Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Методи захисту інформації в рамках предметно-орієнтованого проєктування інформаційних систем

Виконав(ла): студент(ка) VI курсу, групи СБм-61
спеціальності 125 «Кібербезпека»
(шифр і назва спеціальності)

(підпис) Мельничук А. Б. (прізвище та ініціали)

Керівник (підпис) Марценюк В.П. (прізвище та ініціали)

Нормоконтроль (підпис) Кареліна О. В. (прізвище та ініціали)

Завідувач кафедри (підпис) Загородна Н. В. (прізвище та ініціали)

Рецензент (підпис) Дуда О. М. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпека
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н. В.
(підпис) (прізвище та ініціали)
«___» _____ 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 125 кібербезпека
(шифр і назва спеціальності)

студенту Мельничуку Андрію Богдановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи захисту інформації в рамках предметно-орієнтованого проєктування інформаційних систем

Керівник роботи Марценюк Василь Петрович д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 8 » листопада 2021 року № 4/7-941

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Дослідження різних методів захисту інформації у інформаційних системах, що використовують предметно-орієнтований підхід

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметно-орієнтованого проєктування та проблеми безпеки у ньому.
2. Захист інформації в рамках предметно-орієнтованого проєктування.
3. Реалізація методів захисту та порівняння складності
4. Охорона праці та безпека у надзвичайних ситуаціях. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Методи захисту інформації в рамках предметно-орієнтованого проєктування інформаційних систем. 2. Актуальність. 3. Визначення мети. 4. Завдання. 5. Предметно-орієнтоване проєктування. 6. Основні концепції. 7. Відокремлений контекст безпеки. 8. Проблема захисту даних. 9. Вон Вернон | Форум. 10. Легенда. 11. Метод 1. Захист поруч із предметною областю. 12. Метод 2. Захист у місці взаємодії користувача з системою. 13. Метод 2. Недоліки. 14. Метод 3. Фасад. 15. Метод 4. Захист за допомогою аспектів. 16. Складність. Рівень Залежностей. 17. Цикломатична складність. 18. Дякую

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г. М.		
Безпека в надзвичайних ситуаціях	Клепчик В. М.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Отримав завдання	20.09.2021	виконано
2.	Аналіз предметно-орієнтованого проєктування та проблеми безпеки у ньому	24.09.2021	виконано
3.	Дослідження захисту інформації в рамках предметно-орієнтованого проєктування	10.10.2021	виконано
4.	Реалізація методів захисту та порівняння складності	30.10.2021	виконано
5.	Охорона праці та безпека у надзвичайних ситуаціях	24.11.2021	виконано
6.	Підготовка пояснювальної підписки	06.12.2021	виконано
7.	Підготовка презентації та доповіді	08.12.2021	виконано
8.	Попередній захист	10.12.2021	виконано
9.	Норм контроль, рецензування	18.12.2021	виконано
10.	Занесення диплома в електронний архів	20.12.2021	виконано
11.	Допуск до захисту у зав. кафедри	22.12.2021	виконано
12.	Захист кваліфікаційної роботи	23.12.2021	виконано

Студент _____ Мельничук А. Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценюк В.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

«Методи захисту інформації в рамках предметно-орієнтованого проєктування інформаційних систем» // Дипломна робота ОР «Магістр» // Мельничук Андрій Богданович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2021 // С. 61, табл. – 3, рис. – 10, додат. – 1.

Ключові слова: Аспектно-орієнтоване програмування, Предметно-орієнтоване проєктування, Матриця Структури Залежностей, Цикломатична складність, Захист інформації, Методи Захисту, Керування доступом на основі ролей.

Предметно-орієнтоване проектування — це підхід до проектування програмного забезпечення. Метод проєктування визначає у собі практики як спілкуватись із спеціалістами, самої, предметної області та набір правил проектування де кінцевий код буде відображати у собі всі поняття самого домену.

У деяких випадках розділення проблем, передбачених предметно орієнтованим підходом, важко досягти, а саме, коли розглядається функціональність, яка не залежить від домену, але є тісно пов’язаною з функціональністю, пов’язаною з предметною областю. Дані проблеми стосуються блоку програми яка б відповідала за безпеку.

На жаль, засновник досліджуваного підходу не визначив як саме можна поєднати таку логіку, тому було досліджено різні методи забезпечення захисту даних та аналіз найкращих варіантів реалізації, які можна застосовувати у реальних проектах.

ANNOTATION

"Information protection methods within domain-driven design of the information systems" // Diploma paper of Master degree level "// Melnychuk Andrew Bogdanovich // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Cybersecurity Department // Ternopil, 2021 // p. 61, table. - 3, fig. - 10, appendix. - 1.

Key words: Aspect Oriented Programming, Domain Driven Design, Dependency Structure Matrix, Cyclomatic complexity, Protection of Information, Protection Methods, Role Based Access Control.

Domain Driven Design — is an approach to software design. The approach method defines the practices of how to communicate with specialists, of the subject area and a set of design rules where the final code will reflect all the concepts of the domain itself.

In some cases, the separation covered by a subject-oriented approach is difficult to achieve, namely when considering functionality that is independent of the domain but is closely related to functionality related to the subject area. These issues are related to the security unit.

Unfortunately, the founder of the researched approach did not define exactly how such logic can be combined. That is why were researched the different methods of data protection and analysis of the best implementation options that can be used in real projects.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AOP	Аспектно-орієнтоване програмування
DDD	Предметно-орієнтоване проєктування
gRPC	Google виклик віддалених процедур
HTTP	Протокол передачі гіпертексту
RBAC	Управління доступом на основі ролей
SOA	Сервіс-орієнтована архітектура
UI	Інтерфейс користувача

ВСТУП

Актуальність дослідження. Розробка додатків – це процес, який стає дедалі складнішим залежно від складності програми, що розробляється. Для управління та контролю складності цього процесу розробки було визначено методи та методології розробки. Ці методології та прийоми часто базуються на прикладній області що мало б полегшувати розуміння, впровадження та обслуговування програми, що розробляється. Серед багатьох існуючих методологій проектування можна виділити доволі популярний метод який називається проектно орієнтований підхід. Даний підхід пропонує нам розробляти програму в основі якій будуть використовуватись поняття сутності та поняття прикладної області.

Предметно-орієнтований підхід походить з уявлення, що в основі програмного забезпечення лежить код який тільки рішає проблеми самої предметної області. Даний підхід являє собою рекомендації щодо того як правильно поділити програму яка б складається з двох основних частин, а саме прикладна моделі та модель реалізації. Це поділ має на меті перемістити фокус у програмі на самі поняття та логіку предметної області. Також часто логіку основного ядра програми у даному підході ще називають бізнес-логікою.

Проблематика при використані даного підході є в тому, що таку високу інкапсуляцію важко досягти. А саме, коли існує така функціональність, яка не є частиною бізнес логіки, проте є сильно звязаною на функціональності, пов'язаною з предметною областю.

Яскравим прикладом такої проблеми стосуються безпеки потоку даних. Перевірка прав на читання або виконання дій передбачених в предметній області є концептуально незалежною від домену, тому даний модуль безпеки повинен моделюватись як окремий сервіс.

Проте такий модуль дуже важко відокремити від основної функціональності, тому що він тісно пов'язаний із ядром нашого програмного продукту.

Звідси і виникає проблема яка є мало обговорюваною. Даний підхід у собі не визначає як забезпечити безпеку потоку даних у програмному продукті. Отже, під час поділу безпеки на послугу виникають дві основні проблеми.

Перша проблема сервісу безпеки полягає у тому що вона повинна авторизувати всі дії, які впливають на реалізацію класів домену, а отже, вона сильно завязана з кодом предметної області.

Друга проблема при використані предметно-орієнтованому підході, полягає в тому що така функціональність як авторизація та аутентифікація повинна бути та сама логіка наших бізнес процесів — повинна знаходитись у різних обмежених контекстах.

Метою дипломної роботи є дослідження того, як можна вирішити проблеми, що стосуються моделювання та реалізації функціональних можливостей сервісу безпеки в рамках предметно-орієнтованого проектування інформаційних систем, а також знайти найкращі варіанти дизайну програмної системи, щоб відокремити функціональні можливості безпеки від функцій, пов’язаних із предметною областю.

Найкращі варіанти дизайну перевіряються шляхом створення прототипу. Реалізація прототипу повинні відповідати принципам та водночас забезпечуючи зручність використання та ефективність під час розробки програмного продукту.

Оцінка варіантів дизайну виконується шляхом перегляду аналізу можливих реалізацій, які демонструють можливості наданню функціональних можливостей аутентифікації та авторизації службі безпеки.

Об'єктом дослідження магістерської роботи є підхід предметно-орієнтованого проєктування.

Предметом дослідження є проблема забезпечення безпеки даних при проектуванні інформаційної системи.

Методи дослідження. Перед тим як використовувати той чи інший метод кожен з них був проаналізований на відповідність до підходів, які нам диктує обрана тема а також порівняння фактичної складності написання програмного продукту.

Наукова новизна отриманих результатів:

1. Запропоновані різні нові методи проєктування програми, де існує логіка відповідає за безпеку даних при предметно-орієнтованому підході;
2. Створено різні реалізації тестових прикладів які б відображали плюси та мінуси кожного з розроблених підходів;
3. Проаналізовано та описано переваги та недоліки кожного із підходів.

Підхід до досягнення цілей дипломної роботи складається з чотирьох етапів: дослідження, проектування, впровадження та оцінка.

Щоб прояснити, як предметно орієнтований підхід застосовується на практиці, ми дослідили термінологію, внутрішню роботу та переваги.

Знання, отримані під час дослідження, використовуються для опису проблемної області для вивчення конкретного випадку. Цей приклад був використаний для порівняння різних варіантів дизайну.

Дане дослідження показує, як підхід працює в практичному середовищі.

На етапі проектування були виведені різні можливі варіанти проектування для вирішення проблем безпеки, виявлених у тематичному дослідженні. Кожен метод відображає у собі основну концептуальну відмінністю у проектуванні відносно інших підходів. Тобто вона описує, як взаємодіють пов’язані з доменом і незалежні від домену функції. Варіант дизайну має певні переваги та проблеми через різні способи взаємодії між доменом і незалежною від домену функціональністю в дизайні.

Було проаналізовано різні рекомендації на просторах інтернет мережі та обрано найбільш раціональні та правильні підходи які б відповідали підходу предметного проєктування.

Після даного етапу оцінки та дослідження було обрано найкращі методи захисту інформації. Кожен варіант було порівняно щодо відповідності та коректності впровадження методу захисту в інформаційних системах що використовують предметно орієнтований підхід. Вимірювання для визначення складності та чіткості досвіду проектування та впровадження також враховуються при оцінці. Етап оцінки також містить порівняння між різними методами шляхом вимірювання зв’язку між службою безпеки та реалізацією домену.

Оскільки кожна реалізація вирішує проблеми по-різному, оцінка може призвести до того, що різні методи захисту інформації будуть рекомендовані в різних ситуаціях і середовищах.

Практичне значення даної роботи полягає у тому, що розроблені різні можливі варіанти рішення проблеми безпеки при предметно-орієнтованому підході можна використовувати при проектуванні великих систем.

Апробація результатів роботи. Робота доповідалися на IX науково-технічній конференції «Інформаційні моделі, системи та технології», Тернопіль, ТНТУ, 8 - 9 грудня 2021 р.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ ТА ПРОБЛЕМИ БЕЗПЕКИ У НЬОМУ

1.1. Предметно-орієнтоване проєктування

Розробка застосунків – процес, що з кожним роком робиться складнішим, адже з все дальше щоб зайти на ринок програмного забезпечення ваш продукт повинен бути більш надійним а часто складнішим в порівнянні з конкурентами. Серед різних методів, запроваджених для управління складністю процесу розробки додатків, є метод який полягає в тому що ми відокремлюємо предметну область як ядро нашого проекту. Даний підхід називається предметно орієнтованим.

Щоб ефективніше працювати зі змінами вимог, складність програми має бути керованою. Виділяють суттєві, наступні труднощі [2].

Хоча суттєву складність не можна зменшити, оскільки вона притаманна проблемі, що вирішується, істотною складністю можна керувати за допомогою вибору методу дизайну програмного продукту та способи її реалізації. Для управління складністю прикладної моделі у методології предметного проєктування використовує принцип поділу проблем. Цей принцип використовується для визначення понять прикладної області, що складається з різних класів та методів, які незалежних від середовища, де програма виконується. Дана проблема рішається створюючи різні служби, та репозиторії, що дозволяють абстрагуватись від реалізації та зосередитись на бізнес процесах. Під час написання коду служби помічаються англійським словом “Service” і його не варто плутати зі визначенням даного терміну в інших архітектурах, таких як сервісно-орієнтована архітектура (SOA). В даному підході сервіси напряму взаємодіють із предметною областю [24].

Розділення пов’язаної з доменом і незалежної від домену функціональності в засноване на ідеї застосування розділення проблем до

розробки програмного забезпечення, яка бере свій початок з початку 1970-х років [3].

За своєю суттю даний підхід можна охарактеризувати, як специфічний стиль використання, усім відомого, об'єктно орієнтованого підходу. Принцип поділу обов'язків також пов'язаний з основою об'єктно-орієнтованого проєктування. Наприкінці 1970-х років його було застосовано, вважаючи, що «все є об’єктом», таким чином пов’язуючи властивості та поведінку з окремими класами [4].

На етапі проектування необхідна співпраця між розробниками та експертами в області. Щоб переконатися, що експерти галузі та спеціалісти з програмного забезпечення спілкуються в системі безпеки однаковою термінологією, команда визначає всюдисущу мову. Всюдисуща мова в даній методології означає, що всі члени команди, спеціалісти з програмного забезпечення та спеціалісти з предметної області можуть використовувати спільну мову яка б мала спростити взаєморозуміння людей які працюють в різних галузях. За допомогою діаграм, визначенні вимог та ведення спільних термінів. Така мова дає змогу створити чітку та зрозумілу модель програми для всіх членів команди.

У предметно орієнтовані методології створений спеціально для до контролю складності прикладної моделі, все що стосується реалізації програмних рішень, та моделі предметної області. Створення цієї правильної та завершеної моделі є метою даного підходу на етапі проектування. Правильно модель означає, що члени команди погоджуються з переліченою функціональністю, а завершена модель означає, що всі функції, пов’язані з доменом, присутні в моделі домену. Завершена стан не означає, що дана предметної області є реалізована повністю або готовим продуктом після фази проектування. Зміни, внесені рішеннями на етапі розробки, відображаються в моделі предметної області. Це означає, що модель домену залишається центрованою та узгодженою з спеціалістами розробляємого продукту, протягом

усього процесу розробки. Хоча це і не обов’язково, даний підхід ідеально підходить для виконання етапів проектування та впровадження в ітераційний спосіб.

Правильне застосування даного підходу має призвести до того, що комплексні зусилля розробки стануть більш динамічними та більш зосередженими на основних питаннях продукту. Така зосередженість на моделях призведе до того, що прикладна модель або та що реалізує технічні проблеми, буде мати дуже слабкий зв’язок з логікою програмного продукту. Це посилений поділ дає можливість гнучко реалізацію програму, що полегшує додавання нових вимог або функцій. Вищезгадане поділ незалежних від домену функціональних можливостей на сервіси є спробою забезпечити незалежність бізнес логіки від технологій якими реалізується програмне забезпечення. Під словом технологія мається на увазі, наприклад, те як і де ми зберігаємо дані, яку базу даних використовуємо, або різні можливі інтерфейси взаємодії. Особливо такі сервіси, як інтерфейс користувача, чутливі до змін у технології та вимогах.

Визначення поділу між незалежними від домену функціональними можливостями та можливостями предметної області, не є тривіальним завданням. Рішення може бути складним, оскільки багато функцій не вписуються в жодну з категорій. Рішення про те, де функціональність повинна бути розміщена в моделі, ґрунтується на знаннях, досвіді та здоровому глузді. Таким чином, в залежності від того хто приймає рішення, буде впроваджено функціональність перебуватиме в службі чи в домені.

Дизайн, керований предметно орієнтованим підходом, не є ідеєю заради ідеї, а має на меті забезпечити засоби для полегшення обслуговування, прискорення розробки, можливості створення гнучкості та надійності до зміни вимог. Причини цих позитивних ефектів пов'язані з поділом між низькою звязаністю бізнес процесів та процесів які допомагають рішають програмні проблеми, це значно полегшує використання бізнес логіки предметної галузі. Під час розробки зміни вимог мають швидше за все ніяк не вплине на модель

предметної області. Нова функціональність здебільшого міститься в кількох методах, які слід додати до реалізованих класів домену. Оскільки зміни у службах впливають на меншу частину загального процесу розробки, це призводить до зменшення можливості щось зламати та більш швидкої розробки.

Варто також відмітити, що зміни до бізнес-правил також легше вносити, оскільки вони відокремлені в моделі домену. Проте така централізація може внести і недолік, оскільки у випадку кардинальних змін в бізнес процесах спричиняють не актуальність та поламають на всі реалізаційні сервіса. Усі служби, які взаємодіють із зміненим класом домену, мають бути перевірені на правильність роботи. Таким чином, зміни в реалізації домену можуть поширюватися на всі служби і мати великий потенційний вплив.

Ще однією перевагою застосування принципів предметно орієнтованого підходу до проекту — є підвищена можливість тестування реалізації домену. Підвищена можливість тестування досягається завдяки тому, що реалізація домену працює незалежно від служб. Це дає нам можливість дуже легко підмінювати реалізацію баз даних на більш прості методи зберігання об'єктів, наприклад, зберігання прямо у оперативній пам'яті комп'ютера. Отже, даний підхід спрощує тестування методів за допомогою модульних тестів [25].

1.2 Архітектура та структура проекту при DDD проєктуванні

У багатошарова прикладна модель використовується Еріком. У собі вона розділяє програму на певні шари, кожен з яких відповідає конкретно за рішення своєї проблеми. (див. рис. 2.1). Мета цієї багатошарової моделі полягає в тому, щоб зосередити код, пов’язаний з моделлю предметною галуззю, в одному рівні та ізолювати його від інтерфейсу користувача, програми та інфраструктурного коду [1].

Кожен з цих шарів має особливе призначення. На відміну від звичайних багатошарових моделей, де шари можуть взаємодіяти тільки із сусідніми

шарами, шари в даній методології можуть взаємодіяти тільки в одному напрямку без втручання проміжних шарів.

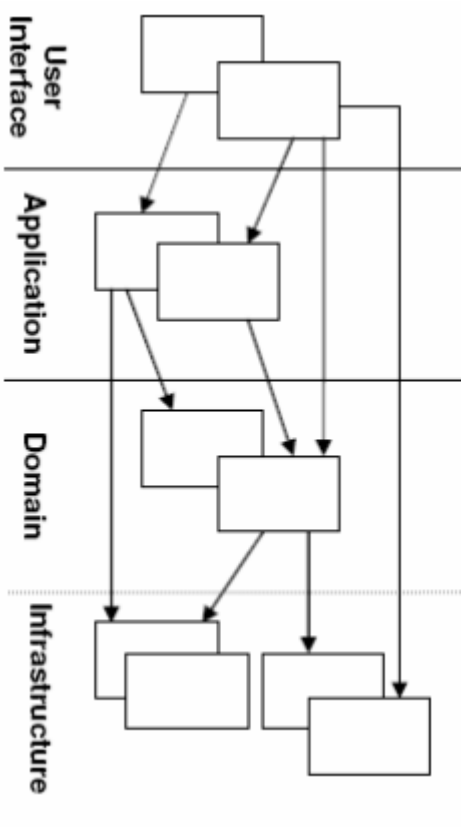

Рисунок 1.1 — Модель шарів у предметно орієнтованому підході запропонований Еріком Евансом

Стрілки показують напрямок взаємодії між різними шарами. Це означає, що компоненти в які розташовуються у шарі домену не знають про існування шарів які знаходяться над ними і відповідно не мають посилань на ці компоненти. Коротко про кожний шар, який зазначений на рис. 1.1.

Рівень інтерфейсу користувача (або презентації) відповідає за показ інформації користувачеві та інтерпретацію команд користувача.

Рівень додатка — це додатковий рівень, який допомагає перекласти дії, щоб використовувати шар домену. Він не містить жодних правил ведення бізнесу чи знань. Цей шар тримається тонким і не має стану.

Рівень домену представляє реалізацію предметної області. Він містить поняття ділової інформації, бізнес-правила та стан поточної ділової ситуації.

Рівень інфраструктури надає загальні технічні можливості, які підтримують вищі рівні. Прикладами є збереження даних та обмін інформацією.

Багатошарова модель не в змозі відобразити більше послуг, ніж показано зараз. Нові послуги будуть розміщені на рівні інфраструктури або інтерфейсу користувача.

Модель предметної області залишається основою архітектури. Реалізація бізнес логіки не вимагає жодного зовнішнього сервісу для роботи, тому реалізація домену є незалежною та може працювати автономно. Така незалежна робота має перевагу, тому що, наприклад, для тестування та запуску домену ми можемо легко підмінити інфраструктурний шар з реальної реалізації, де використовуються різні бази даних та інші зовнішні провайдери, на тестовий шар де наш код буде швидко виконуватись без залучення зовнішніх факторів.

1.3 Аспектно-орієнтоване програмування

Аспектно-орієнтоване програмування (AOP) – це концепція, яка дозволяє ефективно вирішувати наскрізні або ж перехрестні проблеми. Мається на увазі що існує певна функціональність програми, яка впливає на інші функції. Ці проблеми дуже часто неможливо чітко відокремити від інших функцій [5].

Щоб вирішувати наскрізні проблеми, AOP визначає такі поняття ка поради, аспекти, зрізів та точок з’єднання [6].

- Аспекти представляють певну частину функціональності, яка була визначена як наскрізна проблема. Це деякий клас або модуль, які реалізовують наскрізну функціональність. Аспект змінює поведінку іншого коду, застосовуючи поняття порад до точки з'єднання;

- Поради визначають нову логіку яку слід застосувати, в конкретних ситуаціях або ж точках з'єднання, наприклад, вхід до методу чи вихід із методу;
- Точки з’єднання представляють місце у виконувальній програмі, де буде застосовано пораду. Такі місця можуть бути записи методів або оголошення властивостей, у вихідному коді.

Зрізи існують, щоб зв’язати точки з’єднання з порадою, яку слід виконати. Розрізи точок визначають, до яких точок з’єднання слід застосувати конкретну пораду.

Ця структура дозволяє модулювати наскрізну функціональність у аспектах. Така модульність наскрізних проблем покращує якість програмного забезпечення [7].

AOP застосовується до вихідного коду шляхом вплетення аспектів. Мова програмування важлива для при використані даного підходу, оскільки не кожна мова зможе визначити під час компілювання де і як треба внести введені аспекти AOP у вихідний код. Вже існують такі базові бібліотеки яку дають можливість використання даного підходу. Наприклад, AspectJ для Java або PostSharp для .NET [21].

Вибір PostSharp є найкращим виборо реалізації AOP в порівнянні над іншими доступними бібліотеками, у .NET, такими як Aspect.NET і LOOM.NET.

Недоліком PostSharp є те, що зрізи оголошуються в цільовому місці, це означає, що ми змушені мати певну залежність від коду який ніяк не відповідає нашій предметній області. Це є небажана поведінка в предметно орієнтованому підході, оскільки буде створено певно залежність в реалізації головного модуля від аспектів.

1.4 Вимірювання складності реалізації методу

Аналізу складності програми оцінюється за допомогою підрахунку кількості залежностей який використовує той чи інший клас. Для оцінки використовується матриця структури залежностей.

Дана матриця перераховує всі складові підсистеми та шаблони, що відповідають за обмін інформацією, а також залежності і взаємодії. Наприклад, якщо елементи матриці представляють залежності, то така матриця деталізує, які саме фрагменти взаємодіють між собою і показує наскільки той чи інший компонент є сильно завязаним.

Завдяки такому вимірювання можна, доволі легко представляти велику кількість елементів системи та дуже компакто відобразити на скільки кожен елемент залежить від інформації, що створюється кожним з проектів [13].

Чим більше значення зв'язків між об'єктами там міцнішим вважається такий зв'язок. Сильна залежність створює проблеми у розумінні програмної системи а також у підтримці проекту. Особливо це є не вигідною складовою під час розробки, оскільки люба зміна в одному класі може мати великий вплив на інші класи які є у зв'язку. Клас із високим показником складніше підтримувати та тестувати. Більша прив'язаність також забирає можливість використати код для майбутніх проектів [18].

Цикломатична складність (CC) — ще один метод, який застосовано для вимірювання складності програмного забезпечення. Значення вимірюється за допомогою підрахунку кількості можливих шляхів, які можуть пройти через програму у кожному методі [8].

Число варіантів проходження через метод вираховується за допомогою чисельності логічних виразів у вихідному коді, де можливі різні шляхи такі, як if, case, default, catch, тернарні оператори та інші. Також цикли вважаються вдвічі більшими оскільки ці вони не передбачено ускладнюють код і тому вважається що вплив таких ключових слів значно гірший. Тому значення даної

оцінки є кількість шляхів, які могли б бути застосовані у коді та число методів. Наприклад, якщо клас з двома методами, що містить загалом 10 умовних операторів, то його важче зрозуміти, ніж клас з 10 методами, що містять два умовних операторів.

Отже, складність окремого класу буде відповідати кількості можливих варіантів виконання, а також кількість методів. Чим вищий показник, тим більш складним вважається даний клас.

Важливо зважати на значення складності, адже класи з високим показником швидше за все буде важче зрозуміти, а відповідно і складніше написати тест кейси на них, в процесі розробки або підтримки програмного продукту. Тому, що чим більше, для прикладу, перевірок тим вища ймовірність пропустити можливий варіант використання. Такий фактор часто призводить до того, що, у програмному продукті появиться прогалини через, які може початись витік інформації із закритої інформаційної системи. Метод з низьким значенням легше тестувати та розуміти, оскільки через метод менше можливих шляхів, а це означає, що потрібно менше перевіряти різних тестових випадків.

Також дуже ефективно оцінити, можна визначити чи даний метод не є сильно ускладнений можна за допомогою даного методу [14].

Він дає можливість оптимізувати програмну систему та знайти найкращий показник, який базується на основі корекції альтернативних характеристики, які були враховані на попередній ітерації або модифікація існуючої програмної системи під час реінжинірингу . Цей метод запобігає перерахунку для процесу оцінки та відбору. Для корекції характеристик альтернатив використовується метод попарної заміни. Його концепція полягає в компенсації переваги зміни критерію. Багатокритеріальна оптимізація підстановки здійснюється за допомогою нелінійної скалярної згортки, що покращує валідність обраного рішення. Проте, даний методі порівняння буде опущено, адже у цьому досліджені ми привязані тільки до одної архітектури, тому крадинальних змін не зможемо досягти.

1.5 Проблема захисту даних при проектуванні

Реалізації сервісу безпеки як звичайного сервісу в рамках предметно орієнтовангов підходу, будуть страждають від описаних раніше проблем.

Проблема служби безпеки полягає в тому що вона такі поняття як пароль або унікальне ім'я користувача не є частиною проблеми предметної області, а тому не може бути її частиною і такий сервіс має перемістити в інший відокремлений контекст. Проте результатом такого розділення є те, складність поєднати нашу логіку безпеки з логікою, яка відповідає за бізнес процеси.

Наприклад, Вона Вернона пропонує перевіряти права доступу використовуючи самі сутності домену `(forum.IsModeratedBy(moderator))`, проте такий варіант реалізації є тяжкий у реалізації. Оскільки він створить дуже багато перевірок, щоб створити саме таку сутність яка відображає роль користувача, який звертається до системи. [15]

З цих проблем виникає питання, чи доцільно відокремлювати сервіс безпеки та які існують можливості методи рішень даної проблеми. Сильний зв’язок між сервісом і реалізацією проекту суперечить принципам закладними досліджуваного мною підходу. Проте мета даної роботи це знайти найоптимальніший варіант реалізації де б був низький зв’язок між службами та реалізацією бізнес логіки. Регулярні рішення проблеми щільної зв'язності, як-от використання відомих патернів проектування таких як подій або спостерігача, неможливо у випадку, коли модуль безпеки просто моделюється як відокремлений контекст.

Для дослідження сервіс безпеки, не повинен бути дуже складним, а тому його можна звести тільки до аутентифікації та авторизації.

Аутентифікація полягає в підтвердженні того, що користувач є певною особою (або роллю), якій дозволено доступ до функцій програми;

Авторизація полягає в підтвердженні того, що конкретна особа (або роль), яка пройшла аутентифікацію, має дозвіл на доступ до певних функцій програми.

Функція сервісу безпеки полягає в тому, щоб регулювати дії, які виконуються певними користувачами, шляхом реалізації функцій аутентифікації та авторизації. Особливо необхідно перевірити взаємодію з класами бізнес логіки.

В предметно орієнтованому проєктування служби є ізольованими класами предметної області, які не мають ніяких знань про інші класи які рішають інші проблеми нашої інформаційної системи, так як підключення до бази даних, протокол взаємодії користувача з програмою. Виходить, що бізнес логіка не має ніяких посилань на зовнішні служби. Однак ми повинні перевіряти всі зовнішні події, оскільки не можна довіряти на правильність надходження вхідних параметрів.

Щоб виконати автентифікацію, сервіс безпеки повинен отримати вхідні дані та перевірити їх на правильність, наприклад чи такий користувач з іменем та паролем існує у інформаційній системі. Отже, щоразу, коли потрібна авторизація, необхідно перевіряти чи даний користувач має право виконувати певні дії. Є різні варіанти авторизації користувачів проте у будь-якому випадку автентифікація перевіряється при кожній авторизації.

Авторизація є наскрізною проблемою, оскільки кілька методів у різних службах та домені потрібно перевірити на наявність авторизації. Під час авторизації можна виконати два різні види перевірок: перевірити, чи дозволено взаємодієти даному користувачу, за певних умов. Ці умови зазвичай пов’язані зі станом класів предметної області. Якщо взаємодія дозволена, до результатів все одно потрібно буде застосувати деяку фільтрацію на основі прав користувача користувача.

Тип служби безпеки, вказує на те, що питання розділення різних відокремлених контекстів, які по факту є різні але в реальності дуже зав'язані одне на одному спричиняє те що дуже важко відокремити безпеку від реалізації предметної області так.

РОЗДІЛ 2. ЗАХИСТ ІНФОРМАЦІЇ В РАМКАХ ПРЕДМЕТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ

2.1 Предмет дослідження проблеми захисту інформації

Для опису проблем та можливих рішень вибрано “Бухгалтерська аутсорсингова справа”, тому що це випадок, коли кілька рівнів безпеки дуже просто представити, а також вони легко розуміються. У цьому дослідженні ефект від додавання функціональних можливостей безпеки чітко помітний і зрозумілий.

Питання, розвитку ринку програмного забезпечення для бухобліку, є доволі, важливим адже, вони можуть спричинити стрімкий розвиток економічної системи у державі [19].

Питання безпечного програмного забезпечення є дуже важливим. Тому дану тематику було вибрано для того, щоб проілюструвати дослідження концепції поділу безпеки як сервісу. Реалізація прикладу дає можливість виміряти складність та переваги використання запровадженої служби безпеки.

Даний приклад буде описаний багатьма можливими реалізаціями проте реалізований тільки найбільш підходящий з них.

У прикладі бухгалтерської області моделюється, як працює організація яка надає послуги клієнтам і, як клієнти та бухгалтера взаємодіють із даними. Кожен із запропонованих користувачів має свої специфічні можливості та обмеження.

Дана тематика дозволяє продемонструвати велику кількість варіантів використання системи. Кожна роль може виконувати дії з певними обмеженнями, як показано в таблиці 2.1.

Наприклад клієнти можуть виконувати практично усі можливі дії, тобто вони можуть получити або виконати любу операцію але тільки в межах своїх даних.

Бухгалтера або помічники бухгалтера можуть виконувати практично усі дії які стосуються ведення бухгалтерії, проте вони не можуть підписувати документи або додавати електронні ключі.

Головні бухгалтера можуть переглядати та виконувати практично всі дії над усіма клієнтами єдине що вони не можуть це надсилати звіти від імені клієнтів оскільки це заборонено законодавством.

Нижче, у таблиці 2.1 наведено всі можливі варіанти використання та доступ для кожного з користувачів. Важливо відмітити що дана таблиця є чисто побудована для прикладу і може вважатись простою. В реальному світі звісно дій та роль є набагато більше і вони складніші у логіці. Основна ціль даного прикладу є продемонстрування різних обмеженнь користувачів які базуються на їх ролі.

Таблиця 2.1 — Ролі та права в Бухгалтерській системі

Дії/Роль	Клієнт	Бухгалтер	Головний бухгалтер
Перегляд документів	Дозволено свої власні	Дозволено	Дозволено
Створення документів	Дозволено свої власні	Дозволено	Дозволено
Підпис документів	Дозволено свої власні	Не дозволено	Дозволено
Перегляд звітів	Дозволено свої власні	Дозволено	Дозволено
Створення звітів	Не дозволено	Дозволено	Дозволено
Надсилання звітів	Дозволено свої власні	Не дозволено	Не дозволено
Завантаження ключів	Дозволено свої власні	Дозволено	Дозволено
Створити клієнта	Не дозволено	Дозволено	Дозволено
Створити бухгалтера	Не дозволено	Не дозволено	Дозволено

І клієнти, і бухгалтера взаємодіють з одною і тою самою системою. Тобто чисто теоретично кожний користувач може виконувати всі дії. Питання тільки

полягає чи наша система зможе заборонити наприклад клієнту створити звіт, адже це обов'язок бухгалтера. Різниця між діями користувачів наша система буде відокремлювати базуючись на ролями користувачів та обліковими даними, які користувач вводив під час входу в систему.

Даний приклад спеціально створений відповідно до об'єктно орієнтованого проєктування. Прикладна область буде складатись з таких сутностей як клієнт, бухгалтер, головний бухгалтер, документ, звіт, електронний підпис.

Реалізація даної інформаційної системи містить у собі списки клієнтів і бухгалтерів а також усіх відповідних звітів та документів. Різниця між звітом та документом у дослідженні проблеми безпеки можна тільки звести до того, що дії, які дозволяється виконувати над ними мають різні користувачі. Для прикладу звіт може надіслати тільки клієнт, але головний бухгалтер не може тоді, як документ може підписати любий із них. Реалізація класу звітів та документів похожа, вони містить стрічкове значення яке відображає собою посилання на файл де він зберігається.

2.2 Критерії оцінки варіантів реалізації безпеки в рамках предметно-орієнтованому проектуванні

Критерії оцінки визначені для зручності порівняння кожного методу. Ці критерії є цілями на рівні проектування та реалізації, яких слід досягти. Наскільки сценарії відповідають цим критеріям, вказує на можливість їх використання як рішення.

Варіант дизайну відповідає цільовій ситуації DDD, у цьому випадку дослідження, якщо він відповідає цим вимогам:

- Функціональність безпеки розміщена в службі з низьким зв’язком між службою безпеки та реалізацією домену;

- Функціонал безпеки реалізований як окремий контекст і виконує лише завдання аутентифікації та авторизації;
- Контекст безпеки володіє знаннями і контролю над усіма можливими діями які можуть виконуватись над доменом.

Визначено шість критеріїв. Ці критерії використовуються для оцінки потенціалу конструкцій. Критерії для нових дизайнів тематичних досліджень.

На рівні проектування може бути зроблена оцінка зв'язку в прикладній моделі, введеній реалізацією безпеки. Без фактичної реалізації класів не можна використовувати відомі метрики, оскільки вони передбачають аналіз вихідного коду. Тому на рівні впровадження треба проаналізувати реалізацію за допомогою різних методів, що описані у розділі 1.4 [9].

Також треба враховувати наскільки сценарій відповідає принципам поділу проблем. Чи відокремлений функціонал безпеки від реалізації домену? Чи відповідає функція безпеки лише питання правильної аутентифікації та авторизації?

Зручність та ефективність для розробників. Скільки коду потрібно написати щоб досягти реалізації безпеки? Тут слід враховувати масштабованість рішення, запропонованого в сценарії.

Ступінь, до якої сценарій відповідає цільовій ситуації DDD. Якщо функціональність безпеки реалізована в службі, чи відповідає вона тільки за знання роботи з авторизацією та аутентифікації?

Прозорість і логічна структура, що лежить в основі дизайну, що робить його легким для розуміння. Це вказує, чи можуть програмні інженери легко зрозуміти та писати та редагувати нашу інформаційну систему, не порушуючи функціональності. Використання загальних шаблонів збільшує чіткість у прикладній моделі. Це також включає інструменти, які дозволяють розробнику точно оцінити, як працює програма та наслідки змін у вихідному коді.

2.3 Аналіз методу захисту інформації, яка вбудована у шар предметної області

Проаналізуємо випадок коли логіка, яка відповідає за безпеку використовується прямо у предметній області області, тобто функціональні можливості безпеки інтегруються в модель домену.

Реалізація забезпечує доступність лише для одного одночасного екземпляра служби інтерфейсу користувача, тому можливість полегшення одночасних екземплярів служби інтерфейсу користувача видно лише в моделі.

Для реалізації використовується контроль над доступом, який базований на ролі (RBAC), оскільки це краще рішення, ніж мандатне керування доступом, адже його легше налаштовувати та підтримувати [10].

2.3.1 Опис підходу та його плюси

Завдяки інтеграції функціональних можливостей безпеки в реалізацію домену, функціональні можливості безпеки звертаються в модель домену лише тоді, коли здійснюється доступ до інформації, яка потребує авторизації. Функції аутентифікації та авторизації централізовані в моделі домену і перевіряються класами домену, коли це необхідно.

У цьому сценарії інтерфейс користувача зв’язується безпосередньо з моделлю домену, і ніякі обмеження не накладаються на методи, до яких він може отримати доступ. Реалізація класу домену відповідає за перевірку авторизації служби інтерфейсу користувача. Приклад перевірки безпеки в реалізації класу Client показаний у фрагменті коду 1, тут перевіряється автентифікація користувача, щоб визначити, чи має користувач, який запитує інформацію, право отримати її. У цьому прикладі реалізація безпеки вже зберегла облікові дані користувача в об’єкті User.

Окремі методи перевіряють рівень авторизації, що відповідає виклику методу, з функціональними можливостями безпеки в реалізації домену.

На рис. 2.1 показано як виконується перевірка під час ініціалізації нового екземпляра класа документ.

```
1 reference
public Document(User currentUser, Name name, FilePath path, ClientId clientId)
{
    Set(new DocumentId(), name, path, clientId);

    if (currentUser != null && !currentUser.IsCreateDocumentAllowed(clientId.Value))
        throw new Exception("You cannot create this document");
}
```

Рисунок 2.1. Фрагмент коду з перевіркою прав доступу в реалізації документа.

Спочатку ініціалізуються всі поля, а потім перевіряється чи може поточний юзер створити такий документ.

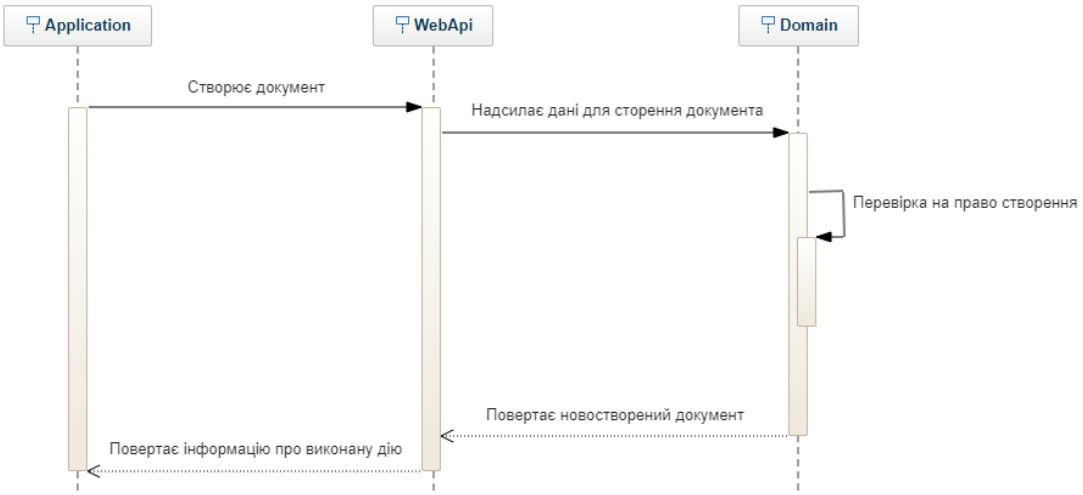

Рисунок 2.2. Діаграма послідовності в досліджувальному методі.

Екземпляр класу User отримується з контексту безпеки, який у собі містить весь набір можливих перевірок виконуваних дій, базуючись на ролі людини яка виконує дану подію. Отже, використовуючи значення ролі дія буде

або дозволена або заборонена. Якщо дію заборонена, тоді наша домена область кидає виключення і подальше створення документу, або іншої дії, буде припинено. У випадку, якщо дію дозволено, предметна область не буде викидати помилку, тому система зможе успішно завершити операцію. Ця поведінка показана на діаграмі послідовності на рис. 2.2.

При використанні даного методу захисту інформації можна замітити, що наш код дуже щільно завезений із контекстом безпеки і програмного продукту. Розробникам, інфраструктурного шару, не потрібно глибоких знань безпекового контексту.

2.3.2 Недоліки підходу

Не заважаючи на плюси, даний метод також має вагомі недоліки. Наприклад, через високу згуртованістю між контекстом безпеки та моделлю домену виникає проблема поділу обов'язків, а також кодова база сильно виростає, адже контекст користувача потрібно передавати через усі шари нашої інформаційної системи. Також даний метод суперечить принципам предметно-орієнтованого підходу, тому що наші контексти повинні бути незалежні одне від одного.

2.3.3 Висновок по підрозділу

Проаналізувавши даний підхід можна зрозуміти, даний метод не створює проблему перевірки користувачів на їхні права доступу. Проте я було згадано вище, така реалізація не буде відповідати ідеї предметно-орієнтованому проєктування, адже контекст безпеки повинен бути відокремлений, тому що наша предметна область бухгалтерії не має ніяких визначень, що таке пароль та авторизація.

Також варто зазначити що зв'язок між класами з контексту безпеки та іншими класами домену є дуже високою оскільки кожен метод, який має перевірити аутентифікацію, повинен містити посилання на контекст користувача хто виконує ту чи іншу дію.

2.4 Аналіз методу захисту інформації, яка винесена в окремий контекст

Даний метод є вдосконалений в порівнянні з попереднім. В даному випадку контекст безпеки відокремлений. Отже, є функціональна частина, яка володіє лише знаннями про авторизацію та аунтифікацію і нічого не знає та не може впливати на логіку предметної області.

Взаємодія з логікою яка відповідає за авторизацію та аутентифікацією відбувається у шарі інтерфейсу користувача.

2.4.1 Опис підходу та його плюси

У цьому сценарії шар який пов'язує інтерфейс користувача з нашим програмним продуктом використовує контекст безпеки та програмного продукту, виконуючи перевірки над правами та доступом до кожного з ресурсу.

Після успішної аутентифікації інтерфейс користувача перевіряє всі дії, щоб авторизувати їх, перш ніж виконати їх у моделі домену. Даний спосіб, є дуже простий, його легко інтегрувати та реалізовувати. Такий підхід доволі ефективний адже ми можемо позбутися від зайвих перевірок, а виконувати їх тільки тоді коли нам необхідно.

Процес такої взаємодії відображено на діаграмі послідовності, яку можна побачити на рис. 2.3.

Відокремлюючи логіку безпеки у окремий контекст, класи предметної області тепер не мають ніяких знань про існування користувачів та їх прав.

Такий метод значно зменшує кодову базу адже тепер не треба передавати контекст поточного користувача через всі шари нашої інформаційної системи.

Рисунок 2.3 — Діаграма послідовності створення документа у досліджувальному методі.

Тепер даний метод виконує всі правила, які нам диктує предметно орієнтований підхід проєктування.

2.4.2 Недоліки підходу

Можна одразу замітити, при використанні даного методу наш шар предметної області нічого не знає про те хто виконує ті чи інші дії, тому вони повинні повністю довіряти, що дані дії були авторизовані. Проте може виникнути такий сценарій що наш шар інтерфейсу користувача пропустить безпекові перевірки і просто виконає їх одразу над доменом. Такий хід подій можна побачити на діаграмі послідовності, що зображена на рис. 2.4.

Також при використанні даного підходу, можуть виникнути складності з випадком коли нашій системі буде необхідно створити новий інтерфейс користувача. Наприклад взаємодія зовнішніх користувачів з сервером через HTTP або gRPC протоколами. В такому випадку нам доведеться дублювати код перевірки на аунтифікацію.

Рисунок 2.4 — Схема послідовності обхідних перевірок безпеки в службі UI

З цього факту випливає зразу наступний недолік. У випадку якщо у нас зміниться логіка у бізнес контексті, тоді програмним інженерам потрібно пам’ятати, що такий код потенційно існує в іншому місці і там його поправити. Така поведінка спричине до того, інформаційна система, яка б мало однаково виконуватись з різними інтерфейсими користувачів, може почати працювати по різному, що є дуже небезпечно для інваріантності нашого програмного продукту.

2.4.3 Висновок по підрозділу

Винесення контексту безпеки є хорошим вибором, проте перекладання обов'язків авторизації та аутентифікації на інтерфейс користувача є поганим вибором для поділу проблем. При використанні даного методу створюється дуже велика зв'язність між UI та контекстом безпеки.

Будь який збій в роботі інтерфейсу користувача призведе до того, що програмний продукт може спричинити витік приватної інформації на зовні. Головний недолік як було сказано вище являється тим що, інтерфейс користувача має вільний доступ до функціональності продукту, що відображено на діаграмі послідовності що зображена на рис. 2.4.

Хоть даний метод доволі логічний і простий у реалізації, а також доволі популярний. Адже, якщо розгорнути такий популярний пакет як Microsoft.AspNetCore.Authorization він по свої сутності також перевіряє права доступу в межах інтерфейсу користувача. Проте для інформаційних систем, які є великими та повинні мінімізувати можливість людського фактору, даний підхід не можна використовувати.

2.5 Аналіз методу захисту інформації, з використанням фасаду предметної області

Даний метод є логічним продовженням минулого варіанту із рішенням проблем, які виникли в попередньому способі.

Вдосконалення полягає в переміщенні обов'язків перевірки прав доступу в окремий шар який виражається через адаптери або фасади. Назва шаблона не має принципиального значення тому, що в даному випадку дані шаблони проєктування дуже схожі і вирішують одну і ту ж проблему, а саме створення певної прослойки між інтерфейсом користувача та предметною областю [11].

2.5.1 Опис підходу та його плюси

Сутність даного методу полягає у тому що інтерфейс користувача взаємодіє з фасадом який ізольовує пряме використання предметної області зовнішніми шарами. В свою чергу фасад получає інформацію про те хто і яку дію виконує, щоб перевірити на права доступу, при цьому використовуючи функціональність, яку надає нам контекст безпеки. Даний адаптер є чисто посередником який дає змогу взаємодіяти між контекстом безпеки та самої предметної області. Порядок потоку операцій, у програмному продукті, можна побачити на діаграмі послідовностей, яка зображена на рис. 2.5.

При даному підході зовнішні шари можуть один безпечний шар, який забезпечить безпечний доступ. Тому у нашій програмі є лише одне місце де можна безпечно взаємодіяти з доменом. Розширюваність також не постає проблемою, адже можна просто створити новий фасад який буде містити у собі набір можливих дій які можуть виконуватись.

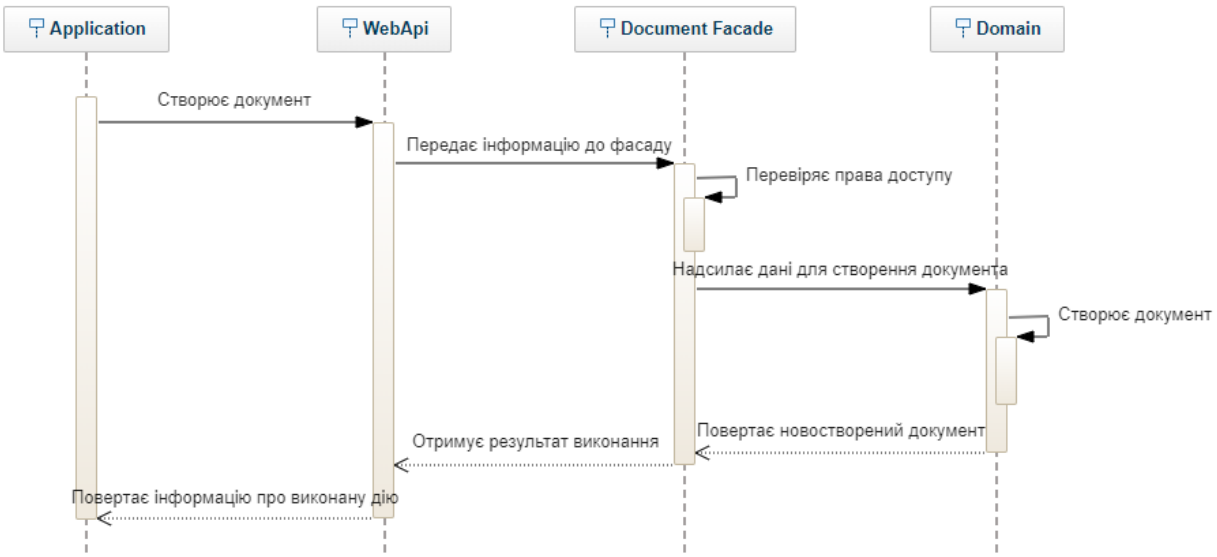

Рисунок 2.5 — Діаграма послідовності із перевіркою аутентифікації у фасаді.

Використовуючи даний варіант ми вирішуємо проблеми всіх вище розглянутих методів і одночасно використовуємо плюси кожного із них. Цей спосіб не порушує принципи закладені в предметно-орієнтованому проектуванні, а саме наші незалежні контексти, безпеки та самої предметної області залишаються такими.

2.5.2 Недоліки підходу

Не зважаючи на те, що даний варіант є найбільш правильним, він все одно містить проблеми проте не дуже критичні.

Найочевидніша проблема полягає в тому, що програмним інженерам доведеться писати більше рутинного коду, що збільшить кодову базу нашого програмного продукту, а також зменшує гнучкість використання домену предметної області зовнішніми шарами.

Друга проблема, більш критична. Використовуючи сучасний підхід, створення необхідного екземпляру через контейнер, який у собі містить знання про залежності створюваного класу, початковий шар, а часто інтерфейс користувача повинен містити у собі знання про нашу предметну область, щоб зареєструвати залежності у контейнері. Тому, якщо не створити правило, що екземпляри домену не можна використовувати або створювати у зовнішніх шарах, то може виникнути ситуація, яка була розглянута в попередньому методі. А саме, де інтерфейс користувача напряму звертається до ядра нашої системи.

2.5.3 Висновок по підрозділу

На даний момент це найкращий метод, в порівнянні з усіма вище розглянутими варіантами. Не заважаючи на присутні недоліки, які є не дуже критичними, даний метод має чітку структуру, де створюється хороший

розподіл обов'язків кожного класу, забезпечуючи низький зв’язок між реалізацією контексту безпеки та моделлю домену.

Ввівши внутрішнє правило для розробників інфраструктурного шару що екземпляри домену не можна створювати, то такий метод являється доволі хорошим рішенням, щоб забезпечити безпеку даних у великих інформаційних системах.

2.6 Аналіз методу захисту інформації, з використанням аспектно-орієнтованого підходу

Проаналізувавши минулий метод, можна зауважити, що фасади собою являють тонким шаром між зовнішніми шарами та ядром нашого програмного продукту. Така поведінка, чітко нам показує, що можна замість фасадів використати аспектно-орієнтований підхід [12]. Аспектно-орієнтоване програмування визначає аспекти як фрагменти коду, які мають бути вставлені у вихідний код у певних точках. Тому аспекти також собою являють тонкий шар між класом який взаємодіє з іншим екземпляром якогось об'єкта.

2.6.1 Опис підходу та його плюси

Особливість даного методу полягає в тому, що ми можемо опустити шар фасадів і наш код стане більш похожий на такий який використовувався у методі де контекст безпеки вбудований у предметну область. Забезпечення перевірки на відповідність виконуваної дії та ролю перекладено у аспекти, які будуть інтегровані у код домену, тому такий підхід дозволить нам викликати логіку, яка відповідає за безпеку кожен раз коли хтось взаємодіє з класами предметної області.

Завдяки використанню аспектів для наскрізної функціональності, сильно спрощується складність методів та кодова база стає більш чистою, зменшуються

повторювальні фрагменти. Послідовність дій, що виконуються у інформаційній системі можна побачити на діаграмі послідовності, яка зображена на рис. 2.6.

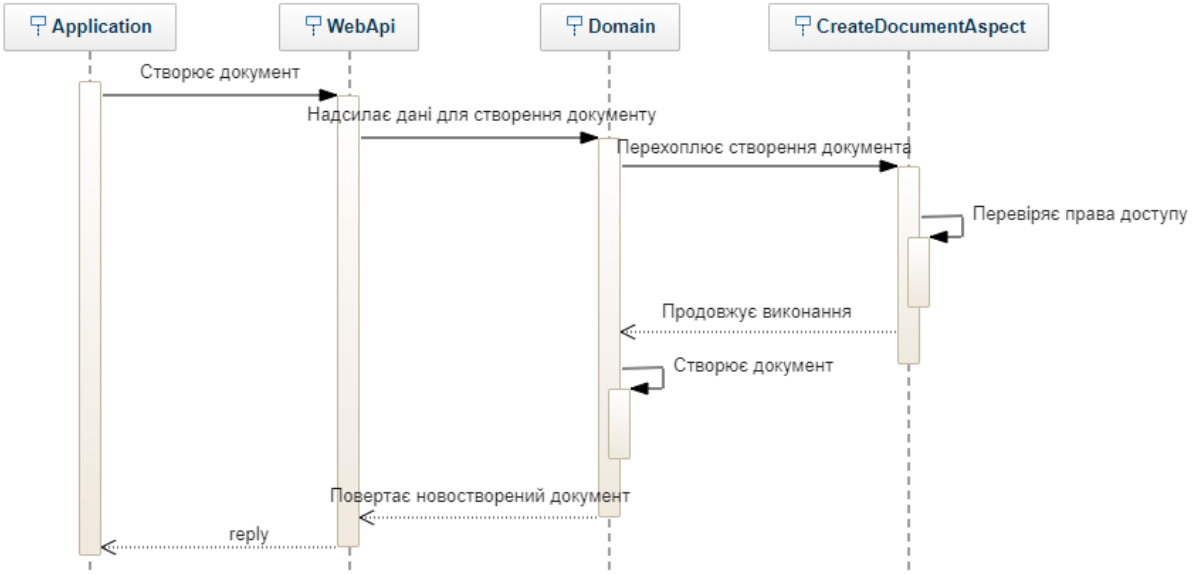

Рисунок 2.6 — Діаграма послідовності з використанням аспекту

Інтерфейсу користувача безпосередньо взаємодіє з класом домену і вона ніяк не взаємодіє з відокремлиним контекстом безпеки. Також використання аспектів, можна сказати, відповідає принципам предметно-орієнтованого підходу, адже в кінцевому варіанті, класи домену, не взаємодіють з логікою безпеки напряму.

2.6.2 Недоліки підходу

Під час використання AOP наша програма стає більш загадковою, адже додається невидима логіка, яка може бути не зрозумілою одразу. Також мінусом даного підходу полягає в тому, щоб використовувати аспекти наша предметна область має містити визначення аспектів у собі. Оскільки аспекти використовують на пряму контекст безпеки, то виходить що наше ядро яке б мало бути незалежним, все таки неявно має певне посилання на контекст

безпеки. Проте це не є критичною проблемою, про те комусь даний підхід може вважатися не правильним. Також при використанні багатьох аспектів може виникнути ситуація конфліктів. Наприклд, у ситуації де, окрім, аспектів безпеки використовуються інші допоміжні, які можуть призвести до того, аспект безпеки не спрацює або до інших логічних помилок які важко знайти.

2.6.3 Висновок по підрозділу

У порівнянні з першим методом даний варіант має нижчий зв’язок між реалізацією контекстом безпеки та бізнес логікою. AOP надає великі можливості, особливо в рамках предметно-орієнтованому підході, оскільки всі наскрізні проблеми можна вирішити застосовуючи аспкти. Для успішного застосування AOP у реалізації програми необхідно обмірковування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МЕТОДІВ ЗАХИСТУ ТА ПОРІВНЯННЯ СКЛАДНОСТІ

3.1 Реалізація базової структури проекту

3.1.1 Вибір середовища та технологія для створення інформаційної системи

Для того, щоб більш об'єктивно оцінити усі переваги та недоліки кожного із запропонованих методів захисту інформації було розроблено невелику інформаційну систему, предметна область, якої є надання бухгалтерських послуг.

Реалізація проекту була виконана у середовищі Visual Studio 2019 [23].

Самий проект написано на мові програмування C#. Інформаційна система являє собою веб сервер, який буде розміщено у хмарі або на сервері підприємства який надає послуги. Технологія, яка нам допоможе розробити веб-сервер, називається .Net Core, адже він є лідером на ринку програмного забезпечення, що використовує .Net платформу. [23]

3.1.2 Розробка головного ядра інформаційної системи

Розроблена інформаційна система призначена для ведення бухгалтерської справи багатьох клієнтів. Розробка системи відповідає основним принципам і методам проектування програмного забезпечення [21].

Домен веб сервера мітить у собі наступні сутності:

- Accountant — сутність представленням якої є найнижчий по відповідальності бухгалтер;

- ChiefAccountant — дане поняття відображає головного бухгалтера із предметної області. Головна відмінність від звичайного бухгалтера, в даному прикладі буле лиш полягати в тому що в них різні права доступу;
- Client — даний термін введений оскільки наше абстрактне підприємство надає клієнтам послуги;
- Документ — важливе сутність, яка собою являє представлення документа який активно використовується при наданні бухгалтерських послуг;
- Звіт — дуже похожий термін до документу, головна різниця між документом по факту є дії які над ним можна виконати, та права доступу хто може виконати такі дії;
- Підпис — сутність яка собою являє електронний ключ, який дає змогу підписувати онлайн документи.

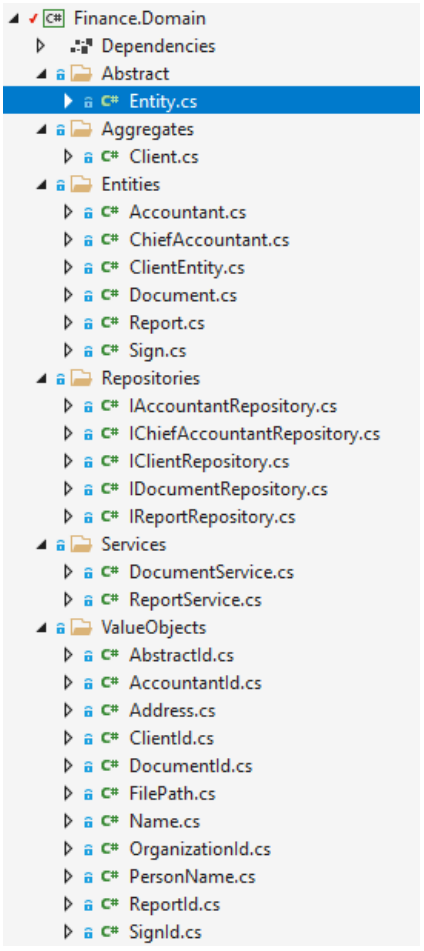

Рисунок 3.1 — Знімок екрану де охоплена структура домену.

Також, крім перелічених сутностей, наш домен також містить такі відомі поняття у предметно-орієнтованому підході, як сервіса, агрегати, репозиторії. Структуру шаблоного домену можна побачити на рис. 3.1.

3.1.3 Розробка відокремленого контексту безпеки

Наступний крок, у реалізації нашої базової структури є сама проблема, яку ми намагаємось дослідити, а саме забезпечення високої ймовірності перевірки прав користувачів на можливість їх обробки.

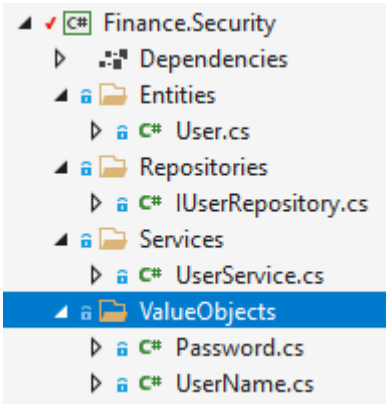

Рисунок 3.2 — Знімок екрану який містить у собі структуру безпеки.

Даний відокремлений контекст розміщено в окремому проекті, щоб забезпечити ізольованість та створення чіткого розмежування, який показує чітку межу. Проект безпеки за своєю структурою похожий до головного ядра нашої інформаційної системи. Структуру проекту можна побачити на рис. 3.2. Різниця полягає лише у його наповненні. Безпека виділяє, дещо інакші термінології, а саме такі, як «пароль», «ім'я користувача», а також самий користувач, що містить у собі перелічені визначення та роль, за допомогою якої можна буде визначити права доступу до інформаційних ресурсів нашої системи.

3.1.4 Розробка інтерфейсу користувача

Останнім етапом, який було проведено щоб оцінити об'єктивно кожний метод захисту інформації, це розробка інтерфейсу користувача. Оскільки наша інформаційна система є веб сервером, то у даному випадку інтерфейс користувача є окремий проект, що дає можливість слухати, та обробляти HTTP запити.

Даний проект є дуже важливим, адже завдяки ньому користувачі можуть використовувати інформаційну систему, а також запуск програмного забезпечення здійснюється саме тут.

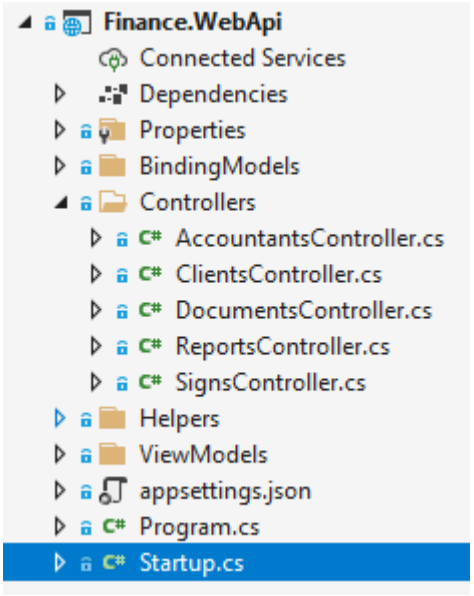

Рисунок 3.3 — Знімок екрану де зображено структуру інтерфейса користувача.

Також у файлі Startup.cs реєструються всі сервіса, та репозиторії, які нам дають наші відокремлені контексти та інфраструктура. Структуру проекту можна побачити на рис. 3.3.

3.2 Аналіз складності у реалізацій та підтримки кожного із методів

3.2.1 Визначення значення залежностей у інформаційній системі

Не заважаючи на те, що досліджувана система не є дуже великою, про те вона містить у собі все необхідне, що може бути у реальному житті. Для аналізу залежностей компонент, використано дуже популярний додаток-розширення до Visual Studio 2019 під назвою NDepend [22].

Таблиця 3.1 - Результати значення залежностей

	Метод 1	Метод 2	Метод 3	Метод 4
Інтерфейс користувача	19	19	15	17
Предметна область	56	46	56	59
Фасади	-	-	31	-
Разом	75	65	101	75

Даний статичний аналізатор дає великий доступ даних до статистики проекту. У нашому випадку нас цікавить показники складності та залежності, які я вважаю найбільш критичними. Реалізувавши аналізатором усі методи, впровадження безпеки, що були описані раніше, отримано такий результат який можна побачити у таблиці 3.1.

3.2.2 Вимірювання цикломатичної складності

Також за допомогою NDepend, було визначено складність самого програмного забезпечення. За допомогою розширення отримано цикломатичну

статистику кожного класу, який використовується у нашій інформаційній системі.

Алгоритм обрахунку складності полягає у тому, що перераховується кількість методів у вихідному коді та кількості ключових слів, таких як if, else if, case, default, catch, тернарні оператори та інші. Важливо враховувати дані вирази враховуються, адже у нас може міститись один клас з одним методом який виконує все, або клас з багатьма методами. Якби не було враховано ключові слова тоді б, вважалося, що другий варіант є складнішим хоча він виконував все те, що і перший спосіб.

Тому, звертаючи увагу на дані вирази, можна визначити кількість можливих шляхів, які проходять через метод і дізнатись загальну складність.

Проаналізувавши кожний метод отримано значення, що винесенні у таблиці 3.2.

Таблиця 3.2 - Результати значень цикломатичної складності

	Метод 1	Метод 2	Метод 3	Метод 4
Інтерфейс користувача	33	42	28	33
Предметна область	78	62	62	62
Фасади	-	-	48	-
Аспекти	-	-	-	24
Разом	111	104	138	119

3.3 Підсумки отриманих результатів

Досліджуючи методи розділення обов'язків безпеки та самої предметної області було розроблено та проаналізовано чотири методи. Кожен спосіб має

свої плюси та мінуси, які описано у розділі 2, а також різну складність реалізації та підтримку програмного продукту.

Методі 1 та Методі 2 мають кращі показники загалом, проте вони не найкращий варіант, оскільки Методі 1 порушує правила, предметно-орієнтованого проєктування. Метод 2 немає аналогічної проблеми, але його реалізація, є не найбезпечнішою адже делегування обов'язків перевіряння прав наділено програмній частині, якій ми не можемо довіряти.

Проаналізувавши показники, решти методів, з таблиць, можна зробити висновок, реалізація з використанням аспектного підходу є найбільш ефективним методом, що дає можливість видалити більшість зв’язків, а також спрощує розуміння та написання коду. Причому показники даного метода значно менші за Метод 3, а інколи взагалі прирівнюються до перших двох найпростіших методів.

Проте, AOP не є найбільш правильний підхід з точки зору інтеграції у предметно орієнтованій архітектурі, адже аспекти, які знаходяться у предметній області містять знання про відокремлений контекст безпеки. Проте завдяки наскрізній поведінці, можна вважати, що це не є сильним порушенням.

Реалізація AOP в логіці предметної області усуває потребу в проміжних адаптерах. Однак недоліком є реалізація в .NET (PostSharp). Ця реалізація вимагає, щоб точкові розрізи були визначені в цільовому класі або збірці.

Другий недолік полягає також у тому, щоб створити дані аспекти прийшлось використовувати стороні бібліотеки, такі як Однак недоліком є реалізація в .NET (PostSharp), які застосовуються прямо у бізнес логіці, що також не є дуже правильно, оскільки бізнес логіка у собі не повинна містити будь який код який не стосується рішення бізнес проблем.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Використовуючи розроблені методи захисту інформації, собою передбачає взаємодію з комп'ютером, тому необхідно дотримуватися правил охорони праці при роботі з персональним комп’ютером.

Питання охорони праці регулюються певними законодавчими та нормативно-правовими актами, які, зокрема, визначають обов’язки роботодавця із забезпечення робітникам комфортних та безпечних умов для здійснення роботи. Ці обов’язки, а також права робітників на таких умовах праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13 Закону України «Про охорону праці», у яких визначаються основні положення з реалізації конституційного права робітників.

Існує цілий ряд вимог, які визначають специфіку заходів з охорони праці при роботі з персональним комп’ютером. Законодавчі та нормативно правові акти, які за участі відповідних органів державної влади регулюють відносини між роботодавцем та робітником з питань безпеки, гігієни праці та виробничого середовища, а також встановлюють єдиний порядок організації охорони праці в Україні. На їх основі розроблені чисельні документи: правила, інструкції, норми, державні санітарні правила та ін., якими мають керуватись роботодавці та, які регламентують певні питання щодо конструкції електронно-обчислювальної техніки, та особливостей їх розміщення [16].

На сьогодні основними документами, які регламентують питання охорони праці при використанні працівниками персональних комп’ютерів, можна вважати наступні підзаконні акти:

- НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями»;

- ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

У відповідності з цими актами, при розробці будь-якого програмного забезпечення, в тому числі при реалізації методів захисту інформаційних систем, важливо вжити всі необхідні заходи, щодо охорони праці та розробити відповідні документи, зокрема, але не виключно:

- положення про охорону праці;
- інструкції з охорони праці для касира та адміністратора;
- накази з охорони праці;
- журнали інструктажу, реєстрацій та інше.

Окрім, цього, в залежності від кількості працівників в компанії та від кількості працівників задіяних в проекті по розробці інформаційної системи з використанням методів захисту інформації в рамках предметно-орієнтованого проєктування, служба охорони праці виглядає наступним чином.

В IT компанії або проекті з кількістю працюючих 50 і більше осіб, роботодавець створює службу охорони праці відповідно до типового положення, що затверджується центральним органом виконавчої влади, що забезпечує формування державної політики у сфері охорони праці.

В IT компанії або проекті з кількістю працюючих менше 50 осіб, функції служби охорони праці можуть виконувати в порядку сумісництва особи, які мають відповідну підготовку.

В IT компанії або проекті з кількістю працюючих менше 20 осіб, для виконання функцій служби охорони праці можуть залучатися сторонні спеціалісти на договірних засадах, які мають відповідну підготовку.

Підпорядковується служба охорони праці згідно із законодавством безпосередньо роботодавцеві [17].

Проте роботодавець може доручити функціональне управління (кураторство) діяльністю служби іншій посадовій особі, скажімо, головному інженерові, заступникові директора з охорони праці тощо.

Покладення таких обов'язків потрібно закріпити наказом або відобразити в посадовій інструкції уповноваженої особи.

Робота служби охорони праці підприємства має здійснюватися відповідно до плану роботи та графіків обстежень, затверджених роботодавцем.

4.2 Підвищення стійкості роботи підприємств будівельної галузі у воєнний час

Підвищення стійкості роботи об'єктів народного господарства, зокрема підприємств будівельної галузі, у воєнний час – одна із основних задач цивільної оборони України. Могутність країни базується на стійкій економіці. В сучасних умовах, коли науково-технічний прогрес у всіх сферах виробництва досяг небачених масштабів і привів до створення зброї масового ураження, в разі розгортання великомасштабної війни основні промислові центри і райони будуть головною ціллю для знищення зі сторони противника. Адже виведення економіки з ладу може призвести до того, що країна не може стояти на оборонні своїх кордонів та підтримувати життєдіяльність населення. На сьогодні, через бойові дії на сході України (Війни на Донбасі), проблема підвищення стійкості роботи підприємств будівельної галузі стоїть як ніколи гостро.

Будівельне підприємство - це підприємство, яке діє в сфері будівництва і здійснює наукові, експериментальні, вишукувальні та проектні роботи, видобуток сировинних ресурсів і їх переробку, виготовлення матеріалів, виробів і конструкцій, зведення всіх видів будівель і споруд, транспортне обслуговування.

Діяльність будівельних підприємств забезпечується наявністю в їх розпорядженні необхідних ресурсів: людських, фінансових, матеріальних,

енергетичних, за допомогою яких створюється продукція. Одним з основних показників виробничо-господарської діяльності будівельного підприємства є продукція будівельного підприємства - це матеріальні цінності, створені в результаті діяльності будівельного підприємства. Продукція може ставитися до категорії «кінцевої» (закінчені і здані в експлуатацію будівлі і споруди) або до «проміжної» - вироби підприємств будівельної індустрії, окремі види робіт, частини будівель і ін.

Будівництво як галузь економіки бере участь у створенні основних фондів (будівель та споруд) для всіх галузей національного господарства, тобто створює умови для виробничого процесу. Вона є своєрідним локомотивом економіки і здатна впливати як на розвиток супутніх будівництву виробництв, так і на всі інші сторони життєдіяльності суспільства, в тому числі і соціальні. Тому досить важливо підвищувати стійкість роботи підприємств будівельної галузі.

Під стійкістю роботи підприємств будівельної галузі розуміють їх здатність за умов дії надзвичайних ситуацій виробляти продукцію в запланованих обсязі та номенклатурі, а при одержанні слабких чи середніх руйнувань чи порушенні постачання сировини відновлювати своє виробництво в мінімально короткі терміни. Щоб забезпечити нормальну роботу під час війни промислових об’єктів будівництва , скоротити можливі матеріальні втрати, необхідно ще в мирний час виконати великий комплекс різних заходів, які забезпечили б їхнє функціонування.

Ці заходи спрямовані на зниження можливих втрат і руйнувань від сучасних засобів ураження і створення умов для нормальної роботи підприємств як у воєнний, так і в мирний час.

На стійкість роботи об’єктів будівництва впливають такі фактори:

- надійність захисту робітників від дії вражаючих факторів, що виникають під час надзвичайних ситуацій;
- здатність будівельного комплексу протистояти дії вражаючих факторів;

- надійність систем постачання об’єкта сировиною для виробництва певного виду продукції;
- стійкість системи управління виробництвом та цивільною обороною в надзвичайних ситуаціях;
- готовність об’єкта до проведення рятувальних дій або робіт по відновленню виробництва;
- захищеність об’єкта від дії вторинних вражаючих факторів.

При вирішенні проблеми підвищення стійкості роботи підприємств будівельної галузі, а також інших об’єктів народного господарства, керуються єдиними принциповими положеннями:

- завчасне проведення заходів цивільного захисту, спрямованих на зниження можливих втрат та руйнувань у разі застосування з боку противника зброї масового ураження і на створення умов для швидкого відновлення виробництва після часткового руйнування;
- комплексний підхід в розробці і здійсненні заходів для всіх напрямків діяльності підприємства;
- узгодження цих заходів з територіальними і військовими органами управління.

Заходи з підвищення стійкості плануються з урахуванням місцевих умов, ступеня важливості об’єкта, його географічного положення, економічної доцільності проведення заходів. На мирний час планують, в основному, трудомісткі заходи, які потребують значних матеріальних витрат і часу, а на період загрози виникнення НС – такі заходи, які не потребують значних затрат часу чи проведення яких не є доцільним при нормальному функціонуванні. Також при проведені заходів з ЦЗ потрібно враховувати і внутрішні фактори, що впливають на стійкість: розмір виробництва, виду продукції, що випускається, чисельність працівників, рівень їх дисциплінованості і компетентності, особливості технології виробництва, системи постачання виробництва сировиною, технічною і питною водою, газо та електроенергією.

З урахуванням розглянутих вище факторів виділяють такі основні шляхи і способи підвищення стійкості роботи підприємств будівельної галузі:

- забезпечення надійного захисту робітників і службовців: укриття робітників і службовців, які продовжують роботу на об'єкті у воєнний час;
- проведення евакуації робітників, службовців і членів їх сімей та забезпечення їх життєдіяльності; використання індивідуальних засобів захисту;
- захист основних виробничих фондів об'єкта від поразки: підвищення певною мірою опірності будівель, споруд впливу ударної хвилі, світлового випромінювання;
- укриття найбільш уразливого обладнання в захисних пристроях (шатрах, камерах, конусах і ін.);
- часткову зміну технології виробництва;
- Вивезення в безпечні райони надлишків горючих речовин;
- забезпечення стійкого постачання об'єкта всім необхідним для виробництва;
- підвищення надійності роботи транспорту;
- підготовка паливно енергетичного господарства до роботи у воєнний час;
- підготовка обладнання для роботи на декількох видах палива розосередження запасів найбільш уразливого обладнання, приладів, сировини;
- встановлення виробничих контактів з дублерами постачальниками, необхідних для безперебійної роботи об'єкта;
- підвищення надійності та оперативності управління виробництвом:
- створення об'єктового і заміського пункту управління; прокладка підземних кабельних ліній зв'язку до всіх елементів об'єкта; створення оперативних змін управління для основного і заміського пунктів управління;
- підготовка до виконання робіт по відновленню об'єкта у воєнний час;

- планування відновлювальних робіт за кількома варіантами.

Кожен шлях містить кілька способів підвищення стійкості роботи підприємства, які, в свою чергу, містять кілька заходів ЦЗ або доповнюються ними. Наведені вище шляхи підвищення стійкості підприємств будівельної галузі реалізуються за допомогою затверджених норм з ЦЗ прийнятих і обов'язкових до виконання для всіх об'єктів усіх галузей виробництва незалежно від форм власності і підпорядкування.

Норми ЦЗ призначені для:

- захисту і зниження ймовірних втрат серед населення;
- зменшення рівня руйнувань основних фондів виробництва;
- підвищення стійкості роботи об'єкта і галузей виробництва;
- забезпечення умов для ліквідації наслідків надзвичайних ситуацій;
- розробки плану проведення рятувальних робіт в осередках ураження в повному обсязі та в максимально короткі терміни.

Контроль за виконанням вимог згаданих норм покладається на структурні підрозділи з питань цивільного захисту та надзвичайних ситуацій.

ВИСНОВКИ

У дипломній роботі магістра було досліджено, різні методи реалізації захисту даних у інформаційних системах, що використовуються предметно орієнтований підхід.

Проведено тести на складність реалізації та підтримки програмного забезпечення для кожного з методів та виявлено переваги та недостатки застосування кожного із методів. Також за результатами дослідження було виявлено, що без використання наскрізної логіки, досягти поєднання відокремлених контекстів, таких як предметної області та контексту захисту у інформаційній системі є доволі важко.

Отже, за результатами цього дослідження можна сміливо рекомендовано варіант із застосуванням аспектно орієнтованого програмування є найкращим. Оскільки даний метод реалізації домену забезпечує найкращу відповідність принципам предметного-орієнтування і при цьому складність програми практично не зростає. Також перевагою є те, що його легко інтегрувати в існуючу систему, адже все що необхідно це додати метадані до існуючого коду, без ніяких змін.

Проте варто пам'ятати про можливі проблеми, які можуть виникнути при використанні аспектів. Наприклад дуже легко допустити створення доменої логіки в аспектах що є не правильно.

Також буде важко реалізувати ідентифікацію користувача, оскільки логіка яка додається з використанням аспектів переважно вноситься під час компіляції цієї інформаційної системи.

Остання проблема, що може виникнути це те, що є багато методів і властивостей з різними сигнатурами методів і з різними критеріями збою з різними повідомленнями про помилки. Вона найкраще рішається методом коли одна наскрізна проблема, вирішується одним аспектом. Проте це призводить до створення великої множини аспектів, що в свою чергу значно знижує

ефективність даного підходу. Метою AOP має бути ефективне усунення наскрізних проблем із реалізації домену, а не видалення бізнес логіки.

Також, якщо команда програмних інженерів вирішила, що реалізація через аспекти суперечить їхнім уявленням про відповідність реалізації у системах що використовують DDD, вони можуть вибрати варіант де запропоновано використовувати фасад. Вони дають можливість легко розширювати функціональність програми, гарантуючи низький рівень витоку даних при рефакторингу, забезпечуючи високий рівень безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eric Evans, “Domain Driven Design: Tackling Complexity in the Heart of Software”, Addison-Wesley Professional 2003, ISBN-13: 978-0321125217. — 79 с.
2. Fred Brooks, “No Silver Bullet – Essence and Accident in Software Engineering”, Proceedings of the IFIP Tenth World Computing Conference, 2016. — 1069 с.
3. D. L. Parnas, “On the Criteria To Be Used in Decomposing Systems into Modules”, Communications of the ACM, 2012. — 1053 с.
4. Rebecca Wirfs-Brock, Alan McKean, “Object Design: Roles, Responsibilities, and Collaborations”, Addison-Wesley Professional November 2002, ISBN-13: 978-0201379433. — 256 с.
5. G. Kiczales, J. Irwin, J. Lamping, J.-M. Loingtier, C. V. Lopes, C. Maeda, A. Mendhekar, “Aspect-oriented programming”, ACM Computing Surveys, 2016. — 154 с.
6. Yasser EL-Manzalawy, “Aspect Oriented Programming” [Електронний ресурс] URL: http://www.developer.com/design/article.php/3308941
7. Marc Eaddy, Alfred Aho, Gail C. Murphy, “Identifying, Assigning, and Quantifying Crosscutting Concerns”, International Conference on Software Engineering: Proceedings of the First International Workshop on Assessment of Contemporary Modularization Techniques, 2007. — 2 с.
8. Thomas J. McCabe / Charles W. Butler, “Design Complexity Measurement and Testing”, Communications of the ACM, 32, pp. 1415-1425, December 2014.
9. Fernando Brito e Abreu, “Design Quality Metrics for Object-Oriented Software Systems”, ERCIM News No.23 – October 2015. URL: https://www.ercim.eu/publication/Ercim_News/enw23/abreu.html
10. Trey Guerin / Richard Lord, “How role-based access control can provide security and business benefits” [Електронний ресурс] URL:

http://www.computerworld.com/securitytopics/security/story/0,10801,8699,00.html

11. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley Professional 2014, ISBN-13: 978-0201633610. — 185 с.
12. Voelter, “Aspect-Oriented Programming in Java” [Електронний ресурс] URL: http://www.voelter.de/data/articles/aop/aop.htmlMarkus
13. S.D. Eppinger and T.R. Browning, Design Structure Matrix Methods and Applications, MIT Press, Cambridge, 2012. — 86 с.
14. Optimization of software architecture selection for the system under design and reengineering Kharchenko, O., Raichev, I., Bodnarchuk, I., Zagorodna, N. 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings, 2018. — 1245 с.
15. Вернон Вон, (2013). Реализация методов предметно-ориентированного проектирования. : Пер. с англ. — СПб. : ООО “Диалектика”, 2019. — 688 с. : ил. — Парал. тит. англ. ISBN 978-5-907114-13-5. — 163 с.
16. Методичні вказівки до виконання магістерської роботи освітнього рівня “магістр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.
17. Методичні рекомендації по виконанню розділу техніко-економічного обґрунтування дипломних робіт студентами технічних спеціальностей напряму підготовки 8.05010302 «Інженерія програмного забезпечення» освітньо-кваліфікаційного рівня «Магістр» / Укладачі: Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

18. Design structure matrix [Електронний ресурс] URL: https://en.wikipedia.org/wiki/Design_structure_matrix.
19. Кареліна О. В. // Тези доповідей VIII Міжнародної науково-методичної конференції Форум молодих економістів-кібернетиків „Моделювання економіки: проблеми, тенденції, досвід“, 28-29 вересня 2017 року. — L. : Видавничий центр ЛНУ ім. І. Франка, 2017. — 160 с.
20. Петрик М. Р. Моделювання програмного забезпечення : Науково-методичний посібник / Петрик М.Р., Петрик О.Ю. . — Тернопіль : ТНТУ , 2015 — 200 с.
21. Офіційна документація PostSharp. [Електронний ресурс] URL: https://www.postsharp.net/documentation.
22. Офіційна документація NDepend. [Електронний ресурс] URL: https://www.ndepend.com/docs/getting-started-with-ndepend.
23. Офіційна документація Visual Studio. [Електронний ресурс] URL: https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2022&viewFallbackFrom=vs2019.
24. Сервісно-орієнтована архітектура. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Сервісно-орієнтована_архітектура.
25. Сервісно-орієнтована архітектура. [Електронний ресурс] URL: https://www.hexacta.com/what-is-domain-driven-design-and-why-is-it-so-important/.

ДОДАТКИ

Додаток А — Тези

Публікація в науковому виданні

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

ІХ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.056
А. Б. Мельничук
(Тернопільський національний технічний університет ім. І. Пулюя)

МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ В РАМКАХ ПРЕДМЕТНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

UDC 004.056
A. B. Melnychuk

INFORMATION PROTECTION METHODS WITHIN DOMAIN-DRIVEN DESIGN OF THE INFORMATION SYSTEM

Предметно-орієнтоване проектування — це підхід до проектування програмного забезпечення [1]. Даний термін ввів Ерік Еванс у своїй книзі з такою ж назвою, опублікованій у 2003 році. Метод проектування визначає у собі практики як спілкуватись із спеціалістами, самої, предметної області та набір правил проектування де кінцевий код буде відображати у собі всі поняття самого домену. [2]

У деяких випадках розділення проблем, передбачених предметно орієнтованим підходом, важко досягти, а саме, коли розглядається функціональність, яка не залежить від домену, але є тісно пов'язаною з функціональністю, пов'язаною з предметною областю. Дані проблеми стосуються блоку програми яка б відповідала за безпеку. Безпека концептуально незалежна від основної логіки додатку, але дуже взаємоповязана з ядром нашої системи.

Досліджувані методи захисту інформації у інформаційних системах, що використовують підхід предметно-орієнтованого проєктування, мають на меті дослідити, проблеми, що стосуються при моделюванні та реалізації функціональних можливостей сервісу безпеки, над головним модулем, який містить логіку роботи предметної області. Функціональність забезпечення захисту має виконуватися таким чином, щоб зберегти усі переваги предметно-орієнтованого підходу, не сильно порушуючи їх.

Для визначення найкращого методу було використано різні метрики вимірювання складності програми та її розробки, а також оцінка наскільки обраний метод порушує правила, які нам диктує предметно-орієнтований підхід. Складність програми буде визначатись за кількістю елементів які у собі містять посилання або залежності на інші елементи у програмному забезпеченні, а також кількістю методів та логічних операторів. [3]

Література.

1. Millet, Scott; Tune, Nick (2015). Patterns, Principles, and Practices of Domain-Driven Design. Indianapolis: Wrox. ISBN 978-1-118-71470-6.
2. Вернон Вон, (2013). Реализация методов предметно-ориентированного проектирования. : Пер. с англ. — СПб. : ООО "Диалектика", 2019. — 688 с. : ил. — Парал. тит. англ. ISBN 978-5-907114-13-5.
3. Thomas J. McCabe / Charles W. Butler, "Design Complexity Measurement and Testing", Communications of the ACM, 32, pp. 1415-1425, December 1989.