Тема : ***«Інформаційна технологія конструювання варіанту використання та концептуальних класів»***

Шифр – ***«Ice Nine»***

ЗМІСТ

ВСТУП

Варіанти використання (use case, прецеденти) часто використовуються в проектуванні інформаційних систем як найбільш точний і повний спосіб формулювання функціональних вимог [1,2]. Опис варіанту використання (ВВ) – трудомісткий і відповідальний етап роботи над формулюванням вимог [4], від якості виконання яких значною мірою залежить успішність проекту. Не дивлячись на досить велику кількість публікацій, що в тій чи іншій мірі визначають правила складання [3-5] і використання [5-8] ВР, до теперішнього часу не існувало класифікації можливих пунктів їх сценаріїв. Це призводило до того, що в проектуванні програмних продуктів (ПП) прецедент розглядався як заздалегідь описаний елемент діаграм [8-10] і питанням автоматизації його складання не приділялася увага.

Виходячи з того, що ВВ – це сценарій роботи користувача з майбутнім ПП, системний аналітик при його описі обов'язково продумує архітектуру ПП. Однак ця інформація ніяк не фіксувалася на етапі формулювання вимог, що фактично призводило до дублювання роботи на етапі реалізації ВР.

В роботі [11] вперше запропоновано класифікацію пунктів сценаріїв прецедентів. На її основі побудовані математичні моделі, які дозволили автоматизувати процес складання ВВ шляхом надання системному аналітику шаблонів для кожного типу пункту. Крім цього була зроблена спроба зафіксувати мінімально необхідну кількість класів (концептуальні класи), необхідних для забезпечення реалізації ВР.

Таким чином, метою представленої наукової роботи є підвищення продуктивності створення програмного забезпечення, зменшення часу виконання проекту, зменшення кількості помилок при написанні сценаріїв варіантів використання, усунення дублювання робіт по відношенню до концептуальних класів.

Завдання, які необхідно вирішити для досягнення мети наукової роботи:

1. Розробка математичної моделі для автоматизованого конструювання варіантів використання, їх сценаріїв, концептуальних класів;
2. Розробка алгоритмів функціонування елементів системи та взаємодії між ними;
3. Програмна реалізація математичної моделі та алгоритмів, що розроблювалися.

У даній роботі представлені результати рішення вище перелічених завдань та наведено висновки щодо повноти досягнення поставленої цілі.

1 КЛАСИФІКАЦІЯ І МОДЕЛІ ПУНКТІВ СЦЕНАРІЇВ ВАРІАНТІВ ВИКОРИСТАННЯ

1.1 Класифікація пунктів сценаріїв

Вимоги до опису прецеденту підрозділяються на вимоги з боку замовника і розробника.

Вимоги з боку замовника інформаційної системи (ІС):

1. Повинен описувати деяку послідовність дій, які переводять проектовану систему в новий стан (відчутний результат);

2. Повинен використовувати мову замовника;

3. Повинен бути схожим на дії замовника в його предметної області.

Вимоги з боку розробника ІС:

1. Опис повинен мати два розділи: основний і додатковий сценарій (розширення);
2. У кожному пункті система повинна виконувати деякі дії;
3. Для кожної дії повинно бути зазначена дійова особа;
4. Якщо інформація, що вводиться вимагає перевірки, то в основному сценарії система дає позитивний результат (підтверджує), а в додатковому – кожен випадок негативного результату представлений окремо своїм сценарієм;
5. Максимальна кількість даних, що вводяться і перевіряються – 2;
6. Якщо система в результаті пошуку або розрахунку виводить деякі дані, які повинен оцінити ініціатор прецеденту, то в основному сценарії він дає позитивну відповідь, а в додатковому сценарії розглядаються дії, пов'язані з його негативною відповіддю;

7. Якщо система виконує деякий розрахунок або перевірку даних, що вводяться, то бажано повідомити, які ресурси будуть при цьому задіяні.

Чинними і зацікавленими особами в прецеденті можуть бути люди і системи:

1. Користувач системи, який буде з нею безпосередньо спілкуватися для виконання своїх професійних обов'язків;
2. Клієнт – замовник деякої послуги;
3. Інша система;
4. Система, що розроблюється.

На підставі вимог, що пред'являються до опису прецеденту, а також аналізу безлічі сценаріїв в різних предметних областях з точки зору характеру взаємодії користувача і програмного продукту запропоновано певний перелік типів пунктів сценаріїв.

У таблиці 1.1 наведено створену класифікацію пунктів прецедентів та їх короткий опис, що пояснює яку роль кожен з них відіграє у повній картині сценарію варіанту використання.

Таблиця 1.1 – Класифікація пунктів сценаріїв

Назва	**Опис**
Створити	Користувач наказує системі створити певний документ (об'єкт), який в залежності від положення відповідного пункту в сценарії може грати роль деякого шаблону для накопичення інформації, або звіту про виконану роботу.
Ввести данні	Користувач вводить в систему ряд даних, для яких система зазвичай має перевірити можливість їх використання для подальшої роботи.

Продовження таблиці 1.1

Запросити значення	Користувач запитує у системи деяке дане. Зазвичай після цього слідує оцінка даного користувачем.
Запросити список	Користувач замовляє список (наприклад, даних, послуг або документів) для подальшого вибору з нього деяких елементів.
Вибрати зі списку	Користувач вибирає зі списку потрібне дане або послугу (документ).
Ввести послугу (документ)	Користувач вводить необхідну послугу або документ, який визначає подальшу послідовність дій. Наприклад, спосіб оплати по банківській картці.
Повторення дій	Користувач має можливість перейти до розташованих вище пунктів сценарію, або відмовитися від їх повторення.
Пункт, що вільно конструюється	Очевидно, можлива ситуація, коли користувач не виявив потрібної йому операції серед запропонованих дій з системою. У цьому випадку йому запропоновано створити пункт, що вільно конструюється.
Завершення прецеденту	Виконання пункту може передбачати прийом деяких даних, які не вимагають аналізу (наприклад, введення прізвища та адреси замовника деякої послуги), реєстрацію певних даних (реєстрація проданого квитка), створення документа (чек, квитанція про сплату послуги).

1.2 Моделі пунктів сценаріїв

На рисунку 1.1 представлена структура моделі пункту сценарію. Модель опису пункту входить в опис прецеденту. Модель проектування пункту не має представлення у вимогах і є початковим етапом проектування.

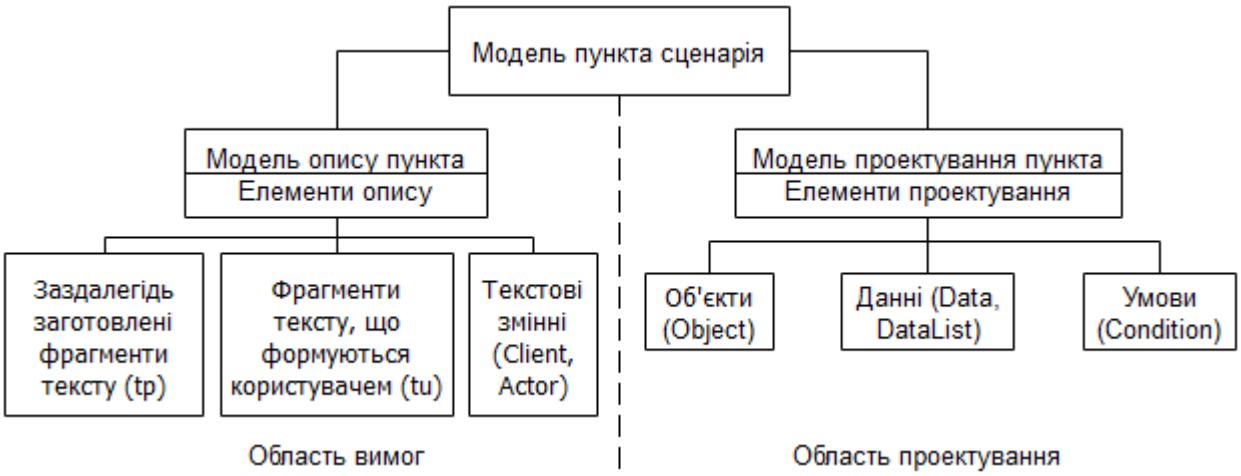

Рисунок 1.1 – Структура моделі пункту сценарія

Елементами опису можуть бути заздалегідь заготовлені фрагменти тексту, фрагменти тексту, що формуються в процесі складання прецеденту, або текстові змінні, що представляють виконавців і зацікавлених осіб. Елементами проектування є об'єкти концептуальних класів або посилання на об'єкти, окремі дані або їх списки, які повинні бути введені в ПП або отримані з ПП, а також умови виконання операцій. Значення даних і умов можуть вводитися системним аналітиком для подальшого використання в процесі тестування прецеденту.

У моделі використовується ряд метасимволов. Окремі елементи або групи елементів, укладені в квадратні дужки – «[]», не обов'язково повинні бути присутніми в описі пункту ВВ. Символ «+» позначає конкатенацію рядків. Символ «/» означає використання одного з двох елементів, розділених цим символом.

Пункт «Створити». Якщо об'єкт створюється в рамках пункту, то попередньо заданих умов немає. Перевіряти дані одночасно з введенням умов

їх перевірки не має сенсу. Тому з моделі пункту виключається необов'язкова умова і відповідний фрагмент тексту – "система підтверджує", а також можлива перевірка атрибутів. Крім цього, уточнюємо, що можливе повторення частини опису. В результаті опис пункту приймає вид (1.1):

$$Create = \langle N, [Client, tp1, Actor, tp2, tu1], Actor, tp3, Object, [tp5, \langle tu2, Data \rangle] \rangle \quad (1.1)$$

Розшифруємо позначення, наведені у формулі (1.1):

- N – номер пункту прецеденту,
- $Client$ – необов’язковий елемент; вводиться, якщо необхідно визначити, від кого надходить ініціатива;
- $tp1$ – «звертається до»;
- $tp2$ – «з приводу»;
- $tp3$ – «створює в системі»;
- $tu1$ – текст, що формується користувачем (наприклад, «замовлення на ремонт устаткування»);
- $Actor$ – користувач, який взаємодіє з системою (повинен бути визначеним у введені до прецеденту);
- $Object$ – назва об’єкта, що створюється; передбачується перевірка на відсутність об’єкта з подібною назвою.

При створені об’єкта з параметрами використовується фрагмент з (1.1) $tp5, \langle tu2, Data \rangle$, де $tp5$ – «з параметрами».

Кортеж $\langle tu2, Data \rangle$ визначає операцію опису і введення значень атрибутів об’єкта, яка може повторюватися:

- $tu2$ – назва атрибута, який формується користувачем (наприклад, номер телефону);
- $Data$ – значення атрибута.

Пункт «Ввести данні». Модель цього пункту описується наступною формулою і приймає вид (1.2):

$$InputData = \langle N, [Client, tp1, tu1], Actor, tp2, DataList, [tp3, tp4] \rangle \quad (1.2)$$

Розшифруємо позначення, наведені у формулі (1.2):

- $tp1$ – «надає дані про»;
- $tp2$ – «вводить у систему»;
- $tp3$ – «Система перевіряє данні»;
- $tp4$ – «Система підтверджує коректність даних»;
- $tu1$ – текст, що формується користувачем, наприклад, «місце проживання»;
- $DataList = \{d_1, d_2, \dots, d_n\}$ – список даних, що вводяться.

Кожне дане представляє собою наступний кортеж (1.3):

$$d_i = \langle name, addrCheck, addrKeep \rangle \quad (1.3)$$

Розшифруємо позначення, наведені у формулі (1.3):

- $name$ – назва даного (запис, що фіксується);
- $addrCheck$ – орієнтовна адреса об’єкта, який виконує перевірку даного (службова інформація). Якщо адреса відсутня, то значення даного не перевіряється. Якщо у списку буде вказано більше двох даних для перевірки, то аналітику виводиться рекомендація щодо розділення даного пункту сценарія на два пункти. Якщо у списку немає даних для перевірки, то аналітику виводиться рекомендація про об’єднання даного пункту з сусіднім пунктом;
- $addrKeep$ – орієнтовна адреса об’єкта, що зберігає значення даного (службова інформація).

Пункт «Запросити значення». Користувач запитує у системи деяке дане. Зазвичай після цього слідує оцінка даного користувачем. Тут вказується на основі яких даних було визначено виведене значення. Наприклад, пункт може бути сформульовано так: «Визначити вартість кошика на підставі кількості та вартості окремих товарів ....». Опис пункту приймає вид (1.4):

$$Inquiry = \langle N, [Client, tp1, tu1], Actor, tp2, Data_q, tp3, Data_b, [tp4, Condition, tp5], tp6 \rangle \quad (1.4)$$

Розшифруємо позначення, наведені у формулі (1.4):

- $Client$ – необов'язковий елемент (вводиться, якщо необхідно визначити, від кого направлена ініціатива);
- $tp1$ – «бажає отримати»;
- $tu1$ – фраза, що формується користувачем, наприклад, «вартість послуги»;
- $tp2$ – «вводить у систему запит на отримання»;
- $Data_q = \langle dataName_q, addrKeep_q \rangle$ – дане, що запитується у системи; $addrKeep_q$ – назва об'єкта, що містить дане;
- $tp3$ – «на основі»;
- $Data_b = \langle dataName_b, addrKeep_b \rangle$ – данні, на основі яких визначається значення, що запитується;
- $tp4$ – «при умові»;
- $Condition = \langle textCond, addrKeep \rangle$ – умова отримання даного, наприклад, $textCond$ може мати значення «за минулий місяць»; $addrKeep$ – назва об'єкта, що здатний перевірити і виконати умову;
- $tp5$ – «Система підтверджує коректність умови» – необов'язковий елемент;
- $tp6$ – «Система виводить + $nameData$ + $Client/Actor$ + згодний».

Пункт «Запросити список». Модель цього пункту описується наступною формулою і приймає вид (1.5):

$$ListInquiry = \langle N, [Client, tp1, tu1], Actor, tp2, List, [tp3, Condition, tp4], tp5 \rangle \quad (1.5)$$

Розшифруємо позначення, наведені у формулі (1.5):

- $tp1$ – «бажає отримати список»;
- $tu1$ – фраза, що формується користувачем, наприклад, «послуг, що надаються».
- $tp2$ – «вводить у систему запит на отримання»;

- $List = \langle listName, addrKeep \rangle$ – список, що запитується у системи; $listName$ – назва списку, $addrKeep$ – місце зберігання (формування) списку;
- $tp3$ – «за умовою»;
- $Condition = \langle textCond, addrKeep \rangle$ – умова отримання даного, наприклад, $textCond$ може бути представлено текстом «за минулий місяць»; $addrKeep$ – назва об'єкта, спроможного перевірити і виконати умову;
- $tp4$ – «Система підтверджує коректність умови";
- $tp5$ – «Система знаходить і виводить $listName$».

Пункт «Вибрати зі списку». Модель цього пункту описується наступною формулою і приймає вид (1.6):

$$SelectElem = \langle N, [Client, tp1], Actor, tp2, Elem, tp3, tp4 \rangle \quad (1.6)$$

Розшифруємо позначення, наведені у формулі (1.6):

- $tp1$ – «вказує на потрібний елемент зі списку»;
- $tp2$ – «обирає зі списку»;
- $Elem = \langle elemName, elemType \rangle$ – елементи можуть бути двох типів (завжди однакові для усього списку): $elemType = Data/Control$; якщо тип елемента $Data$, то система повинна надати $Client/Actor$ дані, що запитуються; якщо тип елемента $Control$, то йому відповідає множина пар $\{\langle elemName_i, N_i \rangle\} i = 1, k$; N_i – номер пункту сценарія, якому буде передано управління, k – кількість елементів списку);
- $tp3$ – «і вводить его в систему»;
- $tp4$ – «Система надає значення + $Data$ + $Client/Actor$ + згодний»;
- $tp5$ – «Відбувається перехід до пункту + N_i + сценарія».

Пункт «Ввести послугу (документ)». Модель цього пункту описується наступною формулою і приймає вид (1.7):

$$InputService = \langle N, [Client, tp1/tp2, tu1], Actor, tp3, Service, tp4 \rangle \quad (1.7)$$

Розшифруємо позначення, наведені у формулі (1.7):

- $Service$ – назва послуги (документа), що визначає подальші дії; $Service = \langle serviceName, Nm \rangle$, де $serviceName$ – назва сервісу (документа), Nm – номер пункту сценарія, котрий відповідає вказаному сервісу;
- $tp1$ – «бажає працювати з документом»;
- $tp2$ – «бажає використати»;
- $tu1$ – текст, що визначає сервіс (наприклад, «огляд ходової частини», «регулювання розвалу/сходження») або документ (наприклад, «заява на надання пільгового тарифу»);
- $tp3$ – «вводить в систему»;
- $tp4$ – «Система перевіряє можливість виконання послуги (документа) і виконавця. Система підтверджує».

Пункт «Повторення дій». Модель цього пункту описується наступною формулою і приймає вид (1.8):

$$CyclAction = \langle N, Client/Actor, tp1, Np \rangle \quad (1.8)$$

Розшифруємо позначення, наведені у формулі (1.8):

- $tp1$ – «бажає повторити дії, починаючи з пункту”
- Np – номер пункту основного сценарія, що розміщений вище пункту N.

Пункт «Пункт, що вільно конструюється». Виходячи з того, що пункти, що входять до класифікації, покривають всі можливі атомарні дії з проектованою системою, користувачеві запропоновано самостійно вибирати дії в довільному порядку і кількості. В результаті модель пункту приймає вид (1.9):

$$FreeConstr = \langle N, [Client, tu1], Actor, tu2, mAction, nFinishPhrast \rangle \quad (1.9)$$

Розшифруємо позначення, наведені у формулі (1.9):

- $mAction$ – мультимножина дій, де кожний елемент відповідає одному з можливих типів пунктів сценарія;
- $nFinishPhrast$ – різні варіанти завершальних фраз пунктів.

Варіанти фраз, що завершують пункти сценарія:

- $el1FinishPhrast$ – «Система підтверджує коректність умови»;
- $el2FinishPhrast$ – «Система перевіряє і підтверджує можливість виконання послуги».
- $el3FinishPhrast$ – «Система підтверджує коректність даних»;
- $el4FinishPhrast$ – «Система підтверджує»;
- $el5FinishPhrast$ – $Client$ + «згодний»;
- $el6FinishPhrast$ – $Actor$ + «згодний».

Пункт «Завершити прецедент». Модель цього пункту описується наступною формулою і приймає вид (1.10):

$$Finish = \langle N, [Elem1], [Elem2][Elem3], tp1 \rangle \quad (1.10)$$

Розшифруємо позначення, наведені у формулі (1.10):

- $tp1$ – «Завершення роботи прецеденту»;
- $Elem1$, $Elem2$, $Elem3$ –т необов'язкові елементи, що дозволяють прийняти дані, що зареєструвати транзакцію і сформулювати документ відповідно.

Прийом даних приймає вигляд (1.11):

$$Elem1 = \langle Client/Actor, tp2, tu1, Data1, tp3, addrKeep1 \rangle \quad (1.11)$$

Розшифруємо позначення, наведені у формулі (1.11):

- $tp2$ – «повідомляє»;
- $tu1$ – назва даного, що надається клієнтом;
- $Data1$ – дане, котре повідомляє клієнт;
- $tp3$ – «Система зберігає дане у»;

- $addrKeep1$ – об’єкт, що приймає дане (об’єкт обирається з створених раніше).

Операція прийому даного може повторюватися декілька разів.

Реєстрація даних приймає вигляд (1.12):

$$Elem2 = \langle tp4, Data2, tp5, addrKeep2 \rangle \quad (1.12)$$

Розшифруємо позначення, наведені у формулі (1.12):

- $tp4$ – «Система реєструє дане:»;
- $Data2$ – дане (обирається з списку створених раніше);
- $tp5$ – «у»;
- $addrKeep2$ – об’єкт, що реєструє дане (можливе створення нового об’єкта).

Операція реєстрації даного може повторюватися декілька разів.

Створення документа приймає вигляд (1.13):

$$Elem3 = \langle tp6, Object, tp7, DataList3 \rangle \quad (1.13)$$

Розшифруємо позначення, наведені у формулі (1.13):

- $tp6$ – «Система видає документ:»;
- $Object$ – створений об’єкт (документ);
- $tp7$ – «що містить»;
- $DataList3$ – список даних, що включаються в документ (данні обираються з створених раніше).

1.3 Апробація

Відповідно до запропонованої моделі пунктів ВВ були розроблені алгоритми їх реалізації. Алгоритм формування пункту «Завершити прецедент» представлено на рисунку 1.2.

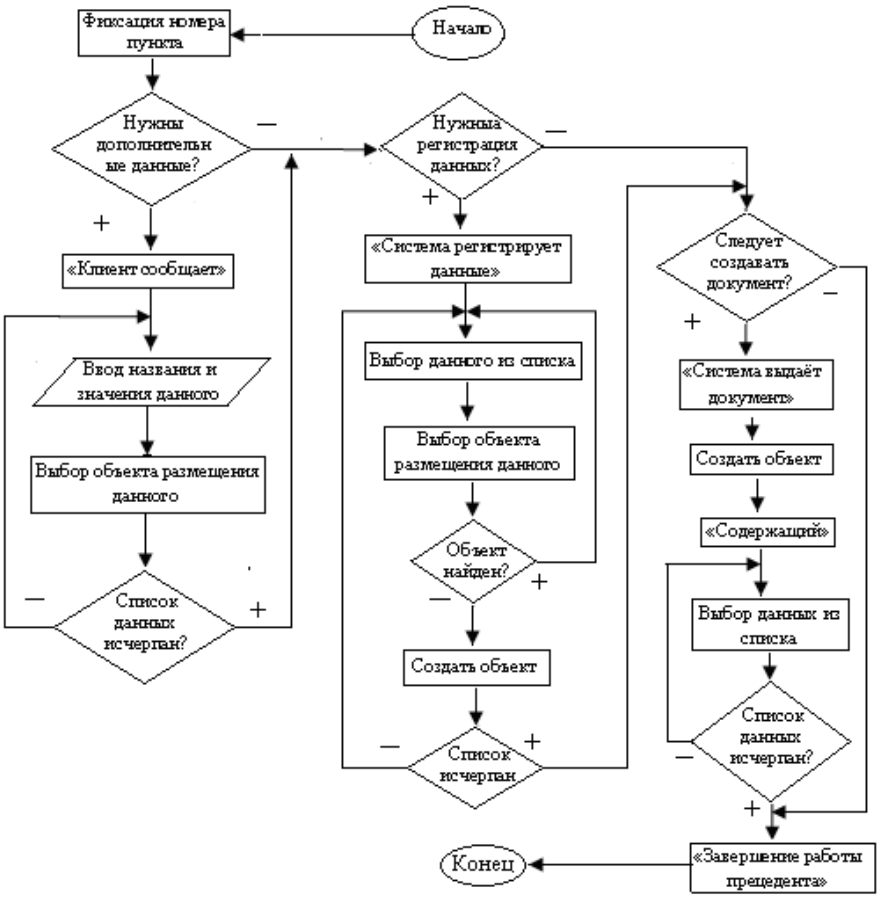

Рисунок 1.2 - Алгоритм формування пункту сценарію «Завершити прецедент»

Прийняті рішення реалізовані в програмному продукті UseCaseEditor. На рисунку 1.3 зображено скриншот вікна програми.

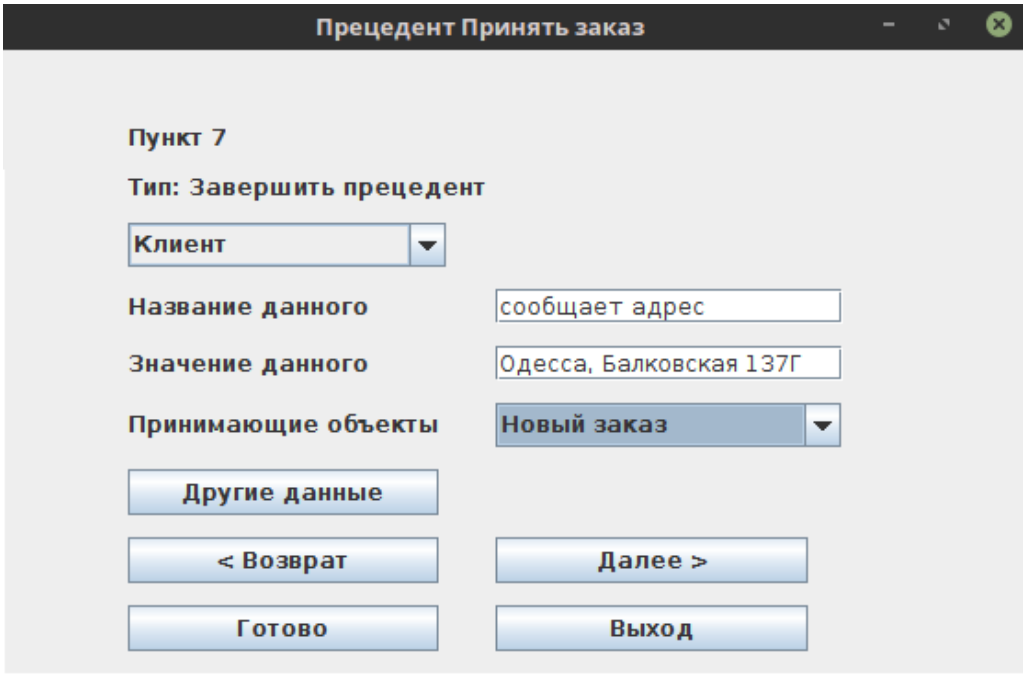

Рисунок 1.3 – Приклад формування пункту прецеденту «Завершити прецедент»

2 МЕТОД ВИЗНАЧЕННЯ КОНЦЕПТУАЛЬНИХ КЛАСІВ У ПРОЦЕСІ ОПИСУ ВАРІАНТІВ ВИКОРИСТАННЯ

2.1 Визначення проблеми

Загалом, моделі автоматизованого опису прецедентів не враховують попередній аналіз предметної області: визначення користувачів, визначення переліку (діаграми) прецедентів, розподіл обов'язків між користувачами. Можна на етапі опису прецедентів описувати концептуальні класи, що забезпечують виконання пунктів прецеденту, однак не фіксується інформація про обставини створення класу і його функціях.

Моделі тестування прецедентів, що забезпечують обхід всіх гілок обходу прецеденту, безпосередньо не пов'язані з функціями, які повинні виконувати концептуальні класи і даними, які повинні в них зберігатися. Потрібно дотримуватися балансу між описом вимог у вигляді прецеденту і попередніми проектуванням концептуальних класів. Другий процес не повинен затримувати перший і повинен бути представлений в обсязі, який необхідний для складання реальних сценаріїв роботи з майбутньою системою.

При описі різних прецедентів можливо дублювання користувачів, введення нових, помилковий перерозподіл їх обов'язків. Відсутність зв'язку концептуальних класів з пунктами прецеденту і фіксації в них виконуваних функцій може привести до дублювання класів і їх функцій. Відсутність зв'язку концептуальних класів з створюваними в них класами може привести до відсутності прецедентів, призначених для ініціалізації структур і даних, що забезпечують роботу користувачів в створюваній системі. Опис прецедентів володіє додатковою інформацією про підтримуючі класи і тестуванні стає громіздким і незрозумілим для замовника.

Можна створити загальну математичну модель опису та проектування прецедентів, що охоплює період попереднього збору необхідної для них інформації.

Також можна створювати і зберігати опис прецеденту в двох документах: «Опис прецеденту», «Проектування прецеденту». Другий документ дублює перший в частині словесного опису прецеденту, але надає всі рішення на рівні попереднього проектування.

Важливо створити механізм опису концептуальних класів, що визначає обставини їх створення і використання, механізм опису тестів, пов'язаний не тільки з гілками прецеденту, але і з класами, що забезпечують майбутнє функціонування системи.

2.2 Математична модель представлення прецеденту

На рисунку 2.1 представлений спрощений фрагмент діаграми діяльності по визначенню прецедентів.

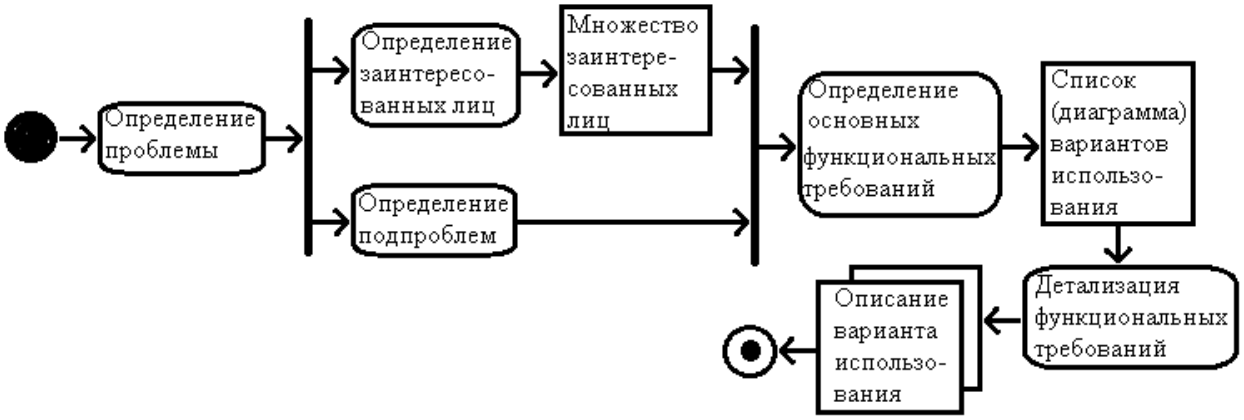

Рисунок 2.1 – Основні етапи визначення вимог

Визначимо зацікавлених осіб у вигляді множини:

$$mA = \{mAp, mAs, mAo\}, \quad (2.1)$$

де mAp – множина основних виконавців (primary actor);

mAs – множина допоміжних виконавців (supporting actor);

mAo – множина закулісних акторів (offstage actor).

Визначимо множину прецедентів:

$$mU = \{\langle uName_i, ma\rangle\} i = 1, k, \quad (2.2)$$

де $uName$ – найменування прецеденту;

ma – множина зацікавлених в прецеденті осіб ($ma \in mA$).

Опис кожного прецеденту представимо у вигляді кортежу:

$$u = \langle uName, pr, sc\rangle, \quad (2.3)$$

де pr – преамбула;

sc – сценарій прецеденту.

Преамбула представлена кортежем:

$$pr = \langle ap, r, ma, da, pc, gm, gs\rangle, \quad (2.4)$$

де ap – головна дійова особа ($ap \in Ap \cap ap \in am$);

r – область дії прецеденту;

da – інтереси учасників;

pc – передумови виконання прецеденту;

gm – мінімальні гарантії;

gs – гарантії успіху.

Часто в преамбулу включають тригер (подія, що приводить до початку виконання прецеденту). Проте в даній технології тригер міститься в першому пункті основного успішного сценарію.

2.3 Модель концептуального класу

Зазвичай використання прецедентів передбачає об'єктно-орієнтовану технологію створення програмного продукту. Наступним етапом після опису сценаріїв є побудова моделі концептуальних класів, який до теперішнього часу виконується вручну. Далі описано створену модель концептуальних класів на основі аналізу пунктів сценаріїв прецедентів.

Виконання всіх розглянутих пунктів сценаріїв повинні в проектованій системі (ПС) забезпечуватися роботою об'єктів класів. Серед запропонованих пунктів сценаріїв є пункт «Створити», який передбачає створення в ПС об'єкта класу. Відповідно до цього пункту ми будемо створювати відповідний клас. Однак при описі прецеденту часто мається на увазі використання об'єктів класів, які не створюються в рамках розглянутого прецеденту. Місцем їх створення може бути інший прецедент, який до моменту написання даного ще не розглядалося. При таких обставинах запропоновано використовувати в подальшому два поняття – власне Клас (виконувався пункт «Створити») і Прототип класу (об'єкт використовувався, але клас ще не створювався).

Введемо поняття множини класів (прототипів) $Mc = \{c\}$.

Кожен клас (прототип) представимо кортежем

$$c = \langle tc, cName, \tau, uName, nP, mData, mFunc \rangle \quad (2.5)$$

де $tc =$ "class"|"$prototype$";

$cName$ – назва класу;

$uName, nP$ – назва прецеденту і номер пункту, де був створений клас;

$\tau =$ "u"|"s" – час життя об’єктів класу (у ході виконання прецеденту – u, або у ході роботи системи – s).

$mData$ – множина даних, які містить клас;

$mFunc$ – множина функцій (методів), які містить клас.

На рівні проектування класів, що розглядається, нас не цікавлять конкретні типи даних, крім цього, виходячи з принципів атомарності представлення вимог, в подальшому ми будемо розглядати дані представлені одним значенням, або безліччю значень одного типу (масив, список). В цьому випадку структуровані дані завжди можна уявити декількома поодинокими даними, або декількома масивами. На відміну від програмних класів введемо поняття розрахованого даного, значення якого не зберігається в об'єкті класу, але може бути отримано шляхом виклику методу класу.

Введемо визначення даного

$$data = \langle dName, td, gH, gL, ref \rangle, \quad (2.6)$$

де $dName$ – назва даного;

$td = "si"|"ar"|"sc"|"ac"$ – тип даного (одиночне значення, масив, одиночне значення, що розраховується, масив, що розраховується);

gH, gL – верхня і нижня границя значень відповідно;

$ref = \{\langle fName, uName, nP \rangle\}$ – посилання на функції (методи), що використовують дане, де $fName$ – назва метода, $uName, nP$ – визначає прецедент і пункт, у якому відбулася зміна.

Таким образом, ref забезпечує трасування вимог, що представлені прецедентом, у програмну функцію (метод класу).

Введемо визначення метода класу

$$func = \langle fName, oData, mArgs, mIData, mCData, mRfFunc \rangle, \quad (2.7)$$

де $oData$ – значення, що повертається методом; якщо $oData = "0"$, то метод нічого не повертає; $oData = "b"$, то метод повертає логічне значення, яке можна не зберігати у об’єкті після завершення роботи метода. У інших випадках ім’я повинно бути включено у множену $mData$ з типом $td = "sc"|"ac"$ і посиланням на даний метод;

$mArgs$ – множина аргументів метода;

$mIData$ – множина даних класу, що приймають нові значення;

$mCData$ – множина даних класу, що використовуються у розрахунках даного метода;

$mRfFunc$ – множина посилань на зовнішні функції (методи інших класів), що використовуються у даному методі. Кожний елемент множени $mRfFunc$ представляється кортежем $mRfFunc_i = \langle cName_j, func_i \rangle, i \geq j,$ де $cName_j$ – клас, якому належить зовнішня функція (у загальному випадку декілька зовнішніх функцій можуть належати одному класу).

2.4 Апробація

На рисунку 2.2 наведена узагальнена схема алгоритму для коригування структури класів при введенні нового пункту в сценарій ВВ. Введення ідентифікатора класу і номера пункту в сценарії дозволяє визначити причину змін в класах, що необхідно при трасуванні і внесення змін до вимоги до програмного продукту.

Перш за все вирішується питання про клас, який може забезпечити виконання відповідного пункту сценарію. Можливо, що опис інших ВВ, які були виконані раніше, привели до необхідності створення класу або прототипу класу, який буде використаний і в даному ВВ. Тому спочатку необхідно проаналізувати наявні класи або прототипи класів. Відповідний прототип класу може бути переведений в категорію класу. Ситуація, коли поточний пункт сценарію повинен створювати клас, але знайдений раніше створений клас, оцінюється як помилкова. Якщо відповідний клас або прототип не знайдено, то створюється новий клас. Після вибору класу визначається наявність в ньому необхідних даних. Дані та методи в обраному класі можуть з'явитися також у процесі опису інших ВВ. При відсутності даних вони створюються. Якщо потрібний метод в обраному класі не знайдений, то він створюється. Якщо створений метод повинен використовувати метод з іншого класу, то всі раніше описані дії щодо вибору класу, даних і методів повторюються.

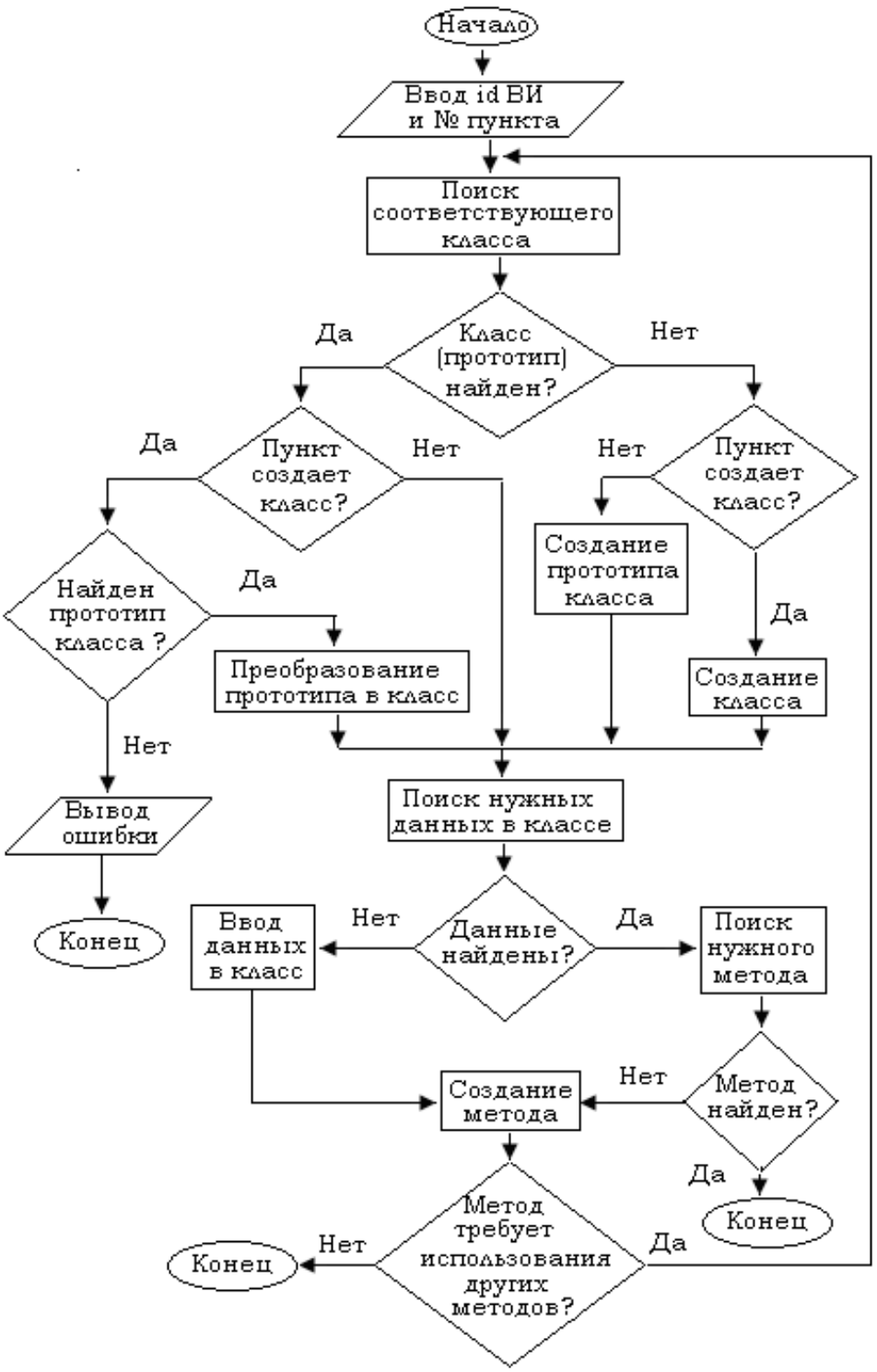

Рисунок 2.2 – Узагальнений алгоритм коригування концептуальних класів

Прийняті рішення реалізовані в програмному продукті UseCaseEditor. На рисунку 2.3 зображено скриншот вікна програми.

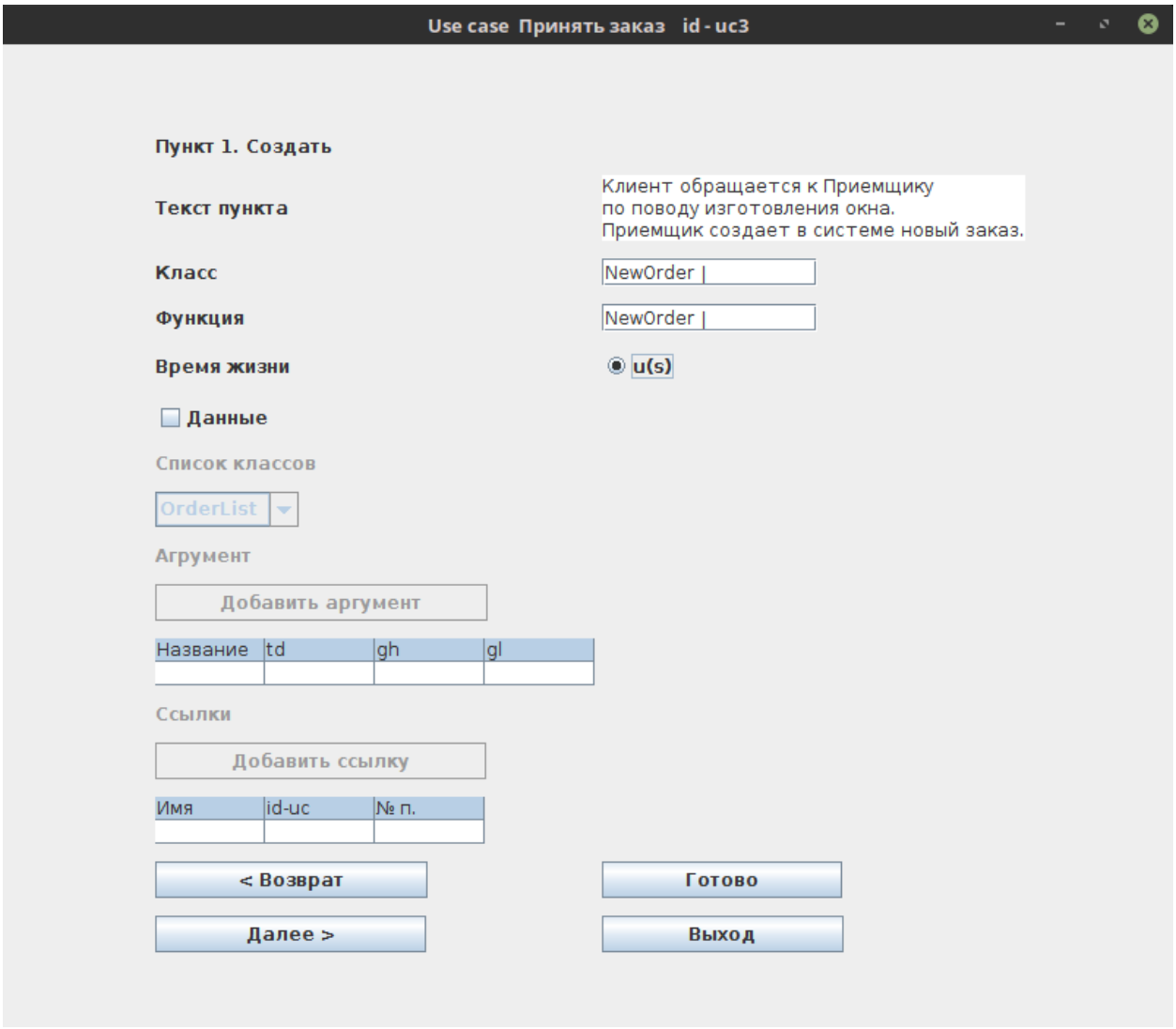

Рисунок 2.3 – Формування класу у відповідності з пунктом «Створити»

ВИСНОВКИ

У ході роботи успішно виконані наступні задачі:

4. Розроблено математичну модель для автоматизованого конструювання варіантів використання, їх сценаріїв, концептуальних класів;
5. Розроблено алгоритми функціонування елементів системи та взаємодії між ними;
6. Програмно реалізовано математичні моделі та алгоритми, що розроблювалися.

Також нами було проведено тестування розроблюваної системи, результати якого наведені у таблиці 3.1. Дві групи студентів використовуючи різні предметні області зі своїх курсових робіт складали сценарії варіантів використання для формалізації функціональних вимог.

Таблиця 3.1 – Результати експерименту

№	Без використання ПП		З використанням ПП	
	Час	Помилки	Час	Помилки
1	30	3	19	0
2	36	5	16	4
3	45	8	28	3
4	22	2	15	2
Середнє значення	33.25	4.5	15.6	2

Ситуації, коли б був потрібен «Пункт, що вільно конструюється» не виникло, що побічно вказує на повноту запропонованої класифікації пунктів сценаріїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alf. Прецеденти у специфікації програм [Електронний ресурс]. – Режим доступу: http://club.shelek.ru/viewart.php?id=232
2. Створення проекту. Аналіз прецедентів. Реалізація прецедентів. Уточнений опис прецеденту [Електронний ресурс]. – Режим доступу: http://vunivere.ru/work72704
3. Леффінгуелл, Д. Уідриг. Принципи роботи з вимогами. Уніфікований підхід. М.: Видавничий дім «Вільямс», 2002. – 450с.
4. Алістер Коберн. Сучасні методи опису функціональних вимог до систем. М.: Лорі, 2002 – 266 с.
5. І.В. Леонов. Фірма "ЭСКЕЙП". Прецеденти (use cases) і їх зв’язки. Виконувачі (actors) і їх зв’язки. [Електронний ресурс]. Дата додавання: 2015-07-09. Режим доступу: http://www.interface.ru/fset.asp?Url=/rational/vmr2.htm
6. Крег Ларман – Використання UML 2.0 і шаблонів проектування. Видавництво Вільямс, 2008 г., 736 с.
7. Rational University – матеріали академічної програми корпорації IBM (див. http://www.ibm.com/ru/software/info/students/): Essentials of visual modeling, Fundamentals of Rational Rose.
8. Лененков А.В. Об’єктно-орієнтований аналіз і проекування з використанням UML IBM та Rational Rose. –ИНТУТЕ.ру, Бином. 2006. – 320 с.
9. Граді Буч, Джеймс Рамбо, Івар Джекобсон – Мова UML. Керівництво користувача. Видавництво ДМК Пресс, 2007 г., 496 с
10. Дж. Рамбо, М. Блаха – UML 2.О. Об’єктно-орієнтоване моделювання і разробка. 2-е вид. — СПб.: Питер, 2007. — 544 с.
11. Ю.Н. Возовіков, Інформаційна технологія автоматизованого створення варіантів використання/ Ю.Н. Возовиков, А.Б. Кунгурцев,

Н.А. Новикова// Наукові праці Донецького національного технічного університету. – Покровськ, 2017. - №1(30). – С. 46-59

12. Кунгурцев А. Б., Поточняк Я. В., Силяев Д. А. Метод автоматизованого створення словника предметної області / А.Б. Кунгурцев, Я.В. Поточняк, Д.Ф. Силяев // Технологічний аудит і резерви виробництва — № 2/2(22), 2015. – С58-63

АНОТАЦІЯ
Наукова робота під шифром «Ice Nine»

Представлена наукова робота присвячена створенню математичної моделі для автоматизованого конструювання варіантів використання, їх сценаріїв, концептуальних класів на основі діаграми варіантів використання та її програмній реалізації.

Варіанти використання часто використовуються в проектуванні інформаційних систем як найбільш точний і повний спосіб формулювання функціональних вимог. Опис варіанту використання – трудомісткий і відповідальний етап роботи з формулювання вимог, від якості виконання яких значною мірою залежить успішність проекту. Не дивлячись на досить велику кількість публікацій, що в тій чи іншій мірі визначають правила складання і використання варіанта використання, до теперішнього часу не існувало класифікації можливих пунктів їх сценаріїв. Це призводило до того, що в проектуванні програмних продуктів варіант використання розглядався як заздалегідь описаний елемент діаграм і питанням автоматизації його складання не приділялася увага.

Метою представленої наукової роботи є підвищення продуктивності створення програмного забезпечення, зменшення часу виконання проекту, зменшення кількості помилок при написанні сценаріїв варіантів використання, усунення дублювання робіт по відношенню до концептуальних класів.

Завдання, які необхідно вирішити для досягнення мети наукової роботи:

7. Розробка математичної моделі для автоматизованого конструювання варіантів використання, їх сценаріїв, концептуальних класів;
8. Розробка алгоритмів функціонування елементів системи та взаємодії між ними;
9. Програмна реалізація математичної моделі та алгоритмів, що розроблювалися.

У науковій роботі розглянуто основні етапи формалізації функціональних вимог, завдяки діаграмі варіантів використання, сценаріїв варіантів використання та формування концептуальних класів, визначено математичні моделі та алгоритми функціонування елементів системи, що проектується, проведено експерименти з використанням системи.

Наукова робота складається з вступу, двох розділів, висновків, списку використаних джерел. Загальний обсяг наукової роботи – 27 сторінок.

Структура наукової роботи:

- пояснювальна записка: 27 сторінок;
- рисунків: 6;
- таблиць: 2;
- використаних наукових джерел: 12.

Ключові слова: варіанти використання; сценарії; моделі; прецеденти; концептуальні класи.