|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт комплексной безопасности и специального приборостроения | | |
| Кафедра КБ-3 «Управление и моделирование систем» | | |

|  |  |
| --- | --- |
| **РАБОТА ДОПУЩЕНА К ЗАЩИТЕ** | |
| Заведующий кафедрой КБ-3 |  |
|  | *Подпись* |
|  | |
| *ФИО* | |
| «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| по направлению подготовки бакалавра | | 09.03.04 |  |  |
|  | | *Код направления подготовки* |  | *Наименование* |
| Программная инженерия | | | | |
| *направления подготовки* | | | | |
| на тему: | Разработка программно-математических средств использования теории | | | |
|  | «управляемого хаоса» в управлении устойчивостью сложных динамических | | | |
| систем в условиях гибридной войны | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Осмоловский Матвей Игоревич\_\_\_\_\_\_\_\_ | |
|  | *Подпись* | *Фамилия Имя Отчество* | |
| Шифр | \_15Б0304\_\_\_о |  |  |
| Группа | БКБО-01-15 о |  |  |
|  |  |  |  |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Заведующий кафедры КБ-8 о | Григорьев Виталий\_\_\_\_\_\_\_\_\_ Робертович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | *Подпись* | *Ученая степень, ученое звание, должность* | *ФИО* |
|  | |  |  |
| Консультант  (*при наличии*) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | *Подпись* | *Ученая степень, ученое звание, должность* | *ФИО* |

Москва 2019 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт комплексной безопасности и специального приборостроения  *наименование института* *(полностью)* |
| Кафедра КБ-3 «Управление и моделирование систем»  *наименование кафедры (полностью)* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СОГЛАСОВАНО | | |  | УТВЕРЖДАЮ | | |
| Заведующий  кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись* | | |  | Директор  института \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись* | | |
| *Фамилия Имя Отчество* | | |  | *Фамилия Имя Отчество* | | |
| «\_\_\_\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 20\_\_\_\_\_\_ г. |  | «\_\_\_\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 20\_\_\_\_\_\_ г. |

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы бакалавра

|  |  |  |
| --- | --- | --- |
| Обучающийся | Осмоловский Матвей Игоревич | |
|  | *Фамилия Имя Отчество* | |
| Шифр | 15Б0304 | |
| Направление подготовки | 09.03.04 | Программная инженерия |
|  | *индекс направления* | *наименование направления* |
| Группа | БКБО-01-15 | |

1. Тема выпускной квалификационной работы

Разработка программно-математических средств использования теории "управляемого хаоса" в управлении устойчивостью сложных динамических систем в условиях гибридной войны\_\_\_\_\_\_\_\_

2. Цель и задачи выпускной квалификационной работы

Цель работы:

* Разработать программно-математическое средство для использования теории управляемого хаоса в управлении устойчивостью сложных динамических систем (СДС)

Задачи работы:

* Обзор подходов к моделированию СДС.
* Выбор структуры, параметров и построение алгоритма моделирования СДС и проверки её устойчивости.
* Разработка программно-математических средств для использования теории управляемого хаоса в управлении устойчивостью СДС.
* Подготовка документации.

3. Этапы выпускной квалификационной работы

|  |  |  |  |
| --- | --- | --- | --- |
| №  этапа | Содержание этапа выпускной квалификационной работы | Результат выполнения этапа ВКР | Срок выполнения |
| 1 | Описание теории управляемого хаоса области. | Исследовательский раздел рукописи п.1.1 |  |
| 2 | Определение СДС как как объектов управления | Исследовательский раздел рукописи п.1.2 |  |
| 3 | Обзор теоретико-графового подхода моделирования устойчивости СДС | Исследовательский раздел рукописи п.1.3 |  |
| 4 | Обзор синергетического подхода моделирования устойчивости СДС | Исследовательский раздел рукописи п.1.4 |  |
| 5 | Обзор принципов построение сложных динамических (социальных) систем на основе использования теории клеточных автоматов | Исследовательский раздел рукописи п.1.5 |  |
| 6 | Обзор системы моделирования WinALT | Исследовательский раздел рукописи п.1.6 |  |
| 7 | Характеристики устойчивости СДС | Специальный раздел рукописи п.2.1 |  |
| 8 | Управление устойчивостью СДС с помощью локальных однородных воздействий | Специальный раздел рукописи п.2.2 |  |
| 9 | Описание особенностей функционирования социальной системы | Специальный раздел рукописи п.2.3 |  |
| 10 | Жизнедеятельность описываемой системы | Специальный раздел рукописи п.2.4 |  |
| 11 | Модель устойчивости сложной динамической социальной системы | Специальный раздел рукописи п.2.5 |  |
| 12 | Выбор технического решения | Технологический раздел рукописи п.3.1 |  |
| 13 | Алгоритмическая поддержка решения | Технологический раздел рукописи п.3.2 |  |
| 14 | Описание основных функций программного обеспечения | Технологический раздел рукописи п.3.3 |  |
| 15 | Тестирование программного обеспечения | Технологический раздел рукописи п.3.4 |  |
| 16 | Использование разработанного программного обеспечения для расчёта модели сложной динамической социальной системы | Технологический раздел рукописи п.3.5 |  |
| 17 | Анализ экономической эффективности | Экономический раздел рукописи п.4.1, п.4.2 |  |
| 18 | Разработка технической документации. | Раздел рукописи техническая документация п.5.1, п.5.2, п.5.3 |  |

4. Перечень разрабатываемых документов и графических материалов

1. Пояснительная записка к ВКР.
2. Слайды презентации по теме ВКР.

5. Руководитель выпускной квалификационной работы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функциональные обязанности | Должность в Университете | Фамилия Имя Отчество | | Подпись | |
| Руководитель ВКР | Заведующий кафедры КБ-8 | Григорьев В.Р. | |  | |
| Задание выдал | | | Задание принял к исполнению | |
| Руководитель ВКР:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | Обучающийся:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| *Подпис*ь | | | *подпис*ь | |
| «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. | | | «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. | |

АННОТАЦИЯ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
| Студент | | Осмоловский М.И. |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
| Руководит. | | Григорьев В.Р. |  |  |  |  |  |  |  |
| Консульт. | |  |  |  | РТУ МИРЭА | | | | |
| Н. Контр. | | Никольский А.Н. |  |  |
| и.о. зав. Каф. | | Пушкин П.Ю. |  |  |

Целью данной бакалаврской работы является разработка программно-математического средства для использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем (СДС).

Задачами данной выпускной квалификационной работы являются:

* Обзор подходов к моделированию СДС;
* Выбор структуры, параметров и построение алгоритма моделирования СДС и проверки её устойчивости;
* Разработка программно-математического средства для использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем;
* Подготовка документации.

Структура выпускной квалификационной работы состоит из нескольких разделов:

* В исследовательском разделе описана предметная область, а так же подходы к моделированию устойчивости сложных динамических систем;
* В специальном разделе рассмотрены характеристики устойчивости СДС, а так же разработана модель устойчивости сложной динамической социальной системы на основе клеточных автоматов;
* В технологическом разделе проведено обоснование выбора инструментальных средств, описаны алгоритмы и структуры программного обеспечения, а также произведено тестирование программного обеспечения и его применение для моделирования и анализа разработанной в специальном разделе модели сложной динамической социальной системы;
* В экономическом разделе определена продолжительность и трудоёмкость разработки, построен ленточный график, произведён расчёт общей сметы, приведён плановый график получения денежных средств;
* В разделе техническая документация приведены: руководство пользователя, руководство программиста, руководство системного администратора;
* В заключении описаны выводы по результатам выполненной работы.

# Содержание

[Введение………………………………………………………………………….7](#_Toc11045493)

[1. Исследовательский раздел………………………………………………….11](#_Toc11045494)

[1.1 Описание теории управляемого хаоса 11](#_Toc11045495)

[1.2 Определение сложных динамических систем как объектов управления 13](#_Toc11045496)

[1.3 Теоретико-графовый подход. Модель структурного разрушения сложных систем при когерентном воздействии 16](#_Toc11045497)

[1.4 Синергетический подход. Использование системы дифференциальных уравнений для описания развития (социальных) систем 18](#_Toc11045498)

[1.5 Принципы построения сложных динамических социальных систем на основе использования теории клеточных автоматов 25](#_Toc11045499)

[1.6 Обзоры системы моделирования WinALT 28](#_Toc11045500)

[Выводы 30](#_Toc11045501)

[2. Специальный раздел………………………………………………………..32](#_Toc11045502)

[2.1 Характеристики устойчивости сложной динамической системы 32](#_Toc11045503)

[2.2 Управление устойчивостью сложной динамической системы с помощью локальных однородных воздействий 35](#_Toc11045504)

[2.3 Описание особенностей функционирования социальной системы 39](#_Toc11045505)

[2.4 Жизнедеятельность описываемой системы 40](#_Toc11045506)

[2.5 Модель устойчивости сложной динамической социальной системы на основе использования теории клеточных автоматов 44](#_Toc11045507)

[Выводы 47](#_Toc11045508)

[3. Технологический раздел……………………………………………………49](#_Toc11045509)

[3.1 Выбор технического решения 49](#_Toc11045510)

[3.2 Алгоритмическая поддержка решения 50](#_Toc11045511)

[3.3 Описание основных функций программного обеспечения 52](#_Toc11045512)

[3.4 Тестирование программного обеспечения 60](#_Toc11045513)

[3.5 Использование разработанного программного обеспечения для расчёта модели сложной динамической социальной системы 63](#_Toc11045514)

[Выводы 68](#_Toc11045515)

[4. Экономический раздел 69](#_Toc11045516)

[4.1 Расчёт сметы затрат на проведение исследования. Построение сетевого графика с расчётом по параметрам 69](#_Toc11045517)

[4.2 Расчёт экономической эффективности 77](#_Toc11045518)

[Выводы 86](#_Toc11045519)

[5. Техническая документация…………………………………………………88](#_Toc11045520)

[5.1. Руководство пользователя 88](#_Toc11045521)

[5.2. Руководство программиста 89](#_Toc11045522)

[5.3. Руководство системного администратора 90](#_Toc11045523)

[Выводы 91](#_Toc11045524)

[Заключение……………………………………………………………………..92](#_Toc11045525)

[Список использованных источников…………………………………………94](#_Toc11045526)

Приложение А. Перечень графического материала…………………………97

Приложение Б. Листинг программы…………………………………………136

# Введение

Сегодня в эпоху тотальной глобализации, ослабления и размытия традиционного понимания государственных границ, стремительного развития средств массовой коммуникации важнейшим фактором стало изменение форм разрешения межгосударственных противоречий. В современных конфликтах все чаще акцент используемых методов борьбы смещается в сторону комплексного применения политических, экономических, информационных и других невоенных мер, реализуемых с опорой на военную силу. Это так называемые гибридные методы ведения противоборства.

Их содержание заключается в достижении политических целей с минимальным вооружённым воздействием на противника, преимущественно за счёт подрыва его военного и экономического потенциала, оказания порой очень агрессивного информационно-психологического давления, активной поддержки извне внутренней оппозиции, партизанских, а по сути, террористических и диверсионных методов. В качестве главного средства достижения цели перехвата управления в стране-жертве используются так называемые «цветные революции», которые, по мнению инициирующих их сторон, должны привести к ненасильственной смене власти в стане оппонента. По сути любая «цветная революция» – это государственный переворот, организованный извне. А в основе этих технологий перехвата власти лежат современные информационно-телекоммуникационные технологии, предусматривающие возможность организации управляемой манипуляции протестным потенциалом населения, подогреваемым извне, в сочетании с другими невоенными ненасильственными средствами.

С точки зрения формализованной науки, такой процесс управления устойчивостью сложной динамической системой (государства) носит бифуркационный характер и может быть описан в терминах и функциях нелинейной динамики, в обиходе получившей название «управляемый хаос».

Математическое моделирование диссипативных систем, оперирующих вдали от состояния термодинамического равновесия открытых систем в условиях проявления нелинейного порядка – детерминированных хаосо- подобных процессов, в настоящее время, имеет важнейшее значение и для моделирования сложных химических, биологических, социологических явлений. Как примеры подобных явлений, можно привести множественные химические реакции, процессы распространения болезней, пожаров, информационные системы, различные социальные процессы. Эти процессы получили название «процессов с обострением», которые достаточно глубоко исследованы в трудах российских и зарубежных учёных. Их моделирование позволяет исследовать особенности их функционирования в различных условиях, наделять их требуемыми характеристиками и снижать возможность возникновения чрезвычайных ситуаций (ЧС).

Традиционным подходом при этом считается моделирование посредством построения систем дифференциальных уравнений, в общем случае, нелинейных. Такие уравнения не имеют аналитических решений, а численные методы требуют больших затрат вычислительных ресурсов.

Вследствие столкновения с подобными проблемами в не теряющей своей актуальности области социологических исследований поведения социальных систем в условиях кризисов, возникает необходимость разработки новых подходов к моделированию. В частности, такими подходами являются модели дискретного и стохастического типа, в которых различные преобразования и пространственные характеристики моделируются непосредственным, прямым образом.

В рамках данной выпускной работы, подробнее будет рассматриваться моделирование социальных процессов, к которым можно отнести электоральные процессы, процессы распространения новостей, идей, в том числе, и деструктивных. Задачи их анализа и прогнозирования являются крайне сложными, так как связаны с прогнозом достижения долгосрочных целей при постоянной необходимости адаптации к изменяющимся условиям внешней среды. Они требуют учёта большого числа факторов, в их решениях присутствует высокая степень неопределённости оценок различных параметров, а так же слабая формализация методов управления и многокритериальность при выборе принимаемых решений.

Общество является хорошим примером сложной социальной системы, в которой случайные процессы самоорганизации, складывающиеся из частных интересов отдельных людей, приводят к устойчивым изменениям всей системы в целом.

Описанием подобных процессов занимается междисциплинарная наука - синергетика. В рамках этой науки происходит изучение процессов самоорганизации в сложных системах, способов управления ими, а так же рассмотрение структурных переходов этих систем от одного состояния к другому.

В противовес к распространённому мнению о статичной природе общества, синергетика утверждает, что оно является неуравновешенной, динамической системой, в которой порядок и хаос взаимодействуют друг с другом, а развитие системы является сложным следствием чередования периодов эволюции с революциями, представляющими из себя бифуркации.

Возможности изменения направления развития и качественных составляющих системы отражены в *теории управляемого хаоса*.

Наряду с теорией управляемого хаоса в последнее время в науке активно развивается теория сложных сетей. Она находит применение во многих сферах человеческой деятельности: от информационных технологий до систем управления социальными сетями. В рамках некоторых направлений были разработаны принципиальные подходы к построению моделей сложных сетей, позволяющие анализировать их поведение в условиях закономерного развития, а также направленных воздействий извне.

Среди таких подходов наибольший интерес представляют теория случайных графов (А.-Л. Барабаши, С. Дороговцев и др.[1-3]) и синергетический подход (Г. Хакен, И. Пригожин и др. [4-6]). Несмотря на внешние различия, эти подходы имеют много общих черт, сходных явлений и процессов.

В настоящей работе проблема описания процессов поведения и развития сложных систем и управления ими в условиях целенаправленных внешних воздействий рассматривается с точки зрения теории самоорганизации. По представленным в математической модели исследуемой системы основным элементам и их качественным характеристикам и особенностям поведения можно будет судить о устойчивости системы к деструктивным воздействиям.

В работе ставится задача анализа и формализации поведения развивающихся сложных сетей в условиях различного рода внешних воздействий. Важно, что когерентные (синхронизированные) воздействия могут быть оказаны как на отдельные элементы сети, так и на сеть в целом, а также на связи между элементами и уровнями в иерархической системе представления СДС.

Объектом исследования дипломной работы является социальная система, а предметом исследования – процесс сохранения устойчивости СДС в условиях влияния внешних деструктивных воздействий на поведение системы.

Целью работы является создание и реализация алгоритма моделирования устойчивости сложной динамической системы в условиях когерентных информационных атак с использованием клеточных автоматов.

Основными задачами при выполнении дипломной работы являются:

* анализ угроз уязвимости сложных динамических систем в условиях когерентных внешних и внутренних воздействий;
* анализ математических моделей распространения по системе поражающих воздействий;
* разработка дискретной модели устойчивости сложной динамической системы на основе использования теории клеточных автоматов.

# 1. Исследовательский раздел

***Открытой*** считается система, которая способна обмениваться веществом, энергией иинформациейс окружающей средой. Таким образом, открытой будет являться любая реальная сложная динамическая система. Находясь вне равновесного состояния, такие системы оказываются неустойчивыми, вследствие чего в точке, называемой ***точкой бифуркации***, система становится непредсказуемой.

Как следствие этой неустойчивости, порой даже несравнимо малое воздействие извне способно повлечь за собой непредсказуемые последствия.

В находящихся далеко от равновесного состояния открытых системах может возникать согласование между элементами на микроскопическом уровне. Последствием такого согласования являются процессы упорядочивания, приводящие к появлению из хаоса новых структур, их преобразованию и усложнению.

В случае систем, для которых отклонения от равновесия играет большую роль, описанное согласование может возникать между отдалёнными и несвязанными, на первый взгляд, областями, их процессами. Нелинейность и наличие обратных связей характеризуют эти процессы, что открывает возможности оказания управляющих воздействий на систему.

## 1.1 Описание теории управляемого хаоса

Возникновение полицентрического мира и связанное с этим ужесточение международной конкуренции, глобализация социальной и экономической сфер жизни – всё это привело к такому состоянию мира, которое более невозможно описать, используя классические подходы, такие как механицистская модель. В поисках выхода из сложившегося положения, учёные-политологи пришли к идее применения математического аппарата описания поведения нелинейных систем – теории хаоса – по отношению к описанию процессов мировой политики. Ведь современный мир, с его бесчисленным множеством действующих элементов, переплетающихся друг с другом в бесчисленном множестве связей, приводящих всю систему целиком к бесчисленному множеству различных состояний, и есть не что иное, как сложная, сложнейшая динамическая система. В таком случае, теория хаоса, как ни что другое подходит к задаче описания этой системы и моделирования её поведения.

Согласно теории хаоса, в сложных системах существует чрезвычайная зависимость от начальных условий, вследствие чего возникает так называемый «эффект бабочки». «Эффектом бабочки» называется свойство хаотических систем, при котором незначительные возмущения (внутренние или внешние), влияющие на некоторый участок системы, могут вылиться в непредсказуемые, обладающие несопоставимым с силой первоначального воздействия масштабом, последствия для системы. Такие последствия могут проявиться на совершенно другом участке системы некоторое время спустя.

Применительно к социоэкономическим системам, теория управляемого хаоса предполагает введение данной системы в хаотическое состояние и использование этого для достижения масштабных качественных изменений системы путём уже достаточных для этого несильных воздействий.

Впервые теория управляемого хаоса была определена дипломатом и советологом Стивеном Манном в 1992 году на проходившей в институте междисциплинарных исследований в Санта-Фе конференции. В своём докладе «Теория хаоса и стратегическая мысль», Манн определил средства создания управляемого хаоса в сферах национальной экономики и социума. Будучи доведёнными до крайне неустойчивого, хаотичного состояния, эти сферы делали всю собственную страну жертвой последующих манипуляций.

Согласно Манну, создание хаоса производится путём вызова недоверия народа к властям, создания паники среди жителей, поощрения протестов.

Таким образом, с позиций «управляемого хаоса» могут быть объяснены нестабильности экономик мировых держав, а так же происходящие в различных углах планеты «цветные» революции.

Так как данная работа не ставит перед собой задачу решения вопросов мировой геополитики, её продуктом будет модель поведения упрощённой социальной системы в условиях внешних деструктивных воздействий и оценка устойчивости этой системы с таким воздействиям.

Далее будет рассмотрено определение понятия сложной динамической системы, а так же описаны подходы к управлению этими системами.

## 1.2 Определение сложных динамических систем как объектов управления

### 1.2.1 Модель сложной динамической системы.

Современные информационные системы (ИС) представляют собой системы, состоящие из большого числа относительно простых элементов. Например, это могут быть персональные компьютеры, образующие ЛВС, процессоры, образующие кластер, персональные странички пользователей некоторого ресурса, образующие социальную сеть. Между этими элементами происходят различного рода взаимодействия, или, другими словами, между ними происходит обмен информацией. С другой стороны, многие из этих частей не следуют никаким «общим правилам», то есть ведут себя хаотично. Тем не менее, в таких системах (то есть в системах, состоящих из большого количества простых элементов, между которыми осуществляется взаимодействие) проявляются процессы самоорганизации, то есть появление в системе свойств и поведения не присущих каждому элементу системы в отдельности. Примеров таких систем множество, в самых разных областях от физики до биологии, от химии до социологии. В последние полвека список таких систем пополнили и вычислительные системы. В свете последних тенденций интеграции социальной и информационной сферы общества, появляется всё больше и больше таких сложных динамических систем, которые характеризуются помимо всего прочего повышенным обменом информацией как между элементами системы, так и между системой и внешней средой.

### 1.2.2 Понятие сложной динамической системы

Перед тем как приступать к рассмотрению подходов к управлению СДС, было бы логично дать определение понятию сложно динамической системы.

Введём следующие определения:

*Определение 1*. Система – это совокупность (множество) объектов и процессов, называемых элементами, взаимосвязанных и взаимодействующих между собой для осуществления какой-либо цели; они образуют единое целое, обладающее свойствами, не присущими составляющим его элементам, взятым в отдельности [7].

Таким образом, система обладает составом, т.е. совокупностью элементов, множеством связей между элементами, структурой и целью.

*Определение 2*. Под динамической системой понимается система, изменяющая своё состояние во времени.

Ввиду отсутствия чёткой границы между простыми и сложными системами, существует множество подходов к оценке их сложности.

Далее приводятся несколько качественно разных определений некоторых авторов. В [8] предлагается подход, согласно которому сложность систем зависит от числа элементов, входящих в систему:

* Малые системы );
* Сложные );
* Ультрасложные );
* Суперсистемы );

Другие авторы, например [9], сложной называет систему, которая «строится для решения многоцелевой, многоаспектной задачи и отражает объект с разных сторон в нескольких моделях». А позднее в книге Ст. Бира «Мозг фирмы» [10] предлагается соотносить сложность системы с числом её возможных состояний: для простых систем - до состояний, сложных от до состояний, и очень сложных имеющих свыше миллиона состояний.

Применительно к задаче, исследуемой в дипломной работы, наиболее подходящим понятием сложной системы является синтез двух походов: элементного [11] и определение сложности через количество состояний [12].

Это связано с тем, что количество элементов в рассматриваемой системе находятся в пределах до [8]. Поэтому в рамках данной работы принято следующее определение сложной системы:

*Определение 3*. Под сложной системой понимается система, состоящая из взаимодействующих элементов и реализующая в процессе своего функционирования порядка от до состояний системы.

### 1.2.3 Подходы к управлению сложными динамическими системами

Рассмотрим вопрос управления такими системами. Под управлением мы будем понимать возможность задать системе соответствующую траекторию развития этой системы. Теория управления в настоящий момент является хорошо развитой и сформировавшейся теорией, однако большинство результатов в этой теории сформулировано для линейных систем. Для нелинейных систем классически применяются методы линеаризации уравнений. Однако эти методы не позволяют описывать поведение сложных динамических систем (СДС) в достаточной мере, в силу того, что эквивалентность исходной нелинейной системы и её линейного приближения сохраняется лишь для ограниченных пространственных или временных масштабов системы, либо для определённых процессов. Причём, если система переходит с одного режима работы на другой, то следует изменить и её линеаризованную модель.

Второй недостаток заключается в том, что СДС характеризуются не только нелинейным поведением, но и большими размерностями и сильной взаимосвязью между параметрами системы. Таким образом, при увеличении числа учитываемых параметров законы управления при данном подходе уже не могут быть получены или это представляется практически невозможным.

Другим подходом к исследованию сложных нелинейных систем является метод аналитического конструирования оптимальных регуляторов (АКОР). Этот метод начал развиваться в 60-х годах ХХ века силами Р. Калмана и А. М. Летова. Целью АКОР является создание такого закона управления, который бы обеспечивал минимум критерия качества на траекториях, описывающих движение объекта от начального состояния к конечному. К сожалению, распространить этот подход на нелинейные системы там и не удалось.

Так как рассматриваются сложные системы в которых протекают процессы самоорганизации, возможно хаотическое поведение, то логично привлечь для построения моделей управления такими системами ту науку, которая такие системы изучает, то есть синергетику. Применение синергетических методов и понятий в управлении СДС позволило сформировать раздел в теории управления, названый синергетической теорией управления (СТУ). Данный подход позволяет аналитически синтезировать асимптотически устойчивые законы управления для нелинейных систем.

Наконец, прежде чем перейти к описанию подходов к построению модели устойчивой СДС, определим само понятие устойчивости сложных динамических систем. По аналогии с представлением понятия устойчивости для биосферы [13], дадим следующее определение: «Устойчивость СДС и её подсистем – это способность поддерживать своё функционирование и возможность реализовать дальнейшие эволюционные процессы, восстанавливаться после умышленных и не умышленных нарушений штатного состояния его работоспособности и негативных внешних и внутренних воздействий».

## 1.3 Теоретико-графовый подход. Модель структурного разрушения сложных систем при когерентном воздействии

Использование теоретико-графовой модели при описании структурных разрушений очевидно в силу того, что графовые операции (стягивание ребра, удаление и добавление вершин и рёбер) удобно описывают происходящие в структуре системы изменения.

Для случая, когда изменения структуры постоянны, разумно ввести понятие, которое бы описывало временное изменение структуры системы. Таким понятием служит *структурная динамика* [14].

Структурные изменения можно разделить на две категории. Первая – положительные изменения – появление новых элементов, способствующих улучшению функционирования системы. Вторая группа – негативные изменения – угнетение (или даже остановка) системы в силу выхода элементов из строя.

Ряд моделей и задач использует потоки в сетях для моделирования всевозможных транспортных процессов (как пример такого процесса – транспортировка газа по сети трубопроводов) [15]. К сожалению, эти модели и задачи не учитывают сценарии выхода из строя всей системы в силу отказа одного из элементов и последующей каскадной реакции выхода из строя смежных элементов.

Как правило, структурное разрушение может продолжаться до отказа системы. В случае разных систем, отказ может наступать в силу различных причин, поэтому разумно использовать различные *критерии отказа*.

Первый, очевидный критерий - критерий полного разрушения, обозначим как . Он зависит от единственного параметра k числа удалённых вершин, и считается наступившим при удалении всех вершин графа.

Общим случаем критерия полного разрушения можно назвать *компонентный критерий*, обозначающий выход системы из строя в случае, когда число удалённых элементов превышает заданное пороговое значение. Обозначается как , и зависит от ранее названного параметра k и заданного порогового значения .

Частным же случаем критерия будет являться *диаметральный критерий* – система вышла из строя в случае, если в процессе разрушения системы диаметр одного из компонентов соответствующего графа станет меньше порогового значения .

Случай нарушения связности структуры графа отражён критерием связности, обозначающимся как и зависящим от ранее названного параметра k.

Систему можно считать вышедшей из строя только в случае соответствия структурных изменений системы критериям отказа.

Исходя из теоретико-графовой модели, система является **абсолютно стойкой** к классу воздействий, если она не переходит в предельное состояние вследствие процесса распространения воздействия по системе.

Руководствуясь данной модель, возможно найти способы наделения системы стойкостью к негативным внешним воздействиям необходимого уровня, и наоборот – смоделировать поведение системы при заданных воздействиях.

Далее будет показан синергетический подход при формализации динамических социальных систем: равновесные состояния и равновесные процессы социальных систем, устойчивость социальных систем.

## 1.4 Синергетический подход. Использование системы дифференциальных уравнений для описания развития (социальных) систем

С практической точки зрения особую актуальность представляет моделирование поведения процессов в социальных системах (сетях). Для решения этих задач используется математический аппарат теории динамических систем. Речь, в первую очередь, идёт о представлении сложной динамических систем с помощью системы дифференциальных уравнений (в общем случае, нелинейных).

Этот подход позволяет описать динамику изменения параметров состояния рассматриваемой системы, бифуркационные процессы, фазовые переходы.

При анализе конкретной динамической системы идеальный вариант – найти и описать все её траектории в явном виде. Для реальных сложных систем это бывает невозможно сделать (главным образом, ввиду их масштабов и сложной структуры связей). В связи с этим представляется возможным выделить комплекс вопросов об общем поведении системы, на которые (по крайней мере, часть из них) можно дать ответ на практике:

1. Что можно сказать о наличии у системы возможности вернуться к некоторому предыдущему состоянию?
2. Как устроен аттрактор (аттракторы) системы (подмножество фазового пространства, к которому стремится большинство траекторий)?
3. Остаются близкими или расходятся траектории, берущие начало в близких точках?
4. Как описывается поведение «усреднённой» динамической системы класса?
5. Насколько похоже поведение «близких» к данной систем?

Пусть некоторая сложная (социальная) система, задаётся системой дифференциальных уравнений

(1)

где – вектор зависимых переменных, характеризующих состояние сложной (социальной) системы, – вектор параметров системы (в общем случае, зависящих от времени), – вектор-функция (в общем случае, нелинейная), отражающая изменение этих переменных во времени.

Тогда решение системы (1) – набор траекторий в фазовом пространстве – удобно представлять в виде следующей схемы (рис. 1.1), позволяющей сделать ряд качественных выводов о поведении такой системы.

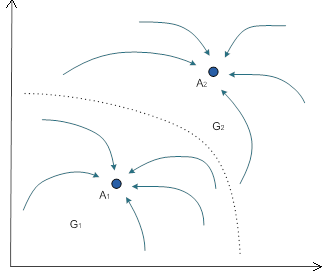


Рисунок 1.1 – Набор траекторий СДС в фазовом пространстве с аттракторами и их областями притяжения

Если система находится в точке фа­зового пространства, принадлежащей об­ластям притяжения, то с течением времени она окажется в точке-аттракторе. Подобный анализ фазо­вых траекторий позволяет сделать заключение о характере развития системы, определить облас­ти её детерминированного поведения и в некоторых случаях предсказать возможное развитие.

Для социальных систем, задаваемых системой уравнений вида (1), точки бифуркации – это определённые интервалы параметров векторов , при ко­торых возникает неустойчивость и происходит изменение числа и/или вида решений системы (1). Посредством уменьшения (увеличения) значений пара­метров вектора можно влиять на изменение поведения сис­темы, вводя систему в неустойчивое состояние. Выявление па­раметров порядка в модели (1), а также их критических значений (при которых проис­ходят бифуркации состояний) помогает выби­рать оптимальные способы управления реальны­ми социальными системами.

Для социальных систем возможны два основных пути развития: ***эволюционный*** и ***революционный (кризисный)***.

Процесс эволюционного развития социальных систем (рис. 1.2) является адаптацией системы к внешним условиям посредством вариации способов функционирования. Большинство из этих вариаций имеют случайный характер и отбраковываются средой.



Рисунок 1.2 – Эволюционное развитие социальных систем

В моменты повышенной чувствительности системы, периодически наступающие в процессе развития, особенно опасны преднамеренные внешние воздействия, так как обладая даже малой интенсивностью, они способны повлечь за собой значительные последствия, вплоть до изменения направления развития системы. Такое развитие называется развитием через кризисы и схематически отражено на рисунке 1.3.



Рисунок 1.3 – Развитие социальных систем через кризисы

(1,2,3 – уровни иерархии (подсистемы))

По мере своего развития, системы приобретают дополнительные уровни иерархии, что повышает эффективность использования имеющихся ресурсов. Но ровно как внешняя среда оказывает влияние на систему, справедливо и обратное. Поэтому в некоторый момент изменения внешней среды могут сделать текущую структуру системы нежизнеспособной, приводя к её кризису, последующим дестабилизации и распаду.

В силу своей узкоспециализированности, более всех страдают верхние слои иерархии.

Далее возможны два исхода. Первый – стабилизация системы в силу успешной реформации верхних слоёв.

Второй – распад системы, выражающийся в разрушении специализированных слоёв, за чем следует новый виток самоорганизации, руководствующийся «генетической» памятью о прошлом опыте. Это наиболее радикальный сценарий развития, за которым следуют принципиальные изменения в системе.

С точки зрения практики наиболее важно искать возможности оказания возможных воздействий на (социальные) системы, не приводящие ко второму типу исходов развития. Эффективно работая на эволюционном уровне, можно управлять функционированием целевой системы без необходимости участия в явных кризисах.

Для построения моделей подобных воздействий необходимо исследовать основные виды бифуркационных переходов к хаосу.

### 1.4.1 Использование теории бифуркаций в определении устойчивости сложных динамических систем

Большую важность с точки зрения управления сложными системами представляет прогнозирование их поведения. Некоторое время назад господствовала точка зрения, что если случайными воздействиями на объект можно пренебречь и последний может быть описан динамической системой вида

то система полностью предсказуема. Синергетика показала, что это не так.

Действительно, на практике, как правило, известно состояние исследуемого объекта с некоторой точностью. Таким образом, приходится иметь дело не с идеальной траекторией и начальным условием , а с решением уравнения , где . Предсказуемость системы определяется разницей между реальной траекторией и той, которую можно прогнозировать – . Оказалось, что в типичных случаях эта разница зависит от времени по экспоненциальному закону:

(2)

Величина *,* называемая ***показателем Ляпунова***, является важнейшей характеристикой исследуемой системы. Если , то близкие траектории стремятся друг к другу, малые погрешности несущественны и поведение системы хорошо предсказывается. В таком случае, исследуемую систему можно назвать **устойчивой**. Если , то близкие траектории расходятся, и через время , называемое ***горизонтом прогноза***, теряется информация о системе и нужно вновь проводить измерения. В таком случае, система является неустойчивой. Поведение системы при разных значениях схематически показано на рисунке 1.4.



Рисунок 1.4 – Поведение системы при разных значениях . Система неустойчива, если , и устойчива, если

В синергетике это явление получило название ***чувствительности к начальным данным*** или ***эффекта бабочки***. Незначительное влияние на систему может иметь большие и непредсказуемые эффекты где-нибудь в другом месте и в другое время. Это связано с экспоненциальным характером нарастания ошибок и неопределённостей в (2).

Оказывается, что при математическом моделировании объектов, связанных с возможностью кризисных ситуаций, есть области, в которых горизонт прогноза резко сокращается. Схема представления сложной динамики системы, в которой возможны катастрофические процессы, изображена на рисунке 1.5.

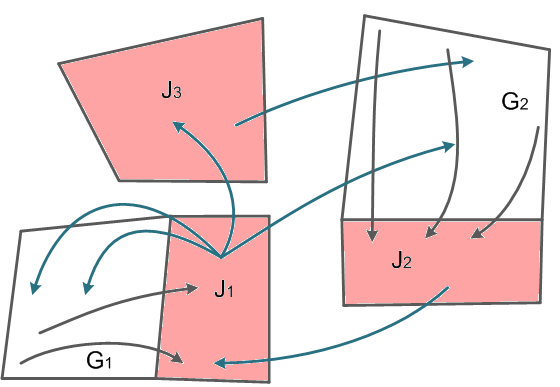


Рисунок 1.5. Схема представления сложной динамики системы, в которой возможны катастрофические процессы.и – русла, , и - джокеры

Черные стрелки показывают характерные траектории для медленной динамики. Зелёные стрелки показывают действие джокеров, когда траектория попадает в область джокера. В этом случае она может с определённой вероятностью направиться в некоторую точку русла или к другому джокеру.

Области джокеров обычно соответствуют различным катастрофам или чрезвычайным ситуациям в исследуемой системе. Для описания таких объектов в синергетике был введён новый класс математических моделей – **динамические системы с джокерами**. Сам джокер – это правило, в соответствии с которым объект в фазовом пространстве может совершить скачок. Это соответствует тому, что в таких системах есть два масштаба времени – «быстрый», на котором мы обычно не успеваем что-либо предпринять и «медленный», на котором в ряде случаев можно предотвратить аварию либо подготовится к ликвидации её последствий.

Наряду с областями джокеров в фазовом пространстве существуют русла, где ход процессов предопределён, и сложные объекты можно описывать достаточно просто.

Для решения практических задач наиболее эффективно поддерживать функционирование целевой системы в рамках «русел», а все воздействия оказывать таким образом, чтобы изменения развития происходили без попадания в области «джокеров».

Синергетический подход с использованием системы дифференциальных уравнений при построении сложных динамических систем (социальных) изучаются уже давно. Исследования в области локальных однородных взаимодействий [16] доказывают возможность использования клеточных автоматов в моделировании экономических, социальных, экологических процессов.

Далее рассматриваются основные понятия аппарата клеточных автоматов, и принципы построения моделей СДС на его основе.

## **1.5 Принципы построения сложных динамических социальных систем на основе использования теории клеточных автоматов**

Клеточные автоматы (КА) дискретные детерминированные системы, поведение которых полностью определяется в терминах локальных взаимодействий [17].

Клеточные автоматы являются полезными дискретными моделями для исследования динамических систем [17]. С возможностью дискретизации модели связанно главное достоинство клеточных автоматов - их абсолютная совместимость с алгоритмическими методами решения задач. Ограниченное множество элементов, в купе с конечным набором формальных правил, приводит к возможности точной алгоритмической реализации. Обратной стороной этого преимущества являются трудности в машинных расчётах, вызванные большими масштабами одновременно обрабатываемых клеток.

Клеточные автоматы были использованы Джоном фон Нейманом для исследования самовоспроизведения [18]. Однако они применяются и для совершенно других целей. Если универсальной моделью для последовательных вычислений считается машина Тьюринга, то клеточные автоматы являются такой моделью для параллельных вычислений [19].

Соединение вычислительной компоненты с данными выделяет клеточные автоматы на фоне прочих средств моделирования.

Стоит отметить, что несмотря на наличие клеточных автоматов прочих измерений, наибольшую практическую ценность представляют двумерные клеточные автоматы.

### 1.5.1 Определение клеточного автомата

Клеточный автомат (КА) представляет собой однородную «решётку» конечных автоматов.

В качестве ячейки двумерного клеточного автомата способна выступать любая фигура, обладающая свойством замощения плоскости, но наиболее распространённой версией является ячейка-квадрат.

На каждом последующем шаге состояние каждой ячейки-конечного автомата определяется через состояния его и его соседей, что является отличительной особенностью клеточных автоматов.

«Соседство» ячеек вытекает из понятия окрестности клетки, что является одним из свойств КА. Свойство это постоянно всюду на решётке.

В случае двумерной решётки с ячейками-квадратами, существует 2 вида окрестностей: окрестной фон Неймана и окрестность Мура.

Окрестность фон Неймана определяет соседями некоторой клетки клетки с индексами . Отметим, что клетка входит в свою собственную окрестность.

Окрестность Мура же, помимо клеток из окрестности фон Неймана, так же добавляет в качестве соседей клетки, касающиеся текущей углами, т.е. имеющие индексы .

Так же существует расширенная окрестность Мура, включающая в себя так же и те клетки, что касаются клеток-соседей оригинальной окрестности.

Описанные окрестности изображены на рисунке 1.6.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Расширенная окрестность *Мура*

Окрестность Мура

Окрестность фон Неймана

Рисунок 1.6 – Описанные окрестности двумерных клеточных автоматов

Отметим другие свойства классической модели клеточных   
автоматов [20]:

• Локальность правил. Последующее состояние клетки определяется самой клеткой и её соседями;

• Однородность системы. Кроме особых, краевых случаев (возникающих в силу ограниченности поля КА), все области решётки идентичны в контексте правил и особенностей. Краевые же случаи, как правило, подвергаются отдельной обработке для избежания данной неравномерности;

• Конечность множества возможных состояний. Это условие гарантирует конечность числа операций в ходе преобразований клеток к новым состояниям.

• Одновременность смены значений ячеек. Данное свойство позволяет избежать искажений результата итерации, вызванных порядком перебора клеток.

Отдельно стоит отметить необязательность последних трёх свойств. В качестве примера можно привести работу [21], в которой решётка КА является неоднородной.

Далее в работе будут рассматриваться только двумерные клеточные автоматы на тетрагональной решётке, а в качестве окрестности будет использоваться окрестность Мура (рис. 1.7).

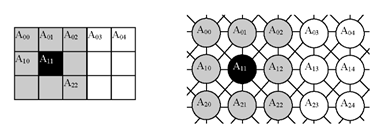


Рисунок 1.7 – Тетрагональная решётка с окрестностью Мура

## 1.6 Обзоры системы моделирования WinALT

Существующим решением, способным наиболее адекватно справиться с целью работы является система имитационного моделирования алгоритмов с мелкозернистым параллелизмом WinALT [25].

Система базируется на **алгоритме параллельных подстановок**.

**Алгоритм параллельных подстановок (АПП)** – это способ представления вычислений над распределёнными в пространстве данными.

АПП отображает “истинный параллелизм” вычислений, когда на каждом шаге выполняются все допустимые действия над всеми имеющимися данными.

Можно выделить следующие преимущества системы WinALT:

* свободное распространение;
* дружественность и адекватность пользовательского интерфейса;
* собственный язык моделирования, заточенный под нужды целевой области;
* расширяемая модульная архитектура системы.

Архитектура системы WinALT отражена на рисунке 1.8.

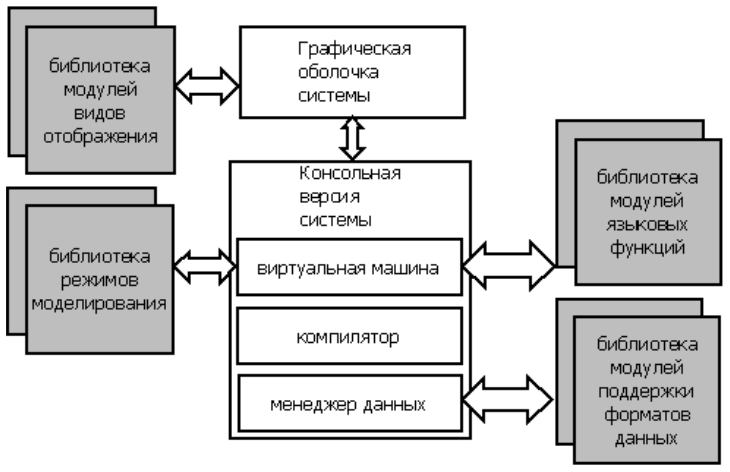


Рисунок 1.8 – Архитектура системы WinALT

Опишем каждую из компонент подробнее.

*Консольная версия системы* – ядро системы WinALT. Ядро состоит из менеджера данных, организующий модули системы и работу с данными, компилятора языка системы и виртуальной машины. Язык реализации консольной версии – C, что обеспечивает мультиплатформенность системы.

*Графическая оболочка системы* – надстройка над консольной версией системы,визуализирующая работу пользователя с моделью.

*Библиотека модулей поддержки форматов данных* обеспечивает поддержку различных форматов представления моделируемых объектов.

*Библиотека модулей языковых функций* обеспечивает поддержку сторонних модулей обработки данных моделей.

*Библиотека модулей видов отображения* обеспечивает разнообразие способов визуализации данных.

*Библиотека режимов моделирования* отвечает за настройку параметров моделирования (таких, например, как синхронность обработки данных).

Обратной стороной наличия внутреннего языка системы является необходимость для пользователя обладать определённым уровнем подготовки, прежде чем начинать работу с системой.

Ещё одним недостатком системы является факт того, что она не обновлялась в течение длительного времени и поддерживается лишь устаревшей на данный момент операционной системой Windows XP.

Поэтому, несмотря на развитость и мощность комплекса моделирования WinALT, было принято решение создать собственную систему для моделирования разрабатываемой в данной работе модели устойчивости СДС.

## Выводы

1. Было дано описание теории управляемого хаоса – подхода к управлению социоэкономическими системами посредством введения их в хаотическое состояние.
2. Введено определение для сложной динамической системы, под которой будет пониматься система, состоящая из взаимодействующих элементов и реализующая в процессе своего функционирования порядка от до состояний системы.
3. Для решения задач анализа устойчивости сложных динамических систем в условиях внешних и внутренних воздействий были рассмотрены два подхода: теоретико-графовый и синергетический.
4. Представлены возможные реализации синергетического подхода к моделированию сложных динамических (социальных) систем.
5. Рассмотрены и проанализированы основные характеристики клеточных автоматов. Указано огромное преимущество моделей на основе КА в виде их совместимости с алгоритмическими методами решения задач.
6. На основе изложенного анализа возможностей реализации СДС сделан вывод о целесообразности использования аппарата клеточных автоматов для моделирования предметной области исследований в выпускной работе. Дискретный характер работы КА, их способность к описанию динамики развития сложных процессов позволяют адекватно описать и смоделировать процессы в сложных динамически изменяемых социотехнических системах, что будет представлено в следующей главе.
7. Так же сделан вывод целесообразности разработки собственного средства моделирования КА.

# 2. Специальный раздел

## 2.1 Характеристики устойчивости сложной динамической системы

Моделирование сложных динамических систем ведёт к довольно серьёзному вопросу – нужно ли брать во внимание поведение каждого из элементов, образующих всю систему, или достаточно рассмотреть некоторые из них? Ведь данная структура может состоять из огромного количества элементов. С точки зрения синергетики и самоорганизации, нам достаточно рассмотреть основные элементы СДС, по поведению которых и можно будет судить о поведении всей системы. Такой процесс исследования, когда не рассматривается детально каждая составляющая СДС, называется *системным синтезом.*

С точки зрения безопасности, у всякой СДС выделяют такие её характеристики, как живучесть, надёжность и стойкость.

Сначала поговорим о надёжности.

Свойство сложной динамической системы в промежутке времени оставаться неизменной, то есть неизменными остаются её параметры, при которых СДС все ещё работает, называется надёжностью [3]. Для того, чтобы определить это понятие, чаще всего используют аппарат математической статистики, называемый деревом событий.

Дерево событий представляет собой диаграмму, отображающую все отказы, которые ведут, появляясь последовательно или параллельно, к критической ситуации или даже чрезвычайному положению [3]. Очевидно, что принимая во внимание совокупность элементов, их отключение может вести к потенциальному прекращению функционирования всей системы.

Чтобы лучше понять, что собой представляет дерево отказов, предлагается рассмотреть следующий пример. Сигнал, поступающий на вершину v1 графа G должен дойти до вершины v3. Обозначим за S – событие, при котором сигнал не достигнет своего пункта назначения, выйдя из начальной вершины. События непрохождения сигнала до вершин vi обозначим, соответственно, Si.

На рисунке 2.1 представлен граф дерева отказов, пунктиром изображён резервный путь и соответствующее ему событие.

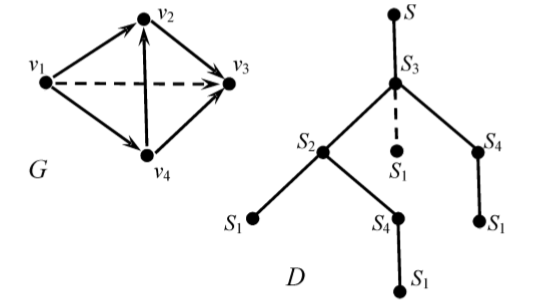


Рисунок 2.1 – Граф дерева отказов

Для примера обозначим вероятность бесперебойной работы в конкретный промежуток времени 0.9. В итоге получаем, что надёжность СДС в виде графа G без учёта резервного пути равна *Р*()=1-*Р*(*S*3*S*4*S*1)*Р*(*S*3*S*2*S*1)*Р*(*S*3*S*2*S*4*S*1)=1-(1-0.93)(1-0.93)(1-0.94)≈0.975, а с учетом резервного соединения – *Р*()≈0.995.

Событие – событие, противоположное событию S, то есть правильное функционирование системы.

Свойство СДС, которое отвечает за способность работать под действиями нагрузок, проходящими в систему из внешней среды, называется живучестью [3].

Исследования данного понятия так же возможно с помощью аппарата математической вероятности. Модель, которая описывает живучесть СДС, называется “нагрузка - прочность” [3]. С течением времени при постоянных внешних импульсах система ослабевает, её прочность снижается и, в конечном счёте, функционирование системы прекращается.

Существует такое понятие, как механика катастроф. Системы, которые принимают на себя перманентные внешние влияния, являются её объектом исследования. С течением времени повреждения накапливаются, но система продолжает функционировать. Все это продолжается до определённого момента, когда СДС переходит в критическое состояние. Это в совокупности с реакцией элементов системы на деструктивные воздействия является предметом исследования механики катастроф. Отдельный интерес вызывает процесс перехода СДС из критического состояния в закритическое, ведь элементы системы при таком развитии событий могут вести себя совершенно непредсказуемым образом. Чаще всего, рано или поздно, они выходят из строя, тем самым ставя под удар целостность всей системы, в состав которой они входят. Это тот случай, когда внешние воздействия ведут к внутренним структурным изменениям и несут в себе негативные последствия, подталкивая систему к переходу в чрезвычайное положение.

Живучесть систем широко используется при анализе сложных коммуникационных систем управления.

Подводя итог разговору про надёжность и живучесть, следует отметить, что они являются довольно важными характеристиками при оценке риска возникновения положения, при котором система будет находится в критическом состоянии, ведь в таком случае они близка к прекращению своего функционирования. Изучая эти параметры, можно планировать и обеспечивать безопасность систем, предупреждая возникновения в них чрезвычайных положений.

Качество системы, которое определяется способностью системы выдерживать деструктивные воздействия, продолжая работать в штатном докритическом режиме, называется стойкостью данной системы [3]. Стойкость напрямую связана с понятием живучести, ведь, судя по определению, она представляет собой живучесть системы под влиянием внешних нагрузок до своего перехода в критическое состояние. Поэтому основная характеристика этого параметра – время, которое требуется системе для перехода в предельное состояние. Чем больше этот период времени, тем меньше шанс возникновения чрезвычайного положения.

Чтобы система правильно работала, правильно взаимодействовать должны все её элементы. Деструктивные влияния, влияющие на показатели надёжности одних элементов, автоматически будут влиять на качественные показатели других составных частей, вызывая при этом лавинную реакцию во всей системе. То есть нагрузка будет на пределе ещё и у тех элементов, на чью структуру воздействие извне никак не повлияло, они будут как бы пассивно заражаться этим вирусом извне. Главная же задача – спроектировать такую систему, при которой мощные внешние нагрузки, адресованные на отдельные элементы, не будут коренным образом влиять на состояние соседних элементов. Важно, чтобы даже при импульсном воздействии, пусть и очень большой мощности, система продолжала быть работоспособной и сохраняла свою функцию.

Не стоит забывать, что сложных динамических систем огромное количество, начиная от технических и заканчивая социальными. Если говорить о военной технике, то они должны быть готовы к пребыванию в экстремальных условиях, ведь вышеописанные импульсы в контексте таких структур – привычное дело. Ведь их задача – правильно функционировать в таких жёстких условиях. Если возвращаться к социально-экономическим сложным системам, то время их пребывания в предкритическом положении чрезвычайно малы и, следовательно, их реакция на непродолжительный, но сильный импульс может оказаться губительной для СДС.

## 2.2 Управление устойчивостью сложной динамической системы с помощью локальных однородных воздействий

В настоящее время огромное внимание привлечено к изучению возможности организации управления большими системами (экономическими, социальными, экологическими) с помощью, быть может, самого универсального способа управления посредством локальных однородных взаимодействий.

Определим понятие социального управления. Считалось (и пусть даже считается), что «в обществе сложи­лись два типа механизма управления — стихийный и сознательный. При стихийном механизме... управляющее воздействие на систему является усреднённым результатом... различных... сил, массы слу­чайных единичных актов; это воздействие автоматично по своей при­роде и не требует вмешательства людей» [22]. Как следствие, управ­ление обществом связывалось лишь с "сознательным механизмом"; при этом влияние "стихийной" составляющей управления как бы иг­норировалось (с чем, конечно, согласиться уже нельзя).

Образно говоря, получалось, что сознательно управляемое об­щество это слон (могучей поступью идущий в светлое будущее), а стихийная компонента управления это какая-то ползающая по сло­ну муха (на которую и внимания-то обращать не за чем). А что, если все наоборот?

И слон (бредущий во времени) это общество, управляемое "стихийным механизмом", а "сознательный механизм" управления в виде мухи (пусть даже и грамотной) ползает по этому слону в поисках эффективного управления им. Очевидно, такой мухе необходимо, по крайней мере, хорошенько изучить слона тем более что “научное управление предполагает активный процесс познания общественных закономерностей” [22]). Но тогда самое время вспомнить, что одним из наиболее эффективных методов изучения больших систем (и сияния на управление ими) является компьютерное моделирование.

Блестящие успехи компьютерного моделирования, продемонстрированные за последнюю четверть 20 века (системная динамика Дж. Форрестера [23], доклады Римского клуба, сценарий "ядерной зимы"   
Н.Н. Моисеева [24] и др.), создали благоприятную атмосферу для вос­приятия подобных методов изучения глобальных процессов учёными самых разных специальностей. В настоящее время можно надеяться на преодоление барьеров узкодисциплинарных подходов и продук­тивную совместную работу междисциплинарных коллективов учёных по созданию новых моделей больших систем. Одной из базовых кон­цепций управления, на основе которой можно было бы вести подоб­ную работу, является управление посредством локальных однородных взаимодействий.

Речь идёт о возможности достижения глобальных целей управ­ления системой посредством управляющих воздействий, имеющих ис­ключительно локальный характер, т.е. основанных только на локаль­ной информации.

Библейские заповеди могут служить примером подобного управления. Правила взаимодействия каждого человека с окружением являются управляющими воздействиями. Причём состояние общества целиком не оказывает влияния на эти правила. Однородность же правил выражена в их массовости, т.е. в распространении на всех индивидов без исключений.

Эти правила настолько эффективны, что заставляют снова и снова восхищаться ими, в отли­чие от многих других исторических попыток навести порядок в об­ществе.

Вообще, управление посредством локальных однородных взаимодействий отличается тем, что управляющие воздействия имеют непрямой, не директивный характер и представляют собой некоторые заданные правила взаимодействия между элементами управляемой системы. Главными свойствами таких правил взаимодействия явля­ются локальность и однородность. Локальность определяет пове­дение всех элементов системы только как зависящее от состояния сосед­них (т.е. от свойств системы только в окрестности данного элемента) и не зависящее от характеристик и состояния всей системы. Свойство однородности указывает на массовость правил взаимодействия элементов системы.

Называемый часто неиндивидуализированным управлением, этот способ имеет ряд важных преимуществ перед пря­мым директивным управлением. В частности, он гораздо более устойчив к дефектам (ошибкам, отсутствию) информации о состоянии элементов, а также к негативному воздействию внешних возму­щений. И даже более, при управлении посредством ло­кальных однородных взаимодействий само устройство управления системой физически может отсутствовать вовсе, поскольку его "техническую" роль выполняют сами элементы системы, а его ин­формационная функция может быть осуществлена различными спо­собами.

Конкретизируем приведённые рассуждения. Рассмотрим более простые, но строгие примеры.

В конце 60-х годов объектом исследований группы московских математиков под руковод­ством И.И. Пятецкого-Шапиро (с середины 70-х годов эти работы ве­дутся A.M. Леонтовичем, в начале 80-х годов к ним присоединился В.Ф. Огарышев) стало управление в морфогенетических процессах.

В рамках этого направления были построены и исследованы компьютерные модели некоторых формообразовательных процессов биологии развития таких, целью которых является создание и со­хранение той или иной геометрической формы биологического объек­та, состоящего из достаточно большого числа клеток (автоматов).

Цель моделирования заключалась в проверке гипотезы о том, что требуемая (глобальная) форма клеточного сообщества может быть достигнута посредством должным образом определённых пра­вил лишь локального (и однородного) взаимодействия между клетка­ми. На базе этой концепции была построена серия моделей биологи­ческих процессов; наиболее содержательная из них задача сферообразования остается предметом исследований уже более 30 лет.

В задаче сферообразования моделируется процесс формиро­вания биологическими клетками однослойной сферической поверх­ности. Классическим примером такого процесса является образова­ние бластулы этапа развития любого организма (в том числе, че­ловека), когда число клеток возрастает от одной (двух) до тысячи (нескольких тысяч). В модели биологические клетки представляются областями двухмерной сети, расположенной на поверхности, типологически эквивалентной сфере. Создание и сохранение (поддержание) глобальной сферической формы клеточного сообщества происходит за счёт перемещения клеток в результате движений их вершин. При этом правила взаимодействия клеток (движения вершин) локальны (т.е. правила движения каждой вершины зависят только от положения соседних с нею вершин) и однородны (т.е. одинаковы для всех вершин и всех клеток сообщества).

В результате компьютерного моделирования были найдены правила перемещения клеток, решающие задачу сферообразования адекватным (биологическому процессу) образом. Тем самым доказана принципиальная возможность управления посредством только локальных средств целыми морфогенетическими процессами на отдельных этапах развития организма. Таким образом, поэтапное раз­витие организма можно представлять как последовательность сменяющих друг друга локальных правил взаимодействия, однородных для достаточно больших групп клеток.

Потенциальная мощь концепции локальных однородных взаимодействий, которую нам удалось почувствовать, вызывает надежду и даже уверенность в возможности применения такого способа управления к экономическим, социальным, экологическим процессам.

## 2.3 Описание особенностей функционирования социальной системы

Социальная система с точки зрения социолога это нечто, что не может быть описано в нескольких словах. Более того, даже в том случае, когда социолог дает ёмкое и достаточно полное (текстовое) описание социальной системы, то легко, обратившись к другому социологу, убедиться, что существует другое, мало чем похожее на первое описание социальной системы. Отчасти это связано с тем, что социология по сей день сугубо гуманитарная наука и в ней отсутствуют формализованные описания социальных систем.

Под формализацией в данном случае мы понимаем символи­ческое описание, т.е. описание, использующее большое количество специальных символов и знаков (например, математических) без то­го, чтобы за этими символами и знаками были закреплены какие-либо конкретные значения, без того, чтобы они были наполнены каким-либо «житейским» смыслом.

Формализация хороша тем, что она позволяет незамедлитель­но начать исследование, используя формальные правила и законы, ранее установленные другими исследователями для данной фор­мальной системы. Иначе говоря, очень быстро могут быть полу­чены конкретные результаты, на основе которых делаются те или иные заключения и прогнозы.

Может оказаться, что формализация была чрезмерно упрощён­ной и это повлекло скудные заключения и прогнозы. Однако важ­но, что они были получены, т.е. налицо некоторое продвижение, прогресс, и это лучше, чем ничего. Тем более, что усложнение фор­мальной системы, как правило, не проблема. Другой вопрос под­дастся ли анализу усложненная формальная система? Последнее было серьёзным препятствием в эпоху, предшествующую появле­нию электронно-вычислительной техники. В настоящее время на­личие мощных вычислительных систем позволяет получать реше­ния и прогнозы там, где ранее исследователь отступал перед чрез­мерно усложнёнными формализованными (математизированными) описаниями природных и, что важно для нас, социальных систем.

Несмотря на распространённость математического моделирования в естественных науках, недостаточная формализация терминов социологии ограничивает его (моделирования) применение в ней.

Социальная система по сравнению с общим понятием сложной динамической системы, имеет специфику, которая заключается в следующем:

1. Элементами социальной системы являются люди, группы людей, предприятия, организации, различные общности.

2. Под социальными связями понимаются осознанные или неосознанные, необходимые или случайные, устойчивые или спонтанные зависимости между элементами, включая информационные взаимодействия.

3. Под социальной структурой понимается совокупность устойчивых связей и отношений между элементами, составляющими систему, которые определяют её качественное своеобразие и строение, обусловленные отношениями социальных групп, разделением труда, характером социальных институтов (государства и др.).

4. Целью социальной системы является приведение социальной системы к гармоничному, устойчивому и безреволюционному состоянию.

## 2.4 Жизнедеятельность описываемой системы

Опишем рассматриваемую в качестве объекта работы социальную систему.

Процессом, протекающим в системе и вызывающим интерес для исследования, является распространение некоторой условной деструктивной идеи.

Отдельным элементом системы является индивид – человек, обладающий определённым кругом общения, и имеющий ряд характеристик, релевантных по отношению к исследуемому процессу распространения деструктивной идеи.

В зависимости от отношения к деструктивной идее, выделим 4 группы людей:

*Распространители (Spreaders, S)* – навязывающие окружающим деструктивную идею активисты.

*Противники (Opposers, O)* – активисты, борющиеся с деструктивной идеей.

*Заражённые (Infected, I)* – люди, поддавшиеся влиянию Распространителей и принявшие деструктивную идею.

И, наконец, *Незаражённые (Normal, N) –* обычные люди.

Для соблюдения рамок выбранного метода моделирования скажем, что Противники распространяют некоторую идею, обратную деструктивной, а группа Незаражённых её принимает. Тогда становится возможным «исцеление» представителей группы Заражённых.

Каждая из групп воздействует на остальные (информационно на них влияя). Связи между ними представлены на рисунке 2.2.

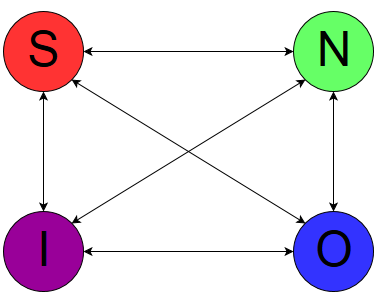


Рисунок 2.2 – Связи между описанными группами людей

С точки зрения распространения двух идей – деструктивной и антидеструктивной – каждая из групп людей характеризуется степенью воздействия на окружающих и степенью невосприятия идей. Дадим этим характеристикам названия:

Destruction\_influence – сила влияния деструктивной идеей;

Anti\_influence – сила влияния идеей, обратной деструктивной;

Destruction\_resistance – уровень устойчивости к деструктивной идее;

Anti\_resistance – уровень устойчивости к идее, обратной деструктивной.

Каждый из этих параметров принимает значение от 0 до 9, где более высокое значение соответствует большей степени воздействия/невосприятия.

Приведём шкалу соответствия значений и их словестных описаний:

0 – отсутствует;

1 – крайне слабый;

2 – очень слабый;

3 –слабый;

4 – ниже среднего;

5 – средний;

6 –выше среднего;

7 – сильный;

8 – очень сильный;

9 – крайне сильный.

В таблице 2.1 дан пример конкретных значений характеристик для каждой из групп.

Таблица 2.1 – Пример значений характеристик для каждой из групп людей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Destruction\_influence | Anti\_influence | Destruction\_resistance | Anti\_resistance |
| S | 7 | 0 | 0 | 9 |
| O | 0 | 6 | 9 | 0 |
| I | 2 | 0 | 0 | 4 |
| N | 0 | 4 | 1 | 0 |

Таким образом, согласно приведённой лингвистической шкале, показанная в примере группа Распространителей деструктивной идеи обладает «сильным» воздействием деструктивной идеей и «крайне сильным» сопротивлением к антидеструктивной идее.

Описываемая система неразрывна с внешней средой, которая и оказывает негативное воздействие в виде навязывания деструктивной идеи активистам. Эти активисты впоследствии распространяют идею на людей, заражая их. Воздействие внешней среды на описываемую систему условно представлено на рисунке 2.3.



Рисунок 2.3 – Негативное воздействие внешней среды на описываемую систему

Жизнедеятельность системы можно представить как переход от начального состояния , когда произошло инициирующее заражение группы Распространителей, вызвавшее появление Противников идеи, до некоторого конечного состояния .

Тогда поведением системы будет набор всех её состояний, начиная от и заканчивая :

Состояние определяется функцией перехода :

В зависимости от группы, к которой принадлежит человек-элемент системы, функция принимает конкретный вид:

,

где – функция перехода i-ой группы людей. Суть функции перехода заключается в том, что если степень воздействия какой-либо идеей на человека превышает порог невосприятия им этой идеи, то он принимает эту идею, изменяя свою групповую принадлежность.

Для определения конечного состояния системы , необходимо сначала определить, что мы будем понимать под устойчивым состоянием системы.

*Определение 4*. Равновесное состояние – состояние системы, в котором она может находиться сколь угодно долго при условии постоянства внешних воздействий (как частный случай, их отсутствия).

*Определение 5*. Устойчивое состояние – равновесное состояние, в которое система возвращается в случае, когда она была выведена из него теми или иными возмущающими воздействиями.

Конечным состоянием назовём состояние системы на i-ом шаге, если оно полностью совпадает с состоянием системы на шаге i+1, т.е.:

(3)

На основании определений 4 и 5, состояние , для которого выполняется (3), назовём устойчивым для рассматриваемого объекта исследования.

## 2.5 Модель устойчивости сложной динамической социальной системы на основе использования теории клеточных автоматов

В модели один человек представляется одной клеткой.

Дадим формальное описание клеточного автомата и опишем каждый из его параметров с точки зрения разрабатываемой модели:

, где

1. поле автомата. В данной модели его размер – клеток.

Начальное распределение людей определяется через величины spreaders\_percentage и opposers\_percentage, где spreaders\_percentage – вероятность клетки стать распространителем деструктивной идеи, opposers\_percentage – вероятность клетки стать распространителем антидеструктивной идеи. Тогда 1-(spreaders\_percentage+ opposers\_percentage) – вероятность клетки стать обычным человеком. Заражённых в начальный момент времени нет.

1. конечное множество возможных состояний клетки. Перечислим возможные состояния:

* 0 – Распространители антидеструктивной идеи (**O**), в графическом представлении модели отмечаются синим цветом;
* 1 – обычные люди (**N**), отмечаются зелёным цветом;
* 2 – заражённые люди (**I**), отмечаются фиолетовым цветом;
* 3 – распространители деструктивной идеи (**S**), отмечаются красным цветом.

Описанные состояния клетки отражены на рисунке 2.4.

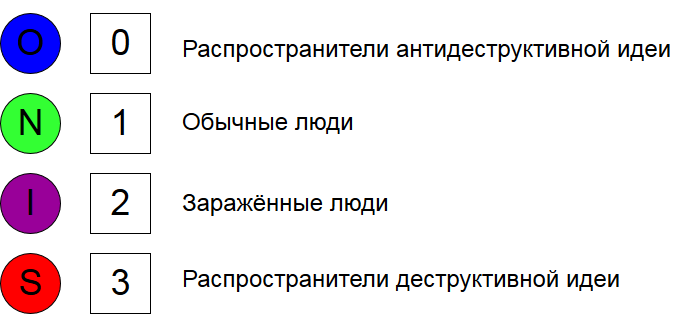


Рисунок 2.4 – Описанные состояния клетки

1. определение окрестности клетки (не путать с обозначением группы Противником деструктивной идеи). В описываемой модели окрестностью клетки является окрестность Мура. Пример соседей клетки **X** при окрестности Мура показан на рисунке 2.5.

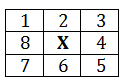


Рисунок 2.5 – Соседи клетки **X** при окрестности Мура

Для избежания краевых эффектов, возникающих на границах решётки КА, при непосредственном расчёте модели, решётка будет топологически сворачиваться, образуя тор [21]. Свёртывание плоскости в тор показано на рисунке 2.6.

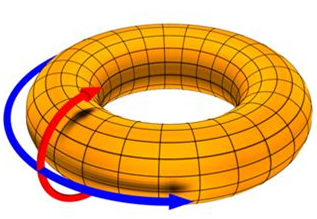


Рисунок 2.6 – Свёртывание плоскости в тор

Непосредственно, это будет означать «закольцованность» решётки по горизонтальной оси и соединение нижней и верхней граней.

Пример свёртывания показан на рисунке 2.7, где изображены соседи ячейки **1** с рисунка 2.5.

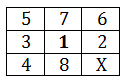


Рисунок 2.7 – Cоседи ячейки **1**

1. функция переходов. Как было сказано в п.2.4, каждой из 4-х групп (см. рис. 2.4) соответствует своя функция перехода:

* Для **O**:

* Для **N**:

* Для **I**:

* Для **S**:

Далее, в разделе 3 будет произведён расчёт построенной модели.

## Выводы

1. Под сложной системой будем понимать систему, состоящую из взаимодействующих элементов и реализующую в процессе своего функционирования порядка от до состояний системы.
2. Устойчивость СДС определяется тремя её свойствами: живучестью, надёжностью и стойкостью.
3. Перспективным способом управления устойчивостью СДС является управление посредством локальных однородных воздействий - управляющих воздействий, имеющих ис­ключительно локальный характер, т.е. основанных только на локаль­ной информации. Такой способ управления имеет ряд важных преимуществ перед пря­мым директивным управлением.
4. Особенностью социальных систем, затрудняющей их исследование и управление, является сложность в формализации описания такой системы.
5. Была описана модель социальной системы, подвергающейся негативному внешнему воздействию в виде деструктивной идеи.

# 3. Технологический раздел

Как уже было сказано в п.1.6, основное существующее средство для моделирования сложных динамических систем на основе аппарата клеточных автоматов не подходит для целей данной работы вследствие своей относительной устарелости и чрезмерной усложнённости в силу излишнего относительно целей работы функционала. Поэтому для моделирования разработанной в п.2.4 и п.2.5 модели сложной динамической социальной системы будет разработано собственное программное решение.

## 3.1 Выбор технического решения

Для разработки программного обеспечения будут использоваться исключительно средства с открытым исходным кодом.

Для написания системы был выбран язык программирования Python (версии 3.6.2) и следующие его библиотеки: Tkinter (версия 8.6), NumPy (версия 1.14.3) и Matplotlib (версия 2.2.2).

Остановимся подробнее на каждом из названных средств.

Python – названный в честь шоу «Летающий цирк Монти Пайтона» высокоуровневый интерпретируемый язык программирования общего назначения. С точки зрения философии разработки, язык ориентирован на удобство написания и чтения кода. Такой подход вкупе с наличием мощных универсальных библиотек, о которых речь пойдёт дальше, стал решающим фактором при выборе этого языка программирования.

Описание библиотек начнём с Tkinter. Tkinter – мультиплатформенная библиотека для написания графического интерфейса на языке Python [26]. Она позволяет гибко описывать индивидуальные элементы интерфейса пользователя, легко организуя их взаимодействие между собой и с внешними участками программы.

Следующая библиотека – NumPy – часть пакета библиотек SciPy [27] для научно-ориентированных вычислений на Python, содержащая методы линейной алгебры, возможность генерации случайных чисел, а так же мощные инструменты для построения многоразмерных контейнеров для хранения и обработки разнородных данных.

Наконец, Matplotlib – так же входящая в пакет SciPy библиотека для визуализации двух- и трёхмерных данных [27].

Каждая из названных библиотек обладает подробной документацией, позволяющей быстро и точно решать возникающие трудности и недопонимания при работе, что является особенным плюсом при разработке комплексных решений.

## 3.2 Алгоритмическая поддержка решения

Моделирование системы происходит на основе аппарата клеточных автоматов, правила перехода и множество состояний клетки описаны в п.2.5, начальное состояние задаётся случайно, в качестве окрестности используется окрестность Мура, для сглаживания краевых эффектов поле КА топологически сворачивается в тор.

Схема работы программного обеспечения представлена на рисунке 3.1.

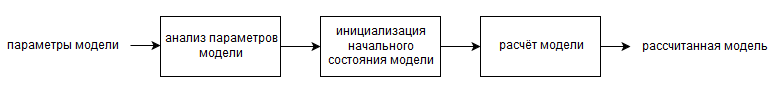


Рисунок 3.1 – Схема работы программного обеспечения

Таким образом, в программном обеспечении выделяется несколько модулей – задание начальных параметров модели, расчёт модели и вывод модели.

Далее, на рисунке 3.2 представлена структурно-функциональная схема ПО, а на рисунке 3.3 – блок-схема алгоритма работы ПО.

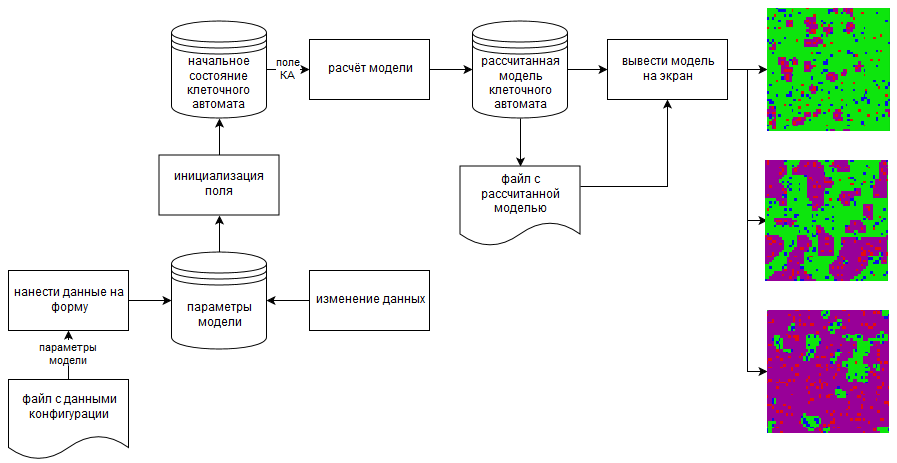


Рисунок 3.2 – Структурно-функциональная схема ПО

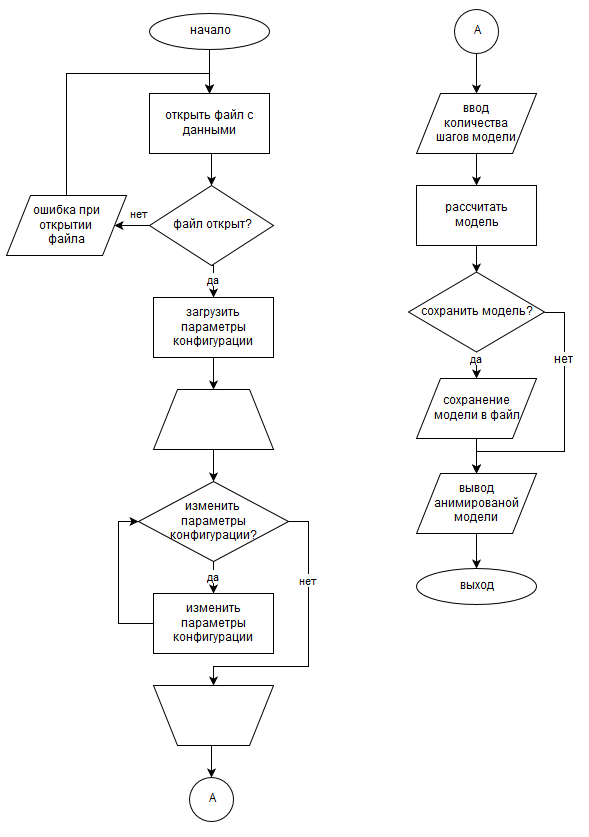


Рисунок 3.3 – Блок-схема алгоритма работы ПО

## 3.3 Описание основных функций программного обеспечения

Хорошей практикой при разработке комплексных решений является разделение решения на отдельные модули. Это приводит к ряду полезных результатов:

* позволяет разработчику легче ориентироваться в тексте программы;
* упрощает процесс овладения кодом для разработчиков, не имевших ранее опыта разработки конкретного решения;
* делает последующие доработку и развитие решения более удобными;
* даёт возможность повторного использования в будущем классов, функций и структур, определённых в решении.

Вследствие вышесказанного, было принято решение разделить по семантическому признаку разрабатываемое программное обеспечение на 4 модуля:

* main.py – основной модуль программы, обеспечивающий глобальную точку входа;
* projectD.py – модуль пользовательского интерфейса;
* modeling.py – модуль средств моделирования;
* miscellaneous.py – модуль прочих объектов, не подпадающих под тематику первых трёх модулей.

Далее представлен разбор и описание содержимого всех четырёх модулей.

**Функции модуля main.py**

**main( )**

**Описание:** главный метод программы, инициализирующий главное окно пользовательского интерфейса.

**Вход:** функция ничего не принимает на вход.

**Выход:** функция ничего не возвращает на выходе.

**Действия:** инициализирует главное окно пользовательского интерфейса.

Так же модуль main.py непосредственно вызывает функцию main(), выполняя роль глобальной точки входа.

**Классы модуля projectD.py**

**TheD(Frame, metaclass=Singleton)**

**Описание:** класс основной формы окна пользовательского интерфейса. Наследуется от класса Frame библиотеки Tkinter, описывающего элемент интерфейса – форму, и от метакласса Singleton, описанного в модуле miscellaneous.py. Класс TheD производит инициализацию элементов интерфейса основного окна, описывая их характеристики и расположение на окне, а так же содержит методы для описания их поведения.

**Методы класса TheD**

**\_\_init\_\_(self, parent)**

**Описание:** конструктор класса TheD.

**Вход:** метод принимает 2 аргумента – ссылку на собственный экземпляр класса (на самого себя) self и ссылку на окно, на которое закрепляется форма, parent.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** производит вызов конструктора родительского класса Frame, инициализирует поля класса, далее использующиеся для считывания данных с элементов формы, и, наконец, вызывает метод init\_ui(), задающий элементы интерфейса.

**init\_ui(self)**

**Описание:** производит инициализацию элементов интерфейса формы.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** инициализирует элементы интерфейса формы, задавая характеристики каждого из них, их расположение на форме, а так же порядок обработки содержащихся в них данных.

**create\_model(self)**

**Описание:** генерирует и выводит на экран рассчитанную по заданным на форме параметрам модель.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** собирает введённые пользователем данные с формы посредством вызова метода get\_settings(), проверяет их корректность, после чего генерирует на их основе модель развития клеточного автомата и выводит окно с анимацией рассчитанного процесса развития модели.

**load\_model(self)**

**Описание:** анимация модели, загруженной из файла.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** открывает диалоговое окно выбора файла. Проверяет корректность выбранного файла, после чего загружает из него модель и выводит её на экран.

**handle\_unsaved\_file(self)**

**Описание:** задаёт порядок обработки не сохранённых данных формы, предлагая пользователю их сохранить в виде файла.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** вывод на экран окно, предлагающее пользователю сохранить текущие данные формы. В случае положительного ответа, вызывает метод их сохранения.

**create\_file(self)**

**Описание:** очищает данные формы, предварительно предложив их сохранить.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** вызывает метод handle\_unsaved\_file(), после чего очищает данные формы.

**open\_file(self)**

**Описание:** позволяет открыть существующий файл конфигурации, предварительно предложив сохранить данные формы.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** вызывает метод handle\_unsaved\_file(), поле чего создаёт диалоговое окно открытия файла. В случае если пользователь открыл файл, извлекает их него данные и переносит их на форму посредством вызова метода set\_settings()

**save\_file(self)**

**Описание:** позволяет сохранить введённые на форму данные, перезаписав ранее открытый файл.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** проверяет, открывался ли ранее файл для считывания настроек и, если да, то перезаписывает этот файл текущими значениями данных на форме, которые собираются с помощью метода get\_settings(). В противном случае, вызывает метод save\_file\_as().

**save\_file\_as(self)**

**Описание:** позволяет сохранить введённые на форму данные в виде нового файла на диске.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** открывает диалоговое окно сохранения файла. Записывает в создаваемый файл данные формы, собранные посредством метода get\_settings().

**get\_settings(self)**

**Описание:** собирает данные с формы.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод возвращает словарь собранных с формы данных.

**Действия:** считывает данные с элементов формы, организует их в словарь, который впоследствии возвращает на выходе.

**set\_settings(self, settings)**

**Описание:** устанавливает значения элементов формы.

**Вход:** метод принимает 2 аргумента - ссылку на экземпляр собственного класса self и словарь данных settings.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** проверяет и корректирует переданные данные settings с помощью функции check\_input(), а затем устанавливает значения элементов settings как значения соответствующих элементов интерфейса.

**close\_window(self)**

**Описание:** задаёт обработку события закрытия окна программы.

**Вход:** метод принимает как аргумент ссылку на экземпляр собственного класса self.

**Выход:** метод нечего не возвращает на выходе.

**Действия:** вызывает метод handle\_unsaved\_file(), после чего закрывает окно программы и завершает её работу.

**Функции модуля projectD.py**

**check\_input(value, left\_boundary, right\_boundary)**

**Описание:**

**Вход:** функция принимает на вход 3 аргумента – проверяемое значение value, его минимальное значение left\_boundary и его максимальное значение right\_boundary.

**Выход:** при необходимости скорректированное значение value.

**Действия:** проверяет, являет ли value целым числом. Если нет, возвращает 0. Далее, проверяет, не является ли value меньше допустимого минимума left\_boundary. Если да, то возвращает left\_boundary. Наконец, проверяет, не является ли value больше допустимого максимума right\_boundary. Если да, то возвращается right\_boundary. Если все проверки пройдены успешно, возвращает неизменённое значение value.

**is\_correct2(value)**

**Описание:** функция для проверки значения, вводимого в поле ввода. Позволяет гарантировать, что в поле ввода нельзя ввести никаких других символов, кроме цифр, а так же, что длинна вводимого числа не будет превышать двух символов.

**Вход:** функция принимает как аргумент переменную value – набранную в поле ввода строку.

**Выход:** функция возвращает True, если введённое в поле ввода значение удовлетворяет заданным критериям, и False в противном случае.

**Действия:** проверяет, состоит ли вводимая строка исключительно из цифр, в противном случае возвращая False, и, если это так, проверяет длину вводимой строки.

**is\_correct3(value)**

**Описание:** функция для проверки значения, вводимого в поле ввода. Позволяет гарантировать, что в поле ввода нельзя ввести никаких других символов, кроме цифр, а так же, что введённое значение не будет превышать 150.

**Вход:** функция принимает как аргумент переменную value – набранную в поле ввода строку.

**Выход:** функция возвращает True, если введённое в поле ввода значение удовлетворяет заданным критериям, и False в противном случае.

**Действия:** проверяет, состоит ли вводимая строка исключительно из цифр, в противном случае возвращая False, и, если это так, проверяет длину вводимой строки.

**Функции модуля modeling.py**

**init\_custom(rows, cols, s\_prob=0.1, o\_prob=0.1)**

**Описание:** инициализирует начальное состояние решётки клеточного автомата с заданными параметрами распределения описанных в п.2.4 групп S и O.

**Вход:** функция принимает на вход 4 аргумента – аргументы rows и cols задают размерность решётки КА, опциональные аргументы s\_prob и o\_prob задают вероятность того, что очередная ячейка КА станет участником группы S или O соответственно.

**Выход:** функция возвращает numpy-массив размерности (1 x rows x cols), содержащий инициализированные ячейки решётки КА.

**Действие:** генерирует случайную величину, распределённую на отрезке [0; 1]. В зависимости от её значения, очередной ячейке КА присваивается значение одной из групп – O, I, S. Таким образом присваиваются значения всем ячейкам КА.

**evolve(cellular\_automaton, timesteps, apply\_rule, r=1, neighbourhood='Moore')**

**Описание:** моделирует развитие клеточного автомата.

**Вход:** функция принимает 5 аргументов – cellular\_automaton – начальное состояние клеточного автомата, timesteps – количество шагов расчёта клеточного автомата, apply\_rule – функция правила перехода, r – радиус окрестности клетки, по умолчанию 1, neighbourhood – вид окрестности клетки, по умолчанию – окрестность Мура.

**Выход:** numpy-массив размерности (timesteps x rows x cols) состояний КА на каждом из шагов его развития.

**Действие:** перебирает все ячейки КА, определяет ячейки, входящие в их окрестность, затем применяет в клетке переданную в качестве входного аргумента функцию перехода. Повторяет процесс указанное количество раз (шагов).

**update\_model\_settings(settings)**

**Описание:** обновляет объявленную в модуле переменную параметров КА.

**Вход:** функция принимает 1 аргумент – словарь параметров КА.

**Выход:** функция ничего не возвращает на выходе.

**Действие:** присваивает переменной модуля model\_settings значение переданного аргумента settings.

**destructive\_distribution\_rule(neighbourhood, c, t)**

**Описание:** применяет к клетке описанные в п.2.5 правила перехода.

**Вход:** функция получает на вход 3 аргумента – neighbourhood – numpy-массив соседей клетки, c – координаты обрабатываемой клетки и t – текущий шаг модели.

**Выход:** функция возвращает на выходе новое состояние клетки.

**Действие:** рассчитывает значения влияния соседних клеток исходя из заданных переменной model\_settings параметров модели, после чего задаёт новое состояние клетки.

**plot\_animate(ca, title='')**

**Описание:** создаёт анимацию развития КА.

**Вход:** функция принимает на входе 2 аргумента – ca – numpy-массив размерности (timesteps x rows x cols) состояний КА на каждом из шагов его развития и t – необязательный аргумент заголовка окна с анимацией.

**Выход:** функция ничего не возвращает на выходе.

**Действие:** анимирует процесс развития КА посредством последовательного построения состояний КА на каждом из шагов её развития.

**Классы модуля miscellaneous.py**

**Singleton(type)**

**Описание:** класс, реализующий одноимённый порождающий шаблон проектирования. Гарантирует, единственность экземпляра некоторого класса, объявленного как синглтон. Используется в модуле projectD.py, где синглтоном является класс TheD, описывающий основное окно пользовательского интерфейса.

**Методы класса Singleton**

**\_\_call\_\_(cls, \*args, \*\*kwargs)**

**Описание:** единственный метод класса Singleton(type).

**Вход:** функция принимает на вход 3 аргумента – тип класса cls, список позиционных аргументов args и список аргументов с ключевыми словами kwargs.

**Выход:** экземпляр класса cls.

**Действия:** Создаёт экземпляр класса, используя переданные тип класса cls, а так же списки его аргументов args и kwargs, если других экземпляров не существует, в противном случае возвращает ссылку на существующий экземпляр.

## 3.4 Тестирование программного обеспечения

Важным этапом разработки программного обеспечения является его тестирование. Процесс тестирования предполагает испытание ПО с целью выявления ошибок и неточностей в работе его функций, несоответствий между его предполагаемым и реальным поведением.

Функциями, в первую очередь нуждающимися в тестировании, являются функции из модуля modeling.py, ведь именно они выполняют наиболее подверженные ошибкам и критичные для работоспособности программы действия.

Для проверки функции генерации начального состояния клеточного автомата init\_custom() сгенерируем с её помощью массив клеток. Подсчитав количество клеток групп O и S вычислим фактическую вероятность их появления и, сравнив её с заданной, определим корректность работы функции.

Размерность генерируемого массива – , таким образом, мы обеспечили достаточный размер выборки. Заданные шансы появления ячеек групп O и S соответственно 10% и 20%.

На рисунке 3.3 показан результат проверки функции init\_custom().



Рисунок 3.3 – Результат проверки функции init\_custom()

Как видно из рисунка 3.3, заданные и фактические вероятности совпадают с точностью до десятитысячных. Таким образом, можно говорить о корректности работы функции init\_custom().

Для проверки функции evolve() сравним результат её работы на одном шаге перехода поля КА размерностью с вычисленным вручную.

Начальное состояние поля КА, заданного для тестирования, представленно на рисунке 3.4.

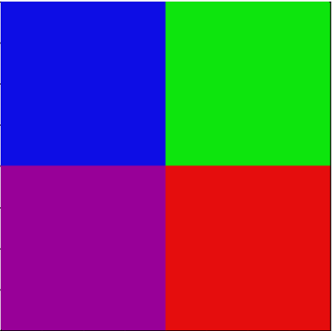


Рисунок 3.4 – Начальное состояние поля КА, заданное для тестирования

Пусть все начальные параметры равны 0, кроме влияния обычных людей, которое равно 9. В таком случае, согласно правилам перехода, описанным в п.2.5, на следующем шаге модели клетка Заражённого (фиолетовая на рис. 3.4) станет клеткой Обычного человека (зелёный), а клетка Распространителя (красная на рис.3.4) станет клеткой Противника (синий).

Следующее состояние системы, рассчитанное функцией evolve() показано на рисунке 3.5.

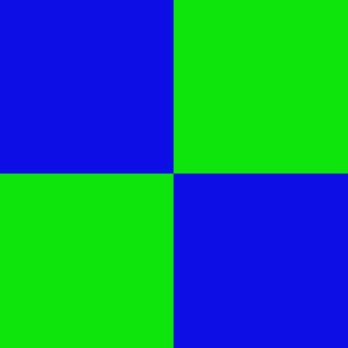


Рисунок 3.5 – Следующее состояние системы, рассчитанное функцией evolve()

Теоретическое состояние модели совпадает с рассчитанным функцией evolve(), из чего можно заключить корректность её работы. Так же отсюда можно сделать вывод о корректности работы функции переходов destructive\_distribution\_rule(), и о корректности работы функции plot\_animate(), выводящей модель на экран.

Не менее важным является тестирование пользовательского интерфейса. Здесь необходимо выявить все возможные ошибки функциональности интерфейса, отсутствие обработок возникающих исключений и искажения передаваемых через интерфейс данных. В силу специфики тестируемого объекта, оно было проведено вручную, путём проверки корректной работы каждого элемента пользовательского интерфейса.

Например, при попытке загрузить некорректный файл модели, программа должна выдавать сообщение о соответствующей ошибке и закрывать файл. Программа работает верно, это сообщение об ошибке показано на рисунке 3.6.

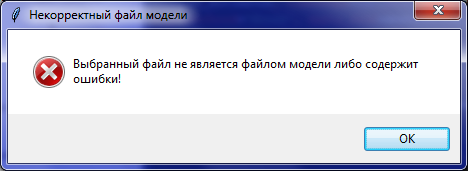


Рисунок 3.6 – Сообщение об ошибке при попытке открыть некорректный файл модели

Оценка быстродействия программы выполнялась на компьютере с процессором Intel(R) Core(TM) i7-4770 и объёмом оперативной памяти в 12 Гб.

Расчёт небольшой модели (размер поля – , 10 шагов) занимает 0.5466 секунды. Расчёт же модели при максимальном размере поля () и максимальном количестве шагов (99) занимает 52.1234 секунды. Такая разница во времени выполнения вызвана необходимостью расчёта переходов для каждой из клеток на каждом из шагов выполнения алгоритма.

При пиковой загруженности, программа использует приблизительно 79 Мб оперативной памяти.

Алгоритм является линейным, т.е. его теоретическая сложность не превышает . Тем не менее, в силу того, что входным данными являются ячейки клеточного автомата, время выполнения алгоритма стремительно увеличивается при увеличении таких начальных параметров модели как размер поля и количество шагов.

## 3.5 Использование разработанного программного обеспечения для расчёта модели сложной динамической социальной системы

Наконец, после завершения процессов разработки и тестирования программного обеспечения, используем его для расчёта разработанной в рамках текущей работы модели устойчивости сложной динамической социальной системы в условиях когерентных информационных атак.

Два основных сценария при первоначальном появлении деструктивной идеи:

* количество противников деструктивной идеи незначительно;
* количество противников деструктивной идеи сопоставимо с количеством её распространителей.

Рассмотрим оба сценария и для каждого из них определим параметры, при которых система будет устойчива.

При исследовании будут использоваться следующие процентные соотношения групп O и S: 5%, 10% и 15%.

Так же установим следующие значение характеристик групп:

* степень воздействия и степень сопротивления группы S – 9;
* степень воздействия группы O – 7, её сопротивления – 9;
* Степень воздействия группы I – 3, её сопротивление – 4;
* Степень воздействия группы N – 2, её сопротивления варьируется.

Результаты моделирования системы при незначительном количестве противников деструктивной идеи отражены в таблице 3.1.

Таблица 3.1 – Результаты моделирования системы при незначительном количестве противников деструктивной идеи

|  |  |  |
| --- | --- | --- |
| Доля группы S в исходном распределении, % | Степень невосприятия группы N | Результат |
| 5 (рис 3.7) | 0-1 | Система неустойчива (рис 3.8) |
| 5 | 2 | Система частично неустойчива |
| 5 | 3+ | Система устойчива |
| 10 (рис 3.9) | 0-3 | Система неустойчива |
| 10 | 4 | Система частично устойчива (рис 3.10) |
| 10 | 5+ | Система устойчива |
| 15 | 0-7 | Система неустойчива |
| 15 | 8-9 | Система частично устойчива |

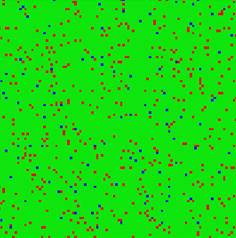


Рисунок 3.7 – Начальное состояние системы с 5% группы S и незначительным количеством группы O



Рисунок 3.8 – Неустойчивая система

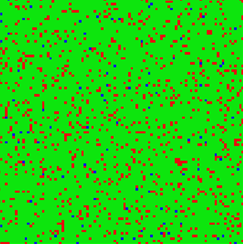


Рисунок 3.9 – Начальное состояние системы с 10% группы S и незначительным количеством группы O

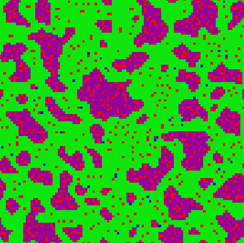


Рисунок 3.10 – Частично неустойчивая система

Результаты моделирования системы при наличии противников деструктивной идеи отражены в таблице 3.2.

Таблица 3.2 – Результаты моделирования системы при наличии противников деструктивной идеи

|  |  |  |  |
| --- | --- | --- | --- |
| Доля группы S в исходном распределении, % | Доля группы O в исходном распределении, % | Степень невосприятия группы N | Результат |
| 5 | 5 | 0+ | Система устойчива |
| 10 | 5 | 0 | Система неустойчива |
| 10 | 10 | 0 | Система частично устойчива |
| 10 | 10 | 1+ | Система устойчива |
| 10 | 15 | 0+ | Система устойчива |
| 15 | 5 | 0-4 | Система неустойчива |
| 15 | 5 | 4-7 | Система частично устойчива |

Продолжение таблицы 3.2.

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | 5 | 9 | Система устойчива |
| 15 | 10 | 0-2 | Система неустойчива |
| 15 | 10 | 3-5 | Система частично устойчива |
| 15 | 10 | 6+ | Система устойчива |
| 15 | 15 | 0-2 | Система неустойчива |
| 15 | 15 | 3-4 | Система частично устойчива |
| 15 | 15 | 5+ | Система устойчива |

Проанализировав полученные результаты, сделаем следующие выводы:

* Даже без наличия противников деструктивной идеи, система достаточно устойчива, и для её дестабилизации необходимо достаточно большое (не менее 10% от общего числа людей) число распространителей деструктивной идеи.
* При этом при преодолении отметки в 15% распространителей, устойчивость системы резко падает.
* Чем инертнее аудитория, тем сильнее колебания общественного мнения.
* В общем случае, люди не склонны поддерживать непопулярные мнения.
* Устойчивость сложной социальной системы сильно зависит от её начального состояния.
* Хотя разработанная модель сильно упрощена относительно реальных социальных систем, она, тем не менее, позволяет наблюдать многие прослеживающиеся в этих системах закономерности, из чего можно сделать вывод о её состоятельности.

## Выводы

1. Дано обоснование выбора технических средств разработки, а именно языка программирования Python и его библиотек Tkinter, NumPy и Matplotlib.
2. Описана работа алгоритма программы, дано описание основных классов и функций программы.
3. Разработано и всесторонне протестировано программное обеспечение для моделирования сложных динамических социальных систем на основе аппарата клеточных автоматов.
4. Практическое применение разработанного ПО было продемонстрированно на примере расчёта построенной во 2 разделе модели сложной динамической социальной системы.
5. Были проанализированы результаты моделирования.

# 4. Экономический раздел

Целью данного раздела является расчёт экономического аспекта в рамках разработки программного продукта. Он включает в себя расчёт затрат для исследований целесообразности разработки, планирования разработки, график выполнения работ, интеграции и сопровождения.

## 4.1 Расчёт сметы затрат на проведение исследования. Построение сетевого графика с расчётом по параметрам

### 4.1.1 Определение списка работ и состава исполнителей

Процесс разработки включает в себя такие этапы как обзор программных средств соответствующей тематики, их анализ, анализ и выбор программных продуктов для создания программы, отладку и испытания. В каждом из этих этапов можно выделить свои подэтапы.

Согласно ГОСТ 23501.1-79 определены следующие стадии создания программного обеспечения:

* техническое задание – ТЗ (ГОСТ 23501.2-79);
* эскизный проект – ЭП (ГОСТ 23501.5-80);
* технический проект – ТП (ГОСТ 23501.6-80);
* рабочий проект – РП (ГОСТ 23501.11-81);
* внедрение – ВП (ГОСТ 23501.15-81).

В разработке проекта принимают участие руководитель проекта, аналитик, разработчик и технический писатель (таблица 4.1).

Таблица 4.1 – Перечень работ и состав исполнителей

|  |  |  |
| --- | --- | --- |
| Стадии разработки | Наименование работы | Ответственные исполнители |
| 1. Техническое задание | 1.1. Постановка задачи | Руководитель проекта  Аналитик |

Продолжение таблицы 4.1.

|  |  |  |
| --- | --- | --- |
|  | 1.2. Подбор и анализ технической литературы | Аналитик |
|  | 1.3. Сбор и анализ исходных данных | Аналитик |
| 1.4. Анализ существующих решений | Аналитик |
| 1.5. Определение этапов, сроков разработки системы и технической документации | Руководитель проекта  Технический писатель |
| 1.6. Согласование и утверждение технического задания | Руководитель проекта  Аналитик  Технический писатель |
| 2. Эскизный проект | 2.1. Разработка структуры входных и выходных данных | Разработчик  Аналитик |
| 2.2. Выбор критериев эффективности и качества системы | Разработчик  Аналитик |
| 2.3. Разработка технологии тестирования и отладки | Разработчик |
| 2.4. Согласование и утверждение эскизного проекта | Руководитель проекта  Разработчик |
| 3. Технический проект | 3.1. Разработка структурной схемы | Руководитель проекта  Разработчик |

Продолжение таблицы 4.1.

|  |  |  |
| --- | --- | --- |
|  | 3.2. Разработка алгоритмического и программного обеспечения | Разработчик |
| 3.3. Разработка интерфейса | Разработчик |
| 3.4. Согласование и утверждение технического проекта | Руководитель проекта  Разработчик |
| 4. Рабочий проект | 4.1. Программирование | Разработчик |
| 4.2. Тестирование | Разработчик |
| 4.3. Отладка | Разработчик |
| 4.4. Разработка программной документации | Технический писатель  Разработчик |
| 4.5. Согласование и утверждение рабочего проекта | Руководитель проекта  Разработчик |
| 5. Внедрение | 5.1. Подготовка и передача системы и технической документации для сдачи | Руководитель проекта  Технический писатель  Разработчик |
| 5.2. Анализ данных полученных в результате эксплуатации | Аналитик  Разработчик |

Продолжение таблицы 4.1.

|  |  |  |
| --- | --- | --- |
|  | 5.3. Корректировки технической документации по результатам испытаний | Технический писатель  Разработчик |

### 4.1.2 Расчёт продолжительности и трудоёмкости работ

В процессе разработки нового программного продукта следует использовать вероятностный метод расчёта на основе экспертных оценок. Для начала, определимся с ожидаемой продолжительностью каждого вида работы, используя формулу (4.1):

 (4.1)

Где  - ожидаемая продолжительность i-ой работы, измеряемая в рабочих днях,  - минимальная ожидаемая продолжительность i-ой работы, измеряемая в рабочих днях,  - максимальная ожидаемая продолжительность i-ой работы, измеряемая в рабочих днях.

Длительность исполнения каждого вида работы рассчитывается по формуле (4.2):

 (4.2)

Где  - коэффициент для пересчёта рабочих дней в календарные. Для вычисления  используется формула (4.3):

 (4.3)

Где  - количество календарных дней в исследуемом году,  - количество рабочих дней в этом году.

Используя формулу (4.3), вычислим количество рабочих дней в 2019 году, в котором 365 дней, из которых 247 рабочих:



Для определения трудоёмкости каждого вида работ, используется формула (4.4):

 (4.4)

Где  - трудоёмкость каждого вида работ, человеко-дни,  - количество исполнителей.

Рассчитанные продолжительность и трудоёмкость работ отображены в таблице 4.2.

Таблица 4.2 – Продолжительность и трудоёмкость работ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Перечень выполняемых работ | Продолжительность работ, рабоч. дн. | | Количество исполнителей, | Трудность работ,  чел-дн. | Продолжительность работ,  кален. дн. |
|  |  |
| 1. Техническое задание | | | | | |
| 1. Постановка задачи | 1 | 2 | 2 | 2,8 | 2,1 |
| 2. Подбор и анализ технической литературы | 2 | 3 | 1 | 2,4 | 3,6 |
| 3. Сбор и анализ исходных данных | 2 | 3 | 1 | 2,4 | 3,6 |
| 4. Анализ существующих решений | 2 | 4 | 1 | 2,8 | 4,2 |

Продолжение таблицы 4.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5. Определение этапов, сроков разработки системы и технической документации | 1 | 2 | 2 | 2,8 | 2,1 |
| 6. Согласование и утверждение технического задания | 1 | 3 | 3 | 3,6 | 2,7 |
| 2. Эскизный проект | | | | | |
| 1. Разработка структуры входных и выходных данных | 1 | 2 | 2 | 2,8 | 2,1 |
| 2. Выбор критериев эффективности и качества системы | 1 | 2 | 2 | 2,8 | 2,1 |
| 3. Разработка технологии тестирования и отладки | 2 | 3 | 1 | 2,4 | 3,6 |
| 4. Согласование и утверждение эскизного проекта | 1 | 3 | 2 | 3,6 | 2,7 |

Продолжение таблицы 4.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3. Технический проект | | | | | |
| 1. Разработка структурной схемы | 2 | 3 | 2 | 4,8 | 3,6 |
| 2. Разработка алгоритмического и программного обеспечения | 3 | 5 | 1 | 3,8 | 5,7 |
| 3. Разработка интерфейса | 1 | 3 | 1 | 1,8 | 2,7 |
| 4. Согласование и утверждение технического проекта | 1 | 3 | 2 | 3,6 | 2,7 |
| 4. Рабочий проект | | | | | |
| 1. Программирование | 7 | 14 | 1 | 9,8 | 14,6 |
| 2. Тестирование | 4 | 7 | 1 | 5,2 | 7,7 |
| 3. Отладка | 4 | 7 | 1 | 5,2 | 7,7 |
| 4. Разработка программной документации | 1 | 3 | 2 | 3,6 | 2,7 |
| 5. Согласование и утверждение рабочего проекта | 1 | 3 | 2 | 3,6 | 2,7 |

Продолжение таблицы 4.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5. Внедрение | | | | | |
| 1. Подготовка и передача системы и технической документации для сдачи | 1 | 4 | 3 | 6,6 | 3,3 |
| 2. Анализ данных полученных в результате эксплуатации | 2 | 3 | 2 | 4,8 | 3,6 |
| 3. Корректировки технической документации по результатам испытаний | 1 | 3 | 2 | 3,6 | 2,7 |
| Итого: | | | | 84,8 | 88,5 |

В результате расчётов была получена продолжительность проекта в 88,5 календарных дней, а трудоёмкость – 84,8 человеко-дней (при условии последовательного выполнения работ).

### 4.1.3 Планирование выполнения разработки программного продукта

В качестве инструмента планирования выполнения работ используется диаграмма Ганта – столбчатая диаграмма, позволяющая графически иллюстрировать процесс протекания проекта. Диаграмма Гранта разработки продукта, составленная на основе данных из таблицы 4.2, предоставлена на рисунке 4.1.

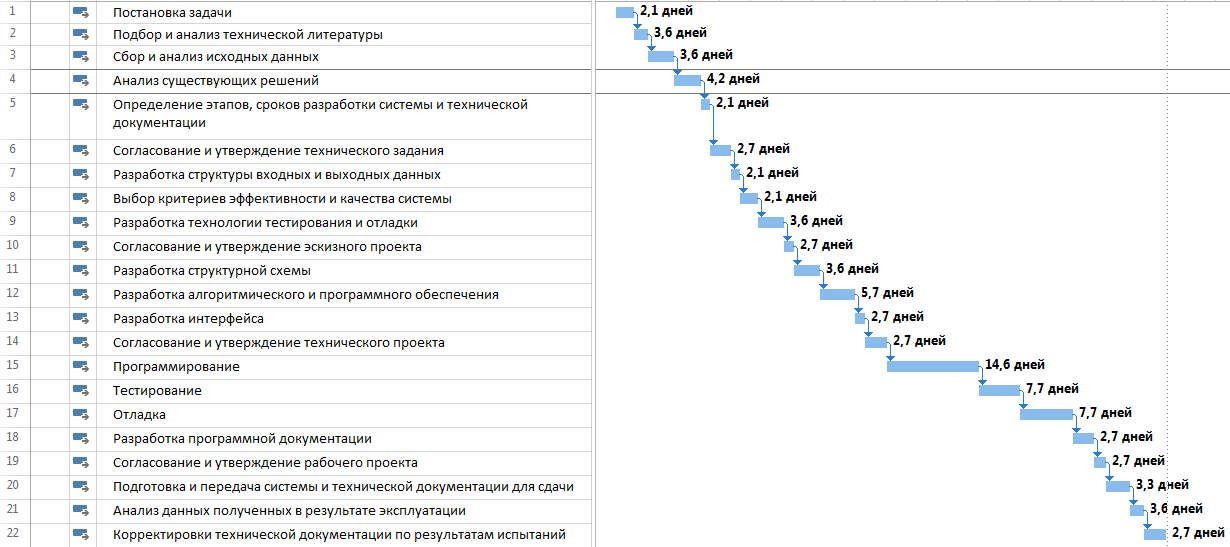


Рисунок 4.1 – Сетевой график выполнения работ

## 4.2 Расчёт экономической эффективности

Важной частью процесса планирования является составление сметы. Смета позволяет учесть потребность в дополнительных денежных ресурсах, и её необходимо учитывать при инвестициях в разрабатываемый проект.

Формула (4.5) позволяет рассчитывать сметную стоимость:

 (4.5)

Где:

*  – затраты на основные материалы, покупаемые полуфабрикаты и комплектующие;
*  – заработная плата исполнителей;
*  – дополнительная заработная плата исполнителей;
*  – отчисления от заработной платы исполнителей;
*  – затраты на машинное время;
*  – накладные расходы;
*  – прочие затраты.

### 4.2.1 Расчёт затрат на основные материалы

Для определение затрат на основные материалы, покупаемые полуфабрикаты и комплектующие используется формула (4.6):

 (4.6)

Где  - количество покупных изделий,  - цена за единицу (руб./ед. изм.), а  - коэффициент транспортных расходов (5%).

Результаты расчётов представлены в таблице 4.3:

Таблица 4.3 – Расчёт затрат на материалы и комплектующие

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Материалы, покупные изделия | Единица измерения | Цена (руб./ед. изм.) | Количество (ед. изм.) | Стоимость (руб.) |
| Магнитная доска | Штука | 2500 | 1 | 2500 |
| Бумага писчая | Листы | 1 | 500 | 500 |
| Маркеры | Штука | 60 | 5 | 300 |
| Ручки | Штука | 20 | 10 | 200 |
| USB-флеш-накопители | Штука | 600 | 10 | 6000 |
| Другие канцтовары | - | - | - | 3000 |
| Итого, общая стоимость: | | | | 12500 |
| Итого, с учётом транспортных расходов: | | | | 13125 |

Таким образом, объём затрат на материалы и комплектующие с учётом транспортных расходов составляет 13125 рублей.

### 4.2.2 Расчёт основной заработной платны исполнителей

Для определения заработной платы исполнителей, используется формула (4.7):

 (4.7)

Где  - затраты времени i-ого работника,  - его тарифная ставка, а  - коэффициент для учёта выплат дополнительных премий.

Результаты расчётов заработной платы отображены в таблице 4.3:

Таблица 4.3 – Расчёт заработной платы исполнителей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнители | Месячная оплата труда (руб./ мес.) | Дневная оплата труда (руб./ дн.) | Время работы (рабоч. дн.) | Общая величина заработной платы (руб.) |
| Руководитель проекта | 60000 | 3000 | 22 | 66000 |
| Аналитик | 30000 | 1500 | 57 | 85500 |
| Разработчик | 45000 | 2250 | 71 | 159750 |
| Технический писатель | 25000 | 1250 | 14 | 17500 |
| Итого: | | | 164 | 262750 |

### 4.2.3 Расчёт дополнительной заработной платы исполнителей

Расчёт дополнительной заработной платы исполнителей производится с помощью формулы (4.8):

 (4.8)

Таким образом, на основании данных таблицы 4.3 и формулы (4.8), можно определить значение основной заработной платы исполнителей как , а дополнительной заработной платы - .

### 4.2.4 Расчёт отчислений от заработной платы исполнителей

Отчисления  от заработной платы исполнителей определяются по формуле (4.9):

 (4.9)

Где  - коэффициент, определяющий размер отчислений.

Отчисления  включают в себя страховые взносы в пенсионный фонд, фонды обязательного медицинского и социального страхования, состав и размер которых определяется действующим на данный момент времени законодательством. По состоянию на 2019 год, согласно федеральным законам №212–ФЗ от 24.07.09 и №333–ФЗ от 02.12.13 и Постановлению Правительства РФ №1101 от 30.11.13, общая величина этих взносов составляет 30% с суммы выплат с начала года на каждого работника до 624000 рублей и 10% с выплат, превышающих эту величину. Кроме того, в страховые взносы входят выплаты на обязательное социальное страхование от несчастных случаев и профессиональных заболеваний, размер которых дифференцирован в зависимости от класса профессионального риска. В соответствии с федеральным законом №323–ФЗ от 02.12.13, для видов экономической деятельности, отнесённой к 1–му классу риска, размер страховых взносов равен 0,2% от заработной платы.

Таким образом, коэффициент  будет равняться 0,302. Отсюда получим :

руб.

### 4.2.5 Расчёт затрат на машинное время

Для определения себестоимости машинного часа, примем срок службы технических средств за 3 года, общую их себестоимость  за 70000 руб., годовые затраты на техническое обслуживание, ремонт, а так же электроэнергию, за 20% от себестоимости технических средств. И, наконец, средний коэффициент использования технических средств  примем равным 0,95.

По формуле (4.10) найдём действительный годовой фонд времени работы оборудования:

 (4.10)

Где:

*  – количество рабочих дней в году (247);
*  – длительность рабочего дня (8 часов);
*  – количество праздничных дней (13 дней);
*  – время сокращения рабочего дня (1 час);
*  – коэффициент потерь рабочего времени, связанных с ремонтом технических средств (примем равным 0,96).

Подставим перечисленные значения в формулу (4.10):

 часа.

Себестоимость машинного часа можно вычислить по формуле (4.11):

 (4.11)

Из (4.11) найдём стоимость машинного часа :

 руб./ ч.

Наконец, затраты на машинное время можно вычислить по формуле (4.12):

 (4.12)

Вычислим :

рублей.

### 4.2.6 Расчёт накладных расходов

Накладные расходы включают в себя командировочные расходы, плату за коммуникацию и электроснабжение, плату за аренду помещений и т.д. Как правило, они составляют от 30% до 50% основной заработной платы и рассчитываются по формуле (4.13):

 (4.13)

Из формулы (4.13) находим величину накладных расходов :

 рублей.

### 4.2.7 Расчёты прочих затрат

К прочим затратам относятся налоги, сборы, отчисления во внебюджетные фонды, платежи по кредитам в пределах ставок, затраты по подготовке и переподготовке кадров, ремонтный фонд, платежи по обязательному страхованию и т.д. Прочие затраты принимаем за 10% от размера общих затрат.



Общая смета затрат на разработку представлена в таблице 4.5.

Таблица 4.5 – Смета затрат на разработку проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Статья затрат | Обозначение | Величина затрат | % затрат к итогу |
| Затраты на основные материалы |  | 13125 | 2,3 |
| Основная заработная плата |  | 262750 | 45,6 |
| Дополнительная заработная плата |  | 34157,5 | 6 |
| Отчисления от заработной платы |  | 89666,07 | 15,6 |
| Затраты на машинное время |  | 20996,48 | 3,7 |
| Накладные расходы |  | 103917,6 | 18,1 |
| Прочие затраты |  | 52461,27 | 9,1 |

Продолжение таблицы 4.5.

|  |  |  |  |
| --- | --- | --- | --- |
| Итого |  | 577073,9 | 100 |

Как видно из таблицы 4.5, себестоимость разработки программных средств составляет 577073,9 рублей.

### 4.2.8 Расчёт затрат на внедрение системы

Затраты на внедрение системы включают в себя:

* стоимость разработки проекта;
* стоимость программного обеспечения, необходимого для разработки проекта;
* стоимость дополнительного оборудования, требующегося для функционирования разработанного ПО.

Стоимость разработки была определена в п.4.2.7.

В процессе разработки использовалось бесплатное программное обеспечение, для его внедрения не требуется дополнительных затрат на ПО.

Так как программное обеспечение в дальнейшем будет размещаться на оборудовании заказчика, то затрат на дополнительное оборудование так же не требуется.

### 4.2.9 Расчёт затрат на сопровождение системы

В таблице 4.6 приведены затраты на оплату труда специалистов.

Таблица 4.6 – Зарплаты специалистов

|  |  |  |  |
| --- | --- | --- | --- |
| Должность | Количество | Заработная плата (с учётом налогов) (руб.) | Сумма (руб.) |
| Руководитель проекта | 1 | 60000 | 60000 |
| Аналитик | 1 | 30000 | 30000 |
| Разработчик | 1 | 45000 | 45000 |

Продолжение таблицы 4.6.

|  |  |  |  |
| --- | --- | --- | --- |
| Технический писатель | 1 | 25000 | 25000 |
| Итого: | | | 160000 |

Таким образом, затраты на сопровождение системы в месяц составляют 160000 рублей.

Результаты расчётов затрат на разработку и дальнейшее сопровождение разработанного ПО отражены в таблице 4.7.

Таблица 4.7 – Затраты на разработку и дальнейшее сопровождение ПО

|  |  |  |
| --- | --- | --- |
| Пункт расходов | | Затраты |
| Разработка | | |
| 1. Затраты на материалы | |  |
| 2. Оплата машинного времени | |  |
| 3. Затраты на оплату труда (всего) | | 386573,6 |
| в том числе: | | |
| 3.1. Основная заработная плата | 262750 | |
| 3.2. Дополнительная заработная плата | 34157,5 | |
| 3.3. Отчисления от заработной платы | 89666,07 | |
| 4. Накладные расходы | 103917,63 | |
| 5. Прочие затраты | 52461,27 | |
| Общая сумма затрат на разработку проекта | 577073,9 | |
| Затраты на внедрение | | |
| 1. Стоимость затрат на оборудование, программное обеспечение | 0 | |
| 2. Стоимость разработки | 577073,9 | |

Продолжение таблицы 4.7.

|  |  |
| --- | --- |
| Общая сумма на внедрение проекта | 577073,9 |
| Затраты на дальнейшее сопровождение | |
| 1. Затраты на оплату труда | 160000 |
| 2. Затраты на содержание и эксплуатацию техники (всего) | 0 |
| в том числе: | |
| 1. Расходы на аренду помещения | 0 |
| Общая сумма затрат на сопровождение (за один месяц) | 160000 |

### 4.2.10 Расчёт срока окупаемости системы

Бизнес-модель разработанного ПО заключается в продаже программы и последующей платной ежемесячной поддержке. При этом процесс реализации ПО будет происходить в течение длительного срока. В стоимость покупки включена стоимость первого месяца поддержки.

Расчёт срока окупаемости производится путём расчёта отношения затраченных средств к прибыли.

 (4.14)

Где  - затраты на сопровождение,  - месячная прибыль.

Установив стоимость продукта равную 300000 рублей, получим значение прибыли с продажи одного продукта  равное:

рублей.

После уплаты налогов в размере 20% от прибыли, получим:

 рублей.

Так как используется оборудование заказчика и закупки ПО для разработки не требуется (см п.4.2.8), то затраты будут целиком состоять из себестоимости разработанного программного продукта .

Используя формулу (4.14) определим срок окупаемости:

 месяца.

Поскольку выбранная бизнес-модель рассчитана на извлечение прибыли при продаже продукта, то для полного окупления продукта необходимы 6 продаж. В течение первых полугода, планируется продажа одного продукта каждые 10-20 дней. В этом случае, возможно извлечение прибыли на 2-ой месяц.

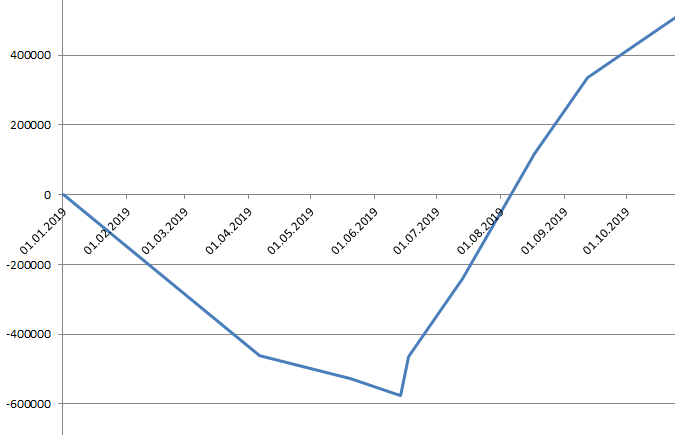
На рисунке 4.2 отображён плановый график получения денежных средств.

Рисунок 4.2 – Плановый график получения денежных средств

## Выводы

В данном разделе был представлен расчёт смет разработки программы, а так же стоимости её внедрения и сопровождения. В конце был приведён плановый график получения денежных средств.

Были получены следующие данные:

* Время реализации проекта составляет 5,16 месяца.
* Смета затрат разработки ПО составляет 577073,9 рублей.
* Расчётный период реализации продукта составляет 4 месяца после окончания разработки продукта, при этом планируется реализовать 12 единиц продукта.

Можно сделать вывод о том, что разработанное программное обеспечение позволяет достаточно быстро окупить затраченные на разработку ресурсы в течение весьма короткого промежутка времени (2 месяца) и получить положительный экономический эффект.

# 5. Техническая документация

## 5.1. Руководство пользователя

Программное средство позволяет моделировать устойчивость сложных динамических социальных систем, общий вид которых описан в п.2.4 и п.2.5.

Пользователь должен иметь уверенные навыки работы с персональным компьютером, а так же ориентироваться в предметной области.

### 5.1.1. Выполнение программы

Запуск программы осуществляется путём исполнения команды консоли Windows «python main.py» в основной директории программы.

После запуска открывается окно программного обеспечения, показанное на рисунке 5.1.

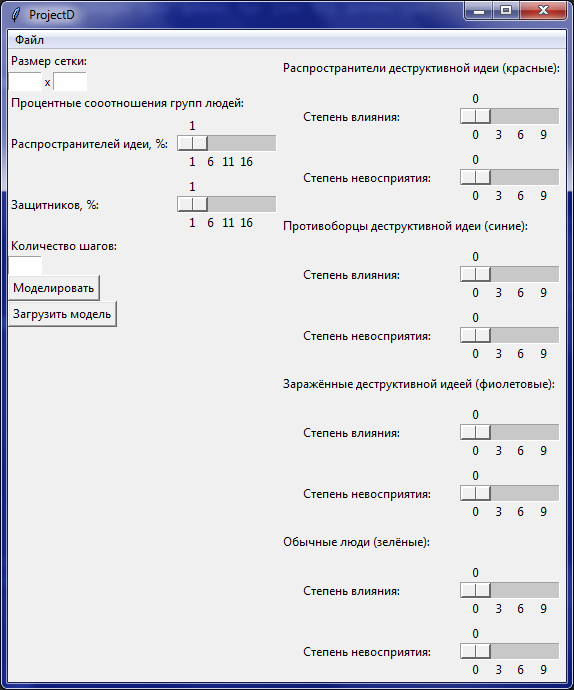


Рисунок 5.1 – Окно программного обеспечения

Для того чтобы начать работу с программой, необходимо заполнить поля «размер сетки» и «количество шагов», а так же задать процентные соотношения групп людей и параметры каждой из групп, после чего нажать кнопку «моделировать».

Так же возможна загрузка параметров из существующего файла конфигурации (меню «Файл» - «Открыть») и файла с рассчитанной моделью (кнопка «Загрузить модель»).

Выход из программы осуществляется нажатием на значок «крестик» в правом верхнем углу окна.

### 5.1.2. Условия выполнения программы

Программа может быть выполнена на платформе допустимой операционной системы. Климатические условия эксплуатации должны удовлетворять требованиям к техническим средствам по части условий их эксплуатации.

Допустимые операционные системы, а так же минимальные аппаратные требования указаны в п.5.3.

### 5.1.3. Сообщение оператору

Программа отлажена, ошибок быть не должно. В случае появления ошибки, перезагрузите программу.

## 5.2. Руководство программиста

В данном руководстве содержатся указания для программистов, желающих внести изменения в функционал программы в процессе её эксплуатации.

### 5.2.1. Назначение и условия применения программы

Назначение и функции, выполняемые программой, а также условия, необходимые для выполнения программы, описаны в разделе руководство пользователя.

### 5.2.2. Характеристика программы

Выполнение разработанного программного обеспечения происходит при использовании интерпретатора языка Python. Временные характеристики, режим работы и средства контроля правильности выполнения определяются особенностями интерпретатора.

### 5.2.3. Обращение к программе

Все основные классы, их методы, а так же функции программы описаны в п.3.3 данной работы.

### 5.2.4. Входные и выходные данные

Используемый в данном программном обеспечении формат входных и выходных данных строго привязан к пользовательскому интерфейсу программы, что обеспечивает их однозначную трактовку.

### 5.2.5. Сообщения

Вследствие того, что преимущественным в программе является графический интерфейс, текстовые сообщения присутствуют лишь в виде диалогов сохранения файлов и окнах ошибок.

## 5.3. Руководство системного администратора

В данном руководстве содержится перечень указаний, необходимых к выполнению для ввода разработанного программного обеспечения в эксплуатацию.

Для использования разработанных в рамках данной выпускной квалификационной работы программных средств требуется персональный компьютер с установленной на нём одной из перечисленных операционных систем: Windows 7, Windows 8.1, Windows 10.

Для запуска программы потребуется установленный интерпретатор языка Python версии не ниже 3.6.2, а так же следующие библиотеки: Tkinter (версия 8.6), NumPy (версия 1.14.3) и Matplotlib (версия 2.2.2).

Минимальные аппаратные требования:

* 64-разрядный (х64) процессор с тактовой частотой 2 ГГц и выше;
* 2 Гб оперативной памяти или больше;
* 17 Мб свободного места на жёстком диске;
* монитор и видеоадаптер;
* клавиатура, мышь или другие совместимые устройства ввода данных.

В перечень задач, выполняемых системным администратором, должны входить:

1. поддержание работоспособности технических средств;
2. установка (инсталляция) и поддержание работоспособности системных программных средств – операционной системы;
3. установка (инсталляция) программы.

## Выводы

В данном разделе было произведено документирование основных возможностей разработанной в рамках выпускной квалификационной работы программы. Были приведены руководства обращения с программой для пользователя, программиста и системного администратора, а так же рассмотрены особенности работы с приложением и сообщения системы.

# Заключение

В результате выполнения выпускной квалификационной работы были разработаны модель устойчивости сложной динамической социальной системы в условиях когерентных информационных атак на основе аппарата клеточных автоматов и программное средство для расчёта подобных моделей. Разработанное программное средство может в дальнейшем использоваться в исследовательских и практических целях в области моделирования сложных динамических социальных систем. Была составлена пояснительная записка.

Таким образом, были решены все задачи, поставленные в выпускной квалификационной работе.

В исследовательском разделе была описана предметная область работы и рассмотрены подходы к моделированию устойчивости сложных динамических систем, а так же сделан вывод о необходимости разработки моделирующего программного обеспечения.

В специальном разделе рассмотрены характеристики устойчивости СДС, управление ею при помощи локальных однородных воздействий и разработана модель сложной динамической социальной системы в условиях когерентных информационных атак.

В технологическом разделе было произведено обоснование выбранных средств для разработки моделирующего программного обеспечения, были описаны алгоритмы, лежащие в основе решения, назначения классов и функций разработанного программного обеспечения, произведено его тестирование. Наконец, с помощью разработанного программного обеспечения был произведён расчёт и анализ построенной в специальном разделе модели устойчивости сложной динамической социальной системы в условиях когерентных информационных атак.

В экономическом разделе был составлен ленточный график выполнения работ, произведён расчёт сметы проекта и сроков его окупаемости.

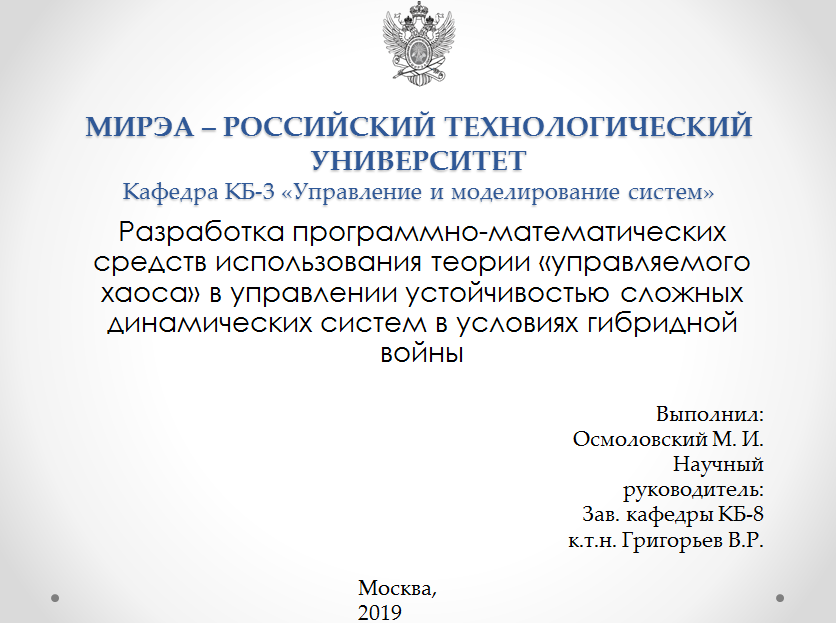
Наконец, в разделе технической документации были составлены руководства для пользователей программы, программистов и системных администраторов.

# Список использованных источников

1. С. Гриняев. Концепция ведения информационной войны в некоторых странах мира. – ПБ ."Зарубежное военное обозрение" от 4/03/2002
2. Фельдбаум А.А. Основы теории оптимальных автоматических систем. М.:Физматгиз, 1963.
3. И.Пригожин. От существующему к возникающему. М., 1985. –   
   327 с.
4. Капица С.П., Курдюмов С.П., Малинецкий Г.Г. Синергетика и прогнозы будущего. М.: Наука, 1997. С.87.
5. С.А.Модестов. «Информационное противоборство как фактор геополитической конкуренции». Издательский центр научных и учебных программ. Москва. 1998г.
6. Князева Е.Н., Курдюмов С.П. Будущее и его горизонты: синергетическая методология в прогнозировании // Синергетика. Труды семинара. Том 4. М.: Издательство МГУ, 2001. С.5-19.
7. В.В. Качала. Основы теории систем и системного анализа. М.: Горячая линия – Телеком, 2007.
8. Поваров В.М. Об уровнях сложности систем // Методологические проблемы кибернетики: материалы к Всесоюзн. Конф. – М.: МГУ, 1970. Т. 2. – с. 176-179.
9. Черняк Ю.И. Анализ и синтез систем в экономике. – М.: Экономика, 1975. – 191 с.
10. Бир Ст. Мозг фирмы. – М.: УСРР, 2005. – 416 с.
11. Theory and applications of cellular automats / Ed. By S.A. Wolfram. — Singapore: World Scientific, 1986.
12. Wolfram S. Statistical mechanics of Cellular automata // Review of Modern Physics. Vol.55. 1993. 607-640.
13. Урсул А.Д., Урсул Т.А. Устойчивое развитие и безопасность. Учебное пособие. – М.: 2013
14. Лекции по теории графов / В.А. Емеличев, О.И. Мельников, В.И. Сарванов, Р.И. Тышкевич. – М.: Наука, 1990.
15. Форд Л., Фалкерсон Д. Потоки в сетях. – М.: Мир,1966.
16. Филлипов. Управление устойчивостью сложной динамической системы посредством локальных однородных взаимодействий. – М.: Мир, 1986.
17. Тоффоли Т., Марголус Н. Машины клеточных автоматов. М.: Мир, 1991.
18. Фон Нейман Дж. Теория самовоспроизводящихся автоматов. М.: Мир, 1971.
19. Шалыто А., Туккель Н. От тьюрингова программирования к автоматному //Мир ПК. 2002, № 2, с. 144—149.
20. Наумов Л.А., Шалыто А.А. Клеточные автоматы. Реализация и эксперименты //Мир ПК. 2003. № 8, с. 64—71.
21. Esser J., Schrechenberg M. Microscopic simulation of urban traffic based on cellular automata //International Journal of Modern Physics, Vol. 8, No. 5 (1997) c. 1025 -1036..
22. М. Мескон и др. Основы менеджмента. - М.: Дело, 1992.
23. Дж. Форрестер. Мировая динамика. - М.: Наука, 1978.
24. Александров В. В. Об одном вычислительном эксперименте, моделирующем последствия ядерной войны. Вычислительная математика и математическая физика, 1984, т. 24, стр. 140—144.
25. Сайт системы имитационного моделирования WinALT. URL: http://winalt.sscc.ru (доступ: 28.01.2019).
26. Сайт документации Python. Режим доступа: [https://docs.python.org/]
27. Библиотека SciPy. Режим доступа: [https://www.scipy.org/]

# Приложение А. Перечень графического материала

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |

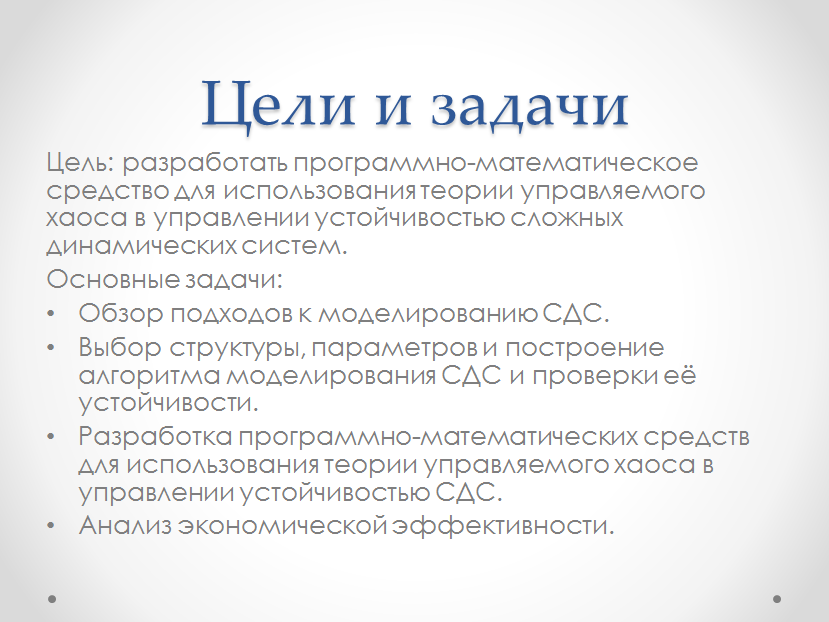




|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |





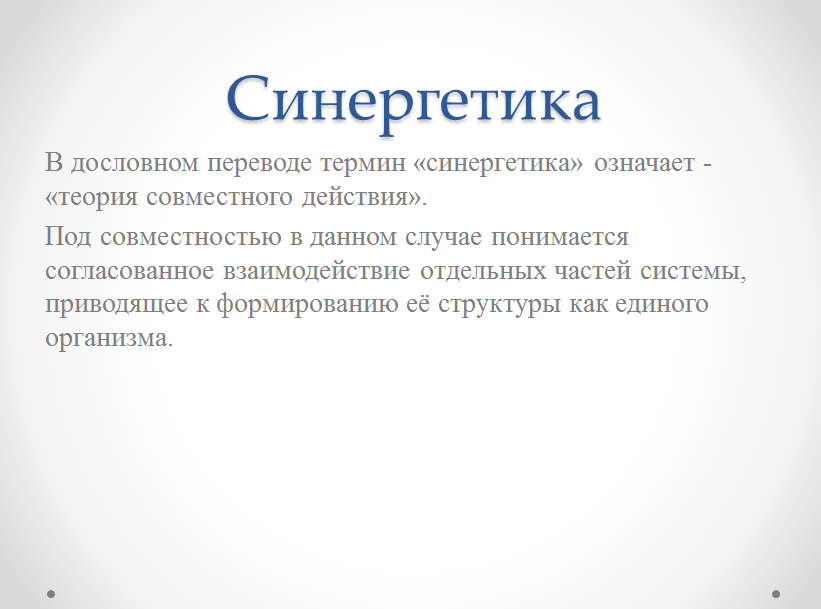
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |

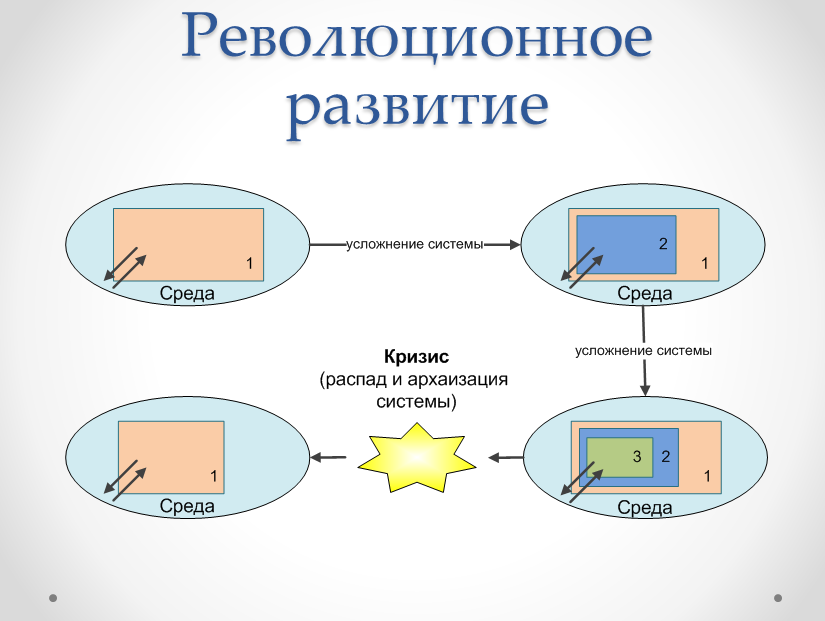




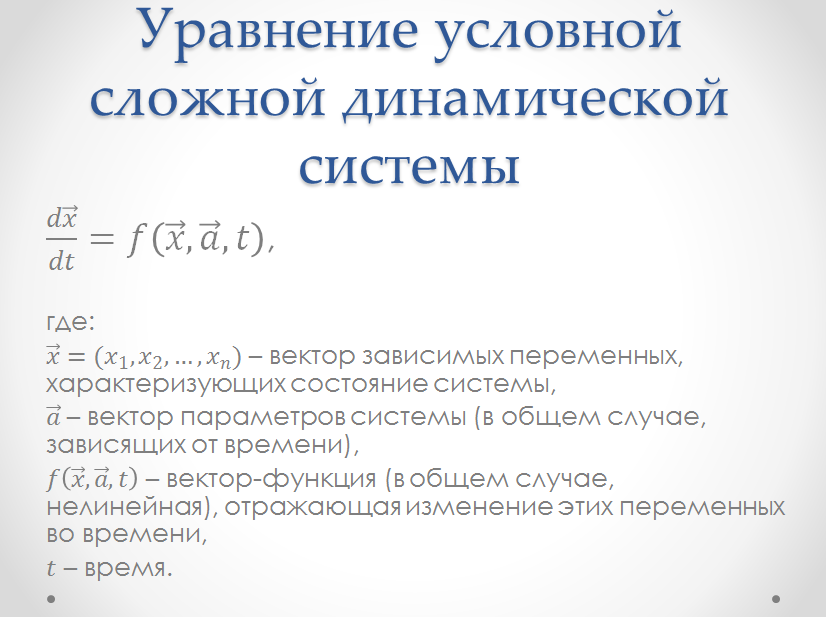
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



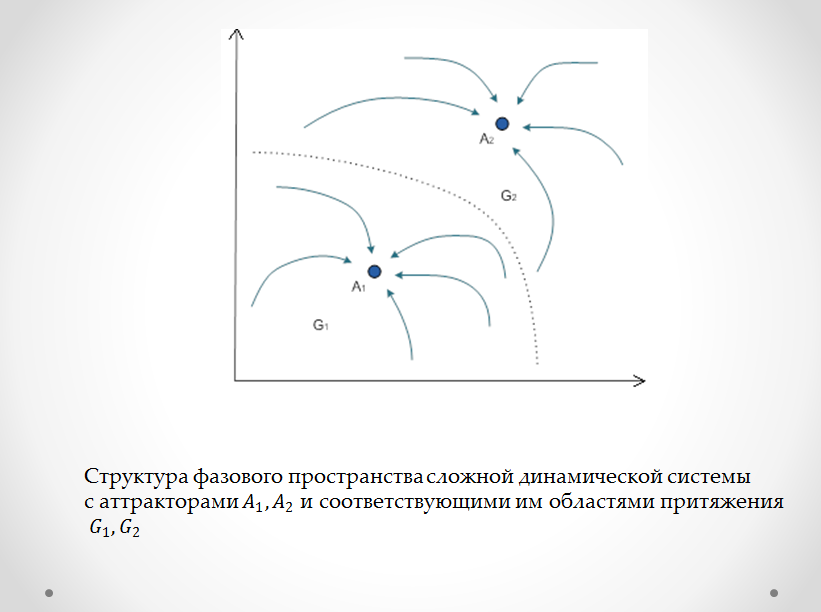
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



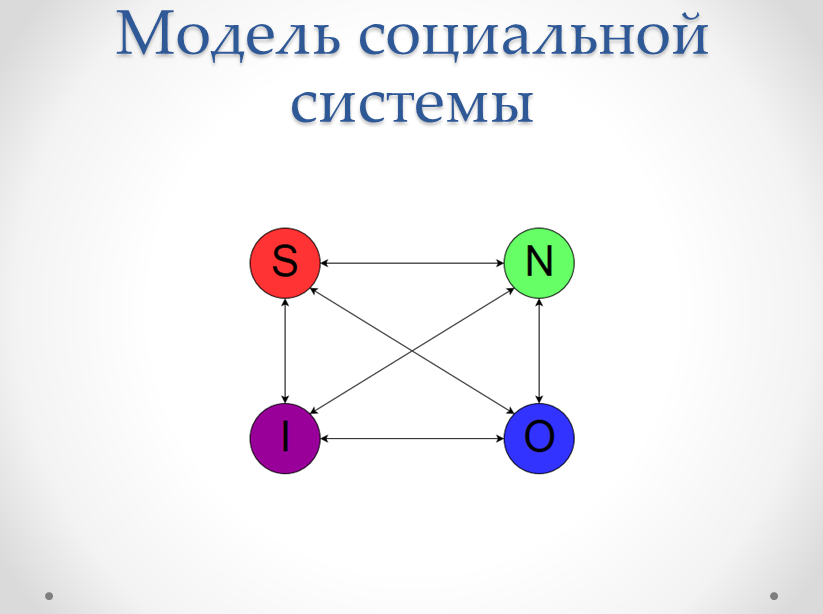
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



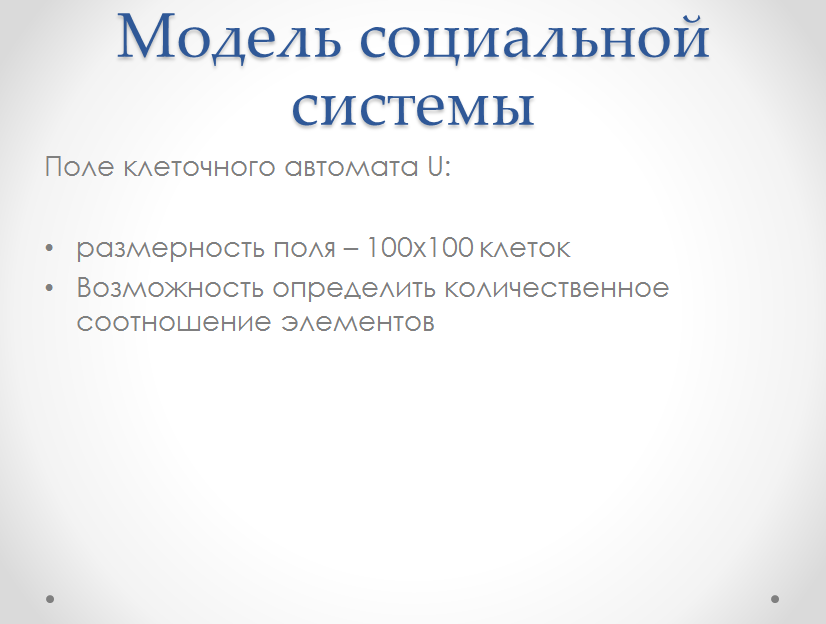
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



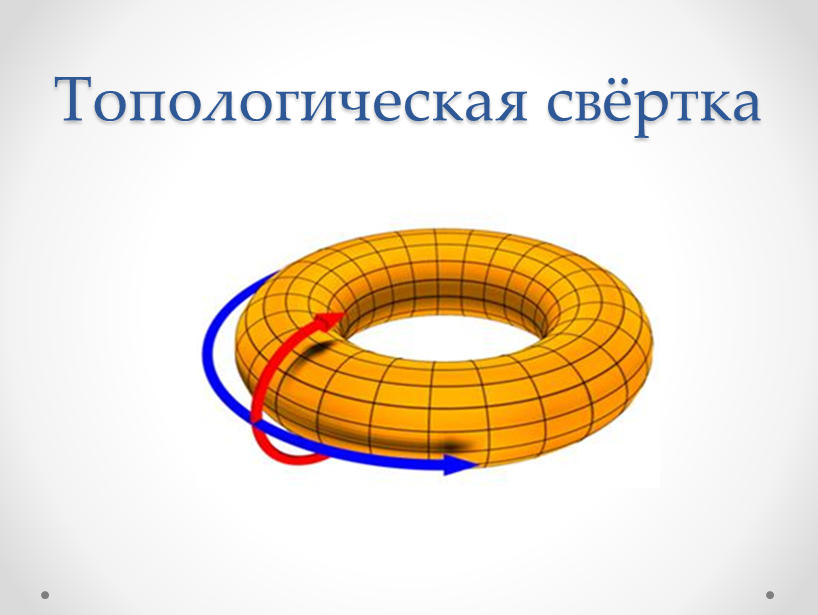
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



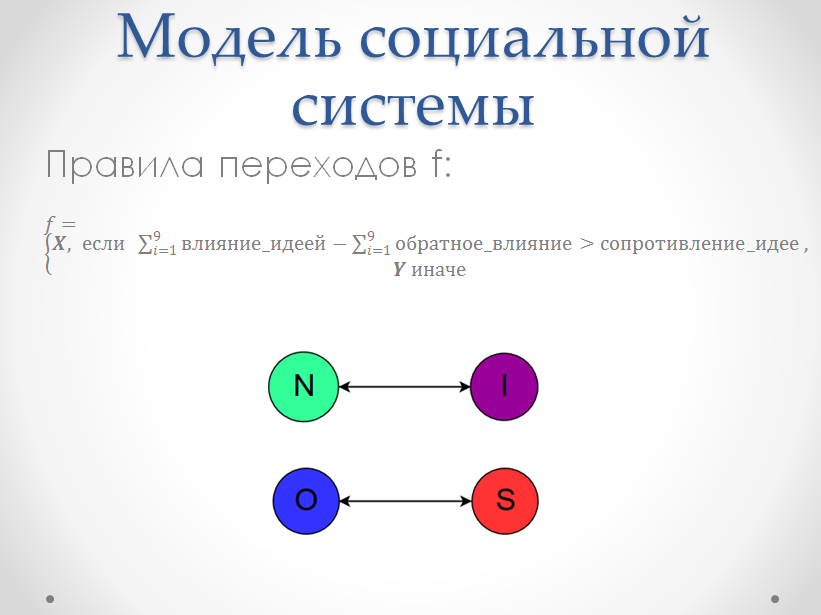
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



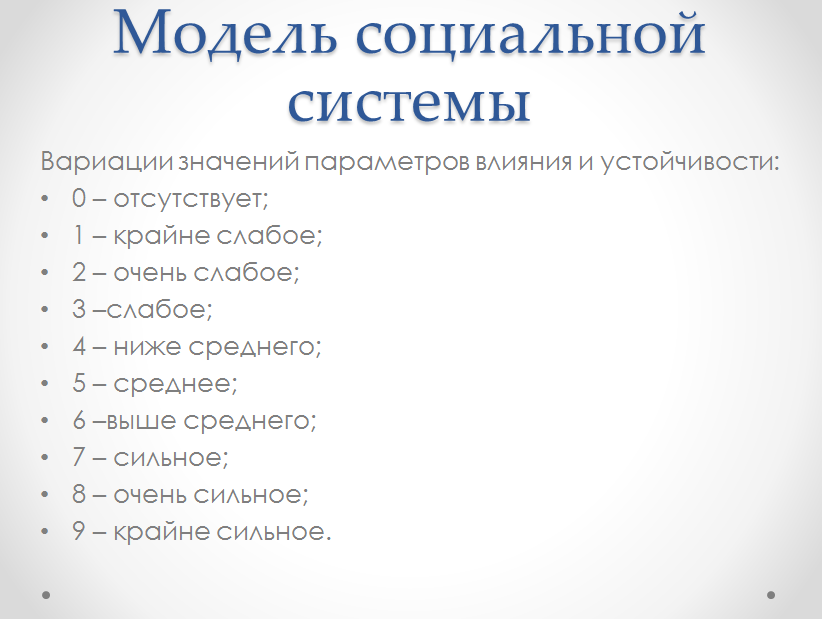
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



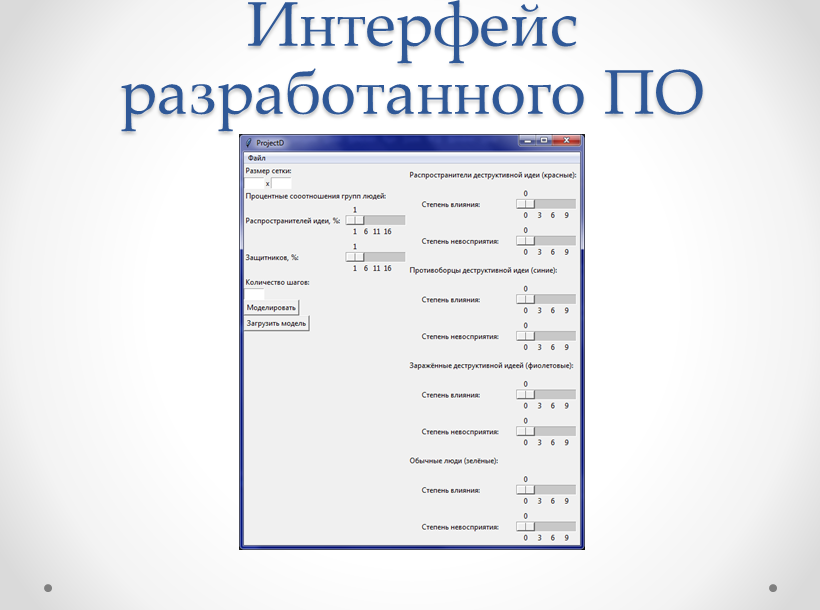
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



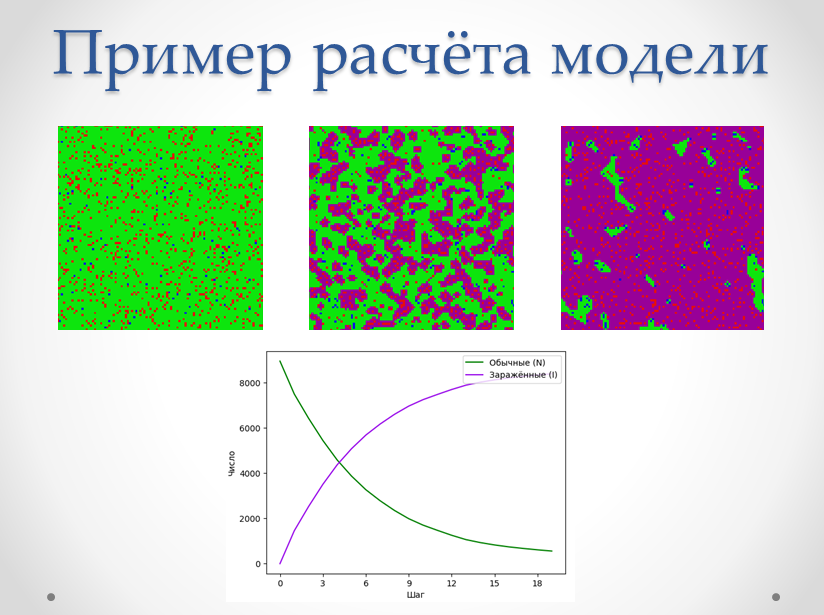
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



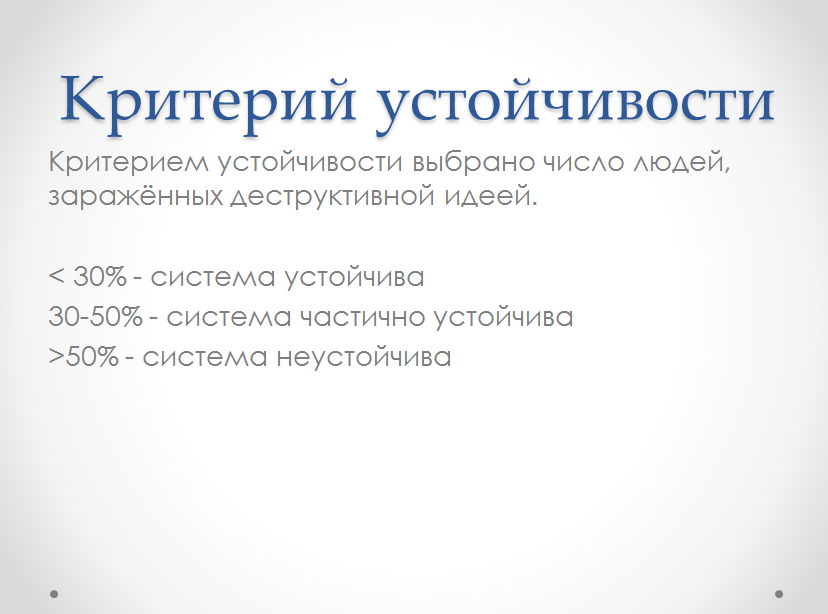
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



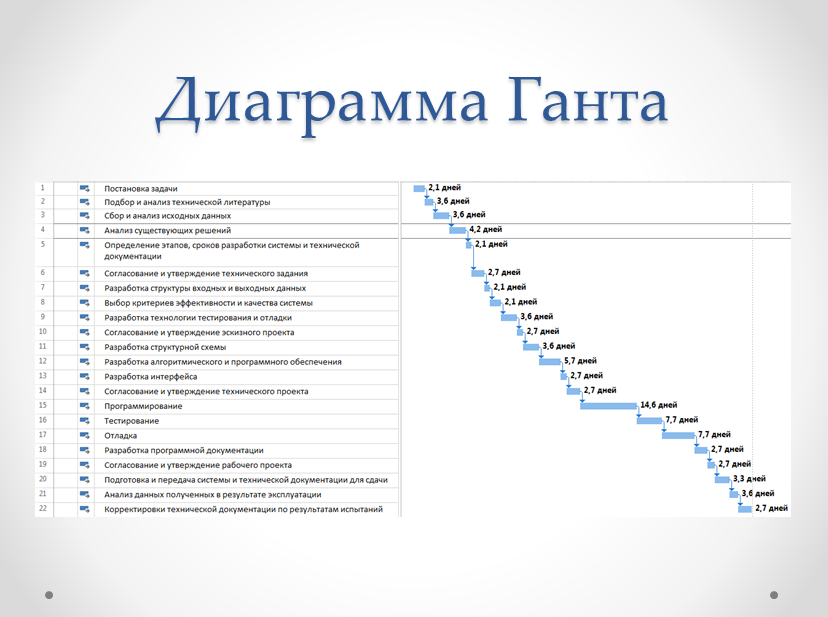
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



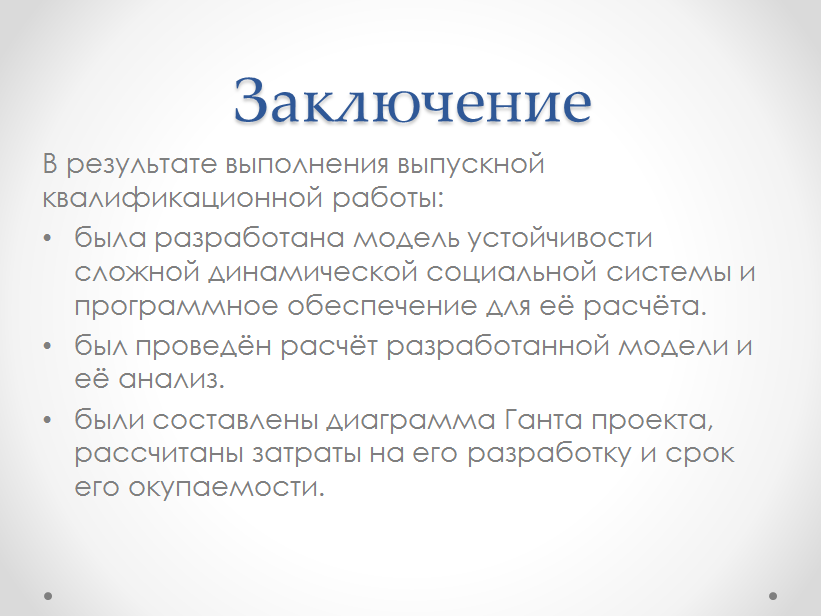
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм. | Лист | Ф.И.О. | Подпись | Дата |
|  | |  |  |  | Разработка программно-математических средств использования теории «управляемого хаоса» в управлении устойчивостью сложных динамических систем в условиях гибридной войны | Литера | | | Лист | Листов |
|  | |  |  |  |  |  |  |  |  |
|  | |  |  |  | РТУ МИРЭА | | | | |
|  | |  |  |  |
|  | |  |  |  |

# Приложение Б. Листинг программы

**Модуль main.py**

from projectD import \*

def main():

# main() method of project

# creates main window via TheD() class from projectD.py

root = Tk()

TheD(root).pack(side="top", fill="both", expand=True)

root.mainloop()

# launching the thing

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Модуль projectD.py**

from tkinter import \*

from tkinter import filedialog

from tkinter import messagebox

from miscellaneous import \*

from json import \*

import os.path

from modeling import \*

fig\_count = -1

class TheD(Frame, metaclass=Singleton):

# class for the main frame

def \_\_init\_\_(self, parent):

# initialization of the main frame

# frame's variables goes here

Frame.\_\_init\_\_(self, parent)

self.parent = parent

self.field\_height\_entry\_value = StringVar() # value for CA's field height

self.field\_width\_entry\_value = StringVar() # value for CA's field width

self.field\_step\_amount = StringVar() # value for amount of steps

self.spreaders\_percentage = IntVar() # value for idea's spreaders percentage

self.opposers\_percentage = IntVar() # value for idea's opposers percentage

# variables for sliders

self.s1 = NONE

self.s2 = NONE

self.s3 = NONE

self.s4 = NONE

self.s5 = NONE

self.s6 = NONE

self.s7 = NONE

self.s8 = NONE

self.f = None # pointer to the current settings' file

self.init\_ui() # initialization of main frame's UI

def init\_ui(self):

# initialization of main frame's UI

# frame's UI's elements goes here

self.parent.title("ProjectD") # setting window's title

# custom function for closing window

self.parent.protocol("WM\_DELETE\_WINDOW", self.close\_window)

# creating menu for file manipulation

menu\_bar = Menu(self.parent)

file\_menu = Menu(menu\_bar, tearoff=0)

file\_menu.add\_command(label="Новый файл", command=self.create\_file)

file\_menu.add\_command(label="Открыть", command=self.open\_file)

file\_menu.add\_command(label="Сохранить", command=self.save\_file)

file\_menu.add\_command(label="Сохранить как...", command=self.save\_file\_as)

menu\_bar.add\_cascade(label="Файл", menu=file\_menu)

self.parent.config(menu=menu\_bar)

row = 0 # row number

# setting's frame (left)

settings\_frame = Frame(self, height=200, width=150)

settings\_frame.grid(row=0, column=0, sticky='n')

# settings

# field's size

# label

Label(settings\_frame, text="Размер сетки: ").grid(row=row, sticky=W)

row += 1

# frame for size entries

size\_entries\_frame = Frame(settings\_frame)

reg\_field = size\_entries\_frame.register(is\_correct3) # register for validating field entries

size\_entries\_frame.grid(row=row, sticky=W)

Entry(size\_entries\_frame, width=5, textvariable=self.field\_height\_entry\_value,

validate='key', validatecommand=(reg\_field, '%P'))\

.grid(row=0, column=0, sticky=W)

Label(size\_entries\_frame, text="x", height=1).grid(row=0, column=1)

Entry(size\_entries\_frame, width=5, textvariable=self.field\_width\_entry\_value,

validate='key', validatecommand=(reg\_field, '%P'))\

.grid(row=0, column=2, sticky=E)

row += 1

col = 0

# faction's sizes (%)

Label(settings\_frame, text="Процентные сооотношения групп людей: ").grid(row=row, sticky=W)

row += 1

# faction sizes frame

faction\_sizes\_frame = Frame(settings\_frame)

faction\_sizes\_frame.grid(row=row, sticky=W)

Label(faction\_sizes\_frame, text="Распространителей идеи, %: ").grid(row=0, column=0, sticky=W)

Scale(faction\_sizes\_frame, from\_=1, to=20, resolution=1, variable=self.spreaders\_percentage,

orient=HORIZONTAL, tickinterval=5).grid(row=0, column=1)

Label(faction\_sizes\_frame, text="Защитников, %: ").grid(row=1, column=0, sticky=W)

Scale(faction\_sizes\_frame, from\_=1, to=20, resolution=1, variable=self.opposers\_percentage,

orient=HORIZONTAL, tickinterval=5).grid(row=1, column=1)

row += 1

# step amount entry

# label

Label(settings\_frame, text="Количество шагов: ").grid(row=row, sticky=W)

row += 1

reg\_steps = size\_entries\_frame.register(is\_correct2) # register for validating step entry

Entry(settings\_frame, width=5, textvariable=self.field\_step\_amount,

validate='key', validatecommand=(reg\_steps, '%P')) \

.grid(row=row, column=0, sticky=W)

row += 1

# model button frame

models\_frame = Frame(settings\_frame)

models\_frame.grid(row=row, sticky=W)

Button(models\_frame , text="Моделировать", command=self.create\_model)\

.grid(row=0, column=0, sticky=W)

Button(models\_frame, text="Загрузить модель", command=self.load\_model).\

grid(row=1, column=0, sticky=W)

row += 1

# model's frame

faction\_settings\_frame = Frame(self, height=200, width=200)

faction\_settings\_frame.grid(row=0, column=1)

row = 0

# Spreaders' frame

# label

Label(faction\_settings\_frame, text="Распространители деструктивной идеи (красные): ", height=2).grid(row=row, sticky=W)

row += 1

# spreaders entry frame

spreaders\_frame = Frame(faction\_settings\_frame)

spreaders\_frame.grid(row=row, sticky=W)

# influence entry

Label(spreaders\_frame, text="Степень влияния: ").grid(row=0, column=0, sticky=W, padx=20)

self.s1 = Scale(spreaders\_frame, from\_=0, to=9, resolution=1, orient=HORIZONTAL, tickinterval=3)

self.s1.grid(row=0, column=1, sticky=W)

row += 1

# resistance entry

Label(spreaders\_frame, text="Степень невосприятия: ").grid(row=1, column=0, sticky=W, padx=20)

self.s2 = Scale(spreaders\_frame, from\_=0, to=9, resolution=1, orient=HORIZONTAL, tickinterval=3)

self.s2.grid(row=1, column=1, sticky=W)

# self.s2.set(self.spreaders\_resistance)

row += 1

# opposers' frame

# label

Label(faction\_settings\_frame, text="Противоборцы деструктивной идеи (синие): ", height=2).grid(row=row, sticky=W)

row += 1

# opposers entry frame

opposers\_frame = Frame(faction\_settings\_frame)

opposers\_frame.grid(row=row, sticky=W)

# influence entry

Label(opposers\_frame, text="Степень влияния: ").grid(row=0, column=0, sticky=W, padx=20)

self.s3 = Scale(opposers\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s3.grid(row=0, column=1, sticky=W)

row += 1

# resistance entry

Label(opposers\_frame, text="Степень невосприятия: ").grid(row=1, column=0, sticky=W, padx=20)

self.s4 = Scale(opposers\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s4.grid(row=1, column=1, sticky=W)

# self.s4.set(self.opposers\_resistance)

row += 1

# Infected's frame

# label

Label(faction\_settings\_frame, text="Заражённые деструктивной идеей (фиолетовые): ", height=2).grid(row=row, sticky=W)

row += 1

# infected's entry frame

infected\_frame = Frame(faction\_settings\_frame)

infected\_frame.grid(row=row, sticky=W)

# influence entry

Label(infected\_frame, text="Степень влияния: ").grid(row=0, column=0, sticky=W, padx=20)

self.s5 = Scale(infected\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s5.grid(row=0, column=1, sticky=W)

row += 1

# resistance entry

Label(infected\_frame, text="Степень невосприятия: ").grid(row=1, column=0, sticky=W, padx=20)

self.s6 = Scale(infected\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s6.grid(row=1, column=1, sticky=W)

# self.s6.set(self.infected\_resistance)

row += 1

# Normal's frame

# label

Label(faction\_settings\_frame, text="Обычные люди (зелёные): ", height=2).grid(row=row, sticky=W)

row += 1

# normal's entry frame

normal\_frame = Frame(faction\_settings\_frame)

normal\_frame.grid(row=row, sticky=W)

# influence entry

Label(normal\_frame, text="Степень влияния: ").grid(row=0, column=0, sticky=W, padx=20)

self.s7 = Scale(normal\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s7.grid(row=0, column=1, sticky=W)

row += 1

# resistance entry

Label(normal\_frame, text="Степень невосприятия: ").grid(row=1, column=0, sticky=W, padx=20)

self.s8 = Scale(normal\_frame, from\_=0, to=9, resolution=1,

orient=HORIZONTAL, tickinterval=3)

self.s8.grid(row=1, column=1, sticky=W)

row += 1

def create\_model(self):

# getting model settings from form and calculating model with these parameters

# checking if dimensions are entered

if self.field\_height\_entry\_value.get() == '' or self.field\_width\_entry\_value.get() == '':

messagebox.showerror("Размерность", "Введите размерность поля!")

return

# checking if fields are not 0

if int(self.field\_height\_entry\_value.get()) == 0 or int(self.field\_width\_entry\_value.get()) == 0:

messagebox.showerror("Недопустимая размерность", "Размерность поля не может быть нулевой!")

return

# checking if step amount is entered

if self.field\_step\_amount.get() == '':

messagebox.showerror("Количество шагов", "Введите количество шагов!")

return

# checking is step count is above 0

if int(self.field\_step\_amount.get()) == 0:

messagebox.showerror("Некорректное количество шагов", "Число шагов должно быть больше 0!")

return

update\_model\_settings(self.get\_settings())

cellular\_automaton = init\_custom(int(self.field\_height\_entry\_value.get()),

int(self.field\_width\_entry\_value.get()),

s\_prob=self.spreaders\_percentage.get() / 100,

o\_prob=self.opposers\_percentage.get() / 100)

cellular\_automaton, demographics = evolve(cellular\_automaton, timesteps=int(self.field\_step\_amount.get()),

neighbourhood='Moore', apply\_rule=destructive\_distribution\_rule)

# saving calculated model

result = messagebox.askquestion("Сохранение модели", "Сохранить модель в файл?", icon='question')

if result == 'yes':

model\_file = filedialog.asksaveasfile(initialdir=os.path.dirname(os.path.abspath(\_\_file\_\_)),

mode='wb',

filetypes=[('Файлы моделей', '\*.model'), ("Все файлы", "\*.\*")],

initialfile="Новый файл модели", defaultextension=".model")

np.savez(model\_file, ca=cellular\_automaton, dem=demographics)

model\_file.close()

messagebox.showinfo("Сохранение файла", "Файл успешно сохранён.", icon='info')

name = 'Несохранённая конфигурация'

# if there is file from which configuration was loaded, then make it's name plot's title

if self.f is not None:

name, \_ = os.path.basename(self.f.name).split('.')

global fig\_count

fig\_count += 2

plot\_animate(cellular\_automaton, fig\_count, title=name, dem=demographics)

def load\_model(self):

# loading and animated model from file

model\_file = filedialog.askopenfile(initialdir=os.path.dirname(os.path.abspath(\_\_file\_\_)),

mode='rb',

filetypes=[('Файлы моделей', '\*.model'), ("Все файлы", "\*.\*")])

try:

model = np.load(model\_file)

except:

messagebox.showerror("Некорректный файл модели",

"Выбранный файл не является файлом модели либо содержит ошибки!")

return

if (type(model['ca']) == np.ndarray) and (type(model['dem'] == np.ndarray)):

name, \_ = os.path.basename(model\_file.name).split('.')

global fig\_count

fig\_count += 2

plot\_animate(model['ca'], fig\_count, name, model['dem'])

else:

messagebox.showerror("Некорректный файл модели",

"Выбранный файл не является файлом модели либо содержит ошибки!")

model\_file.close()

def handle\_unsaved\_file(self):

# handling unsaved file by offering to save it

result = messagebox.askquestion("Несохранённый файл", "Сохранить файл?", icon='question')

if result == 'yes':

if self.f is None:

self.save\_file\_as()

else:

self.save\_file()

messagebox.showinfo("Сохранение файла", "Файл успешно сохранён.", icon='info')

def create\_file(self):

# creating new settings' file, clearing window's parameters' values

self.handle\_unsaved\_file()

# resetting file pointer

self.f = None

# setting all parameters to 0

self.field\_height\_entry\_value.set(0)

self.field\_width\_entry\_value.set(0)

self.spreaders\_percentage.set(0)

self.opposers\_percentage.set(0)

self.s1.set(0)

self.s2.set(0)

self.s3.set(0)

self.s4.set(0)

self.s5.set(0)

self.s6.set(0)

self.s7.set(0)

self.s8.set(0)

def open\_file(self):

# opening settings' file

self.handle\_unsaved\_file()

self.f = filedialog.askopenfile(initialdir=os.path.dirname(os.path.abspath(\_\_file\_\_)),

mode='rt',

filetypes=[('Файлы конфигурации', '\*.json'), ("Все файлы", "\*.\*")])

if self.f is None: # askopenfile return `None` if dialog closed with "cancel"

return

# trying to load settings from file and putting them into form

try:

self.set\_settings(load(self.f))

except JSONDecodeError: # catching incorrect configuration file

messagebox.showerror("Ошибка", "Выбранный файл не является подходящим файлом конфигурации!")

self.f.close()

def save\_file(self):

# saving existing file

# if there is no current settings' file, invoke save\_file\_as() function

if self.f is None:

self.save\_file\_as()

else:

self.f = open(self.f.name, mode='w') # getting name from file pointer, opening file by his name

dump(self.get\_settings(), self.f)

self.f.close()

def save\_file\_as(self):

# saving file as new file

self.f = filedialog.asksaveasfile(initialdir=os.path.dirname(os.path.abspath(\_\_file\_\_)),

mode='w',

filetypes=[('Файлы конфигурации', '\*.json'), ("Все файлы", "\*.\*")],

initialfile="Новый файл конфигурации", defaultextension=".json")

if self.f is None: # asksaveasfile return `None` if dialog closed with "cancel"

return

dump(self.get\_settings(), self.f)

self.f.close()

def get\_settings(self):

# method for extracting parameters from form

# handling empty entries

if self.field\_height\_entry\_value.get() is '':

field\_height = 0

else:

field\_height = int(self.field\_height\_entry\_value.get())

if self.field\_width\_entry\_value.get() is '':

field\_width = 0

else:

field\_width = int(self.field\_width\_entry\_value.get())

# extracting values from corresponding sliders

settings = {'field\_height': field\_height,

'field\_width': field\_width,

'spreaders\_percentage': self.spreaders\_percentage.get(),

'opposers\_percentage': self.opposers\_percentage.get(),

'spreaders\_influence': self.s1.get(), 'spreaders\_resistance': self.s2.get(),

'opposers\_influence': self.s3.get(), 'opposers\_resistance': self.s4.get(),

'infected\_influence': self.s5.get(), 'infected\_resistance': self.s6.get(),

'normal\_influence': self.s7.get(), 'normal\_resistance': self.s8.get()}

return settings

def set\_settings(self, settings):

# method for verifying parameters and putting them into form

self.field\_height\_entry\_value.set(check\_input(settings['field\_height'], 0, 150))

self.field\_width\_entry\_value.set(check\_input(settings['field\_width'], 0, 150))

self.spreaders\_percentage.set(check\_input(settings['spreaders\_percentage'], 1, 20))

self.opposers\_percentage.set(check\_input(settings['opposers\_percentage'], 1, 20))

self.s1.set(check\_input(settings['spreaders\_influence'], 0, 9))

self.s2.set(check\_input(settings['spreaders\_resistance'], 0, 9))

self.s3.set(check\_input(settings['opposers\_influence'], 0, 9))

self.s4.set(check\_input(settings['opposers\_resistance'], 0, 9))

self.s5.set(check\_input(settings['infected\_influence'], 0, 9))

self.s6.set(check\_input(settings['infected\_resistance'], 0, 9))

self.s7.set(check\_input(settings['normal\_influence'], 0, 9))

self.s8.set(check\_input(settings['normal\_resistance'], 0, 9))

def close\_window(self):

self.handle\_unsaved\_file()

# method for closing window

self.parent.destroy()

def check\_input(value, left\_boundary, right\_boundary):

# checking if value is integer and whether it is in set boundaries

if not isinstance(value, int):

return 0

elif value < left\_boundary:

return left\_boundary

elif value > right\_boundary:

return right\_boundary

return value

def is\_correct2(value):

# function for checking correctness of steps' inputs

# it should be integers-only, no longer than 2 digits

if value.isdigit():

if len(value) < 3:

if value is 0:

return False

return True

else:

return False

elif value == "":

return True

else:

return False

def is\_correct3(value):

# function for checking correctness of dimensions' inputs

# it should be integers-only, no longer than 3 digits

if value.isdigit():

if len(value) < 4:

if int(value) > 150:

return False

return True

else:

return False

elif value == "":

return True

else:

return False

**Модуль modeling.py**

import numpy as np

from random import random

from matplotlib.colors import ListedColormap

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib.ticker import MaxNLocator

model\_settings = {} # global variable for model settings

def init\_custom(rows, cols, s\_prob=0.1, o\_prob=0.1):

"""

Returns matrix initiated with custom distribution of "o" (0) and "s" (3) groups

with values consisting of numbers in {0,...,3}.

:param rows: the number of rows in the matrix

:param cols: the number of columns in the matrix

:param k: the number of states in the cellular automaton (2, by default)

:param s\_prob: probability of spreader's spawn (0.1, by default)

:param o\_prob: probability of opposer's spawn (0.1, by default)

:return: a tensor with shape (1, rows, cols), randomly initialized with numbers in {0,...,k - 1}

"""

random\_numbers = []

for r in range(rows):

random\_row = []

for c in range(cols):

rnd = random()

if rnd < o\_prob:

random\_row.append(0)

elif rnd < o\_prob+s\_prob:

random\_row.append(3)

else:

random\_row.append(1)

random\_numbers.append(random\_row)

return np.array([random\_numbers])

def evolve(cellular\_automaton, timesteps, apply\_rule, r=1, neighbourhood='Moore'):

"""

:param cellular\_automaton:

:param timesteps: the number of time steps in this evolution; note that this value refers to the total number of

time steps in this cellular automaton evolution, which includes the initial condition

:param apply\_rule: a function representing the rule to be applied to each cell during the evolution; this function

will be given three arguments, in the following order: the neighbourhood, which is a numpy

2D array of dimensions 2r+1 x 2r+1, representing the neighbourhood of the cell (if the

'von Neumann' neighbourhood is specified, the array will be a masked array); the cell identity,

which is a tuple representing the row and column indices of the cell in the cellular automaton

matrix, as (row, col); the time step, which is a scalar representing the time step in the

evolution

:param r: the neighbourhood radius; the neighbourhood dimensions will be 2r+1 x 2r+1

:param neighbourhood: the neighbourhood type; valid values are 'Moore' or 'von Neumann'

:return:

"""

\_, rows, cols = cellular\_automaton.shape

array = np.zeros((timesteps, rows, cols), dtype=cellular\_automaton.dtype)

array[0] = cellular\_automaton

# demographics for initial state

demographics = np.zeros((2, timesteps)) # variable for N/I ration graphic

n\_amount = 0

i\_amount = 0

for x in cellular\_automaton[0]:

for y in x:

if y == 1:

n\_amount += 1

demographics[0][0] = n\_amount

demographics[1][0] = i\_amount

von\_neumann\_mask = np.zeros((2\*r + 1, 2\*r + 1), dtype=bool)

for i in range(len(von\_neumann\_mask)):

mask\_size = np.absolute(r - i)

von\_neumann\_mask[i][:mask\_size] = 1

if mask\_size != 0:

von\_neumann\_mask[i][-mask\_size:] = 1

def get\_neighbourhood(cell\_layer, row, col):

row\_indices = range(row - r, row + r + 1)

row\_indices = [i - cell\_layer.shape[0] if i > (cell\_layer.shape[0] - 1) else i for i in row\_indices]

col\_indices = range(col - r, col + r + 1)

col\_indices = [i - cell\_layer.shape[1] if i > (cell\_layer.shape[1] - 1) else i for i in col\_indices]

n = cell\_layer[np.ix\_(row\_indices, col\_indices)]

if neighbourhood == 'Moore':

return n

elif neighbourhood == 'von Neumann':

return np.ma.masked\_array(n, von\_neumann\_mask)

else:

raise Exception("unknown neighbourhood type: %s" % neighbourhood)

for t in range(1, timesteps):

cell\_layer = array[t - 1]

n\_amount = 0

i\_amount = 0

for row, cell\_row in enumerate(cell\_layer):

for col, cell in enumerate(cell\_row):

n = get\_neighbourhood(cell\_layer, row, col)

array[t][row][col] = apply\_rule(n, (row, col), t)

if array[t][row][col] == 1:

n\_amount += 1

elif array[t][row][col] == 2:

i\_amount += 1

demographics[0][t] = n\_amount

demographics[1][t] = i\_amount

return array, demographics

def update\_model\_settings(settings):

# method for updating global variable of model settings

global model\_settings

model\_settings = settings

def update\_demographics(value):

# method for updating global variable of model settings

global demographics

demographics = value

def destructive\_distribution\_rule(neighbourhood, c, t):

"""

:param neighbourhood: numpy array of dimensions 2r+1 x 2r+1, representing the neighbourhood

:param c: the cell identity, which is a tuple representing the row and column indices of the cell in the cellular

automaton matrix, as (row, col)

:param t: the time step, which is a scalar representing the time step in the evolution

:return: cell's new state

"""

# total values of neighbours' influence

total\_destruction\_influence = 0

total\_anti\_influence = 0

global model\_settings # using set on the form model settings

for neighbour\_row in neighbourhood:

for neighbour in neighbour\_row:

if neighbour == 0: # opposers

total\_anti\_influence += model\_settings['opposers\_influence']

elif neighbour == 1: # normal

total\_anti\_influence += model\_settings['normal\_influence']

elif neighbour == 2: # infected

total\_destruction\_influence += model\_settings['infected\_influence']

elif neighbour == 3: # supporters

total\_destruction\_influence += model\_settings['spreaders\_influence']

else:

pass

if neighbourhood[1][1] == 0:

return 3 if total\_destruction\_influence - total\_anti\_influence > model\_settings['opposers\_resistance'] else 0

if neighbourhood[1][1] == 1:

return 2 if total\_destruction\_influence - total\_anti\_influence > model\_settings['normal\_resistance'] else 1

if neighbourhood[1][1] == 2:

return 1 if total\_anti\_influence - total\_destruction\_influence > model\_settings['infected\_resistance'] else 2

if neighbourhood[1][1] == 3:

return 0 if total\_anti\_influence - total\_destruction\_influence > model\_settings['spreaders\_resistance'] else 3

return -1

# creating new color scheme, which is corresponding to paper's model group's colors

N = 4

values = np.ones((N, 4))

values[0] = np.array([14/256, 14/256, 230/256, 1])

values[1] = np.array([14/256, 230/256, 14/256, 1])

values[2] = np.array([153/256, 0/256, 153/256, 1])

values[3] = np.array([230/256, 14/256, 14/256, 1])

newcmp = ListedColormap(values)

def plot\_animate(ca, fig\_count, title='', dem=None):

"""

:param ca: array of all CA's states

:param fig\_count: index of new figure

:param title: figure title

:param dem: array of demographics

:return:

"""

cmap = plt.get\_cmap(newcmp)

fig = plt.figure(fig\_count - 1)

fig.canvas.set\_window\_title(title)

im = plt.imshow(ca[0], animated=True, cmap=cmap)

i = {'index': 0}

def updatefig(\*args):

i['index'] += 1

if i['index'] == len(ca):

i['index'] = 0

im.set\_array(ca[i['index']])

return im,

ani = animation.FuncAnimation(fig, updatefig, interval=100, blit=True)

if dem is not None:

plot\_demographics(dem, fig\_count, 'Соотношение Обычных и Заражённых людей для ' + title)

plt.show()

def plot\_demographics(dem\_value, fig\_count, title=''):

time = np.arange(0, len(dem\_value[0]))

fig\_number = fig\_count

plt.figure(fig\_count)

plt.figure(fig\_count).canvas.set\_window\_title(title)

lines = plt.plot(time, dem\_value[0], time, dem\_value[1])

plt.setp(lines[0], linestyle='-', color='green')

plt.setp(lines[1], linestyle='-', color='xkcd:violet')

plt.ylabel('Число')

plt.xlabel('Шаг')

ax = plt.figure(fig\_count).gca()

ax.xaxis.set\_major\_locator(MaxNLocator(integer=True))

plt.legend(('Обычные (N)', 'Заражённые (I)'), loc='upper right')

plt.show()

**Модуль miscellaneous.py**

class Singleton(type):

# Singleton class

# for main window

\_instances = {}

def \_\_call\_\_(cls, \*args, \*\*kwargs):

if cls not in cls.\_instances:

cls.\_instances[cls] = super(Singleton, cls).\_\_call\_\_(\*args, \*kwargs)

return cls.\_instances[cls]