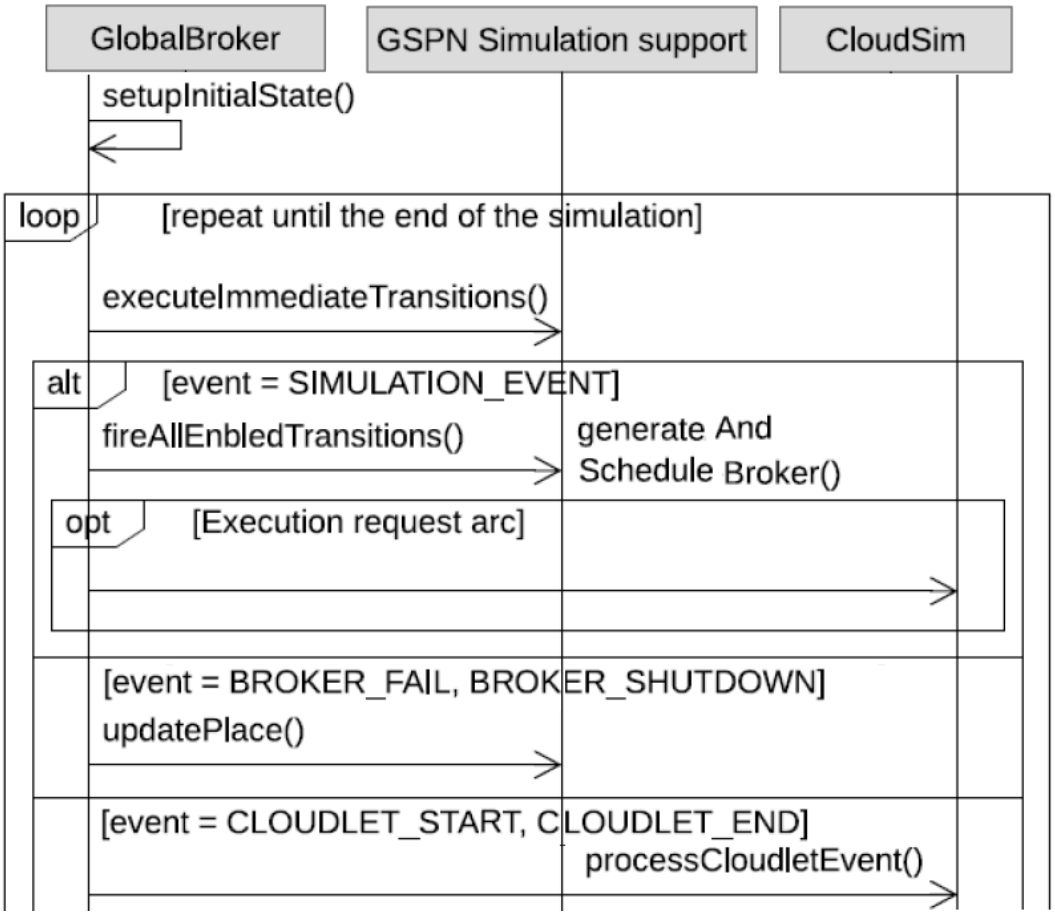


Рис. 3.4. Диаграмма последовательности выполнения процесса моделирования

3.4. Исследование пограничных вычислений

Чтобы продемонстрировать эффективность этого подхода, приведен простой пример, представляющий сценарий пограничных вычислений.

3.4.1. Исходные данные

Приложение в режиме реального времени частично выполняется в облаке на стохастической основе. Выходные данные приложения имеют смысл только в том случае, если они были созданы правильно и вовремя, поэтому ошибки (например, из-за невозможности выделить ресурсы, необходимые для выполнения облачных приложений, отправленных брокеру) не переносятся на выполнение. Несмотря на это, количество ошибок явля-

ется важным показателем, который необходимо оценить, чтобы понять поведение приложения. Одно и то же приложение выполняется циклически, с N параллельными экземплярами, M из которых могут выполняться одновременно в облаке, при M≤N. Каждое выполнение состоит из предварительной (вычислительной) фазы, которая выполняется на месте (с вероятностью p) или в облаке (с вероятностью 1-p), а затем переходит в фазу ожидания. Первоначально предполагалось, что все фазы будут обозначены экспоненциально распределенными длительностями, с разными значениями для граничной фазы и фазы облачных вычислений, хотя для фазы облачных вычислений требуется более точная оценка. Следовательно, оценка выполняется с помощью моделирования, в то время как фаза вычислений описывается в соответствии с фактической используемой облачной конфигурацией.

Приложение смоделировано так, как показано на рис. 3.5. Временной переход T0 представляет фазу ожидания, T1 - предварительную фазу и T5 - фазу локальных вычислений. Элемент broker B1 моделирует фазу выполнения в облаке. Временные переходы T2 и T3 представляют собой вероятностный выбор между двумя альтернативами на этапе вычисления, путем присвоения веса p T2 и веса 1-p T3. Место P1 изначально подготавливает все N параллельных запусков к предварительной фазе, а отметка места P5 устанавливает ограничение на M одновременных исполняемых рабочих нагрузок в облаке.

Узлы h1s и h2s настроены с параметрами H1и H2. B1 формирует рабочие нагрузки cls cloudlet C1. Этот же B1 требует BM, у которых параметры сформированы по параметрам VM1. Cls cloudlet C1 отрабатывают в ЦОД DC1, который состоит из узлов h1s, настроенных в соответствии с параметрами H1, и узлов h2s, настроенных в соответствии с параметрами H2. При успешном выполнении токен вставляется на место P4, в то время как ошибка при выполнении приводит к появлению токена на месте P7,

что в конечном итоге приводит к немедленному запуску перехода T7. Все синхронизированные переходы выполняются с использованием одного сервера. Таблица 3.3 определяет кратность дуг на рис. 3.5; В таблице 3.3 определены базовые значения, присвоенные параметрам модели для того, что касается подмодели сети Петри; параметры для VM1 определены в таблице 3.5; параметры для C1 определены в таблице 3.6; параметры для H1 и H2 определены в таблице 3.7; параметры для DC1 определены в таблице 3.8.

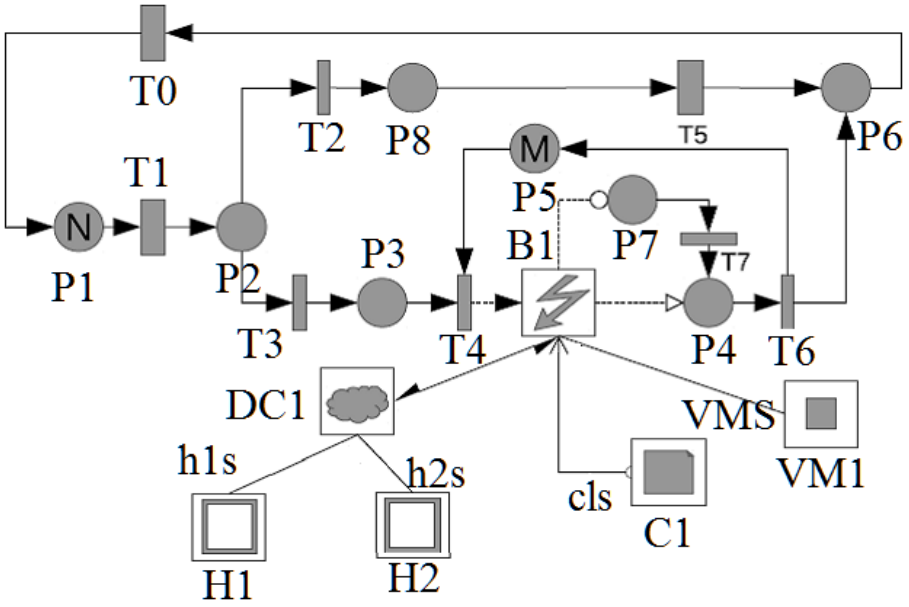


Рис. 3.5. Модель, отражающая объект исследования

Таблица 3.3

Базовые параметры модели: кратность дуги

cls (количество cloudlet)	10
vms (количество виртуальных машин на одного брокера)	1…10
h1s (количество хостов типа H1)	1
h2s (количество хостов типа H2)	1

3.4.2. Эксперимент

Моделирование было повторено 30 раз для повышения точности и получения 95%-х доверительных интервалов. Каждый цикл моделирова-

ния длился от 25000 до 500000 с во время моделирования, плюс еще 10000 с были отброшены в начале и в конце, чтобы устранить начальные и конечные переходные процессы. Эксперимент повторили, варьируя количество виртуальных машин на одного брокера, vms=[1...10].

На рис. 3.6 показано среднее время отклика для выполнения одного cloudlet и для всего рабочего процесса (на рисунке обозначено как ”Broker R.T.”). Первый набор показателей получен путем усреднения информации, возвращаемой Cloudsim, описывающей время выполнения каждого облачного пакета. Второй определяется с применением закона Литтла к компоненту сети Петри, определяющему выполнение одного задания (табл. 3.4):

$$R_{\text{Broker}} = \frac{N - E[P1]}{XT1} \tag{3.1}$$

где E[P1] и XT1 обозначают соответственно среднюю отметку места P1 и пропускную способность перехода T1.

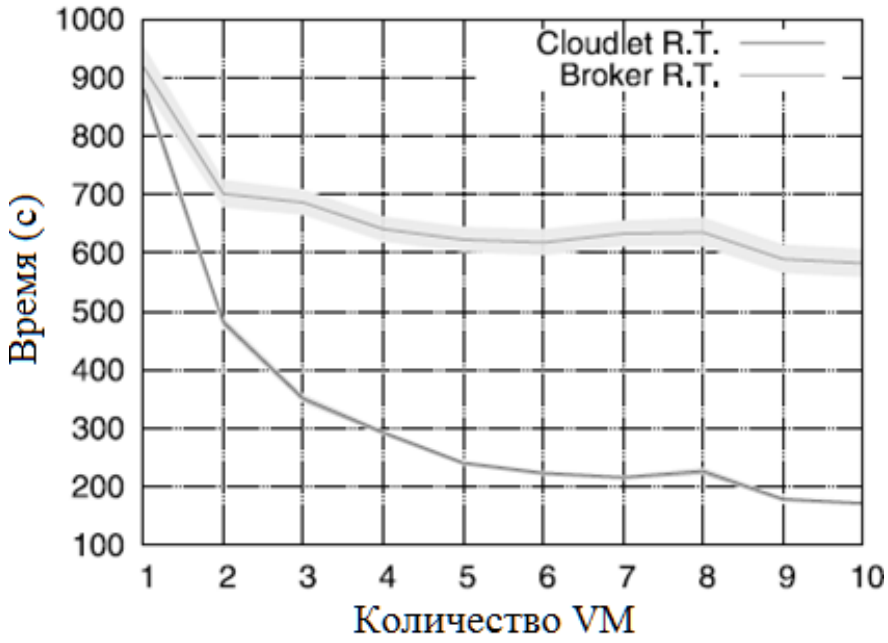


Рис. 3.6. Среднее время отклика при изменении количества виртуальных машин для каждого брокера

Как было отмечено в [3.24], основное сокращение времени вычислений происходит при уменьшении максимального количества облачков, на-

значенных для каждой виртуальной машины. Это событие происходит с дискретным интервалом: например, в этом сценарии с N=10 заметны значительные улучшения в работе виртуальных машин {1, 2, 4, 5, 10}. Интересно отметить, что, однако, между двумя значительными улучшениями время отклика, как правило, увеличивается из-за плохого использования ресурсов.

Таблица 3.4

Базовые параметры модели: компонент сети Петри

Среднее время срабатывания T0	50 s
Среднее время срабатывания T1	100 s
Вес T2	p=0.7
Вес T3	1-p=0.3
Вес T4, T6, T7	1
Среднее время срабатывания T5e	200 s
Число экземпляров N	5
Количество в облаке M	3

На рис. 3.7 показана пропускная способность различных переходов. Как следует из модели, пропускная способность сгруппирована в три группы: переходы T0 и T1 представляют собой пропускную способность системы и имеют наибольшее значение. Переходы T2 и T5 учитывают задания, выполняемые локально, в то время как переходы T3, T4 и T6 учитывают задания, выполняемые брокером в облаке. Как и ожидалось, между тремя группами существует тесная взаимосвязь, например:

$$X_{T1} = X_{T3} + X_{T3} \quad (3.2)$$

Что касается пропускной способности, то локальные максимумы кривых еще более заметны, чем время отклика. Например, мы можем утверждать, что vms=6 обеспечит лучшие результаты, чем vms=8, и что было бы целесообразно иметь больше, чем vms=6, только если мы можем позволить себе хотя бы vms=9. С другой стороны, если vms≥9, распределение некоторых виртуальных машин начинает сбоить, и работа некоторых бро-

керов прервана (запуск перехода T7). Таким образом, этот анализ подтверждает, что либо мы переезжаем в более крупный центр обработки данных, способный предоставить больше виртуальных машин, либо ограничиваем нашу конфигурацию значением vms=6 виртуальных машин.

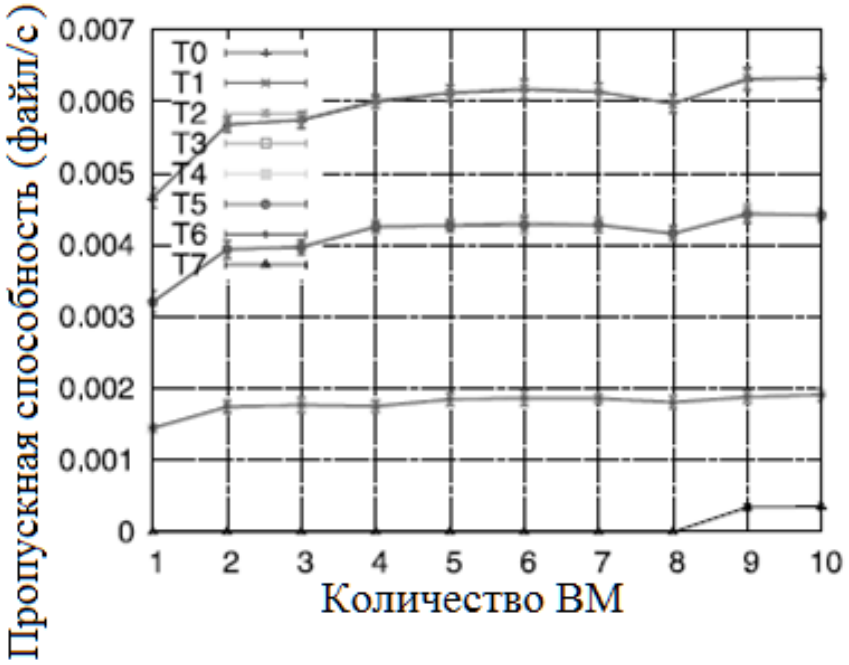


Рис. 3.7. Пропускная способность переходов при изменении количества виртуальных машин для каждого брокера

Таблица 3.5

Базовые параметры модели: виртуальные машины

Размер образа	10000 MB
RAM	512 MB
MIPS	250
Пропускная способность	1000 Mbps
Количество CPU	1
Технология виртуализации	Xen

Средняя маркировка мест, которые имеют ненулевое количество жетонов в течение неисчезающего времени, показана на рис. 3.8. Обратите внимание, что из-за мгновенных переходов места P2 и P4 остаются отмеченными только в течение бесконечно малого промежутка времени, что

приводит к нулевому среднему значению.

Таблица 3.6

Базовые параметры модели: cloudlet

(средний) Размер	40000 строк
Размер входного файла	300 MB
Размер выходного файла	300 MB
Количество CPU	1
Модель использования	full

Таблица 3.7

Базовые параметры модели: host

Хранилище	100000
RAM	16384 MB
MIPS	1000
Пропускная способность	10000 Mbps
Количество CPU	1
Количество ядер	4 (тип 1), 2 (тип 2)
Инструмент подготовки RAM	Simple
Инструмент подготовки пропускной способности	Simple
Инструмент подготовки ядер	Simple
Планировщик виртуальных машин	Разделенный по времени

Таблица 3.8

Базовые параметры модели: datacenter

Пользователи	2
Архитектура	x86
ОС	Linux
VMm	Xen
Цена обработки	3.0
Цена памяти	0.05
Цена накопителей	0.1
Цена пропускной способности	0.1
Инструмент подготовки RAM	simple
Политика распределения виртуальных машин	simple

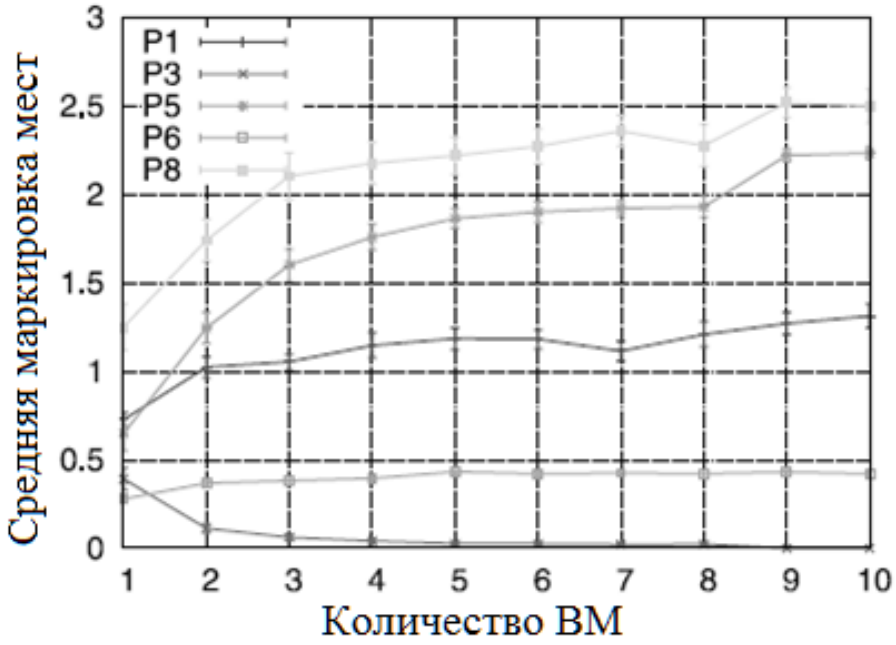


Рис. 3.8. Средняя маркировка мест, занятых во время неразрушающей маркировки, при различном количестве виртуальных машин на брокера

Интересно отметить, что единственная уменьшающаяся отметка относится к позиции P3, которая представляет задания, необходимые для выполнения в облаке, которые ожидают допуска в удаленную инфраструктуру. Это происходит потому, что с увеличением количества виртуальных машин среднее время отклика облачных узлов уменьшается, что снижает вероятность ожидания входа в облако. Место с наибольшей средней оценкой - P8, из-за длительности локальной обработки, представленной переходом T5.

Из облачной части модели можно рассчитать конкретные показатели производительности, как показано на рис. 3.9. В частности, мы можем рассчитать среднее количество брокеров, работающих в системе, количество облачных сервисов и количество виртуальных машин, назначенных двум хостам центра обработки данных. В этом сценарии интересно посмотреть, как неудачное распределение брокера повлияет на среднее количество за-

пускаемых облачных приложений. Когда используется одна виртуальная машина, то хост H1 вполне способен обрабатывать все брокерские услуги, а на втором хосте H2 виртуальная машина не запускается.

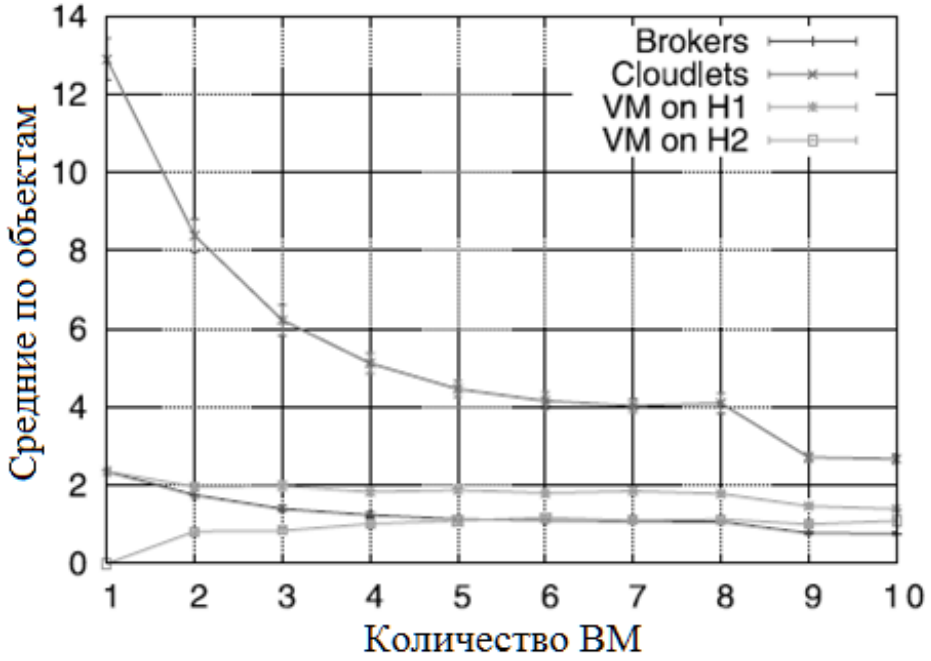
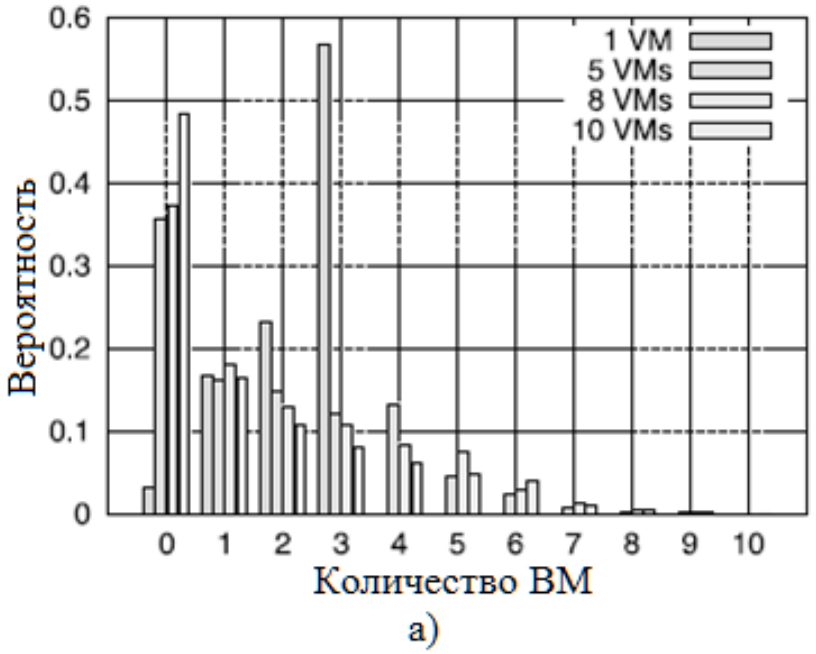


Рис. 3.9. Среднее количество брокеров, облачных сервисов и виртуальных машин, выделенных для каждого типа хоста, в зависимости от количества виртуальных машин для каждого брокера

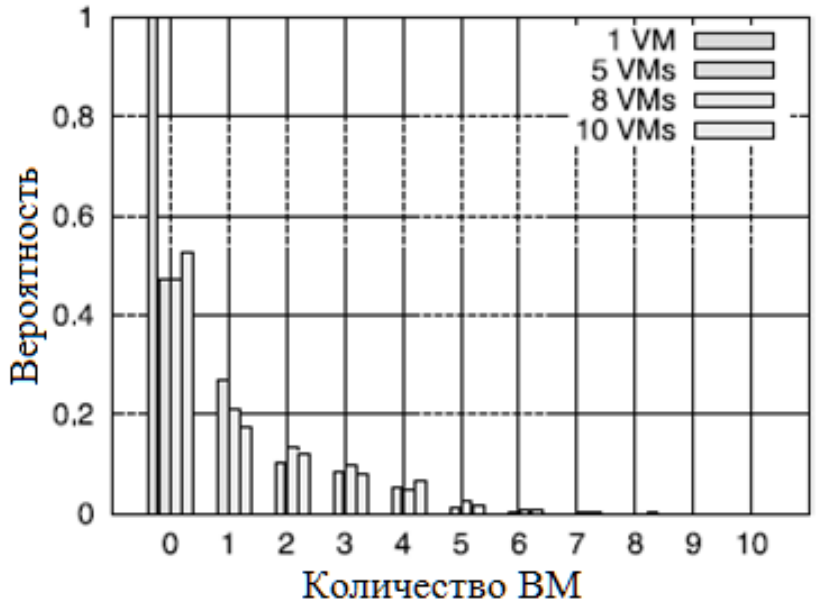
Использование моделирования позволяет также вычислять более сложные результаты, такие как распределение количества виртуальных машин, выделяемых на каждый хост (рис. 3.10). В частности, можно видеть, что, за исключением случая, когда используется одна виртуальная машина для каждого брокера, второй хост используется более или менее одинаково независимо от конфигурации системы.

3.4.3. Результаты

В этом подразделе кратко излагаются (табл. 3.9) основные наблюдения, полученные в ходе эксперимента.



а)



б)

Рис. 3.10. Распределение количества виртуальных машин, выделенных для каждого типа хоста, для 1, 5, 8 и 10 виртуальных машин для каждого брокера: а) Хост H1 (четырехъядерный), б) Хост H2 (двухъядерный).

Таблица 3.9

Результаты

Ссылка	**Примечания**
Рис. 3.6	Между двумя отмеченными точками улучшения время отклика

Ссылка	Примечания
	увеличивается из-за плохого использования имеющихся ресурсов
Рис. 3.7	Необходимо либо рассмотреть возможность создания более крупного центра обработки данных, способного предоставить больше виртуальных машин, либо ограничить текущую конфигурацию
Рис. 3.8	По мере увеличения числа виртуальных машин и уменьшения среднего времени отклика облачных модулей вероятность ожидания входа в облако уменьшается
Рис. 3.9	Происходит сбой при распределении брокера. Когда используется одна виртуальная машина, хост H1 вполне способен обрабатывать все брокеры, а на втором хосте H2 виртуальная машина не запускается
Рис. 3.10	Второй хост используется более или менее одинаково, независимо от конфигурации системы

3.5. Выводы к главе 3

1. Определен мультиформальный подход, позволяющий интегрировать ситуационный подход к высокоуровневому моделированию с известным симулятором облачных архитектур для получения более реалистичных результатов в процессе анализа производительности. Этот подход призван заполнить пробел между предварительными (аналитическими) моделями анализа производительности и подробными имитационными моделями на основе поэтапного моделирования.

2. Подход основан на процессе моделирования, при котором разрабатываются модели, состоящие из двух (или двух наборов) подмоделей: первая подмодель представляет приложение, вторая - используемую облачную инфраструктуру. В то время как первое представляет собой нечто, что полностью контролируется разработчиком, о втором ничего не известно с ранних этапов процесса проектирования, и оно становится частично известным, когда выбирается окончательная облачная платформа.

3. Имеются некоторые ограничения. В частности, CloudSim предлагает характеристики, которые не полностью используются в модели GSPN,

что ограничивает ее использование. Другой аспект касается мультиформальной системы, которая может быть расширена за счет включения сетей массового обслуживания и деревьев отказов, а также соображений, касающихся надежности и безопасности. Кроме того, облачная рабочая нагрузка должна также включать локальную обработку. Наконец, выбор тематического исследования был обусловлен целью смоделировать как локальные, так и облачные взаимодействия в сценарии передовых вычислений. Выбор параметров модели был обусловлен, главным образом, необходимостью:

1) изучения набора специфических характеристик, используемых для моделирования локальных взаимодействий, где использование модели явно выгодно

2) рассмотрения аспектов проектирования модели и относительного цикла разработки, а не оптимальности моделирования.

4. Будущие направления работы включают, помимо изучения вышеупомянутых ограничений, определение комплексной методологии моделирования и дальнейшую проверку представленного решения на конкретных примерах более высокого уровня, основанных на реальных крупномасштабных приложениях и трассировках облачных архитектур, а также уточнение уровня интеграции предлагаемого решения в структуре моделирования SIMTHESys.

4. ОБЛАЧНОЕ УПРАВЛЕНИЕ ДОСТУПНОСТЬЮ И ПРОИЗВОДИТЕЛЬНОСТЬЮ ИНФОРМАЦИОННО-КОММУНИКАЦИОННОЙ ПОДСИСТЕМЫ

4.1. Облачное управление оперативной идентификацией отказов информационно-коммуникационной подсистемы

Для решения проблем, связанных с тем, что традиционный метод имеет длительное время отклика на мониторинг перегрузки сети связи, а эффект обнаружения не идеален, предлагается метод мониторинга в реальном времени, основанный на облачных вычислениях для блокировки сети связи. Во-первых, устанавливается точка мониторинга сети связи, и приемник завершает процесс сбора коммуникационных данных. На основе собранных данных выполняется постоянный расчет трафика для определения наличия аварийного состояния блокировки в канале сети связи и определения точного местоположения точки блокировки. Таким образом, информация генерирует тревожное сообщение для получения результатов мониторинга. Экспериментально проанализированы время работы в режиме реального времени и точность метода мониторинга. Установлено, что метод мониторинга позволяет контролировать время задержки в пределах 0,2 с, а частота ошибок мониторинга является низкой.

4.1.1. Особенности идентификации отказов при работе коммуникационной сети

В процессе построения коммуникационной сети (КС) часто возникают различные помехи, которые влияют на производительность сети; если проблема помех не устранена, то оптимизации сети будет трудно выполнить. Среди проблем, связанных с помехами, проблема блокировки помех является серьезной. Если она не будет решена, построение сети ста-

нет невозможным. Блокирование помех заключается в том, что при одновременном поступлении в приемник сигнала сильной помехи и полезного сигнала нелинейные компоненты канала связи приемника будут насыщены, что приведет к нелинейным искажениям и блокировке приемника, который выходит за пределы рабочего диапазона усилителя и микшера, что делает приемник неспособным к демодуляции как правило, это создает помехи в работе приемника, что приводит к невозможности нормально сообщать о нижнем уровне шума в сети связи. При слишком сильном сигнале использование полного сигнала также приведет к амплитудному сжатию и блокировке. Основной причиной блокировки является нелинейность устройства, особенно многоступенчатые эффекты интермодуляции. В то же время ограничение динамического диапазона приемника также приведет к возникновению помех. Блокировка приведет к сбою в работе приемника, а длительная блокировка может также привести к необратимому снижению производительности приемника.

Для подтверждения необходимо выполнить следующие действия:

Сдвиг частоты. В соответствии с принципом блокирования помех, для радиочастотного терминала, тип фильтра которого - фильтр if, сильный сигнал помех может быть исключен из приема радиочастотных сигналов путем сдвига частоты (изменения центральной частотной точки приема радиочастотных сигналов), так что уровень сигнала, попадающего на прием радиочастотных сигналов, составляет менее -40 дбм. Если RTWP сети связи снижен, можно определить, что сеть помех заблокирована.

Ослабление сигнала VGA. Для одномодовых станций, которые, как предполагается, подвержены помехам, если степень помех не очень велика, можно использовать ослабление сигнала VGA для определения того, подавлена ли сеть связи из-за возможности подавления сигнала помех. Если в процессе ослабления сигнала VGA RTWP сети связи резко меняется, это означает, что ячейка заблокирована.

Волновая ловушка. Режекторный фильтр (полосовой ограничивающий фильтр) может быть настроен таким образом, чтобы в разумной степени ослаблять сильный сигнал помех в соответствии с реальной ситуацией на объекте, так что общий принимаемый сигнал при радиочастотном приеме будет ниже порогового значения блокировки помех, что позволит определить, заблокирована ли сеть с нарушенным сигналом.

Отключение источника помех. Этот метод является самым простым. Если RTWP сети возвращается в нормальное состояние после обнаружения предполагаемого источника помех (сигнал помех не попадает в принимающую беспроводную сеть), это может свидетельствовать о том, что сеть связи заблокирована.

В разделе представлена технология облачных вычислений, позволяющая осуществлять постоянный мониторинг перегрузки коммуникационной сети.

Сеть использует физические каналы связи для соединения изолированных рабочих станций или хостов с целью формирования канала передачи данных для совместного использования ресурсов и коммуникации [4.1]. Однако с усложнением информации в сети связи и увеличением объема данных в сети связи возникают некоторые сетевые проблемы, такие как перегрузка сети [4.2]. Производительность передачи данных по сети снижается из-за ограниченных ресурсов узлов хранения и пересылки.

Что касается архитектуры Интернета, то возникновение перегрузки является неотъемлемым атрибутом. Однако, если условие блокировки имеет определенную длительность, то при исчерпании места в кэше маршрутизатор отбрасывает пакет только для того, чтобы сеть могла избежать состояния блокировки. В большой среде облачных вычислений, чтобы поддерживать нормальную работу сети и избегать негативного воздействия перегрузки на сеть, необходимо принять некоторые контрмеры для поддержания нормального режима работы сети связи.

4.1.2. Разработка метода мониторинга блокировки сети в режиме реального времени

Установка точки мониторинга сети

На определенном узле КС необходимо установить устройство сетевого мониторинга, чтобы получить данные о параметрах производительности всех каналов, связанных с этим узлом. Необходимо учитывать, на каких узлах установлены устройства сетевого мониторинга, и можно получить данные о производительности всех каналов и включить мониторинг.

Несколько устройств сетевого мониторинга могут быть разделены на несколько областей сетевого мониторинга. Одно устройство сетевого мониторинга может логически принадлежать нескольким областям сетевого мониторинга, то есть нескольким службам мониторинга производительности сети для нескольких пользователей, что позволяет сократить количество устройств и снизить затраты на проектирование [4.3]. Это требует разумного расположения точек сетевого мониторинга и рационального разделения зон сетевого мониторинга и управления ими. Чтобы полностью реализовать функцию точки мониторинга, ее структура разделена на часть локальной вычислительной сети, серверную систему, различные рабочие станции, сервер терминалов, преобразователь протоколов и удаленное сетевое устройство. Структура точки мониторинга показана на рис. 4.1.

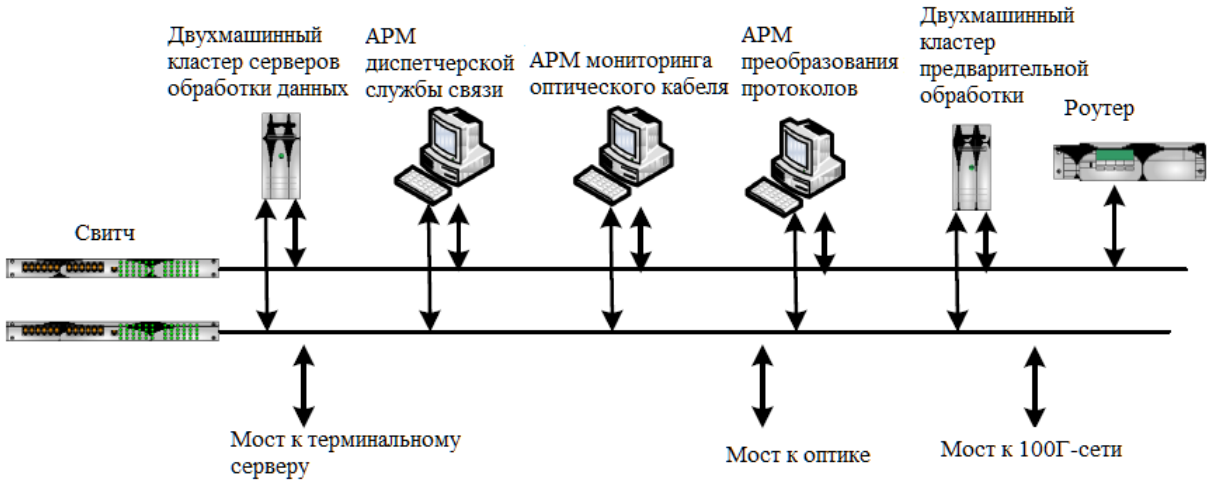


Рис. 4.1. Принципиальная схема структуры точки мониторинга

Точка мониторинга оснащена двумя сетевыми коммутаторами. Каждый сервер и рабочая станция оснащены двумя сетевыми интерфейсами, что гарантирует работу в случае отказа любого сетевого коммутатора. Настроены два сетевых сегмента, внутренней сети и внешней сети, с помощью функции моста в программе Network Switch, чтобы гарантировать, что точки доступа внутреннего и внешнего сетевых сегментов не будут создавать помех друг другу. Функция подключения к удаленной локальной сети реализуется через маршрутизатор. Кластер серверов с двумя главными серверами, устанавливает связь между двумя серверами и переводит оба сервера в режим работы кластера. Эти два сервера являются резервными копиями данных друг друга, и система гарантирует, что они не будут обнаружены ни на одном из серверов. После сбоя работа системы не нарушается.

Добавлены высокопроизводительные периферийные устройства ввода-вывода, такие как лазерные принтеры, для совместного использования с онлайн-пользователями. Данные с каждой подстанции передаются на центральную станцию через мост. Каждый мост оснащен двойным сетевым портом для обеспечения соединения с RC [4.4]. В то же время предусмотрен двойной порт WAN для реализации двойного канала E1, основного и резервного, и обеспечивается автоматическое переключение двух каналов. Кроме того, требуется процессор преобразования протоколов для ввода данных из различных подсистем мониторинга связи по различным протоколам.

Радиочастотный приемник сбора данных о КС

Сбор и хранение данных являются основой для мониторинга производительности, позволяя получать и сохранять необработанную информацию. Радиочастотный канал оснащен регулируемым цифровым аттенюатором и VGA-интерфейсом, обеспечивающими достаточную динамику для

выполнения требований по блокировке. Однако, если блокирующий сигнал находится далеко от рабочей частоты, он может находиться в непосредственной близости от АЦП или другой полосы дискретизации Найквиста и быть отобран АЦП [4.5]. Если частота помех выбрана и попадает в диапазон "использовать полный сигнал", что приводит к искажению сигнала, радиочастотный и цифровой фильтры не подавляют сигнал. В этом случае необходим фильтр промежуточной частоты, чтобы предотвратить цифровую выборку нежелательных сигналов. Структура таблицы данных мониторинга производительности показана на рис. 4.2.

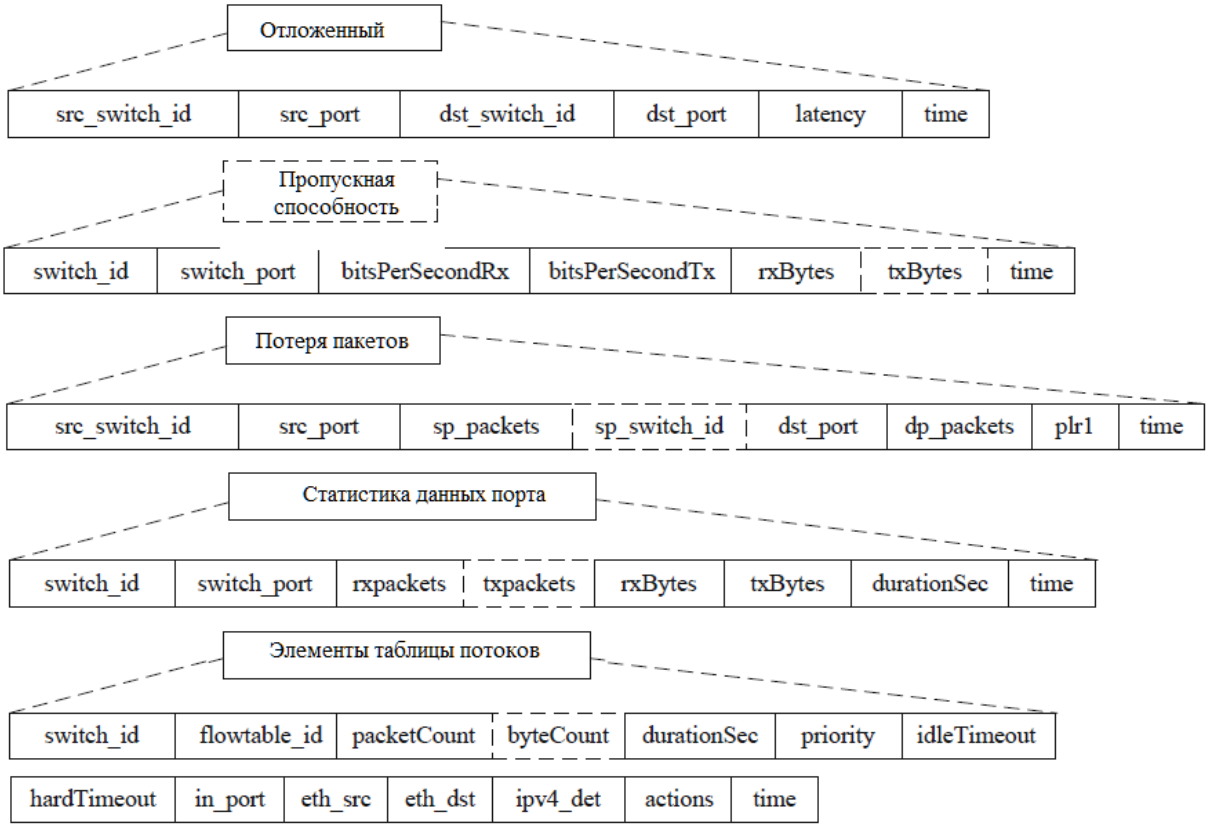


Рис. 4.2. Схема цепочки сбора коммуникационных данных

Создание вышеуказанной таблицы и подключение к базе данных в контроллере осуществляется с помощью интерфейса JDBC. В сочетании со скоростью, дешевизной и открытым исходным кодом базы данных MySQL, использование JDBC в качестве интерфейса MySQL стало распространенным явлением. JDBC - это стандартный API для языка Java для

взаимодействия с базами данных. Приложения Java могут напрямую обращаться к базе данных через этот стандартный интерфейс. JDBC также поддерживает возможность единообразного доступа к различным типам баз данных. Создайте поток сбора данных о производительности в режиме реального времени и сохраните собранные данные. Модуль взаимосвязи между реализацией сбора и хранения данных в основном разделен на три пакета реализации: пакет подключения к базе данных CLOS.sql, основная функция которого заключается в реализации функций подключения и управления базой данных; пакет Java Bean colSto.bean в основном определяет структуру данных различных таблиц данных о производительности. Пакет бизнес-операций cloSto.op в основном выполняет функции приема и хранения данных. Когда модуль мониторинга производительности получает параметры производительности, он вызывает соответствующий класс в пакете бизнес-операций. Операции по хранению данных для обеспечения распределенного хранения собранных данных в режиме реального времени.

4.1.3. Расчет данных в сети связи в режиме реального времени

Трафик данных, собранный в сети связи, рассчитывается в режиме реального времени, и вычислительная структура в режиме реального времени сравнивается с максимальным объемом памяти и максимальной нагрузкой на пропускную способность, и проверяется, может ли трафик данных плавно достигать указанного объема памяти через полосу пропускания для хранения данных [4.6]. Расчетная формула для расчета потока данных в сети связи в режиме реального времени выглядит следующим образом:

$$Q = \min\left\{ \sum_{k\in I_l} \frac{F_l}{\sum\limits_{k\in I_l} P_{lk} y_{lk} - F_l} + G \sum_{\substack{l\in L \\ k\in I_l}} \left(S_{lk} y_{lk} + d_i m_{lk} \right) + V \sum_{\substack{l\in L \\ k\in I_l}} \left(C_{lk} F_{y_{lk}} \right) \right\} \qquad (4.1)$$

где F_l - локальный трафик данных по каналу КС;

Q – общий трафик данных;

I_l - набор индикаторов модели потенциального соединения для первой связи;

P_{lk} - индекс модели первого звена, выбранного в качестве пропускной способности связи k;

S_{lk} - возможный набор маршрутов для узла первой связи;

C_{lk} - коэффициент линейной переменной при выборе индекса модели первого звена;

d_i – длина первой связи;

m_{lk} - соединитель;

I_l - скорость поступления пакетов;

G, V – фиксированный весовой коэффициент;

L – множество всех связей КС.

Ограничения:

$$\sum_{k \in S_p} x_p = 1 \; \forall p \in \Pi \tag{4.2}$$

$$\sum_{k \in I_l} y_{lk} = 1 \; \forall l \in L \tag{4.3}$$

где P - набор всех пар узлов связи в сети;

x_p - переменная оптимизации, равна 1, когда маршрут выбран, иначе 0;

y_{lk} - переменная оптимизации. Когда выбран индекс модели первого канала «ask», значение равно 1, в противном случае оно равно 0; может быть получен коммуникационный трафик в реальном времени по определенному каналу, а также могут быть получены максимальный предел и пропускная способность дискового пространства. Для определения расхода потока в нормальных условиях сравнивается максимальный предел производительности.

4.1.4. Распознавание аномальных блокировок и анализ характеристик

Помимо того, что трафик данных в КС, превышающий максимальную пропускную способность КС, может привести к перегрузке сети, аномалии передачи данных также являются другой причиной перегрузки. Чтобы осуществлять мониторинг аномальных узлов передачи данных в режиме реального времени, необходимо автоматическое инспектирование узлов. Программа мониторинга узла считывает IP-адрес устройства из базы данных и циклически отправляет команду проверки связи. Полученный результат фиксируется в рабочем состояние устройства узла. Если время ожидания результата истекло, это означает, что устройство узла работает неправильно. Как только сетевой трафик станет ненормальным, IP-адрес и распределение портов изменятся. Если произойдет ошибка конфигурации сети, исходный IP-адрес и IP-адрес назначения увеличатся, что приведет к резкому увеличению количества пакетов, отправляемых хостом. В соответствии с этой особенностью для анализа дисперсии характеристик распределения трафика используется метод матрицы сетевого трафика. Предположим, что характеристика трафика есть A, общее количество выборок есть B, количество выбранных выборок есть C, а количество повторений конкретной характеристики трафика i есть n_i. Таким образом, выборка характеристик трафика может быть осуществлена следующим образом:

$$F(x) = -\sum_{i=1}^{C}\left(\frac{n_i}{B}\right)\log_2\left(\frac{n_i}{B}\right) \tag{4.4}$$

Если все выборки дают одинаковый результат, то F(x)=0; если все выбранные выборки имеют большую степень разброса, то $F(x)=\log_2 C$ может описывать аномальное поведение различных характеристик потока, а затем выполнять обработку захвата пакетов.

Когда система мониторинга трафика фиксирует передаваемый кадр Ethernet, ей необходимо сначала проанализировать пакет данных, а затем

извлечь соответствующие данные и сохранить результат извлечения в БД, что удобно для анализа аномального сетевого трафика в режиме реального времени. Чтобы обеспечить точность системного перехвата, необходимо сначала прочитать конфигурационный файл; затем установить соединение с БД, зарегистрировать источник данных ODBC, использовать функцию openDatabase() в файле скрипта Python для подключения к БД; настроить драйвер перехвата для создания различных типов, создать таймеры и потоки, мониторинг аномального трафика в режиме реального времени с использованием таймеров; наконец, создать сервер мониторинга в режиме реального времени и вызвать функцию Bind(), чтобы получить IP-адрес сервера мониторинга из файла конфигурации, тем самым завершив процесс перехвата пакетов. Пакет данных о ненормальном трафике может быть перехвачен в режиме реального времени, и может быть добавлена функция отображения мониторинга в режиме реального времени, чтобы пользователь мог просматривать результат захвата пакета в режиме реального времени, а затем находить аномальный узел.

4.1.5. Результаты эксперимента

Чтобы проверить целесообразность исследования метода мониторинга блокировки КС в режиме реального времени, были проведены эксперименты. В качестве экспериментальных объектов выбираются 50 наборов данных за определенный период времени, и данные о нормальном состоянии получаются в соответствии с историческими записями за предыдущие периоды, после чего выполняется стандартизированная обработка. Использовался симулятор Mininet для создания топологии сети связи на виртуальной машине и подключения к удаленному контроллеру для реализации построения сетевой среды связи. В качестве экспериментальных показателей были взяты ошибка мониторинга в режиме реального времени и время задержки связи в КС. Традиционный метод мониторинга сравнива-

ется с мониторингом трафика в сети связи в режиме реального времени. В ходе эксперимента сравниваются традиционный метод мониторинга и разработанный метод мониторинга в режиме реального времени, и результаты сравнения показаны на рис. 4.3.

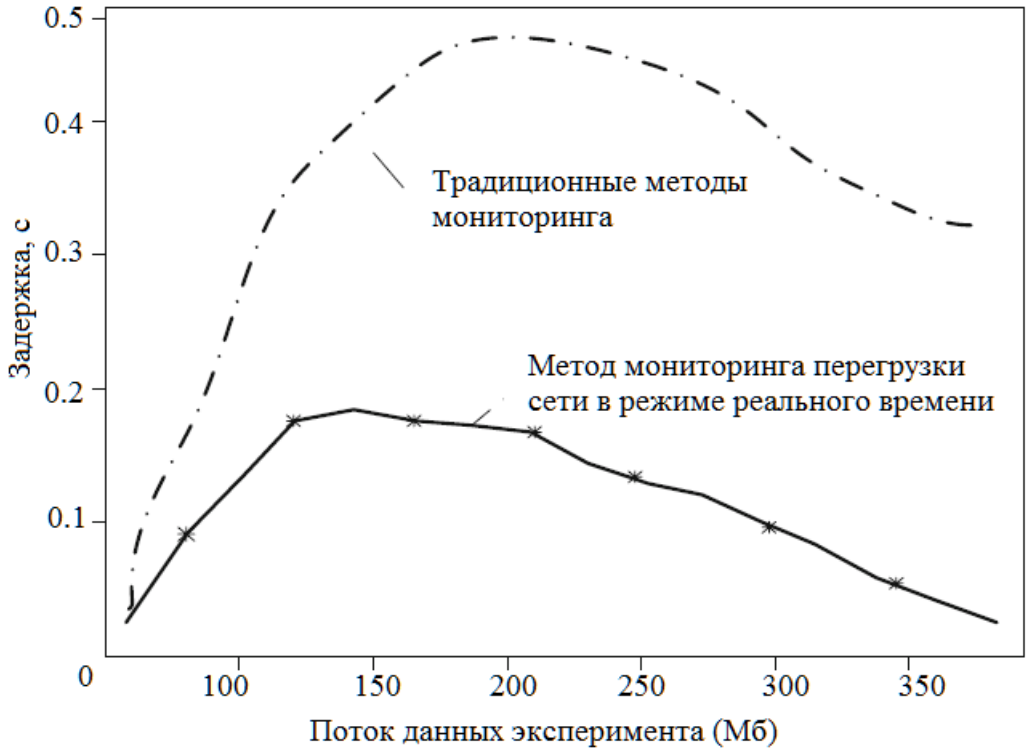


Рис. 4.3. Результаты сравнения времени задержки связи

Время задержки при использовании метода блокировки КС для мониторинга в режиме реального времени напрямую отражает эффективность метода. Из сравнения экспериментальных результатов видно, что время задержки связи при использовании традиционного метода мониторинга будет увеличиваться с увеличением объема данных. Несмотря на явную тенденцию к сокращению задержки при достижении количества 280 М, время задержки всегда превышает 0,3 с. Напротив, в методе мониторинга в режиме реального времени, блокирующем сеть связи, используется метод сбора данных и расчета в режиме реального времени, поэтому экспериментальные результаты показывают идеальный результат задержки, время задержки всегда контролируется в пределах 0,2 с, а когда поток дан-

ных превышает 250 м, наблюдается явная тенденция к снижению, что полностью отражает характер метода мониторинга в режиме реального времени. Кроме того, точность метода мониторинга результатов мониторинга также очень важна. Результаты информации, полученной с помощью разработанного метода мониторинга в режиме реального времени, могут быть точно определены в том месте, где происходит блокировка, и рассчитано значение блокировки, что удобно для своевременной блокировки аварийных ситуаций. Метод имеет низкую погрешность мониторинга и высокую точность.

Для дальнейшей проверки точности мониторинга с помощью этого метода используются традиционный метод и разработанный. Результаты приведены в табл. 4.1.

Таблица 4.1

Точность мониторинга КС

Время мониторинга (мин.)	Точность мониторинга КС (%)	
	Традиционный метод	Предложенный метод
10	67	89
20	69	96
30	71	93
40	70	92
50	65	95
Среднее значение	68.4	93

Согласно табл. 4.1, точность мониторинга данных зависит от времени мониторинга. Когда время мониторинга составляет 10 минут, показатель точности мониторинга данных сети связи традиционным методом составляет 67%, а предложенным методом - 89%. При продолжительности мониторинга 50 минут точность мониторинга данных в КС традиционным методом составляет 65%, а в предложенном методе - 95%. Средний показатель точности традиционного метода составляет 68,4%, в то время как предложенного - 93%.

4.2. Высокоуровневая модель анализа производительности и прогнозирования доступности многоуровневой облачной среды

4.2.1. Многоуровневое моделирование производительности облака

Эффективность оценивается с точки зрения поставщиков услуг и с точки зрения потребителей.

4.2.2. Оценка на основе инфраструктуры

• Оценка на основе инфраструктуры является очень важной категорией для поставщиков облачных услуг. В ходе этой оценки определяются машины, необходимые для создания облачных сервисов. Для каждой машины задается набор параметров для мониторинга производительности машины и частоты отказов. Компонентами инфраструктуры, участвующими в мониторинге и оценке производительности, являются виртуальная машина, хост-машина, хранилище, сеть и т.д. Поскольку отдельные компоненты не могут обеспечить общую производительность многоуровневого облака. Нам необходимо интегрировать все измерения отдельных компонентов в единую многоуровневую облачную систему для оценки доступности всей системы в целом. Основные требования к управлению производительностью на основе инфраструктуры в многоуровневой облачной среде.

• Поддержка любых приложений и служб, размещенных в облачной среде. Сбой, возникший в приложении, должен быть спрогнозирован автоматически.

• Динамический мониторинг виртуальной машины, хост-компьютера и ресурсов, используемых приложением, для контроля частоты отказов и обработки данных о скорости восстановления.

• Частота отказов рассчитывается по всей глубине и ширине многоуровневой облачной среды. Следует учитывать все возможности системы.

• Готовность в любой ситуации справиться с отказом компонента и запустить процесс восстановления отказавшего компонента. Компонент резервного копирования должен быть включен в момент сбоя, чтобы заменить вышедший из строя компонент. Он также поддерживает несколько платформ, таких как Hyper-V, VMware, OpenVZ, Hen, KVM и т.д.

Далее перечислены параметры (рис. 4.4), используемые для мониторинга и оценки производительности коммерческого облака. Этот список разделен на категории в зависимости от типа машины.

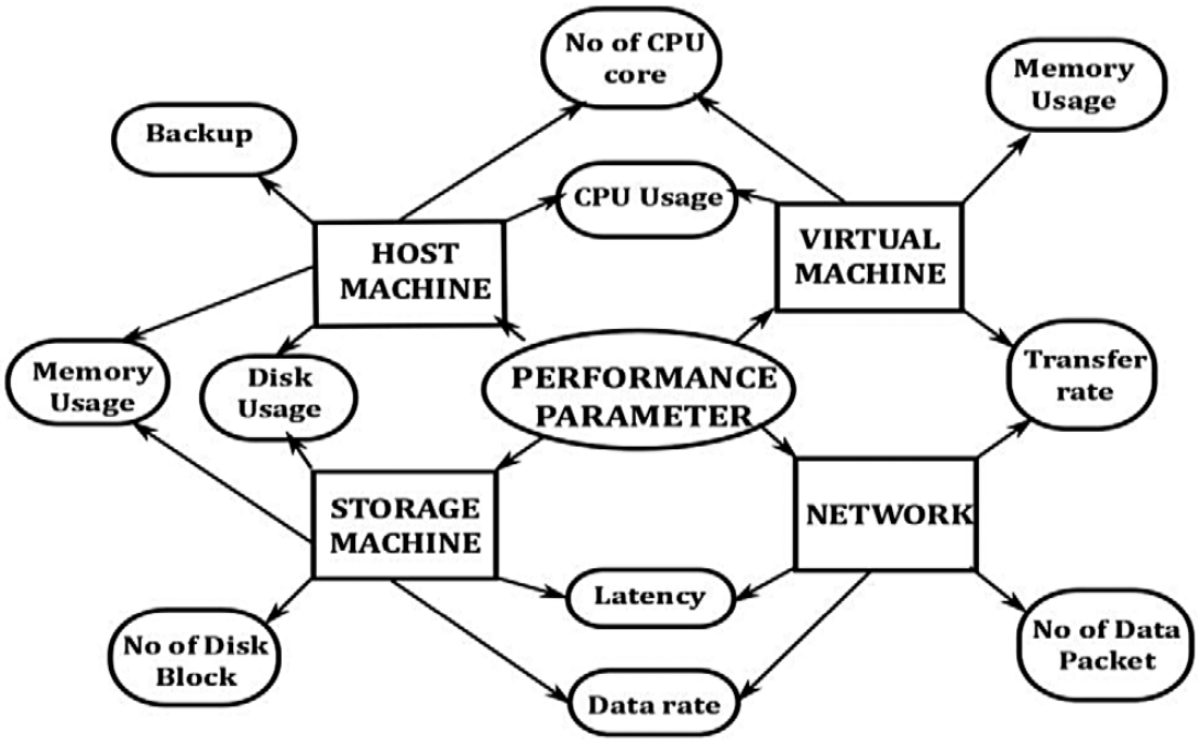


Рис. 4.4. Параметры, используемые при оценке на основе инфраструктуры

Эти категории параметров отслеживаются для оценки частоты отказов компонента в многоуровневой облачной системе. Для вышеуказанного показателя устанавливается пороговое значение. Когда пороговое значение этих показателей достигнуто, компонент или тип машины считаются вышедшими из строя. Частота отказов этой системы оценивается для расчета доступности многоуровневой облачной системы. Чтобы упростить эту

операцию мониторинга, четыре категории типов компьютеров объединены в два типа, такие как виртуальная машина и хост-машина. Тип хранилища относится к хост-машине, а тип сети - к виртуальной машине.

Оценка на основе приложений

Здесь оценивается производительность приложения, размещенного в многоуровневом облаке. Эта оценка проводится для удовлетворения потребностей пользователей облака. Размещенное приложение необходимо отслеживать и сопоставлять для измерения производительности приложения. При оценке производительности используются такие показатели, как время выполнения приложения, использование ресурсов, необходимые сервисы, сеть и качество приложения. Эти показатели отслеживаются, и на их основе оценивается производительность.

Важные параметры для поставщика облачных услуг

Поставщикам облачных услуг необходимо получить подробное представление об анализе производительности всего облака, чтобы оценить ситуацию. Приведенные ниже данные используются для оценки реальной производительности поставщика облачных услуг.

Использование ресурсов

Использование ресурсов является важным параметром для мониторинга и измерения производительности. Использование памяти и диска оценивается для прогнозирования частоты отказов компонента в многоуровневой облачной среде. Рассмотрим многоуровневую облачную систему, в которой задания поступают случайным образом в любое время, поступают для получения услуг и отправляются после завершения обслуживания. Для каждой услуги используется необходимое количество ресурсов в соответствии с моделью организации очередей. Задание поступает в систему упорядоченным образом, а остальные задания будут ждать в очереди до завершения обслуживания этого задания. Если произойдет какая-либо задержка или потеря ресурсов, это затронет все задания, ожидающие пре-

доставления услуг, и они будут отложены. Этот сценарий показывает, что измерение использования ресурсов необходимо для оценки производительности.

Использование ресурсов для каждого компонента представлено в виде $R_i=\{R_1,R_2, \ldots, R_n\}$ со временем доступа $A_i = A_1, A_2, . , A_n$.

Полезность ресурсов

$$R_u = \frac{\sum_{i=0}^{n} R_i}{\sum_{i=0}^{n} A_i}$$

Показатели хоста

В предыдущем разделе мы обсудили параметры, связанные с хост-машиной. Этот параметр влияет на производительность и доступность приложения, что, в свою очередь, влияет на соглашение об уровне обслуживания.

Виртуальные показатели

Аналогично хост-машине, мы должны накапливать данные об использовании ресурсов, ЦП виртуальной машины. Другие показатели, учитываемые в VM, включают количество виртуальных машин, используемых приложением, продолжительность создания новой виртуальной машины, время, необходимое для перемещения приложения с одной виртуальной машины на другую, и промежуток времени для распределения ресурсов по виртуальной машине

Метрика транзакций

Скорость транзакций и передачи данных - это показатели, которые влияют на объем хранилища и сети. Необходимо измерить этот показатель для анализа производительности многоуровневой облачной системы.

4.2.3. Срок службы компонентов многоуровневой облачной системы

Первым шагом для оценки доступности многоуровневой облачной системы является оценка срока службы элемента, задействованного в облачной среде. Рассмотрим многоуровневую архитектурную облачную систему, как показано на рис. 4.5, с несколькими компонентами, такими как хост-машина, виртуальная машина, сервер приложений, сервер баз данных. M (H+V) обозначает активный номер компонента виртуальной машины (V) и хост-машины (H). S обозначает деактивированный или находящийся в режиме ожидания компонент как виртуальной машины, так и HM. Процесс мониторинга выполняется для каждого компонента VM и HM, который имеет набор параметров, описанных в предыдущем разделе, для оценки срока службы. Активный компонент имеет экспоненциальный выход из строя параметра λ, а неактивный компонент имеет постоянную частоту отказов μ ($0 \leq \mu \leq \lambda$).

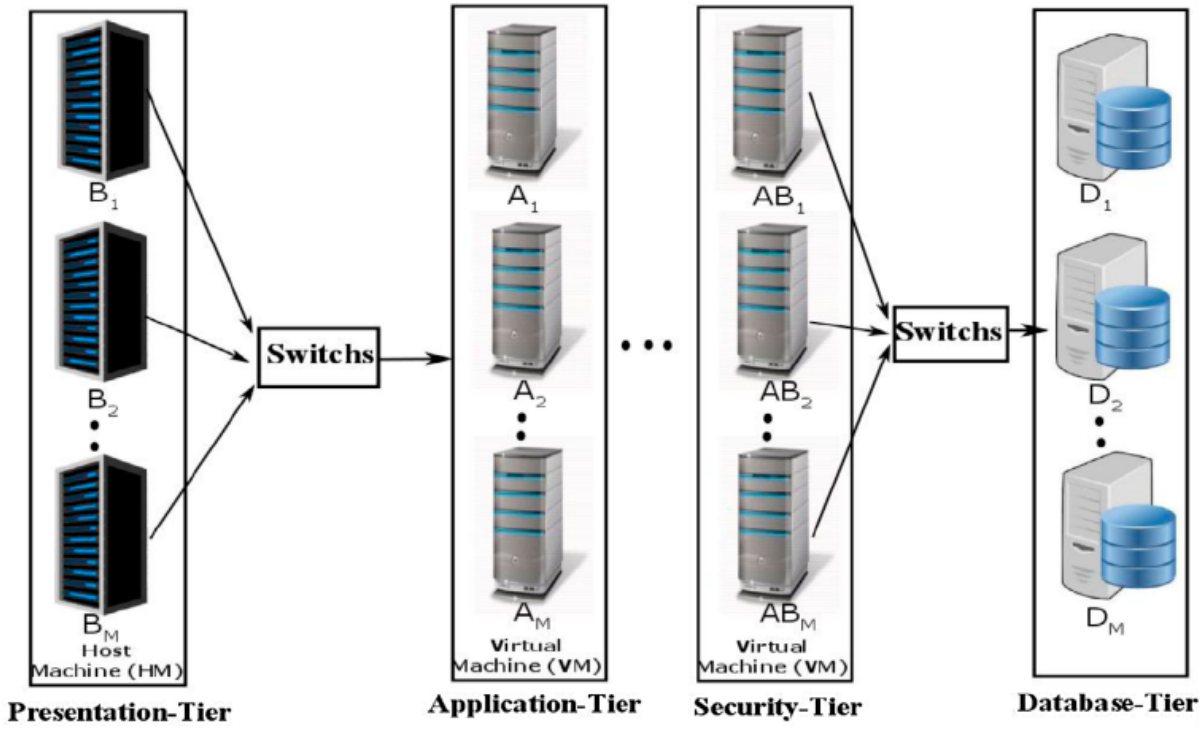


Рис. 4.5. Высокоуровневая мультиоблачная архитектура

Частота отказов активного компонента λ определяется на основе мониторинга набора соответствующих компонентов. Пусть A_i ($1 \le i \le M$) обозначает время жизни активного компонента V, пусть B_i ($1 \le i \le M$) обозначает время жизни активного компонента H и пусть D_j ($1 \le j \le S$) обозначает время жизни неактивного компонента. Тогда время жизни многоуровневой облачной системы $L_T(n|M,C)$ определяется как

$$\begin{aligned} & L_T(n \mid M,S) = L_T(n \mid V,H,S) = \\ & = \min\left(A_1, A_2, ..., A_m; B_1, B_2, ..., B_m; D_1, D_2, ..., D_S\right) + \\ & + L_T(n \mid M, S-1) = \\ & = W_F(H,V,S) + L_T(n \mid H,V,S-1) \end{aligned} \tag{4.5}$$

$W_F(H,V,S)$ − это время, в течение которого в системе происходит сбой между H+V+S компонентами системы и неисправные компоненты удаляются, облачная система имеет M активных и S-1 деактивированных компонентов. Для M+S−1 нет времени на старение.

$$L_T(n \mid H,V,0) = L(n \mid H,0) + \sum_{j=1}^{S} W_F(H,j) + L(n \mid V,0) + \sum_{j=1}^{S} W_F(V,j) \tag{4.6}$$

Здесь LT(n|H,V,0)=LT(n| H,V) - это просто время жизни системы вне H&V и, следовательно, статистика l-го порядка с l=H+V−n+1. Для определения вероятности жизни мы используем гипоэкспоненциальное распределение. Распределение $L_T(n|H,V,0)$ является гипоэкспоненциальным с параметрами (H+V)λ, ((H+V)-1)λ,.... nλ. Это может быть индивидуально распределено L(n|H,0) как Hλ, (H−1)λ,.... nλ и L(n|V,0) как Vλ, (V−1)λ, ... nλ. У активной системы есть частота отказов λ в зависимости от загрузки процессора, диска и задержки. Для расчета надежности системы оценивается частота отказов и частота ремонтов. Таким образом, мы приходим к выводу, что $L_T(n|M,S)$ имеет (H+V+S– (n+1() ступенчатое гипоэкспоненциальное распределение с параметром (H+V)λ+Sμ, (H+V)λ+(S−1) μ, …, (H+V) λ+μ, (H+V)λ, ((H+V)-1) λ,…, nλ.

Надежность - это один из параметров оценки производительности, который тесно связан с доступностью системы. В определенный промежу-

ток времени способность системы выполнять определенную операцию определяется как надежность. Этот параметр необходим для оценки производительности многоуровневой облачной системы в непредвиденной ситуации.

Пусть $R_L[n|M,S](t)$ обозначает надежность всего многоуровневого облака, тогда

$$R_L\left[n \mid H,V,S\right](t) = \sum_{j=1}^{S} a_j e^{-((H+V)\lambda+j\mu)t} + \sum_{j=1}^{H+V} b_j e^{-(\lambda j)t} \tag{4.7}$$

$$\begin{aligned} R_L\left[n \mid M,S\right](t) &= \sum_{j=1}^{S} a_j e^{-(M\lambda+j\mu)t} + \sum_{j=n}^{H+V} b_j e^{-(\lambda j)t} = \\ &= \sum_{j=1}^{S} a_j e^{-(H\lambda+j\mu)t} + \sum_{j=1}^{S} a_j e^{-(V\lambda+j\mu)t} + \sum_{j=n}^{H} b_j e^{-(j\lambda)t} + \sum_{j=n}^{V} b_j e^{-(j\lambda)t} \end{aligned} \tag{4.8}$$

где a_j - время жизни неактивной системы с постоянной частотой отказов μ, b_j - время жизни активной системы с частотой отказов λ.

Из приведенного выше уравнения (4.8) время жизни отдельных компонентов VM и HM можно определить как

$$R_L\left[n \mid H,V,S\right](t) = \sum_{j=n}^{H+V} b_j e^{-(\lambda j)t} \tag{4.9}$$

Генерация MTTF

MTTF генерируется, когда время жизни компонентов VM и HM достигает порогового значения t. В зависимости от времени жизни компонента определяется надежность этого компонента. Пусть L_T обозначает время жизни многоуровневого облачного компонента, так что его надежность определяется

$$R_L(t) = P(L_T > t) \tag{4.10}$$

где R_L указывает на надежность многоуровневой облачной системы и

$$R_L(t) = -g(t) \tag{4.11}$$

Затем рассчитывается среднее время наработки на отказ (MTTF) с учетом доступности системы, которое определяется по формуле:

$$E(L) = \int_0^\infty tg(t)dt \tag{4.12}$$

Подставив (4.11) в (4.12), получим

$$E(L) = -\int_0^\infty tR'_L(t)dt \tag{4.13}$$

Интегрируя уравнение (4.13), получим:

$$E(L) = -tR_L(t)\,|\,0^\infty + \int_0^\infty R_L(t)dt \tag{4.14}$$

$$E(L) = \int_0^\infty R(t)dt \tag{4.15}$$

В теории надежности более позднее выражение для среднего времени до отказа используется как обычное

$$E(L^k) = \int_0^\infty t^k g(t)dt \tag{4.16}$$

Подставим уравнение (4.11) в (4.16)

$$E(L^k) = -\int_0^\infty t^k R'_L(t)dt = -t^k R_L(t)\,|\,0^\infty + \int_0^\infty kt^{k-1} R_L(t)dt \tag{4.17}$$

Путем упрощения мы получаем

$$E(L^k) = \int_0^\infty kt^{k-1} R_L(t)dt \tag{4.18}$$

После нахождения ожидаемого

$$Var(L) = \int_0^\infty 2tR_L(t)dt - \left[\int_0^\infty R_L(t)dt\right]^2 \tag{4.19}$$

Если время жизни компонента зависит от экспоненты, мы получаем для компонента HM & VM (см. (4.9))

$$R_L(t) = \sum_{j=n}^{H+V} b_j e^{-(\lambda j)t} \tag{4.20}$$

Теперь подставим уравнение (4.20) в (4.15), получим

$$E(L) = \int_0^{\infty} \sum_{j=n}^{H+V} b_j e^{-(\lambda j)t} dt$$

Поскольку b_j - это постоянный член, мы можем исключить его,

$$b_j = \sum_{j=n}^{H+V} \frac{1}{\lambda j} = \frac{1}{\lambda} \sum_{j=n}^{H+V} \frac{1}{j} = \frac{H_n}{\lambda} \quad (4.21)$$

$$Var(L) = \int_0^{\infty} 2te^{-\lambda t} dt - \left[\frac{1}{\lambda}\right]^2 = \frac{1}{\lambda^2} \quad (4.22)$$

Таким образом, среднее время жизни и дисперсия времени жизни вычисляются для всей многоуровневой среды. При этом вычислении учитывается загрузка процессора отдельным компонентом, использование диска, задержка и т.д.

4.2.4. Анализ доступности

Отказ компонента в облаке может быть восстановлен как исправный.

Исходя из частоты отказов и восстановления компонента, мы собираемся проанализировать доступность многоуровневой облачной системы.

Пусть T_k - время, затрачиваемое на выполнение функции, а D_k - время простоя, в течение которого выполняется ремонт или восстановление. Последовательность случайных величин $\{Y_k=T_k+D_k\}$ (k=1,2,...) предполагается взаимно независимой.

Предположим, что T_k уникально распределены с помощью CDF U(t) и PDF u(t), и что D_k уникально распределены с помощью CDF H(t) и PDF h(t).Тогда Y_k также распределены однозначно, следовательно, {Yk| k=1,2,…} - это процесс обновления. Точкой обновления в этом процессе является вероятность завершения ремонта. Базовая плотность f (t) является сверткой u и h. Таким образом

$$L_f(S) = L_u(s)L_h(s) \quad (4.23)$$

И используя свойство свертки, мы получаем

$$L_m(s) = \frac{L_u(s)L_h(s)}{1\text{-}L_u(s)L_h(s)} \tag{4.24}$$

Среднее количество ремонтов или восстановлений M(t) в интервале (0,t) получено преобразованием Лапласа:

$$L_m(s) = \frac{L_u(s)L_h(s)}{s\left[1\text{-}L_u(s)L_h(s)\right]} \tag{4.25}$$

Мгновенная доступность A(t) облачной системы определяется как вероятность того, что система будет нормально функционировать в момент времени t. Доступность A(t), которая просто равна R_L, равна 1-u(t), когда нет элемента для восстановления. Вероятность, связанная со вторым случаем, равна $\int_0^t R_L(t-x)m(x)dx$.

Отсюда

$$A(t) = R_L(t) + \int_0^t R_L(t-x)m(x)dx \tag{4.26}$$

Рассчитанная мгновенная доступность больше или равна надежности. Используя преобразования Лапласа с обеих сторон уравнения (4.26), получим

$$L_A = L_{R_L}(s) + L_{R_L}(s)L_m(s) = L_{R_L}(s)\left[1 + L_m(s)\right] + \frac{L_{R_L}(s)}{\left[1 - L_u(s)L_h(s)\right]} \tag{4.27}$$

Используем уравнение (3.5), поскольку $R_L(t)$=1-U(t):

$$L_A = L_{R_L}(s) = \frac{1}{s} - L_U(s) = \frac{L_u(s)L_h(s)}{s\left[1 - L_u(s)L_h(s)\right]} \tag{4.28}$$

Приведенное выше уравнение вычисляет мгновенную доступность A(t) облачной системы на основе времени сбоя и времени ремонта облачной системы

Рассмотрим сценарий на многоуровневой облачной машине; множество приложений обрабатывается на сервере приложений с номером n на виртуальной машине и с помощью сервера базы данных с номером m. Ко-

гда приложение с высоким приоритетом попадает в облачную систему, мы должны запланировать это приоритетное приложение на сервере приложений. Чтобы запланировать это приложение на определенный промежуток времени, мы используем ограничение доступности для расчета доступности n-го номера сервера. После определения доступности сервера приложений в указанный промежуток времени это приложение с приоритетами назначается доступному серверу приложений. Доступность сервера определяется на основе срока его службы в системе. Для большей точности при расчете доступности системы используется предельная вероятность. Предельная вероятность A - это ожидаемый период времени t, в течение которого система работает. Предельная вероятность определяется как:

$$\lim_{t\to\infty} R_L = \lim_{t\to\infty}[1-u(t)] = 0 \tag{4.29}$$

Для получения выражения предельной вероятности используется теорема Лапласа о конечном значении. Пусть $H(t) = \int_0^t h(x)dx + H(0^-)$ и, применив преобразование Лапласа, мы получим

$$sL_H(s) - H(0^-) = L_h(s) = \int_0^t e^{-st}h(t)dt$$

Следовательно,

$$\lim_{t\to\infty} sL_H(s) = \int_0^\infty h(x)dx + H(0^-) = \lim_{t\to\infty} sL_H(s)\left[\int_0^\infty h(x)dx\right] + H(0^-) = \\ = \lim_{t\to\infty} H(t) \tag{4.30}$$

Преобразование Лапласа также применяется для ограниченной вероятности

$$A = \lim_{t\to\infty} A(t) = \lim_{t\to\infty} sL_A(s)$$

Для малого значения s аппроксимация используется следующим образом:

$$A = \int_0^{\infty} e^{-st} u(t)dt \cong 1 - L_U = \int_0^{\infty} e^{-st} u(t)dt \cong 1 - \frac{s}{\lambda} \tag{4.31}$$

где $MTTF = \frac{1}{\lambda}, L_h(s) = 1 - \frac{s}{\mu}$, $MTTR = \frac{1}{\mu}$, then the limiting availability,

$$A = \lim_{t \to \infty} \left[\frac{1 - \left(1 - \frac{s}{\lambda}\right)}{1 - \left(1 - \frac{s}{\lambda}\right)\left(1 - \frac{s}{\mu}\right)} \right] = \frac{MTTF}{MTTF + MTTR} \tag{4.32}$$

Приведенное выше уравнение (4.32) показывает, что предельная эксплуатационная готовность основана на среднем времени до отказа (MTTF) и среднем времени ремонта (MTTR). Используя эту ограниченную доступность системы, мы можем оценить доступность компонента в многоуровневой облачной системе в ожидаемый промежуток времени. Мгновенная доступность используется для оценки доступности всей функции многоуровневой облачной системы. С помощью этого метода обеспечения доступности измеряется производительность многоуровневой облачной системы.

4.2.5. Экспериментальная оценка

Доступность измеряется с помощью инструмента SHARPE [4.23]. Символьно-иерархическая автоматизированная оценка надежности и производительности (SHARPE) - это инструмент моделирования, который сочетает в себе гибкость марковской модели и эффективность комбинаторной модели. Этот инструмент переносим на большинство архитектур и операционных систем. Он используется во многих областях исследований, образования и инженерной практики. Графический интерфейс пользователя (GUI) доступен и запрограммирован на языке Си. Этот инструмент разработан только для анализа производительности, надежности и быстродействия. В этот инструмент встроено множество методов, основанных на

производительности. Методами измерения доступности и надежности являются дерево отказов, блок-схема надежности, MRGP, цепочка Маркова, график надежности, сеть Петри и график задач. В нем есть возможность печатать модели и выводить данные. Инструмент SHARPE очень эффективен и популярен при оценке производительности. В этом эксперименте инструмент SHARPE используется для разработки высокоуровневой архитектуры многоуровневого облака (рис. 4.6) и применения предложенного уравнения к разрабатываемому облаку. Инструмент SHARPE был выбран из-за его гибкого характера, позволяющего легко разрабатывать и применять предложенную методологию.

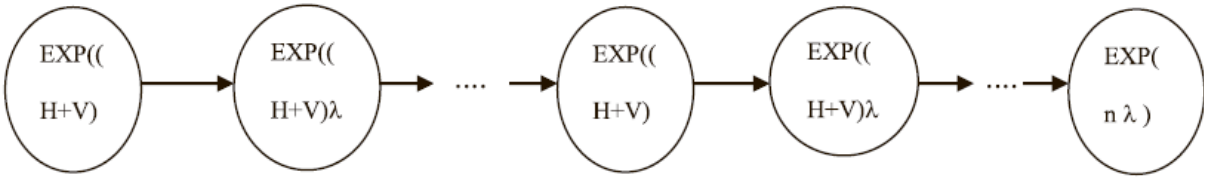


Рис. 4.6. Диаграмма состояний компонента H+V в многоуровневом облаке

Экспериментальная установка

Мы создали модель для многоуровневой облачной системы, аналогичную нашей высокоуровневой архитектуре (рис. 4.5). Этот симулятор обеспечивает облачный дизайн. Каждому процессору (виртуальным машинам) в проекте назначена модель частоты отказов, скорости восстановления и мгновенной доступности. Используя приведенную выше математическую формулу, оценивается доступность каждого сервера. В этом проекте многоуровневое облако состоит из трех облачных сервисов, таких как служба приложений, службы безопасности и службы баз данных. Доступность оценивается для этих трех сервисов и для всего многоуровневого облака. Результатом нашей модели будет оценка доступности облачной системы. Поскольку выбрано многоуровневое облако, ни один из сервисов не может быть разработан и поддержан. Но для целей эксперимента были вы-

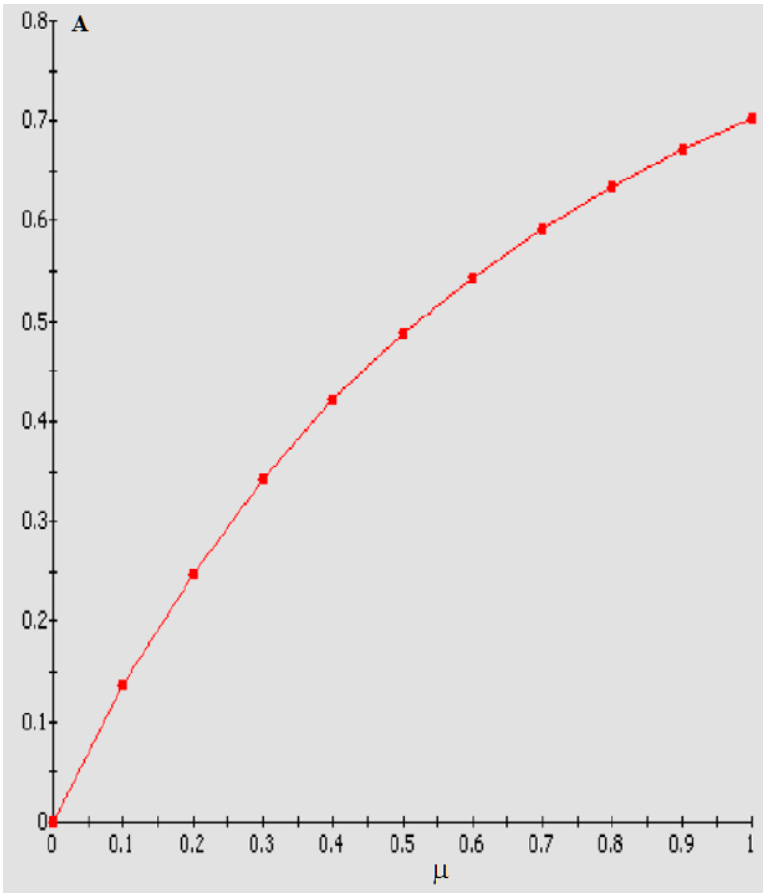
браны и продемонстрированы только три облачных сервиса. Построена центральная часть многоуровневой облачной системы. Из этой системы управляются все остальные сервисы. Следующим является уровень приложений, который содержит сервер приложений, инкапсулированный в несколько виртуальных машин. Соединительный уровень, который предоставляет пользователю ресурсы для выполнения приложения, - это уровень базы данных. Модель базы данных содержит блоки хранения данных. Дополнительный уровень, подключенный к службе безопасности, называется Security Tier. Служба безопасности содержит несколько служб, выполняющих функции по обеспечению безопасности.

Оценка эффективности

Как показано на рис. 4.5, многоуровневое облако создается с помощью инструмента SHARPE. Анализ доступности оценивается по формуле (4.32). В работе процесс оценки выполняется для построенного многоуровневого облака. Предложенная модель может прогнозировать доступность для любого типа среды. Созданное многоуровневое облако построено с дополнительным уровнем безопасности в качестве сервиса. Для проверки того, что построенная модель работает лучше по сравнению со всеми другими моделями, используется предложенная доступность.

а) По результатам оценки можно спрогнозировать производительность модели на основе качества обслуживания. Используя этот результат, можно также гарантировать уровень обслуживания на основе значения доступности. На графике (рис. 4.7а) показано, что доступность построенной модели мгновенно возрастает в зависимости от рабочей нагрузки. Значение прогнозируемого значения доступности может быть использовано для сравнения моделей разных уровней. Результат сравнительной оценки показывает наилучшую модель производительности.

а)



б)

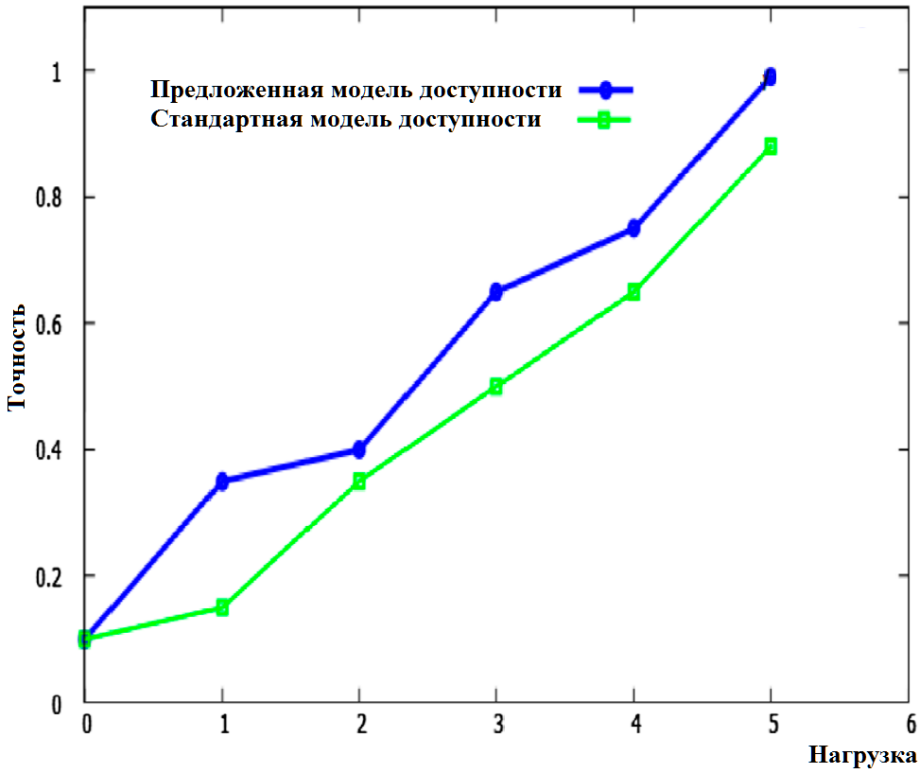


Рис. 4.7 а) Моделирование доступности для многоуровневой облачной системы, б) сравнение точности

На рис. 4.7б показано сравнение точности доступности между предложенной моделью доступности и моделью доступности [4.25]. Результат показывает, что предложенная модель является более точной по сравнению со стандартными моделями. Она предсказывает значение доступности с точностью приблизительно 99%, на 8.3% точнее стандартных.

Разработан прототип программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов (рис. 4.8).

Рисунок 4.8. Структура и функции прототипа программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов

4.3. Выводы к главе 4

1. В среде облачных вычислений для обеспечения стабильности работы компьютерной сети необходимо обеспечить, чтобы сетевой трафик всегда находился в нормальном состоянии. С помощью метода мониторинга КС в режиме реального времени можно отслеживать текущее состояние всей КС и своевременно обнаруживать проблему блокировки сети, что облегчает непрерывное управление и техническое обслуживание. В процессе разработки метода мониторинга было обнаружено, что, хотя метод обладает высокой точностью мониторинга, долгосрочную стабильную работу метода еще предстоит изучить, и в будущем будет разработан высокопроизводительный метод мониторинга для поддержки мониторинга сети в режиме реального времени.

2. Предложенный метод оценки доступности позволяет оценить производительность облака в многоуровневом приложении. Эта оценка реализована в инструменте SHARPE. Многоуровневый облачный дизайн разработан в инструменте SHARPE, и доступность для этого проекта проверяется с помощью предложенной методологии. Приведенная математическая модель доступности используется для точной оценки доступности многоуровневой облачной системы. Эта модель доступности может быть использована для любого типа облачных систем.

3. В работе также проводится оценка надежности и срока службы для прогнозирования доступности системы. При оценке многоуровневой облачной системы учитываются все аспекты системы. Предлагаемая модель доступности позволяет оценить QoS для различных моделей. Его можно использовать для сравнения различных моделей и оценки уровня их производительности во всех аспектах. При оценке производительности облачной системы также учитываются аспекты надежности, параллельный и последовательный режимы работы. Таким образом, результаты предлагае-

мой работы полезны:

1. Для оценки производительности различных типов облачных систем.

2. Для разработки комплексной модели, применимой ко всем типам облачных систем.

3. Для улучшения качества обслуживания.

4. Для обеспечения соблюдения соглашения об уровне обслуживания.

Будущая работа заключается во внедрении решения для обеспечения доступности на удаленном сервере. С помощью этой модели можно оценить серверы, расположенные в удаленных местах, и измерить производительность. Модель доступности может быть применена и реализована в различных типах облаков.

ЗАКЛЮЧЕНИЕ

В процессе выполнения диссертационного исследования получены следующие основные результаты:

1. Создана ситуационная модель анализа организации-разработчика программного обеспечения с открытым программным кодом, обеспечивающая определение степени открытости программного кода и уровень стратегической открытости организации в экосистеме жизненного цикла.

2. Предложена модификация технологии стохастического моделирования облачной архитектуры, позволяющая автоматически генерировать стохастические имитационные модели с высоким уровнем согласованности с поведением облачной архитектуры

3. Разработан алгоритм идентификации состояния инфокоммуникационной системы на основе облачных вычислений, обеспечивающий расчет трафика для определения наличия состояния блокировки в системе и определения точного местоположения точки блокировки

4. Предложена модель анализа производительности и прогнозирования доступности многоуровневой облачной среды, обеспечивающая учет времени жизни основных компонентов и оценку доступности облачной среды. Модель предсказывает значение доступности на 8.3% точнее стандартных.

5. Разработана структура программного обеспечения распознавания блокировок и оптимизации доступности облачных сервисов, обеспечивающая реконфигурацию инфокоммуникационной системы в зависимости от параметров инфраструктуры и качества обслуживания.

6. Элементы программного обеспечения зарегистрированы в ФИПС.

Рекомендации и перспективы дальнейшей разработки темы

1. Результаты исследования рекомендуются к применению в задачах исследования и разработки программного обеспечения с открытым исходным кодом в облачных архитектурах.

2. Дальнейшая разработка темы будет направлена на практическую реализацию теоретических и алгоритмических результатов, интеграцию в проекты наиболее распространенных распределенных систем. Развитие результатов будет направлено на автоматизацию реконфигурации инфокоммуникационных систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Амоа К.К.А.Ж. Многоагентный подход к построению распределенной архитектуры веб-приложения// Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. - 2022. - №10. - С. 47-51.

2. Амоа Куадио-кан Армел Жеафруа, Баталов Д.И., Балдин А.В., Мутина Е.И. Моделирование облачных систем с использованием формализмов и программных интерфейсов CloudSim// Системы управления и информационные технологии, №3(97), 2024. С. 55-59.

3. Амоа Куадио-кан Армел Жеафруа, Рындин Н.А. Модель OSE в экосистеме организации-разработчика программного обеспечения// Экономика и менеджмент систем управления, №2(52), 2024. – С. 39-48.

4. Амоа Куадио-кан Армел Жеафруа, Сидоренко Е.В., Рындин Н.А. Мониторинг состояния коммуникационных сетей на основе облачных вычислений в режиме реального времени// Моделирование, оптимизация и информационные технологии. 2025;13(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1809 DOI: 10.26102/2310-6018/2025.48.1.014

5. Амоа Куадио-кан Армел Жеафруа. Оперирующая система в экосистеме организации-разработчика программного обеспечения// Информационные технологии моделирования и управления, №2(136), 2024. – С. 88-100.

6. Амоа Куадио-Кан Армел Жеафруа. Проблемы и особенности использования формализмов при исследовании облачных систем// Наука и технологии: перспективы развития и применения: сб. статей VII Междунар. НПК. - Петрозаводск: МЦНП «НОВАЯ НАУКА», 2024. - С. 59-64.

7. Амоа Куадио-кан Армел Жеафруа. Программные решения для имитационного моделирования сложных систем с участием облачных тех-

нологий// Информационные технологии моделирования и управления, №3(137), 2024. – С. 222-238.

8. Атласов И.В., Горшков А.В., Каперко А.Ф., Амоа Куадио-кан Армел Жеафруа, Коптелова А.С. Информационное представление метода и модели кэширования на основе использованием временных меток// Системы управления и информационные технологии, №2(96), 2024. - С. 46-50.

9. Бегичева С. В. Облачные технологии в практике управления малым и средним бизнесом : учеб. пособие /С. В. Бегичева, А. Д Назаров, Д. М. Назаров ; М-во науки и высш. образования Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург, 2017. - 103 с.

10. Брозгунова Н. П. Роль и значение информационных технологий в управлении предприятием / Н. П. Брозгунова, И. М. Жамкова, И. И. Осокин, С. В. Выговский // Экономика и предпринимательство. - 2021. - № 5. - С. 1029-1033.

11. Василишина А. А. Стратегические технологии как инструменты управления цифровым гостиничным и туристическим бизнесом = Strategic technologies as tools for managing the digital hotel and tourism business / А. А. Василишина, Е. С. Кваша, Е. Ю. Никольская // Гостиничное дело. - 2022. - № 2. - С. 141-147.

12. Гаврилов Л. П. Инновационные технологии в коммерции и бизнесе : учебник / Л. П. Гаврилов. - Москва : Юрайт, 2013. - 372 с.

13. Горохов Е. Постизоляционные тренды удаленной работы и облачных решений от M1Cloud / Е. Горохов // Экономист. - 2020. - № 6. - С. 93-94.

14. Горохов Е. Тренды облачного рынка в 2021 г. / Е. Горохов // Экономист. - 2021. - № 3. - С. 58-59.

15. Гузев И. Почему банки не витают в облаках? / И. Гузев // Bis journal. - 2022. - № 1. - С. 40-41.

16. Догучаева С. М. Инновационный подход облачных технологий в

условиях пандемии / С. М. Догучаева // Риск: ресурсы, информация, снабжение, конкуренция. - 2022. - № 1. - С. 71-74.

17. Иванов Ф. Д. Электронная логистика как инструмент повышения конкурентоспособности / Ф. Д. Иванов, М. О. Петраков // Экономика и предпринимательство. - 2021. - № 6. - С. 824-828.

18. Ильяшенко О. Ю. Современное состояние развития облачных технологий / О. Ю. Ильяшенко, В. М. Ильяшенко, Е. Л. Лукьянченко // Экономика и предпринимательство. - 2020. - № 10. - С. 1219-1223.

19. Исследование PwC "Страх облаков" / Аудиторская компания "PricewaterhouseCoopers - PwC" (Москва) // Акционерное общество: вопросы корпоративного управления. - 2020. - № 12. - С. 26-31.

20. Каменский А. М. Корпоративное управление в условиях коронавируса / А. М. Каменский, С. А. Данилова, Л. М. Миронова [и др.] // Акционерное общество: вопросы корпоративного управления. - 2020. - № 5. - С. 28-38.

21. Кишкович Ю. П. Облачная ITSM-система - решающий фактор развития компании / Кишкович Ю. П. // Финансы и кредит. - 2021. - Т. 27, вып. 9. - С. 1997-2007.

22. Кишкович Ю. П. Стимулирование и границы применения облачной ITSM- системы российскими компаниями / Ю. П. Кишкович // Региональная экономика: теория и практика. - 2021. - Т. 19, вып. 10. - С 2001-2008.

23. Клунко Н. С. Основные тренды цифровой трансформации фармацевтической отрасли / Н. С. Клунко, Н. В. Сироткина // Организатор производства. - 2021. - № 2. - С. 89-97.

24. Куадио-кан Армел Жеафруа. Исследование пограничных вычислений с использованием Cloudsim// Сб. тр. VI Всеросс. НПК «Информационные технологии в экономике и управлении». – Махачкала, 2024. С. 59-64.

25. Манучарян С. К. Влияние IT-технологий на развитие управленческого учета / С. К. Манучарян // Экономика и предпринимательство. - 2021. - № 1. - С. 1362-1365.

26. Минина Е. Е.Распределенные системы и облачные технологии : учеб. пособие / Е. Е. Минина ; М-во науки и высш. образования Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург : Изд-во Урал. гос. экон. ун-та, 2020. -122 с.

27. Нетёсова О. Ю. Информационные системы и технологии в экономике : учеб. пособие / О. Ю. Нетёсова. - 3-е изд., испр. и доп. - Москва : Юрайт, 2017. - 146 с.

28. Никульчев Е. В., Лукьянчиков О. И., Ильин Д. Ю. Облачные технологии: учебное пособие. – М.: МИРЭА, 2019.

29. Обухов Е. Ожидается сильная облачность / Е. Обухов, К. Пахунов // Эксперт. - 2021. - № 24. - С. 37-41.

30. От хранения данных к управлению информацией : учебник / под ред.: Н. Римицан; пер. с англ. Н. Вильчинского. - 2-е изд. - СПб : Питер, 2016. - 543 с.

31. Парчинский К. С. Облачные технологии в управлении современным производством / Парчинский К. С. // Конкурентоспособность территорий : материалы XXII Всеросс. экономич. форума молодых ученых и студентов (Екатеринбург, 22-26 апр. 2019 г.) : в 5 ч. / М-во науки и высш. образования Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург, 2019. - Ч.3. - С.107 -109.

32. Пичугин А Плужник Е.В., Никульчев Е.В. Персонифицированные информационные системы и слабоструктурированные базы данных в облачных технологиях // Всероссийская конференция: Индустриальные информационные системы. - http://conf.nsc.ru/iis2013/ru/reportview/161281.

33. 5. Программа интерактивного управления очередью системы мониторинга/ Е.В. Сидоренко, М.В. Кочегаров, В.А.К. Камиль, К.А.Ж.

Амоа, О.А. Ющенко. Свидетельство о регистрации программы для ЭВМ № 2025615513 от 12.02.2025. М.: ФИПС, 2025.

34. Пузийчук С. В. Технологии "умного" туризма для лучшего клиентского опыта = Technologies of "Smart" Tourism For The Best Customer Experience / С. В. Пузийчук // Вестник Национальной академии туризма. - 2020. - № 1. - С. 34-36.

35. Современные информационно- коммуникационные технологии для успешного ведения бизнеса : учеб. пособие / Ю. Д. Романов, Л. П. Дьяконова, Н. А. Жарова, [и др.] . - Москва : ИНФРА - М, 2019. -279 с.

36. Соловьева И. П. Облачные технологии как элемент цифровизации производственных процессов / И. П. Соловьева, Е. Н. Евдокимова, М. В. Куприянова, И. П. Симикова // Экономика и предпринимательство. - 2020. - № 2. - С. 645-648.

37. Филатов Е. Банки, использующие облака, растут в два раза быстрее конкурентов / Е. Филатов // Банковские технологии. - 2021. - № 2. - С. 32.

38. Чиркин М. А. Роль инновационных облачных услуг в развитии среднего и малого бизнеса Урала / М. А. Чиркин // BI- технологии и корпоративные информационные системы в оптимизации бизнес-процессов цифровой экономики : материалы VIМеждународ. науч.-практ. очно-заоч. конф. (Екатеринбург, 5 дек. 2018 г.) / М-во науки и высш. образования Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург, 2019. - С. 103-107.

39. A survey of mathematical models, simulation approaches and test-beds/ G. Sakellari, G. Loukas// Research in cloud computing 39 (2013) 92–103.

40. A taxonomy of model transformation/ T. Mens, P.V. Gorp// Electron. Notes Theor. Comput. Sci. 152 (2006) 125–142.

41. A toolbox for functional and quantitative analysis of deds/ F. Bause, P. Buchholz, P. Kemper// Proc. of the 10th Int. Conf. on Computer Performance Evaluation: Modelling Techniques and Tools, Tools ’98, Springer-Verlag, Lon-

don, UK, 1998, pp. 356–359.

42. Abdelmaboud, A., Jawawi, D.N., Ghani, I., Elsafi, A., & Kitchenham, B. (2015). Quality of service approaches in cloud computing: A systematic mapping study. Journal of Systems and Software, 101, 159–179. Bauer E., & Adams, R. (2012). Reliability and Availability of Cloud Computing. JohnWiley & Sons.

43. Alspaugh T.A., Asuncion H.U., Scacchi, W., 2008. The role of software licenses in open architecture ecosystems// Proceedings of the First International Workshop on Software Ecosystems, Washington, Virginia.

44. Amoa Kouadio-kan Armel Geoffroy, Ryndin N.A. Research of boundary computing based on modeling of cloud systems using formalisms and software interfaces CloudSim// Modern informatization problems in the technological and telecommunication systems analysis and synthesis (MIP-2025'AS): Proc. of the XXX-th Int. Open Science Conf.. - Yelm, WA, USA: Science Book Publishing House, 2025. – pp. 140-148.

45. Anvaari M., Jansen S., 2010. Evaluating architectural openness in mobile software platforms// Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, Copenhagen, Denmark, ACM, pp.85–92.

46. Badii C. et al. Icaro cloud simulator exploiting knowledge base, Simul. Modell. Pract. Theory 62 (2016) 1–13.

47. Bala Iyer N.V., Lee C.-H., 2006. Managing in a "small world ecosystem": lessons from the software sector, Harvard Business Review.

48. Bannerman P., Zhu L., 2008. Standardization as a business ecosystem enabler// Proceedings of the International Workshop on Enabling Service Business Ecosystems (ESBE'08), Sydney, Australia.

49. Barbierato E. et al. Modeling and evaluating the effects of big data storage resource allocation in global scale cloud architectures, Int. J. Data Warehouse. Min. 12 (2) (2016) 1–20, https://doi.org/10.4018/IJDWM.2016040101.

50. Barbierato E. et al. Exploiting multiformalism models for testing and performance evaluation in SIMTHESys, Proceedings of 5th International ICST Conference on Performance Evaluation Methodologies and Tools - VALUETOOLS 2011, (2011).

51. Barbierato E. et al. Performability modeling of exceptions-aware systems in multiformalism tools, ASMTA, (2011), pp. 257–272.

52. Barbierato E. et al. Performance evaluation of NoSQL big-data applications using multi-formalism models, Fut. Gener. Comput. Syst. 37 (0) (2014) 345–353.

53. Barbosa O., Alves C., 2011. A systematic mapping study on software ecosystems// Proceedings of the third Workshop on Software Ecosystems, CEUR-WS, vol. 746, pp. 15–26.

54. Bi, J., Zhu, Z., Tian, R., & Wang, Q. (2010, July). Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In 2010 IEEE 3rd International Conference on Cloud Computing (pp. 370–377). IEEE.

55. Bonaccorsi A., Rossi C., 2003. Why open source software can succeed// Research Policy 32(7), 1243–1258.

56. Bosch J., 2009. From software product lines to software ecosystems// Proceedings of the 13th International Conference on Software Product Lines, San Francisco, USA, pp. 111–119.

57. Boucharas V., Jansen S., Brinkkemper S., 2009. Formalizing software ecosystem modeling// IWOCE'09: Proceedings of the 1st international workshop on Open component ecosystems. ACM, New York, NY, USA, pp. 41–50.

58. Bruneo, D. (2014). A stochastic model to investigate data center performance and qos in iaas cloud computing systems. IEEE Transactions on Parallel and Distributed Systems, 25, 560–569.

59. Calheiros R.N. et al. Cloudsim: a toolkit for modeling and simulation

of cloud computing environments and evaluation of resource provisioning algorithms// Softw. 41 (1) (2011) 23–50.

60. Cardinale C., Brunton S.L., Colonius T. Spectral proper orthogonal decomposition using sub-Nyquist rate data. arXiv preprint arXiv:2501.02142, 2025

61. Castiglione A. et al. Modeling performances of concurrent big data applications// Softw. 45 (8) (2015) 1127–1144.

62. Cataldo M., Herbsleb J.D., 2010. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration// Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA ’10, ACM, New York, NY, USA, pp. 65–72.

63. Chen, Y., Iyer, S., Liu, X., Milojicic, D., & Sahai, A. (2007, June). SLA decomposition: Translating service level objectives to system level thresholds. In Fourth International Conference on Autonomic Computing (ICAC’07) (pp. 3–3). IEEE.

64. Chief Information Officer Council, United Kingdom, Government ICT Strategy: Smarter, Cheaper, Greener, 2010.

65. Ciardo G. et al. Logic and stochastic modeling with smart// Perform. Eval. 63 (2006) 578–608.

66. Ciardo G., Miner A.S. SMART: the stochastic model checking analyzer for reliability and timing// Quantitative Evaluation of Systems, Int. Conf. (2004), pp. 338–339.

67. Classification of model transformation approaches/ K. Czarnecki, S. Helsen// Proc. of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, (2003).

68. CMMI Product Team, 2006. Capability Maturity Model for Development, Version 1.2.

69. Courtney T. et al. Mobius 2.3: an extensible tool for dependability, security, and performance evaluation of large and complex system models//

DSN, IEEE, 2009, pp. 353–358.

70. Cusumano M.A., 2008. The changing software business: moving from products to services// IEEE Computing 41 (1), 20–27.

71. Dai, Y.S., Yang, B., Dongarra, J., & Zhang, G. (2009, November). Cloud service reliability: Modeling and analysis. In 15th IEEE Pacific Rim International Symposium on Dependable Computing (pp. 1–17).

72. Dantas, J., Matos, R., Araujo, J., & Maciel, P. (2015). Eucalyptus-based private clouds: Availability modeling and comparison to the cost of a public cloud. Computing, 97, 1121–1140.

73. Davis E., Spekman R., 2003. Extended Enterprise, the Gaining Competitive Advantage Through Collaborative Supply Chains. FT Press.

74. De Lara J., H. Vangheluwe, Atom3: a tool for multi-formalism and meta-modelling, in: R.-D. Kutsche, H. Weber (Eds.), FASE, Lecture Notes in Computer Science, 2306 Springer, 2002, pp. 174–188.

75. De Molder J., van Lier B., Jansen S., 2011. Clopenness of systems: the interwoven nature of ecosystems// Proceedings of the third Workshop on Software Ecosystems, Brussels, Belgium, CEUR-WS, vol. 746, pp. 52–64.

76. Deavours D.D. et al. The Mobius framework and its implementation, 2002.

77. Deelman E. et al. Panorama: an approach to performance modeling and diagnosis // Extreme scale workflows, 31 (2015) 1–15.

78. Der Hartigh E., Tol M., Visscher W., 2006. The health measurement of a business ecosystem// Proceedings of the European Chaos/Complexity in Organisations Network (ECCON) Conference.

79. Dos Santos R.P., Werner C.M.L., 2010. Revisiting the concept of components in software engineering from a software ecosystem perspective// Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA ’10, ACM, New York, NY, USA, pp. 135–142.

80. Drucker P., 1954. The Practice of Management. Harper and Row, New York.

81. Dubinsky Y., Kruchten P., 2009. Software development governance (sdg)// 2nd workshop ACM SIGSOFT Software Engineering Notes 34 (5), 46–47.

82. Edwards, D.J. (2024). Network Monitoring and Defense. In: Critical Security Controls for Effective Cyber Defense. Apress, Berkeley, CA. https://doi.org/10.1007/979-8-8688-0506-6_13

83. Fabian, B., Baumann, A., & Lackner, J. (2015). Topological analysis of cloud service connectivity. Computers & Industrial Engineering, 88, 151–165.

84. Farbey B., Finkelstein A., 1999. Exploiting software supply chain business architecture: a research agenda// Proceedings of the 1st Workshop on Economics-Driven Software Engineering Research (EDSER-1), Los Angeles, USA, 1999.

85. Feature-based survey of model transformation approaches/ K. Czarnecki, S. Helsen// IBM Syst. J. 45 (3) (2006) 621–645.

86. Fernndez-Cerero D. et al. Score: simulator for cloud optimization of resources and energy consumption// Simul. Modell. Pract. Theory 82 (2018) 160–173.

87. Filho M.C.S. et al. Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness// 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), (2017), pp. 400–406.

88. Gawer A.G., Cusumano M.A., 2002. Platform Leadership. Harvard Business School Press, Boston, MA, USA.

89. Ghosh, R., Longo, F., Frattini, F., Russo, S., & Trivedi, K.S. (2014). Scalable analytics for iaas cloud availability. IEEE Transactions on Cloud Computing, 2, 57–70.

90. Gorschek T., Fricker S., Brinkkemper S., Ebert C., 2010// Third International Workshop on Software Product Management, Atlanta, USA. SIGSOFT Software Engineering Notes 35 (2), 25–29.

91. Gribaudo M.I. An introduction to multiformalism modeling, in: M. Gribaudo, M. Iacono (Eds.), Theory and Application of Multi-Formalism Modeling, IGI Global, Hershey, 2014, pp. 1–16.

92. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing/ R. Buyya, M. Murshed// Concurrency and computation: practice and experience 14 (13–15) (2002) 1175–1220.

93. Hamedani M.N. et al. Evaluation of performance modelling: optimizing simulation tools to stages of architectural design// Procedia Eng. 118 (2015) 774–780.

94. Hevner A.R., March S.T., Park J., 2004. Design Science in Information Systems Research// MIS Quarterly 28, 75–99.

95. Howell F., McNab R. Simjava: a discrete event simulation library for java// Simul. Series 30 (1998) 51–56.

96. http://dev.eclipse.org/viewcvs/.

97. http://www.eclipse.org/org/.

98. http://www.opendesign.org.

99. https://bugs.eclipse.org/bugs/.

100. Hu, Z., Zhu, L., Ardi, C., Katz-Bassett, E., Madhyastha, H. V., Heidemann, J., et al. (2014). The need for end-to-end evaluation of cloud availability. In M.Faloutsos & A.Kuzmanovic (Eds.), Passive and active measurement. Proceedings of the 15 th international conference (PAM 2014), Los Angeles, CA, USA. LNCS (Vol. 8362, pp. 119–130). Springer.

101. Iansiti M., Levien R., 2004. Strategy as ecology// Harvard Business Review 82 (3), 68–78.

102. Improving energy efficiency for transactional workloads in cloud en-

vironments/ N.T.T. Ho, M. Gribaudo, B. Pernici// Proc. of the Eighth Int. Conf. on Future Energy Systems, e-Energy 2017, Shatin, Hong Kong, China, May 16–19, 2017, (2017), pp. 290–295.

103. Improving reliability and performances in large scale distributed applications with erasure codes and replication/ M. Gribaudo, M. Iacono, D. Manini// Fut. Gener. Comput. Syst. 56 (2016) 773–782.

104. Iqbal, W., Dailey, M.N., Carrera, D., & Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Generation Computer Systems, 27, 871–879.

105. Jakobik A. et al. Towards secure non-deterministic meta-scheduling for clouds// 30th European Conf. on Modelling and Simulation, ECMS 2016, Regensburg, Germany, May 31, - June 3, 2016, Proceedings. (2016), pp. 596–602.

106. Jakobik A., D. Grzonka, F. Palmieri, Non-deterministic security driven meta scheduler for distributed cloud organizations, Simul. Modell. Pract. Theory (2018), https://doi.org/10.1016/j.simpat.2016.10.011.

107. Jansen S., Brinkkemper S., 2008. Applied multi-case research in a mixed-method research project: customer configuration updating improvement// Steel A.C., Hakim L.A. (Eds.), Information Systems Research Methods, Epistemology and Applications, 2008.

108. Jansen S., Brinkkemper S., Finkelstein A., 2000. Business network management as a survival strategy: a tale of two software ecosystems// CEUR-WS Proceedings of the 1st International Workshop on Software Ecosystems, Washington, USA.

109. Jansen S., Finkelstein A., Brinkkemper S., 2009. A sense of community: a research agenda for software ecosystems// Proceedings of the International Conference on Software Engineering, pp. 187–190.

110. Jayakumari D. S., Dr. Mathusoothana S Kumar R, Dr. P. Venkadesh and Dr. S.V. Divya. Computer Networks. San International, 2024.

https://doi.org/10.59646/cn/283

111. Kabbedijk J., Brinkkemper S., Jansen S., van der Veldt B., 2009. Customer involvement in requirements management: lessons from mass market software development// Proceedings of the Requirements Engineering Conference, 2009 (RE ’09), Atlanta, USA, pp. 281–286.

112. Kecskemeti G., Dissect-cf: a simulator to foster energy-aware scheduling in infrastructure clouds, Simul. Modell. Pract. Theory 58 (2015) 188–218.

113. Khazaei H., J. Misic, V.B. Misic, A fine-grained performance model of cloud computing centers, IEEE Trans. Parallel Distrib. Syst. 24 (11) (2013) 2138–2147.

114. Khazaei, H., Misic, J., & Misic, V.B. (2012). Performance analysis of cloud computing centers using m/g/m/m+r queuing systems. IEEE Transactions on Parallel and Distributed Systems, 23, 936–943.

115. Kliazovich D., P. Bouvry, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. 62 (3) (2012) 1263–1283.

116. Ko D.-G., Kirsch L.J., King W.R., 2005. Antecedents of knowledge transfer from consultants to clients in enterprise system implementations// MIS Quarterly 29, 59–85.

117. Kosar T., P.E.M. Lpez, P.A. Barrientos, M. Mernik, A preliminary study on various implementation approaches of domain-specific language, Inf. Softw. Technol. 50 (5) (2008) 390–405.

118. Kravets O.Ja., Aksenov I.A., Redkin Yu.V., Rahman P.A., Mutin D.I., Amoa Kouadio-kan Armel Geoffroy, Ermolova M.A. Tools for designing complex software systems based on special linguistic constructions and algorithms for their processing// *International Journal on Information Technologies and Security,* vol.16, no.4, 2024, pp. 15-24. https://doi.org/10.59035/XOIF5262. WOS:001369038100002.

119. Kryvinska, N., & Strauss, C. (2013). Conceptual model of business services availability vs. Interoperability on collaborative IoT-enabled eBusiness platforms. In Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence (pp. 167–187). Berlin Heidelberg: Springer.

120. Le Berre D., Rapicault P., 2009. Dependency management for the eclipse ecosystem: eclipse p2, metadata and resolution// Proceedings of the 1st International Workshop on Open Component Ecosystems, IWOCE '09, ACM, New York, NY, USA, pp. 21–30.

121. Li Z.E., H. Zhang, L. O'Brien, R. Cai, S. Flint, On evaluating commercial cloud services: a systematic review, 86 (2013) 2371–2393.

122. Liu H. Research on control method of blocking jamming in HF communication system. Digit. Technol. Appl. 37(1), 29–30 (2019)

123. Liu P., Cai Y., Lu G. Space environment data transfer system based on BBR congestion control algorithm. Chin. J. Space Sci. 39(1), 117–123 (2019)

124. Liu, Y., Liu, W., Song, J., & He, H. (2015). An empirical study on implementing highly reliable stream computing systems with private cloud. Ad Hoc Networks, 35, 37–50.

125. Lloyd, W., Pallickara, S., David, O., Lyon, J., Arabi, M., & Rojas, K. (2013). Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: Towards performance modeling. Future Generation Computer Systems, 29, 1254–1264.

126. Magalhães, D., Calheiros, R.N., Buyya, R., & Gomes, D.G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. Computers & Electrical Engineering, 47, 69–81.

127. Marsan M.A. et al. Modelling with Generalized Stochastic Petri Nets, first, John Wiley & Sons, Inc., New York, NY, USA, 1994.

128. Messerschmitt D.G., Szyperski C., 2003. Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, Cambridge, MA, USA.

129. Ministry of Information and Communication Technology, India, Draft Policy on Open Standards for Egovernance, 2008.

130. Model Reduction and Approximation/ P. Benner, M. Ohlberger, A. Cohen, K. Willcox// Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017, https://doi.org/10.1137/1.9781611974829.

131. Performance and Reliability Analysis of Computer Systems; An Example-based Approach Using the SHARPE Software Package/ R.A. Sahner, K.S. Trivedi, A. Puliafito. - Kluwer Academic Publisher, 1996.

132. Performance modeling and analysis of software architectures: an aspect-oriented UML based approach/ K. Cooper, L. Dai, Y. Deng// Sci. Comput. Program. 57 (1) (2005) 89–108.

133. Performance Modeling of Big Data-Oriented Architectures/ M. Gribaudo, M. Iacono, F. Palmieri. - Springer International Publishing, Cham, pp. 3–34. 10.1007/978-3-319-44881-7_1.

134. Popp K., 2010. Goals of software vendors for partner ecosystems – a Pratitioner's view// Proceedings of the First International Conference on Software Business, Brussels, Belgium, pp. 185–192.

135. Rajala R., Rossi M., Tuunainen V.K., 2003. A framework for analyzing software business models// Proceedings of the 11th European Conference on Information Systems, Naples, Italy.

136. Ranekc M., Ylitalo J., Peltonen J., Koivisto N., Mutanen O.-P., Autere J., Valtakoski A., Pentikdinen P., 2009. National Software Industry Survey. Helsinki University of Technology.

137. Raymond, E.S., 1999. The Cathedral and the Bazaar, 1st ed. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

138. Riehle D., 2009. The commercial open source business model// Proceedings of the 15th Americas Conference on Information Systems, San Francisco, USA, Lecture Notes in Business Information Processing, pp. 18–30.

139. Runeson P., Hust M., 2009. Guidelines for conducting and reporting case study research in software engineering// Empirical Software Engineering 14 (2), 131–164.

140. Sahner, R.A., Trivedi, K., & Puliafito, A. (2012). Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package. Springer Science & Business Media.

141. Sanders W. Integrated frameworks for multi-level and multi-formalism modeling// Petri Nets and Performance Models, 1999. Proc. The 8th Int. Workshop on, (1999), pp. 2–9.

142. Scacchi W., 2007. Free/open source software development: recent research results and methods// Zelkowitz M.V. (Ed.), Architectural Issues, vol. 69 of Advances in Computers. Elsevier, pp. 243–295.

143. Serrano, D., Bouchenak, S., Kouki, Y., de Oliveira Jr, F.A., Ledoux, T., Lejeune, J., Sens, P. (2016). SLA guarantees for cloud services. Future Generation Computer Systems, 54, 233–246.

144. Sherriff M., Williams L., 2006. Devcop: a software certificate management system for eclipse// Proceedings of the International Symposium on Software Reliability Engineering, IEEE Computer Society, Los Alamitos, CA, USA, pp. 375–384.

145. Simonin B.L., 1999. Ambiguity and the process of knowledge transfer in strategic alliances// Strategic Management Journal 20 (7), 595–623.

146. Singleton P. Performance modelling — what, why, when and how// BT Technol. J. 20 (3) (2002) 133–143.

147. Souer J., van Mierloo M., 2008. A component based architecture for web content management: runtime deployable web manager component bundles// Proceedings of the International Conference on Web Engineering, Yorktown Hights, USA, pp. 366–369.

148. Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 34(1), 1–11.

149. Sun, D.W., Chang, G.R., Gao, S., Jin, L.Z., & Wang, X.W. (2012). Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. Journal of Computer Science and Technology, 27, 256–272.

150. Tavis, M., & Fitzsimons, P. (2012). Web application hosting in the AWS cloud.

151. The SIMTHESys multiformalism modeling framework/ M. Iacono, E. Barbierato, M. Gribaudo// Comput. Math. Appl. (64) (2012) 3828–3839.

152. Tian W. et al. Open-source simulators for cloud computing: comparative study and challenging issues// Simul. Modell. Pract. Theory 58 (2015) 239–254.

153. Toosi, A.N., Calheiros, R.N., & Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. ACM Computing Surveys, 47, 7:1–7:47.

154. Trivedi K.S. Symbolic hierarchical automated reliability and performance evaluator// DSN ’02: Proc. of the 2002 Int. Conf. on Dependable Systems and Networks, IEEE Computer Society, Washington, DC, USA, 2002, p. 544.

155. Tuffin B. et al. Simulation versus analytic-numeric methods: illustrative examples// Proc. of the 2Nd Int. Conf. on Performance Evaluation Methodologies and Tools, ValueTools ’07, , ICST, Brussels, Belgium, Belgium, 2007, pp. 63:1–63:10. http://dl.acm.org/citation.cfm?id=1345263.1345344.

156. Turkarslan, S. (2015). Technical article for SQL server and Azure, Application patterns and development strategies for SQL server in Azure virtual machines, Oct. 2014, Available online at: http://msdn.microsoft. com/en-

us/library/azure/dn574746. aspx#comparison ; 2015, (acce ssed on 22.05.2015).

157. Van Angeren J., Kabbedijk J., Jansen S., Popp K., 2011. A survey of associate models used within large software ecosystems// Proceedings of the third Workshop on Software Ecosystems, Brussels, Belgium, CEUR-WS, pp. 27–39, volume 746.

158. Van de Weerd I., Brinkkemper S., Nieuwenhuis R., Versendaal J., Bijlsma L., 2006. Towards a Reference Framework for Software Product Management// IEEE Computer Society, Los Alamitos, CA, USA, pp. 319–322.

159. Van der Schuur H., Jansen S., Brinkkemper S., 2011. The power of propagation: on the role of software operation knowledge within software ecosystems// Proceedings of the Conference on Management of Emergent Digital Ecosystems (MEDES 2011), San Francisco, USA.

160. Vishwanath, K.V., & Nagappan, N. (2010, June). Characterizing cloud computing hardware reliability. In Proceedings of the 1st ACM symposium on Cloud computing (pp. 193–204). ACM.

161. Visser E. WebDSL: A Case Study in Domain-Specific Language Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 291–373.

162. Vittorini V. et al. The OsMoSys approach to multi-formalism modeling of systems// Softw. Syst. Model. 3 (1) (2004) 68–81.

163. Wei, Bing & Xiao, Limin & Wei, Wei & Song, Yao & Huo, Zhisheng. (2020). A high-bandwidth and low-cost data processing approach with heterogeneous storage architectures. Personal and Ubiquitous Computing. 27. 10.1007/s00779-020-01383-6.

164. West J., 2007. The economic realities of open standards: black, white and many shades of gray// Greenstein, S., Stango, V. (Eds.), Standards and Public Policy. Cambridge University Press, pp. 87–122.

165. When and how to develop domain-specific languages/ M. Mernik, J. Heering, A.M. Sloane// ACM Comput. Surv. 37 (4) (2005) 316–344.

166. Xu L., Brinkkemper S., 2007. Concepts for product software. European Journal of Information Systems 5, 531–541.

167. Xu, X., Jin, H., Wu, S., & Wang, Y. (2015). Rethink the storage of virtual machine images in clouds. Future Generation Computer Systems, 50, 75–86.

168. Yin R.K., 2003. Case study research// Vol. 5 of Applied Social Research Methods, 3rd ed. SAGE.

169. Zhang, Q., Cherkasova, L., & Smirni, E. (2007, June). A regression based analytic model for dynamic resource provisioning of multitier applications. In Fourth International Conference on Autonomic Computing (ICAC'07) (pp. 27–27). IEEE.