1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Збирання та аналіз вимог

Аналіз вимог полягає в визначенні потреб та умов які висуваються щодо нового, чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників.

Аналіз вимог є критичним для успішної розробки проекту. Вимоги мають бути задокументованими, вимірними, тестовними, пов'язаними з бізнес-потребами, і описаними з рівнем деталізації достатнім для конструювання системи.

1.1.1 Класифікація вимог

Вимоги споживача представляють собою вирази фактів та припущень які описують очікування до системи в термінах цілей, середовища, обмежень, та міри ефективності й придатності.

Архітектурні вимоги пояснюють що має бути зроблено ідентифікацією необхідної системної архітектури.

Структурні вимоги пояснюють що має бути зроблено ідентифікацією необхідної структури системи.

Поведінкові вимоги пояснюють що має бути зроблено ідентифікацією необхідної поведінки системи.

Функціональні вимоги пояснюють що має бути зроблено ідентифікацією необхідної задачі, дії, чи діяльності які мають виконуватись. Аналіз функціональних вимог буде використаний в функціях верхніх рівнів для функціонального аналізу.

Нефункціональні вимоги задають критерій для оцінки операцій системи, замість її поведінки.

Вимоги продуктивності пояснюють, до якої міри місії чи функції повинні бути виконані; зазвичай вимірюється в термінах кількості, якості, охопленні, своєчасності чи готовності. Протягом аналізу вимог, вимоги продуктивності будуть інтерактнивно розроблятись вздовж всіх виявлених функції що базуються на факторах життєвого циклу системи, і характеризуються в термінах ступеня визначеності в їх оцінках, ступеня критичності успіху системи, і їх відношення до інших вимог.

Вимоги дизайну.

Успадковані вимоги, тобто вимоги які маються наувазі вимогами вищого рівня, чи перетворені з них.

Розподілені вимоги визначені поділом, чи іншим перерозміщенням високорівневих вимог в кілька низькорівневих вимог.

1.2 Документування вимог

Документ, який описує вимоги, є результатом етапів виявлення та аналізу вимог. Документ опису вимог розроблюється відповідно до раніше визначеного шаблону. Шаблон визначає структуру та стиль документу.

Частина документу опису вимог, яка містить **стислий опис проекту**, переважно орієнтує тих керівників та учасників проекту, які відповідні за прийняття рішення, але ймовірно не стануть детально вивчати документ повністю. На початку документу необхідно визначити цілі та межі проекту, а потім описати діловий контекст системи.

Також слід визначити учасників проекту системи. При цьому важливо, щоб замовник виступав не як безвиразно представлений підрозділ або офіс, необхідно привести конкретні імена.

Хоча документ опису вимог може бути далеким від технічних рішень, важливо визначити ідеї, які стосуються рішення, на початкових етапах життєвого циклу розробки. Важливо також проаналізувати варіант придбання готового продукту замість його розроблення «з нуля». Документ опису вимог повинен надавати перелік існуючих програмних компонентів та пакетів, які необхідно в подальшому вивчити в якості варіантів можливих рішень.

Основна частина документу опису вимог присвячена визначенню **системних сервісів**. Ця частина може займати до половини всього обсягу документу. Це єдина частина документу, яка може містити узагальнені моделі – моделі бізнес-вимог.

Межі системи можна моделювати за допомогою діаграм контексту. У поясненнях до діаграми контексту повинні бути чітко визначені межі системи. Без подібного визначення проект не може бути застрахованим від спроб «розтягнути» його межі.

Функціональні вимоги можна моделювати за допомогою діаграм бізнес-прецедентів. Однак, діаграми охоплюють перелік функціональних вимог тільки в загальному вигляді. Усі вимоги треба позначити, класифікувати та визначити.

Вимоги до даних можна моделювати за допомогою діаграми бізнес-класів. Так само як і у випадку функціональних вимог, діаграма бізнес-класів не дає повного визначення структур даних для бізнес-процесів. Кожний бізнес-клас вимагає подальших пояснень. Необхідно описати атрибут ненаповнення класів та визначити ідентифікуючі атрибути класів. У протилежному випадку неможливо правильно представити асоціації.

Системні сервіси визначають, що повинна робити система. Системні обмеження визначають, на скільки система обмежена під час виконання обслуговування. Системні обмеження пов'язані з наступними видами вимог:

- вимоги до інтерфейсу;
- вимоги до продуктивності;
- вимоги до безпеки;
- експлуатаційні вимоги;
- політичні та юридичні вимоги.

Вимоги до інтерфейсу визначають, як система взаємодіє з користувачем. У документі опису вимог визначаються тільки «відчуття» від GUI-інтерфейсу. Початкове проектування GUI-інтерфейсу виконується під час специфікації вимог та пізніше під час системного проектування.

У залежності від галузі застосування вимоги до продуктивності можуть грати доволі важливу роль в успіху проекту. В обмеженому розумінні вони задають швидкість (час відгуку системи), з якою повинні виконуватися різноманітні завдання. У широкому розумінні вимоги до продуктивності включають інші обмеження: щодо надійності, готовності, пропускної здатності тощо.

Вимоги до безпеки описують користувацькі права доступу до інформації, що контролюються системою. Користувачам може бути наданий обмежений доступ до даних або обмежені права на виконання деяких операцій з даними.

Експлуатаційні вимоги визначають програмно-технічне середовище, якщо воно відоме на етапі проектування, у якому повинна функціонувати система. Ці вимоги можуть впливати на інші сторони проекту, такі як: підготовка користувачів та супроводження системи.

Важливі й інші види обмежень. Наприклад, у відношенні деяких систем можуть висуватися вимоги щодо легкості їх використання (ви-

моги щодо придатності їх використання) або легкості їх супроводження (вимоги щодо придатності до супроводження).

1.3 Розроблення технічного завдання

Технічне завдання (ТЗ) – вихідний документ для розробки автоматизованої системи або створення програмного продукту, відповідно до якого проводиться виготовлення, приймання при введенні в дію та експлуатація відповідного об'єкта. ТЗ є основним документом, що визначає вимоги і порядок створення (розвитку або модернізації) інформаційної системи.

1.3.1 Етапи складання списку вимог ТЗ

Робота над ТЗ включає виконання низки етапів, а невизначеність, властива цій роботі, викликає проходження їх по кілька разів, ітераційно, від більш загальної постановки завдання до детального опрацювання (проектування носить ітераційний характер і те, що не враховано на початку, може бути враховано на наступних етапах).

Основні етапи розробки ТЗ:

– аналіз завдання заказника;
– конкретизація цілей проектування;
– обробка зібраної інформації:
    1) узагальнення та абстрагування;
    2) перевірка на суперечливість;
    3) розмежування вимог на умови, обмеження та показники якості;
    4) параметризація;
    5) зменшення списку вимог;
    6) зведення вимог та затвердження замовником.

1.4 Проектування інтерфейсу веб-застосунків

Інтерфейс (від англ. Interface – поверхня розділу, перегородка) – сукупність засобів і методів взаємодії між елементами системи. Залежно від контексту, поняття застосовне як до окремого елементу (інтерфейс елементу), так і до зв'язків елементів (інтерфейс сполучення елементів).

Інтерфейс користувача – сукупність засобів, за допомогою яких користувач спілкується з різними пристроями:

інтерфейс командного рядка – інструкції програмою або пристроєм здійснюється шляхом введення з клавіатури текстових рядків;

графічний інтерфейс – управління програмними функціями реалізовано графічними елементами екрану.

1.5 Розгортання локального серверу

Веб-сервер – це сервер, який приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, і видає їм HTTP-відповіді разом із затребуваними даними: HTML-сторінкою, зображенням, файлом, медіапотоком і т.д.

На серпень 2011 року найбільш поширеним веб-сервером, що займає більше 65% ринку, є вільний веб-сервер Apache.

Ядро Apache включає в себе основні функціональні можливості, такі як обробка конфігураційних файлів, протокол HTTP і система завантаження модулів. Apache HTTP Server підтримує модульність.

У модулях реалізуються такі можливості, як:

– підтримка мов програмування;

– додавання функціоналу;

– виправлення помилок або модифікація основних функцій;

– посилення безпеки.

LAMP – акронім, що позначає набір (комплекс) серверного програмного забезпечення, який широко використовують в Інтернеті. LAMP названий за першими літерами продуктів, що входять до його складу:

– Linux – операційна система Linux;

– Apache – веб-сервер;

– MySQL – СУБД;

– PHP – мова програмування, що використовується для створення веб-додатків (крім PHP можуть матися на увазі інші мови, такі як Perl і Python).

Аналогічно йому існує WAMP:

– Windows – операційна система від компанії Microsoft;

– Apache;

– MySQL;

– PHP.

Хоча спочатку ці програмні продукти не розроблялися спеціально для роботи один з одним, однак така зв'язка стала вельми популярною через свою гнучкості, продуктивності та низької вартості.

Розгортання веб-застосунку, що розробляється засобами фрймворку в більшості випадків складається з таких етапів:

– створення бази даних в доступній СУБД (MySQL найчастіше);

– попередня настройка через систему веб-доступу;

– підключення до бази даних;

– встановлення.

1.6 Реалізація основних сторінок вебзастосунку

При створенні веб-застосунку з використанням PHP використовуються поняття **пакету** та **репозиторію**.

Пакет – власне, та сама бібліотека, яка створюється або використовується в проекті, як залежність.

Репозиторій (Registry) – сховище пакетів PHP, яке називається Packagist. Кожен бажаючий може опублікувати пакет в Packagist, витративши буквально хвилину часу, а інші зможуть його використовувати. Пакети в PHP ніяк не пов'язані з Git і GitHub.

Для вирішення проблем з повторним використанням та управлінням пакетами використовуються менеджери пакетів.

Composer – менеджер пакетів прикладного рівня для мови програмування PHP, що забезпечує стандартний формат для управління залежностями у програмному забезпеченні та необхідними бібліотеками.

Composer працює з командного рядка і встановлює залежності (наприклад, бібліотек) для застосунку. Він також дозволяє користувачам встановлювати PHP пакети, доступні на ресурсі «Packagist», який є його основним сховищем, що містить доступні пакети. Він також реалізує автозавантажувач класів, для встановлених бібліотек і це полегшує використання коду від сторонніх розробників.

Composer використовується як складова частина декількох популярних PHP проектів з відкритим вихідним кодом, серед яких є більшість популярних фреймворків: Symfony, CodeIgniter, CakePHP, FuelPHP та, звісно ж, Laravel й Yii. Серед CMS Composer використовує Drupal починаючи з 8-ої версії.

1.7 Аналіз якості та оцінка вебзастосунків

Тестування програмного забезпечення (Software Testing) – це процес технічного дослідження, який виконується на вимогу замовників, і призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись.

Якість програмного забезпечення – характеристика програмного забезпечення, ступінь відповідності ПЗ до вимог. Частіше за все, використовують визначення ISO 9001, згідно з яким якість – це «ступінь відповідності наявних характеристик вимогам».

Фактори якості – це нефункціональні вимоги до ПЗ, що відносяться до, наприклад, надійності та продуктивності програм.

Деякі з факторів якості:

Зрозумілість. Призначення ПЗ повинно бути зрозумілим з самої програми та документації.

Повнота. Всі необхідні частини програми повинні бути представлені та реалізовані.

Стислість. Відсутність надлишкової інформації та такою, що дублюється. Реалізація принципів DRY.

Можливість портування. Легкість в адаптації програми до інших умов: архітектури, платформі, операційній системі тощо.

Узгодженість. Вся документація та код повинні виконуватися за єдиними угодами, використовувати єдині формати та позначення

Покриття тестуванням.

Зручність використання.

Надійність.

Безпечність.

Модульне тестування тестує мінімальний компонент програми, або модуля. Кожний модуль тестується для перевірки правильності його реалізації.

Інтеграційне тестування виявляє дефекти в інтерфейсах та у взаємодії між компонентами (модулями).

Системне тестування тестує інтегровану систему для перевірки відповідності всім вимогам.

Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог.

Приймальне тестування може проводитись кінцевим користувачем, замовником, або клієнтом для перевірки, чи може продукт бути прийнятий до використання:

– альфа-тестування – це симульоване або реальне операційне тестування потенційними користувачами/замовником або командою тестувальників на боці розробника.

– бета-тестування йде після альфа-тестування. Версії програмного забезпечення, відомі як бета-версії, надаються у користування обмеженій кількості людей поза компанією для того, щоб упевнитись, що програма не містить великої кількості помилок.

2 ПОРЯДОК ВИКОНАННЯ САМОСТІЙНОГО ЗАВДАННЯ

2.1 Обрати тему для подальшого проектування системи.

2.2 Провести документування специфікації вимог до проекту: сформулювати цілі та межі проекту, аналіз функціональних вимог.

2.3 Визначити системні сервіси: межі системи, функціональні вимоги, вимоги до даних.

2.4 Сформулювати системні обмеження: вимоги до інтерфейсу, продуктивності, безпеки та експлуатаційні вимоги.

2.5 Розробити абстрактну модель системи за обраною темою на основі обраного архітектурного патерну (архітектурного шаблону програмного забезпечення).

2.5.1 Обґрунтувати вибір способу побудови архітектури веб-застосунку: на основі предметної області або фреймворку.

2.5.2 Розробити архітектуру вебзастосунку на основі обраного підходу (та обґрунтувати вибір фреймворку/CMS у випадку використання даного підходу).

2.6 Спроєктувати інтерфейс для вебзастосунку, що реалізує систему (5 вайрфреймів, 2 мокапи, 1 прототип).

2.7 Обґрунтувати вибір запропонованих елементів інтерфейсу, ґрунтуючись на вимогах до інтерфейсу та ергономічних показниках інтерфейсу.

2.8 Виконати конфігурацію сервера.

2.8.1 Навести обґрунтування вибору сервера.

2.8.2 Навести обґрунтування вибору хостингу, у випадку, якщо обрано віддалений хостинг.

2.8.3 Якщо обрано підхід на основі фреймворку, інсталювати обраний фреймворк на сервер відповідно до інструкцій для даного фреймворку.

2.9 Розробити основні сторінки вебзастосунку у вигляді структури, що реалізує основні вимоги на основі обраного підходу (від 5 сторінок).

2.10 Розробити чек-листи для тестування верстки та функціонування вебзастосунку.

2.11 Розробити сценарій та відтворити тестування згідно сценарію. Навести результати тестування.

2.12 Провести тестування верстки.

2.13 Занотовувати опис програми.

3 ВИМОГИ ДО ЗМІСТУ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

3.1 Реферат

Реферат призначений для ознайомлення із пояснювальною запискою. Він має бути стислим, інформативним і містити відомості, які дозволяють прийняти рішення про доцільність читання всієї пояснювальної записки. Реферат повинен містити: відомості про обсяг роботи, кількість частин пояснювальної записки, кількість ілюстрацій, таблиць, додатків, кількість джерел згідно з переліком посилань (усі відомості наводять, включаючи дані додатків); текст реферату; перелік ключових слів. Текст реферату повинен відбивати подану у пояснювальній записці інформацію і, як правило, у такій послідовності: об'єкт дослідження або розробки; мета роботи; методи дослідження та апаратура; результати та їх новизна; основні конструктивні, технологічні й техніко-експлуатаційні характеристики та показники; ступінь впровадження; взаємозв'язок з іншими роботами; рекомендації щодо використання результатів роботи; галузь застосування; економічна ефективність; значущість роботи та висновки; прогнозні припущення про розвиток об'єкта дослідження або розроблення. Частини тексту реферату, щодо яких відсутні відомості, випускають. Реферат належить виконувати обсягом не більш як 500 слів, і, бажано, щоб він уміщувався на одній сторінці формату А4. Ключові слова, що є визначальними для розкриття суті роботи, вміщують після тексту реферату. Перелік ключових слів містить від 5 до 15 слів (словосполучень), надрукованих великими літерами в називному відмінку в рядок через коми.

3.2 Зміст

Зміст розташовують безпосередньо після реферату, починаючи з нової сторінки. До змісту включають: перелік умовних позначень, символів, одиниць, скорочень і термінів; передмову; вступ; послідовно перелічені назви всіх розділів, підрозділів, пунктів і підпунктів (якщо вони мають заголовки) суті пояснювальної записки; висновки; рекомендації; перелік посилань; назви додатків і номери сторінок, які міс-

тять початок матеріалу. У змісті можуть бути перелічені номери й назви ілюстрацій та таблиць з зазначенням сторінок, на яких вони вміщені. Зміст складають, якщо пояснювальна записка містить не менш ніж два розділи або один розділ і додаток за загальної кількості сторінок не менше десяти.

3.3 Перелік умовних позначень, символів, одиниць, скорочень і термінів

Усі прийняті у записці малопоширені умовні позначення, символи, одиниці, скорочення і терміни пояснюють у переліку, який вміщують безпосередньо після змісту, починаючи з нової сторінки.

Незалежно від цього за першої появи цих елементів у тексті записки наводять їх розшифровку. Перелік умовних позначень, символів, одиниць, скорочень і термінів повинен розташовуватись стовпцем. Ліворуч в абетковому порядку наводять умовні позначення, символи, одиниці, скорочення і терміни, праворуч - їх детальну розшифровку.

3.4 Вступ

Вступ розташовують на окремій сторінці. У вступі коротко викладають: оцінку сучасного стану проблеми, відмічаючи практично розв’язані задачі, прогалини знань, що існують у даній галузі, провідні фірми та провідних вчених і фахівців даної галузі; світові тенденції розв’язання поставлених задач; актуальність даної роботи та підставу для її виконання; мету роботи та галузь застосування; взаємозв’язок з іншими роботами.

3.5 Основна частина

Основна частина пояснювальної записки складається з розділів, підрозділів, пунктів і підпунктів і містить наступні розділи:

– вимоги до системи;

– розроблення моделей системи;

– розроблення архітектури та структури системи;

– звернення та використання системи;

– тестування та аналіз результатів тестування.

3.5.1 Вимоги до системи

В першому розділі виконується стислий огляд стану проблеми, де зокрема проводиться аналіз існуючих рішень даної проблеми, та збираються вимоги до програмного забезпечення з метою визначення потреб, які має задовольнити розроблення даної системи.

У межах виконання огляду стану проблеми, зокрема, має бути проведено аналіз аналогічного програмного забезпечення та виконано обґрунтування доцільності власної розробки.

Вимоги до програмного забезпечення мають включати наступні пункти:

– стислий опис проекту: цілі та межі проекту, учасники проекту, існуючі рішення проблеми;

– системні сервіси: межі системи, функціональні вимоги, вимоги до даних;

– системні обмеження: вимоги до інтерфейсу, продуктивності, безпеки, експлуатаційні вимоги;

– проектні питання: попередній план-графік виконання основних проектних завдань, попередній бюджет, нерозглянуті питання, які впливають на успіх проекту.

3.5.2 Розроблення архітектури та структури системи

У третьому розділі виконується розроблення архітектури та структури системи, що проектується, а також інтерфейсу веб-застосунку, що реалізує дану систему.

Архітектура та структура системи проектується на обраного архітектурного патерну (MVC, MVP, MVVM) або на основі предметної області. У подальшому архітектура модифікується на основі застосування обраного фреймворку та модулів для розширення функціональності системи або на основі реалізації функціональності за використання підходу на основі предметної області.

Інтерфейс веб-застосунку має враховувати вимоги до інтерфейсу програмного забезпечення на основі документу опису вимог та те-

хнічного завдання та особливості роботи кожної групи користувачів з кожним елементом взаємодії користувача з веб-застосунку.

У даному розділі мають бути наведені наступні схематичні зображення та виконано їх змістовний опис:

– архітектура веб-застосунку на основі обраного підходу;

– інтерфейс веб-застосунку, що реалізує систему;

– структура веб-застосунку на основі використання обраного фреймворку;

– архітектура веб-застосунку за використання модулів;

– структура веб-застосунку за використання модулів.

3.5.3 Звернення та використання системи

Четвертий розділ пояснювальної записки повинен містити відомості про звернення до системи для запуску та опис процесу використання системи.

Тобто розділ має мітити такі підрозділи:

– призначення й умови застосування програми;

– звертання до програми;

– вхідні і вихідні дані;

– використання програми.

У залежності від особливостей документа допускається поєднувати окремі розділи чи вводити нові.

У розділі «Призначення й умови застосування програми» повинні бути зазначені призначення і функції, які виконуються програмою. Окремо слід виділити умови, необхідні для виконання програми розділивши їх на:

– вимоги до апаратної та програмної частин сервера;

– вимоги до апаратної та програмної частин клієнта.

У розділі «Звертання до програми» повинен бути наведений опис процедур виклику програми (способи передачі керування і параметрів даних та ін.).

У розділі «Вхідні та вихідні дані» повинен бути приведений опис організацій початкової і вихідної інформації, що використовується, при необхідності, її кодування.

У розділі «Використання програми» повинно бути приведено результати використання програми згідно із сценарієм описаним у діаграмі послідовностей та опису потоків.

3.5.4 Тестування та аналіз результатів тестування

П’ятий розділ присвячено виконанню тестування та аналізу результатів тестування розробленого веб-застосунку.

Розроблений веб-застосунок має пройти декілька етапів тестування для винесення рішення про його закінченість та готовність до експлуатації.

У даному розділі мають бути наведені наступні наступні матеріали та виконано їх змістовний опис:

– розроблені чек-листи для тестування верстки та функціонування веб-застосунку;

– результати проведеного тестування верстки.

– результати аналізу реалізованості функціональних вимог до ПЗ.

– результати аналізу тестування функціональності веб-застосунку.

– результати аналізу тестування стабільності.

3.6 Висновки

Висновки вміщують безпосередньо після викладення суті записки, починаючи з нової сторінки. У висновках наводять найбільш важливі результати, одержані в розрахунково-графічному завданні, виконують оцінку одержаних результатів роботи з урахуванням світових тенденцій вирішення поставленої задачі; можливі галузі використання результатів роботи; народногосподарську, наукову, соціальну значущість роботи.

У висновках необхідно наголосити на якісних та кількісних показниках отриманих результатів, обґрунтувати достовірність результатів, викласти рекомендації щодо їх використання.

Текст висновків може поділятись на пункти.

3.7 Перелік посилань

Перелік посилань наводять у кінці тексту записки, починаючи з нової сторінки. У відповідних місцях тексту мають бути посилання.

Перелік посилань є складовою частиною роботи і відображує ступінь вивчення даної проблеми автором. Рекомендований обсяг переліку посилань – не менше 25 літературних джерел.

Бібліографічні описи в переліку посилань подають у порядку, за яким вони вперше згадуються в тексті. Порядкові номери описів у переліку є посиланнями в тексті (номерні посилання). За необхідності джерела, на які є посилання тільки в додатку, наводять у окремому переліку посилань, який розташовують у кінці цього додатка.

Бібліографічний опис джерел складають відповідно до чинних стандартів з бібліотечної та видавничої справи. Зокрема, потрібну інформацію можна одержати із таких стандартів: ГОСТ 7.1 – 84 "Библиографическое описание документа. Общие требования и правила составления", ДСТУ 3582-97 "Інформація та документація. Скорочення слів в українській мові в бібліографічному описі. Загальні вимоги та правила", ГОСТ 7.12-93 "Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила".

У якості обов'язкових елементів бібліографічного опису книги виступають: відомості про назву, перелік авторів, місце видання, рік видання, кількість сторінок.

Книга може мати окрім основної назви додаткову. Перед додатковою назвою, що пояснює зміст основної назви, а також перед іншими відомостями, які відносяться до назви (підручник, навчальний посібник та ін.), ставлять двокрапку.

Якщо книга має більше трьох авторів, відомості про авторів (редакторах, укладачах) наводять після даних, що відносяться до області назви. Перед прізвищами авторів, редакторів, укладачів ставиться коса риска. При кількості авторів більше чотирьох після косої риски наводять прізвища та ініціали перших трьох авторів з додаванням скорочення «та ін.».

Найменування місця видання дається повністю у називному відмінку. У скороченому вигляді прийнято вказувати тільки назви наступних міст: Київ (К.).

Перед назвою видавництва ставиться двокрапка, а після нього – кома та рік видання.

На складову частину видання (статтю, главу, розділ та ін.) складають бібліографічний опис, який містить відомості про статтю (главу, частину, розділ) та відомості про видання де вона опублікована.

Перед відомостями про видання ставляться дві косі риски. Замість загальної кількості сторінок вказують початкову та кінцеву сторінки частини видання.

ЛІТЕРАТУРА

1. Wiegers K. Software Requirements. 3rd edition [Text] / K. Wiegers, J. Beatty. – Redmond : Microsoft Press, 2013. – 672 p.

2. Leffingwell D. Managing Software Requirements: A Use Case Approach [Text] / D. Leffingwell, D. Widrig. – Boston : Addison-Wesley Longman Publishing, 1999. – 377 p.

3. Wiegers K. More About Software Requirements: Thorny Issues and Practical Advice [Text] / K. Wiegers. – Redmond : Microsoft Press, 2005. – 224 p.

4. Starck E. Agile Project Management QuickStart Guide: A Simplified Beginners Guide To Agile Project Management [Text] / E. Starck. – Albany : ClydeBank Media LLC, 2017. – 166 p.

5. Schwaber K. Agile software development with Scrum [Text] / K. Schwaber, M. Beedle. – New Jersey : Prentice Hall, 2002. – 176 p.

6. Rothman J. Create Your Successful Agile Project [Text] / J. Rothman. – Raleigh : The Pragmatic Programmers, 2017. – 225 p.

7. Kniberg H. Kanban and Scrum - making the most of both [Text] / H. Kniberg, M. Skarin. – Morrisville : Lulu Press, 2010. – 120 p.

8. Anderson D. Essential Kanban Condensed [Text] / D. Anderson, A. Carmichael. – Seattle : Lean Kanban University Press, 2016. – 92 p.

9. Booch G. The Unified Modeling Language User Guide [Text] / G. Booch, J. Rumbaugh, I. Jacobson. – Boston : Addison-Wesley Professional, 1998. – 391 p.

10. Mitchell W. H. UML, the Unified Modeling Language [Text] / W. H. Mitchell. – Tucson : Mitchell Software Engineering, 2003. – 28 p.

11. Rumpe B. Modeling with UML: Language, Concepts, Methods [Text] / B. Rumpe. – New York : Springer International Publishing, 2016. – 228 p.

12. Rumpe B. Modeling with UML: Language, Concepts, Methods [Text] / B. Rumpe. – New York : Springer International Publishing, 2016. – 228 p.