Міністерство освіти і культури України
Чернівецький національний університет ім. Юрія Федьковича

Лабораторна робота № 3
Назва: **Визначення надійності програмного забезпечення на етапі проектування.**

Мета роботи : Оволодіти знаннями про визначення надійності програмного забезпечення на етапі проектування.

Завдання:

1. Вивчити математичну модель прогнозування надійності програмного забезпечення на етапі проектування на основі моделі Холстеда.
2. Отримати практичні навички розрахунку проектної надійності програмного забезпечення.

Порядок виконання роботи

1. Вивчіть основні моделі оцінки проектної надійності програмного забезпечення.
2. Розрахуйте показники надійності відповідно до завдання до лабораторної роботи.
3. Дайте відповідь на контрольні питання.
4. Складіть звіт в електронному варіанті, який повинен утримувати титульний лист, мета лабораторної роботи, отримані результати та висновки.

Виконав студент 543 групи
Дем’янов Р.І.

2.Розрахунок показників надійності:

– n_1 – кількість різних операторів, які у програмі;

– n_2 – кількість різних операндів, що використовуються в програмі;
– N_1 – кількість усіх операторів, які у програмі ;
– N_2 – кількість усіх операндів, які у програмі;

Код застосунку для автоматичного збору ID сторінок в соціальній мережі Facebook, мова програмування JS (Frontend), NodeJC(Backend), CSS, HTML, база даних MongoDB.

```
{{> head }}
<body>
        {{> navbar }}
        {{{ body }}}
        {{{> footer }}}
</body>
</html>
<!DOCTYPE html>
<head>
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
        <link rel = "stylesheet" href = "/general.css">
        <link href = "https://fronts.googleapis.com/css2?family=Kaushan+Script&family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
        <title>{{tittle}}</tittle>
</head>

<header class = "header">
        <div class = "conteiner">
                <div class = "header_inner">
                        <div class = "header_logo"><a href = "/">GetId</a></div>
                                <nav class = "nav">

                                        <a class = "nav_link" href = "/getList">Швидкий список</a>
                                        {{#if isAuth}}
                                                <a class = "nav_link" href = "/add">СФормувати список</a>
                                                <a class = "nav_link" href = "/userLists">Мої списки</a>
                                                <a class = "nav_link" href = "/auth/logout">Вийти</a>
                                        {{else}}
```

```
                                    <a class = "nav_link" href = "/auth/login">Увійти</a>
                                {{/if}}

                            </nav>
                        </div>
                    </div>
                </header>

<script src="/addFields.js"></script>
<script src="/deleteFields.js"></script>
<script src="/login.js"></script>
const {Schema, model} = require('mongoose');

const userSchema = new Schema({
    name: {
       type: String,
       required: true,
    },
    password: {
        type: String,
        required: true,
    },
    userData: {
        items: [{
            tittle: {
                type: String,
                required: true,
            },
        }],
    },
    groupToken: {
       type: String,
    }
}0;

    router.get('/logout', async(req, res) =>{
    req.session.destroy( ()=>{
        res.redirect('/auth/login');
    });
});

    router.post('login', loginValidators, async (req, res) =>{
        try {
            const{name, password} = req.body;

            const errors = validationResult(req);
```

```
                if(!errors.isEmpty()){
                        req.flash('regError', errors.array()[0].msg;
                        return res.status(422).rediirect('/auth/login')
                }
                const candidate = await User.findOne({name});
                req.session.user =candidate;
                req.session.isAuthenticated = true;
                req.session.save(error => {
                        if(error){
                                throw error;
                        }
                        res.redirect('/userLists');
                        })

                        } catch (error) {
                                console.log(error);
                                }

                                });
                module.exportst = function(req, res, next){
                        if(!req.session.isAuthenticated){
                                return res.redirect('/auth/login');
                        }

                        next();
                }
router.post('/register', registerValidators, astnc (req, res) =>
                try{
        const{name, passsword} =  req.dody;
        if(!errors.isEmpty()){
                req.flash('regError', errors.array()[0].msg);
                return res.status(422).redirect('/auth/login')
        }
        const hashPassword = await bcrypt.hash(password, 10)
        const user = new User({
                name, password: hashPassword, lists: {items: []}
        });
        await user. save():
        res.redirect('/auth/login);

}catch(error){
        console.log(error);
        }
})
                module.exports = function(req, res, next) {
                        res.locals.isAuth = req.session.isAuthenticated;
                        res.locals.csrf = req.csrfToken();
```

```
                next();
        }

    <form class = "inputGroup" id = "pasteFacebookLink" action ="/add" method ="post">
    <input id =tittle" type ="text" class = "inputField" name = "tittle" placeholder = "Назва списку" required value="{{data.tittle}}">
    <input id =tittle" type ="text" class = "inputField" name = "link" placeholder = "Посилання на пост" required value="{{data.link}}">

    <input type = "hidden" name = "_csrf" value = "{{csrf}}">

    <button class = "submitButton" id = "buttonVk">Сформувати</button>
    </form>

    exports.registerValidators = [
    body('name').custom(async (value, {req}) => {
    try {
    const user = await User.findOne({name: value});
    if(user){
        return Promise.reject('Користувач вже існує');
    }
    })
    .isLenght({min: 2})
    .withMessage('ім'я не може містити менше 2 символів')
    .trim(),

    body('password','Пароль не може бути менше 6 символів')
    .isLength({min: 6, max: 56})
    .isAlphanumeric()
    .trim(),

    body('confirm').custom(async (value,{req}) => {
        if(value !== req.body.password){
            throw new Error('Паролі не збігаються');
            }
                return true;
                })
                .trim(),
                ];
    module.exports ={
        MONGODB_URI: process.env.MONGODB_URI,
        SESSION_SECRET: process.env.SESSION_SECRET,
        }
        if(process.env.NODE_ENV === 'production'){
            module.exports = require('./keys.prod');
            } else{
```

```
                module.exports = require('./keys.dev');
                }

router.get('/',auth, async (req, res) => {
const user = await req.user.populate('userData.items');
const lists = user.userData. items.map(l => ){
        ...l._doc
        }));
res.render('userLists', {
        tittle: 'My lists',
                lists
                });

router.get('/:id/edit', auth, async (req, res) => {
        if(!req.query.allow){
                return res.redirect('/');
                }
router.get('/:id/edit', auth, async (req, res) => {
        if(!req.query.allow){
                return res.redirect('/');
                }
        let list = await req.user.findListId(req.params.id);
                list = list._doc;
        res.render('list-edit', {
                tittle: `Редактувати ${list.tittle}`,
                        list,
                        });
                        });

router.post('edit', auth, async (req, res) => {
        await req.user.update(req.body);
        res.redirect('/userLists');
        })
```

– n_1 – кількість різних операторів, які у програмі= 7

– n_2 – кількість різних операндів, що використовуються в програмі=63
– N_1 – кількість усіх операторів, які у програмі = 100
– N_2 – кількість усіх операндів, які у програмі = 162

Словник $n = n_1+n_2= 7+63=70$
Довжина реалізації $N = N_1 + N_2$.= 100+162=262

Час кодування програми:

$$T = \frac{n_1 N_2 (n_1 \log_2 n_1 + n_2 \log_2 n_2) \log_2 n_1}{2 n_2 S} = \frac{7+162(7*2.8+63)2.8}{2*63*18} = 17.23$$

Обсяг програми:

$$V = (2 + n_2^*)\log(2 + n_2^*) = (2 + 100)6.25 = 13125$$

Вихідна кількість дефектів у програмі $N_И$:

$$N_И = \frac{V_П^2}{lV_У} = \frac{43120}{3829} = 11,26$$

$$V_У = \frac{24^3}{l^2} = \frac{24^3}{1.9^2} = 3829$$

3.Відповіді на контрольні запитання

1)У чому особливості програмного забезпечення як об'єкта аналізу надійності?

Програмне забезпечення має свої особливі критерії вимірювання надійності та довговічності роботи, не притаманні іншим матеріальним об’ктам. (механізмам, конструкціям). Показники надійності є важливими чинниками, які впливають на рішення чи програма буде випущена в експлуатацію, тому постала потреба у різного виду моделях та класах надійності ПЗ. В залежності від моделі надійності діляться на два класи : статичні і динамічні.

Динамічні класи припускають, що в програмі кількість помилок є величиною дискретною динамічні моделі розраховують величину початкових дефектів на етапі відлагодження ПЗ і кількість дефектів, отримується на кінець етапу відлагодження, також динамічні моделі припускають, що інтенсивність відмов є сталим числом або функцією часу відлагодження або ж випадковим числом із заданим законом розподілу.

Моделі, які використовуються для визначення надійності ПЗ, поділяють на два класи: ймовірнісні та детерміністичні

Можна виділити такі класи моделей: моделі, які засновані на висіванні помилок, апроксимації залежностей, інтенсивності відмов, на підставі неоднорідного пуассонового процесу, на підставі марковської структури та на підставі зростання надійності.

Однією з особливостей комп‘ютерних систем і мереж порівняно з іншими, наприклад, механічними , є те, що їх надійність визначає надійність елементів та надійність програмного забезпечення.

2. Зобразіть залежності зміни надійності програмних та апаратних засобів ІС від часу.

Надійність прогамних та апаратних засобів ІС від часу залежить від такого показника ,як ,,Інтенсивність відмов”

Інтенсивність відмов; λ(t) (instantaneous failure rale; failure rale**) -** умовна густина імовірності виникнення відмови об'єкта, яка визначається за умови, що до цього монету відмова не виникла.

Інтенсивність відмов є показником безвідмовності не ремонтопридатних і не відновлюваних об'єктів.

$$\lambda(t) = \frac{n(t)}{N_P \cdot \Delta t}$$

де N_P – кількість працездатних примірників ПЗ;

$n(t)$ – кількість відмов ПЗ за час Δt.

Практично для всіх систем інтенсивність відмов залежить від часу і має характеристику у вигляді “ванни” (рис. 1).

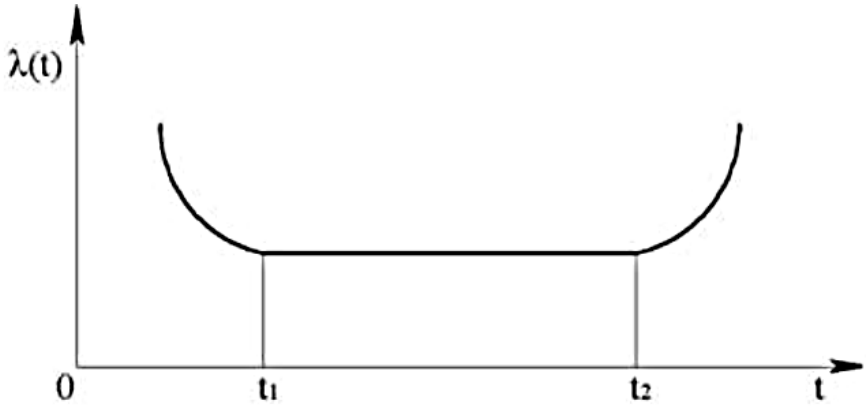

Рис. 2.1. Зміна інтенсивності відмов ПЗ:

0 – t_1 - період припрацювання;

t_1–t_2 - період нормальної експлуатації;

$t > t_2$ – період зношування та старіння.

3. Дайте визначення поняття «надійність програмного забезпечення» відповідно до стандарту ДСТУ ISO/ІЄС 9126-1:2013

Надійність це здатність програмного продукту підтримувати встановлений рівень експлуатаційних характеристик під час використовування за заданих умов.

4, 5 Назвіть основні причини помилок програмного забезпечення та джерела помилок (загрози надійності) програмного забезпечення

1)Людський фактор (експлуатаційні відмови в результаті порушення встановлених правил і (або) умов експлуатації ПЗ)

2)Граничний стан безвідмовності об’єкта, коли об’єкт вичерпав заданий працездатний час функіонування в результаті відмови в процесі наробітку, з лекції №2 - Наробіток до відмови ($t_{сер}$) **(**operating time to failure**)** - наробіток об'єкта від початку експлуатації до виникнення першої відмови.

3)Граничний стан довговічності, коли об’єкт перестає виконувати свої функції в

результаті закінчення технічного ресурсу, сумарний наробіток об'єкта від початку його експлуатації чи поновлення після ремонту до переходу в граничний стан, або терміну служби

4)Відмови які є залежні та незалежні від інших об’єктів.

5)Порушення правил встановлених правил проектування і розробки ПЗ
6)Недосконаліть виготовлення, супроводу або встановлення ПЗ

6.У чому полягають відмінності програмних та апаратних відмов?
Несправності апаратного забезпечення – це переважно фізичні несправності. Помилки програмного забезпечення – це помилки конструкції, які важко візуалізувати, класифікувати, виявити та виправити. Апаратні компоненти зазвичай виходять з ладу через зношення. Програмний компонент виходить з ладу через помилки.