викладачів, співробітників, аспірантів і студ. фак-ту технічних систем та енергоефективних технологій (м. Суми, 14-17 квітня 2015 р.) / Редкол.: О Г. Гусак, В.Г. Євтухов. – Суми: СумДУ, 2015. – Ч.2. – С. 58-59.

2. Мілтих В.С. Підвищення енергоефективності насосної станції з насосами типу д впливом геометричних параметрів їх робочих коліс на форму напірної характеристики: дисертація на здобуття наукового ступеня / Мілтих В.С.; наук. кер. М.І. Сотник. – Суми: СумДУ, 2016. – 132 с.

3. Ратушный А.В. Повышение напорности ступени центробежного насоса путем усовершенствования лопастной решетки рабочего колеса: диссертация на соискание научной степени канд. технических наук / А.В. Ратушный; Науч. рук. И.А. Ковалев. – Сумы: СумГУ, 2015. – 154 с.

4. Пфлейдерер К. Центробежные и пропеллерные насосы / К. Пфлейдерер – М.-Л.: ИКТН, 1937. – 495 с.

4. Жумагулов Н.Ж. Исследование работы консольного центробежного насоса К 45/30 при различных загибах лопаток рабочего колеса: автореф. дисс. канд. техн. наук: 05.04.13 / Н.Ж. Жумагулов. – Алма-Ата., 1988. – 16 с.

5. Анисимов С.А. Влияние числа лопаток на эффективность центробежного колеса с однорядной решеткой / С.А. Анисимов, Ф.С. Ренстин, К.П. Селезнёв // Труды ЛПИ. – 1962. – №.221. – С. 32-46.

6. Малюшенко В.В Определение оптимального числа лопастей робочих колёс питательных насосов / В.В. Малюшенко // Изветия вузов. Сер. Энергетика. – 1964. – № 4. – С. 58-65.

7. Тхи Д.С. Исследование потока в рабочем колесе центробежного насоса низкой быстроходности / Д.С. Тхи, В.И. Арсеньев // Гидравлические машины. – 1967. – № 6. – С. 116-123.

8. Зотов В.Н. Выбор числа лопаток колеса и направляющего апарата центробежного колеса / В.Н. Зотов // Вестник машиностроения. – 1976. – № 11. – С. 33-36.

**Робачинська І.В.**

*студентка,*

*Черкаський державний технологічний університет*

## ПОСТАНОВКА ПРОБЛЕМИ ВИЗНАЧЕННЯ РИЗИКУ НЕКОРЕКТНОЇ РОБОТИ ПРОГРАМИ ПРИ ТЕСТУВАННІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Може оцінюватись:

- відповідність вимогам, якими керувалися проектувальники та розробники;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з програмним забезпеченням та операційними системами;

– відповідність задачам замовника.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів (помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів так і для замовників [1].

Процес тестування починається з першого етапу розробки програмного продукту, а виконуються тести, як тільки створено виконуваний код (навіть частково).

На сьогодні існує дуже багато різних видів та підходів тестування програмного забезпечення. Кожен тестувальник може сам підібрати зручні для нього засоби тестування та збереження їх результатів. Але програмний продукт така річ, що помилки в ньому не закінчуються, і тому постає питання: коли тестування повинно закінчитись?

Існує багато способів оцінки стану програмного продукту, але жоден з цих методів не може швидко і точно дати відповідь на питання: чи можна вже зараз впроваджувати програмну систему в роботу, та, які ризики при цьому виникнуть.

Оцінку стану програмного продукту можна отримати попрацювавши з метриками тестування. Метрика – це кількісний масштаб і метод, який може використовуватися для вимірювання. Якщо згрупувати метрики за типами сутностей, що беруть участь у забезпеченні якості та тестуванні програмного забезпечення, то можна виділити:

- Метрики за тестовими випадками;
- Метрики по багам/дефектам;
- Метрики за завданнями [2].

Звичайно, що немає ідеального підходу для визначення стану системи, але ознайомившись з найпоширенішими підходами можна виділити деякі найважливіші для нас критерії, які стануть основою нового методу оцінювання ступеню завершеності та визначення ризику некоректної роботи програмного продукту.

Дослідження критеріїв. Одним з прийомів тестування є автоматизація. Автоматизація здатна скоротити витрати часу на тестування та мінімізувати його трудомісткість. Такі тести видають зрозумілі результати: пройшов (passed), не вдався (failed) або заблокований (blocked). Вони створюються на основі тест-кейсів або тестових випадків. Тестовий випадок (Test Case) – це артефакт, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується, або її частини. Тест-кейси мають такі ж самі результати як і автоматизовані тести, адже останні на них базуються [3].

При тестуванні програмного продукту знаходяться баги, які описуються в баг репортах. Баг або дефект репорт – це документ, що описує ситуацію або послідовність дій, яка призвела до некоректної роботи об'єкта тестування, із зазначенням причин і очікуваного результату.

Баг репорт має такі властивості: серйозність та пріоритет. Серйозність (Severity) – це атрибут, що характеризує вплив дефекту на працездатність програми. Пріоритет (Priority) – це атрибут, який вказує на черговість виконання завдання або усунення дефекту. Чим вище пріоритет, тим швидше потрібно виправити дефект.

Градація Серйозності дефекту:

− S1 Блокуюча (Blocker). Блокуюча помилка, що приводить додаток в неробочий стан, в результаті якого подальша робота з тестованою системою або її ключовими функціями стає неможлива. Рішення проблеми необхідно для подальшого функціонування системи.

− S2 Критична (Critical). Критична помилка, неправильно працює ключова бізнес-логіка, діра в системі безпеки, проблема, яка призвела до тимчасового падіння сервера або приводить в неробочий стан деяку частину системи, без можливості вирішення проблеми, використовуючи інші вхідні точки. Рішення проблеми необхідно для подальшої роботи з ключовими функціями системи, що тестується.

− S3 Значна (Major). Значна помилка, частина основної бізнес-логіки не функціонує належним чином. Помилка не критична або є можливість для роботи з тестованою функцією, використовуючи інші вхідні точки.

− S4 Незначна (Minor). Незначна помилка, що не порушує бізнес-логіку частини програми, що тестується, очевидна проблема призначеного для користувача інтерфейсу.

− S5 Тривіальна (Trivial). Тривіальна помилка, яка не стосується бізнес-логіки додатка, погано відтворена проблема, малопомітна засобами призначеного для користувача інтерфейсу, проблема сторонніх бібліотек або сервісів, проблема, не надає ніякого впливу на загальну якість продукту [4].

Кожен баг репорт можна прив’язати до деякого тест-кейса, а отже, серйозність баг репорта та тест-кейса однакові.

В даному разі пріоритет не важливий для розрахунків, адже він вказує лише на порядок виконання роботи.

Таким чином для визначення ризику некоректної роботи програми при тестуванні достатньо 2-х критеріїв: серйозності та результату тест-кейсу.

Дані параметри використовує кожен тестувальник у своїй роботі, отже, працювати з таким підходом буде дуже легко та зручно, а його результат буде не суб’єктивним, а підтвердженим конкретними цифрами.

Проведене дослідження дало змогу зробити наступні висновки. Встановлено одну з основних потреб галузі тестування програмного забезпечення. Розглянуто існуючі методи, які можуть допомогти у вирішенні цього питання. Визначено, що є потреба у створенні універсального методу для швидкого отримання необхідної для роботи інформації. Також розглянуто критерії для оцінки стану програми та виділено основні з них.

Перспективи подальших досліджень полягають у розгляді та визначенні методів теорії прийняття рішень, які допоможуть у створенні універсального методу.

**Список використаних джерел:**

1. Державний університет комунікацій [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: http://www.dut.edu.ua/ua/news/1/category/1009/view/1503 – Тестування програмного забезпечення.
2. ПроТестинг [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: http://www.protesting.ru/qa/metrics.html – Метрики по обеспечению качества.
3. ПроТестинг [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: http://www.protesting.ru/testing/testcase.html – Тестовый случай (Test Case).
4. ПроТестинг [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: http://www.protesting.ru/testing/bugpriority.html – Серьезность и Приоритет Дефекта.

**Романюк О.Н.**
*доктор технічних наук, професор;*
**Мельник О.В.**
*аспірант,*
*Вінницький національний технічний університет*

## МОРФОЛОГІЧНИЙ АНТИАЛІАЙЗИНГ ДЛЯ ГЕКСАГОНАЛЬНОГО РАСТРУ

MLAA – Morphological Anti-Aliasing – морфологічний антиаліайзинг – один з методів постобробки зображення. MLAA призначений для згладжування зображень, що відображаються на екрані, без додаткових вибірок [1]. Метод складається з трьох основних етапів: а) знаходження межі між ділянками контурного розділення в зображенні; б) визначення відповідності геометрії знайдених меж стандартним зразкам U, Z, L; в) змішування кольорів в околі цих знайдених меж.

**Алгоритм морфологічного антиаліайзингу**

На першому кроці алгоритму, знаходять всі лінії, що розділяють чорні пікселі на одній стороні і білі пікселі на іншій, зберігаючи найдовші з цих знайдених ліній [2]. Це досягається шляхом порівняння двох сусідніх рядків і двох сусідніх стовпців (рис. 1).

На другому кроці алгоритму, знаходять інші розділювальні лінії, ортогональні до поточної в крайній вершині [2]. Більшість розділювальних лінії матиме дві ортогональні лінії, включно з лініями, що виходять по краях зображення.

Це спостереження дозволяє класифікувати розділові лінії на наступні три категорії: 1. Z-подібної кривої, яка виникає, якщо дві ортогональних лінії перетинаються обидві напівплощині сформовані з розділювальної лінії. Одним із прикладів є фігура B5R, {CDE}, E4R на рис. 1.; 2. U-подібної кривої, для якої обидві ортогональні лінії розміщуються на тій же стороні, наприклад c2r, {de} 2b, E2R.; 3. L-подібної кривої, яка утворюється, тільки якщо ортогональна лінія бере початок на зображенні границі, наприклад {AB} 5t, B5R.