МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО



МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«**ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ**»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТІ 122 – «КОМП'ЮТЕРНІ НАУКИ»
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«КОМП'ЮТЕРНІ НАУКИ»
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

КРЕМЕНЧУК 2020

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Технології створення програмних продуктів» для студентів денної форми навчання зі спеціальності 122 – «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні науки» освітнього ступеня «Бакалавр»

Укладачі: асист. О. П. Слюсаренко,
старш. викл. Т. В. Горлова
Рецензент д. т. н., проф. А. П. Оксанич

Кафедра автоматизації та інформаційних систем

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського
Протокол № ___ від _____
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

Вступ……4
1 Вимоги до оформлення звітів з лабораторних робіт……6
2 Перелік лабораторних робіт……7
Лабораторна робота № 1 Основи життєвого циклу програмного забезпечення (ПЗ)……7
Лабораторна робота № 2 Основи методологій розробки……12
Лабораторна робота № 3 Основи тестування ПЗ……17
Лабораторна робота № 4 Специфікація вимог до ПЗ……20
3 Критерії оцінювання студентів……24
Список літератури……25
Додаток А……26

ВСТУП

Виробництво і використання комп'ютерних програм на сьогодні є масовою діяльністю. У зв'язку з постійно зростаючими обсягами програмних розробок потрібно готувати фахівців, здатних розв'язувати проблеми створення нових програмних продуктів на інженерній підставі, використовуючи накопичений запас знань у галузі програмування та управління системами. Сформовану структуру та зміст підготовки фахівців потрібно розширити методами управління, планування і регулювання робіт, адаптуючи їх до умов колективної розробки програмних систем з гарантованою якістю.

Предметом вивчення навчальної дисципліни «Технологія створення програмних продуктів» є такі основні розділи. Життєвий цикл і стандарти ПП. Поняття програмного забезпечення та проблема розробки складного ПЗ. Життєвий цикл і процеси розробки програмного забезпечення. Міжнародні та національні стандарти розробки складних програмних продуктів. Методології розробки ПЗ (Agile, Scrum, XP, Lean, Kanban). Командна розробка ПЗ. Моделі співпраці. Моделі розробки ПЗ. Тестування ПЗ. Специфікація вимог. Засоби автоматизації розробки ПЗ.

Міждисциплінарні зв'язки

Навчальна дисципліна «Технологія створення програмних продуктів» базується на вивченні навчальних дисциплін «Системний аналіз», «Алгоритмізація та програмування», «Об'єктно-орієнтоване програмування», «Організація баз даних та знань», «Крос-платформне програмування».

Навчальна дисципліна забезпечує засвоєння навчальних курсів «Проектування інформаційних систем», «Технології розподілених систем та паралельних обчислень», а також виконання дипломного проектування.

Метою викладання навчальної дисципліни «Технологія створення програмних продуктів» є одержання компетенцій, достатніх для аналізу вимог до програмних систем, їхнього документування, проектування, розробки, тестування, упровадження, керування програмними проектами та керування якістю розробки програмних систем.

Завданням навчальної дисципліни є набуття та розвиток знань, умінь і навичок для виробничо-технологічної, організаційно-управлінської, проектної та науково-дослідної діяльності.

Згідно з вимогами освітньо-професійної програми студенти повинні

знати:

– етапи життєвого циклу програмного продукту;

– сучасні технології розробки програмного забезпечення та їхні моделі процесу розробки;

– стандарти та методології побудови моделей предметних галузей для використання їх у процесі аналізу вимог до програмного забезпечення;

– методики аналізу вимог і стандарти документування вимог;

– основи процесу розробки програмного забезпечення, орієнтованого на архітектуру;

– основи маркетингу програмного забезпечення;

уміти:

– розробляти специфікацію вимог (SRS);

– розробляти програмні системи зі стійкою архітектурою.

– проводити тестування програмних засобів;

– проводити випробування та супровід програмних продуктів.

1 ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ З ЛАБОРАТОРНИХ РОБІТ

Метою лабораторних робіт є закріплення теоретичних знань лекційного матеріалу, а також розвиток навичок самостійної роботи, уміння робити висновки за результатами проведених досліджень.

Виконавши роботу, студенти оформлюють звіт відповідно до вимог роботи. Під час захисту роботи необхідно відповідати на контрольні питання й уміти пояснювати і коментувати порядок виконання лабораторної роботи.

Звіт про виконання лабораторних робіт (далі – звіт) має бути написаний студентом власноруч, розбірливим почерком, чисто й акуратно, однаковим чорнилом (синім або фіолетовим) чи пастою на аркушах білого паперу форматом А4 (210×297 мм) або комп'ютерним способом (за допомогою комп'ютерної техніки) шрифтом Times New Roman, розміром 14 пунктів на одному боці аркуша. На одній сторінці допускається не більш ніж три виправлення, зроблені охайно та розбірливо (припускається застосування коректора).

Звіт має містити титульну сторінку (додаток А) та звіти про виконання кожної лабораторної роботи.

Звіт про кожну лабораторну роботу має містити:

– тему лабораторної роботи;

– мету роботи;

– порядок виконання роботи – короткий опис виконання кожного пункту роботи;

– письмові відповіді на контрольні питання;

– висновки про виконану роботу.

2 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Основи життєвого циклу програмного забезпечення (ПЗ)

Мета: ознайомлення з принципами розробки програмного забезпечення.

Короткі теоретичні відомості

Життєвий цикл ПЗ

Життєвий цикл ПЗ – це умовна схема, що передбачає окремі етапи, які є стадіями процесу створення ПЗ. При цьому на кожному етапі виконуються різні дії.

Цикл розробки пропонує шаблон, використання якого полегшує проектування, створення та випуск якісного ПЗ. Це методологія, що визначає процеси й засоби, необхідні для успішного завершення проекту.

Мета використання моделі життєвого циклу – створити ефективний, економічно вигідний і якісний програмний продукт.

Процеси під час розробки ПЗ

Аналіз вимог

Життєвий цикл ПЗ починається зі стадії аналізу, під час якого учасники процесу обговорюють вимоги до кінцевого продукту. Мета цієї стадії – визначення детальних вимог до системи. Окрім цього, необхідно переконатися у тому, що всі учасники правильно зрозуміли поставлені завдання та те, як саме кожна вимога буде реалізована на практиці.

Етапи збирання й аналізу вимог

Процес роботи з вимогами до продукту можна розділити на 4 етапи.

1. Визначення концепції продукту.
2. Збір вимог.
3. Аналіз вимог.
4. Проектування системи.

Визначення концепції продукту

На цьому етапі проводиться робота з інвестором продукту, метою якої є вироблення єдиного бачення майбутнього продукту. По закінченню цього етапу проводиться висновок про те, чи буде цей продукт розроблятися чи ні.

Збирання вимог

На цьому етапі основна робота проводиться із замовником системи і її майбутніми користувачами. Мета етапу – точно визначити функції продукту і способи його інтеграції в сучасні процеси.

Якісне виконання робіт на цьому етапі гарантує те, що майбутній продукт відповідатиме очікуванням замовника. Чітка розстановка пріоритетів забезпечує реалізацію найбільш затребуваної функціональності та виключення незатребуваної функціональності, що заощадить бюджет і терміни.

Аналіз вимог

На цьому етапі відбувається структуризація вже зібраних раніше вимог. Мета етапу – надати чіткий перелік недубльованих вимог до системи, які мають бути виділені з надлишкових і частково дубльованих сценаріїв і призначених для користувача історій, які були отримані на попередньому етапі.

Правильно згруповані вимоги допоможуть обійтися мінімальною кількістю функціоналу для задоволення максимально великої кількості цілей, а це допоможе заощадити бюджет і не дасть «розповзтися» рамкам проекту.

Проектування системи

Метою всіх попередніх етапів було збирання інформації про те, кому та навіщо потрібен майбутній проект. Етап проектування – це етап, на якому група розробки приймає проектні рішення про те, які функції виконуватиме продукт, щоб задовольнити користувачів.

Результатом цього етапу є закінчене технічне завдання (ТЗ) до продукту. Воно має містити повний опис поведінки майбутнього продукту та не містити неоднозначностей і питань.

На підставі ТЗ починається моделювання роботи продукту з кінцевими користувачами та проводиться тестування ТЗ. Це дозволяє збільшити якість продукту та знизити його вартість, оскільки вартість внесення змін до ТЗ завжди менша, ніж у кінцевий продукт.

Кожен з чотирьох названих вище етапів проводиться на різних рівнях, а тому має різну ступінь впливу на характеристики майбутнього продукту. Унаслідок цього неможливо лінійно оцінити, який етап важливіший. Але це питання неминуче постане під час планування витрат часу.

Наводиться вплив кожного з етапів роботи з вимогами на характеристики продукту. Залежно від того, які характеристики продукту для нас важливіші, можна визначити, на який етап витратити більше часу, а на який менше.

Проектування

На стадії проектування розробники та системні архітектори, дотримуючись вимогами, розробляють високорівневий дизайн системи.

Різноманітні технічні питання, що виникають у процесі проектування, обговорюються з усіма зацікавленими сторонами, у тому числі із замовником. Визначаються технології, які використовуватимуться у проекті, завантаження команди, обмеження, тимчасові рамки та бюджет. Відповідно до уточнених вимог вибирають проектні рішення.

Метою проектування є визначення внутрішніх властивостей системи та деталізації її зовнішніх властивостей на підставі виданих замовником вимог до ПЗ. Ці вимоги піддаються аналізу.

Проектування ПЗ передбачає такі підставі види діяльності:

– вибір методу та стратегія розв’язання;

– вибір уявлення внутрішніх даних;

– розробка основного алгоритму;

– документування ПЗ;

– тестування та підбір тестів;

– вибір зображення вхідних даних.

Спочатку програма розглядається як чорний ящик. Порядок виконання процесу проектування та його результати залежать не від складу вимог, але й від вибраної моделі процесу та досвіду проектувальника.

Модель предметної галузі накладає обмеження на бізнес-логіку і структуру даних.

Залежно від класу створюваного ПЗ, процес проектування може забезпечуватися як ручним проектуванням, так і різними засобами його автоматизації. У процесі проектування ПЗ для вираження його характеристик використовують різні нотації – блок-схеми, ER-діаграми, UML-діаграми, DFD-діаграми, а також макети.

Проектуванню зазвичай підлягають:

– архітектура ПЗ;

– пристрій компонентів ПЗ;

– призначені для користувача інтерфейси.

Розробка та програмування

Після того, як вимоги та дизайн продукту затверджені, відбувається перехід до наступної стадії життєвого циклу – безпосередньо розробки. Починається написання розробниками коду програми відповідно до раніше визначених вимог.

Системні адміністратори налагоджують програмне оточення, front-end програмісти розробляють призначений для користувача інтерфейс програми і логіку її взаємодії із сервером.

Окрім того, програмісти пишуть Unit-тести для перевірки правильності роботи коду кожного компонента системи, проводять рев'ю написаного коду, створюють білди та розгортають готове ПЗ у програмному середовищі. Цей цикл повторюється, доки всі вимоги не будуть реалізовані.

Програмування передбачає 4 основні стадії:

1) розробка алгоритмів – фактично, створення логіки роботи програми;

2) написання вихідного коду;

3) компіляція;

4) тестування та налагодження (юніт-тестування).

Документація

Цей етап виокремлюють достатньо умовно, оскільки, як видно, ті чи інші документи створюються на усіх стадіях життєвого циклу програми. Проте, окрім проектної документації та супроводжувальних розробки записів, існують також й інші текстові документи, що описують, наприклад, функції програми та способи її використання.

Існує 4 основні типи документації на ПЗ.

1. Архітектурна/проектна – огляд ПЗ, що містить опис робочого середовища та принципів, які мають бути використані для створення ПЗ.

2. Технічна – документація на код, алгоритми, інтерфейси, API.

3. Призначена для користувача – керівництва для кінцевих користувачів.

4. Маркетингова – рекламні матеріали, мета яких – зацікавити людей, звернувши їхню увагу на продукт.

Тестування

Тестування ПЗ – процес дослідження програмного продукту, що має дві різні цілі:

– продемонструвати розробникам і замовникам, що програма відповідає вимогам;

– виявити ситуації, у яких програма працює неправильно, небажано або не відповідає специфікації.

Тестувальники займаються пошуком дефектів у програмному забезпеченні та порівнюють описану у вимогах роботу системи з реальною.

У фазі тестування виявляються пропущені під час розробки баги. У разі виявлення дефекту тестувальник складає звіт про помилку, який передається розробникам. Останні його виправляють, після чого тестування повторюється – але цього разу для того, щоб переконатися, що помилка була виправлена, а саме виправлення не виявилося причиною появи нових дефектів у продукті. Тестування повторюється, доки не будуть досягнуті критерії його закінчення.

Упровадження та супровід

Коли програма протестована і в ній більше не залишилося серйозних дефектів, починається реліз і передавання її кінцевим користувачам. Після випуску нової версії програми в роботу включається відділ технічної підтримки. Його співробітники забезпечують зворотний зв'язок з користувачами, їх консультування та підтримку.

У разі виявлення користувачами тих чи інших пост-релізних багів інформація про них передається у вигляді звітів про помилки команді розробки, яка, залежно від серйозності проблеми, або негайно випускає виправлення (т. зв. hot-fix), або відкладає його до наступної версії програми.

Окрім того, команда технічної підтримки допомагає збирати та систематизувати різні метрики – показники роботи програми у реальних умовах.

Завдання до лабораторної роботи

1. Кожному члену команди визначити свою роль.

2. Відповідно до своєї теми кожна команда проводить збір й аналіз вимог до програмного додатка.

3. Запроектувати високорівневий дизайн додатка.

Контрольні питання

1. Назвіть основні етапи життєвого циклу ПЗ.

2. Що таке документація ПЗ?

3. Що таке тестування ПЗ?

4. Що таке аналіз вимог?

5. На які етапи поділяють проектування ПЗ?

Література: [1].

Лабораторна робота № 2

Тема. Основи методологій розробки

Мета: ознайомлення з методологією SCRAM та її аспектами

Короткі теоретичні відомості

Методологія – це система принципів, а також сукупність ідей, понять, методів, способів і засобів, що визначають стиль розробки програмного забезпечення.

Методологія – це реалізація стандарту. Самі стандарти лише вказують на те, що повинно бути, залишаючи свободу вибору і адаптації.

Конкретні дії реалізуються за допомогою вибраної методології. Саме вона визначає, як виконуватиметься розробка. Існує багато успішних методологій створення програмного забезпечення. Вибір конкретної методології залежить від розміру команди, від специфіки і складності проекту, від стабільності та зрілості процесів у компанії та від особистих якостей співробітників.

Методології – це ядро теорії управління розробкою програмного забезпечення. До наявної класифікації, залежно від використовуваної у ній моделі життєвого циклу, додалася більш загальна класифікація на прогнозовані й адаптивні методології.

Agile – це серія підходів до розробки ПЗ, орієнтованих на використання ітеративної розробки, динамічне формування вимог і забезпечення їх реалізації завдяки постійнії взаємодії всередині самоорганізованих груп, що складаються з фахівців різного профілю.

Існує кілька методик, що належать до класу гнучких методологій розробки: екстремальне програмування (XP), DSDM, Scrum, FDD і т. д.

Основні ідеї:

– люди і взаємодія важливіші за процеси та інструменти;

– працюючий продукт важливіший за вичерпну документацію;

– співпраця із замовником важливіша за узгодження умов контракту;

– готовність до змін важливіша за проходження за попереднім планом.

Принципи:

– задоволення клієнта завдяки ранньому та безперебійному постачанню цінного ПЗ;

– сприйняття змін вимог навіть у кінці розробки;

– часта поставка робочого ПЗ;

– тісне, щоденне спілкування замовника з розробниками протягом усього проекту;

– проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою та довірою;

– рекомендований метод передавання інформації – особиста розмова;

– ПЗ, що працює – кращий вимірювач прогресу;

– спонсори, розробники й користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;

– постійна увага поліпшенню технічної майстерності та зручному дизайну;

– простота – мистецтво не робити зайвої роботи;

– кращі технічні вимоги, дизайн і архітектура – у самоорганізованої команди;

– постійна адаптація до обставин, що змінюються.

SCRUM – методологія гнучкої розробки програмного забезпечення, яка робить акцент на якісному контролі процесу розробки.

SCRUM – це набір принципів, на яких будується процес розробки, що дозволяє у жорстко фіксовані та невеликі за часом ітерації (спринти), надавати кінцевому користувачеві ПЗ, що працює з новими можливостями, для яких визначено найбільший пріоритет. Можливості ПЗ до реалізації у черговому спринті визначаються на початку спринту на етапі планування та не можуть змінюватися протягом його тривалості. При цьому строго фіксована невелика тривалість спринту надає процесу розробки передбачуваність і гнучкість.

Спринт – ітерація у скрам, під час якої створюється функціональне зростання ПЗ. Воно жорстко фіксоване за часом: тривалість одного спринту – від 2 до 4 тижнів. Результатом спринту є готовий продукт, який можна передавати замовнику (система має бути готова до показу замовнику). Для оцінювання обсягу робіт у спринті можна використовувати попередню оцінювання, що вимірюється в Story Points.

У методології SCRUM всього три функції:

1. Scrum Master.

2. Product Owner.

3. Team.

Scrum Master – найважливіша функція у Scrum. Він відповідає за успіх у проекті й по суті він є інтерфейсом між менеджментом і командою. Зазвичай цю функцію у проекті виконує ПМ або Тім Лід. До того ж, що Scrum Master не роздає завдання членам команди, оскільки у гнучких методологіях команда є самоорганізованою і самокерованою.

Product Owner – це людина, яка відповідає за розробку продукту. Зазвичай це Product Manager для продуктової розробки, ПМ для внутрішньої розробки та представник замовника для розробки на замовлення. Product Owner ставить завдання команді, але він не має права ставити завдання конкретному члену команди протягом спринту.

Team (команда) – команда бере на себе зобов'язання щодо виконання обсягу робіт на спринт перед Product Owner. Робота команди оцінюється як робота єдиної групи. Обов'язки команди такі:

– відповідає за оцінювання елементів беклога;

– приймає рішення щодо дизайну та імплементації;

– розробляє софт і надає його замовнику;

– відстежує власний прогрес;

– відповідає за результат перед Product Owner.

Команда у Scrum кроссфункціональна. У її складі – люди з різними навичками – розробники, аналітики, тестувальники.

Артефакти Scrum

Product Backlog – це пріоритезувати список наявних на даний момент бізнес вимог і технічних вимог до системи. Product Backlog постійно переглядається й доповнюється – до нього додаються нові вимоги, видаляються непотрібні, переглядаються пріоритети.

Sprint Backlog – містить функціональність, вибрану Product Owner з Product Backlog. Усі функції розподілені за завданнями, кожна з яких оцінюється командою.

Sprint Planning розділяється на два мітинги. Метою першого є визначення мети спринту та Беклога-спринту – тобто вибір функціональності, яка буде розроблена протягом наступного спринту. Другий мітинг – визначення того, як саме буде розроблятися певна функціональність. Для кожного елемента Спринт-Беклога визначається список завдань і оцінюється їх тривалість.

Daily Meeting – цей мітинг проходить щоранку на початку дня. Він призначений для того, щоб усі члени команди знали, хто і чим займається у проекті. Тривалість не має перевищувати 15 хвилин. Мета мітингу – поділитися інформацією. Він не призначений для розв’язання проблем у проекті (nfrs питання мають бути винесені за межі дейлі мітингу). Дейлі-мітинг проводить Скрам Майстер, який кожному члену команди по черзі задає такі питання:

1. Що зроблено вчора?
2. Що буде зроблено сьогодні?
3. З якими проблемами зіткнувся?

Demo – команда демонструє інкремент продукту, створений за останній спринт. Product Owner, менеджмент, замовники, користувачі водночас його оцінюють. Команда розповідає про поставлені завдання, про те, як вони були розв’язані, які перешкоди були у них на шляху, які були прийняті рішення і які проблеми залишилися нерозв’язаними.

Sprint Retrospective проводиться в кінці спринту. На ретроспективі члени команди висловлюють свою думку про минулий спринт, відповідаючи на два питання:

1. Що було зроблено добре в минулому спринті?
2. Що потрібно поліпшити в наступному?

Завдання до лабораторної роботи

Розробити перелік завдань на весь час розробки програмного забезпечення.

Контрольні питання

1. Що таке методологія?
2. Що таке SCRAM?
3. Що таке Daily Meeting?
4. Що таке Sprint Planning?
5. Що таке Product Backlog?

Література:[5].

Лабораторна робота № 3

Тема. Основи тестування ПЗ

Мета: ознайомлення з основними принципами та підходами тестування програмного забезпечення.

Короткі теоретичні відомості

Тестування ПЗ – це процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний за допомогою спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, вибраних певним чином.

Можна визначити такі основні цілі тестування ПЗ:

– надання інформації про якість ПЗ кінцевому замовнику;

– підвищення якості ПЗ;

– запобігання багів.

Головне завдання тестування – пошук багів.

Помилка – це дія людини, яка породжує неправильний результат.

Баг – недолік компонента або системи, який може привести до відмови певної функціональності.

Під час виконання коду програми, баги, закладені ще під час його написання, можуть проявлятися: програма може не робити того, що повинна або навпаки – робити те, чого не повинна – відбувається збій.

Збій – невідповідність фактичного результату роботи компонента або системи очікуваного результату.

Збій у роботі програми може бути індикатором наявності у ній бага.

Отже, баг існує за умови одночасного виконання трьох умов:

– відомий очікуваний результат;

– відомий фактичний результат;

– фактичний результат відрізняється від очікуваного результату.

Усього існує кілька джерел багів, і, відповідно, збоїв:

– помилки в специфікації, дизайні або реалізації програмної системи;

– помилки використання системи;

– несприятливі умови навколишнього середовища;

– умисне заподіяння шкоди;

– потенційні наслідки попередніх помилок, умов або навмисних дій.

Дефекти можуть виникати на різних рівнях, і від того, чи будуть вони виправлені й коли, буде прямо залежати якість системи.

Якість – ступінь, за якої сукупність притаманних характеристик відповідає вимогам.

Якість ПЗ – це сукупність характеристик ПЗ, що відображають його здатність задовольняти встановлені й передбачувані потреби.

Вимога – потреба або очікування, яке встановлено. Зазвичай передбачається або є обов’язковим.

Умовно можна визначити 5 причин появи багів у програмному коді.

1. Недолік або відсутність спілкування в команді.

2. Складність ПЗ.

3 Зміни вимог.

4. Погано документований код.

5. Засоби розробки ПЗ (також можуть бути погано документовані).

Отже, чим пізніше баг був виявлений, тим дорожче обійдеться його виправлення й тим більше зусиль для цього буде потрібно. Окрім того, баги, закладені в систему на ранніх рівнях проектування, особливо підступні – їх важко відстежити і правильно інтерпретувати. Висновок: чим раніше в

життєвому циклі почнеться тестування, тим більшою мірою ми можемо бути впевнені в її якості.

Коли починати й закінчувати тестування

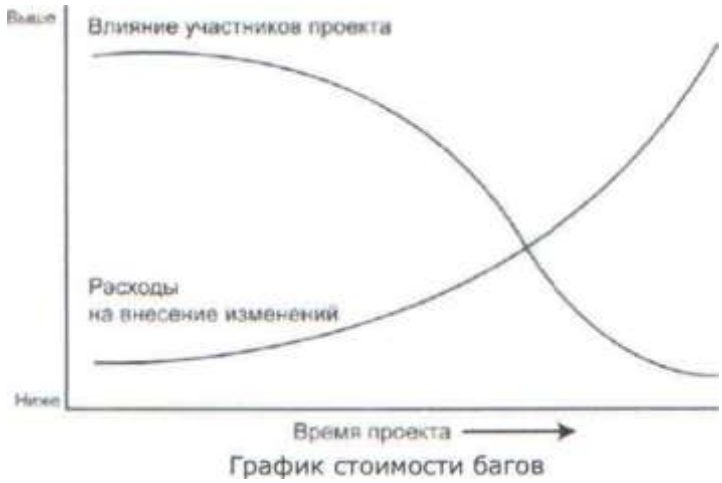

Рисунок 2.1 – залежність вартості виправлення помилок від часу розробки

Важче визначити критерій закінчення тестування, оскільки, згідно з принципами тестування, ніколи не можна бути впевненим у тому, що у програмі 100 % немає багів. Тому тут використовуються інші умови.

1. Граничні терміни, установлені заздалегідь.

2. Виконання всіх передбачених тест-кейсів.

3. Досягнення певного рівня тестового покриття.

4. Коли після певного моменту ми майже не знаходимо нових багів або критичних багів.

5. Рішення менеджменту.

Завдання до лабораторної роботи

Відповідно до таблиці 3.1 описати кожен знайдений баг у проекті та продемонструвати його виправлення.

Таблиця 3.1

Короткий опис	
Проект	
Компонент додатка	
Важливість	
Пріоритет	
Автор	
Оточення	
ОС\Сервис-пак\Браузер + версія	
Опис	
Кроки відтворення	
Фактичний результат	
Очікуваний результат	

Контрольні питання

1. Що таке баг-репорт?
2. Що таке димне тестування?
3. Що таке чек лист?
4. Що таке якість ПЗ?
5. Що таке валідація і верифікація?

Література: [2, 3].

Лабораторна робота № 4

Тема: Специфікація вимог до ПЗ

Мета: набуття навичок написання технічного завдання та описання програмного продукту.

Короткі теоретичні відомості

Специфікація вимог до ПЗ

Технічне завдання (ТЗ) – вихідний документ на проектування технічного об’єкта. ТЗ встановлює основне призначення розроблюваного об’єкта, його

технічні характеристики, показники якості й техніко-економічні вимоги, приписи щодо виконання необхідних стадій створення документації та її склад, а також спеціальні вимоги.

ТЗ є юридичним документом – як додаток міститься в договір між замовником і виконавцем на проведення проектних робіт і є його підставою: визначає порядок та умови робіт, у тому числі мета, завдання, принципи, очікувані результати та терміни виконання. Тобто, повинні бути об'єктивні критерії, за якими можна визначити, чи зроблений той або інший пункт робіт, чи ні.

Усі зміни, доповнення та уточнення формулювань ТЗ обов'язково узгоджуються із замовником та затверджуються ним. Це необхідно тому, що в разі виявлення під час розв'язання проектної задачі неточностей або помилковості вихідних даних, виникає необхідність визначення ступеня вини кожної зі сторін-учасниць розробки, розподілу понесених чере це збитків.

ТЗ як термін у галузі інформаційних технологій – це юридично значущий документ, що містить вичерпну інформацію, необхідну для постановки завдань виконавцям з розробки, упровадження або інтеграції програмного продукту, інформаційної системи, сайту, порталу або іншого IT-сервісу.

– ТЗ ставить завдання, а значить, воно має йти перед прототипом;

– ТЗ має сформулювати бізнес-цілі проекту;

– ТЗ має містити чітко вимірні показники;

– ТЗ має обов'язково узгоджуватися із загальним бізнес-планом замовника;

– кожне внесення правок у готове ТЗ має коштувати грошей.

SRS (Software Requirement Specification) – спеціальна документація для програмного забезпечення, яка містить в собі інформацію про те, як повинна діяти система, які функції виконувати, яке навантаження витримувати і тощо.

Вступ. Мета. У цій секції описується продукт, функціонал якого описуватиметься у SRS.

Вступ. Умовні позначення. Тут описуються всі незрозумілі технічні слова або терміни, які зустрічаються в SRS. До того ж, опис одного незрозумілого терміна не повинен містити інший незрозумілий термін. Намагайтеся розписати термін простою і зрозумілою для всіх мовою, як можна докладніше.

Вступ. Посилання. У цій секції описуються посилання на літературу, у якій можна знайти підстави використаних технологій і фактів. посилання та їх опис мають бути максимально повними, щоб за потреби можна було знайти цей матеріал.

Загальний опис. Функціональність продукту. У цьому розділі описуються частини функціонала на високому рівні. Детальніше кожна частина функціонала буде описана у відповідному розділі нижче. Тут бажано розмістити DFD-діаграму, яка покаже загальну взаємодію системи.

Загальний опис. Середовище функціонування продукту. Тут описується оточення, у якому працюватиме продукт. ОС, версії компіляторів, бази даних, сервера та інше, що необхідно для роботи продукту.

Загальний опис. Обмеження проектування та реалізації. У цьому розділі описуються вибрані стандарти розробки. Наприклад: мова програмування, база даних, стандарти кодування, стандарти обміну даними, обмеження, що накладаються середовищем функціонування або бізнес-логікою.

Загальний опис. Документація для користувачів. Опис списку документації, яку необхідно вивчити користувачеві для роботи з продуктом.

Функціональність системи. Функціональність X. Називаємо функціональність продукту й даємо їй унікальний ідентифікатор.

Функціональність системи. Функціональність X. Опис і пріоритет. Описуємо детально функціональність проекту. Для чого вона? Що повинна робити? Який у неї пріоритет виконання? З цього розділу людині, що читає SRS, має бути зрозуміло, навіщо ця функціональність існує в системі.

Функціональність системи. Функціональність X. Причинно-наслідкові зв’язки. Тригер запуску функціональності. Коли вона запускається і як діє під час запуску?

Функціональність системи. Функціональність X. Функціональні вимоги. Детальний опис функціональності. Описуємо все: як працює, як реагує на помилки, що потрібно перевіряти, як відображати дані і т. д.

Вимоги до зовнішніх інтерфейсів. Опис того, як система взаємодіятиме із зовнішнім світом. Які API, інтерфейси і т. д. будуть з нею взаємодіяти?

Нефункціональні вимоги. Тут слід назвати вимоги, які неможливо описати як функціональність. Наприклад, вимоги до безпеки.

Нефункціональні вимоги. Вимоги до продуктивності. Це не функціональність, проте вони накладають певні обмеження. Наприклад, база даних проекту повинна витримувати 1000 запитів за секунду і т. д.

Нефункціональні вимоги. Критерії якості ПЗ. Опис вимог до якості коду. Які тести використовувати? Які використовувати метрики для визначення якості коду? Скільки коду має бути покрито тестами?

Додаток A. Словник. Цей додаток актуальний, коли виникає необхідність у проясненні будь-яких концепцій.

Додаток B. Моделі процесів і ПЗ. Цей розділ визначає, які діаграми потрібно використовувати для написання SRS.

Appendix C: Список ключових завдань. Цей розділ використовується для дуже формальних SRS. Описує пункти To Be Done – що в майбутньому має бути зроблено, але не описано в SRS.

Завдання до лабораторної роботи

Написати повну специфікацію розробленого програмного продукту відповідно до вимог.

Контрольні питання

1. Що таке загальний опис продукту?
2. Що таке функціональні вимоги?
3. Що таке нефункціональні вимоги?
4. Що таке функціональність системи?
5. Що таке технічне завдання?

3 КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

Перелік лабораторних робіт складається із чотирьох робіт, які спрямовані на перевірку знань студентів з теорії та практики розв’язання прикладних задач.

Виконання лабораторних робіт і написання звітів є обов’язковим для всіх студентів. За чотири роботи студент може отримати 35 балів.

За кожну лабораторну роботу виставляють:

– 7 балів, якщо студент виконав лабораторну роботу, написав звіт і захистив його, тобто правильно та вичерпно відповів на всі питання. Допускаються незначні неточності під час відповіді на одне з контрольних питань;

– 4 бали виставляють, якщо студент виконав лабораторну роботу, написав звіт, але не захистив його;

– у решті випадків виставляють 0 балів.

СПИСОК ЛІТЕРАТУРИ

1. Зиглер К. Проектирование программного обеспечения. Москва: Мир, 1985. 328 с.

2. Майерс Г. Надежность программного обеспечения. Москва: Мир, 1980. 359 с.

3. Чиж С. М. Надежность программной продукции. Санкт-Петербург: Питер, 2001. 128 с.

4. Sandi Metz Practical Object-Oriented Design, Second Edition / Sandi Metz: Pearson Education, 2019. 254 с.

5. Грин Д. Постигая Agile / Д. Грин: «Манн, Иванов и Фербер», 2015. 1272 с.

Додаток А

Зразок оформлення титульної сторінки звіту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ
ІЗ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«ТЕХНОЛОГІЇ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ»

Виконав студент групи _____
ПІБ студента
Перевірив посада, ПІБ викладача

КРЕМЕНЧУК 20__

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Технології створення програмних продуктів» для студентів денної форми навчання зі спеціальності 122 – «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні науки» освітнього ступеня «Бакалавр»

Укладачі: асист. О. П. Слюсаренко,
старш. викл. Т. В. Горлова
Відповідальний за випуск к. т. н., доц. І. Г. Оксанич

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. _____. Наклад _____ прим. Зам. № _____. Безкоштовно.

Редакційно-видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600