**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ**

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

СТАНДАРТИЗАЦІЯ ТА СЕРТИФІКАЦІЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ

**Лабораторний практикум
для студентів спеціальності 7/8.05010101
“Інформаційні управляючі системи
та технології (за галузями)”**

Київ 2012

МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

СТАНДАРТИЗАЦІЯ ТА СЕРТИФІКАЦІЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ

Лабораторний практикум
для студентів спеціальності 7/8.05010101
“Інформаційні управляючі системи
та технології (за галузями)”

Київ 2012

УДК 004.057.2 (076.5)
ББК З 973.20-018.2Ц.Я7
С 764

Укладачі: *І.Е. Райчев, О.Г. Харченко*

Рецензенти: д.т.н., пров. наук. спів. *Алішов Н.І.,*
д.т.н., проф. *Воронін А.М.,*
директор *Мішарін І.В.*

Затверджено методично-редакційною радою Національного авіаційного університету (протокол №5/12 від 14.06.2012).

С 764 **Стандартизація та сертифікація інформаційних управляючих систем :** Лабораторний практикум для студентів спеціальності 7/8.05010101 “Інформаційні управляючі системи та технології (за галузями)” / уклад.: І.Е.Райчев, О.Г.Харченко. – К.: Видав. Нац. авіац. ун-ту “НАУ-друк”, 2012. – 48 с.

Містить описання принципів формування вимог до проектованих програмних систем з подальшою побудовою моделей якості, що відображають вимоги та відповідають рекомендаціям міжнародних та національних стандартів з якості.

Розглянуто основні поняття та методи визначення якості відповідно до змісту ядра професійних знань інженерії якості програмних систем. Викладено технології визначення якості інформаційних систем.

ВСТУП

Лабораторні роботи виконуються згідно з робочою навчальною програмою дисципліни “Стандартизація та сертифікація інформаційних управляючих систем”, яка призначена для спеціалістів та магістрів спеціальності 7/8.05010101 “Інформаційні управляючі системи та технології (за галузями)”. Мета виконання робіт – набуття студентами практичних навичок та закріплення теоретичних знань з питань технологій забезпечення якості програмних систем (ПС), які є ядром кожної інформаційної системи, що реалізується впровадженням вимог національних та міжнародних стандартів в процеси розробки, атестації та сертифікації ПС. В сучасних умовах особливо важливою є об'єктивна оцінка якості як кінцевого програмного продукту (ПП), так і його оцінка на кожному етапі життєвого циклу (ЖЦ).

Вивчення матеріалу дисципліни побудовано відповідно до вимог кредитно-модульної системи оцінювання знань. Передбачено виконання трьох лабораторних робіт у першому модулі й двох – у другому. Виконуючи лабораторні роботи для першого модулю студенти отримують знання щодо методів формування вимог до властивостей ПС, складу класів вимог до оцінюваних ПС та методів формалізованого представлення вимог. Крім того, лабораторні роботи першого модулю орієнтовані на вивчення студентами структури та процедур побудови узагальненої та частинних моделей якості ПС (зовнішньої, внутрішньої та експлуатаційної), які відповідають сформованим вимогам до ПС.

У процесі вивчення другого модуля студенти виконують лабораторні роботи, які орієнтовані на знайомство студентів із методами та засобами визначення досягнутих значень показників якості відповідно до побудованої моделі якості ПС. Студенти вивчають і застосовують методи оцінки атрибутів характеристик якості, користуючись відповідними метриками, що запропоновані в міжнародних стандартах з якості ПС.

Загальною метою проведення лабораторних робіт є поглиблення та закріплення знань з розділів дисципліни “Стандартизація та сертифікація інформаційних управляючих систем”. У процесі виконання лабораторних робіт 1.1–2.2 студенти докладно знайомляться з методами та засобами оцінювання якості ПЗ, що входить до складу інформаційних управляючих систем.

На виконання кожної роботи відводиться від 8 до 14 академічних годин згідно програми. За цей час студент повинен:

– одержати у викладача індивідуальний номер варіанта;
– виконати передбачене у номері варіанта завдання;
– розробити відповідні діаграми та моделі;
– побудувати власний проект оцінювання якості ПС;
– зробити висновки щодо лабораторної роботи;
– підготувати протокол звіту для даної лабораторної роботи;
– відповісти на контрольні питання.

Звіт про виконання лабораторної роботи має містити:

1. Титульний аркуш.
2. Мету роботи та стислі теоретичні відомості.
3. Порядок виконання лабораторної роботи та висновки.

До оформлення звіту висуваються такі вимоги:

1. Робота оформлюється на аркушах формату А4.
2. На титульному аркуші мають бути вказані:

– тема лабораторної роботи,
– назва дисципліни та номер варіанта предметної області;
– ким виконано роботу (ПІБ, номер групи, факультет);
– ким прийнята робота (ПІБ).

3. У стислих теоретичних відомостях викладаються основні положення даної теми.

4. Хід роботи повинен містити:

– вихідні дані та основні етапи їх перетворення;
– побудовані візуальні діаграми вимог до ПС за допомогою CASE-засобу Visio та мови UML;
– побудовані візуальні діаграми якості ПС за допомогою CASE-засобу Visio та відповідні описи моделей якості;
– висновки за результатами виконаної роботи.

МОДУЛЬ 1. ІНЖЕНЕРІЯ ВИМОГ ТА ДОСЛІДЖЕННЯ МОДЕЛЕЙ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

Модуль вміщує такі теми: “Життєвий цикл і процеси розробки ПЗ”, “Об’єктно-орієнтована інженерія вимог. Проектування та реалізація ПС”, “Якість ПС. Метрики та міри якості. Модель якості ПС згідно стандарту ISO/IEC 9126”,лабораторні роботи 1.1, 1.2, 1.3.

Лабораторна робота 1.1

МЕТОД ІНЖЕНЕРІЇ ВИМОГ С.ШЛЕЙЄР І С.МЕЛЛОРА

1. Мета роботи

Метою цієї роботи є дослідження методології опису вимог до інформаційної системи (ІС) із використанням підхода Шлейєр і Меллора, що використовується для формалізації вимог. Студенти вчаться застосовувати відповідні діаграми для опису вимог до ІС, яка буде функціонувати у визначеному предметному середовищі.

2. Теоретичні відомості

Інформаційні система та її ПЗ згідно цього методу створюється як сукупність певної множини доменів проблемних областей, кожний з яких являє собою окремий світ зі своїми об'єктами. При цьому кожний такий домен аналізується незалежно від інших.

Продуктом аналізу домену є три моделі [1]:

- інформаційна модель системи (онтологія домену);
- модель станів об'єктів, визначеної у складі інформаційної моделі (або онтології);
- модель процесів, що забезпечують переходи із одного стану об'єкта в інший.

2.1. Інформаційна модель домену

На першому етапі треба визначити існуючі об'єкти й виявити зв'язки між ними. Знайдемо об'єкти і дамо їм унікальні імена [1].

Категорії об'єктів, серед яких має сенс проводити пошук:

- реальні предмети світу (стілець, Дніпро і т.п.);
- абстракції фізичних предметів (будинок, літак);
- ролі предметів (для університету – ректор, студент, декан);
- інциденти – події, які впливають на зміну стану об'єктів (повінь, вибори, дефекти тощо);
- взаємодії – це відношення між об'єктами (контракт);
- специфікації – це подання правил, стандартів, критеріїв якості й обмежень на використання системи (правила, стандарти).

Атрибути об'єктів визначаються для кожного об'єкта. Вони являють собою характерні його властивості. Для кожного з атрибутів задаються можливі значення (числовий діапазон, список значень, правила генерації).

Після цього визначається ідентифікатор об’єкта (один або більше атрибутів, які точно вирізняють об’єкт серед інших).

Представлення інформаційної моделі в цьому методі базується на відомій реляційній моделі даних. Об’єкти нумеруються і зображуються у прямокутнику, всередині якого подається номер об’єкта, ім’я об’єкта, а також імена його атрибутів [1].

Об’єкти різних класів можуть брати участь у попарних зв’язках з іншими об’єктами. Розрізняють три види зв’язків:

1. 1:1 ⟷
2. 1:n ⟷▸
3. m:n ◂⟷▸

Над стрілкою може вказуватися ім’я (ідентифікатор) зв’язку, а на кінцях зв’язку можуть вказуватися також ролі об’єктів.

Інформаційна модель метода Шлеєр і Меллора базується на відомій моделі Чена (E-R діаграма), що відтворює рис.1.1.

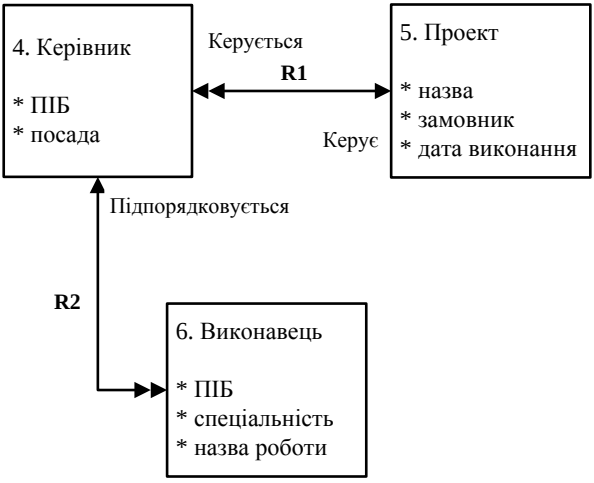


Рис.1.1. Приклад інформаційної моделі по методу Шлеєр і Меллора

Тут R1, R2 – ідентифікатори зв’язків. Крім інформаційної моделі як правило будуються відношення успадкування за допомогою відповідних діаграм (рис.1.2).

Рис.1.2. Приклад відношення спадкування

2.2. Модель станів

Дана модель призначена для відображення динаміки змін, що відбуваються в стані кожного з екземплярів класу. Нагадаємо базові поняття моделі динаміки поведінки об'єктів:

- стан об'єкта визначається поточними значеннями його окремих атрибутів;
- стан об'єкта змінюється в результаті подій, що відбулися, або появи певних стимулів;
- стан домену визначається сукупністю станів його об'єктів;
- зміна стану об'єкта супроводжується деякими процесами, які наперед визначені для кожного стану.

Для фіксації динамічних аспектів вимог є дві нотації: діаграма переходів у стани (ДПС) і таблиця переходів у стани (ТПС) [1].

Обидві нотації базуються на автоматі Мура. Для відображення поведінки об'єктів, у вимогах на розробку ПС потрібно:

1) визначити множину станів;

2) визначити множину інцидентів або подій, які спонукують екземпляри класів змінювати свій стан (переходити у новий стан);

3) визначити правила переходу для кожного із зафіксованих станів, що вказують у який стан переходить екземпляр класу;

4) визначити процеси, що виконуються при переході.

Графічна нотація ДПС передбачає наступне:

- кожний стан об'єкта має назву й порядковий номер;
- кожній події надається унікальна мітка й назва;
- стан позначається рамкою з номером та назвою стану;
- перехід позначається лінією зі стрілкою;

- використовуються спеціальні системні об'єкти – ***таймери*** як механізми виміру інтервалів часу (атрибути таймеру: унікальний ідентифікатор; інтервал часу, через який подається сигнал про настання певної події; мітка події, яка наступить, коли залишок часу буде дорівнювати 0).

Розглянемо ДПС та ТПС пральної машини (рис.1.3 та рис.1.4).

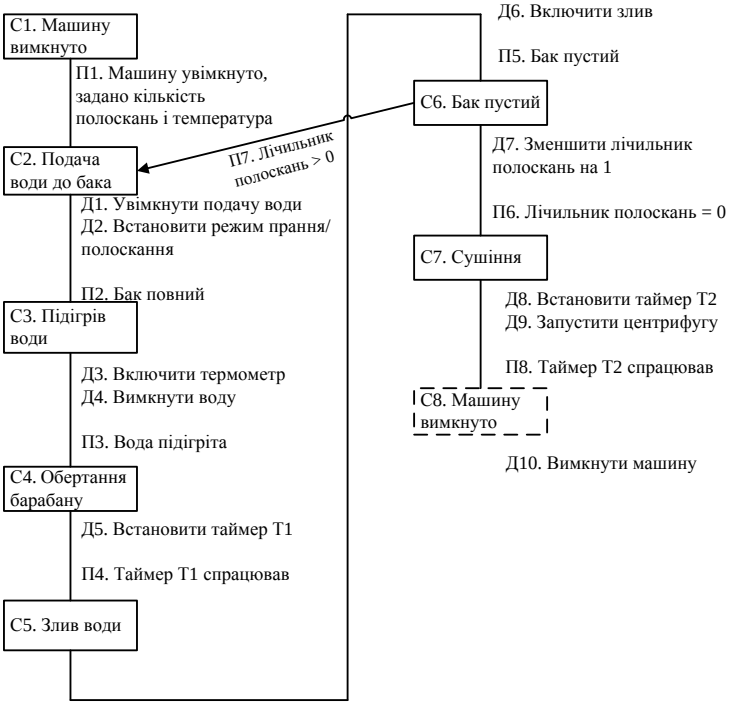


Рис.1.3. ДПС автоматичної пральної машини

	П1	П2	П3	П4	П5	П6	П7	П8
С1	2							
С2		3						
С3			4					
С4				5				
С5					6			
С6						7	2	
С7								1(8)

Рис.1.4. ТПС автоматичної пральної машини

ТПС складається з рядків і стовпців, причому кожний з можливих станів об’єкта представляється рядком, а кожна з можливих подій – стовпцем. Клітина ТПС визначає стан у який переходить об’єкт. Якщо порівняти ДПС і ТПС, то перевагою ДПС є наочність і визначення дій, однак ТПС дозволяє зафіксувати всі можливі комбінації стан-подія. За допомогою побудови ТПС забезпечують повноту та несуперечність представлення вимог.

Все, що подано вище, відноситься до окремих об’єктів, як базових складових архітектури ПС. Для системи в цілому будуються схеми взаємодіючих моделей поведінки об’єктів. Взагалі ПС розглядається як взаємодія об’єктів. Така взаємодія моделюється як обмін повідомленнями між об’єктами системи і зовнішніми об’єктами. Оскільки поведінка об’єктів представляється відповідними ДПС, то поведінка системи в цілому представляється як схема взаємодіючих ДПС.

2.3. Модель процесів

Дії є алгоритмами, які виконуються системою, як реакція на зовнішні події. Набір таких дій визначає поведінку системи, її функціональність. Розуміння вимог до системи має на увазі розуміння цих дій, які можуть бути досить складними [2].

Для подолання складності розуміння дій, виконують їх декомпозицію на окремі складові, які називають процесами. Послідовність процесів визначає потік керування. При цьому процеси обмінюються даними, які утворюють потік даних. Для представлення моделі алгоритмів дій системи використовуються діаграми потоків даних дій (ДПДД) [1, 2].

Як джерела даних допускаються:

- атрибути об'єктів (зберігаються у файлах або базах даних);
- системні годинники й таймери;
- дані подій і повідомлення.

Правила побудови ДПДД

Кожному стану з ДПС може відповідати тільки одна ДПДД. Процес на ДПДД зображується у вигляді овалу (усередині дається назва), потоки даних зображуються стрілками (на котрих вказуються ідентифікатори даних, напрямок до овалу – вхідні, а від

овалу – вихідні дані), джерела даних зображуються прямокутниками або рамками з відкритими сторонами (рис.1.5).

Потоки даних від таймера маркуються номером таймера.

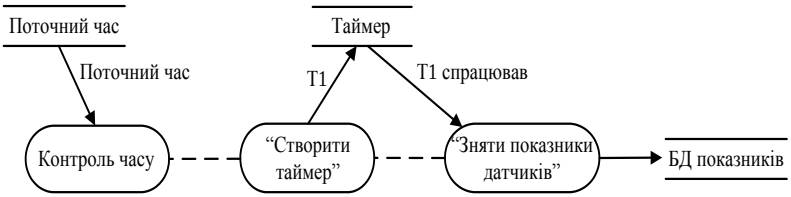


Рис.1.5. Приклад діаграми потоків даних дій (ДПДД) для стану “Запис показників датчиків”

Після побудови ДПДД варто побудувати загальну таблицю процесів із наступними колонками:

- ідентифікатор процесу;
- тип процесу (аксесор – доступ до архівів даних, генератор подій, процес-обчислення, перевірка умов);
- повна назва процесу;
- назва стану, у якому визначений процес та назва дії стану;

За допомогою цієї таблиці перевіряється:

1. несуперечність назв та ідентифікаторів процесів;
2. повнота подій та процесів;
3. виявляються спільні процеси для різних дій чи станів.

2.4. Продукти інженерії вимог за методом Шлеєр і Меллора

Відповідно до методу, результатами аналізу вимог до створюваних ПС є наступні продукти [1]:

1. Інформаційна модель системи (онтологія) у формі:
   - діаграми сутність-зв’язок;
   - опис об’єктів з атрибутами та зв’язків між об’єктами.
2. Модель поведінки об’єктів системи у формі:
   - ДПС і ТПС;
   - описи дій для ДПС та описи подій для ДПС та ТПС.
3. Модель процесів для станів об’єктів у вигляді:
   - ДПДД та таблиці процесів станів;
   - описи процесів (природною мовою).

3. Порядок виконання роботи

В процесі виконання роботи студенти з метою формалізації вимог до ПС в межах заданої предметної галузі мають побудувати 4 типи діаграм і одну таблицю ТПС, виконавши наступні кроки:

1) Шляхом аналізу предметної області слід виділити суттєві для системи об'єкти (5-6 об'єктів) та побудувати інформаційну модель системи. При цьому можна використати досвід побудови E-R діаграм для заданої предметної області.

2) Віднайти для складних об'єктів відношення спадкування і побудувати для них відповідні діаграми спадкування властивостей.

3) Розглянути модель поведінки для довільного об'єкта системи. Об'єкт має мати не менше 6-7 незалежних станів. Для об'єкта треба побудувати діаграму переходів у стани (ДПС) і відповідну парну до цієї діаграми таблицю переходів у стани (ТПС). При цьому користаємося діаграмами активності мови UML.

4) Для довільного стану об'єкта із побудованої ДПС побудувати модель процесів у вигляді ДПДД (5-6 процесів/дій та 4-5 джерел даних). Для побудови цих діаграм можна скористатися діаграмами станів та діаграмами потоків даних. Для ДПДД побудувати загальну таблицю процесів із наступними колонками: ідентифікатор процесу; тип процесу (аксесор, генератор подій, процес-обчислення, перевірка умов); повна назва процесу; назва стану, у якому визначений процес; назва дії стану.

4. Завдання

Для заданого опису предметної області визначити вимоги до проектованої системи відповідно до методу Шлеєр і Меллора:

1. Діяльність підприємства "Будинок музики"

Підприємство "Будинок музики" займається продажами і ремонтом музичних інструментів (електрогітар, клавішних та ударних інструментів) та апаратури. Крім того підприємство проводить заходи, пов'язані з музичною діяльністю (концерти, джем-сейшени, майстер-класи). Напрямками діяльності є такі: виконання продажів музичних інструментів, ремонт інструментів і обладнання відповідно до замовлень клієнтів, організація музичних заходів, навчання в групах та індивідуально. Покупець оформляє замовлення інструментів у продавця або через Інтернет.

2. Діяльність Інтернет-провайдера

Інтернет-провайдер має автоматизувати обслуговування клієнтів. Основними функціями організації є підключення нових абонентів до мережі Інтернет, здійснення контролю за оплатою ними послуг, оброблення позаштатних ситуацій (включаючи налагодження з'єднання з боку клієнта) та виконання аналізу якості сервісу. Організація поділяється на чотири відділи: абонентський, фінансовий, експлуатації та будівництва мережі. Кожен з відділів для виконання функцій використовує менші структурні одиниці.

3. Діяльність ріелтерської агенції

В агенції, що купує й продає нерухомість, із клієнтами працюють брокери. Агенція має назву, ліцензію, адресу, телефон, директора. На операцію з нерухомістю брокер оформляє накладну, в якій зазначено номер, дату, тип операції (купівля, продаж, оренда), ПІБ брокера, тип застави, атрибути клієнта, адреса об'єкта нерухомості. Нерухомість характеризується державним реєстраційним номером, адресою, вартістю, приналежністю. Клієнтом може бути як фізична, так і юридична особа.

4. Діяльність видавництва

У видавництві, що видає печатну продукцію, клієнти роблять замовлення. Співробітники видавництва характеризуються посадою, фахом і працюють у різних відділах, котрі мають назву та список технічних засобів для друку. Перш ніж оформити замовлення, замовники вивчають прайси, в яких перераховані види й найменування продукції (книжки, журнали, газети, листівки, конверти, плакати), із зазначенням обсягів та термінів робіт, а також вартістю одиниці продукції. Замовлення має список печатної продукції та банківські рахунки виконавця та замовника.

Контрольні питання

1. Характеризувати етап формування вимог до ПС.
2. Дати визначення специфікації та аналізу вимог до ПС. Функціональні та нефункціональні вимоги. Привести приклади.
3. Як будується модель поведінки об'єктів системи за допомогою ТПС і ДПС та модель процесів стану об'єкта ДПДД?
4. Зробити висновки щодо достатності або недостатності продуктів інженерії вимог за методом Шлеєр і Меллора для подальшого проектування та реалізації ПС.

Література: [1, 2].

Лабораторна робота 1.2
МЕТОД ІНЖЕНЕРІЇ ВИМОГ І.ДЖЕКОБСОНА

1. Мета роботи

Метою цієї роботи є дослідження методології опису вимог до програмної системи (ПС) із використанням підхода Джекобсона, який використовується не тільки для формалізації вимог до ПС, а і для визначення складу об'єктів, якими буде оперувати ПС. Студенти вчаться застосовувати діаграми Джекобсона для опису вимог до ПС, що функціонує у визначеній предметній галузі.

2. Теоретичні відомості

Метод Джекобсона – це єдиний метод, що вказує послідовний достатньо формалізований підхід до виявлення об'єктів, суттєвих для даної предметної області [3].

2.1. Концепція моделі сценаріїв для збору вимог

Метод Джекобсона базується на варіантах використання системи. На першому кроці складна система декомпозується на більш прості складові [1, 3]. Розробка системи починається з осмислення мети системи, тобто для кого й для чого створюється ПС. Складність загальної мети виражається через окремі складові мети. Складові мети можуть відповідати функціональним і нефункціональним вимогам, а також проектним рішенням.

Функціональні вимоги **–** це вимоги до функцій системи.

Нефункціональні вимоги **–** це вимоги, наприклад, до надійності, безпеки, тестованості тощо [2, 4, 5].

Складові мети є джерелом вимог до системи і класифікуються на обов'язкові й бажані. Складові мети можуть перебувати в певних відношеннях (узгодженість, конфлікт, залежність тощо).

Другий крок – визначення носіїв інтересів, яким відповідає кожна складова мети, і можливих сценаріїв. Відбувається послідовна декомпозиція складних проблем:

1. проблеми трансформуються в сукупність складових мети;
2. кожна із складових мети трансформується в сукупність можливих прикладів використання системи;
3. сценарії трансформуються в процесі аналізу в сукупність взаємодіючих об'єктів.

У такий спосіб будується ланцюжок трансформації: проблема – складові мети – сценарії – об’єкти. Проведена трансформація відображається у термінах базових понять проблемної області.

Кожний сценарій запускається користувачем (актором), який є зовнішнім чинником ІС. Кожний сценарій запускає один актор.

Сценарій складається з ряду операцій і має стани й поведінку.

Сценарій – це повний ланцюжок подій, ініційованих актором, і специфікація реакції на них [2, 3, 6].

Сукупність сценаріїв визначає всі можливі шляхи використання системи. Для моделі сценаріїв визначається спеціальна графічна нотація: актор – іконка людини, під якою вказується назва актора; сценарій – овал (усередені овалу – назва сценарію); зв’язки між ними – стрілки. Кожного актора обслуговує своя сукупність сценаріїв (див. рис.2.1).

Рис.2.1. Діаграма сценаріїв бібліотеки

Відношення розширення “extend” вказує на те, що функції одного сценарію є доповненням до функцій іншого. Відношення використання “include” означає, що певний сценарій може бути використаний декількома іншими сценаріями (аналог процедури).

Продуктом першої стадії інженерії вимог – збір вимог по Джекобсону – є модель вимог, що складається із онтології домену, моделі сценаріїв і опису інтерфейсів сценаріїв [1, 2, 4].

Онтологія домену може бути зображена за допомогою нотації моделі сутність-зв'язок (ERD). Модель сценаріїв супроводжується неформальним описом кожного сценарію, що складається з:

- назви, анотації та акторів, які запускають сценарій;
- опису аспектів дій акторів по взаємодії з системою;
- передумови та функції, які виконуються в сценарії;
- нестандартні ситуації й реакція на них;
- умови завершення сценарію і постумови.

Далі сценарій використання трансформується в сценарій поведінки системи шляхом додавання елементів, які пов'язані з конструктивним рішенням цільового продукту. Наприклад, механізми запуску сценарію (пункти меню), механізми вводу даних (ввід пароля), поведінка у випадках виникнення нештатних ситуацій. Опис інтерфейсів дається теж неформально, однак корисно узгоджувати інтерфейси ПС із майбутніми користувачами.

2.2. Модель аналізу вимог

Модель вимог піддається аналізу з метою подальшого структурування проблеми, що вирішує дана ПС. Оскільки обрано об'єктну архітектуру, то результатом структурування є об'єкти, взаємодія яких визначає функціональність ПС. Сукупність об'єктів знаходимо шляхом послідовної декомпозиції сценаріїв на об'єкти.

Слід визначити такий набір об'єктів, який дасть можливість ПС бути легко адаптованою у випадку зміни вимог до неї [1, 6].

Аксіома. *Всяка працююча ПС згодом вимагає змін*.

Тому стратегія вибору об'єктів заснована на таких постулатах:

1. зміна вимог неминуча;
2. об'єкт повинен модифікуватися тільки внаслідок зміни відповідних вимог до ПС;
3. об'єкт повинен бути стійким до модифікації;
4. під стійкістю до модифікацій (стабільність ПС) мається на увазі, що зміни будь-якої з вимог спричиняють собою зміну найменшої кількості об'єктів ПС (в ідеалі - одного).

Вищенаведені принципи приводять до того, що простір, у якому функціонує ІС, потрібно структурувати в ***трьох*** вимірах:

1. інформація, що обробляє система (її внутрішній стан);
2. поведінка системи та її презентація (інтерфейси ПС).

Результати досліджень свідчать, що на протязі ЖЦ найбільша кількість змін відбувається в інтерфейсах. Поведінка системи більш консервативна, але також зазнає змін. Характер і структура інформації, яку обробляє ПС, є найбільш стійкими до змін.

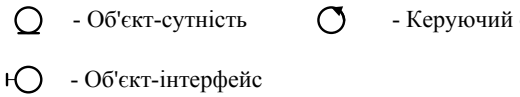
Доцільно для кожного з вимірів мати свою сукупність об’єктів, що його відображає. За допомогою такого поділу ми локалізуємо в тексті вимог найбільш стабільні, мобільні й проміжні елементи.

Об’єкт-сутність моделює довгоживучу інформацію, яка зберігається після виконання сценаріїв. Більшість із цих об’єктів виявляються на етапі аналізу проблемної області, але до уваги беруться тільки ті, на які є посилання в сценарії.

Об’єкти-інтерфейси містять функціональності, які залежать безпосередньо від оточення системи. Ці об’єкти визначаються шляхом аналізу опису інтерфейсу сценарію з моделі вимог. За допомогою інтерфейсів актори взаємодіють із кожним зі сценаріїв системи. Завдання об’єкта – трансляція інформації, яку вводить актор, у події, на які реагує система та зворотня трансляція.

Керуючі об’єкти – це об’єкти, які перетворюють інформацію, введену об’єктами інтерфейсу й представлену об’єктами-сутностями, в інформацію, що виводиться інтерфейсними об’єктами. Керуючі об’єкти відповідають функціям обробки інформації і є перетворювачами.

Модель аналізу вимог має відповідну графічну нотацію:



Діаграма взаємодії об’єктів в рамках сценарію будується на основі аналізу вимог до цього сценарію. Приклад такої діаграми показано для сценарію обробки замовлень в бібліотеці на рис.2.2.

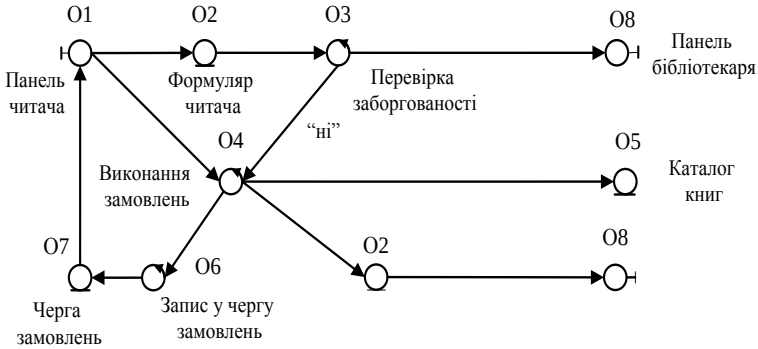


Рис.2.2. Модель аналізу вимог: асоціації між об’єктами бібліотечної системи

Таким чином, послідовний підхід до виявлення об'єктів, суттєвих для даної предметної області, складається з таких кроків:

1. Множина можливих об'єктів визначається на етапі побудови інформаційної моделі проблемної галузі (ERD домену).

2. Склад об'єктів уточнюється після побудови комплекту діаграм сценаріїв для моделювання вимог до проектованої ПС (можуть з'явитися нові об'єкти та скоротитися склад об'єктів з п.1, якщо вони не використовуються в сценаріях).

3. Остаточно склад об'єктів визначається після побудови діаграм аналізу вимог для кожного сценарію (можуть з'явитися нові об'єкти та скоротитися склад об'єктів з п.2, якщо вони не використовуються у відповідних діаграмах взаємодії).

2.3. Продукти інженерії вимог по методу Джекобсона

1. Онтологія домену (для нотації проблемної області можна скористатися діаграмами Чена або Баркера, тобто ERD).
2. Моделі сценаріїв та неформальний опис сценаріїв і акторів.
3. Детальний опис інтерфейсів, сценаріїв і акторів.
4. Діаграми взаємодії об'єктів в рамках сценаріїв (будуються на основі аналізу вимог).

Подальша деталізація вимог відбувається на етапах ЖЦ ПС, при цьому зберігається трасування вимог (відстеження відповідних об'єктів протягом ЖЦ принаймні на всіх етапах розробки ПС).

3. Порядок виконання роботи

У процесі виконання роботи студенти з метою формалізації вимог та визначення складу об'єктів в межах заданої предметної галузі, мають побудувати 2 типи діаграм.

1) На першому кроці шляхом аналізу предметної області слід виділити суттєві для системи об'єкти (5-6 об'єктів) та побудувати інформаційну модель системи. Для цього скористаємося результатами лабораторної роботи 1.1. Для побудови інформаційної моделі використовується, також, діаграма класів.

2) На другому кроці необхідно побудувати повну діаграму сценаріїв для ПС, що створюється. В діаграмі повинно бути 5-6 базових функціональних сценаріїв та 4-5 сценаріїв типу розширення або використання ("extend", "include"). Для побудови повної розгорнутої діаграми сценаріїв ПС можна скористатися діаграмами варіантів використання (use case diagrams, мова UML).

Після побудови діаграми сценаріїв необхідно скласти неформальний опис всіх сценаріїв і акторів, наявних на діаграмі.

3) На третьому кроці слід побудувати дві діаграми аналізу вимог (діаграми взаємодії об'єктів в рамках сценаріїв) для двох відповідних сценаріїв ПС із діаграми сценаріїв (п.2). На кожній побудованій діаграмі має бути не менше 2-3 об'єктів-інтерфейсів та по 4-5 керуючих об'єктів і об'єктів-сутностей. Для кожного знайденого об'єкта треба виконати детальний опис.

4. Завдання

Для заданого опису предметної області визначити вимоги до проектованої системи відповідно до методу Джекобсона:

1. Робота готелю

Обслуговування клієнтів готелю полягає в забезпеченні клієнтів послугами та перевірці якості цих послуг. Послуги забезпечує персонал готелю, а перевірку здійснюють менеджер та адміністратори. Відповідно до створеного дирекцією бізнес-плану готелю надаються такі послуги: забезпечення їжею та напоями як з доставкою у номери готелю, так і у ресторанах та барах готелю; надання номерів зі зручностями та комплексів для відпочинку; забезпечення безпеки мешкання; надання різного роду інформації для гостей. Щомісячно складається бізнес-звіт діяльності готелю.

2. Діяльність букмекерської контори

Букмекерська контора приймає ставки на результати наступних спортивних змагань. Оператор-букмекер приймає від клієнта прогнози на результати матчів, а ставки клієнт сплачує касиру в касу закладу, отримуючи квітанцію. Після оприлюднення результатів клієнт (у разі виграшу) може отримати суму свого виграшу в касі, надаючи відповідні квітанції. Раз у тиждень готівку із каси закладу інкасатори вивозять у банк. Персоналу контори суворо забороняється обслуговувати неповнолітніх, що перевіряється зовнішніми інспекторами щоденно.

3. Робота житлово-комунальної контори

В кожному місті маються житлово-комунальні контори (ЖКК), котрі здійснюють обслуговування жителів будинків, постачаючи газ, електроенергію та воду. В ЖКК мається декілька відділів (дирекція, бухгалтерія, інспектори по прийому громадян, водії та робітники різних спеціальностей), Жителі будинків сплачують

вартість комунальних послуг і отримують квитанції про сплату. На балансі ЖКК, як правило, знаходиться декілька будинків. На виконання робіт жителі будинку подають заявку, а майстер оформляє накладну, де зазначено: дату, адресу, ПІБ замовника, список матеріалів та виконаних робіт із зазначенням вартості.

Контрольні питання

1. Дати визначення поняттям: актори, ролі, сценарії в методі І.Джекобсона. Привести приклади відношень “розширює” та “використовує” між сценаріями.

2. Як згідно методу Джекобсона будується діаграма взаємодії об’єктів в межах сценарію і визначаються її об’єкти? Перерахувати продукти інженерії вимог за методом Джекобсона.

Література: [1–6].

Лабораторна робота 1.3

ПОБУДОВА МОДЕЛЕЙ ЯКОСТІ ПРОГРАМНОЇ СИСТЕМИ ВІДПОВІДНО ДО СТАНДАРТУ ISO/IEC 9126 (PARTS 1, 2, 4)

1. Мета роботи

Метою цієї роботи є дослідження методів побудови моделей якості ПС відповідно до рекомендацій стандарту ISO/IEC 9126 (частини 1, 2, 4). Студенти вчаться застосовувати характеристики, підхарактеристики, атрибути та метрики якості стандарту ISO/IEC 9126 для побудови моделей зовнішньої та експлуатаційної якості ПС, що функціонує у визначеній предметній галузі.

2. Теоретичні відомості

2.1. Визначення якості ПС

Визначення якості сформульоване в ДСТУ 2844-94 «Програмні засоби. Забезпечення якості. Терміни та визначення»:

Якість **–** це сукупність властивостей ПС, які забезпечують її здатність задовольняти встановлені або передбачувані потреби відповідно до призначення ПС [4, 7, 8].

Для оцінки ПС використовуються в основному два процеси: ***атестація*** та ***сертифікація***. Як правило ***атестація*** проводиться організацією розробником для перевірки того, чи відповідає ПС заданим вимогам. У процесі атестації широко використовується тестування. ***Сертифікація*** проводиться третьою стороною, а саме спеціальним органом, ліцензованим державою для таких дій [9].

Сертифікація **–** це оцінка відповідності властивостей ПС вимогам до неї. *Сертифікаційні випробування* виконують лабораторії, що призначаються органом сертифікації [9].

2.2. Оцінювання якості ПС

Атрибути ПС, які характеризують якість, виміряються за допомогою ***метрик якості.***

В ієрархічній моделі якості ПС, яка складається із множини характеристик, введені такі позначення: C_i – характеристики, S_i– підхарактеристики, A_i – атрибути якості ПС.

Питання побудови моделей якості та оцінювання якості розглянуті у стандартах ISO/IEC 9126 (частини 1, 2, 3 та 4): модель якості – частина 1, зовнішні метрики – частина 2, внутрішні метрики – частина 3, експлуатаційні метрики – частина 4.

2.3. Моделі якості ПС

Модель якості ПС представляє множину взаємозалежних характеристик, які утворюють базис для специфікації вимог до якості і оцінки якості ПС [7]. Як еталон слугує стандарт ISO/IEC9126 (parts 1-4). Зовнішні й внутрішні метрики служать для вимірювання атрибутів моделі. Модель якості показана на рис.3.1.

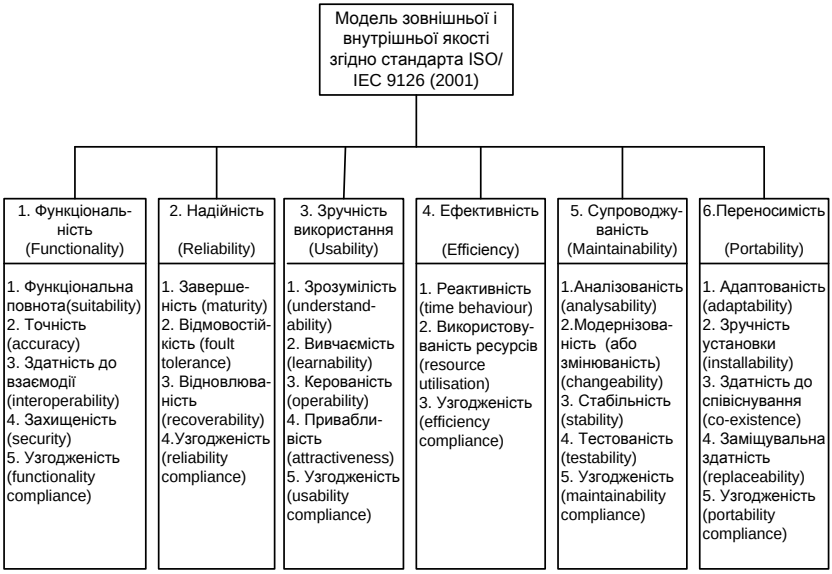


Рис.3.1. Модель якості ПС

Функціональність (functionality) – властивість ПС виконувати функції у відповідності встановленим і очікуваним потребам при використанні у вказаних умовах.

Надійність (reliability) – властивість ПС зберігати рівень функціонування при роботі в зазначених умовах.

Зручність використання (useability) – властивість ПС бути зрозумілою, освоюваною, зручною і привабливою для користувачів, при використанні в зазначених умовах.

Ефективність (efficiency) – властивість ПС забезпечувати раціональне використання виділених ресурсів при роботі у встановлених умовах.

Супроводжуваність (maintainability) – властивість ПС забезпечувати можливість її ефективної модифікації. Модифікація може включати корегування, вдосконалення або адаптацію ПС до змін середовища, вимог або функціональних специфікацій.

Переносимість (portability) – властивість ПС бути перенесеною з одного середовища до іншого [7, 8].

Для кожної підхарактеристики якості існує декілька різних метрик якості.Тому в моделі якості запровадимо атрибути. Кожний атрибут буде вимірюватися за допомогою відповідної метрики. Таким чином,***загальна формула моделі якості*** буде виглядати так:

$$Q = \left\{ C_i \left\{ S_{ij} \left\{ A_{ijk} \left(M_{ijk}, W_{ijk} \right) \right\}_{k=1}^{K_{ij}} \right\}_{j=1}^{J_i} \right\}_{i=1}^{I} \qquad \textbf{(3.1)}$$

У формулі (3.1): Q – множина взаємозалежних характеристик (властивостей) якості, C_i – i-та характеристика якості, S_{ij} – j-та підхарактеристика i-ї характеристики якості, A_{ijk} – k-й атрибут j-ї підхарактеристики i-ї характеристики якості, M_{ijk} – метрика k-го атрибута j-ї підхарактеристики i-ї характеристики якості, W_{ijk} – коефіцієнт ваги (значимості) k-го атрибута j-ї підхарактеристики i-ї характеристики якості; i, j, k – індекси, I – загальна кількість характеристик якості в моделі якості ($2 \le I \le 6$), J_i – загальна кількість підхарактеристик i-ї характеристики якості, K_{ij} – загальна кількість атрибутів j-ї підхарактеристики i-ї характеристики якості.

Після визначення всіх елементів моделі (3.1) можна перейти до випробувань, в ході яких обчислюються фактичні значення атрибутів підхарактеристик якості. Ці значення порівнюються із обмеженнями, заданими у вимогах [10].

Якщо отримані фактичні значення показників якості відповідають вимогам, то подальшу оцінку можна провести, використовуючи інтегральний показник якості. Інтегральний показник можна застосувати для вибору кращого ПЗ із декількох конкуруючих в даній галузі при умові погодження конфліктуючих атрибутів якості. Але, оскільки в моделі є показники з різними метриками, такими, як неперервні числові, бальні, якісні та інші, необхідно попередньо провести узгодження та нормування метрик, що можна виконати, наприклад, шляхом уведення шкал для якісних та категорійних критеріїв і заданням вагових множників.

Модель експлуатаційної якості

На відміну від моделі зовнішньої і внутрішньої якості, для опису експлуатаційної якості служить однорівнева модель, яка розподіляє атрибути якості по чотирьом характеристикам [7].

Характеристики експлуатаційної якості:

- *результативність (effectiveness)* – ступінь у якій користувач досягає заданих цілей по точності й повноті вирішення задач у контексті використання ПС;
- *продуктивність (productivity)* – ступінь у якій витрачаються ресурси на досягнення користувачем заданої ефективності у встановленому контексті використання ПС;
- *безпека (safety)* – рівень ризику завдання матеріальних і моральних збитків, тобто шкоди здоров'ю людей, бізнесу, майну, навколишньому середовищу при використанні ПС у встановленому контексті її використання;
- *задоволеність (satisfaction)* – ступінь у якій користувач задоволений ПС у певному контексті її використання.

3. Порядок виконання роботи

Вважається, що ПС створюється для функціонування в межах заданої предметної області відповідно до вимог, визначених студентами у лабораторних роботах 1.1 та 1.2.

У процесі виконання роботи студенти мають побудувати дві моделі якості для оцінювання рівня якості ПС. Це модель зовнішньої якості ПС і модель якості у використанні (модель експлуатаційної якості ПС). Для цього слід виконати такі кроки:

1). В процесі виконання роботи студенти з метою побудови моделі зовнішньої якості ПС користуються рекомендаціями

стандарту ISO/IEC 9126-1. Необхідно обрати не менше 3-5 характеристик якості і для них загалом не менше 9-10 підхарактеристик. Кожну підхарактеристику будемо оцінювати за допомогою одного атрибута якості.

2) Необхідно провести відображення функціональних та нефукціональних вимог до ПС на підхарактеристики та атрибути якості моделі із рис.3.1. Після цього потрібно обгрунтувати обрані властивості якості, що увійшли у модель, тобто пояснити чому ті, а не інші атрибути цікавлять нас більше. На наступному кроці потрібно знайти попарно конфліктуючі між собою властивості якості (не менше двох атрибутів). Наприклад, здатність до співіснування та захищеність і т.ін. Далі студенти обирають вагові коефіцієнти для атрибутів, підхарактеристик та характеристик якості. Побудована модель якості має відповідати формулі (3.1).

3) Користуючись стандартом ISO/IEC 9126-4, студенти будують експлуатаційну модель якості ПС, що функціонує в заданій предметній галузі. Необхідно обрати не менше чотирьох характеристик якості у використанні та відповідні метрики для обчислення досягнутого рівня експлуатаційної якості. Слід визначити яким чином характеристики та атрибути зовнішньої якості відображаються на характеристики експлуатаційної якості.

4. Завдання

Для заданої предметної галузі побудувати моделі якості ПС:

1. Робота метеорологічної станції

Метеорологічна станція забезпечує автоматичний контроль наступних погодних характеристик: швидкість та напрям вітру, температура повітря, атмосферний тиск, вологість повітря. Інформаційна система станції дає можливість визначати коефіцієнт різкості погодних умов, зміну температури та тиску. Станція має приміщення для устаткування та установки приладів, обслуговуючий персонал та комп'ютерні засоби. На основі вимірів погодних параметрів складається прогноз погоди, який курьер станції доставляє на радіо та телебачення.

2. Діяльність підприємства по розробці програмних продуктів

Приватні підприємства, що розробляють програмні продукти (ПП), мають декілька відділів, де працюють співробітники (дирекція, бухгалтерія, відділ розробки ПП, відділ тестування ПП,

відділ супроводження та ін.). Програмний продукт поставляється замовнику і має назву, список розробників, вартість, документацію, дистрибутив, правила використання. Замовниками можуть бути як фізичні, так і юридичні особи. Кожний замовник може замовити декілька ПП, на кожний з яких він отримує ліцензію, в якій вказано назву продукту, дату продажу, вартість, терміни апгрейдів.

3. Робота поліклініки

В кожному районі міста мається поліклініка, яка обслуговує пацієнтів. Пацієнти записуються на прийом до лікарів-спеціалістів, а також до медсестер на процедури на певну дату і час у деякі кабінети. Пацієнти характеризуються ПІБ, номером медичної картки, номером медичної страховки, адресою. Лікарі назначають пацієнтам лікування у вигляді ліків та процедур. Ліки мають назву, номер ліцензії МОЗ, ціну, правила вживання, термін придатності. Процедури мають назву, список необхідних ліків, тривалість.

Контрольні питання

1. Визначення якості ПС. Основні стандарти з якості ПС. Моделі якості. Цілі моделювання якості.
2. Визначення характеристик якості. Три типи якості. Дати кожній з них стислу характеристику.

Література: [4, 7–10].

МОДУЛЬ 2. СТАНДАРТИ ТА ТЕХНОЛОГІЇ ОЦІНЮВАННЯ ЯКОСТІ ПРОГРАМНИХ СИСТЕМ

Модуль вміщує такі теми: “Технології оцінювання якості ПС”, “Тестування та супроводження ПП”, лабораторні роботи 2.1 і 2.2.

Лабораторна робота 2.1

ОЦІНЮВАННЯ РІВНЯ ЯКОСТІ ПРОГРАМНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ МЕТРИК СТАНДАРТУ ISO/IEC 9126 (PARTS 2,4)

1. Мета роботи

Метою цієї роботи є дослідження методів оцінки рівня якості ПП відповідно до побудованих моделей якості, що створені згідно рекомендацій стандарту ISO/IEC 9126. Студенти вчаться застосовувати характеристики, підхарактеристики та метрики якості стандартів ISO/IEC 9126 (частини 1, 2 та 4) для побудови моделей зовнішньої та експлуатаційної якості ПС, що функціонує у визначеній предметній галузі. Після цього студенти оцінюють рівень зовнішньої якості ПС, використовуючи зовнішні метрики, а рівень експлуатаційної якості – метрики якості у використанні.

2. Теоретичні відомості

2.1. Оцінювання якості ПС

Атрибути ПС, які характеризують якість, вимірюються за допомогою ***метрик якості.***

Метрика – це комбінація конкретного *методу вимірювання* атрибута сутності й *шкали вимірювання* [4, 7].

Метод вимірювання визначає спосіб одержання значень атрибута якості певної сутності. *Шкала* використовується для структурування отриманих значень. ***Метрика*** визначає *міру* атрибута, тобто змінну, котрій присвоюється значення в результаті вимірювання. Наприклад, сутність – це звіт про виявлення дефектів у ПС, атрибут – список дефектів, метод вимірювання – підрахувати кількість дефектів у списку, шкала – цілі числа більше 0, міра атрибута – загальне число дефектів, ім'я метрики (однойменне мірі) – загальне число дефектів [7].

Міра атрибута є безпосередньою, якщо вона не залежить від мір інших атрибутів, або побічною, якщо утворюється на основі мір інших атрибутів. Питання, пов'язані з побудовою моделей якості докладно розглянуті в лабораторній роботі 1.3.

Оскільки метрика якості ПС – це модель вимірювання атрибута, то метрики служать індикатором одного або декількох атрибутів. Метрика називається ***базовою***, якщо в її основі лежить елементарний метод вимірювання атрибута. Стандарт ISO/IEC 9126-2 рекомендує застосовувати 5 видів шкал виміру значень:

1. *номінальна шкала* (класифікаційна шкала);
2. *порядкова шкала* дозволяє впорядковувати властивості по зростанню/зменшенню шляхом порівняння із базовим значенням;
3. *інтервальна шкала* – відзначає дистанцію між властивостями об'єкта (використовується в арифметичних операціях та операціях порівняння);
4. *відносна шкала* – значення розрізняються відносно обраної одиниці вимірювання(вважається найбільш пріорітетною шкалою);
5. *абсолютна шкала* є спеціальним випадком відносної шкали, де вказується абсолютне значення величини.

Обчисленні значення метрики самі по собі не несуть інформації про рівень задоволення вимог до якості. Для цих цілей шкалу, як правило, ділять на області (ранги), які відповідають

різним ступеням задоволеності вимог. Наприклад, можна розподілити шкалу на 4 категорії, як показано на рис.4.1.

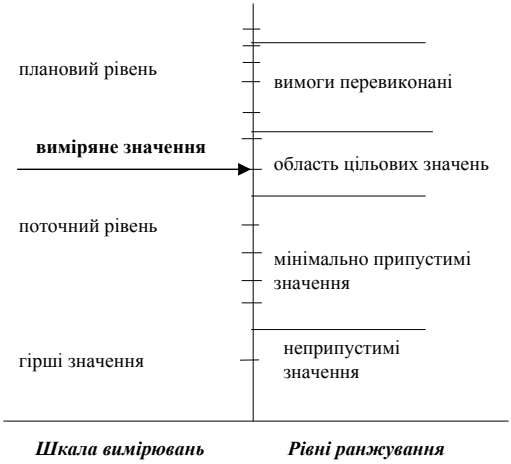


Рис.4.1. Рівні ранжування метрик

Стосовно об'єкта вимірювання міри й метрики підрозділяються на *зовнішні*, *внутрішні* й *метрики використання*.

Зовнішні метрики використовують міри ПП, який працює на комп'ютері. Ці міри отримуються в результаті вимірювання поведінки ПП у ході тестування й функціонування. Відповідні метрики розроблюються та використовуються для демонстрації якості ПП, а також для підтвердження того, що програмний продукт задовольняє зовнішнім вимогам до якості.

Приклади зовнішніх метрик для характеристики надійність – число усунутих дефектів, середній час між відмовами.

Внутрішні метрики дають можливість оцінити якість ПС безпосередньо по її властивостях (об'єм коду, число цикломатичності програми тощо). Внутрішні метрики відображають якість проміжного й кінцевого програмного продукту по тим характеристикам, які визначені в моделі. Ці метрики використовуються для верифікації того, що проміжний та кінцевий ПП задовольняють вимогам до внутрішньої якості. Розробка внутрішніх метрик ґрунтується на виконанні вимірювань статичних атрибутів, які визначаються й оцінюються по тексту

вихідного коду, а також по графічому представленню потоків керування, чи по документації на ПС.

Приклади внутрішніх метрик для характеристики надійність – число помилок знайдених при інспекції коду, число усунутих дефектів при інспекції [7].

Метрики якості у використанні (експлуатаційні метрики). Ці метрики вимірюють ступінь, у якій ПП задовольняє потреби користувачів в ефективності, продуктивному й безпечному вирішенні завдань і оцінюють видимі результати експлуатації ПС.

Приклади – повнота досягнення цілей користувачів, точність результатів, продуктивність праці, думка користувачів.

Вищевикладені питання розглянуті у стандарті ISO/IEC 9126 (parts 1-4): модель якості – part 1; зовнішні метрики – part 2; внутрішні метрики – part 3; експлуатаційні метрики – part 4.

2.2. Оцінювання рівня якості ПС

З*агальна формула моделі якост*і (3.1) приведена в лабораторній роботі 1.3. Таким чином, враховуючи (3.1), інтегральний (узагальнений) рівень якості ПС U_q можна обчислити як середньозважений показник [10]:

$$U_q = \sum_{i=1}^{I} W_i \sum_{j=1}^{J_i} W_{ij} \sum_{k=1}^{K_{ij}} W_{ijk} Q_{ijk} \qquad \textbf{(4.1)}$$

У формулі (4.1): W_i – коефіцієнт ваги *i*-ї характеристики якості, W_{ij} – коефіцієнт ваги *j*-ї підхарактеристики *i*-ї характеристики якості, W_{ijk} – коефіцієнт ваги *k*-го атрибута *j*-ї підхарактеристики *i*-ї характеристики якості, Q_{ijk} – рівень якості *k*-го атрибута *j*-ї підхарактеристики *i*-ї характеристики якості, який обчслюється по наступній формулі:

$$Q_{ijk} = \frac{P_{ijk}}{PB_{ijk}} \qquad \textbf{(4.2)}$$

У формулі (4.2): P_{ijk} – обчислений рівень якості атрибута, PB_{ijk} – базовий рівень якості згідно обраної метрики (максимально можливе значення міри атрибута).

Приведемо приклад обчислення рівня якості *атрибута*, який вимірюється відповідно до метрики *completeness (закінченість, завершеність)* підхарактеристики *suitability* (*функціональна*

повнота) характеристики *функціональність*. Це буде перший атрибут першої підхарактеристики першої характеристики якості моделі, тобто: Q_{111}. Міра цього атрибута вимірюється відповідно до метрики *X*. Рівень якості цього атрибута обчислюється по формулі:

$$Q_{111} = X = 1 - A / B \quad , \qquad \textbf{(4.3)}$$

У формулі (4.3): *A* – кількість нереалізованих функцій; *B* – кількість функцій, котрі мають бути реалізовані відповідно до специфікацій та описів ПП. Тут $0 \le X \le 1$. Природно, що чим *X* ближче до 1, тим краще.

3. Порядок виконання роботи

Вважається, що ПС створена для функціонування в межах заданої предметної області відповідно до вимог, визначених студентами у лабораторних роботах 1.1 і 1.2. ПС має повністю відповідати переліку вимог із лабораторних робіт 1.1 та 1.2.

В процесі виконання роботи 1.3 студенти побудували дві моделі якості для оцінювання рівня якості програмної системи. Це модель зовнішньої якості і модель якості у використанні (модель експлуатаційної якості). Рівень якості ПС слід розраховувати так:

1) Рівень якості, досягнутий кожним атрибутом із формули (3.1), будемо вимірювати за допомогою зовнішніх метрик, які обирає студент із стандарту ISO/IEC 9126-2 для відповідних підхарактеристик. Для значень кожної метрики треба побудувати відповідну шкалу ранжування (див. рис.4.1). У цій шкалі мають бути відображені відмінні, добрі, задовільні та незадовільні множини числових значень у балах.

2) Якщо всі конфлікти у моделі узгоджені та виміряні значення показників якості відповідають вимогам, то подальшу оцінку можна провести, використовуючи інтегральний показник якості. Інтегральний рівень якості ПС будемо обчислювати по формулі (4.1). Міри якості атрибутів будемо обчислювати, зважаючи на формули (4.2) та (4.3). Однак маємо на увазі, що формули метрик для обчислення відповідних мір обираємо із стандарту ISO/IEC 9126-2 для кожного атрибута якості ПС. В результаті виконаних дій студенти повинні отримати загальну шкалу ранжування, яку можна застосувати для оцінювання інтегрального рівня якості ПС. У цій шкалі мають бути відображені сукупні множини відмінних,

добрих, задовільних та незадовільних числових значень рівня якості ПС із зазначенням граничних балів.

3) Користуючись стандартом ISO/IEC 9126-4, студенти в лабораторній роботі 1.3 побудували експлуатаційну модель якості ПС, що функціонує в заданому предметному середовищі. Слід обрати не менше чотирьох характеристик якості у використанні та відповідні метрики для обчислення досягнутого рівня експлуатаційної якості. Необхідно також визначити, як (яким чином) характеристики та атрибути зовнішньої якості відображаються на характеристики експлуатаційної якості ПС. Інтегральний рівень експлуатаційної якості ПС обчислюється так само, як він обчислювався для зовнішньої якості.

4. Завдання

Для заданої предметної галузі розрахувати рівень якості ПС:

1. Підготовка та захист дипломних проектів

Дипломники виконують дипломні проекти на випускаючій кафедрі по конкретній спеціальності. Дипломник має керівників по основній частині проекту та по охороні праці. Кожний керівник може керувати декількома дипломниками. Випускаюча кафедра характеризується назвою, номером і назвою спеціальності. Дипломник характеризується ПІБ, № групи, темою диплому. Керівник характеризується ПІБ, вчена ступінь, посада. Дипломні проекти захищаються у Державній екзаменаційній комісії.

2. Діяльність будівельної організації

Будівельна організація будує різні будинки (житлові, котеджі, офіси та ін.) на замовлення. На будівництво будинку оформлюється наряд, де вказано номер, постанову міськради, адресу будівельного майданчика, вартість, дату початку й дату закінчення робіт. В будівництві приймають участь декілька бригад робітників різних спеціальностей, які використовують різні матеріали. Будівля характеризується типом, адресою майданчика, виконробом, датою здачі. Одна бригада може працювати на декількох будовах.

3. Діяльність авіакомпанії

Авіакомпанія виконує пасажиро- та вантажоперевезення й характеризується назвою, номером ліцензії, адресою, аеропортом базування. Перевезення виконуються повітряними судами (ПС) певними рейсами. Рейс має назву, номер, аеропорт відправлення та

аеропорт прибуття, кількість пасажирів і/або вантажу, дату, час. На перевезення вантажів виписується накладна, із зазначенням вартості перевезення одиниці вантажу. В авіакомпанії на різних посадах працюють співробітники, які обслуговують ПС та рейси і характеризуються ПІБ, спеціальністю та посадою.

Контрольні питання

1. Описати типи метрик та види шкал їх вимірювання згідно рекомендацій стандарту ISO/IEC 9126-2 та ISO/IEC 9126-4.
2. Що таке міра якості? Пояснити різницю між мірою та метрикою якості. Привести приклад.

Література: [4, 7, 10].

Лабораторна робота 2.2
ОЦІНЮВАННЯ ВНУТРІШНЬОЇ ЯКОСТІ ПС НА ОСНОВІ ISO/IEC 9126-3. ЗАСТОСУВАННЯ СИСТЕМИ МЕТРИК М.ХОЛСТЕДА ТА Т.МАККЕЙБА

1. Мета роботи

Метою цієї роботи є дослідження методів оцінки рівня якості ПС відповідно до загальної моделі якості ПС, що запропонована у стандарті ISO/IEC 9126-1. Студенти вчаться застосовувати характеристики, підхарактеристики та метрики якості стандартів ISO/IEC 9126-1 та ISO/IEC 9126-3 для побудови внутрішньої моделі якості ПС, що функціонує у визначеній предметній галузі. Після побудови моделі студенти оцінюють рівень внутрішньої якості ПС, використовуючи внутрішні метрики якості (стандарт ISO/IEC 9126-3) із залученням системи метрик М.Холстеда та Т.Маккейба, особливо для прогнозування кількості дефектів та складності програмних модулів та підсистем.

2. Теоретичні відомості

2.1. Основи метричної теорії програм

Метричну теорію програм представляють різні математичні моделі визначення чисельних значень характеристик програмного забезпечення, у тому числі характеристик якості.

Кожна модель представляє ту чи іншу метрику програми. По типу одержуваної інформації про метрики програм ці моделі можна розбити на наступні групи: оцінюючі відхилення від норми, а також ті, що прогнозують значення характеристик, які й формують прийняття рішення про відповідність програмного забезпечення заданим вимогам. По типу використовуваної інформації про програми в моделях розрізняють метрики,

засновані на лексичному аналізі програм; на аналізі потоку управління; на аналізі модульних та міжмодульних зв'язків і метрики, а також засновані на аналізі потоку даних [2, 4, 6].

Метрики всіх цих груп використовують, головним чином, для прогнозування й оцінки складності й коректності програм. До них відносять метрики Холстеда, Джилба, Маккейба, Майерса та ін. Метрики модульних і міжмодульних зв'язків є основними характеристиками складності програмних комплексів у фазі проектування. Акумулюючи потік керування і потоки даних, ці метрики утворюють шкали функціональної міцності (зв'язності) і зчеплення модулів. Можлива непряма оцінка надійності за принципом: «чим менше складність, тим більше надійність». При цьому задача проектування надійного ПЗ зводиться до досягнення максимальної міцності і мінімального зчеплення модулів.

Історично першими з'явилися математичні моделі, що представляють метрики програм, засновані на аналізові лексики і потоку управління програм, що реалізують заданий алгоритм.

Представниками таких метрик є метрики Холстеда і Маккейба.

2.2. Метрики Холстеда

Метрики Холстеда відбивають лексичний підхід до виміру характеристик програмного забезпечення, заснований на вимірних властивостях алгоритмів. Властивості будь-якого опису алгоритму (або програми), на думку Холстеда, можуть бути виміряні чи обчислені на основі наступних метричних характеристик:

n1 **-** кількість різних (відмінних один від одного) операторів програми;

n2 **-** кількість різних (відмінних один від одного) операндів програми;

N1 **-** загальна кількість операторів програми;

N2 **-** загальна кількість операндів програми.

На цій основі Холстед визначає наступні метрики:

словник програми (в умовних одиницях)

$$n = n1+n2 \qquad \textbf{(5.1)}$$

довжина реалізації (в умовних одиницях)

$$N = N1+N2 \qquad \textbf{(5.2)}$$

довжина програми (в умовних одиницях)

$$\tilde{N} = (n1 \times \log_2 n1)+(n2\times\log_2 n2) \qquad \textbf{(5.3)}$$

обсяг програми (у бітах)

$$V = (N1+N2) \times \log_2(n1+n2) \quad \textbf{(5.4)}$$

При визначенні обсягу програми припускаємо, що позначення операндів і операторів потребує не більше одного символу.

потенційний обсяг програми

$$V^* = (n2^*+2) \times \log_2(n2^*+2), \quad \textbf{(5.5)}$$

де ***n2**** **-** загальне число вхідних і вихідних параметрів.

Припускаємо, що в програмах ідеальних з погляду економії витрат пам'яті, по-перше: оператори та операнди не повторюються; по-друге: всі операнди є або вхідними, або вихідними параметрами; по-третє: для запису тексту програми досить двох операторів (опису заголовка процедури-функції і присвоювання значення). Використаємо вираз (5.4) для обсягу програми, за умови, що N1=n1=2 і N2=n2=n2* .

рівень програми (в умовних одиницях)

$$L = V^*/ V \cong (2\times n2)/(n1\times N2), \quad \textbf{(5.6)}$$

якщо n2*=2 .

рівень мови

$$\lambda = L\times V^*, \quad \textbf{(5.7)}$$

інтелектуальний зміст програми (в умовних одиницях)

$$I = L\times V \cong (2\times n2/n1\times N2)\times(N1+N2)\times\log_2(n1+n2) \quad \textbf{(5.8)}$$

робота з програмування (в умовних одиницях)

$$E = V/L = V^2/V^* \quad \textbf{(5.9)}$$

час на програмування (в умовних одиницях)

$$T = E/S, \quad \textbf{(5.10)}$$

чи $T \cong (n1\times N2\times\log_2 n\times(n1\times\log_2 n1+n2\times\log_2 n2))/(2\times n2\times S)$, **(5.11)**

де ***S*** – число Страуда (***5<S<20***).

Число Страуда ***S*** визначається як число «страудовських моментів» у секунду. «Страудовський момент» - це час, необхідний людині для виконання елементарного розрізнення об'єктів, подібно розрізненню кадрів фільму. Страуд винайшов, що людина здатна розрізняти від 5 до 20 об'єктів у секунду.

Потенційний обсяг програми є мірою мінімально необхідного обсягу програми з заданим словником. Потенційний обсяг не залежить від мови реалізації. При перекладі програми з однієї мови на інший потенційний обсяг не міняється, але дійсний обсяг V чи збільшується, чи зменшується в залежності від мови реалізації.

Використовуючи вираження для потенційного обсягу програми, Холстедом отримані наступні метрики:

Рівень програми $L \leq 1$ характеризує ефективність реалізації алгоритму щодо витрат пам'яті. Тільки для найбільш компактної форми реалізації алгоритму $(V=V^*)$ рівень програми дорівнює 1. Всім іншим варіантам реалізації відповідають значення $L<1$.

Рівень мови λ – це коефіцієнт пропорційності зміни обсягу програми при перекладі з однієї мови на іншу так, що добуток рівня програми на потенційний обсяг залишається незмінним.

Інтелектуальний зміст характеризує міру «сказаного» у програмі, чи її «інформативність». Інтелектуальний зміст (рівень) програми сильно корелює з потенційним обсягом ($I \approx V^*$) і теж не залежить від мови реалізації.

Робота з програмування (рівняння розумової роботи) характеризує величину розумової роботи, зв'язаної із написанням програмного коду. Оскільки сума квадратів двох величин завжди менше квадрата їхньої суми, рівняння роботи дає підставу для розбиття програми на складові частини – модулі. Модульність знижує об'єм програмування. Найбільш продуктивною є ситуація, при якій для одержання одного результату використовується не більше п'яти об'єктів. У прикладному відношенні цей результат називають гіпотезою про «шість об'єктів». Для визначення кількості модулів M у програмі Холстед рекомендує використовувати вираз:

$$M = n2^*/6, \qquad \textbf{(5.12)}$$

де $n2^*$ – загальна кількість вхідних і вихідних змінних у програмі.

З рівняння роботи отримаємо наступне рівняння помилок :

$$B = L \times E/E_0, \qquad \textbf{(5.13)}$$

де B – кількість помилок у програмі, E_0 – середнє число елементарних відмінностей між помилками програмування.

Використовуючи перетворене рівняння роботи:

$$E = (V^*)^3/\lambda^2, \qquad \textbf{(5.14)}$$

а також значення рівня англійської мови (λ=2,16), як аналог мови програмування і гіпотезу про «шість об'єктів» ідеальної по витратах пам'яті програми ($n1=n1^*=2$, $n2=n2^*=6$), Холстед вивів наступне рівняння для прогнозу кількості помилок у програмі:

$$B = E^{2/3}/3000, \qquad \textbf{(5.15)}$$

чи: B= V/3000, **(5.16)**

де ***V*** **–** обсяг програми (5.4).

Крім свого прямого призначення ***в практичному відношенні*** метрики довжини програми (5.3) і довжини реалізації (5.2) можна використовувати для виявлення недосконалостей програмування. Якщо розрахунки довжини програми і довжини реалізації відрізняються більш ніж на ***десять відсотків***, то це свідчить про можливу наявність у програмі таких *6 класів недосконалостей:*

1. Наявність послідовності доповнюючих один одного операторів до того ж самого операнду, наприклад, A+C–A.
2. Наявність неоднозначних операндів, наприклад, A=D і A=C.
3. Наявність операндів-синонімів, наприклад, A=B и T=B.
4. Наявність загальних підвиразів: (A+B)×C + D×(A+B).
5. Непотрібне присвоювання, наприклад C=A+B, якщо змінна C використовується в програмі тільки один раз.
6. Наявність виразів, що не представлені в згорнутому виді як добуток множників, наприклад X×X+2×X×Y+Y×Y не представлено як (X+Y)×(X+Y).

Довжину реалізації ***N*** (5.2) можна використати для *прогнозу кількості фактичних машинних команд* ***P*** за допомогою виразу:

$$N = \frac{8}{3} \times P, \qquad \textbf{(5.17)}$$

чи більш грубо, за допомогою нерівності:

$2 \times P \le N \le 4 \times P$ **(5.18)**

Рівняння роботи (5.9) можна використати для *оцінки економічної ефективності використання* тієї чи іншої мови програмування. Відносне скорочення роботи з програмування в залежності від рівня мови використовують як покажчик ефективності впровадження мови програмування у практику.

Рівень програми $0 < L \le 1$ можна використовувати для оцінки *складності варіантів реалізації заданого алгоритму* ***D*** (чим менше витрат пам'яті, тим складніше варіант програми):

$$D = 1/L \approx \frac{n1 \times N2}{2 \times n2} \qquad \textbf{(5.19)}$$

Фахівець, який добре знає мову програмування, зрозуміє програму тим швидше, чим менше її обсяг, тобто вище рівень. Але для людини, що менше разбирається в програмуванні, потрібно

більший її обсяг і менший рівень. Встановлено, що для будь-якого алгоритму, описаного різними мовами, зі збільшенням обсягу програми ***V*** рівень програми ***L*** зменшується в тій же пропорції. Тому добуток рівня програми на обсяг є постійною величиною, рівною потенційному обсягу реалізації даного алгоритму:

$$L \times V = V^* = const. \quad \textbf{(5.20)}$$

Якщо мова не міняється, а міняється тільки алгоритм, то для будь-якої мови добуток потенційного об'єму на рівень програми залишається постійною величиною, рівною рівню мови:

$$L \times V^* = \lambda = const. \quad \textbf{(5.21)}$$

2.3. Метрика Маккейба

Метрика Маккейба заснована не на аналізі лексичних особливостей програмної реалізації різноманітних алгоритмів рішення задач, а на аналізі потоку передач управління від одного оператора до іншого. Це дозволяє, на відміну від метрик Холстеда, врахувати логіку програми при оцінці її складності.

Програма (алгоритм, специфікація) має бути представлена у вигляді управляючого орієнтованого графа G=(V, E) із ***V*** вершинами і ***E*** дугами, де вершини відповідають операторам, а дуги – переходам від одного оператора до іншого. Граф, що описує програму у виді вершин-операторів і дуг-переходів, називають *графом* керування чи управляючим графом програми.

Приклад. Нехай мається програма, що зчитує із вхідного потоку символи доти, поки не буде виявлений символ «#». Текст програми мовою Сі представлений на рис.5.1. Відповідний граф керування програми показаний на рис.5.2.

Для представлення програми у виді графа необхідно визначити угоду про те, що вважати вузлом графа, бо синтаксис операторів у мовах програмування є досить різноманітним. Звичайно враховують тільки виконувані оператори, виключаючи оператори опису даних. Бажано підібрати такі синтаксичні форми операторів, що найбільшою мірою підходять для відображення вузлом графа. Лінійні ділянки програми можна замінити одним вузлом графа. У цьому відношенні використання схем чи алгоритмів опису програм може виявитися навіть більш кращою формою опису програми, ніж текст мовою програмування. У будь-якому випадку, бажано перетворити оператори циклу в еквівалентну послідовність

операторів розгалуження, додавши, при необхідності, лічильники числа повторень циклу з «верхнім» чи «нижнім» завершенням.

```
main()
{ int zeich = 'x';
   while(zeich != '#')
          {  printf(" Продовжити введення ");
             zeich = getchar();
          }
   printf("Введення завершене "); }
```

Рис. 5.1. Програма введення даних

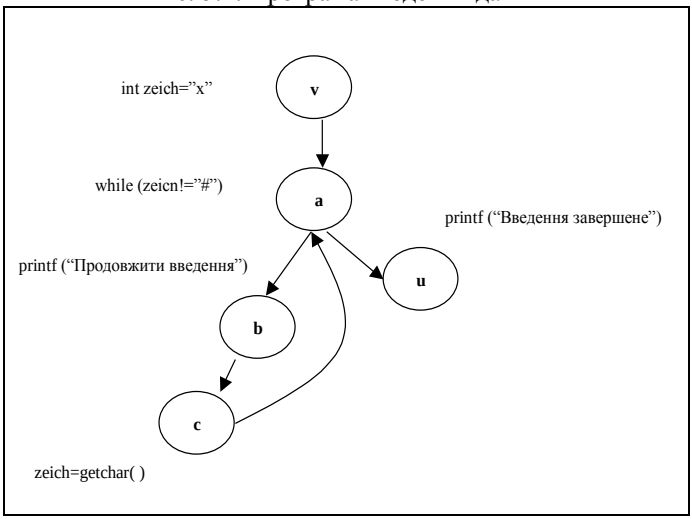


Рис. 5.2. Граф керування програми введення даних

Метрика Маккейба є цикломатичним числом графа передач управління програми і визначається виразом:

$$M = m - n + 2, \qquad \textbf{(5.22)}$$

де ***m*** – кількість ребер графа, ***n*** – кількість вершин графа. Величину M, розраховану по формулі (5.22), називають ***цикломатичним числом Маккейба***.

Теоретичною базою визначення цикломатичного числа Маккейба є теорія графів. У теорії графів цикломатичне число орієнтованого графа визначається виразом:

$$Z = m - n + 2 \times p, \qquad \textbf{(5.23)}$$

де ***m*** – кількість ребер, ***n*** – кількість вершин, ***p*** – кількість компонентів зв'язності графа.

Число компонентів зв'язності ***p*** можна розглядати як мінімально необхідну кількість ребер, які потрібно додати до графа, щоб зробити його повнозв'зним. Повнозв'язним вважають граф, у якого існує шлях з будь-якої вершини графа в будь-яку іншу вершину графа. Як показали дослідження, для графів керування програм повнозв'язність забезпечується додаванням однієї фіктивної дуги з кінцевої вершини в початкову, тобто у нашому прикладі із вершини **u** у вершину **v**.

Тому справедливо вважати, що для будь-якого графа управління програми число компонентів зв'язності дорівнює одиниці (***p=1)***. Підстановка ***p=1*** у формулу (5.23) дає цикломатичне число Маккейба (5.22). Визначимо цикломатичне число Маккейба для графа керування програми, зображеного на рис.5.2. Кількість ребер графа дорівнює п'яти, кількість вершин теж дорівнює п'яти, тому цикломатичне число дорівнює: M = 5 – 5 + 2 = 2.

Розглянемо фізичний зміст цикломатического числа.

Для кожної дуги ***e=(v, u)| e∈E*** графа керування перша вершина (***v***) є вихідною, а друга (***u***) – кінцевою. *Шлях* від однієї вершини до іншої вершини графа, якщо він складається більш ніж з однієї дуги, описують послідовністю вершин відповідних дуг ***(v, a,…,u)*** так, що вихідна вершина буде першою в перерахуванні, а кінцева – останньою. *Контур* – це площина, обмежена циклічним шляхом, у якій початкова і кінцева вершина графа збігаються. Кожному контуру відповідає обмежуючий його шлях, що веде з початкової вершини графа в кінцеву. *Цикломатичне число* визначає кількість незалежних контурів у повнозв'язному графі і, як наслідок, кількість різних шляхів, що ведуть з початкової вершини в кінцеву.

У практичному аспекті цикломатичне число є мірою складності програми і визначає кількість тестів, достатніх для тестування за критерієм покриття всіх гілок програми. При оцінці складності програми діє наступне правило: якщо цикломатичене число більше десяти, програма має зайву складність і її варто розбити на складові частини (незалежні модулі) з меншим значенням цикломатичного числа.

Таким чином, програма організації введення даних, що має граф керування з цикломатичним числом, рівним 2, не має зайву

складність і для її тестування досить підібрати два тести, що покривають гілки (a, b, c) і (a, u) (рис.5.2). Помітимо, що вершини **v** і **a** графа керування програми можна об'єднати, що не приведе до зміни цикломатичного числа Маккейба.

Цикломатичне число залежить тільки від кількості управляючих операторів, що містять умовні вирази (предикати). При цьому складність предиката не враховується. Якщо в операторі циклу *while* програми організації введення даних (рис.5.1) ускладнити предикат, змінивши заголовок:

while(zeich != '#') на while(zeich != '#' || zeich != ''),*

граф керування і цикломатичне число залишаться тими ж самими.

Якщо програма містить тільки оператори розгалуження (без операторів вибору і переключення), цикломатичне число можна визначити, використовуючи вираз: M=u+1, **(5.24)**

де u – кількість операторів розгалуження.

При необхідності оператори вибору чи переключення з n гілками, можна розглядати як n операторів розгалуження.

Дослідження показують, що не існує єдиної метрики, яка б забезпечила універсальний підхід до кількісної оцінки якості ПС. Виміри й оцінка якості дають спектр метрик, що є основою для прийняття рішень у процесі розробки і супроводження ПЗ.

3. Порядок виконання роботи

Вважається, що ПС створена для функціонування в межах заданої предметної області відповідно до вимог, визначених студентами у лабораторних роботах 1.1 та 1.2.

В процесі виконання роботи 1.3 студенти побудували дві моделі якості для оцінювання рівня якості програмної системи. Це модель зовнішньої якості ПС і модель якості у використанні (модель експлуатаційної якості ПС). При виконанні лабораторної роботи 2.1 студенти обчислили рівень якості ПП із залученням зовнішніх метрик та метрик експлуатаційної якості.

В роботі 2.2 необхідно визначити показники внутрішньої якості та обчислити інтегральний рівень якості ПП відповідно до стандарту ISO/IEC 9126-3 (внутрішні метрики) із залученням метрик Холстеда та Маккейба. Слід встановити взамозв'язок між внутрішніми та зовнішніми показниками властивостей ПП і доповнити модель якості ПП. Для цього виконаємо такі кроки:

1) Основну увагу необхідно зосередити на нефункціональних характеристиках ПП (надійність, зручність використання, ефективність, супроводжуваність, переносимість). Особливо важливою з них є характеристика надійність. Для усіх трьох її підхарактеристик слід розрахувати оцінки за допомогою метрик Холстеда (формули (5.1)-(5.14)) і окремо побудувати прогнозні оцінки для помилок (формули (5.15), (5.16)). Надалі шляхом обчислення мір підхарактеристик: завершеність, відмовостійкість і відновлюваність перевірити прогнозні оцінки. Міри цих підхарактеристик обчислюються відповідно до метрик, які залежать від результатів спільних перевірок програмного коду, які будемо вважати відомими (ISO/IEC 9126-3). Крім того, ці оцінки мають корелюватися із відповідними зовнішніми метриками, які можуть бути обчислені при тестуванні ПС (ISO/IEC 9126-2).

2) Із характеритики функціональність слід виконати розрахунки тільки для підхарактеристики захищеність (security). Кількість різних операторів та операндів у кожній підсистемі ПС, а також загальну їх кількість вважаємо відомою. Відомими, також, є код та документація на ПП. Складність алгоритмів оцінюється для розроблених у роботі 1.2 двох діаграм взаємодії об'єктів в межах сценаріїв, для яких виконаємо всі дії згідно підходу Маккейба для розрахунку цикломатичного числа відповідних алгоритмів.

4. Завдання

Для проблемної галузі обчислити рівень внутрішньої якості ПС:

1. Діяльність спортивного клубу

Деякий спортивний клуб має декілька команд в ігрових видах спорту, зокрема футбольну команду, яка приймає участь у чемпіонаті та проводить вдома ігри на певному стадіоні. Спортивний клуб має назву, кольори, місто базування, Раду директорів. Команда має назву, тренера, адресу базування, гравців. Кожний гравець характеризується ПІБ, амплуа (нападник, захисник і т.п.), зростом, вагою, віком. Команда проводить матчі на стадіонах, які відвідують болільники.

2. Обслуговування банківських рахунків

Клієнт має банківські рахунки в різних банках, які характеризується назвою, адресою, номером ліцензії. Кожним банком керує Рада директорів. Клієнт характеризується ПІБ,

адресою, ідентифікаційним кодом. Банківський рахунок характеризується номером, видом рахунку, сумою. Банківський рахунок можна відкрити або закрити, зняти чи переказати з нього певну суму коштів, додати готівку, нарахувати відсотки тощо. Банк також надає клієнтам кредити, які характеризуються назвою, типом, датами видачі та погашення, річним відсотком.

3. Діяльність проектної установи

Проектна установа ліцензована на розробку архітектурних проектів і складається з декількох відділів. Кожен відділ очолює завідувач і в ньому працюють співробітники на певних посадах. Відділи мають назву та розташовані в певних приміщеннях, де маються телефони. Співробітники виконують проекти, котрі мають назву, дату початку і закінчення, а також керівника. Один співробітник може виконувати декілька проектів. Виконані проекти розглядаються та затверджуються Державною архітектурною комісією, котра розглядає проекти в зазначені терміни.

4. Діяльність загальноосвітньої школи

Загальноосвітні школи, в яких навчаються учні, характеризуються номером, назвою, адресою, ПІБ директора. В школах мається певна кількість класів, котрі мають назву, класного керівника, список учнів. Дисципліни викладаються педагогами, причому один вчитель може викладати декілька предметів, а однакові дисципліни можуть викладатися різними вчителями. Предмети викладаються згідно розкладу у кабінетах, котрі мають номер, назву, відповідне обладнання. Предмети мають назву, кількість годин вивчення, список навчальних посібників.

Контрольні питання

1. Визначити основні метрики Холстеда (довжина реалізації; словник, довжина та обсяг програми та ін.) та число Маккейба.

2. Які метрики вважаються зовнішніми, а які – внутрішніми? Привести приклади. Про що йдеться в стандарті ISO/IEC 9126-3?

Література: [2, 4, 6].

СПИСОК ЛІТЕРАТУРИ

1. *Бабенко Л.П., Лавріщева К.М.* Основи програмної інженерії. Навч. посіб. –К.: Т-во "Знання", КОО, 2001. – 269 с.

2. *Брауде Э.* Технология разработки программного обеспечения. – СПб.: Питер, 2004. – 655 с.

3. *Райчев І.Е., Харченко О.Г., Замковий В.В.* Принципи проектування відкритих розподілених систем : навч. посіб. –К.: Вид-во Нац. авіац. ун-ту "НАУ-друк", 2010. – 240 с.

4. *Лавріщева К.М.* Програмна інженерія. Підручн. –К.: Академперіодика, 2008. – 320 с.

5. *Канер Сэм, Фолк Дж., Нгуен Енг Кен.* Тестирование программного обеспечения : Пер. с англ. –К.: Диасофт, 2001. –544с.

6. *Соммервилл И.* Инженерия программного обеспечения. – М.: Изд. дом Вильямс, 2002. – 624 с.

7. *Основы инженерии качества программных систем* / Ф.И.Андон, Г.И.Коваль., Т.М.Коротун, Е.М.Лаврищева, В.Ю.Суслов. –К.: Академпериодика, 2007. –672 с.

8. *Липаев В.В.* Выбор и оценивание характеристик качества программных средств. Методы и стандарты. –М.: СИНТЕГ, 2001. – 228 с.

9. *ДСТУ 2462–94.* Сертифікація. Основні поняття. Терміни та визначення. – Чинний від 01.01.95. –К.: Держстандарт України, 1994. – 27 с.

10. *Райчев І.Е., Харченко О.Г.* Концепція побудови сертифікаційної моделі якості програмних систем // Проблемы программирования. –2006. –№2-3. – С. 275–281.

ЗМІСТ

Навчально-методичне видання

СТАНДАРТИЗАЦІЯ ТА СЕРТИФІКАЦІЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ

Лабораторний практикум
для студентів спеціальності 7/8.05010101
“Інформаційні управляючі системи
та технології (за галузями)”

Укладачі: РАЙЧЕВ Ігор Едуардович,
ХАРЧЕНКО Олександр Григорович

Редактор *С.М. Барабаш*
Технічний редактор *А.І. Лавринович*
Коректор *І.М. Вихованець*
Комп’ютерна верстка *О.М. Іваненко*

Підп. до друку 16.08.2012. Формат 60x84/16. Папір офс.
Офс.друк. Ум. друк. арк. 2,79. Обл. вид. арк. 3,0.
Тираж 100 пр. Замовлення № 148-1.

Видавець і виготівник
Національний авіаційний університет
03680, Київ-58, проспект Космонавта Комарова, 1
Свідоцтво про внесення до Державного реєстру ДК № 977 від 05.07.2002
Тел. (044) 406-78-28, Тел./факс (044) 406-71-43
E-mail: publish@nau.edu.ua