МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ
имени И.Т. ТРУБИЛИНА»

Кафедра информационных систем и технологий

ТЕЗИС

ИНСТРУМЕНТЫ И ИНФРАСТРУКТУРА ПРОГРАММНОЙ РЕАЛИЗАЦИИ ТРЕХЦВЕТНОЙ КЛЕТОЧНО-АВТОМАТНОЙ ПРОГНОЗНОЙ МОДЕЛИ

ПО КУРСОВОЙ РАБОТЕ

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ
ТРЁХЦВЕТНОЙ КЛЕТОЧНО-АВТОМАТНОЙ
ПРОГНОЗНОЙ МОДЕЛИ

ВЫПОЛНИЛ:
магистрант направления подготовки
09.04.03 Прикладная информатика
профиль «Менеджмент проектов
в области информационных систем»
Макаров Юрий Юрьевич

РУКОВОДИТЕЛЬ:
доктор экономических наук, кандидат физико-математических наук
Попова Елена Витальевна

Краснодар 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Разработка программной реализации трехцветной клеточно-автоматной прогнозной модели требует построения полноценной вычислительной инфраструктуры, обеспечивающей:

- загрузку и хранение временных рядов произвольной длины
- предварительную обработку данных
- реализацию алгоритма клеточного автомата
- расчет метрик качества прогнозирования
- визуализацию результатов
- формирование сравнительных таблиц
- создание автономного исполняемого файла (exe)

В рамках курсовой работы была реализована программная система, ориентированная на прогнозирование временных рядов котировок акций компаний, полученных как из CSV-файлов, так и через API Московской биржи.

1 ВЫБОР ТЕХНОЛОГИЧЕСКОГО СТЕКА

1.1 Язык программирования C# и платформа .NET

Для реализации модели использована платформа .NET 8 и язык программирования C#, что обусловлено следующими факторами:

- высокая производительность
- удобство работы с числовыми данными
- развитые средства создания десктопных приложений
- возможность сборки standalone-exe
- удобная интеграция с CSV и HTTP-API

C# является строго типизированным языком, что позволяет минимизировать ошибки на этапе компиляции и повысить устойчивость программного кода. Для задачи моделирования нелинейной динамики экономических процессов это особенно важно, поскольку алгоритм включает работу с числовыми данными, вероятностями переходов, массивами состояний и вычислением метрик качества прогноза. Статическая типизация упрощает контроль корректности вычислений и делает код более прозрачным.

Платформа .NET 8 была выбрана как современная и поддерживаемая версия среды выполнения. Она обеспечивает высокую производительность, удобные инструменты работы с коллекциями данных, поддержку асинхронного программирования и встроенные механизмы публикации автономных приложений.

В рамках курсовой работы это позволяет сформировать из UI слоя исполняемый файл формата exe, который может запускаться на целевой системе без необходимости дополнительной установки зависимостей, а также сформировать файлы библиотек dll для слоя работы с данными и математическими расчетами.

Для запуска программы необходим SDK - набор инструментов разработки либо использование self-contained сборки, которая формирует папку проекта с SDK внутри, что делает разработанное приложение полностью автономным.

Графический интерфейс реализован с использованием WPF (Windows Presentation Foundation), что позволило:

- организовать ввод параметров модели
- реализовать визуализацию временных рядов
- обеспечить экспорт результатов
- реализовать загрузку данных из различных источников

Несмотря на то, что WPF работает исключительно в среде Windows, данный выбор является оправданным, поскольку целью работы является создание настольного инструмента анализа временных рядов, а не кроссплатформенного сервиса. WPF позволяет реализовать удобный пользовательский интерфейс с возможностью загрузки данных, задания параметров модели, визуализации исходного и прогнозируемого временного ряда, а также вывода сравнительных таблиц.

Для описания внешнего вида окон интерфейса приложения используется технология XAML, что обеспечивает декларативное описание интерфейса, а механизм привязки данных позволяет автоматически обновлять элементы интерфейса при изменении вычислительных результатов.

В основе организации интерфейса фреймворка WPF используется паттерн MVVM (Model–View–ViewModel). Данный шаблон проектирования позволяет отделить визуальную часть приложения от вычислительной логики. Каждый логический слой выполняют свою задачу:

- Представление(View) отвечает исключительно за отображение данных.
- Связка(ViewModel) управляет сценарием работы и связывает интерфейс с алгоритмом
- Модель(Model) содержит структуру данных и математическую реализацию клеточного автомата.

Такое разделение делает программу более тестируемой и поддерживаемой. Логика прогнозирования может быть протестирована отдельно от графического интерфейса, что повышает качество реализации.

Для обработки действий пользователя применяется паттерн Command. Вместо прямых обработчиков событий кнопок используются команды, что позволяет централизовать управление выполнением операций.

2 Архитектура программной системы

Для построения приложения выбрана слоистую архитектура, которая позволяет легче поддерживать код, расширять приложение и тестировать.

Идея слоистой архитектуры в том, что приложение делится на части по ответственности. Это называется принцип разделение ответсвенности(Separation of Concerns).

Один слой отвечает за интерфейс и взаимодействие с пользователем, другой — за работу с файлами и сетью, третий — за сам алгоритм клеточного автомата и расчёты. При таком подходе каждый слой знает только то, что ему нужно, и не лезет в детали других.

2.1 Слой данных (Data Layer)

Выполняемые задачи:

- загрузка временных рядов из CSV
- загрузка данных через MOEX ISS API(API московской биржи)
- хранение котировок
- проверка корректности входных данных
- обеспечение работы с временными рядами большой длины

Основные структуры данных:

- CsvImportedData - ранит уже распарсенный временной ряд
- TimeSeries - хранит упорядоченную последовательность наблюдений во времени:
- ClosePrices[] - это массив цен закрытия

2.2 Слой алгоритма (Core Layer)

Выполняемые задачи:

- реализация трехцветного клеточного автомата
- механизм формирования состояний
- расчет вероятностных переходов
- реализация Лапласского сглаживания
- расчет глубины памяти
- расчет прогноза
- расчёт метрик качества (MAE, RMSE, MAPE, Accuracy)

2.3 Слой представления (UI Layer)

Позволяет:

- ввод параметров модели
- запуск вычислений
- отображение исходного временного ряда
- отображение прогноза
- вывод сравнительной таблицы по 5 компаниям
- экспорт результатов в CSV

3 Инфраструктура загрузки и хранения данных

3.1 CSV-загрузка

Реализована возможность загрузки временных рядов любой длины:

1. Чтение файла
2. Парсинг дат и цен закрытия
3. Сортировка по дате
4. Проверка пропусков

3.2 Загрузка через API MOEX

Для получения исторических котировок используется HTTP-запрос к сервису ISS Московской биржи.

Параметры HTTP-запроса:

- from=YYYY-MM-DD — начальная дата периода
- till=YYYY-MM-DD — конечная дата периода
- iss.meta=off — отключение мета-информации
- history.columns=TRADEDATE,CLOSE — выбор только необходимых столбцов

Реализовано:

- автоматическая подгрузка всех страниц (history.cursor)
- объединение данных
- преобразование в внутренний формат

3.3 Организация хранения больших временных рядов

Для обеспечения работы с максимально возможной длиной ряда:

- используется массив double[]
- вычисления проводятся без хранения избыточных копий
- глубина памяти задается параметром L

- используется скользящее окно

Временная сложность алгоритма O(N)

ЗАКЛЮЧЕНИЕ

В результате первого этапа работы была спроектирована и реализована программная инфраструктура для решения задачи прогнозирования временных рядов на основе трехцветной клеточно-автоматной модели.

Выбор языка C#, платформы .NET 8 и технологии WPF позволил создать автономное настольное приложение с удобным пользовательским интерфейсом и возможностью формирования исполняемого файла.

Применение слоистой архитектуры обеспечило разделение ответственности между загрузкой данных, вычислительным ядром и представлением результатов, что повысило структурированность, масштабируемость и устойчивость системы.