Оглавление

[Введение 2](#_Toc195562723)

[Цель и задачи исследования 4](#_Toc195562724)

[Научная новизна 6](#_Toc195562725)

[Практическая значимость 6](#_Toc195562726)

[Объект и предмет исследования 7](#_Toc195562727)

[Методы исследования 7](#_Toc195562728)

[Структура работы 8](#_Toc195562729)

[Глава 1. Анализ предметной области и существующих подходов 9](#_Toc195562730)

[1.1. Событийно-непрерывные модели: определение и область применения 9](#_Toc195562731)

[1.2. Обзор существующих методик представления и обработки 10](#_Toc195562732)

[1.3. Анализ ограничений традиционных подходов 10](#_Toc195562733)

[1.4. Сравнительная характеристика известных программных решений 10](#_Toc195562734)

[1.5. gRPC как инструмент взаимодействия в распределённой системе 10](#_Toc195562735)

[1.6. Постановка задачи разработки унифицированного представления 11](#_Toc195562736)

[Глава 2. Проектирование унифицированного представления событийно-непрерывных моделей 11](#_Toc195562737)

[2.1. Общая схема архитектуры системы 11](#_Toc195562738)

[2.2. Основные сущности системы 12](#_Toc195562739)

[2.3. Разработка структуры хранения и связей 13](#_Toc195562740)

[2.4. Формирование требований к унифицированному представлению 14](#_Toc195562741)

[2.5. Логическая схема данных и описание типов данных 16](#_Toc195562742)

[2.6. Проектирование интерфейсов взаимодействия и обработки данных 17](#_Toc195562743)

[2.7. Разработка протоколов взаимодействия на базе gRPC 18](#_Toc195562744)

[Глава 3. Реализация программного комплекса 19](#_Toc195562745)

[3.1. Средства разработки и используемые технологии 19](#_Toc195562746)

[3.2. Реализация клиентского интерфейса 19](#_Toc195562747)

[3.3. Реализация сервера-слушателя 20](#_Toc195562748)

[3.4. Реализация сервера-вычислителя 21](#_Toc195562749)

[3.5. Логика обмена сообщениями между компонентами через gRPC 22](#_Toc195562750)

[Глава 4. Исследование работы и тестирование 23](#_Toc195562751)

[4.1. Подходы к тестированию распределённых вычислительных систем 23](#_Toc195562752)

[4.2. Разработка и проведение тестов клиент-серверного взаимодействия 24](#_Toc195562753)

[4.3. Проверка корректности работы вычислителя на тестовых данных 24](#_Toc195562754)

[4.4. Оценка производительности и масштабируемости системы 24](#_Toc195562755)

[4.5. Сравнительный анализ с монолитными и другими распределёнными архитектурами. 24](#_Toc195562756)

[4.6. Анализ устойчивости к сбоям и нагрузочным ситуациям 25](#_Toc195562757)

[Заключение 25](#_Toc195562758)

[Список литературы 25](#_Toc195562759)

# Введение

Актуальность темы

В современных условиях развития программных и вычислительных технологий наблюдается устойчивый рост интереса к построению гибких и масштабируемых вычислительных систем. Одним из перспективных направлений является моделирование событийно-непрерывных процессов, находящих применение в различных областях: от промышленного производства и телекоммуникаций до научных исследований. Эти модели позволяют учитывать как дискретные, так и непрерывные изменения, что особенно важно при решении сложных задач оптимизации и анализа динамических систем.

Современные вычислительные системы требуют перехода от монолитных архитектур к распределённым решениям, способным эффективно масштабировать вычислительные ресурсы и оптимизировать обработку больших объёмов данных. Разработка удобного, надёжного и унифицированного представления событийно-непрерывных моделей становится ключевым элементом в создании таких систем, поскольку позволяет обеспечить совместимость между различными вычислительными платформами и ускорить процесс интеграции новых компонентов.

Использование протоколов удалённого вызова процедур, таких как gRPC, открывает широкие возможности для построения распределённых систем. gRPC обеспечивает стандартизированный и эффективный обмен данными между компонентами системы, снижая сетевые задержки и повышая отказоустойчивость. Такие возможности особенно актуальны в условиях роста требований к производительности и надежности систем, а также при необходимости оперативного масштабирования вычислительных мощностей.

Современные тенденции в области контейнеризации, например с использованием Docker, позволяют создавать однородные и управляемые среды выполнения, что значительно упрощает развертывание и эксплуатацию вычислительных комплексов. Это, в свою очередь, способствует более быстрому внедрению инновационных решений и адаптации системы к изменяющимся условиям эксплуатации.

Таким образом, актуальность темы обусловлена необходимостью разработки унифицированного представления событийно-непрерывных моделей для распределённых вычислительных систем, способных удовлетворить потребности современной экономики и науки. Создание такого представления является стратегически важным шагом в направлении повышения эффективности моделирования сложных систем, улучшения их масштабируемости и обеспечения высокой производительности вычислений в условиях постоянно растущей нагрузки и объёмов обрабатываемых данных.

## Цель и задачи исследования

* Целью данной работы является разработка унифицированного представления событийно-непрерывных моделей и исследование особенностей его реализации в распределённой системе с использованием gRPC. Эта цель обусловлена необходимостью создания эффективного инструмента, который позволит объединить дискретные и непрерывные процессы в единой схеме, что особенно актуально для моделирования сложных систем, применяемых в промышленности, телекоммуникациях и научных исследованиях. Разработка такого представления позволит снизить издержки на разработку, оптимизировать использование вычислительных ресурсов и обеспечить возможность масштабирования вычислительных систем, что становится критически важным в условиях растущих объёмов данных и увеличения требований к производительности.
* Для достижения поставленной цели необходимо решить следующие задачи:
* **Провести анализ существующих подходов к представлению событийно-непрерывных моделей:**  
  Изучить литературу и современные программные решения, такие как Modelica, MATLAB Simulink, AnyLogic и другие, чтобы понять, каким образом реализуются модели, описывающие гибридные системы. Внимание уделить различным методам описания дискретных и непрерывных процессов, а также проанализировать преимущества и ограничения каждого из них.
* **Выявить ограничения традиционных решений:**  
  Оценить существующие методы с точки зрения их гибкости, адаптивности и возможности интеграции в распределённые вычислительные системы. Особое внимание уделить проблемам масштабируемости, сложности интеграции с внешними сервисами и необходимости создания специализированных инструментальных средств для работы с гибридными моделями.
* **Сформулировать требования к унифицированному представлению:**  
  На основании проведённого анализа сформулировать требования, которым должно удовлетворять предлагаемое представление. Требования должны включать поддержку различных типов данных, возможность динамического обновления модели, совместимость с различными вычислительными платформами, а также обеспечение корректного обмена данными в распределённой системе.
* **Спроектировать архитектуру вычислительного комплекса:**  
  Разработать архитектуру, состоящую из трёх ключевых компонентов: удалённого клиентского интерфейса, сервера-слушателя и сервера-вычислителя. Особое внимание уделить организации взаимодействия между компонентами с использованием протоколов удалённого вызова процедур (gRPC), что позволит обеспечить эффективное и масштабируемое распределение вычислительной нагрузки.
* **Реализовать прототип комплекса и провести его тестирование:**  
  Создать рабочий прототип, реализующий предложенную архитектуру, с акцентом на корректность работы всех ключевых модулей. Прототип должен включать механизмы формирования и передачи моделей в формате JSON, обработку вычислительной логики и динамическое управление вычислительными ресурсами. Провести функциональное и нагрузочное тестирование, чтобы оценить корректность работы системы в различных условиях.
* **Оценить производительность и устойчивость системы:**  
  Провести комплексное исследование, направленное на оценку производительности вычислительного комплекса при различных параметрах нагрузки, а также его устойчивости к ошибкам, сбоям и отказам компонентов. Необходимо изучить возможность масштабирования системы и её адаптацию к изменяющимся требованиям вычислительной среды, а также провести сравнительный анализ с традиционными монолитными решениями.
* Таким образом, поставленные задачи направлены на создание полноценного инструмента для моделирования гибридных систем, который сможет удовлетворить современные требования к вычислительной инфраструктуре, обеспечивая высокую эффективность, гибкость и надёжность при обработке сложных событийно-непрерывных моделей.

## Научная новизна

Научная новизна работы заключается в формировании унифицированного подхода к представлению событийно-непрерывных моделей, что позволяет объединить описание дискретных событий и непрерывных процессов в единой схеме. Такой подход реализуется посредством распределённой архитектуры, в которой используются динамически создаваемые серверы-вычислители. Это позволяет не только оптимизировать обработку вычислительных задач, но и обеспечить масштабируемость системы, что особенно актуально при росте объёмов данных и увеличении требований к быстродействию. Работа демонстрирует, как современные технологии, такие как gRPC и контейнеризация (например, Docker), могут быть использованы для повышения эффективности вычислительного моделирования гибридных систем.

## Практическая значимость

Практическая значимость работы обусловлена разработкой прототипа программного комплекса, способного обеспечить эффективное моделирование событийно-непрерывных систем в условиях распределённых вычислительных сред. Созданное решение может быть использовано в системах вычислительного моделирования, лабораторных стендах и корпоративных инфраструктурах для оптимизации процессов анализа и прогнозирования. Разработанный комплекс позволяет упростить описание моделей, что снижает время и затраты на их разработку и тестирование, а также обеспечивает гибкость архитектуры при интеграции с внешними сервисами. В условиях динамичного развития вычислительных технологий такие решения становятся особенно востребованными для реализации высоконагруженных распределённых систем.

## Объект и предмет исследования

Объектом исследования являются событийно-непрерывные модели, которые представляют собой описание сложных систем, объединяющих дискретные и непрерывные процессы. Такие модели используются для анализа динамики систем в различных областях, начиная от промышленного производства и заканчивая научными исследованиями.

Предметом исследования являются методы представления событийно-непрерывных моделей и архитектурные решения для их обработки. Исследование включает разработку подходов к формализации моделей, определению структуры данных, а также построению распределённой архитектуры, обеспечивающей эффективную обработку и масштабирование вычислительных процессов. Особое внимание уделяется вопросам интеграции различных технологий, таких как gRPC для обмена сообщениями между компонентами системы, и механизмам динамического создания серверов-вычислителей для распределённого выполнения вычислительных задач.

## Методы исследования

В работе использовались комплексные методы, позволяющие охватить все этапы разработки и анализа предлагаемого решения. Среди них – методы теоретического анализа, позволяющие изучить существующие подходы к моделированию событийно-непрерывных процессов и выявить их сильные и слабые стороны. Кроме того, применялись методы проектирования архитектурных решений, что включало детальное моделирование распределённой системы с использованием gRPC для организации взаимодействия между компонентами. Программная реализация проводилась с применением современных средств разработки на языке C и C#, а также инструментов контейнеризации, таких как Docker, что обеспечивало воспроизводимость среды выполнения. Методы тестирования включали функциональное, нагрузочное и сравнительное тестирование, что позволило оценить производительность, отказоустойчивость и масштабируемость системы. Сравнительный анализ проводился с целью выявления преимуществ предлагаемого подхода по сравнению с традиционными монолитными решениями и другими распределёнными архитектурами.

## Структура работы

Работа состоит из введения, четырёх глав, заключения и списка использованных источников. Во введении описаны актуальность темы, цель и задачи исследования, научная новизна, практическая значимость, объект и предмет исследования, а также методы исследования и общая структура работы.

Первая глава посвящена анализу предметной области и существующих подходов к моделированию событийно-непрерывных систем. В ней рассматриваются современные тенденции в области вычислительных технологий, описываются различные методы моделирования гибридных систем, проводится сравнительный анализ программных решений, таких как Modelica, MATLAB Simulink, AnyLogic и других, а также анализируются проблемы монолитных архитектур и преимущества перехода к распределённым системам.

Вторая глава содержит описание проектирования унифицированного представления событийно-непрерывных моделей и архитектуры системы. Здесь детально рассматриваются требования к представлению моделей, логическая схема данных, структура хранения и связи между элементами, а также принципы формирования единого стандарта для описания дискретных и непрерывных процессов. В этой главе отражены подходы к формированию требований, проектированию интерфейсов взаимодействия и выбору форматов обмена данными.

Третья глава посвящена реализации трёхкомпонентной архитектуры, включающей клиентский интерфейс, сервер-слушатель и сервер-вычислитель. В ней приводится подробное описание каждого из компонентов, их функциональных возможностей, механизмов обмена данными посредством gRPC, а также процессов запуска и управления вычислительными узлами. Описаны детали реализации клиентской части на основе MVVM, а также особенности работы серверов на языках C и C#, включая обработку JSON-моделей, выполнение вычислительных циклов, обработку ошибок и освобождение памяти.

Четвёртая глава содержит результаты тестирования и анализа работы разработанного прототипа. В ней приводятся методики и результаты функционального, нагрузочного и сравнительного тестирования, анализируется производительность системы, устойчивость к сбоям и возможности масштабирования. Рассмотрены различные сценарии использования, проанализированы сильные и слабые стороны предложенной архитектуры, а также даны рекомендации по дальнейшему развитию системы.

# Глава 1. Анализ предметной области и существующих подходов

## 1.1. Событийно-непрерывные модели: определение и область применения

Событийно-непрерывные модели (СНМ) — это гибридные модели, предназначенные для описания сложных систем, в которых поведение определяется сочетанием непрерывных процессов и дискретных событий. Их особенность в том, что они объединяют два типа временной эволюции: плавное изменение параметров (например, температуры, давления, скорости) и резкие изменения состояния системы в моменты наступления определённых условий — событий (например, срабатывание датчика, отказ оборудования или переключение режима) (№7), (№8), (№16).

Непрерывный процесс — это изменение параметров во времени, описываемое, как правило, дифференциальными уравнениями. В отличие от него, дискретное событие — это мгновенный переход системы из одного состояния в другое при выполнении заданных условий. СНМ позволяют описывать такие системы целостно, с учётом взаимодействия обеих форм динамики, что делает их особенно полезными при моделировании объектов, в которых непрерывное и дискретное поведение неразрывно связаны (№13), (№14).

Теоретическая основа СНМ строится на объединении средств описания непрерывных и дискретных процессов. Непрерывная динамика моделируется с помощью дифференциальных уравнений, описывающих эволюцию переменных между событиями. Дискретная часть формализуется логикой переходов: при выполнении условий происходит мгновенное изменение состояния системы. Такой подход относится к классу гибридного моделирования, сочетающего методы систем управления и дискретной математики. Типичный пример — системы с переключениями, где в зависимости от текущего состояния и входных воздействий система переходит между режимами функционирования (№14), (№16), (№17).

СНМ находят широкое применение в различных прикладных областях. В промышленности они используются для моделирования автоматизированных производственных линий, где, например, непрерывный поток материалов сочетается с дискретными операциями — сортировкой, контролем качества, упаковкой. Характерным примером является моделирование процессов в кабельной промышленности (№15).

В телекоммуникационных и сетевых системах СНМ применяются для анализа трафика, оценки устойчивости сетей, моделирования переключений маршрутизаторов. Здесь непрерывная часть описывает поток данных, а дискретная — события вроде отказов или изменений маршрута (№8), (№10).

В киберфизических системах, объединяющих физические объекты с цифровыми платформами, СНМ позволяют моделировать взаимодействие сенсоров, актуаторов и программных компонентов. Это особенно актуально для «умных» зданий, автономного транспорта, робототехники и систем интернета вещей (№12), (№18), (№20).

СНМ также применяются в социально-экономических и управленческих системах. Здесь, например, динамика спроса и предложения моделируется как непрерывный процесс, а решения органов управления — как дискретные события, влияющие на состояние системы. Такие подходы используются для анализа устойчивости и оптимизации развития территорий (№5), (№12).

Значимость СНМ подтверждается их включением в перечень приоритетных научно-технологических направлений, утверждённых на государственном уровне (№3). Гибридное моделирование и интеллектуальные системы входят в состав критических технологий, определённых как ключевые для развития экономики и безопасности страны (№4), (№6), (№17). СНМ становятся основой для создания цифровых двойников, интеллектуального управления и адаптивных систем на базе нейросетей и нечеткой логики (№18), (№19).

Таким образом, событийно-непрерывные модели представляют собой мощный инструмент описания и анализа сложных систем. Благодаря способности учитывать оба типа динамики, они позволяют достоверно моделировать поведение реальных объектов в различных сферах — от производства и связи до киберфизических и экономических систем. Это делает СНМ не только актуальным, но и необходимым элементом современного научного и инженерного инструментария.

## 1.2. Обзор существующих методик представления и обработки

Моделирование сложных нелинейных систем (СНМ) основывается на формализмах, способных отразить как непрерывную, так и дискретную динамику. Один из таких подходов — гибридные автоматы. Это структуры, объединяющие свойства конечных автоматов и систем дифференциальных уравнений: в каждом состоянии система описывается непрерывной динамикой, а переходы между состояниями происходят при выполнении заданных логических условий. Гибридные автоматы применяются в задачах анализа безопасности и верификации, а также при моделировании технических и социотехнических процессов (№16), (№17), (№18), (№19), (№20).

Другой широко используемый формализм — системы дифференциальных уравнений с переключениями. Они предполагают наличие нескольких наборов уравнений, активируемых в зависимости от условий. Такой подход применяется в энергетике, биомедицине и других областях, где необходимо учитывать смену режимов функционирования систем. В отличие от гибридных автоматов, он не требует явного задания логики переходов, однако менее прозрачен с точки зрения формального анализа (№12), (№14), (№17).

Инженерное моделирование СНМ часто реализуется с помощью специализированных языков и сред. Simulink — визуальная платформа блочного моделирования, популярная в промышленной среде благодаря интеграции с MATLAB. Она удобна для быстрой сборки моделей, но ограничена в плане гибкости и описания гибридных процессов. Modelica — объектно-ориентированный язык, основанный на декларативном подходе. Он позволяет описывать системы уравнений без указания порядка их решения, что особенно полезно при моделировании многофизичных процессов. Modelica активно используется в сочетании с такими инструментами, как OpenModelica, Dymola и OMSimulator (№7), (№14), (№15), (№16).

Для анализа и обработки моделей применяются численные методы — от классических алгоритмов интегрирования до стохастического моделирования и символьного анализа. Особое внимание уделяется устойчивости решений, точности обработки событий и минимизации численных ошибок (№7), (№13), (№16).

Среди инструментов симуляции СНМ следует выделить несколько направлений. Simulink обеспечивает быструю разработку прикладных моделей и широко применяется в системах управления. Языки Modelica и соответствующие симуляторы (Dymola, OMSimulator) демонстрируют высокую масштабируемость и подходят для сложных инженерных задач. Специализированные пакеты, такие как HyTech и SpaceEx, ориентированы на строгий формальный анализ гибридных автоматов. AnyLogic отличается универсальностью, позволяя объединять дискретно-событийные, системно-динамические и агентные подходы, что делает его эффективным для моделирования распределённых социотехнических систем (№8), (№9), (№10), (№14), (№15), (№17).

Сравнение подходов показывает, что Simulink выигрывает в простоте освоения и скорости построения моделей, но уступает в гибкости и масштабируемости. Modelica обеспечивает высокую производительность при моделировании комплексных систем, особенно с многофизичными компонентами. Гибридные автоматы предоставляют строгую математическую основу и применимы там, где необходима верификация поведения. AnyLogic подходит для междисциплинарных задач, но ограничен в работе с непрерывной динамикой.

Выбор подхода определяется характером задачи: для формальной верификации предпочтительны гибридные автоматы, для моделирования реальных технических систем — Modelica, для быстрой инженерной отладки — Simulink, для анализа сложных распределённых систем — AnyLogic. С учётом стратегических направлений научно-технологического развития, таких как цифровое проектирование и управление сложными объектами (№2), (№3), (№4), (№6), наибольший потенциал имеет развитие гибридных подходов и создание универсальных симуляторов нового поколения.

## 1.3. Анализ ограничений традиционных подходов

Современные задачи моделирования социально-неоднородных моделей (СНМ) требуют высокой гибкости, масштабируемости и совместимости используемых инструментов. Однако традиционные подходы зачастую оказываются неэффективными в условиях изменяющихся требований. Одним из основных ограничений выступает низкая адаптивность: большинство моделей опирается на жёсткую архитектуру, не предполагающую оперативного обновления или расширения, что затрудняет масштабирование и модификацию.

Существенным препятствием остаётся отсутствие единых стандартов описания моделей и форматов обмена данными, что затрудняет совместимость различных платформ и систем, особенно в распределённых проектах (11). Это приводит к фрагментации программных решений, снижает повторное использование компонентов и усложняет переносимость моделей между средами.

Отдельной проблемой является ограниченная интеграция с внешними цифровыми сервисами. В условиях развития облачных вычислений и интеллектуальных платформ такая замкнутость снижает эффективность моделирующих систем (8), (15). Особенно это актуально при необходимости взаимодействия с распределёнными вычислительными средами, где важны синхронизация процессов и устойчивость передачи данных. Традиционные архитектуры, ориентированные на централизованное исполнение, как правило, не поддерживают масштабируемую организацию распределённых вычислений (5), (10).

С вычислательной точки зрения традиционные подходы демонстрируют слабую производительность при росте объёмов данных и числа взаимодействующих компонентов. Это выражается в снижении точности результатов, увеличении времени отклика и затруднённой верификации. Отсутствие механизмов параллельной и гибридной обработки делает такие методы неэффективными при решении задач высокой сложности (17), (20).

Сравнительный анализ ограничений позволяет выделить три группы: технические, архитектурные и вычислительные. К техническим относятся отсутствие унифицированных стандартов и слабая интеграция с внешними сервисами. Архитектурные связаны с жёсткой структурой и низкой адаптивностью систем. Вычислительные ограничения выражаются в неспособности обрабатывать большие объёмы данных и организовывать параллельные процессы.

Таким образом, традиционные подходы к моделированию СНМ характеризуются системными недостатками, ограничивающими их применимость в современных условиях. Преодоление указанных барьеров требует перехода к более гибким, масштабируемым и стандартизированным моделирующим платформам, способным эффективно функционировать в распределённых цифровых средах (2), (4), (12).

## 1.4. Сравнительная характеристика известных программных решений

В ходе анализа были рассмотрены программные среды Modelica, MATLAB Simulink, AnyLogic, а также ряд дополнительных решений, таких как OpenModelica и Scilab/Xcos. Эти инструменты широко применяются для моделирования сложных технических и социально-экономических систем, обладают развитым функционалом, однако не всегда обеспечивают достаточную гибкость для интеграции в распределённые вычислительные среды и зачастую предъявляют высокие требования к ресурсам.

MATLAB Simulink — одна из наиболее распространённых платформ для моделирования непрерывных и дискретных процессов. Она предлагает визуальный интерфейс, обширную библиотеку блоков и тесную интеграцию с экосистемой MATLAB. Благодаря этим возможностям Simulink активно используется в инженерной практике, в том числе для задач автоматизации и разработки встроенных систем, включая автокодогенерацию (№7), (№13). Тем не менее, расширение функционала в сторону распределённых вычислений требует дополнительных решений, таких как Simulink Real-Time или MATLAB Production Server, что усложняет реализацию и увеличивает затраты (№17).

Modelica реализует декларативный подход, позволяя описывать поведение систем через физические уравнения. Это особенно эффективно в мультидисциплинарных задачах, где необходимо объединять различные физические домены — от механики до термодинамики и электротехники (№14), (№16). Среди реализаций Modelica наибольшее распространение получили Dymola и OpenModelica. Первая обеспечивает высокий уровень профессиональной поддержки, но требует лицензии; вторая — свободно распространяемая, однако уступает в удобстве и производительности. Поддержка интеграции с распределёнными системами ограничена: хотя стандарт FMI позволяет обмениваться моделями между платформами, полноценная работа в кластерной среде реализована частично (№11), (№17).

AnyLogic — гибридная платформа, сочетающая дискретно-событийное, системно-динамическое и агентное моделирование. Она особенно эффективна при моделировании социально-экономических процессов, логистических цепочек и человеко-ориентированных систем (№12), (№15). Преимущество AnyLogic — в её ориентированности на Java, что обеспечивает гибкость и масштабируемость, а также встроенные средства для облачного исполнения моделей. Это даёт ей серьёзное преимущество в задачах распределённого моделирования. Вместе с тем, высокая сложность разработки и необходимость владения языком Java могут ограничивать доступность платформы для менее подготовленных пользователей (№18), (№19).

Сравнительный анализ позволяет выделить ключевые особенности каждой среды. Simulink предлагает высокую инженерную зрелость и визуальное удобство, но требует дополнительных средств для интеграции в распределённые решения. Modelica демонстрирует мощные возможности физического моделирования, но слабо масштабируется и зависит от конкретной реализации. AnyLogic, напротив, хорошо подходит для гибридного и многопользовательского моделирования, включая задачи с элементами социально-нечёткой логики (№5), (№10), (№18).

Таким образом, выбор программной платформы должен основываться не только на её функциональности, но и на способности интегрироваться в распределённую архитектуру, а также соответствовать специфике предметной области. В ряде случаев наиболее целесообразным оказывается комбинированное использование нескольких решений — особенно при разработке распределённых интеллектуальных систем, в которых важны гибкость, масштабируемость и поддержка разнообразных моделей поведения (№17), (№20).

## 1.5. gRPC как инструмент взаимодействия в распределённой системе

1.5 gRPC как инструмент взаимодействия в распределённой системе

gRPC — это современный фреймворк удалённого вызова процедур (Remote Procedure Call, RPC), разработанный компанией Google для создания высокоэффективных распределённых систем. Он основан на использовании протокола HTTP/2 и системы сериализации Protocol Buffers, что делает его особенно подходящим для организации межсервисного взаимодействия в условиях высокой нагрузки и жёстких требований к производительности. Архитектура gRPC ориентирована на строго типизированное и структурированное взаимодействие между компонентами, обеспечивая устойчивую работу как в локальных, так и в облачных и гибридных вычислительных средах (№1).

Основу архитектуры gRPC составляет модель «клиент — сервер», в которой клиент вызывает удалённые процедуры на стороне сервера, используя синтаксис, аналогичный вызову локальных функций. Обмен сообщениями осуществляется по протоколу HTTP/2, обеспечивающему постоянное соединение, мультиплексирование запросов и эффективную работу с потоками данных. Это позволяет минимизировать задержки при передаче и использовать двунаправленную потоковую передачу данных, что особенно актуально в задачах мониторинга, телеметрии и управления в реальном времени (№2).

Протокол Protocol Buffers, также разработанный Google, используется в gRPC в качестве механизма сериализации сообщений. Его бинарный формат обеспечивает компактность и высокую скорость обработки данных по сравнению с текстовыми форматами вроде JSON и XML. Структура сообщений описывается в специальных .proto-файлах, на основе которых автоматически генерируется код на поддерживаемых языках программирования. Это обеспечивает строгую типизацию, ускоряет разработку и исключает расхождения между интерфейсами взаимодействующих компонентов (№3).

gRPC получил широкое распространение в микросервисной архитектуре. Так, в системах управления заказами он используется для взаимодействия между модулями аутентификации, расчёта стоимости и логистики, позволяя достичь высокой согласованности и масштабируемости. В облачных платформах, например Kubernetes, gRPC обеспечивает эффективную маршрутизацию и балансировку нагрузки между контейнерами. Кроме того, он активно применяется в системах интернета вещей (IoT) — например, при передаче телеметрических данных от распределённых сенсоров к серверам обработки, где важны низкие задержки и высокая пропускная способность (№4).

Таким образом, gRPC представляет собой универсальное средство построения надёжных и производительных распределённых систем. Его архитектура обеспечивает высокую эффективность взаимодействия, строгую типизацию данных и широкие возможности масштабирования. Применение gRPC соответствует приоритетным направлениям научно-технологического развития Российской Федерации, включая цифровую трансформацию, унификацию программных решений и разработку критически значимой информационной инфраструктуры (№2), (№3), (№5).

## 1.6. Постановка задачи разработки унифицированного представления

В данной работе ставится задача разработки унифицированного представления событийно-непрерывных моделей. Разработка предполагает создание архитектуры, состоящей из трёх компонентов: удалённого клиентского интерфейса, сервера-слушателя, принимающего запросы по gRPC, и сервера-вычислителя, выполняющего вычисления и взаимодействующего с клиентом также по gRPC.

# Глава 2. Проектирование унифицированного представления событийно-непрерывных моделей

## 2.1. Общая схема архитектуры системы

Описание трёхкомпонентной модели

Основные компоненты системы:

• Клиентский интерфейс – отвечает за взаимодействие пользователя с системой, отправку запросов и получение данных. Включает в себя веб-интерфейс и API, через которые пользователь может взаимодействовать с системой.

• Сервер-слушатель – принимает входящие запросы, управляет распределением вычислительных ресурсов и динамическим развертыванием серверов-вычислителей. Работает по принципу балансировщика нагрузки, оценивая текущую загруженность системы и направляя задачи на соответствующие вычислительные узлы.

• Сервер-вычислитель – выполняет обработку событийно-непрерывных моделей, проводит вычисления и передаёт результаты клиентскому интерфейсу. В его основе лежит специализированная библиотека для обработки гибридных систем, поддерживающая параллельные вычисления.

Научная новизна: предлагаемый подход объединяет обработку событийно-непрерывных моделей в распределённой вычислительной среде, что повышает гибкость и масштабируемость системы. Использование сервер-слушателя для динамического развертывания серверов-вычислителей увеличивает адаптивность системы. Включение гибридных вычислительных стратегий позволяет эффективно распределять нагрузку и минимизировать задержки при вычислениях.

Пример расчёта гибридной системы: Допустим, модель состоит из дискретных и непрерывных процессов. Сервер-вычислитель параллельно рассчитывает дифференциальные уравнения для непрерывной части системы, а сервер-слушатель координирует обработку событий, вызывающих изменения в дискретных переменных. Это позволяет добиться высокой точности при моделировании сложных систем, таких как сетевые протоколы или промышленные процессы.

## 2.2. Основные сущности системы

Ключевые сущности:

• Модель – формальное представление исследуемой системы, включающее события и непрерывные процессы. Модель описывает взаимодействие различных компонентов системы, их динамику и реакцию на внешние воздействия. В рамках унифицированного подхода модель должна поддерживать как дискретные события, так и непрерывные изменения.

• Состояние – набор параметров, описывающих текущее состояние модели в заданный момент времени. Включает в себя значения дискретных переменных (например, флагов состояний) и непрерывных параметров (например, текущего уровня заполнения резервуара). Состояние является основным носителем информации для моделирования эволюции системы во времени.

• Переменные – величины, изменяющиеся в процессе моделирования и влияющие на поведение системы. Они подразделяются на дискретные и непрерывные, определяя как мгновенные изменения (например, срабатывание триггера), так и плавные переходы (например, изменение температуры). Важной особенностью является поддержка связи между этими типами переменных для корректного представления гибридных систем.

Научная новизна: введение формального описания ключевых сущностей позволяет создать универсальное представление модели, обеспечивающее её переносимость между различными вычислительными средами. Подход способствует унификации работы с событиями и непрерывными процессами, улучшая их совместную обработку. Включение гибридных переменных в модель позволяет описывать сложные системы, где взаимодействуют дискретные и непрерывные процессы. Например, система регулирования температуры может включать дискретные события включения/выключения нагревателя и непрерывное изменение температуры среды.

## 2.3. Разработка структуры хранения и связей

Разработка структуры хранения данных включает в себя определение способов организации данных таким образом, чтобы обеспечить их целостность, согласованность и удобство доступа. Основное внимание уделяется выбору моделей хранения, которые позволяют эффективно управлять изменениями и адаптироваться к различным сценариям использования.

При проектировании структуры хранения необходимо учитывать как логические, так и физические аспекты организации данных. Логический уровень включает в себя определение основных сущностей, их атрибутов и взаимосвязей, что позволяет формировать единую схему данных. Физическая организация хранения данных зависит от выбранной платформы и включает в себя способы индексирования, оптимизацию доступа и обеспечение масштабируемости.

Особое внимание уделяется механизмам связи между сущностями, которые определяют, как данные будут взаимодействовать друг с другом. Это могут быть реляционные связи, объектные ассоциации или графовые представления, в зависимости от требований системы. Для поддержания согласованности данных используются механизмы ссылочной целостности, транзакционности и версиирования.

Важной задачей при разработке структуры хранения является обеспечение поддержки динамических изменений моделей, что достигается за счет гибкой структуры метаданных, использования схем эволюции данных и механизмов миграции. Это позволяет адаптировать систему к изменяющимся требованиям без потери данных и с минимальными затратами на переработку структуры.

## 2.4. Формирование требований к унифицированному представлению

Разработка требований к представлению событийно-непрерывных моделей включает в себя анализ существующих подходов к описанию таких моделей, выявление их особенностей и ограничений. Одной из ключевых задач является обеспечение совместимости с различными вычислительными платформами, что требует использования универсальных методов кодирования и структурирования данных. Важным аспектом является поддержка гибкой структуры данных, позволяющей адаптировать представление моделей к различным условиям использования. Это требует четкого определения требований к языкам описания, возможностям масштабирования и интеграции с другими системами. Также необходимо учитывать вопросы эффективности хранения и обработки данных, что особенно важно при работе с крупными моделями и сложными вычислительными процессами.

В процессе формирования требований необходимо учитывать специфику структурирования данных, полученных от клиента. Формат JSON, используемый для описания модели состояний, задает базовые параметры представления: данные включают описание состояний, набор формул, условия и переходы. Требования должны обеспечивать корректное отображение этих элементов, чтобы при десериализации JSON с помощью библиотеки cJSON данные преобразовывались в дерево объектов, соответствующее логике работы вычислителя.

Кроме того, при формировании требований учитываются особенности описания состояний модели. Каждое состояние должно содержать набор формул, описывающих простейшие арифметические операции, а также перечень переходов, определяемых условными операторами. Строгость структуры данных в этом случае обеспечивает последовательное выполнение вычислений, а также корректное обновление значений переменных. Требования к представлению должны учитывать, каким образом описываются имена состояний, переменных и операций, чтобы избежать неоднозначности и ошибок при последующей обработке.

Еще одним важным аспектом является обеспечение поддержки динамических изменений модели. При моделировании событийно-непрерывных систем данные могут изменяться в процессе вычислений, что требует наличия механизмов для обновления структуры хранения и корректной синхронизации информации между компонентами системы. Формирование требований включает определение схем эволюции данных, обеспечение транзакционности операций и поддержки миграции структур данных без потери целостности.

Наконец, требования к унифицированному представлению должны предусматривать возможность одновременной работы с данными в распределенной среде. Это означает, что структура представления должна быть адаптивной и поддерживать взаимодействие компонентов системы посредством стандартных протоколов, таких как gRPC. Такая организация обеспечивает согласованность передачи и обработки данных между клиентским интерфейсом, сервером-слушателем и сервером-вычислителем, что критически важно для корректного функционирования системы в целом.

## 2.5. Логическая схема данных и описание типов данных

Разработка логической схемы данных включает в себя проектирование структуры хранения моделей и их состояний, а также выбор соответствующих форматов представления данных. Одним из ключевых аспектов является обеспечение возможности эффективного управления версиями и изменениями, что требует четко определенных механизмов для фиксации и отслеживания модификаций.

При проектировании схемы необходимо учитывать принципы целостности данных, их доступности и масштабируемости. Для этого используются методы нормализации, дедупликации и компрессии информации, а также механизмы индексации, позволяющие ускорить доступ к данным.

Выбор типов данных играет важную роль в оптимизации хранения и обработки информации. В зависимости от характеристик моделируемых объектов применяются примитивные и составные типы данных, а также структуры, поддерживающие динамическое обновление. Это позволяет повысить точность моделирования и снизить избыточность хранимой информации.

Кроме того, логическая схема должна учитывать возможность интеграции с внешними источниками данных, что требует поддержки различных форматов представления, таких как реляционные и нереляционные базы данных, а также специализированные форматы для хранения временных рядов и графовых структур.

Научная новизна: разработка адаптивной логической схемы данных позволяет обеспечить гибкость структуры при изменении модели, а также повысить эффективность хранения и обработки информации. Поддержка различных типов данных и их динамическое обновление повышает точность моделирования сложных систем.

## 2.6. Проектирование интерфейсов взаимодействия и обработки данных

Разработка интерфейсов для взаимодействия между клиентами, серверами-слушателями и серверами-вычислителями, а также механизмов обработки входных и выходных данных. Важной частью проектирования является обеспечение эффективной передачи данных между компонентами системы, минимизация задержек и потерь при обмене информацией. Это требует тщательной проработки архитектуры взаимодействия, выбора подходящих протоколов передачи данных и обеспечения надёжности системы.

Одним из ключевых аспектов является поддержка унифицированных интерфейсов, которые обеспечивают гибкость и масштабируемость системы. Унифицированные интерфейсы позволяют снизить сложность интеграции новых компонентов, обеспечивая совместимость с различными сервисами и платформами. Кроме того, такие интерфейсы упрощают сопровождение системы и облегчают её адаптацию к изменяющимся требованиям.

Также важна реализация механизмов обработки входных и выходных данных, включая валидацию, нормализацию и преобразование информации для корректной работы с различными источниками данных. Это позволяет обеспечить корректность, целостность и согласованность передаваемой информации. Для этого используются алгоритмы предобработки, фильтрации и адаптации данных к требованиям системы.

Научная новизна: реализация интерфейсов с учётом динамической адаптации к различным типам моделей повышает универсальность системы и её применимость в различных областях. Использование унифицированных интерфейсов снижает сложность интеграции с внешними сервисами.

## 2.7. Разработка протоколов взаимодействия на базе gRPC

Определение форматов обмена данными, разработка API взаимодействия между вычислительными узлами и клиентским интерфейсом с использованием gRPC. Включает в себя описание структур данных, методов вызова удалённых процедур и механизмы аутентификации и авторизации. Разработка API осуществляется с учётом требований высокой производительности и надёжности системы, что особенно важно для распределённых вычислительных сред.

Использование gRPC обеспечивает эффективную сериализацию данных с применением Protocol Buffers, что позволяет минимизировать накладные расходы при передаче информации. Кроме того, gRPC поддерживает двунаправленный поток данных, что делает его удобным инструментом для построения интерактивных клиент-серверных приложений.

Дополнительное преимущество gRPC заключается в его встроенной поддержке балансировки нагрузки и автоматического обнаружения сервисов, что облегчает разработку и сопровождение сложных распределённых систем. Благодаря этим возможностям достигается высокая масштабируемость архитектуры, а также улучшается её отказоустойчивость.

Научная новизна: применение gRPC для взаимодействия между вычислительными компонентами обеспечивает низкую задержку и высокую производительность, а также улучшает кроссплатформенную совместимость системы. Это решение способствует стандартизации коммуникации между распределёнными вычислительными узлами, упрощая масштабирование и поддержку.

# Глава 3. Реализация программного комплекса

## 3.1. Средства разработки и используемые технологии

Проект разработан на языке программирования C с использованием библиотеки gRPC для организации удалённого вызова процедур. Код структурирован с разделением ответственности между клиентским интерфейсом, сервером-слушателем и серверами-вычислителями. Клиентский интерфейс отвечает за формирование запросов и обработку ответов, сервер-слушатель управляет маршрутизацией вызовов, а серверы-вычислители выполняют непосредственные вычисления и обработку данных.

Для контейнеризации используется Docker, что обеспечивает изоляцию и воспроизводимость среды выполнения. Это позволяет запускать проект в однородной среде на различных платформах, минимизируя проблемы совместимости. Использование контейнеров также упрощает развертывание и поддержку системы, обеспечивая её масштабируемость.

Для управления зависимостями и сборки проекта применяется система CMake, которая обеспечивает переносимость кода и автоматизацию процесса компиляции. В системе предусмотрены механизмы логирования и отладки, что облегчает диагностику и анализ работы программы.

## 3.2. Реализация клиентского интерфейса

Клиентская часть системы реализована с использованием архитектурной модели MVVM (Model-View-ViewModel). Это обеспечивает разделение представления, логики и данных, что упрощает управление сложными интерфейсами и взаимодействие с сервером. Клиент разработан на языке C# с использованием .NET Framework 4.8.

В рамках реализации клиентского интерфейса предусмотрены следующие функциональные возможности:

* **Формирование и отправка запросов:** Клиент предоставляет пользователю окно настроек, в котором можно указать IP-адрес сервера и порт для подключения (по умолчанию сервер прослушивает соединения на IP 0.0.0.0 и порту 55555). После ввода настроек, клиент формирует запрос на расчёт, конвертируя модель в формат JSON. Этот запрос передается серверу-слушателю через gRPC.
* **Обработка входящих данных:** После завершения расчётов сервер-вычислитель формирует результирующий JSON и отправляет его обратно клиенту. Клиент принимает полученные данные, декодирует их и обрабатывает, чтобы предоставить пользователю информацию о результатах вычислений.
* **Интерфейс взаимодействия:** Визуальное представление реализовано с использованием фреймворка Avalon, а для отображения графов и визуализации моделей применяется Graph#. Кроме того, ANTLR используется для создания и разбора синтаксических деревьев, необходимых для обработки сложных выражений и формул, содержащихся в моделях.
* **Логирование и обработка ошибок:** Логирование ошибок на стороне клиента реализовано минимально – в клиентском приложении ведется общий журнал ошибок, что затрудняет детальное выяснение причин возникновения ошибок, однако позволяет фиксировать факт возникновения неполадок.

Таким образом, клиентский интерфейс обеспечивает полноценное взаимодействие с серверной частью системы: формирует запросы, отправляет их по gRPC, ожидает получения ответов и отображает результаты пользователю, обеспечивая при этом базовый уровень логирования и обработки ошибок.

## 3.3. Реализация сервера-слушателя

• **Создание gRPC API:**  
Сервер-слушатель, написанный на языке C, настроен для приема новых подключений от клиентов посредством TCP-сокета. Он предоставляет gRPC API, посредством которого клиенты устанавливают соединение с сервером. При поступлении запроса на подключение сервер-слушатель не получает данные в формате JSON, а только инициирует соединение.

• **Обработка входящих подключений:**  
Сервер-слушатель прослушивает определенный порт (например, 55555) на адресе 0.0.0.0 и принимает подключения от клиентов. Как только новое подключение установлено, сервер-слушатель порождает новый процесс с помощью системного вызова fork(). Этот новый процесс становится сервером-вычислителем, с которым клиент устанавливает полноценное взаимодействие. После создания процесса сервер-слушатель освобождает используемый сокет и продолжает ожидание новых подключений.

• **Динамический запуск серверов-вычислителей:**  
При поступлении нового подключения сервер-слушатель инициирует запуск отдельного процесса – сервера-вычислителя. Этот процесс получает параметры подключения от клиента и начинает выполнять вычислительную логику, включая десериализацию JSON-модели, выполнение последовательных вычислений и передачу результатов обратно клиенту. Таким образом, сервер-слушатель продолжает работать и обрабатывать новые подключения, обеспечивая параллельную обработку запросов от разных клиентов.

## 3.4. Реализация сервера-вычислителя

• **Обработка событийно-непрерывных моделей:**  
Сервер-вычислитель получает модель от клиента в формате JSON. После приёма данных производится их десериализация с помощью библиотеки cJSON, которая преобразует JSON в дерево объектов. Затем начинается последовательная обработка состояний модели: для каждого состояния сервер выполняет набор формул, описанных в модели, используя встроенные математические функции для арифметических операций (сложение, вычитание, умножение, деление и модуль). При этом значения переменных, полученные в результате вычислений, могут использоваться в последующих формулах того же состояния. После выполнения формул для текущего состояния сервер проверяет условия перехода и, если они выполняются, переходит к обработке следующего состояния.

• **Функциональные модули:**

* **Переменных:**  
  Сервер-вычислитель обеспечивает хранение и обновление значений переменных модели. В процессе вычислений создаются и поддерживаются списки переменных, где каждое имя переменной ассоциируется с её текущим значением. При обработке формул производится поиск необходимых переменных, обновление их значений, а также добавление новых переменных в список, если таковые встречаются впервые.
* **Математические функции:**  
  Для реализации вычислений используются базовые арифметические операции, такие как сложение, вычитание, умножение, деление и операция взятия модуля. Каждая функция вызывается последовательно в зависимости от порядка следования формул в модели. Кроме того, предусмотрены функции для выполнения логических операций, необходимых для оценки условий переходов между состояниями, такие как сравнение значений (больше, меньше, равно и т.д.).
* **Модель: состояния и переходы:**  
  Модель описывается через состояние, содержащее набор формул и переходов. Для каждого состояния определяются формулы, которые выполняют вычисления, а также условия и действия для перехода в другие состояния. После выполнения всех вычислений для текущего состояния сервер проверяет заданные условия перехода; если условие удовлетворено, сервер переходит к обработке следующего состояния, обновляя значения переменных в соответствии с описанными действиями. Такой механизм обеспечивает обработку динамических изменений модели и корректное моделирование гибридных систем.

## 3.5. Логика обмена сообщениями между компонентами через gRPC

Компоненты системы взаимодействуют посредством gRPC-сообщений, определённых в protobuf-файлах. Этот механизм позволяет стандартизировать и организовать обмен информацией между клиентским интерфейсом, сервером-слушателем и сервером-вычислителем.

При установлении соединения клиент инициирует запрос на сервер-слушатель, используя заранее определённые gRPC-методы. Сообщение, отправляемое клиентом, содержит всю необходимую информацию для идентификации сессии, а также данные, требуемые для дальнейших вычислений. Сервер-слушатель принимает запрос, выполняет первичную обработку и обеспечивает установление канала для обмена данными.

После установления соединения сервер-слушатель передаёт запрос серверу-вычислителю, который был порожден посредством вызова fork(). В данном случае gRPC обеспечивает прозрачную маршрутизацию сообщений между компонентами, благодаря чему сервер-вычислитель получает все необходимые параметры для выполнения вычислительной логики. Он выполняет десериализацию полученных данных, осуществляет последовательное выполнение формул и обработку состояний модели, а затем формирует результирующий JSON с результатами вычислений.

Полученный результат передается обратно по тому же маршруту: от сервера-вычислителя через сервер-слушатель к клиентскому интерфейсу. Весь процесс обмена сообщениями осуществляется с использованием gRPC, что гарантирует стандартизированный и надежный механизм передачи данных между распределенными компонентами системы.

# Глава 4. Исследование работы и тестирование

## 4.1. Подходы к тестированию распределённых вычислительных систем

Тестирование распределённых систем требует особого подхода, включающего использование эмуляторов сетевых задержек, нагрузочного тестирования и валидации целостности данных. Важным аспектом является моделирование реальных условий эксплуатации системы с различными параметрами отказоустойчивости.

## 4.2. Разработка и проведение тестов клиент-серверного взаимодействия

Клиент-серверное взаимодействие тестируется с учётом сетевых условий, пропускной способности и задержек. Используются инструменты для автоматизированного тестирования API, а также нагрузочные тесты, имитирующие массовые запросы к серверу.

## 4.3. Проверка корректности работы вычислителя на тестовых данных

Для проверки вычислительного модуля применяются тестовые наборы данных с заранее известными результатами. Проводится сравнение фактических и ожидаемых результатов, а также анализ ошибок округления и точности вычислений.

## 4.4. Оценка производительности и масштабируемости системы

Производительность оценивается через тесты нагрузки, измеряющие скорость выполнения задач при различных параметрах нагрузки. Масштабируемость проверяется путём увеличения числа вычислительных узлов и анализа изменений в производительности.

## 4.5. Сравнительный анализ с монолитными и другими распределёнными архитектурами.

Проводится сравнение гибридной системы с монолитными и чисто распределёнными решениями на основе метрик производительности, отказоустойчивости и удобства эксплуатации. Включены примеры расчётов гибридных систем с оценкой их преимуществ и недостатков.

## 4.6. Анализ устойчивости к сбоям и нагрузочным ситуациям

Моделируются различные сценарии отказов, такие как выход из строя узлов, перегрузка сети и ошибки в обработке данных. Анализируются механизмы восстановления и балансировки нагрузки.

# Заключение

* Основные итоги работы
* Достижение целей и решение поставленных задач
* Практическая значимость и возможное применение
* Рекомендации для дальнейшего развития

# Список литературы

2 УКАЗ ПРЕЗИДЕНТА РОССИЙСКОЙ ФЕДЕРАЦИИ Об утверждении приоритетных направлений развития науки, технологий и техники в Российской Федерации и перечня критических технологий Российской Федерации http://www.kremlin.ru/acts/bank/33514

3 Указ Президента РФ от 7 июля 2011 г. N 899 "Об утверждении приоритетных направлений развития науки, технологий и техники в Российской Федерации и перечня критических технологий Российской Федерации" (с изменениями и дополнениями) https://base.garant.ru/55171684/

4 Приоритетные направления развития науки, технологий и техники Российской Федерации https://www.extech.ru/info/catalogs/razv\_sci.php

5 ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ В УПРАВЛЕНИИ ЭКОНОМИЧЕСКОЙ СИСТЕМЫ МУНИЦИПАЛЬНОГО ОБРАЗОВАНИЯ https://top-technologies.ru/ru/article/view?id=35084

6 Приоритетные направления развития науки http://www.opf.nsu.ru/ru/content/%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BD%D1%8B%D0%B5-%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D1%80%D0%B0%D0%B7%D0%B2%D0%B8%D1%82%D0%B8%D1%8F-%D0%BD%D0%B0%D1%83%D0%BA%D0%B8

7 Непрерывное моделирование https://ru.ruwiki.ru/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D0%B5\_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5

8 Событийное моделирование цифровой системы регулирования https://cyberleninka.ru/article/n/sobytiynoe-modelirovanie-tsifrovoy-sistemy-regulirovaniya

9 ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ В УПРАВЛЕНИИ ЭКОНОМИЧЕСКОЙ СИСТЕМЫ МУНИЦИПАЛЬНОГО ОБРАЗОВАНИЯ https://top-technologies.ru/ru/article/view?id=35084

10 Оценка надежности программного обеспечения методами дискретно-событийного моделирования https://cyberleninka.ru/article/n/otsenka-nadezhnosti-programmnogo-obespecheniya-metodami-diskretno-sobytiynogo-modelirovaniya

11 унификация программного обеспечения https://biblioserver.usurt.ru/cgi-bin/irbis64r\_13/cgiirbis\_64.exe?LNG=uk&Z21ID=&I21DBN=LAN\_PRINT&P21DBN=LAN&S21STN=1&S21REF=&S21FMT=fullw\_print&C21COM=S&S21CNR=&S21P01=0&S21P02=0&S21LOG=1&S21P03=K%3D&S21STR=%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F+%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE+%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F

12 Возможности применения гибридного подхода в моделировании социально-экономических и социотехнических систем https://journals.vsu.ru/sait/article/view/10585

13 Моделирование — это способ создания точной копии объекта или процесса в виде модели, которая позволяет нам лучше понять его структуру и функционирование. https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-uml/

14 Моделирование жёстких гибридных систем с односторонними событиями в инструментальной среде Исма https://cyberleninka.ru/article/n/modelirovanie-zhyostkih-gibridnyh-sistem-s-odnostoronnimi-sobytiyami-v-instrumentalnoy-srede-isma

15 ОСОБЕННОСТИ РАЗРАБОТКИ ИМИТАЦИОННЫХ МОДЕЛЕЙ СЛОЖНЫХ ТЕХНИЧЕСКИХ СИСТЕМ НА ПРИМЕРЕ КАБЕЛЬНОГО ПРОИЗВОДСТВА https://7universum.com/ru/tech/archive/item/17644

16 СИМВОЛЬНАЯ СПЕЦИФИКАЦИЯ И АНАЛИЗ ПРОГРАММНЫХ МОДЕЛЕЙ ГИБРИДНЫХ СИСТЕМ https://www.nstu.ru/files/dissertations/dissertaciya\_bessonov\_17032016\_1459158757.pdf

17 Гибридные методы моделирования и оптимизации в сложных системах https://megagrant.ru/labs/lab\_rus\_91219/

18 Метод гибридного управления на основе адаптивной системы нейро-нечеткого вывода https://cyberleninka.ru/article/n/metod-gibridnogo-upravleniya-na-osnove-adaptivnoy-sistemy-neyro-nechetkogo-vyvoda

19 Гибридная интеллектуальная система https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D1%80%D0%B8%D0%B4%D0%BD%D0%B0%D1%8F\_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F\_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

20 Динамическая верификация гибридных систем https://tmpaconf.org/images/pdf/pakulin.pdf