УДК 004.896+004.5+004.942+004.67
КП
№ держреєстрації 0115U001569
Інв. №

Міністерство освіти і науки України
Сумський державний університет
(Сум ДУ)
40007, м.Суми, вул.Римського-Корсакова, 2
тел. (0542) 33 53 83; факс 33 40 58

ЗАТВЕРДЖУЮ
Проректор з наукової роботи,
д-р. физ.-мат. наук, професор
_____А. М. Чорноус
28.12.2017 р. М.П.

ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ
Моделі та інформаційні технології проектування і управління в складних системах

МОДЕЛІ ОЦІНКИ ЯКОСТІ ТА ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ ФУНКЦІЙ В АВТОМАТИЗОВАНИХ СИСТЕМАХ. РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ТА ФУНКЦІОНАЛЬНИХ ВИМОГ ДО КОМПЛЕКСУ ЗАСОБІВ АВТОМАТИЗАЦІЇ ПРОЕКТУВАЛЬНИХ РОБІТ. МОДЕЛІ ТА МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ В УМОВАХ НЕВИЗНАЧЕНОСТІ ПРИ УПРАВЛІННІ ЕНЕРГОЗАБЕЗПЕЧЕННЯМ

(проміжний)

Начальник НДЧ
канд. физ.-мат. наук, с. н. с _____ Д. І. Курбатов

Керівник НДР
канд. техн. наук _____ Е. Г. Кузнєцов

2017

Рукопис закінчено 27 грудня 2017 р.
Результати цієї роботи розглянуто науковою радою СумДУ,
протокол від 28.12.2017 № 4

СПИСОК АВТОРІВ

Керівник теми канд. техн. наук, старший викладач		Кузнєцов Е. Г. (розділ 1.2, п.п 2.2)
Відповідальний виконавець доктор техн. наук, професор		Лавров Є. А. (розділи 1.1-1.2, висновки)
Відповідальний виконавець канд. техн. наук, доцент		Неня В. Г. (розділи 2.1-2.2, висновки)
Відповідальний виконавець канд. техн. наук, доцент		Шендрик В. В. (розділи 3.1-3.9, висновки)
Канд. техн. наук, доцент		Алексенко О.В. (розділи 3.2-3.4)
Канд. техн. наук, доцент		Баранова І. В. (розділ 2.1)
Канд. техн. наук, доцент		Ващенко С. М. (розділ 2.2, розділи 3.1-3.3)
Канд. техн. наук, доцент		Гайдабрус Б. В. (розділ 2.2)
Канд. техн. наук, доцент		Концевич В. Г. (розділ 1.2)
Канд. техн. наук, доцент		Марченко А. В. (розділ 2.2, розділи 3.3-3.5)
Канд. техн. наук, старший викладач		Нагорний В. В. (розділ 2.2)
Канд. техн. наук, старший викладач		Парфененко Ю. В. (розділи 3.5-3.7)

Канд. техн. наук, старший викладач		Федотова Н. А. (розділи 3.4-3.6)
Канд. техн. наук, доцент		Чибіряк Я. І. (розділ 2.2)
Канд. техн. наук, асистент		Бойко О. В. (розділ 3.8)
Асистент		Захарченко В. П. (розділи 2.1, 2.2)
Канд. техн. наук, доцент, СНАУ		Пасько Н. Б. (розділ 1.2)
Старший викладач, СНАУ		Барченко Н. Л. (розділ 1.2)
Аспірант		Войцеховский Я. С. (розділи 1.1-1.2)
Аспірант		Омельченко Д. Є. (розділи 3.6-3.8)
Аспірант		Плакс Р.Д. (розділ 1.2)
Аспірант		Товкус О. І. (розділи 1.1-1.2)
Аспірант		Шендрик С. О. (розділи 3.6-3.8)
Студент, гр. ІТ-31		Антипенко Б. А. (розділ 2.2)
Студент, гр. ІТ.м-61		Бахмач М. В. (розділ 1.2)
Студент, гр. ІТ.м-61		Бичко Д. В. (розділ 3.1)
Студент, група ІТ-31		Єлісєєва А. Р. (розділ 3.3)
Студент, гр. ІТ.мз-62с		Коваленко Р. Ю. (розділ 2.2)

Студент, гр. ІТ-32		Ковтун А. А. (розділ 3.5)
Студент, гр. ІТ.м-61		Кошара В. С. (розділ1.2)
Студент, гр. ІТ.м-61		Криштоп М. О. (розділ 2.1)
Студент, гр. ІТ-51		Михайленко Ю. С. (розділ 3.1)
Студент, гр. ІТ-52		Нечепорук О. А. (розділ 3.1)
Студент, гр. ІТ.м-61		Рудакова Н. О. (розділ 1.2)
Студент, гр. ІТ-41		Федорова А. В. (розділ 1.2)
Студент, гр. ІТ.м-61		Шапочка Ю. С. (розділ 1.2)
Студент, гр. ІТ-51		Щербань Т. В.. (розділ 1.2)
Студент, гр. ІТ-52		Ясінська Т. А. (розділ 3.1)

РЕФЕРАТ

Звіт про НДР: 84 с., 7 табл., 13 рис., 50 джерела.

ІТ, ІС, АВТОМАТИЗАЦІЯ, ЛЮДИНО-МАШИННА ВЗАЄМОДІЯ, СИСТЕМА ІНФОРМАЦІЙНОЇ ПІДТРИМКИ.

Об'єкт дослідження: інформаційні технології та інформаційні системи.

Мета роботи: розробка теоретико-методологічних і науково-практичних основ розробки інформаційних технологій управління та інформаційних систем на потреб галузей суспільного виробництва та соціальної сфери.

Методи дослідження: системний та функціональний аналіз, дискретна математика та математична логіка.

Результатом роботи аналіз сучасного стану розвитку теорії складних систем та інформаційних технологій для автоматизації їх роботи та обґрунтування шляхів вирішення поставлених завдань.

Взаємозв'язок з іншими роботами: дана робота пов’язана із науковими дослідженнями аспірантів спеціальності інформаційні технології та студентів напряму комп’ютері науки.

Рекомендації по використанню результатів роботи: розробка науково-методичних основ за досліджуваними напрямками, формування тематики досліджень для магістрів та аспірантів.

Галузь застосування: промисловість, будівництво, міське господарство.

Значущість роботи і висновки: створює теоретично-методичну базу для автоматизації складних ситем та удосконалення людино-машинної взаємодії.

Прогнозні припущення про розвиток об’єкту дослідження: подальше поглиблення наукового обгрунтування дослідження та проектування складних систем та взаємодії їх з людиною-оператором.

ЗМІСТ

ВСТУП

Основні ідеї та принципи управління та проектування в складних системах виражені в системному підході. Для фахівця в області системотехніки вони є очевидними і природними, проте, їх дотримання і реалізація найчастіше пов'язані з певними труднощами, які зумовлені особливостями проектування. Як і більшість дорослих освічених людей, правильно використовують рідну мову без залучення правил граматики, розробники використовують системний підхід без звернення до посібників з системного аналізу. Однак інтуїтивний підхід без застосування правил системного аналізу може виявитися недостатнім для вирішення, задач інженерної діяльності, задач, що все більш ускладнюються. Основний загальний принцип системного підходу полягає в розгляді частин явища або складної системи з урахуванням їх взаємодії. Системний підхід виявляє структуру системи її внутрішні та зовнішні зв'язки.

Збільшення продуктивності праці розробників нових програмних і матеріальних об'єктів, скорочення термінів проектування, підвищення якості розробки проектів – найважливіші проблеми, вирішення яких визначає рівень прискорення науково-технічного прогресу суспільства. Розвиток систем автоматизованого проектування і управління спирається на міцну науково-технічну базу. Це, на сам перед, сучасні засоби обчислювальної техніки, нові способи подання та обробки інформації, створення нових чисельних методів розв'язання інженерних задач і оптимізації. Системи автоматизованого управління та проектування дають можливість на основі новітніх досягнень фундаментальних наук відпрацьовувати і удосконалювати методологію проектування, стимулювати розвиток математичної теорії проектування складних систем і об'єктів. В даний час створені і застосовуються в основному засоби і методи, що забезпечують автоматизацію рутинних процедур і операцій, таких, як підготовка управлінської документації, перетворення технічних креслень і моделей, отримання графічних зображень процесів і т.д.

Взаємодія підрозділів проектних та керуючих організацій з комплексом засобів автоматизації регламентується організаційним забезпеченням.

1 МОДЕЛІ ОЦИНКИ ЯКОСТІ ТА ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ ФУНКЦІЙ В АВТОМАТИЗОВАНИХ СИСТЕМАХ

1.1 Вступ

В якості основної методології оцінювання процесів (як показано в попередніх дослідженнях) доцільно використовувати методологію функціонально-структурну методологію ерготехнічних систем.

Для оцінки якості реалізації функцій автоматизованої системи, як показано в результаті виконання робіт з попереднього етапу необхідно розробити моделі автоматичного синтаксичного аналізу й редукції функціональної мережі (ФМ), що описує заданий процес функціонування.

1.2 Синтаксичний аналіз і редукція функціональних мереж, що описують процес реалізації функцій автоматизованої системи

Модель редукції ФМ. Формалізованим описом редукції ФМ назвемо сукупність оцінних операцій із множини M_4 (див. попередній етап) над елементами оцінювання разом з відповідними їм елементами опису, що задають позначення ТФЕ в структурі алгоритму, виявлені для згортання на поточному кроці, і елементами опису, що задають позначення еквівалентної ТФЕ:

$$\Pr_{FS} = < \{s_i, \{te_{ij}\} \mid j = 1,2,...m_{s_i}, Fs_i, o_{ee\,i}, \{y_{i_l}\} \mid l = 1,2,...,n_i \} \mid i = 1,2,...n >, \quad (1.1)$$

де s_i – номер кроку редукції;

n – кількість кроків редукції;

te_{ij} – позначення в структурі алгоритму, що звиваються на кроці si ТФЕ;

m_{s_i} – кількість, що згортаються на кроці s_i ТФЕ;

Fs_i – позначення ТФМ, виявленої в структурі алгоритму для згортання на кроці si;

o_{eei} – описовий елемент, що задає позначення еквівалентної ТФЕ;

$\{y_{i_l}\}$ – *l-я* оцінна операція на *i-му* кроці редукції, застосовувана для визначення *l-го* показника якості еквівалентної ТФЕ, $\{y_{i_m}\} \in M_4$.

Опис ТФМ і узагальненої структури алгоритму розробленим способом забезпечує можливість їх порівняння, що дозволяє виявляти ТФМ у структурі алгоритму й заміняти їх на еквівалентні ТФЕ.

Справедливо наступне твердження 1: «ТФМ буде виявлена в структурі алгоритму на одному із кроків редукції, якщо в структурі алгоритму існує послідовність ТФЕ, аналогічна по складу ТФЕ даної ТФМ, а всі значення елементів зв'язки, що описують дані ТФЕ в структурі алгоритму, відрізняються від відповідних значень елементів зв'язку, що описують такі ж ТФЕ в ТФМ, на те саме число, рівне номеру першої ТФЕ, виявленої в структурі алгоритму, зменшене на одиницю».

Функціональна структура Fs_i буде виявлена в структурі алгоритму на одному із кроків редукції, якщо $\exists$ r таке, що $1<r<n$ і $\exists\, O_{FS_r} \in O_{FS}$, і $O_{FS_r} ::=<\{o_{e_h}, te_h, N_h, \{V_{hl}, L_{hl}\} | l = 1,2,...\eta_h, [k_{c_h}]\} | h = r, r+1,..., r+k_i-1>$, і для $\forall j = 1,2,...,z_i$ виконується: $o_{e_{ij}} = o_{e_{r+j-1}}$; $N_{ij} = N_{r+j-1} - r + 1$; $\eta_{i_j} = \eta_{r+j-1}$; $V_{ij_l} = V_{(r+j-1)l}$; $L_{ij_l} = L_{(r+j-1)l} - r + 1$.

Сформульоване твердження слугувало основою для розробки правил виявлення основних ТФМ в узагальненій структурі алгоритму й редукції ФМ. Процес зменшення розмірності ФМ зводиться до маніпулювання рядковими описами ФМ, ТФМ і ТФЕ. Складемо укрупнений алгоритм згортання функціональної мережі:

1. Представити алгоритм діяльності оператора у вигляді ФМ;

2. Описати сконструйовану мережу у табличному (рядковому) виді з іменем O_{FS}: $O_{FS} =<\{o_{e_j}, te_j, N_j, \{V_{j_l}, L_{j_l}\} | l = 1,2,...\eta_j, [k_{c_j}]\} | \ j = 1,2,...n>$;

3. Привласнити номеру підстановки, що укрупнює, і лічильнику ТФМ значення 1*: i*=1*; m*=1;

4. Створити тимчасовий опис ФМ із іменем O_{FS_1} у табличному (рядковому) виді. Вибрати в описі заданої ФМ рядка, що описують послідовне виконання робочих операцій, і помістити їх в опис ФМ із іменем O_{FS_1};

5. В описі ФМ із іменем O_{FS_1} кожну групу рядків, що описують послідовні робочі операції, замінити на рядок еквівалентної ТФЕ;

6. Створити опис протоколу редукції PR_{FS} мовою опису ФМ по формулі (1.1) і помістити рядка опису протоколу згортання кроку $i=1$. У вхідному описі ФМ O_{FS} обрані рядки, що описують послідовне виконання робочих операцій, замінити на рядки еквівалентних їм ТФЕ. Вилучити всі рядки з опису O_{FS_1};

7. Перевірити кількість рядків в описі заданої ФМ із іменем O_{FS}, *що* описують ТФЕ: *k=кількість ТФЕ*;

8. Якщо $k=1$, перейти до 24, інакше перейти до 9;

9. Збільшити лічильник ТФМ на 1: $m=m+1$;

10. В описі ФМ із іменем O_{FS} вибрати послідовності рядків, що представляють *m*-у ТФМ і помістити в опис ФМ із іменем O_{FS_1};

11. Якщо кількість рядків в описі ФМ із іменем O_{FS_1} порожньо, то перейти до 19, якщо немає – замінити в O_{FS_1} рядка, що описують *m-е* ТФМ, на рядки, що описують еквівалентні їм ТФЕ;

12. Збільшити крок згортання на 1: $i=i+1$. В опис протоколу редукції PR_{FS} додати записи протоколу згортання *m*-х ТФМ;

13. В описі ФМ O_{FS} записи, що описують обрані *m*-е ТФМ, замінити на записі з опису O_{FS_1}, *що* представляють еквівалентні їм ТФЕ;

14. Вилучити всі рядки з опису O_{FS_1};

15. Вибрати в описі ФМ O_{FS} рядки, що представляють послідовності робітників ТФЕ, і помістити їх в опис ФМ із іменем O_{FS_1};

16. Якщо кількість рядків в описі ФМ із іменем O_{FS_1} порожньо, то перейти до 19, інакше – замінити записи, що представляють послідовності робітників ТФЕ, на записі еквівалентних їм ТФЕ;

17.Збільшити крок згортання на 1: $i=i+1$. В опис протоколу редукції PR_{FS} додати записи, що описують протокол згортання послідовностей робітників ТФЕ;

18. У вихідному описі ФМ із іменем O_{FS} обрані рядки, що описують послідовності робочих операцій, замінити на рядки, що описують їм еквівалентні ТФЕ. Вилучити всі рядки з опису O_{FS_1};

19. Перевірити кількість рядків в описі заданої ФМ O_{FS}, *що* описують ТФЕ: *k=кількість ТФЕ.* Якщо $k=1$, перейти до 24, інакше перейти до 20;

20. Перевірити умову " Є ще ТФМ?". Якщо "Так", то збільшити лічильник ТФМ на одиницю: $m=m+1$, інакше перейти до кроку 22;

21. Перейти до кроку 10;

22. Перевірити кількість рядків в описі заданої ФМ із іменем O_{FS};

23. Якщо $k>1$, перевірити опис ФМ;

24. Кінець згортання ФМ.

Вимагають детального розгляду пункти зазначеного алгоритму, що стосуються виявлення скорочених фрагментів ФМ, тобто ТФМ, і заміни їх на еквівалентні ТФЕ.

Визначимо в термінах мови опису ФМ синтаксис для операцій з рядковими (табличними) описами ФМ, ТФЕ, ТФМ, протоколів редукції, який будемо використовувати надалі.

Основною синтаксичною одиницею є вираження, що робить опис ФМ, ТФМ, ТФЕ або протоколу редукції мовою опису ФМ. Будь-яке вираження є похідним описом. Розкриємо послідовно суть вираження і його операндів:

вираження ::= унарне-*вираження* | бінарне-*вираження*;
унарное-вираження ::= *перейменування* | *вибірка* | *проекція*;
перейменування ::= *терм* RENAME *атрибут* AS *атрибут*;
терм ::= *опис* | (*вираження*);
вибірка ::= *терм* WHERE *предикат*;
предикат ::= *порівняння* | *предикат* AND *предикат* |
предикат OR *предикат* | NOT *предикат*;

порівняння ::= *символ* θ *символ*;

символ ::= *атрибут* | *константа;*

θ ::= < | > | =|>=|<=;

проекція ::= *терм* | *терм* [*список*];

бінарне-вираження ::= *проекція*| *бінарна-операція;*

бінарна-операція ::= *UNION* | *INTERSECT* | *MINUS* | *TIMES* |*EKVIVALENT*.

Тут: *атрибут* – ідентифікатор змінної, елемента групи мови опису ФМ; *список* – список розділених комами атрибутів; *константа* – літеральне значення; *UNION* – об'єднання двох описів; *INTERSECT* – перетинання двох описів; *MINUS* – різниця двох описів; *TIMES* – вибірка на дві або декількох описах; *EKVIVALENT* – заміна опису ТФМ на опис еквівалентної ТФЕ. Основні позначення бінарних операцій аналогічні позначенням операцій над відносинами в реляційній алгебрі.

Позначимо опис у загальному виді (тобто, для ФМ, або для ТФМ, або для ТФЕ, або для протоколу редукції) символом «*R*», рядок опису в загальному виді – *r*.

Уведемо поняття сумісних описів: два описи будуть сумісні, якщо вони мають однаковий склад елементів у рядку опису.

Крім зазначених позначень бінарних операцій уведемо позначення цих операцій у функціональному виді й визначимо суть кожної операції.

Нехай задано два сумісні описи R_1 і R_2.

Об'єднанням *Union* двох сумісних описів R_1 і R_2 буде опис *R*, яке містить сукупність рядків заданих описів, причому, першими в описі *R* вказуються рядки опису R_1:

$$R=R1\ Union\ R2::=\left\{r \mid\ r \in R_1 \vee r \in R_2\right\}.$$

Позначення операції *Union* у функціональному виді:

$R_1\ Union\ R_2::=Union(R_1, R_2)$.

Перетинання *Intersect* двох сумісних описів R_1 і R_2 буде опис *R*, яке містить множина рядків, що належать одночасно опису R_1 і опису R_2:

$R=R_1$ *Intersect* $R_2::=\{r \mid r \in R_1 \wedge r \in R_2\}$.

Позначення операції *Intersect* у функціональному виді:

R_1 *Intersect* $R_2::=$*Intersect* (R_1, R_2).

Різниця *Minus* двох сумісних описів R_1 і R_2 буде опис R, яке містить множина рядків, що належать опису R_1 і не приналежних опису R_2:

$R= R_1$ *Minus* $R_2::=\{r \mid r \in R_1 \wedge r \notin R_2\}$.

Позначення операції *Minus* у функціональному виді:

R_1 *Minus* $R_2::=$*Minus* (R_1, R_2).

Вибором *Where* на описі R_1 за умовою $\alpha(r)$ буде опис R, рядка r належати опису R_1, і умова $\alpha(r)$ ухвалює значення *«Істина»*:

$R= R_1$ *Where* $\alpha::=\{r \mid r \in R_1 \wedge \alpha(r)=$"*Истина* "$\}$.

Позначення операції *Where* у функціональному виді:

R_1 *Where* $\alpha::=$ *Where*(R_1, α).

Операцію вибору із двох (або декількох) описів R_1, R_2 за умовою α позначимо через *Times*. У цьому випадку функціональне вираження вибору буде мати вигляд:

(R_1 *Times* R_2) *where* $\alpha::=$*Timeswhere*(R_1, R_2, α) – результат вибору з описів R_1 і R_2 за умовою α. Умова α містить операції порівняння атрибутів описів R_1 і R_2 між собою, з константами й може бути як завгодно складним.

У тих операціях, де об'єднання описів, як результат вибору, здійснюється саме із собою, або один опис з'єднується двічі з іншим описом, використовуємо псевдоніми (аліаси), які дозволяють розрізняти копії, що з'єднуються, описів. Псевдоніми вказуються після назви опису через символ "пробіл":

$R=$ *Timeswhere*(R_1 P_1, R_1 P_2 ...,R_1 *Pn*, α) – опис R є результатом вибору з опису R_1 за умови α, яке містить операції порівняння різних атрибутів між собою.

Моделі синтаксичного аналізатора ФМ. Процес редукції ФМ полягає в тому, що виділення на кожному кроці редукції в структурі алгоритму ТФМ і заміна їх на еквівалентні ТФЕ триває доти, поки вся ФМ не буде зведена до

одній еквівалентної ТФЕ, якщо це можливо. Тобто, на кожному кроці редукції структура АФ характеризується певним станом. Уведемо позначення:

k – крок редукції, $k \in N$;

$f_k = f(k)$ – стан функціональної мережі на k-*ом* – кроці редукції;

G_f – множина станів функціональної мережі в процесі згортання, $f_k \in G_f$;

$k_0 = 0$ – споконвічний крок редукції;

$f_0 = f(k_0)$ – споконвічний стан функціональної мережі, $f_0 \in G_f$;

G_α – множина правил розпізнання скорочених фрагментів функціональної мережі;

$\alpha_i \in G_\alpha$ – правило виявлення в структурі алгоритму і-*й* ТФМ для згортання;

$u_k = u(k)$ – протокол згортання на k-*му* кроці редукції.

Синтаксичний аналізатор ФМ (*САФМ*) – це математична структура, яка представляється перехідним відображенням $R:(k;k_0,f,\alpha) \rightarrow f(k)$ для стану структури алгоритму функціонування $f(k) \in G_f$, досягнутого на кроці редукції $k \in N$ при застосуванні правил виявлення ТФМ $\alpha \in G\alpha$, якщо на вихідному кроці редукції k_0=0 вихідний стан структури алгоритму функціонування був $f_0 = f(k_0)$, $k_0 \in N$, а також відображенням виходу – протоколу редукції $u(k) = \eta(k,f(k))$. Відображення R і η задаються наборами правил заміни ТФМ на еквівалентні ТФЕ й математичних моделей визначення показників якості виконання еквівалентних ТФЕ. За допомогою *САФМ* можна представити співвідношення зміни на кроці редукції стану ФМ залежно від вихідних правил розпізнавання ТФМ:

$$f_k \xrightarrow{\bar{a}, \bar{Y}} f_{k+1}, \qquad (1.2)$$

де f_k (fk_{+1}) – стан ФМ на k-*м* (k+ *1*-*м*) кроці редукції;

$\bar{a} = (\alpha_1, \alpha_2, \ldots, \alpha_{Kfs})$ – узагальнений вектор правил розпізнавання ТФМ у структурі АФ на k-*му* кроці редукції,

α_i – правило виявлення в структурі алгоритму і-*й* ТФМ для згортання, *Kfs*=18 – кількість ТФМ.

Правила розпізнавання ТФМ задаються предикатами, операнди яких належать множинам елементів опису й операцій описи й використовуються при описі ФМ.

З метою апроксимації моделі (1.2), і враховуючи, що стан ФМ f_k представляється в рядковому виді, відображення $R:f_k,\alpha_k \rightarrow f_{k+1}$ досліджуємо як послідовність виражень, що приводять рядковий опис, що відповідає стану ФМ f_k (позначимо опис через O_{FS_k}), до рядкового опису, що відповідає стану ФМ f_{k+1} (позначимо опис через O_{FS_k+1}). Аналогічно, через PR_{FS_k} позначимо рядковий опис (формула 1.1), що містить протокол редукції, що відповідає *k*–му кроку редукції, і через PR_{FS_k+1} – протокол редукції, що відповідає *k+1-му* кроку редукції. Тоді редукція ФМ може бути подана у вигляді послідовності ітераційних формул, застосованих до описів. Для викладу використовуємо прийняті вище позначення й синтаксис.

Опису O_{FS_1} і O_{FS_E}, що мають однаковий склад елементів у рядку опису, що й опис O_{FS_k}, беруться для вибору рядків, що представляють ТФМ, з відносини O_{FS_k} і для вистави еквівалентних ТФЕ, відповідно. Розпізнавання й вибір скорочених фрагментів ФМ і-*й* ТФМ виконується функцією $Timeswhere(O_{FS\_k},...,\alpha_i)$. Тут правила виявлення скорочених фрагментів ФМ відповідають твердженню 1 і задаються умовою α_i. Заміна фрагментів, що редуцюються, функціональної мережі ТФМ на еквівалентні ТФЕ здійснюється послідовним виконанням операцій *Minus* і *Union* над описами O_{FS_k+1}, O_{FS_1} і O_{FS_E}. Опис O_{FS_E} формується на основі опису O_{FS_1}, певного виконанням функції $Timeswhere(O_{FS\_k},...,\alpha i)$. При цьому кожної ТФМ, представленої в OFS_1, відповідає один запис опису O_{FS_E}. Значення атрибутів цьому запису, що визначають показники якості виконання еквівалентної ТФЕ, розраховуються, у випадку бінарних помилок, по відомих моделях оцінки надійності процесів функціонування ЧМС, і по формулах, виведених у п.1.3. – у випадку декількох типів помилок. Для відображення протоколу редукції береться опис PR_{FS_k}, яке

крім атрибутів, значеннями яких є показники якості виконання еквівалентних ТФЕ, містить атрибути для значень кроку редукції, позначення еквівалентної ТФЕ й переліку ТФЕ, що редуцюються. Таким чином, формула (1.2) апроксимується як послідовність ітераційних формул над зазначеними описами:

$$\begin{aligned} O_{FS\_1} &= Timeswhere(O_{FS\_k}\ o_1, O_{FS\_k}\ o_2, \ldots, \alpha_i); \\ O_{FS\_k+1} &= Minus(O_{FS\_k}, O_{FS\_1}); \\ O_{FS\_E} &= Ekvivalent(O_{FS\_1}); \\ O_{FS\_k+1} &= Union(O_{FS\_k+1}, O_{FS\_E}); \\ PR_{FS\_k+1} &= Union(PR_{FS\_k}, O_{FS\_E}). \end{aligned} \quad (1.3)$$

Символ три крапки в першій формулі означає, що число «аліасов» залежить від типу розпізнаваної ТФМ у структурі алгоритму.

Задамо правила формування опису O_{FS_1} для основних ТФМ.

1. Розпізнавання ТФМ «послідовне виконання робочих операцій» –*Fsrr*:

$$O_{FS\_1} = Timeswhere(O_{FS\_k}\ o_1, O_{FS\_k}\ o_2, \alpha_1), \quad (1.4)$$

де

$$a_1 = ((o_1 o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge (o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 1)) \vee$$
$$((o_1.o_{e_j} = "R" \wedge o_1.L_{j_1} = o_1.N_j + 1) \wedge (o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_2.N_j + 1)).$$

2. Розпізнавання ТФМ «робоча операція з контролем функціонування без обмеження на кількість циклів» – *Fsrk*:

$$OFS\_1 = Timeswhere(OFS\_k, npo_1, OFS\_k, o_2, \alpha_2), \quad (1.5)$$

де

$$a_2 = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge$$
$$(o_2.o_{e_j} = "K" \wedge o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 1 \wedge o_2.L_{j_2} = o_1.N_j)) \vee$$
$$((o_1.o_{e_j} = "K" \wedge o_1.L_{j_1} = o_1.N_j + 1 \wedge o_1.L_{j_2} = o_1.N_j - 1) \wedge$$
$$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.L_{j_2} \wedge o_2.L_{j_1} = o_2.N_j + 1)).$$

3. Розпізнавання ТФМ «робоча операція з контролем функціонування, доробкою й повторенням робочої – Fsrkr :

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, O_{FS\_k}, o_3, \alpha_3), \qquad (1.6)$$

де: $a_3 = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge (o_2.o_{e_j} = "K" \wedge$
$o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 2 \wedge o_2.L_{j_2} = o_2.N_j + 1 \wedge$
$o_3.o_{e_j} = "R" \wedge o_3.N_j = o_2.L_{j_2} \wedge o_3.L_{j_1} = o_1.N_j)) \vee$
$((o_1.o_{e_j} = "K" \wedge o_1.L_{j_1} = o_1.N_j + 2 \wedge o_1.L_{j_2} = o_1.N_j + 1) \wedge$
$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_1.N_j) \wedge$
$(o_3.o_{e_j} = "R" \wedge o_3.N_j = o_1.N_j + 1 \wedge o_3.N_j = o_1.L_{j_2} \wedge o_3.L_{j_1} = o_2.N_j)) \vee$
$((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_3.N_j + 1 \wedge o_1.L_{j_1} = o_2.N_j) \wedge$
$(o_3.o_{e_j} = "K" \wedge o_3.L_{j_2} = o_1.N_j \wedge o_3.L_{j_1} = o_1.N_j + 1) \wedge$
$o_2.o_{e_j} = "R" \wedge o_2.L_{j_1} = o_3.N_j \wedge o_2.N_j + 2 = o_1.N_j \wedge o_2.N_j + 1 = o_3.N_j)).$

4. Розпізнавання ТФМ «робоча операція з контролем працездатності без обмеження на кількість циклів» – Fspr :

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, \alpha_4), \qquad (1.7)$$

де: $a_4 = ((o_1.o_{e_j} = "P" \wedge o_1.N_j = o_1.L_{j_3} - 1 \wedge o_1.L_{j_1} = o_1.N_j + 2) \wedge$
$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.L_{j_3} \wedge o_2.L_{j_1} = o_1.N_j)) \vee$
$((o_1.o_{e_j} = "R" \wedge o_1.L_{j_1} = o_1.N_j - 1 \wedge o_2.N_j = o_1.N_j - 1) \wedge$
$(o_2.o_{e_j} = "P" \wedge o_2.N_j = o_1.L_{j_1} \wedge o_2.L_{j_1} = o_2.N_j + 2 \wedge o_2.L_{j_3} = o_1.N_j)).$

5. Розпізнавання ТФМ « n-кратне повторення робочої операції із прийманням по всім успішним результатам – Fs_{CRF} :

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, \alpha_5), \quad (1.8)$$

де: $a_5 = ((o_1.o_{e_j} = "R" \wedge o_1.L_{j_1} = o_1.N_j + 1 \wedge o_2.o_{e_j} = "C_F" \wedge o_1.N_j = o_2.N_j - 1 \wedge$
$o_2.L_{j_4} = o_1.N_j \wedge o_2.L_{j_5} = o_2.N_j + 1) \vee (o_1.o_{e_j} = "C_F" \wedge o_1.L_{j_5} = o_1.N_j + 1 \wedge$
$o_1.L_{j_4} = o_1.N_j - 1 \wedge o_2.o_{e_j} = "R" \wedge o_2.L_{j_1} = o_2.N_j + 1 \wedge o_2.L_{j_1} = o_1.N_j)).$

6. Розпізнавання ТФМ «n-кратне повторення робочої операції із прийманням при наявності хоча б одного успішного результату» – Fs_{CRO}:

$$OFS\_1 = Timeswhere(OFS\_k, npo_1, OFS\_k, o_2, \alpha_6), \quad (1.9)$$

де: $a_6 = ((o_1.o_{e_j} = "R" \wedge o_1.L_{j_1} = o_1.N_j + 1 \wedge o_2.o_{e_j} = "C_O" \wedge o_1.N_j = o_2.N_j - 1 \wedge$
$o_2.L_{j_4} = o_1.N_j \wedge o_2.L_{j_5} = o_2.N_j + 1) \vee (o_1.o_{e_j} = "C_O" \wedge o_1.L_{j_5} = o_1.N_j + 1 \wedge$
$o_1.L_{j_4} = o_1.N_j - 1 \wedge o_2.o_{e_j} = "R" \wedge o_2.L_{j_1} = o_2.N_j + 1 \wedge o_2.L_{j_1} = o_1.N_j)).$

7. Розпізнавання ТФМ «послідовне виконання робочої операції й операції контролю» – Fsrk1:

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, \alpha_7), \quad (1.10)$$

де: $a_7 = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge$
$(o_2.o_{e_j} = "K" \wedge o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 1 \wedge o_2.L_{j_2} > o_2.N_j + 1)) \vee$
$((o_1.o_{e_j} = "K" \wedge o_1.L_{j_1} = o_1.N_j + 1 \wedge o_1.L_{j_2} > o_1.N_j + 1) \wedge$
$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_2.N_j + 1)).$

8. Розпізнавання ТФМ «послідовне виконання робочої операції й операції контролю працездатності» – Fsrp1 :

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, \alpha_8), \quad (1.11)$$

де: $a_8 = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge$

$(o_2.o_{e_j} = "P" \wedge o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 1 \wedge o_2.L_{j_2} > o_2.N_j + 1)) \vee$

$((o_1.o_{e_j} = "P" \wedge o_1.L_{j_1} = o_1.N_j + 1 \wedge o_1.L_{j_2} > o_1.N_j + 1) \wedge$

$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_2.N_j + 1)).$

9. Розпізнавання ТФМ «робоча операція з контролем функціонування й виправленням помилки без циклів» – Fs_{RKR1}:

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, O_{FS\_k}, o_3, \alpha_9), \qquad (1.12)$$

де: $a_9 = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge (o_2.o_{e_j} = "K" \wedge$

$o_2.N_j = o_1.N_j + 1 \wedge o_2.L_{j_1} = o_2.N_j + 2 \wedge o_2.L_{j_2} = o_2.N_j + 1 \wedge$

$o_3.o_{e_j} = "R" \wedge o_3.N_j = o_2.L_{j_2} \wedge o_3.L_{j_1} = o_3.N_j + 1)) \vee$

$((o_1.o_{e_j} = "K" \wedge o_1.L_{j_1} = o_1.N_j + 2 \wedge o_1.L_{j_2} = o_1.N_j + 1) \wedge$

$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_1.N_j) \wedge$

$(o_3.o_{e_j} = "R" \wedge o_3.N_j = o_1.N_j + 1 \wedge o_3.N_j = o_1.L_{j_2} \wedge o_3.L_{j_1} = o_3.N_j + 1)) \vee$

$((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_3.N_j + 1 \wedge o_1.L_{j_1} = o_1.N_j + 1) \wedge$

$(o_3.o_{e_j} = "K" \wedge o_3.L_{j_2} = o_1.N_j \wedge o_3.L_{j_1} = o_1.N_j + 1) \wedge$

$o_2.o_{e_j} = "R" \wedge o_2.L_{j_1} = o_3.N_j \wedge o_2.N_j + 2 = o_1.N_j \wedge o_2.N_j + 1 = o_3.N_j)).$

10. Розпізнавання ТФМ «робоча операція з контролем функціонування, доробкою й повторенням контролю робочої операції без обмеження на кількість циклів» - Fsrkrk :

$$O_{FS\_1} = Timeswhere(O_{FS\_k}, o_1, O_{FS\_k}, o_2, O_{FS\_k}, o_3, \alpha_{10}), \qquad (1.13)$$

де

$$a_{10} = ((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_1.L_{j_1} - 1) \wedge (o_2.o_{e_j} = "K" \wedge o_2.N_j = o_1.N_j + 1 \wedge$$
$$o_2.L_{j_1} = o_2.N_j + 2 \wedge o_2.L_{j_2} = o_2.N_j + 1 \wedge$$
$$o_3.o_{e_j} = "R" \wedge o_3.N_j = o_2.L_{j_2} \wedge o_3.L_{j_1} = o_2.N_j)) \vee$$
$$((o_1.o_{e_j} = "K" \wedge o_1.L_{j_1} = o_1.N_j + 2 \wedge o_1.L_{j_2} = o_1.N_j + 1) \wedge$$
$$(o_2.o_{e_j} = "R" \wedge o_2.N_j = o_1.N_j - 1 \wedge o_2.L_{j_1} = o_1.N_j) \wedge$$
$$(o_3.o_{e_j} = "R" \wedge o_3.N_j = o_1.N_j + 1 \wedge o_3.N_j = o_1.L_{j_2} \wedge o_3.L_{j_1} = o_1.N_j)) \vee$$
$$((o_1.o_{e_j} = "R" \wedge o_1.N_j = o_3.N_j + 1 \wedge o_1.L_{j_1} = o_2.N_j) \wedge$$
$$(o_3.o_{e_j} = "K" \wedge o_3.L_{j_2} = o_1.N_j \wedge o_3.L_{j_1} = o_1.N_j + 1) \wedge$$
$$(o_2.o_{e_j} = "R" \wedge o_2.L_{j_1} = o_3.N_j \wedge o_2.N_j + 2 = o_1.N_j \wedge o_2.N_j + 1 = o_3.N_j)).$$

У формулах (1.4) – (1.13) використані псевдоніми для опису O_{FS_k} позначені через "o_1", "o_2", "o_3".

1.3. Висновки

Розроблені математичні моделі є основою для автоматичного розпізнавання типових функціональних структур і дозволяють реалізувати технологію автоматичної редукції функціональної мережі, що описує логікові процесу функціонування автоматизованої системи.

2 РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ТА ФУНКЦІОНАЛЬНИХ ВИМОГ ДО КОМПЛЕКСУ ЗАСОБІВ АВТОМАТИЗАЦІЇ ПРОЕКТУВАЛЬНИХ РОБІТ

2.1 Розробка інформаційного забезпечення КЗАПР

Для розробки інформаційного забезпечення комплексу засобів автоматизації проектувальних робіт (КЗАПР) запропоновано інформаційну модель комплексу у вигляді сукупності множин [1], які описують окремі складові (компоненти) комплексу (об’єкти та суб’єкти). Таких множин, на основі яких між описуваними складовими встановлено зв’язки, запропоновано шістнадцять.

На основі введених до розгляду множин та зв'язків було створено реляційну базу даних [2 – 4], яка складається з 16 наступних взаємопов’язаних таблиць:

1. Contract.
2. ToR.
3. Project.
4. ObjectStructure.
5. Status.
6. TaskType.
7. ModelsRepository.
8. Specialization.
9. Software.
10. Department.
11. Position.
12. absp_users.
13. Expert.
14. ExpertToSpecialization.
15. ManufacturingOperation.
16. LogBook.

Таблиця «*Contract*» містить дані про контракти, які уклала проектна організація. Вона має наступні поля:

– *id* – ідентифікатор контракту;

– *NameClient* – назву юридичної або фізичної особи, з якою проектна організація уклала контракт;

– *DataContractStarts* – дата укладення контракту;

– *DataContractFinishes* – запланована дата завершення дії контракту;

– *Budget* – бюджет, який виділено на виконання контракту;

– *Completed* – логічне поле, яке показує чи завершений контракт.

Таблиця «*ToR*» містить дані про технічні завдання, які було розроблено згідно існуючим контрактам проектної організації, для виконання їх відповідних вимог. Вона має наступні поля:

– *id* – ідентифікатор технічного завдання;

– *idContract* – ідентифікатор контракту, до якого відноситься технічне завдання;

– *FileName* – посилання на директорію, де знаходиться файл, у якому описане технічне завдання на виконання відповідного контракту.

Таблиця «*Project*» містить дані про проекти, які були створені згідно існуючим технічним завданням проектної організації, для їх реалізації. Вона має наступні поля:

– *id* – ідентифікатор проекту;

– *idToR* – ідентифікатор технічного завдання, за яким ініціюється проект;

– *Name* – назва проекту.

Таблиця «*ObjectStructure*» містить дані про структуру об’єктів, які проектуються. Вона має наступні поля:

– *id* – ідентифікатор елементу структури проектованого об’єкту;

– *idProject* – ідентифікатор проекту, за яким розроблюється елементи структури проектованого об’єкту;

– *Name* – назва елементу структури проектованого об’єкту;

– *idPObjectStructure* – ідентифікатор батьківського елементу структури проектованого об’єкту, до складу якого відноситься розроблюваний елемент структури проектованого об’єкту;

– *Component* – позначка належності елементу структури проектованого об’єкту до складу елементу структури більш високого рівня.

Таблиця «*Status*» містить дані про статуси виробничих завдань. Вона має наступні поля:

– *id* – ідентифікатор статусу;

– *Name* – назва статусу;

– *Description* – опис статусу.

Таблиця «*TaskType*» містить дані вид проектувальних робіт. Вона має наступні поля:

– *id* – ідентифікатор виду проектувальних робіт;

– *Name* – назва виду проектувальних робіт.

Таблиця «*ModelsRepository*» містить дані про шаблони проектних документів та готові проектні рішення. Вона має наступні поля:

– *id* – ідентифікатор елементу репозитарію;

– *Name* – назва елементу репозитарію;

– *Description* – опис елементу репозитарію;

– *idTaskType* – ідентифікатор виду проектувальних робіт, якому відповідає елементу репозитарію;

– *FileName* – посилання на директорію, де знаходиться файл шаблону проектного документу або готового проектного рішення.

Таблиця «*Specialization*» містить дані про спеціалізації фахівців. Вона має наступні поля:

– *id* – ідентифікатор спеціалізації;

– *Name* – назва спеціалізації;

– *idTaskType* – ідентифікатор виду проектувальних робіт, до якого відноситься спеціалізація.

Таблиця «*Software*» містить дані про програмне забезпечення. Вона має наступні поля:

– *id* – ідентифікатор програмного забезпечення;

– *idSpecialization* – ідентифікатор спеціалізації, до якої відноситься програмне забезпечення;

– *Name* – назва програмного забезпечення.

Таблиця «*Department*» містить дані про відділи проектної організації. Вона має наступні поля:

– *id* – ідентифікатор відділу;

– *Name* – назва відділу;

– *idTaskType* – ідентифікатор виду проектувальних робіт, на якому спеціалізується відділ.

Таблиця «*Position*» містить дані про посади фахівців проектної організації. Вона має наступні поля:

– *id* – ідентифікатор посади;

– *Name* – назва посади;

– *Salary* – посадовий оклад.

Таблиця «*absp_users*» містить унікальні дані про зареєстрованих користувачів. Вона має наступні поля:

– *id* – ідентифікатор облікового запису;

– *name* – ПІБ користувача;

– *username* – логін користувача;

– *email* – електронна пошта користувача;

– *password* – пароль користувача;

– *usertype* – логічне поле, яке показує чи є користувач адміністратором;

– *block* – логічне поле, яке показує чи є користувач заблокований;

– *registerDate* – дата реєстрації користувача;

– *lastvisitDate* – дата останнього сеансу входу користувача;

– *activation* – логічне поле, яке показує чи активований обліковий запис;

– *params* – набір прав користувача;

– *lastResetTime* – зарезервоване поле для подальшого розвитку;

– *resetCount* – зарезервоване поле для подальшого розвитку.

Таблиця «*Expert*» містить дані про фахівців проектної організації. Вона має наступні поля:

– *id* – ідентифікатор фахівця;

– *idUsers* – ідентифікатор користувача, якому відповідає фахівець;

– *idPosition* – ідентифікатор посади, яку займає фахівець;

– *idDepartment* – ідентифікатор відділу, у якому працює фахівець.

Таблиця «*ExpertToSpecialization*» містить дані про профільні та непрофільні спеціалізації фахівців проектної організації. Вона має наступні поля:

– *id* – ідентифікатор запису;

– *idSpecialization* – ідентифікатор спеціалізації;

– *idExpert* – ідентифікатор фахівця;

– *MainTask* – логічне поле, яке показує чи є спеціалізація профільною у фахівця.

Таблиця «*ManufacturingOperation*» містить дані про виробничі операції. Вона має наступні поля:

– *id* – ідентифікатор виробничої операції;

– *idObjectStructure* – ідентифікатор елементу структури проектованого об’єкту

– *idTaskType* – ідентифікатор виду проектувальних робіт, до якого відноситься виробнича операція;

– *idExpert* – ідентифікатор фахівця, якого призначено на виконання виробничої операції;

– *DateManufacturingOperationStarts* – запланована дата початку виконання виробничої операції;

– *DateManufacturingOperationFinishes* – запланована дата закінчення виконання виробничої операції;

– *idSoftware* – ідентифікатор програмного забезпечення, яке використовується для виконання виробничої операції;

– *idModels* – ідентифікатор елементу репозитарію, який використовується для виконання виробничої операції;

– *idStatus* – ідентифікатор статусу виробничої операції;

– *urgency* – зарезервоване поле для подальшого розвитку.

Таблиця «*LogBook*» містить дані про журнал обліку. Вона має наступні поля:

– *id* – ідентифікатор запису;

– *idManufacturingOperation* – ідентифікатор виробничої операції;

– *idExpert* – ідентифікатор фахівця, який виконує виробничу операцію;

– *IP* – адреса комп'ютера в мережі, на якому фахівцем виконується виробнича операція;

– *TimeStart* – час початку сеансу виконання користувачем виробничої операції;

– *TimeFinish* – час закінчення сеансу виконання користувачем виробничої операції;

– *TimeManufacturingOperation* – загальний час виконання користувачем виробничої операції.

У результаті створено базу даних «КЗАПРБД» до комплексу засобів автоматизації проектувальних робіт для організації основного ієрархічного сховища даних, які необхідні для інформаційного забезпечення належного функціонування комплексу засобів автоматизації проектувальних робіт.

Схема даних розробленої бази даних представлена на рисунку 2.1.

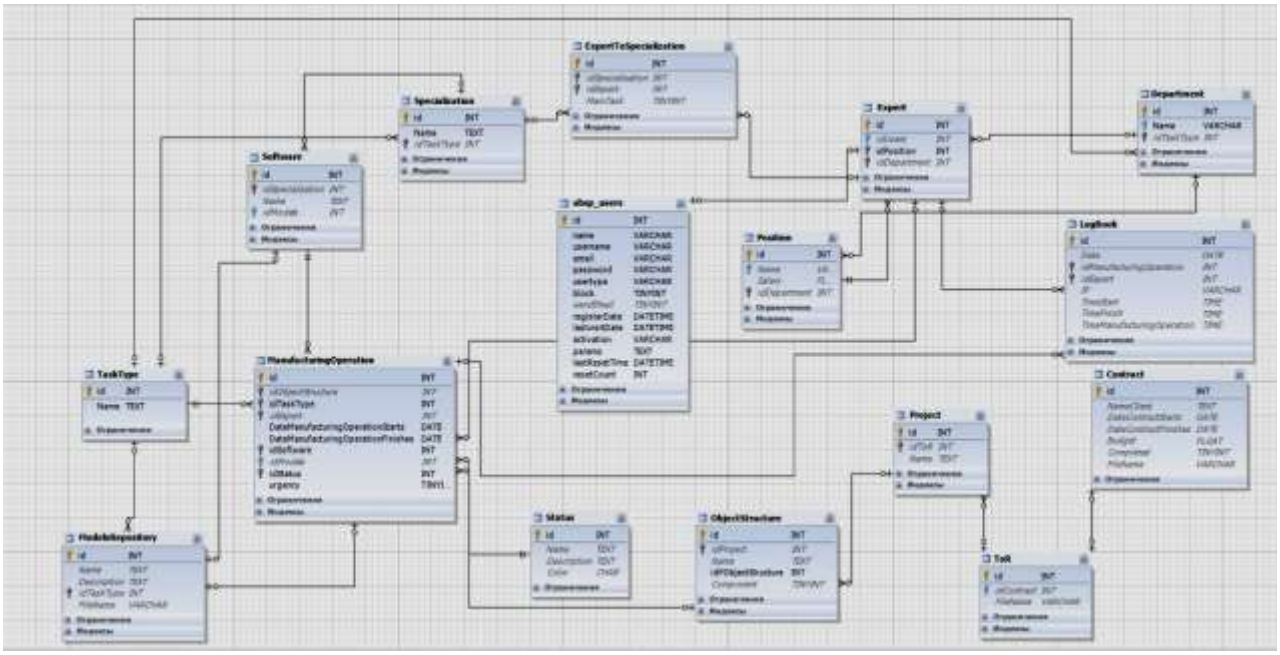

Рисунок 2.1 – Схема даних розробленої бази даних КЗАПР

2.2 Розробка функціональних вимог до КЗАПР з метою їх надійної верифікації

2.2.1 Обґрунтування вибору способу опису функціональних вимог

При створенні програмного забезпечення для комплексу засобів автоматизації проектувальних робіт однієї з основних проблем є забезпечення його якості. Основним підходом до розв'язку цієї проблеми є верифікація програм [5]. Виділяються два основні підходи до верифікації: динамічний і статичний (формальна верифікація) [6].

При динамічному підході програма перевіряється в процесі виконання. Найбільш частим застосуванням цього підходу є тестування програм. Однак, як показав Е. Дейкстра [7], тестування не може гарантувати коректність програми.

Формальна верифікація дозволяє довести, що програма відповідає специфікації [8]. При цьому під специфікацією програми будемо розуміти високорівневі вимоги до її поведінки. У рамках даного підходу виділяються два основні напрямки: доказова верифікація [9] і верифікація моделі програми (model checking) [10].

У загальному випадку, доказова верифікація є алгоритмічно нерозв'язним завданням, навіть у такому окремому випадку, як доказ зупину програми [11]. Таким чином, цей підхід пов'язаний з величезною ручною роботою, що робить його малозастосовним на практиці.

Для верифікації моделі програми потрібно розв'язати наступні завдання:

а) побудова по програмі моделі з кінцевим числом станів [10] і

б) формальний опис вимог до програми термінах одного з видів темпоральної логіки [12, 13].

У результаті верифікації моделі або підтверджується, що модель задовольняє формалізованим вимогам, або будується контрприклад. В останньому випадку потрібно визначити причину некоректності: помилка у вихідній програмі або помилка при побудові моделі. При цьому часто потрібно розв'язати завдання переносу контрприкладу у вихідну програму.

Відзначимо, що рішення перерахованих завдань для програм загального виду погано піддаються автоматизації.

В [14] було запропоновано застосовувати кінцеві автомати для опису поведінки програм. Цей підхід був названий «Автоматне програмування», а відповідні йому програми – автоматними [15]. Програми цього класу мають наступну специфіку:

- кінцеві автомати є добре визначеною абстракцією;
- кінцевий автомат — одна з найбільш простих моделей дискретної математики [27];
- простота опису взаємодії автоматів;
- централізація логіки роботи в автоматах і логічна простота вхідних і вихідних впливів.

Розглянемо вплив цих факторів на верифікацію автоматних програм.

При динамічній верифікації програм часто висувається вимога повного покриття тестами операторів або умов [6]. У термінах автоматної програми покриття операторів означає, що кожний вхідний і вихідний вплив був перевірений у процесі тестування, а покриття умов – що перевірений кожний

перехід. Це дозволяє автоматизувати створення тестів для автоматних програм. Наприклад, набір регресійних тестів [16], що покриває всі переходи, може бути генерований досить просто (наприклад, на основі методів, запропонованих в [17]).

Доказова верифікація автоматних програм може бути розбита на дві підзадачі, кожна з яких досить проста: верифікація автоматів і верифікація вхідних/вихідних впливів. Для верифікації автоматів застосовуються методи, розроблені в теорії компіляторів [18], а тому що вхідні й вихідні впливи звичайно не містять складних умов і циклів, то в багатьох випадках вони можуть бути верифіковані автоматично.

При верифікації моделі, звичайно, по програмі будується модель Кріпке [10], яка, фактично, є спеціальним видом автомата, що спрощує її побудова по автоматній програмі. Тому можливо створення формальних методів побудови моделі Кріпке по автоматній програмі й доказ коректності цих методів. Таким чином, для програм цього класу можна виключити помилки при побудові моделі по програмі.

Побудова формальних вимог до моделі по специфікації автоматних програм також звичайно не представляє складності. Наприклад, такі вимоги, як «Після стану «відкрите» завжди випливає стан «закрите»» або «Вихідний вплив z1 не повинне з'являтися після вхідного впливу x1» можуть бути записані в термінах темпоральної логіки для станів і переходів автоматів, а не моделі програми, як це робиться, наприклад, в [19, 20]. Специфікація, записана в термінах автоматів, може бути автоматично перетворена у формальні вимоги до моделі.

У виді того, що модель гарантовано відповідає верифікованій програмі, обробка контрприкладів також спрощується. При цьому шлях, що приводить до помилки в моделі Кріпке, може бути легко відображений у термінах автоматної програми [21]. При цьому наочно демонструється порушення специфікації.

Таким чином, верифікація автоматних програм є суттєво більше простим завданням, чому верифікація програм загального виду. При цьому багато етапів верифікації можуть бути автоматизовані, що дозволить більш широко застосовувати її в реальних проектах.

Відзначимо, що застосування засобу автоматичної генерації коду дозволить не проводити його верифікацію, тому що він буде повністю відповідати вже верифікованому набору автоматів. Перспективність такого підходу підтверджується досвідом NASA в області побудови надійного програмного забезпечення [24]. При цьому автоматична генерація коду по автоматах виконувалася за допомогою інструментального засобу Stateflow [25]. Застосування розробленого верифікатора разом із програмним комплексом автоматного програмування Unimod [26] дозволяє сподіватися на одержання програм аналогічної якості.

Досягнуті останнім часом результати у цьому напрямку [27-31] дозволяють мати впевненість у перспективності зробленого вибору.

2.2.2 Математичний апарат опису функціональних вимог

Для розробки програмного забезпечення комплексу засобів автоматизації проектувальних робіт запропоновано використовувати модель програми як скінченого автомату. Розглядається один із способів моделювання, специфікації та верифікації програм, побудованих на основі моделі скінчених автоматів. Технологія створення програм на такій основі є досить ефективною при створенні програмного забезпечення для реагуючих систем і систем логічного керування. З погляду моделювання й аналізу програмних систем ця технологія має ряд переваг у порівнянні із традиційним підходом, тому що виключає проблему адекватності побудованої програмної моделі вихідній програмі. Набір взаємодіючих автоматів, що описує логіку програми, уже є адекватною моделлю, по якій формальним образом будуються програмні модулі. Властивості програмної системи у вигляді автоматів можуть бути сформульовані та специфіковані звичайнім і зрозумілим чином. Перевірка

властивостей здійснюється в термінах, які природно випливають із автоматної моделі програми. Практичним результатом роботи є застосування логіки LTL, мови Promela та інструментального засобу SPIN для специфікації й верифікації ієрархічних автоматних програм.

При до проектування й побудові програм на основі моделі скінченого автомату виділяються дві частини: системно незалежна й системно залежна [5, 6, 7, 8, 9]. Перша частина реалізує логіку програми й задається системою взаємодіючих автоматів Мура-Милі. Проектування кожного автомата полягає в створенні по змістовному опису схеми зв'язків, яка описує його інтерфейс, і графа переходів, який визначає його поведінку. По цих двом документам формально та адекватно можуть бути побудовані програмні модулі, відповідні до автоматів. Вже потім реалізується системно залежна частина.

Розглянемо технологія специфікації й верифікації автоматних програм, які побудовані за ієрархічною моделлю [9], котра ґрунтується на застосуванні методу перевірки моделі програми [10, 11]. З погляду простоти й адекватності сприйняття дослідимо можливість специфікації структурних і семантичних властивостей автоматних програм за допомогою темпоральної логіки лінійного часу LTL та оцінимо зручність і доцільність застосування для аналізу коректності автоматних програм широко відомого LTL верифікатора SPIN. Запропонована технологія специфікації й верифікації автоматних програм, побудованих за ієрархічною моделлю.

Основний автомат АТ відповідає за логіку взаємодії з користувачем через панель керування. Автомат АТ реагує на команди (натискання кнопок на панелі інструментів), робить процедуру читання/запису службової інформації у пам’ять автомата, запитує необхідну інформацію в інформаційній мережі й, крім того, постійно (по системному таймеру, що генерує подію e0) перевіряє можливість (умови) переходу в наступний стан. Автомат АТ має вкладені автомати A1 і A2, взаємодія з якими здійснюється передачею керування за допомогою відправлення їм певних подій і відстеженням їх поточних станів. Схема зв'язків і граф переходів автомата керування представлена на рисунку 3.

Автомат *A1* взаємодіє з автоматом керування *AT*, передаючи останньому свій поточний стан і одержуючи від нього події «виконати своє функціональне призначення» і «перевірити переходи». Також автомат *A1* звертається із запитами параметрів до інформаційної підсистеми і системного таймеру здійснює вихідні впливи відповідно до свого функціонального призначення. Аналогічно працює і автомат *A2*. Схеми зв'язків і графи переходів автоматів керування *A1* і *A2* представлені на рисунку 4.

Основи специфікація й верифікація автоматних моделей. При побудові автоматної програми в рамках ієрархічної моделі логіка програми зосереджує в основному автоматі, який розподіляє керування вкладеним автоматам залежно від поведінки керованого об'єкта. Кожний автомат програми взаємодіє тільки зі своїми головним і вкладеними автоматами, що полегшує розуміння програми. Під час проектування або верифікації такої автоматної програми можливий розгляд частини або деякого піддерева системи автоматів залежно від тієї функції, яка реалізується виділеною підсистемою автоматів. Будь-яка підсистема взаємодіючих автоматів в ієрархічній моделі являє собою дерево автоматів, яке можна розглядати як окрему систему (як окрему автоматну програму). Це дозволяє міняти масштаб усієї системи, відносячи до зовнішнього середовища, тобто до зовнішнього об'єкта керування, її частину, яка не цікавлять проектувальника в цей момент часу, і втримувати в увазі тільки аналізовану підсистему. І, нарешті, при верифікації специфікація й аналізі властивостей автоматної програми проводяться простіше при зрозумілій і не складній за структурою верифіковуваній моделі.

Автоматна програмна модель є дуже зручним об'єктом для автоматичної верифікації методом перевірки моделі (model checking) [10, 11], зміст якого полягає в наступному.

Поведінка програмної моделі описується кінцевою системою переходів, яка називається структурою Кріпке. Скінченість системи переходів означає, що система має скінчене число станів. При деякому спрощенні скінчену систему

переходів можна представити як скінчений орієнтований граф з явно виділеною початковою вершиною.

Далі в рамках структури Кріпке з використанням мови темпоральної логіки специфікуються властивості програмної моделі, істинність яких для цієї моделі потім і перевіряється. Використовуючи такі прості об'єкти, як скінчена система переходів і формули темпоральної логіки, можливо автоматичним способом виконати перевірку властивостей моделі, заданих у вигляді формул.

Структура Кріпке автоматної моделі. *Структурою Кріпке* над множиною елементарних висловлень *P* називається система переходів

$$S = (S, S_0, \rightarrow, L),$$

де S — скінчена множина станів;

S_0 — початковий стан;

$\rightarrow \subset S \times S$ — тотальне відношення переходів (тотальність означає, що для кожного стану $s \in S$ повинен існувати стан $s' \in S$, для якого має місце $(s, s') \in \rightarrow$, тобто $s \rightarrow s'$);

$L: S \rightarrow 2^P$ - функція, яка позначає кожний стан множиною елементарних висловлень, дійсних у цьому стані.

Шлях у структурі Кріпке зі стану S_0 — це нескінченна послідовність станів $\pi = s_0 s_1 s_2 \ldots$ така, що для всіх $i \geq$ □ виконується $s_i \rightarrow s_{i+1}$. Для деякого шляху $\pi = s_0 s_1 s_2 s_3 \ldots$ позначимо π^i суфікс π, який виходить видаленням з π перших i станів - наприклад, $\pi^1 = s_1 s_2 s_3 \ldots$, а $\pi(i)$ буде позначати i-ий стан шляху, $\pi(0) = s_0$, $\pi(1) = s_1$ і т.д.

Розглянемо для довільної програми, побудованої на принципах автоматного програмування, її ієрархічну автоматну модель *A, яка* представляє собою набір взаємодіючих автоматів $(A_0, A_1, \ldots, A_n)$. Для цієї автоматної моделі A побудуємо структуру Кріпке, яка, з одного боку, описує всі можливі стани системи A, а з іншої сторони, задає семантику елементарних висловлень,

дійсних у цих станах.

Для цього виділимо в кожному автоматі A_k, крім основних станів, множину його проміжних станів, у яких автомат перебуває під час переходу з одного основного стану в інший. Проміжний стан переходу автомата будемо фіксувати щораз, коли автомат зробить одну з елементарних дій, тобто автомат відреагує на деяку подію e_k, звернеться до об'єкта керування із запитом значень вхідних змінних x_k ($!x_k$) або зробить деякий вихідний вплив на об'єкт керування або вкладений автомат z_k.

Продемонструємо ідею виділення проміжного стану з переходу довільного автомата A_k (рис. 2.2).

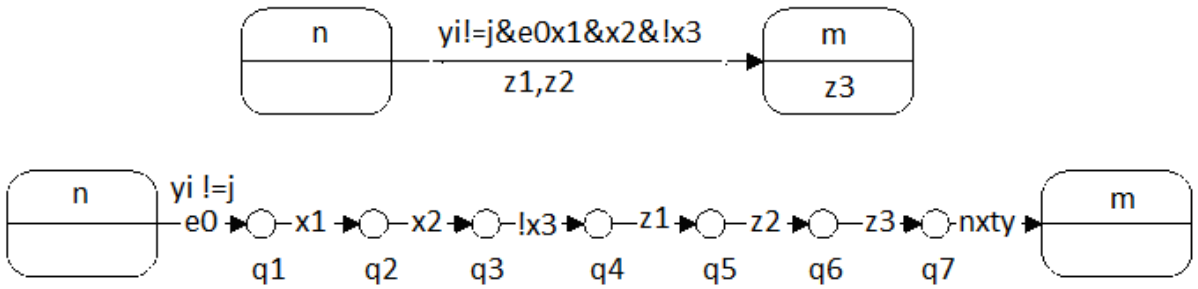

Рисунок 2.2 – Виділення проміжних станів для автоматного переходу

Проміжний перехід з позначкою *nxty* уводиться для того, щоб мати можливість безпосередньо відслідковувати момент переходу вкладеного автомата в наступний основний стан з одночасною передачею керування головному автомату. У випадку проміжного переходу *nxty* основного автомата, активним залишається як і раніше основний автомат. Внутрішній перехід з позначкою e0 може відбутися при настанні події e0, якщо виконується умова *yi*!=j. Усі інші переходи не мають умов і є активними при переході автомату у відповідні проміжні стани. У зазначеній на рисунку послідовності переходів по внутрішніх станах обов'язково спочатку йде перехід, відповідний до вхідної події, далі переходи з позначками вхідних запитів, а потім переходи, позначені

вихідними впливами.

Далі для довільного автомата під переходом у наступний стан будемо припускати *внутрішній* перехід в основний або проміжний стани.

Кожному автомату $Ak \in A$ зіставимо змінні *yk, xmk, zdk, evk* і *stk* ($0 \le k \le n$). А для всієї системи автоматів *A* у цілому введемо змінні *ev, xm, zd, act, auto* і *evnt*. Кожна з уведених змінних має наступний зміст.

1. Змінна y_k зберігає останній основний стан автомата A_k.
2. Змінна xm_k містить останній виконаний запит параметрів об'єкта керування. Значеннями xm_k є імена вхідних впливів у прямій або інверсній формі, тобто xm_k може містити або *x*, або !*x*, де *x* належить множині X_{A_k} вхідних запитів автомата A_k.
3. Змінна zd_k використовується для зберігання імені останнього виконаного вихідного впливу $z \in ZA_k$, де ZA_k— множині вихідних впливів автомата A_k.
4. Змінна ev_k містить ім'я останньої обробленої автоматом A_k вхідної події $e \in EA_k$, тобто ім'я такої події, на яку A_k відреагував; E_{A_k} — множина подій для автомата A_k.
5. Змінна st_k зберігає поточний стан автомата A_k. Значеннями st_k є як основні, так і проміжні стани автомата.
6. Змінна *auto* містить номер активного в цей момент часу автомата з набору *A*, тобто номер останнього автомата, що одержав керування.
7. Змінна *evnt* зберігає ім'я вхідної події, яка попала на обробку, для вкладеного автомата, якому було передане керування. Значеннями змінної є імена вхідних подій усіх вкладених автоматів з *A* і порожнє значення 0. Змінна *evnt* має значення 0 тоді, коли вхідна подія була оброблена або ж зігнорована автоматом (якщо в поточному стані автомат не може відреагувати на дану вхідну подію). У змінну *evnt* записується ім'я деякої події, якщо відбулася передача керування із цією подією від головного автомата до вкладеного.
8. Змінна *act* використовується для зберігання імені останньої елементарної дії, що відбувся в системі автоматів А. Значеннями змінної *act*

можуть бути імена вхідних подій $e \in E_A$, вхідних запитів x і $!x$, де $x \in X_A$, вихідних впливів $z \in Z_A$, а також імена спеціальних дій 0, *nxty* і *end*. Після переходу активного автомата в новий проміжний або основний стан змінна *act* містить мітку цього переходу. Змінна *act* має значення 0, коли вкладений автомат, перебуваючи в основному стані, не може обробити вхідну подію й ігнорує її. При цьому автомат робить порожню дію 0 і переходить у той же самий основний стан, у якому перебував. Якщо *act = nxty*, то це означає, що деякий вкладений автомат перейшов у свій наступний основний стан з одночасною передачею керування головному автомату. Значення *end* використовується для ідентифікації тупикового стану всієї системи автоматів. Для того щоб виконати умова тотальності відносини переходів для структури Кріпке, установлюється, що з тупикового стану (структури Кріпке) є всього лише один перехід сам у себе й при цьому відбувається дія *act = end*.

9. Змінні *ev, xm* і *zd* використовуються для зберігання імен останніх виконаних вхідних подій, вхідних запитів і вихідних впливів відповідно в рамках усієї системи *A* у цілому.

10. Тоді станом *s* структури Кріпке автоматної моделі *A* є вектор значень змінних

(act, auto, evnt, ev, xm, zd, evo, xmo, zdo, yo, sto,.. ., evn, xmn, zdn, yn, stn).

11. У початковому стані S_0 структури Кріпке S_A усі змінні *yi* і *sti* містять початкові стани відповідних автоматів A_i, змінна *auto* = 0, тобто активним є основний автомат A_0, *evnt* не містить подій і встановлена в 0, а всім іншим змінним привласнюється початкове значення *intl*, яке вводиться спеціально для ініціалізації й далі не використовується.

Відношення переходів структури Кріпке автоматної моделі *A* визначається відповідно до поведінки системи автоматів *A*.

1. Перехід системи відповідає тільки одному із внутрішніх переходів деякого автомата з множини A.

2. При виконанні умов переходу деякий автомат *Ak* може зробити цей перехід, якщо в даному стані він є активним (йому передане керування), що

висвітлює значення змінної *auto = k.*

3. Для активного вкладеного автомата *Ak* перехід з позначкою e, де *e* — деяка вхідна подія, може відбутися, якщо змінна *evnt = e* й виконується умова для цього переходу — дійсний предикат над значеннями змінних станів *yi,* зіставлений переходу. Після спрацьовування цього переходу змінюються значення змінних *act, evnt, ev, evk* і *stk*. Змінна *evnt* присвоює нульове значення 0, що означає, що подія була оброблена, а змінним *act, ev* і *evk* присвоюється ім'я події e. В *stk* міститься внутрішній стан, у який відбувся перехід. Якщо активним автоматом є основний автомат A0, то перехід з позначкою *e* відбувається при виконанні умови переходу, а *evnt* = 0.

4. Якщо вкладений автомат *Ak* одержав від головного автомата керування з подією e, яке перебуває в змінній *evnt,* але не може на нього відреагувати — не існує з даного основного стану переходу з позначкою *e* або для нього не виконані умови переходу, то відбувається порожня дія *act :=* 0, змінна *evnt* встановлюється рівною нулю й керування передається головному автомату — у змінну *auto* заноситься номер головного автомата, а всі інші змінні не змінюють своїх значень.

5. Якщо відбувся перехід с позначок x, де *x* — деякий вхідний запит у прямій або інверсній формі (x може мати вигляд *!x'*), те відбуваються зміни наступних змінних: *act := x, xm := x, xmk := x,* а *stk* одержує значення внутрішнього стану, у який був зроблений перехід.

6. При спрацьовуванні переходу автомата *Ak* з позначкою z, де *z* — вихідний вплив, відбуваються присвоювання *act :=* z, *zd := z, zdk* := z, і *stk* одержує значення внутрішнього стану, у який був зроблений перехід. Більше того, якщо z належить до вихідних впливів другого типу, тобто z = *Ak'*(e), де *Ak'* — вкладений автомат, а *e* — передана разом з керуванням вхідна подія для автомата *Ak'*, то *evnt* одержує значення e, і активним стає вкладений автомат *Ak'* за рахунок присвоювання *auto := k'.*

7. Якщо відбувся перехід з позначкою *nxty* в основний стан *j* у вкладеному автоматі *Ak'*, то відбуваються присвоювання *act := nxty, yk'* := j,

stk' := j і передача керування головному автомату *Ak* через *auto := k.* Якщо перехід *uxty* відбувся в основному автоматі A0, то змінна *auto* = 0 залишається без зміни.

8. Якщо з деякого стану автомата *Ak* існує перехід, який позначений змінними z, x, *!x* або *nxty,* то для його спрацьовування не потрібно додаткових умов, крім активності цього автомата *auto = k.*

9. Якщо система взаємодіючих автоматів *A* потрапила в тупиковий стан, що рівносильно неможливості зробити перехід з основного стану основного автомата A0 через порушення умов переходів, то відбувається спеціально введений перехід *act := end,* який веде в той же самий стан, а значення всіх інших змінних залишаються без змін.

Завдяки побудованій структурі Кріпке при специфікації й верифікації є можливість як елементарні висловлення використовувати предикати над значеннями введених змінних, що дозволяє виражати наступні властивості станів автоматних моделей.

1. У кожному стані автоматної моделі існує можливість відслідковувати, яка остання дія відбулася перед переходом у поточний стан за допомогою змінної *act.*

2. Є можливість визначення типу останньої дії, тобто чи відбулася вхідна подія, вхідний запит або вихідний вплив, за допомогою виражень *act = ev, act = xm* і *act = zd.* Наприклад, якщо в поточному стані автоматної моделі виконується *act = zd,* це означає, що останньою дією, яка відбулася при переході в даний стан, є деякий вихідний вплив. Уточнити, до якого автомата системи *A* належить вихідний вплив, можна за допомогою виразів виду *act = zd_i.* Нарешті, істинність вираження *act = z* означає, що останньою дією при переході в поточний стан був вихідний вплив *z.*

3. Через змінну *auto* у кожному стані автоматної моделі визначається активний у цей момент часу автомат. Наприклад, якщо в стані системи *A* виконується *auto = 0*, це означає, що активним є основний автомат A_0.

4. Для кожного автомата A_i у поточному стані системи *A* можна

довідатися через змінну y_i останній основний стан, у якому він перебував. Більше того, вираження $y_i = st_i$ означає, що автомат A_i у даному стані системи A перебуває у своєму основному стані.

5. За допомогою виразу *act = end* можна відслідковувати тупикові стани системи автоматів *A*. А також через *act = nxty* відслідковується перехід одного з автоматів системи у свій новий основний стан з одночасною передачею керування головному автомату.

Темпоральна логіка LTL для автоматної моделі. Одними з найбільш популярних темпоральних логік для специфікації й верифікації властивостей програмних систем є логіка CTL (branching-time logic або computation tree logic) і логіка лінійного часу LTL (linear-time logic). Для цілей верифікації автоматних програм логіка LTL заслуговує на особливу увагу, оскільки будь-яка формула в рамках цієї логіки, по суті, являє собою автомат Бюхі, який описує нескінченні припустимі шляхи структури Кріпке, яка у свою чергу задає поведінку (усі можливі виконання) автоматної програми, що перевіряється на коректність. Це дозволяє при верифікації й специфікації автоматних програм оперувати в основному таким простим поняттям, як «автомат». Таким чином, можна в деякому змісті говорити, що LTL є більш природнім засобом специфікації властивостей автоматних програм.

Формули логіки LTL для структури Кріпке SA автоматної моделі *A* будуються по наступній граматиці:

$$\phi ::= \text{true} \mid \text{false} \mid p \in P \mid \neg \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid X\phi \mid \phi U \phi \mid \phi \bar{U} \phi \mid [] \phi,$$

де $p \in P$ — елементарні висловлення, визначені над множиною станів автоматної моделі А.

Формули темпоральної логіки лінійного часу інтерпретуються через виконання системи переходів (структури Кріпке). Поведінка системи задовольняє формулі ϕ логіки LTL, якщо всі шляхи, що виходять із початкового стану, задовольняють даній формулі ϕ. Для структури Кріпке SA=

(S, so,→,L) позначимо Ωs множину усіх можливих шляхів, що виходять із початкового стану s0, а Ω — множину усіх можливих шляхів з будь-якого стану.

Відношення виконуваності |= формули *ф* логіки LTL для деякого шляху π = S0S1S2S3 ... $\in$ Q структури Кріпке індуктивно визначається в такий спосіб:

π|= true і |≠ false ;

π|= p для p ∈ P ⇔□ □ ∈ L(π(0));

π |= ¬ф ⇔π |≠ ф;

π |= ф ∧ ф ⇔π |= ф і π |= ψ;

π |= ф ∨ ф⇔□π |= ф або π |= ψ ;

π |= *X ф* ⇔ π(0) →π(1) і π*j* |= *ф;*

π |= *ф P ф* ⇔існує таке *j* > 0, що π*j* |= і при цьому для всіх і, 0 ≤ *i* < *j*, *nj* |= ф;

π|= *ф V ф* ⇔для будь-якого *j*≥*1* такого, що π*j* |≠ψ, існує і, 0 ≤ *i* < *j*, що π*г* |= *ф;*

π = ⟨⟩*ф* ⇔ *true U ф* — існує таке *j* ≥ 0, що *j* |= ф;

π = [] *ф*⇔ *false V ф* — протягом усього шляхи *n* формула *ф* виконується.

Множина [[ф]]s являє собою множину усіх шляхів структури Кріпке SA, для яких дійсна формула ф, тобто [[*ф*]]s = { | $\pi \in \Omega$., π |= ф}.

Структура Кріпке буде задовольняти формулі *ф* логіки LTL, якщо Ωs ⊆ [[ф]]s

Крім уведених логічних зв'язувань ∧ і ∨ традиційно використовуються зв'язування → і ↔:

$$ф \rightarrow ψ \equiv \neg ф \vee ψ;$$

$$ф \leftrightarrow ψ \equiv (ф \rightarrow ψ) \wedge (ψ \rightarrow ф) = (\neg ф \vee ψ) \wedge (\neg ψ \vee ф).$$

Далі поряд із символами ¬, ∧ і ∨ будуть також використовуватися символи !, && і ||.

2.2.3 Функціональні вимоги до КЗАПР

Темпоральні властивості автоматних моделей

Розглянемо кілька прикладів темпоральних властивостей, які є загальними для будь-якої ієрархічної системи взаємодіючих автоматів і які у значній мірі визначають показники якості розроблюваного програмного забезпечення.

«*Тупиковий стан».* Властивість, що описує неможливість переходу автоматної програми в тупиковий стан, з якого не можна вийти, застосовний до моделі будь-якої автоматної програми. У рамках описаної структури Кріпке ця властивість може бути описане мовою темпоральної логіки LTL у такий спосіб:

!<>(act == end) або [](act != end).

Ця формула означає, що протягом усіх шляхів починаючи з початкового стану структури Кріпке не існує переходу з позначкою «end». По побудові структури Кріпке SA такий перехід відбувається тоді, коли немає можливості вийти з основного стану автомата A0 через порушення умов переходів або взагалі повної відсутності переходів з даного основного стану.

«Тупиковий стан вкладеного автомата». Припустимо, що структура Кріпке перейшла в новий стан за допомогою переходу деякого вкладеного автомата *Ak* з *A* у свій наступний основний стан з одночасною передачею керування головному автомату. Далі, нехай протягом усіх можливих шляхів із цього стану структури Кріпке будь-яка вхідна подія, яка передається разом з керуванням для автомата *Ak*, буде ігноруватися їм або через порушення умов переходів, або через відсутність переходів, позначених цією вхідною подією. Виходить, що вкладений автомат потрапив у деякий основний стан, з якого він більше ніколи не вийде, незважаючи на те, що регулярно одержує керування. І цей основний стан у цей момент часу можна назвати тупиковим для автомата *Ak*, хоча в іншій ситуації воно може й не бути таким. Властивість, що виражає відсутність тупикового стану вкладеного автомата *Ak*, задається в такий спосіб:

!<>([](auto == k -> act != evk) && []<>(auto == k)).

«Забутий автомат». Розглянемо ще один різновид тупикового стану вкладеного автомата. Починаючи з деякого моменту часу вкладеному автомату *Ak* ніколи більше не буде передане керування. Таким чином, автомат *Ak* з деякого моменту часу назавжди залишиться в одному зі своїх основних станів. Властивість, що виражає відсутність таких шляхів, виглядає так (автомат ніколи не буде забутий):

!<>([](auto != k) або []<>(auto == k).

«Коректне завершення». Якщо автомат перейшов у стан виконання своєї основної функції, то він обов'язково повернеться в початковий стан, і до того, як він повернеться у стан здатності знову виконувати свою функцію, усі вкладені автомати повернуться у свої початкові стани. Для будь-якого шляху структури Кріпке автоматної моделі системи керування автоматом повинна виконуватися наступна вимога. Якщо система перейде в стан s0 = 9, то в майбутньому, коли відбудеться вихідний вплив z01, вкладені автомати будуть перебувати у своїх початкових станах *st1* = 0 і *st2* = 0. Замінивши темпоральні зв'язування темпоральними операторами, одержимо наступну формулу:

[](y0==9 -> (act!=z01 U act==z01 && st1==0 && st2==0)).

«Скидання в початковий стан». Якщо надано команду «Скидання» до переходу автомата в стан виконання основної функції, то в кожному разі до того, як автомат почне новий цикл роботи, усі автомати керування повернуться в початкові стани. Ця властивість запишеться в такий спосіб:

[](y0 != 9 && y0 != 10 && y0 != 11 && act == e03 ->
(act != z01 U act == z01 && st1 == 0 && st2 == 0)).

«Коректне скидання». Після переходу банкоматом уостанній робочий стан *st*9 аж до початкового стану *st*0. не можна зробити операцію скидання (тобто завершити виконувану транзакцію поданням команди «Скидання»):

[](st0 == 9 -> (act != e03 && st0 != 0 U st0 == 0)).

«Коректна робота після виконання функції». Якщо відбулося виконання запланованої функції, усі автомати повернуться у свої початкові стани до того, як буде наступати здатність повторного виконання функції.

[]((act == z25 || act == z24) ->
(act != z01 U act == z01 && st1 == 0 && st2 == 0)).

Специфікація властивостей у рамках логіки LTL здійснюється двома способами. Або за допомогою формули логіки LTL описуються всі припустимі шляхи структури Кріпке, або задається шлях, якого не повинно існувати в моделі, а потім будується його заперечення.

Відмітимо, що властивості «Скидання в початковий стан», «Коректне завершення», «Коректне скидання» і «Коректна робота після вилучення» мають загальну структуру:

[](p1 -> (p2 U p3)),

де *pi* — елементарне висловлення.

У зв'язку із цим важливим є питання про *шаблони* (структури) темпоральних властивостей, найбільш застосовних і адекватних для верифікації автоматних програм. Наявність таких шаблонів дозволяла б говорити про *класи* темпоральних властивостей автоматних моделей, що, безсумнівно, полегшувало б побудова технологічної схеми перевірки автоматних програм на коректність щодо специфікації.

Верифікація автоматних програм у системі SPIN. Визначення структури Кріпке ієрархічної автоматної моделі й описана специфікація властивостей дозволяють застосувати для верифікації автоматних програм метод перевірки моделі. Для цього надалі доцільно використовувати вже існуючі пакети прикладних програм-верифікаторів, які розробляються й підтримуються провідними науковими лабораторіями протягом досить тривалого часу (більш десяти років). Одним з таких засобів верифікації є система SPIN [33], розроблювальна в лабораторії Bell.

SPIN — це система верифікації моделей для логіки LTL з використанням явного перерахування станів і редукції часткових порядків, що представляє собою інструментальний засіб, який використовується головним чином для верифікації асинхронних програмних систем.

Вхідною мовою системи SPIN для опису моделей і специфікації властивостей є мова Promela. До складу цієї мови входять синтаксичні конструкції декількох різних мов програмування. Логічні й арифметичні вирази мови Promela успадковані від мови C. Синтаксис канальної взаємодії процесів орієнтований на CSP (Communicating Sequential Processes) Хоара. Умовні оператори й оператори циклу засновані на охоронюваних командах Дейкстри; їхній синтаксис наведений нижче:

```
if  : guardl -> S1      do      : guardl -> S1
    : guard2 -> S2              : guard2 -> S2
    : else -> Sn                else -> Sn
fi                      od
```

При використанні оператора if і при кожній ітерації оператора циклу do недетерміновано вибирається одна з істиних охорон guardi і виконується відповідна їй команда і, якщо всі охорони неправильні, то відбувається перехід до команді Sn по оператору else. Допускається використання цих конструкцій і без else.

Розглянемо нижче, яким чином відбувається опис поведінки автоматних моделей (завдання структури Кріпке) у рамках мови Promela на прикладі автомата керування механізмом видачі грошей *A1* системи керування банкоматом.

Далі при використанні конструкції atomic зміна значень змінних усередині фігурних дужок відбувається одночасно й розцінюється системою SPIN як один крок, один перехід у новий стан моделі.

Помітимо, що слово auto у мові Promela є зарезервованим, тому ми використовуємо змінну automaton для того, щоб відслідковувати, який з автоматів у цей момент є активним.

```
proctype A1() { /* опис поведінки моделі автомата A1 */
do
:: automaton == 1 -> /* автомат A1 є активним */
if /* основні стани */
:: st1==0 -> if
:: evnt==e11 -> atomic{act=e11;evnt=0;ev=e11;ev1=e11;st1=2;}
:: else -> atomic{act=0; evnt=0; automaton=0;} fi
:: st1==1 -> if
:: evnt==e0 -> atomic{act=e0;evnt=0;ev=e0;ev1=e0;st1=7;}
:: else -> atomic{act=0; evnt=0; automaton=0;}
fi
/* додаткові стани */
::    st1==2      ->    atomic{act  = z13; zd = z13; zd1 = z13; st1 = 3};
::    st1==3      ->    atomic{act  = z11; zd = z11; zd1 = z11; st1 = 4};
::    st1==4      ->    atomic{act  = nxty; y1 = 1; st1 = 1; automaton = 0};
::    st1==5      ->    atomic{act  = nxty; y1 = 0; st1 = 0; automaton = 0};
::    st1==6      ->    atomic{act  = z12; zd = z12; zd1 = z12; st1 = 5};
 ::   st1==7      ->    if
:: atomic{act = x11; xm = x11; xm1 = x11; st1 = 6};
```

```
:: atomic{act=no_x11; xm = no_x11; xm1 = no_x11; st1 = 10};
fi
::    st1==8      ->    atomic{act   = z12; zd = z12; zd1 = z12; st1 = 5};
::    st1==9      ->    atomic{act   = z14; zd = z14; zd1 = z14; st1 = 8};
:: st1==10 -> atomic{act = x12; xm = x12; xm1 = x12; st1 = 9};
fi
od }
```

Перевірка істинності темпоральної властивості, представленої у вигляді формули ϕ логіки LTL, для моделі (описаної мовою Promela) відбувається в такий спосіб.

Заперечення властивості, яка перевіряється, автоматично перетвориться в автомат Бюхі. Цей автомат описується за допомогою спеціальної синтаксичної конструкції мови Promela, яка називається never claim. Така назва пояснюється тим, що автомат, отриманий у результаті трансляції заперечення властивості, що перевіряється, задає такі шляхи структури Кріпке, які не повинні існувати в поведінці моделі.

Нижче приводиться конструкція never claim для властивості «Коректне завершення» системи керування головним автоматом.

```
#define p1 (y0 == 9)
#define p2 (act != z01)
#define p3 (act == z01 && st1 == 0 && st2 == 0) never {    /* ![](p1 -> (p2 U p3)) */
T0_init:
if
:: (! ((p3)) && (p1)) -> goto accept_S4
:: (! ((p2)) && ! ((p3)) && (p1)) -> goto accept_all
:: (1) -> goto T0_init fi;
accept_S4:
if
```

```
:: (! ((p3))) -> goto accept_S4
:: (! ((p2)) && ! ((p3))) -> goto accept_all
fi;
accept_all: skip }
```

Мітка кожної початкової вершини містить слово init, а мітка кожної допустимої вершини містить у собі слово accept.

Система SPIN будує перетинання автомата never claim і автомата (структури Кріпке), виділеного із програми (написаної на Promela). Перетинання будується «з лету», не очікуючи повної побудови структури Кріпке. Якщо перетинання не порожнє, то видається траса помилки.

Необхідно відзначити обмеження, які накладаються на формули логіки LTL у рамках системи SPIN. При виділенні структури Кріпке автоматної моделі з Promela-Програми система SPIN породжує й допоміжні службові стани такі, як вхід і вихід з конструкцій do і if, які порушують описану вище поведінку автоматної моделі (структуру Кріпке). У зв'язку із цим не рекомендується використовувати у формулах логіки LTL оператор *X* (next), тому що на перевірку істинності темпоральної властивості, вираженої без цього оператора, службові стани (вбудовані в «чисту» структуру Кріпке) ніякого впливу не виявляють. З цього погляду говорити про адекватну перевірку властивості не доводиться.

Втім, як показує практика, у більшості випадків вдається успішно проводити LTL специфікацію автоматних моделей і без застосування оператора *X*. Так у роботі [34] побудовані темпоральні властивостей системи керування банкоматом, запропонованої у [32], і ефективно підтверджені за допомогою системи SPIN. Переконливі приклади наведені також у роботах [35 – 37].

Це дає підстави на успішне використання цього інструментального засобу для перевірки моделі розроблюваного програмного забезпечення комплексу засобів автоматизації проектувальних робіт на основі автоматного підходу. SPIN є безкоштовною й порівняно невеликою програмою, дистрибутив

якої розміром 300 Kb доступний на сайті [33]. Цей інструмент успішно використовується багатьма відомими організаціями в різних цілях. У перелік клієнтів входять NASA, наукові центри збройних сил різних країн, найбільші виробники аерокосмічної галузі. Існує велике світове співтовариство користувачів SPIN, щороку (починаючи з 1995 р.) проводяться міжнародні конференції користувачів.

2.3 Висновки

У результаті проведених досліджень установлено факт приділення недостатньої уваги питанням управління при комп'ютеризації проектних робіт. Для реалізації можливості управління процесом проектування зроблено наступне.

1. Проведено системний аналіз процесу проектування технічних об'єктів із метою розробки системи автоматизації проектних робіт. Запропоновано такі зв'язки системи автоматизації проектних робіт із зовнішнім середовищем, які дозволять як виконувати проектування, так і вдосконалювати та розвивати саму систему. Установлена необхідність наявності трьох видів програмного забезпечення для функціонування системи: програми-клієнта для взаємодії проектувальника з системою, програми-монітора для відслідковування стану системи та пакету прикладних програм проектування, які реалізують процес комп'ютеризації проектних робіт.

2. Представлено обґрунтоване виконання проектних робіт за схемою «зверху-вниз», при якій кожен об'єкт проектується як система. На кожну задіяну в проекті підсистему повинне складатися технічне завдання на її проектування.

3. Складено інформаційну модель для підтримки процесу проектування. Дана модель підтримує структуру проектів технічних об’єктів і забезпечує формалізацію опису процесу проектування.

Таким чином, на концептуальному рівні отримано архітектурне рішення для системи автоматизації проектних робіт в організаціях.

4. У результаті проведених досліджень також виявлено відсутність інформаційного опису процесів діяльності, який дозволяє їх супроводжувати протягом усього життєвого циклу; запропоновано комплексний опис процесів; різні за формою, але сумісні за змістом, описи інформації доповнюють один одного, забезпечують реалізацію вирішуваних задач від початкового етапу концептуального моделювання процесів до супроводження їх виконання, управління та удосконалення.

Єдина модель сприяє автоматизації роботи виконавців із різними рівнями повноважень, забезпечує однозначність розуміння ними виконуваних завдань та зручність узгодження різних складових опису при коригуванні моделі.

3 МОДЕЛІ ТА МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ В УМОВАХ НЕВИЗНАЧЕНОСТІ ПРИ УПРАВЛІННІ ЕНЕРГОЗАБЕЗПЕЧЕННЯМ

3.1 Використання інформаційних систем при плануванні та управлінні гібридними енергомережами

Сучасний стан гібридних енергосистем (ГЕ) потребує аналізу альтернативних варіантів їх розвитку. Існуючі тенденції привели не тільки до ускладнення структури систем управління ГЕ, а й обумовило їх перетворення у єдиний технологічний комплекс, який отримав характерні риси штучних систем кібернетичного типу. У першу чергу це стосується розвитку технологій прийняття рішень при управлінні енергозабезпеченням за рахунок розподіленої генерації електроенергії від різних типів відновлювальних джерел енергії (сонячних панелей, вітрогенераторів та інших).

Проблема енергоефективності ГЕ у більш практичній постановці пов’язана саме з оптимальним плануванням структури ГЕ, управлінням виробництвом та використанням електроенергії. У контексті цього велике значення має розвиток та удосконалення моделей та інформаційних технологій прийняття рішень при плануванні та управлінні.

На основі огляду наукових публікацій можна виділити ряд напрямів, які забезпечують подолання проблем планування та правління.

Першу групу напрямів можна визначити як розробку систем підтримки прийняття концептуальних рішень, у рамках яких створюються нові методи та алгоритми оптимізації процесів. Їх вирішення пов’язане у першу чергу з подоланням проблеми невизначеності як зовнішньої так і внутрішньої інформації для системи управління ГЕ. Ця невизначеність має глобальний характер, що обумовлено загальним рівнем знань у області процесу електрозабезпечення як комплексу процесів, які описують фізичні, технологічні, геофізичні, соціальні та інші відомі та невідомі впливи на ГЕ. Невизначеність може носити будь-який характер, але на сучасному етапі можливо математично описати невизначеність лише двох типів: випадковість та нечіткість.

Невизначеність типу випадковість застосовується до процесів, які пов’язані з появою подій, наслідки яких нам невідомі. Нечіткість виникає у наслідок неоднозначного рішення наслідків подій, які вже відбулися.

Інша група напрямів пов’язана з вирішенням проблеми ускладнення оптимізаційних задач. Зазначене призводить до необхідності проводити багатокритеріальну оптимізацію з пошуком паритетних рішень. Крім того, при оптимальному управлінні вимушено використовуються інтегральні значення параметрів протягом визначеного проміжку часу та прогнозні значення параметрів. Це накладає додаткові вимоги до отриманих рішень в плані їх стійкості до вихідних умов оптимізації, що обумовлює доцільність подальшого розвитку нечітких методів багатокритеріальної оптимізації, які забезпечують більш стійкі рішення.

Для практичного використання досить розповсюджені наступні моделі та методи розрахунку ефективного планування та управління режимами ГЕ:

1. Енергетичні моделі, що виконують розрахунок попиту і пропозиції.

2. Прогнозні моделі:

2.1. Енергетичні моделі для визначення економічних показників.

2.2. Моделі для розрахунку показників відновлювальних джерел енергії:

2.2.1. Моделі, що використовуються для сонячної енергетики.

2.2.2. Моделі, що використовуються для визначення енергії вітру.

2.2.3. Моделі, що використовуються для визначення енергії, отриманої від біомаси та біоенергії.

3. Оптимізаційні моделі.

4. Енергетичні моделі на основі нейронних мереж.

5. Моделі, що використовують для розрахунку скорочення викидів.

Найчастіше для отримання більш ефективних рішень щодо управління ГЕ декілька моделей об'єднують у більш складні.

Для моделювання процесів у реальному часі використовують:

1. Моделювання на базі програмних агентів.
2. Імітаційне моделювання.

3. Мережеве моделювання.
4. Оптимізаційне моделювання.
5. Моделі підтримки прийняття рішень.

Разом із використання моделей необхідним є забезпечення інтерактивності майбутньої інформаційної системи. Тому структура системи постає більш складною. Разом із моделюванням енергетичних процесів в системі є необхідність в одержанні зведеної інформації про роботу енергосистеми.

Для вирішення таких завдань можуть створюватися системи підтримки прийняття рішень (СППР). Це інтерактивні системи, що спроможні виробляти інформацію та сприяти її розумінню з метою надання допомоги у вирішенні складних і погано структурованих завдань.

Застосування СППР при плануванні та управління структури енергосистеми дозволяє отримати рекомендації щодо вибору складових структури енергосистеми та прогнозу роботи енергосистеми за умов невизначеності, враховуючи при цьому зовнішні фактори впливу (споживання в господарстві, особливості ринку установок відновлювальних джерел енергії (ВДЕ), метеорологічні умови тощо).

СППР відрізняються між собою в залежності від кінцевою мети роботи:

– системи визначення оптимального виду ВДЕ для потреб в господарстві;
– системи дослідження продуктивності енергосистеми;
– системи прогнозування генерації електроенергії у відповідності до зовнішніх умов;
– системи керування ГЕ.

Також СППР відрізняються у залежності від вибору підходу до моделювання:

– Системи з використанням динамічного моделювання.
– Системи з використанням аналітичних моделей.

Окрім того, підтримка прийняття рішень може забезпечуватися поєднанням та одночасним використанням декількох різних інформаційних систем. При цьому необхідно забезпечити їх коректну взаємодію.

3.2 Функціональне моделювання інформаційних систем

Більшість підприємств України сьогодні відчувають різкий спад виробництва та трудової активності. Це пов’язано як з об’єктивними причинами (фінансово-економічними), так і суб’єктивними (неготовністю до конкуренції, низьким рівень IT-готовності). Важливим фактором є те, що національним виробникам зараз необхідно конкурувати з світовими підприємствами, у яких співвідношення ціна/якість на товари та послуги є більш вигідною.

Перші кроки для покращення свого стану, підприємства повинні робити на базі діючих технологій, оптимізуючи організацію роботи компанії. Нажаль, більшість національних підприємств роблять акцент на комерційні цілі, поступово виводячи на другий план задачі господарчої діяльності.

Світовий досвід довів, що для досягнення успіху компаніям необхідно збалансувати фінансові, виробничі на комерційні цілі, тобто необхідно робити акцент на підвищення власного потенціалу, що підвищує «якість» підприємства в цілому. Дані кроки дають можливість поступово збільшувати життєспроможність підприємств, дозволяючи отримувати прибуток в майбутньому.

Сьогодні ефективна робота більшості підприємств можлива лише при «гнучкому» управлінні. Оскільки об’єм оброблюваної інформації зростає в геометричній прогресії, то значно ускладняється досягнення цієї цілі. Тому необхідною умовою досягнення «гнучкості» є використання підприємством інформаційних систем (ІС).

Вивівши на перше місце комерційні цілі, більшість вітчизняних підприємств не акцентували увагу на застосуванні сучасних інформаційних

технологій, це призвело до низько рівня комп’ютеризації. У теперішній фінансово-економічній ситуації, більшість компаній не мають змоги купувати сучасні ІС, що забезпечують повне управління процесом діяльності підприємства, тому вони або інтегрують вже закуплені ІС, які автоматизують деякі процеси роботи підприємства, або проектують власні інформаційні системи під власні потреби. Слід зазначити, що останнє має певні переваги для підприємств, що займаються проектуванням ГЕ, адже такі підприємства вирішують не типові завдання, постійно враховуючи мінливі потреби замовників. Потреби замовників в інформаційній підтримці роботи ГЕ можуть стосуватися вирішення окремих задач планування, проектування, установки ГЕ, підтримки їх експлуатації, керування роботою ГЕ. Така особливість процесу роботи обумовлює складності при впровадженні сучасних інформаційних систем, які вирішують задачі різної спрямованості.

3.3 Аналіз підходів для функціонального моделювання

Під час функціонального моделювання інформаційних систем виникає основна загальна проблема - дотримання коректності (достатньої формалізованості) відображення існуючих процесів, які відбуваються під час вирішення окремих задач з планування/керування ГЕ. Тому для коректного порівняння існуючих методів проектування необхідно сформулювати кількісні та якісні критерії, на основі яких можна робити висновки, щодо адекватності функціональної моделі.

Сформулювати такі критерії можливо лише в результаті аналізу вимог, які висунуті до моделі. Її коректність, тобто найбільша відповідність до заявлених функцій інформаційної системи є основною умовою вибору критеріїв. Проаналізувавши різні вимоги можна виділити загальні якісні критерії:

- Гнучкість – можливість розробленої моделі адаптуватися до різнотипних змін (зміна процесу за яким відбувається планування/керування ГЕ).
- Універсальність – можливість інтеграції на іншому виробництві.
- Масштабованість – можливість розширення моделі, щодо кількості заявлених виконавців (робітників).
- Функціональність складає окрему комплексну групу якісних критеріїв, до яких відносять такі показники: стійкість, надійність, незалежність.

Клас кількісних критеріїв складають показники, від яких залежить загальна фінансова складова (сумарні фінансові витрати, які необхідні для забезпечення моделювання):

- Часовий інтервал, який необхідний для проектування або впровадження моделі.
- Складність розробки або впровадження характеризується фінансовими витратами, які необхідно нести підприємству при впровадженні інформаційної системи, на основі спроектованої моделі.
- Вартість супроводження інформаційної системи.

Аналогічно як функціональність є комплексним якісним критерієм, так і вартість є комплексним кількісним критерієм, який включає в себе багато факторів, таких як:

- Вартість CASE-засобів, або вартість ІС.
- Вартість впровадження.
- Господарські затрати.
- Заробітна плата працівникам, що розроблюють або впроваджують.

Для формалізації моделей використовують різні підходи – системний, ситуаційний, функціональний, процесний. Кожен з яких по-різному інтерпретує виконання бізнес-логіки інформаційної системи. Завдяки своїй простоті та зручності функціональний підхід довгий час був найбільш вживаним.

3.4 Функціональний підхід при проектуванні моделей інформаційних систем

Функціональний підхід широко використовується у програмуванні, проектуванні, керуванні. Він протягом останніх 30 років дуже успішно застосовувався під час проектування інформаційних систем. Особливо вдало процес розробки використовувався, коли область «обробки» обмежувалась у рамках (розмірах), умовах виконання, часі тощо.

Під час проектування функціональний підхід можна використовувати як методологічну основу для моделей ІС. Він обумовлює деяке абстрагування між різними областями моделей, сконцентровуючи «увагу» на спільності функцій. Іншою особливістю функціонального підходу є його комплексність – система розглядається з боку функцій, абстрагуючись від їх внутрішнього змісту.

Рисунок 3.1 – Загальна схема моделі ІС при використанні функціонального підходу

3.5 Функціональне моделювання процесів в системи підтримки прийняття рішень при плануванні структури енергетичної мережі

Використовуючи функціональний підхід, а саме методологію структурного аналізу SADT та нотації IDEF0, IDEF3, можна реалізувати

функціональні моделі окремих процесів, що відбуваються в ІС планування структури ГЕ,

До таких процесів віднесено процес збору та попередньої обробки інформації, процес планування структури енергосистеми з ВДЕ, процес визначення її техніко-економічних параметрів, процес формування рішення щодо вибору оптимальної структури ГЕ серед множини запропонованих.

Підтримка прийняття рішень при плануванні структури ГЕ здійснюється за схемою, наведеною на рисунку 3.2.

Рисунок 3.2 – Схема процесу прийняття рішень

У ході збирання та попередньої обробки інформації до бази даних (БД) надходять дані про значення факторів впливу на структуру енергомережі.

Інформаційна підтримка прийняття рішень при плануванні структури ГЕ здійснюється з використанням комплексу моделей: збору та обробки інформації, визначення техніко-економічних параметрів енергетичної мережі, формування множини альтернатив структури енергетичної мережі, модель формування ефективних рішення щодо планування структури ГЕ, модель визначення оціночних критеріїв.

Для формування рішення щодо планування оптимальної структури мережі використовуються знання експертів, що зберігаються в базі знань. Особа, яка приймає рішення (ОПР), на основі результату аналізу усіх даних, що використовуються у процесі прийняття рішення, отримує техніко-економічне обґрунтування оптимального варіанту структури мережі.

Основними діями, які повинна забезпечувати інформаційна система, є - збір, передача і зберігання інформації, аналіз даних, моделювання процесів, прогнозування параметрів, оптимізація споживання, надання рекомендацій для прийняття рішень, відображення характеристик поточного стану, аналізу і прогнозу в зручній та зрозумілій формі для користувача.

Було вирішено проводити планування структури ГЕ на основі моделювання її роботи у середовищі Matlab з використанням даних про технічні характеристик обладнання та метеорологічні умови місцевості. Оптимізацію рішення щодо вибору структури ГЕ проводити на основі багатокритеріальної моделі у середовищі Matlab. Всю вихідну інформацію виводити для перегляду на сайті для інтерактивності та зручного перегляду інформації у зрозумілому вигляді. Накопичені та оброблені дані зберігати в БД. Таким чином при інформаційній підтримці процесу планування структури ГЕ пропонується використовувати декілька різних інформаційних систем.

Контекстна діаграма процесу планування енергозабезпечення будівель з використанням відновлювальних джерел енергії представлена в нотації IDEF0 на рисунку 3.3.

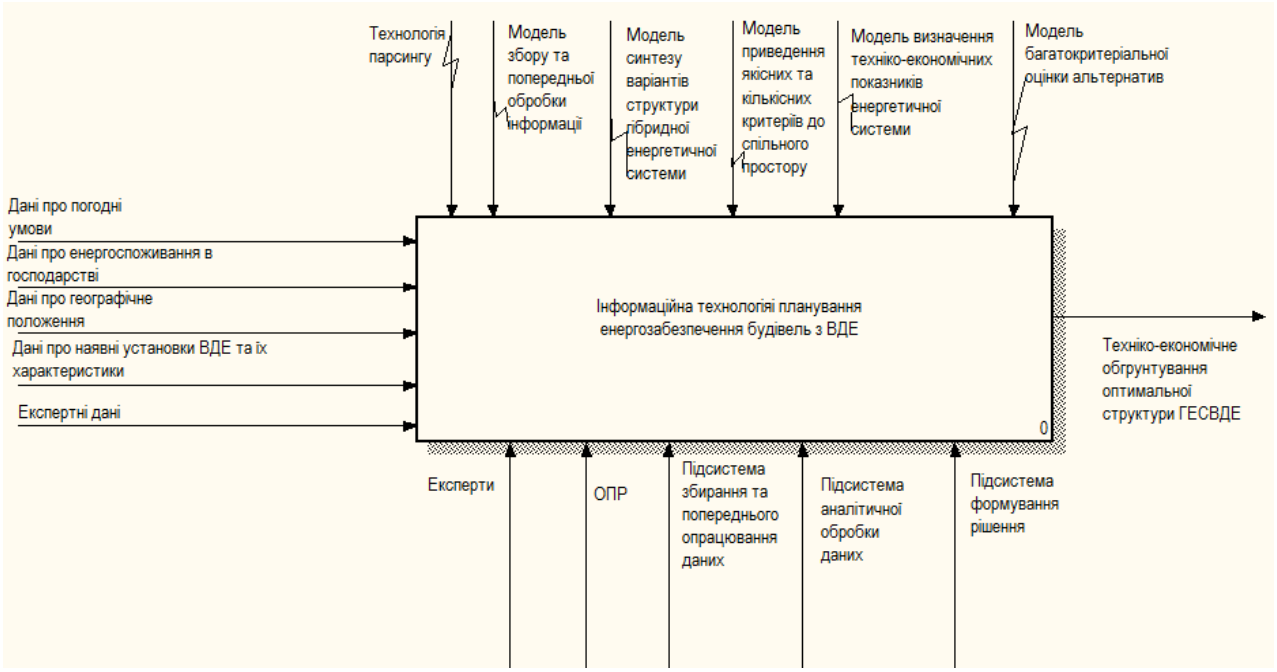

Рисунок 3.3 – Контекстна діаграма інформаційної технології планування енергозабезпечення будівель з відновлювальними джерелами енергії

Інформаційна технологія планування енергозабезпечення будівель з ВДЕ складається із п’яти взаємозв’язаних етапів. Функціональна модель запропонованої інформаційної технології у нотації IDEF0 представлена на рисунку 3.4.

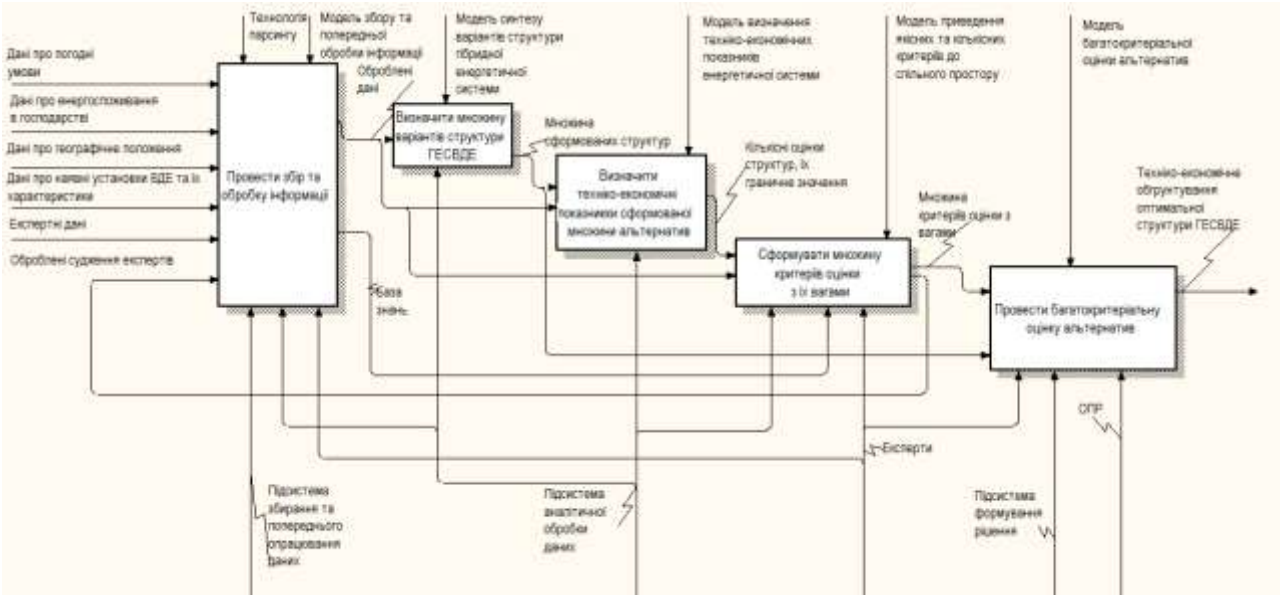

Рисунок 3.4 – Функціональна модель інформаційної технології планування енергозабезпечення будівель з використанням ВДЕ

Компонентами функціонального наповнення процесу збору та попередньої обробки інформації є «Збір даних», «Перевірка на коректність», «Збереження у базі даних», «Формування бази знань» та «Розрахунок найгірших погодних умов». Діаграму декомпозиції процесу збору та попередньої обробки інформації зображено на рисунку 3.5.

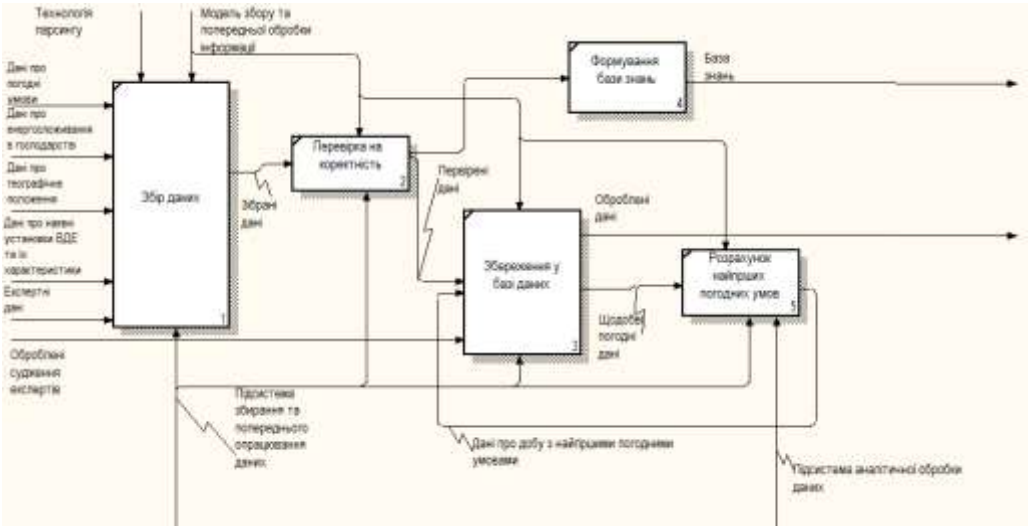

Рисунок 3.5 – Діаграма процесу збору та попередньої обробки інформації

Процес витягу даних відбувається з використанням технології парсингу. Парсинг даних про установки ВДЕ виконується з наступних веб-сайтів: http://ecost.lviv.ua; http://alteco.in.ua; https://energostar.kiev.ua. Парсинг погодних даних визначається погодинно з сайту gismeteo.ua.

Діаграму декомпозиції процесу «Формування множини варіантів структури енергетичної системи» в нотації IDEF3 зображено на рисунку 3.6.

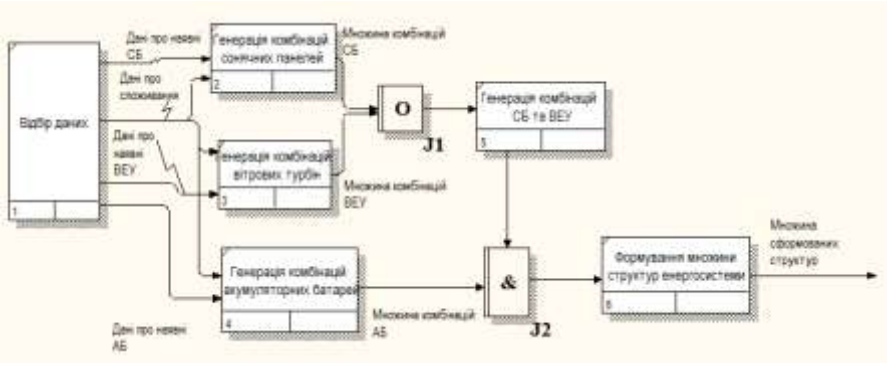

Рисунок 3.6 – Діаграма декомпозиції процесу формування множини варіантів структури енергетичної системи

Компонентами функціонального наповнення процесу є «Відбір даних», що реалізовується підсистемою збору та попереднього опрацювання інформації, а також «Генерація комбінацій сонячних панелей», «Генерація комбінацій вітрових турбін», «Генерація комбінацій акумуляторних батарей», «Генерація комбінацій СБ та ВЕУ», «Формування множини структур енергосистеми», що реалізовуються підсистемою аналітичної обробки інформації.

Діаграму декомпозиції процесу «Визначення техніко-економічних параметрів» представлено на рисунку 3.7.

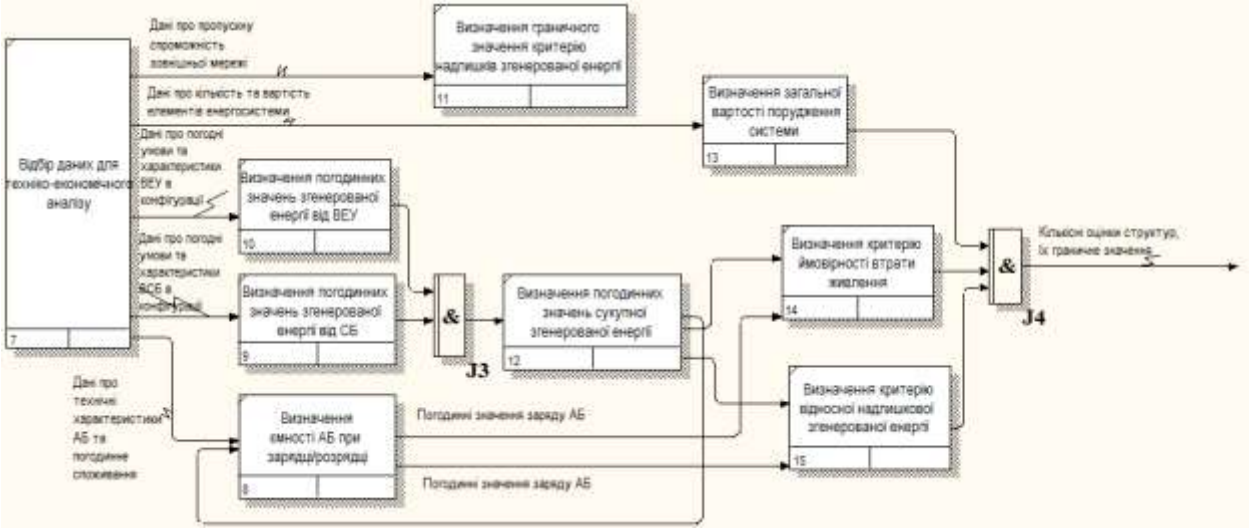

Рисунок 3.7 – Діаграма декомпозиції процесу визначення техніко-економічних параметрів множини структур

Компонентами функціонального наповнення процесу є «Відбір даних для техніко-економічного аналізу», «Визначення погодинних значень згенерованої енергії від СБ», «Визначення погодинних значень згенерованої енергії від ВЕУ», «Визначення ємності АБ при зарядці/розрядці», «Визначення погодинних значень сукупної згенерованої енергії», «Визначення загальної вартості спорудження системи», «Визначення критерію ймовірності втрати живлення», «Визначення критерію відносної надлишкової енергії», «Визначення граничного значення критерію відносної надлишкової енергії».

Діаграму декомпозиції процесу «Формування множини критеріїв оцінки з їх вагами» в нотації IDEF3 представлено на рисунку 3.8.

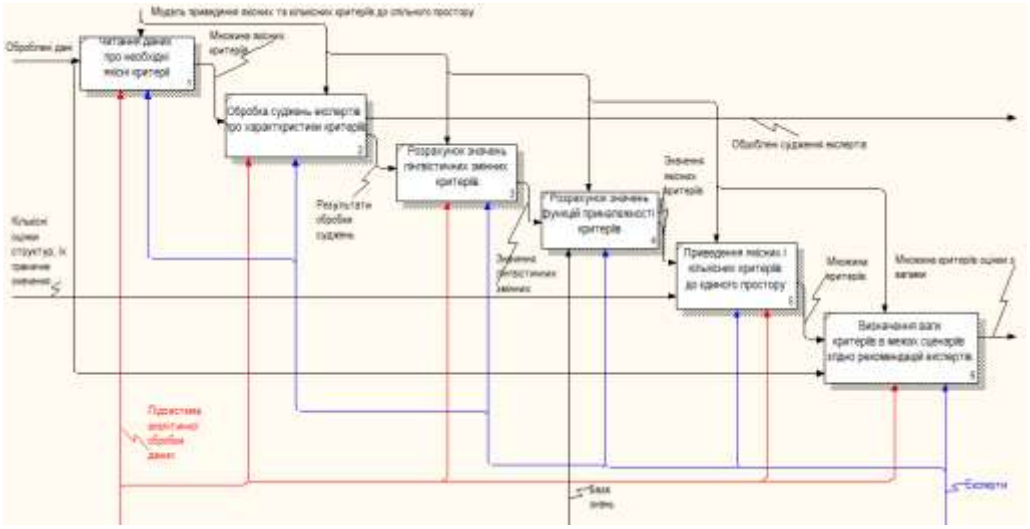

Рисунок 3.8 – Діаграма декомпозиції процесу формування множини критеріїв оцінки з їх вагами

Процес формування множини критеріїв оцінки з їх вагами є останнім етапом, що виконується підсистемою аналітичної обробки інформації. Результати, а саме значення критеріїв з їх вагами для кожного із сценаріїв, використовуються в процесі багатокритеріальної оцінки альтернатив.

Загалом сукупність задач багатокритеріальної оцінки альтернативних структур можна подати у вигляді функціональної моделі в нотації IDEF3 (рис. 3.9).

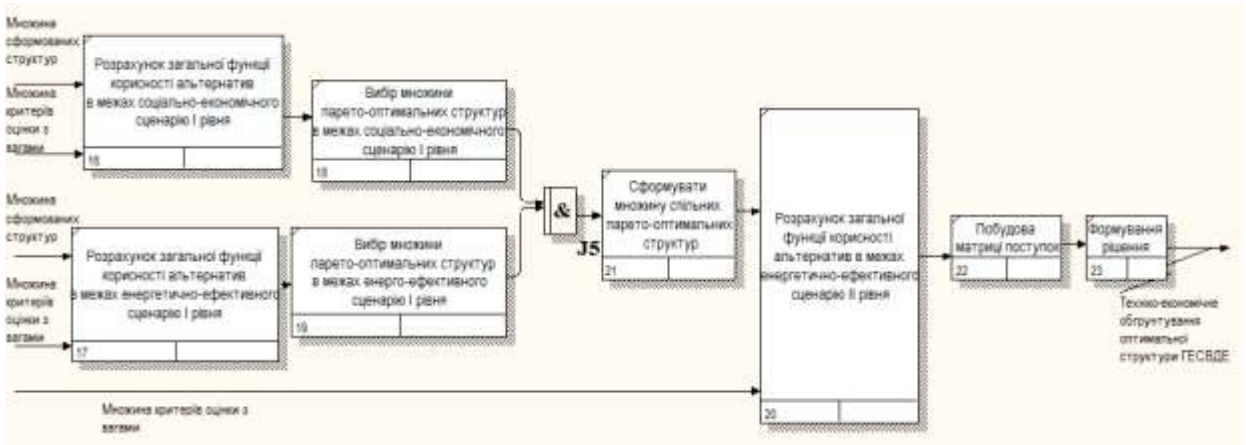

Рисунок 3.9 – Функціональна модель процесу багатокритеріальної оцінки альтернатив

Запропонована схема процесу багатокритеріальної оцінки альтернатив дозволяє структурувати процес визначення оптимального варіанту енергосистеми та отримати оцінки ефективності енергосистеми з більшою точністю.

3.6 Архітектура системи підтримки прийняття рішень при плануванні структури енергетичної мережі

Відповідна СППР для планування структури ГЕ реалізована у вигляді web-додатку з використанням трирівневої архітектури типу клієнт-серверної. СППР складається із інформаційних підсистем:

1. Підсистема збирання та попереднього опрацювання даних (SCP).

Збір інформації про метеорологічні умови, техніко-економічні характеристики обладнання доступного для продажу із сайтів компаній в мережі Інтернет. Дані зберігаються в SQL таблицях та використовуються в інших модулях. На виході отримуємо погодинні погодні дані для доби з найгіршими погодними умовами, характеристики установок ВДЕ, споживання в господарстві, параметри зовнішньої мережі.

2. Підсистема аналітичної обробки даних (SAPD).

Визначення альтернативних структур ГЕСВДЕ, їх техніко-економічних параметрів з використанням Matlab, визначення множин критеріїв. На вхід до підсистеми подаються вихідні дані підсистеми збирання та попереднього опрацювання даних. На виході отримуємо множину альтернативних структур гібридної енергетичної системи, множину критеріїв.

3. Підсистема формування рішень (DM).

Оцінка альтернативних проектних рішень, кінцеве формування оптимального рішення. На вхід подаються вихідні дані підсистеми SAPD. На виході отримуємо список проранжованих альтернатив разом з їх характеристиками.

4. Підсистема зберігання даних (SDS).

Формування бази даних на основі даних SCP та SAPD. На вхід подаються вихідні дані підсистем SCP, DM та SAPD. На виході отримуємо опрацьовані дані для роботи підсистем SCP, DM та SAPD.

5. Представлення результатів.

Використовується web-інтерфейс, через який користувачі отримують доступ до системи та сформованого рішення, що представляє собою техніко-економічне обґрунтування оптимального варіанту ГЕСВДЕ із зазначенням виду та кількості установок ВДЕ, а також користувач отримує перелік проранжованих альтернативних рішень, найближчих до оптимальних. На вхід подаються дані із бази даних, які запитує користувач. На виході користувач отримує звітну інформацію у вигляді таблиць та графіків на web-сторінках.

Обробка неузгоджених даних при інтеграції інформаційних систем потребує застосування спеціальних підходів, які відповідають можливостім середовища, де проектуються ІС. В запропонованій СППР планування ГЕ використано інтегровану інформаційну технологію обробки інформації, що використовує ієрархічне впорядкування:

Пошук інформаційних джерел I_{fc} *в підсистемі збирання та попереднього опрацювання даних* →

Web-інтеграція I_{web} *в підсистемі збирання та попереднього опрацювання даних* →

Інтеграція на рівні баз даних I_{wh} *в підсистемі збирання та попереднього опрацювання даних, підсистемі аналітичної обробки даних, підсистемі формування рішень та підсистемі зберігання даних*

Пошук інформаційних джерел відбувається у підсистемі збирання та попереднього опрацювання даних. Робота такого пошуку базується на основі посередника, у роботі запропоновано використовувати програми-парсери. Програма посередника отримує модель вимог користувача до даних, що він шукає.

Web-інтеграція відбувається у підсистемі збирання та попереднього опрацювання даних. Під web-інтеграцією розуміють методи обробки і представлення інформаційних ресурсів за допомогою Web-технологій. Зберігання і передавання даних відбувається з використанням мови XML, призначеної для організації взаємодії з різними застосуваннями. За пошук інформації відповідають два різні програмні парсери, які взаємодіють з підсистемами за допомогою мови XML.

Інтеграція на рівні баз даних відбувається у підсистемах збирання та попереднього опрацювання даних, аналітичної обробки даних, формування рішень та зберігання даних. Метою інтеграції даних на рівні баз даних є використання систем управління базами даних для запису і читання значень із спільних таблиць.

В СППР база даних призначена для накопичення структурованих даних і метаданих про параметри, що впливають на проектування розподілених енергетичних систем. З БД постачається інформація іншим програмним модулям.

Оскільки між складовими підсистемами СППР існує інтеграція, то необхідно продумати механізм кооперації з файлами, що містять дані в різних форматах, та інформація з яких необхідна при роботі різним підсистемам. Вирішенням даної проблеми стало впровадження «Підсистеми збирання та попереднього опрацювання даних». Через цю підсистему наповнюється база даних, що знаходиться під управлінням СКБД MySQL. Підсистема зберігання даних має забезпечувати надійне зберігання великих обсягів даних, що будуть використані підсистемами в процесі формування рішень.

3.7 Концептуальне моделювання предметної області при інтеграції на рівні баз даних

Концептуальне, чи інфологічне проектування – це процес створення моделі використовуваної на підприємстві інформації, що не залежить від будь-

яких фізичних аспектів її представлення. Інфологічне проектування має своїм результатом одержання семантичних моделей, що відбивають інформаційний зміст конкретної предметної області, тобто концептуальної моделі даних для аналізованої частини підприємства.

Дана модель створюється на основі аналізу інформації, записаної в специфікації користувача, і сформованої в ході підготовки до проектування користувальницької моделі бази даних. Ця модель абсолютно не залежить від логічних і фізичних подробиць реалізації бази даних. На цьому етапі виконується сприйняття реальної дійсності, абстрагування, вивчення й опис предметної області. У результаті цього визначаються об'єкти, властивості і зв'язки об'єктів, які мають відбиватися при проектуванні бази даних.

Розрізняють два головних підходи до моделювання даних при концептуальному проектуванні:

− семантичні моделі;

− об'єктні моделі.

Семантичні моделі головну увагу приділяють структурі даних. Найбільш поширеною семантичною моделлю є модель "сутність – зв'язок" (Entity Relationship model, ER-модель). ER-модель складається із сутностей, зв'язків, атрибутів, доменів атрибутів, ключів. Моделювання даних відображає логічну структуру даних, так само, як блок-схеми алгоритмів відображають логічну структуру програми.

Об'єктні моделі головну увагу приділяють поведінці об'єктів даних і засобам маніпуляції даними. Головне поняття таких моделей − об'єкт, тобто сутність, яка має стан і поведінку. Стан об'єкта визначається сукупністю його атрибутів, а поведінка об'єкта визначається сукупністю операцій специфікованих для нього.

Зближення цих моделей реалізується в розширеному ER-моделюванні (Extended Entity Relationship model, EER-модель).

Спочатку створюємо концептуальну модель об’єктів (табл. 3.1), потім додаємо атрибути (табл. 3.2 – табл. 3.7). Далі створюємо схему концептуальної моделі бази даних (рис. 3.8).

Таблиця 3.1 – Таблиця об'єктів бази даних

Назва об'єкту	**Характеристика об’єкта**
Сонячні батареї (PV)	Дані про різні види сонячних батарей
Вітрові установки (WP)	Дані про різні види вітрових установок
Акумуляторні батареї (B)	Дані про різні види акумуляторних батарей
Конфігурації (Configurations)	Інформація з сформованими конфігураціями сонячних батарей, вітроустановок та акумуляторів
Погодні дані дня (Weather_data)	Детальна інформація про найгірший день
Коефіцієнти (Coefficients)	Розраховані коефіцієнти, що характеризують конфігурації – гібридну вітро-сонячну систему електропостачання

Таблиця 3.2 – Таблиця атрибутів сутності «PV»

Назва атрибута	**Характеристика атрибута**
PV_id	Атрибут, який виступає первинним ключем відношення PV і позначає його порядковий номер. Зв’язує таблиці PV і Configurations
PV	Назва сонячної батареї
PV_Power	Потужність сонячної батареї
PV_Voltage	Напруга сонячної батареї
PV_Amperage	Сила струму сонячної батареї
PV_Max_Vol	Номінальне (максимальне) значення напруги сонячної батареї

Продовження таблиці 3.2

PV_Length	Довжина сонячної батареї
PV_Height	Висота сонячної батареї
PV_Width	Ширина сонячної батареї
Min_Work_Temp	Мінімальна робоча температура сонячної батареї
Max_Work_Temp	Максимальна робоча температура сонячної батареї
PV_Weight	Вага сонячної батареї
PV_Price	Вартість сонячної батареї

Таблиця 3.3 – Таблиця атрибутів сутності «WP»

Назва атрибута	**Характеристика атрибута**
WP_id	Атрибут, який виступає первинним ключем відношення WP і позначає його порядковий номер. Зв’язує таблиці WP і Configurations
WP	Назва вітрової установки
WP_Power	Потужність вітроелектроустановки
WP_Start_wind_speed	Стартова швидкість вітру вітрової установки
WP_Start_work_wind_speed	Стартова робоча швидкість вітру вітроелектроустановки
WP_Max_wind_speed	Номінальна (максимальна) швидкість вітру вітроелектроустановки
WP_Voltage	Напруга вітрової установки
WP_Diameter	Діаметр вітроротора
WP_Number_of_blades	Кількість лопатей вітроелектроустановки
WP_Price	Вартість вітрової установки

Таблиця 3.4 – Таблиця атрибутів сутності «B»

Назва атрибута	**Характеристика атрибута**
B_id	Атрибут, який виступає первинним ключем відношення B і позначає його порядковий номер. Зв’язує таблиці B і Configurations

Продовження таблиці 3.4

Назва атрибута	Характеристика атрибута
B	Назва акумуляторної батареї
B_Power	Потужність акумуляторної батареї
B_Volume	Ємність акумуляторної батареї
B_Length	Довжина акумуляторної батареї
B_Width	Ширина акумуляторної батареї
B_Height	Висота акумуляторної батареї
B_Weight	Вага акумуляторної батареї
B_Price	Вартість акумуляторної батареї

Таблиця 3.5 – Таблиця атрибутів сутності «Configurations»

Назва атрибута	Характеристика атрибута
Conf_id	Атрибут, який виступає первинним ключем відношення Configurations і позначає його порядковий номер. Зв’язує таблиці Configurations і Coefficients
El1_id	Атрибут, який виступає зовнішнім ключем відношення Configurations і зв’язує таблиці Configurations і PV
Element1	Назва сонячної батареї
El2_id	Атрибут, який виступає зовнішнім ключем відношення Configurations і зв’язує таблиці Configurations і WP
Element2	Назва вітроелектроустановки
El3_id	Атрибут, який виступає зовнішнім ключем відношення Configurations і зв’язує таблиці Configurations і B
Element3	Назва акумуляторної батареї
W_id	Атрибут, який виступає зовнішнім ключем відношення Configurations зв’язує таблиці Configurations і Weather_data
N_El1	Кількість елементів сонячних батарей

Продовження таблиці 3.5

Назва атрибута	Характеристика атрибута
N_El2	Кількість елементів вітроелектроустановок
N_El3	Кількість елементів акумуляторних батарей

Таблиця 3.6 – Таблиця атрибутів сутності «Coefficients»

Назва атрибута	Характеристика атрибута
Coef_id	Атрибут, який виступає первинним ключем відношення Coefficientsi позначає його порядковий номер.
REPG_z	Значення критерію надлишковості енергії за рік роботи, коли акумуляторна батарея заряджається
REPG_r	Значення критерію надлишковості енергії за рік роботи, коли акумуляторна батарея розряджається
DPSP_z	Значення критерію дефіциту енергії за рік роботи, коли акумуляторна батарея заряджається
DPSP_r	Значення критерію дефіциту енергії за рік роботи, коли акумуляторна батарея розряджається
COE	Вартість системи
Conf_id	Атрибут, який виступає зовнішнім ключем відношення Configurations зв’язує таблиці Configurations і Coefficients

Таблиця 3.7 – Таблиця атрибутів сутності «Weather_data»

Назва атрибута	Характеристика атрибута
W_id	Атрибут, який виступає первинним ключем відношення Weather_data і позначає його порядковий номер. Зв’язує таблиці Weather_day і Weather_data

Продовження таблиці 3.7

Назва атрибута	Характеристика атрибута
Duration_of_the_day	Продовжність дня з найгіршими погодними даними
Time	Період часу дня
Chromancy	Хмарність
Temperature	Температура
Wind_speed	Швидкість вітру
Gtilt	Погодинні значення притоку сонячної радіації
EL	Погодинні значення використання електроенергії людиною у господарстві
Date	Дата

Зв’язки між об’єктами:

- Конфігурації (Configurations) складаються з сонячних батарей (PV).
- Конфігурації (Configurations) складаються з вітрових установок (WP).
- Конфігурації (Configurations) складаються з акумуляторних батарей (B).
- Сонячні батареї (PV) містяться в конфігураціях (Configurations).
- Вітрові установки (WP) містяться в конфігураціях (Configurations).
- Акумуляторні батареї (B) містяться в конфігураціях (Configurations).
- Коефіцієнти гібридної вітро-сонячної системи електропостачання (Coefficients) розраховуються на основі згенерованих конфігурацій (Configurations).
- Конфігурації (Configurations) визначають коефіцієнти гібридної вітро-сонячної системи електропостачання (Coefficients).

- Конфігурації (Configurations) розраховуються для дня з найгіршими погодними умовами (Weather_data).
- День з найгіршими погодними умовами (Weather_data) визначає конфігурації (Configurations).

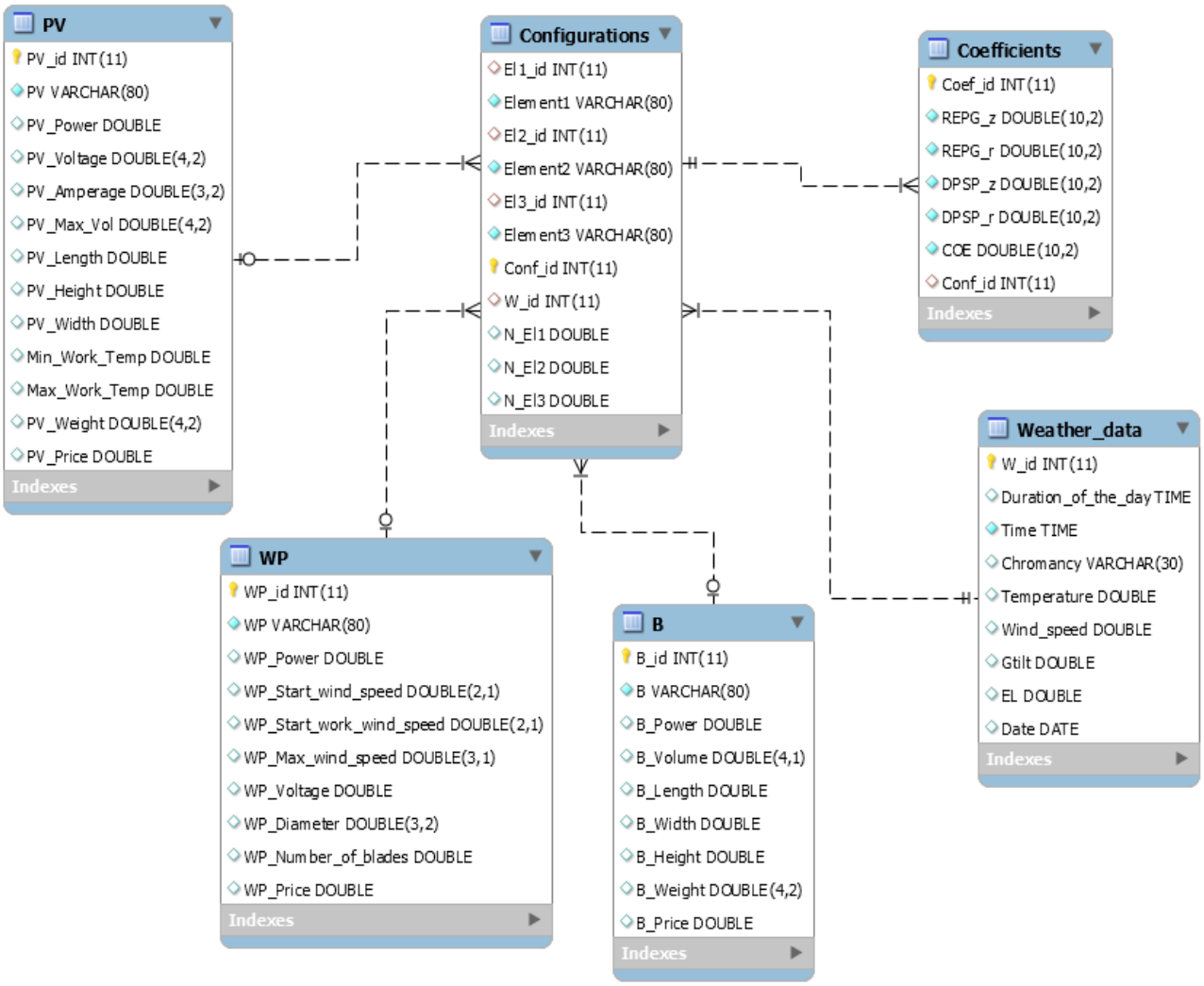

Рисунок 3.10 – Концептуальна модель бази даних

3.8 Логічне проектування предметної області при інтеграції на рівні баз даних

Логічне проектування — це розроблення структур зберігання, методів доступу й логічної структури системи баз даних без прив’язки до конкретної системи управління базами даних (СУБД).

Ціль етапу – створення логічної моделі даних на рівні моделі записів шляхом уточнення і перетворення концептуальної моделі з урахуванням особливостей обраної організації даних у цільовий СУБД (наприклад, реляційна чи мережева модель). Для реляційних моделей на цьому етапі фіксується набір і структура таблиць, що представляють сутності, визначаються ключі і проводиться нормалізація таблиць.

Нормалізація таблиць. Проаналізувавши сутності «PV», «WP» та «B» визначаємо, що вони не потребують нормалізації, так як вони перебувають в атомарному стані та всі неключові атрибути відношення залежать тільки від первинного ключа.

Виконаємо аналіз сутності «Weather_data». Атрибути «Gtilt» та «EL» функціонально залежить від первинного ключа, ці дані не стосуються погоди, тому доцільно було б виділити їх в окрему сутність «Additionally». Так як розрахунок конфігурацій відбувається подобово, а не погодинно, то слід виділити окрему сутність «Connect», яка буде пов’язувати таблиці All_Configurations та Additionally. У розрахунках визначається день з найгіршими погодними умовами, тому таблицю слід розділити та додаткові три сутності:«Weather_Archive», «Weather_data» та окрему таблицю для збереження дня з найгіршою погодою «Weather_day».

Проаналізуємо таблицю «Configurations». Не ключові відношення залежать як від первинного ключа так і від не первинних. Атрибути сутності «Configurations» перебувають не в атомарному (простому) стані, саме тому слід виділити з атрибутів даної сутності інші сутності, щоб вони складалися з неподільних значень. Виділяємо дві окремі сутності «All_Configurations», «Working_Configurations» та «Working_Configurations_with_AB». У сутності «All_Configurations» будуть міститися дані про конфігурації, створені з альтернативних джерел енергії, а у сутності «Working_Configurations» будуть міститися вже проаналізовані конфігурації з визначеною кількістю елементів сонячних батарей та вітроелектроустановок. Сутність «Working_Configurations_with_AB» пов’язана з сутністю

«Working_Configurations», до якої додаються акумуляторні батареї. На рис.3.11 зображена логічна модель бази даних.

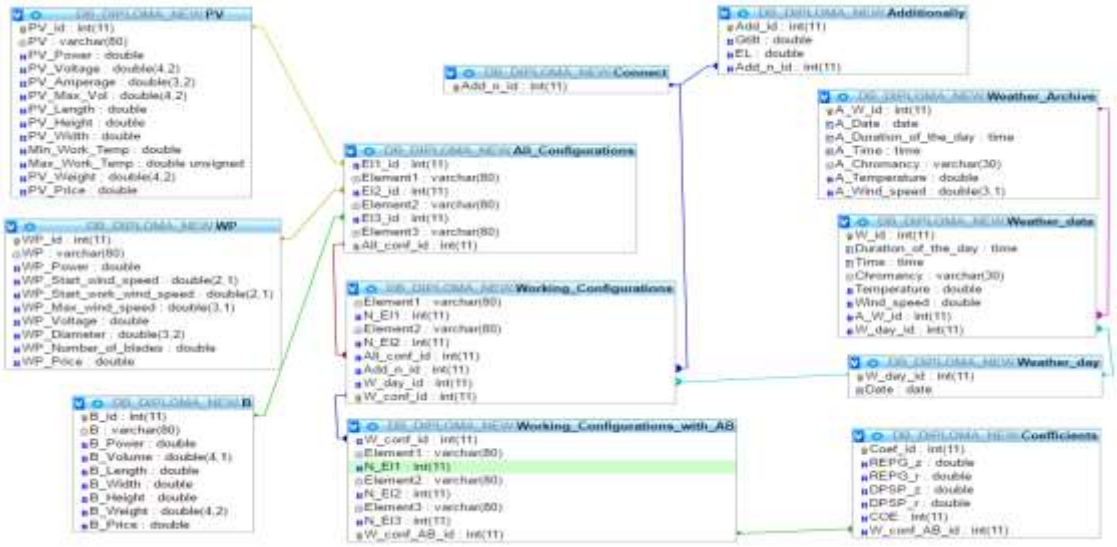

Рисунок 3.11 – Логічна модель бази даних

3.9 Функціональні залежності при проектування бази даних інформаційної системи

Функціональна залежність — концепція, що лежить в основі багатьох питань, пов'язаних з реляційними базами даних, включаючи, зокрема, їхнє проектування. Математично являє собою бінарне відношення між множинами атрибутів даного відношення і є, по суті, зв'язком типу «один-до-багатьох».

Функціональні залежності відображають зв'язки між атрибутами, які властиві реальному об'єкту. Атрибут B функціонально залежить від A, якщо кожному значенню A відповідає в точності одне значення B.

Функціональні залежності розроблюваної бази даних:

- сутність «PV»: «PV», «PV_Power», «PV_Voltage», «PV_Amperage», «PV_Max_Vol», «PV_Length», «PV_Height», «PV_Width», «Min_Work_Temp», «Max_Work_Temp», «PV_Weight», «PV_Price» → {«PV_id»};

- сутність «WP»: «WP», «WP_Power», «WP_Start_wind_speed», «WP_Start_work_wind_speed», «WP_Max_wind_speed», «WP_Voltage», «WP_Diameter», «WP_Number_of_blades», «WP_Price» → {«WP_id»};
- сутність «B»: «B», «B_Power», «B_Volume», «B_Length», «B_Width», «B_Height», «B_Weight», «B_Price» → {«B_id»};
- сутність «All_Configurations»: «El1_id» → {«Element1»};
- сутність «All_Configurations»: «El2_id» → {«Element2»};
- сутність «All_Configurations»: «El3_id» → {«Element3»};
- сутність «All_Configurations»: «Element1», «Element2», «Element3» → {«All_conf_id»};
- сутність «Working_Configurations»: «W_conf_id», «Add_n_id», «W_day_id» → {«Element1», «Element2»};
- сутність «Working_Configurations_with_AB»: «W_conf_id» → {«Element1», «Element2», «Element3»};
- сутність «Working_Configurations_with_AB»: «Element1», «Element2», «Element3» → {«W_conf_AB_id»};
- сутність «Working_Configurations»: «Element1», «N_El1», «Element2», «N_El2» → {«W_conf_id»};
- сутність «Additionally»: «Gtilt» → {«Add_id»};
- сутність «Additionally»: «Add_n_id» → {«EL»};
- сутність «Weather_day»: «W_day_id» → {«Date »};
- сутність «Weather_data»: «A_W_id» → {«Duration_of_the_day», «Time», «Chromancy», «Temperature», «Wind_speed», «W_day_id»};
- сутність «Weather_data»: «Duration_of_the_day», «Time», «Chromancy», «Temperature», «Wind_speed» → {«W_id»};
- сутність «Weather_Archive»: «A_Date», «A_Duration_of_the_day», «A_Time», «A_Chromancy», «A_Temperature», «A_Wind_speed» → {«A_W_id»};

- сутність «Coefficients»: «W_conf_AB_id» → {«REPG_z», «REPG_r», «DPSP_z», «DPSP_r», «COE»};
- сутність «Coefficients»: «REPG_z», «REPG_r», «DPSP_z», «DPSP_r», «COE» » → {«Coef_id»}.

Визначення первинних та зовнішніх ключів:

Для сутності «PV»: «PV_id» – первинний.

Для сутності «WP»: «WP_id» – первинний.

Для сутності «B»: «B_id» – первинний.

Для сутності «All_Configurations»: «All_conf_id» – первинний.

Для сутності «Working_Configurations»: «W_conf_id» – первинний, «All_conf_id», «Add_n_id», «W_day_id» – зовнішні.

Для сутності «Working_Configurations_with_AB»: «W_conf_AB_id» – первинний, «W_conf_id » – зовнішній.

Для сутності «Coefficients»: «Coef_id» – первинний, «W_conf_AB_id» – зовнішній.

Для сутності «Connect»: «Add_n_id» – первинний.

Для сутності «Additionally»: «Add_id» – первинний, «Add_n_id» – зовнішній.

Для сутності «Weather_day»: «W_day_id» – первинний.

Для сутності «Weather_data»: «W_id» – первинний, «A_W_id», «W_day_id» – зовнішні.

Для сутності «Weather_Archive»: «A_W_id» – первинний.

3.10 Висновки

Функціональне моделювання процесів, які забезпечують інформаційну підтримку прийняття рішень при плануванні структури ГЕ дозволило отримати моделі та алгоритмічне забезпечення реалізації задач процесу збору та обробки інформації, визначення загальної кількості альтернатив та техніко-економічних

параметрів, формування рішення щодо оптимальної структури енергетичної системи.

У роботі наведено функціональну модель, опис етапів розробленої прикладної інформаційної технології підтримки прийняття рішень при плануванні структури ГЕ.

Для реалізації запропонованих функціональних моделей запропоновано відповідну СППР, що є програмним веб-додатком із трирівневою архітектурою типу клієнт-сервер.

ВИСНОВКИ

Запропонований комплекс моделей для СППР оператора-керівника автоматизованих систем керування створює основу для автоматизації процесу надання операторові-керівникові інформаційної підтримки при ухваленні рішення про прийняття ергономічних розв'язків

У результаті проведених досліджень установлено факт приділення недостатньої уваги питанням управління при комп’ютеризації проектних робіт. Для реалізації можливості управління процесом проектування зроблено системний аналіз процесу проектування технічних об’єктів, запропоновано зв’язки системи автоматизації проектних робіт із зовнішнім середовищем, встановлена необхідність наявності трьох видів програмного забезпечення для функціонування системи, представлено обґрунтоване виконання проектних робіт за схемою «зверху-вниз», складено інформаційну модель для підтримки процесу проектування та отримано архітектурне рішення для системи автоматизації проектних робіт в організаціях, виявлено відсутність інформаційного опису процесів діяльності, який дозволяє їх супроводжувати протягом усього життєвого циклу.

Єдина модель сприяє автоматизації роботи виконавців із різними рівнями повноважень, забезпечує однозначність розуміння ними виконуваних завдань

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Нікольський Ю.В. Дискретна математика./ Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина — К.: Видавнича група BHV, 2007. — 368 с. —ISBN 966-552-201-9.
2. Пасічник В.В. Організація баз даних та знань. / В.В. Пасічник, В.А. Резніченко — К.: Видавнича група BHV, 2006. — 384 с. — ISBN 966-552-156-X.
3. Дейт К. Введение в системы баз данных,/ К. Дж. Дейт.8-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1328 с. - ISBN 978-5-8459-0788-2.
4. Карпова, Т. С. Базы данных: модели, разработка, реализация / Т. С. Карпова. - СПб.: Питер, 2002. - 304 с. - ISBN: 5-272-00278-4.
5. Кулямин В.В. Методы верификации программного обеспечения /В.В. Кулямин // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. - 117 с.
6. Abran A., Moore J. Swebok: Guide to the Software Engineering Body of Knowledge. http://www.swebok.org/
7. Дейкстра Э. Дисциплина программирования / Э. Дейкстра Пер. с англ. Под ред. Э.З.Любимского - М.: Мир, 1978 - 276 с. (Серия Программное обеспечение ЭВМ)
8. Formal Verification http://www.nist.gov/dads/HTML/formalverf.html
9. Грис Д. Наука программирования. М.: Мир. 1984. – 416 с.
10. Кларк Э.М. Верификация моделей программ. Model Checking./ Э.М. Кларк, О. Грамберг, Д. Пелед - М.: МЦНМО. 2002. — 416 с. — ISBN 978-5-94057-054-7.
11. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем / Ю.Г. Карпов - СПб.: БХВ-Петербург, 2010. — 560 с. — ISBN: 978-5-9775-0404-1.

12. Sergio Tessaris; Enrico Franconi; Thomas Eiter (2009). Reasoning Web. Semantic Technologies for Information Systems: 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures. Springer. p. 112. — ISBN 978-3-642-03753-5.
13. Emerson E. A. Temporal and modal logic // Handbook of Theoretical Computer Science. Chapter 16. 1990.
14. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. — СПб.: Питер, 2009. — 176 с. — ISBN 978-5-388-00692-9.
15. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. http://is.ifmo.ru/books/switch/1
16. Kaner C., Falk J., Nguyen Q. Testing Computer Software. NY: Wiley. 1999. p. 496 ISBN 978-0-471-35846-6
17. Бурдонов И. Б., Косачев А. С., Кулямин В. В. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай // Программирование. 2003, № 5. http://www.ispras.ru/~igor/doctor/papers_2003/3/1.html
18. Хопкрофт Дж Введение в теорию автоматов, языков и вычислений./ Дж. Хопкрофт, Р. Мотвани, Дж. Ульман - 2-е изд. Пер. с англ. — М.: Вильямс, 2008. — 528 с.: ил. — ISBN 978-5-8459-1347-0
19. Васильева К.А. Верификация автоматных программ с использованием LTL / К. А. Васильева, Е.В. Кузьмин// Моделирование и анализ информационных систем. Ярославль: ЯрГУ. Т. 14. № 1. 2007. http://is.ifmo.ru/verification/_LTL_for_Spin.pdf
20. Alur R., Yannakakis M. Model Checking of Hierarchical State Machines // Proceedings of the 6th ACM Symposium on Foundations of Software Engineering. 1998 P. 175-188
21. Вельдер С.Э., Шалыто А. А. Введение в верификацию автоматных программ на основе метода Model checking. http://is.ifmo.ru/verification/modelchecking/

22. Holzmann G. Spin Model Checker: Primer and Reference Manual. NJ: Addison-Wesley Professional. 2003. p. 596. - ISBN 978-0321773715
23. Bogor. Software Model Checking Framework. 2005 http://bogor.projects.cs.ksu.edu/manual/
24. Риган П., Хэмилтон С. NASA: Миссия надежна // Открытые системы. 2004, № 3. http://www.osp.ru/os/2004/03/184060/
25. Stateflow — Design and simulate state machines and control logic http://www.mathworks.com/products/stateflow/
26. Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А. UML. SWITCH-Технология. Eclipse //Информационно-управляющие системы. 2005. №6(13). http://is.ifmo.ru/works/uml-switch-eclipse/
27. Нікольський Ю.В. Дискретна математика./ Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина — К.: Видавнича група BHV, 2007. — 368 с. —ISBN 966-552-201-9.
28. Пасічник В.В. Організація баз даних та знань. / В.В. Пасічник, В.А. Резніченко — К.: Видавнича група BHV, 2006. — 384 с. — ISBN 966-552-156-X.
29. Лукин Верификация параллельных автоматных программ /М.А. Лукин // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики, 2014. № 1 (89), с. 145 – 162.
30. Вельдер С.Э. Верификация автоматных программ: Учебное пособие./ С.Э. Вельдер, М.А. Лукин, А.А. Шалыто, Б.Р. Яминов - СПб.: СПбГУ ИТМО, 2011. - 242 с.
31. Егоров К.В. Методика верификации автоматных программ / К.В. Егоров, А.А. Шалыто // Информационно-управляющие системы № 5, 2008 – С. 15-21.
32. Первушин Е.В. Моделирование банкомата,/ Е.В. Первушин, А.А. Шалыто, СПбГУ ИТМО, 2003, http://is.ifmo.ru/projects/.
33. SPIN, http://spinroot.com/spin/whatispin.html.

34. Васильева К.А. Верификация автоматных программ с использованием LTL / К. А. Васильева, Е. В. Кузьмин, // Моделирование и анализ информационных систем, 14:1 (2007), С. 31–43.
35. Безверха М.А. Конструювання та верифікація програм на основі специфікацій у композиційно-номінативній мові CNLS / М.А. Безверха, П.П. Процик// Проблеми програмування. — 2010. — № 2-3. — С. 340-348
36. Максимец А.Н. Верификация программ: состояние, проблемы, экспериментальные результаты. I / А.Н. Максимец // Проблеми програмування. — 2013. — № 4. — С. 53-63.
37. Максимец А.Н. Верификация программ: состояние, проблемы, экспериментальные результаты. II / А.Н. Максимец // Проблеми програмування. — 2014. — № 1. — С. 76-89.
38. Kulvatunyou, B. A functional approach to enterprise-based engineering activities [Text] / B. Kulvatunyou, R. A. Wysk // Journal of Manufacturing Systems. – 2000. – V. 19. – №. 2. – P. 156 – 171.
39. O'Brien, W. J. Challenges, approaches and architecture for distributed process integration in heterogeneous environments [Text] / W.J O'Brien, H. Joachim, M. Siddiqui, O. Topsakal // Advanced Engineering Informatics. – 2008. – V. 22. – №. 1. – P. 28 – 44.
40. Kishore, R. Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems [Text] / R. Kishore, H. Zhang, R. Ramesh // Decision Support Systems. – 2006. – V. 42. – №. 1. – P. 48 – 78.
41. Feng, S. C. A manufacturing process information model for design and process planning integration [Text] / S. C.Feng, E. Y. Song // Journal of Manufacturing Systems. – 2003. – V. 22. – №. 1. – P. 1 – 15.
42. Файзрахманов, Р. А. Структурно функциональный подход к проектированию информационных технологий и автоматизированных систем с использованием case-средств [Текст] / Р. А. Файзрахманов, К. А. Селезнев ; Перм. гос. техн. ун-т. – Пермь, 2005. – 245 с.

43. Kettinger, W. J. Business process change: a study of methodologies, techniques, and tools [Text] / W. J.Kettinger, J. T. C. Teng, S.Guha // MIS quarterly. – 1997. – P. 55 – 80.
44. Репин, В. В. Процессный подход к управлению. Моделирование бизнес-процессов [Текст] / В. В. Репин, В. Елиферов. – М. : Манн, Иванов и Фербер, 2013. – 544 с.
45. Ferrer, I. An approach to integrate manufacturing process information in part design phases [Text] / I. Ferrer, J.Rios, J. Ciurana // Journal of Materials Processing Technology. – 2009. – V. 209. – №. 4. – P. 2085 – 2091.
46. Ramachandra T. V. RIEP: Regional integrated energy plan / Ramachandra. // Renewable and Sustainable Energy Reviews. – 2009. – №13. – С. 285–317.
47. Кочовий А. Б. Оперативна оптимізація усталених режимів електричних мереж енергопостачальних компаній за умов невизначеності / Андрій Богданович Кочовий // Автореферат дисертації на здобуття наукового ступеня кандидата технічних наук. – Львів: «Львівська політехніка», 2010. – 17 с.
48. Лежнюк П. Д. Застосування принципу найменшої дії для оптимізації режимів електроенергетичних систем / П. Д. Лежнюк, В. В. Нетребський // Вісник національного університету «Львівська політехніка». – 2009. - No637. – С. 44 – 50.
49. Мірошник О. О. Структура втрат електричної енергії та методика їх розрахунку / О. О. Мірошник // Вісник ХДТУСГ. Проблеми енергозабезпечення та енергозбереження в АПК України. – Х: ХДТУСГ, 2004. – вип. 27. – т. 1. – С. 25 – 30.
50. Тимчук С. А. Совершенствование методологии поиска рациональных решений в условиях многокритериальности и неопределенности исходной информации на примере системы электроснабжения / С. А. Тимчук, Н. М. Черемисин // Енергетика та електрифікація. – 2013. - №4. – С. 53 – 60.