**Практична робота №6-7. Визначення ризиків та керування ризиками.**

**Мета роботи**. Набуття навичок визначення ризиків та керування ризиками

**Завдання**

1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи (лекція 27).

2. Для власного проекту потрібно визначити:

2.1. можливі ризики;

2.2. спланувати реагування на ризики;

2.3. процедуру моніторингу і контролю ризиків для вашого проекту.

3. Надайте письмові відповіді на контрольні питання.

3. Оформити звіт, де надати у відповідні пояснення щодо обраного рішення та відповіді на контрольні запитання. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва практичної роботи.
* Прізвище, група
* Назва проекту.

**Контрольні запитання**

1. Визначте поняття керування ризиком.
2. Що розуміється під процесом керування конфігурацією ПЗ?
3. Наведіть основні завдання керування конфігурацією.
4. Які дії виконуються в процесі керування версіями ПЗ?
5. Сформулюйте основні завдання обліку й аудиту.

Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді текстових файлів з іменем у форматі **OPI<Номер групи><Номер лекції / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, R - лабораторна]<Прізвище англійською>**. Наприклад, **OPI3112R**buts.doc. Відповіді повинні бути не довгими і змістовними. Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності відповідей-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк відсилки ЛР IПЗ41 – 07.06.2023**

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Важливою частиною роботи менеджера проекту є оцінка ризиків, які можуть вплинути на графік робіт або на якість створюваного програмного *продукту,* і розробка заходів щодо запобігання ризиків. Результати аналізу ризиків повинні бути відображені в плані проекту. Визначення ризиків і розробка заходів щодо зменшення їхнього впливу на хід виконання проекту називається *керуванням ризиками.* Спрощено ризик можна розуміти як імовірність прояву яких-небудь несприятливих обставин, що негативно впливають на реалізацію проекту. Ризики можуть загрожувати проекту в цілому, створюваному програмному продукту або організації-розроблювачеві. Можна виділити три типи ризиків.

1. *Ризики для проекту,* які впливають на графік робіт або ресурси, необхідні для виконання проекту.
2. *Ризики для розроблювального продукту, що* впливають на якість або продуктивність розроблювального програмного продукту.
3. *Бізнес-Ризики, що* відносяться до організації-розроблювача або постачальнику.

Звичайно, ці типи ризиків можуть перетинатися. Наприклад, якщо досвідчений програміст залишає проект, це буде ризиком для проекту (оскільки затримується строк здачі готового продукту), ризиком для продукту (тому що новий програміст, що замінив збіглого, може виявитися не занадто досвідченим і зробити помилки в програмі) і бізнес-ризиком (оскільки затримка даного проекту може негативно вплинути на майбутні ділові контакти між замовником і організацією-розроблювачем).

Конкретні типи ризиків, які можуть вплинути на даний проект, залежать від виду створюваного програмного продукту й від організаційного оточення, де реалізується програмний проект. Разом з тим багато типів ризиків здатні вплинути на будь-які програмні проекти, ці ризики наведені в табл. 1.

Таблиця 1. Можливі ризики програмних проектів

| Ризик | Тип ризику | Опис ризику |
| --- | --- | --- |
| Плинність розроблювачів | Ризик для проекту | Досвідчені розроблювачі залишають проект до його завершення |
| Зміна в керуванні організацією | Ризик для проекту | Організація змінює свої пріоритети в керуванні проектом |
| Неготовність апаратних засобів | Ризик для проекту | Апаратні засоби, які необхідні для проекту, не надійшли вчасно або не готові до експлуатації |
| Зміна вимог | Ризик для проекту й для розроблювального продукту | Поява великої кількості непередбачених змін у вимогах, пропонованих до розроблювального ПЗ |
| Затримка в розробці специфікації | Ризик для проекту й для розроблювального продукту | Специфікації основних інтерфейсів підсистем не *надійшли* до розроблювачів відповідно до графіка робіт |
| Недооцінка розміру розроблювальної системи | Ризик для проекту й для розроблювального продукту | Розмір системи значно перевищив первісну оцінку |
| Недостатня ефективність CASE-засобів | Ризик для проекту й для розроблювального продукту | CASE-засоби, призначені для підтримки проекту, виявилися менш ефективними, ніж очікувалося |
| Зміни в технології розробки ПЗ | Ризик для розроблювального продукту | Основні технології побудови програмної системи заміняються новими |
| Поява конкуруючого програмного продукту | Бізнес-Ризик | На ринку програмних продуктів до закінчення проекту з'явилася конкуруюча програмна система |

Схема процесу керування ризиками складається із чотирьох стадій.

*Визначення ризиків.* Визначаються можливі ризики для проекту, для розроблювального продукту й бізнес ризики.

*Аналіз ризиків.* Оцінюється ймовірність і послідовність появи ризикових ситуацій.

*Планування ризиків.* Плануються заходи щодо запобігання ризиків або мінімізації їхнього впливу на проект.

*Моніторинг ризиків.* Постійне оцінювання ймовірностей ризиків і виконання заходів щодо зм'якшення наслідків прояву ризикової ситуацій.

Процес керування ризиками, як і інші процеси планування, є ітераційним, виконуваним протягом усього строку реалізації проекту. Спочатку розробляються плани керування ризиками, потім постійно відслідковується ситуація навколо реалізації проекту. При надходженні нової інформації про можливі ризики заново проводиться аналіз ризиків і першорядна увага приділяється новим ризикам. У міру надходження нової інформації також змінюються плани заходів щодо запобігання й пом'якшення ризиків.

Результати процесу керування ризиками документуються у вигляді планів керування ризиками. Вони повинні включати опис можливих проектних ризиків, їхній аналіз і перелік заходів, необхідних для керування ризиками.

### Визначення ризиків

Визначення ризиків - перша стадія процесу керування ризиками. На цій стадії описуються ризики, які можуть виявитися при реалізації проекту. У принципі на цій стадії не повинна оцінюватися ймовірність і значимість ризиків, але на практиці малоймовірні ризики з незначними наслідками зазвичай відкидаються відразу.

Визначення ризиків може виконуватися в режимі командної роботи з використанням підходу "мозковий штурм" або ґрунтуватися на досвіді менеджера. При визначенні ризиків може допомогти наведений нижче список можливих категорій ризиків.

1. *Технологічні ризики.* Виникають із програмних і апаратних технологій, на основі яких розробляється система.

*2. Ризики, пов'язані з персоналом.* Пов'язані зі членами команди розроблювачів.

*3. Організаційні ризики.* Виникають із організаційного оточення, у якому виконується проект.

*4. Інструментальні ризики.* Пов'язані з використовуваними CASE-засобами й іншими засобами підтримки процесу створення ПЗ.

5. *Ризики, пов'язані із системними вимогами.* Проявляються при зміні вимог, пред’явлених  до розроблювальної системи.

6. *Ризики оцінювання.* Зв'язані оцінюванням характеристик програмної системи й ресурсів, необхідних для реалізації проекту.

У табл. 2 представлені деякі приклади, що ставляться до кожної з описаних категорій ризиків. Результатом етапу визначення ризиків буде довгий список можливих ризиків, які можуть вплинути на розроблювальний програмний продукт, проект або організацію-розроблювача.

Таблиця 2. Категорії ризиків

|  |  |
| --- | --- |
| Категорія ризиків | Приклади  ризиків |
| Технологічні ризики | База даних, що використовується в програмній системі, не забезпечує обробку очікуваного обсягу транзакцій.  Програмні компоненти, які використовуються повторно, мають дефекти, що обмежують їхні функціональні можливості |
| Ризики, пов'язані з персоналом | Неможливо підібрати працівників з необхідним професійним рівнем.   Провідний розроблювач занедужав у самий критичний час.  Неможливо організувати необхідне навчання персоналу |
| Організаційні ризики | В організації, що виконує розробку ПЗ, відбулася реорганізація, у результаті чого змінилися пріоритети в керуванні проектом.  Фінансові труднощі в організації привели до зменшення бюджету проекту |
| Інструментальні ризики | Програмний код, що генерується CASE-засобами, не ефективний.  CASE-Засобу неможливо інтегрувати з іншими засобами підтримки проекту |
| Ризики, пов'язані із системними вимогами | Зміни вимог приводять до значних повторних робіт із проектування системи.  Первісне нечітке формулювання користувальницьких вимог привело до значних змін системних вимог, що виявилися на пізніх стадіях розробки проекту |

**Аналіз ризиків**

При аналізі для кожного певного ризику підраховується ймовірність його прояву й збиток, що він може нанести. Не існує простих методів виконання аналізу ризиків - значною мірою він заснований на думці й досвіді менеджера. Можна привести наступну шкалу ймовірностей ризиків і їхніх наслідків.

Імовірність ризику вважається дуже низкою, якщо вона має значення менш 10%; низкою, якщо її значення від 10 до 25 %; середньою при значеннях від 25 до 50%; високою, якщо значення коливається від 50 до 75%; дуже високою при значеннях більше 75%.

Можливий збиток від ризиків ситуацій можна підрозділити на катастрофічний, серйозний, терпимий і незначний.

Результати аналізу ризиків повинні бути представлені у вигляді таблиці ризиків, упорядкованих по ступені можливого збитку. У табл. 3 наведений упорядкований список ризиків, описаних у табл.2; там же зазначені ймовірності цих ризиків. Тут імовірності ризиків і ступінь збитку від них зазначені довільно. На практиці для їхнього визначення необхідна докладна інформація про проект, технологію створення ПЗ, команді розроблювачів і про саму організацію.

Таблиця 3. Список ризиків після проведення їхнього аналізу

|  |  |  |
| --- | --- | --- |
| Ризик | Імовірність | Ступінь збитку |
| Фінансові труднощі в організації привели до зменшення бюджету проекту | Низька | Катастрофічна |
| Неможливо підібрати працівників із професійним рівнем, що вимагається | Висока | Катастрофічна |
| Провідний розроблювач занедужав у самий критичний час | Середня | Серйозна |
| Програмні компоненти, використовувані повторно, мають дефекти, що обмежують їхні функціональні можливості | Середня | Серйозна |
| Зміни вимог приводять до значних повторних робіт із проектування системи | Середня | Серйозна |
| В організації, що виконує розробку ПЗ, відбулася реорганізація, у результаті чого змінилися пріоритети в керуванні проектом | Висока | Серйозна |
| База даних, що використовується в програмній системі, не забезпечує обробку очікуваного обсягу транзакцій | Середня | Серйозна |
| Недооцінки часу виконання проекту | Висока | Серйозна |
| CASE-засоби неможливо інтегрувати з іншими засобами підтримки проекту | Висока | Терпима |
| Первісне нечітке формулювання користувальницьких вимог привело до значних змін системних вимог, що виявилися на пізніх стадіях розробки проекту | Середня | Терпима |
| Неможливо організувати необхідне навчання персоналу | Середня | Терпима |
| Швидкість виявлення дефектів у системі нижче раніше запланованої | Середня | Терпима |
| Розмір системи значно перевищує спочатку розрахований | Висока | Терпима |
| Програмний код, що згенерований CASE-засобами, неефективний | Середня | Незначна |

Звичайно, як імовірність ризиків, так і можливий збиток від них повинні переглядатися при надходженні додаткової інформації про ці ризики й у міру реалізації заходів щодо керування ними. Тому подібні таблиці ризиків повинні перероблятися на кожній ітерації процесу керування ризиками.

Після проведення аналізу ризиків визначаються найбільш значимі ризики, які потім відслідковуються протягом усього строку виконання проекту. Визначення цих значимих ризиків залежить від їхніх ймовірностей і можливого збитку. У загальному випадку завжди відслідковуються ризики з катастрофічними наслідками, а також ризики із серйозним збитком, значення ймовірності яких вище за середнє.

Зазвичай рекомендується визначити й відслідковувати "10 верхніх" ризиків. Кількість ризиків, які необхідно відслідковувати, залежить від конкретного проекту. Це може бути п'ять ризиків, а може - п'ятнадцять. Але, звісно, кількість ризиків, по яких проводиться моніторинг, повинне бути доступним для огляду. Велика кількість ризиків, що відслідковуються, зажадає величезної кількості збираємої інформації. Зі списку ризиків, представлених у табл. 6, для моніторингу варто відібрати всі вісім ризиків, які можуть привести до катастрофічних і серйозних наслідків.

Вимоги до програмної системи часто класифікуються як функціональні, нефункціональні й вимоги предметної області.

1. *Функціональні вимоги.* Це перелік сервісів, які повинна виконувати система, причому повинне бути зазначене, як система реагує на ті або інші вхідні дані, як вона поводиться в певних ситуаціях тощо. У деяких випадках вказується, що система не повинна робити.

2. *Нефункціональні вимоги.* Описують характеристики системи і її оточення, а не поводження системи. Тут також може бути наведений перелік обмежень, що накладаються на дії й функції, виконувані системою. Вони включають тимчасові обмеження, обмеження на процес розробки системи, стандарти тощо.

3. *Вимоги предметної області.* Характеризують ту предметну область, де буде експлуатуватися система. Ці вимоги можуть бути функціональними й нефункціональними.

У дійсності чіткої границі між цими типами вимог не існує. Наприклад, користувальницькі вимоги, що стосуються безпеки системи, можна віднести до нефункціональних. Однак при більш детальному розгляді така вимога можна віднести до функціональних, оскільки воно породжує необхідність включення в систему засобу авторизації користувача. Тому, розглядаючи далі ці види вимог, ми повинні завжди пам'ятати, що дана класифікація в значній мірі штучна.

### Функціональні вимоги

Ці вимоги описують поводження системи й сервіси (функції), які вона виконує, і залежать від типу розроблювальної системи й від потреб користувачів. Якщо функціональні вимоги оформлені як користувальницькі, вони, як правило, описують системи в узагальненому виді. На противагу цьому функціональні вимоги, оформлені як системні, описують систему максимально докладно, включаючи її вхідного й вихідного дані, виключення тощо.

Функціональні вимоги для програмних систем можуть бути описані різними способами. Розглянемо для приклади функціональні вимоги до бібліотечної системи університету, призначеної для замовлення книг і документів з інших бібліотек.

1. Користувач повинен мати можливість проводити пошук необхідних йому книг і документів або по всій множині доступних каталожних баз даних або по певній їхній підмножині.

2. Система повинна надавати користувачеві підходящий засіб перегляду бібліотечних документів.

3. Кожне замовлення повинно бути постачений унікальним ідентифікатором, що копіюється у формуляр користувача для постійного зберігання.

Ці функціональні користувальницькі вимоги визначають властивості, якими повинна володіти система. Вони взяті з документа, що містить користувальницькі вимоги, і показують, що функціональні вимоги можуть бути описані з різним рівнем деталізації (порівняєте першу й третю вимоги).

Багато проблем, що виникають при розробці систем, пов'язані з неточністю й "розмитістю" специфікації вимог. Природно, розроблювачі інтерпретують вимоги, що допускають двояке тлумачення, так, щоб систему було простіше реалізувати. Але це тлумачення може не збігатися з очікуваннями замовника. Така ситуація приводить до розробки нових вимог і внесенню змін у систему. Це, у свою чергу, веде до затримки здачі готової системи і її подорожчанню.

Наприклад, до бібліотечної системи, може надаватися вимога "підходящий засіб перегляду документів". Бібліотечна система може надавати документи в широкому спектрі форматів. У вимозі мається на увазі, що система повинна надати спосіб для перегляду документів у будь-якому форматі. Але оскільки ця умова чітко не виписана, розроблювачі у випадку дефіциту часу можуть використовувати простий засіб для перегляду текстових документів і наполягати на тому, що саме таке рішення треба з даної вимоги.

Специфікація функціональних вимог повинна бути комплексною й несуперечливою. Комплексність має на увазі опис (визначення) всіх системних сервісів. Несуперечність означає відсутність несумісних і взаємовиключних визначень сервісів. На практиці для великих і складних систем украй важко розробити комплексну й несуперечливу специфікацію функціональних вимог. Причина криється частково в складності самої розроблювальної системи, а частково у неузгоджених точках зору на те, що повинна робити система. Ця непогодженість може не виявитися на етапі первісного формулювання вимог - для її виявлення необхідний більше глибокий аналіз специфікації. Коли непогодженість системних функцій виявиться на якому-небудь етапі життєвого циклу програми, у системну специфікацію прийде внести відповідні зміни.

### Нефункціональні вимоги

Нефункціональні вимоги не позв'язані безпосередньо з функціями, виконуваними системою. Вони пов'язані з такими інтеграційними властивостями системи, як надійність, час відповіді або розмір системи. Крім того, нефункціональні вимоги можуть визначати обмеження на систему, наприклад на пропускну здатність пристроїв введення-виведення, або формати даних, використовуваних у системному інтерфейсі.

Багато нефункціональних вимог відносяться до системи в цілому, а не до окремих її засобів. Це означає, що вони більш значимі й критичні, ніж окремі функціональні вимоги. Помилка, допущена у функціональній вимозі, може знизити якість системи, помилка в нефункціональних вимогах може зробити систему непрацездатною.

Разом з тим нефункціональні вимоги можуть ставитися не тільки до самої програмної системи: одні можуть ставитися до технологічного процесу створення ПЗ, інші - містити перелік стандартів якості, що накладаються на процес розробки. Крім того, у специфікації нефункціональних вимог може бути зазначено, що проектування системи повинне виконуватися тільки певними САSЕ-засобами, і наведено опис процесу проектування, якому необхідно слідувати.

Нефункціональні вимоги відображають користувальницькі потреби; при цьому вони ґрунтуються на бюджетних обмеженнях, ураховують організаційні можливості компанії-розроблювача й можливість взаємодії розроблювальної системи з іншими програмними й обчислювальними системами, а також такі зовнішні фактори, як правила техніки безпеки, законодавство про захист інтелектуальної власності тощо.

Всі нефункціональні вимоги можна розбити на три великі групи.

1. *Вимоги до продукту.* Описують експлуатаційні властивості програмного продукту. Це вимоги до продуктивності системи, обсягу необхідної пам'яті, надійності (визначає частоту можливих збоїв у системі), переносимості системи на різні комп'ютерні платформи й зручності експлуатації.

2. *Організаційні вимоги.* Відображають політику й організаційні процедури замовника й розроблювача ПЗ. Вони включають стандарти розробки програмного продукту, вимоги до реалізації ПЗ (тобто до мови програмування й методам проектування), вихідні вимоги, які визначають строки виготовлення програмного продукту, і супровідну документацію.

3. *Зовнішні вимоги.* Враховують фактори, зовнішні стосовно розроблювальної системи й процесу її розробки. Вони включають вимоги, що визначають взаємодію даної системи з іншими системами, юридичні вимоги, проходження яким гарантує, що система буде розроблятися й функціонувати в рамках існуючого законодавства, а також етичні вимоги. Останні повинні гарантувати, що система буде прийнятна для користувачів або замовника.

Далі наведений приклад вимог до продукту, організаційних і зовнішніх вимог. Вимоги до продукту позв'язані із середовищем програмування APSE для мови Ada. Це обмежує свободу проектувальника системи у виборі символів - можна використовувати тільки символи з користувальницького інтерфейсу APSE. Організаційні вимоги вказують, що система повинна розроблятися відповідно до внутрішнього стандарту компанії на розробку ПЗ, що має код ХYZсо-SP-STAN-95. Зовнішні вимоги випливають із необхідності дотримання законодавства про збереження конфіденційності. Внаслідок цього системні оператори не будуть мати доступ до тих даним, які їм не потрібні для роботи із системою.

### Приклад нефункціональних вимог

### Вимоги до продукту

Всі взаємодії між інтерфейсом АР5Е и користувачем здійснюються на основі стандартної безлічі символів мови Ada.

Організаційні вимоги

Розробка системи й створення супутньої документації виконуються на основі стандарту ХYZCo-SP-STAN-95.

Зовнішні вимоги

Система не повинна розкривати конфіденційної інформації про замовника системи, крім його ім'я, а також телефонного номера системних операторів

Основна проблема нефункціональних вимог полягає в тому, що їхнє виконання важко перевірити. Часто вони пишуться для того, щоб відобразити загальні цілі замовника системи, такі, як простота експлуатації, можливість відновлення після збоїв або швидка відповідь на запити користувача. Реалізація подібних вимог може виявитися складної для системних розроблювачів, оскільки вони нечітко сформульовані й відкривають простір для різних тлумачень. Одним з основних показників (цілей) системи зазначена простота експлуатації, що у вигляді нефункціональних вимог можна виразити різними способами. У цьому випадку вимога сформульована так, що його можна перевірити.

### Системні цілі й перевірка вимог

### Системна мета

Система повинна бути простою в експлуатації для досвідченого оператора й зводити кількість його помилок до мінімуму.

Нефункціональна вимога, що перевіряється

Досвідченому операторові повинні бути доступні всі системні функції після двох годин навчання роботи з даною системою. Після такого навчання середнє число помилок оператора не повинне перевищувати двох за робочий день.

В ідеалі нефункціональні вимоги повинні виражатися через кількісні показники, які можна об'єктивно виміряти. У табл. 4 наведені показники, за допомогою яких можна специфікувати нефункціональні системні властивості.

На практиці виразити нефункціональні вимоги за допомогою кількісних показників досить важко. Часто замовник ПО не може оформити своє бачення майбутньої системи за допомогою вимог, виражених кількісними показниками. Або деякі системні вимоги, наприклад зручність супроводу, взагалі не можна виразити через кількісні показники. Крім того, витрати на об'єктивний вимір кількісних нефункціональних вимог можуть виявитися вкрай високими. Тому часто документ, що специфікує вимоги до системи, містить опис системних цілей разом із чітко сформульованими вимогами. Ці системні цілі корисні, оскільки відбивають подання (і пріоритети) замовника про майбутню систему. Разом з тим замовник повинен розуміти, що його системні цілі можуть трактуватися різними способами і їх неможливо об'єктивно проконтролювати.

Таблиця 4. Кількісні показники для нефункціональних вимог

| Показник | Одиниці виміру |
| --- | --- |
| Швидкість | Кількість виконаних транзакцій в секунду; |
|  | час реакції на дії користувача; |
|  | час відновлення екрана |
| Розмір | Кілобайти; |
|  | кількість модулів пам'яті |
| Простота експлуатації | Час навчання персоналу; |
|  | кількість статей у довідковій системі |
| Надійність | Середня тривалість часу між двома послідовними проявами помилок у системі; |
|  | імовірність виходу системи з ладу; |
|  | коефіцієнт готовності системи |
| Стійкість до збоїв | Час відновлення системи після збою; |
|  | відсоток подій, що приводять до збоїв; |
|  | імовірність псування даних при збоях |
| Переносимість | Відсоток машинно-залежних операторів; |
|  | кількість машинно-залежних підсистем |

Нефункціональні вимоги часто вступають у конфлікт із іншими вимогами, пропонованими системі. Наприклад, відповідно до одним із системних вимог розмір системи не повинен перевищувати 4 Мбайт, оскільки вона повинна повністю поміститися в постійний запам'ятовувальний пристрій обмеженої ємкості. Інша вимога зобов'язує використовувати для написання системи мова програмування Ada, що часто застосовується для створення критичних систем реального часу. Але, допустимо, відкомпільована системна програма, написана мовою Ada, займає більше 4 Мбайт. Отже, одночасне виконання цих вимог неможливо. У цій ситуації варто відмовитися від одного з вимог. Можна або застосувати інша мова програмування, або збільшити обсяг пам'яті, виділюваний для системи.

У принципі функціональні й нефункціональні вимоги в документі, що описує вимоги до системи, повинні бути рознесені по різних розділах. Але на практиці ця умова виконати непросто. Якщо нефункціональні вимоги помістити окремо від функціональних, буде важко простежити взаємозв'язки між ними. Якщо всі вимоги зібрані в одному списку, складно провести аналіз функціональних і нефункціональних вимог окремо й визначити вимоги, що ставляться до системи в цілому. Вид подання вимог в одному документі також істотно залежить від типу спецификуємої системи. Але в кожному разі повинні бути виділені вимоги, що описують інтеграційні властивості системи. Для цього їх можна помістити в окремий розділ або який-небудь інший спосіб відокремити від інших вимог.

*Для самостійного вивчення*: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

*Література*

1. Лавріщева К.М. Програмна інженерія. Електронний підручник. URL: <http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf>.

2. Соммервил И. Инженерия программного обеспечения / И.Соммервил.–М. : Издательский дом «Вильямс», 2002. –623 с.

3. *Боэм Б.У.* Инженерное проектирование программного обеспечения. – М.: Радио и связь, 1985. – 511 с.