Вдосконалення системи управління якості програмного забезпечення

Студент групи ІЯП-18м Петровський Д. Ю.

Керівник: к.т.н., доцент Маньковська В.С.

Актуальність

- Сьогодні галузь розробки програмного забезпечення (ПЗ) вважається найбільшою галуззю світової економіки. За деякими оцінками на 2015 рік кількість розробників ПЗ у світі сягнула 19 мільйонів осіб, а до 2019 року прогнозується зростання їх кількості до 25 мільйонів.
- За час становлення галузі, завдання, їх складність, форми подання даних та методи їх обробки сильно змінились, але до сьогодні розробка якісного програмного забезпечення не стала нормою.
- Тому пошук шляхів покращення системи управління якістю програмного забезпечення залишається **актуальною** задачею.

Мета та задачі

- Метою роботи є пошук шляхів вдосконалення системи управління якості програмного забезпечення в рамках існуючих методик та підходів до оцінки якості.

Для досягнення мети сформульовано наступні задачі:

- провести огляд існуючих систем якості;
- проаналізувати методики та підходи до оцінки якості програмного забезпечення;
- визначити найбільш ефективні, визначити недоліки, запропонувати шляхи покращення;
- підвищити ефективність методів та підходів до визначення якості ПЗ.

Об’єкт та предмет дослідження

- **Об’єктом** роботи є система управління якості програмного забезпечення.
- **Предметом** роботи є процес вдосконалення існуючої системи управління якості програмного забезпечення, підвищення ефективності методів та підходів до визначення якості ПЗ.

Якість ПЗ та методи її контролю

- Однією з найважливіших проблем забезпечення якості програмних засобів є формалізація характеристик якості і методологія їх оцінки.

Основні міжнародні стандарти в області програмних засобів

Якість ПЗ та методи її контролю

До методів контролю якості програмного забезпечення, серед іншого відносять:

- інспекції (огляди);
- тестування програмного забезпечення;
- статичний аналіз програмного забезпечення;
- імовірнісна оцінка надійності програмного забезпечення;
- аналіз видів, наслідків і критичності відмов програмного забезпечення

Якість ПЗ (Інспекції)

Етапи проведення інспекції за Фаганом

Етапи проведення інспекції за Глібом

Якість ПЗ (Тестування)

- У більш широкому сенсі тестування – це одна з технік контролю якості, що включає в себе активності з планування робіт (Test Management), проектування тестів (Test Design), виконання тестування (Test Execution) та аналізу отриманих результатів (Test Analysis).
- Основною метою процесу тестування – є доказ того, що результат розробки відповідає пред'явленим до нього вимогам. Основне завдання тестування ПЗ: отримання інформації про статус готовності заявленої функціональності системи або програми.

Якість ПЗ (Інші методи)

- Статичний аналіз коду (*англ.* Static Code Analysis) – аналіз програмного забезпечення, що виконується (на відміну від динамічного аналізу) без реального виконання досліджуваних програм
- Надійність програмного забезпечення набагато важливіше інших його характеристик, наприклад, часу виконання, і хоча абсолютна надійність сучасного програмного забезпечення, мабуть, недосяжна, досі не існує загальноприйнятої міри надійності комп'ютерних програм.

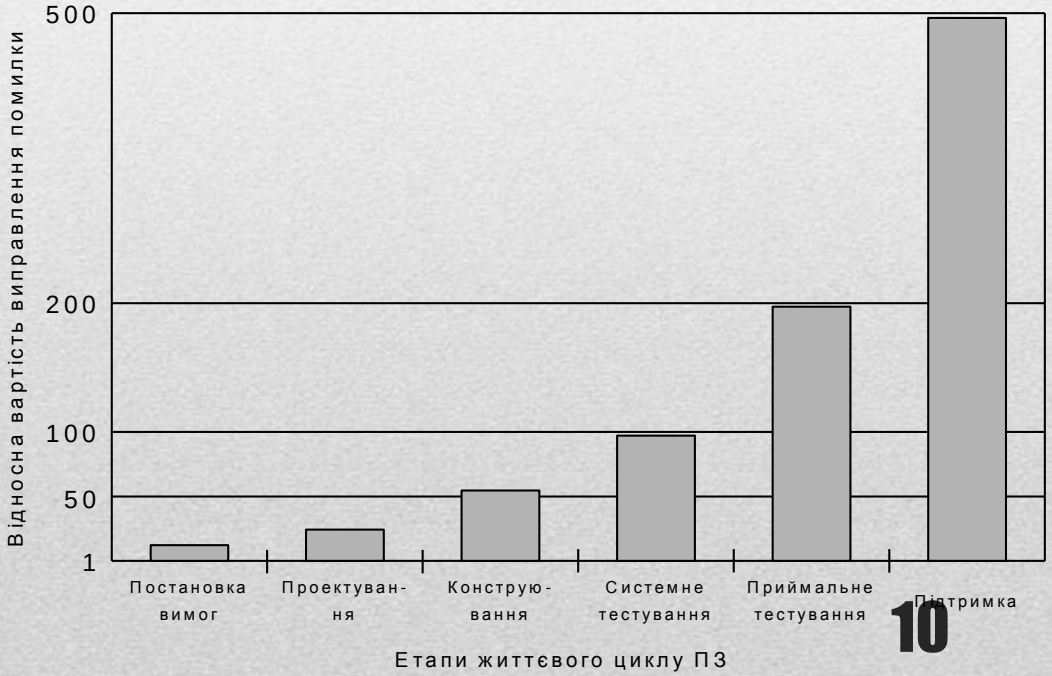
Визначення метрик якості

Розподіл помилок, які були допущені на різних стадіях розробки ПЗ

Етап життєвого циклу ПЗ	Об'єм програмного забезпечення				
	до 2КБ	до 8КБ	до 32КБ	до 128КБ	до 512КБ
Формулювання вимог	10%	15%	20%	22%	23%
Проектування	15%	19%	25%	28%	32%
Конструювання	75%	66%	55%	50%	45%

чим більший об'єм програмного забезпечення тим більше помилок вноситься саме на ранніх етапах створення програмного забезпечення.

Слід відмітити і той факт, що вартість виправлення помилки тим більша чим більший за номером етап розробки



Метрики якості та інтервали їх значень

Значення інтервалів метрик якості ПЗ

Метрики з точними значеннями	Значення
Метрика звертання до глобальних змінних	0..1
Кількість виявлених помилок при інспектуванні	0..2000
Метрики з прогнозованими значеннями	
Цикломатична складність	10..100
Відносна гранична складність програми	0..1

Ефективність метрик

Значення ефективності метрик на ранніх етапах життєвого циклу

№	Метрика якості	l_1	l_2	l_3	l_4	A
1	Метрика звертання до глобальних змінних	1	0,33	0,667	0,25	0,141
2	Кількість виявлених помилок при інспектуванні	1	0,25	0,33	0,25	0,114
3	Цикломатична складність	1	0,25	0,45	0,25	0,09
4	Відносна гранична складність програми	1	1	0,333	0,25	0,161

$$A = \left(\frac{1}{n}\right) \cdot \sum_{i=1}^{n} (l_i \cdot g_i)$$

Показник «розуміння інформації» l_1 дорівнює 1 для кожної з метрик.
Показник «повнота висвітлення предмету інтересу» l_2 залежить від повноти висвітлення метрикою інформації.
Показник «своєчасність інформації та її достатність для прийняття рішень» l_3 залежить від достатності інформації про оброблювану інформацію.
Показник «кількість інформації» l_4 дорівнюватиме одиниці

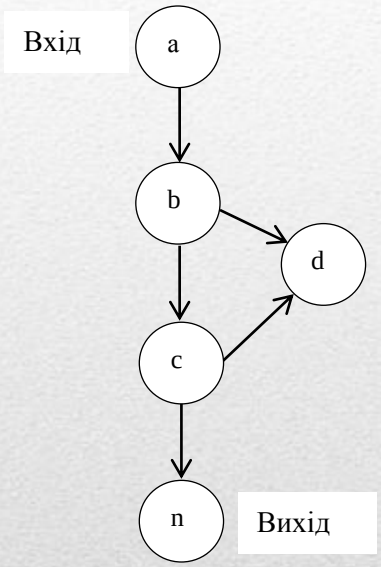
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Для розуміння послідовності дій, які описують програму від початку до кінця, було складане алгоритм у вигляді блок-схеми. Далі, було розроблено програмний код. Провівши огляд розробленого програмного коду та виявивши помилки складено звіт.

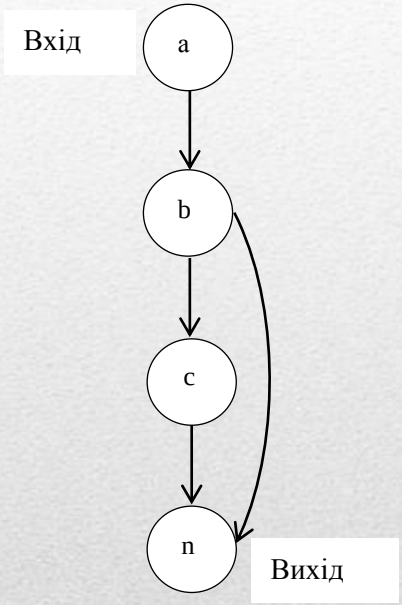
Оглядові помилки

Номер помилки	Назва модуля/функції	Опис помилки	Важливість помилки (висока, середня, низька)	Помилка виправлена Так/Ні
1	entry	Не вірно складена умова продовження виконання цикла	Середня	Так
2	regit	Умова не відповідає алгоритму	Середня	Ні
3	show per	Некоректно застосовується конструкція умовного переходу	Середня	Так
4	main	Кількість умовних розгалужень не відповідає завданню	Середня	Так

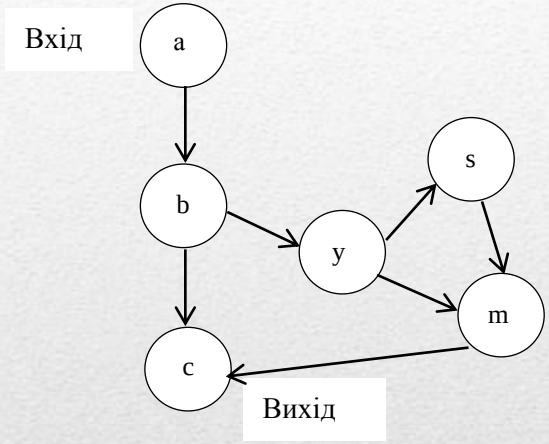
Для кожного модуля будуємо графи та обчислюємо циклометричні числа. Розробляємо тестові випадки для кожного графу та представляємо їх у вигляді таблиці.



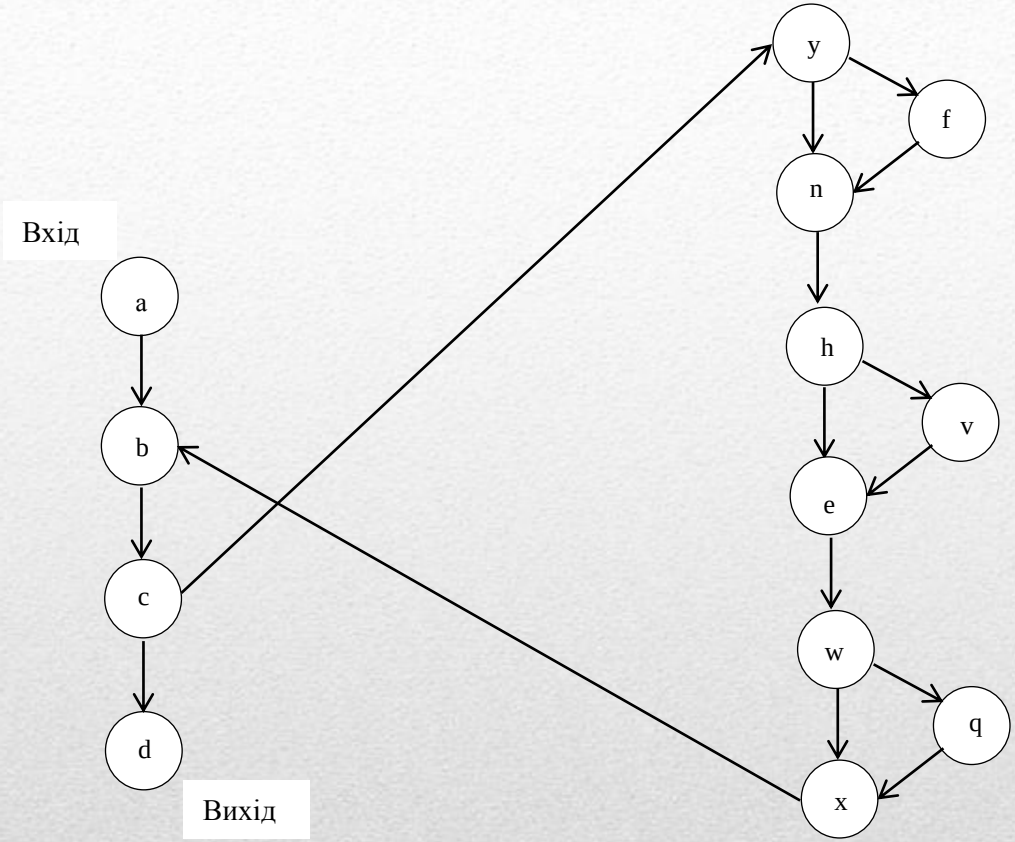
Граф модуля «entry»



Граф модуля «regit»



Граф модуля «show per»



Граф модуля «main»

Тестові випадки

G	Номер сценарія	Опис проходу	Контрольні приклади, що дозволяють реалізувати описану ситуацію	Тест пройдений Так/Ні
Модуль entry				
2	1	a-b-c-d-b-c-n	a>0	Так
	2	a-b-c-n	a=0	Ні
Модуль regit				
2	1	a-b-n	fout.is_open=1	Так
	2	a-b-c-n	fout.is_open=0	Так
Модуль show per				
2	1	a-b-y-s-m-c	file.is_open=1, file.eof=1	Так
	2	a-b-c	file.is_open=0, file.eof=0	Так
Модуль main				
4	1	a-b-c-y-f-n-h-e-w-x-b-c-d	true=1, ans =1	Так
	2	a-b-c-y-f-n-h-y-e-w-x-b-c-d	true=1, ans =2	Так
	3	a-b-c-y-f-n-h-y-e-w-q-x-b-c-d	true=1, ans =3	Так
	4	a-b-c-d	true=0	Так

Обчислимо циклометричну складність для кожного модуля за формулою:

$$M = E - N + 2P,$$

де M – цикломатична складність;

E – кількість ребер в графі;

N – кількість вершин в графі;

P – кількість компонентів зв'язності.

Отже, отримаємо такі результати циклометричної складності для модулів «entry», «regit», «show per», та «main», відповідно:

$$M_{entry} = 5 - 5 + 2 \times 1 = 2.$$

$$M_{regit} = 4 - 4 + 2 \times 1 = 2.$$

$$M_{show\_per} = 7 - 6 + 2 \times 1 = 3.$$

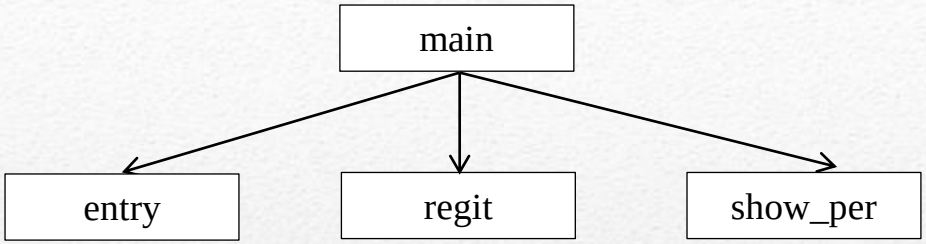
$$M_{main} = 16 - 13 + 2 \times 1 = 5.$$

Далі проводимо модульне тестування згідно складеним тестовим випадкам. Складаємо звіт про кожну знайдену похибку.

Модульне тестування

Номер помилки	Назва модуля	Опис проходу	Контрольні приклади, які дозволяють реалізувати описану ситуацію	Опис помилки	Важливість помилки	Помилка виправлена Так/Ні
1	entry	a-b-c-n	a=0	Некоректно відображаються введені дані	Середня	Так

Будуємо схему взаємодії модулів.



Розробляємо стратегію тестування взаємодії модулів у вигляді таблиці.

Номер в послідовності	Опис послідовності	Контрольні приклади, які дозволяють реалізувати описану ситуацію	Тест пройдений Так/Ні
1	main→entry	true=1	Так
2	main→regit	ans=1	Так
3	main→show_per	ans=2	Так

Висновки

- Аналіз проведений в роботі вказує на те, що сьогодні існує велика кількість стандартів та технологій покликаних підвищувати якість програмного забезпечення, проте якість готової продукції як і раніше залишається на достатньо низькому рівні.
- В результаті дослідження були визначені метрики якості, та визначена їх ефективність. Обраховані граничні значенні та діапазони зміни спираючись на дані проведених досліджень.
- Серед найбільш ефективних метрик якості придатних до застосування на ранніх етапах життєвого циклу ПЗ були обрані: метрика звертання до глобальних змінних, кількість виявлених помилок при інспектуванні, цикломатична складність, відносна гранична складність програми.
- Як перспективний напрямок розвитку системи якості був обраний напрямок розвитку статичного коду для контролю якості ПЗ на ранніх етапах життєвого циклу
- Була проведена оцінка результатів, в ході якої були проаналізовані сучасні програмні продукти, покликані автоматизувати процес визначення якості ПЗ на ранніх етапах життєвого циклу.
- Ефективність цикломатичної складності піднімається за рахунок попередньої кількісної оцінки алгоритмічних структур програми, які відрізняються від елементарних типу розгалуження, циклів на поряд, а той два і застосування цих заздалегідь отриманих даних для оцінки більш об’ємних структур.

Дякую за увагу