МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра системного анализа и
автоматического управления

# РАЗРАБОТКА ФРЕЙМВОРКА ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 481 группы
направления 27.03.03 — Системный анализ и управление
факультета КНиИТ
Заварзина Алексея Сергеевича

Научный руководитель
к. ф.-м. н., доцент _____ О. А. Осипов

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2021

# ВВЕДЕНИЕ

**Актуальность темы.** Для исследования и оптимизации систем используется моделирование, методы которого можно разделить на две большие группы – аналитическое и имитационное [1–3]. При аналитическом подходе описание системы может быть выполнено, например, посредством дифференциальных или разностных уравнений, графовых методов и т.д., которые не всегда позволяют найти аналитическое решение существующими методами. Имитационное моделирование позволяет описывать модели систем через процессы или события, которые в них происходят. При таком подходе модель системы реализуется в виде компьютерной программы. Отличие имитационного моделирования от аналитического заключается в том, что намного легче описывать сложные системы и находить для них численные характеристики.

Имитационное моделирование может применяться в самых разных сферах деятельности. Приведем список конкретных задач, при решении которых моделирование особенно эффективно:

— проектирование и анализ производственных систем;
— оценка различных систем вооружений и требований к их материально-техническому обеспечению;
— определение требований к оборудованию и протоколам сетей связи;
— определение требований к оборудованию и программному обеспечению различных компьютерных системы;
— проектирование и анализ работы транспортных систем;
— оценка проектов создания различных организаций массового обслуживания;
— модернизация различных процессов в деловой сфере;
— анализ финансовых и экономических систем.

За последнее время имитационное моделирование стало одним из основных и наиболее распространённых инструментов исследования сложных систем. Роль имитационного моделирования в промышленном и информационном мире велика. Благодаря возможности построения имитационной модели реальной системы исследователи могут проводить большое количество экспериментов, анализировать поведение системы, собирать статистические данные и делать соответствующие выводы. Эксперименты над имитационной моделью, в отличие от экспериментов над реальной системой, позволяют

сэкономить временные и денежные ресурсы. Таким образом, имитационное моделирование является важным инструментом исследования и анализа поведения реальных систем, который в последнее время активно развивается. Для создания имитационных моделей используются различные подходы:

— программирование компьютерных моделей с помощью универсальных языков (C++, Java, Delphi);
— программирование компьютерных моделей с применением специализированных языков моделирования (SIMULA, AnyLogic);
— построение компьютерных моделей и проведение имитационных экспериментов при помощи специализированных компьютерных сред с графическим редактором процессов (AnyLogic, Arena, SIMUL8, NetLogo);
— включение средств имитационного моделирования в стандартные математические компьютерные системы (пакет Simulink системы Matlab).

Большинство компьютерных сред имитационного моделирования обладают некоторыми недостатками. Например, многие продукты вынуждают пользователя использовать свой собственный язык программирования, некоторые системы с хорошо проработанным функционалом (AnyLogic, SIMUL8) достаточно дороги для обычного пользователя (хоть и зачастую имеют пробную/бесплатную версию с сильно ограниченными возможностями), другие являются примитивными, медленными, сложными.

В данной работе была поставлена цель реализовать основные компоненты системы для дискретно-событийного имитационного моделирования. В качестве языка разработки для данной системы использовался строго типизированный объектно-ориентированный язык программирования общего назначения Java [4–6], что упростит использование и позволит в дальнейшем расширять указанную систему.

**Цель бакалаврской работы** — разработать систему дискретно-событийного имитационного моделирования.

Поставленная цель определила **следующие задачи**:

1. Ознакомиться с основами имитационного моделирования;
2. Изучить дискретно-событийное моделирование;
3. Изучить существующие решения для моделирования систем;
4. Разработать основные модули фреймворка дискретно-событийного мо-

делирования;

5. Изучить алгоритм обучения с подкреплением;
6. Внедрить алгоритм обучения с подкреплением в имитационную модель системы массового обслуживания с управлением.

**Методологические основы** имитационного моделирования и, в частности, дискретно-событийного имитационного моделирования представлены в работах С. С. Журавлева [1], Ю. И Рыживов, Б. В Соколов, Р. М Юсупов [2], В. Кельтон, А. Лоу [3].

**Практическая значимость бакалаврской работы**. В ходе выполнения выпускной квалификационной работы была разработана система дискретно-событийного имитационного моделирования, с помощью которой можно создавать имитационные модели систем и сетей массового обслуживания и исследовать их .

**Структура и объем работы**. Бакалаврская работа состоит из введения, 4 разделов, заключения, списка использованных источников и цифрового носителя в качестве приложения. Общий объем работы — 54 страницы, из них 52 страницы — основное содержание, включая 11 рисунков и 1 таблицу, список использованных источников информации — 26 наименований.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

**Первый раздел «Основные понятия имитационного моделирования»** посвящен описанию основных понятий имитационного моделирования и рассмотрению механизмов дискретно-событийного моделирования.

В подразделе 1.1 приведено описание дискретно-событийного имитационного моделирования, описание основных компонентов дискретно-событийной модели, приведен механизм продвижения времени в дискретно-событийных моделях.

Подраздел 1.2 посвящен методу обратного преобразования для генерирования случайных величин .

В подразделе 1.3 приводится обзор существующих систем и сред имитационного моделирования.

**Второй раздел «Архитектура разработанной системы дискретно-событийного моделирования»** посвящен описанию архитектуры разработанной системы дискретно-событийного моделирования. Также описаны

как основные компоненты модуля для дискретно-событийного имитационного моделирования, так и компоненты модуля для построения моделей систем и сетей из теории массового обслуживания (*queueing system*).

В подразделе 2.1 описывается общий архитектурный подход к разработке фреймворка. Рассматривается архитектура, управляемая событиями (*event-driven architecture, EDA*).

Подраздел 2.2 посвящен перечислению и краткому описанию всех модулей разработанного фреймворка, представленных на рис. 1.

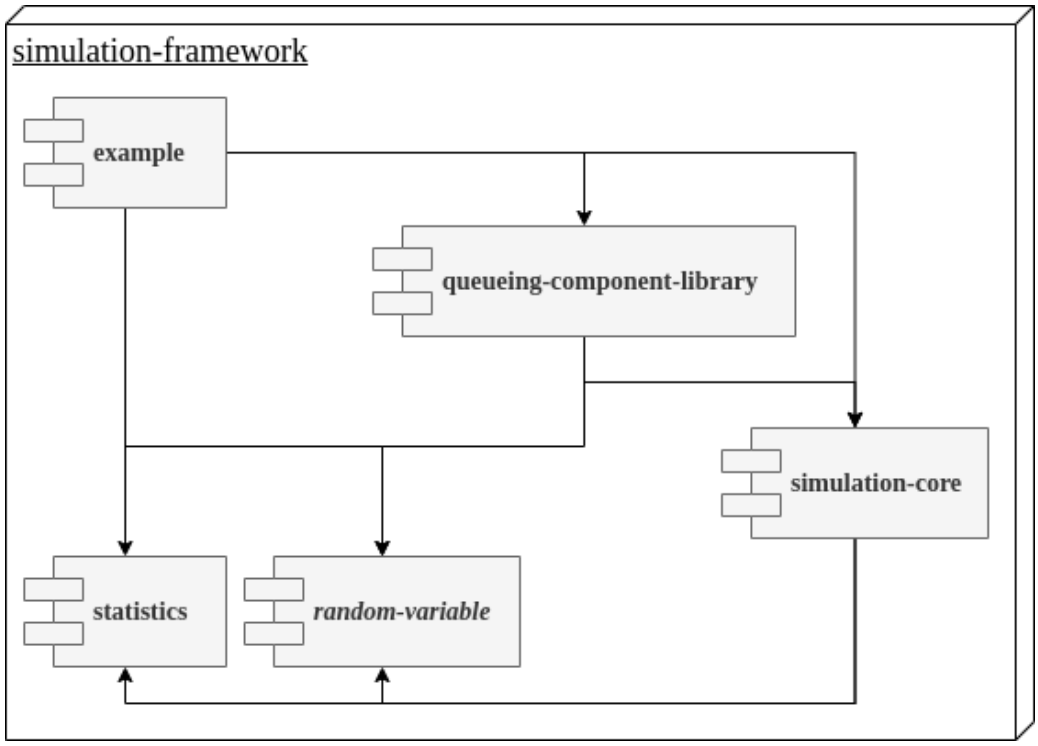


Рисунок 1 – Модули системы имитационного моделирования

— *simualtion-core* – модуль, содержащий основные компоненты для дискретно-событийной модели (события, провайдеры событий, обработчики, часы модельного времени и т.д.). Данный модуль является главным в разработанной системы, с помощью компонентов и объектов, реализованных в нем, можно создавать простые модели для моделирования;
— *queueing-component-library* – модуль, содержащий необходимый набор компонентов для создания имитационных моделей процессов обслуживания;
— *random-variable* – модуль, содержащий датчики псевдослучаных чисел с различными распределениями, основной задачей данного модуля является генерация случайных чисел для моделирования различных по-

токов;

— *statistic* – модуль, предназначенный для сбора и расчета статистических данных;

— *examples* – примеры имитационных моделей, реализованных с помощью компонентов системы.

Подраздел 2.3 посвящен описанию основного цикла имитационного моделирования и класса основном модуле *simualtion-core* разработанного фреймворка, в котором реализована работа данного цикла. Кратко, данный цикл описывается следующими шагами:

1. Из хранилища событий извлекается ближайшее по времени наступления событие.
2. Часы модельного времени обновляются на время активации данного события.
3. Происходит наступление и обработка данного события.
4. Выполняется действие из пункта 1.

И данный цикл будет продолжаться до тех пор, пока не будет выполнено установленное ранее условие останова моделирования. Этим условием может быть, например, достижение ранее заданных единиц модельного времени или отсутствие в хранилище событий для извлечения.

Базовая реализация цикла моделирования описана в абстрактном классе *AbstractSimulationModel* методами *run* и *step*:

```
@Override
public void run() {
    while (!this.stopCondition.test(simulationContext)) {
        step();
    }
}
```

```
@Override
public void step() {
    Event event = simulationContext.getEventProvider().getNext();
    simulationContext.getClock().setCurrentTime(event.getActivateTime());
    simulationContext.updateDeltaTimeLastTwoEvents();
    event.activate();

    LOGGER.debug("STEP: The current time: {} \n", simulationContext.getCurrentTime());
}
```

В подразделе 2.4 описывается реализация событий (через интерфейс *Event* и абстрактный класс *AbstractEvent*). Описывается реализация обработ-

чика событий, которые можно добавлять к событию для реализации дополнительной логики событий (логирование, сбор статистических данных и т.д.). Также описывается хранилище для событий *EventProvider* и *EventProviderImpl*, в которых описывается логика хранения и доступа к событиям.

В подразделе 2.5 описывается объекты-агенты, которые могут инкапсулировать в себе связанные между собой события. В агенте описывается логика событий посредством написания функций. В разработанной системе с помощью данных агентов реализуется возможность построения моделей на основе агентного моделирования [7]. Для этой цели служат такие классы и интерфейсы, как *Agent*, *AbstractAgent*, *AgentAction* и *AgentBasedSimulationModel*. Расписана работа с агентами и приводится пример создания агента-источника (*Source*), объекта из теории массового обслуживания. Источник служит для создания и отправки требований в систему для обслуживания, т.е. целью источника является генерировать поток поступающих в систему требований. Опишем источник с его логикой в виде агента системы как примера.

```
public class SourceAgent extends AbstractAgent implements Source {
    private final RandomVariable interArrivalTimes; private Receiver receiver;

    public SourceAgent(SimulationContext simulationContext, RandomVariable
        interArrivalTimes, String sourceName) {
        super(sourceName);
        this.context = simulationContext;
        this.interArrivalTimes = interArrivalTimes;
    }

    @Override
    public void sendDemand() {
        Demand demand = new SimpleDemand(context.getCurrentTime());
        receiver.receive(demand);

        performActionAfterTimeout(this::sendDemand, interArrivalTimes.nextValue());
    }
}
```

Логика (поведение) источника описывается в методе *sendDemand*. Все что в данном методе, это создание нового объекта требования (*demand*), отправка его определенному получателю (*receiver*) и планирование следующего повторения описанных действий (*performActionAfterTimeout*).

Подраздел 2.6 содержит описание остальных компонентов модуля *simualtion-core*, таких как, *SimulationContext*, *Clock*, *Environment*, *SimulationComponent*,

*Parcel*, которые также служат для корректной работы имитационного моделирования.

В подразделе 2.7 описывается реализация разработанного модуля *queueing-component-library*, который содержит в себе компоненты, необходимые для построения моделей теории массового обслуживания. Основная его цель – облегчить жизнь исследователям в сфере массового обслуживания и упростить реализацию систем и сетей обслуживания разной сложности.

Подраздел 2.8 содержит описание реализаций вспомогательных модулей фреймворка.

— Модуль *random-variable* – модуль содержит классы, реализующие генераторы случайных величин с разными законами распределения. Модуль является одним из ключевых, т.к. генерирование случайных величин является важным аспектом как в имитационном моделировании (генерируются значения для продвижения часов имитационного моделирования), так и в теории массового обслуживания (для имитации разных потоков).

  Для реализации генераторов используется интерфейс *RandomVariable* с единственным методом *nextValue*, который будет возвращать случайное значение, распределенное по некоторому закону.

  Для генерации псевдослучайных чисел используется метод обратного преобразования.

— Модуль *statistics* – модуль содержит классы *StatisticCalculation* и *StatisticAccumulator* для сбора данных и расчета статистических характеристик имитационной модели.

Подраздел 2.9 содержится информация о документации к разработанному фреймворку дискретно-событийного моделирования.

**Третий раздел «Машинное обучение в задачах оптимального управления системами массового обслуживания»** посвящен описанию машинного обучения в целом и, в частности, метода *Q-learning* обучения с подкреплением для оптимального управления системой массового обслуживания. В данном разделе кратко описывается понятие машинного обучения и описываются основные методы машинного обучения.

В подразделе 3.1 описывается метод обучения с подкреплением (*reinforcement learning*) [8], в ходе которого испытуемая система (*агент*) обу-

чается, взаимодействуя с некоторой *средой*. Откликом среды на принятые решения являются сигналы подкрепления, поэтому такое обучение является частным случаем обучения с учителем, но учителем является среда или её модель.

В подразделе 3.2 представлен один из алгоритмов обучения с подкреплением $Q$-обучение (*Q-learning*)

Подраздел 3.3 содержит описание проблемы медленного прибора [9] (задача управления системой массового обслуживания $M/M/K/N$ с $K$ неоднородными обслуживающими приборами и очередью вместимостью $N \leq \infty$, в которую поступает простейший входящий поток требований интенсивности $\lambda$.) из области массового обслуживания. Содержится реализация данной системы обслуживания в разработанном фреймворке. Проводится внедрение алгоритма $Q$-обучения для нахождения оптимального управления данной системой обслуживания. Приводятся результаты оптимизации алгоритмом $Q$-обучения.

**Четвертый раздел «Примеры использования системы дискретно-событийного моделирования»** посвящен примерам работы с разработанным фреймворком для построения и моделирования разных систем.

Подраздел 4.1 содержит пример построения простой имитационной модели таймера, основанной на описании события и его обработки.

В подразделе 4.2 описана реализация имитационной модели системы обслуживания типа M/M/1 двумя способами: через описание простых событий, которые могут происходить в данной системе (поступление требования, начало обслуживания требования и завершение обслуживания требования), и через описание агентов, которые инкапсулируют в себе логику основных компонентов данной системы обслуживания (источник требований и обслуживающая система).

## ЗАКЛЮЧЕНИЕ

В результате выполнения данной выпускной квалификационной работы разработан комплекс дискретно-событийного моделирования на языке программирования общего назначения Java.

Разработанный комплекс представляет собой набор модулей, который может быть использован, например, для создания имитационных моделей

систем и сетей массового обслуживания. Поддерживается сбор и обработка необходимой статистики. Важным достоинством фреймворка является возможность его расширения и легкость создания имитационных моделей. В работе приведены архитектура и основы работы фреймворка, а также некоторые примеры реализации моделей.

**Отдельные части бакалаврской работы были представлены на конференции:**

1. Заварзин А. С., Осипов О. А. Разработка фреймворка дискретно-событийного моделирования процессов массового обслуживания. – Математическое и программное обеспечение информационных, технических и экономических систем (МПОИТЭС-2020): Материалы Международной научной конференции. Томск, 28-30 мая 2020 г. Томск: Изд-во ТГУ, 2020, 265-270.

**Основные источники информации:**


1. Журавлев, С. С. Краткий обзор методов и средств имитационного моделирования производственных систем. //Журнал: Проблемы информатики //Рубрика: Имитационное моделирование технических систем и технологических процессов, 2009. С. 47 – 53.
2. Рыживов Ю. И, Соколов Б. В, Юсупов Р. М. Проблемы теории и практики имитационного моделирования //Сб. докл. III Всерос. науч.-прак. конф. “Имитационное моделирование. Теория и прак-тика” (ИММОД-2007). Санкт-Петербург, 17-19 окт. 2007. Т. 1. С. 58 – 70.
3. Кельтон В., Лоу А. Имитационное моделирование. // Классика CS. 3-е изд. СПб.: Питер: Киев: Издательская группа BHV, 2004. 847 с.
4. Эккель Б. Философия Java. //4-е изд. СПб.: Питер , 2019. – 1168 с.
5. Блох Д. Java. Эффективное программирование. //3-е изд. Вильямс, 2018. – 464 с.
6. Мартин Р. Чистый код: создание, анализ и рефакторинг. //Библиотека программиста. – СПб.: Питер, 2010. – 464 с.
7. Акопов А. С., Хачатрян Н. К. Агентное моделирование: учебно-методическое пособие . //Москва: ЦЭМИ РАН, 2016.
8. Liu B., Xie Q., Modiano E. Reinforcement Learning for Optimal Control of Queueing Systems // 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2019, pp. 663-670.

9. Рыков В. В., Ефросинин Д. В. К проблеме медленного прибора. //Автомат. и телемех., 2009, выпуск 12, 81–91 с.