Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Програмний проєкт аналізу стабільності енергетичних комплексів в сучасних умовах

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–23ск

_____ / В. В. Прохоренко/

Керівник кваліфікаційної роботи _____/ І. А. Котов /

Завідувач кафедри _____/ А. М. Стрюк /

Кривий Ріг

2026

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«____» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ–23ск Прохоренко Віталію Вадимовичу

1. Тема: Програмний проєкт аналізу стабільності енергетичних комплексів в сучасних умовах

затверджена наказом по КНУ № 283с від «28» травня 2026р.

2. Термін подання студентом закінченої роботи: «10» червня 2026р.

3. Вихідні дані по роботі: програмний комплекс для автоматизації аналізу стабільності енергетичних комплексів в сучасних умовах.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): проаналізувати підходи до автоматизації контролю стабільності енергетичних комплексів в сучасних умовах, проаналізувати існуючи програмні системи автоматизації контролю стабільності енергетичних комплексів в сучасних умовах, спроектувати додаток, розробити програмне забезпечення системи автоматизації контролю стабільності енергетичних комплексів в сучасних умовах, здійснити тестування розробленого застосунку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема алгоритму, зображення екранних форм додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Пошук науково-практичних джерел та інтернет-ресурсів з тематики роботи	02.02.26 – 09.02.26
2	Аналіз існуючих підходів і методів вирішення проблеми	10.02.26 – 23.02.26
3	Формулювання актуальності і постановка завдань роботи	24.02.26 – 02.03.26
4	Оформлення матеріалів першого розділу роботи	03.03.26 – 09.03.26
5	Розробка структурної і функціональної схем та основних алгоритмів програмного проєкту	10.03.26 – 23.03.26
6	Оформлення матеріалів другого розділу роботи	24.03.26 – 06.04.26
7	Розробка візуального інтерфейсу, баз даних і програмних модулів проєкту	07.04.26 – 20.04.26
8	Оформлення матеріалів третього розділу роботи	21.04.26 – 11.05.26
9	Оформлення додатків	12.05.26 – 18.05.26
10	Тестування розробленої програмного комплексу	19.05.26 – 01.06.26
11	Оформлення пояснювальної записки та супровідних документів	02.06.26 – 08.06.26

Дата видачі завдання: «02» лютого 2026 р.

Студент: _____ / В. В. Прохоренко /

Керівник роботи: _____ / І. А. Котов /

РЕФЕРАТ

АВТОМАТИЗАЦІЯ, БАЗА ДАНИХ, БЛОК-СХЕМА, ЕНЕРГЕТИКА, ІНТЕРФЕЙС, МОДЕЛЬ, НАДІЙНІСТЬ, ПРОЄКТ, УПРАВЛІННЯ, ТЕХНОЛОГІЯ, SCADA

Пояснювальна записка: 77 с., 45 рис., 1 дод., 23 джерела.

Кваліфікаційна робота: «Програмний проєкт аналізу стабільності енергетичних комплексів в сучасних умовах».

Мета випускової кваліфікаційної роботи: проєктування і розробка застосунку автоматизації процесів аналізу стабільності енергетичних комплексів в сучасних умовах.

Об’єктом дослідження і проектування є: процеси функціонування промислового обладнання енергетичних комплексів в сучасних умовах.

Теоретична частина роботи присвячена аналізу проблеми надійності промислового обладнання, яке працює в сучасних умовах нестабільності електропостачання. Сформульовано актуальність проектування системи для автоматизації аналізу стабільності енергетичних комплексів в сучасних умовах.

Практична частина кваліфікаційної роботи присвячена розробці основних програмних модулів системи автоматизації для аналізу стабільності компонентів енергетичних комплексів, розробці і наповненню бази даних показників надійності для побудови автоматизованої системи, реалізації форм візуального користувацького інтерфейсу системи автоматизації.

Аналіз розробленого програмного комплексу аналізу стабільності енергетичних комплексів в сучасних умовах показує, що отримані програмні результати можуть бути використані для аналізу роботи промислового обладнання, так і у вищих навчальних закладах при проведенні лекційних і практичних занять.

ABSTRACT

AUTOMATION, DATABASE, BLOCK DIAGRAM, POWER ENGINEERING, INTERFACE, MODEL, RELIABILITY, PROJECT, CONTROL, SCADA, TECHNOLOGY

Explanatory note: 77 p., 45 fig., 1 supplement, 23 sources.

Qualification work: «Software project for analyzing the stability of energy complexes in modern conditions».

The purpose of the final qualifying work: design and develop an application for automating the analysis of the stability of power systems in modern conditions.

The object of research and design are: the operational processes of industrial equipment in energy complexes under modern conditions.

The theoretical part of the work is devoted to the analysis the issue of reliability of industrial equipment operating under modern conditions of unstable power supply. The relevance of designing a system for automating the stability analysis of energy complexes under modern conditions is formulated.

The practical part of qualification work is devoted to the development of the main software modules of the automation system for analyzing the stability of power complex components, the development and populating of a database of reliability indicators for building the automated system, and the implementation of the visual user interface forms of the automation system.

An analysis of the developed software complex for analyzing the stability of power systems under modern conditions shows that the obtained software results can be used both for analyzing the operation of industrial equipment and in higher education institutions during lectures and practical classes.

ЗМІСТ

ВСТУП

У світі, що стає дедалі більш взаємопов’язаним, де попит на електроенергію постійно зростає, а очікування споживачів зміщуються у бік більш ефективних, стійких та персоналізованих послуг, модернізація мереж розподілу стала нагальною необхідністю. У цьому контексті кожна країна стикається з унікальним викликом. Регіон характеризується різноманітним ландшафтом, де країни перебувають на різних етапах економічного та технологічного розвитку. Однак усі вони мають спільну потребу в підвищенні стійкості, ефективності та стійкості своїх систем розподілу електроенергії. Інтеграція відновлюваних джерел енергії, зменшення технічних та нетехнічних втрат, а також підвищення якості обслуговування — це лише деякі з цілей, у досягненні яких можуть допомогти цифровізація та автоматизація [1].

Цифровізація та автоматизація (D&A) систем розподілу електроенергії є однією з найбільш трансформаційних тенденцій в енергетичній інфраструктурі на глобальному рівні. Протягом останніх десятиліть стрімкий розвиток передових технологій дозволив зробити мережі розподілу електроенергії більш ефективними, стійкими та пристосованими до зростаючих енергетичних потреб та інтеграції відновлюваних джерел енергії. Цей процес модернізації переосмислив традиційні парадигми роботи електричних мереж, зосередивши увагу на цифровій взаємодії, оптимізації в режимі реального часу та здатності адаптуватися до енергетичного середовища, що постійно змінюється [1]. Крім того, інтеграція штучного інтелекту (ШІ) та великих даних покращує управління електричними мережами в режимі реального часу, оскільки дозволяє прогнозувати попит, оптимізувати розподіл та мінімізувати перебої, що створює мультиплікативний ефект у створенні цінності для розподільчих компаній та користувачів. Нижче наведено деякі технології та рішення в галузі мереж та

обладнання, систем IT/OT, аналітичних інструментів, а також комунікаційних та інформаційних послуг [2].

На основі проведеного аналізу сформульований перелік завдань кваліфікаційної роботи: дослідження математичних моделей забезпечення стабільності компонентів енергооб'єктів; розробка структури програмної системи забезпечення стабільності енергооб'єктів; розробка функціональної моделі оцінки надійності енергооб'єктів засобами програмного комплексу [1,2]; розробка моделі бази даних автоматизованої програмної системи; розробка алгоритмів функціональних блоків автоматизованої програмної системи; проєктування елементів інтерфейсу користувача програмного комплексу; розробка основних програмних модулів системи автоматизації для аналізу стабільності компонентів енергетичних комплексів; розробка і наповнення бази даних показників надійності для побудови автоматизованої системи; реалізація форм візуального користувацького інтерфейсу системи автоматизації; розробка засобів відображення результатів оцінки стабільності компонентів енергетичних комплексів.

Розроблена автоматизована система аналізу стабільності енергетичних комплексів в сучасних умовах дає змогу підвищити ефективність використання промислового обладнання [2]. Система забезпечує можливості оперативного аналізу надійності електротехнічного обладнання та формування аналітичних звітів.

На основі отриманих результатів можна зробити висновок, що виконана робота характеризує актуальність теми, демонструє ефективність розроблених програмних модулів [3].

Результати роботи можуть бути використані для аналізу роботи промислового обладнання, так і у вищих навчальних закладах при проведенні лекційних і практичних занять.

1 АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЗАЦІЇ ЗАБЕЗПЕЧЕННЯ СТАБІЛЬНОСТІ ЕНЕРГЕТИЧНИХ КОМПЛЕКСІВ В СУЧАСНИХ УМОВАХ

1.1. Аналіз проблеми автоматизації засобів забезпечення стабільності енергетичних комплексів

Цифровізація та автоматизація систем розподілу електроенергії кардинально змінюють підходи до управління та експлуатації енергетичної інфраструктури на глобальному рівні. У світі, що стає дедалі більш взаємопов’язаним, де попит на електроенергію постійно зростає, а очікування споживачів зміщуються у бік більш ефективних, стійких та персоналізованих послуг, модернізація мереж розподілу стала нагальною необхідністю [2,3]. У цьому контексті кожна країна стикається з унікальним викликом. Регіон характеризується різноманітним ландшафтом, де країни перебувають на різних етапах економічного та технологічного розвитку. Однак усі вони мають спільну потребу в підвищенні стійкості, ефективності та стійкості своїх систем розподілу електроенергії [3]. Інтеграція відновлюваних джерел енергії, зменшення технічних та нетехнічних втрат, а також підвищення якості обслуговування — це лише деякі з цілей, у досягненні яких можуть допомогти цифровізація та автоматизація. Електрична мережа показана на рис. 1.1.

Рисунок 1.1 – Структура електроенергетичної системи

Одним із ключових аспектів є визначення основних рушійних сил цієї трансформації, з особливим акцентом на регуляторних рамках та державній політиці [3]. У багатьох випадках відсутність належного регуляторного середовища була значною перешкодою для впровадження інноваційних технологій. Тому цей документ також має на меті надати рекомендації щодо того, як уряди, регуляторні органи та підприємства можуть сприяти створенню сприятливого середовища для цифровізації та автоматизації розподілу електроенергії [3]. Крім того, наведено кілька міжнародних прикладів успіху, які можуть слугувати орієнтиром для країни. Ці приклади демонструють, як успішне впровадження цифрових технологій в інших контекстах може бути адаптоване для вирішення конкретних викликів регіону, підкреслюючи необхідність адаптації цих рішень до місцевих реалій з урахуванням таких факторів, як існуюча інфраструктура, технологічні можливості та соціально-економічні умови [3].

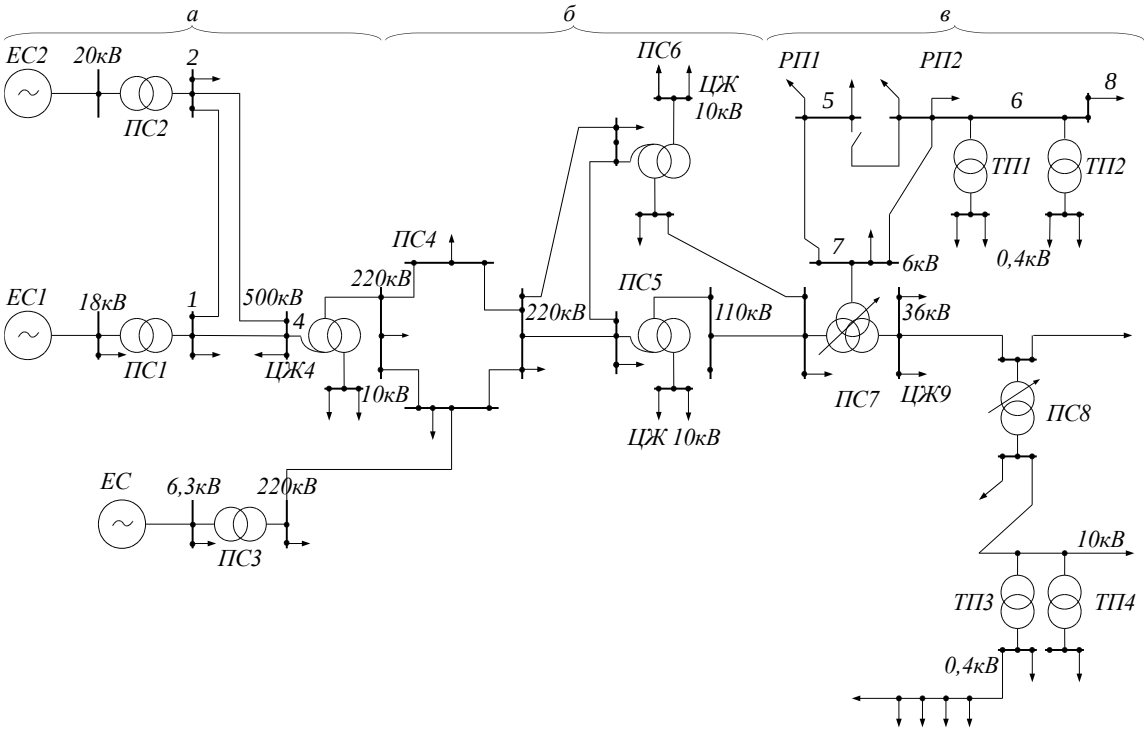


Рисунок 1.2 – Приклад схеми електричних мереж

Розглянемо глобальні тенденції, нові технології та нові рішення. Цифровізація та автоматизація (D&A) систем розподілу електроенергії є однією з найбільш трансформаційних тенденцій в енергетичній інфраструктурі на глобальному рівні [3]. Протягом останніх десятиліть стрімкий розвиток передових технологій дозволив зробити мережі розподілу електроенергії більш ефективними, стійкими та пристосованими до зростаючих енергетичних потреб та інтеграції відновлюваних джерел енергії. Цей процес модернізації переосмислив традиційні парадигми роботи електричних мереж, зосередивши увагу на цифровій взаємодії, оптимізації в режимі реального часу та здатності адаптуватися до енергетичного середовища, що постійно змінюється. Крім того, інтеграція штучного інтелекту (ШІ) та великих даних покращує управління електричними мережами в режимі реального часу, оскільки дозволяє прогнозувати попит, оптимізувати розподіл та мінімізувати перебої, що створює мультиплікативний ефект у створенні цінності для розподільчих компаній та користувачів [3,4]. Нижче наведено деякі рішення в галузі електричних мереж та обладнання, рис. 1.3.

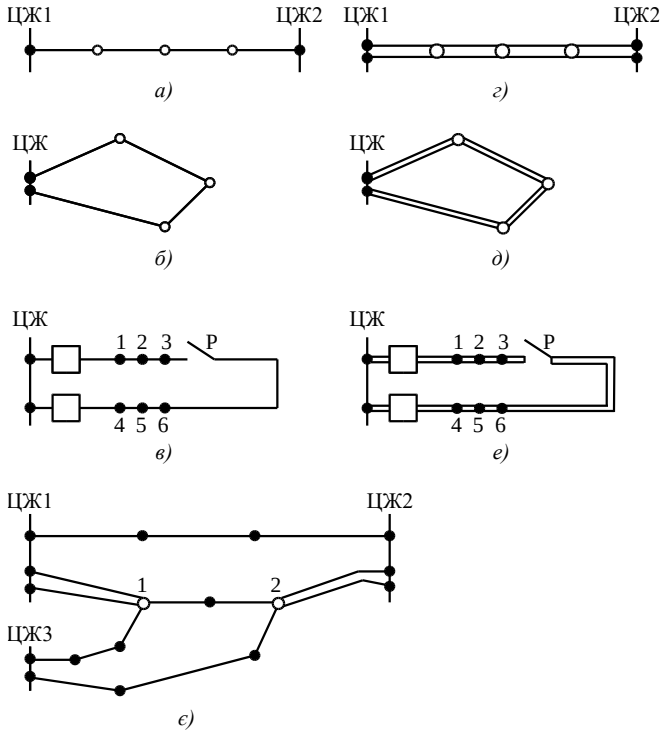


Рисунок 1.3 – Схеми розімкнутих електричних мереж

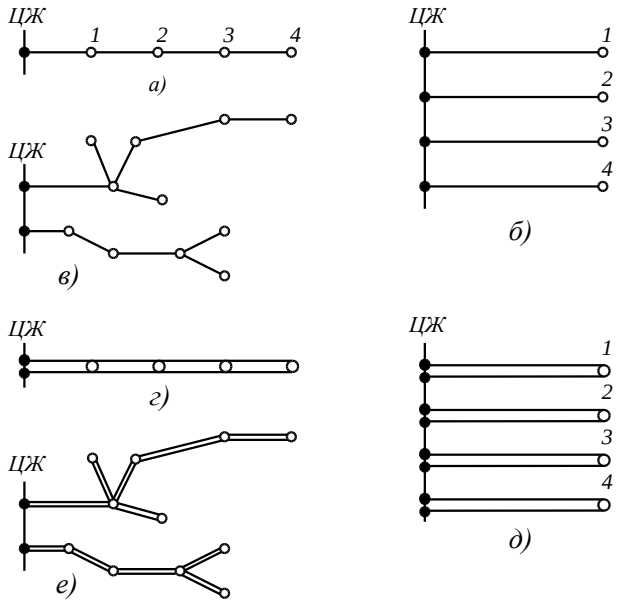


Рисунок 1.4 – Прості замкнуті і складні замкнуті електричні мережі

Одним із основних напрямків модернізації електромереж є впровадження «розумних» мереж (Smart Grids). На відміну від традиційних мереж, що працюють в односторонньому режимі, «розумні» мережі забезпечують двосторонній обмін даними між мережею та споживачами [4]. Ця трансформація відбувається завдяки впровадженню сучасних датчиків, підключених пристроїв та систем моніторингу в режимі реального часу, що надає операторам мереж можливість керувати постачанням та споживанням більш ефективно та гнучке [4].

1.2. Аналіз загальних принципів та моделей забезпечення надійності енергетичних комплексів

Системи автоматизації та управління мережами кардинально змінили підхід розподільчих компаній до управління та експлуатації своїх електричних інфраструктур, підвищивши операційну ефективність, знизивши витрати та збільшивши стійкість системи [3]. Системи IT/OT дозволяють електричним мережам функціонувати більш незалежно, ефективно та безпечно, що має

вирішальне значення в умовах зростаючого попиту на енергію та більшого проникнення відновлюваних джерел енергії [4]. Для цього було розроблено платформи, що інтегрують різноманітні функції контролю та управління для оптимізації використання енергетичних ресурсів, покращення взаємодії зі споживачами та спрощення управління активами. Платформи SCADA (системи контролю та збору даних) відіграли вирішальну роль у забезпеченні більш ефективної та безпечної роботи розподільчих мереж. Ці платформи дозволяють здійснювати моніторинг та контроль мережевого обладнання в режимі реального часу, що сприяє швидкому реагуванню на надзвичайні ситуації, поломки або коливання попиту [5]. SCADA забезпечують віддалений доступ до розподілених пристроїв, таких як вимикачі, трансформатори та інші критичні компоненти мережі, що покращує здатність реагувати на непередбачені ситуації. Ключовою перевагою цих систем є зменшення втручання людини в критичних ситуаціях. Завдяки автоматизації певних функцій, таких як виявлення несправностей та автоматичне відновлення з'єднання, час реагування мінімізується, що призводить до зменшення перебоїв та загального підвищення якості обслуговування, як показано на рис. 1.5.

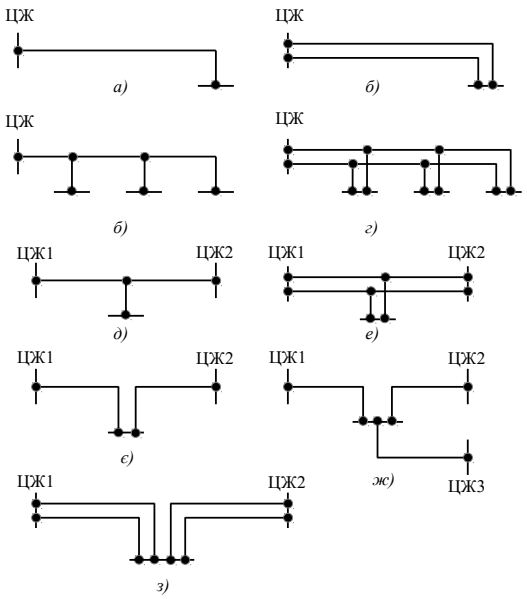


Рисунок 1.5 – Варіанти приєднання підстанцій до електричної мережі

Під терміном «система електроенергетики» (СЕЕ) або система електричної потужності позначається сукупність обладнання, що складається з електростанцій, підстанцій підвищення та пониження напруги, повітряних та підземних ліній передачі та розподілу електроенергії. Мета системи — надійне забезпечення споживачів електроенергією необхідної потужності з високими якісними характеристиками та за низькою вартістю [4].

Розглянемо в загальному вигляді проблему надійності електроенергетичних систем. Світові потреби у споживанні електроенергії за останні роки різко зросли. Щоб задовольнити постійно зростаючий попит на електроенергію, були створені складні енергетичні системи [5]. Галузь виробництва та постачання енергії, яка сформувалася для задоволення цих потреб, має багато технічних проблем, що становлять виклик для інженера. Успішна робота будь-якої системи електропостачання значною мірою залежить від здатності забезпечувати надійне та безперебійне обслуговування навантажень [3].

Згідно з Міжнародною електротехнічною комісією (International Electrotechnical Commission - IEC) надійність (IEV 692-01-14) електроенергетичної системи визначається як здатність системи адекватно задовольняти попит за певних умов експлуатації протягом певного періоду часу. Термін «надійність», є функцією двох окремих понять, а саме достатності та безпеки [4]. Під достатністю (IEV 692-01-05) розуміється здатність системи забезпечувати необхідну сумарну електричну потужність та енергію, не перевищуючи характеристик системи або меж її функціонування, при цьому напруги на шинах та частота системи мають залишатися в межах норми з урахуванням запланованих або незапланованих відключень складових системи. Безпека (IEV692-01-11) електроенергетичної системи — це її здатність витримувати подію без втрати навантаження або перевантаження елементів системи, або відхилення від встановлених меж напруги та частоти. На рис. 1.6 показана зміна інтенсивності відмов під час експлуатації [5].

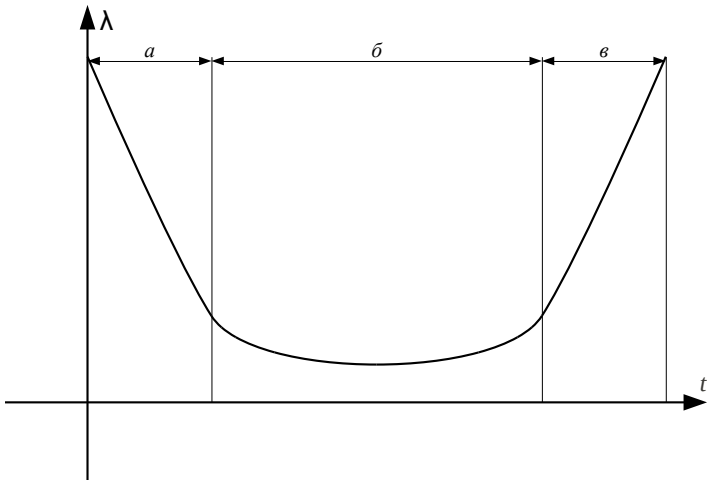


Рисунок 1.6 – Зміна інтенсивності відмов під час експлуатації

Ці рівні ієрархії також можуть використовуватися для оцінки достатності системи електропостачання. Ієрархічний рівень 1 стосується виключно об’єктів виробництва. Ієрархічний рівень 2 охоплює як об'єкти виробництва, так і об'єкти передачі, тоді як рівень 3 охоплює всі три функціональні зони [4]. Проведення досліджень рівня 3 у системах реального розміру, як правило, уникають через їхні розміри та складність. Існує можливість проведення дослідження одного рівня, в якому не будуть включені вищі рівні дослідження, щоб розглянути конкретну модифікацію системи або топологію, і зазвичай вони зосереджуються на мережі підпередачі або розподілу [5]. Ймовірність відмови і безвідмовної роботи при експоненціальному розподілу наробітку до відмови показано на рис. 1.7.

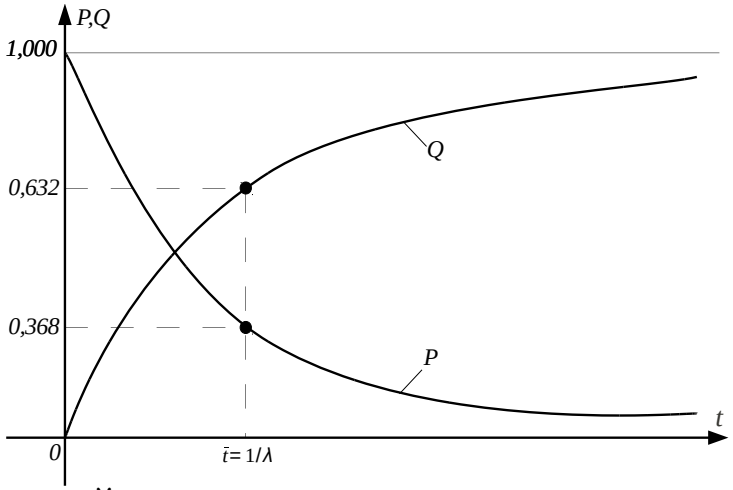


Рисунок 1.7 – Ймовірність відмови і безвідмовної роботи при експоненціальному розподілу наробітку до відмови

Розрахунок статичної стійкості ЕЕС є одним із найважливіших етапів під час проектування та експлуатації енергосистеми. Під час проектування електромережі це дослідження використовується для визначення експлуатаційної ефективності системи та встановлення необхідності змін у разі її розширення через збільшення навантажень. Дослідження статичної безпеки є діагностичним інструментом, метою якого є перевірка здатності системи протистояти можливим порушенням [4, 5]. Розробка починається з визначення збурень, які можуть виникнути в реальному часі в енергосистемі. Вибираються всі можливі сценарії, які можуть призвести до порушення меж напруги на шинах та допустимих меж навантаження на лініях.Можливими порушеннями, а отже, і сценаріями дослідження, є відмова генератора або трансформатора, або відключення лінії електропередачі [5]. Загальний огляд включає наступні критерії безпеки для визначення порушень, рис. 1.8.

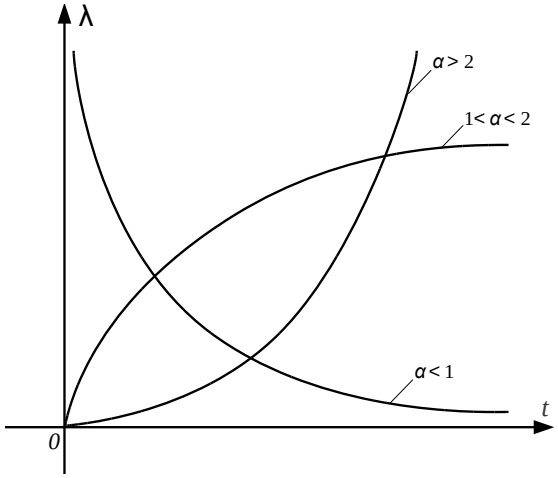


Рисунок 1.8 – Залежність інтенсивності відмов від часу за законом Вейбулла

Навіть найменша електроенергетична система є мережею з високим рівнем складності. Її структура визначається переважно її розміром, що зумовлює існування великої різноманітності мереж залежно від їхньої складності [6]. Не існує загальних правил щодо побудови системи, які були б спільними для всіх систем. Однак усі системи мають таку спільну рису щодо своєї структури. Вони працюють на різних рівнях напруги, розділених між собою трансформаторами. Загальноприйнятою практикою є поділ системи електроенергетики на структури залежно від функції, яку виконує кожна з них.

1.3. Аналіз аварійних ситуацій в енергетичних комплексах в сучасних умовах

Вимоги до вимірювання надійності енергопостачання. Схема збору даних для статистики аварій та доступності мережі зазвичай планується до запровадження, щоб забезпечити надійне виконання всіх вимог з питань енергетики щодо збору даних про надійність енергопостачання. Показники, визначені до цього часу, є системними характеристиками, що визначаються для окремих зон постачанням [5]. Однак, насамперед, промислові споживачі цікавляться не показниками надійності загальної мережі, а своєю індивідуальною ситуацією з постачанням. Щоб надати цю додаткову інформацію, саме для середньої напруги необхідна статистика аварій по окремих підстанціях [6]. Тому в майбутньому реєстрація перебоїв на рівні середньої напруги повинна здійснюватися більш детально, оскільки саме цей рівень мережі відіграє важливу роль у надійності електромережі. Такий аналіз стосується саме трансформаторної станції, з якої забезпечується об’єкт. Така інформація може бути не тільки важливою з огляду на заплановані регуляторними органами вимоги щодо стандартів надійності для різних груп споживачів, але й мати велике значення для планування мережі самої компанії. Тому, зокрема, доцільно надавати більш точний опис надійності постачання для окремих споживачів середньої напруги разом із часовим описом перебігу порушення. Реєстрація перебігу порушення в мережах середньої напруги, включаючи час, коли порушення закінчилося, слугує основним показником для розрахунків надійності обладнання [6]. Ключовим моментом тут є реєстрація всіх відключень у мережі середньої напруги стосовно кожної окремої трансформаторної станції з урахуванням обладнання, що вийшло з ладу. Для цього необхідні додаткові розширення існуючої системи реєстрації. Автоматичне прив'язування окремого елемента мережі (наприклад, повітряної лінії від станції А до станції Б), щоб забезпечити регіональну та точну за обладнанням ідентифікацію у разі скупчення порушень. Реєстрація перебоїв у

постачанні з урахуванням конкретної станції, при цьому також зберігається інформація про перервану потужність (максимальна потужність станції або номінальна потужність трансформатора), щоб разом із тривалістю перебоїв можна було оцінити також енергію, яка не була поставлена вчасно [6]. Приклад аварійної ситуації приведений на рис. 1.9.

Рисунок 1.9 – Електрична дуга на електропідстанції.

Обладнання слід експлуатувати та обслуговувати належним чином, завжди дотримуючись інструкцій, наданих монтажниками та виробниками. Зокрема, необхідно дотримуватися їхньої навантажувальної здатності, ні в якому разі не перевищуючи її; дбати про їхню чистоту та санітарний стан (особливо це стосується установок з відкритими повітряними лініями, що проходять поблизу лісових масивів); а також періодично перевіряти роботу систем захисту [6]. Електротехнічні норми встановлюють вимоги до технічного обслуговування, перевірок та інспекцій, яким повинні підлягати електричні установки як під час їх монтажу, так і протягом терміну їх експлуатації. Однак стосовно інспекцій та перевірок слід враховувати можливу наявність додаткових положень на рівні автономних спільнот та в галузевих сферах, таких як гірнича промисловість або транспорт [7]. Пожежа на енергооб’єкті, рис. 1.10.

Рисунок 1.10 – Пожежа на енергооб’єкті

Пожежа, що виникла внаслідок іскри в електричному трансформаторі (рис. 1.10), призвела до загоряння на заводі з виробництва палива. Вогонь поширився по рідкому паливі, хоча його причиною була інша причина [7].

Щодо використання, контролю та технічного обслуговування низьковольтних електричних установок, відповідно до статті 19 REBT, монтажна компанія повинна надати власнику будь-якої електричної установки (у вигляді додатка до сертифіката про монтаж) інструкції щодо правильного використання та технічного обслуговування такої установки. Крім того, стаття 20 зазначеного регламенту встановлює, що власники установок повинні підтримувати їх у належному робочому стані, використовуючи їх відповідно до їхніх характеристик та утримуючись від втручання в них з метою їх модифікації [6]. Якщо модифікації необхідні, їх має здійснювати монтажна компанія. Щодо використання, контролю та технічного обслуговування електричного обладнання, крім дотримання зазначених електротехнічних правил, необхідно дотримуватися вимог, якими встановлюються мінімальні вимоги безпеки та охорони здоров’я щодо використання працівниками робочого обладнання [7]. На рис. 1.11 представлена ілюстрація небезпеки з боку електротехнічного обладнання, ознаки небезпеки, що стосуються електричних установок.

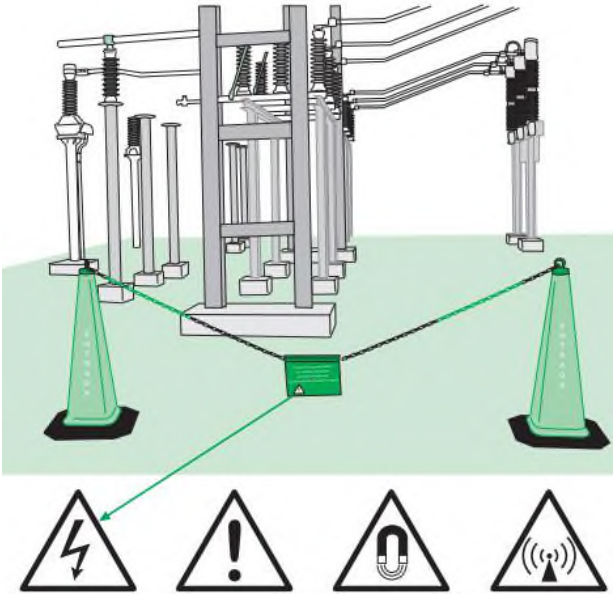


Рисунок 1.11 – Ознаки небезпеки, що стосуються електричних установок

Рішення про проведення робіт під напругою не може прийматися довільно; воно має ґрунтуватися на потребах, зумовлених умовами експлуатації об’єкта або необхідністю забезпечення безперебійного електропостачання [7]. Такі потреби можуть мати різні причини; наприклад: через вимоги щодо дотримання нормативно-правових актів, що регулюють вимоги до якості та безперебійності електропостачання або коли відключення електропостачання створює ризики для безпеки чи здоров'я населення або його певної групи.

1.4. Порівняльний аналіз існуючих автоматизованих систем забезпечення надійності енергетичних комплексів

У всьому світі зростає зацікавленість підприємств в автоматизації своїх виробничих систем з метою підвищення ефективності, що має забезпечити надійне та недороге обслуговування для кінцевих споживачів [6]. Підприємства, що надають такі послуги, як-от виробники електроенергії, повинні докладати більше зусиль для підвищення ефективності своєї діяльності, що гарантуватиме належне функціонування систем управління. Для цього значна частина підприємств енергетичного сектору базує свою

діяльність на програмних системах типу SCADA (Supervisory Control And Data Acquisition) — системах управління, нагляду та збору даних, що дозволяють збирати інформацію з польових пристроїв, завдяки чому можна дистанційно керувати та контролювати процеси, рис. 1.12 [7].

Рисунок 1.12 – Диспетчерський пульт як складова SCADA

Системи управління SCADA. Системи управління SCADA складаються з низки компонентів, які забезпечують комплексне управління складними промисловими процесами, зокрема, у контексті цієї статті — на підприємствах з виробництва електроенергії. Серед різноманітного обладнання, що входить до складу такої системи, можна виділити наступне [7]. Апаратне забезпечення зовнішньої станції: віддалені пристрої управління підстанціями, що контролюють такі параметри, як стан навантаження, трансформатор струму (CT), трансформатор напруги (VT), паливні клапани та вимикачі, якими можна керувати локально або дистанційно. Процесори місцевих підстанцій: збирають дані з польових приладів та апаратного обладнання, включаючи RTU, PLC та інтелектуальні електронні пристрої (IED), зокрема цифрові реле та цифрові лічильники [8]. Місцевий процесор відповідає за велику кількість аналогових та цифрових входів/виходів IED та комутаційного обладнання. Цифрові прилади: зазвичай встановлюються на місці або в локальному приміщенні та вимірюють такі параметри, як струм, напруга, інтенсивність сонячного

випромінювання, температура, тиск, швидкість вітру та витрата. Комунікаційні пристрої: можуть бути коротко- або далекодіючими. Короткодіючі комунікації встановлюються між локальними RTU (віддаленими термінальними пристроями), приладами та оперативним обладнанням.

Диспетчерська керуюча система [7]. Однак самого лише забезпечення роботи окремих блоків на електростанціях за допомогою систем SCADA недостатньо для задоволення вимог сучасного ринку, оскільки слід враховувати, що більшість електростанцій та розподільчих систем працюють у мережі й повинні спільно реагувати на сукупний попит [7]. Отже, централізація та комплексний контроль стають важливим фактором для функціонування підприємств комунальних послуг та систем електроенергетики. Для диспетчерських центрів на енергогенеруючих підприємствах оцінка стану потужності та роботи системи становить основу функцій моніторингу, аналізу та управління системою в режимі реального часу. Ця інформація діє як фільтр між необробленими вимірюваннями, отриманими від системи, та іншими вимірюваннями експлуатаційних функцій, які вимагають збору даних з вищою надійністю, необхідною для визначення поточного стану роботи системи, і в цілому представляє виклик з точки зору обробки даних, оцінки стану, оцінки похибок параметрів та топології обробки, серед інших аналізів (там само) [8].

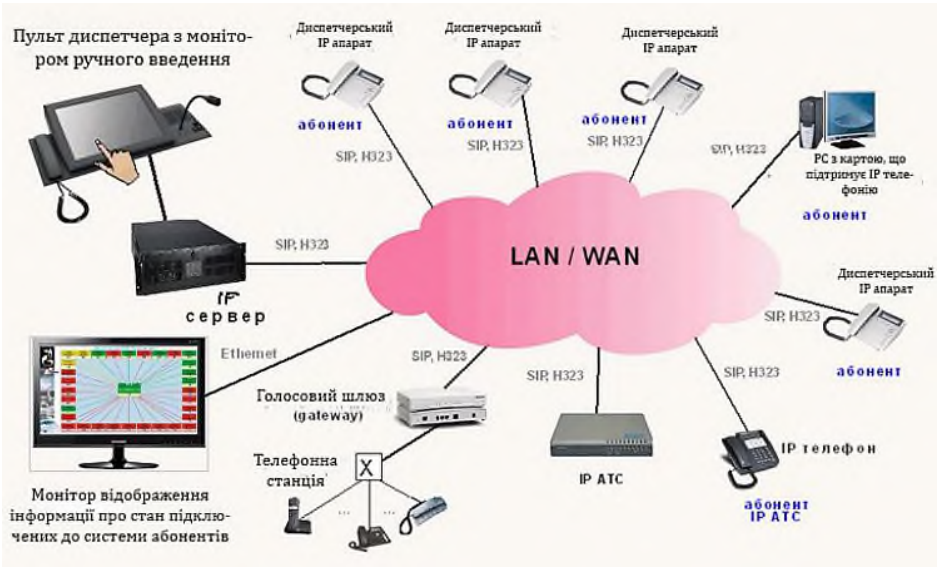


Рисунок 1.13 – Загальна структура системи SCADA

Робота має на меті ознайомитись з структурою системи SCADA, яка буде застосована до центру управління, що дозволяє підприємствам електроенергетичного сектору мати більший контроль над своїми активами та їхньою роботою, що призведе до підвищення ефективності як їхньої діяльності, так і наданих послуг [7, 9]. Система SCADA промислового підприємства показана на рис. 1.14.

Рисунок 1.14 – Система SCADA промислового підприємства

Ілюстрація SCADA-системи Measurement & Automation Explorer, рис. 1.15.

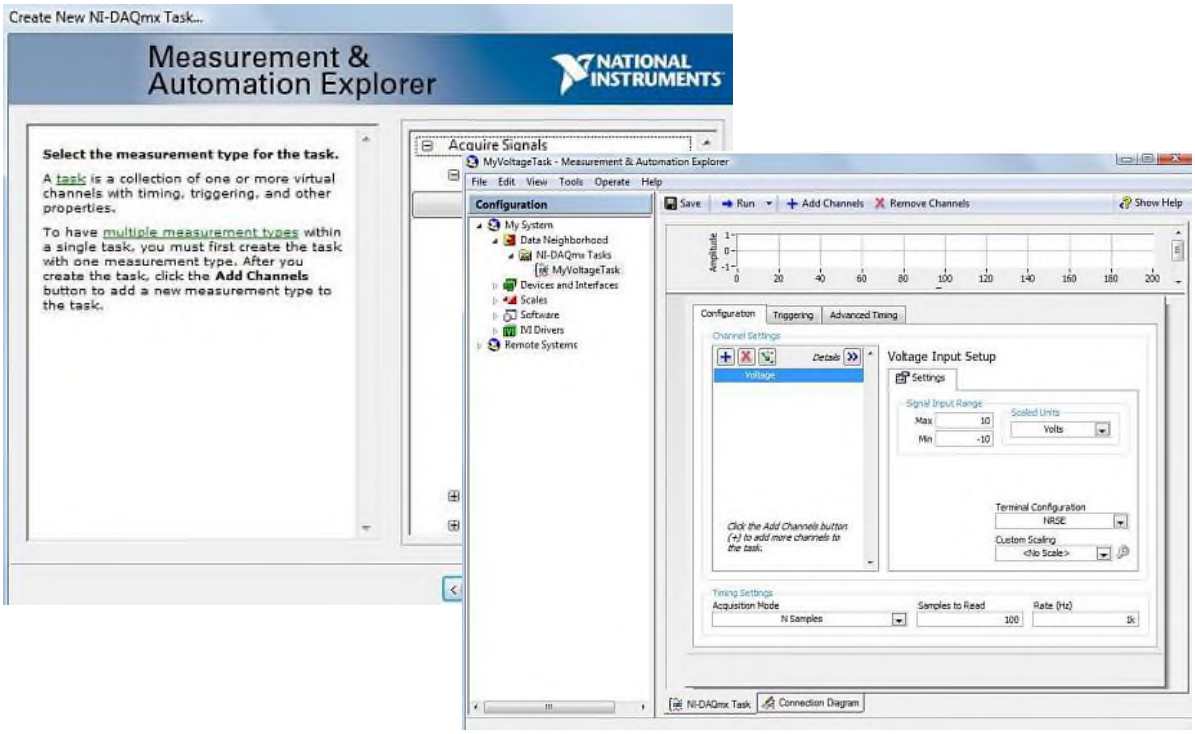


Рисунок 1.15 – SCADA-система Measurement & Automation Explorer

Збір даних (data acquisition, DAQ) за допомогою карт DAQ та вимірювальних приладів, підключених до комп'ютера через основні протоколи зв'язку (RS232, GPIB, Ethernet) [7,8]. Крім того, включає в себе вступ до основних елементів обробки сигналів (аналіз FFT, фільтри, вікна згладжування), до технік моніторингу та дистанційного через веб-браузер, а також у завантаженні та обміні даними між віддаленими модулями, підключеними через мережу TCP [8,9]. Мовою програмування додатків, яка використовується протягом усього курсу, є графічна мова програмування LabVIEW. Система SCADA Trace Mode показана на рис. 1.16.

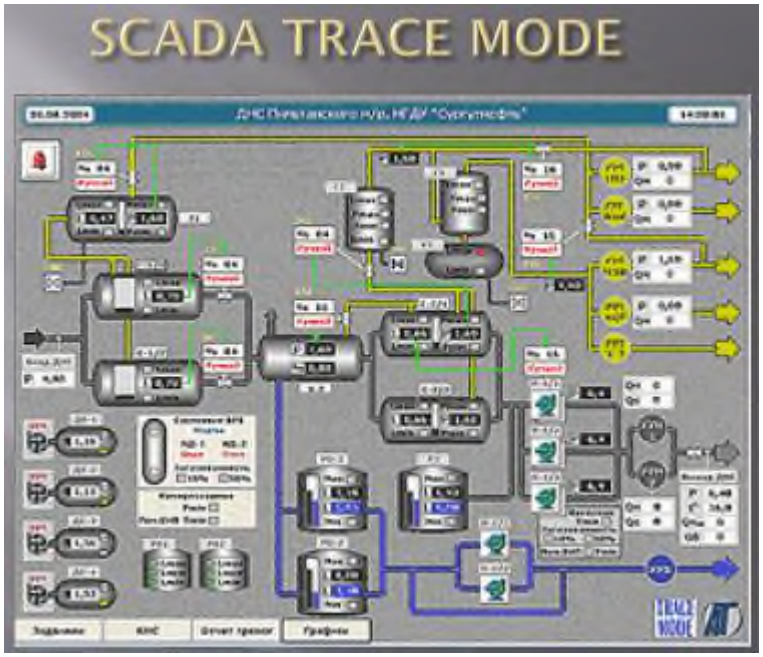


Рисунок 1.16 – Система SCADA Trace Mode

1.5. Постановка комплексу завдань для розробки автоматизованого програмного застосунку контролю стабільності енергетичних комплексів

У світі, що стає дедалі більш взаємопов'язаним, де попит на електроенергію постійно зростає, а очікування споживачів зміщуються у бік більш ефективних, стійких та персоналізованих послуг, модернізація мереж розподілу стала нагальною необхідністю [9]. У цьому контексті будь-яка країна стикається з унікальним викликом. Наш регіон характеризується

різноманітним ландшафтом, де країни перебувають на різних етапах економічного та технологічного розвитку [8]. Однак усі вони мають спільну потребу в підвищенні стійкості, ефективності та стійкості своїх систем розподілу електроенергії. Інтеграція відновлюваних джерел енергії, зменшення технічних та нетехнічних втрат, а також підвищення якості обслуговування — це лише деякі з цілей, у досягненні яких можуть допомогти цифровізація та автоматизація [9].

Цифровізація та автоматизація (D&A) систем розподілу електроенергії є однією з найбільш трансформаційних тенденцій в енергетичній інфраструктурі на глобальному рівні. Протягом останніх десятиліть стрімкий розвиток передових технологій дозволив зробити мережі розподілу електроенергії більш ефективними, стійкими та пристосованими до зростаючих енергетичних потреб та інтеграції відновлюваних джерел енергії [9]. Цей процес модернізації переосмислив традиційні парадигми роботи електричних мереж, зосередивши увагу на цифровій взаємодії, оптимізації в режимі реального часу та здатності адаптуватися до енергетичного середовища, що постійно змінюється [9]. Крім того, інтеграція штучного інтелекту (ШІ) та великих даних покращує управління електричними мережами в режимі реального часу, оскільки дозволяє прогнозувати попит, оптимізувати розподіл та мінімізувати перебої, що створює мультиплікативний ефект у створенні цінності для розподільчих компаній та користувачів.

Підприємства, що надають такі послуги, як-от виробники електроенергії, повинні докладати більше зусиль для підвищення ефективності своєї діяльності, що гарантуватиме належне функціонування систем управління [8]. Для цього значна частина підприємств енергетичного сектору базує свою діяльність на програмних системах типу SCADA (Supervisory Control And Data Acquisition) — системах управління, нагляду та збору даних, що дозволяють збирати інформацію з польових пристроїв, завдяки чому можна дистанційно керувати та контролювати процеси.

Однак самого лише забезпечення роботи окремих блоків на електростанціях за допомогою систем SCADA недостатньо для задоволення вимог сучасного ринку, оскільки слід враховувати, що більшість електростанцій та розподільчих систем працюють у мережі й повинні спільно реагувати на сукупний попит [9]. Отже, централізація та комплексний контроль стають важливим фактором для функціонування підприємств комунальних послуг та систем електроенергетики. Для диспетчерських центрів на енергогенеруючих підприємствах оцінка стану потужності та роботи системи становить основу функцій моніторингу, аналізу та управління системою в режимі реального часу [9]. Ця інформація діє як фільтр між необробленими вимірюваннями, отриманими від системи, та іншими вимірюваннями експлуатаційних функцій, які вимагають збору даних з вищою надійністю, необхідною для визначення поточного стану роботи системи, і в цілому представляє виклик з точки зору обробки даних, оцінки стану, оцінки похибок параметрів та топології обробки, серед інших аналізів.

На основі аналізу поставлено такі завдання:

– дослідження математичних моделей забезпечення стабільності компонентів енергооб’єктів;

– розробка структури програмної системи забезпечення стабільності енергооб’єктів;

– розробка функціональної моделі оцінки надійності енергооб’єктів засобами програмного комплексу;

– розробка моделі бази даних автоматизованої програмної системи;

– розробка алгоритмів функціональних блоків автоматизованої програмної системи;

– проєктування елементів інтерфейсу користувача програмного комплексу;

– розробка основних програмних модулів системи автоматизації для аналізу стабільності компонентів енергетичних комплексів;

– розробка і наповнення бази даних показників надійності для побудови автоматизованої системи;

– реалізація форм візуального користувацького інтерфейсу системи автоматизації;

– розробка засобів відображення результатів оцінки стабільності компонентів енергетичних комплексів.

2 ПРОЄКТУВАННЯ МАТЕМАТИЧНИХ ТА СТРУКТУРНИХ МОДЕЛЕЙ ПРОГРАМНОГО КОМПЛЕКСУ ЗАБЕЗПЕЧЕННЯ СТАБІЛЬНОСТІ ЕНЕРГООБ'ЄКТІВ

2.1. Дослідження математичних моделей забезпечення стабільності компонентів енергооб'єктів

Теорія надійності складається з сукупності ідей, моделей та методів, що мають на меті вирішення проблем, пов'язаних з обчисленням, оцінкою, оптимізацією ймовірності функціонування або очікуваного терміну служби, або, загалом, розподілу тривалості терміну служби одиниці або системи одиниць. Термін «надійність» стосується здатності механізму (одиниці або системи) функціонувати без відмов протягом певного часу [10]. Функціонування (без відмов) одиниці — це збереження її характеристик у визначених межах та за заданих умов. Збій одиниці або системи — це подія, після настання якої певні характеристики одиниці перевищують допустимі заздалегідь встановлені межі [9,10]. Одиниці або системи поділяються на:

– невідновлювані (якщо після однієї відмови вони виходять з ладу і не можуть знову працювати);

– відновлювані (якщо після однієї відмови вони можуть знову працювати, наприклад, після ремонту або заміни).

Деякі приклади відмов «одиниць» або «систем одиниць»: припинення роботи («перегорання») електричної лампи, монітора або, загалом, електричного приладу; поломка поршня в двигуні, аварія на атомній електростанції, припинення виробничого процесу (наприклад, на заводі) через поломку машини [10].

Ось декілька прикладів систем. Послідовна система – Serial System (SS) [10]. Складається з n одиниць і виходить з ладу, коли хоча б одна з n одиниць виходить з ладу, або, що еквівалентно, працює, коли всі n одиниці працюють. Цю систему можна зобразити графічно наступним чином, рис. 2.1.

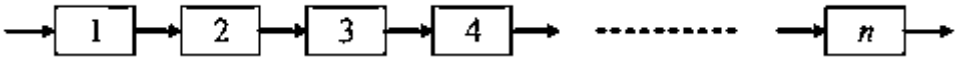


Рисунок 2.1 – Приклад послідовної системи

Тобто вважається, що для того, щоб «сигнал» пройшов зліва направо, він повинен пройти через усі блоки [10]. Отже, «сигнал» пройде (тобто система працює, $\varphi = 1$), коли всі її елементи працюють ($x1 = x2 = \ldots = xn = 1$). Якщо якесь із $x_i = 0$, то $\varphi = 0$. Отже, функція структури послідовної системи може бути записана так:

$$\varphi(x) = \min\{x_1, x_2, \ldots, x_n\} = \prod_{i=1}^{n} x_i. \qquad (2.1)$$

Паралельна система – Parallel system (PS) складається з n модулів і виходить з ладу, коли всі її модулі виходять з ладу, або, що еквівалентно, працює, коли принаймні один її модуль працює [10]. Цю систему можна зобразити графічно так:

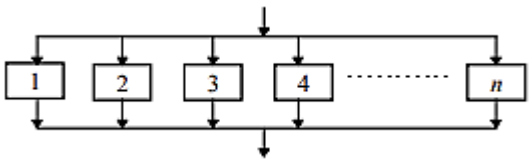


Рисунок 2.2 – Приклад паралельної системи

«Сигнал» може пройти зверху вниз, якщо хоча б один із модулів працює, а отже, $\varphi = 1$, якщо хоча б одне x_i дорівнює 1. Отже, функція структури паралельної системи може бути записана так:

$$\varphi(x) = \max\{x_1, x_2, \ldots, x_n\} = 1 - \prod_{i=1}^{n}(1 - x_i). \qquad (2.2)$$

Тепер розглянемо структурні моделі надійності компонентів енергооб'єктів [10]. Для цього розглянемо схеми заміщення найпростіших з них — ліній електропередачі і трансформаторів.

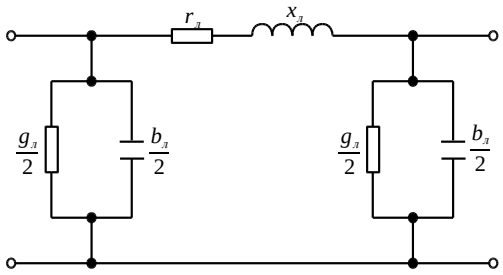


Рисунок 2.3 – П-образна схема заміщення повітряної лінії електропередачі

Схема заміщення двообмоткового трансформатора показана на рис. 2.4

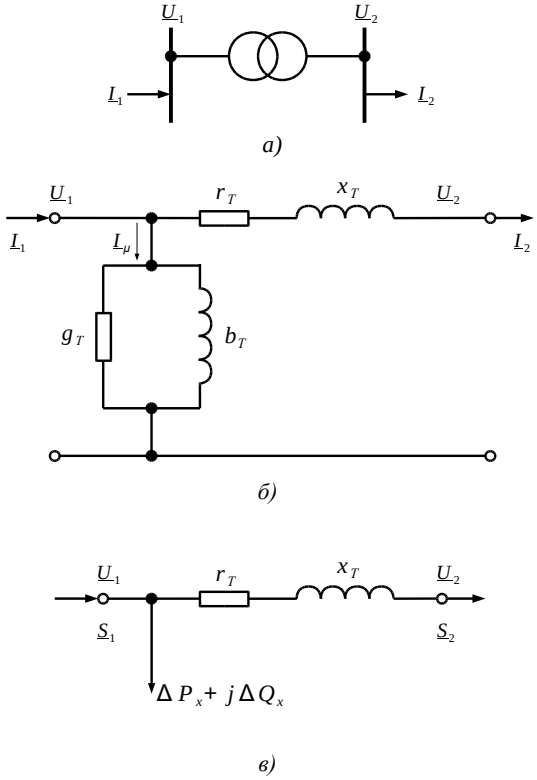


Рис. 2.4. Трансформатор з двома обмотками
а – умовне позначення трансформатору [11]; б – Г-образна схема заміщення; в – спрощена схема заміщення

Виходячи з наведених схем заміщення, можна побудувати моделі надійності [11]. Структурна схема надійності нерезервованої системи показана на рис. 2.5.

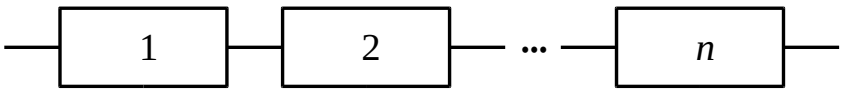


Рисунок 2.5 – Структурна схема нерезервованої системи

Нижче, на рис. 2.6, показано структурні схеми надійності резервованих систем.

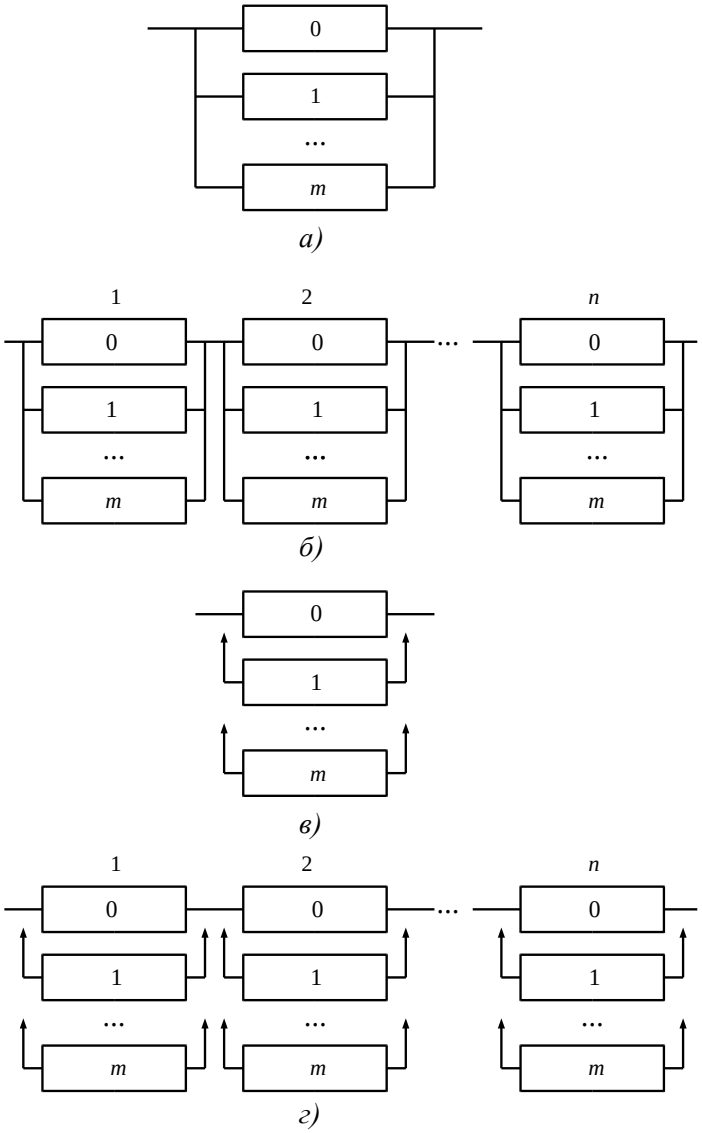


Рисунок 2.6 – Структурні схеми резервованих систем
а – загальне резервування з постійно включеним резервом [11]; б – окреме резервування з постійно включеним резервом; в – загальне резервування заміщенням; г – окреме резервування заміщенням

Розглянемо нормальну модель надійності. Нехай t – час життя технічного об’єкту [12]. Якщо припустити, що ця змінна підпорядковується нормальному розподілу з середнім значенням μ та стандартним відхиленням σ, то функція надійності матиме вигляд

$$R(t) = 1 - F(t) = 1 - \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi} \cdot \sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = 1 - \varphi\left(\frac{t-\mu}{\sigma}\right). \qquad (2.3)$$

Імовірність відмови для цього розподілу становить

$$\lambda(t) = \frac{f(t)}{R(t)}. \quad (2.4)$$

Як уже зазначалося, вираз функції щільності, коли термін служби пристрою підпорядковується експоненційному розподілу, має вигляд

$$f(t) = \lambda e^{-\lambda t},$$

де λ – додатна константа (λ > 0).

Оскільки функція розподілу, тобто її ненадійність, буде [12]

$$F(t) = 1 - e^{-\lambda t}, t > 0,$$

функція надійності, ймовірність роботи протягом часу *t*, дорівнює

$$R(t) = 1 - F(t) = e^{-\lambda t}.$$

Цікавим є значення, яке отримуємо [12], якщо $t = 1/\lambda$, тобто коли тривалість часу збігається із середнім терміном служби, у цьому випадку $R(1/\lambda)=0,37$.

2.2. Розробка структури програмної системи забезпечення стабільності енергооб’єктів

Базову архітектуру системи управління та моніторингу – SCADA для електрогенеруючої компанії показано на рис. 2.7. В системі SCADA передбачені диспетчерські станції, здатні дистанційно та бездротово передавати дані про такі параметри, як струм, напруга, показники датчиків тиску, стан насосів та клапанів, які підключаються через маршрутизатор до загальної мережі SCADA, яка разом із серверним обладнанням отримує та обробляє інформацію, централізовано передаючи її на інженерні комп'ютери та робочі станції операторів [11].

У більшості випадків система SCADA здійснює моніторинг та вносить незначні зміни в роботу обладнання, щоб підвищити його ефективність та продуктивність [12]. Системи SCADA вважаються системами управління із замкнутим контуром і функціонують із мінімальним втручанням людини.

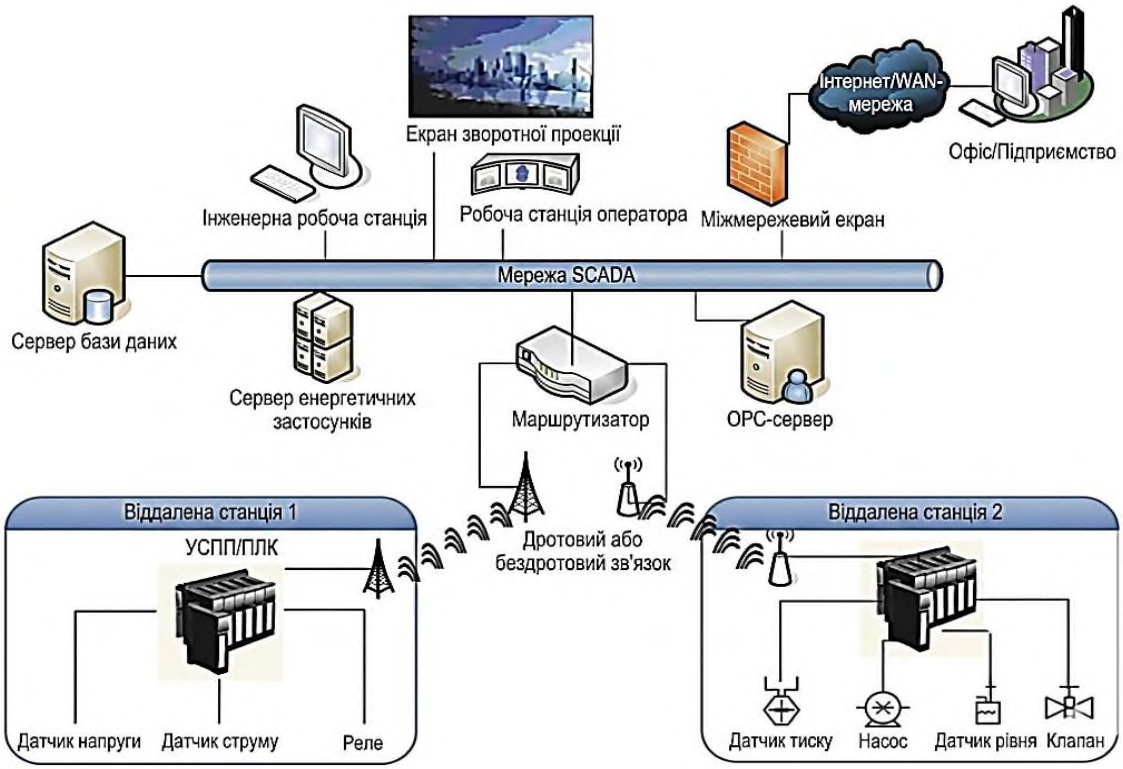


Рисунок 2.7 – Загальна схема архітектури системи SCADA

Однією з ключових переваг систем SCADA є їхня здатність контролювати всю систему або сукупність систем у режимі реального часу [12]. Це здійснюється шляхом збору даних, включаючи зчитування показань лічильників та перевірку стану датчиків, які передають інформацію через регулярні короткі проміжки часу [11]. Система SCADA як система промислової автоматизації отримує дані від приладів та датчиків, розташованих у віддалених місцях, і передає їх до центрального головного блоку управління з метою контролю або моніторингу.

Дистанційні пристрої управління та зв'язку використовуються для забезпечення оптимального рішення для кожної операції завдяки гнучким та сучасним структурам управління та функціонування [12]. Це досягається за допомогою використання ПЛК-контролерів, пристроїв збору даних RTU та сучасних каналів зв'язку у поєднанні з програмним та апаратним забезпеченням SCADA на електростанціях та енергогенеруючих об'єктах. Структура SCADA в енергетичній галузі контролює різні операційні завдання,

включаючи функції захисту, управління та моніторингу планових і позапланових процедур технічного обслуговування [12]. Функції управління системою SCADA в енергетиці включають:

– постійний моніторинг швидкості та частоти;

– спостереження за динамічним станом вимикачів, перемикачів та реле захисту;

– планування операцій з виробництва електроенергії;

– контроль активної та реактивної потужностей;

– захист від вітру / пари / газової турбіни;

– моніторинг паливної системи та допоміжних служб станції;

– програмування навантаження на основі напруги та частоти;

– обробка історичних даних для параметрів, пов'язаних з виробництвом;

– спостереження за метеостанціями у випадку вітрових та сонячних електростанцій.

Функціональна блокова структура системи SCADA приведена на рис. 2.8 [12].

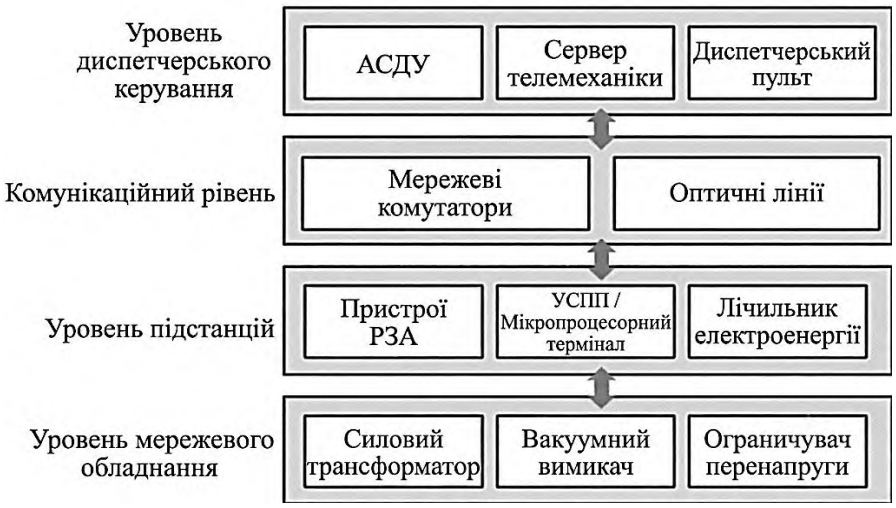


Рисунок 2.8 – Функціональна блокова структура системи SCADA

Централізовані системи SCADA мають широке коло застосування. Воно охоплює як переваги централізованого управління та налаштування безпеки, так і стандартизацію операційної діяльності та прийняття рішень у великому масштабі [13]. Управління тривогами та сигналізацією подій має велике

значення, тому створення централізованої архітектури дозволяє управляти тривогами в підсистемах з точки зору підприємства в частині визначення пріоритетів, розподілу відповідальності, узгодженості та аналізу даних. Ключові галузеві стандарти надають важливі орієнтири для встановлення стандартизованої політики щодо подій, які можуть застосовуватися на рівні всієї мережі [13].

Необхідно забезпечити безпечне кодування, щоб мінімізувати вразливості в додатках та сервісах, які пропонує система SCADA. Вразливості в програмному забезпеченні можуть бути використані в зловмисних цілях, що змушує системних адміністраторів уникати внесення змін після початкового налаштування [12]. Перевірки програмного забезпечення та дослідження з реверс-інжинірингу показують, що програмне забезпечення SCADA не завжди має безпечну структуру. Для оцінки вразливості систем управління розподілом енергії, включаючи мережі SCADA, розроблено програму National SCADA TestBed. Встановлено, що вразливості програмного забезпечення SCADA, виявлені під час оцінок, є результатом незахищеного програмного забезпечення та недостатнього тестування [13]. Три найважливіші виявлені вразливості: перевірка вхідних даних, автентифікація та контроль доступу. Також було встановлено, що віддалене виконання коду спричиняє небезпечні функції у більшості вразливостей. Вищезазначений сценарій показує, що необхідна постійна оцінка програмного коду та постійне оновлення.

2.3. Розробка функціональної моделі оцінки надійності енергооб’єктів засобами програмного комплексу

Модель процесу розробки програмного забезпечення — це спрощене зображення цього процесу. Кожна модель процесу відображає його з певної точки зору і надає лише часткову інформацію про цей процес [13]. Наприклад, модель активності процесу показує дії та їхню послідовність, але, можливо, не відображає ролі осіб, які беруть участь у цих діях. Розглянемо деякі дуже

загальні моделі процесів («парадигми процесів») з архітектурної точки зору. Іншими словами, розглядається framework процесу, але не деталі конкретних дій. Такі загальні моделі не є остаточними описами процесів розробки програмного забезпечення [11]. Скоріше, це абстракції процесу, які використовуються для пояснення різних підходів до розробки програмного забезпечення. Їх можна розглядати як фреймворки процесу, які розширюються та адаптуються для створення більш конкретних процесів програмної інженерії. Виходячи з цих міркувань, можна розробити загальну функціональну модель програмної системи для оцінки надійності енергооб'єктів, яка показана на рис. 2.9.

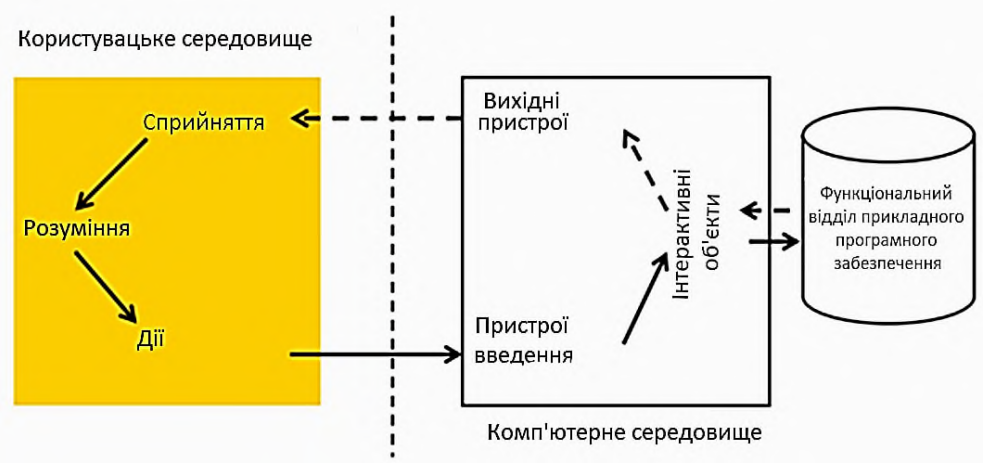


Рисунок 2.9 – Загальна функціональна модель програмної системи для оцінки надійності енергооб'єктів

Ця функціональна схема може розглядатися як прототип для багатьох систем такого типу [12]. Прототипи системи дають користувачам змогу оцінити, наскільки добре система сприяє їхній роботі. Вони можуть отримати нові ідеї щодо вимог та виявити сильні й слабкі сторони програмного забезпечення. Після цього вони пропонують нові системні вимоги. Більше того, у міру розробки прототипу можуть виявитися помилки та прогалини у запропонованих вимогах [13]. Функція, описана у специфікації, може здаватися корисною та чітко визначеною. Однак, коли така функція поєднується з іншими операціями, користувачі часто виявляють, що їхнє початкове бачення було неправильним або неповним. Тоді специфікацію

системи змінюють, щоб відобразити нове розуміння вимог [13]. Під час розробки системи для проведення експериментів з проектування прототип системи слугує для перевірки здійсненності запропонованого проекту. Наприклад, можна створити прототип проекту бази даних і протестувати його з метою перевірки, чи ефективно він підтримує доступ до даних для найпоширеніших запитів користувачів [13]. Крім того, створення прототипів є важливою частиною процесу проектування користувацького інтерфейсу. Через природну динаміку користувацьких інтерфейсів текстові описи та діаграми недостатні для вираження вимог до користувацького інтерфейсу. Тому швидке створення прототипів за участю кінцевого користувача є єдиним раціональним способом розробки графічних користувацьких інтерфейсів для програмних систем [14]. Інтерактивні функції системи взаємодії з оператором показано на ілюстрації рис. 2.10.

Рисунок 2.10 – Модель комплексу інтерактивних функцій системи автоматизації

Наступний етап процесу полягає у вирішенні, що включити, а що — можливо, ще важливіше — виключити з прототипу [12]. Щоб зменшити витрати на створення прототипів та прискорити терміни реалізації, можна виключити з прототипу певну функціональність, а також зробити

нефункціональні вимоги, такі як час відгуку та використання пам’яті, більш гнучкими. Обробку та управління помилками можна проігнорувати, якщо метою прототипу не є створення користувацького інтерфейсу. Крім того, можна знизити стандарти надійності та якості програми. Останнім етапом процесу є оцінка прототипу [14]. На цьому етапі слід подбати про навчання користувачів та використати цілі прототипу для розробки плану оцінки. Користувачам потрібен час, щоб звикнути до нової системи та перейти до звичайного режиму використання. Коли вони починають користуватися системою у звичайному режимі, вони виявляють помилки та прогалини у вимогах [14]. Загальна схема розробки інтерактивної системи приведена на рис. 2.11.

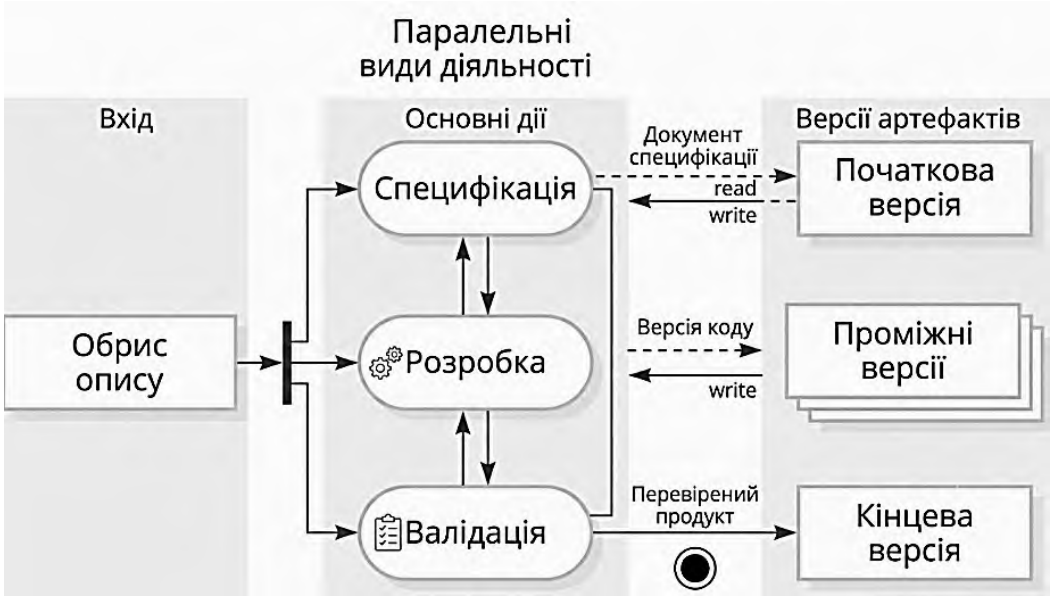


Рисунок 2.11 – Загальна схема розробки інтерактивної системи

Проблема при створенні прототипів полягає в тому, що прототип не обов’язково використовуватиметься так само, як і кінцева система [13]. Особа, яка перевіряє прототип, може не бути типовим користувачем системи. Крім того, час на навчання під час оцінки прототипу може виявитися недостатнім. Якщо прототип працює повільно, оцінювачі можуть змінити свій стиль роботи та уникати тих функцій системи, які мають повільний час відгуку. Коли кінцева система реагує краще, її можна використовувати по-іншому.

Отже, після формулювання схеми розробки інтерактивної програмної системи можна сформувати функціональну схему застосунку оцінки надійності електромережі, яка показана на рис. 2.12.

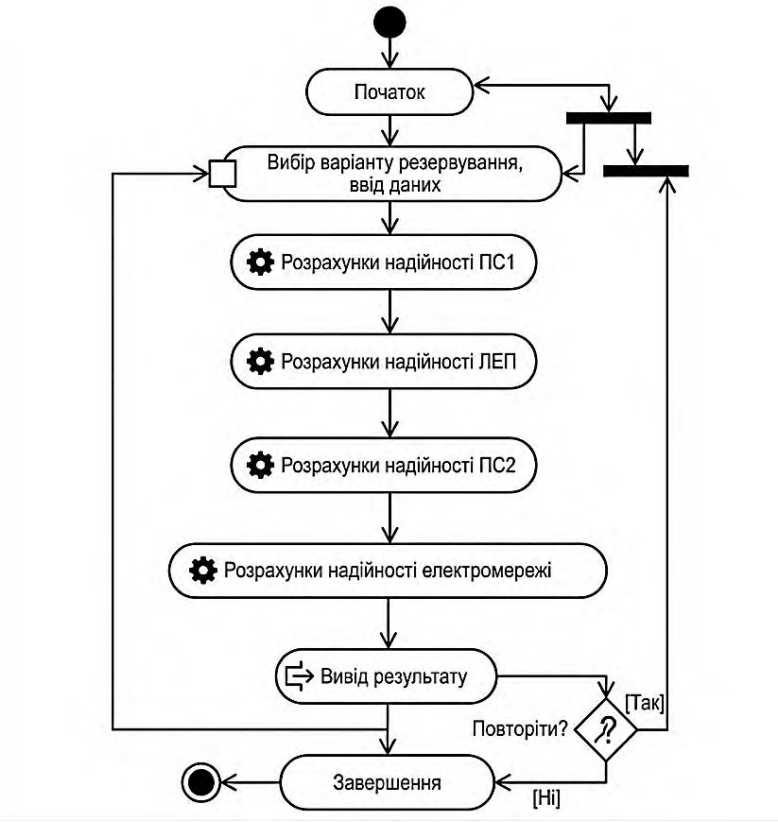


Рисунок 2.12 – Структурна функціональна схема застосунку для оцінки надійності електромережі

Наразі підприємства, – ті, що пов’язані з електроенергетичним сектором, докладають значних зусиль для створення систем централізованого контролю та моніторингу, які виконують низку функцій з управління системою та збалансування попиту й пропозиції, що забезпечує стабільне енергопостачання та злагоджену роботу системи в цілому [14]. Водночас вони мають низку функцій управління, що застосовуються для аналізу системи, та функцій моніторингу безпеки напруги, які надають необхідну інформацію в режимі реального часу, що приносить користь як підприємству, так і користувачам. Завдяки централізації конфігурацій апаратного забезпечення та об'єднанню функцій програмного забезпечення в системах SCADA додаються

власні функціональні можливості для експлуатації кожного генераторного агрегату, що дозволяє створювати високогнучкі системи, досягаючи більшої ефективності в роботі енергетичної системи та швидшої реакції на збої з центральної диспетчерської [14]. Розвиток цих систем передбачає наявність більш ефективних схем безпеки для запобігання кібератакам або порушенням систем, що відповідають за експлуатацію [13]. У найближчому майбутньому очікується використання машинного навчання для виявлення моделей роботи на основі історичних даних; ці системи навчання дозволять створювати зв'язки між характеристиками поведінки та класами, що допоможе зміцнити системи безпеки та управління електроенергетичних підприємств.

2.4. Розробка моделі бази даних автоматизованої програмної системи

Для розробки моделі бази даних автоматизованої програмної системи кожен етап задачі повинен розбиватися на кілька етапів. На кожному кроці досліджується набір семантичних припущень, які дають початок підсхемі E/R. На кожному кроці до підсхеми, отриманої на попередньому кроці, будуть додаватися елементи, і так послідовно, доки не буде завершено дослідження всіх семантичних припущень, передбачених у формулюванні проблеми [15]. Припустимо, що формулювання є правильним описом проблемної області. Для найпростіших практичних випадків розв’язання буде здійснено за один крок. Підхід, який зазвичай використовується при побудові E/R-схем, полягає в тому, щоб спочатку визначити сутності, потім взаємозв’язки, а насамкінець — атрибути сутностей та взаємозв’язків [15].

Іноді також використовують інші типи інструментів, які допомагають виявити інформацію, що не вказана прямо в формулюванні завдання, і які виявляються дуже корисними. Отже, методологія створення концептуальної схеми бази даних складатиметься з таких етапів [15].

Аналіз формулювання, що описує проблемну сферу, та складання двох списків: один із кандидатів на об’єкти, а інший із можливими взаємозв’язками

разом із типом їхньої відповідності (1:1, 1:N). Крім того, слід вказати ті сумнівні поняття, щодо яких невідомо, як їх представити (як об’єкт чи як взаємозв’язок) [13].

Побудова матриці сутностей, у якій рядки та стовпці є назвами сутностей, а кожна комірка може містити або не містити назви взаємозв'язків. Ця матриця має такий вигляд: сутності — це E_1, E_2, ..., E_N, а взаємозв'язки — це I_1, I_2, ..., I_N. Оскільки матриця симетрична, комірки, позначені хрестиком, відповідають взаємозв'язкам, які вже вказані в іншій половині матриці. Символ - у комірці вказує на відсутність взаємозв'язку між двома згаданими сутностями [15]. Крім того, слід вказати типи відповідності для кожного взаємозв'язку. Наприклад, I_1 може бути взаємозв'язком 1:N. Важливо підкреслити, що ця матриця не відображає взаємозв'язки вищого ступеня. Під час розробки цієї матриці можна виявити взаємозв’язки, які не представлені явно в описі, але які, тим не менш, було б цікаво відобразити в схемі E/R [15]. Цей тип взаємозв’язків зазвичай виявляється завдяки здоровому глузду, хоча завжди необхідно підтверджувати їх з користувачем.

Використовуючи матрицю сутностей, будується перша схема E/R із сутностями, атрибутами, взаємозв’язками та їхніми типами відповідності. До цієї схеми додаються мінімальні та максимальні кардинальності [16].

Не всю семантику формулювання проблем можна буде відобразити на діаграмі E/R, тому буде включено розділ семантичних припущень, не відображених на діаграмі E/R. Остаточна схема E/R складається з діаграми E/R та розділу семантичних припущень, не відображених на діаграмі [15].

На останньому етапі схема E/R, отримана на попередньому етапі, уточнюється шляхом аналізу можливих надлишків, за умови наявності циклів із семантично еквівалентними взаємозв’язками. Надлишок у схемі E/R має місце, коли одна й та сама семантика повторюється, тому таку схему можна представити, зберігши ту саму семантику, але з меншою кількістю елементів [15]. Загалом, надмірність може бути присутньою, коли в схемі E/R існують цикли (кілька сутностей, пов’язаних кількома семантично пов’язаними

взаємозв’язками, що утворюють цикл). У цьому випадку слід перевірити, чи при видаленні одного взаємозв’язку семантика, представлена в ньому, може бути отримана за допомогою решти взаємозв’язків [16].

На основі цих принципів побудована схема (модель) бази даних системи оцінки надійності, яка показана на рис. 2.13.

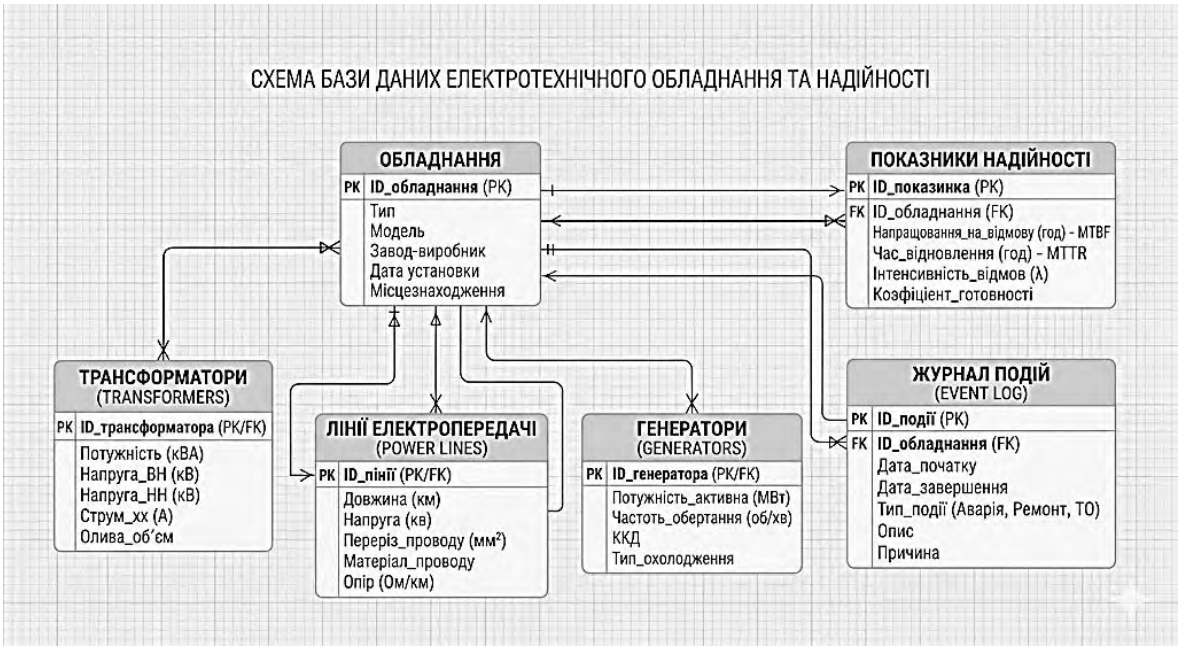


Рисунок 2.13 – Схема бази даних програмного застосунку

В принципі можуть існувати надлишкові атрибути або надлишкові взаємозв’язки. Щоб перевірити, що на діаграмі на рисунку 2.13 немає надлишкових атрибутів, буде показано, що семантику, пов’язану з кожним атрибутом на діаграмі, неможливо отримати на основі іншого атрибута або їхньої сукупності. У цій задачі надлишкових атрибутів немає [16].

2.5. Розробка алгоритмів функціональних блоків автоматизованої програмної системи

Алгоритми роботи функціональних блоків автоматизованої програмної системи оцінки стабільності енергетичних комплексів розробляються на основі методики оцінки надійності промислового обладнання. Зазвичай людям легше сприймати реальні приклади, ніж абстрактні описи [17]. Вони можуть зрозуміти та проаналізувати сценарій взаємодії з програмною

системою. Інженери з вимог використовують інформацію, отриману в ході такого обговорення, для формулювання фактичних вимог до системи [17]. Сценарії особливо корисні для деталізації ескізного опису вимог. Це приклади описів сеансів взаємодії. Кожен сценарій зазвичай охоплює одну взаємодію або невелику кількість можливих взаємодій [18]. Розробляються різні форми сценаріїв, що містять різну інформацію з різним рівнем деталізації про систему. Сценарій починається з нарису взаємодії. Під час процесу збору інформації до нього додаються деталі, щоб створити повне уявлення про цю взаємодію. У найзагальнішому вигляді сценарій може включати:

– опис того, чого очікують система та користувачі на початку сценарію;

– опис у сценарії нормального перебігу подій;

– опис того, що може піти не так і як це буде вирішуватися;

– інформацію про інші дії, що відбуваються одночасно;

– опис стану системи, коли сценарій завершується.

Ця модель враховує підходи до розробки, в яких вимоги формулюються з різним рівнем деталізації [18]. Кількість ітерацій спіралі зазвичай варіюється, тож спіраль завершується після збору деяких або всіх вимог користувача. Замість створення прототипів можна використовувати гнучку розробку, щоб одночасно розробляти вимоги та реалізацію системи. Деякі люди розглядають інженерію вимог як процес застосування методу структурованого аналізу, такого як об'єктно-орієнтований аналіз [18]. Це передбачає аналіз системи та розробку набору графічних моделей системи, таких як моделі випадків використання, які потім слугують специфікацією системи. Набір моделей описує поведінку системи та доповнюється додатковою інформацією, що описує, наприклад, необхідну продуктивність або надійність системи [19]. Хоча структуровані методи відіграють певну роль у процесі інженерії вимог, інженерія вимог набагато ширша, ніж те, що охоплюють ці методи. Зокрема, збір вимог — це діяльність, орієнтована на людей, а люди не люблять обмежень, що накладаються жорсткими моделями системи. Практично у всіх системах вимоги змінюються. Залучені особи краще розуміють, що саме вони хочуть від програмного забезпечення; організація, яка купує систему,

змінюється; вносяться зміни до апаратного та програмного забезпечення, а також до організаційного середовища системи [19]. Процес управління такими мінливими вимогами називається управлінням вимогами. На основі сформульованих вимог можна синтезувати блок-схему алгоритму головного блоку програмного застосунку, яка показана на рис. 2.14.

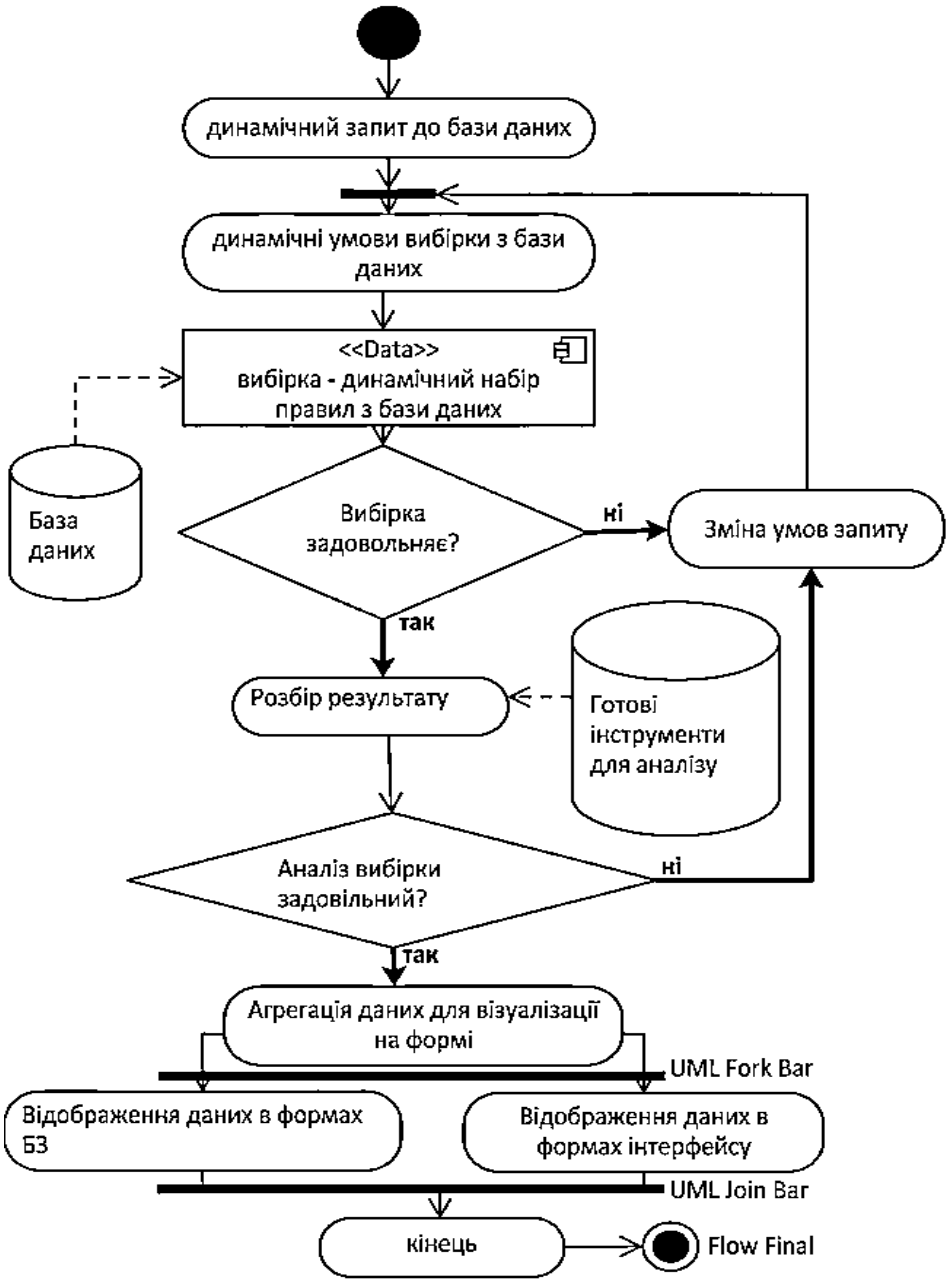


Рисунок 2.14 – Блок-схема алгоритму головного блоку програмного застосунку

Модельно-орієнтована розробка (MDE, Model-Driven Engineering) — це підхід до розробки програмного забезпечення, за якого головними результатами процесу розробки є не програми, а моделі [20]. На основі модельно-орієнтованої моделі розроблено блок-схему алгоритму формування даних для розрахунку надійності, показану на рис. 2.15.

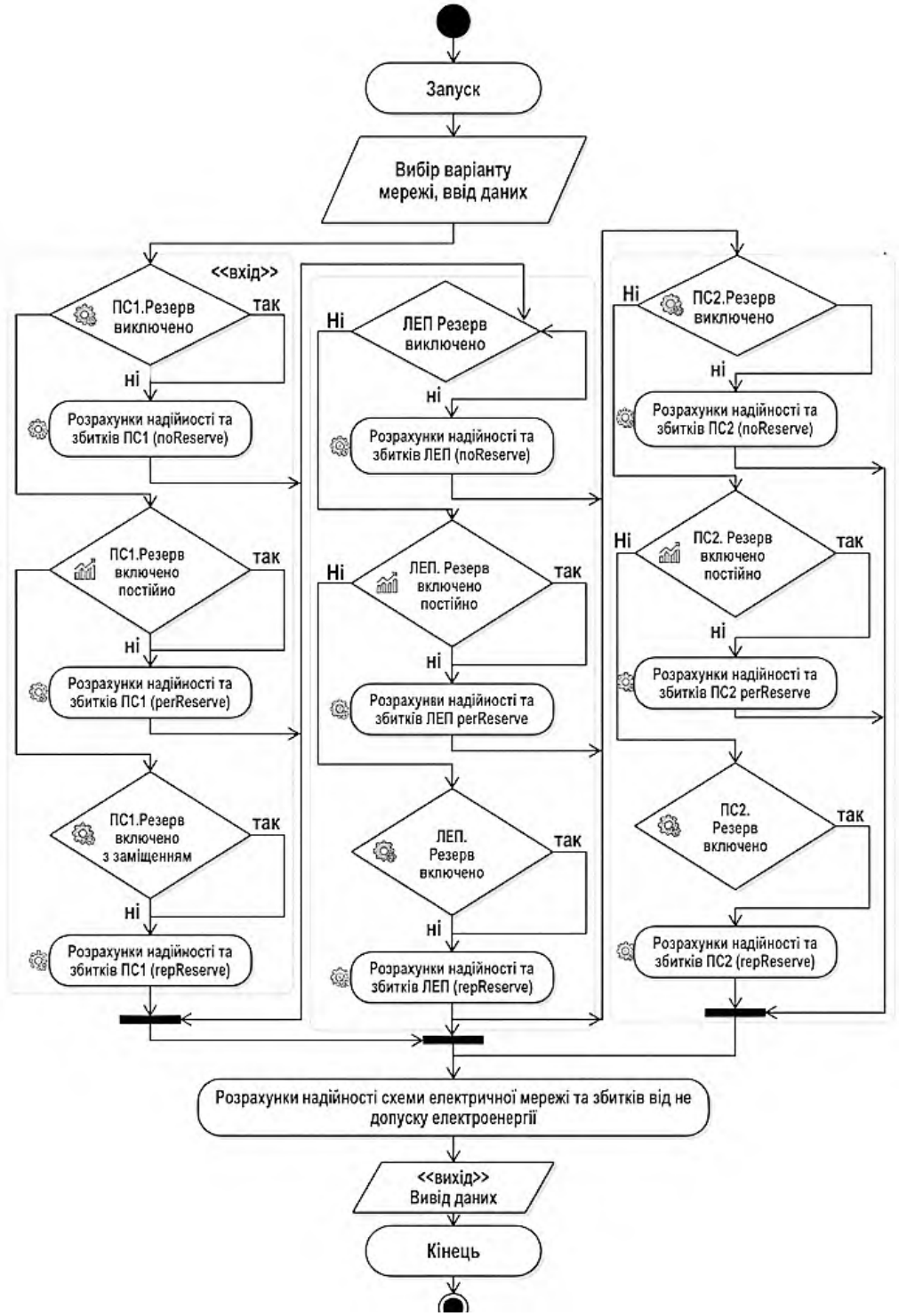


Рисунок 2.15 – Блок-схема алгоритму формування даних для розрахунку надійності

Програми, що виконуються на апаратно-програмній платформі, у цьому випадку генеруються автоматично на основі моделей. Прихильники MDE стверджують, що це підвищує рівень абстракції в програмній інженерії, оскільки інженери більше не мають турбуватися про деталі мови програмування або зазвичай не звертають на них уваги. Однак вважається, що MDE має ширшу сферу застосування, ніж MDA [18].

MDA зосереджується на етапах проектування та реалізації розробки програмного забезпечення, тоді як MDE охоплює всі аспекти процесу програмної інженерії. Отже, такі теми, як модельно-орієнтована інженерія вимог, процеси модельно-орієнтованої розробки програмного забезпечення та модельно-орієнтоване тестування, є частиною MDE, але на даний момент не входять до MDA [19]. Модельно-орієнтована інженерія має свої корені в модельно-орієнтованій архітектурі (MDA, Model-Driven Architecture), яка була запропонована Object Management Group (OMG) як нова парадигма розробки програмного забезпечення [19].

Основна ідея, що лежить в основі модельно-орієнтованої інженерії, полягає в тому, що має бути можливим повністю автоматизоване перетворення моделей у код. Щоб досягти цього, розробники повинні мати можливість будувати графічні моделі, семантика яких чітко визначена. Також потрібен спосіб додавання до графічних моделей інформації про те, як реалізуються операції, визначені в моделі [19]. Це можливо за допомогою підмножини UML-2, яка називається виконуваним UML або xUML.

2.6. Проєктування елементів інтерфейсу користувача програмного комплексу

Проєктування елементів інтерфейсу користувача програмного комплексу потрібно починати з визначення всіх аспектів інженерії візуального інтерфейсу в сучасних програмних застосунках [20]. UI — це абревіатура від терміна «User Interface» (користувацький інтерфейс). Під інтерфейсом ми

маємо на увазі «поверхню взаємодії», яку користувач бачить на своєму екрані і яка охоплює всі ті творчі елементи, що її складають, а саме: кольори, іконки, графіку, контент. Вдалий дизайн UI — це той, у якому користувач відразу розуміє, що бачить і чому це бачить. Важливими елементами вдалого дизайну UI є: узгодженість між елементами, передача зрозумілого повідомлення через дизайн, баланс між порожнім простором, контентом, легке розпізнавання елементів дизайну користувачем, «адаптивний дизайн» для застосунків та, загалом, правильне відображення дизайну на всіх екранах [20].

Щоб дати уявлення про те, що включає в себе дизайн, орієнтований на користувацький досвід (User Experience), існують такі приклади. Швидкий пошук того, що шукає відвідувач в інтерфейсі (навігаційні ланцюжки, фільтри, кнопки навігації, вбудована панель пошуку тощо) [21]. Функціональні точки взаємодії (елементи, на які можна натиснути/переміщати/наводити курсор). Наразі User Experience впливає і на результати SEO, оскільки, згідно з Google та Search Engine Journal, візуальний інтерфейс є вирішальним фактором. На основі цих принципів розроблений проєкт головної форми застосунку, показаної на рис. 2.16.

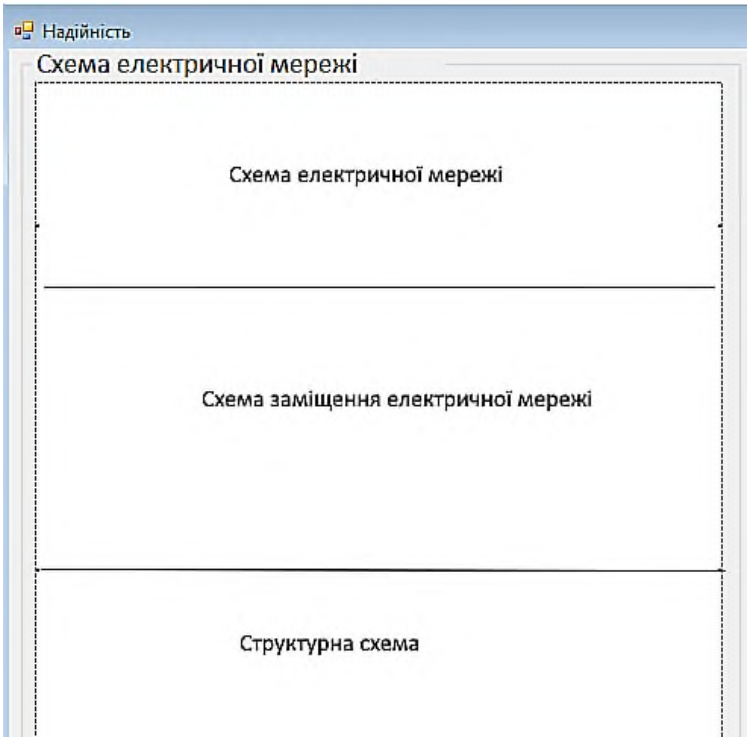


Рисунок 2.16 – Прототип головної форми застосунку

Найкращі практики, орієнтовані на оператора застосунку, такі моделі створюють належні умови для користувача. Коли користувач стикається з елементами інтерфейсу, спочатку він оглядає форми, кольори, артефакти, освітлення. Потім дивиться, як можно взаємодіяти в цьому просторі. Чи буде він користуватися? Яким би способом він не взаємодіяв з застосунком, швидше за все, весь його досвід з застосунком буде вважатися позитивним [19].

Те саме має відбуватися і під час користування будь яким програмним застосунком або платформою користувачем. Що означає UI/UX для показників ефективності застосунку: завдяки кращому User Experience ми можемо розраховувати на кращий досвід. Простіше кажучи, коли ми пропонуємо кращий досвід користувачеві, зростає коефіцієнт конверсії (Conversion Rate) і зменшується коефіцієнт відмов (Drop Off Rate). Автоматично це означає і менші витрати на ремаркетинг застосунку. Завдяки продуманому дизайну UI/UX зростає і CTR [22]. Користувачі діють більш правильно, клікають там, де їм цікаво, знаходять те, що шукають. Якісний дизайн інтерфейсу користувача також забезпечує кращі результати в рейтингу програм. Розробляючи дизайн UI/UX, ми закладаємо основу для найкращих результатів у майбутньому. Покращення таких показників, як ті, що були згадани, свідчать на користь реалізації правильної стратегії [21].

Розробка користувацького інтерфейсу буде здійснюватися таким чином, щоб у центрі уваги перебував користувач. Основною методологією розробки користувацького інтерфейсу є створення прототипу. Під прототипом ми розуміємо спрощену модель системи, яка була створена з метою опису її функціональності [20]. Методологія створення прототипу включає три етапи, яки показані на рис. 2.17.

Визначення вимог до інтерфейсу супроводжується аналізом професійної сфери, визначенням призначення програмного забезпечення та поставлених цілей, а також визначенням середовища, в якому працюватиме застосунок. Проектування, на якому буде обрано найбільш прийнятний варіант оформлення користувацького інтерфейсу серед запропонованих [21].

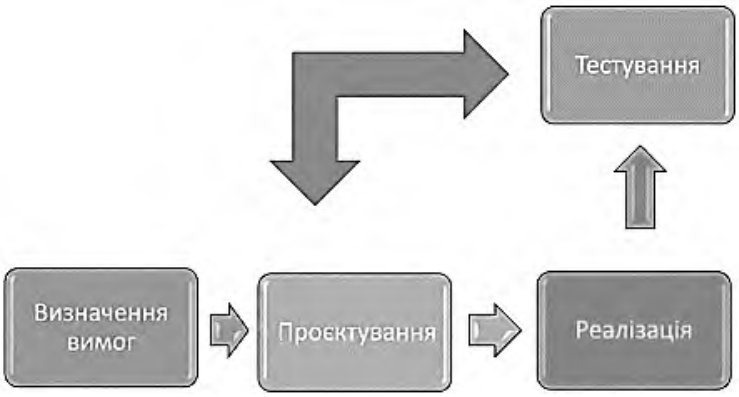


Рисунок 2.17 – Етапи створення прототипу інтерфейса застосунка

Тестування, під час якого експериментально перевіряється ефективність користувацького інтерфейсу в пілотному режимі і розпочинається процес вдосконалення в тих місцях, де це вважається необхідним, з поверненням до етапу проектування для реалізації необхідного вдосконалення [20]. Проєкт форми для розміщення результатів розрахунків показників надійності обладнання показана нижче, на рис. 2.18.

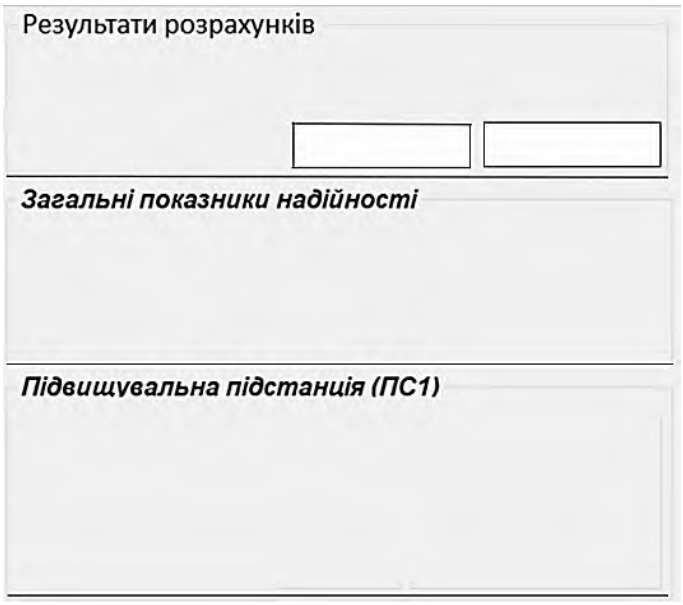


Рисунок 2.18 – Проєкт форми для розміщення результатів розрахунків показників надійності обладнання

Одними з найважливіших аспектів, які будуть враховані під час створення користувацького інтерфейсу застосунка, є вибір кольорів елементів управління та дизайн повідомлень про помилки [22].

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ АВТОМАТИЗАЦІЇ ДЛЯ АНАЛІЗУ СТАБІЛЬНОСТІ ЕНЕРГЕТИЧНИХ КОМПЛЕКСІВ В СУЧАСНИХ УМОВАХ

3.1. Розробка основних програмних модулів системи автоматизації для аналізу стабільності компонентів енергетичних комплексів

Visual Studio .NET — це повний набір інструментів для розробки веб-додатків ASP, XML-веб-сервісів, настільних та мобільних додатків. Visual Basic .NET, Visual C++ .NET та Visual C# .NET використовують одне й те саме інтегроване середовище розробки (IDE), що дозволяє їм спільно використовувати інструменти та полегшує створення рішень на різних мовах. Крім того, ці мови використовують функції .NET Framework, що надає доступ до ключових технологій для спрощення розробки веб-додатків ASP та веб-сервісів XML [22].

Форми Windows Forms — це нова платформа для розробки додатків для Microsoft Windows, що базується на .NET Framework. Цей фреймворк надає чіткий, об’єктно-орієнтований та розширюваний набір класів, що дозволяє розробляти складні додатки для Windows [23]. Крім того, форми Windows Forms можуть виступати в ролі локального користувацького інтерфейсу в багаторівневому розподіленому рішенні. Для отримання додаткової інформації [19].

Робоче середовище пропонує програмістам уніфікований, об’єктно-орієнтований, ієрархічний та розширюваний набір бібліотек класів (API). Наразі програмісти C++ використовують Microsoft Foundation Classes, а програмісти C# — Windows Foundation Classes. Робоче середовище об'єднує ці розрізнені моделі, надаючи програмістам Visual Basic та JScript можливість також отримати доступ до бібліотек [18]. Завдяки створенню набору загальних API для всіх мов програмування, Common Language Runtime забезпечує спадковість, контроль помилок та налагодження між мовами. Усі мови

програмування, від JScript до C++, можуть мати доступ до робочого середовища подібним чином, і програмісти можуть вільно обирати мову, яку бажають використовувати [20]. Зовнішній вигляд інтегрованого середовища розробки Visual Studio приведений на рис. 3.1.

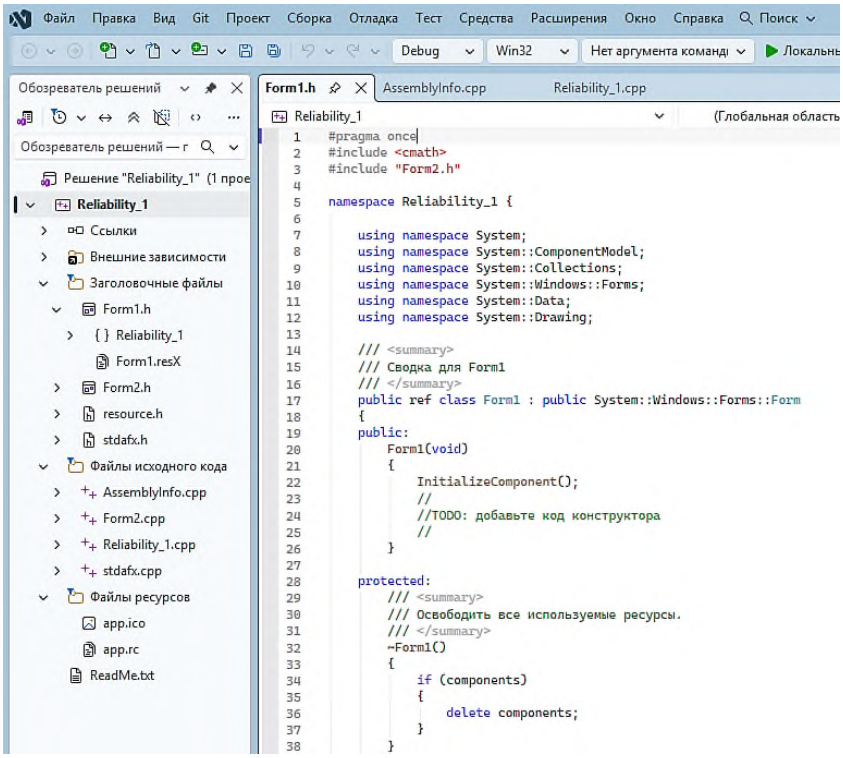


Рисунок 3.1 – Зовнішній вигляд інтегрованого середовища розробки Visual Studio

Приблизно кожні десять років спільнота розробників програмного забезпечення переосмислює «проблему», переносячи акцент з аспектів продукту на аспекти процесу [21]. Так, перейшли на мови структурованого програмування (продукт), за якими пішли методи структурованого аналізу (процес), за якими пішла інкапсуляція даних (продукт), за якими пішов нинішній акцент на моделі зрілості компетенцій Інституту програмної інженерії для розробки програмного забезпечення (процес) за яким слідують об'єктно-орієнтовані методи, за якими йде гнучка розробка програмного забезпечення. Оскільки природна тенденція маятника полягає в досягненні стану спокою в середині між двома крайніми положеннями, увага спільноти розробників програмного забезпечення постійно змінюється, оскільки при

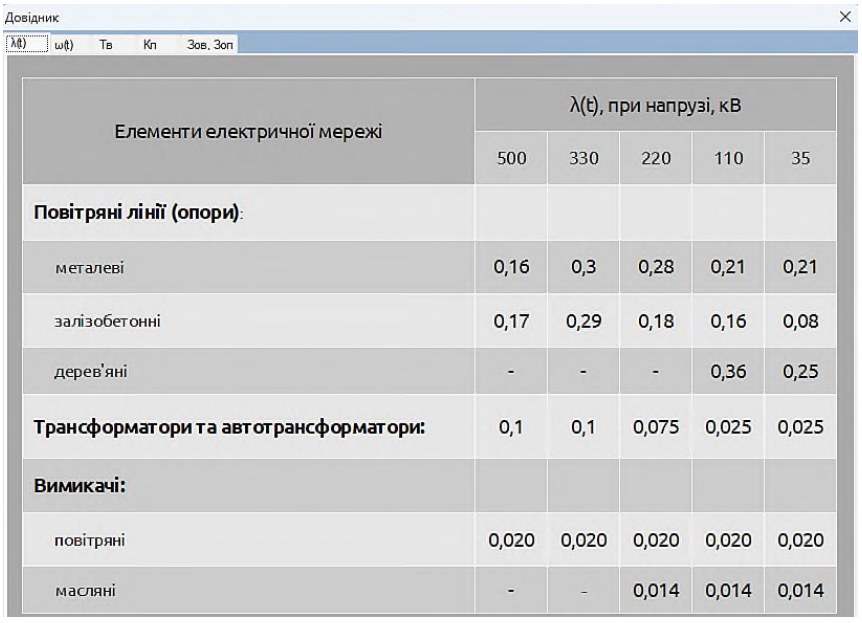
завершенні останнього коливання застосовується нова сила. Ці коливання є шкідливими самі по собі, оскільки вони заплутують пересічного фахівця з програмного забезпечення, радикально змінюючи уявлення про те, що означає якісно виконувати роботу [22].

На лістингу нижче, рис. 3.2, показане оголошення класу головної форми застосунку.

```
#pragma once
#include <cmath>
#include "Form2.h"

namespace Reliability_1 {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;


    public ref class Form1 : public System::Windows::Forms::Form


    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO
            //
        }

    protected:
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::GroupBox^


                                              groupBox1;
    protected:
    private: System::Windows::Forms::PictureBox^  pictureBox1;
    private: System::Windows::Forms::GroupBox^  groupBox2;
    private: System::Windows::Forms::GroupBox^  groupBox3;
    private: System::Windows::Forms::GroupBox^  groupBox4;
    private: System::Windows::Forms::GroupBox^  groupBox5;
    private: System::Windows::Forms::GroupBox^  groupBox6;
    private: System::Windows::Forms::GroupBox^  groupBox10;
    private: System::Windows::Forms::GroupBox^  groupBox7;
    private: System::Windows::Forms::GroupBox^  groupBox11;
    private: System::Windows::Forms::GroupBox^  groupBox8;
    private: System::Windows::Forms::GroupBox^  groupBox12;
    private: System::Windows::Forms::GroupBox^  groupBox9;
    private: System::Windows::Forms::Label^  label2;
```

Рисунок 3.2 – Оголошення класу головної форми застосунку

Хоча швидке засвоєння цілей повторного використання у розробці програмного забезпечення потенційно підвищує задоволення, яке отримують фахівці з програмного забезпечення від своєї роботи, воно також збільшує нагальність прийняття дуальності продукту та процесу [21]. Розглядати артефакт, що підлягає повторному використанню, як лише продукт або лише процес, затуманює контекст і способи його застосування, або приховує той факт, що кожне використання дає результат у вигляді продукту, який, у свою чергу, буде використаний як вхідні дані для якоїсь іншої діяльності з розробки програмного забезпечення [22]. Надання переваги одній точці зору над іншою значно зменшує можливості для повторного використання і, отже, втрачається можливість підвищити задоволення від роботи. Люди отримують стільки ж (або навіть більше) задоволення від творчого процесу, скільки й від кінцевого продукту. Як творчий фахівець у галузі програмного забезпечення, ви також повинні отримувати стільки ж задоволення від процесу, скільки й від кінцевого продукту [23]. Двоїстість продукту та процесу є важливим елементом для залучення творчих людей у міру розвитку програмної інженерії. Друга форма візуального інтерфейсу з відображенням довіднику нормативних параметрів надійності обладнання показана на рис. 3.3.



Елементи електричної мережі	λ(t), при напрузі, кВ				
	500	330	220	110	35
Повітряні лінії (опори):					
металеві	0,16	0,3	0,28	0,21	0,21
залізобетонні	0,17	0,29	0,18	0,16	0,08
дерев'яні	-	-	-	0,36	0,25
Трансформатори та автотрансформатори:	0,1	0,1	0,075	0,025	0,025
Вимикачі:					
повітряні	0,020	0,020	0,020	0,020	0,020
масляні	-	-	0,014	0,014	0,014

Рисунок 3.3 – Друга форма візуального інтерфейсу довіднику нормативних параметрів надійності обладнання

Visual Studio містить компонентні засоби розробки, такі як Visual C++, а також різноманітні допоміжні технології, що спрощують проектування, розробку та впровадження рішень у команді. Visual Studio підтримує середовище Microsoft .NET Framework, яке надає Common Language Runtime та уніфіковані класи програмування; ASP.NET використовує ці компоненти для створення веб-додатків ASP.NET та XML-веб-служб. Він також включає бібліотеку MSDN, яка містить всю документацію щодо цих інструментів програмування [22].

Завдяки зазначеним засобам середовища розробки створено програмний код методів класів програмного проєкту [23]. Лістинг методу ініціалізації елементів користувацького інтерфейсу наведено на рис. 3.4.

```
void InitializeComponent(void)
{
    System::ComponentModel::ComponentResourceManager^
      resources = (gcnew
      System::ComponentModel::ComponentResourceManager
      (Form1::typeid));
    this->groupBox1 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->pictureBox5 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox13 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox12 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox11 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox10 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox9 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox8 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox7 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox6 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox4 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox3 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox2 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox1 = (gcnew
      System::Windows::Forms::PictureBox());
    this->groupBox2 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->button4 = (gcnew System::Windows::Forms::Button());
    this->button3 = (gcnew System::Windows::Forms::Button());
    this->button2 = (gcnew System::Windows::Forms::Button());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->textBox2 = (gcnew System::Windows::Forms::TextBox
      ());
    this->textBox1 = (gcnew System::Windows::Forms::TextBox
      ());
    this->label2 = (gcnew System::Windows::Forms::Label());
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->groupBox3 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->numericUpDown4 = (gcnew
      System::Windows::Forms::NumericUpDown());
```

Рисунок 3.4 – Лістинг методу ініціалізації елементів користувацького інтерфейсу

Серед основних переваг, які надає успадкування при розробці додатків, можна виділити такі. Повторне використання коду – у випадках, коли потрібно створити клас, який, окрім власних методів, повинен містити методи, визначені в іншому класі, успадкування дозволяє уникнути необхідності переписувати всі ці методи в новому класі [23]. Обслуговування існуючих програм – використовуючи спадковість, якщо ми маємо клас з певною функціональністю і нам потрібно розширити цю функціональність, нам не потрібно модифікувати існуючий клас (який можна продовжувати використовувати для того типу програми, для якого він був розроблений), а можна створити клас, який успадковує перший, отримуючи всю його функціональність і додаючи свою власну [21].

Лістинг заповнення та обробки форми з параметрами надійності елементів електроенергетичного обладнання наведено нижче, на рис. 3.5.

```
private: System::Void checkBox2_CheckedChanged(System::Object^
  sender, System::EventArgs^  e) {
            //Включення резерву
            if(this->checkBox2->Checked == true)   //Якщо параметр
              "Checked" включений, тоді
            {                                      //стають
              активними слідуючі елементи
                this->numericUpDown21->Enabled = true; //Поля для
                  вводу параметрів надійності L2(ЛЕП)
                this->numericUpDown22->Enabled = true;
                this->numericUpDown23->Enabled = true;
                this->numericUpDown24->Enabled = true;
                this->numericUpDown42->Enabled = true;
                this->checkBox5->Enabled = true; //Включення
                  можливості вибору АВР
                this->pictureBox6->Visible = true; //Влючення ел. на
                  схемі
                this->pictureBox7->Visible = true;
                this->pictureBox8->Visible = true;
            }
            //Виключення резерву
            if(this->checkBox2->Checked == false)  //Якщо параметр
              "Checked" виключений, тоді
            {                                      //перестають бути
              активними слідуючі елементи
                this->numericUpDown21->Enabled = false; //Поля для
                  вводу параметрів надійності L2(ЛЕП)
                this->numericUpDown22->Enabled = false;
                this->numericUpDown23->Enabled = false;
                this->numericUpDown24->Enabled = false;
                this->numericUpDown42->Enabled = false;
                this->checkBox5->Enabled = false; //Виключення
                  можливості вибору АВР
                this->checkBox5->Checked = false; // параметру
                  "Checked" ел. вибору АВР присвоюэться значення
                  "Выключено"
                this->pictureBox6->Visible = false; //Виключення ел.
                  на схемі
                this->pictureBox7->Visible = false;
                this->pictureBox8->Visible = false;
                this->pictureBox9->Visible = false;
            }
        }
```

Рисунок 3.5 – Лістинг заповнення та обробки форми з параметрами надійності елементів електроенергетичного обладнання

Незважаючи на відмінності між цими визначеннями, у них є одна спільна риса — всі вони розглядають програмну інженерію як дисципліну або сферу формальних знань, пов’язану з процесами розробки програмного забезпечення [20]. Розглядати її як технологічну та управлінську дисципліну надзвичайно важливо, оскільки неможливо створювати високоякісне програмне забезпечення без застосування управлінських принципів і процесів.

3.2. Розробка і наповнення бази даних показників надійності для побудови автоматизованої системи

Система управління базами даних (СУБД) складається з набору взаємопов’язаних даних та набору програм для доступу до цих даних. Набір даних, який зазвичай називають базою даних, містить інформацію, що має значення для підприємства. Системи баз даних призначені для управління великими обсягами інформації [23]. Управління даними передбачає як визначення структур для зберігання інформації, так і забезпечення механізмів для її обробки. З метою полегшення розуміння запропонованих рішень проблем моделювання, у випадку найскладніших формулювань кожна проблема буде довільно розбита на кілька етапів. На кожному кроці буде досліджено набір семантичних припущень, які дадуть початок підсхемі E/R. На кожному кроці до підсхеми, отриманої на попередньому кроці, будуть додаватися елементи, і так послідовно, доки не буде завершено дослідження всіх семантичних припущень, передбачених у формулюванні проблеми [23]. Припустимо, що формулювання є правильним (і майже завжди повним) описом універсуму дискурсу. Для найпростіших практичних випадків розв’язання буде здійснено за один крок. Підхід, який зазвичай використовується при побудові E/R-схем, полягає в тому, щоб спочатку визначити сутності, потім взаємозв’язки, а насамкінець — атрибути сутностей та взаємозв’язків.

Іноді також зазвичай використовують інші типи інструментів, які допомагають виявити інформацію, що не представлена явно в твердженні, і які виявляються дуже корисними для недосвідчених розробників. Отже, пропозиція методології створення концептуальної схеми, що враховує ці аспекти, складатиметься з таких кроків. Вивчити задачу, що описує світогляд, та скласти два списки: один із кандидатами на сутності, а інший із можливими взаємозв'язками разом із типом відповідності (1:1, 1:N, N:M) [22]. Крім того, слід вказати ті сумнівні поняття, щодо яких невідомо, як їх представити (як сутність чи як взаємозв'язок). Схема бази даних приведена на рис. 3.6.

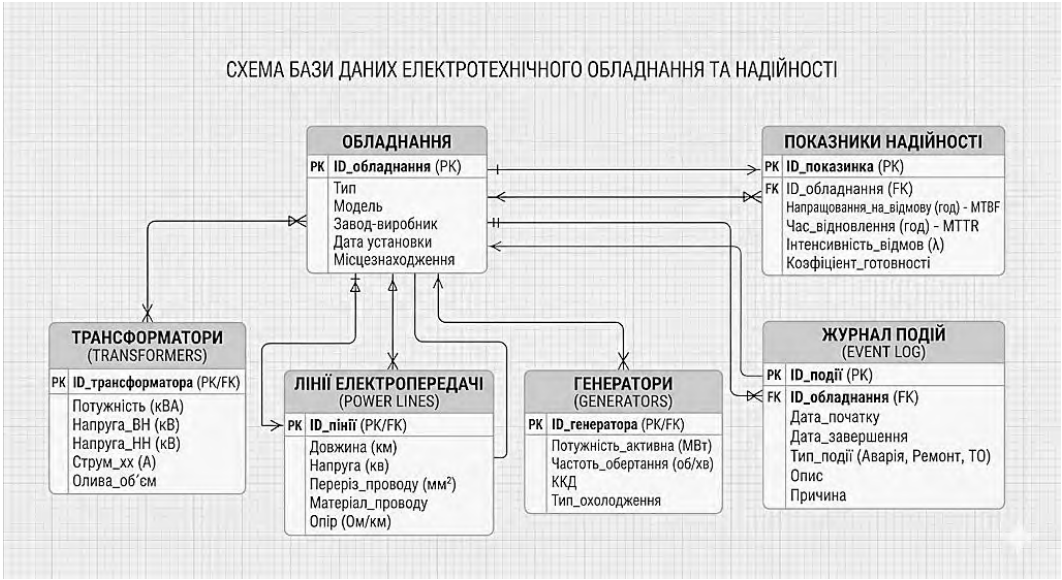


Рисунок 3.6 – Схема бази даних програмного застосунку

Часто доводиться перепроектувати базу даних, яка погано справляється з обробкою запитів. Процес адаптації схеми у цьому випадку називається еволюцією схеми (schema evolution). У базах, які вже використовуються, існують програми, що витягують або модифікують дані. Часто найкращий спосіб змінити схему, не впливаючи на існуючі програми, — це використовувати подання (views) [23].

Реляційна модель використовує набір таблиць для представлення як даних, так і їхніх взаємозв'язків. Кожна таблиця має кілька стовпців, а кожен стовпець має унікальну назву. Реляційна модель є прикладом моделі,

заснованої на записах. Моделі, засновані на записах, називаються так тому, що база даних структурована у вигляді записів фіксованого формату різних типів. Кожна таблиця містить записи певного типу [18]. Кожен тип запису визначає фіксовану кількість полів або атрибутів.

Заповнена таблиця з характеристиками дротів показана на рис. 3.7.

ID_лінії (PK/FK)	Довжина (км)	Напруга (кВ)	Переріз_проводу (мм²)	Матеріал_проводу	Опір (Ом/км)
101	45.3	110	240	AC	0.125
102	120.7	330	330	AC	0.098
103	15.1	35	150	AC	0.210
104	78.5	220	220	AC	0.115
105	189.2	500	500	AC	0.082
106	9.4	10	95	C	0.350
107	33.8	110	110	AC	0.160
108	62.0	150	150	AC	0.130
109	215.1	750	750	AC	0.055
110	11.2	35	120	AC	0.245

Рисунок 3.7 – Заповнена таблиця з характеристиками дротів

Зовнішні ключі дозволяють встановлювати зв’язки між рядками в таблицях. Для цього зовнішній ключ містить набір атрибутів однієї таблиці, які посилаються на первинний ключ іншої таблиці (або навіть тієї самої таблиці). Мета зовнішніх ключів — встановити зв'язок із первинним ключем, на який вони посилаються [23]. Отже, значення зовнішнього ключа мають бути присутніми у відповідному первинному ключі або бути нульовими. В іншому випадку зовнішній ключ є неправильним посиланням.

База даних складається з таблиць, які певним чином пов’язані між собою, щоб надати системі сенсу та якомога точніше відобразити те, що відбувається в реальному світі. Зв’язки можна класифікувати за їхніми властивостями, наприклад за кардинальністю. Кардинальність визначає кількість екземплярів одного об’єкта, які можуть бути пов’язані з певною кількістю екземплярів іншого об’єкта [20].

З точки зору виробництва, якість не залежить від якості проектування і визначається здатністю виробничого процесу відповідати специфікаціям, які вибрало керівництво для даного продукту. У тій мірі, в якій функціонування виробничого процесу не створює одиниць продукції, що відрізняються від

передбачених у специфікаціях, ми вважаємо, що маємо високу якість виробництва навіть для товарів, які не відрізняються високою якістю проектування. Рівень якості виробництва залежить від того, наскільки ефективно відповідальні особи узгодили технічні характеристики продукту з можливостями виробничого процесу. Найбільше значення для конкурентоспроможності підприємства, — це уявлення про якість, яке формує клієнт [21]. У прагненні поліпшити якість підприємство спочатку намагається підвищити якість виробництва, що дозволить йому стати кращим у своїй категорії. Потім стратегія якості змінюється, надаючи пріоритет якості проектування продукту, що таким чином сприятиме підвищенню продукту до вищої категорії.

3.3. Реалізація форм візуального користувацького інтерфейсу системи автоматизації

Для створення об’єктів інтерфейсу користувача нашого додатка ми додаємо елементи управління з панелі інструментів до форми. Спочатку панель інструментів розташована в лівій частині середовища розробки. На ній є кілька вкладок для різних категорій елементів управління, таких як «Windows Forms» та «Дані» [20]. Робота панелі інструментів Панель інструментів містить різноманітні елементи керування, які ми можемо використовувати для додавання ілюстрацій, міток, кнопок, списків, смуг прокрутки, меню та геометричних фігур до інтерфейсу користувача. Кожен елемент керування, який ми додаємо до форми, стає програмованим об’єктом інтерфейсу користувача в нашій програмі. Ці об’єкти видно користувачам під час роботи програми і вони функціонують як стандартні об’єкти будь-якої програми на базі Windows [22]. Закриття та відкриття панелі інструментів Закриття та відкриття панелі інструментів Щоб закрити панель інструментів, натисніть кнопку «Закрити» (x) у верхньому правому куті панелі інструментів. Щоб відкрити панель інструментів, у меню «Вид» натисніть «Панель

інструментів». Щоб панель інструментів залишалася відкритою, натисніть значок закріплення на заголовній панелі панелі інструментів. Приховування та повторне відкриття панелі інструментів Приховування та повторне відкриття панелі інструментів Щоб приховати панель інструментів, клацніть на значку на заголовній панелі панелі інструментів [23]. Щоб знову відкрити панель інструментів, коли вона прихована, у меню «Вид» клацніть на «Панель інструментів». Зміна розташування панелі інструментів Зміна розташування панелі інструментів [23].

За допомогою інструментів для створення користувацького інтерфейсу було створено форму з відображенням схем ліній електропередачі, яка показана на рис. 3.8.

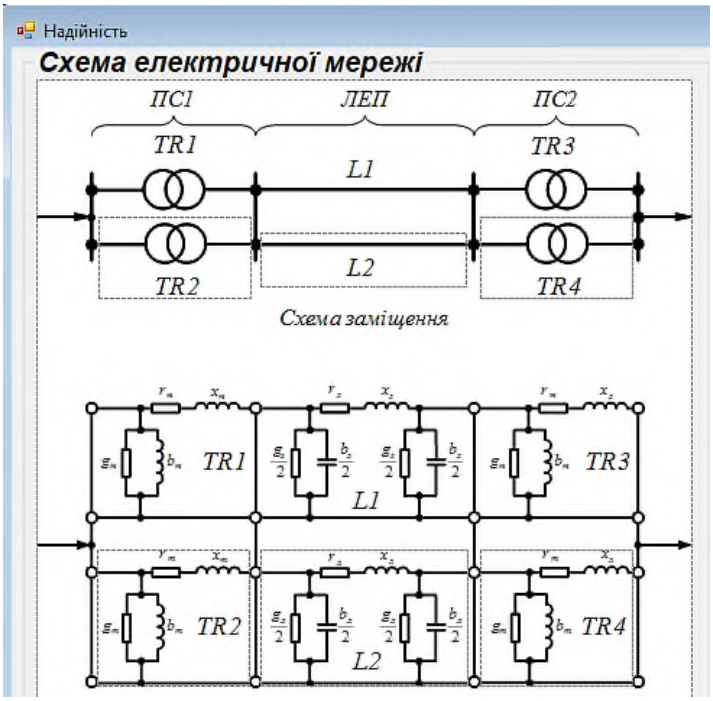


Рисунок 3.8 – Форма з відображенням схем ліній електропередачі

Коли ми розпочинаємо проект у .NET, конструктор Windows Forms відкривається у режимі «Дизайн», відображаючи форму Form1 проекту. Ми розмістимо елементи керування з панелі інструментів на формі, щоб створити інтерфейс користувача вікна, яке використовується в нашій програмі [21]. Форма за замовчуванням містить мінімальні елементи, які використовуються в більшості форм: рядок заголовка, поле управління та кнопки «Згорнути»,

«Розгорнути» і «Закрити». Перегляд форми Перегляд форми в режимі «Дизайн» У Провіднику рішень двічі клацніть на формі. У Провіднику рішень виберіть форму та натисніть кнопку «Переглянути конструктор» на панелі інструментів. Першим кроком під час створення нової програми є створення інтерфейсу користувача. У .NET можна швидко створити інтерфейс користувача, додаючи елементи управління до форми з панелі інструментів. Додавання елементів управління до форми: Якщо панель інструментів не відкрита, у меню «Вид» натисніть «Панель інструментів» [22]. У панелі інструментів клацніть на елементі управління, який потрібно додати до форми, і перетягніть його у потрібне місце на формі. Зміна положення та розміру елементів управління. Щоб змінити положення елемента управління, клацніть на ньому, щоб виділити його, і перетягніть у потрібне місце на формі. Щоб змінити розмір елемента управління, клацніть на ньому, щоб виділити його, перетягніть один із краників регулювання розміру, доки елемент управління не набуде потрібного розміру, і відпустіть кнопку миші [20].

Форма інтерфейсу з параметрами для розрахунків показана на рис. 3.9.

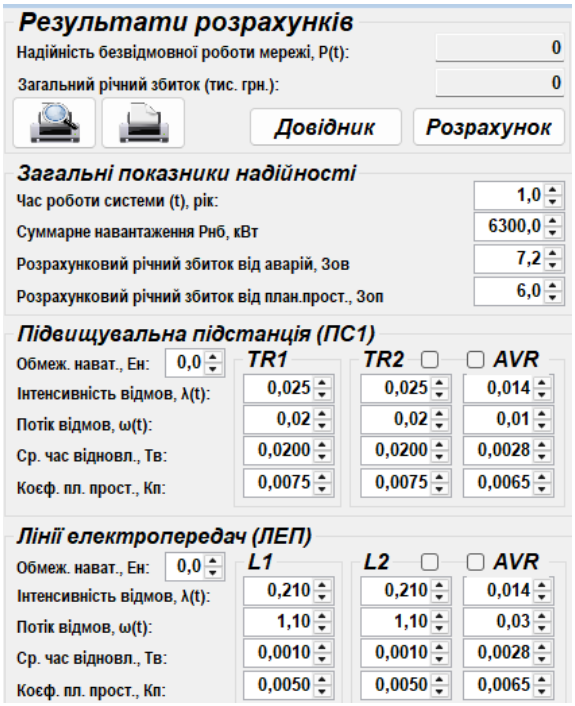


Рисунок 3.9 – Форма інтерфейсу з параметрами для розрахунків

Ми можемо змінювати властивості, виділяючи об’єкти у формі та змінюючи їхні налаштування у вікні «Властивості». Налаштування властивостей у вікні «Властивості». Якщо вікно «Властивості» не відкрито, у меню «Вигляд» натисніть «Вікно властивостей». Клацніть на елементі управління, для якого потрібно налаштувати властивість. У вікні «Властивості» виберіть властивість і встановіть потрібне значення [20].

Крім того, інформація, пов’язана з пересуванням користувача, така як час, проведений на певному вузлі, допомагають зробити висновки щодо того, наскільки релевантним є вміст вузла для поточної мети, тоді як кількість відвідувань дозволяє передбачити рівень обізнаності користувача щодо тематично пов’язаного вузла, який він відвідав. Така інформація не є надійною і не повинна використовуватися як єдине джерело для створення моделі користувача.

3.4. Розробка засобів відображення результатів оцінки стабільності компонентів енергетичних комплексів

Методи формування звітів Далі описано метод формування звітів та наведено їхні типи. Метод автоматичного формування звітів визначає послідовність дій, якої слід дотримуватися під час формування звітів. Нижче наведено графік, що ілюструє цей процес [23].

На наведеному графіку видно, що дані спочатку готуються, а потім вводяться в систему аналізу, де, залежно від їхньої структури, переходять до наступного етапу обробки або формується звіт. Варто зазначити, що квадрат із пунктирною рамкою позначає систему формування звітів [23]. Під час формування звітів з даних необхідно оцінити, чи є цей набір інформації простими даними чи змінними даними, оскільки залежно від цього процес повинен змінюватися, інакше він може стати неефективним. Нижче наведено ілюстрацію автоматизованого формування звітів, рис. 3.10.

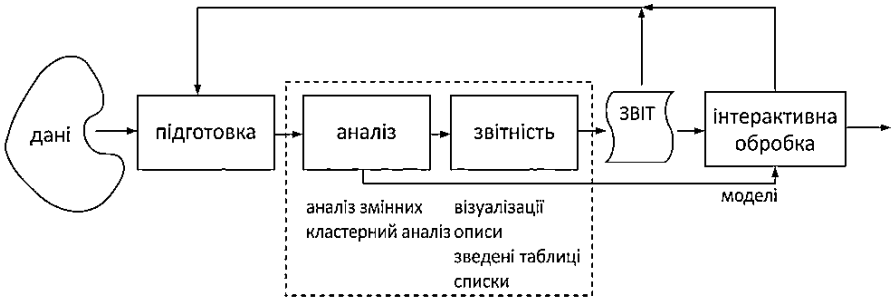


Рисунок 3.10 – Інструмент для автоматизованого формування звітів на етапі аналізу даних

Сервер звітів підтримує різні режими розгортання для екземплярів звітів. Існують такі режими: Нативний режим: цей режим передбачає використання сервера звітів як автономного сервера додатків, який забезпечує повну візуалізацію, управління, обробку та доставку звітів і моделей звітів, що можуть бути створені [22]. Цей режим є типовим під час встановлення нового екземпляра SQL Server. Інтегрований режим SharePoint: у цьому режимі сервер звітів повинен працювати в рамках ферми серверів SharePoint. Кожен із цих сайтів забезпечує фронтальний доступ до вмісту сервера та всіх його операцій [23]. Сервер звітів забезпечує всю обробку та візуалізацію звітів. Цей режим вимагає, щоб на кожному сервері ферми були встановлені SharePoint Foundation та SharePoint Server. Також потрібні Windows SharePoint Services або Office SharePoint Server.

Ви можете прив’язати звіт до одного з принтерів, встановлених у системі. Це означає, що вибраний принтер стане принтером за замовчуванням під час друку цього звіту. Це може бути корисно у випадках, коли в системі є кілька різних принтерів; наприклад, текстові документи можна прив’язати до монохромних принтерів, а документи з графікою — до кольорових. У налаштуваннях принтера є пункт «За замовчуванням» — якщо його вибрати, звіт не буде прив’язаний до жодного конкретного принтера, а друкуватиметься на системному принтері за замовчуванням [22].

На рис. 3.11 показаний сформований звіт в режимі попереднього перегляду з можливістю налаштування режимів друку.

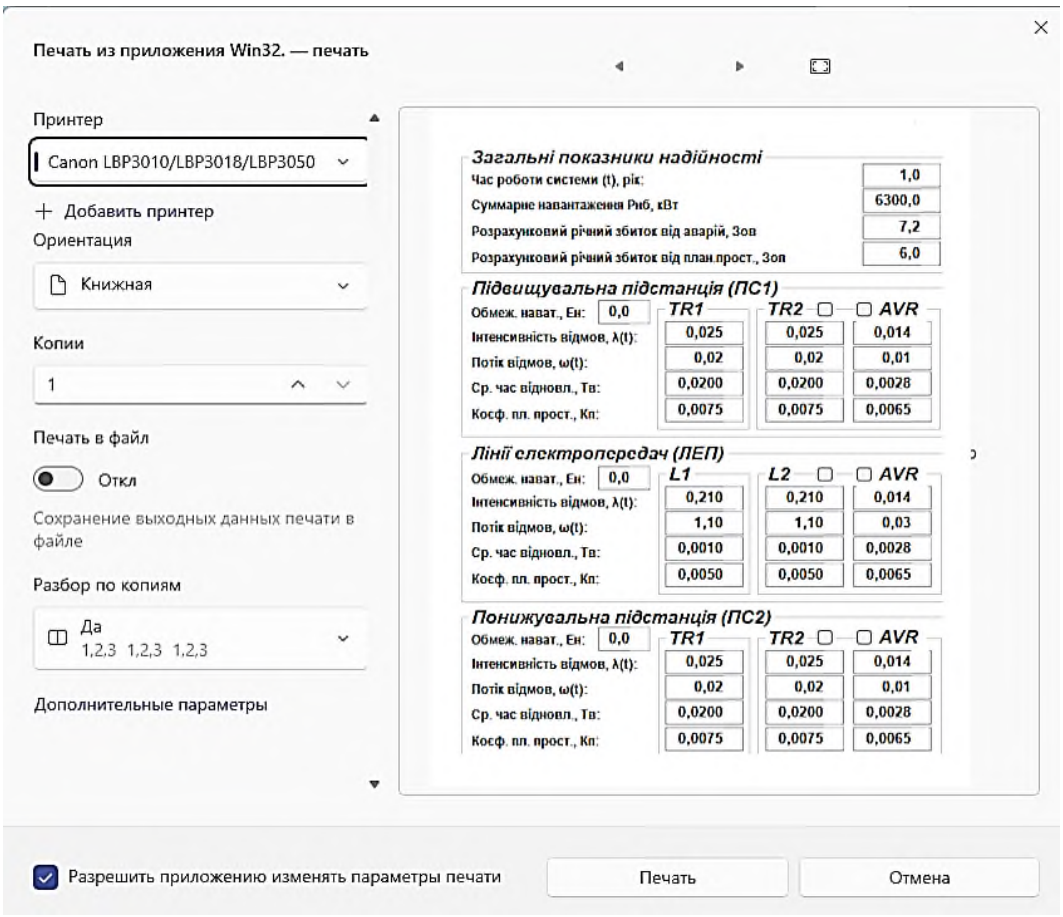


Рисунок 3.11 – Сформований звіт в режимі попереднього перегляду

Ви також можете встановити кількість копій звіту, що друкуються, та вказати, чи слід сортувати вихідні дані. Значення, встановлене в цьому діалоговому вікні, відображатиметься у вікні «Друк» під час друку звіту. Якщо встановлено прапорець «Подвійний прохід», формування звіту відбуватиметься у два етапи. Під час першого проходу створюється чернетка звіту, яка поділяється на сторінки, але не доступна для попереднього перегляду. Під час другого проходу чернетка звіту перетворюється на стандартний звіт, який потім зберігається у потоці попереднього перегляду [21].

Зазвичай за формування звіту відповідає ядро – генератор звітів. Воно відображає смуги звіту в певному порядку стільки разів, скільки цього вимагають дані, створюючи таким чином повний звіт [22]. Іноді виникає необхідність відобразити звіт у нестандартній формі, чого ядро зробити не може. У цьому випадку можна побудувати звіт вручну, використовуючи подію на сторінці дизайну звіту. Якщо для цієї події визначено обробник, то ядро

передає йому управління, коли потрібно вивести дані. Водночас ядро автоматично обробляє відображення тих смуг, які розташовані на сторінці, таких як «Заголовок звіту», «Заголовок сторінки», «Заголовок стовпця», «Нижній колонтитул звіту», «Нижній колонтитул сторінки», «Нижній колонтитул стовпця» та «Фон». Ядро також обробляє створення нових сторінок і стовпців [21]. Мета обробника події— відображати смуги даних, їхні заголовки та нижні колонтитули в порядку, який контролює користувач. Тобто суть обробника полягає в тому, щоб давати ядру команди на відображення смуг у певний час. Решту ядро робить самостійно: створює нові сторінки, щойно на поточній не залишається вільного місця, обробляє скрипти, прив’язані до подій, тощо.

ВИСНОВКИ

Була виконана випускна кваліфікаційна робота: " Програмний проєкт аналізу стабільності енергетичних комплексів в сучасних умовах». Робота присвячена актуальній темі – розробці автоматизованої системи для оперативного аналізу і розрахунку параметрів надійності елементів електроенергетичного комплексу. Актуальність теми кваліфікаційної роботи полягає у зростаючій складності сучасних електроенергетичних комплексів і зростання збитків від аварій [18]. Під час здійснення кваліфікаційної роботи було спроєктовано та реалізовано програму розрахунку параметрів надійності схем з'єднання компонентів обладнання [21].

Під час виконання кваліфікаційної роботи були отримані наступні результати:

– проведено дослідження математичних моделей забезпечення стабільності компонентів енергооб'єктів;

– розроблені структури програмної системи забезпечення стабільності енергооб'єктів;

– розроблені функціональні моделі оцінки надійності енергооб'єктів засобами програмного комплексу;

– розроблені моделі бази даних автоматизованої програмної системи;

– розроблені алгоритми функціональних блоків автоматизованої програмної системи;

– спроєктовані елементи інтерфейсу користувача програмного комплексу;

– розроблені основні програмні модулі системи автоматизації для аналізу стабільності компонентів енергетичних комплексів;

– розробка і наповнення бази даних показників надійності для побудови автоматизованої системи;

– реалізовані форми візуального користувацького інтерфейсу системи автоматизації;

– розроблені засоби відображення результатів оцінки стабільності компонентів енергетичних комплексів.

Розроблені основні програмні модулі системи автоматизації для аналізу стабільності компонентів енергетичних комплексів, розробці і наповненню бази даних показників надійності для побудови автоматизованої системи, реалізації форм візуального користувацького інтерфейсу системи автоматизації [22].

Аналіз розробленого програмного комплексу аналізу стабільності енергетичних комплексів в сучасних умовах показує, що отримані програмні результати можуть бути використані для аналізу роботи промислового обладнання, так і у вищих навчальних закладах при проведенні лекційних і практичних занять [22].

На основі отриманих результатів можна зробити висновок, що виконана робота характеризує актуальність теми, демонструє ефективність розроблених програмних модулів.

Результати роботи можуть бути використані для організації, проведення, обліку результатів та оцінювання параметрів надійності електротехнічного обладнання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Лежнюк П. Д., Нетребський К. В. Надійність та оптимізація режимів роботи електричних мереж : монографія. Вінниця : ВНТУ, 2020. 164 с.

2. Кирик В. В. Надійність систем електропостачання : підручник. Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2021. 284 с.

3. Каплін С. Ю. Надійність об’єктів електроенергетики та електротехнічних систем : навч. посіб. Кременчук : КрНУ, 2022. 190 с.

4. Li, W. *Risk Assessment of Power Systems: Models, Methods, and Applications*. 2nd ed. Wiley-IEEE Press, 2022. 384 p.

5. Zaporozhets, A. *Reliability and Efficiency of Energy Systems: Models and Methods of Analysis*. Cham : Springer, 2023. 240 p.

6. Гончаренко О. В., Сидоренко Д. М. Критерії оцінки структурної надійності схем видачі потужності helio-систем великої продуктивності. *Сучасні проблеми електроенерготехніки та автоматики* : матер. XXV Міжнар. наук.-техн. конф. молодих учених (Київ, 21–23 травня 2024 р.). Київ : КПІ, 2024. С. 89–92.

7. Яценко М. В., Козирєва Л. В. Діагностика та надійність електроенергетичного обладнання станцій і підстанцій : навч. посібник. Дніпро : НТУ «Дніпровська політехніка», 2023. 215 с.

8. Кулик М. М., Дерій В. О. Оцінювання надійності та живучості магістральних електричних мереж в умовах операційних обмежень. *Технічна електродинаміка*. 2022. № 4. С. 58–66. DOI:

9. Бункевич О. М., Самойленко І. О. Застосування SCADA-систем для підвищення надійності диспетчерського керування розподільними електричними мережами. *Енергетика і автоматика*. 2023. № 2. С. 45–53.

10. Манілов А. В. Підвищення надійності схем релейного захисту та автоматики елементів систем електропостачання. *Електричні мережі та системи*. 2021. № 1. С. 32–38.

11. Калиняк Б. М., Федорів М. П. Аналіз стійкості та надійності енергосистем з високим рівнем проникнення сонячних та вітрових електростанцій. *Відновлювана енергетика*. 2024. № 3. С. 74–81.

12. Баран П. М. Моделювання деградації ізоляції силових кабельних ліній для прогнозування безвідмовної роботи розподільних мереж. *Праці Інституту електродинаміки НАН України*. 2025. Вип. 70. С. 112–119.

13. Billinton, R., Allan, R. N. *Reliability Evaluation of Power Systems*. New York : Springer Science & Business Media, 2021. 614 p. (Reprint/Updated Text).

14. Cherkes, B., Kyryk, V. Cyber-physical security and reliability of automated electrical dispatch control systems. IEEE International Conference on Modern Electrical and Energy Systems (MEES). 2024. P. 114–119.

15. Трофименко О. Г., Прокоп Ю. В., Швайко І. Г. Програмування мовою C++ : навч. посіб. Одеса : Фенікс, 2021. 344 с.

16. Josuttis, N. M. C++20 - The Complete Guide. Independence : O'Reilly Media, 2021. 450 p.

17. Meyers, S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 (with updates for C++20). New York : O'Reilly Media, 2020. 336 p.

18. Костін Ю. Д., Організація обчислювальних процесів та програмування на C++ : підручник. Харків : ХНУРЕ, 2022. 280 с.

19. Lippman, S. B., Lajoie, J., Moo, B. E. C++ Primer. 6th ed. Boston : Addison-Wesley, 2023. 920 p.

20. Gregoire, M. Professional C++. 6th ed. Indianapolis : John Wiley & Sons, 2024. 1184 p.

21. Креневич А. П. C++ Орієнтоване програмування : підручник. Київ : ВПЦ "Київський університет", 2020. 255 с.

22. Васильєв О. М. Програмування на C++ для початківців : навч. посіб.

23. Stroustrup, B. A Tour of C++. 3rd ed. Boston : Addison-Wesley, 2022. 320 p.

ДОДАТОК А

Лістинги програмних модулів застосунку

Оголошення класу головної форми застосунку

```
#pragma once
#include <cmath>
#include "Form2.h"

namespace Reliability_1 {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;


    public ref class Form1 : public System::Windows::Forms::Form


    {
    public:
        Form1(void)
        {
            InitializeComponent();
            //
            //TODO
            //
        }

    protected:
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::GroupBox^


                                                groupBox1;
    protected:
    private: System::Windows::Forms::PictureBox^  pictureBox1;
    private: System::Windows::Forms::GroupBox^  groupBox2;
    private: System::Windows::Forms::GroupBox^  groupBox3;
    private: System::Windows::Forms::GroupBox^  groupBox4;
    private: System::Windows::Forms::GroupBox^  groupBox5;
    private: System::Windows::Forms::GroupBox^  groupBox6;
    private: System::Windows::Forms::GroupBox^  groupBox10;
    private: System::Windows::Forms::GroupBox^  groupBox7;
    private: System::Windows::Forms::GroupBox^  groupBox11;
    private: System::Windows::Forms::GroupBox^  groupBox8;
    private: System::Windows::Forms::GroupBox^  groupBox12;
    private: System::Windows::Forms::GroupBox^  groupBox9;
    private: System::Windows::Forms::Label^  label2;
```

Лістинг методу ініціалізації елементів користувацького інтерфейсу

```
void InitializeComponent(void)
{
    System::ComponentModel::ComponentResourceManager^ 
      resources = (gcnew 
      System::ComponentModel::ComponentResourceManager 
      (Form1::typeid));
    this->groupBox1 = (gcnew System::Windows::Forms::GroupBox 
      ());
    this->pictureBox5 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox13 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox12 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox11 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox10 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox9 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox8 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox7 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox6 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox4 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox3 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox2 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->pictureBox1 = (gcnew 
      System::Windows::Forms::PictureBox());
    this->groupBox2 = (gcnew System::Windows::Forms::GroupBox 
      ());
    this->button4 = (gcnew System::Windows::Forms::Button());
    this->button3 = (gcnew System::Windows::Forms::Button());
    this->button2 = (gcnew System::Windows::Forms::Button());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->textBox2 = (gcnew System::Windows::Forms::TextBox 
      ());
    this->textBox1 = (gcnew System::Windows::Forms::TextBox 
      ());
    this->label2 = (gcnew System::Windows::Forms::Label());
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->groupBox3 = (gcnew System::Windows::Forms::GroupBox 
      ());
    this->numericUpDown4 = (gcnew 
      System::Windows::Forms::NumericUpDown());
```

Лістинг заповнення та обробки форми з параметрами надійності елементів електроенергетичного обладнання

```
private: System::Void checkBox2_CheckedChanged(System::Object^
  sender, System::EventArgs^  e) {
            //Включення резерву
            if(this->checkBox2->Checked == true)   //Якщо параметр
              "Checked" включений, тоді
            {                                      //стають
              активними слідуючі елементи
                this->numericUpDown21->Enabled = true; //Поля для
                  вводу параметрів надійності L2(ЛЕП)
                this->numericUpDown22->Enabled = true;
                this->numericUpDown23->Enabled = true;
                this->numericUpDown24->Enabled = true;
                this->numericUpDown42->Enabled = true;
                this->checkBox5->Enabled = true; //Включення
                  можливості вибору АВР
                this->pictureBox6->Visible = true; //Влючення ел. на
                  схемі
                this->pictureBox7->Visible = true;
                this->pictureBox8->Visible = true;
            }
            //Виключення резерву
            if(this->checkBox2->Checked == false)  //Якщо параметр
              "Checked" виключений, тоді
            {                                      //перестають бути
              активними слідуючі елементи
                this->numericUpDown21->Enabled = false; //Поля для
                  вводу параметрів надійності L2(ЛЕП)
                this->numericUpDown22->Enabled = false;
                this->numericUpDown23->Enabled = false;
                this->numericUpDown24->Enabled = false;
                this->numericUpDown42->Enabled = false;
                this->checkBox5->Enabled = false; //Виключення
                  можливості вибору АВР
                this->checkBox5->Checked = false; // параметру
                  "Checked" ел. вибору АВР присвоюэться значення
                  "Выключено"
                this->pictureBox6->Visible = false; //Виключення ел.
                  на схемі
                this->pictureBox7->Visible = false;
                this->pictureBox8->Visible = false;
                this->pictureBox9->Visible = false;
            }
        }
```

```
void InitializeComponent(void)
{
    System::ComponentModel::ComponentResourceManager^
      resources = (gcnew
      System::ComponentModel::ComponentResourceManager
      (Form1::typeid));
    this->groupBox1 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->pictureBox5 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox13 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox12 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox11 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox10 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox9 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox8 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox7 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox6 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox4 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox3 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox2 = (gcnew
      System::Windows::Forms::PictureBox());
    this->pictureBox1 = (gcnew
      System::Windows::Forms::PictureBox());
    this->groupBox2 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->button4 = (gcnew System::Windows::Forms::Button());
    this->button3 = (gcnew System::Windows::Forms::Button());
    this->button2 = (gcnew System::Windows::Forms::Button());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->textBox2 = (gcnew System::Windows::Forms::TextBox
      ());
    this->textBox1 = (gcnew System::Windows::Forms::TextBox
      ());
    this->label2 = (gcnew System::Windows::Forms::Label());
    this->label1 = (gcnew System::Windows::Forms::Label());
    this->groupBox3 = (gcnew System::Windows::Forms::GroupBox
      ());
    this->numericUpDown4 = (gcnew
      System::Windows::Forms::NumericUpDown());
```

```
this->numericUpDown3 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown2 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown1 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->label6 = (gcnew System::Windows::Forms::Label());
this->label5 = (gcnew System::Windows::Forms::Label());
this->label4 = (gcnew System::Windows::Forms::Label());
this->label3 = (gcnew System::Windows::Forms::Label());
this->groupBox4 = (gcnew System::Windows::Forms::GroupBox
  ());
this->numericUpDown41 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->label11 = (gcnew System::Windows::Forms::Label());
this->label10 = (gcnew System::Windows::Forms::Label());
this->label9 = (gcnew System::Windows::Forms::Label());
this->label8 = (gcnew System::Windows::Forms::Label());
this->label7 = (gcnew System::Windows::Forms::Label());
this->groupBox10 = (gcnew System::Windows::Forms::GroupBox
  ());
this->checkBox4 = (gcnew System::Windows::Forms::CheckBox
  ());
this->checkBox1 = (gcnew System::Windows::Forms::CheckBox
  ());
this->numericUpDown16 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown15 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown14 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown13 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown12 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown11 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown10 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown9 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->groupBox7 = (gcnew System::Windows::Forms::GroupBox
  ());
this->numericUpDown8 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown7 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown6 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown5 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->groupBox5 = (gcnew System::Windows::Forms::GroupBox
  ());
```

```
this->numericUpDown42 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->label12 = (gcnew System::Windows::Forms::Label());
this->label13 = (gcnew System::Windows::Forms::Label());
this->label14 = (gcnew System::Windows::Forms::Label());
this->label15 = (gcnew System::Windows::Forms::Label());
this->label16 = (gcnew System::Windows::Forms::Label());
this->groupBox11 = (gcnew System::Windows::Forms::GroupBox
  ());
this->checkBox5 = (gcnew System::Windows::Forms::CheckBox
  ());
this->checkBox2 = (gcnew System::Windows::Forms::CheckBox
  ());
this->numericUpDown28 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown27 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown26 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown25 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown24 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown23 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown22 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown21 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->groupBox8 = (gcnew System::Windows::Forms::GroupBox
  ());
this->numericUpDown20 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown19 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown18 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->numericUpDown17 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->groupBox6 = (gcnew System::Windows::Forms::GroupBox
  ());
this->numericUpDown43 = (gcnew
  System::Windows::Forms::NumericUpDown());
this->label17 = (gcnew System::Windows::Forms::Label());
this->label18 = (gcnew System::Windows::Forms::Label());
this->label19 = (gcnew System::Windows::Forms::Label());
this->label20 = (gcnew System::Windows::Forms::Label());
this->label21 = (gcnew System::Windows::Forms::Label());
this->groupBox12 = (gcnew System::Windows::Forms::GroupBox
  ());
this->checkBox6 = (gcnew System::Windows::Forms::CheckBox
  ());
this->checkBox3 = (gcnew System::Windows::Forms::CheckBox
```

```
            this->numericUpDown40 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown39 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown38 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown37 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown36 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown35 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown34 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown33 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->groupBox9 = (gcnew System::Windows::Forms::GroupBox
              ());
            this->numericUpDown32 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown31 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown30 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->numericUpDown29 = (gcnew
              System::Windows::Forms::NumericUpDown());
            this->printDocument = (gcnew
              System::Drawing::Printing::PrintDocument());
            this->pageSetupDialog = (gcnew
              System::Windows::Forms::PageSetupDialog());
            this->printPreviewDialog = (gcnew
              System::Windows::Forms::PrintPreviewDialog());
            this->printDialog = (gcnew
              System::Windows::Forms::PrintDialog());
            this->groupBox1->SuspendLayout();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ
              e^>(this->pictureBox5))->BeginInit();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ
              e^>(this->pictureBox13))->BeginInit();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ
              e^>(this->pictureBox12))->BeginInit();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ
              e^>(this->pictureBox11))->BeginInit();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ
              e^>(this->pictureBox10))->BeginInit();

              (cli::safe_cast<System::ComponentModel::ISupportInitializ

              if(pageSetupDialog->ShowDialog() ==
                System::Windows::Forms::DialogResult::OK)
                 printPreviewDialog->ShowDialog();
        }
private: System::Void button4_Click(System::Object^  sender,
  System::EventArgs^  e) {
              if(printDialog->ShowDialog() ==
                System::Windows::Forms::DialogResult::OK)
                 printDocument->Print();
        }
};
}
```