

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.04.04 «Программная инженерия»
Магистерская программа Разработка программно-информационных систем

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
на тему
Методика и инструментальные средства
управления жизненным циклом активных
правил

Студент Курбатова Мария Николаевна (ФИО полностью)
Руководитель Шибанов С.В. (фамилия, инициалы)
Нормоконтролер Такташкин Д.В. (фамилия, инициалы)
Рецензент А.А.А. (фамилия, инициалы)
(должность, место работы) Н.П.П., Критический

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой Макарычев П.П. (фамилия, инициалы)

Работа защищена с отметкой 5 (протокол заседания ГЭК от 30.06.21 № 1)

Секретарь ГЭК Попова Н.А. (фамилия, инициалы)

Пенза, 2021

Содержание

Введение	5
1 Анализ задачи управления жизненным циклом активных правил.....	9
1.1 Актуальность информационных систем на основе обработки событий	9
1.1.1 Задача обработки событий в информационно-управляющих системах...	9
1.1.2 Концепция и виды активных правил.....	16
1.1.3 Анализ систем обработки активных правил	21
1.2 Анализ процессов управления в жизненном цикле правил	23
1.3 Сравнительный анализ методик управления и нотаций моделирования активных правил	25
1.3.1 Современные модели и нотации представления правил.....	26
1.3.2 Сравнительный анализ моделей и нотаций	27
1.3.3 Обоснование выбора моделей и нотаций активных правил для сервиса управления жизненным циклом	31
1.4 Анализ требований к инструментальным средствам поддержки жизненного цикла активных правил	33
1.4.1 Функциональные требования	33
1.4.2 Требования к пользовательскому интерфейсу	34
1.4.3 Архитектурные требования	35
Выводы	40
2 Методика и алгоритмы управления жизненным циклом активных правил	41
2.1 Формальное представление активных правил.....	41
2.1.1 Формальное представление обособленных активных правил	41
2.1.2 Формальное представление взаимодействующих правил	42
2.1.3 Сценарии исполнения активных правил	44
2.2 Объектная модель метаданных.....	46

	3
2.3 Методика управления жизненным циклом активных правил	47
2.3.1 Процессы жизненного цикла активных правил	47
2.3.2 Подготовка метаданных для автоматизации процессов управления активными правилами.....	48
2.3.3 Создание типов отслеживаемых событий.....	48
2.3.4 Создание типов активных правил	50
2.4 Анализ корректности и эффективности активных правил	53
Выводы	56
3 Инструментальные средства управления жизненным циклом активных правил в рамках микросервисной архитектуры	57
3.1 Сервис потоковой обработки событий и исполнения активных правил...	57
3.1.1 Назначение и функциональные возможности сервиса потоковой обработки событий и исполнения активных правил.....	57
3.1.2 Архитектура сервиса потоковой обработки событий и исполнения активных правил	61
3.1.3 Обоснование выбора технологий и средств разработки	63
3.2 Инструментальные средства управления жизненным циклом активных правил в составе сервиса	64
3.3 Информационное и программное обеспечение инструментальных средств	67
3.3.1 Репозиторий метаданных для СУБД PostgreSQL	67
3.3.2 Программное обеспечение REST-сервера	68
3.3.3 Программное обеспечение клиентского приложения	70
Выводы	71
Заключение	72
Список использованных источников	74
Приложение А ГЛОССАРИЙ.....	77

	4
Приложение Б КОД ПРОГРАММЫ	79

Введение

Большинство баз данных и других систем, обрабатывающих различные данные, включают в себя систему правил, в соответствии с которыми информационная система реагирует на какие-либо события, происходящие во внешней среде. Многие событийно-ориентированные системы работают в пассивном режиме и реагируют на события только после соответствующей команды от пользователя. Причем анализ произошедших событий также зачастую проводится самим пользователем либо системой при его непосредственном участии. В случае систем с активными правилами, они не требуют участия пользователя и выполняют заданные правилами действия в реальном времени сразу после событий, произошедших во внешней среде. Такая оперативная реакция позволяет значительно уменьшить обязанности пользователя, освобождая для него временные ресурсы, а также обеспечить своевременную реакцию на любые события, влияющие на данные, содержащиеся в системе. В последние несколько лет в мире ежесекундно происходит множество событий. Часть из них делают многие данные, содержащиеся в базах, неактуальными и тем самым вынуждают постоянно обновлять содержимое баз данных. В связи с этим особенную актуальность приобретают событийно-ориентированные системы, способные моментально реагировать на события или предоставлять пользователю информацию о них и возможность предпринять какие-либо действия после анализа произошедших событий. Большое количество составных событий увеличивает степень необходимой автоматизации системы. Пользователю также приходится выполнять довольно трудоемкую задачу – решение, как система должна реагировать на сложные правила, состоящие из нескольких, и как реагировать на события, произошедшие с разницей в определенный интервал времени. Записи таких правил могут превратиться в большие формулы, в которых трудно ориентироваться и редактировать.

Для того чтобы упростить задачу пользователя по составлению правил, можно разработать методику составления активных правил. Ориентируясь на

эту методику, пользователь сможет легко составлять и редактировать активные правила обработки событий, в результате чего необходимость его вмешательства в работу системы значительно снизится. В условиях современного мира освобождение человеческих ресурсов является одной из самых актуальных задач в сфере информационных технологий. Один из способов решения этой проблемы – создание наиболее точной и быстрой системы работы с жизненным циклом правил с удобным интерфейсом, позволяющим с минимальными трудозатратами создать новые либо отредактировать существующие правила реакции на события. Также необходимо разработать методику управления жизненным циклом активных правил, которая поможет эффективно хранить и исполнять их.

Объектом исследования является жизненный цикл активных правил, процессы конструирования и анализа коллизий взаимодействия активных правил, принципы построения автоматизированных средств поддержки жизненного цикла активных правил. **Предметом исследования** являются модельные представления активных правил, методика конструирования и алгоритмы статического анализа активных правил, программная архитектура и пользовательский интерфейс инструментальных средств управления жизненным циклом активных правил.

Цель и задачи работы.

Целью являются исследование и разработка автоматизированных инструментальных средств на основе единой методики управления жизненным циклом активных правил, включая конструирование и модификацию активных правил, анализ корректности и эффективности активных правил, аудит активных правил в процессе исполнения.

Для выполнения поставленной цели требуется выполнить следующие задачи:

- 1) исследовать и систематизировать процессы жизненного цикла активных правил в событийно-ориентированных системах;

2) провести анализ современных подходов к моделированию бизнес-логики информационно-управляющих систем и выработать требования к нотации представления активных правил;

3) исследовать и обосновать выбор модельного представления активных правил для инструментальных средств управления жизненным циклом активных правил;

4) разработать методику управления жизненным циклом активных правил, включая алгоритмы конструирования, анализа корректности и аудита активных правил;

5) разработать программный прототип инструментальных средств, поддерживающих методику управления жизненным циклом активных правил, обладающих развитым пользовательским интерфейсом и обеспечивающих конструирование, анализ и аудит активных правил для сервиса исполнения активных правил.

Научная новизна, оригинальность идеи в работе:

- 1) определены основные этапы жизненного цикла активных правил;
- 2) разработаны алгоритмы конструирования событий;
- 3) разработаны алгоритмы конструирования правил;
- 4) разработан алгоритм статического анализа правил.

Методы проведенных исследований. В ходе исследования и разработки применялись методы информационного поиска, методы сравнительного анализа и проектирования систем и программного обеспечения.

Основные положения, выносимые на защиту:

- 1) формальное представление обособленных и взаимодействующих активных правил с помощью алгебры Хоара;
- 2) методика управления жизненным циклом активных правил: алгоритмы конструирования и анализа правил и событий;
- 3) микросервисная архитектура и программная реализация сервиса потоковой обработки событий и исполнения активных правил.

Апробация результатов исследований. Основные положения и результаты исследований докладывались на следующих научно-технических конференциях:

- XVII Международная научно-техническая конференция «Новые информационные технологии и системы» (НИТиС-2020);
- VII Ежегодная Всероссийская межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. проблемы и перспективы» (Пенза, 2020);
- VII Ежегодная Всероссийская межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. проблемы и перспективы» (Пенза, 2021);
- научно-технический журнал «Промышленные АСУ и Контроллеры» (Москва, 2021).

Структура и объем диссертации. Диссертация состоит из введения, 3 глав, заключения, списка использованных источников, включающего 25 источников и 2 приложения. Материал диссертации изложен на 84 листах, включает 25 рисунков, 9 таблиц и 8 страниц приложений.

1 Анализ задачи управления жизненным циклом активных правил

Данная глава посвящена анализу и сравнению процессов управления жизненным циклом активных правил, нотаций моделирования, а также обзору систем, использующих активные правила, и выявлению их преимуществ и недостатков.

1.1 Актуальность информационных систем на основе обработки событий

Работа событийно-ориентированных систем строится вокруг событий, происходящих во внешней среде или внутри системы, и реакции на эти события. Большинство событийно-ориентированных систем являются пассивными. Это означает, что они требуют участия пользователя, который решает, как поведет себя система при определенных условиях и событиях, и производит изменения в системе. В случае с активными правилами система хранит определенный набор правил, согласно которому реагирует на произошедшие события без участия пользователя.

1.1.1 Задача обработки событий в информационно-управляющих системах

Обработка событий является неотъемлемой частью современных информационных систем. Любые действия, совершенные пользователем, пришедшие с датчиков, считывающих данные наблюдательных приборов, и действия внутри самой системы могут в определенной степени влиять на ее работу. Авторы в [1 – 2] определяют событие (event) как любое существенное явление, которое происходит или может произойти. Событие либо происходит полностью, либо не происходит вообще, и является существенным, поскольку может влиять на некоторое действие. Событие может быть частью бизнес-процесса (например, выдача заказа, приземление самолета, считывание данных датчиком) или мониторинговой информацией об информационной инфраструктуре, программном обеспечении и бизнес-процессах.

События можно классифицировать по различным критериям. Основные виды событий представлены в таблице 1.

Таблица 1 – Классификация событий в системах с активными правилами

Вид события	Описание
По уровню сложности	
Простое	Событие, которое невозможно разложить на более простые.
Сложное	Событие, содержащее в себе два или более простых событий или других сложных событий, связанных между собой операциями алгебры логики.
По уровню соответствия заданной структуре	
Стандартное	Событие, имеющее стандартную структуру.
Нестандартное	Событие, не подходящее под стандартную структуру.
По уровню планирования	
Ожидаемое	Событие, которое ожидает поток обработки событий.
Неожидаемое	Событие, которое не ожидает ни один из потоков обработки событий.
По уровню предсказуемости	
Запланированное	Событие, наступление которого запланировано в данный момент времени.
Незапланированное	Событие, наступление которого не запланировано в данный момент времени.
По уровню формальности	
Зарегистрированное	Событие, которое зарегистрировано в системе.
Незарегистрированное	Событие, которое не зарегистрировано в системе.

В разных источниках существует множество предложений классификаций событий с учетом контекста предметной области или функциональности правила. Типы зависят от конкретных характеристик организации или предприятия. Таким образом, существуют разные типы правил в зависимости от сферы деятельности. В большинстве систем есть события делятся на простые и сложные. Простым называется событие, не содержащее в себе элементы других событий. Для сложного события исследователи вводят в [3] такие понятия, как инициатор, детектор и терминатор. Инициатором сложного события является первое элементарное событие, возникновение которого начинает сложное событие. Детектором сложного события является элемент, возникновение которого обнаруживает

сложное событие. Терминатор сложного события является элементом, означающим прекращение сложного события.

Функции обработки событий также делятся на простые и сложные. Когда происходит существенное событие, оно инициирует последующее действие (действия), и каждое появление события обрабатывается независимо, действует простая обработка событий. Она включает в себя такие виды обработки, как преобразование схемы событий из одной формы в другую, пополнение события полезной нагрузкой с дополнительными данными, перенаправление события из одного канала или потока в другой, генерирование нескольких событий на основе полезной нагрузки одного из них.

Сложная обработка событий предусматривает выявление шаблонов, охватывающих несколько независимых событий, для порождения новых составных событий. Сложная обработка событий включает в себя обработку наборов событий с целью обнаружения важной бизнес-ситуации. Обычно такая обработка означает применение к набору событий ряда оценочных условий или ограничений. События могут принадлежать к различным типам и происходить в течение определенного периода времени. Они могут коррелировать по многим аспектам, включая причинно-следственную связь, временное и пространственное соотношение и т.д. Получаемая при такой обработке событий информация является объемной, но при этом не должна являться сложной для пользователя.

Обработка событий применяется в довольно многих видах информационных систем (ИС) в различных предметных областях: фондовые рынки, здравоохранение, военная сфера, транспорт, энергетика и многие другие. Приведем примеры некоторых из них.

Исследователи К. Канди и В. Шульц занимались изучением обработки событий в области логистики и транспорта. В [4] они приводят пример использования таких событий, как прибытие и отбытие транспортного средства определенного типа, прибытие товара в пункт назначения, различные штатные ситуации, происходящие во время доставки товаров.

В [5] авторы описывают возможность применения обработки событий при изменении цен на акции, также при их покупке или продаже. В военной сфере важно отслеживать как можно большее число событий для их последующего анализа и принятия решений. Джонс М. приводит [6] примеры обработки таких событий, как обнаружение вражеской техники или солдат и их перемещение. В [2] авторы рассуждают, как можно применить обработку событий в здравоохранении: отслеживать переход от препандемического состояния к пандемическому в случае эпидемии, также возможно отслеживать изменение температуры, давления и других показателей у пациентов.

На данный момент чаще всего применяется потоковая обработка и обработка сложных событий. Исследователи Д. Ванг и Р. Эллисон в своей работе [7] рассматривают обработку сложных событий в CEP-системах (Complex Event Processing, Обработка Сложных Событий). Все интересующие события приходят из потенциально бесконечного потока. Каждый экземпляр события (например, e_i) учитывается модулем отслеживания событий и имеет две метки времени: время приложения и системное время. Время приложения e_i относится к дискретному моменту его возникновения, назначенному источником события, обозначаемому как $e_i.ts$. При этом системное время экземпляра события назначается механизмом CEP с использованием системного времени, обозначаемого как $e_i.sts$. Подобные экземпляры события могут быть сгруппированы по типам событий (например, детекторы). В реальных приложениях необработанные данные о событиях обычно дополняются семантически более богатой информацией. Следовательно, семантическая информация, касающаяся приложения, должна поддерживаться в хранилище. Главными операциями над потоком событий являются удаление из очереди и постановка в очередь, то есть добавление события в конец потока. Эти основанные на очереди операции являются общими для большинства CEP- и потоковых систем. Запрос CEP использует входной поток, выполняет указанную семантику запроса и затем добавляет результирующие события в выходной поток (если таковой есть).

Для потоковой обработки событий имеется, по крайней мере, 3 разных технологии программных систем, являющихся потенциально применимыми к решению обработки объемных данных с приемлемыми задержками. Это технологии СУБД и процессоров потоковой обработки, описанные в [2], а также системы SCADA, описанные в [8].

Системы управления базами данных (СУБД) широко используются по причине их возможности надежно хранить большие наборы данных и эффективно обрабатывать поступающие от людей запросы. СУБД, управляющие базами данных в основной памяти, могут обеспечить более высокую производительность, чем традиционные СУБД, за счет устранения обращений к дискам для большинства операций при наличии достаточного объема основной памяти. Поточковые данные поступают в СУБД напрямую или через загружающее их приложение. Данными, поддерживаемыми СУБД, может манипулировать набор приложений. Клиент может использовать эти заранее построенные приложения, часто с параметрами, задаваемыми во время выполнения, а также может написать новые приложения на языках общего назначения, таких как C++ или Java, используя встраиваемые SQL-вызовы СУБД.

Процессоры потоковой обработки (Stream Processing Engines, SPE) специально разрабатываются для работы с потоковыми данными и в последнее время привлекают внимание в качестве альтернативы СУБД. SPE выполняет обработку в стиле SQL над входящими сообщениями по мере их поступления, без обязательного их сохранения. Для сохранения состояния в случае необходимости можно использовать традиционную SQL-базу данных, встроенную в систему. Для выражения логики потоковой обработки в SPE используются специализированные примитивы и конструкции (например, временные окна).

SCADA-системы (Supervisory Control and Data Acquisition, диспетчерское управление и сбор данных) предназначены для контроля над технологическими процессами в реальном времени. На данный момент во многих подобных системах используются алгоритмы линейного сжатия

данных, в которых неактуальные события удаляются из репозитория. Идея удаления старых данных в пользу новых не подходит для всех сценариев приложений, в которых интересные события, хотя и истекли, могут по-прежнему иметь отношение к задачам обнаружения знаний по потокам данных. Поэтому модель линейного сжатия должна быть адекватно улучшена за счет внедрения удобства достижения так называемой модели сжатия потока нелинейных данных. Эта модель преследует идею сохранения более низкой степени аппроксимации для совокупной информации об этих показаниях потока данных, связанных с интересными событиями, и, наоборот, более высокой степени приближения для совокупной информации о других показаниях потока данных, т.е. малоинтересных. Методологии обнаружения событий по потокам данных могут давать гораздо лучший результат благодаря объединению промежуточных и окончательных результатов добычи в самых последних и прошлых чтениях, которые относятся к значимым событиям с истекшим сроком годности.

Обработка событий в системах с пассивными правилами осуществляется средствами систем управления базами данных с помощью механизма хранимых процедур и триггеров. Триггер – это хранимая процедура особого типа, которую СУБД запускает при возникновении сопоставленных триггеру событий. Триггеры предназначены для реализации узкоспециализированных, но полезных в ряде ситуаций сценариев обработки событий. Например, в Oracle присутствует механизм автономных транзакций. Данный механизм позволяет успешно завершить вложенную транзакцию, откатив родительскую. В результате триггер становится способен не только отменить нежелательное действие, но и сохранить сведения о попытке его совершения. Специализированными средствами выявления событий на разных уровнях абстракции являются системы обработки сложных событий (Complex Event Processing, CEP). Ключевой особенностью таких систем является возможность создания пользовательских типов событий. События могут быть сгенерированы как внешней средой, так и самой системой обработки событий по заданным

пользователем правил. В качестве примеров программных продуктов, реализующих данную концепцию, можно выделить IBM InfoSphere, Oracle CEP, Microsoft StreamInsight и SAP Event Insight. Перечисленные продукты имеют возможность глубокой интеграции с СУБД своего производителя, что позволяет обеспечить высокую скорость обработки событий. Данный класс систем реализует процессный подход к управлению компанией и позволяет автоматизировать сквозные бизнес-процессы, а также быстро вносить в них изменения. В таблице 2 перечислены наиболее известные системы обработки событий.

Таблица 2 – Средства обработки событий

Название	Ключевые особенности
Microsoft StreamInsight	<ul style="list-style-type: none"> – непрерывная обработка бесконечных последовательностей событий (потоков событий) из нескольких источников; – отслеживание, анализ и выполнение действий с данными в практически реальном времени.
SAP BusinessObjects Event Insight	<ul style="list-style-type: none"> – интеллектуальная сетка событий, которая распределяет обработку и мониторинг событий в режиме реального времени из нескольких источников данных в распределенной сети; – механизм обработки сложных событий.
Oracle Complex Event Processing	<ul style="list-style-type: none"> – выявление закономерностей в реальном времени; – высокая доступность, масштабируемость и надежность; – легкая интеграция: анализ событий в гетерогенных источниках.
IBM WebSphere iLOG BRMS	<ul style="list-style-type: none"> – создание и редактирование правила, используя естественный язык; – оперирование запросами при поиске правил в репозитории, проведение анализа изменений в правилах; – проверка правил на синтаксические ошибки или опечатки; – встроенный редактор запросов позволяет создавать запросы на языке Business Query Language; – тестирование и проведение симуляционного анализа.

В исследовании [9] автор Перри Л. упоминает обработку событий в системах, использующих концепцию Интернета вещей (Internet of things, IoT), а также применение данной концепции в различных производственных областях.

Обработка событий в IoT включает в себя следующие этапы:

- поглощение;
- оценка;
- агрегация;
- корреляция;
- временной анализ.

При поглощении адаптеры принимают потоки данных из различных источников. Не имеет смысла хранить все данных, генерируемые датчиками, особенно если часть данных не представляет интереса. Обработка потока событий может стандартизировать данные по мере их поступления, применяя простые преобразования и правила, чтобы определить, требуется ли дополнительная последующая обработка. Если нет, данные (или событие) быстро отбрасываются, не занимая дополнительного потока обработки. Агрегация означает, что обработка потока событий может непрерывно рассчитывать необходимые метрики через временные окна. Корреляция означает, что обработка потока событий позволяет подключаться к нескольким потокам данных, чтобы отследить последовательность неких событий. Обработка нескольких потоков событий поможет, например, сравнить показания разных датчиков. При обработке потока событий используется понятие времени как контекста произошедшего события. Временной анализ может проводиться, например, для обнаружения мошенничества при слишком частых денежных транзакциях на банковском счету или для составления статистики по историческим данным.

1.1.2 Концепция и виды активных правил

В научной литературе существует множество вариантов классификаций правил с учетом бизнес-контекста или функциональности системы. Если это бизнес-контекст, типы правил определяются в соответствии с конкретными характеристиками организации. Таким образом, существуют разные типы правил в зависимости от сферы деятельности или политики каждой организации. В данном разделе рассматриваются основные типы правил в соответствии с функциональными возможностями

правила [10]. Основной проблемой классификации правил является отсутствие консенсуса. В большинстве классификаций использовались разные имена для одного и того же типа правила, а в некоторых случаях существуют типы правил, которые рассматриваются в некоторой классификации, но не в других. Объединение классификаций правил предлагается в, чтобы реорганизовать и представить все основные типы правил учетом системного представления и тех свойств, которые могут интересовать пользователя системы, создающего и редактирующего их. В таблице 3 приведены виды правил, предложенные классификациями основных правил.

Таблица 3 – Классификация правил

Виды правил	Описание
Структурные правила	Этими правилами являются предложения, которые определяют бизнес-концепции или некоторые функции, связанные со структурой организации.
Ограничения целостности	Это ограничения целостности бизнес-правил; поэтому любое действие, которое пытается их нарушить, будет отвергнуто.
Дифференцирующие правила	Этот тип правила позволяет генерировать новую информацию на основе известной информации.
Правила деятельности	Они определяют набор действий или последовательность операций, которые должны выполняться для достижения определенной цели.
Активные правила	Они определяют действие, которое должно выполняться как ответ на возникновение предопределенного события. Этот тип правила характеризуется наличием формата Событие-Условие-Действие (СУД). Условие является необязательным; поэтому иногда эти правила имеют формат СД.

Приведем примеры перечисленных выше правил.

Структурные правила:

- каждый проект должен быть связан с конкретным отделом;
- у каждого отдела должен быть менеджер.

Ограничения целостности:

- бюджет любого проекта не может превышать 300 000 долларов;
- крайний срок завершения любого проекта должен быть равен или меньше 24 месяцев.

Дифференцирующее правило: если категория сотрудника равна A1, то его бонус составляет 3 000 долларов США в месяц. Если категория A2, то его бонус составляет 2 000 долларов США в месяц;

Правило деятельности: окончательная степень студента по предмету под названием «Структуры данных» должен быть рассчитан следующим образом: добавьте оценки двух частичных оценок и разделите результат на два; эта степень считается Степень 1. Добавить оценку практической работы со степенью выпускного экзамена: Степень 2. Конечная степень получается следующим образом: $(\text{Степень 1} + \text{Степень 2}) / 2$.

Активные правила:

- каждый проданный продукт должен обновить свой запас;
- каждый раз, когда значение запаса обновляется; он должен быть проверен, если это значение равно или меньше минимального запаса.

Помимо классификации на основе событий и целей правила можно также разделить на простые и сложные. Сложные правила подразумевают вложенность и взаимозависимость правил друг от друга. Действие одного правила, как описано в [11], в реальном времени может инициировать одно или несколько других активных правил, которые могут, в свою очередь, запускать еще одно несколько других, и такое инициирование может продолжаться до прерывания выполнения внутренним или внешним событием.

Для представления активных правил используется классическая модель ЕСА (Event-Condition-Action). Авторы работы «Active database systems» выделяют 3 составные части правила: произошедшее событие, условие для срабатывания действия и само действие, которое выполнит система в случае истинности условия [12].

В системах управления активными правилами реализована, помимо функциональности стандартной базы данных, обработка произошедших событий и правил, активируемых этими событиями. В общем случае ЕСА-правила имеют структуру «вследствие события, если условие, то действие».

Условие может быть как простым, так и сложным (состоящим из нескольких условий, объединенных логической операцией). Действий, выполняющихся в случае истинности условия, также может быть несколько, а в случае ложности дополнительные действия могут быть не предусмотрены.

Характеристики активных правил определяются исследователями Ромеро Д. и Руттенем Е. в [11] следующим образом:

- правило активируется только событиями;
- его выполнение является автономным и не зависит от других правил в системе;
- реализует реакцию на входящее событие;
- он содержит защитное условие для выполнения таких действий.

Помимо обычных ЕСА-правил правил, включающих стандартный набор из события, условия и действия, существуют их модификации.

Безусловное правило (ЕА-правило) – правило с опущенным условием. Такое правило содержит в себе только событие и действие. Действия из данного правила являются обязательными к исполнению, так как для них не описано условие. Возможен вариант ЕА-правила, когда условие является частью действия.

Правило с неявным событием (СА-правило) – правило, в котором описано только условие и действие. Данный тип правил используется в ситуациях, когда одна из частей системы определяет событие. Например, изменение данных, для которых существуют ограничения, или добавление новой записи в журнал. Система в таком случае сама отправляет сигнал на вход обработчика событий. В таких правилах событие определено неявно – пользователь определяет лишь условие и действие для правила, а событие происходит автоматически.

Существуют также расширенные ЕСА-правила, в которых в случае невыполнения условия исполняется альтернативное действие. Формально они выглядят как «вследствие события, если условие, то действие 1, иначе действие 2».

ЕСА-модель активных правил обладает определенными недостатками. В качестве примера рассмотрим систему умного дома. Предположим, что два светильника в комнате запрограммированы с учетом различных форм поведения, следующие правила генерируют проблему цикличности. Пользователем задан следующий набор правил:

- переключен выключатель 1 – если светильник 1 включен – выключить светильник 2;
- переключен выключатель 1 – если светильник 1 выключен – включить светильник 2;
- переключен выключатель 2 – если светильник 2 включен – включить светильник 1;
- переключен выключатель 2 – если светильник 2 выключен – выключить светильник 1.

Первые два правила будут пытаться изменить состояние второго светильника в зависимости от состояния первого. Третье и четвертое правила будут пытаться поддерживать оба светильника в одном (текущем) состоянии. Подобные действия будут непрерывно генерировать цикличное выполнение правил, если не поступит другое событие [11]. Во избежание подобных ситуаций можно использовать правила с дополнительным компонентом – состоянием.

Классическая ЕСА-модель описывает реакцию системы на внешние события, не учитывая, что в реальных программных продуктах поведение объекта зачастую зависит от его состояния. Например, в приложении, где несколько пользователей могут редактировать одну таблицу, у каждой ячейки могут быть состояния «редактируется» и «свободна». Когда один пользователь редактирует содержимое ячейки, она недоступна другим пользователям.

Под состоянием понимается набор значений и связей объекта. При определении состояния не учитываются атрибуты, не оказывающие влияния на поведение объекта. Состояние включает в себя все комбинации значений и связей, характеризующиеся одинаковым откликом на события. Объекты

некоторого класса обладают конечным числом возможных состояний. В каждый момент времени объект может находиться только в одном состоянии. Объекты могут несколько раз менять состояние в течение времени своего существования.

Состояние можно рассматривать как дополнительное условие для выполнения действия, заданного в правиле: в конкретном состоянии игнорируются любые события, за исключением тех, поведение которых описано явным образом. Это свойство состояний позволяет наделить объекты системы свойством изменчивости – способностью изменять свое поведение в зависимости от возникающих событий.

Добавление в классическую ЕСА-модель понятия состояния сокращает количество необходимых правил и упрощает процесс разработки активной системы. Возможные типы поведения объекта группируются в соответствии с его состояниями, поведение объекта структурируется и становится более доступным для понимания. Модель правила, учитывающая состояние объекта, называется расширенной ЕСА-моделью или SECA-моделью (State – Event – Condition – Action).

1.1.3 Анализ систем обработки активных правил

Описание не только события, но и всего процесса реагирования на высоком уровне абстракции позволяет существенно сократить внесение изменений в логику работы информационной системы по мере развития бизнес-процессов за счет снижения доли участия программистов. Для решения задачи описания бизнес-логики на высоком уровне абстракции предназначены системы управления бизнес-правилами (Business Rule Management System, BRMS). Бизнес-правило – это логическое утверждение, которое может быть создано пользователями и является исполнимым. Примерами программных продуктов, реализующих концепцию систем управления бизнес-правилами, являются системы, описанные в 1.1.1. Продукты предоставляют программные средства для создания, развертывания, исполнения, настройки и анализа бизнес-правил. Данные

системы нашли свое применение в обработке платежей, управлении цепями поставок (supply chain management), межкорпоративном взаимодействии, принятии решений и составлении отчетности.

Можно сделать вывод о недостаточной общности существующих подходов к обработке событий, а также о необходимости единой концепции реагирования. В этом качестве может быть рассмотрена концепция активных баз данных (АБД). Система управления активными базами данных (СУАБД) в дополнение к функциональности традиционных СУБД должна быть способна обнаруживать события и реагировать на них на том же уровне абстракции, что и системы обработки сложных событий. В таблице 4 представлены системы, сочетающие логику реляционных и объектных СУБД и активных правил.

Таблица 4 – Реляционные и объектные активные СУБД

Название	Ключевые особенности
Starburst Rule system	<ul style="list-style-type: none"> – наличие языка создания правил; – сетевой эффект набора изменений данных. Использование переходных таблиц; – масштаб транзакции – набор.
Ariel	<ul style="list-style-type: none"> – поддержка правил «условие– действие»; – язык правил базируется на языке OPS5.
Ode	<ul style="list-style-type: none"> – основана на объектной парадигме; – использует язык запросов O++, основанный на C++.
HiPAC	<ul style="list-style-type: none"> – связана с пассивной объектной СУБД PROBE; – параллельное выполнение вызванных правил как подтранзакций; – сетевой эффект набора изменений.
Chimera	<ul style="list-style-type: none"> – поддержка дедуктивных правил; – условия и действия правил могут получать доступ к предыдущим состояниям БД либо к началу текущей транзакции, либо когда правило было запущено.
SAMOS	<ul style="list-style-type: none"> – реализована на верхнем уровне объектной СУБД ObjectStore; – наличие детектора событий, основанного на сетях Петри.
NAOS	<ul style="list-style-type: none"> – объектно-ориентированная СУБД; – каскадное исполнение и иерархия правил; – язык запросов основан на SQL.

В большинстве систем обработки активных правил пользователи сталкиваются с такой проблемой, как терминальность исполнения правил., которая означает, что порядок выполнения правил влияет на конечный результат. В [12] Д. Ванг и Р. Эллисон подробно рассматривают данную

коллизии и приводят в пример систему отслеживания процедуры дезинфекции медицинских работников, где состояние работника (безопасное / небезопасное) хранится в репозитории. Система решает, можно ли допустить сотрудника к работе на основании этого состояния. Процедура дезинфекции проводится некоторое время, и из-за задержки в обработке событий к работе может быть допущен неготовый сотрудник из-за того, что из репозитория два потока пытались считать и записать данные. Данная проблема присуща большинству систем обработки активных правил.

На основании проведенного анализа современных технологий обработки событий, можно отметить, что каждое средство способно эффективно решать задачи обработки событий, но только в определенной предметной области, и не является универсальным [13].

1.2 Анализ процессов управления в жизненном цикле правил

В данном подразделе проводится анализ процессов управления активными правилами на разных этапах их создания. Последовательность формирования активного правила приведена на рисунке 1.



Рисунок 1 – Жизненный цикл активного правила

Далее более подробно будет рассмотрен каждый из этапов.

Основополагающим этапом конструирования правила является выбор типа правила, так как от него зависят дальнейшие этапы описания правила. Тип правила определяет набор атрибутов, которые пользователю необходимо заполнить. В данном случае тип правила будет определен, когда пользователь закончит заполнять все его атрибуты.

Анализ корректности активных правил проводится для предупреждения возможных ошибок в процессе их исполнения. При добавлении нового правила в систему пользователь может не продумать вероятность появления коллизий при одновременной работе двух правил или возможность бесконечного вызова одного или нескольких правил.

Для обеспечения стабильной работы системы необходимо учитывать взаимодействие при их конструировании и (или) редактирования содержимого репозитория. Это означает, что при каждом изменении активных правил должна предоставляться информация о том, как это отразится на работе других правил.

Коллизии работы активных правил можно разделить на три группы [14]:

- избыточность;
- несогласованность;
- цикличность.

Избыточность означает, что в системе есть два или более правил, функциональность которых повторяется. Подобная проблема может возникнуть в больших системах, где правила создаются разными пользователями.

Несогласованность является следствием противоречивых команд в системе. Также она может возникнуть в том случае, когда порядок выполняемых действий в системе оказывает влияние на их конечный суммарный результат. Пример несогласованности был приведен в разделе 1.1.2.

Третья проблема, цикличность, возникает в тех случаях, когда правила инициируют выполнение других правил, из-за чего система не переходит в какое-либо стабильное состояние. Может сложиться такая ситуация, при которой одни правила запускают действия, которые являются событиями активации других правил. Цепочка действий и условий в определенный момент может замкнуться, благодаря чему цикл выполнения правил никогда не завершится.

Проверка активных правил при верификации называется статической проверкой. Она заключается в проверке вариантов взаимодействия правил и выделении среди них потенциальных ошибок, влияющих на стабильность работы системы. Динамическая проверка правил позволяет выявить некорректные взаимодействия уже во время обработки событий. С точки зрения грамотного проектирования, проверка правил до внедрения его в репозиторий является более предпочтительной, так как позволяет выявить угрозы сбоя системы заранее. Статическая проверка может проводиться как непосредственно перед добавлением правил в репозиторий, так и в дополнение к тестированию.

Управление правилами включает в себя такие действия как изменение, включение правила в работу (активация) и выключение.

Изменение правила может включать в себя как изменение любой из его частей, так и смену типа правила. В случае любого изменения правила оно должно заново пройти этап анализа корректности.

За активное и неактивное состояние правила может отвечать всего одна переменная булевского типа, но так как при выполнении правил между ними могут возникать коллизии, то для включения правила в работу необходимо учитывать уже работающие правила и то, как активация еще одного правила повлияет на поведение системы. Для контроля исполнения правила можно установить максимальное число правил, взаимодействующих с определенным объектом, или задать время выполнения.

1.3 Сравнительный анализ методик управления и нотаций моделирования активных правил

Нотация – это набор знаков и правил, которые используются для описания и моделирования бизнес-процессов. Нотация определяет, как на схеме обозначаются процессы, операции, события и т.д., и по каким правилам они соединяются между собой [7]. Бизнес-процесс определяется как логически завершенная цепочка взаимосвязанных и взаимодействующих повторяющихся видов деятельности [8]. В состав цепочки входят действия,

выполняемые по определенным бизнес-правилам. Под бизнес-правилами понимают способы реализации бизнес-функций в рамках бизнес-процесса, а также характеристики и условия выполнения бизнес-процесса [8].

В работе «Моделирование бизнес-правил для спецификации приложения для Активной Базы данных. Спецификация приложения» [9] есть некий набор описаний спецификаций моделирования бизнес-правил. Бизнес-правила представляют собой существенные элементы в парадигме деятельности. Они могут быть выражены с использованием различных методов моделирования, включая диаграммы перехода потока данных и состояния или диаграммы сети Петри. Каждый из этих методов кратко описан. Что касается подходов к моделированию, бизнес-правила появляются через модели данных и поведения.

1.3.1 Современные модели и нотации представления правил

В различных исследовательских работах, посвященных изучению активных правил, предлагается множество вариантов их представления. Наиболее часто используемые нотации приведены в таблице 5.

Таблица 5 – Нотации и модели представления активных правил

Наименование	Описание	Авторы и время появления
Семейство IDEF	<ul style="list-style-type: none"> – Прекращено развитие; – ограничено число процессов; – независимость размера блок-схемы от ПО; – сложность в освоении; – поддерживает большое количество ПО [7]. 	Научно-исследовательский Центр CALS, 1993 г.
eEPC	<ul style="list-style-type: none"> – Четко виден порядок выполнения; – можно описать большую часть составляющих бизнес-процесса; – неочевидно взаимодействие участников процесса; – отсутствие типов событий; – поддерживает большое количество ПО. 	Август-Вильгельм Шеер, 1990 г. В рамках разработки систем ARIS.

Продолжение таблицы 5

Наименование	Описание	Авторы и время появления
BPMN	<ul style="list-style-type: none"> – Развитие продолжается в настоящий момент; – наличие «дорожек»; – наличие разных типов потоков; – несколько вариаций моделирования потоков; – возможность перевода модели в программный код. 	Разработана организацией «Business Process Management Initiative (BPMI)» в 2004 г. , поддерживается группой компаний «Object Management Group».
DFD (диаграмма потоков данных)	<ul style="list-style-type: none"> – Используются для описания потока данных между процессами или между процессом и внешним блоком или сохраненными данными; – компонент действия структуры ЕСА соответствует процессу в структуре DFD; – не позволяют представить условия синхронизации для выполнения процесса. 	Разработали Гейн и Карсон в начале 90-х годов XX века.
STD (диаграмма состояний-перехода)	<ul style="list-style-type: none"> – Используется для представления поведенческих требований в терминах состояний и переходов; – видны взаимодействия между правилами; – не видны эффекты правил над объектами. 	–
Сети Петри	<ul style="list-style-type: none"> – Используются для описания и моделирования поведения системы; – узлы используются для представления событий и (или) условий, а переходы представляют собой действия; – высокая сложность спецификаций; – подходит для всех типов правил. 	Карл Петри в 1962 г.
ER	<ul style="list-style-type: none"> – Не позволяет дизайнеру формулировать отдельные события, условия и действия; – невозможно представить активные правила; – подходит для описания отношений между событиями. 	Питер Чен в 1976 г.

Для обзора были выбраны наиболее популярные и современные нотации и модели представления правил и бизнес-процессов.

1.3.2 Сравнительный анализ моделей и нотаций

Сравнение нотаций и моделей представления активных правил можно проводить по нескольким критериям. При выборе нотации для сервиса

конструирования и исполнения активных правил важно учесть следующие критерии:

- этап исполнения правила, на котором используется нотация, и конкретный процесс жизненного цикла;
- уровень стандартизации и формализации;
- удобство создания правил и восприятия их пользователем.

Этапы процесса исполнения были перечислены в разделе 1.1.2. На каждом этапе необходимо выделить разные части составляющие набора активных правил. На этапе возникновения события необходимо как можно более подробно описать его: к какому типу оно принадлежит, является ли оно зарегистрированным, иногда может играть роль время его возникновения и т.д. Срабатывание правила происходит в тот момент, когда возникает событие, указанное в нем. На данном этапе понадобится нотация, которая может представить событие как составляющую правила. Оценка условий исполнения правила проверяет, является ли истинным указанное в правиле условие. Составные условия объединены логическими операциями, поэтому для этого этапа подойдут нотации, в которых удобно представлять логические выражения. Планирование одновременного исполнения нескольких правил является самой сложной частью исполнения активных правил. Необходимо учитывать конфликты, которые могут возникать между правилами, взаимодействующими с одним объектом, для этого нужно представить одновременно весь набор правил, исполняющихся в данный момент. Во время выполнения действий, указанных в правиле, нужно представить данные и объекты, с которыми произойдут изменения.

Такой критерий, как степень стандартизации и формализации, определяет возможность приведения правил к общему виду, что предоставляет возможность работы с правилами как с объектами определенного типа, обладающего неким набором свойств.

Удобство создания и восприятия правил пользователем необходимо для обеспечения высокой скорости и эффективности создания правил. В этот

критерий входят простота синтаксиса нотации, возможности описать как можно больше элементов правила и связь между правилами.

Результаты сравнительного анализа нотаций и моделей, описанных в разделе 1.3.1, приведены в таблице 6.

Таблица 6 – Сравнительный анализ нотаций и моделей

Наименование	Номер этапа ЖЦ	Уровень стандартизации и формализации	Удобство
Семейство IDEF	2, 3	Высокий	Низкое
eEPC	2, 3, 5	Высокий	Среднее
BPMN	1, 2, 3, 4, 5	Низкий	Высокое
DFD	3	Средний	Среднее
STD	2, 3, 4, 5	Средний	Среднее
Сети Петри	1, 2, 3, 4, 5	Высокий	Среднее
ER	1, 2	Средний	Высокое

Нотации IDEF (методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем) обладают высоким уровнем формализации и позволяют представить все компоненты активного правила [10]. Но из-за сложности восприятия и громоздкого синтаксиса нотация не представляется удобной для пользователя.

В нотации eEPC(от англ. extended event driven process chain, расширенная событийная цепочка процессов [7]) моделирование сосредоточено на событиях. На ее схемах хорошо виден порядок выполнения процессов, события, документы, информационные системы и т.д. Минусом является невозможность представить взаимодействие между участниками процесса, что усложняет процесс редактирования правил пользователем и делает нотацию непригодной для моделирования взаимодействия правил в процессе исполнения.

BPMN (Business Process Model and Notation, нотация управления бизнес-процессами) является удобной, гибкой и функциональной [7]. В данной нотации определены так называемые «дорожки» – области в модели процесса, которые отображает все, что выполняет конкретный объект в

данном процессе. С помощью дорожек можно отобразить взаимодействие исполнителей исследуемого процесса. Набор обозначений нотации может быть использован для описания любого процесса или события, что делает BPMN подходящей для любого этапа исполнения активного правила. Также существует разделение потоков на типы, что облегчает чтение и анализ моделей бизнес-процессов [7]. Также имеется возможность дополнить стандартный набор значков своими вариантами, что позволяет адаптировать нотацию к любым потребностям. Существует множество вариаций моделирования процесса. Это позволяет на основе BPMN создать собственную нотацию.

Диаграммы потоков данных (DFD) используются для описания потока данных между процессами или между процессом и сохраненными данными. Компонент действия структуры ЕСА соответствует процессу в структуре DFD. Эти диаграммы без расширений не могут представлять контрольные и временные события в качестве потока данных [11]. Кроме того, DFD не позволяют представить условия синхронизации для выполнения процесса. Зато данные диаграммы удобны для отображения течения информации в пределах процесса или системы, также для изображения входных и выходных данных, точек хранения информации и путей ее передвижения между источниками.

Диаграммы состояния-перехода (STD) хорошо описывают взаимозависимости между правилами, но эффекты правил над объектами представить с их помощью нельзя. Графики состояния-перехода не содержат явных символов для всех компонентов бизнес-правила [12]. Хотя переходы могут быть помечены событиями, условиями и действиями, нотация диаграмм не поддерживает маркировку правил. В целом, нотация удобна и наглядна для пользователя.

Сети Петри (СП) часто используются для описания и моделирования поведения системы [14]. Они представляют из себя двудольный граф, в котором есть последовательность мест и переходов в пути, образованном последовательными ребрами. СП характеризуется такими свойствами, как

бесконфликтная, безопасная и достижимая. В моделировании бизнес-правил узлы используются для представления событий и (или) условий, а переходы представляют собой действия. В [12] было предложено использовать расширенные СП для описания элементарных или сложных событий. Расширенные СП используют маркированные кромки для моделирования альтернативных результатов в зависимости от условия. Используя эти расширения, можно смоделировать структуру всех бизнес-правил. Однако недостатком использования СП является сложность, связанная с их спецификацией. СП не дает пользователю представления о сложности поведения системы.

В модели Сущность-Связь (ER) правила выражаются через семантику модели, включающей отношения, мощности и экзистенциальные зависимости. Модель ER не позволяет пользователю формулировать отдельные события, условия и действия. Правило в этой модели может представлять только ограничение целостности. Отношения между событиями или между событием и правилом обозначаются довольно удобно.

1.3.3 Обоснование выбора моделей и нотаций активных правил для сервиса управления жизненным циклом

В результате проведенного в разделе 1.3.2 анализа нотаций представления активных правил были выбраны несколько из них (таблица 7):

Таблица 7 – Выбранные нотации

Этап исполнения	Наименование нотации
Возникновение события	Сети Петри
Срабатывание правила	Сети Петри
Оценка условий исполнения правила	Сети Петри
Планирование одновременного исполнения нескольких правил	BPMN
Выполнение действий	BPMN

Для надежной и стабильной работы система обработки активных правил должна обеспечивать контроль взаимодействия правил как при их проектировании, так и в процессе исполнения. Статический анализ активных

правил в процессе их проектирования направлен на выявление возможных вариантов взаимодействия правил и выделение среди них потенциальных угроз с точки зрения стабильности работы системы. Динамический анализ активных правил обеспечивает контроль корректности исполнения и поиска наиболее эффективного варианта исполнения. Исследования А. Зудова и С. Шибанова [15] показывают, что классический подход к описанию потенциальных взаимодействий основан на принципе иницируемости вызова правил. Одним правилом потенциально иницируется вызов другого, если действие первого правила подразумевает формирование события того же типа, к которому привязано второе правило. Пусть $t(e_k)$ – тип события e_k . Тогда иницируемость вызова правила r_j правилом r_i означает выполнение условия $t(e_g|e_g \in r_i^k) = t(e_h|\exists \alpha r_i^k)$. В рамках данного подхода в качестве модели описания взаимодействия правил предлагается граф срабатываний G_{tr} (triggering graph), вершины которого соответствуют активным правилам $\{r_1 \dots r_n\}$, а ориентированные рёбра означают иницируемость вызова правил: $G_{tr} = (\{r_1 \dots r_n\}, \{r_i, r_j | t(e_a|e_a \in r_i^e) = t(e_a|\exists \alpha r_i^e)\})$. Такой подход обеспечивает возможность правильно представить как события, так и правила на тот момент, когда они должны быть запущены, и правильно оценить условия их срабатывания.

Основным недостатком описанного подхода является то, что вызов активных правил не всегда приводит к запуску действия, так как контекст события должен удовлетворять условию правила. Данное обстоятельство означает, что наличие путей в графе срабатывания не гарантирует, что соответствующий вариант последовательного инициирования правил возможен. По данной причине для представления правил на двух последних этапах их работы правила была выбрана нотация BPMN. В этой нотации используется понятие потока, с помощью которого управление разделяется на части. Это позволяет планировать одновременное исполнение нескольких правил и выполнение действий [15]. Примеры представлены на рисунках 2 и 3.



Рисунок 2 – Разделение потока управления



Рисунок 3 – Выполнение действий

Таким образом, для сервиса конструирования и исполнения активных правил были выбраны 2 нотации представления: Сети Петри и BPMN.

1.4 Анализ требований к инструментальным средствам поддержки жизненного цикла активных правил

Анализ требований позволяет определить, какие задачи решает разрабатываемая система, какие возможности она предоставляет пользователю и то, как именно он может с ней взаимодействовать. Также в процессе определения требований выбирается архитектура системы и технологии для ее программной реализации.

1.4.1 Функциональные требования

Функциональные требования описывают поведения системы с точки зрения пользователя.

В [16] Форс упоминает функции систем управления активными правилами. Они включают в себя работу с активными правилами (создание, удаление, изменение), отслеживание и обработку событий, а также анализ активных правил на возможные коллизии.

Функциональные требования к сервису управления активными правилами представлены в таблице 8.

Таблица 8 – Функциональные требования

Роль	Группа функций
Разработчик правил	Конструирование активных правил
Администратор сервиса	Настройка сервиса
Система	Анализ коллизий взаимодействия активных правил
	Обработка событий
	Исполнение активных правил
	Журнализация и анализ данных журналов событий

Перечисленный функционал является обобщенным вариантом и в ходе исследования будет уточнен.

1.4.2 Требования к пользовательскому интерфейсу

Пользовательский интерфейс совмещает все аспекты технологии и дизайна, которые обеспечивают взаимодействие человека с технической системой [15].

В понятие пользовательского интерфейса компьютерной системы входят следующие составляющие:

- графическая среда – то, что видит пользователь на экране;
- набор управляющих элементов (кнопки, списки, диалоговые окна и т.д.) и их расположение на экране;
- технологии взаимодействия пользователя с системой.

Основные требования к пользовательскому интерфейсу следующие:

- функциональность (соответствие задачам пользователя);
- соответствие технологии;
- понятность и логичность;
- обеспечение высокой скорости работы пользователя;
- обеспечение защиты от человеческих ошибок;
- быстрое обучение пользователя;
- субъективное удовлетворение пользователя.

Пользовательский интерфейс сервиса управления жизненным циклом активных правил должен включать в себя возможность создания правила, просмотра списка, выбора и редактирования правила, а также проведения статического и динамического анализа.

Автор исследования [16] Форс. Т. описывает концепцию пользовательского представления правил в активных базах данных. Он предлагает создать графический интерфейс, в котором составные части правила будут изображены в виде отдельных элементов, соединенных линиями. Было решено, что наиболее удобным и лаконичным станет интерфейс, состоящий из двух частей. Графическая часть представляет из себя поле, на котором компоненты представлены в виде геометрических фигур, а линиями показаны связи между ними. Пользователь может перемещать эти объекты по полю, добавлять и удалять связи и компоненты с помощью добавления или удаления графических объектов. Табличная часть необходима для подробного редактирования правила. Эта часть представляет из себя таблицу со всеми атрибутами правила, в которой можно указать численные значения для его компонентов, например, время срабатывания, число одинаковых событий, необходимых для срабатывания правила и т.д.

Переход из графической части в табличную осуществляется двойным щелчком мыши по компоненту правила, а из табличной в графическую – нажатием кнопки «Сохранить» или «Отмена». При переключении между режимами интерфейса пользователь видит то же правило, что было открыто в предыдущем режиме, что позволяет ему лучше представлять общую карту правил.

1.4.3 Архитектурные требования

В исследовательских работах [17, 18] рассматриваются концепции возможной архитектуры систем, основанных на обработке событий. Реализация архитектуры осуществляется в контексте систем обработки потока событий. Исследователи выделяют три основные группы архитектур:

платформы обработки потока событий, платформы распределенной обработки потоков событий и библиотеки CEP (или движки CEP).

Платформы обработки потоков событий (EPP) – это программное обеспечение обработки потоков событий, которое предоставляет модели программирования высокого уровня, такие как выразительные языки обработки событий (EPL) и встроенные функции для фильтрации событий, корреляции и абстракции. IBM InfoSphere Streams, DataTorrent Streaming (RTS), Apache Apex, SQL-Stream s-Server, WSO2 CEP, StreamMine3G и т. д. – некоторые известные примеры EPP-систем. В них события общего характера доставляются заинтересованным сторонам через так называемого посредника сообщений (то есть службу уведомлений). Посредники сообщений могут быть сконфигурированы для фильтрации определенных нежелательных событий, так что дальнейшая обработка может быть выполнена в чистой системе обработки событий.

Платформы обработки распределенных потоков событий (DSCP) (также известные как механизмы обработки распределенного потока (DSPE)) – это платформы обработки потока событий, которые предоставляют явную поддержку для распределения вычислений по нескольким узлам в вычислительном кластере.

Существуют четко различимые категории DSCP, основанные на их архитектуре. В литературе чаще всего выделяют такие категории, как DSCP первого поколения, второго поколения и геораспределения.

DSCP имеют много общего с реляционными базами данных. Ярким примером этой категории является Borealis [17]. Проект Borealis подчеркнул три фундаментальных требования к механизмам потоковой обработки, таких как динамическая проверка результатов запроса, динамическая модификация запроса и гибкая и масштабируемая оптимизация.

К представителям второго поколения относятся S4 и Storm. Существует два основных варианта DSCP второго поколения, называемых обработкой по событию и микропакетированием (то есть обработкой дискретизированного/пакетного потока). Системы EP для обработки каждого

события обрабатывают все события индивидуально. Они обеспечивают минимальные задержки при низкой пропускной способности по сравнению со своими аналогами (микропакетирование). Примеры систем, которые реализуют обработку для каждого события, включают в себя S4, Storm, Flink и т. д. Новая парадигма распределенной обработки данных, называемая пакетной обработкой потока, в последнее время изменила концепцию архитектуры систем EP. Эти системы обеспечивают высокую пропускную способность, но обеспечивают относительно высокую задержку. Comet была одной из первых систем, которая внедрила эту технику, и она была интегрирована с DryadLINQ, Spark Streaming, Storm Trident и т.д. Spark-Streaming, Samza, Apache Flink (ранее Stratosphere), Trill, являются примерами для DSCP, которые имеют возможности как пакетной, так и потоковой обработки.

Геораспределенная обработка потоков отличается проведением высокопроизводительной обработки потока событий через центры обработки данных. Чтобы обеспечить эффективную передачу событий, в таких системах должна быть адаптирована пакетная передача. Модели задержки могут быть созданы для поддержки динамической адаптации размера пакета к облачной среде и для обеспечения многопутевой потоковой передачи через облачные узлы.

Библиотека CEP – это программный компонент, который предназначен для обнаружения сложных событий. Библиотека CEP должна уметь идентифицировать значимые закономерности, отношения и абстракции данных среди явно не связанных между собой событий и немедленно запускать ответ. Это легковесный компонент, который может быть встроен в другие системы обработки событий, такие как DSCP и EPP. Некоторые известные примеры библиотек CEP, появившихся недавно: Esper, Siddhi [17], ruleCore, Cayuga.

Во всех перечисленных видах архитектур в [16] выделяются следующие компоненты:

- источники и приемники событий;

- адаптеры и соединители событий;
- хранилище событий;
- ядро обработки событий;
- адаптеры действий.

Источники предоставляют события для последующей обработки. Приемники событий различных типов получают события, создаваемые системой обработки событий. Любой технологический ресурс потенциально является как источником, так и приемником событий. Адаптеры и соединители предоставляют средства для интеграции с источниками и приемниками событий. Помимо оперативного хранения в памяти данных событий для их захвата, накопления и дальнейшей обработки, необходимо постоянное хранилище (например на основе интегрированной РСУБД), предоставляющее средства для тестирования и симуляции. Кроме того, важно обладать средствами поддержки, предоставляющими доступ к предыдущим данным, используемым для идентификации сложных событий. Ядро событий включает в себя набор функций, отвечающий за логику обработки. Как правило, эта логика представляется в форме правил среды разработки, разделяющих события на актуальные и неактуальные. Для системы важно включать адаптеры для сторонних приложений мониторинга бизнес-активности и аналитических панелей; кроме того, важно учитывать то, как действия инициируют автоматизированные бизнес-процессы посредством, например, интеграции с решениями по управлению бизнес-процессами, а также отправляют события внешним потребителям посредством промежуточного ПО и промышленных шин.

Важной частью системы является репозиторий, хранящий не только компоненты активных правил, но и данные, составляющие их контекст. В ходе исследований на кафедре МОиПЭВМ была предложена архитектура репозитория метаданных.

Репозиторий активных правил представляет собой базу данных, содержащую составные части правил и производимые с БД операции. Добавление нового типа события подразумевает создание метаданных для

функций условий правил и выходных параметров, а также реализует возможность вычисления значений этих функций и позволяет хранить области значений. Репозиторий может управляться не той же СУБД, которая используется для источников событий. Кроме того, добавление типа правил может повлечь добавление ещё одного наблюдаемого объекта.

В информационных системах чаще всего используются две категории информационных ресурсов – данные и метаданные. Данные представляют сущности предметной области системы, их свойства и связи с другими сущностями. Метаданные – это данные о данных, они могут описывать свойства как самих данных, так и информационной системы в целом, отдельных ее функций, поддерживаемых технологий, пользователей и т.д. Как описано исследователями в [19], конкретные функции метаданных и их состав в значительной мере зависят от специфики рассматриваемой системы и характера конкретных информационных ресурсов. В общем случае роль метаданных в информационных системах можно описать следующим образом:

- описание свойств информационной системы, ее механизмов и информационных ресурсов в среде инструментов автоматизации разработки таких систем (CASE-средств);
- обмен сведениями между различными инструментами CASE и/или приложениями информационных систем;
- хранение сведений о свойствах и содержании информационных ресурсов для механизмов управления данными в информационных системах любого рода;
- обеспечение механизмов интеграции и информационных ресурсов из разных источников сведениями о свойствах этих ресурсов;
- хранение информации, необходимой для осуществления реинжиниринга информационных систем;
- хранение сведений о системе, ее информационных и других ресурсах для различных приложений и пользователей системы.

Репозиторий, как предлагается в [17], отвечает за поддержку модели контекстуальных знаний, которая собирает соответствующую информацию о среде, в которой развернуто приложение.

Выводы

Результаты исследований и анализа, представленные в настоящей главе, приведены далее.

Системы, нацеленные на обработку событий, широко применяются в таких областях, как здравоохранение, транспорт, энергетика, фондовые рынки, военная сфера, интернет вещей и многих других. На данный момент отсутствует единый подход к обработке событий в разных по своей концепции информационных системах.

Для обработки событий в информационно-управляющих системах и реагирования на их возникновение в виде выполнения тех или иных действий могут применяться активные правила, которые представляют собой структуру вида Event-Condition-Action (ECA) (Событие-Условие-Действие).

Современные нотации моделирования бизнес-процессов не могут быть использованы для представления активных правил. Специализированные нотации активных правил широко применяются для их документирования, но имеют ограничения по применению в программных инструментальных средствах.

Основными этапами жизненного цикла активных правил являются конструирование, анализ корректности активных, тестирование, управление состоянием активных правил.

Выделенные этапы должны быть автоматизированы в инструментальных средствах управления активными правилами.

2 Методика и алгоритмы управления жизненным циклом активных правил

Данная глава диссертации посвящена разработке методики и алгоритмов конструирования, анализа и исполнения активных правил. Приведены ситуации, возникающие при обработке событий, которые выявляются в ходе проведения динамического анализа. Предложены методы устранения коллизий и алгоритмы оценки эффективности работы активных правил.

2.1 Формальное представление активных правил

2.1.1 Формальное представление обособленных активных правил

Как было описано ранее, классической моделью активных правил является ЕСА-модель (Event-Condition-Action). Правило можно представить как

$$r = \langle e, c, a \rangle.$$

Существуют также расширенные активные правила, описанные в разделе 1.1.2. Такие правила имеют альтернативное действие или набор действий на случай невыполнения условия.

$$r = \langle e, c, \{a_{\text{main}}, a_{\text{alt}}\} \rangle,$$

где a_{main} — основное действие;

a_{alt} — дополнительное действие.

Классическая языковая конструкция активного правила с дополнительным действием представлена на листинге 1.

Листинг 1 – Языковое представление активного правила

```
CREATE RULE <имя_правила>
  ON <тип_события> TO <условие_проверки_состояния>
  WHEN <условие>
    DO <основное_действие>
    [ELSEDO <дополнительное_действие>]
```

Аналогичным способом можно представить другие виды активных правил.

Активные ЕСА-правила включают в себя произошедшее в системе или во внешней среде событие (event), одно или несколько условий срабатывания правила (condition) и действия (action), которые необходимо выполнить. ЕСА-правило не учитывает состояние объекта, которое может повлиять на его поведение. Состояние позволяет наделить объекты системы свойством изменчивости. По этой причине появились расширенные модели ЕСА, в которых появился еще один компонент – состояние. Такие модели имеют название SECA-модели (State–Event–Condition–Action). Добавление в модель ЕСА-правил понятия состояния сокращает количество порождаемых событий, уменьшает объем обрабатываемых правил и упрощает процесс разработки системы. Активные правила SECA-модели, наряду с компонентами ЕСА-правил, включают условие проверки состояния объекта-источника событий

$$r = \langle s, e, c, a \rangle.$$

При возникновении события проверяется состояние наблюдаемого объекта. Если состояние объекта не совпадает с заданным состоянием (условие проверки состояния принимает значение «ложь»), то событие не генерируется. Подобный подход описан исследователями в [20].

2.1.2 Формальное представление взаимодействующих правил

Для описания взаимодействия активных правил была выбрана нотация взаимодействующих последовательных процессов (CSP, Communicating Sequential Processes) или алгебра Хоара [22]. В нотации CSP поведение процесса рассматривается относительно событий, в которых процесс принимает участие. Если процессы участвуют в одном событии, то они являются взаимодействующими.

Каждому процессу X ставится в соответствие множество событий αX , в которых он выполняется. Процесс, не участвующий в событиях, обозначается «STOP». Процессы могут представлять из себя события и подпроцессы и обозначаются с помощью символа префикса $\rightarrow$. Запись вида $(ek \rightarrow X)$ обозначает процесс, который сначала участвует в событии ek , а

затем ведет себя как подпроцесс X . Для описания процессов, которые ведут себя по-разному в зависимости от того, каким оказалось исходное событие, используется оператор выбора $|$. Процессы, работающие как независимые подпроцессы с общими событиями, описываются с помощью оператора параллельного выполнения $\parallel$.

Множество экземпляров событий обозначается как E , а множество элементарных событий через Ω . Активное правило, вызванное для обработки некоторого события, является процессом, который совершает последовательность событий-операций с i и затем останавливается. В нотации Хоара оно будет выглядеть следующим образом

$$rik = (di1k \rightarrow (di2k \rightarrow \dots (dink \rightarrow STOP))),$$

где k – номер экземпляра события $e \in E$;

rik – правило при обработке экземпляра события ek ;

$dink \in \Omega \cup E$ – некоторое элементарное событие или экземпляр промежуточного события, генерируемый правилом напрямую.

Активное правило, обрабатывающее по-разному различные экземпляры событий, представим как процесс с выбором

$$r_i = (|(e_k \rightarrow r_i^k)|),$$

где $i=1,2,3\dots$ – номер активного правила;

$k=1,2,3\dots$ – номер экземпляра события.

Активная база данных в целом описывается как процесс, в котором правила и триггеры обрабатывают события параллельно

$$ADB = (||r_i||)(||tr_k||),$$

где $i=1,2,3\dots$ – номер активного правила;

$k=1,2,3\dots$ – номер экземпляра события.

Вызову процедуры генерации экземпляра события ставится в соответствие процесс, запускающийся элементарным событием и генерирующий экземпляр события

$$tr_k' = (\omega \rightarrow (e' \rightarrow STOP)),$$

где e' – генерируемый экземпляр события;

k' – номер триггера.

2.1.3 Сценарии исполнения активных правил

В [17] исследователи рассматривают концепцию сценариев исполнения активных правил, при которых они могут иметь несколько вариантов завершения.

Для обработки подобных ситуаций существуют многоблочные правила, включающие в себя альтернативные действия, охватывающие полную группу событий.

На рисунках 4 и 5 представлены сценарии исполнения классического и расширенного ЕСА-правила.



Рисунок 4 – Сценарий исполнения классического правила

Исполнение расширенного правила отличается от исполнения обычного проверкой выполнения условия, активирующего действие, и исполнением альтернативного действия.



Рисунок 5 – Сценарий исполнения расширенного ЕСА-правила

Рассмотрим пример офисного помещения с электронной системой отслеживания времени прихода сотрудников на работу. Если сотрудник должен отметить у проходной и непосредственно у входа в офис, то возможны 4 комбинации событий:

- отметка у проходной, отметка у офиса;
- отметка у проходной, нет отметки у офиса;
- нет отметки у проходной, отметка у офиса;
- нет отметки у проходной, нет отметки у офиса.

Первый из перечисленных случаев является нормальной комбинацией событий, при которой система не предпринимает никаких действий, кроме записи времени прибытия на работу. Остальные комбинации событий

являются отклонениями от нормы и каждое из них требует определенных действий со стороны системы.

2.2 Объектная модель метаданных

Для обеспечения процессов конструирования и анализа активных правил требуется создать репозиторий, который позволит хранить все виды правил, а также всю информацию, необходимую для взаимодействия с ними. Также для разных пользователей необходимо разграничить доступ к базе данных и предоставить возможность выбрать при регистрации роль, от которой будет зависеть возможность взаимодействия с правилами.

На рисунках 6 и 7 представлены структуры фрагментов репозитория, в которых хранятся данные пользователей и активные правила.

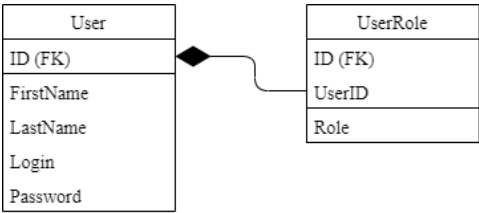


Рисунок 6 – Логическая модель репозитория пользовательских данных

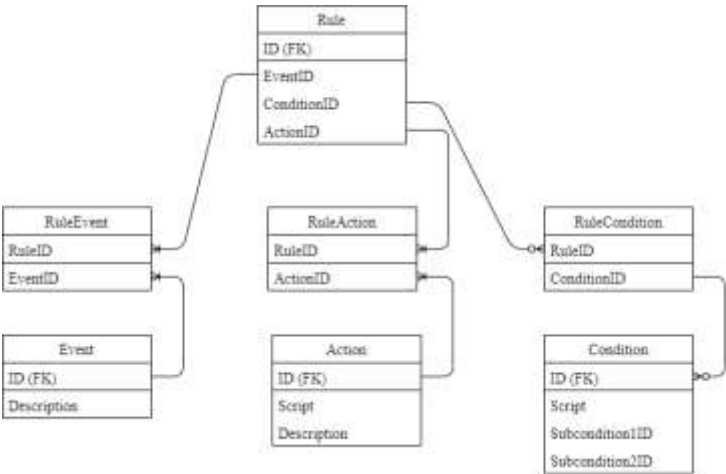


Рисунок 7 – Логическая модель репозитория активных правил

В репозитории активных правил содержатся данные о правилах и составляющих их событиях, условиях и действиях. Условия сложного типа содержат в себе ссылки на составные условия. Также условие может отсутствовать.

2.3 Методика управления жизненным циклом активных правил

Управление жизненным циклом активных правил является комплексной задачей с особенным подходом к решению на каждом отдельном его этапе. Необходимо включить в методику управления способы формализации правил и событий, а также методы их создания.

2.3.1 Процессы жизненного цикла активных правил

Жизненный цикл активного правила включает в себя несколько этапов:

- конструирование правила;
- проверка правила и анализ корректности;
- включение в работу;
- исполнение;
- отключение правила.

Последовательность этапов из раздела 1.2 дополнена в ходе исследования и представлена и на рисунке 8.



Рисунок 8 – Жизненный цикл активного правила

Создание правила происходит по разным алгоритмам конструирования в зависимости от его типа. Процесс создания правила также включает в себя процесс создания события. Проверка корректности правила проходит в 2 этапа: статический и динамически анализ. Статическая проверка состоит в анализе вариантов взаимодействия правил и выделении среди них потенциальных ошибок, влияющих на стабильность работы системы. Динамическая проверка позволяет выявить некорректные взаимодействия правил во время обработки событий. Активация правила включает его в

список действующих правил. С момента включения правила система будет отслеживать описанное в нем действие и проверять условие. Исполнение правила представляет из себя выполнение действия, описанного в правиле. Отключение правила означает его удаление из списка действующих.

2.3.2 Подготовка метаданных для автоматизации процессов управления активными правилами

2.3.3 Создание типов отслеживаемых событий

Процесс создания события включает в себя выбор его типа, наблюдаемого объекта и само формальное описание события. Алгоритм создания события отличается в зависимости от того, является событие простым или сложным. Создание события представлено на рисунках 9–10.

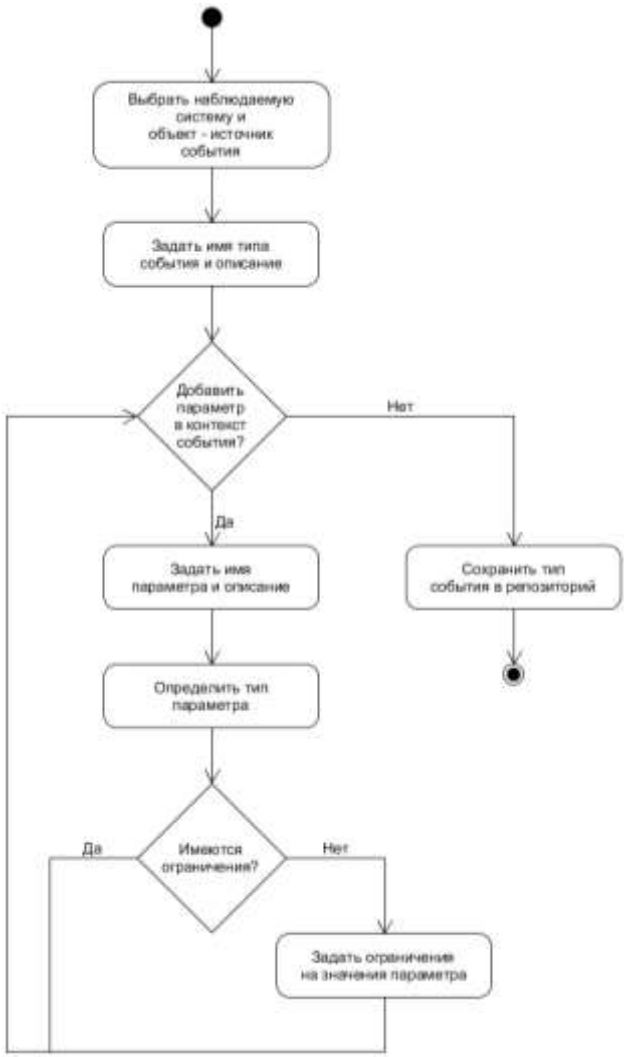


Рисунок 9 – Алгоритм создания простого события

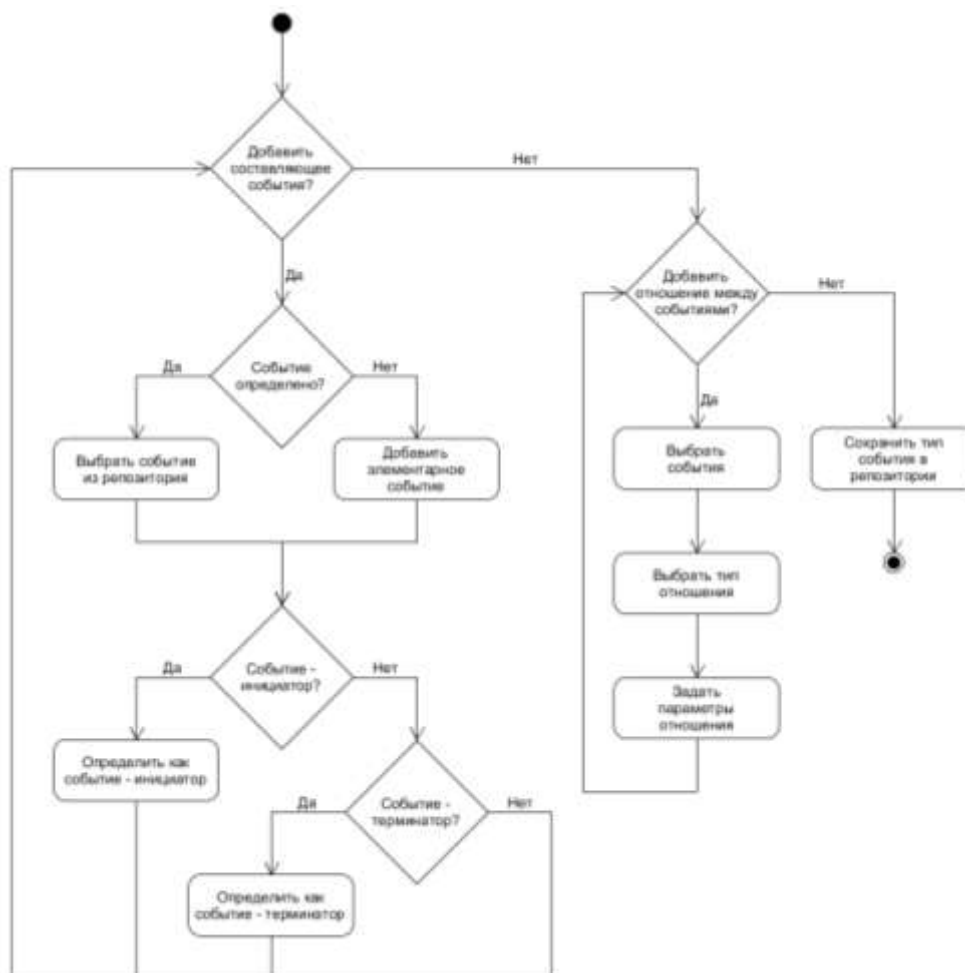


Рисунок 10 – Алгоритм создания сложного события

Помимо создания, управление событиями также включает в себя их отслеживание и обработку. Для обработки события необходимо учитывать его контекст. Он включает в себя подсистему, из которой было передано событие, и время его прихода. В [23] контекст определяется как обстоятельства, при которых произошло событие. Оценка контекста напрямую влияет на обработку события, то есть на то, будет ли оно учитываться системой и запустит ли действие. Поэтому в модуль обработки происходящих событий необходимо передавать объект, содержащий в себе событие и его контекст. Если событие является сложным, то есть состоит из нескольких атомарных событий, то для его описания применяется пара событий: начальное (инициатор) и конечное (терминатор). Между простыми событиями в составе сложного могут быть описаны логические отношения, имеющие тип и параметры. В промежутке между ними могут происходить также и другие события, а значит, продолжающиеся во времени события

могут перекрываться, что является еще одной причиной введения в сервис потоковой обработки событий и исполнения активных правил.

2.3.4 Создание типов активных правил

Как и в случае с событием, управление активным правилом включает в себя его создание, проверку, изменение в случае необходимости, включение и выключение, а также динамический анализ выполнения правила.

Конструирование активного правила также зависит от его типа. Виды активных правил перечислены в разделе 1.1.2. На рисунках 11 и 12 представлены разновидности алгоритма конструирования правила.

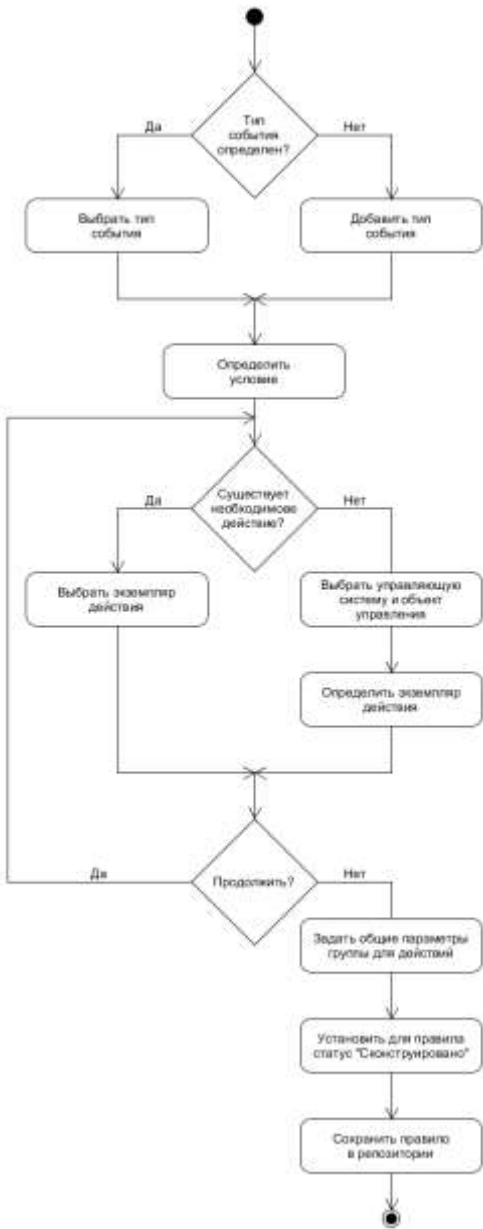


Рисунок 11 – Создание классического правила

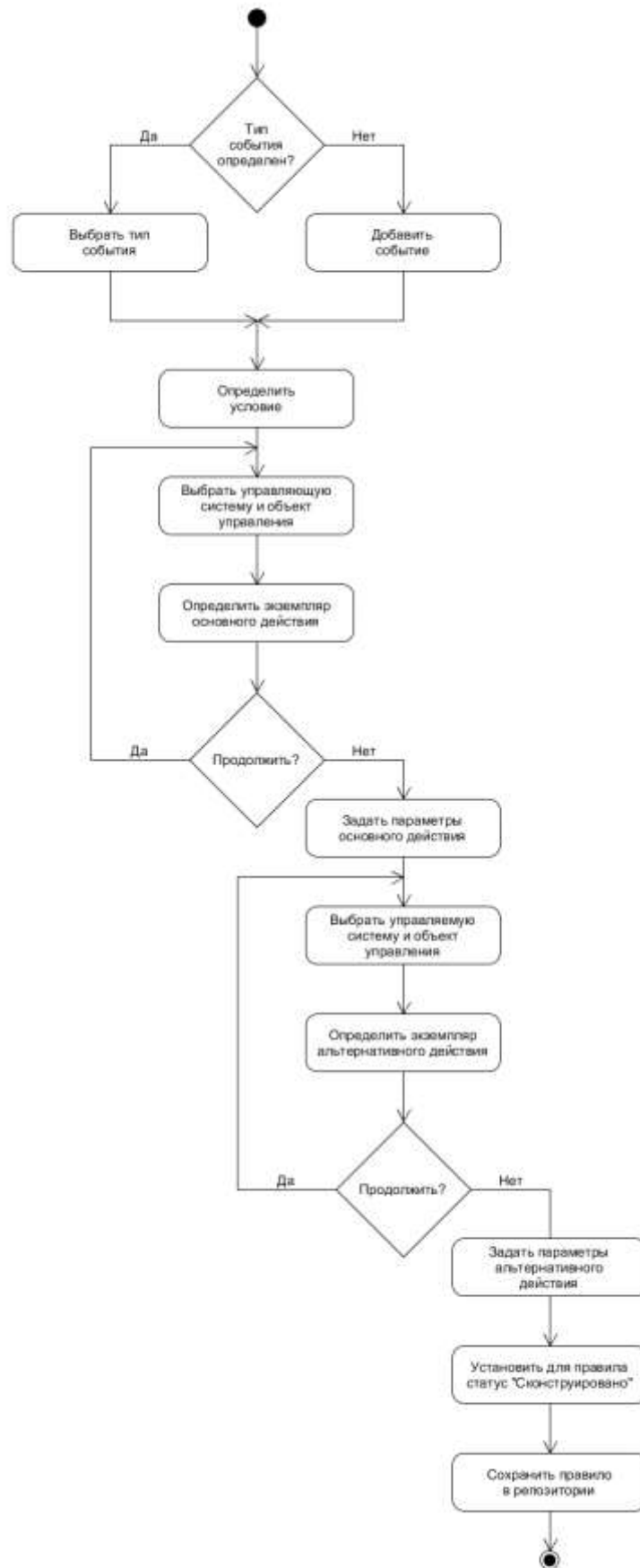


Рисунок 12 – Создание расширенного правила

Помимо перечисленных типов правил можно также отдельно выделить алгоритм создания многоблочного правила, который представлен на рисунке 13.

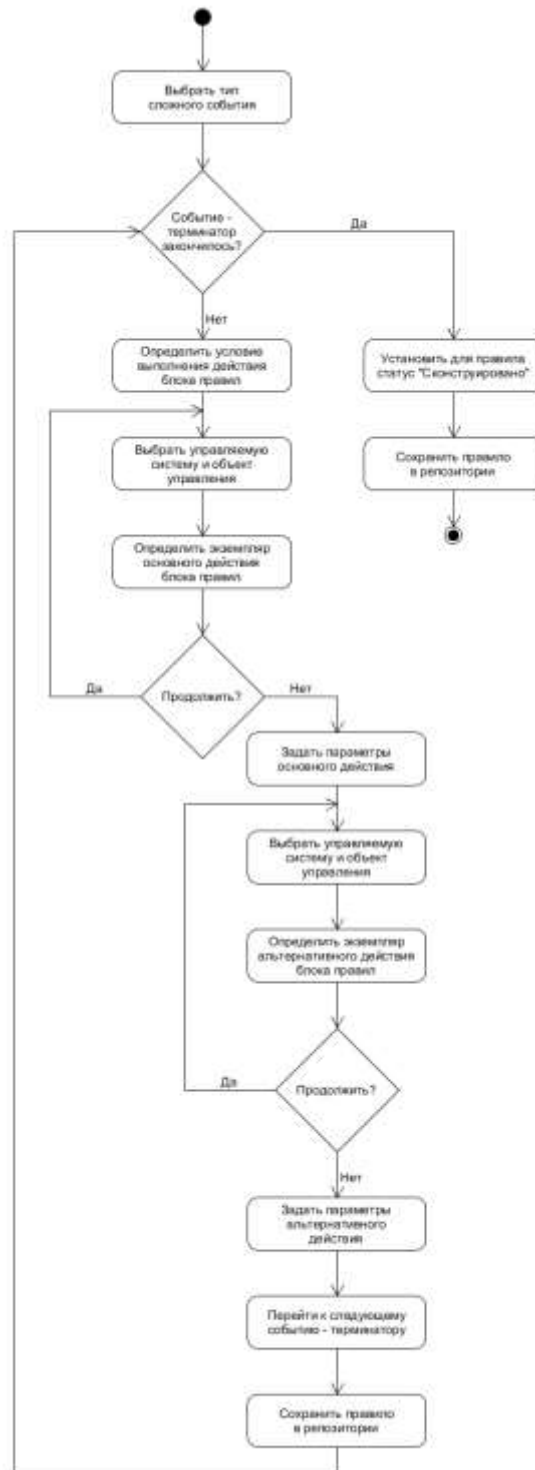


Рисунок 13 – Создание многоблочного правила

Тип правила влияет главным образом на набор атрибутов, доступных пользователю при создании правила.

2.4 Анализ корректности и эффективности активных правил

Во избежание неполадок в работе системы необходимо осуществлять анализ создаваемых правил непосредственно после их конструирования, а также на этапе включения их в работу.

Существует 3 основных вида коллизий взаимодействия активных правил, описанных в [12] Ромеро, Делавалем и Руттенем:

- избыточность, при которой правила выполняют одни и те же действия при одинаковом наборе происходящих событий;
- заикливание, означающее, что исполнение одного правила запускает цикл, который приводит к его исполнению;
- транзитивность, при которой порядок исполнения правил влияет на конечное состояние объекта.

Анализ активных правил в момент верификации называется статической проверкой. Статическая проверка состоит в анализе вариантов взаимодействия правил и выделении среди них потенциальных ошибок, влияющих на стабильность работы системы. Динамическая проверка позволяет выявить некорректные взаимодействия правил во время обработки событий. Статическая проверка может проводиться как непосредственно перед добавлением правил в репозиторий, так и в дополнение к тестированию, тогда как динамическая проверка может проводиться, только когда правило уже добавлено в репозиторий, поэтому статическая проверка является более важным этапом тестирования правил.

Именно на этапе статической проверки выявляется избыточность активных правил. Для этого необходимо сравнить условия и действия для каждого правила. Также статическая проверка помогает выявить заикленности в правилах. Для этого необходимо отследить активацию одних правил другими.

Так как для представления активного правила на этапе конструирования была выбрана нотация раскрашенных сетей Петри (CCPN),

то эта же нотация будет использоваться для анализа правила перед его добавлением в репозиторий.

Динамическая проверка проводится для анализа коллизий, возникающих в процессе работы активных правил. Динамический анализ производится модулем исполнения сервиса потоковой обработки событий и исполнения активных правил при обработке событий непосредственно перед записью результатов работы правил в базу. Общий алгоритм проверки на коллизии представлен на диаграмме деятельности на рисунке 14. Основные виды коллизий перечислены в разделе 1.2.2.

На рисунке 15 представлен алгоритм проверки активных правил на избыточность, то есть на наличие правил, выполняющих одинаковые действия.



Рисунок 14 – Анализ коллизий исполнения активных правил

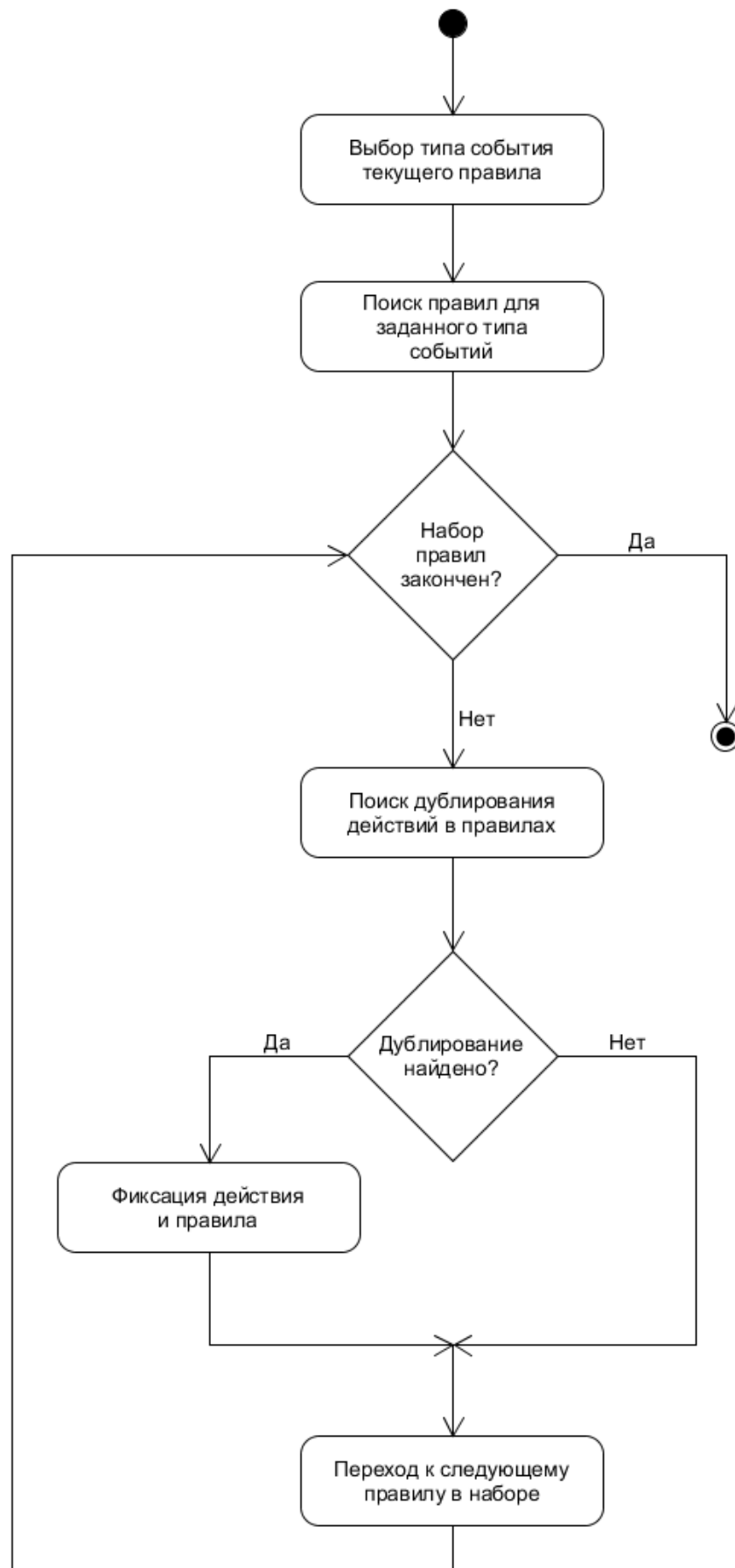


Рисунок 15 – Анализ избыточности активных правил

Объектом проверки являются вызываемые правила и экземпляры событий. Динамический анализ требуется как на стадии запуска правила для проверки выходных параметров, так и на стадии вызова для контроля выполнения активного правила [23].

Выводы

По результатам исследований, представленных в данной главе, можно сделать выводы, перечисленные ниже:

Наряду с классической ЕСА-моделью активных правил, выделяются расширенная ЕСА-модель, модель многоблочных правил, расширенная SECA-модель.

Для описания взаимодействия активных правил выбрана нотация взаимодействующих последовательных процессов (CSP, Communicating Sequential Processes), разработанная Ч. Хоаром и относящаяся к классу алгебр событий.

Представлена методика управления жизненным циклом активных правил, включающая соответствующие процессы для основных этапов жизненного цикла активных правил. В основе разработанной методики лежит богатый набор метаданных, позволяющих автоматизировать соответствующие процессы в рамках инструментальных средств.

Предложены алгоритмы создания различных видов событий и активных правил с учетом многообразия их структуры.

3 Инструментальные средства управления жизненным циклом активных правил в рамках микросервисной архитектуры

В ходе выполнения работы было составлено техническое задание на разработку сервиса потоковой обработки событий и исполнения активных правил, сформулированы функциональные и нефункциональные требования к разрабатываемому программному обеспечению, а также архитектурные требования. Были проанализированы существующие подходы к реализации подсистем исполнения активных правил, а также выбраны технологии и средства для реализации сервиса. В данной главе изучены исследования средств исполнения активных правил и определены основные моменты практической реализации программы. Определены микросервисы, входящие в состав сервиса управления жизненным циклом активных правил.

3.1 Сервис потоковой обработки событий и исполнения активных правил

3.1.1 Назначение и функциональные возможности сервиса потоковой обработки событий и исполнения активных правил

Сервис потоковой обработки событий и исполнения активных правил предназначен для реализации событийной обработки данных в сторонней наблюдаемой системе. Механизм активных правил может использоваться, например, в СУБД, где зачастую возникают ситуации, в которых необходимо реагировать на события добавления записи, когда необходимо пересчитать какие-либо значения, или удаления записи, которая связана внешними ключами с записями в других таблицах. Также данный механизм может использоваться в системе так называемого умного дома, в котором при срабатывании определенных условий сервис может включить или отключить какие-либо приборы или управлять системой безопасности дома. Еще один вариант применения связан с покупкой-продажей акций на биржах, в этом случае при изменении стоимости акций до порогового значения активирует соответствующее правило.

Во всех этих случаях сервис потоковой обработки событий и исполнения активных правил можно описать, как систему, на вход которой поступает событие и его контекст, а на выходе мы получаем определенную реакцию сервиса.

На рисунке 16 представлена диаграмма вариантов использования сервиса потоковой обработки событий и исполнения активных правил.

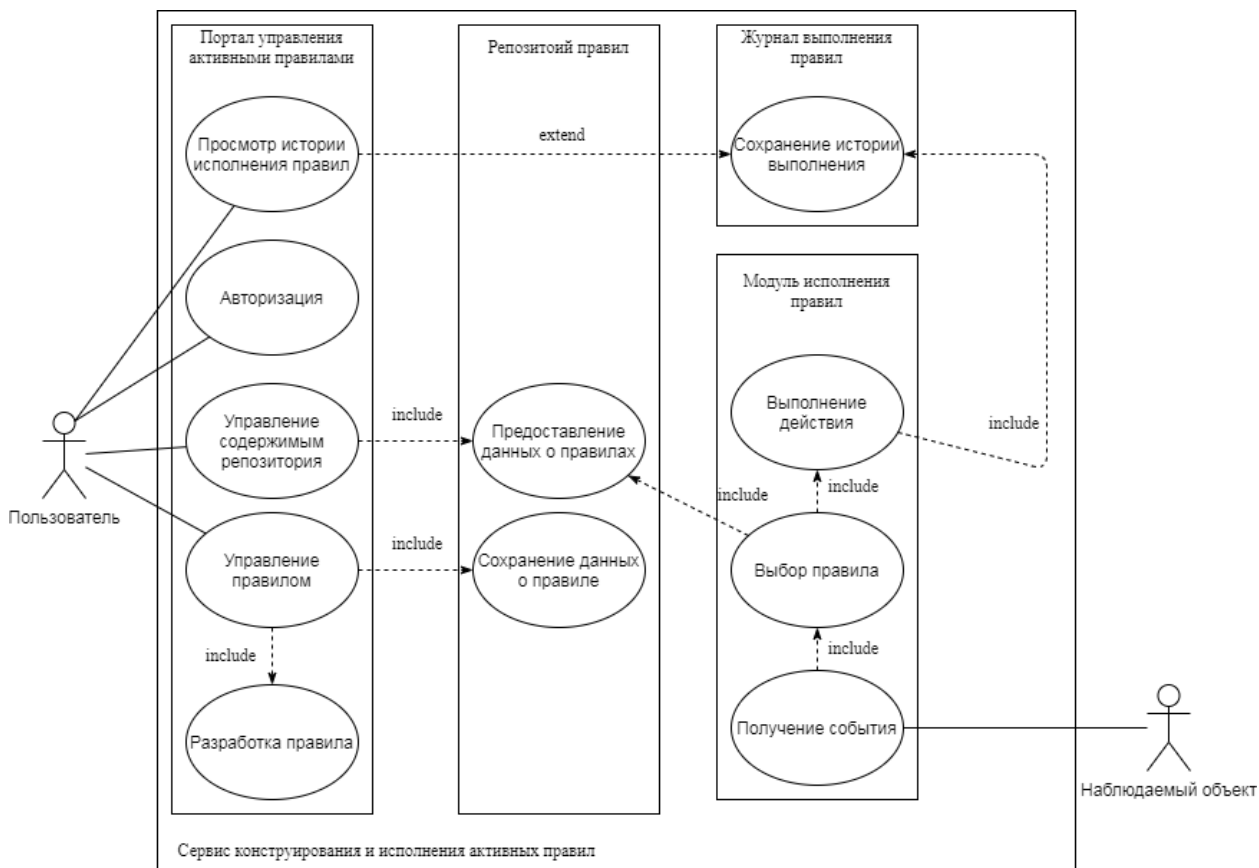


Рисунок 16 – Диаграмма вариантов использования сервиса потоковой обработки событий и исполнения активных правил

Рассмотрим основные функции, которые должен реализовывать сервис потоковой обработки событий и исполнения активных правил:

- разработка активных правил – при помощи клиентского приложения пользователь может сконструировать активное правило: описать входящие в его состав события, условия и действие;
- управление правилами – в клиентском приложении должна быть возможность изменять правило, включать активное правило в обработку, а также отключать его, при необходимости;

- ведение аудита исполнения активных правил – пользователь имеет возможность просматривать историю исполнения активных правил;
- настройка подключения к наблюдаемому объекту – данная функция необходима для настройки подключения программных средств к наблюдаемой системе;
- отслеживание событий в наблюдаемой системе – система отслеживает наступление события и в случае необходимости начинает анализ и исполнение активного правила и запуск действия, соответствующего правилу.

Расширенный перечень функций из раздела 1.4.1 приведен в таблице 9.

Таблица 9 – Функции сервиса

Роль	Группа функций	Функция
Разработчик правил	Конструирование активных правил	Создание, редактирование, проверка, подключение к исполнению / отключение от исполнения типов активных правил
		Создание, редактирование, проверка, подключение к наблюдению / отключение от наблюдения типов входных элементарных (простых) событий от объектов наблюдения, в том числе контекста (набора параметров) события
		Создание, редактирование, проверка, подключение к отслеживанию / отключение от отслеживания типов сложных (составных) событий
		Создание, редактирование, проверка, подключение доступности / отключение доступности условий исполнения действий с учетом контекста (набора параметров) соответствующего типа события и данных, выбираемых из оперативного хранилища
		Создание, редактирование, проверка, подключение к использованию / отключение от использования типов действий с объектами управления и данных
Администратор сервиса	Настройка и масштабирование сервиса	Настройка параметров функционирования отдельных компонент с учетом результатов аудита
		Масштабирование отдельных компонент (изменение параметров и распределение очередей и потоков обработки) и сервиса в целом (подключение/отключение компонентов)
		Синхронизация глобальных и локальных метаданных, описывающих конфигурацию системы и отдельных компонент

Продолжение таблицы 9

Роль	Группа функций	Функция
Аналитик	Аудит	Аудит обрабатываемых событий в режимах онлайн (на основе актуальной информации) и офлайн (на основе данных журналов), выявление «проблемных» событий
		Аудит исполнения активных правил в режимах онлайн (на основе актуальной информации) и офлайн (на основе данных журналов), выявление «тяжело исполняемых» активных правил
Система	Анализ коллизий взаимодействия активных правил	Статический анализ коллизий взаимодействия активных правил в процессе конструирования
		Динамический анализ коллизий взаимодействия активных правил в процессе исполнения
	Обработка событий	Первичная обработка экземпляров элементарных (простых) входных событий от объектов наблюдения: идентификация события, принятие решение по событию в зависимости от профиля события (зарегистрированное/незарегистрированное, информационное/управляющее, элемент сложного события и т.д.)
		Передача экземпляров событий для дальнейшей обработки компонентам сервиса в соответствии со сценарием обработки категорий типов событий, заданных в профиле события;
		Отслеживание появления экземпляров сложных (составных) событий в соответствии с заданными шаблонами
	Исполнение активных правил	Выбор экземпляра активного правила (группы экземпляров активных правил) для исполнения
		Вычисление условия экземпляра активного правила, принятие решение о выполнении действия экземпляра активного правила
		Динамический анализ коллизий взаимодействия экземпляров активных правил и оперативное реагирование на возникновение коллизий
	Журнализация и анализ данных журналов событий	Журнализация обрабатываемых событий, исполнения активных правил, обработки компонент активных правил
		Выявление закономерностей в журналах событий
		Моделирование новых событий и активных правил на основе выявленных закономерностей

К списку пользователей был добавлен аналитик, проводящий аудит исполняемых правил для выявления возможных проблем. Также функции остальных пользователей получили подробное описание.

3.1.2 Архитектура сервиса потоковой обработки событий и исполнения активных правил

Для разработки сервиса управления жизненным циклом активных правил была выбрана микросервисная архитектура. Это принципиальная организация распределенной системы на основе микросервисов и их взаимодействия друг с другом и со средой по сети, а также принципов, направляющих проектирование архитектуры, её создание и эволюцию [21]. В приложении выделяется несколько крупных задач и для каждой из них выделяется микросервис, который будет за нее отвечать.

Микросервисная архитектура обладает рядом свойств, необходимых при создании многомодульных систем. Модули можно легко заменить в любое время, акцент делается на простоту, независимость развёртывания и обновления каждого из микросервисов. Каждый модуль организован вокруг функции – микросервис по возможности выполняет только одну достаточно элементарную функцию. Модули могут быть реализованы с использованием различных языков программирования, фреймворков, связующего программного обеспечения, выполняться в различных средах контейнеризации, виртуализации, под управлением различных операционных систем на различных аппаратных платформах: приоритет отдаётся в пользу наибольшей эффективности для каждой конкретной функции, нежели стандартизации средств разработки и исполнения. Архитектура является симметричной, а не иерархической – зависимости между микросервисами одноранговые [21].

Такая архитектура позволяет разработчикам системы действовать независимо друг от друга и переиспользовать существующие модули программы в других системах. Также она позволяет безболезненно для системы вносить изменения любой сложности в микросервис, не затрагивая при этом остальные.

В системе управления жизненным циклом активных правил можно выделить следующие модули:

- сервис запросов на клиентской части системы;
- сервис, обрабатывающий события;
- сервис конструирования активных правил;
- сервис статического анализа правил;
- сервис исполнения активных правил;
- сервис динамического анализа правил;
- база данных;
- сервис журналирования.

Связь между компонентами системы представлена на рисунке 17.

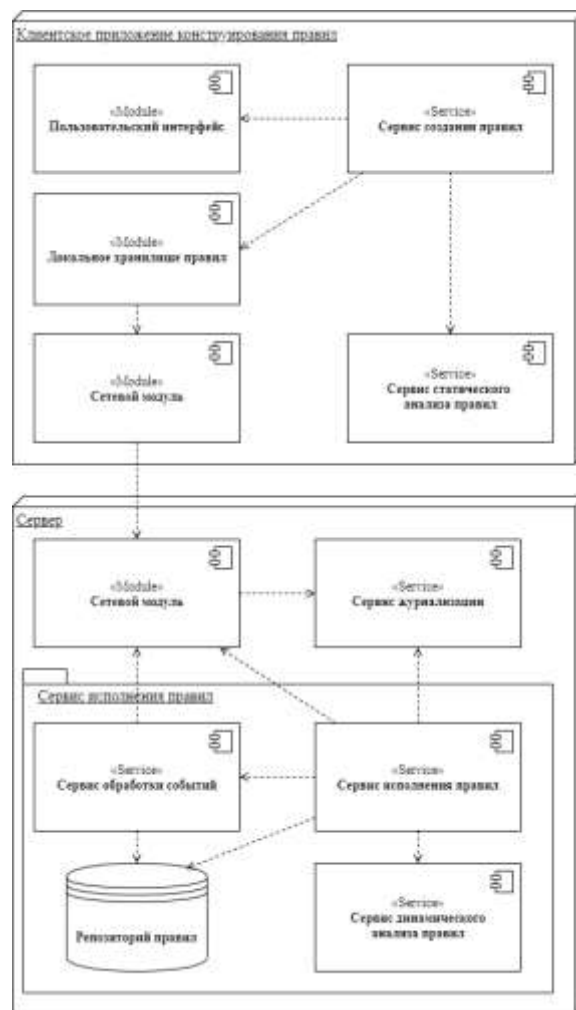


Рисунок 17 – Диаграмма компонентов сервиса потоковой обработки событий и исполнения активных правил

Из диаграммы видно, что главным модулем в клиентской части системы является интерфейс, который также выступает точкой входа в приложение. События, приходящие из внешней среды, вызывают

срабатывания методов в сервисе, ответственном за их обработку. Эти методы, в свою очередь, вызывают методы сервиса запросов и локального хранилища. На сервере через специальный сервис происходит обращение к базе данных.

3.1.3 Обоснование выбора технологий и средств разработки

В качестве технологий разработки клиентской части сервиса были выбраны язык программирования C# и библиотека Qt. Разработка велась в среде Visual Studio. C# является высокоуровневым объектно-ориентированным языком программирования, стандартные средства которого позволяют создавать WPF-приложения (Windows Presentation Foundation), которые имеют легкий для создания и использования графический интерфейс. Библиотека Qt позволяет создавать элементы графического интерфейса любой сложности, а также прописывать взаимодействие элементов: перетаскивание, добавление, удаление и т.д., как описано в [24].

Для создания серверной части использовался язык программирования Python и среда разработки PyCharm. Python представляет из себя скриптовый язык программирования, который часто используется для написания серверной части системы. Оно обладает большим количеством различных библиотек, рассчитанных на написания сервера. В сервисе потоковой обработки событий и исполнения активных правил были использованы библиотеки Django и TastyPie, позволяющие легко реализовать все функции REST-сервера [25].

Для обработки потоков была выбрана библиотека Kafka Streams [22]. Данная библиотека позволяет использовать хранилища состояния для поиска по справочнику и запоминания уже просмотренных данных, □ соединять потоки данных для получения дополнительной информации, а также ставить временные метки для журналирования событий.

3.2 Инструментальные средства управления жизненным циклом активных правил в составе сервиса

Инструментальные средства сервиса управления жизненным циклом активных правил включают в себя средства работы с наблюдаемой и управляемой системами, а также подсистему конструирования и проверки активных правил, функции которой представлена на рисунке 18.

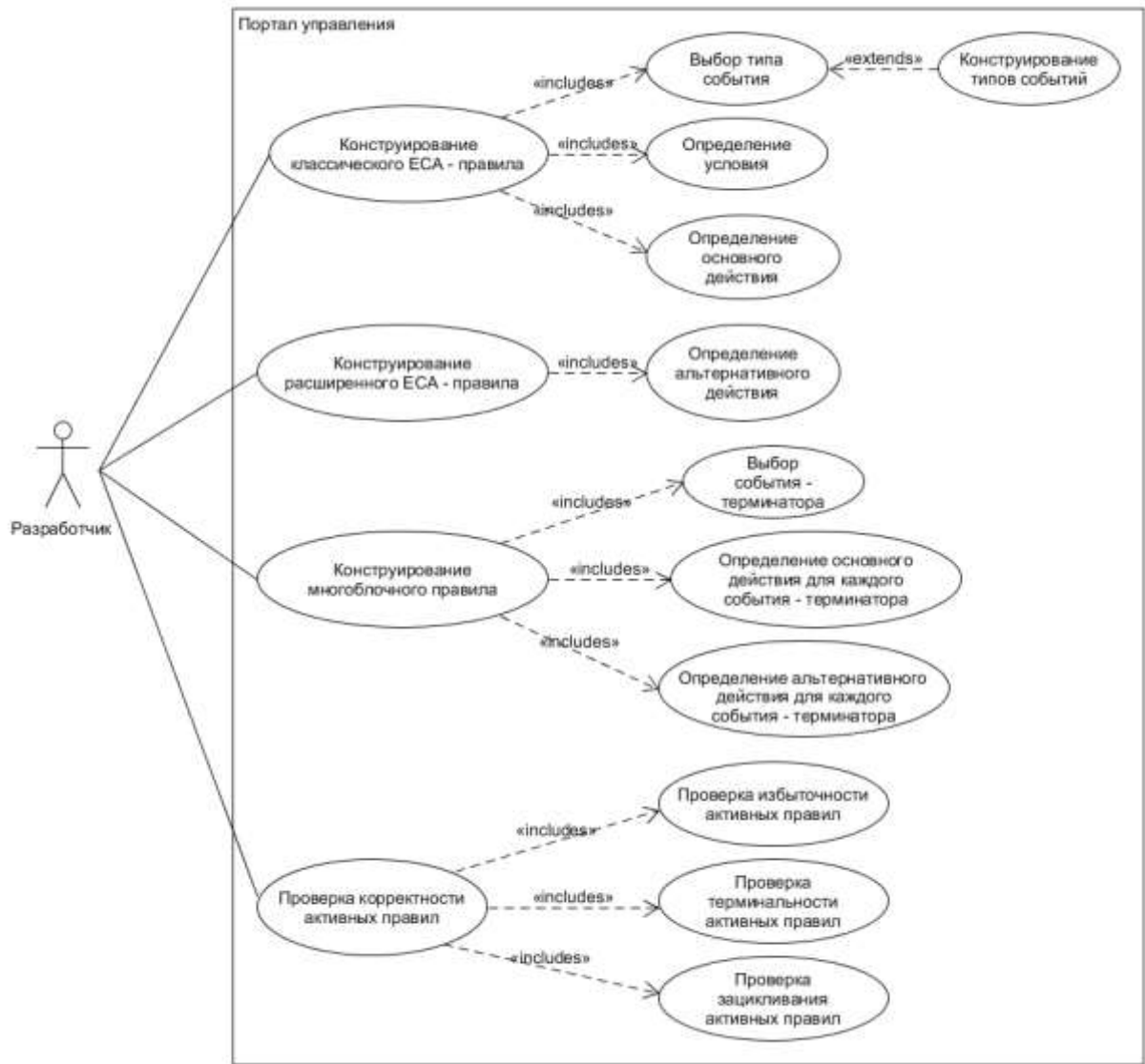


Рисунок 18 – Диаграмма вариантов использования подсистемы конструирования и проверки активных правил

Как видно из диаграммы, разработчик может выбрать тип создаваемого правила. Выбор повлияет на доступные опции, для некоторых видов правил

доступны дополнительные действия и события. Также разработчик может произвести проверку правил на возможные коллизии.

Для работы с наблюдаемой и управляемой системами необходимо хранить метаданные о них. На рисунках 19 и 20 представлены процессы подготовки метаданных об этих системах.

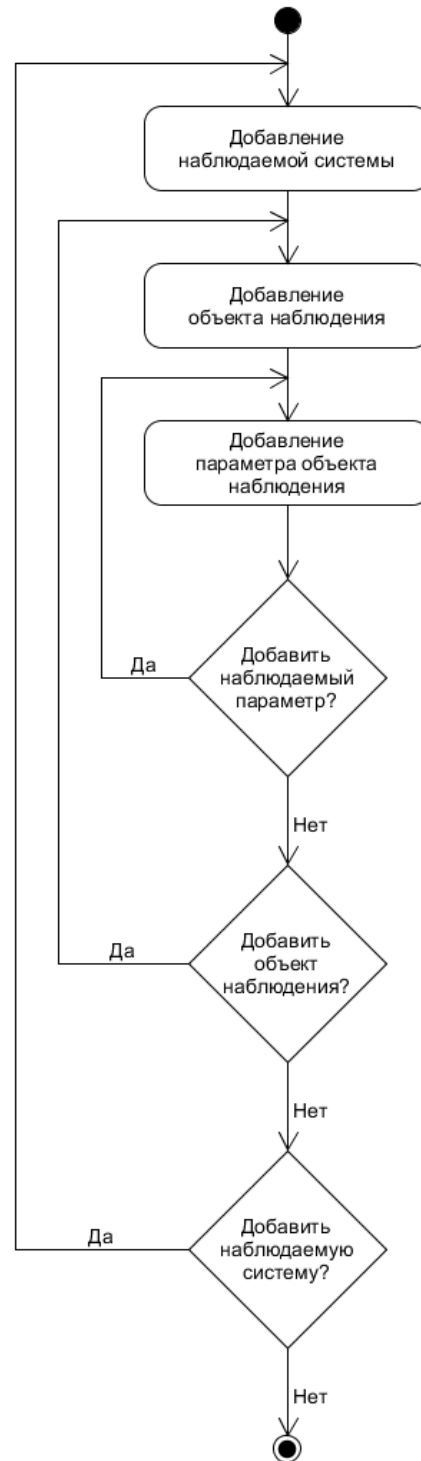


Рисунок 19 – Подготовка метаданных о наблюдаемых системах

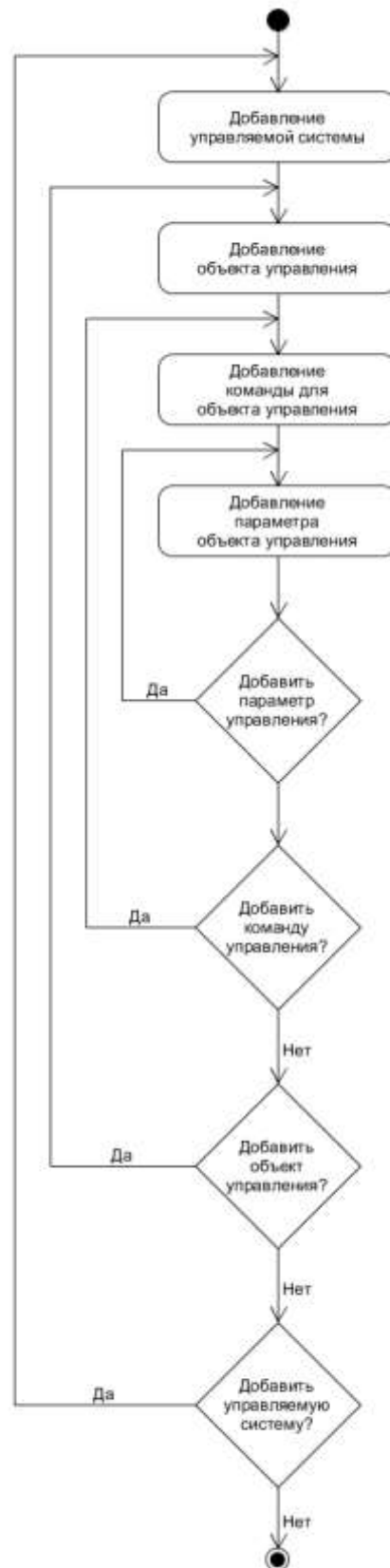


Рисунок 20 – Подготовка метаданных об управляемых системах

В обоих типах систем пользователь выбирает объект для управления или наблюдения, добавляет для них необходимые команды и параметры.

3.3 Информационное и программное обеспечение инструментальных средств

3.3.1 Репозиторий метаданных для СУБД PostgreSQL

Для обеспечения процессов конструирования и анализа активных правил требуется создать репозиторий, который позволит хранить все виды правил, а также всю информацию, необходимую для взаимодействия с ними. Также для разных пользователей необходимо разграничить доступ к базе данных и предоставить возможность выбрать при регистрации роль, от которой будет зависеть возможность взаимодействия с правилами.

На рисунках 21 и 22 представлены структуры фрагментов репозитория, в которых хранятся данные пользователей и активные правила.

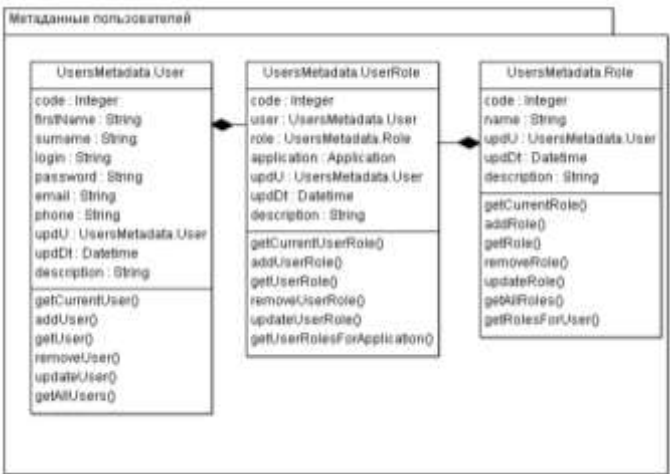


Рисунок 21 – Логическая модель репозитория пользовательских данных

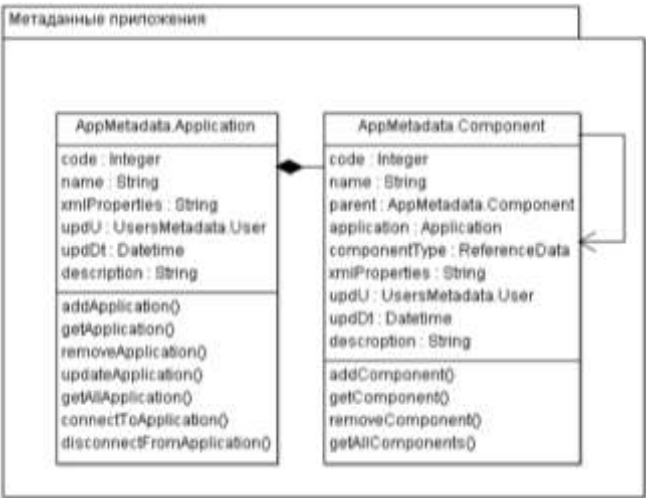


Рисунок 22 – Логическая модель репозитория активных правил

В репозитории активных правил содержатся данные о правилах и составляющих их событиях, условиях и действиях. Условия сложного типа содержат в себе ссылки на составные условия. Также условие может отсутствовать.

3.3.2 Программное обеспечение REST-сервера

Архитектура проекта основана на использовании Rest-API и включает в себя две традиционные части: клиентскую и серверную. REST расшифровывается как Representational State Transfer. Данная архитектура лучше всего подходит для многопользовательской системы, в которой имеется единая база данных и возможность работы с ней для нескольких удалённых пользователей.

Передача данных в Rest-API осуществляется по протоколу HTTP (Hyper Text Transfer Protocol). На клиентской части системы формируется запрос, включающий в себя метод, заголовки и тело запроса. Метод представляет собой последовательность из любых символов, кроме управляющих и разделителей, и определяет операцию, которую нужно осуществить с указанным ресурсом. Спецификация HTTP не ограничивает количество разных методов, которые могут быть использованы, однако в целях соответствия общим стандартам и сохранения совместимости с максимально широким спектром программного обеспечения, как правило, используются лишь некоторые, наиболее стандартные методы, смысл которых однозначно раскрыт в спецификации протокола. Стандартными методами являются следующие:

- GET – запрашивает данные с сервера;
- POST – добавляет на сервер новые данные;
- PUT – изменяет существующие данные;
- DELETE – удаляет данные с сервера.

В заголовках запроса представлена различная дополнительная информация для сервера. Они не являются обязательной частью запроса. В теле запроса содержится его основная информация.

База данных активных правил размещается на сервере. Также на сервере размещается интерфейс для работы с базой данных, включающий в себя функции создания новой БД, добавления данных в существующую или их изменение. Структура базы данных представлена на рисунке 23.

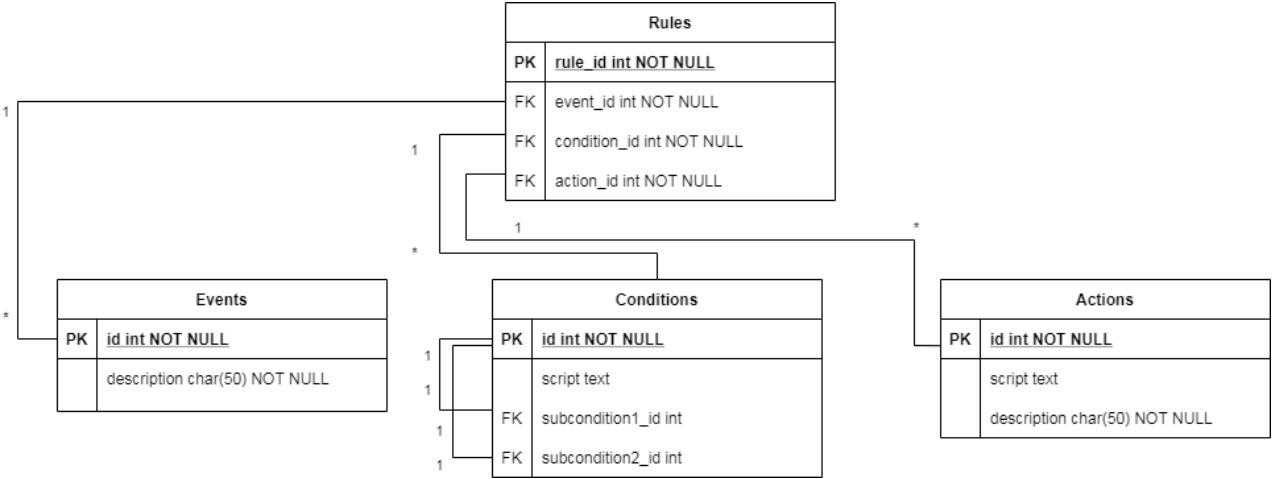


Рисунок 23 – Обобщенная структура репозитория правил

Репозиторий правил включает в себя таблицы правила, события, условия и действия. Правило имеет ссылки на свои компоненты. Условие и действие включают поле скрипта, хранящее их запись в нотации BPMN. Условие также может включать ссылки на составные части.

Связь между клиентской и серверной частью системы представлена на рисунке 24.

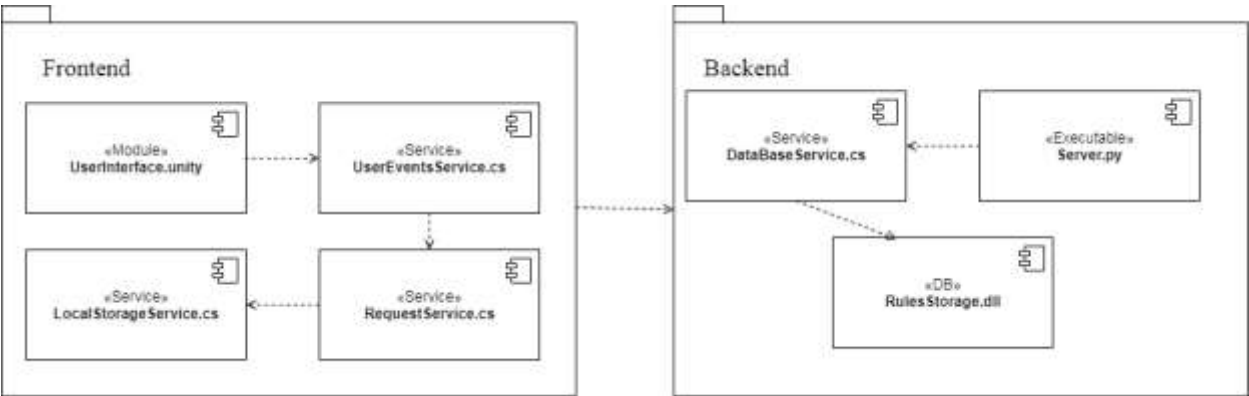


Рисунок 24 – Структура сервиса потоковой обработки событий и исполнения активных правил

На серверной части системы формируется ответ, который включает в себя код, по которому можно определить, что запрос обработан удачно, или,

в случае неудачи, понять, какая произошла ошибка. Ответ также содержит тело и заголовки.

3.3.3 Программное обеспечение клиентского приложения

Клиентское приложение представляет собой WPF-приложение, включающее 2 основных модуля – интерфейс пользователя и модуль взаимодействия с API, в котором выполняются запросы на получение, отправку и изменение данных в таблицах и обрабатываются ответы сервера.

На рисунке 25 представлена частичная диаграмма классов клиентского приложения.

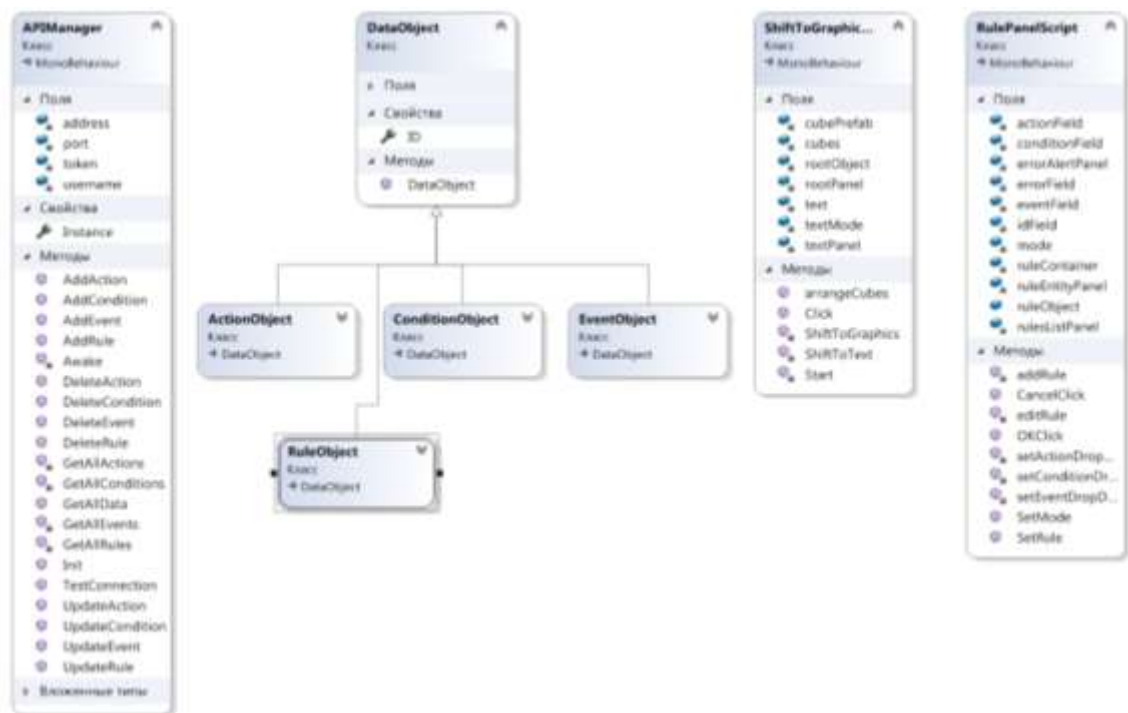


Рисунок 25 – Диаграмма классов клиентского приложения

Класс APIManager отвечает за отправку запросов на сервер и обработку ответов. DataObject и его наследники отвечают за создание активного правила и его компонентов. В ShiftToGraphicsMode обрабатывается переключение интерфейса между табличным и графическим представлением, а RulePanelScript отвечает за создание и редактирование активных правил в графическом режиме.

Выводы

По результатам исследований, выполненных в настоящей главе, можно сделать выводы, перечисленные ниже.

Сервис потоковой обработки событий и исполнения активных правил предназначен для событийной обработки данных, полученных от наблюдаемых систем. Реакцией на возникновение событий является исполнение активных правил по отношению к управляемым системам.

Основным назначением портал управления является управление активными правилами на всех этапах жизненного цикла.

Для эффективной работы различных компонент сервиса создан централизованный репозиторий метаданных, в котором хранится вся информация о правилах, включая информацию о событиях, условиях и действиях.

Портал управления активными правилами реализован в клиент-серверной архитектуре на основе REST API.

Заключение

В ходе исследования было установлено существующие решения проблемы обработки событий в информационных системах имеют ряд ограничений. В данной работе предлагается использование единого модельного представления активных правил на основе нотации BPMN и условных раскрашенных сетей Петри. В ходе работы были решены следующие задачи:

- исследованы и систематизированы процессы жизненного цикла активных правил в событийно-ориентированных системах;
- проведен анализ современных подходов к моделированию бизнес-логики информационно-управляющих систем и выработаны требования к нотации представления активных правил;
- исследован и обоснован выбор модельного представления активных правил для инструментальных средств управления жизненным циклом активных правил;
- разработана методика управления жизненным циклом активных правил, включая алгоритмы конструирования, анализа корректности и аудита активных правил;
- разработан программный прототип инструментальных средств, поддерживающих методику управления жизненным циклом активных правил, обладающих развитым пользовательским интерфейсом и обеспечивающих конструирование, анализ и аудит активных правил для сервиса исполнения активных правил.

В ходе работы были исследованы и проанализированы коллизии взаимодействия в процессе исполнения взаимодействующих правил в системах обработки событий. В ходе работы была проведена формализация взаимодействия активных правил на основе алгебры событий Хоара, позволяющая строго определить каждый из видов конфликтов, рассмотрены модели, применяющиеся для статического и динамического анализа с целью выявления и устранения конфликтов и отмечена необходимость разработки

единой модели взаимодействия правил, которая может применяться на стадии динамического анализа. В качестве такой модели предлагается использовать нотацию BPMN и условные раскрашенные сети Петри. Сети Петри на основе единого формализма позволяют анализировать правила на этапе исполнения. В данной диссертационной работе приведены результаты анализа существующих подходов к обработке событий и обосновано применение концепции технологии исполнения активных правил как более универсального решения, охватывающего ряд важных предметных областей. Результаты исследования позволяют строить системы хранения и исполнения активных правил, обеспечивающие хранение правил не зависимо от их типа или структурных особенностей, исполнение правил с высокой скоростью, путем применения механизмов динамического анализа, обработку сложных событий, т.е. цепочки связанных между собой простых событий, журнализацию исполнения правил. Так же результатами являются: модельное представление репозитория активных правил, алгоритм параллельного исполнения активных правил, модельное представление и алгоритм динамического анализа исполнения активных правил.

Список использованных источников

1. Dayarathna M., Perera S. / Recent Advancements in Event Processing // ACM Computing Surveys – 2018– С. 8–9
2. Stonebraker M., Getintemel U., Zdonic S / The 8 Requirements of Real-Time Stream Processing // SIGMOD Record, Vol. 34, No. 4 – 2005 – С. 1–2
3. Adaikkalavan R. Event Operators: Formalization, Algorithms, and Implementation Using Interval-Based Semantics. / R.Adaikkalavan. – Arlington. : MS Thesis, 2002. – p.4 – 5
4. Chandy, K. M. Event Processing: Designing IT Systems for Agile Companies / K. M. Chandy, W. R. Schulte. – McGraw-Hill, 2009. – С. 251
5. Atzmueller, M. Enterprise Big Data Engineering, Analytics and Management /M. Atzmueller, S. Oussena, T. Roth-Berghofer. – IGI Global, 2016. – С. 272
6. Джонс, М. Обработка больших данных реального времени с помощью Twitter Storm // Электронные текстовые данные [Электронный ресурс]. URL: <https://www.ibm.com/develop-perworks/ru/library/os-twitterstorm/> (дата обращения 12.02.2021)
7. Di Wang, Elke A. Rundensteiner, Richard T. Ellison / Active Complex Event Processing over Event Streams // Proceedings of the VLDB Endowment – 2011 – С. 2–5
8. Paton, N.W. Active database systems / N.W.Paton, O.Diaz // ACM Computing Surveys (CSUR). – 1999. – №31.1. – С. 63-103
9. Перри Л. / Архитектура интернета вещей // пер. с англ. М. А. Райтмана. – М.: ДМК Пресс, 2019– С. 7
10. Chakravarthy, S. Early Active Database Efforts: A Capsule Summary / Chakravarthy S. // IEEE Transactions on Knowledge and Data Engineering. – 1995. – Vol. 7. – №6. – С. 1008- 1010
11. Paton, N.W. Active Database Systems / Paton N.W., Diaz O. // ACM Computing Surveys. – 1999. – Vol. 31. – №1. – С. 64-103

12. Romero J.A.C., Delaval G., Rutten E. // Coordination of ECA Rules by Verification and Control – C. 2
13. Di Wang, Elke A. Rundensteiner / Active Complex Event Processing: Applications in Real-Time Health Care // Univ. of Massachusetts Medical School – 2011 . – C. 4-8
14. Schaff, M. «An Active DBMS Style Activity Service for Cloud Environments», / Schaff M., Koschel A., Grivas S.G., Astrova I. // The International Conference on Cloud Computing, GRIDs, and Virtualization, pp.80-85, IARIA, ISBN: 978-1-61208-106-9, 2010
15. Требования к дизайну пользовательского интерфейса // Основы композиции и дизайна [Электронный ресурс]. URL: http://www.okd.mdk.ksue.edu.ua/index.php?option=com_content&view=ar%20article&id=174&Itemid=90 (дата обращения 18.10.2020)
16. Fors T., Visualization of rule behaviour in active databases / Department of Computer Science // University of Skovde, Skovde, Sweden – 1995 – C. 12 – 13
17. Neil Macehiter, Neil Ward-Dutton / The business value of event processing: preparing the path – 2009 – C. 7 – 10
18. S. Suhothayan, I. L. Narangoda, S. Chaturanga / Siddhi: A Second Look at Complex Event Processing Architectures – 2011 – C. 5 – 9
19. J.Medina-Marin, G.Perez-Lechuga, J.C.Seck-Tuoh-Mora. ECA rule analysis in a Distributed Active Database // International Conference on Computer Technology and Development, 2009 – C.113-120
20. Schaff, M. An Active DBMS Style Activity Service for Cloud Environments / Schaff M., Koschel A., Grivas S.G., Astrova I. // The International Conference on Cloud Computing, GRIDs, and Virtualization. – 2010. – C. 80-85
21. Ричардсон К. Микросервисы. // Паттерны разработки и рефакторинга. – СПб.: Питер, 2019, с. 59
22. Беджек Б. Kafka Streams в действии. // Приложения и микросервисы для работы в реальном времени. – СПб.: Питер, 2019, с. 117

23. Naqvi, W. Rule and knowledge management in an active database system // Proceedings of the First international workshop on rules in database systems / W. Naqvi M. Ibrahim // Springer-Verlag, 1994. P. 58-69

24. Qt vs .Net // Qt Documentation [Электронный ресурс]. URL: <https://doc.qt.io/qt-5/activeqt-dotnet.html> (дата обращения 28.01.21)

25. Getting Started with Tastypie // TastyPie Documentation [Электронный ресурс]. URL: <https://django-tastypie.readthedocs.io/en/latest/tutorial.html> (дата обращения 16.10.20)

Приложение А
ГЛОССАРИЙ
(справочное)

ИС – информационная система.

АП – активное правило.

БД – база данных.

ЖЦ – жизненный цикл.

ПО – программное обеспечение.

СУБД – система управления базами данных.

АСУБД – система управления активными базами данных.

CEP (Complex Event Processing) – обработка сложных событий.

ЕСА (Event–Condition–Action)-правило – правило, состоящее из события, условия, действия.

СА (Condition-Action)-правило – правило, состоящее из условия и действия.

ЕА (Event-Action)-правило – правило, состоящее из события и действия.

URL (Uniform Resource Locator) – единый указатель ресурса.

API (Application Programming Interface) – интерфейс прикладного программирования.

Приложение Б
КОД ПРОГРАММЫ
(рекомендуемое)

Листинг Б.1 – Код участка главного модуля REST-сервера

```

from tastypie.resources import ModelResource
from .models import Event, Condition, Action, Rule
from django.contrib.auth.models import User
from django.db.models import signals
from tastypie.models import create_api_key
from tastypie import fields
from tastypie.authentication import ApiKeyAuthentication
from tastypie.authorization import DjangoAuthorization
from django.db import models

class EventResource(ModelResource):
    class Meta:
        queryset = Event.objects.all()
        resource_name = 'event'
        authentication = ApiKeyAuthentication()
        authorization = DjangoAuthorization()

    # def post(self, request):
    #     print(request)
    #     return super().post(request)

class ConditionResource(ModelResource):
    class Meta:
        queryset = Condition.objects.all()
        resource_name = 'condition'
        authentication = ApiKeyAuthentication()
        authorization = DjangoAuthorization()

    left_argument = fields.ToOneField('self', 'argument1',
null=True)
    right_argument = fields.ToOneField('self', 'argument2',
null=True)

    #left_argument_id =
fields.IntegerField(attribute='left_argument_id', null=True)
    #right_argument_id =
fields.IntegerField(attribute='right_argument_id', null=True)

class ActionResource(ModelResource):
    class Meta:
        queryset = Action.objects.all()
        resource_name = 'action'
        authentication = ApiKeyAuthentication()
        authorization = DjangoAuthorization()

class RuleResource(ModelResource):
    class Meta:
        queryset = Rule.objects.all()
        resource_name = 'rule'

```

```

        authentication = ApiKeyAuthentication()
        authorization = DjangoAuthorization()

        event_id = fields.ToOneField(EventResource, 'event')
        condition_id = fields.ToOneField(ConditionResource,
'condition')
        action_id = fields.ToOneField(ActionResource, 'action')

        #event_id = fields.IntegerField(attribute='event_id')
        #condition_id = fields.IntegerField(attribute='condition_id')
        #action_id = fields.IntegerField(attribute='action_id')

signals.post_save.connect(create_api_key, sender=User)

```

Листинг Б.2 – Код участка настроек REST-сервера

```

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tastypie',
    'serverapp.apps.ServerappConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Server.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',

```

```

'django.contrib.messages.context_processors.messages',
    ],
},
],

WSGI_APPLICATION = 'Server.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'Server',
        'USER': 'DjangoUser',
        'PASSWORD': '1234',
        'HOST': 'localhost',
        'PORT': '5432',
        # 'OPTIONS': {
        #     'init_command': 'ALTER DATABASE server CHARACTER
SET utf8mb4 COLLATE utf8mb4_general_ci',
        # }
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-
password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]

```

```
# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
```

Листинг Б.3 – Участок класса ShiftToGraphicMode.cs

```
public class ShiftToGraphicMode : MonoBehaviour
{
    [SerializeField]
    GameObject textPanel;
    [SerializeField]
    GameObject rootPanel;
    [SerializeField]
    GameObject rootObject;
    [SerializeField]
    GameObject cubePrefab;
    [SerializeField]
    GameObject text;

    bool textMode = true;
    List<GameObject> cubes;
    private void Start()
    {
        cubes = new List<GameObject>();
    }
    private void ShiftToGraphics()
    {
        textPanel.SetActive(false);

        foreach(var r in RuleStorage.Instance)
        {
            GameObject cubeInstantiate = Instantiate(cubePrefab,
rootPanel.transform);
            GraphicsLogic curGL =
cubeInstantiate.GetComponent<GraphicsLogic>();
            cubes.Add(cubeInstantiate);
            curGL.TextViewPanel = text;
            curGL.Rule = r;
        }
    }
}
```

```

    }
    rootObject.SetActive(true);
    rootPanel.SetActive(true);
    arrangeCubes();
}
private void ShiftToText()
{
    rootPanel.SetActive(false);
    rootObject.SetActive(false);

    var gls =
rootPanel.GetComponentsInChildren<GraphicsLogic>();
    foreach (var cube in gls)
    {
        Destroy(cube.gameObject);
    }
    cubes.Clear();

    textPanel.SetActive(true);
}

public void Click()
{
    if(textMode)
    {
        ShiftToGraphics();
    } else
    {
        ShiftToText();
    }
    textMode = !textMode;
}
public void arrangeCubes()
{
    float inset = 0;
    foreach(var cube in cubes)
    {
        cube.transform.position = new Vector3(inset,
cube.transform.position.y, cube.transform.position.z);
        inset += 1.5f;
    }
}
}

```