Довідка
про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України
Український державний університет науки та технологій

Кафедра «Комп'ютерні інформаційні технології»

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти
Мурковича Миколи Сергійовича
(прізвище, ім'я, по батькові)
на тему: *Методи поетапного моделювання складних процесів*
в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР _____ Олександра ГОРБОВА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /Вадим ГОРЯЧКІН/

«20» 12 2021 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Методи поетапного моделювання складних процесів**

Theme **Methods of step-by-step modeling of complex processes**

Керівник дипломної роботи доц. _____ Олександра ГОРБОВА

Нормоконтролер доц. _____ Олена КУРОПЯТНИК

Студент групи ПЗ2021 _____ Микола МУРКОВИЧ

Student Mykola MURKOVYCH

Дніпро – 2021

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Комп'ютерних технологій і систем
Кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. В. І.Шинкаренко
(підпис)
«18» жовтня 2021 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС магістр
(освітній ступінь)

студента групи ПЗ2021 М.С. Муркович
(номер групи) (ПІБ)

1 Тема дипломної роботи: Методи поетапного моделювання складних процесів
затверджена наказом по університету від «18» листопада 2020 р. № 690.

2 Термін подання студентом закінченої роботи 3 грудня 2021 р.

3 Вихідні дані до дипломної роботи набір правил переходів, набір фатів о стані процесу або середовища, інформація о часі затраченому на переходи з одного стану в інший

4 Зміст пояснювальної записки (перелік питань до розробки) аналіз сучасного стану дослідження та програмно-апаратного забезпечення, обґрунтування експериментального методу дослідження складних технологічних процесів за допомогою поетапного моделювання, проектування й розробка інструментального забезпечення для дослідження методу поетапного моделювання складних процесів, дослідження можливості використання методу поетапного моделювання за допомогою декомпозиції та реалізацією на мовах програмування c# та prolog для моделювання складних процесів, охорона праці та безпека в надзвичайних ситуаціях

5 Перелік демонстраційного матеріалу презентація, доповідь, демонстраційне відео

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	М.В. Гненний	26.10.21	23.10.21
Охорона праці	О.І. Саблін	26.10.21	24.11.21

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	При-мітка
1	Вступ	1.09.21	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	2.09.21 – 15.09.21	від 70 джерел
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	16.09.21 – 10.10.21	
4	Постановка задачі, технічне завдання	11.10.21 – 17.10.21	30%
5	Техніко-економічні показники	18.10.21 – 19.10.21	
6	Розробка інструментальних засобів дослідження	20.10.21 – 7.11.21	
7	Виконання досліджень	8.11.21 – 14.11.21	60%
8	Оформлення тез доповідей	15.11.21 – 18.11.21	
9	Оформлення статті у фаховий журнал	19.11.21 – 22.11.21	
10	Оформлення пояснювальної записки	23.11.21 – 28.11.21	
11	Розробка демонстраційних матеріалів	29.11.21 – 5.12.21	100%

Дата видачі завдання «19» листопада 2020р.

Керівник дипломної роботи _____ О.В. Горбова
(підпис) (ПІБ)

Завдання прийняв до виконання _____ М.С. Муркович
(підпис) (ПІБ)

РЕФЕРАТ

Темою магістерської роботи є: «Методи поетапного моделювання складних процесів».

Наукова новизна. Наукова новизна роботи полягає у:

– вперше були використані методи поетапного моделювання з такими набором етапів та таким видом реалізації як, фізичне моделювання, математичне моделювання та імітаційне моделювання;

– застосуванні загальної методології при моделюванні процесів за допомогою методу поетапного моделювання;

– удосконаленні формалізації методу поетапного моделювання;

– представленні процесів за допомогою побудови дерева рішень та аналізу часу його виконання в умовах де можуть відбуватися випадкові події що будуть впливати на час.

Методи дослідження. При моделюванні складного технологічного процесу виділяються етапи фізичного, математичного та імітаційного моделювання.

Дипломний проект: пояснювальна записка містить 87 стор., 35 рисунок, 6 таблиць, 29 літературних джерел та 6 додатків.

Об’єктом дослідження є підходи та методики моделювання процесів та систем у часі.

Мета і задачі дослідження. Метою магістерської роботи – моделювання складних процесів на основі методики поетапного моделювання шляхом декомпозиції складних процесів. Поставлена мета досягається в результаті розв’язання таких завдань:

– вивчення різних підходів моделювання складних процесів та систем на основі поетапного моделювання;

– розробка методики моделювання складних процесів та систем на основі поетапного моделювання;

– формалізації процесу дослідження алгоритму для моделювання імітаційної системи для розрахунку часової складової для обраних об’єктів складних систем.

Одержані результати – розроблений інструментальний засіб, що дозволяє проводити моделювання складних процесів за допомогою методу поетапного моделювання, проведено дослідження що підтверджує можливість використання відповідного підходу моделювання.

Ключові слова: *поетапне моделювання, метод декомпозиції, складні технічні та технологічні процеси, Visual Studio, Prolog, логічне програмування, інтерфейс, статистична інформація, випадкові події, розрахунок часу*.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

ПП – програмний продукт;

БД – база даних;

ЕОМ – електрона обчислювальна машина;

ПК – персональний комп’ютер;

НПАОП – нормативно-правові акти з охорони праці;

ДСанПіН – державні санітарні правила та норми.

ВСТУП

Актуальність теми. В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області. Під час моделювання технологічних процесів застосовують різні методи й підходи.

Розвиток та ускладнення або навпаки спрощення технологій та технологічних процесів з часом призводить до нагромадження різних зайвих процедур, дій, зайвої роботи або навпаки може бути дещо спрощено, для цього необхідно постійно слідкувати за всією системою. Для поліпшення цього процесу та надання більшої наочності складним процесам та системам необхідно дослідити різні підходи та методики розробки моделювання методом поетапного моделювання.

Побудова моделей технологічних процесів є досить важливим завданням, тому що дозволяє передбачити їх особливості, що сприяють досягненню необхідних характеристик та властивостей. Моделювання передбачає побудову такого опису об'єкта, яке із заданою точністю буде збігатися з функціонуванням реального процесу.

Побудова дерева рішення дозволяє покращити оптимізацію складних процесів, шляхом поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі. Це дозволяє збільшити ефективність реалізації цих процесів у кілька разів і помітно поліпшити якість кінцевого продукту.

При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання спрощення процесу моделювання. На базі формалізованого підходу до побудови і дослідження моделей є доцільним створення програмного комплексу, що буде допомагати та прискорить діяльність фахівців.

У зв’язку з усім, що було перераховано вище, тема роботи, що спрямована на розв’язання вказаних завдань – є актуальною.

Об’єктом дослідження є підходи та методики моделювання процесів та систем у часі.

Предметом дослідження є методологія дослідження середніх витрат часу всіх технологічних ланок у складних системах та процесах.

Мета і задачі дослідження. Метою магістерської роботи – моделювання складних процесів на основі методики поетапного моделювання шляхом декомпозиції складних процесів. Поставлена мета досягається в результаті розв’язання таких завдань:

– вивчення різних підходів моделювання складних процесів та систем на основі поетапного моделювання;

– розробка методики моделювання складних процесів та систем на основі поетапного моделювання;

– формалізації процесу дослідження алгоритму для моделювання імітаційної системи для розрахунку часової складової для обраних об’єктів складних систем.

Методи дослідження. При моделюванні складного технологічного процесу виділяються етапи фізичного, математичного та імітаційного моделювання.

Перший етап являє собою не більш ніж збір статистичної інформації та фактичної інформації о стані об’єктів моделі з фізично існуючої та функціонуючої моделі або використання поточної інформації що може бути отримана в режимі реального часу, також можливе використання інформації яка будуються на припущеннях для тих моделей процесів що ще не існують, або знаходяться в такому стані з котрого не можливо провести фізичний збір інформації або перевірити стан об’єктів моделі.

Другий етап характеризується створенням математичної моделі на основі вхідних даних, котрі можуть представляти з себе опис об’єктів

дослідження та їх вхідний стан або це також може бути декілька таких наборів об’єктів.

Етап імітаційного моделювання пов'язаний з дослідженням часової характеристики реалізованого на попередньому етапі дерева рішення. Застосування статистичних та аналітичних інструментів обробки результатів чисельних експериментів дозволяє отримати не тільки окремі результати функціонування моделі технологічного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе процес ітеративного запуску отриманої моделі, що являє собою дерево рішень, з однаковими вхідними параметрами але з можливістю того що може трапитись якась випадкова подія з тих що передбачені та закладені перед запуском програмного комплексу, що буде впливати на час виконання процесу. Отримані результати підлягають подальшій обробці для виявлення слабких місць процесу, функціональних залежностей і узагальнення результатів.

Таким чином, поставлена задача вирішується за допомогою формалізації складного процесу та імітаційного моделювання, а також дослідження результатів моделювання складних процесів. Це може бути використано при розробленні та/або удосконаленні складних процесів, що можуть бути представлені у вигляді наборів фактів.

Наукова новизна. Наукова новизна роботи полягає у:

– вперше були використані методи поетапного моделювання з такими набором етапів та таким видом реалізації як, фізичне моделювання, математичне моделювання та імітаційне моделювання;

– застосуванні загальної методології при моделюванні процесів за допомогою методу поетапного моделювання;

– удосконаленні формалізації методу поетапного моделювання;

– представленні процесів за допомогою побудови дерева рішень та аналізу часу його виконання в умовах де можуть відбуватися випадкові події що будуть впливати на час.

Практичне значення. Результати виконаних досліджень та впровадження запропонованого методу може бути використано при розробці, удосконаленні та/або формалізації, моделюванні складних технологічних процесів, устрою систем та структур підприємства або організації.

Апробація результатів дослідження та публікації. Основні положення магістерської роботи доповідалися та були схвалені на XIV міжнародній науково-практичній конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті», яка відбулася в Дніпровському національному університеті залізничного транспорту імені академіка В. Лазаряна (ДНУЗТ) 15 – 16 грудня 2020 року, див. додаток В.

На основні положень магістерської роботи було створено тези доповіді, котрі були схвалені на 81 міжнародній науково-практичній конференції "Проблеми та перспективи розвитку залізничного транспорту", яка відбулася в Дніпровському національному університеті залізничного транспорту імені академіка В. Лазаряна (ДНУЗТ) 22 – 23 квітня 2021 року та на 81 всеукраїнській науково-технічній конференції молодих учених, магістрантів та студентів «Наука і сталий розвиток транспорту», яка відбулася в Дніпровському національному університеті залізничного транспорту імені академіка В. Лазаряна (ДНУЗТ) 28 жовтня 2021 року, див. додаток Г та додаток Д.

Опубліковано наукову статтю «МОДЕЛЮВАННЯ СКЛАДНИХ ПРОЦЕСІВ НА ОСНОВІ ПОЕТАПНОГО МОДЕЛЮВАННЯ», «Вісник Дніпропетровського національного університету залізничного транспорту» (подана до друку), див. додаток Е.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ ТА ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз предметної області

1.1.1 Опис проблеми. Актуальність дослідження

В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області, процесів або складових системи. Під час моделювання технологічних процесів застосовують різні методи й підходи.

Розвиток та ускладнення або навпаки спрощення технологій та технологічних процесів з часом призводить до нагромадження різних зайвих процедур, дій, зайвої роботи або навпаки може бути дещо спрощено, для цього необхідно постійно слідкувати за всією системою. Для поліпшення цього процесу та надання більшої наочності складним процесам та системам необхідно дослідити різні підходи та методики розробки моделювання методом поетапного моделювання.

Побудова моделей технологічних процесів є досить важливим завданням, тому що дозволяє передбачити їх особливості, що сприяють досягненню необхідних характеристик та властивостей. Моделювання передбачає побудову такого опису об'єкта, яке із заданою точністю буде збігатися з функціонуванням реального процесу.

Побудова дерева рішення дозволяє покращити оптимізацію складних процесів, шляхом поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі. Це дозволяє збільшити ефективність реалізації цих процесів у кілька разів шляхом знаходження слабких місць у цьому процесі.

При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання формалізації процесу моделювання. На базі

формалізованого підходу до побудови і дослідження моделей є доцільним створення програмного комплексу, що автоматизує і впорядковує діяльність фахівців.

У зв’язку з усім, що було перераховано вище, тема роботи, що спрямована на розв’язання вказаних завдань – є актуальною. Також доцільним є необхідність перевірки коректності використання такого моделювання у різних сферах.

1.1.2 Огляд останніх досліджень і публікацій

Одне з останніх досліджень та статей у сфері моделювання технологічних процесів, є стаття – котра присвячена питанням формалізації і автоматизації процесу синтезу моделей систем автоматичного управління технологічними процесами. Пропонується підхід поетапної побудови моделі, що включає такі етапи, як фізичне моделювання, математичне моделювання, дискретне комп'ютерне моделювання та імітаційне моделювання, див рис. 1.1 [1]. На кожному з етапів передбачається активна взаємодія з базою знань з метою накопичення типових рішень їх подальшого використання. Для синтезу моделей об'єктів і систем управління, а також для адаптації типових рішень пропонується застосування інтелектуального структурно-параметричного синтезу, заснованого на ідеї використання еволюційних алгоритмів як засоби автоматичної генерації моделі у вигляді мережі Петрі. Запропонований метод реалізується у вигляді прикладного програмного комплексу, призначеного для автоматизації науково-дослідних і конструкторських робіт.

Для кращого уявлення сутності процесу моделювання слід виділити типові етапи, а також завдання та механізми, використовувані на них.

Більшість дослідницьких робіт, пов'язаних з побудовою та використанням моделей технологічних процесів, піддаються поділу на схожі етапи. Типова послідовність робіт має на увазі лінійне проходження наступних фаз: фізичне моделювання, математичне моделювання, дискретне комп'ютерне моделювання, імітаційне моделювання.

Рисунок 1.1 – Схема поетапного моделювання

На етапі фізичного моделювання проводиться дослідження цікавить технологічного процесу. Для цього може бути проведений ряд експериментів з існуючим обладнанням. При відсутності такої можливості створюються фізичні моделі, що відображають основні закономірності досліджуваних процесів.

Основною метою побудови фізичної моделі є отримання статичних і динамічних характеристик процесу, що модулюється. Таким чином, реалізована і вивчена фізична модель дає можливість формально описати властивості досліджуваного процесу, визначити чисельні значення актуальних параметрів, оцінити їх кореляцію і динаміку зміни.

Після збору і обробки даних про функціонування фізичної моделі настає етап математичного моделювання. Залежно від особливостей технологічного процесу і цілей моделювання можуть бути обрані різні математичні апарати.

У разі прикладних розробок важливим фактором є трудомісткість. Часто буває доцільним використовувати свідомо менш точну методологію моделювання, проте що є простіший у використанні.

У разі моделювання схожих процесів доцільно використання шаблонних підходів, що дають приріст швидкості синтезу моделі. В цьому випадку виникає задача зберігання бази шаблонів і вибору необхідного шаблону моделі за заданими критеріями. завдяки застосуванню систем управління базами даних і базами знань можливе створення автоматизованої системи, що пропонує на етапі математичного моделювання найбільш відповідну методологію і шаблон, використовуючи які створюється математичне опис досліджуваного процесу.

У деяких випадках побудова математичної моделі може бути ускладнене в зв'язку зі складністю опису досліджуваного об'єкта. У цьому випадку може бути застосований підхід «чорного ящика», коли модель синтезується лише на підставі накопиченої статистичної інформації про поведінку об'єкта. для автоматизації цього процесу пропонується використання еволюційного алгоритму як засобу спрямованого пошуку структури і параметрів моделі, адекватно описує отримані раніше вихідні емпіричні дані.

У зв'язку з тим, що створення кошти інтелектуального синтезу, яке могло б працювати з різними математичними апаратами, представляється скрутним, то для спрощення пропонується використання тільки математичного апарату мереж Петрі.

Автоматизований структурно-параметричний синтез дозволить не тільки створювати нові моделі за вихідними даними, але і уточнювати (адаптувати) вже наявні або спрощені моделі з метою підвищення їх адекватності та точності. В цьому випадку можна говорити про підхід «сірого ящика», коли в якості вихідних даних для інтелектуального синтезу моделі крім емпіричних відомостей про вхідних і вихідних сигналах використовується узагальнена або спрощена модель досліджуваного об'єкта.

Дискретне комп'ютерне моделювання стало практично, обов'язковим етапом процесу синтезу та аналізу моделей технологічних процесів. Це пов'язано з тим, що чисельне моделювання в порівнянні з аналітичним аналізом моделей дозволяє проводити дослідження моделі значно швидше, а в деяких випадках і точніше. Крім того, коригування параметрів і модифікація структури моделі при чисельній комп'ютерної реалізації відбувається значно оперативніше.

Реалізація дискретної чисельної моделі повинна проводитися з урахуванням особливостей переходу від математичної моделі до програмного алгоритму.

В першу чергу це пов'язано з процесом дискретизації, який може привести не тільки до появи похибок, але і до функціональних відмінностей з вихідної математичною моделлю (наприклад, втрата стійкості). Перевірку коректності переходу від математичної моделі до дискретного комп'ютерного поданням технологічного процесу можна представити в якості типових методик, що містять в собі ряд формальних операцій зіставлення чисельних і функціональних характеристик моделі до і після дискретизації. Процес застосування цих методик також підлягає автоматизації завдяки використанню програмних засобів підтримки наукових досліджень.

Етап імітаційного моделювання пов'язаний з дослідженням властивостей реалізованої на попередньому етапі дискретної комп'ютерної моделі. Застосування статистичних та аналітичних інструментів обробки результатів чисельних експериментів, а також їх планування дозволяє отримати не тільки окремі чисельні результати функціонування моделі технологічного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе ітеративний процес запуску дискретної комп'ютерної моделі з різними наборами вихідних даних.

Отримані результати підлягають подальшій обробці для виявлення функціональних залежностей і узагальнення результатів.

В узагальнення про дане дослідження можливо зробити висновок про схожість основної структури програмних засобів даної категорії, але і про різницю в підходах в реалізації та властивостях програмних засобів.

Інша стаття о моделюванні складних процесів соціальної взаємодії [2]. В даній статті показано, що для дослідження складних процесів соціальної взаємодії на сучасному етапі розвитку системного моделювання все частіше використовуються ієрархічні системи когнітивних моделей, когнітивні архітектури та інтегровані системи моделювання. Що стосується даним процесам система моделей повинна відображати різні абстракції опису структури, різноманітні аспекти її поведінки, етапи (ітерації) її еволюції в процесі функціонування і розвитку. Пропонується розглядати архітектуру сукупності моделей в контексті необхідності для вирішення завдання на конкретному етапі дослідження.

Як показує практика системного моделювання найбільш значущими і, в той же час, найбільш складними є завдання аналізу і прогнозування розвитку процесів соціальної взаємодії. Такого роду завдання пов'язані з прогнозом досягнення довгострокових цілей шляхом адаптації до змін зовнішнього середовища. Завдання складні і вимагають врахування великої кількості факторів, інтересів, ризиків і наслідків, в їх рішеннях присутня висока ступінь невизначеності в оцінці зовнішнього середовища, слабка формалізація методів управління і широке використання експертних оцінок і знань, багатокритеріальність при оцінці прийнятих рішень. Характеризуючи проблему модельного підходу до аналізу та прогнозування складних процесів соціальної взаємодії, необхідно враховувати, що від постановки задачі моделювання до інтерпретації отриманих результатів, існує велика група складних науково-технічних проблем, до основних з яких можна віднести наступні: ідентифікацію реальних об'єктів, вибір виду моделей, побудова моделей і їх програмну

реалізацію, взаємодія дослідника з моделлю в ході комп'ютерного експерименту, перевірку правильності отриманих в ході моделювання результатів, виявлення основних закономірностей, досліджених в процесі моделювання. Залежно від об'єкта моделювання та виду використовуваної моделі ці проблеми можуть мати різну значимість.

Зазначено, що відповідно до принципу необхідної різноманітності, «різноманітність засобів моделювання повинно бути більше або, принаймні, так само різноманітності об'єкта моделювання». В цьому випадку традиційний спосіб опису різних аспектів системи і процесу її створення, що передбачає послідовний розвиток однієї моделі, не є ефективним. Сама наявність такої єдиної моделі викликає сумніви, тому що занадто складний і багатогранний предмет опису – від реальних об'єктів і процесів до абстрактних інформаційних та інших об'єктів.

Вихід в подібній ситуації був запропонований свого часу розробниками імітаційної моделі екосистеми Азовського моря, очолюваними Ю.А. Ждановим. Вже тоді (а це було більше 30 років тому) було показано, що «... рішення природничо-наукових і практичних питань, пов'язаних з проблемою Азовського моря, можливо тільки на базі імітаційної моделі, що відбиває всі основні процеси життєдіяльності екологічної системи моря, що дозволяє кількісно аналізувати закономірності її функціонування і розраховувати наслідки різних антропогенних впливів ». При створенні моделі такого рівня складності авторам і довелося зіткнутися як з проблемами масштабування, так і з значними розмірами та кількістю параметрів, з необхідністю врахування різноманітних аспектів дослідження ... «Завдання, таким чином, полягала в тому, щоб формалізувати і кількісно описати життєдіяльність великого числа компонент цієї системи, математично відобразити процеси їх взаємодії і розвитку». Ю.А Жданов зазначав, що «методологічні принципи, покладені в основу розробленої імітаційної моделі, ... сформувалися в ході самої розробки, бо практично був відсутній який-небудь досвід моделювання екосистем з великою кількістю

параметрів». У цій ситуації було запропоновано і потім реалізовано понад 20 аспекту моделей, інтегрованих в рамках спільного використання вихідних даних в єдиний моделює комплекс. Отримана імітаційна модель на той момент за масштабами не мала прецедентів у практиці дослідження екосистем. Зокрема зазначається, що «за допомогою моделі був проведений детальний аналіз різних варіантів екопроектів. ...прогноз здійснювався для 120 модельованих компонент екосистеми. Було розраховано понад 100 стратегій ... ». Підсумки робіт зі створення імітаційної моделі Азовського моря були включені в зведені звіти СРСР з водного проекту Міжнародного інституту прикладного системного аналізу, в наукову програму обміну між СРСР і США, опубліковані в працях Агентства з охорони навколишнього середовища США, яке використовувало модель для дослідження екосистеми озера Гурон.

Розглянемо сучасні рішення проблем формування систем моделювання складних процесів соціальної взаємодії. Як відомо при формуванні моделі прагнуть досягати відповідності між способом організації соціального світу і способом, яким модель описує цей світ, між апаратом, використовуваним в процесі моделювання, і концептуальним апаратом моделюється теорії, між теорією і соціальним світом. У соціальному моделюванні використовується весь спектр методів та інструментарію від змістовних моделей (наприклад, сформульованих на природній мові) до формальних моделей (наприклад, реалізованих за допомогою традиційних математичних моделей або сучасних парадигм імітаційного моделювання).

В якості вихідної змістовної моделі дослідники соціальних систем нерідко вибирають когнітивну (наприклад, когнітивну карту) або концептуальну (наприклад, реалізовані в рамках технологій функціонально-структурних або об'єктно-орієнтованих мов моделювання) модель. Такого роду змістовна модель відображає базові концепти і конструкти досліджуваної предметної області знання і дозволяє досягнення певного рівня абстрагування на шляху від попереднього опису об'єкта до його формальної моделі.

Зазначається, що когнітивні моделі можуть бути корисним інструментом для формування і уточнення гіпотези про функціонування досліджуваного об'єкта, що розглядається як складна система. Для того щоб зрозуміти і проаналізувати поведінку складної системи будується когнітивна карта, що представляє собою структурну схему причинно-наслідкових зв’язків. Такого роду модель дозволяє візуалізувати інформацію і аналізувати ситуацію на основі виявлених сутностей і зв'язків в рамках заданих обмежень. У цьому випадку кожна з безлічі вершин когнітивної карти відповідає одному фактору або елементу соціальної системи, а дуга, що зв'язує вершини, відповідає причинно-наслідкового зв'язку, і тим самим, когнітивна карта надає схематичне, спрощене опис соціальної системи в конкретній проблемній ситуації. Поряд з власне дослідними особливо слід підкреслити комунікативні можливості когнітивного інструментарію. Саме в комунікативній сфері знаходяться найбільш очевидні ресурси підвищення ефективності вирішення багатьох соціальних проблем.

В світлі розглянутих проблем моделювання відзначено і принципову сумісність когнітивних моделей з соціальними моделями більш високого рівня формалізації. Зокрема, розглянуто рішення задачі стійкого розвитку соціальної системи як послідовне вирішення завдань когнітивного моделювання (результат когнітивні карти) і далі завдання моделювання перехідного процесу в системі з оцінкою її характеристик стійкості. Зазначений підхід в даний час також досить широко апробується в рішенні задач оцінки стійкості розвитку соціально-економічних систем. Формально він зводиться до моделювання імпульсних процесів в вершинах когнітивної моделі, одержуваних при різних керованих і некерованих впливах в вершини, а результат моделювання являє собою можливі альтернативні варіанти розвитку системи – модельні сценарії розвитку. Апарат сценарного моделювання дозволяє формально будувати сценарії як гіпотетичні траєкторії руху модельованої системи в фазовому

просторі її змінних (факторів) на основі інформації про її структурі та бажаних програмах розвитку.

У найзагальнішому випадку завдання побудови сценаріїв на когнітивних моделях формується так: на основі вивчення складної, динамічної, відкритої, керованої, але не порожниною спостерігається системи необхідно описати можливі напрямки її розвитку декількома (бажано небагатьма) варіантами так, щоб в рамках поставленої змістовної завдання дати найбільш повне уявлення про можливі майбутні станах і траєкторіях розвитку системи.

Але найбільш поширеним варіантом спільного використання є імітаційне моделювання, що реалізує парадигму системної динаміки. Зазначена парадигма моделювання передбачає побудову для досліджуваної системи графічної діаграми причинних зв'язків і глобальних впливів одних параметрів на інші параметри в часі, а потім модель, створена на основі цих діаграм, імітується на комп'ютері. Системна динаміка представляє сьогодні парадигму на основі фундаментальних робіт Дж. Форрестера, метод і графічна мова для представлення моделей складних систем, а також для їх імітаційного комп'ютерного виконання. Графічна нотація для моделювання всіх компонентів системи і їх взаємозв'язків роблять системну динаміку дуже зручним інструментом візуального представлення будь-якої системи, в тому числі і соціальної. Складні зв'язки і взаємні впливу процесів часто зустрічаються в соціальних системах і тому системна динаміка виявилася дуже ефективним методом для представлення та аналізу проблем динаміки такого роду систем. Як приклад можна вказати на вирішення найбільш поширеною завдання дослідження демографічних і міграційних процесів в соціальних системах різного рівня.

Очевидні переваги спільного використання когнітивних і системно-динамічних моделей при дослідженні соціальних систем породжують все нові спроби розширення функціональності досліджень, коректного подолання проблем масштабованості об'єкта в заданій предметній області і, нарешті,

постановки і рішення задач опису та моделювання соціальної поведінки в рамках все більш складної архітектури систем моделювання. З усього різноманіття прикладів на даному етапі можна відзначити ієрархічні системи когнітивних моделей, когнітивні архітектури і інтегровані системи моделювання.

Побудова ієрархічних систем з когнітивних моделей дозволяє, залишаючись в рамках єдиної обраної парадигми моделювання, переходити на наступних (нижніх) рівнях ієрархії до більш детального опису сутностей і зв'язків предметної області. Наприклад, при необхідності, одна з вершин когнітивної карти верхнього рівня може бути представлена в свою чергу у вигляді когнітивної карти на нижньому рівні, і, таким чином, формується ієрархічна система когнітивних моделей. У такому послідовному просуванні при вивченні слабко структурованих проблем соціальних систем сукупність ієрархічних когнітивний моделей (когнітивних карт) дозволяє отримувати досить суворі формалізації первинного (якісного) опису соціальних процесів і систем.

Однак вказане звуження області використання когнітивного моделювання різко контрастує в порівнянні з зарубіжними публікаціями, де когнітивне моделювання охоплює більшу частину етапів дослідження соціальних систем, забезпечуючи потреби і в якісних, і в кількісних прогнозах шляхом формування когнітивної архітектури. Термін архітектура передбачає підхід, при якому моделюється не тільки внутрішня структура, але і соціальну поведінку. Когнітивна архітектура моделює пізнання в цілому, а не окремі його механізми, як, наприклад, когнітивна модель (когнітивна карта). В основі закладені такі механізми: ієрархічна система когнітивних моделей, що забезпечує спадний процес моделювання, моделі на основі нейронних процесів, що накладає відповідні обмеження і сприятиме точності, і, нарешті, інтегровані сховища вже реалізованих архітектур, обчислювальних моделей, середовищ і даних. Найбільш поширена когнітивна архітектура ACT-R застосовується як

для соціально-психологічних експериментів, так і для складних імітацій (авто або авіа трафіку і ін.). У ній описується структура наборів модулів і відповідні рівні імітації, що дозволяють враховувати всю область людського пізнання, зокрема, нейронний рівень, соціальний рівень, мережевий рівень і рівень інтеграції моделей і імітацій.

Таким чином, когнітивна архітектура являє собою систему статичних і поведінкових моделей, що забезпечує цілісний підхід до моделювання пізнання соціальних процесів, від отримання емпіричних даних до кількісної оцінки результатів моделювання, від рівня нейронного опису до системного рівня, при цьому когнітивна модель масштабується до більш високого рівня з збереженням якісних властивостей, а платформою узгодження служить сховище даних.

Наступний крок у дослідженні складних процесів соціальної взаємодії пропонується виконувати на основі інтегрованих систем моделювання. Принципова їхня відмінність від розглянутих вище полягає в використанні різних парадигм моделювання для дослідження відповідних властивостей соціальних процесів.

Слідуючи в руслі вище зазначених рішень відзначимо, що для подібних систем завдання, таким чином, зводиться до формування мінімально необхідної сукупності моделей m, кожна з яких є однією з проекцій процесів в області рішень, а всі разом вони утворюють систему моделей SM, що забезпечує з належною ступенем якості Q (повноти, правильності та адекватності) цільове дослідження зазначених процесів, формула 1.1.

$$\begin{aligned} &S_M =< Z, M,\ R,\ Q >,\ M \neq \varnothing, \\ &M \coloneqq\ \{m \mid F(m)\},\ \left|\{m_i\}\right| = min. \end{aligned} \tag{1.1}$$

де Z – множина (структура) цілей,

M – множина, що складається з моделей m таких, що m є однією з проекцій F (m) властивостей P процесів,

R – множина механізмів і відносин, що забезпечують інтеграцію моделей mi в систему SM, що володіє, в свою чергу, відповідними інтеграційними властивостями,

Q – множина вимог до якості дослідження процесів.

Стосовно до складних процесів соціальної взаємодії сукупність моделей повинна відображати різні абстракції опису структури, різноманітні аспекти її поведінки, етапи (ітерації) її еволюції в процесі функціонування і розвитку. Кожна з моделей має унікальні властивості, які відсутні у інших, і тому в різного ступеня відповідає реальним процесам. Використовувана модель, що інтерпретує досліджуване властивість процесів, в свою чергу, визначає і те, як буде осмислюватися проблема, і які рішення будуть прийматися. Необхідно розглядати архітектуру сукупності моделей в контексті необхідності для вирішення завдання на конкретному етапі дослідження.

1.2 Огляд програмних аналогів

На ринку ПЗ існують різні програмні рішення для моделювання бізнес процесів, наприклад, такі як:

– Bizagi Process Modeler – безкоштовне програмне забезпечення для створення діаграм процесів і документації в нотації стандарту BPMN. Відмінний інструмент побудови бізнес-процесів. Допомагає не тільки створити, але і опублікувати результати роботи в різних форматах, включаючи MS Word і інтерактивний HTML;

– Intalio BPMS – Безкоштовна. Open Source Business Management System. Програма для побудови і аналізу бізнес процесів;

– ARIS Express – Досить простий в установці і використанні інструмент, так що його можуть застосовувати і починаючі користувачі. ARIS Express належить до сімейства засобів моделювання ARIS (ARchitecture of Integrated Information Systems) і включає не тільки інструменти моделювання бізнес-

процесів і публікації моделей, а й інтегруються між собою кошти розробки системи збалансованих показників, оцінки і оптимізації вартості бізнес-процесів, інструменти, що спрощують впровадження ERP-систем, а також контрольні заходи за виконанням бізнес-процесів.

Також як аналоги для математичного моделювання процесів можливо використання Maple, MathCad и MatLab.

Maple – комерційна система комп'ютерної алгебри від компанії Waterloo Maple. Остання версія містить понад 5000 функцій для більшості розділів сучасної математики, моделювання та інтерактивної візуалізації, підтримує мову програмування Maple, і дозволяє комбінувати алгоритми, результати обчислення, математичні формули, текст, графіку, діаграми та анімацію зі звуком в електронному документі.

Mathcad – система комп'ютерної алгебри з класу систем автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводом, відрізняється легкістю використання і застосування для колективної роботи.

Mathcad має інтуїтивний і простий для використання інтерфейс користувача. Для введення формул і даних можна використовувати як клавіатуру, так і спеціальні панелі інструментів.

Незважаючи на те, що ця програма, в основному, орієнтована на користувачів, які не є програмістами, Mathcad також використовується в складних проектах, щоб візуалізувати результати математичного моделювання шляхом використання розподілених обчислень і традиційних мов програмування.

MATLAB – пакет прикладних програм для вирішення задач технічних обчислень.

MATLAB надає користувачеві велику кількість (кілька сотень) функцій для аналізу даних, які покривають майже всі області математики, зокрема:

– матриці і лінійна алгебра – алгебра матриць, лінійні рівняння, власні значення і вектори, сингулярності, факторизація матриць та інші.

– багаточлени і інтерполяція – корені многочленів, операції над многочленами і їх диференціювання, інтерполяція і екстраполяція кривих і інші.

– математична статистика та аналіз даних – статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур'є та інші.

– обробка даних – набір спеціальних функцій, включаючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування (в квадратурі) та інші.

– диференціальні рівняння – рішення диференціальних і диференціально-алгебраїчних рівнянь, диференціальних рівнянь із запізненням, рівнянь з обмеженнями, рівнянь в приватних похідних і інші.

– розріджені матриці – спеціальний клас даних пакету MATLAB, що використовується в спеціалізованих додатках.

– цілочисельна арифметика – виконання операцій над цілими числами в середовищі MATLAB.

В якості систем імітаційного моделювання наведено наступні продукти:

Система Actor Pilgrim – система імітаційного моделювання тимчасової, просторової і фінансової динаміки економічних процесів. Система дозволяє працювати з багатошаровими імітаційними моделями. Підтримувані види (технології) моделювання: дискретне і дискретно-безперервне, механізм віртуального таймера дискретно-подієвий, одночасна реалізація тимчасової, просторової і фінансової динаміки;

AGNES (AGent NEtwork Simulator) – система імітаційного моделювання великих систем з дискретними подіями. Система AGNES є кросплатформенною, можливий розподілений запуск (на локальній мережі або обчислювальному кластері). Область застосування: локальні комп'ютерні

мережі, бездротові сенсорні мережі, обчислення на суперЕОМ. AGNES поширюється безкоштовно за ліцензією LGPL;

Система моделювання AnyLogic підтримує три підходи до створення імітаційних моделей: процесно-орієнтований (дискретно-подієвий), системно-динамічний і Агентно, а також будь-яку їх комбінацію. Графічний інтерфейс AnyLogic, інструменти та бібліотеки дозволяють швидко створювати моделі для широкого спектра задач – від моделювання виробництва, логістики, бізнес-процесів до стратегічних моделей розвитку компанії і ринків. AnyLogic став корпоративним стандартом на бізнес-моделювання в багатьох транснаціональних компаніях, широко використовується в освіті;

Arena – система дискретного моделювання. Сфера основних додатків системи – імітаційне моделювання виробничих технологічних процесів і операцій, складський облік, банківська діяльність, оптимізація обслуговування клієнтів в сфері послуг, транспортні завдання. Arena забезпечена зручним об'єктно-орієнтованим інтерфейсом і володіє можливостями адаптації до різних предметних областях. Система не вимагає написання програмного коду і виключно проста у використанні, але для її освоєння потрібні значний час і досить глибокі знання теорії ймовірностей, математичної статистики, теорії систем масового обслуговування, мереж Петрі.

1.3 Призначення та сфера застосування

Призначення даного ПО та дослідницької роботи:

– аналіз складу та опису складного процесу та побудова відповідного дерева рішень;

– перевірка часової ефективності отриманої моделі;

– дослідження та порівняння моделей отриманих при різних умов застосування, перевірка коректності використання даного підходу.

Найкраща сфера застосування та ефективність даного ПО буде знайдена як результат даного дослідження, але можливість застосування в моделюванні

для дослідження часової ефективності та оптимізації складних технічних процесів та підпроцесів різних ланок заздалегідь гарантоване.

1.4 Постановка задачі

Розробити програмне середовище для дослідження та моделювання складних процесів, що включає в себе такі функції та можливості:

– аналіз об’єкту що досліджується та його стану;

– складання математичної моделі процесів, у вигляді графів переходів;

– побудова дерева рішень для об’єкту дослідження;

– виконання імітації виконання процесу за згенерованим деревом рішення;

– дослідження виконання процесу по дереву рішень за часовим показником та знаходження слабких місць складних процесів.

Висновки до розділу 1

Аналіз предметної сфери показав відсутність інструментальних середовищ здатних виконувати оцінку алгоритмів за показниками часової ефективності або складність їх використання, чи високу ціну на ПО.

Існуючі програмні рішення не відповідають повною мірою поставленим вимогам. Отже, завдання з розробки програмної системи є актуальним.

2 ОБҐРУНТУВАННЯ ЕКСПЕРИМЕНТАЛЬНОГО МЕТОДУ ДОСЛІДЖЕННЯ СКЛАДНИХ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ ЗА ДОПОМОГОЮ ПОЕТАПНОГО МОДЕЛЮВАННЯ

Для рішення поставленої задачі розроблено новий алгоритм, що відповідає потребам, які виникають під час дослідження та є ефективним рішенням задачі. Розроблений підхід може бути використаний для дослідження прикладної задачі або для дослідженням складних технологічних процесів на підприємстві. Крім того необхідно враховувати постійний розвиток технологій та обчислюваних машин та слід пам’ятати про постійне збільшення потужності обчислюваних машин та зростання об’єму доступної пам’яті, але також треба не забувати про ті випадки коли ці ресурси обмежені, або про можливості змін умов використання. Тому слід розробляти алгоритм, враховуючі всі ці фактори, та намагатися не тільки націлити його на отримання необхідного результату але і мати вектор подальшого розвитку алгоритму та самого ПЗ.

У задачі «розробка алгоритму» у більшості випадків мається на увазі використання відомих методик, їх комбінування та використання різних підходів до реалізації для отримання необхідного результату. Для розв’язку задачі використовується методологія з використанням поетапного моделювання. Моделювання передбачає 3 етапи та використовує алгоритм декомпозиції, тобто розглядає задачу від глобального до детального.

Головною задачею цього експериментального методу дослідження довести або спростувати гіпотезу про те що, поетапний метод моделювання, реалізований за допомогою мови логічного програмування Prolog у парі з мовою програм C#, може бути ефективним рішення для рішення задач, у сфері моделювання та контролю функціонування складних технічних, технологічних, бізнес або інших процесів що можливо представити у необхідній формі.

2.1 Перший етап

На першому етапі даної реалізації проводиться збір необхідної інформації для проведення експерименту. Ця інформація представляється у вигляді статистики, наборів інтервалів або в будь якому іншому вигляді, лише за одним обмеженням, щоб цю інформацію представити згідно вимог методики.

Інформація представляється згідно вимог:

інформація про процес, що досліджується, представляється у вигляді ієрархічного дерева процесів, де кожен вузол має свій номер, там до кожного вузлу зіставляється умова, за якою система буде розуміти куди буде просуватися процес за ходом виконання алгоритму. Номер необхідний для розуміння місцеположення та формування шляху проходження цього дерева, приклад інтерпретації зовнішнього вигляду дерева наведено на рис. 2.1.

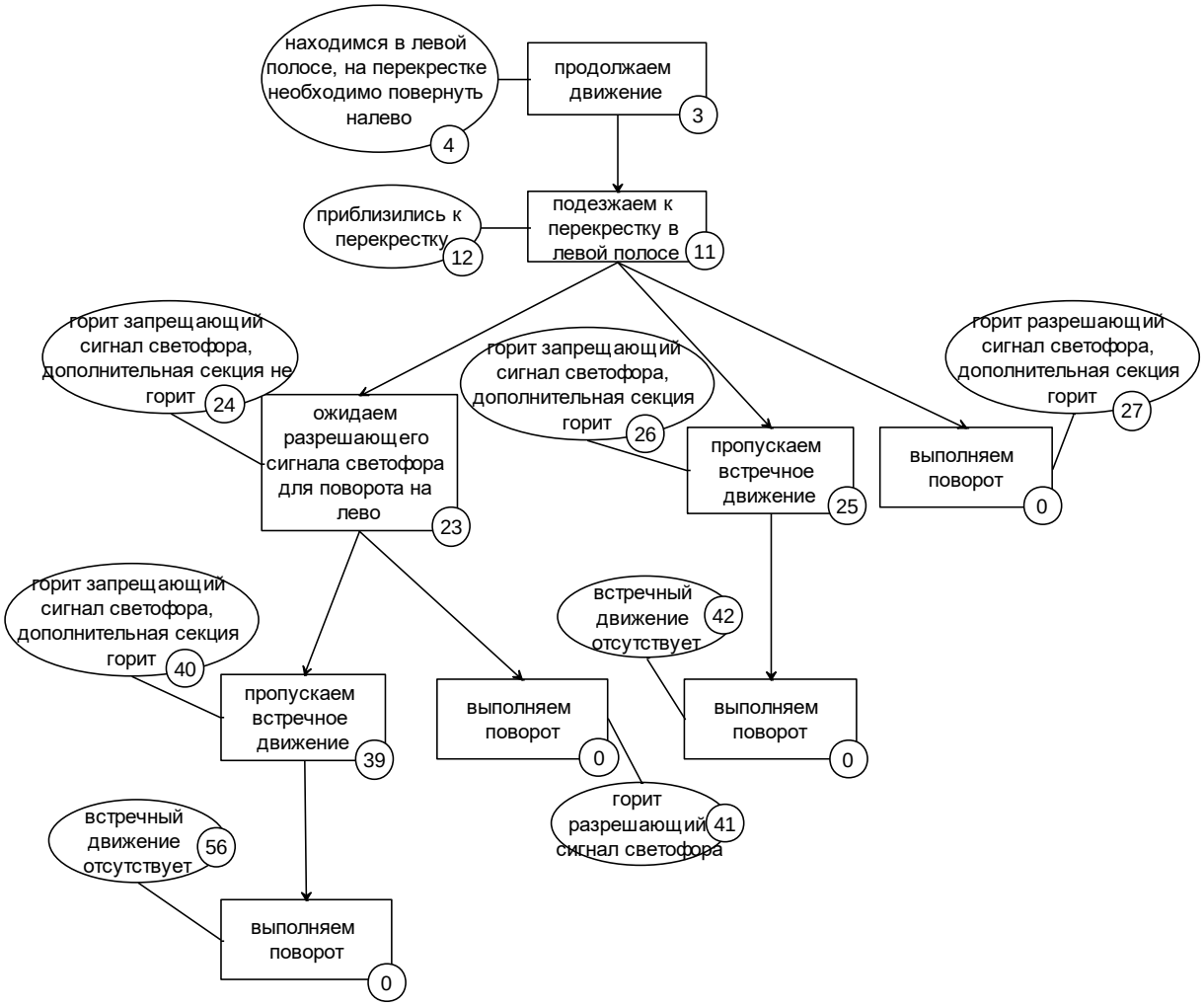


Рисунок 2.1 – Приклад схема переходів у дереві переходів

Після завантаження вся ця структура має бути записана у відповідному вигляді, для зчитування системою необхідної інформації та виконати необхідні дії, приклад запису вхідного файлу з правилами переходів наведено на рис. 2.2;

– інформація про час виконання кожного вузлу або час необхідний для переходу з одного вузлу до іншого повинен бути записаний у окремий файл у відповідній формі, приклад файлу зі вказаним необхідним часом зображено на рис. 2.4;

– необхідно вказувати кількість повторень моделювання часу проходження отриманого ланцюга.

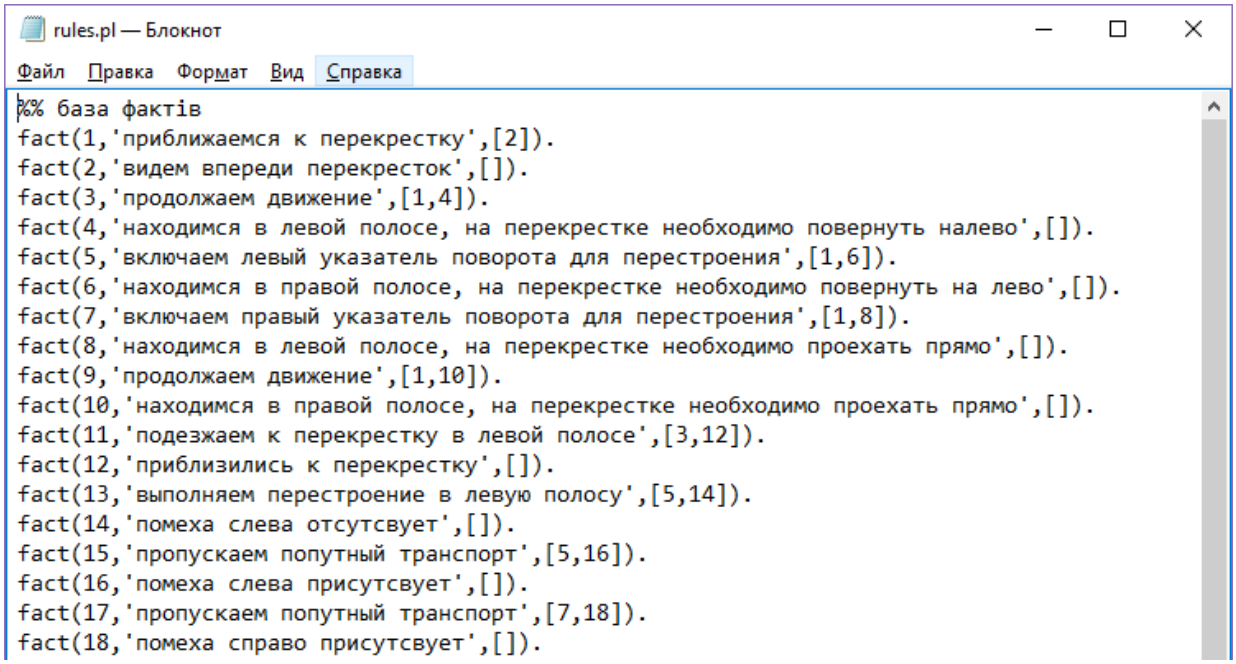


Рисунок 2.2 – Приклад запису вхідного файлу з правилами переходів

Легенда до рис. 2.2 наведно на рис. 2.3.

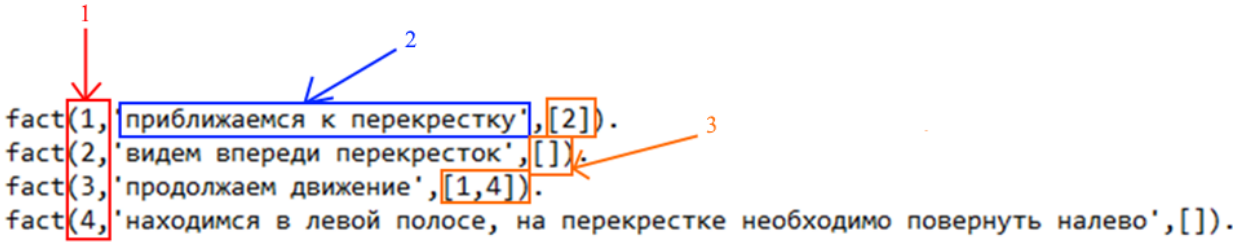


Рисунок 2.3 – Поянення до рис. 2.2

На рис. 2.3 наведено:

1 – номер вузлу;

2 – опис вузлу;

3 – пов’язані вузли.

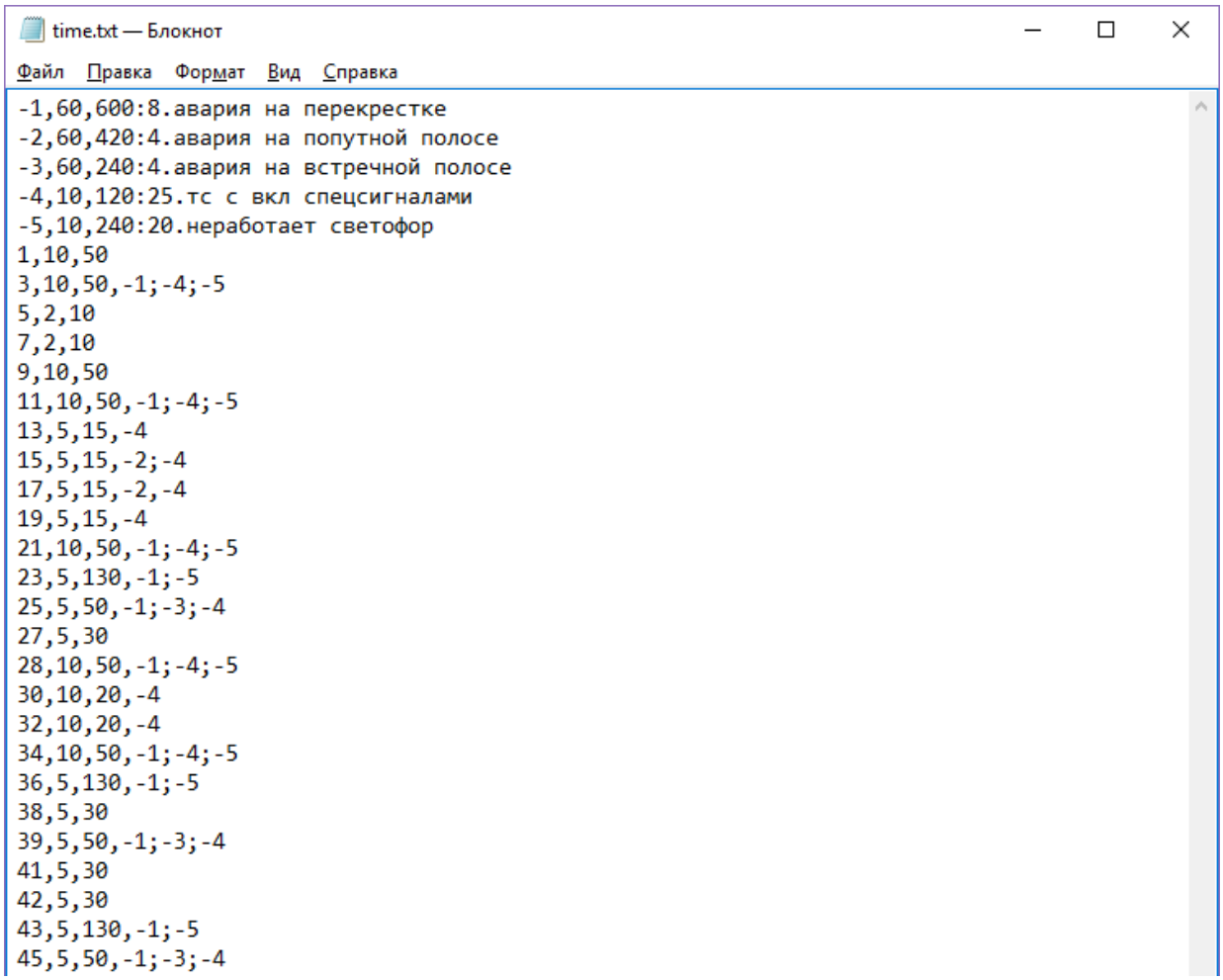


Рисунок 2.4 – Приклад файлу зі вказаним необхідним часом

Легенда до рис. 2.4 наведно на рис. 2.5.

Рисунок 2.5 – Пояснення до рис. 2.4

На рис. 2.5 наведено:

1 – номер вузлу;

2 – початок интервалу часу виконання;

3 – кінець інтервалу часу виконання;

4 – номер вузлів вірогідних випадковиї події;

5 – вірогідність виникнення події;

6 – пояснювальні коментарі.

Для подальшої обробки різниці, яким шляхом було отримано ці данні не є важливим, але для подальшого використання цього алгоритму та методики на практиці на цьому етапі може залучається автоматизована система. Завданням такої системи стає відстежування станів об’єктів системи під час роботи та передачі цих даних для подальшої обробки. Тобто, у перспективі є можливість повної автоматизації системи з використанням цього алгоритму, за умови, що до процесу, що відстежується є можливість використання автоматизованого підходу.

2.2 Другий етап

На цьому етапі відбувається подальша обробка інформації що була надана на першому етапі.

Обробка виконується шляхом перевірки на відповідність вхідних даних та процесу з питанням, як повинен виконуватися цей процес. Алгоритм на відповідність вхідних даних та процесу виконується за допомогою програми, написаної на мові Prolog. Як результат роботи програми, будується дерево переходів що відповідає тим підпроцесам, що відбуваються в системі або об’єкті під час виконання процесу. Побудова дерева виконуються за допомогою зворотнього ланцюжку виводу. У результаті проходження даного етапу буде отримано проміжний результат у вигляді ланцюга переходів, приклад проміжного файлу із ланцюгом виконання наведено на рис. 2.6.

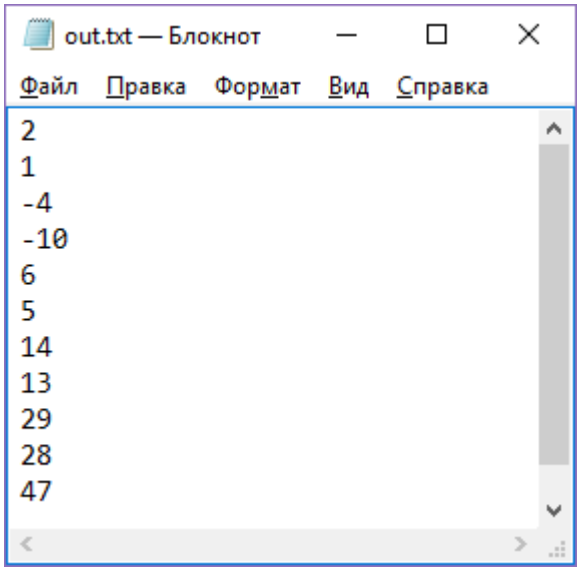


Рисунок 2.6 – Приклад проміжного файлу із ланцюгом виконання

2.3 Третій етап

На останньому етапі відбувається моделювання проходів цього фрагменту, котрий представлений ланцюгом переходів, отримання статистики часової ефективності цього процесу, слабкі місця процесу та можливість порівняти результати отримані під час моделювання та у реальному процесі, а також можливість спрогнозувати результати та дії на майбутнє. В залежності від вказаної кількості повторень результат може істотно відрізнятися, чим більша кількість повторень тим точніший буде результат. Треба зазначити що у даній реалізації використовуються простий метод генерування випадкових подій, для покращення генерування вірогідності виникнення випадкових подій можлива зміна генератору випадкових подій на інший, з використанням кращих методів, але це може привезти як покращенню ефективності так і погіршенню не тільки ефективності генерації випадкових подій, але і к погіршенню часової ефективності. Все це можливо також доповнити тим що, алгоритм може бути покращено за допомогою оптимізації, якщо це буде необхідно.

Висновки до розділу 2

У розділі обґрунтовано використання саме цього методу проведення дослідження для підтвердження або спростування гіпотези. Описано вибір етапів моделювання та їх функціональну необхідність.

Описано місця та методологію можливостей покращення алгоритму.

3 ПРОЕКТУВАННЯ Й РОЗРОБКА ІНСТРУМЕНТАЛЬНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОСЛІДЖЕННЯ МЕТОДУ ПОЕТАПНОГО МОДЕЛЮВАННЯ СКЛАДНИХ ПРОЦЕСІВ

3.1 Формалізація задачі

Формалізація задачі на рівні зовнішнього проектування наведена у вигляді діаграми варіантів використання. Користувач представлений у вигляді актора-дослідника, що взаємодіє із системою.

Схема використання системи зображена на рис. 3.1.

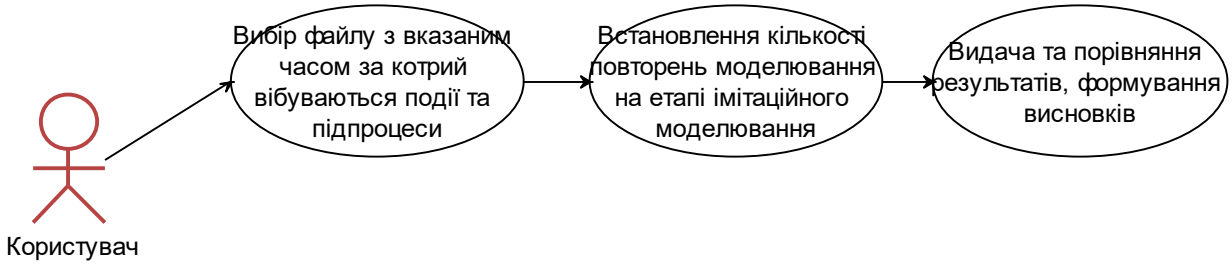


Рисунок 3.1 – Схема використання системи

Де на рис. 3.1 наведено:

 – Образне представлення користувача системою;

У овалах наведено можливості що може виконувати користувач.

3.2 Базова архітектура системи

Проектування інструментального засобу одразу було розпочато з урахуванням розробки програмного продукту під 3-рівневу модель MVC. Основна парадигма – об’єктно-орієнтована, функціональна, але на рівні контролера присутній зв'язок модулем що працює на мові логічного програмування Prolog з декларативним підходом до програмування, що базується на наборах фактів та правил. Схема взаємодії системи в рамках MVC моделі зображена на рис. 3.2.

Концепція патерну MVC передбачає поділ додатка на три компонента:

Модель (Model): описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як

правило, об'єкти моделей зберігаються в базі даних або локальній пам’яті робочої станції [3].

У MVC моделі представлені двома основними типами: моделі уявлень, які використовуються уявленнями для відображення і передачі даних, і моделі домену, які описують логіку управління даними.

Модель містить дані та зберігає логіку управління цими даними. У той же час модель не повинна містити логіку взаємодії з користувачем і не має визначати механізм обробки запиту. Також модель не повинна містити логіку відображення даних в поданні.

Подання (View): відповідають за візуальну частину або призначений для користувача інтерфейс, нерідко html-сторінка або Win-форма, через які користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час уявлення не повинно містити логіку обробки запиту користувача або управління даними.

Контролер (Controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей [3].

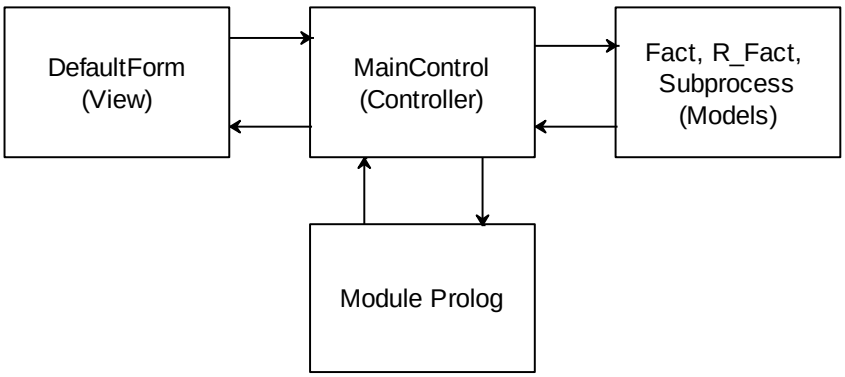


Рисунок 3.2 – Схема взаємодія частин архітектурної моделі MVC

Більш детальна діаграма взаємодії компонентів системи наведена на рис. 3.3.

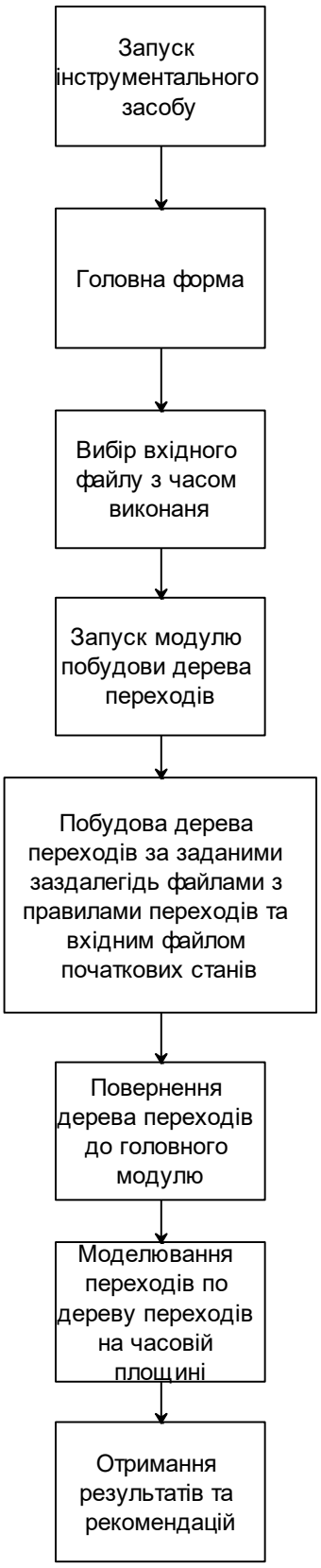


Рисунок 3.3 – Діаграма взаємодії компонентів системи

3.3 Внутрішнє проектування

3.3.1 Вибір мови програмування

Для розробки інструментального засобу було обрано C# та мову логічного програмування Prolog.

C# – об'єктно-орієнтована мова програмування. Розроблено в 1998-2001 роках групою інженерів компанії Microsoft під керівництвом Андерса Хейлсберг і Скотта Вільтаумота як мову розробки додатків для платформи Microsoft .NET Framework. Згодом був стандартизований як ECMA-334 і ISO / IEC 23270 [6].

C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі в форматі XML.

Перейнявши багато від своїх попередників – мов C++, Delphi, Модула, Smalltalk і, особливо, Java – C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинна реалізація інтерфейсів) [6].

C# розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#, яка відображає BCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід очікувати і в подальшому (проте, ця закономірність була порушена з виходом C# 3.0, що представляє собою розширення мови, що не

спираються на розширення платформи .NET). CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, прибирання сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так же, як це робиться для програм на VB.NET, J# і ін.

Пролог (англ. Prolog) – мова і система логічного програмування, засновані на мові предикатів математичної логіки диз'юнктів Хорна, що представляє собою підмножина логіки предикатів першого порядку [7].

Мова зосереджений навколо невеликого набору основних механізмів, включаючи зіставлення зі зразком, деревовидного представлення структур даних і автоматичного перебору з поверненнями. Добре підходить для вирішення завдань, де розглядаються об'єкти (зокрема структуровані об'єкти) і відносини між ними. Пролог, завдяки своїм особливостям, використовується в області штучного інтелекту, комп'ютерної лінгвістики і нечислового програмування в цілому. У деяких випадках реалізація символьних обчислень на інших стандартних мовах викликає необхідність створювати велику кількість коду, складного в розумінні, в той час як реалізація тих же алгоритмів на мові Пролог дає просту програму, легко поміщається на одній сторінці.

Prolog є декларативною мовою програмування: логіка програми виражається в термінах відносин, представлених у вигляді фактів і правил. Для того щоб ініціювати обчислення, виконується спеціальний запит до бази знань, на які система логічного програмування генерує відповіді «істина» і «брехня». Для узагальнених запитів зі змінними в якості аргументів створена система Пролог виводить конкретні дані на підтвердження істинності узагальнених відомостей і правил виведення [7].

Інакше кажучи, предикат можна визначити як функцію, яка буде показувати безліч довільної природи в безліч булевих значень {помилково, істинно}. Завдання пролог-програми полягає в тому, щоб довести, чи є задане цільове твердження наслідком з наявних фактів і правил.

3.3.2 Технологічна платформа

В розробці даного інструментального засобу використовувалися наступні середовища розробки програмного забезпечення:

Microsoft Visual Studio 2019 – продукт компанії Microsoft, що включає інтегроване середовище розробки програмного забезпечення і ряд інших інструментів. Дані продукти дозволяють розробляти як консольні додатки, так і ігри та програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінгу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення [4].

SWI Prolog – Реалізація мови програмування Prolog. SWI-Prolog – це потужне середовище розробки з набором графічних інструментів XPCE. Ядро системи ліцензовано під GNU LGPL, бібліотеки – під GNU GPL з додатковою умовою, що дозволяє використання в пропрієтарних додатках. Розвиток SWI-

Prolog почалося в 1987 р, і сьогодні він широко використовується в дослідницьких і освітніх цілях, а також в комерційних додатках.

SWI Prolog – досить популярна система, в основному завдяки зручній середовищі та яку переносять бібліотеці для створення графічного інтерфейсу. SWI-Prolog, як майже всі реалізації мови, здебільшого реалізує Edinburgh Prolog, але також містить окремі елементи ISO Prolog.

SWI-Prolog включає в себе швидкий компілятор, профілювальник, набір бібліотек та зручний інтерфейс для підключення C-модулів. Він реалізований для ряду UNIX-платформ, таких, як HP, IBM Linux, для NeXT, OS / 2, Sun і Sparc [5].

3.3.3 Ієрархія та взаємодія класів системи

Результати проектування та розробки системи класів зображено на рис. 3.4.

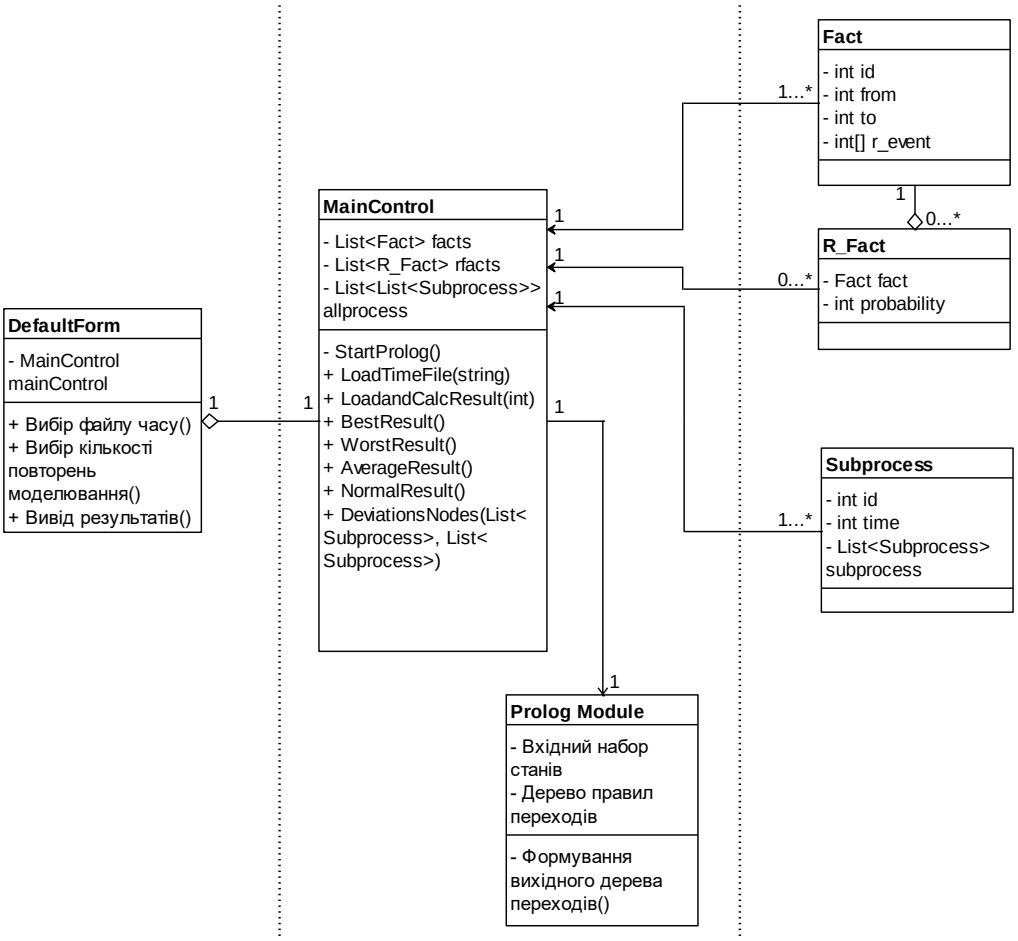


Рисунок 3.4 – Діаграма класів

3.3.4 Ієрархія та схема переходів у дереві переходів

В ході розробки було використано набір фактів для мови логічного програмування Prolog, приклад схема переходів між фактами наведено на зображені на рис. 3.5 – 3.11.

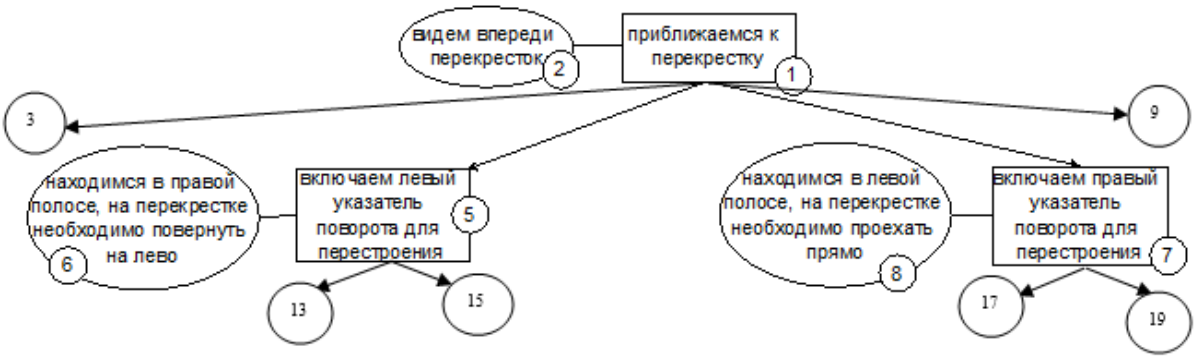


Рисунок 3.5 – Схема переходів частина 1

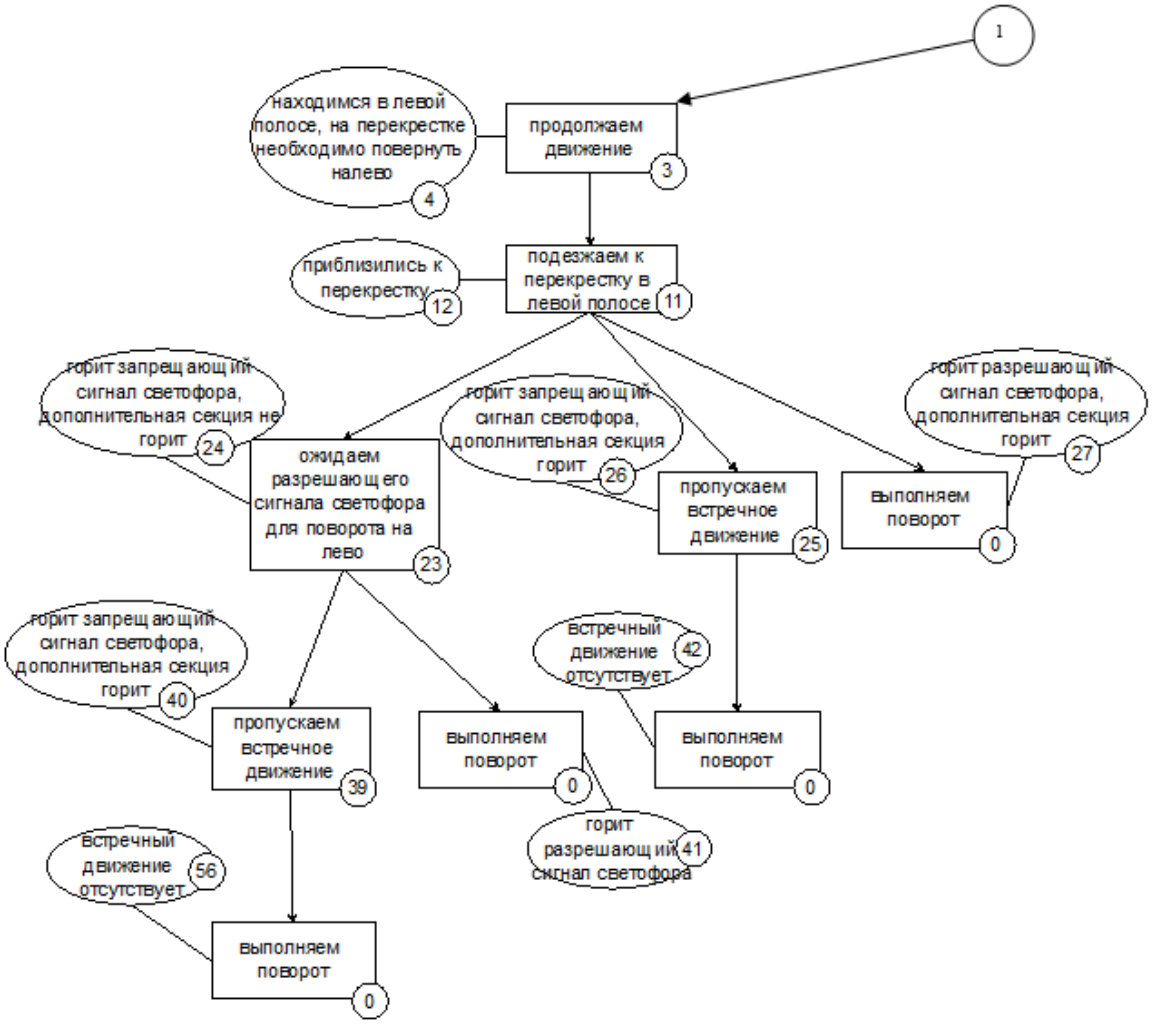


Рисунок 3.6 – Схема переходів частина 2

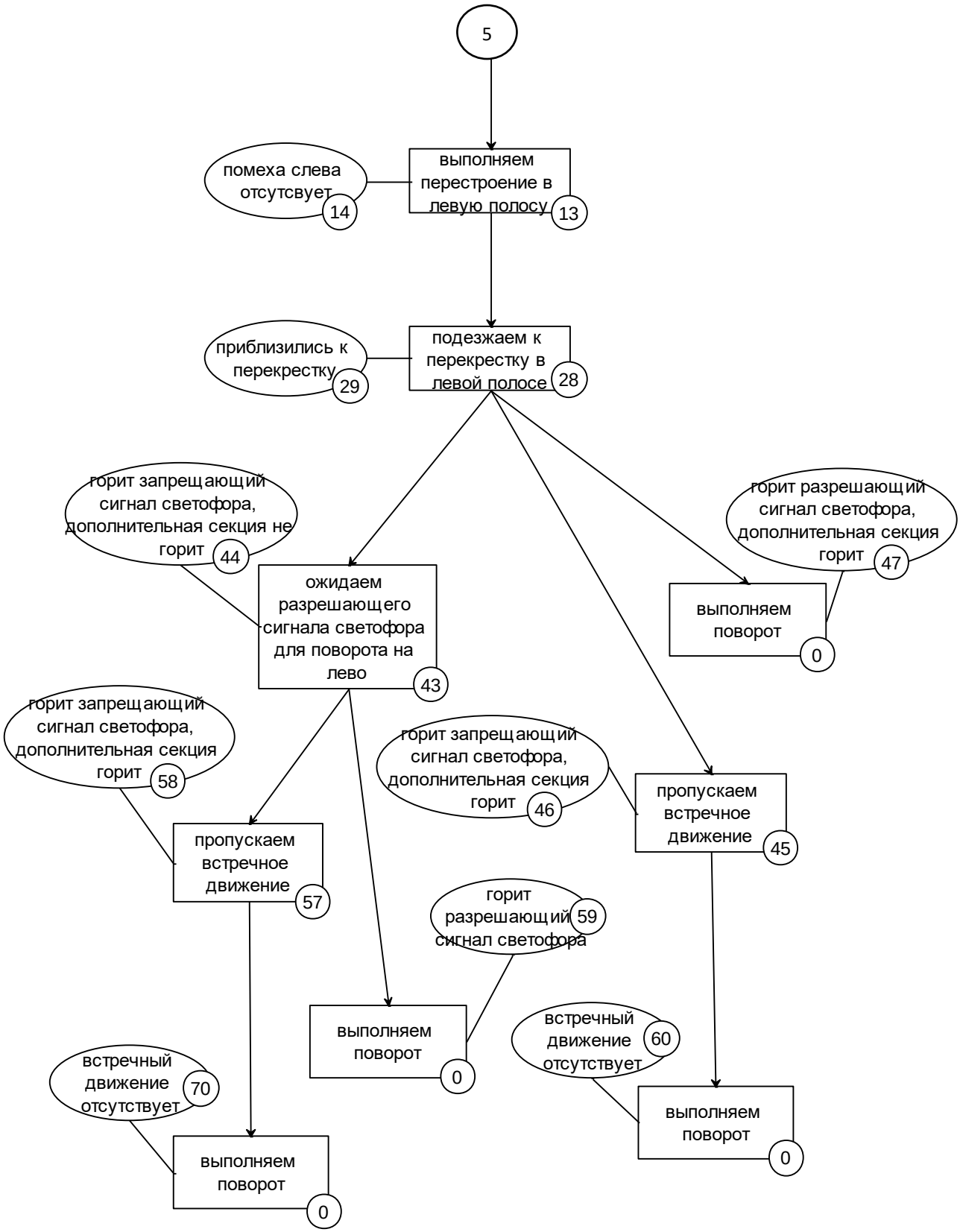


Рисунок 3.7 – Схема переходів частина 3

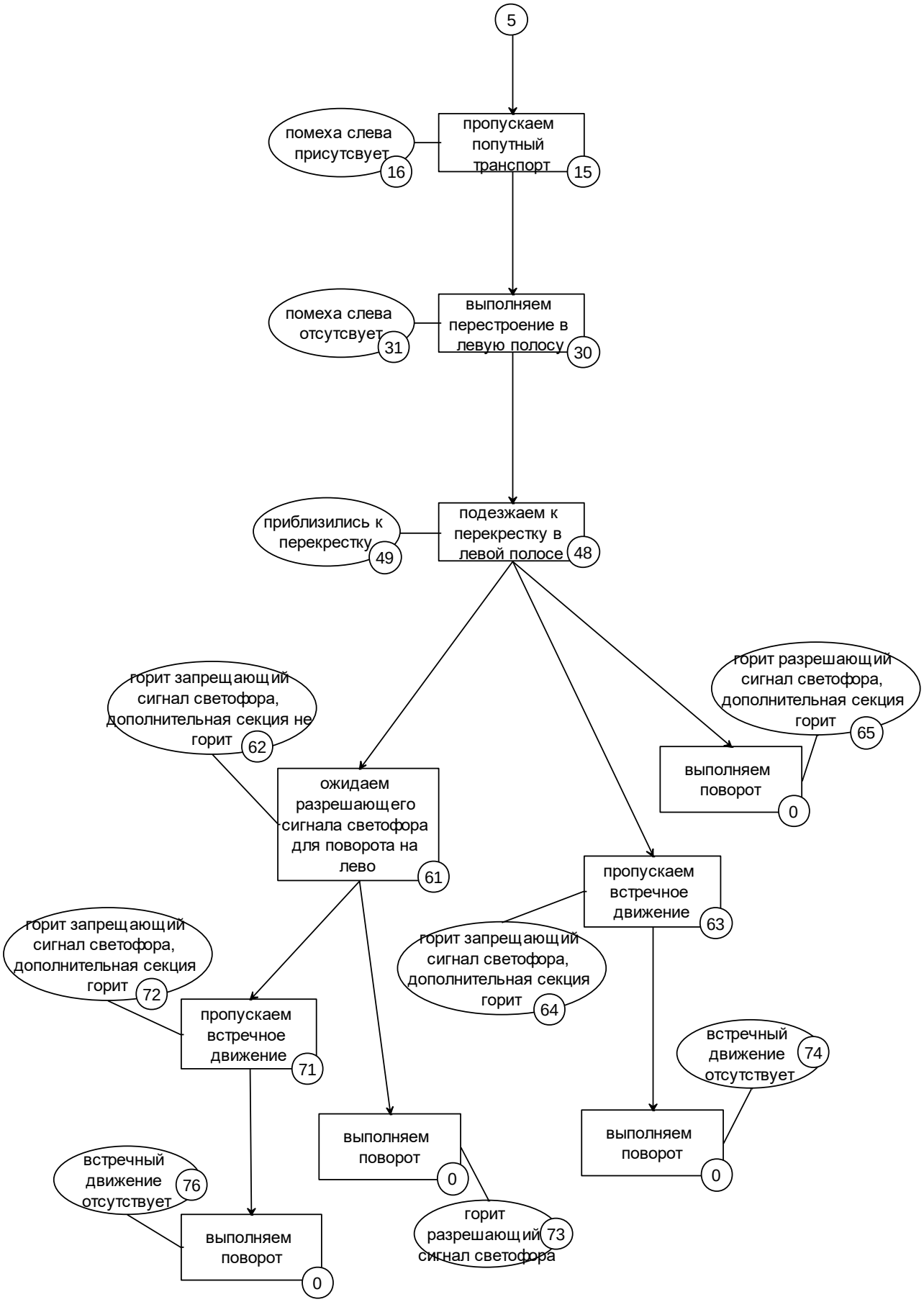


Рисунок 3.8 – Схема переходів частина 4

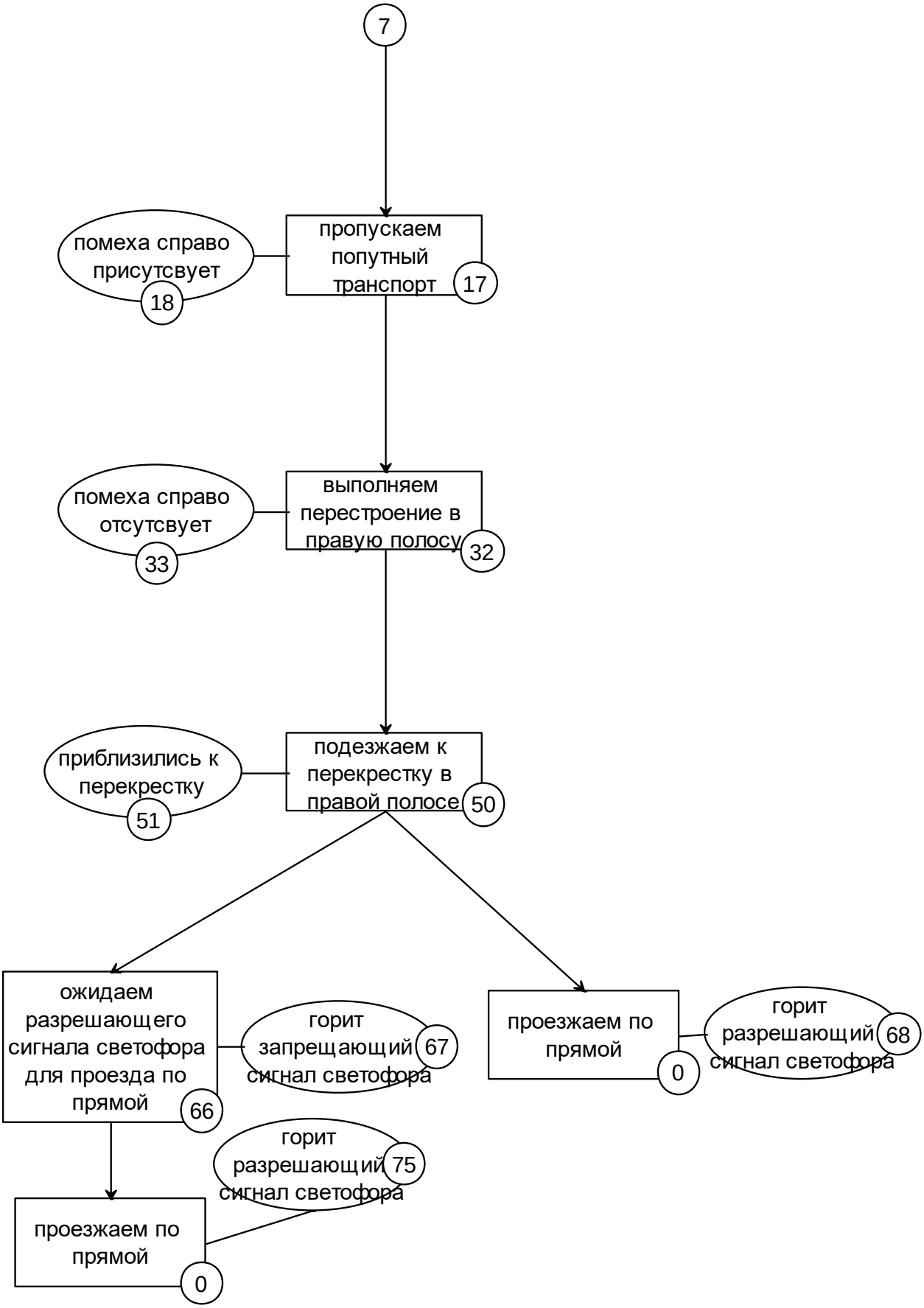


Рисунок 3.9 – Схема переходів частина 5

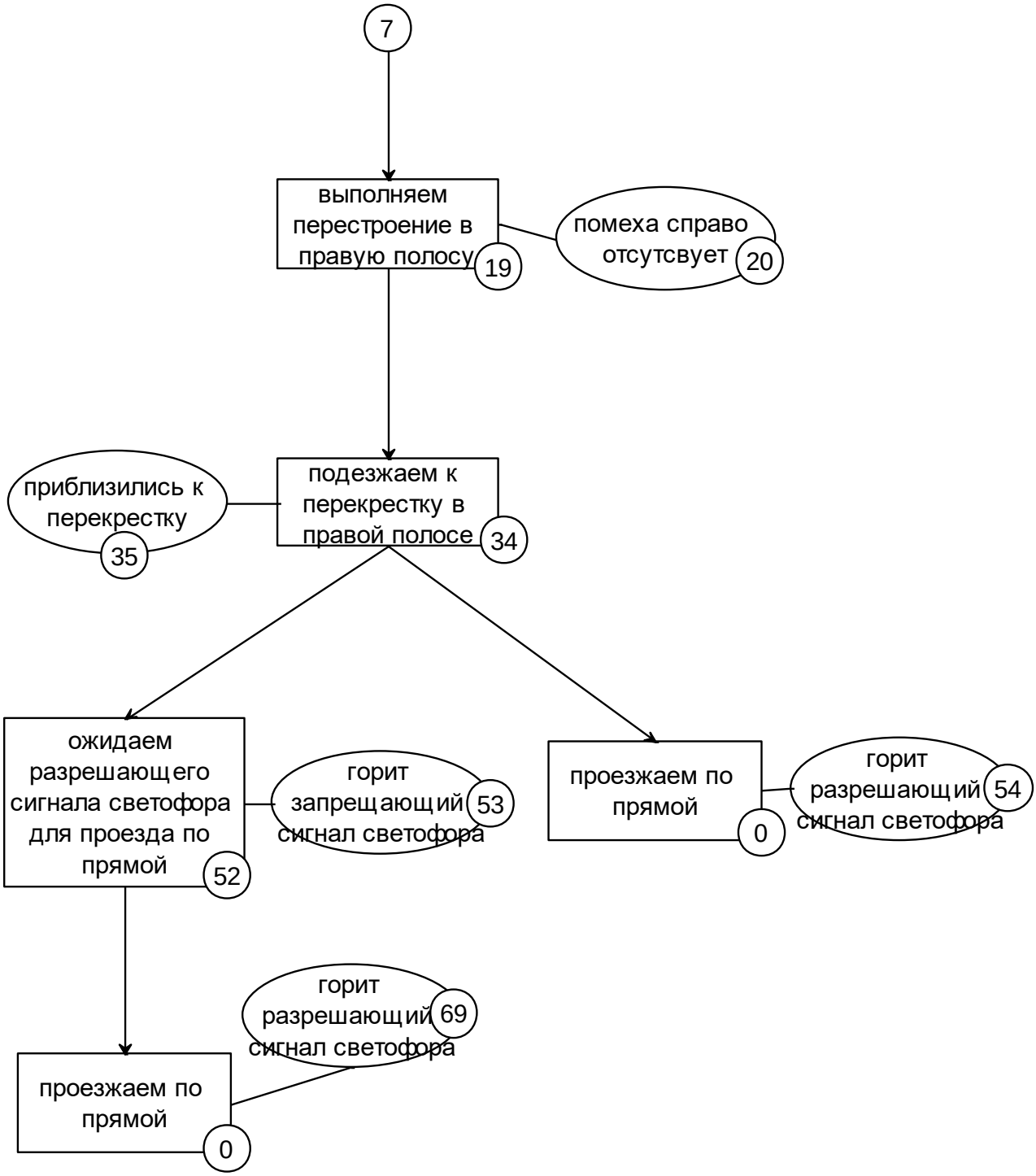


Рисунок 3.10 – Схема переходів частина 6

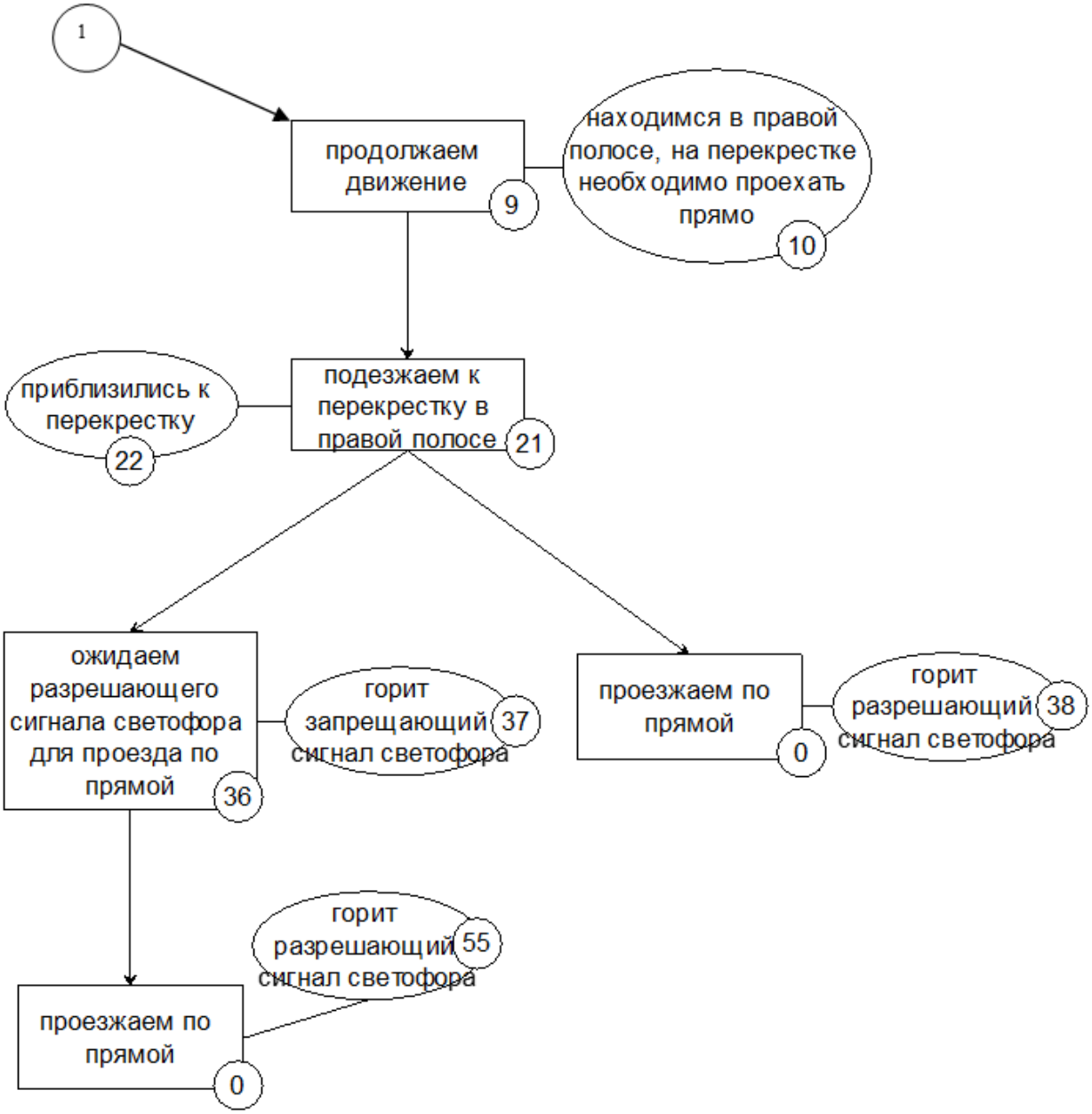


Рисунок 3.11 – Схема переходів частина 7

3.3.5 Використані принципи проектування

Для створення структури системи було використано принципи об’єктно-орієнтованого програмування та принципи SOLID:

– абстракція для виділення в об’єкті, що моделюється, важливого для вирішення конкретного завдання за предметом, в кінцевому рахунку – контекстне розуміння предмета, формалізуються у вигляді класу [9];

– інкапсуляція для швидкої і безпечної організації власне ієрархічної керованості: щоб було достатньо простої команди «що робити», без одночасного уточнення як саме робити, так як це вже інший рівень управління;

– успадкування для швидкої і безпечної організації родинних понять: щоб було достатньо на кожному ієрархічному кроці враховувати тільки зміни, що не дублюючи все інше, врахованих на попередніх кроках [9];

– поліморфізм для визначення точки, в якій єдине управління краще розпаралелити або навпаки – зібрати воєдино;

– принцип єдиного обов'язку (Single responsibility principle): Кожен об'єкт має виконувати лише один обов'язок [8];

– принцип відкритості/закритості (Open/closed principle): Програмні сутності повинні бути відкритими для розширення, але закритими для змін. Розширення певного класу/інтерфейсу може здійснюватися через його успадкування;

– принцип підстановки Лісков (Liskov substitution principle): Об'єкти в програмі можуть бути заміненими їх нащадками без зміни коду програми [8];

– принцип розділення інтерфейсу (Interface segregation principle): Багато спеціалізованих інтерфейсів краще за один універсальний. Інтерфейс може бути поділений на спеціалізовані ще на стадії проектування, заради майбутньої гнучкості програмних компонентів;

– принцип інверсії залежностей (Dependency inversion principle): Залежності всередині системи будуються на основі абстракцій, що не повинні залежати від деталей; навпаки, деталі мають залежати від абстракцій. Модулі вищих рівнів не мають залежати від модулів нижчих рівнів [8].

3.4 Розробка інтерфейсу користувача

3.4.1 Розробка структури екранів системи

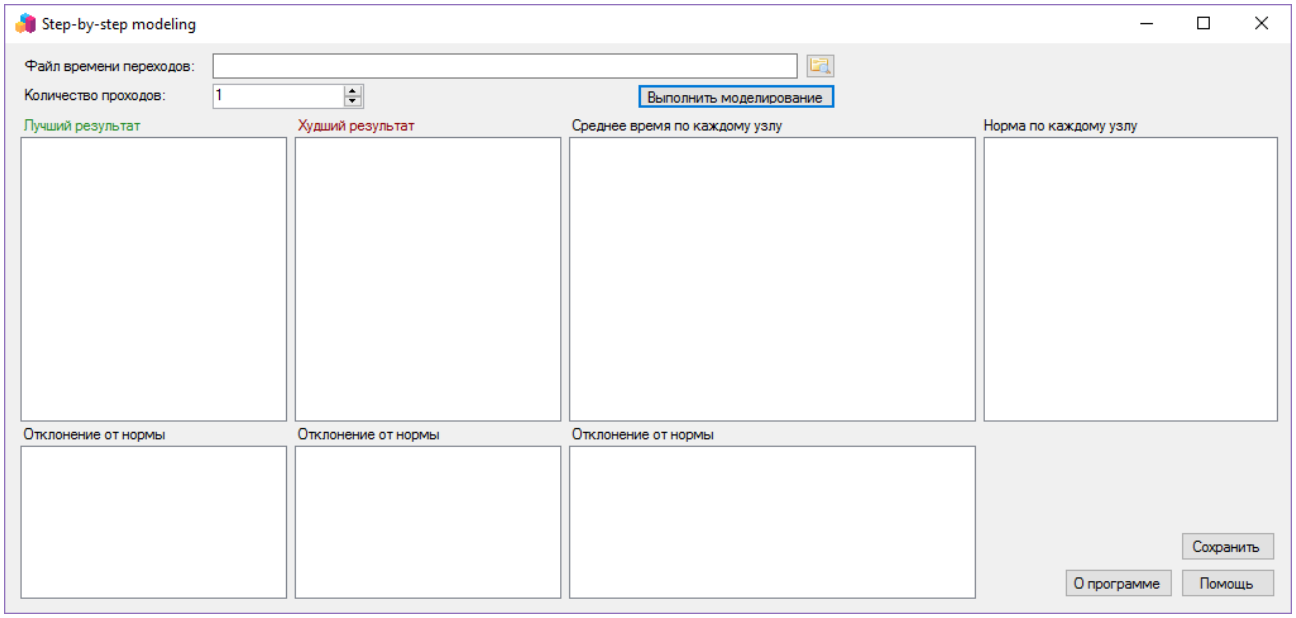


Рисунок 3.12 – Прототип головної сторінки

Інтерфейс був написаний на мові C#.

Об’єкти що були використанні при розробці інтерфейсу та можливі дії над ними зображені у таблиці 3.1.

Таблиця 3.1 – Об’єкти що були використанні при розробці інтерфейсу та можливі дії над ними

назва об’єкту	тип	дії над об’єктом
1	2	3
label	контейнер	Вивід підписів
textBox3	контейнер	Введення посилання на файл
listBox1	контейнер	Вивід результатів
listBox2	контейнер	Вивід результатів

Продовження таблиці 3.1

1	2	3
listBox3	контейнер	Вивід результатів
listBox4	контейнер	Вивід результатів
listBox5	контейнер	Вивід результатів
listBox6	контейнер	Вивід результатів
listBox7	контейнер	Вивід результатів
openFileDialog1	засіб керування	Вибір файлів для завантаження
button1	засіб керування	Необхідний для взаємодії з програмою
button4	засіб керування	Необхідний для взаємодії з програмою
numericUpDown	засіб керування	Необхідний для задання числових значень

3.4.2 Реалізація інтерфейсу користувача

Нижче наведено рисунки, які ілюструють результати розробки інтерфейсу системи, див. рис. 3.13 – 3.19.

На рис. 3.13 зображена головна форма програми що включає в себе усе необхідне для роботи з програмним засобом, а саме, поля вводу посилання на файл часу переходів, поле для вибору кількості повторень моделювання, та декілька віконець для виводу усіх отриманих результатів.

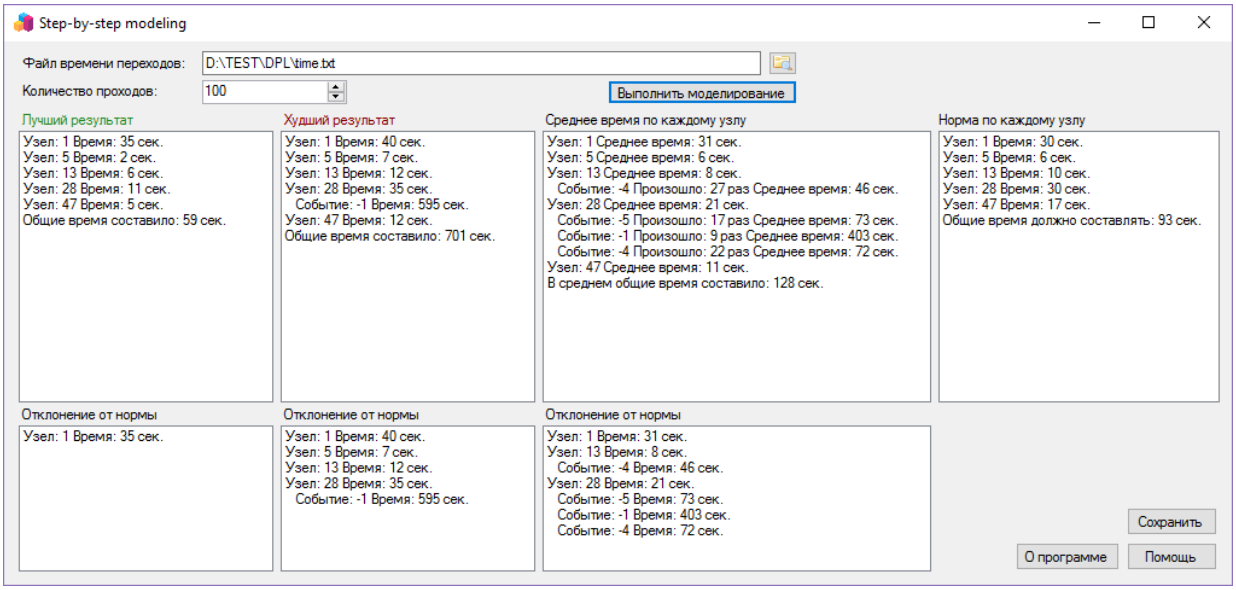


Рисунок 3.13 – Головна форма програми

На рис. 3.14 зображена головна форма програми при умові натискання клавіші «Помощь».

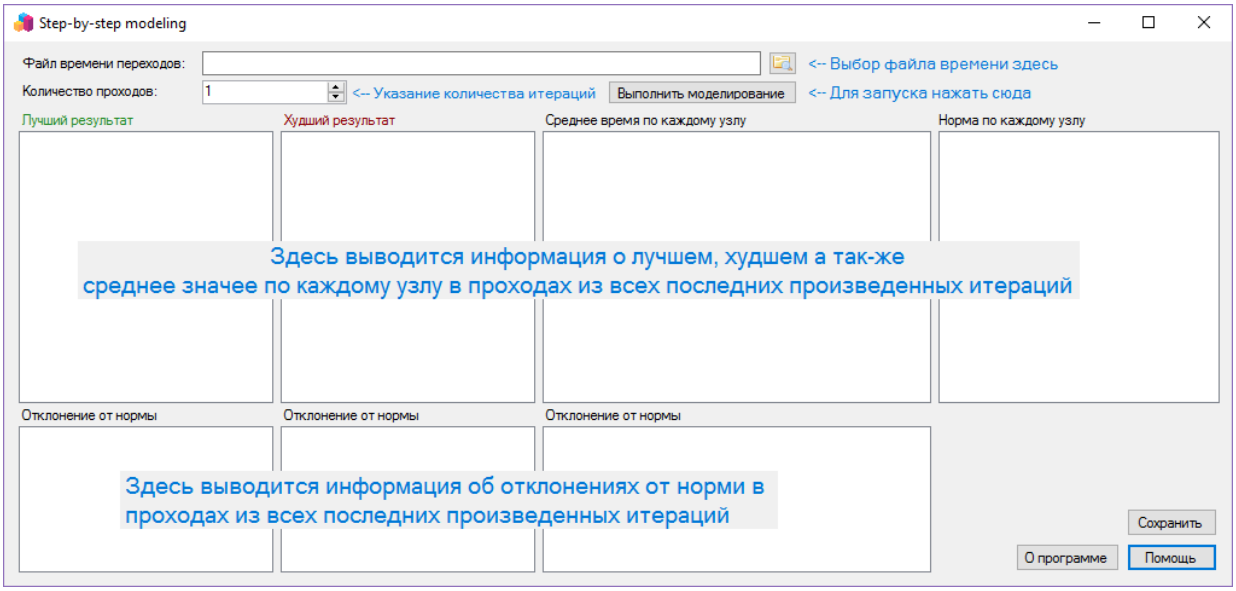


Рисунок 3.14 – Головна форма програми

На рис. 3.15 зображена вікно з інформацією про програму, для виклику необхідно натиснути клавішу «О программе».

На рис. 3.16 зображена вікно в котрому відбувається вибір файлу з часом.

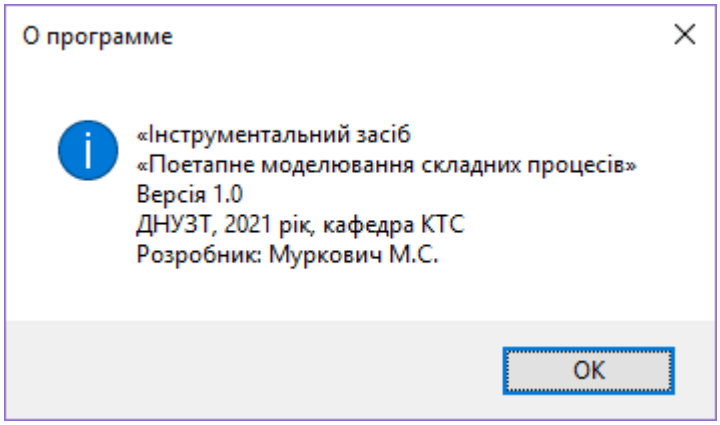


Рисунок 3.15 – Вікно «О программе»

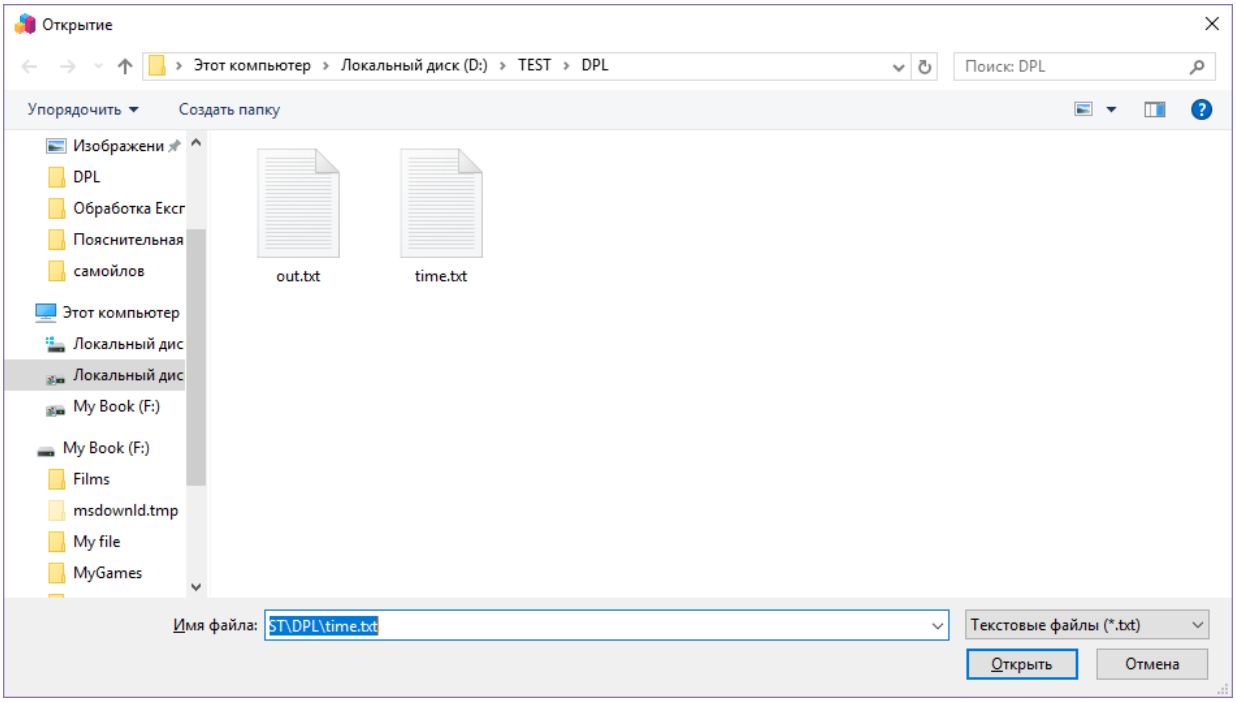


Рисунок 3.16 – Вікно вибору файлу з часом

На рис. 3.17 зображена вікно що буде виведено після успішного завершення моделювання.

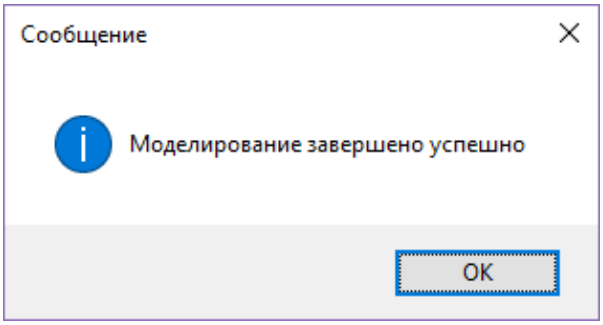


Рисунок 3.17 – Вікно повідомлення про успішне завершення моделювання

На рис. 3.18 зображена вікно в котрому виконується збереження файлу.

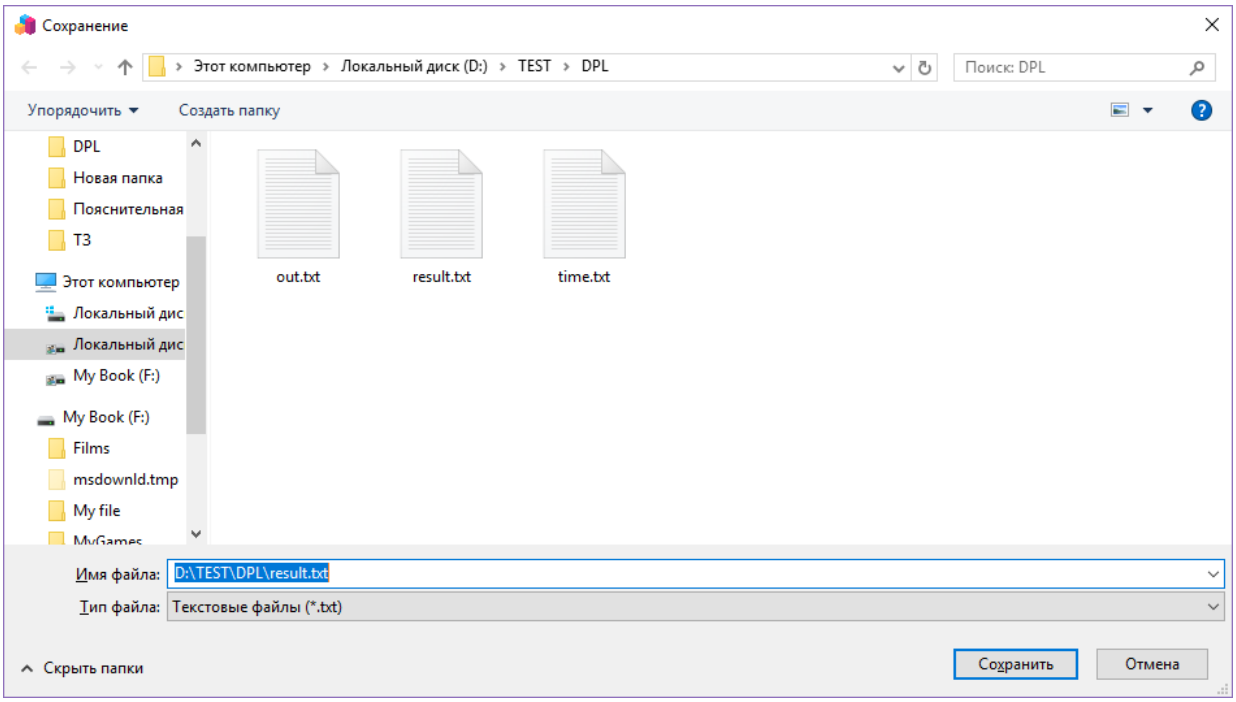


Рисунок 3.18 – Вікно збереження кінцевого файлу

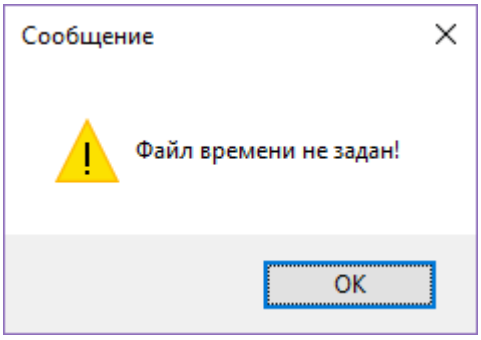


Рисунок 3.19 – Вікно повідомлення про помилку

На рис. 3.19 зображена вікно що буде виведено у разі не вказання файлу часу перед початком моделювання.

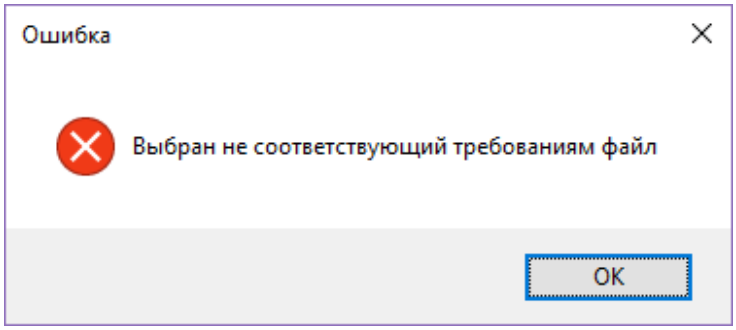


Рисунок 3.20 – Вікно повідомлення про помилку

На рис. 3.20 зображена вікно що буде виведено у разі вказання не вірного файлу часу перед початком моделювання.

3.5 Тестування та налагодження програми

3.5.1 Аналіз методів тестування та відлагодження

В розробці програмного забезпечення одним з найважливіших етапів є тестування та відлагодження програми. Цілю цього етапу є находження та виправлення помилок що було допущено при написанні. Також завдяки цьому етапові можуть бути найдені більш глобальні помилки що були допущені при проектуванні програмного засобу, а саме логіки. У випадку находження таких помилок виправлення роботи програми може забрати багато часу, тому слід проводити тестування такого роду не після завершення написання програмного продукту, а під час самого процесу написання.

Також необхідно зазначити що неможливо бути в повній мірі впевненим що після проходження цього етапу усі помилки що містяться у програмі були виправленні або інакше кажучи, завжди існує можливість находження нової помилки, але в цей же час цю можливість можливо зменшити до мінімуму.

Для ефективного тестування рекомендується використовувати що відносяться як до методів «чорної скриньки», так і до методів «білої скриньки».

В даному випадку було використано методи покриття операторів, а також метод покриття переходів для тестування коректності роботи частини програмного продукту що написано на мові програмування Prolog. Для тестування другої частини було використано метод припущення про помилку.

3.5.2 Тестування частини системи методом покриття операторів та методом покриття переходів

У ході тестування системи методом покриття операторів та методом покриття переходів було складено тест-кейс, у склад котрого входили деяка кількість тестів що дозволила виконати повне покриття усіх операторів та

переходів. Кожний тест представляє з себе одноразовий запуск відповідної частини програмного продукту з специфічними вхідними даним.

Таким чином маючи набір з деякої кількості різних файлів з різною вхідною інформацією забезпечується повне покриття усіх методів та переходів що містяться у частині програмного продукту що створено на мові програмування Prolog. У ході тестування цієї частини програми були виявлено декілька незначних помилок, що були одразу виправлені, оскільки вони відносились до частини що відповідала за формування зовнішнього вигляду проміжного файлу, а отже мало лише не значний вплив на роботу програми.

3.5.3 Тестування системи методом припущення про похибку

Друга частина програмного продукту, а саме частина що була написана на мові програмування C# була протестована методом припущення про помилку, це метод добре підходить поставленим цілям, бо має меншу трудомісткість та добре підходить для тестування простих лінійних функцій та зв’язків між методами. У парі з цим було використано декілька Unit-тестів для перевірки правильності роботи самих алгоритмів всередині методів. Оскільки результатом роботи деяких алгоритмів можуть бути випадкові числа то провести тестування цілковито тільки за допомогою Unit-тестів не можливо.

У ході тестування цієї частини програми були виявлено декілька помилок, що відносились до частини з завантаження вхідних файлів та їх розбору для подальшої обробки отриманої інформації, ці помилки були критичні для правильної роботи програми, але все ж вони були швидко виправлені. У частині що відповідала за моделювання помилок виявлено не було.

Висновки до розділу 3

Виконано проектування архітектури інструментального засобу для дослідження моделювання складних технологічних процесів. В ході проектування системи використовувалися SOLID-принципи та принципи ООП.

Проектом забезпечується можливість подальшої модернізації, оптимізації або адаптації до необхідностей середовища.

Завдяки тому що при розробці було використано загально прийняті принципи та стандарти, будь-якому спеціалісту у відповідній галузі буде зрозуміла архітектура та внутрішня структура програмного засобу.

Було створено розділ з описом проектування інтерфейсу користувача При розробці інтерфейсу користувача були використані стандарті елементи. Інтерфейс взагалом простий та зрозумілий, при необхідності є можливість вивести підказки до інтерфейсу, інтерфейс дружньою реагує на всі дії користувача та при необхідності повідомляє про помилки або результат дій користувача.

Виконано тестування усіх частин системи за допомогою методів чорної та білої скриньки, а саме методів покриття операторів, методів покриття переходів, Unit-тестів та методу припущення про помилку.

Система готова до впровадження та використання.

4 ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ВИКОРИСТАННЯ МЕТОДУ ПОЕТАПНОГО МОДЕЛЮВАННЯ ЗА ДОПОМОГОЮ ДЕКОМПОЗИЦІЇ ТА РЕАЛІЗАЦІЄЮ НА МОВАХ ПРОГРАМУВАННЯ C# ТА PROLOG ДЛЯ МОДЕЛЮВАННЯ СКЛАДНИХ ПРОЦЕСІВ

4.1 Підготовка до експерименту

Для проведення експерименту необхідно підготувати вхідні дані, а саме, набір правил переходів, в залежності від сфери процесу що досліджується.

Для демонстрації була вибрана сфера моделювання процесів автомобільних перехресть, а саме процес проїзду Т-образного перехрестя з світлофорним регулюванням.

Також необхідно підготувати набори вхідних фактів стану об’єкту що досліджується, на демонстраційному прикладі це легкове авто, тобто в нашому випадку набором фактів може бути побудований маршрут навігатором та ряд інших факторів навколишнього середовища, наприклад присутність або відсутність зустрічного руху або стан світлофора, інше.

Оскільки ми маємо раду працювати з моделюванням процесу проїзду перехрестя, але це може знадобиться і в інших сферах, нам необхідно врахувати випадкові події що можуть відбутися під час проїзду перехрестя, тобто під час модулювання процесу. Цю інформацію можна дістати за допомогою спостереження за об’єктом моделювання або взяти із статистичної інформації якщо така існує.

Після збору усієї необхідної інформації, її необхідно сформувати у файли котрі зможе розпізнати пролог при зчитуванні.

Після того як усе програмне забезпечення встановлене та налаштоване, вхідні файли підготовлені, можливо приступити до проведення експерименту.

Для більшої наочності було проведено декілька експериментів на базі одного середовища, але з використанням різних вхідних даних імітуючи один процес але при різних подіях.

Перший набір вхідної інформації наведено на рис. 4.1.

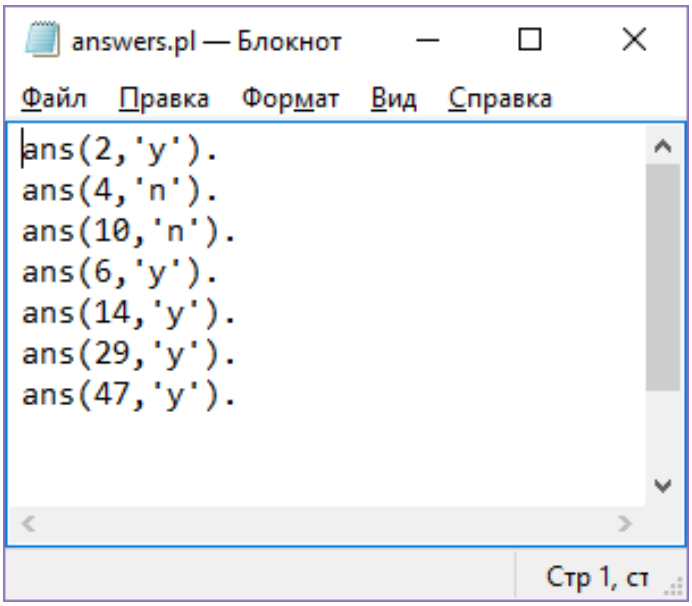


Рисунок 4.1 – Перший набір вхідних значень стану об’єктів

Другий набір вхідної інформації наведено на рис. 4.2.

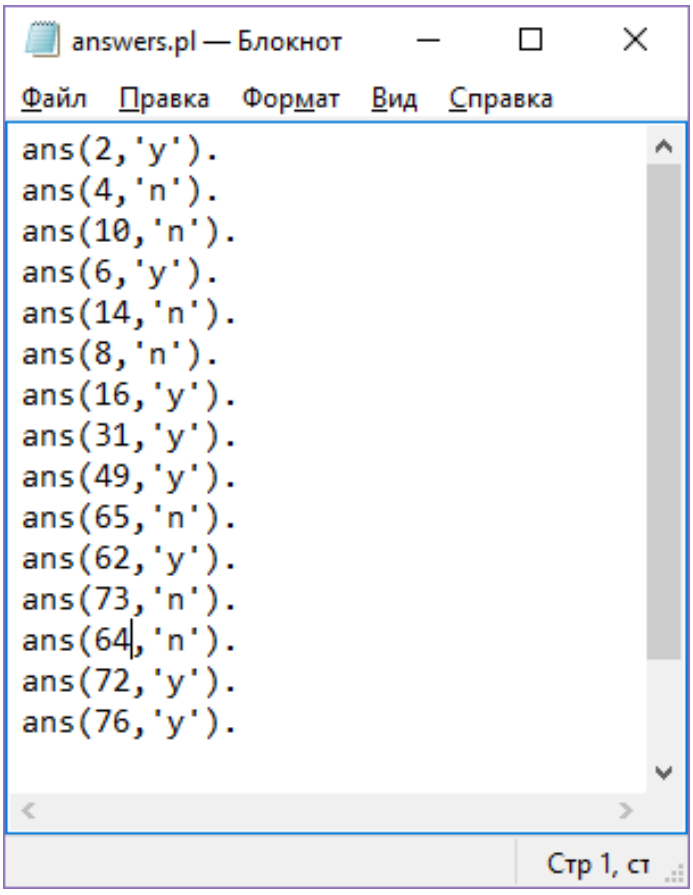


Рисунок 4.2 – Другий набір вхідних значень стану об’єктів

4.2 Проведення експерименту

Проведення експерименту полягає в запуску інструментального засобу та проведення моделювання за підготовленою інформацією.

Для цього необхідно:

1 Запустити інструментальний засіб.

2 Вибрати файл в котрому знаходиться час за котрий виконується перехід з одного стану в інший або виконується той або інший процес.

3 Вказати кількість ітерацій моделювання для етапу імітаційного моделювання.

4 Запустити моделювання та чекати на закінчення моделювання.

5 Отримати результати.

6 Проаналізувати отримані результати на правдоподібність та відповідність інформації про процес, котра була підготовлена за заздалегідь.

4.3 Результати експерименту

Результатом експерименту є статистична інформація: інформація о найкращій, найгіршій та середньо статистичній ітерації з вказанням у кожній слабких місць що відхиляються від норми, нормою є середнє значення проходження кожного вузлу та ітерації в цілому, що залежить від вказаних у вхідних файлах проміжків часу на кожний вузол.

Результати проведених експериментів зображено на рис 4.3 – 4.8.

При проведенні першого, другого та третього експериментів було використано однакові вхідні дані окрім кількості повторень моделювання, як результат є можливість побачити тенденцію, при котрій наочно видно що при збільшенні кількості ітерацій збільшується точність моделювання.

При проведенні четвертого, п’ятого та шостого експериментів було використано другий набір вхідних даних, що дозволяє порівняти результати при різних реалізаціях одного процесу.

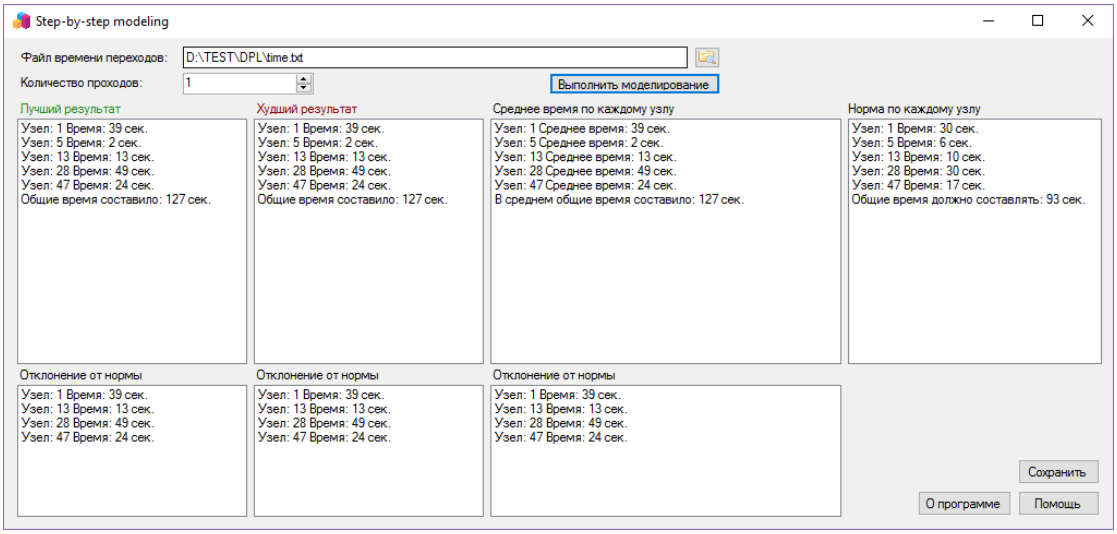


Рисунок 4.3 – Результати першого експерименту

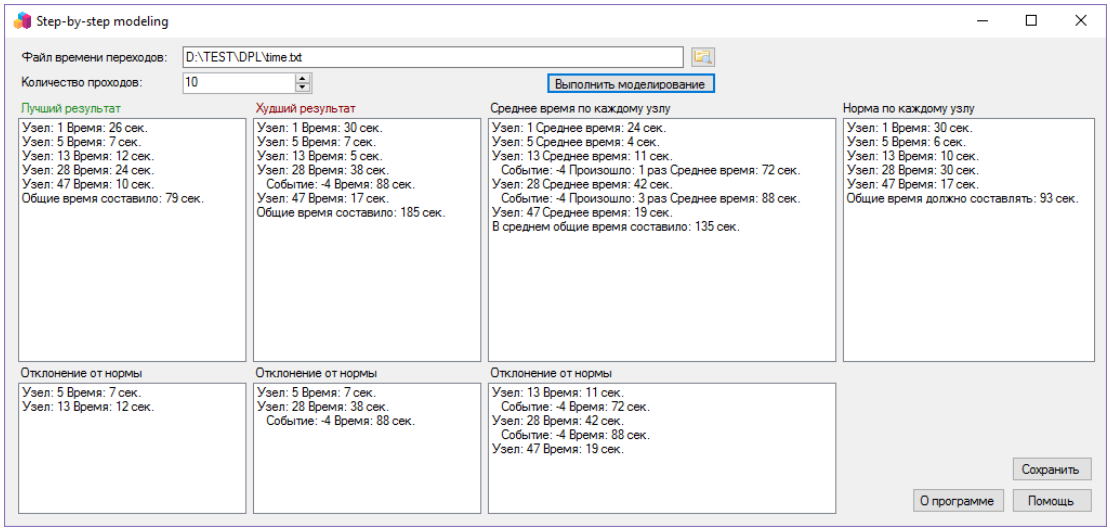


Рисунок 4.4 – Результати другого експерименту

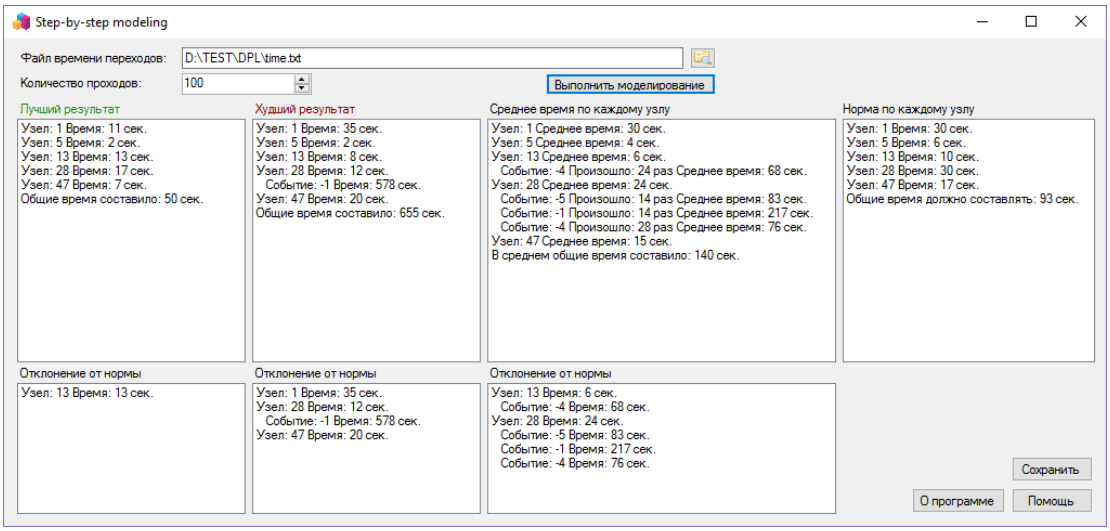


Рисунок 4.5 – Результати третього експерименту

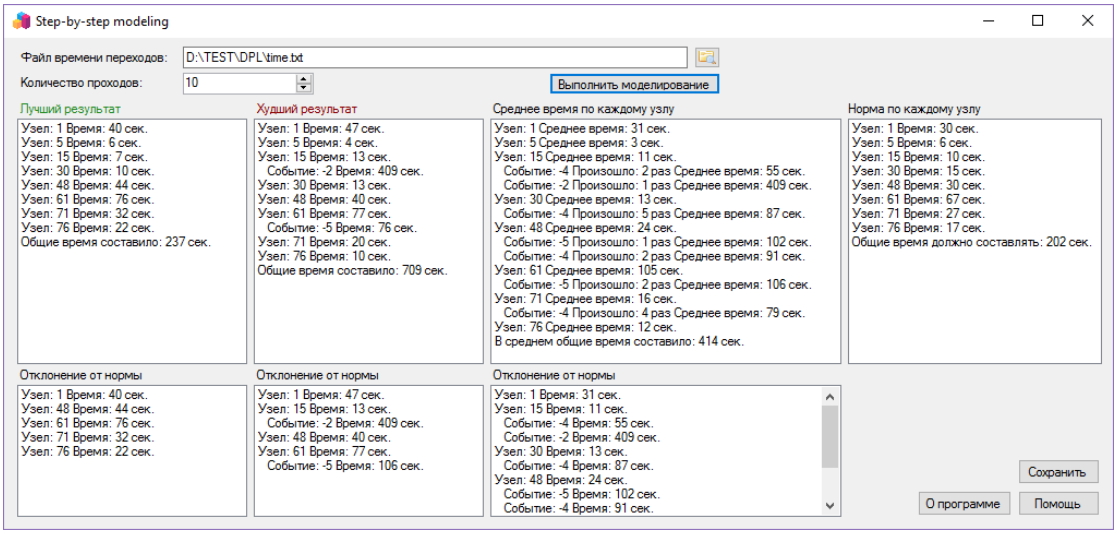


Рисунок 4.6 – Результати четвертого експерименту

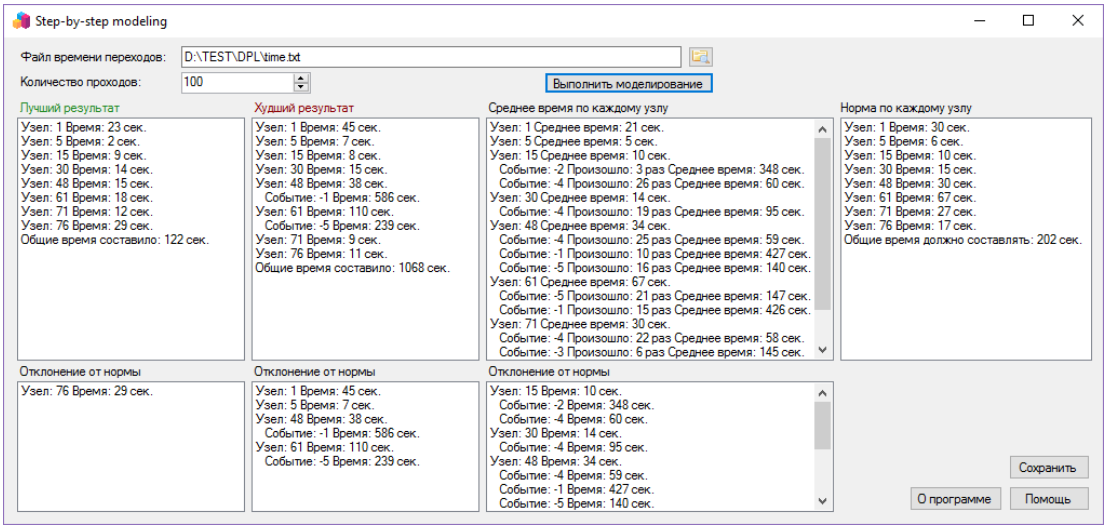


Рисунок 4.7 – Результати п’ятого експерименту

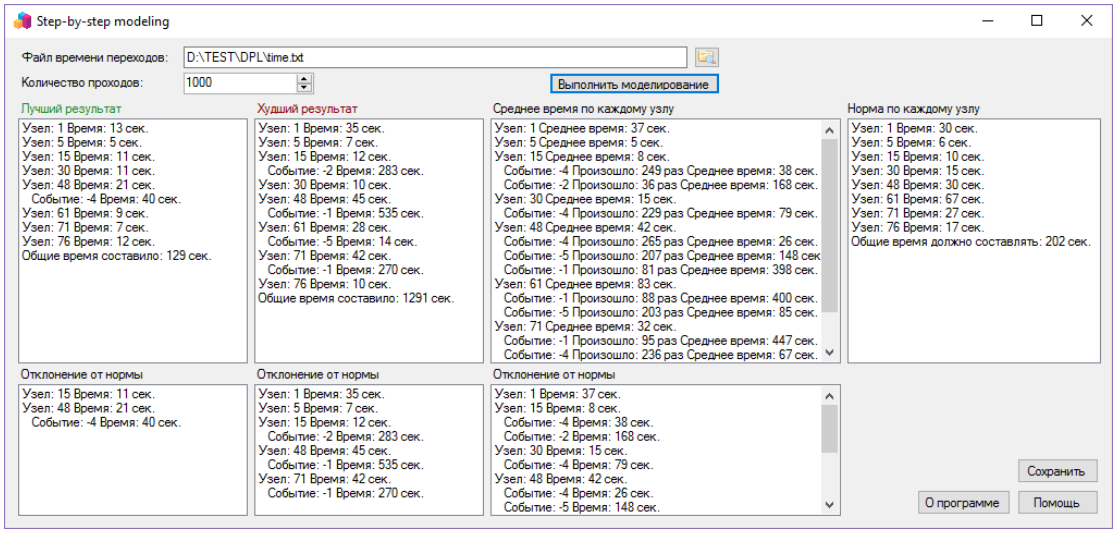


Рисунок 4.8 – Результати шостого експерименту

Висновки до розділу 4

У розділі описано процес досліду та результати аналізу методу поетапного моделювання складних процесів.

Висновки що були отримані після проведення експерименту:

– Підхід має право на існування, та їм можливо користуватися для моделювання складних процесів для отримання інформації о часовій ефективності, знаходження слабких місць та закономірностей у виниканні випадкових подій.

– Під час проведення дослідження стало зрозуміло, що використання цього підходу може добре вписатися у рамках систем в котрих необхідний постійний контроль у режимі реального часу, оскільки даний засіб можливо модифікувати для цього додаванням наборів датчиків що будуть постійно посилати інформацію до системи або оборудувати додатковою системою що буде надавати вже готові пакети інформації. Тобто цей підхід може добре використовуватися у системах контролю реального часу.

– Також цей підхід можливо модифікувати у протилежному напрямку, тобто для використання у мануальному режимі, шляхом модифікації системи створення та збору інформації необхідної для моделювання, а отримані результати можуть бути корисні при побудові нового процесу або модифікації будь-якого іншого процесу, також теоретично можливе використання цього підходу для роботи з бізнес процесами, але для підтвердження цього необхідне більше детальне дослідження цього питання, що не очікувалося у рамках цього дослідження.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона праці — це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я та працездатності людини в процесі праці.

Відповідно до статті 18 Закону України "Про охорону праці", працівник зобов'язаний «знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту, проходити у встановленому порядку попередні та періодичні медичні огляди» [11].

5.1 Аналіз умов праці на робочому місці

Впровадження ЕОМ у різні галузі виробництва позитивно вплинуло на умови праці, її якість та продуктивність. Разом з тим робота за монітором, електронною обчислювальною машиною(ЕОМ), ноутбуком та персональним комп’ютером (ПК) має низку чинників, які у разі порушення правил експлуатації можуть негативно впливати на стан здоров'я користувачів.

Користувачі повинні знати потенційно шкідливі та небезпечні виробничі чинники під час роботи за монітором, ПК, ноутбуком та виконувати вимоги НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» і ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ» та НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями» щодо виключення або зменшення їх негативного виливу.

5.1.1 Аналіз шкідливих та небезпечних виробничих факторів

Аналіз шкідливих та небезпечних виробничих факторів проводять згідно з ГОСТ 12.1.003-74. Небезпечними та шкідливими називають фактори, що

призводять до раптового погіршення здоров’я людини, або навіть смерті, під час її трудової діяльності.

Згідно з класифікацією ГОСТ 12.1.003-74 до числа небезпечних та шкідливих факторів, що виникають при роботі з ПК слід віднести:

– нервово-психічні перевантаження (розумове перевантаження, монотонність праці, емоційні перевантаження);

– фізичне перевантаження;

– невідповідність параметрів мікроклімату робочої зони санітарним нормам;

– недостатня або надмірна освітленість робочої зони;

– несприятливе забарвлення стін та підлоги, віддзеркалення;

– робота з комп’ютером [12].

Робоче місце оснащене ПК, принтером, монітором та іншими периферійними пристроями. Розміщення принтера або інших пристроїв введення-виведення інформації на робочому місці має забезпечувати добру видимість екрану, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті від 900 до 1300 мм, по глибині від 400 до 500 мм. Під матричні принтери потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

Площа, виділена для одного робочого місця ПК повинна складати не менше 6 $м^2$, а об'єм — не менше 20 $м^3$. Робочі місця з моніторами відносно світлових прорізів повинні розміщуватися так, щоб природне світло падало збоку, переважно зліва.

Режим праці та відпочинку.

При організації праці, пов'язаної з використанням ПК, для збереження здоров'я працівника передбачаються:

– перерви для відпочинку і вживання їжі;

– перерви для відпочинку й особистих потреб;

– додаткові перерви.

5.1.2 Вимоги до приміщень, розміщення в них моніторів, ПК, ноутбуків та організації лінії робочих місць

Об'ємно-планувальні рішення будівель і приміщень для роботи з моніторами, ПК, ноутбуками мають відповідати вимогам чинних нормативних актів, зокрема ДБН В.2.2-28-2010 «Будинки адміністративного та побутового призначення», ДСанПіН 3.3.2.007-98, НПАОП 0.00-7.15-18 та іншим і мати ступінь вогнестійкості не нижчу II.

Заборонено розміщувати робочі місця з моніторами, ПК та ноутбуками у підвальних приміщеннях, на цокольних поверхах, поряд з приміщеннями, в яких рівні шуму та вібрації перевищують допустимі значення (поряд з механічними цехами, майстернями тощо), з мокрими виробництвами, з вибухо-пожежо-небезпечними приміщеннями категорій А і Б, а також над такими приміщеннями або під ними.

Площу приміщень визначають із розрахунку, що на одне робоче місце вона має становити не менше ніж 6 м^2, а об'єм не менше ніж 20 м^2 з урахуванням максимальної кількості осіб, які одночасно працюють у зміні.

Приміщення мають бути оснащені природним і штучним освітленням відповідно до ДБН В.2.5-28:2018. Приміщення мають бути обладнані системами водяного опалення, кондиціонування або припливно-витяжною вентиляцією відповідно до ДБН В.2.5-67:2013. Заземлені конструкції приміщення (батареї опалення, водопровідні труби, кабелі з заземленим відкритим екраном тощо) надійно захищені діелектричними щитками або сітками від випадкового дотику [13].

Для всіх споруд і приміщень, у яких експлуатують монітори, ПК і ноутбуки визначають категорію з вибухопожежної та пожежної безпеки відповідно до ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків, установок за вибухопожежною та пожежною небезпекою» і НАПБ В.01.053-2016/520 «Правила пожежної безпеки в галузі зв'язку» та клас вибухонебезпепечних зон відповідно до НПАОП 0.00-1.32-01 «Правила будови

електроустановок. Електрообладнання спеціальних установок». Відповідні позначення наносять на вхідні двері приміщення. Крім того, приміщення з моніторами, ПК і ноутбуками оснащують системою пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 од. на кожні 20 м^2 площі приміщення. Підходи до засобів пожежогасіння мають бути вільними.

Робочі місця з моніторами, щодо світлових прорізів розміщують так, щоб природне світло падало збоку, переважно зліва.

Екран монітора і клавіатура мають розміщуватися на оптимальний відстані від очей користувача, але не ближче 600 мм з урахуванням розміру алфавітно-цифрових знаків і символів. Відстань від екрана до ока працівника залежить від діагоналі екрана та наведена у таблиці 5.1.

Таблиця 5.1 – Відстань від екрана до ока працівника

Діагональ екрана	Відстань від екрана
35/38 см (14"/15")	від 600 до 700 мм
43 см (17")	від 700 до 800 мм
48 см (19")	від 800 до 900 мм
53 см (21")	від 900 до 1000 мм

Розміщення екрана монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом ±30° від лінії зору працівника.

Обладнання й організація робочих місць з моніторами, ПК і ноутбуками мають забезпечувати відповідність усіх елементів робочого місця та їх розміщення ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги», характеру й особливостям діяльності. Конструкція робочого місця має забезпечити підтримання оптимальної робочої пози й оптимальне

розміщення на робочій поверхні використовуваного обладнання (екрану, клавіатури, принтера) та документів.

Робочий стіл, як правило, обладнують підставкою для ніг ширшою не менше 300 мм, глибиною не менше 400 мм з можливістю регулювання по висоті в границях 150 мм та кутом нахилу опорної поверхні в границях 20°. Підставка має бути з рифленою поверхнею і бортиком по передньому краю заввишки 10 мм.

Робоче сидіння (стілець, крісло) має складатися з таких основних елементів: сидіння, спинки, стаціонарних або змінних підлокітників. Стілець має бути підйомно-поворотним, таким, що регулюється за висотою, кутом нахилу сидіння і спинки, за відстанню спинки до передньою краю сидіння, висотою підлокітників. Крок регулювання для лінійних розмірів — від 15 до 20 мм, для кутових — від 2 до 5°. Зусилля регулювання — не більше 20 Н.

Клавіатуру слід розташовувати на поверхні столу або на спеціальній регульованій за висотою робочій поверхні окремій віл столу на відстані від 100 до 300 мм від краю, ближчого до працівника. Кут нахилу клавіатури має бути у границях від 5 до 15°.

Розміщення принтера або іншого пристрою вводу-виводу інформації на робочому місці має забезпечувати гарну видимість екрана монітора і зручність ручного керування в зоні досяжності моторного поля.

Робочі місця слід оснащувати пюпітром (тримачем) для документів, що легко перемішувати. Його встановлюють вертикально (або з нахилом) на тому ж рівні та відстані від очей користувачів, що і монітор.

5.1.3 Вимоги до виробничого середовища приміщень з моніторами, ПК і ноутбуками

Згідно з Гігієнічною класифікацією праці за показниками шкідливості та небезпечності чинників виробничого середовища, тяжкості та напруженості трудового процесу умови праці користувачів ПК і ноутбуків мають відповідати І класу (оптимальним) або ІІ класу (допустимим) умовам праці.

Приміщення з моніторами, ПК і ноутбуками мають бути забезпечені природним і штучним освітленням. Коефіцієнт природного освітлення (КПО) має бути не нижчим 1,5%. Розраховують площу світлових прорізів, яка забезпечує нормоване значення КПО в робочій зоні користувачів комп'ютерів, відповідно до ДБН В.2.5-28:2018.

Штучне освітлення має бути загальним, робочим і рівномірним. У випадку, коли робота переважно з документами, допускається додатково використовувати місцеве освітлення. Рівень освітленості в зоні розташування документів має бути в границях від 300 до 500 лк.

Систему загального освітлення має бути виконано у вигляді суцільних або переривчатих ліній світильників, що розмішують збоку від виробничих місць (переважно зліва), паралельно лінії зору працівників. Допускається застосовувати світильники таких класів світлорозподілу: світильники прямого світла — П; переважно прямого світла — Н; переважно відбитого світла — В.

У разі розташування моніторів, ПК по периметру приміщення лінії світильників мають бути розміщені локально над робочими місцями.

Уміст шкідливих речовин у повітрі робочої зони не має перевищувати ГДК відповідно до ГОСТ 12.1.005 — 88 уміст озону не більше 0,1 мг/м^3, вміст оксидів азоту — не більше 5 мг/м^3. уміст пилу — не більше 4 мг/м^3.

Параметри мікроклімату мають відповідати вимогам ДСН 3.3.6.042 «Санитарные нормы микроклимата производственных помещений».

Оптимальні величини температури, відносної вологості та швидкості руху повітря для приміщень з моніторами, ПК і ноутбуками наведені у таблиці 5.2.

Рівні іонізації повітря приміщень під час роботи з моніторами, ПК і ноутбуками наведено у таблиці 5.3.

Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот представлений в таблиці 5.4.

Таблиця 5.2 – Оптимальні величини температури, відносної вологості та швидкості руху повітря для приміщень з моніторами, ПК і ноутбуками

Період року	Категорія робіт	Температура повітря, °С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
холодний	Легка-1 а	від 22 до 24	від 40 до 60	0,1
	Легка-16	від 21 до 23	від 40 до 60	0,1
теплий	Легка-1а	від 23 до 25	від 40 до 60	0,1
	Легка-16	від 22 до 24	від 40 до 60	0,2

Таблиця 5.3 – Рівні іонізації повітря приміщень під час роботи з моніторами, ПК і ноутбуками

Рівні іонізації повітря	Кількість іонів у 1 $см^3$ повітря	
	n^+	n^-
Мінімально необхідні	400	600
Оптимальні	від 1500 до 3000	від 3000 до 5000
Максимально допустимі	50000	50000

Таблиця 5.4 – Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот

Види трудової діяльності	Рівні звукового тиску, дБ, в октавних смугах із середньо геометричними частотами, Гц									Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв
	31.5	63	125	250	500	1000	2000	000	000	
Програмісти ПК	6	1	1	4	9	5	2	0	8	50
Оператори в залах оброблення інформації на ПК та оператори комп'ютерного набору	6	3	4	8	3	0	7	5	4	65

Для підтримання оптимальних значень параметрів повітря робочої зони потрібно застосовувати вентиляцію приміщень, кондиціонування повітря, використовувати установки або прилади зволожування та штучної іонізації.

У приміщеннях з ПК рівні звукового тиску, рівні звуку та еквівалентні рівні звуку мають відповідати вимогам ДСН 3.3.6.037-99 та ДСанПіН 3.3.2.007 98. Устаткування, яке є джерелом шуму, слід розташовувати поза приміщеннями з ПК і ноутбуками.

Значення напруженості електромагнітних полів на робочих місцях із моніторами мають відповідати нормативним значенням ДСанПіН 3.3.2.007-98 [14].

Допустимі рівні електромагнітного випромінення радіочастотного діапазону наведено у таблиці 5.5.

Таблиця 5.5 – Допустимі рівні електромагнітного випромінення радіочастотного діапазону

Діапазон частот, МГц	Допустимі рівні ЕМП	
	Електрична складова *Е, В/м*	Магнітна складова Н, А/м
від 0.06 до 3,0	50	5
від 3.0 до 30,0	20	-
від 30.0 до 50.0	10	0.3
від 50,0 до 300,0	5	-

5.2 Дії працівників (персоналу) в аварійних (надзвичайних) ситуаціях

Типова інструкція щодо дій персоналу підприємств при загрозі або виникненні надзвичайних ситуацій.

Аварійною ситуацією треба вважати порушення меж та/або умов безпечної експлуатації об'єкта (устаткування, обладнання), яке не перейшло в аварію, за якого всі несприятливі впливи джерел небезпеки на персонал,

населення та навколишнє середовище утримуються в прийнятних межах за допомогою відповідних технічних засобів.

Аварією слід уважати раптову подію, таку як потужний викид небезпечних речовин, пожежа або вибух, у наслідок порушення експлуатації підприємства (об’єкта), яка призводить до негайної та/або наступної загрози для життя та здоров'я людей, довкілля, матеріальних цінностей на території підприємства та/або за його межами [15].

Джерелами небезпечних і шкідливих виробничих чинників можуть бути:

– нерегламентовані режими роботи технологічного устаткування,
– транспортні засоби,
– вантажопідіймальне устаткування,
– механізми, обладнання,
– деталі та вироби, що рухаються;
– устаткування, що працює під тиском;
– електромережі, електрифіковане устаткування та інструменти;
– інженерні комунікації;
– роботи, що призводять до психофізіологічних перевантажень;
– токсичні речовини;
– помилкові дії працівників,
– аварії.

У разі виникнення аварійної ситуації чи аварії працівники зобов’язані діяти тверезо й спокійно, не панікувати, точно й оперативно слідувати вказівкам керівництва підприємства, осіб, відповідальних за цивільний захист (цивільну оборону) та техногенну безпеку, протипожежну безпеку, охорону праці, а також представників аварійно-рятувальних, та газорятувальних, пожежних, медичних підрозділів.

Кожний працівник, який першим виявив загрозу виникнення аварійної ситуації, повинен негайно припинити роботу та подати команду "СТОП!"

Команда "СТОП!", подана будь-яким працівником, має негайно бути виконаною всіма працівниками, котрі її почули.

У випадку виникнення аварійних ситуацій або пожежі кожний працівник мусить:

1. Припинити роботу (якщо це дозволено технологічним процесом виробництва);

2. Якнайшвидше сповістити про аварію (пожежу) керівника та відповідальну посадову особу;

3. Приступити до ліквідації (локалізації) аварії (пожежі) наявними засобами;

4. За необхідності викликати інші аварійно-рятувальні (пожежну, медичну газорятувальну тощо) служби.

Працівники та службовці, які входять до складу невоєнізованих (цивільних) формувань цивільної оборони, у разі виникнення аварійної ситуації повинні прибути на пункт збору формувань та взяти участь у локалізації й ліквідації аварії [15].

Керівництво підприємства, а також особи, відповідальні за цивільний захист (цивільну оборону) та техногенну безпеку, протипожежну безпеку, охорону праці, зобов’язані в разі виникнення аварійної ситуації (аварії):

1. Перевірити та продублювати повідомлення про аварію (пожежу), довести це до відома керівника підприємства;

2. Оцінити умови, з'ясувати кількість і місцезнаходження людей, захоплених аварією, за потреби вжити заходів щодо оповіщення працівників, населення про аварію;

3. Під час загрози для життя людей негайно організувати їх рятування (евакуацію), використовуючи для цього наявні сили й засоби;

4. Вжити негайних заходів щодо локалізації та ліквідації аварії;

5. Вжити заходів щодо оточення району аварії (небезпечної зони);

6. Забезпечити виведення з небезпечної зони людей, які не беруть безпосередньої участі в ліквідації аварії;

7. Обмежити допуск людей та транспортних засобів до небезпечної зони;

8. Контролювати правильність дій персоналу щодо рятування людей, локалізації й ліквідації аварії;

9. У разі необхідності виконати: відключення електроенергії (за винятком систем протипожежного захисту), зупинку транспортувальних пристроїв, агрегатів, апаратів, перекриття сировинних, газових, парових комунікацій, зупинку систем вентиляції в аварійному приміщенні (за винятком пристроїв протидимового захисту) та вжити інших заходів, що сприяють ліквідації (локалізації) аварії (пожежі);

10. Сприяти діями аварійно-рятувальних, газорятувальних, пожежних, медичних підрозділів щодо рятування людей, локалізації, ліквідації аварії на підприємстві;

11. Організувати надання медичної допомоги потерпілим, харчування та відпочинок осіб, які беруть участь у ліквідації аварії;

12. Інформувати безпосереднє керівництво, уповноважені органи державної влади та місцевого самоврядування про хід і характер аварії, потерпілих у ході рятувальних робіт.

У разі дій щодо локалізації (ліквідації) аварійної ситуації (аварії) потрібно:

1. Постійно враховувати реальні можливості й ресурси підприємства, накопичений персоналом підприємства досвід дій під час аварійних ситуацій та аварій, ступінь небезпеки для життя та здоров'я людей, довкілля;

2. Організувати оповіщення й зустріч підрозділів аварійно-рятувальних та інших служб, забезпечити узгодженість дій персоналу підприємства й підрозділів аварійно-рятувальної, медичної та інших служб;

3. У випадку необхідності організувати евакуацію персоналу (частини персоналу) та матеріальних цінностей.

Якщо того вимагає характер аварійної ситуації (аварії), особи, які задіяні в оповіщенні, локалізації та ліквідації аварійної ситуації чи аварій, повинні використовувати спеціальний одяг та взуття, промислові засоби захисту органів дихання [15].

У разі нещасного випадку:

1. Працівник, який його виявив, або сам потерпілий повинен терміново повідомити безпосереднього керівника та/або іншу посадову особу підприємства та вжити заходів щодо надання потерпілим необхідної допомоги;

2. Працівник, який його виявив, потерпілий чи його безпосередній керівник невідкладно повідомляє особу, відповідальну за стан цивільного захисту (цивільної оборони) та техногенної безпеки на підприємстві;

3. Слід терміново організувати медичну допомогу потерпілому, а за необхідності — доставити його до лікувального закладу;

4. Повідомити про те, що сталося, керівника підприємства або його заступника, а також представника трудового колективу підприємства (професійної спілки);

5. Зберегти до прибуття комісії з розслідування обстановку на робочому місці та устаткування в такому стані, у якому вони були на момент події (якщо це не загрожує життю й здоров'ю інших працівників і не призведе до більш тяжких наслідків);

6. Вжити заходів щодо недопущення подібних випадків у ситуації, що склалася.

У разі виникнення пожежі слід діяти згідно з Інструкцією з пожежної безпеки на підприємстві.

У випадку припинення подачі електроенергії треба від'єднати електричні прилади від мережі.

Виявивши обрив електропроводів, пошкодження їхньої ізоляції, не варто торкатися їх. Необхідно повідомити про це безпосереднього керівника та/або особу, відповідальну за стан цивільного захисту (цивільної оборони) та техногенної безпеки на підприємстві.

Евакуація має забезпечити захист працюючого персоналу в разі неможливості вжиття інших заходів цивільного захисту під час виникнення надзвичайних ситуацій. Рішення про евакуацію приймається керівником підприємства або особою, яка його заміщує. Підставою для прийняття рішення про практичне здійснення евакуаційних заходів є фактичні показники стану довкілля у випадку надзвичайної ситуації та відповідне рішення Кабінету Міністрів України, органів місцевої державної влади, територіальних органів МНС [15].

У разі евакуації:

1. На керівника підприємства покладається:

1.1 планування й проведення евакуації працівників;

1.2 подання (за потребою) до відповідних транспортних органів розрахунків необхідності в транспортних засобах для вивезення працівників до безпечного району;

1.3 контроль за плануванням, підготовкою й проведенням евакуаційних заходів;

1.4 визначення та підготовка безпечного району для розміщення евакуйованих працівників;

2. На особу, відповідальну за стан цивільного захисту (цивільної оборони) та техногенної безпеки на підприємстві, покладають:

2.1 оповіщення працівників, уточнення даних про транспортні засоби, що виділяються для евакуації, термін їхньої подачі, маршрути та порядок руху;

2.2 організацію й контроль посадки евакуйованих працівників на транспортні засоби та відправку колон;

2.3 інформування керівництва підприємства та вповноважених органів влади про хід евакуації.

Задоволення мінімуму життєвих потреб працівників, які потерпіли (можуть потерпіти) від наслідків надзвичайних ситуацій, включає, зокрема:

1. їх тимчасове розміщення (розселення) у безпечних районах;

2. організацію харчування, медичного обслуговування та санітарно-епідеміологічного нагляду в районах тимчасового розселення;

3. забезпечення потерпілих одягом, взуттям і товарами першої необхідності;

4. надання фінансової допомоги потерпілим.

Посадові особи, на яких чинними нормативно-правовими актами покладаються обов'язки щодо локалізації (ліквідації) аварійної ситуації (аварії), несуть відповідальність згідно із законодавством України [15].

5.2.1 Дії щодо забезпечення електробезпеки

Відповідно до ст. 13 розд. III Закону України «Про охорону праці» від 14 жовтня 1992 р. № 2694-XII роботодавець зобов’язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці [16].

Заходи щодо виконання вимог електробезпеки офісних працівників регламентують наступні нормативні документи:

– Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.12-05), затверджене наказом Державного комітету України з нагляду за охороною праці України від 26 січня 2005 р. № 15;

– Правила технічної експлуатації електроустановок споживачів, затверджені наказом Міністерства палива та енергетики України від 25 липня 2006 р. № 258 (далі — ПТЕЕС);

– Правила безпечної експлуатації електроустановок споживачів, затверджені наказом Міністерства праці та соціальної політики України, Комітету по нагляду за охороною праці від 9 січня 1998 р. № 4;

– Правила пожежної безпеки в Україні, затверджені наказом Міністерства внутрішніх справ України від 30 грудня 2014 р. № 1417.

Відповідно до п. 2.2 розд. ІІ Загальних вимог стосовно забезпечення роботодавцями охорони праці працівників, затверджених наказом Міністерства надзвичайних ситуацій від 25 січня 2012 р. № 67, роботодавець має забезпечити повну і вичерпну інформацію працівників та їх уповноважених представників з питань охорони праці про можливі небезпечні ситуації, про вжиті заходи для їх запобігання або їх ліквідації та про дії працівників у аварійних ситуаціях.

Електробезпека — система організаційних та технічних заходів і засобів, що забезпечують захист людей від шкідливого та небезпечного впливу електричного струму, електричної дуги, електромагнітного поля і статичної електрики [16].

Тобто електробезпека — це відсутність будь-яких факторів з боку електроустановки, які можуть створити загрозу і небезпеку життю і здоров'ю людини. Не варто під терміном «електроустановка» розуміти щось таке, що може знаходитись поза межами офісу. Адже, наприклад, настільна лампа це також електроустановка. Тобто пристрій, в якому є перетворення електричної енергії в той чи інший вид енергії (світлову, механічну, теплову), і буде вважатись електроустановкою.

Заходи електробезпеки, на які необхідно звернути особливу увагу:

1. Облаштування електромережі, зокрема:

– правильний розподіл навантаження на всі приміщення офісу;

– правильний розподіл електромережі за призначенням (наприклад: освітлення — це одна група, робоча зона — інша);

– якість самих комплектуючих електромережі (розетки, вимикачі, лампи, світильники);

– чи є потенціал для збільшення навантаження (на випадок створення додаткових робочих місць чи розширення компанії);

– використання офісного обладнання, в якому електроенергія застосовується за призначенням згідно з технічними рекомендаціями виробника.

2. Виважений підхід до питання використання стаціонарних або мобільних електрогенераторів для зменшення енергозалежності:

– консультація зі спеціалістом електротехнічного фаху (якщо не призначений відповідальний за електрогосподарство) щодо вибору генератора відповідно до потреб енергоспоживання;

– якщо електрогенератор стаціонарний — необхідно виконати вимоги, зазначені в п. 9.1-9.20 розд. VIII ПТЕЕС;

– якщо електрогенератор мобільний (бажано з вмонтованою автоматикою введення резерву) — правильно виконати підключення з урахуванням можливих наслідків у випадку появи струму в зовнішній електромережі.

Згідно з п. 1.2-1.6 розд. IV ПТЕЕС керівник повинен створити відповідні служби щодо забезпечення безпечної та надійної експлуатації електроустановок [16].

3. Проведення інструктажів з охорони праці з питань електробезпеки.

Одним з важливих заходів попередження електротравматизму є проведення інструктажів з охорони праці з питань електробезпеки. Доцільно проводити інструктажі у формі співбесіди.

Роботи, пов’язані з обслуговуванням електромережі, повинні виконуватись підготовленим персоналом відповідно до п. 2.2, 2.3 розд. IV ПТЕЕС.

5.2.2 Дії щодо забезпечення пожежної безпеки

Організаційні заходи щодо забезпечення пожежної безпеки до адміністративно-офісних приміщень викладені відповідно до розділу ІІ «Правил пожежної безпеки в Україні» НАПБ А.01.001-2014.

Керівник підприємства повинен визначити обов'язки посадових осіб щодо забезпечення пожежної безпеки.

Керівник підприємства повинен призначити відповідальних за пожежну безпеку приміщень, а також за утримання й експлуатацію засобів протипожежного захисту.

Відповідним документом (наказом, інструкцією тощо) повинен бути встановлений протипожежний режим.

Для кожного приміщення об'єкта мають бути розроблені та затверджені керівником об'єкта інструкції про заходи пожежної безпеки.

На об'єктах з постійним або тимчасовим перебуванням на них 100 і більше осіб або таких, що мають хоча б одне окреме приміщення із одночасним перебуванням 50 і більше осіб, котрі мають два поверхи і більше, у разі одночасного перебування на поверсі більше 25 осіб, а для одноповерхових – більше 50 осіб, мають бути розроблені і вивішені на видимих місцях [17].

У разі зміни планування або функціонального призначення будинків (приміщень), технології виробництва, штатного розкладу персоналу плани евакуації та інструкції повинні бути відкориговані плани (схеми) евакуації людей на випадок пожежі.

У приміщеннях на видимих місцях біля телефонів слід вивішувати таблички із зазначенням номера телефону для виклику пожежно-рятувальних підрозділів.

Будинки, приміщення мають бути забезпечені відповідними знаками безпеки.

Знаки безпеки, їх кількість, а також місця їх встановлення повинні відповідати вимогам ДСТУ ISO 6309:2007, (ISO 6309:1987, IDT).

Роботу новоутвореного підприємства чи використання суб’єктом господарювання об’єктів нерухомості необхідно розпочинати на підставі поданої декларації відповідності матеріально-технічної бази суб’єкта господарювання вимогам законодавства з питань пожежної безпеки відповідно до статті 57 Кодексу цивільного захисту України.

Посадові особи та працівники повинні пройти навчання та перевірку знань з питань пожежної безпеки у порядку, встановленому постановою Кабінету Міністрів України від 26 червня 2013 року № 444 «Про затвердження Порядку здійснення навчання населення діям у надзвичайних ситуаціях».

Усі працівники при прийнятті на роботу на робочому місці повинні проходити інструктажі з питань пожежної безпеки [17].

Висновки до розділу 5

В даному розділі було освітлено теми охорони праці та типові інструкції для персоналу при виникненні надзвичайних ситуацій.

Описані умови праці персоналу у офісних приміщеннях, вимоги до мікроклімату, освітленню, випромінюванню, вимоги до робочого місця, комплектація робочого місця. Описані вимоги пожежно-вибухової безпеки у приміщеннях де використовуються екранні пристрої, приведені вимоги та можливі заходи до електробезпеки.

Усі вимоги та правила що приведені відповідають чинному законодавству, для кожної вимоги приведено посилання на нормативний документ до якого воно відноситься.

ВИСНОВКИ

У роботі було виконано дослідження методу поетапного моделювання складних технічних та технологічних процесів.

Створено інструментальний засіб необхідний для проведення моделювання складних процесів для отримання статистичної інформації.

Під час розробки використовувалися як сучасні технології так і технології що вже добре себе зарекомендували у продовж років. Завдяки використанню об’єктно-орієнтованого підходу у парі з логічним програмуванням система має просту структуру розбиту на модулі що дає змогу для подальшого вдосконалення або модифікації, зміни.

Під час аналізу сфери використання було розглянуто основні принципи, необхідності та проблеми у моделюванні складних процесів, було оглянуто достатня кількість програмних аналогів.

Інтерфейс програмного додатку був спроектований у мінімалістичному стилі, у результаті чого було отримано простий інтерфейс з інтуїтивно зрозумілим користуванням, та можливостями для додавання нових можливостей у випадку розширення функціоналу.

Інструментальний засіб було протестовано та відлагоджено з використанням сучасних засобів та методик тестування, що дало змогу отримати стабільний продукт що має малий відсоток помилок.

Було проведено дослідження економічної доцільності розроблюваного додатку. Виконані розрахунки дають змогу підтвердити доцільність розробки та виконання відповідного дослідження сфери.

У пояснювальній записці були розглянуті питання з охорони праці, а саме вплив комп’ютерів на здоров’я користувачів, режими праці та відпочинку під час роботи за ПК та вимоги до приміщень з комп’ютерною технікою.

У результаті проведення дослідження, підкріплене експериментами, отримані вичерпні відповіді на поставлені питання та отримано

інструментальний засіб для проведення дослідження, котрий також можливо використовувати при рішенні подібних задач у різних сферах у майбутньому.

Після закінчення розробки системи була написана уся необхідна програмна документація. Наявність якісного керівництва оператора дозволить кінцевим користувачам швидко вивчити можливості програми та почати її використовувати, за умови наявності необхідних знань предметної сфери. Задокументований опис та текст програми спростить подальший супровід системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петросов Д.А., Ігнатенко В.А. / Поэтапное моделирование технологических процессов с использованием интеллектуального структурно-параметрического синтеза / Д.А. Петросов, В.А. Ігнатенко, «Фундаментальные исследования» в розділі «Технические науки», 28.12.2017 р.

2. Розін М.Д., Свічкарьов В.П. / Проблемы системного моделирования сложных процессов социального взаимодействия / М.Д. Розін, В.П. Свічкарьов, «Инженерный вестник Дона», 2012 р.

3. Введение в MVC [Електронний ресурс] / METANIT.COM – Сайт о программировании. – URL: https://metanit.com/sharp/aspnet5/3.1.php, (дата звернення 16.11.2021 р.)

4. Microsoft Visual Studio [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio, (дата звернення 16.11.2021 р.)

5. SWI-Prolog [Електронний ресурс] / Progopedia – URL: http://progopedia.ru/implementation/swi-prolog/, (дата звернення 16.11.2021 р.)

6. C Sharp [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://ru.wikipedia.org/wiki/C_Sharp, (дата звернення 16.11.2021 р.)

7. Пролог (язык программирования) [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://ru.wikipedia.org/wiki/Пролог_(язык_программирования), (дата звернення 16.11.2021 р.)

8. SOLID (об'єктно-орієнтоване програмування) [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://uk.wikipedia.org/wiki/SOLID_(об%27єктно-орієнтоване_програмування), (дата звернення 16.11.2021 р.)

9. Объектно-ориентированное программирование [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование, (дата звернення 16.11.2021 р.)
10. Одиночка [Електронний ресурс] / Рефакторинг.Гуру – URL: https://refactoring.guru/ru/design-patterns/singleton, (дата звернення 16.11.2021 р.)
11. Охорона праці [Електронний ресурс] / Вікіпедія — вільна енциклопедія. – URL: https://uk.wikipedia.org/wiki/Охорона_праці, (дата звернення 16.11.2021 р.)
12. Система стандартів безпеки праці. Небезпечні і шкідливі виробничі фактори. ГОСТ 12.1.003-74. — ГОСТ 12.1.003-74. — [Чинний від 1976—01—01]. — К. : Держспоживстандарт України, 2004.
13. Природне і штучне освітлення. ДБН В.2.5-28:2018. — ДБН В.2.5-28:2018. — [Чинний від 2019—03—01]. — К. : Державні будівельні норми України, 2018.
14. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. ДСанПІН 3.3.2.007-98. — ДСанПІН 3.3.2.007-98. — [Чинний від 1998—12—10]. — К. : Міністерство Охорони Здоров'я України, 2004.
15. Інструкція щодо дій персоналу підприємства в разі виникнення аварійної ситуації (аварій) [Електронний ресурс] / Охорона праці та пожежна безпека в Україні – URL: https://otipb.at.ua/load/civilna_bezpeka/instrukcija_shhodo_dij_personalu_pidpriemstva_v_razi_viniknennja_avarijnoji_situaciji_avarij/36-1-0-3372, (дата звернення 16.11.2021 р.)
16. Про заходи з електробезпеки для працівників офісу [Електронний ресурс] / Охорона праці та пожежна безпека в Україні – URL: https://oppb.com.ua/articles/pro-zahody-z-elektrobezpeky-dlya-pracivnykiv-ofisu, (дата звернення 16.11.2021 р.)
17. Організаційні заходи щодо забезпечення пожежної безпеки до адміністративно-офісних приміщень [Електронний ресурс] / ТЕХНОСПЕКТР-

СЕРВІС – URL: https://ts.kiev.ua/orhanizatsiini-zakhody-shchodo-zabezpechennia-pozhezhnoi-bezpeky/, (дата звернення 16.11.2021 р.)
18. Марков В. / Книга Современное логическое программирование на языке Visual Prolog 7.5 / В. Марков – СПб.: БХВ-Петербург, 2016. – 544 с.
19. Братко І. / Язык PROLOG (Пролог): алгоритмы искусственного интеллекта 3-е изд. / Пер. с англ. – М. : Издательский дом "Вильяме", 2004. – 640 с.
20. Клоксин У., Меллиш К. / Программирование на языке ПРОЛОГ 2-е изд. / У. Клоксин, К. Меллиш – Кембридж, 1984. – 334 с.
21. Прайс Марк Дж. / C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов 3-е изд. / Марк Дж. Прайс – СПб.: Питер, 2018. – 640 с.
22. Троелсен Є., Джепикс Ф. / Язык программирования C# 7 и платформы .NET и .NET Core 8-е изд. / Є. Троелсен, Ф. Джепикс – СПб.: ООО «Диалектика», 2019. – 1328 с.
23. Албахари Д., Абахари Б. / C# 7.0. Справочник. Полное описание языка / Д. Албахари, Б. Абахари – ООО «И.Д. Вильямс», 2013. – 281 с.
24. Щекатуров А.М. / Книга Методика моделирования динамики октокоптера / А.М. Щекатуров – ДМК Пресс, 2021. – 228 с.
25. Купер А., Рейманн Р., та ін. / Интерфейс. Основы проектирования взаимодействия. 4-е изд. / А. Купер, Р. Рейманн, та ін. – Питер, 2017. – 720 с.
26. Норман Д. / Дизайн привычных вещей. / Д. Норман – Вильямс, 2006. – 384с.
27. Петров А.В. / Моделирование процессов и систем. / А.В. Петров – Лань, 2015. – 288 с.
28. Ильин В.В. / Моделирование бизнес-процессов. Практическое использование ARIS / В.В. Ильин – Вильямс, 2006. – 176 с.
29. Барабаш М.С. / Компьютерное моделирование процессов жизненного цикла объектов строительства / М.С. Барабаш, Монография. - К.: Изд-во «Сталь», 2014. – 301 с.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ДОДАТОК А

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР
__.__.__

Методи поетапного моделювання складних процесів

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01208-01 ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
__.__.__

Керівник розробки
Олександра ГОРБОВА
__.__.__

Виконавець
Микола МУРКОВИЧ
__.__.__

Нормоконтролер
Олена КУРОП'ЯТНИК
__.__.__

2021

АНОТАЦІЯ

Документ 1116130.01208-01 «Методи поетапного моделювання складних процесів. Технічне завдання» є керівництвом для проектування та розробки програмного додатку для поетапного моделювання складних процесів, та входить до складу документації на проект.

Програмне забезпечення розробляться для проведення дослідження у сфері створення та оптимізації складних процесів, шляхом розрахунку часової ефективності складних процесів, формалізації процесів у вигляді дерева рішень та подальшого аналізу. В документі містяться основні вимоги до розробки програмного додатку та його функціональні можливості.

ЗМІСТ

ВСТУП

Розробка інструментального засобу для магістерської роботи «Методи поетапного моделювання складних процесів» необхідна для можливості дослідження часової ефективності складних процесів та можливості формалізації їх у вигляді дерева рішень, для подальшого аналізу.

Основні терміни:

– ПЗ – програмне забезпечення;

– ПП – програмний продукт;

– БД – база даних.

В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області. Під час моделювання технологічних процесів застосовують різні методи й підходи.

При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання спрощення процесу моделювання. На базі формалізованого підходу до побудови і дослідження моделей є доцільним створення програмного комплексу, що буде допомагати та прискорить діяльність фахівців.

Мається на увазі програмний продукт, що включає в себе можливості внесення необхідних наборів фактів об об’єктах, побудова дерева рішень та перевірка ефективності цього дерева у часі.

Основні завдання ПП:

– формалізація складних процесів за допомогою дерева рішень;

– перевірка часової ефективності складного процесу.

Таким чином для спеціалістів з’являється можливість більш комфортної та швидшої перевірки ефективності складних процесів.

Необхідність розробки виникла у результаті необхідності проведення дослідження щодо доцільності використання даного підходу та через відсутність на ринку аналогів з задовільною ціною та необхідним функціоналом, було прийнято рішення з створення нового ПЗ у відповідності з усіма вимогами.

Отриманий інструментальний засіб створено для проведення дослідження та може використовуватися у ДНУЗТ або будь-якому іншому підприємстві для аналізу ефективності складних процесів.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ № 690 ст. від 18.11.2020 ректора Дніпровського національного університету залізничного транспорту імені академіка В. Лазаряна «Про затвердження керівників та затвердження тем магістерських робіт».

Тема проекту: «Методи поетапного моделювання складних процесів».

Керівник дипломного проекту: доцент кафедри «КІТ» О.В. Горбова.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Основним функціональним призначенням програмного продукту є надання можливостей дослідження часової ефективності складних процесів або їх частин за допомогою побудови дерева рішень та подальшого імітаційного моделювання з урахуванням виникнення раптових подій.

2.2 Експлуатаційне призначення

Головне призначення програмного продукту полягає у наданні можливості проведення дослідження, перевірки можливості застосування поетапного моделювання у сфері оптимізації процесів з застосуванням дерева рішень. Програмний комплекс дозволить дослідити часову ефективність складного процесу та як результат знайти слабкі місця процесу.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

– введення вхідних даних шляхом завантаження файлу з наборами фактів за посиланням;

– побудова дерева рішення за допомогою мови логічного програмування Prolog;

– проведення імітаційного моделювання та збір статистичної інформації для дослідження часової ефективності складних процесів з урахуванням можливості виникнення раптових подій.

Вхідні дані

Метод організації вхідних даних полягає в наданні користувачу зручного інтерфейсу для введення даних у поля зі строгим дотриманням встановленого формату. Файли з наборами фактів що подаються на вхід мають мати формат файлу – «.pl». Файл з інформацією о часі проходження кожного вузлу, що подаються на вхід, мають мати формат файлу – «.txt».

Вхідні дані можна розділити на наступні підгрупи:

– цифрова інформація, кількість повторень моделювання, вводяться за допомогою користувацького інтерфейсу;

– посилання на файли, вводяться за допомогою користувацького інтерфейсу.

Вихідні дані

До вихідних даних відносяться:

– файл з побудованим деревом рішення;

– результати імітаційного моделювання, а саме статистична інформація о найкращому, найгіршому та середньому результаті проходженні кожного вузлу та усього дерева в цілому, а також інформацію про найслабкіші місця.

3.2 Вимоги до надійності

Програмний продукт має забезпечити стійку роботу, коректне виконання своїх основних функцій та цілісність і збереженість даних. Повинні виконуватися наступні вимоги:

– стійка робота при вводі некоректної вхідної інформації;

– відновлення працездатності програми не пізніше однієї год. після збою;

– наявність резервної копії бази даних на зовнішньому носії;

– наявність архівної копії тексту програми на зовнішньому носії;

– час роботи на відмову не менше 1500 години (відмовою системи слід вважати короткочасне порушення робочого стану системи, яке самоусувається при повторній спробі).

3.3 Умови експлуатації

Для нормального функціонування програмного продукту необхідно виконання наступних вимог:

– обчислювальна техніка, на якій буде експлуатуватися програма, повинна знаходитися в приміщенні з температурою повітря від 21º С до 25º С, вологістю повітря 40 – 60%, швидкістю руху повітря не більш 0,2 м/с;

– з метою полегшення надійного функціонування програми періодично проводити резервне копіювання даних на зовнішні носії;

– стан технічних пристроїв повинен задовольняти відповідні норми і вимоги;

– працювати з програмою може людина, яка володіє навичками роботи з комп’ютером на рівні користувача та ознайомлена з керівництвом користувача.

3.4 Вимоги до складу та параметрів технічних засобів

Мінімальна конфігурація, що забезпечить нормальну експлуатацію програмного продукту:

– процесор: Intel Core 2 Duo 2.4 GHz або краще;

– оперативна Пам'ять: 2 GB DDR2 або краще;

– вільний дисковий простір 50 Mb;

– наявність CD/DVD приводу або USB роз’ємну для встановлення необхідного ПЗ;

– монітор з роздільною здатністю екрану 1024x768;

– клавіатура;

– маніпулятор «миша».

3.5 Вимоги до інформаційної і програмної сумісності

Програма повинна виконуватися під керуванням ОС Windows XP/Windows7 або Windows10. Система повинна розроблятись в середовищі візуального програмування (наприклад Visual Studio).

3.6 Вимоги до маркування та упаковки

Програма може зберігатись на жорсткому диску або на знімних носіях (флеш-носії, CD диски). На упаковці повинно бути вказано назву продукту, його серійний номер, версія, вимоги до складу та параметрів технічних засобів, вимоги до інформаційної та програмної сумісності.

Приклад маркування наведено на рис. 3.1.

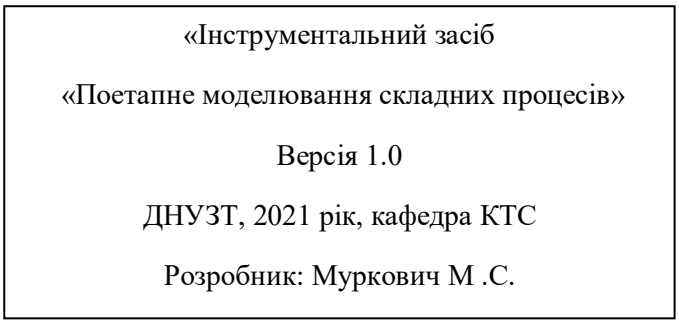


Рисунок 3.1 – Приклад маркування

3.7 Вимоги по транспортуванню та зберіганню

Транспортування програмного продукту може виконуватися шляхом його переносу на знімних інформаційних носіях чи по інформаційних каналах зв’язку через мережу Інтернет. Не допускається механічний вплив на носії. Інформаційні носії повинні зберігатися в сухому теплому місці, не допускати попадання прямих сонячних променів, температура повітря 18-40^{0}С, при вологості 50-70%, не допускати попадання пилу.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація на даний програмний продукт повинна включати наступні документи:

– специфікації програми;

– текст програми;

– опис програми;

– керівництво користувача. Керівництво експерту з побудови технологічних процесів.

5 ТЕХНІКО–ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Техніко–економічне обґрунтування (ТЕО) – це обов'язкова складова частина будь–якого інвестиційного проекту, тобто проекту, що потребує певних фінансових витрат. Основна мета розробки ТЕО – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку, економічну доцільність його розробки та впровадження.

Нова техніка, технологія, засоби автоматизації, що розробляються і впроваджуються у виробництво, повинні приносити певний корисний результат – ефект. Ефект може проявлятися у поліпшенні умов праці працюючих (соціальний), в зниженні шкідливого впливу виробництва на навколишнє середовище (екологічний), у підвищенні безпеки держави (оборонний), та, врешті, в економії витрат підприємства на виробництво продукції та збільшенні його прибутку (економічний).

Абсолютна величина економічного ефекту без співставлення його з витратами підприємства не дозволяє однозначно оцінити, наскільки вдалим виявився відповідний інноваційний проект. Таку оцінку дають показники економічної ефективності (прибутковості) проекту.

При впровадженні інвестиційного проекту підприємство несе разові витрати, пов'язані з розробкою проекту, а також з придбанням і налагодженням необхідного обладнання, засобів програмного забезпечення і таке інше.

Такі разові витрати називають капітальними витратами або інвестиціями. При використанні інновацій підприємство отримує певний ефект, що зазвичай виражається приростом прибутку. При розрахунках ефективності необхідно врахувати додаткові річні витрати підприємства, пов'язані з експлуатацією нового обладнання. Величина щорічного прибутку, додатково одержуваного підприємством за рахунок впровадження інвестиційного проекту, повинна бути достатньо високою у порівнянні з капітальними

витратами підприємства та у порівнянні з іншими можливими варіантами вкладення коштів у розвиток виробництва.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості (кількісний або якісний).

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Припустимо, що розмір програмного коду програмного засобу – 1240 рядків:

$$E = 2{,}4 \cdot 1{,}24^{1{,}05} \cdot 1 = 3{,}00 \tag{5.2}$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 3,00 людино-місяці.

Розрахуємо вартість розробки програмного забезпечення «Методичне забезпечення з навчальної дисципліни». Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;
- витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера–програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Рекомендована кількість виконавців – 1 *чол*; тривалість розробки – 3 місяців. Розрахунок зарплати проводиться по формі таблиці 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, *грн/міс*	Кількість		Сума зарплати, *грн*
			чол	місяців	
1	інженер-програміст	13000	1	3	39 000

Інформація взята з сайту ua.trud.com, див. рис. 5.1.

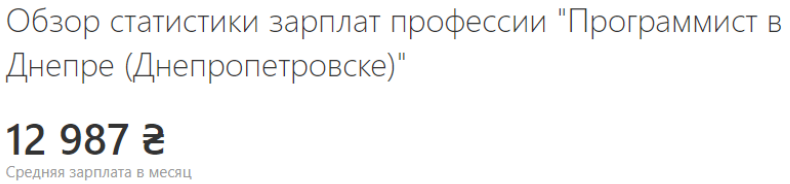


Рисунок 5.1 – Середня зарплата інженер-програмісту в Дніпрі

Описаний в проекті програмний продукт розроблений одним програмістом в період з 01.09.21 до 5.12.21, що складає 67 днів або 13,4 робочих тижнів. Витрати робочого часу приймемо 40 часів у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 73,8 *грн/год*. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} * N_{\text{тиж}} * N_{\text{год}}, \quad (5.3)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол*;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, *год*;

$$t_{\text{розробки}} = 1 \cdot 13{,}4 \cdot 40 = 536 \text{ чол/год}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{\text{КВ}}, \quad (5.4)$$

де $t_{\text{розробки}}$ – витрати праці у *чол/год;*

N – погодинна ставка;

$K_{КВ}$ – коефіцієнт кваліфікації програміста, приймаємо 0.75.

$$\text{ОЗП} = 536 * 73{,}8 * 0{,}75 = 29\,667{,}60 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати:

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%}, \quad (5.5)$$

$$C_{\text{соц}} = \frac{29\,667{,}60 * 22\%}{100\%} = 6\,526{,}87 \text{ грн.}$$

Отримані результати за (5.2) – (5.3) підсумовуються. Вони складають 36 194,47 *грн.* та визначають основні прямі витрати.

Накладні витрати враховують загально-господарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30–40 %) від суми прямих витрат:

$$C_{\text{накл}} = \frac{(\text{ОЗП} + C_{\text{соц}}) \cdot 35\%}{100\%}, \quad (5.6)$$

$$C_{\text{накл}} = \frac{(29\,667{,}60 + 6\,526{,}87) \cdot 35\%}{100\%} = 12\,668{,}06 \text{ грн.}$$

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– витрати на електроенергію;

– вартість витратних матеріалів;

– витрати на ремонт;

– заробітна плата ремонтника;

– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп’ютер і програмне забезпечення.

Витрати на електроенергію ($C_{ел}$) визначаються за формулою:

$$C_{ел} = \text{Р} \cdot \text{В} \cdot \text{Т}_{розр}, \quad (5.7)$$

де P – потужність комп’ютера та допоміжних споживачів електричної енергії, приймаємо 0,55 *кВт/год*;

B – вартість 1 *кВт/год* складає 1,68 *грн;*

Інформація взята з сайту yasno.com.ua, див. рис. 5.2.

Рисунок 5.2 – Вартість 1 *кВт/год* у Дніпрі

$T_{розр}$ – час роботи з ЕВМ, прийнято рівним робочому часу.

$$C_{ел} = 0{,}55 * 1{,}68 * 536 = 495{,}26 \text{ грн.}$$

Витрати на витратні матеріали ($C_{вм}$) протягом всього терміну експлуатації приблизно 10 % від вартості комп'ютеру. Вартість комп'ютеру приймаємо 25000 *грн*, дані взяті із сайту https://can.ua, див рис. 5.3, термін експлуатації – 2 роки.

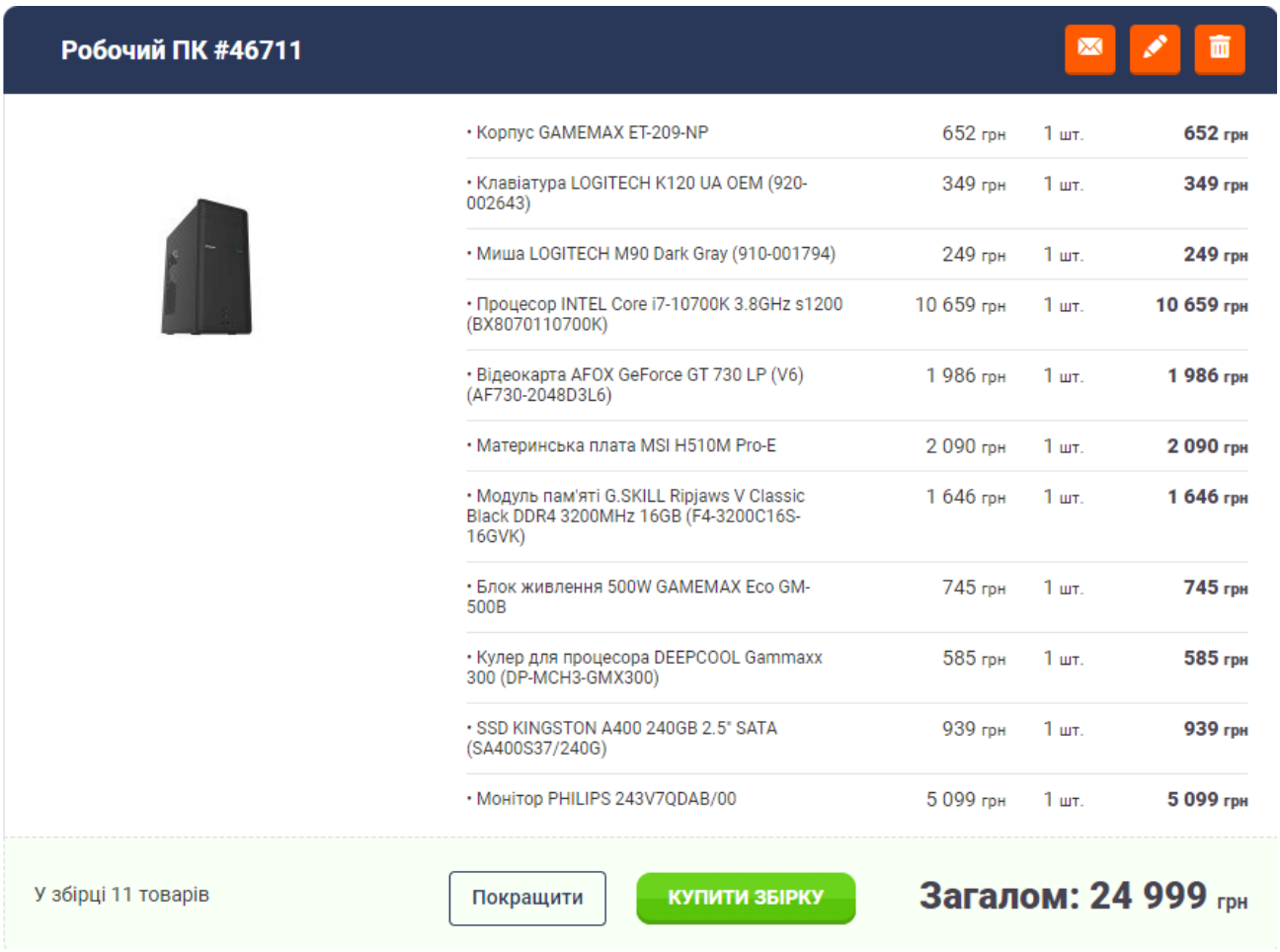


Рисунок 5.3 – Набір комплектуючих робочої машини з вартістю

Отже, можна визначити ці витрати за період створення програмного засобу:

$$C_{вм} = В_{ком} \cdot \frac{N_д}{N_{екс} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (5.8)$$

де $B_{ком}$ – вартість персонального комп’ютеру;

$N_{Д}$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

$$C_{вм} = 25000 * \frac{67}{2 \cdot 365} \cdot \frac{10}{100} = 229{,}45 \text{ грн}$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 10 комп’ютерів потрібен один інженер–системотехнік. Його середньомісячна заробітна плата приймається 7000 *грн*.

Інформація взята з сайту ua.trud.com, див. рис. 5.4.

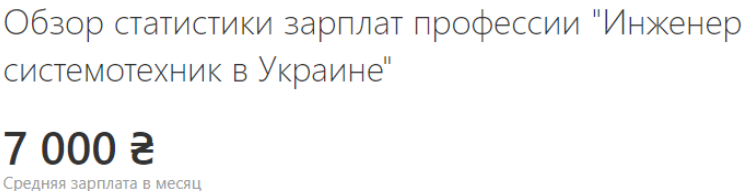


Рисунок 5.4 – Середня зарплата інженер-системотехнік в Дніпрі

Тоді в перерахунку на один комп’ютер його заробітна плата складає:

$$C_{рем} = \frac{C'_{рем}}{N_{ком}}, \quad (5.9)$$

де $C'_{рем}$ – середньомісячна заробітна плата;

$N_{ком}$ – кількість комп’ютерів на одного ремонтника.

$$C_{рем} = \frac{7000}{10} = 700 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($C_{ком}$) для ремонту персонального комп’ютера складає 10 % від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали.

$$C_{ком} = C_{вм} = 229{,}45 \text{ грн}, \quad (5.10)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АПК} = \text{В}_{\text{ком}} \cdot \frac{N_{\text{д}}}{N_{\text{експ}} \cdot 365}, \tag{5.11}$$

$$\text{АПК} = 25000 \cdot \frac{67}{3 \cdot 365} = 1\,529{,}68 \text{ грн.}$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння таким же, як у персонального комп'ютера, то амортизаційні відрахування на програмне забезпечення за 3 роки дорівнюють його вартості. Для функціонування персонального комп'ютера використовувалася операційна система Windows 10 Professional, для написання експертної системи оболонка програми. Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Інформація про вартість програмного забезпечення взята з сайту https://soft.rozetka.com.ua.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, *грн*	Амортизаційні відрахування, *грн*
Windows 10 Professional	8000	788,95
MS Office-19	3300	325,44
MS Visual Studio Professional	3700	364,89
	Всього:	1 479,28

Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, оренда, комунальні послуги важко оцінити точно і прийняти рівними 50 % заробітної плати інженера–системотехніка, тобто 3500 *грн*, з якої доля оренди складає 2000 *грн.*, – вартість оренди 10 м², дані взяти з сайту https://dom.ria.com, див. рис. 5.5, де приведено приклад приміщення розміром 30 м², отже вартість одного м² складає 200 *грн*.

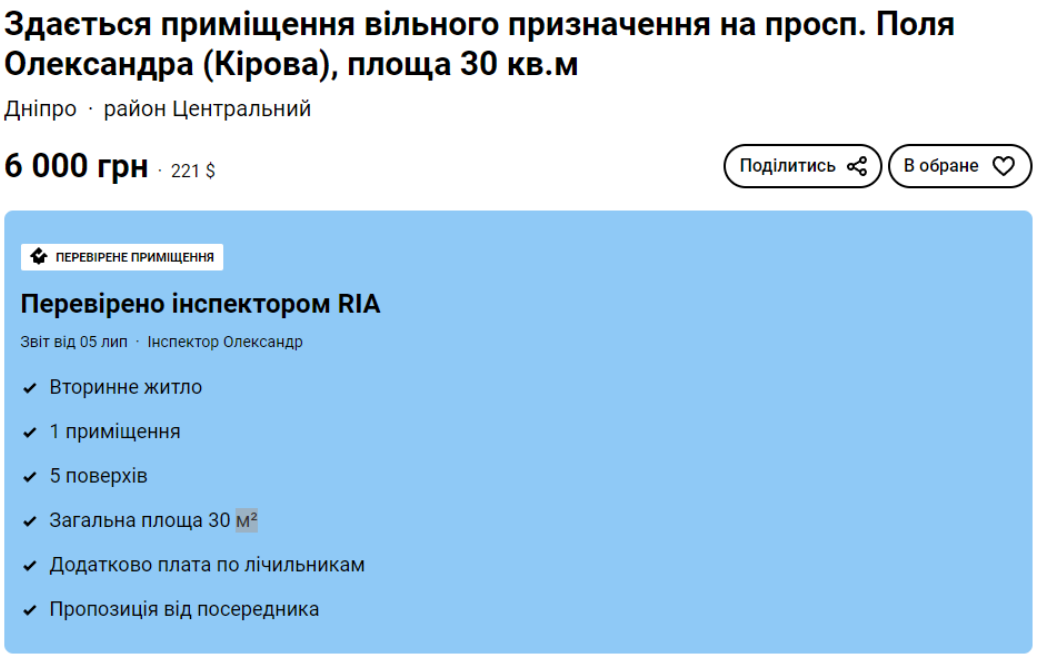


Рисунок 5.4 – Середня зарплата інженер-системотехнік в Дніпрі

Сумарні експлуатаційні витрати на один персональний комп’ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{вм}} + C_{\text{рем}} + C_{\text{ком}} + \text{АПК} + \text{АПО} + C_{\text{дод}}, \quad (5.12)$$

$$C_{\text{експ}} = 495{,}26 + 229{,}45 + 700{,}00 + 229{,}45 + 1\,529{,}68 + 1\,479{,}28 + 3\,500{,}00 == 8\,163{,}12 \text{ грн.}$$

Результати розрахунків зводимо у табл. 5.3.

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}, \quad (5.13)$$

$$C_{\text{розробки}} = 29\,667{,}60 + 6\,526{,}87 + 12\,668{,}06 + 8\,163{,}12 = 57\,025{,}65 \text{ грн.}$$

Розрахунок витрат зводимо у табл. 5.4.

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПО

Найменування витрат	Витрати, *грн*
Витрати на електроенергію	495,26
Вартість витратних матеріалів	229,45
Витрати на ремонт	229,45
Заробітна плата інженера системотехніка	700,00
Амортизація персонального комп'ютера	1 529,68
Амортизація програмного забезпечення	1 479,28
Додаткові витрати	3 500,00
Всього	8 163,12

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, *грн*
Основна заробітна плата	29 667,60
Відрахування на соціальні потреби	6 526,87
Накладні витрати	12 668,06
Експлуатаційні витрати	8 163,12
Всього	57 025,65

За отриманими значеннями техніко–економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення призначеного для проведення дослідження у сфері програмної інженерії. За результатами розрахунків, приблизна вартість розробки складає 57 025,65 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програмного продукту «Методи поетапного моделювання складних процесів» представлені в таблиці 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	При-мітка
1	Вступ	1.09.21	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	2.09.21 – 15.09.21	від 70 джерел
2.1	Дослідження існуючих рішень та інших аспектів дослідження	4.09.21 – 10.09.21	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	16.09.21 – 10.10.21	
4	Постановка задачі, технічне завдання	11.10.21 – 17.10.21	30%
4.1	Розробка постановки задачі	11.10.21 – 13.10.21	
4.2	Розробка технічного завдання	14.10.21 – 16.10.21	
4.3	Затвердження постановки задачі та технічного завдання	17.10.21	
5	Техніко-економічні показники	18.10.21 – 19.10.21	
5.1	Розробка техніко-економічного обґрунтування розробки проекту	18.10.21	
5.2	Затвердження техніко-економічного обґрунтування розробки проекту	19.10.21	
6	Розробка інструментальних засобів дослідження	20.10.21 – 7.11.21	
6.1	Проектування інструментального засобу	20.10.21 – 25.10.21	
6.2	Розробка алгоритмів та програмування інструментального засобу	26.10.21 – 3.11.21	

Продовження таблиці 6.1

6.3	Розробка та програмування інтерфейсу користувача	4.11.21 – 6.11.21	
6.4	Тестування та відлагодження інструментального засобу	7.11.21	
7	Виконання досліджень	8.11.21 – 14.11.21	60%
8	Оформлення тез доповідей	15.11.21 – 18.11.21	
9	Оформлення статті у фаховий журнал	19.11.21 – 22.11.21	
10	Оформлення пояснювальної записки	23.11.21 – 28.11.21	
11	Розробка демонстраційних матеріалів	29.11.21 – 5.12.21	100%

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль здійснюється за допомогою виконання набору тестів з метою виявлення помилок в програмному продукті та його специфікаціях. Контроль за виконанням роботи здійснює головний керівник розробки.

Прийом здійснюється державною комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1 Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ДОДАТОК Б

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР
__.__.__

Методи поетапного моделювання складних процесів

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01208 ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
__.__.__

Керівник розробки
Олександра ГОРБОВА
__.__.__

Виконавець
Микола МУРКОВИЧ
__.__.__

Нормоконтролер
Олена КУРОП'ЯТНИК
__.__.__

2021

ЗАТВЕРДЖЕНО

1116130.01208-01-ЛЗ

Методи поетапного моделювання складних процесів

Специфікація

1116130.01208-01

Аркушів 2

2021

Позначення	Найменування	Примітка
1116130.01208-01-ЛЗ	Лист затвердження	
1116130.01208-01 12 01-ЛЗ	Лист затвердження	
1116130.01208-01 12 01	Текст програми	
1116130.01208-01 13 01-ЛЗ	Лист затвердження	
1116130.01208-01 13 01	Опис програми	
1116130.01208-01 І3 01-ЛЗ	Лист затвердження	
1116130.01208-01 І3 01	Керівництво користувача. Керівництво експерту з побудови технологічних процесів	

ЗАТВЕРДЖЕНО

1116130.01208-01 12 01-ЛЗ

Методи поетапного моделювання складних процесів

Текст програми

1116130.01208-01 12 01

Листів 17

2021

АНОТАЦІЯ

Документ 1116130.01208-01 12 01 «Методи поетапного моделювання складних процесів. Текст програми» є складовою частиною документації на програму, що призначена для розробки програмного додатку «Інструментальний засіб «Методи поетапного моделювання складних процесів», та входить до складу документації на проект.

В документі міститься текст програми. Програма реалізована на мові високого рівня C# у програмному середовищі Visual Studio, а також частина написана на мові логічного програмування Prolog. Об’єм пам’яті що займає програмний комплекс складає 2Мб. Конфігурація комп’ютера стандартна. Програма функціонує в середовищі MS Windows 7.

ЗМІСТ

1 КОРОТКИЙ ОПИС МОДУЛІВ ПРОГРАМИ

Програма складається з восьми модулів. Опишемо призначення кожного модуля програми:

Program.cs – виконавчий модуль запуску програми

DefaultForm.cs – модуль що надає інтерфейс користувача.

Fact.cs – модуль що містить у собі однойменний клас, за допомогою цього класу надається можливість зберігати інформацію про вузли у дереві процесів.

MainControl.cs – головний контролюючий модуль з набором функцій виконавчої частини програмного комплексу, функції для проведення імітаційного моделювання.

R_Fact.cs – модуль що містить у собі однойменний клас, за допомогою цього класу надається можливість зберігати інформацію про випадкові події що можуть трапитись.

Subprocess.cs – модуль що містить у собі однойменний клас, за допомогою цього класу надається можливість зберігати інформацію у вигляді ланцюга підпроцесів.

dpl1.pl – модуль з набором функцій виконавчої частини програмного комплексу, функції для побудови дерева переходів.

rules.pl – модуль з набором правил переходів.

2 ТЕКСТ ПРОГРАМИ

2.1 Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace diplom
{
   static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new DefaultForm());
      }
   }
}
```

2.2 DefaultForm.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace diplom
{
   public partial class DefaultForm : Form
   {
      public DefaultForm()
      {
         InitializeComponent();
      }
      MainControl mainControl = new MainControl();

      private void button1_Click(object sender, EventArgs e)
      {
         if (textBox3.Text != "")
         {
            mainControl.StartProlog();
            mainControl.LoadandCalcResult(Convert.ToInt32(numericUpDown1.Value));
            ShowResults();
            MessageBox.Show("Моделирование завершено успешно", "Сообщение", MessageBoxButtons.OK,
MessageBoxIcon.Information);
         }
         else
         {
            MessageBox.Show("Файл времени не задан!", "Сообщение",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Warning);
         }
      }

      private void button4_Click(object sender, EventArgs e)
      {
         try
         {
            openFileDialog1.Filter = "Текстовые файлы (*.txt)|*.txt";
            openFileDialog1.InitialDirectory = "D:\\TEST\\DPL\\";
```

```
            openFileDialog1.ShowDialog();
            textBox3.Text = openFileDialog1.FileName;
            if (textBox3.Text != "" && textBox3.Text != "openFileDialog1")
            {
                mainControl.LoadTimeFile(openFileDialog1.FileName);
            }
            else { textBox3.Text = ""; }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Выбран не соответствующий требованиям файл","Ошибка",
MessageBoxButtons.OK,MessageBoxIcon.Error);
        }
    }

    private void ShowResults()
    {
        #region BestResults
        listBox1.Items.Clear();
        List<Subprocess> BestP = new List<Subprocess>();
        int time = 0;
        (BestP,time) = mainControl.BestResult();
        foreach (var subp in BestP)
        {
            string item = "Узел: "+subp.getId().ToString() +" Время: "+ subp.getTime().ToString() + " сек.";
            listBox1.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                    listBox1.Items.Add(item);
                }
            }
        }
        listBox1.Items.Add("Общие время составило: "+time+" сек.");
        #endregion

        #region WorstResults
        listBox2.Items.Clear();
        List<Subprocess> WorstP = new List<Subprocess>();
        (WorstP,time) = mainControl.WorstResult();
        foreach (var subp in WorstP)
        {
            string item = "Узел: " + subp.getId().ToString() + " Время: " + subp.getTime().ToString() + " сек.";
            listBox2.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                    listBox2.Items.Add(item);
                }
            }
        }
        listBox2.Items.Add("Общие время составило: " + time + " сек.");
        #endregion

        #region AverageResults
        listBox3.Items.Clear();
        List<Subprocess> AverageP = new List<Subprocess>();
        int avgtime = 0;
        Dictionary<(int, int), int> dicevent = new Dictionary<(int, int), int>();
        (AverageP, avgtime, dicevent) = mainControl.AverageResult();
        time = 0;
        foreach (var subp in AverageP)
        {
            string item = "Узел: " + subp.getId().ToString() + " Среднее время: " + subp.getTime().ToString() + " сек.";
```

```
            listBox3.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() +" Произошло: "+ dicevent[(subp.getId(), ssubp.getId())] + " раз
Среднее время: " + ssubp.getTime().ToString() + " сек.";
                    listBox3.Items.Add(item);
                }
            }
        }
        listBox3.Items.Add("В среднем общие время составило: " + avgtime + " сек.");
        #endregion

        #region NormalResults
        listBox4.Items.Clear();
        List<Subprocess> NormP = new List<Subprocess>();
        (NormP, time) = mainControl.NormalResult();
        foreach (var subp in NormP)
        {
            string item = "Узел: " + subp.getId().ToString() + " Время: " + subp.getTime().ToString() + " сек.";
            listBox4.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                    listBox4.Items.Add(item);
                }
            }
        }
        listBox4.Items.Add("Общие время должно составлять: " + time + " сек.");
        #endregion

        #region Deviations
        listBox5.Items.Clear();
        List<Subprocess> DevN = new List<Subprocess>();
        DevN = mainControl.DeviationsNodes(BestP,NormP);
        foreach (var subp in DevN)
        {
            string item = "Узел: " + subp.getId().ToString() + " Время: " + subp.getTime().ToString() + " сек.";
            listBox5.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                    listBox5.Items.Add(item);
                }
            }
        }

        listBox6.Items.Clear();
        DevN = mainControl.DeviationsNodes(WorstP, NormP);
        foreach (var subp in DevN)
        {
            string item = "Узел: " + subp.getId().ToString() + " Время: " + subp.getTime().ToString() + " сек.";
            listBox6.Items.Add(item);
            if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
            {
                foreach (var ssubp in subp.getSubprocess())
                {
                    item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                    listBox6.Items.Add(item);
                }
            }
        }
```

```
            listBox7.Items.Clear();
            DevN = mainControl.DeviationsNodes(AverageP, NormP);
            foreach (var subp in DevN)
            {
                string item = "Узел: " + subp.getId().ToString() + " Время: " + subp.getTime().ToString() + " сек.";
                listBox7.Items.Add(item);
                if (subp.getSubprocess() != null && subp.getSubprocess().Count != 0)
                {
                    foreach (var ssubp in subp.getSubprocess())
                    {
                        item = "   Событие: " + ssubp.getId().ToString() + " Время: " + ssubp.getTime().ToString() + " сек.";
                        listBox7.Items.Add(item);
                    }
                }
            }
            #endregion
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (label10.Visible == false)
            {
                label10.Visible = true;
                label11.Visible = true;
                label12.Visible = true;
                label13.Visible = true;
                label14.Visible = true;
            }
            else {
                label10.Visible = false;
                label11.Visible = false;
                label12.Visible = false;
                label13.Visible = false;
                label14.Visible = false;
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            MessageBox.Show("«Інструментальний засіб \n«Поетапне моделювання складних процесів»\nВерсія 1.0\nДНУЗТ,
2021 рік, кафедра КТС\nРозробник: Муркович М.С.","О программе", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                saveFileDialog1.Filter = "Текстовые файлы (*.txt)|*.txt";
                saveFileDialog1.InitialDirectory = "D:\\TEST\\DPL\\";
                saveFileDialog1.ShowDialog();
                using (System.IO.StreamWriter sw = new System.IO.StreamWriter(saveFileDialog1.FileName))
                {
                    sw.WriteLine(label1.Text);
                    for (int i = 0; i < listBox1.Items.Count; i++)
                        sw.WriteLine(listBox1.Items[i].ToString());
                    sw.WriteLine(label7.Text);
                    for (int i = 0; i < listBox5.Items.Count; i++)
                        sw.WriteLine(listBox5.Items[i].ToString());
                    sw.WriteLine(label2.Text);
                    for (int i = 0; i < listBox2.Items.Count; i++)
                        sw.WriteLine(listBox2.Items[i].ToString());
                    sw.WriteLine(label8.Text);
                    for (int i = 0; i < listBox6.Items.Count; i++)
                        sw.WriteLine(listBox6.Items[i].ToString());
                    sw.WriteLine(label5.Text);
                    for (int i = 0; i < listBox3.Items.Count; i++)
                        sw.WriteLine(listBox3.Items[i].ToString());
                    sw.WriteLine(label9.Text);
```

```
                for (int i = 0; i < listBox7.Items.Count; i++)
                    sw.WriteLine(listBox7.Items[i].ToString());
                sw.WriteLine(label6.Text);
                for (int i = 0; i < listBox4.Items.Count; i++)
                    sw.WriteLine(listBox4.Items[i].ToString());
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
  }
}
```

2.3 Fact.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace diplom
{
   public class Fact
   {
      int id;
      int from;
      int to;
      int[] r_event;

      public int  getId()
      {
         return id;
      }
      public int getFrom()
      {
         return from;
      }
      public int getTo()
      {
         return to;
      }
      public int[] getR_event()
      {
         return r_event;
      }
      public Fact(int v1, int v2, int v3, int[] ev)
      {
         id = v1;
         from = v2;
         to = v3;
         r_event = ev;
      }
   }
}
```

2.4 MainControl.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace diplom
{
```

```
class MainControl
{
    public List<Fact> facts = new List<Fact>();
    public List<R_Fact> rfacts = new List<R_Fact>();
    public List<List<Subprocess>> allprocess;
    public int[] sequence;

    public void StartProlog()
    {
        Process _process = new Process();
        ProcessStartInfo startInfo = new ProcessStartInfo();

        startInfo.FileName = "D:\\TEST\\DPL\\dpl1.pl";

        _process.StartInfo = startInfo;
        _process.Start();
        _process.WaitForExit();
    }

    public void LoadTimeFile(string path)
    {
        try
        {
            string f = "";
            using (StreamReader sr = new StreamReader(path))
            {
                f = sr.ReadToEnd();
            }
            string[] lines = f.Split('\n');
            foreach (string s in lines)
            {
                string[] parts = s.Split(',');
                if (parts[0].Contains('-'))
                {
                    Fact fact = new Fact(Convert.ToInt32(parts[0]), Convert.ToInt32(parts[1]), Convert.ToInt32(parts[2].Split(':').First()),
null);
                    R_Fact rfact = new R_Fact(fact, Convert.ToInt32(parts[2].Split(':').Last().Split('.').First()));
                    rfacts.Add(rfact);
                }
                else
                {
                    if (parts.Length > 3)
                    {
                        string[] arr = parts[3].Split(';');
                        int[] iarr = new int[arr.Length];
                        int i = 0;
                        foreach (string a in arr)
                        {
                            iarr[i] = Convert.ToInt32(a);
                            i++;
                        }
                        Fact fact = new Fact(Convert.ToInt32(parts[0]), Convert.ToInt32(parts[1]), Convert.ToInt32(parts[2]), iarr);
                        facts.Add(fact);
                    }
                    else
                    {
                        Fact fact = new Fact(Convert.ToInt32(parts[0]), Convert.ToInt32(parts[1]), Convert.ToInt32(parts[2]), null);
                        facts.Add(fact);
                    }

                }
            }
        }
        catch (Exception e)
        {
            throw e;
        }
    }
```

```
public void LoadandCalcResult(int count)
{
   List<Subprocess> process;
   allprocess = new List<List<Subprocess>>();
   string f = "";
   using (StreamReader sr = new StreamReader("D:\\TEST\\DPL\\out.txt"))
   {
      f = sr.ReadToEnd();
   }
   string[] lines = f.Split('\n');
   sequence = new int[lines.Length-1];
   int i = 0;
   foreach (string s in lines)
   {
      try
      {
         sequence[i] = Convert.ToInt32(s);
         i++;
      }
      catch{}
   }
   i = 0;
   Random r = new Random();
   while (i < count)
   {
      process = new List<Subprocess>();
      foreach (int j in sequence)
      {
         var found = facts.Find(p => p.getId() == j);
         if (found != null)
         {
            int time = r.Next(found.getFrom(), found.getTo());

            if (found.getR_event() != null)
            {
               int[] r_e = found.getR_event();
               int[] rep = new int[r_e.Length];
               R_Fact r_f = null;
               for (int a = 0; a < r_e.Length; a++)
               {
                  r_f = rfacts.Find(p => p.getFact().getId() == r_e[a]);
                  if (a == 0)
                     rep[a] = r_f.getProbability();
                  else rep[a] = rep[a - 1] + r_f.getProbability();
               }
               int prob = r.Next(0, 100);
               List<Subprocess> rsub = new List<Subprocess>();
               for (int a = 0; a < r_e.Length; a++)
               {
                  if (a == 0)
                  {
                     if (rep[a] >= prob)
                     {
                        r_f = rfacts.Find(p => p.getFact().getId() == r_e[a]);
                        rsub.Add(new Subprocess(r_e[a], r.Next(r_f.getFact().getFrom(), r_f.getFact().getTo()), null));
                     }
                  }
                  else
                        if (rep[a] >= prob && rep[a - 1] < prob)
                  {
                     r_f = rfacts.Find(p => p.getFact().getId() == r_e[a]);
                     rsub.Add(new Subprocess(r_e[a], r.Next(r_f.getFact().getFrom(), r_f.getFact().getTo()), null));
                  }
               }
               process.Add(new Subprocess(found.getId(), time, rsub));
            }
            else
```

```
            {
                process.Add(new Subprocess(found.getId(), time, null));
            }
        }
    }
    i++;
    allprocess.Add(process);
  }
}

public (List<Subprocess>,int) BestResult()
{
  List<Subprocess> BestP = new List<Subprocess>();
  int mintime = 0x7FFFFFFF;
  foreach (var list in allprocess)
  {
    int time = 0;
    foreach (var pr in list)
    {
      time += pr.getTime();
      if (pr.getSubprocess() != null && pr.getSubprocess().Count != 0)
        foreach (var ssubp in pr.getSubprocess())
          time += ssubp.getTime();
    }
    if (time < mintime)
    {
      mintime = time;
      BestP = list;
    }
  }
  return (BestP,mintime);
}

public (List<Subprocess>, int) WorstResult()
{
  List<Subprocess> WorstP = new List<Subprocess>();
  int maxtime = 0;
  foreach (var list in allprocess)
  {
    int time = 0;
    foreach (var pr in list)
    {
      time += pr.getTime();
      if (pr.getSubprocess() != null && pr.getSubprocess().Count != 0)
        foreach (var ssubp in pr.getSubprocess())
          time += ssubp.getTime();
    }
    if (time > maxtime)
    {
      maxtime = time;
      WorstP = list;
    }
  }
  return (WorstP,maxtime);
}

public (List<Subprocess>, int, Dictionary<(int, int), int>) AverageResult()
{
  List<Subprocess> AverageP = new List<Subprocess>();
  int avgtime = 0;
  Dictionary<(int, int), int> dicevent = new Dictionary<(int, int), int>();
  bool flag = false;
  foreach (var list in allprocess)
  {
    int time = 0;
    foreach (var pr in list)
    {
      time += pr.getTime();
```

```
            if (AverageP.Find(p => p.getId() == pr.getId())!=null)
            {
               AverageP.Find(p => p.getId() == pr.getId()).setTime((AverageP.Find(p => p.getId() ==
pr.getId()).getTime()+pr.getTime())/2);
               if (pr.getSubprocess() != null && pr.getSubprocess().Count != 0)
                  foreach (var ssubp in pr.getSubprocess())
                  {
                     time += ssubp.getTime();
                     if (AverageP.Find(p => p.getId() == pr.getId()).getSubprocess().Find(p => p.getId() == ssubp.getId()) != null)
                     {
                        AverageP.Find(p => p.getId() == pr.getId()).getSubprocess().Find(p => p.getId() == ssubp.getId()).setTime(
                           (AverageP.Find(p => p.getId() == pr.getId()).getSubprocess().Find(p => p.getId() ==
ssubp.getId()).getTime() + ssubp.getTime()) / 2);
                        dicevent[(pr.getId(), ssubp.getId())]++;
                     }
                     else
                     {
                        AverageP.Find(p => p.getId() == pr.getId()).getSubprocess().Add(ssubp);
                        dicevent.Add((pr.getId(), ssubp.getId()),1);
                     }
                  }
            }
            else
            {
               AverageP.Add(pr);
               if (pr.getSubprocess() != null && pr.getSubprocess().Count != 0)
                  foreach (var ssubp in pr.getSubprocess())
                  {
                     time += ssubp.getTime();
                     dicevent.Add((pr.getId(), ssubp.getId()), 1);
                  }
            }
         }
         if (flag)
            avgtime = (avgtime + time) / 2;
         else
         {
            avgtime = time;
            flag = true;
         }
      }
      return (AverageP, avgtime, dicevent);
   }

   public (List<Subprocess>, int) NormalResult()
   {
      List<Subprocess> NormP = new List<Subprocess>();
      int alltime = 0;
      foreach (int j in sequence)
      {
         var found = facts.Find(p => p.getId() == j);
         if (found != null)
         {
            int time = (found.getFrom()+found.getTo())/2;
            alltime += time;
            NormP.Add(new Subprocess(found.getId(), time, null));
         }
      }
      return (NormP, alltime);
   }

   public List<Subprocess> DeviationsNodes(List<Subprocess> Input, List<Subprocess> Normal)
   {
      List<Subprocess> DevN = new List<Subprocess>();
      foreach (Subprocess sub in Input)
      {
         var found = Normal.Find(p => p.getId() == sub.getId());
         if (found != null)
```

```
            {
                int time = sub.getTime();
                if (sub.getSubprocess() != null && sub.getSubprocess().Count != 0)
                    foreach (var s in sub.getSubprocess())
                        time += s.getTime();
                if (time > found.getTime())
                    DevN.Add(sub);
            }
        }
        return DevN;
    }
  }
}
```

2.5 R_Fact.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace diplom
{
  public class R_Fact
  {
    Fact fact;
    int probability;

    public Fact getFact()
    {
      return fact;
    }

    public int getProbability()
    {
      return probability;
    }

    public R_Fact(Fact fact, int v)
    {
      this.fact = fact;
      this.probability = v;
    }
  }
}
```

2.6 Subprocess.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace diplom
{
  public class Subprocess
  {
    int id;
    int time;
    List<Subprocess> subprocess;

    public int getId()
    {
      return id;
    }

    public int getTime()
    {
```

```
        return time;
    }

    public void setTime(int Time)
    {
        this.time = Time;
    }

    public void setSubprocess(List<Subprocess> subprocess)
    {
        this.subprocess = subprocess;
    }

    public List<Subprocess> getSubprocess()
    {
        return subprocess;
    }

    public Subprocess(int id, int time, List<Subprocess> s)
    {
        this.id = id;
        this.time = time;
        subprocess = s;
    }

  }
}
```

2.7 dpl1.pl

```
:- dynamic analise/4, fact/3, ans/2, data/1, svar/1.
:- multifile fact/3, ans/2.

readchar(Ch,Elem):- ans(Elem,Ch).

%считывание динамических данных из файлалов,
 readFile:-consult('rules.pl'),consult('answers.pl').

analise(E1,E1,y) :- !.
analise(E1,E2,n) :- E1 =:= -E2,!.
analise(E1,E1,n) :- !,fail.
analise(E1,E2,y) :- E1 =:= -E2,!,fail.

putfact(Elem,y) :- assertz(data(Elem)).
putfact(Elem,n) :- E1 is -Elem,assertz(data(E1)).

analize(y,y,_,_,_).
analize(n,n,_,_,_).
analize(' ',Ans,Mess,Msg,Elem) :- getreaction(Ans,Mess,Msg,Elem).

getreaction(Ans,Mess,Msg,Elem) :- writef('%t?',[Elem]),readchar(Ans1,Elem),writef(' %t',[Ans1]),nl,
      analize(Ans1,Ans,Mess,Msg,Elem).

abss(A,A):-A>0,!.
abss(A,B):-B is -A.

proveel(_,y,_) :-!.
proveel(_,n,_) :- !,fail.
proveel(E1,' ',Msg) :- abss(E1,Elem),fact(Elem,Mess,[]),
    getreaction(Ans,Mess,Msg,Elem),putfact(Elem,Ans),!,
    analise(E1,Elem,Ans).
proveel(E1,' ',Msg) :- fact(E1,Svar,List),pr(List,Msg),
     assertz(data(E1)),asserta(svar(Svar)).

getinfo(E1,y) :- data(E1),!.
getinfo(E1,n) :- Elem is -E1,data(Elem),!.
getinfo(_,' ').

pr([],_).
```

```
pr([E1|List],Mess) :- getinfo(E1,Inf),!,proveel(E1,Inf,Mess),!, pr(List,Mess).

getfact(E1) :- E1>0,fact(E1,_,_).
getfact(E1) :- E1<0,E11 is -E1,fact(E11,_,_).

recogn(y) :- retract(data(E1)), getfact(E1), write(E1),nl,fail.
recogn(_) :- retractall(data(_)).

provefact :-
        fact(0,Mess,List),pr(List,Mess),
        writef('%t',[Mess]),nl, tell('out.txt'),recogn(y), told.


expert:-
       readFile, provefact,halt(0);
       svar(X),writef('Возможно %t',[X]),!;
       writeln('Не могу установить ...').

%%цель
:-expert.
```

2.8 rules.pl

```
%% база фактів
fact(1,'приближаемся к перекрестку',[2]).
fact(2,'видем впереди перекресток',[]).
fact(3,'продолжаем движение',[1,4]).
fact(4,'находимся в левой полосе, на перекрестке необходимо повернуть налево',[]).
fact(5,'включаем левый указатель поворота для перестроения',[1,6]).
fact(6,'находимся в правой полосе, на перекрестке необходимо повернуть на лево',[]).
fact(7,'включаем правый указатель поворота для перестроения',[1,8]).
fact(8,'находимся в левой полосе, на перекрестке необходимо проехать прямо',[]).
fact(9,'продолжаем движение',[1,10]).
fact(10,'находимся в правой полосе, на перекрестке необходимо проехать прямо',[]).
fact(11,'подезжаем к перекрестку в левой полосе',[3,12]).
fact(12,'приблизились к перекрестку',[]).
fact(13,'выполняем перестроение в левую полосу',[5,14]).
fact(14,'помеха слева отсутсвует',[]).
fact(15,'пропускаем попутный транспорт',[5,16]).
fact(16,'помеха слева присутсвует',[]).
fact(17,'пропускаем попутный транспорт',[7,18]).
fact(18,'помеха справо присутсвует',[]).
fact(19,'выполняем перестроение в правую полосу',[7,20]).
fact(20,'помеха справо отсутсвует',[]).
fact(21,'подезжаем к перекрестку в правой полосе',[9,22]).
fact(22,'приблизились к перекрестку',[]).
fact(23,'ожидаем разрешающего сигнала светофора для поворота на лево',[11,24]).
fact(24,'горит запрещающий сигнал светофора, дополнительная секция не горит',[]).
fact(25,'пропускаем встречное движение',[11,26]).
fact(26,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(27,'горит разрешающий сигнал светофора, дополнительная секция горит',[]).
fact(28,'подезжаем к перекрестку в левой полосе',[13,29]).
fact(29,'приблизились к перекрестку',[]).
fact(30,'выполняем перестроение в левую полосу',[15,31]).
fact(31,'помеха слева отсутсвует',[]).
fact(32,'выполняем перестроение в правую полосу',[17,33]).
fact(33,'помеха справо отсутсвует',[]).
fact(34,'подезжаем к перекрестку в правой полосе',[19,35]).
fact(35,'приблизились к перекрестку',[]).
fact(36,'ожидаем разрешающего сигнала светофора для проезда по прямой',[21,37]).
fact(37,'горит запрещающий сигнал светофора',[]).
fact(38,'горит разрешающий сигнал светофора',[]).
fact(39,'пропускаем встречное движение',[23,40]).
fact(40,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(41,'горит разрешающий сигнал светофора',[]).
fact(42,'встречный движение отсутствует',[]).
fact(43,'ожидаем разрешающего сигнала светофора для поворота на лево',[28,44]).
fact(44,'горит запрещающий сигнал светофора, дополнительная секция не горит',[]).
fact(45,'пропускаем встречное движение',[28,46]).
```

```
fact(46,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(47,'горит разрешающий сигнал светофора, дополнительная секция горит',[]).
fact(48,'подезжаем к перекрестку в левой полосе',[30,49]).
fact(49,'приблизились к перекрестку',[]).
fact(50,'подезжаем к перекрестку в правой полосе',[32,51]).
fact(51,'приблизились к перекрестку',[]).
fact(52,'ожидаем разрешающего сигнала светофора для проезда по прямой',[34,53]).
fact(53,'горит запрещающий сигнал светофора',[]).
fact(54,'горит разрешающий сигнал светофора',[]).
fact(55,'горит разрешающий сигнал светофора',[]).
fact(56,'встречный движение отсутствует',[]).
fact(57,'пропускаем встречное движение',[43,58]).
fact(58,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(59,'горит разрешающий сигнал светофора',[]).
fact(60,'встречный движение отсутствует',[]).
fact(61,'ожидаем разрешающего сигнала светофора для поворота на лево',[48,62]).
fact(62,'горит запрещающий сигнал светофора, дополнительная секция не горит',[]).
fact(63,'пропускаем встречное движение',[48,64]).
fact(64,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(65,'горит разрешающий сигнал светофора, дополнительная секция горит',[]).
fact(66,'ожидаем разрешающего сигнала светофора для проезда по прямой',[50,67]).
fact(67,'горит запрещающий сигнал светофора',[]).
fact(68,'горит разрешающий сигнал светофора',[]).
fact(69,'горит разрешающий сигнал светофора',[]).
fact(70,'встречный движение отсутствует',[]).
fact(71,'пропускаем встречное движение',[61,72]).
fact(72,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(73,'горит разрешающий сигнал светофора',[]).
fact(74,'встречный движение отсутствует',[]).
fact(75,'горит разрешающий сигнал светофора',[]).
fact(76,'встречный движение отсутствует',[]).
fact(0,'выполняем поворот',[11,27]).
fact(0,'проезжаем по прямой',[21,38]).
fact(0,'выполняем поворот',[23,41]).
fact(0,'выполняем поворот',[25,42]).
fact(0,'выполняем поворот',[28,47]).
fact(0,'проезжаем по прямой',[34,54]).
fact(0,'проезжаем по прямой',[36,55]).
fact(0,'выполняем поворот',[39,56]).
fact(0,'выполняем поворот',[43,59]).
fact(0,'выполняем поворот',[45,60]).
fact(0,'выполняем поворот',[48,65]).
fact(0,'проезжаем по прямой',[50,68]).
fact(0,'проезжаем по прямой',[52,69]).
fact(0,'выполняем поворот',[57,70]).
fact(0,'выполняем поворот',[61,73]).
fact(0,'выполняем поворот',[63,74]).
fact(0,'проезжаем по прямой',[66,75]).
fact(0,'выполняем поворот',[71,76]).

:- dynamic fact/3.
:- multifile fact/3.
```

ЗАТВЕРДЖЕНИЙ

1116130.01208-01 13 01-ЛЗ

Методи поетапного моделювання складних процесів

Опис програми

1116130.01208-01 13 01

Листів 21

2021

АНОТАЦІЯ

Документ 1116130.1208-01 13 01 «Методи поетапного моделювання складних процесів. Опис програми» є складовою частиною документації на програму, що призначена для розробки програмного додатку «Інструментальний засіб «Методи поетапного моделювання складних процесів», та входить до складу документації на проект.

В документі наведені призначення програми, умови її використання, опис, формат вхідних та вихідних даних. Програма реалізована на мові високого рівня C# у програмному середовищі Visual Studio. Об’єм пам’яті що займає програмний комплекс складає 2Мб. Конфігурація комп’ютера стандартна. Програма функціонує в середовищі MS Windows 7.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Розроблена програма має назву «Інструментальний засіб «Методи поетапного моделювання складних процесів».

Програма розрахована на функціонування в операційних системах MS Windows 7/8/8.1/10.

Програма реалізована на мові C# у програмному середовищі Visual Studio та на мові програмування Prolog. Об’єм пам’яті що займає програмний комплекс складає 2Мб. Конфігурація комп’ютера стандартна.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Основним функціональним призначенням програмного продукту є надання можливостей дослідження часової ефективності складних процесів або їх частин за допомогою побудови дерева рішень та подальшого імітаційного моделювання з урахуванням виникнення раптових подій. Програмний комплекс дозволить провести дослідження доцільності використання даної реалізації методу поетапного моделювання в сфері дослідження та оптимізації складних технічних процесів та бізнес процесів, а також наддасть можливість аналізу та пошуку слабких місць у дереві процесів, що полегшить роботу відповідних спеціалістів з відстежування та контролю якості виконання процесів у компаніях різної направленості.

Даний програмний продукт не слід використовувати у сферах не пов’язаних з ієрархічною структурою виконання процесів, а також у тих сферах навчання де не можливо використати ієрархічну структуру виконання процесів, або при відсутності кваліфікованого персоналу з контролю якості виконання процесів.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритми роботи програми

Структурна схема взаємодії складових програми наведена на рис. 3.1.

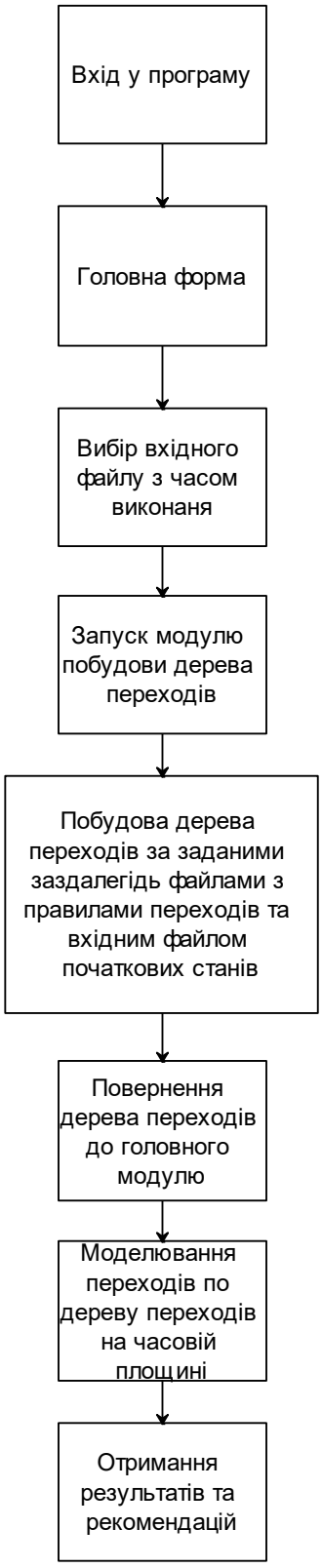


Рисунок 3.1 – Структурна схема взаємодії складових програми

В програмі передбачено використання наборів фактів для мови логічного програмування Prolog, приклад схема переходів між фактами наведено на зображені на рис. 3.2 – 3.8.

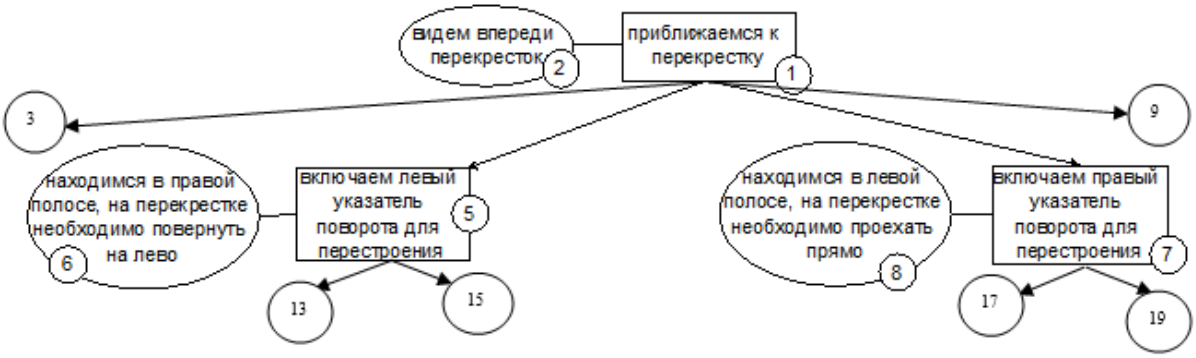


Рисунок 3.2 – Схема переходів частина 1

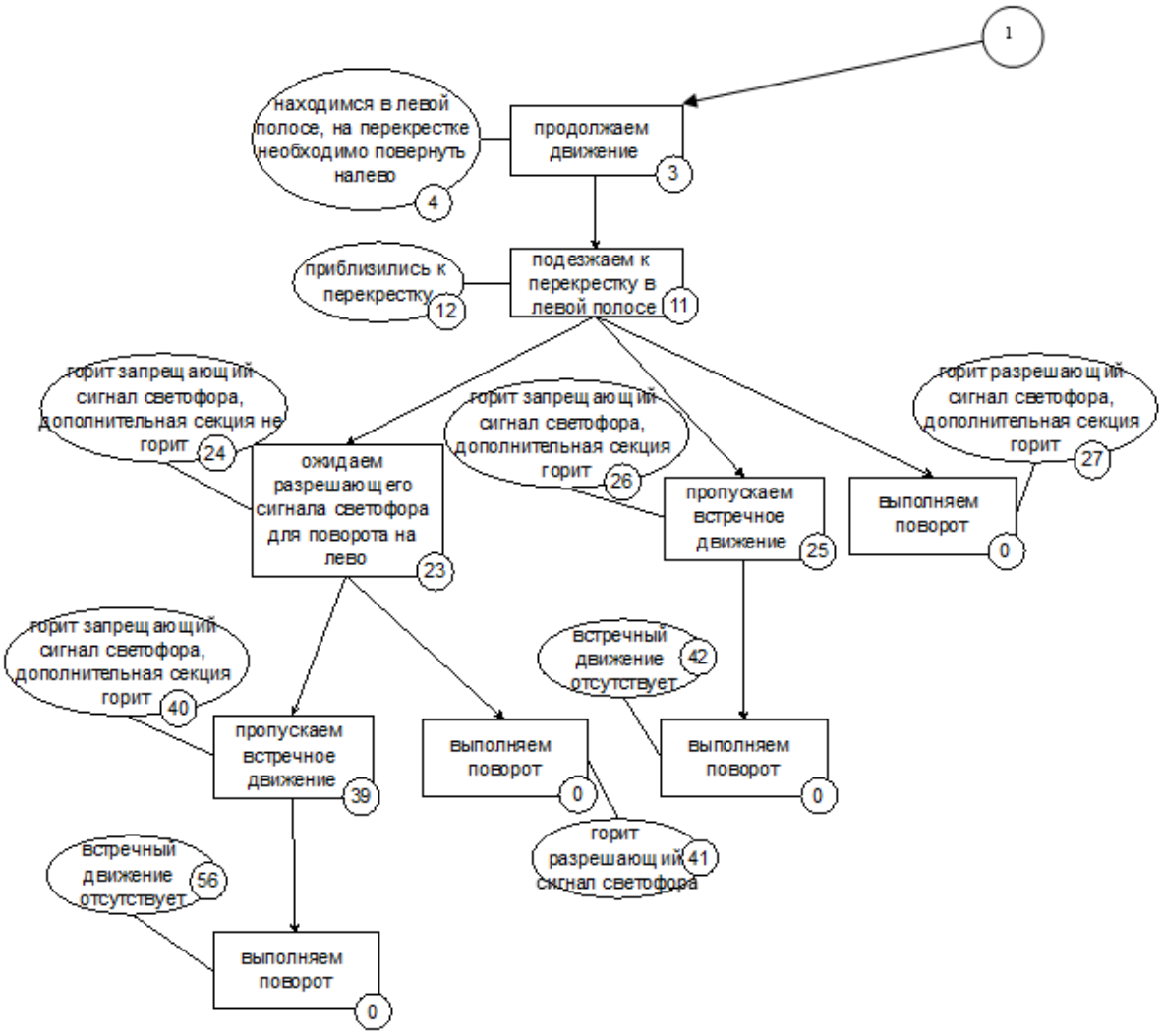


Рисунок 3.3 – Схема переходів частина 2

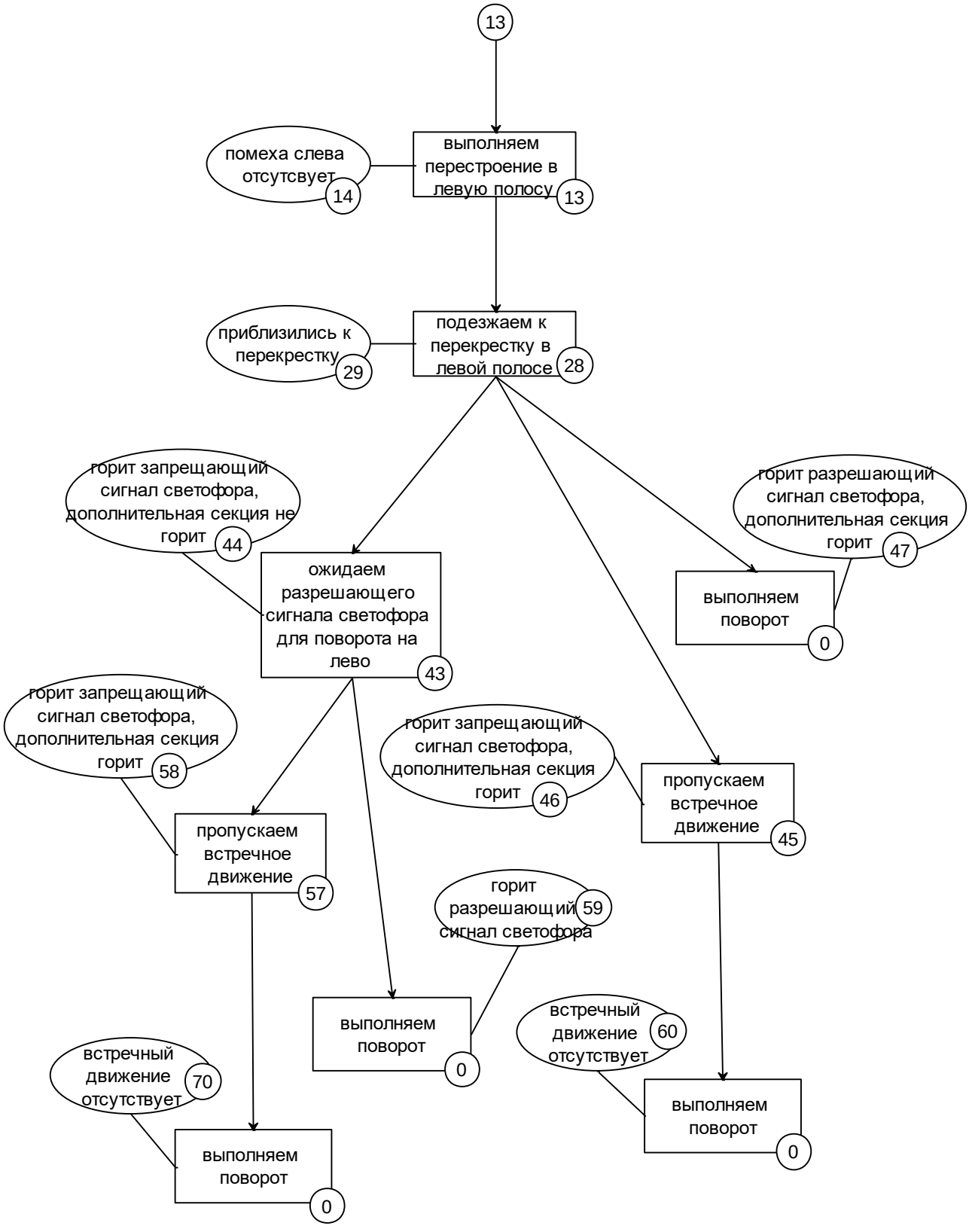


Рисунок 3.4 – Схема переходів частина 3

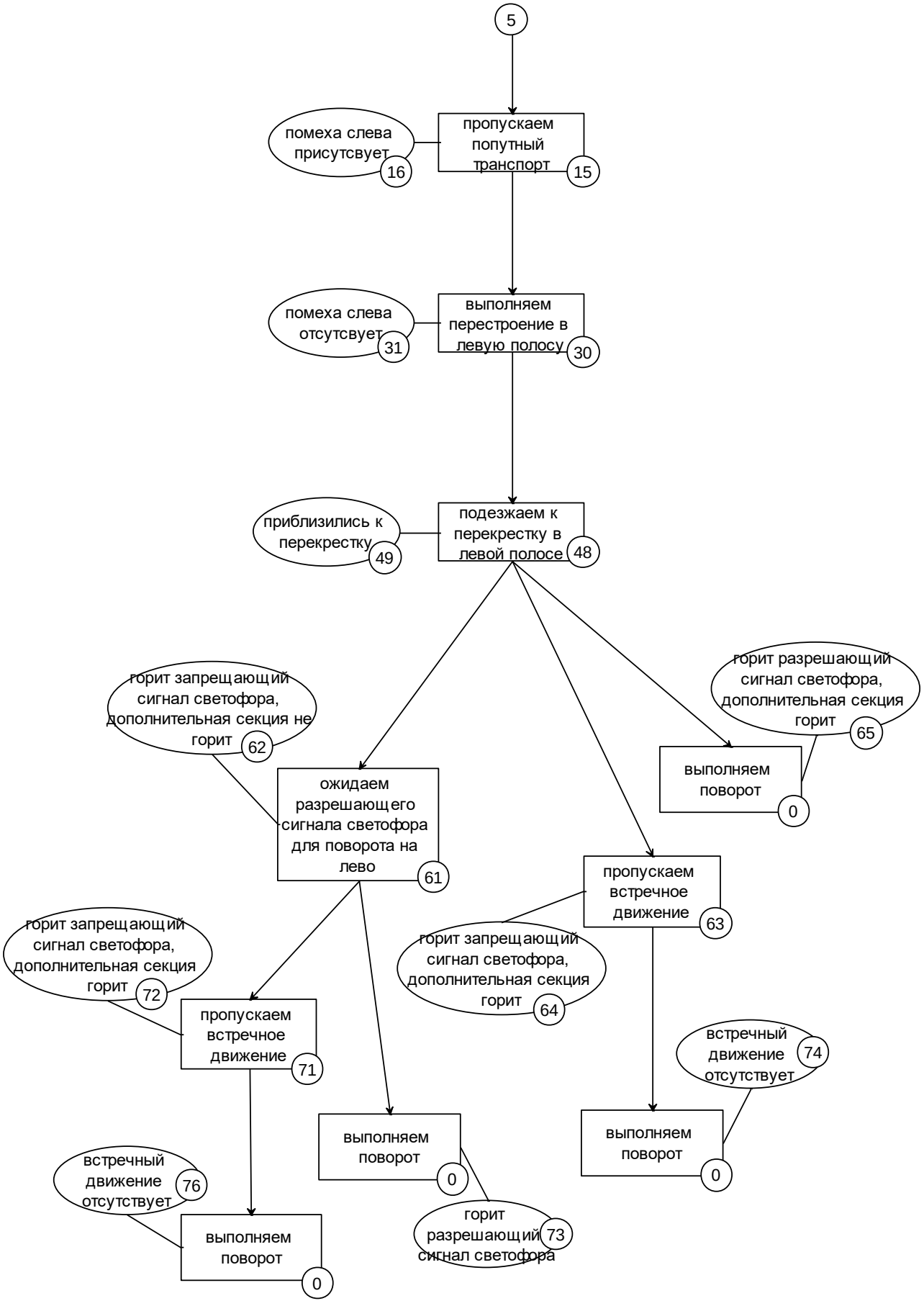


Рисунок 3.5 – Схема переходів частина 4

Рисунок 3.6 – Схема переходів частина 5

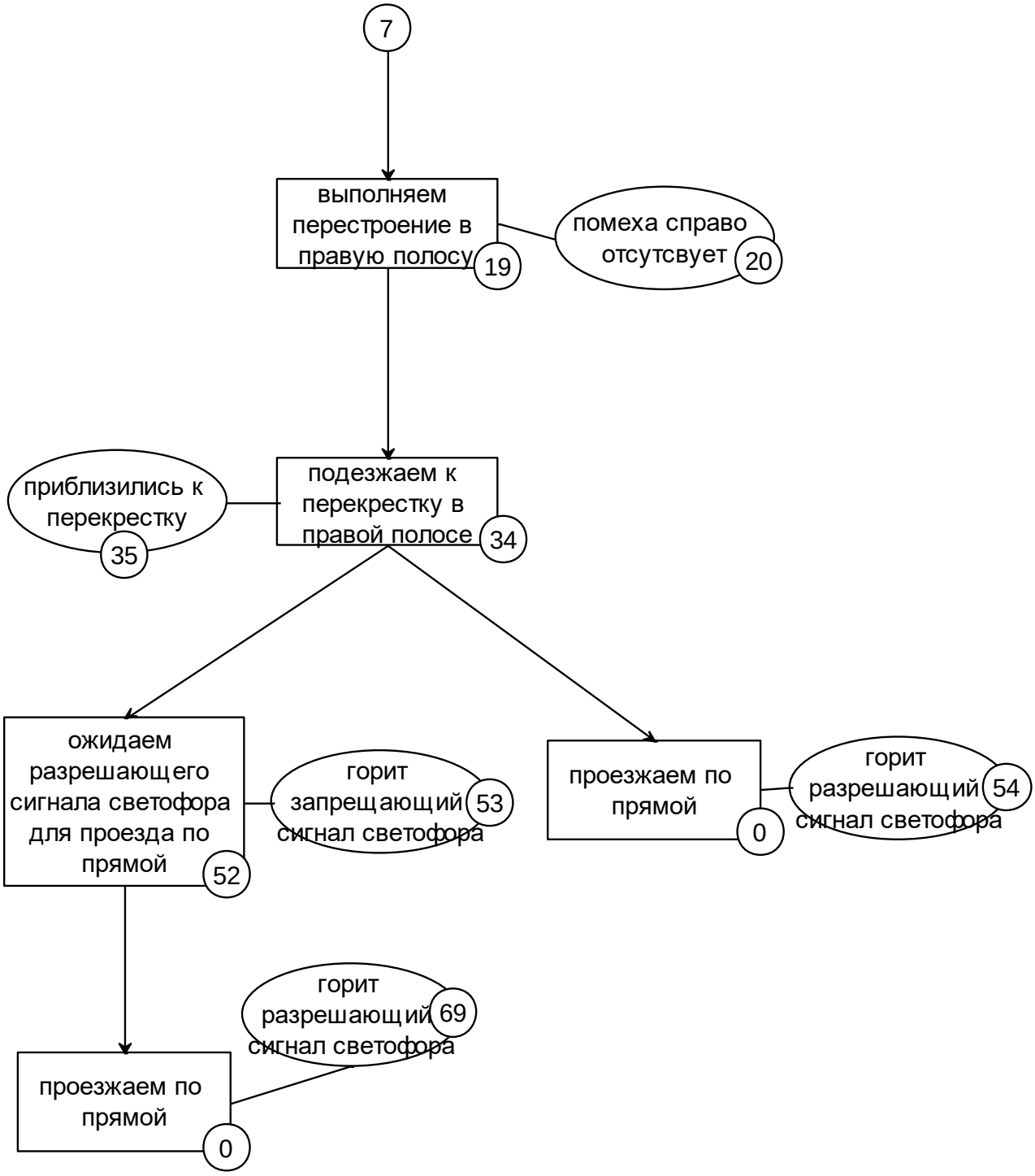


Рисунок 3.7 – Схема переходів частина 6

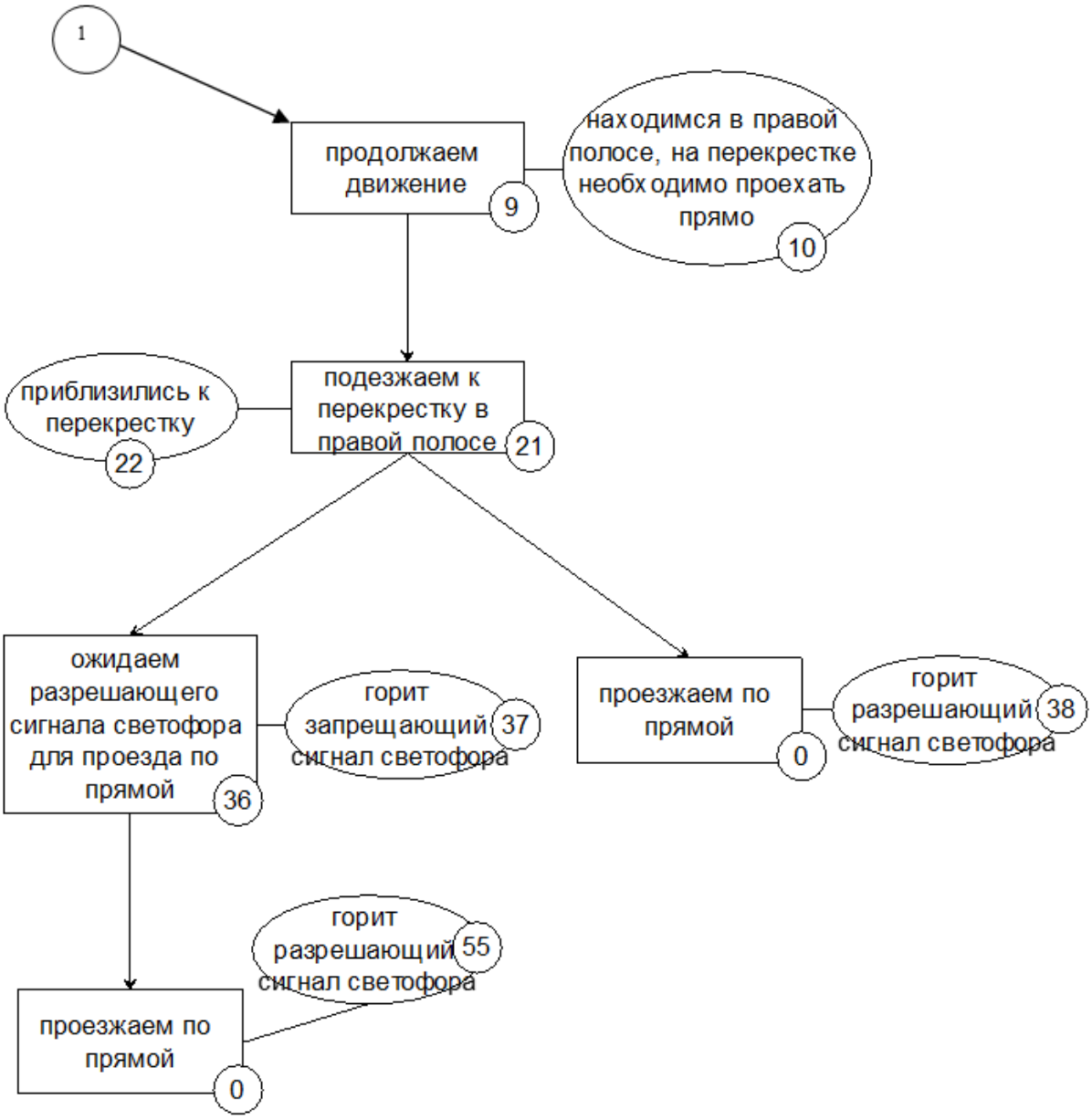


Рисунок 3.8 – Схема переходів частина 7

3.2 Використані методи

В ході розробки програми були використані наступні методи та методології:

1 Програмний засіб працює з використання методології поетапного моделювання, з наступними етапами: збір інформації о початкових станах, побудова дерева переходів та імітаційне моделювання.

2 Програмний продукт розроблено згідно моделі програмування «MVC», в програмі присутня головна форма для введення та виведення

інформації, моделі представлення інформації та головний контролер для виконання головної логіки інструменту.

3 Для роботи з фактами та побудови дерева переходів використовується мова логічного програмування Prolog.

4 Для локального (тимчасового) зберігання підсумкової інформації використовується ієрархічна система класів, кожен клас містить в собі відповідну інформацію.

3.3 Структура програми з описом функції

Програма складається з чотирьох основних модулів. Призначення модулів програми:

Program.cs – виконавчий модуль запуску програми

DefaultForm.cs – модуль з набором функцій для вводу необхідної інформації та виводу результатів моделювання.

Fact.cs – клас являє собою модель для представлення та тимчасового зберігання інформації о вузлах та подія що можуть відбутися під час переходів з цього вузлу та час котрий може бути затрачений для цього.

R_Fact.cs – клас являє собою модель для представлення та тимчасового зберігання інформації о випадкових подія що можуть відбутися під час переходів з вузлів та час котрий може бути затрачений для цього.

Subprocess.cs – клас являє собою модель для представлення та тимчасового зберігання підсумкової інформації о вузлах, випадкових подіях що відбулися під час переходів з вузлів та час котрий був затрачений для цього.

MainControl.cs – головний модуль що містить в собі головні функції, функції підготовки інформації для виводу, а також функції з імітаційного моделювання.

3.4 Зв’язки програми з іншими програмами

Для роботи з даним ПО на робочій станції повинно бути встановлено та налаштовано мову логічного програмування Prolog.

4 ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ

Мінімальна конфігурація, що забезпечить нормальну експлуатацію програмного продукту:

– процесор: Intel Core 2 Duo 2.4 GHz або краще;

– оперативна Пам'ять: 2 GB DDR2 або краще;

– вільний дисковий простір 50 Mb;

– наявність CD/DVD приводу або USB роз’ємну для встановлення необхідного ПЗ;

– монітор з роздільною здатністю екрану 1024x768;

– клавіатура;

– маніпулятор «миша».

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Для початку роботи програми на сервері необхідно встановити, налаштувати та заповнити фали з використанням Prolog. Для початку роботи програми на робочій станції необхідно виконати файл «diplom\bin\diplom.exe» Якщо програма знаходиться на зовнішньому носії даних (CD, DVD, Flash Memory), то її можна завантажити з них або скопіювавши на жорсткий диск.

Об’єм програми складає 2 Мбайт.

6 ВХІДНІ ДАНІ

Метод організації вхідних даних полягає в наданні користувачу зручного інтерфейсу для введення даних у поля зі строгим дотриманням встановленого формату. Файли з наборами фактів що подаються на вхід мають мати формат файлу – «.pl». Файл з інформацією о часі проходження кожного вузлу, що подаються на вхід, мають мати формат файлу – «.txt».

Вхідні дані можна розділити на наступні підгрупи:

– цифрова інформація, кількість повторень моделювання, вводяться за допомогою користувацького інтерфейсу;

– посилання на файли, вводяться за допомогою користувацького інтерфейсу.

7 ВИХІДНІ ДАНІ

До вихідних даних відносяться:

– файл з побудованим деревом рішення;

– результати імітаційного моделювання, а саме статистична інформація о найкращому, найгіршому та середньому результаті проходженні кожного вузлу та усього дерева в цілому, а також інформацію про найслабкіші місця.

8 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

Після запуску програми (diplom\bin\diplom.exe) на екрані з'являється вікно головної форми, представлене на рис. 8.1.

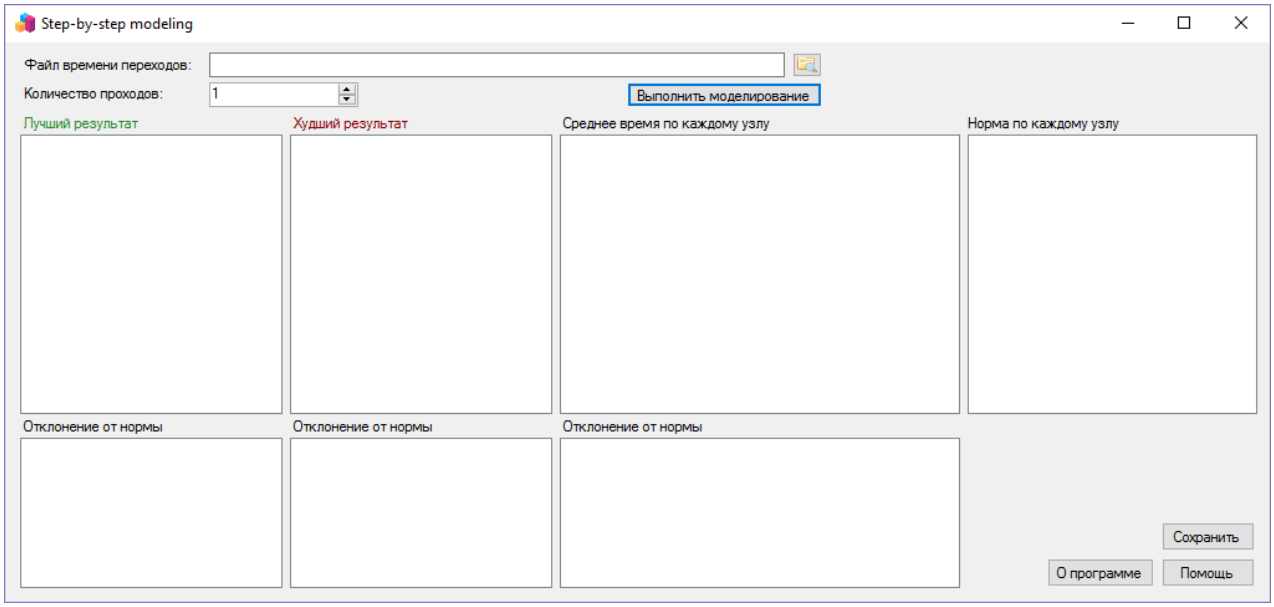


Рисунок 8.1 – Головна форма програми

На головній формі є поле для вводу посилання на файл з часом або вибору за допомогою натискання відповідної кнопки на формі, файл необхідно вказати до початку моделювання, якщо цього не було зроблено буду виведено відповідне повідомлення, див. рис. 8.2.

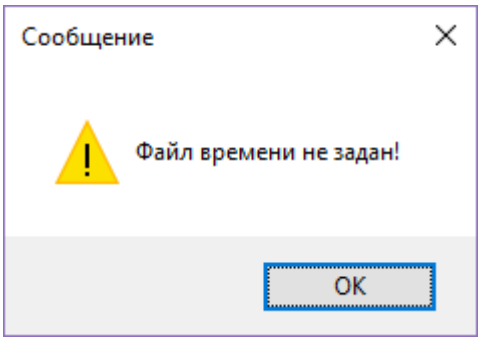


Рисунок 8.2 – Повідомлення про помилку

Також перед початком моделювання необхідно вказати кількість повторень моделювання, після чого натиснути кнопку «Выполнить моделирование». Після завершення моделювання на головній формі з’являться результати, див. рис. 8.3.

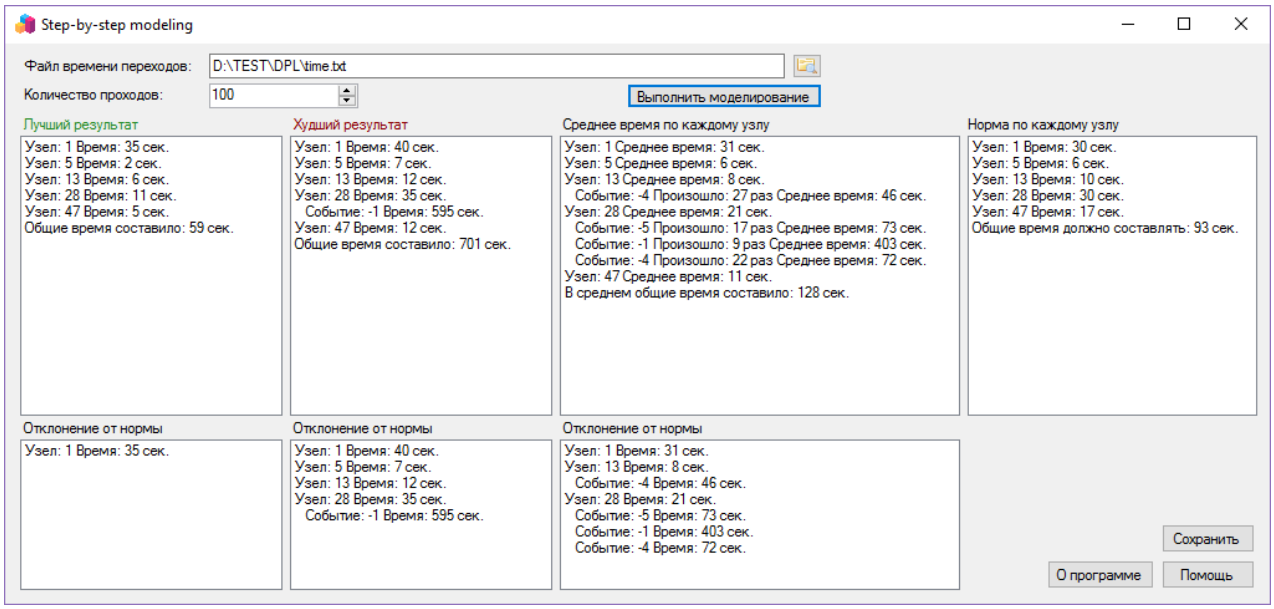


Рисунок 8.3 – Головна форма програми після завершення моделювання

У вікні «Лучший результат» виведено найкращій результат моделювання за створеним деревом переходів, у вікнах «Худший результат» та «Среднее время по каждому узлу» виводиться найгірший результат та середній результат за кожним вузлом що відбувалися за час усіх проходів відповідно. У вікні «Норма по каждому узлу» виводиться канонічний час за котрий повинно бути пройдено усі вузли. У полях що знаходяться нижче виводяться ті вузли що перевищують очікуваний(канонічний) час, тобто слабкі місця у цьому технологічному процесі.

Для завершення роботи програми необхідно натиснути на кнопку виходу на головній формі (рис. 8.3).

9 ПОРЯДОК РАБОТИ З ПРОГРАМОЮ

Технологію роботи з програмою і порядок виконання самої програми:

1 Запуск системи, 1 – 5 хв.

2 Додавання у систему посилання на файл у котрому вказано кількість часу для кожного вузлу та події, 1 – 10 хв.

3 Вказування необхідної кількості повторень, 1 – 5 хв.

4 Проходження побудови дерева переходів та імітаційного моделювання, 2 – 20 хв, залежить від кількості повторень та потужності робочої станції.

5 Перегляд результатів, 5 – 15 хв.

10 ПОВІДОМЛЕННЯ КОРИСТУВАЧУ

У таблиці 10.1 наведені повідомлення користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
"Выбран не соответствующий требованиям файл"	Вказано не вірне посилання або вибрано файл не відповідає вимогам	Натиснути кнопку «ОК», обрати відповідний файл
"Файл времени не задан!"	Не вказали посилання на файл	Натиснути кнопку «ОК», обрати відповідний файл
"Моделирование завершено успешно"	Успішне завершення моделювання	Натиснути кнопку «ОК»

ЗАТВЕРДЖЕНО

1116130.01208-01 ІЗ 01-ЛЗ

Методи поетапного моделювання складних процесів

Керівництво користувача

Керівництво експерту з побудови технологічних процесів

1116130.01208-01 ІЗ 01

Листів 11

2021

АНОТАЦІЯ

Документ 1116130.01208-01 ІЗ 01 «Методи поетапного моделювання складних процесів. Керівництво користувача. Керівництво експерту з побудови технологічних процесів» є складовою частиною документації на програму, що призначена для розробки програмного додатку «Інструментальний засіб «Методи поетапного моделювання складних процесів», та входить до складу документації на проект.

В документі наведені докладний перелік точних дій користувача при роботі з програмою. Програма реалізована на мові високого рівня C# у програмному середовищі Visual Studio, а також частина написана на мові логічного програмування Prolog. Об’єм пам’яті що займає програмний комплекс складає 2Мб. Конфігурація комп’ютера стандартна. Програма функціонує в середовищі MS Windows 7.

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Основним функціональним призначенням програмного продукту є надання можливостей дослідження часової ефективності складних процесів або їх частин за допомогою побудови дерева рішень та подальшого імітаційного моделювання з урахуванням виникнення раптових подій. Програмний комплекс дозволить провести дослідження доцільності використання даної реалізації методу поетапного моделювання в сфері дослідження та оптимізації складних технічних процесів та бізнес процесів, а також наддасть можливість аналізу та пошуку слабких місць у дереві процесів, що полегшить роботу відповідних спеціалістів з відстежування та контролю якості виконання процесів у компаніях різної направленості.

Даний програмний продукт не слід використовувати у сферах не пов’язаних з ієрархічною структурою виконання процесів, а також у тих сферах навчання де не можливо використати ієрархічну структуру виконання процесів, або при відсутності кваліфікованого персоналу з контролю якості виконання процесів.

2 УМОВИ ЗАСТОСУВАННЯ

Розроблювальний програмний продукт розрахований на використання на персональних комп'ютерах під керування операційної системи Windows Win7 або краще, що мають наступні мінімальні характеристики:

– процесор Intel Core 2 Duo 2.4 GHz або краще;

– пам'ять 2 GB DDR2 або краще;

– вільний дисковий простір на програмний комплекс 50 МБ або більше;

– наявність CD привода або USB роз’ємну для встановлення необхідного ПЗ;

– монітор з відео настройкою екрана 1024x768 або вище;

– клавіатура;

– маніпулятор миша.

Програма повинна виконуватися під керуванням ОС Windows WIN7 або новішою з використанням програми Prolog.

3 ВИКОНАННЯ ПРОГРАМИ

Для запуску програми знайдіть на робочій машині файл «diplom\bin\diplom.exe» та за допомогою миші оберіть програму. Після запуску програми на екрані з'являється вікно головної форми, представлене на рис 3.1.

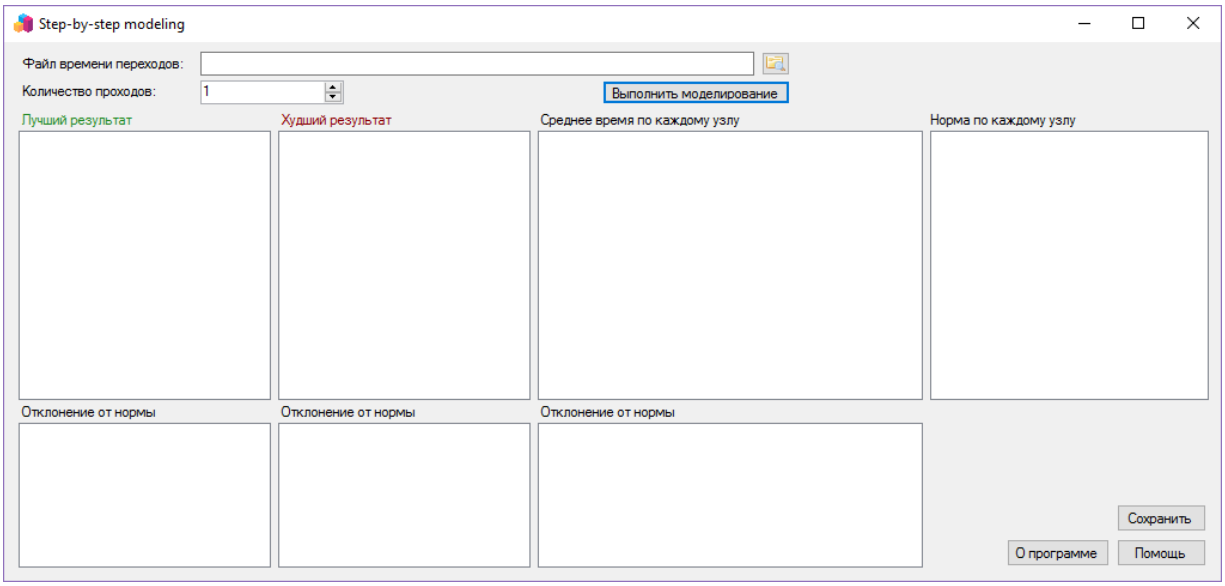


Рисунок 3.1 – Головна форма програми

Перед запуском моделювання вкажіть файл з часом переходів та встановіть кількість повторень моделювання. Для цього на головній формі є поле для вводу посилання на файл з часом або вибору за допомогою натискання відповідної кнопки на формі див. рис. 3.2.

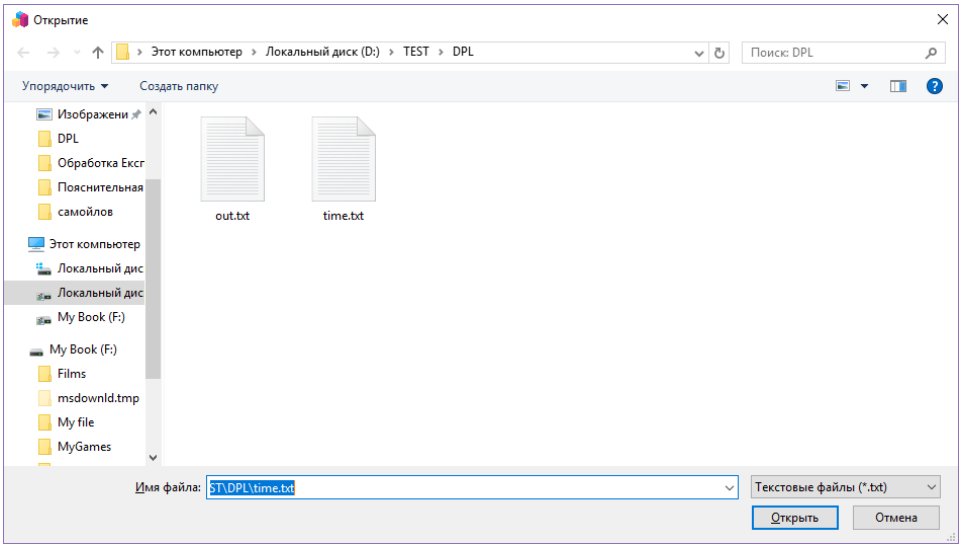


Рисунок 3.2 – Вікно вибору файлу з часом

Файл необхідно вказати до початку моделювання, якщо цього не було зроблено буду виведено відповідне повідомлення, див. рис. 3.3.

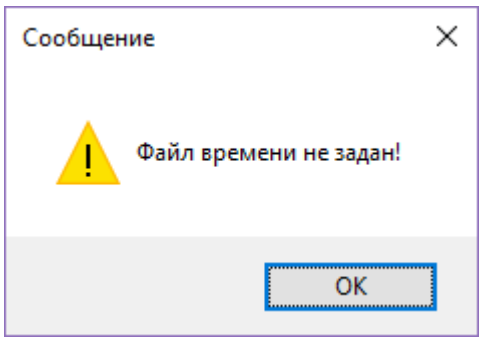


Рисунок 3.3 – Повідомлення про помилку

Якщо це сталося натисніть кнопку «ОК» та ввести посилання на файл.

Якщо буде введено не вірне посилання на файл буде видано наступне повідомлення, див. рис. 3.4.

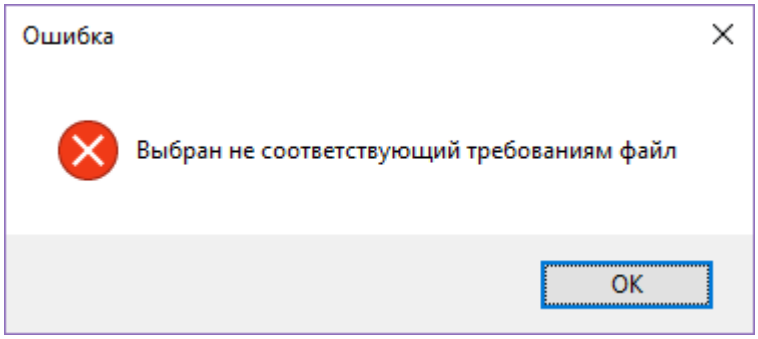


Рисунок 3.4 – Вікно повідомлення про помилку

Якщо це сталося натисніть кнопку «ОК» та ввести посилання на файл.

Також перед початком моделювання вкажіть кількість повторень моделювання, після чого натисніть кнопку «Выполнить моделирование». Після завершення моделювання з’явиться повідомлення о успішному закінченні моделювання, див. рис. 3.5.

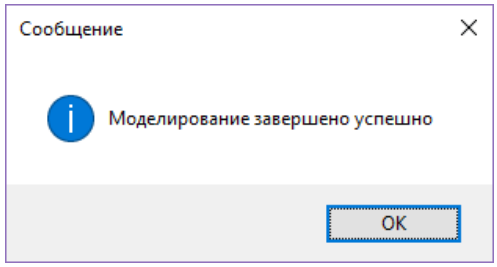


Рисунок 3.5 – Вікно повідомлення про успішне завершення моделювання

Натисніть кнопку «ОК», на головній формі з’являться результати, див. рис. 3.6.

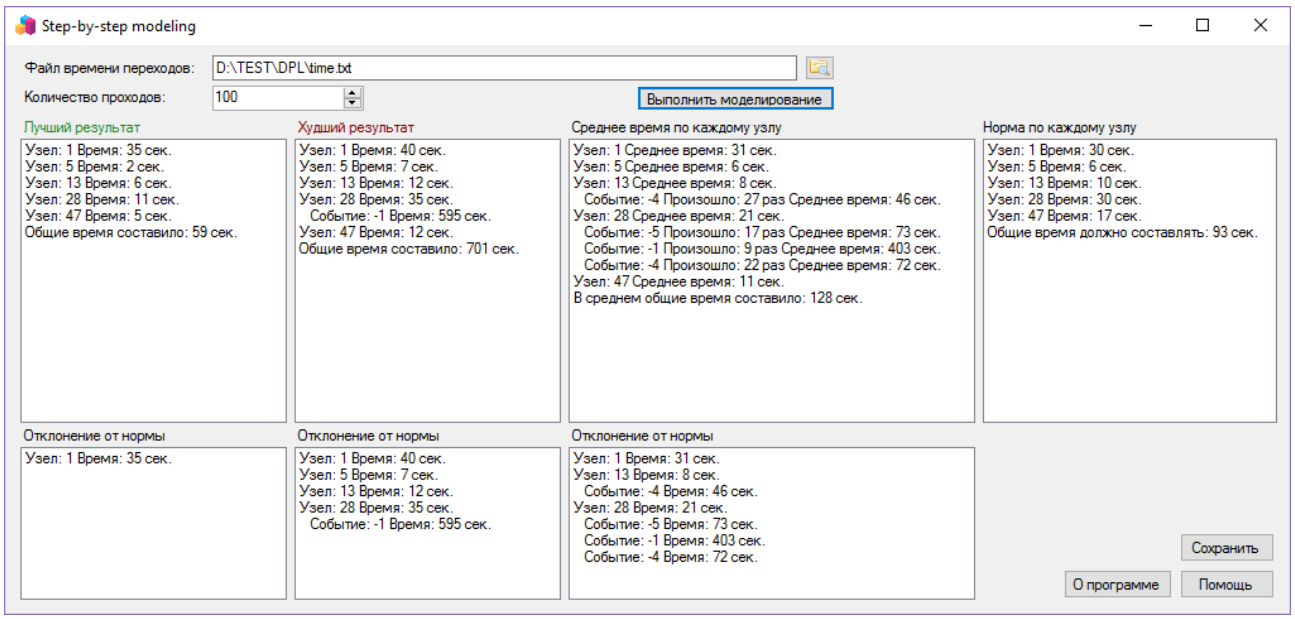


Рисунок 3.6 – Головна форма програми після завершення моделювання

У вікні «Лучший результат» виведено найкращій результат моделювання за створеним деревом переходів, у вікнах «Худший результат» та «Среднее время по каждому узлу» виводиться найгірший результат та середній результат за кожним вузлом що відбувалися за час усіх проходів відповідно. У вікні «Норма по каждому узлу» виводиться канонічний час за котрий повинно бути пройдено усі вузли. У полях що знаходяться нижче виводяться ті вузли що перевищують очікуваний(канонічний) час, тобто слабкі місця у цьому технологічному процесі. Для збереження результатів натисніть кнопку «Сохранить» на головній формі програми, після чого на екрані з’явиться віконце в котрому оберіть куди зберегти файл та вкажіть назву файлу, див. рис. 3.7.

Якщо файл з таким ім’ям вже існує буде виведено відповідне повідомлення, якщо необхідно перезаписати файл натисніть кнопку «Да», якщо ні то «Нет», див. рис. 3.8.

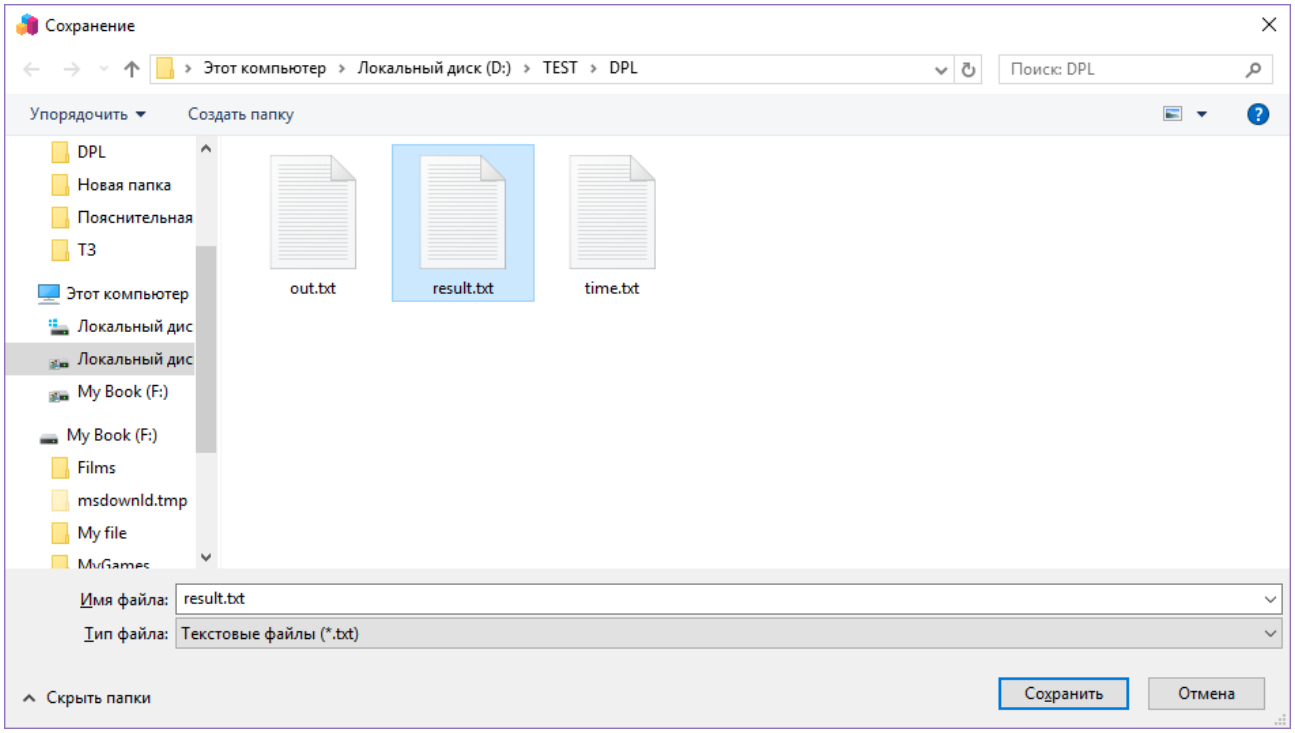


Рисунок 3.7 – Вікно збереження файлу

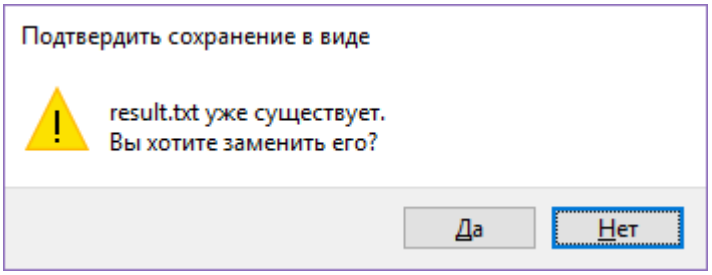


Рисунок 3.8 – Повідомлення про існування файлу з таким ім’ям

При необхідності отримати інформацію о програмі натисніть на головній формі на кнопку «О программе», після чого буде видано вікно з інформацією о програмі, див. рис. 3.9.

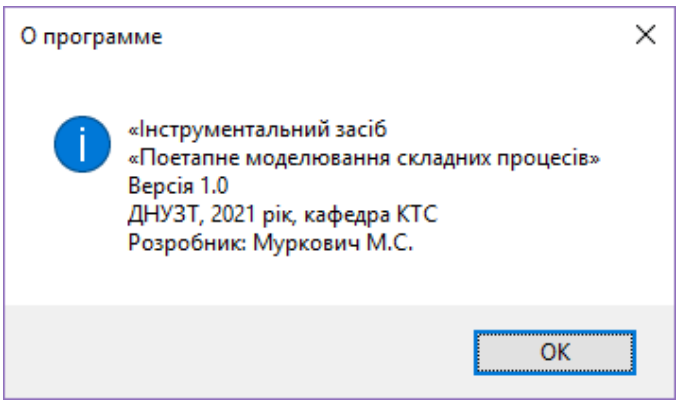


Рисунок 3.9 – Вікно «О программе»

Для виходу з цього вікна натисніть на кнопку «ОК».

У разі необхідності швидкої допомоги для знайомства з інтерфейсом натисніть кнопку «Помощь» на головній формі, після чого на головній формі з’являться додаткові написи з описом інтерфейсу, див. рис. 3.10.

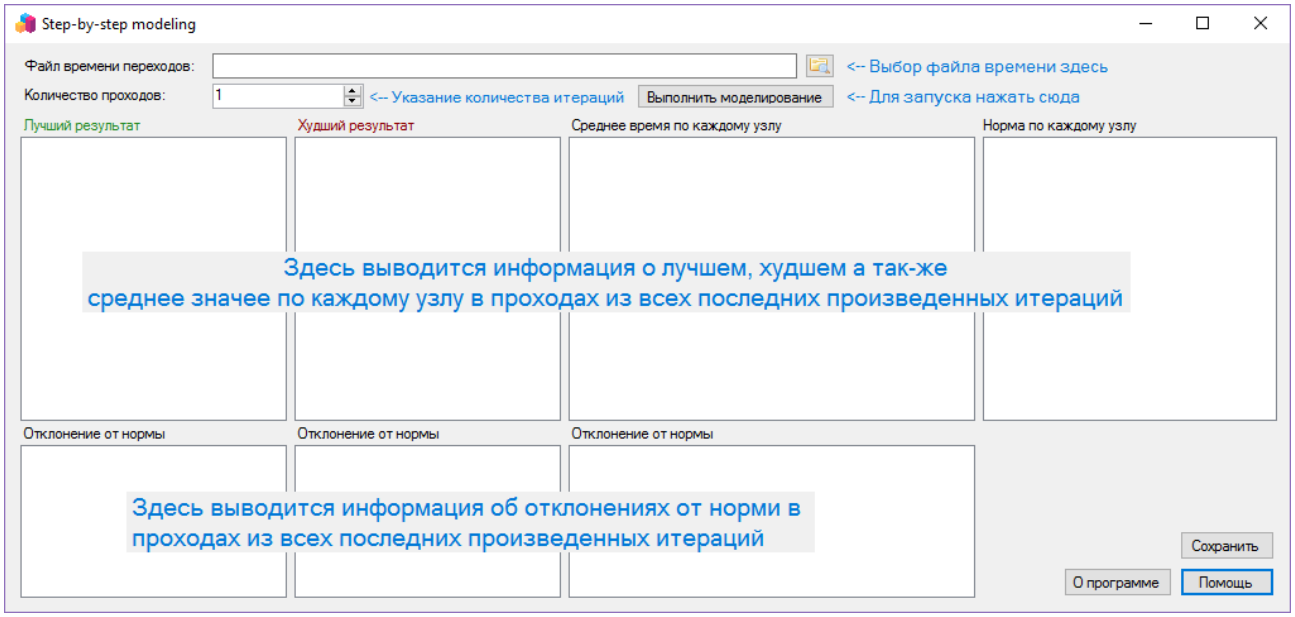


Рисунок 3.10 – Головна форма програми з підказками

Для завершення роботи програми необхідно натиснути на кнопку виходу (хрестик) на головній формі.

4 ПОВІДОМЛЕННЯ

У таблиці 4.1 наведені повідомлення користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 4.1 – Повідомлення користувачу

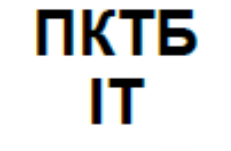
Текст повідомлення	Опис ситуації	Рекомендовані дії
"Выбран не соответствующий требованиям файл"	Вказано не вірне посилання або вибрано файл не відповідає вимогам	Натиснути кнопку «ОК», обрати відповідний файл
"Файл времени не задан!"	Не вказали посилання на файл	Натиснути кнопку «ОК», обрати відповідний файл
"Моделирование завершено успешно"	Успішне завершення моделювання	Натиснути кнопку «ОК»

ДОДАТОК В

Міністерство освіти і науки України

Дніпровський національний університет
залізничного транспорту ім. акад. В. Лазаряна

Східний науковий центр транспортної академії наук



TEMPUS: CITISET & SEREIN & CRENG

ТЕЗИ
XIV Міжнародної науково-практичної конференції
«СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ»

ABSTRACTS
of the XIV International Conference
«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»

ТЕЗИСЫ
XIV Международной научно-практической конференции
«СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ, В ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ»

15.12.2020 – 16.12.2020

Дніпро
2020

Методи поетапного моделювання складних процесів

Горбова О.В., Муркович М.С.

Дніпровський національний університет залізничного
транспорту імені академіка В. Лазаряна,
(Україна)

В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області. Під час моделювання технологічних процесів застосовують різні методи й підходи.

Розвиток та ускладнення або навпаки спрощення технологій та технологічних процесів з часом призводить до нагромадження різних зайвих процедур, дій, зайвої роботи або навпаки може бути дещо спрощено, для цього необхідно постійно слідкувати за всією системою. Для поліпшення цього процесу та надання більшої наочності складним процесам та системам необхідно дослідити різні підходи та методики розробки моделювання методом поетапного моделювання.

Візуальне модулювання за допомогою UML діаграм дозволяє покращити оптимізацію складних процесів, шляхом візуалізації та поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі. Це дозволяє збільшити ефективність реалізації цих процесів у кілька разів і помітно поліпшити якість кінцевого продукту.

При моделюванні складного технологічного процесу виділяються наступні етапи: математичне моделювання, графічне моделювання та імітаційне моделювання.

Створення математичної моделі характеризується створенням математичної моделі на основі вхідних даних, котрі можуть представляти з себе статистичні ряди вхідної інформації та побудованими законами розподілу за цими рядами.

Створення графічної моделі характеризується побудовою ефективного графічного представлення з використанням математичних моделей, орієнтованого на візуалізацію процесу у виді UML діаграм.

Етап імітаційного моделювання пов'язаний з дослідженням властивостей реалізованої на попередньому етапі графічної моделі. Застосування статистичних та аналітичних інструментів обробки результатів чисельних експериментів, а також їх планування дозволяє отримати не тільки окремі результати функціонування моделі технологічного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе ітеративний процес запуску графічної моделі з різними наборами вихідних даних. Отримані результати підлягають подальшій обробці для виявлення функціональних залежностей і узагальнення результатів.

Таким чином, поставлена задача вирішується за допомогою формалізації, візуалізації та імітаційного моделювання, а також дослідження складних процесів. Це може бути використано при розробленні та/або удосконаленні будь яких процесів, що можуть бути представлені у вигляді статистики або іншої текстової інформації для аналізу системою та подальшої обробки.

ДОДАТОК Г

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

АТ «УКРАЇНСЬКА ЗАЛІЗНИЦЯ»

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ІМЕНІ АКАДЕМІКА В. ЛАЗАРЯНА

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

81 Всеукраїнської науково-технічної конференції молодих учених, магістрантів та студентів

«НАУКА І СТАЛИЙ РОЗВИТОК ТРАНСПОРТУ»

28 жовтня 2021 року

INFORMATION-TELECOMMUNICATION TECHNOLOGY TA COMPUTER MODELING

CONFERENCE PROCEEDINGS

81th all Ukrainian Scientific and Technical Conference of young scientists, masters and students

"SCIENCE AND SUSTAINABLE DEVELOPMENT OF TRANSPORT"

October 28, 2021

ДНІПРО
2021

ЗМІСТ

Методи поетапного моделювання складних процесів

Горбова О.В., Муркович М.С.

Дніпровський національний університет залізничного
транспорту імені академіка В. Лазаряна,
(Україна)

В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області. Під час моделювання технологічних процесів застосовують різні методи й підходи.

Розвиток та ускладнення або навпаки спрощення технологій та технологічних процесів з часом призводить до нагромадження різних зайвих процедур, дій, зайвої роботи або навпаки може бути дещо спрощено, для цього необхідно постійно слідкувати за всією системою. Для поліпшення цього процесу та надання більшої наочності складним процесам та системам необхідно дослідити різні підходи та методики розробки моделювання методом поетапного моделювання.

Представлення процесу у вигляді графу переходів дозволяє покращити оптимізацію складних процесів, шляхом візуалізації та поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі. Це дозволяє збільшити ефективність реалізації цих процесів у кілька разів і помітно поліпшити якість кінцевого продукту.

При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання спрощення процесу моделювання. На базі формалізованого підходу до побудови і дослідження моделей є доцільним створення програмного комплексу, що буде допомагати та прискорить діяльність фахівців.

При моделюванні складного технологічного процесу виділяються етапи математичного та імітаційного моделювання.

Перший етап характеризується створенням наборів фактів на основі вхідних даних, котрі можуть представляти з себе опис об’єкту дослідження та його вхідний стан або набори таких об’єктів.

Після занесення наборів фактів до системи відбуваються побудова дерева рішення досліджуваного об’єкту для подальшого дослідження.

Спроектована система, котра побудована та працює за схожими алгоритмами з мовою логічного програмування Prolog, також має зручним графічний інтерфейс, та дозволяє працювати людям без спеціальної підготовки у напрямку програмування.

Етап імітаційного моделювання пов'язаний з дослідженням часової характеристики реалізованого на попередньому етапі дерева рішення. Застосування статистичних та аналітичних інструментів обробки результатів чисельних експериментів, а також їх планування дозволяє отримати не тільки окремі результати функціонування моделі технологічного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе ітеративний процес запуску математичної моделі з різними наборами подій, що будуть впливати на час виконання процесу. Отримані результати підлягають подальшій обробці для виявлення

функціональних залежностей і узагальнення результатів.

Таким чином, поставлена задача вирішується за допомогою формалізації складного процесу та імітаційного моделювання, а також дослідження моделі складного процесу. Це може бути використано при розробленні та/або удосконаленні складних процесів, що можуть бути представлені у вигляді наборів фактів.

ДОДАТОК Д

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ІМЕНІ АКАДЕМІКА В. ЛАЗАРЯНА

АТ «УКРАЇНСЬКА ЗАЛІЗНИЦЯ»

ПАТ «КРЮКІВСЬКИЙ ВАГОНОБУДІВНИЙ ЗАВОД»

АТ «ДНІПРОПЕТРОВСЬКИЙ СТРІЛОЧНИЙ ЗАВОД»

ТОВ «ЗАВОД РЕЙКОВИХ СКРІПЛЕНЬ»

INSTYTUT KOLEJNICTWA

КОРПОРАЦІЯ «ДЕТАЛЬ ВАГОН ГРУП»

ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

МАТЕРІАЛИ
81 МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

22.04.2021–23.04.2021

Дніпро – 2021

СЕКЦІЯ 14 «ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА КІБЕРБЕЗПЕКА НА ТРАНСПОРТІ»

МОДЕЛЮВАННЯ СКЛАДНИХ ПРОЦЕСІВ ШЛЯХОМ ДЕКОМПОЗИЦІЇ ТА ПОЕТАПНОГО МОДЕЛЮВАННЯ ДЛЯ ДОСЛІДЖЕННЯМ ЧАСОВОЇ ЕФЕКТИВНОСТІ

Горбова О.В., Муркович М.С.

Дніпровський національний університет залізничного
транспорту імені академіка В. Лазаряна,
(Україна)

Gorbova Alexandra, Murkovich Mykola. Modeling of complex processes by decomposition and step-by-step modeling for the study of time efficiency.

Summary. *The development and complication or the simplification of technologies and technological processes over time leads to the accumulation of various unnecessary procedures, actions, unnecessary work or, conversely, can be somewhat simplified, for this it is necessary to constantly monitor the entire system. To improve this process and provide greater clarity to complex processes and systems, it is necessary to explore different approaches and techniques for developing modeling by stepwise modeling.*

В сучасному світі існує безліч різноманітних сфер застосування моделювання, це може бути моделювання за допомогою математичного апарату, створення моделей шляхом збору статистичної інформації, проведення різноманітних випробовувань для подальшого імітаційного моделювання та багато яких інших.

Розвиток технологій, поява нових напрямків виробництва або зміна цілей підприємства може призвести до появи зайвих процедур, дій або навіть цілої низки функцій. Для вирішення цієї проблеми існують окремі спеціалісти, що займаються оптимізацією та зміною процесів на підприємствах, для цього їм необхідно постійно слідкувати за станом усієї системи або окремої її частини. Для покращення результату та надання більшої наочності та точності складним процесам, необхідно програмне дослідження структури процесу, адже комп'ютер може надати більш точні дані та у більшому спектрі ситуацій ніж можливо перевірити на виробництві або підприємстві.

Для складних процесів найчастіше є можливість провести процес декомпозиції, в ході якого складний процес буде розбито на підпроцеси, і з'явиться можливість працювати з різними рівнями абстракції, що дозволить провести поетапне моделювання процесу.

При поетапному моделюванні можливо виділити етапи математичного та імітаційного моделювання.

Представлення процесу у вигляді графу переходів дозволяє покращити оптимізацію складних процесів, шляхом візуалізації та поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі.

Перший етап характеризується створенням наборів фактів на основі вхідних даних, котрі повинні представляти з себе опис об'єкту дослідження, вхідний стан об'єкту, властивості процесу або набори таких даних.

Після занесення наборів фактів до системи відбуваються побудова дерева рішення досліджуваного об'єкту або процесу для подальшого дослідження.

Спроектована система, котра побудована та працює за схожими алгоритмами з мовою логічного програмування Prolog, також має зручним графічний інтерфейс, та дозволяє працювати людям без спеціальної підготовки у напрямку програмування.

Наступний етап – етап імітаційного моделювання, пов'язаний з дослідженням часової характеристики реалізованого на попередньому етапі дерева рішення з урахуванням виникнення випадкових подій, що впливають на час. За допомогою статистичних та аналітичних інструментів обробки результатів чисельних експериментів дає можливість отримати не тільки окремі результати функціонування моделі складного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе ітеративний процес запуску математичної моделі для отримання як найбільш достовірних результатів, так і максимальних результатів. Отримані дані підлягають подальшій обробці для виявлення функціональних залежностей і узагальнення результатів.

Таким чином, поставлена задача вирішується за допомогою декомпозиції, дослідження, формалізації складного процесу та імітаційного моделювання, а також дослідження отриманих результатів. Це може бути використано при розробленні та/або удосконаленні складних процесів, що можуть бути представлені у вигляді наборів фактів.

ДОДАТОК Е

Подано до друку

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ

О. В. ГОРБОВА[1], М. МУРКОВИЧ[2]

[1]Кафедра «Комп'ютерні інформаційні технології», Дніпровський національний університет залізничного транспорту імені академіка В.Лазаряна, вул. Лазаряна, 2, Дніпро, Україна, 49010, тел.. +38 (056) 373-15-35, ел. пошта alexandra.gorbova@gmail.com, ORCID - 0000-0002-5612-2715.

[2]Кафедра «Комп'ютерні інформаційні технології», Дніпровський національний університет залізничного транспорту імені академіка В.Лазаряна, вул. Лазаряна, 2, Дніпро, Україна, 49010, тел.. +38 (056) 373-15-35, ел. пошта nikolay.murkovich@gmail.com, ORCID - 0000-0002-1031-8245.

МОДЕЛЮВАННЯ СКЛАДНИХ ПРОЦЕСІВ НА ОСНОВІ ПОЕТАПНОГО МОДЕЛЮВАННЯ

Мета. При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання формалізація процесу моделювання та використання методики поетапного моделювання для проектування технологічних процесів. Такий підхід дозволяє проектувати процеси та завдання поетапно та включає такі етапи: фізичне моделювання, математичне моделювання, дискретне комп'ютерне моделювання та імітаційне моделювання. **Методика.** Для розв’язку задачі використовується методологія з використанням поетапного моделювання. Моделювання передбачає 3 етапи та використовує алгоритм декомпозиції, тобто розглядає задачу від глобального до детального. На першому етапі даної реалізації проводиться збір необхідної інформації для проведення експерименту. Ця інформація представляється у вигляді статистики. На другому етапі відбувається подальша обробка, що виконується шляхом перевірки на відповідність вхідних даних та процесу з питанням, як повинен виконуватися цей процес. На останньому етапі відбувається моделювання проходів цього фрагменту, який представлений ланцюгом переходів, отримання статистики часової ефективності цього процесу, слабкі місця процесу та можливість порівняти результати отримані під час моделювання та у реальному процесі, а також можливість спрогнозувати результати та дії у системі на майбутнє. **Результати.** Методика може бути використана для дослідженням складних технологічних процесів на підприємстві. Методика дозволяє моделювання складних процесів для отримання інформації про часову ефективність виконання технологічної операції, знаходження слабких місць у ній та закономірностей у виниканні випадкових подій, що можуть вплинути на операцію. Використання цього підходу може бути дуже ефективним у рамках систем в яких необхідний постійний контроль у режимі реального часу, оскільки даний засіб можливо модифікувати для цього додаванням наборів датчиків що будуть постійно посилати інформацію до системи або оборудувати додатковою системою що буде надавати вже готові пакети інформації. **Наукова новизна**. Удосконалено методику поетапного моделювання представлення, що полягає у одночасному використанні фізичного, математичного та імітаційного моделювання складних процесів на базі методології поетапного моделювання з набором етапів реалізації процесів. **Практична значимість**. Запропонована методика призначена для поетапного моделювання технологічного процесу с подальшою побудовою імітаційного програмування.

Ключові слова: технологічний процес; імітаційне програмування; моделювання технологічного процесу; математичне моделювання, моделювання автотранспортних пригод.

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

Вступ

В основі проектування технічного забезпечення, автоматизованих систем управління, розробки різноманітних технологічних процесів лежить моделювання предметної області, процесів або складових системи. Під час моделювання технологічних процесів застосовують різні методи й підходи.

Побудова моделей технологічних процесів є досить важливим завданням, тому що дозволяє передбачити їх особливості, що сприяють досягненню необхідних характеристик та властивостей. Побудова дерева рішення дозволяє покращити оптимізацію складних процесів, шляхом поетапного моделювання складних процесів, а завдяки імітаційному моделюванні можливо побачити та перевірити цю модель у часі.

При вирішенні практичних завдань, що вимагають створення і подальшого аналізу моделі, важливим критерієм є трудомісткість моделювання. Внаслідок цього виникає завдання формалізації процесу моделювання. На базі формалізованого підходу до побудови і дослідження моделей є доцільним створення програмного комплексу, що автоматизує і впорядковує діяльність фахівців.

У [10] автори пропонують підхід поетапної побудови моделі, що включає такі етапи, як фізичне моделювання, математичне моделювання, дискретне комп'ютерне моделювання та імітаційне моделювання (рис. 1).

На кожному з етапів передбачається активна взаємодія з базою знань з метою накопичення типових рішень їх подальшого використання. Для синтезу моделей об'єктів і систем управління, а також для адаптації типових рішень пропонується застосування інтелектуального структурно-параметричного синтезу, заснованого на ідеї використання еволюційних алгоритмів як засоби автоматичної генерації моделі у вигляді мережі Петрі.

На етапі фізичного моделювання проводиться дослідження технологічного процесу. Для цього може бути проведений ряд експериментів з існуючим обладнанням. При відсутності такої можливості створюються фізичні моделі, що відображають основні закономірності досліджуваних процесів.

Рис. 1– Схема поетапного моделювання

Основною метою побудови фізичної моделі є отримання статичних і динамічних характеристик процесу, що модулюється. Таким чином, реалізована і вивчена фізична модель дає можливість формально описати властивості досліджуваного процесу, визначити чисельні значення актуальних параметрів, оцінити їх кореляцію і динаміку зміни.

Після збору і обробки даних про функціонування фізичної моделі настає етап математичного моделювання. Залежно від особливостей технологічного процесу і цілей моделювання можуть бути обрані різні математичні апарати.

У разі моделювання схожих процесів доцільно використання шаблонних підходів, що дають приріст швидкості синтезу моделі. В цьому випадку виникає задача зберігання бази шаблонів і вибору необхідного шаблону моделі за заданими критеріями. Завдяки застосуванню систем управління базами даних і базами знань можливе створення автоматизованої системи, що пропонує на етапі математичного моделювання найбільш відповідну методологію і шаблон, використовуючи які створюється математичний опис досліджуваного процесу.

У деяких випадках побудова математичної моделі може бути ускладнена в зв'язку із складністю опису досліджуваного об'єкта. У цьому випадку може бути застосований підхід «чорного ящика», коли модель синтезується

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

лише на підставі накопиченої статистичної інформації про поведінку об'єкта. Для автоматизації цього процесу пропонується використання еволюційного алгоритму як засобу спрямованого пошуку структури і параметрів моделі, що адекватно описує отримані раніше вихідні емпіричні дані.

Етап імітаційного моделювання пов'язаний з дослідженням властивостей реалізованої на попередньому етапі дискретної комп'ютерної моделі. Застосування статистичних та аналітичних інструментів обробки результатів чисельних експериментів, а також їх планування дозволяє отримати не тільки окремі чисельні результати функціонування моделі технологічного процесу для одиничних вибірок вихідних умов, але і виявити якісні особливості поведінки модельованих систем і об'єктів, що володіють схожими характеристиками.

Імітаційне моделювання, як правило, включає в себе ітеративний процес запуску дискретної комп'ютерної моделі з різними наборами вихідних даних. Отримані результати підлягають подальшій обробці для виявлення функціональних залежностей та узагальнення результатів.

В узагальнення про дане дослідження можливо зробити висновок про схожість основної структури програмних засобів даної категорії, але і про різницю в підходах в реалізації та властивостях програмних засобів.

В статті [11] показано, що для дослідження складних процесів соціальної взаємодії на сучасному етапі розвитку системного моделювання все частіше використовуються ієрархічні системи когнітивних моделей, когнітивні архітектури та інтегровані системи моделювання. Що стосується даних процесів, система моделей повинна відображати різні абстракції опису структури, різноманітні аспекти її поведінки, етапи (ітерації) її еволюції в процесі функціонування і розвитку. Пропонується розглядати архітектуру в сукупності моделей в контексті необхідності для вирішення завдання на конкретному етапі дослідження.

Як показує практика системного моделювання найбільш значущими і, в той же час, найбільш складними є завдання аналізу і прогнозування розвитку процесів соціальної взаємодії. Такого роду завдання пов'язані з прогнозом досягнення довгострокових цілей шляхом адаптації до змін зовнішнього середовища. Завдання складні і вимагають врахування великої кількості факторів, інтересів, ризиків і наслідків, в їх рішеннях присутня висока ступінь невизначеності в оцінці зовнішнього середовища, слабка формалізація методів управління і широке використання експертних оцінок і знань, багатокритеріальність при оцінці прийнятих рішень. Характеризуючи проблему модельного підходу до аналізу та прогнозування складних процесів соціальної взаємодії, необхідно враховувати, що від постановки задачі моделювання до інтерпретації отриманих результатів, існує велика група складних науково-технічних проблем, до основних з яких можна віднести наступні: ідентифікацію реальних об'єктів, вибір виду моделей, побудова моделей і їх програмну реалізацію, взаємодія дослідника з моделлю в ході комп'ютерного експерименту, перевірку правильності отриманих в ході моделювання результатів, виявлення основних закономірностей, досліджених в процесі моделювання. Залежно від об'єкта моделювання та виду використовуваної моделі ці проблеми можуть мати різну значимість.

Зазначено, що відповідно до принципу необхідної різноманітності, «різноманітність засобів моделювання повинно бути більше або, принаймні, так само різноманітності об'єкта моделювання». В цьому випадку традиційний спосіб опису різних аспектів системи і процесу її створення, що передбачає послідовний розвиток однієї моделі, не є ефективним. Сама наявність такої єдиної моделі викликає сумніви, тому що занадто складний і багатогранний предмет опису – від реальних об'єктів і процесів до абстрактних інформаційних та інших об'єктів.

Наступний крок у дослідженні складних процесів соціальної взаємодії пропонується виконувати на основі інтегрованих систем моделювання. Принципова їх відмінність полягає в використанні різних парадигм моделювання для дослідження відповідних властивостей соціальних процесів.

Стосовно до складних процесів соціальної

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

взаємодії сукупність моделей повинна відображати різні абстракції опису структури, різноманітні аспекти її поведінки, етапи (ітерації) її еволюції в процесі функціонування і розвитку. Кожна з моделей має унікальні властивості, які відсутні у інших, і тому в різного ступеня відповідає реальним процесам. Використовувана модель, що інтерпретує досліджуване властивість процесів, в свою чергу, визначає і те, як буде осмислюватися проблема, і які рішення будуть прийматися. Необхідно розглядати архітектуру сукупності моделей в контексті необхідності для вирішення завдання на конкретному етапі дослідження.

У [9] запропоновано методи моделювання технологічних процесів із позиції системного аналізу. У роботі запропоновано формальні підходи до описання математичних моделей та показано моделювання технологічних об'єктів, що розрізняються ступенем деталізації процесів.

Для роботи [3] характерним є підхід у створенні моделі технологічного процесу з позиції представлення технологічного виробничого процесу з використання методології поетапного моделювання та її формалізація.

Під час моделювання технологічних процесів застосовують різні методи й підходи. Усі вони мають свої особливості, проте до моделювання предметної області висуваються такі вимоги:

– однозначний опис структури предметної області;

– зрозумілість результатів передпроектного обстеження для замовників і розробників на основі застосування графічних засобів відображення моделі;

– реалізованість, під якою розуміють наявність засобів фізичної реалізації моделі предметної області в інформаційних системах;

– забезпечення оцінки ефективності реалізації моделі предметної області на основі певних методів та обчислюваних показників.

Ефективним способом описання функціонування об'єктів, що забезпечує високу інформативність та інтуїтивно зрозуміле представлення інформації, є візуальне моделювання. Під час візуального моделювання кожен елемент виробничого процесу зображують у вигляді графічного позначення. Графічну модель можна створювати як на паперових носіях, так і за допомогою спеціалізованого програмного забезпечення на ЕОМ. Відображення наявних виробничих процесів у вигляді простих діаграм і коротких описів допомагає досягти єдиного розуміння чинних норм та оперативних процедур між розробником і замовником проектів розвитку залізничних станцій.

Мета

Основною метою цієї статті є представлення методики моделювання складних процесів на основі поетапного моделювання шляхом декомпозиції складних процесів.

Методика

Для рішення поставленої задачі розроблений алгоритм, що відповідає потребам, які виникають під час дослідження та є ефективним рішенням задачі.

У задачі «розробка алгоритму» у більшості випадків є використання відомих методик, їх комбінування та використання різних підходів до реалізації для отримання необхідного результату. Для розв'язку задачі використовується методологія з використанням поетапного моделювання. Моделювання передбачає 3 етапи та використовує алгоритм декомпозиції, тобто розглядає задачу від глобального до детального.

Перший етап. На першому етапі даної реалізації проводиться збір необхідної інформації для проведення експерименту. Ця інформація представляється у вигляді статистики, наборів інтервалів або в будь якому іншому вигляді, лише за одним обмеженням, щоб цю інформацію можливо було представити згідно вимог методики.

Вимоги до представлення інформації:

– інформація про процес, що досліджується, представляється у вигляді ієрархічного дерева процесів, де кожен вузол має свій номер, там до кожного вузлу зіставляється умова, за якою система буде розуміти куди буде просуватися процес за ходом виконання алгоритму. Номер необхідний для розуміння місцеположення та

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

формування шляху проходження цього дерева, приклад інтерпретації зовнішнього вигляду дерева наведено на рис. 2. Після завантаження вся ця структура має бути записана у відповідному вигляді, для зчитування системою необхідної інформації та виконання необхідних дій, приклад запису вхідного файлу з правилами переходів наведено на рис. 3; інформація про час виконання кожного вузлу або час необхідний для переходу з одного вузлу до іншого повинна бути записана у окремий файл у відповідній формі, приклад файлу зі вказаним необхідним часом зображено на рис. 5;

– необхідно вказувати кількість повторень моделювання часу проходження отриманого

–

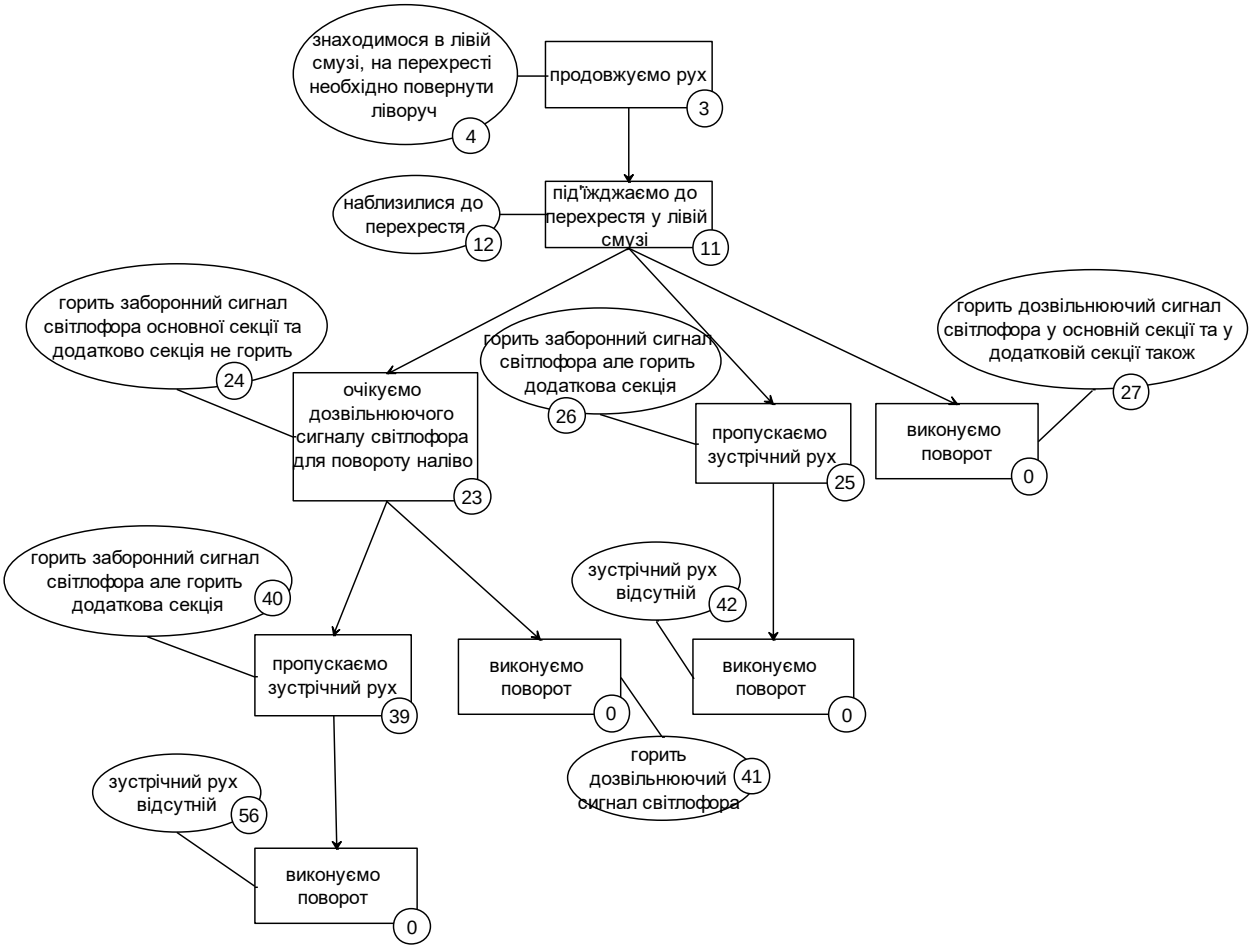


Рис. 2 – Схема переходів у дереві ланцюга

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

rules.pl — Блокнот

Файл Правка Формат Вид Справка

```
%% база фактів
fact(1,'приближаемся к перекрестку',[2]).
fact(2,'видем впереди перекресток',[]).
fact(3,'продолжаем движение',[1,4]).
fact(4,'находимся в левой полосе, на перекрестке необходимо повернуть налево',[]).
fact(5,'включаем левый указатель поворота для перестроения',[1,6]).
fact(6,'находимся в правой полосе, на перекрестке необходимо повернуть на лево',[])
fact(7,'включаем правый указатель поворота для перестроения',[1,8]).
fact(8,'находимся в левой полосе, на перекрестке необходимо проехать прямо',[]).
fact(9,'продолжаем движение',[1,10]).
fact(10,'находимся в правой полосе, на перекрестке необходимо проехать прямо',[]).
fact(11,'подезжаем к перекрестку в левой полосе',[3,12]).
fact(12,'приблизились к перекрестку',[]).
fact(13,'выполняем перестроение в левую полосу',[5,14]).
fact(14,'помеха слева отсутсвует',[]).
fact(15,'пропускаем попутный транспорт',[5,16]).
fact(16,'помеха слева присутсвует',[]).
fact(17,'пропускаем попутный транспорт',[7,18]).
fact(18,'помеха справо присутсвует',[]).
fact(19,'выполняем перестроение в правую полосу',[7,20]).
fact(20,'помеха справо отсутсвует',[]).
fact(21,'подезжаем к перекрестку в правой полосе',[9,22]).
fact(22,'приблизились к перекрестку',[]).
fact(23,'ожидаем разрешающего сигнала светофора для поворота на лево',[11,24]).
fact(24,'горит запрещающий сигнал светофора, дополнительная секция не горит',[]).
fact(25,'пропускаем встречное движение',[11,26]).
fact(26,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(27,'горит разрешающий сигнал светофора, дополнительная секция горит',[]).
fact(28,'подезжаем к перекрестку в левой полосе',[13,29]).
fact(29,'приблизились к перекрестку',[]).
fact(30,'выполняем перестроение в левую полосу',[15,31]).
fact(31,'помеха слева отсутсвует',[]).
fact(32,'выполняем перестроение в правую полосу',[17,33]).
fact(33,'помеха справо отсутсвует',[]).
fact(34,'подезжаем к перекрестку в правой полосе',[19,35]).
fact(35,'приблизились к перекрестку',[]).
fact(36,'ожидаем разрешающего сигнала светофора для проезда по прямой',[21,37]).
fact(37,'горит запрещающий сигнал светофора',[]).
fact(38,'горит разрешающий сигнал светофора',[]).
fact(39,'пропускаем встречное движение',[23,40]).
fact(40,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(41,'горит разрешающий сигнал светофора',[]).
fact(42,'встречный движение отсутствует',[]).
fact(43,'ожидаем разрешающего сигнала светофора для поворота на лево',[28,44]).
fact(44,'горит запрещающий сигнал светофора, дополнительная секция не горит',[]).
fact(45,'пропускаем встречное движение',[28,46]).
fact(46,'горит запрещающий сигнал светофора, дополнительная секция горит',[]).
fact(47,'горит разрешающий сигнал светофора, дополнительная секция горит',[]).
fact(48,'подезжаем к перекрестку в левой полосе',[30,49]).
fact(49,'приблизились к перекрестку',[]).
fact(50,'подезжаем к перекрестку в правой полосе',[32,51]).
fact(51,'приблизились к перекрестку',[]).
```

Рис. 3 – Вид вхідного файлу

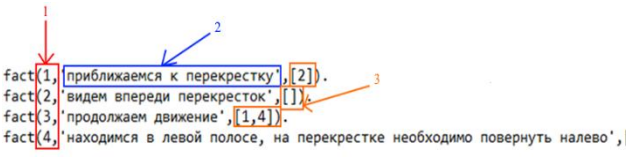


Рис. 4 – Пояснення до рис. 3

На рис. 4 наведено:
1 – номер вузлу;
2 – опис вузлу;
3 – пов’язані вузли.

time.txt — Блокнот

Файл Правка Формат Вид Справка

```
-1,60,600:8.авария на перекрестке
-2,60,420:4.авария на попутной полосе
-3,60,240:4.авария на встречной полосе
-4,10,120:25.тс с вкл спецсигналами
-5,10,240:20.неработает светофор
1,10,50
3,10,50,-1;-4;-5
5,2,10
7,2,10
9,10,50
11,10,50,-1;-4;-5
13,5,15,-4
15,5,15,-2;-4
17,5,15,-2,-4
19,5,15,-4
21,10,50,-1;-4;-5
23,5,130,-1;-5
25,5,50,-1;-3;-4
27,5,30
28,10,50,-1;-4;-5
30,10,20,-4
32,10,20,-4
34,10,50,-1;-4;-5
36,5,130,-1;-5
38,5,30
39,5,50,-1;-3;-4
41,5,30
42,5,30
43,5,130,-1;-5
45,5,50,-1;-3;-4
47,5,30
48,10,50,-1;-4;-5
50,10,50,-1;-4;-5
52,5,130,-1;-5
54,5,30
55,5,30
56,5,30
57,5,50,-1;-3;-4
```

Рис. 5 – Файл із необхідним часом

Легенда до рис. 5 наведно на рис. 6.

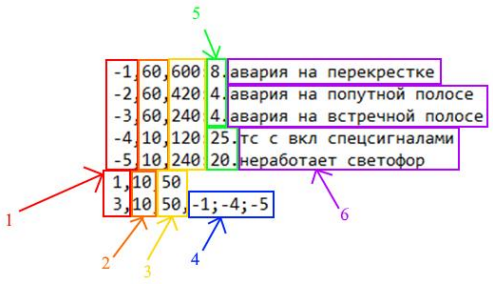


Рис. 6 – Пояснення до рис. 5

На рис. 6 наведено:
1 – номер вузлу;
2 – початок интервалу часу виконання;
3 – кінець інтервалу часу виконання;
4 – номер вузлів вірогідних випадковиї події;
5 – вірогідність виникнення події;
6 – пояснювальні коментарі.

Для подальшої обробки те, яким шляхом було отримано ці данні не є важливим, але для

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

подальшого використання цього алгоритму та методики на практиці на цьому етапі може залучається автоматизована система. Завданням такої системи стає відстежування станів об'єктів системи під час роботи та передачі цих даних для подальшої обробки. Тобто, у перспективі є можливість повної автоматизації системи з використанням цього алгоритму, за умови, що до процесу, що відстежується є можливість використання автоматизованого підходу.

Другий етап. На цьому етапі відбувається подальша обробка інформації що була надана на першому етапі.

Обробка виконується шляхом перевірки на відповідність вхідних даних та процесу з питанням, як повинен виконуватися цей процес. Алгоритм на відповідність вхідних даних та процесу виконується за допомогою програми, написаної на мові Prolog. Як результат роботи програми, будується дерево переходів що відповідає тим підпроцесам, що відбуваються в системі або об'єкті під час виконання процесу. Побудова дерева виконуються за допомогою зворотнього ланцюжку виводу. У результаті проходження даного етапу буде отримано проміжний результат у вигляді ланцюга переходів, приклад проміжного файлу із ланцюгом виконання операцій наведено на рис. 7.

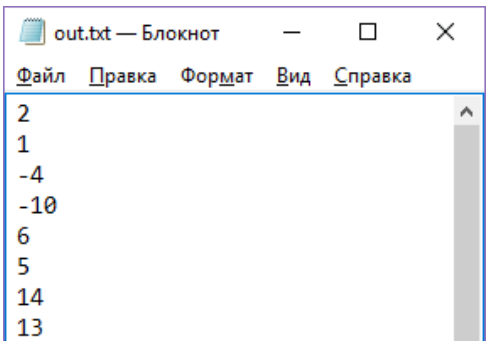


Рис. 7 – Проміжного файлу із ланцюгом виконання операцій

Третій етап. На останньому етапі відбувається моделювання проходів цього фрагменту, який представлений ланцюгом переходів, отримання статистики часової ефективності цього процесу, слабкі місця процесу та можливість порівняти результати отримані під час моделювання та у реальному процесі, а також можливість спрогнозувати результати та дії у системі на майбутнє.

В залежності від вказаної кількості повторень результат може істотно відрізнятися, чим більша кількість повторень тим точніший буде результат.

Результати

Розроблена методика може бути використаний для дослідження прикладної задачі або для дослідженням складних технологічних процесів на підприємстві. Крім того, методика дозволяє моделювання складних процесів для отримання інформації про часову ефективність виконання технологічної операції, знаходження слабких місць у ній та закономірностей у виниканні випадкових подій, що можуть вплинути на операцію. Використання цього підходу може бути дуже ефективним у рамках систем в яких необхідний постійний контроль у режимі реального часу, оскільки даний засіб можливо модифікувати для цього додаванням наборів датчиків що будуть постійно посилати інформацію до системи або оборудувати додатковою системою що буде надавати вже готові пакети інформації. Також цей підхід можливо модифікувати для використання у мануальному режимі, шляхом модифікації системи створення та збору інформації необхідної для моделювання, а отримані результати можуть бути використані при побудові нового процесу або модифікації будь-якого іншого процесу, також теоретично можливе використання цього підходу для роботи з бізнес-процесами, але для підтвердження цього необхідне більше детальне дослідження цього питання.

Наукова новизна та практична значимість

Наукова новизна роботи полягає у одночасному використанні фізичного, математичного та імітаційного моделювання складних процесів на базі методології поетапного моделювання з набором етапів реалізації процесів.

Модель дозволяє будувати дерево рішень задачі, що розв'язується та завдяки їй виконується аналіз часу виконання

ISSN 2307–3489 (Print), ISSN 2307–6666 (Online)
Наука та прогрес транспорту. Вісник Дніпропетровського національного університету залізничного транспорту, 2021, № 5 (95)

технологічної задачі в умовах випадкових подій.

Висновки

У цій статті запропоновано методику дослідженням складних технологічних процесів та операцій на основі методу імітаційного програмування та застосування методики поетапного моделювання.

Завдяки опису мови UML користувач має можливість в описі технологічних процесів методом візуального програмування, проте використання методики поетапного моделювання дозволить виділити об’єкти, наділити їх технологічними алгоритмами та переходами, коли імітаційне програмування дозволить змоделювати ситуації для виділення вузьких місць.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бобровский В. И. Функциональное моделирование работы железнодорожных станций: монография / В. И. Бобровский, Д. Н. Козаченко, Р. В. Вернигора, В. В. Малашкин; Днепропетр. нац. ун-т ж.-д. трансп. им. акад. В. Лазаряна. – Днепропетровск, 2015. – 269 с.
2. Глушков, В. М. Синтез цифровых автоматов / В. М. Глушков. – Москва: Физматлит, 1962. – 476 с.
3. Горбова, О. В. Формалізація технологічних процесів залізничних станцій на основі поетапного моделювання/ Горбова О. В. – Дніпро : Днепров. нац. ун-т ж.-д. трансп. им. акад. В. Лазаряна, 2016. – 167 с.
4. Дозорцев, В. М. Динамическое моделирование в оптимальном управлении и автоматизированном обучении операторов технологических процессов. Ч. 2. Компьютерные тренажеры реального времени / В. М. Дозорцев // Приборы и системы управления. – 1996.– № 8. – С.41–50.
5. Козаченко, Д. Н. Автоматизированное формирование функциональных моделей железнодорожных станций / Д. Н. Козаченко, Р. В. Вернигора, В. В. Малашкин // Транспортні системи та технології перевезень : зб. наук. пр. Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна. – Дніпропетровськ, 2014. – Вип. 8. – С. 65–73.
6. Козаченко, Д. Н. Объектно-ориентированная модель функционирования железнодорожных станций / Д. М. Козаченко // Наука та прогрес трансп. – 2013. – № 4 (46). – С. 47–55.
7. Козаченко, Д. М. Програмний комплекс для імітаційного моделювання роботи залізничних станцій на основі добового плану графіку / Д. М. Козаченко, Р. В. Вернигора, Р. Г. Коробйова // Залізн. трансп. України. – 2008. – № 4 (70). – С. 18–20.
8. Леоненков, А. В., Самоучитель UML / А. В. Леоненков. – СПб. : БХВ–Петербург, 2002.
9. Мальков, М. В. Моделирование технологических процессов: методы и опыт / М. В. Мальков, А. Г.Олейник, А. М. Федоров // Труды КНЦ РАН, 2010. – С. 93–101.
10. Петросов, Д. А. Поэтапное моделирование технологических процессов с использованием интеллектуального структурно-параметрического синтеза / Д. А. Петросов., В. А. Игнатенко // Фундаментальные исследования. – 2017. – № 12 (1). – С. 97–102. doi:10.17513/fr.41986
11. Розін М.Д. / Проблемы системного моделирования сложных процессов социального взаимодействия / М.Д.Розін, В.П. Свічкарьов // «Инженерный вестник Дона» - 2012.
12. Bianco, Vieri del. A formalization of UML statecharts for real-time software modeling [Електронний ресурс] / Vieri del Bianco, L. Lavazza, M. Mauri. – Режим доступу: https://cutt.ly/QeZxM3q – Назва з екрана. – Перевірено : 20.11.2019.
13. Gorbova, O. V. Modeling Work of Sorting Station Using UML / O. V. Gorbova // Наука та прогрес транспорту. – 2015. – № 1 (55). – С. 129–138. doi 10.15802/STP2015/38260
14. Harel, D. Statecharts: A visual formalism for complex systems / D. Harel //Science of Computer Programming. – North-Holland, 1987. – Vol. 8. – Iss. 3. – P. 231–274. doi:10.1016/0167-6423(87)90035-9
15. Harel, D., Statecharts: A visual formalisms / Devid Harel // Communications of the ACM. – New York, 1988. – Vol. 31, Iss. 5, P. 514–530.
16. Silva, M. On Modelling of Hierarchical and Distributed Discrete-Event Systems / M. Silva, J.-M. Colom, J. Julvez, C. Mahulea, J. H. van Schuppen, R. Su, J. Komenda, J. Raisch, S. Geist, P. Darondeau // The DISC Project Perspective. – 2007. – p. 85.
17. Zimmermann, A. Eine Quantitative Untersuchung des European Train Control System mit UML State Machines [Електронний ресурс] / A. Zimmermann, J. Trowitzsch. – Режим доступу: https://cutt.ly/HeZxV5r – Назва з екрана. – Перевірено : 20.11.2019.

А. В. ГОРБОВА[1*], Н. МУРКОВИЧ[2*]

[1*]Каф. «Компьютерные информационные технологии», Днипровский национальный университет железнодорожного транспорта имени академика В. Лазаряна, ул. Лазаряна, 2, Днипро, Украина, 49010, тел. +38 (056) 373 15 35, эл. почта alexandra.gorbova@gmail.com, ORCID 0000-0002-5612-2715

[1*]Каф. «Компьютерные информационные технологии», Днипровский национальный университет железнодорожного транспорта имени академика В. Лазаряна, ул. Лазаряна, 2, Днипро, Украина, 49010, тел. +38 (056) 373 15 35, эл. почта nikolay.murkovich@gmail.com, ORCID 0000-0002-1031-8245.

МОДЕЛИРОВАНИЕ СОСТАВНЫХ ПРОЦЕССОВ НА ОСНОВЕ ПОЭТАП-НОГО МОДЕЛИРОВАНИЯ

Цель. При решении практических задач, требующих создания и дальнейшего анализа модели, важным критерием является трудоемкость моделирования. В результате возникает задача формализация процесса моделирования и использования методики поэтапного моделирования для проектирования технологических процессов. Такой подход позволяет проектировать процессы и задачи поэтапно и включает в себя следующие этапы: физическое моделирование, математическое моделирование, дискретное компьютерное моделирование и имитационное моделирование. **Методика.** Для решения задачи используется методология с использованием поэтапного моделирования. Моделирование предусматривает 3 этапа и использует алгоритм декомпозиции, т.е. рассматривает задачу от глобального к детальному. На первом этапе данной реализации производится сбор необходимой информации для проведения эксперимента. Эта информация представляется посредством статистики. На втором этапе происходит дальнейшая обработка, выполняемая путем проверки на соответствие входных данных и процесса с вопросом, как должен выполняться этот процесс. На последнем этапе происходит моделирование проходов этого фрагмента, представленного цепью переходов, получение статистики временной эффективности этого процесса, слабые места процесса и возможность сравнить результаты полученные во время моделирования и реального процесса, а также возможность спрогнозировать результаты и действия в системе на будущее. **Результаты.** Методика может быть использована для исследования сложных технологических процессов на предприятии. Методика позволяет моделирование сложных процессов для получения информации о временной эффективности выполнения технологической операции, нахождении слабых мест в ней и закономерностей в возникновении случайных событий, которые могут повлиять на операцию. Использование этого подхода может быть очень эффективным в рамках систем, в которых необходим постоянный контроль в режиме реального времени, поскольку данное средство возможно модифицировать для этого добавлением наборов датчиков, которые будут постоянно посылать информацию к системе или оборудовать дополнительной системой, которая будет предоставлять уже готовые пакеты информации. **Научная новизна.** Усовершенствована методика поэтапного моделирования представления, заключающаяся в одновременном использовании физического, математического и имитационного моделирования сложных процессов на базе методологии поэтапного моделирования с набором этапов реализации процессов. **Практическая значимость**. Предложенная методика предназначена для поэтапного моделирования технологического процесса с последующим построением имитационного программирования.

Ключевые слова: технологический процесс; имитационное программирование; моделирование технологического процесса; математическое моделирование.

A. V. GORBOVA[1*], N. MURKOVICH[2*]

[1*]Dep. «Computer Information Technologies», Dnipro National University of Railway Transport named after Academician V. Lazaryan, Lazaryana St., 2, Dnipro, Ukraine, 49010, tel. +38 (056) 373 15 35, e-mail alexandra.gorbova@gmail.com, ORCID 0000-0002-5612-2715

[1*]Dep. «Computer Information Technologies», Dnipro National University of Railway Transport named after Academician V. Lazaryan, Lazaryana St., 2, Dnipro, Ukraine, 49010, tel. +38 (056) 373 15 35, e-mail nikolay.murkovich@gmail.com, ORCID 0000-0002-1031-8245

MODELING OF FOLDING PROCESSES ON THE BASIS OF STEP-BY-STOP MODELING

Goal. When solving practical problems that require the creation and further analysis of the model, an important criterion is the complexity of modeling. As a result, there is a problem of formalizing the modeling process and using the method of step-by-step modeling for the design of technological processes. This approach allows you to design processes and tasks in stages and includes the following stages: physical modeling, mathematical modeling, discrete computer modeling and simulation. **Method.** To solve the problem, a methodology using step-by-step modeling is used. The simulation involves 3 stages and uses the decomposition algorithm, ie considers the problem from global to detailed. At the first stage of this implementation, the necessary information is collected for the experiment. This information is presented in the form of statistics. In the second stage, further processing takes place, which is performed by checking the compliance of the input data and the process with the question of how this process should be performed. The last stage is the simulation of passages of this fragment, which is represented by a chain of transitions, obtaining statistics of time efficiency of this process, weaknesses of the process and the ability to compare results obtained during modeling and in the real process. **Results.** The technique can be used to study complex technological processes in the enterprise. The technique allows modeling of complex processes to obtain information about the time efficiency of the technological operation, finding weaknesses in it and patterns in the occurrence of random events that may affect the operation. Using this approach can be very effective in systems that require constant real-time monitoring, as this tool can be modified by adding sensor kits that will constantly send information to the system or equip an additional system that will provide ready-made information packets.

Scientific novelty. The method of step-by-step modeling of representation has been improved, which consists in the simultaneous use of physical, mathematical and simulation modeling of complex processes based on the methodology of step-by-step modeling with a set of stages of process implementation. **Practical significance.** The proposed technique is designed for step-by-step modeling of the technological process with the subsequent construction of simulation programming.

Keywords: technological process; simulation programming; technological process modeling; mathematical modeling.

REFERENCES

1. Bobrovsky V. I. Functional modeling of the work of railway stations: monograph / V. I. Bobrovsky, D. N. Kozachenko, R. V. Vernigora, V. V. Malashkin; Dnipropetro. nat. un-t railway transport them. acad. V. Lazaryan. - Dnepropetrovsk, 2015 .-- 269 p.
2. Glushkov, VM Synthesis of digital automata / VM Glushkov. - Moscow: Fizmatlit, 1962 .-- 476 p.
3. Gorbova, O. V. Formalization of technological processes in private stations on the basis of a step-by-step fashion-luvannya / Gorbova O. V. - Dnipro: Dniprov. nat. un-t railway transport them. acad. V. Lazaryan, 2016 .-- 167 p.
4. Dozortsev, VM Dynamic modeling in optimal control and automated training of operators of technological processes. Part 2. Real-time computer simulators / V. M. Dozortsev // Instruments and control systems. - 1996.– No. 8. - P.41–50.
5. Kozachenko, DN Automated formation of functional models of railway stations / DN Kozachenko, RV Vernigora, VV Malashkin // Transport systems and technologies for transportation: zb. sciences. Dnipropetr ave. nat. un-tu zalizn. transport im. acad. V. Lazaryan. - Dnipropetrovsk, 2014. - VIP. 8. - S. 65–73.
6. Kozachenko, DN Object-oriented model of functioning of railway stations / DM Kozachenko // Science and progress of transport. - 2013. - No. 4 (46). - S. 47–55.
7. Kozachenko, DM Program complex for the imitational model of robotic hall stations on the basis of an additional graphical plan / DM Kozachenko, RV Vernigora, RG Korobyova // Zalizn. transport Ukraine. - 2008. - No. 4 (70). - S. 18–20.
8. Leonenkov, A. V., UML Self-Tutorial / A. V. Leonenkov. - SPb. : BHV – Petersburg, 2002.
9. Malkov, M. V. Modeling of technological processes: methods and experience / M. V. Malkov, A. G. Oleinik, A. M. Fedorov // Proceedings of the KSC RAS, 2010. - pp. 93–101.
10. Petrosov, D. A. Phased modeling of technological processes using intelligent structural-parametric synthesis / D. A. Petrosov., V. A. Ignatenko // Fundamental research. - 2017. - No. 12 (1). - S. 97-102. doi: 10.17513 / fr.41986
11. Rozin M.D. / Problems of system modeling of complex processes of social interaction / M.D. Rozin, V.P. Svichkarov // "Engineering Bulletin of the Don" - 2012.
12. Bianco, Vieri del. A formalization of UML statecharts for real-time software modeling [Electronic resource] / Vieri del Bianco, L. Lavazza, M. Mauri. - Access mode: https://cutt.ly/QeZxM3q - Screen name. - Revised: 20.11.2019.
13. Gorbova, O. V. Modeling Work of Sorting Station Using UML / O. V. Gorbova // Science and progress of transport. - 2015. - No. 1 (55). - S. 129-138. doi 10.15802 / STP2015 / 38260
14. Harel, D. Statecharts: A visual formalism for complex systems / D. Harel // Science of Computer Programming. - North-Holland, 1987. - Vol. 8. - Iss. 3. - P. 231–274. doi: 10.1016 / 0167-6423 (87) 90035-9
15. Harel, D., Statecharts: A visual formalisms / Devid Harel // Communications of the ACM. - New York, 1988. - Vol. 31, Iss. 5, P. 514-530.
16. Silva, M. On Modeling of Hierarchical and Distributed Discrete-Event Systems / M. Silva, J.-M. Colom,
17. J. Julvez, C. Mahulea, J. H. van Schuppen, R. Su, J. Komenda, J. Raisch, S. Geist, P. Darondeau // The DISC Project Perspective. - 2007. - p. 85.
18. Zimmermann, A. Eine Quantitative Untersuchung des European Train Control System mit UML State Machines [Electronic resource] / A. Zimmermann, J. Trowitzsch. - Access mode: https://cutt.ly/HeZxV5r - Screen name. - Revised: 20.11.2019