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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Модель управління ризиками в Agile-проєктах по розробці програмного забезпечення на основі програмних агентів

Виконав(ла): студент(ка) 6 курсу, групи СТм-61
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(підпис) Войтюк Р.О. (прізвище та ініціали)

Керівник (підпис) Марценко С.В. (прізвище та ініціали)

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

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Оробчук О.Р. (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 комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«21» вересня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

студенту Войтюк Руслан Олегович
(прізвище, ім'я, по батькові)

1. Тема роботи **Модель управління ризиками в Agile-проєктах по розробці програмного забезпечення на основі програмних агентів**

Керівник роботи **к.т.н., доц. Марценко С.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *жовтня* 2021 року № *4/7-911*

2. Термін подання студентом завершеної роботи 20 грудня 2021 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

ВСТУП; 1 ПРОБЛЕМИ РИЗИКУ В ПРОЄКТАХ AGILE ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 1.1 Підхід до вирішення проблеми управління ризиками 1.2 Модель Agile Risk Tool (ART) 1.3 Результати реєстрації ризиків; 2 ТЕМАТИЧНІ ДОСЛІДЖЕННЯ 2.1 Методологія дослідження проєктів 2.2 Вибір даних для ілюстрації методу 2.3 Приклад використання методу 3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ 3.1 Узагальнення даних проєкту CSA 3.2 Резюме даних проєкту CSB; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; 4.1 Охорона праці та її актуальність в IT-сфері; ВИСНОВКИ СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ; ДОДАТКИ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Дмитроца Л.П., доц.		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викл.		

7. Дата видачі завдання 21 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	21.09.21-27.09.21	*Виконано*
2.	Підбір наукових джерел по темі роботи	28.09.21-04.10.21	*Виконано*
3.	Переклад та опрацювання наукових джерел	05.10.21-11.10.21	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо огляду атак на	12.10.21-18.10.21	*Виконано*
	комп'ютерні системи		
5.	Оформлення першого розділу	19.10.21-25.10.21	*Виконано*
6.	Оформлення другого розділу	26.10.21-01.11.21	*Виконано*
7.	Оформлення третого розділу	02.11.21-08.11.21	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	09.11.21-15.11.21	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	16.11.21-22.11.21	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	23.11.21-29.11.21	*Виконано*
11.	Нормоконтроль	30.11.21-05.12.21	*Виконано*
12.	Перевірка на плагіат	05.12.21	*Виконано*
13.	Попередній захист кваліфікаційної роботи	14.12.21	*Виконано*
14.	Захист кваліфікаційної роботи	20.12.2121	

Студент _____ Войтюк Р.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марценко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Модель управління ризиками в Agile-проєктах по розробці програмного забезпечення на основі програмних агентів" // Войтюк Руслан Олегович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТм-61 // Тернопіль, 2021 // с. – 65, рис. – 13, табл. – 9, джерел – 46.

Ключові слова: УПРАВЛІННЯ РИЗИКАМИ, AGILE РИЗИКИ, AGILE ПРОЄКТИ, · ПРОГРАМНІ АГЕНТИ

Управління ризиками є важливим процесом у програмній інженерії. Однак це можна сприймати як дещо протилежне більш легким процесам, які використовуються в методах Agile. Таким чином, потрібна відповідна та реалістична модель управління ризиками, а також інструментальна підтримка, яка мінімізує людські зусилля. Ми пропонуємо використовувати програмні агенти для виконання завдань з управління ризиками та використовувати дані, зібрані з середовища проєкту, для виявлення ризиків. У цій роботі описується базова модель управління ризиками в спеціальному програмному інструменті Agile для ризиків, де програмні агенти використовуються для підтримки ідентифікації, оцінки та моніторингу ризику.

ANNOTATION

"The model of risk management in software development Agile-projects with program daemons" // Ruslan Voitiuk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СТм-61 // Ternopil, 2021 // p. – 65, fig. – 13, tables. – 9, ref. – 46.

Keywords: RISK MANAGEMENT, AGILE RISKS, AGILE PROJECTS, SOFTWARE AGENTS

Risk management is an important process in software engineering. However, this can be seen as the opposite of the easier processes used in Agile methods. Thus, an appropriate and realistic risk management model is needed, as well as instrumental support that minimizes human effort. We suggest using software agents to perform risk management tasks and use data collected from the project environment to identify risks. This paper describes the basic model of risk management in the special software tool Agile for Risk, where software agents are used to support risk identification, assessment and monitoring.

ЗМІСТ

ВСТУП 6
1 ПРОБЛЕМИ РИЗИКУ В ПРОЄКТАХ AGILE ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9
1.1 Підхід до вирішення проблеми управління ризиками 12
1.2 Модель Agile Risk Tool (ART) 13
1.2.1 Вхідні дані 13
1.2.2 Процес управління ризиками 19
1.3 Результати реєстрації ризиків 21
2 ТЕМАТИЧНІ ДОСЛІДЖЕННЯ 24
2.1 Методологія дослідження проєктів 24
2.2 Вибір даних для ілюстрації методу 26
2.2.1 Дані середовища 27
2.2.2 Правила ідентифікації ризику 29
2.3 Приклад використання методу 31
2.3.1 Альфа версія прикладного дослідження (CSA) 32
2.3.2 Бета-версія прикладного дослідження (CSB) 38
3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ 41
3.1 Узагальнення даних проєкту CSA 41
3.2 Резюме даних проєкту CSB 46
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 49
4.1 Актуальність забезпечення процесів охорони праці як фактор успішної реалізації проєктів в ІТ-сфері 49
4.2 Шкідлива дія шуту та вібрації і захист від неї 53
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62
ДОДАТКИ

ВСТУП

Актуальність задачі.

Управління ризиками визнано ключовою сферою процесу в розробці програмного забезпечення. Більшість літератури з управління ризиками стосується важких процесів, керованих планом, і зазвичай припускають, що, наприклад, вимоги були узгоджені та підписані до початку розробки. З іншого боку, Agile Software Development використовує ітераційний підхід до конструювання програмного забезпечення, спрямований на скорочення часу розробки, визначення пріоритетності цінності, одночасно покращуючи якість програмного забезпечення та знижуючи ризики [11].

У підході, який використовується, керівник проєкту повинен визначити ці елементи: цілі проєкту, сценарії проблеми, наслідки, індикатори ризику, дані про середовище проєкту, а також визначити правила ризику за допомогою попередньо визначеного «шаблона правила». Далі обговорюється запропонована модель інструменту ризику Agile (Agile Risk Tool – ART), зосереджена на розробці цього інструменту. Це показує, як діяльність з управління ризиками розкладається на агентів, а також як взаємодія між агентами використовується для забезпечення належного управління ризиками. Використання реєстру ризиків представлено там, де на панелі інструментів відображається список ризиків, ініційованих у проєкті. Великою перевагою підходу є те, що програмні агенти можуть використовуватися для виявлення ризиків і динамічної реакції на зміни в середовищі Agile проєкту.

Мета роботи. Ця робота має на меті продемонструвати ідею програмних агентів, які допомагають керувати ризиками при розробці проєкту. Це досягається шляхом використання програмних агентів для ідентифікації ризиків, оцінки ризиків та моніторингу ризиків, при цьому агенти використовують дані, зібрані з середовища проєкту.

Управління ризиками в наукових роботах завжди визнається надзвичайно важливим. Щоб визначити, що необхідно, ми використали існуючу роботу [34] щодо

дослідження бар’єрів для управління ризиками. Результати цього розслідування дозволили зробити висновок:

– Що не існує стандартного або загальноприйнятого процесу та/або інструменту управління ризиками, які використовуються в ситуаціях розробки програмного забезпечення.

– Ідентифікація ризиків була найбільш трудомістким процесом, і, крім того, 30% погодилися, що моніторинг ризику є найскладнішим і потребує більше зусиль.

– Найбільшою перешкодою було те, що видимим (і відчутним) витратам на розробку приділяється більше уваги, ніж нематеріальним, таким як втрата чистого прибутку та зобов’язання щодо подальшого розвитку.

– Незважаючи на визнання того, що методи управління ризиками сприяють розвитку системи, наданню цих методів не можна знайти підтримки [38].

Об’єкт дослідження: процеси розробки програмного забезпечення.

Предмет дослідження: наукові публікації на тему оптимізації процесів розробки ПЗ.

Методи дослідження. Для досягнення мети кваліфікаційної роботи використовувались:

– метод систематичних оглядів літератури (systematic literature reviews – SLR);

– емпіричні методи наукового дослідження, зокрема метод узагальнення .

Практичне значення отриманих результатів. У цій роботі стверджується, що методи управління ризиками при розробці програмного забезпечення не є вичерпними, оскільки вони мають справу з конкретними типами ризиків [2]. Крім того, незважаючи на те, що було запроваджено багато добре відомих підходів до управління ризиками, управління ризиками все ще не практикується належним чином [21, 36]. Як повідомлялося в [3], найбільш поширені підходи до управління ризиками, знайдені в літературі, виділяють такі практики, як контрольні списки, аналітичні рамки, моделі процесів і стратегії реагування на ризики. Багато дослідників проводили дослідження щодо адаптації управління ризиками, надаючи різні підходи.

Однак лише кілька досліджень повідомляють про інтеграцію управління ризиками із сучасним програмним забезпеченням [32].

Апробація результатів та особистий внесок здобувача. Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції, яка проводились у ТНТУ.

1 ПРОБЛЕМИ РИЗИКУ В ПРОЄКТАХ AGILE ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Зрозуміло, що проблеми з людьми є найбільш критичними в проєктах із застосуванням Agile, і що вони повинні бути вирішені, щоб Agile було успішно запроваджено [7]. Справді, одним з найважливіших факторів успіху в Agile-проєкті є індивідуальна компетентність [11]. Крім того, оцінка зусиль є постійною проблемою в роботі з розробки, особливо коли вона виконується вперше [16], і є проблеми з гнучкими навичками та плинністю кадрів, а також незадоволеність роботою [6, 27, 28]. У Scrum індивідуальна мотивація дуже важлива і впливає на те, наскільки старанні члени команди; наприклад, відвідування щоденних зустрічей Scrum [20]. Визнання недотримання усталеної практики може дати ранні ознаки ризиків, наприклад, низький моральний дух, виражений під час щоденних зустрічей, або уникнення обговорення проблем у разі відставання від графіка [26].

Через те, що Agile методи багато в чому залежать від довіри до людей, залучених до проєктів [11, 31], а також від їх мотивації у застосуванні методів Agile [25, 14], більшість проблем, з якими стикаються, стосуються людей та залучених практик. Це перегукується з одним із цінностей у маніфесті Agile, тобто «особи та взаємодія щодо процесів та інструментів» [18]. Це означає, що відсутність потрібних людей, які виконують правильний процес, буде джерелом ризику.

В роботі [10] колектив авторів розробив деякі дослідницькі роботи з питань і проблем гнучкої розробки програмного забезпечення за допомогою Scrum. Дослідницька робота в основному спрямована на надання рекомендацій, які допоможуть компаніям уникнути та подолати бар’єри у прийнятті методу. Було проведено поглиблене дослідження між двома різними компаніями з використанням різних методів збору якісних даних. У дослідженні були представлені різні загальні категорії проблем і проблем у Agile-проєктах, серед яких наступні моменти обговорюються в таблиці 1.1.

Автор [9] окреслив загальні підводні камені в Agile проєктах, у яких він запропонував, що керівнику проєкту важливо розуміти ці підводні камені, щоб

зменшити ризики. Серед важливих аспектів, які обговорюються, є: нездатність забезпечити достатню підготовку з методології Agile; незнайомство керівника проєкту з Agile методами; погана участь Власника продукту; команда, яка практикує «єдиного експерта» без обміну знаннями, а також має пасивних членів команди.

Таблиця 1.1 – Категорії проблем/викликів у Agile проєктах

Категорії	**Знайдено проблеми**
Людський ресурс	Формування команди, коли команда організувалася без урахування її необхідних навичок та знань Багато обов'язків, коли один член команди відповідає за багато завдань Деякі розробники не знають про переваги застосування методів Agile, і тому неохоче застосовують такі методи. Відсутність відповідальності, коли члени команди не несуть відповідальності за будь-яке відкладене завдання в поєднанні з відсутністю нагляду Співпраця між членами команди складна, особливо коли вони не знаходяться разом
Структурований процес розробки	Scrum зустрічі; щоденні наради, планування спринту та оглядові наради іноді неефективні там, де вони проводяться занадто часто, або забирають занадто багато часу або складно налаштувати нараду Труднощі в оцінці роботи проєкту над успадкованим кодом
Екологічні	Погане залучення клієнтів і незрозумілі вимоги до продукту Індивідуальний внесок не визнається, і не існує жодних рекомендацій щодо точного вимірювання індивідуальних результатів
Інформаційні системи та технології	Відсутність комунікації між членами команди, які знаходяться разом, спричиняє дублювання у вирішенні проблем Нещодавно найняті члени команди, як правило, створюють помилки в коді через незнайомство з програмним забезпеченням

Автори [11] підкреслили, що одним з найважливіших факторів успіху проєкту є індивідуальна компетентність. Вони підкреслюють якості людей, які беруть участь у проєкті, де хороші люди завершать проєкт, а якщо члени команди не мають достатньо навичок, жоден процес не зможе компенсувати їх недолік. Це також підтверджують автори [7], де проблеми з людьми є найбільш критичними, і було сказано, що дуже важливо їх вирішувати, перш ніж прийняти та інтегрувати Agile-практики в проєкт. У цій статті представлено перелік перешкод на шляху успішного використання гнучких методів, а серед важливих проблем, які висвітлюються

стосовно людей, є їхні ролі, відповідальність і навички, а також їх здатність передбачати й бути обізнаними.

Автори [16] обговорюють загальні проблеми в Scrum. Однією з поставлених проблем є здатність команди дати оцінку зусиль у своїй роботі з розвитку, особливо коли вона виконується вперше. Більшість команд не можуть виконати поставлені завдання через погані навички аналізу та оцінки завдань. Коли це відбувається, команда прагне подовжити тривалість спринту, а не навчитися робити правильні оцінки. Це може спричинити проблеми у досягненні стабільного темпу, оскільки команда не зможе працювати належним чином через затримку у виконанні інших завдань у проєкті.

Наявність члена команди, який є скептиком щодо гнучкості, тобто він протистоїть гнучким методам, може мати величезний вплив на команду в цілому. Це пов’язано з тим, що гнучка команда в значній мірі покладається на довіру та обмін негласними знаннями для підтримки таких важливих практик, як парне програмування та спільне володіння.

Іншою важливою практикою в Agile-процесі є колективне володіння кодом. Результати дослідження, обговорені в [25] вказує, що колективне володіння кодом дає переваги з точки зору обміну знаннями в команді. Однак недоліком цього є те, що команда буде схильна вибирати найшвидше рішення, ігноруючи його якість, припускаючи, що вони не єдині, хто відповідає за якість коду чи інше. Якщо не призначати права власності на частину роботи, це може демотивувати команду в написанні коду, який відповідає стандартам або необхідним рівням якості. Інші виявлені проблеми працюють у стабільному темпі та з точністю оцінок. Ці проблеми пов’язані з такими ситуаціями, як наявність агресивних дат, необхідність працювати понаднормово, а також недооцінка завершення часу.

Колектив авторів [30] запровадив техніку управління ризиками, яку можна застосувати в процесах, що працюють у Agile. У статті також наводиться аргумент щодо того, що Agile керується ризиком, оскільки він неявно керує ризиком у процесі. Один із неявних методів, що використовуються, – це визначення пріоритетів завдань на початку ітерації. Проте просто присвоєння вищого пріоритету завданню з високим

рівнем ризику не вважається управлінням ризиком. Це зменшує ризик для проєкту, якщо пов’язане завдання виконується раніше, але поки ризик не буде вирішено або більше не буде застосовним, завдання потрібно відстежувати на предмет ризику та вживати заходів, якщо необхідно. Коли визначення правильного ризику для завдання та його аналіз не виконується належним чином, представити відповідний план для пом’якшення ризику важко [1, 41].

1.1 Підхід до вирішення проблеми управління ризиками

Як результат виявлених проблем, є сильна мотивація покращити управління ризиками в Agile проєктах, не зменшуючи гнучкість проєктів. Насправді, сучасне управління ризиками має бути в змозі інтегруватися в гнучкий процес для підтримки прийняття рішень. Це включає в себе врахування людських факторів, таких як навички та здібності розробника, а також їх поведінку під час виконання завдань.

У зв’язку з тим, що Agile в значній мірі покладається на компетенцію залучених людей, ми перетворили ці проблеми на фактори ризику, тобто ситуації або події, які можуть призвести до збитків і, отже, ми повинні відстежувати в проєкті, як показано на рис. 1 . Для того, щоб створити основу моделі ART щодо управління ризиками, модель називається ціле-спрямованою.

Модель управління ризиками розробки програмного забезпечення (GSRM), запропонована [22], була повторно використана. Ця модель використовується для визначення вхідних даних для проєкту, які складаються з типу ризиків та індикаторів ризику, а також даних про навколишнє середовище, які можна використовувати для визначення ризиків для проєкту.

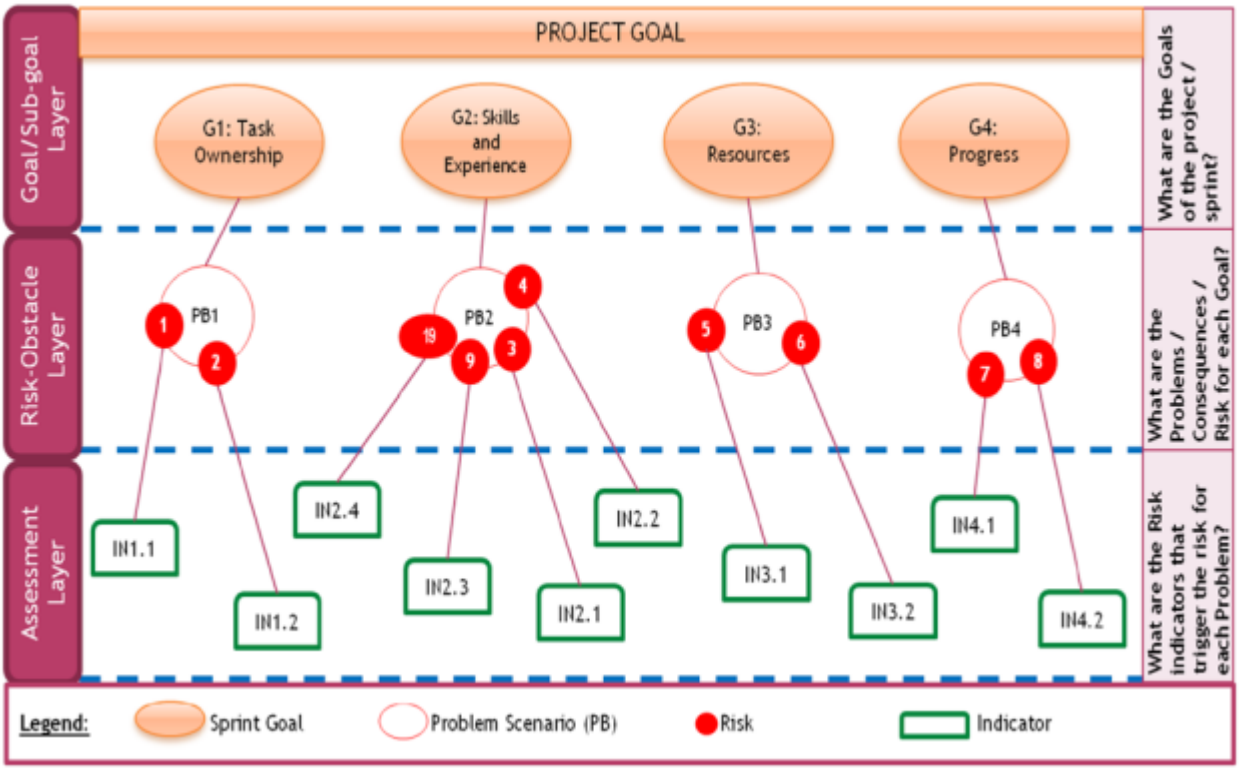


Рисунок 1.2 – Модель ART

Таким чином обговорені раніше питання перетворюються на набір цілей спринту. Пізніше вони будуть використовуватися для визначення ризиків та їх індикаторів, що дозволить безперервно відстежувати ризики.

У наступному розділі буде розглянуто розвиток моделі ART і те, як вона використовується для динамічного управління ризиками.

1.2 Модель Agile Risk Tool (ART)

Розробка моделі ART розпочалася зі встановлення уявлення про те, як управління ризиками може застосовуватися в гнучкому середовищі. На рисунку 2 нижче зображено огляд отриманої моделі.

1.2.1 Вхідні дані

Модель показує, як ризики збираються та керуються в рамках гнучкого проєкту. На етапі введення гнучкий процес починається з планування та збору вимог.

На цьому етапі, під час підготовки проєкту, водночас може розпочатися збір даних про ризики.

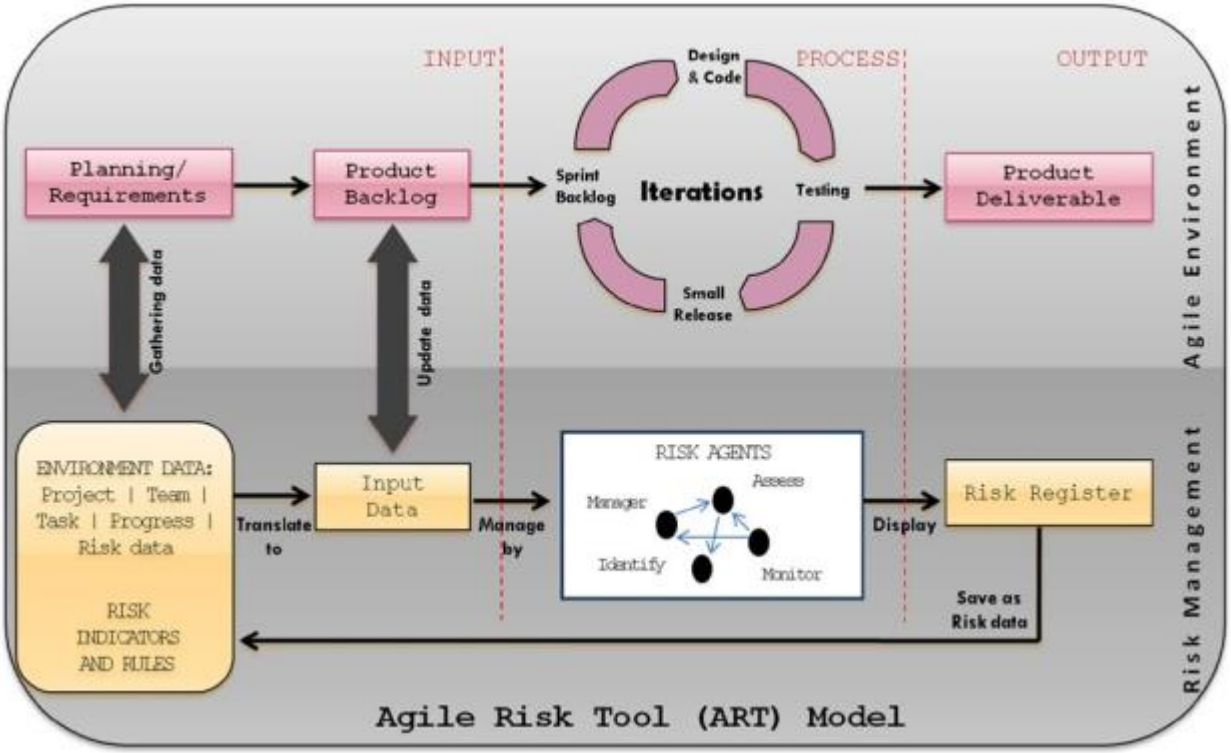


Рисунок 1.3 – Модель інструменту Agile для ризиків (ART), що описує застосування управління ризиками в середовищі Agile

Вимоги в Agile процесах найчастіше представлені у вигляді історій користувачів. Це текстові описи, які містять уточнення замовником потреб у необхідній системі. Відставання продукту – це підмножина цих вимог, які будуть вибрані на основі пріоритету.

Використані дані середовища містять:

– Проєкт у цьому контексті – це набір історій користувачів, членство в яких не фіксується на жодній точці його життя. Кожен проєкт пов’язує унікальну назву проєкту, набір цілей проєкту, коли він почався і коли закінчився.

– Команда – це набір осіб, де кожна людина складається з набору атрибутів, що описують людину. Кожна команда працює над досягненням цілей проєкту. Для кожного члена команди є конкретна інформація, наприклад, про тип навичок, якими

володіє член команди, а також його рівень знань у визначених навичках, зазначений у вигляді цілого числа;

– Історії користувачів поділені на завдання. Завдання відноситься до текстового опису завдання, пов’язаного з передбачуваними годинами виконання, ім’ям відповідального за завдання та прогресом виконання завдання;

– Прогрес відноситься до додаткової інформації про хід виконання конкретного завдання, яку повідомляє особа, відповідальна за виконання завдання. Сюди входить інформація про присутність члена команди на щоденній зустрічі Scrum і про те, чи повідомляється про прогрес чи перешкоду для виконання завдання;

– Дані про ризики представляють інформацію про ризики, отримані інструментом. Інформація містить назву ризику, його серйозність, власника ризику, місце розташування ризику, а також дату виникнення та усунення ризику.

Індикатори та правила ризику відносяться до набору заздалегідь визначених факторів ризику, які команда проводить мозковий штурм на ранній стадії проєкту та закодована як правила (про це детальніше буде розказано в наступному підрозділі). Індикатори ризику містять текстовий опис, що вказує на поріг або стан, який спричинить ризик. Одним із прикладів може бути, коли завдання з високим пріоритетом вибирається в спринті розробником із занадто низьким попередньо визначеним порогом навичок. Правила містять список умов для події, закодованих у операторах IF/THEN. Пізніше ця інформація зберігається в механізмі правил. Вхідні дані відносяться до набору зібраних даних із середовища та переведених у набір шаблонів, які можна читати інструментом.

На етапі процесу проєкт виконується у вигляді ітерацій, які включають відставання в спринту, дизайн і код, тестування та невеликі випуски вимог до продукту. Ітерації містять фіксовану довжину за часом розробки. Агенти ризику (або агенти ART) керуватимуть ризиком на основі вхідних даних, визначених раніше. Цей процес ризику є автономним, де програмні агенти; виявлення, оцінка та моніторинг ризику на основі вхідних даних із навколишнього середовища. Щойно будь-який ризик спрацьовує, дані про ризики відображатимуться в Реєстрі ризиків. Будь-які

зміни чи оновлення середовища вплинуть на дані про ризики (незалежно від того, позначено ризик чи ні).

На етапі вихідних даних остаточні дані про ризики можна отримати після доставки продукту та під час зустрічі з огляду Sprint. Реєстр ризиків забезпечує перегляд усіх ідентифікованих даних ризиків. Зрештою, дані, відображені в Реєстрі ризиків, можуть бути записані та збережені в сховищі даних ризиків, де цю інформацію можна використовувати для планування майбутніх проєктів.

Модель була продемонстрована далі та використана як частина роботи [34]. Саме тут було продемонстровано запропоновану архітектуру ART для вивчення застосування управління ризиками в гнучких додатках. Цей документ доповнює це, зосереджуючись на розробці ART-агентів, що використовуються на стадії процесу.

Одним із способів перейти до автоматизації є надання програмним агентам відповідальності за виявлення, оцінку та моніторинг ризиків. Ці агенти в ідеалі повинні мати можливість автономно реагувати на зміни середовища, коли середовищем в даному випадку є середовище розробки програмного забезпечення, включаючи набір інструментів, що використовуються.

Щоб зменшити бар’єри у застосуванні управління ризиками, необхідний легкий підхід до управління ризиками. Новопропонований підхід включає три основні кроки в управлінні ризиками; ідентифікація ризиків, оцінка ризиків та моніторинг ризиків. Обґрунтування цього полягало в наступному:

1) розробці реалістичного та прийнятного процесу управління ризиками, який може вписуватися в гнучкі методи;

2) емпіричне дослідження [34] підтвердило, що найскладнішими кроками в управлінні ризиками є ідентифікація ризиків та моніторинг ризиків.

Крім того, до цього розділу були встановлені докази, які стверджували, що управління ризиками було складним головним чином через необхідні людські зусилля.

З огляду на це, мета полягає в тому, щоб замінити частину людської участі автономними програмними агентами з метою, щоб вони могли керувати ризиками та мінімізувати потребу в ручному введенні даних. Таким чином, автоматизовані агенти

можуть допомогти полегшити робоче навантаження в управлінні ризиками, зокрема щодо виявлення, оцінки та моніторингу ризиків.

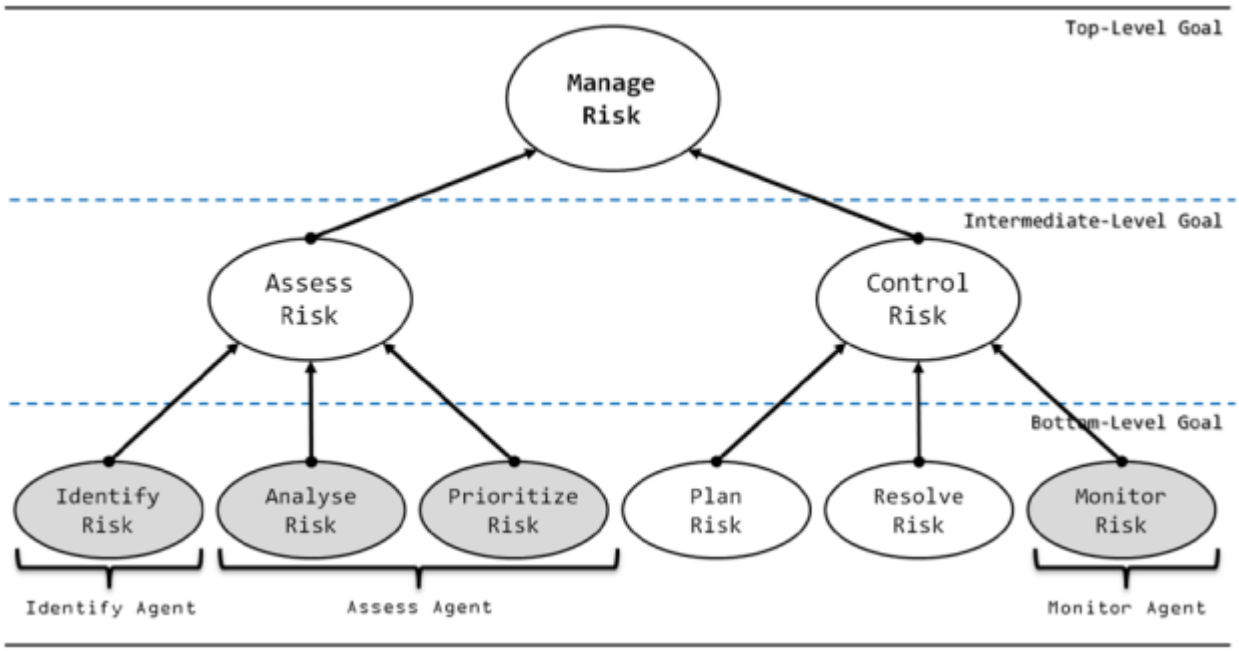


Рисунок 1.4 – Графік декомпозиції ризиків
для агентів інструменту Agile ризиків (ART) для чотирьох видів діяльності з управління ризиками

Розклад ризиків на види діяльності є звичним явищем. Використовується один приклад, розглянутий у [24] декомпозиція ризику на концептуальні елементи, такі як фактор ризику, подія ризику, результат ризику, реакція на ризик, ефект ризику та втрата корисності. Зовсім недавно техніка розкладання цілі зверху вниз описана в [8] та [15]. Справді, в [5] розкладає управління ризиками на види діяльності. У цій роботі категорія або тип використовуваних агентів були отримані на основі початкової декомпозиції цілі агента, як показано на рисунку 1.3 , на основі роботи Боема.

Загальною метою цієї роботи є пошук шляхів зниження бар'єрів для застосування ризик-менеджменту. Однією з цілей є використання агентів, оскільки поведінка агента є більш адаптивною і може діяти від імені менеджера проєкту Agile проєкту. У цьому випадку частина зусиль керівника проєкту замінюється виконанням агента, щоб він реагував автоматично відповідно до власних цілей. При визначенні

цілей для агентів, ціль верхнього рівня починається, щоб застосувати управління ризиками в проєкті розробки програмного забезпечення, особливо в проєктах, що працюють у Agile. Ця ціль далі розкладається на дві проміжні підцілі; оцінка ризику та контроль ризику.

Ці підцілі потім розкладаються на шість менших підцілей; визначати, аналізувати, розставляти пріоритети, планувати, вирішувати та контролювати. У результаті декомпозиції цілі агенти були призначені на основі найменших підцілей, які підтримували ціль верхнього рівня. Оскільки найбільш інтенсивними кроками, визначеними раніше, були ідентифікація та моніторинг, тим часом обидві підцілі були обрані на додаток до аналізу та визначення пріоритетів цілей, як показано на рисунку 1.3. Варто зауважити, що тут задіяні лише цілі нижнього рівня; припущення полягає в тому, що цілі верхнього і середнього рівнів можуть мати в значній мірі контрольну функцію, але, тим не менш, мають власні цілі щодо того, як агенти нижчого рівня повинні взаємодіяти.

Подальші ART-агенти були розроблені для цієї роботи як чотири агенти; менеджера агента, ідентифікувати агента, оцінити агента (поєднує аналіз і визначення пріоритетів цілей) і моніторинг агента. Це зображено, як на рисунку 1.4, де показано взаємодію (зв'язок через передачу повідомлення) між агентом менеджера та агентами ідентифікації, оцінки та моніторингу. В залежності від того, як агенти реагують на дані із середовища, щоб динамічно виявляти ризик за допомогою виконання правил, де правила викликаються з механізму правил. Спілкування агентів ART описано нижче.

Є чотири агенти ART, і кожен з них має визначену мету. Мета та призначення кожного з них обговорюються нижче.

– Агент менеджера виступає посередником між трьома іншими агентами. Він керує та виконує правила, отримує дані із середовища та повідомляє агента Identify, якщо спрацьовує будь-який ризик.

– Ідентифікувати агента повідомляється, якщо виникає будь-який ризик. Він запитує в агента менеджера, який ризик був виявлений, і повідомляє агента з оцінки.

– Агент оцінки викликає агент Identify, і його мета полягає в оцінці ризику (Risk Evaluation – RE) ідентифікованого ризику, де RE = ймовірність (P) × вплив (I). Виявлений ризик буде оцінено як високий, середній або низький, і агент моніторингу буде сповіщений, щоб він вжив наступних заходів.

– Агент моніторингу викликає агент оцінки з деякими даними: RE та ранг ідентифікованого ризику. Агент моніторингу встановить місцезнаходження ідентифікованого ризику разом із власником ризику. Потім ці дані відображаються в Реєстрі ризиків, який пізніше можна записати та зберегти в сховищі даних ризиків.

1.2.2 Процес управління ризиками

На етапі процесу агенти ART будуть контролювати ризик, визнаючи будь-які правила або індикатори ризику ініціюється відповідно до шаблону ART. Агенти ART ініціюють спілкування між ними. Повідомлення передаються відповідно до запиту, і кожен агент сповіщає іншого агента, пропонуючи будь-які подальші дії. Приклад спілкування агентів ART був представлений раніше в цій главі (див. рис. 1.5).

На рисунку 1.6 показано взаємодію між агентами ART, що починається, коли виникає ризик. На малюнку показано, як агенти передають повідомлення за допомогою агента Sniffer на платформі JADE. Відповідаючи своїй назві, sniffer agent – це чисто java-додаток, який відстежує повідомлення в середовищі JADE. Це корисно під час налагодження поведінки агента та для аналізу передачі повідомлень за допомогою графічного інтерфейсу аналізатора [4].

Правила та дані середовища можна динамічно редагувати. У випадку, коли необхідно внести зміни, можна змінити дані середовища (які були раніше переведені в шаблон ART), а також правила ризику та індикатори, використовуючи надану область головного екрана. З іншого боку, розробляючи можливі ризики, пов'язані з правилами та індикаторами ризику, можна виявити, що раніше даних про навколишнє середовище було недостатньо для виявлення певних ризиків. У деяких випадках невелика зміна в зборі даних про навколишнє середовище дозволить визначити або виявити більше ризиків.

Наприклад, додавання інформації про навички розробника дозволить контролювати можливості програмування розробника, особливо під час виконання завдань високого пріоритету. Приклад синтаксису правила, який можна використати: «ЯКЩО рівень кваліфікації розробника «низький» І розробник, залучений до завдання «високого» пріоритету, ТОДІ Є ймовірність, що завдання не може бути виконано вчасно через погані навички програмування розробника».

Агенти ART динамічно реагуватимуть на вхідні дані, оброблятимуть вхідні дані, оцінюючи будь-який ініційований ризик, і вироблятимуть результат ризику в Реєстрі ризиків.

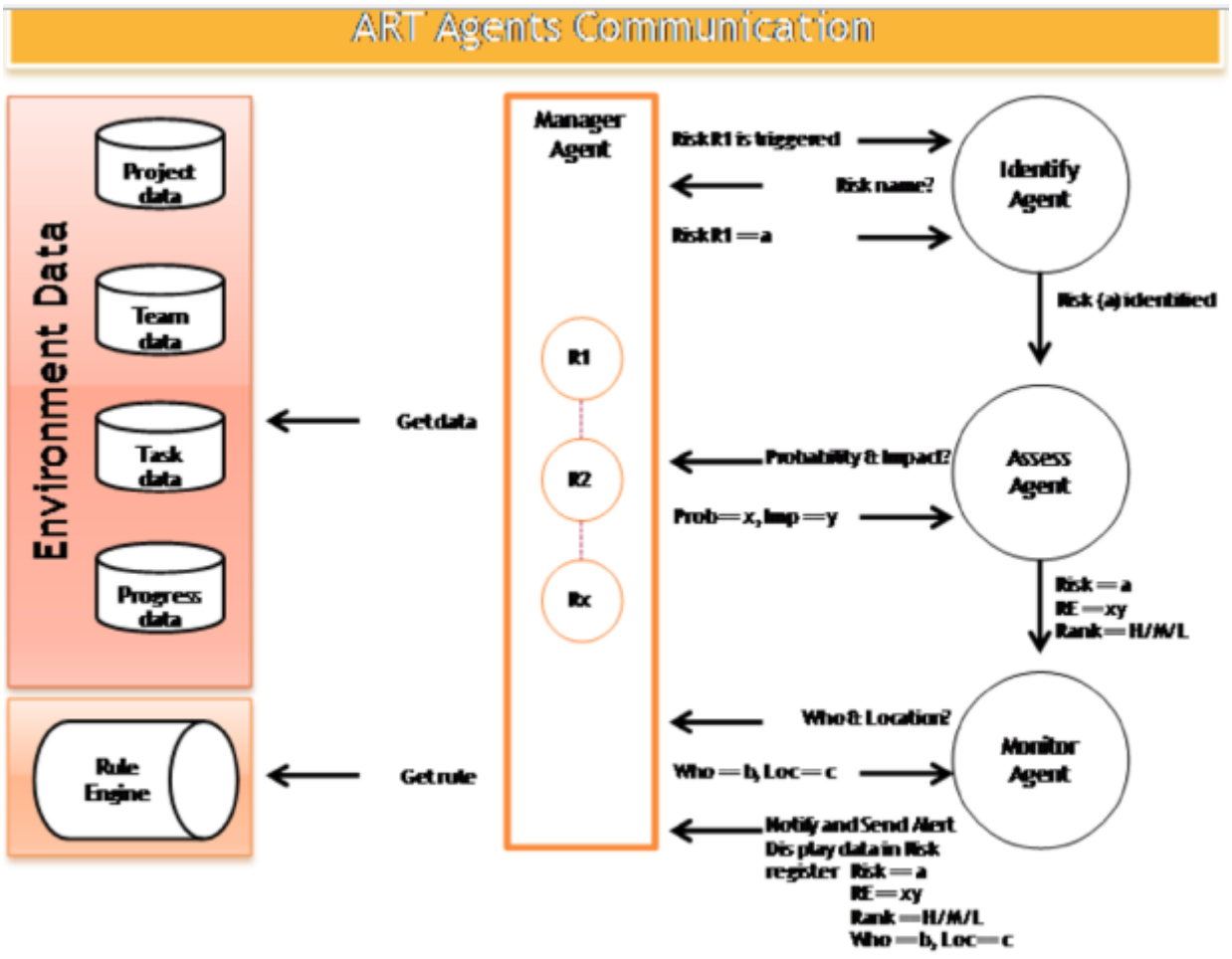


Рисунок 1.5 – Зв'язок між агентами ART і як вони взаємодіють із даними середовища та механізмом правил

1.3 Результати реєстрації ризиків

Ідея реєстру ризиків була визначена методика[40], яка стверджує, що «реєстр ризиків виконує дві основні ролі. Перший – це сховище сукупності знань… Друга роль реєстру ризиків – ініціювати аналіз і плани, які з нього випливають». У той час як [35] повідомили, що дуже мало розробки та створення реєстрів ризиків, хоча вони зазвичай використовуються в управлінні ризиками.

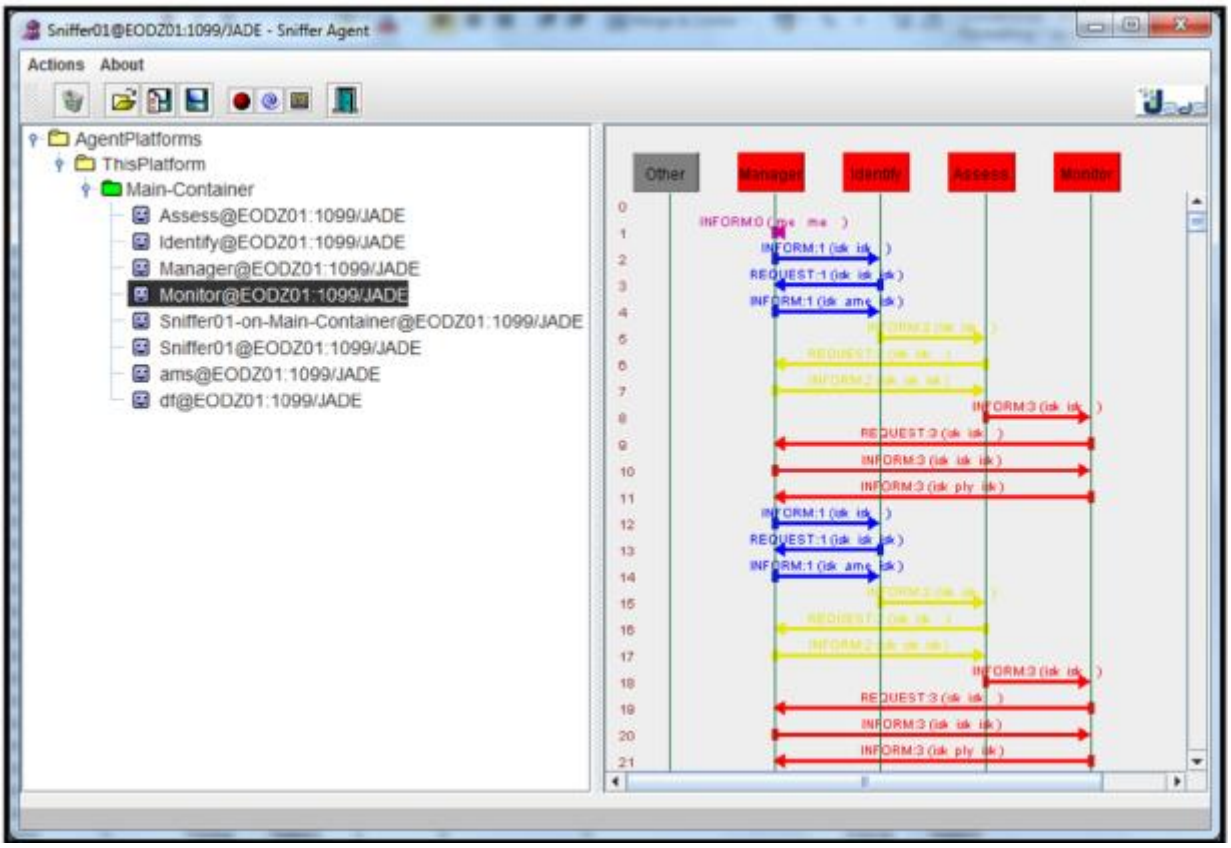


Рисунок 1.6 – Sniffer агент

Таким чином, реєстр ризиків, розроблений у цій роботі, може представляти собою панель ризиків, на якій можна побачити список ризиків, ініційованих агентами ART. На рисунку 1.7 показаний приклад регістру ризиків, який використовується для візуалізації результатів у цьому інструменті.

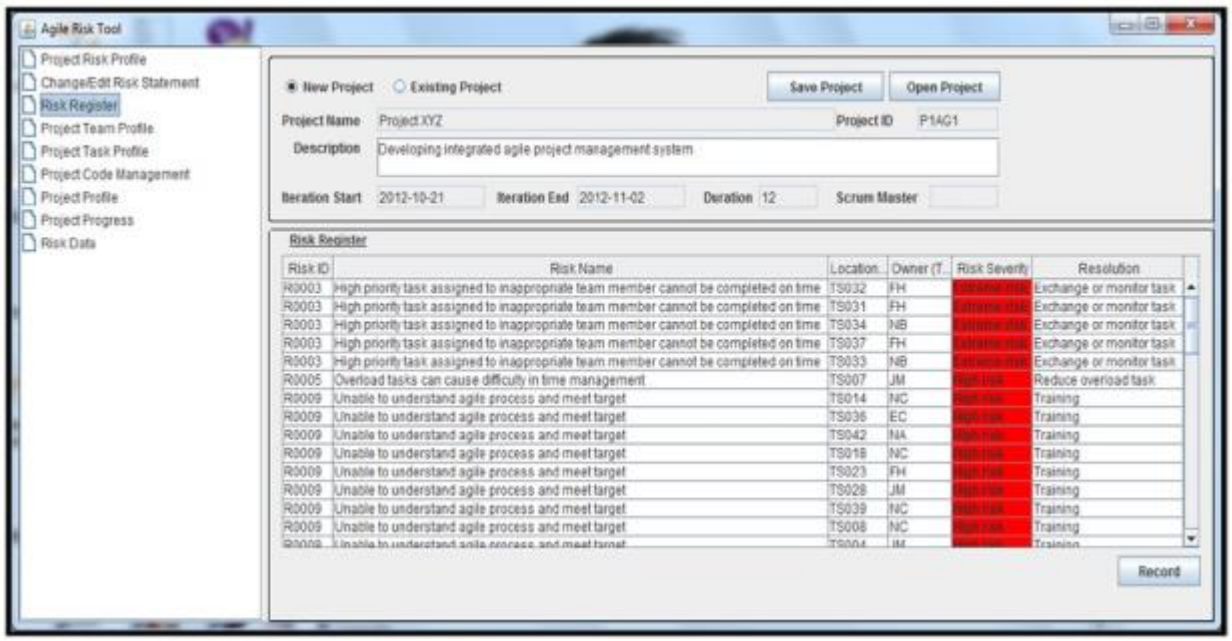


Рисунок 1.7 – Інструмент Agile для ризиків. Реєстр ризиків

Нижче наведено огляд основного алгоритму з реєстрації ризиків.

Лістинг 1.1 – Псевдокод алгоритму реєстрації ризиків

```
Ввід даних
env_data = Елементи Environment_data (проєкт, команда, завдання, прогрес, ризик), отримані за допомогою інструментів.
// Тут ми будемо посилатися на загальний тип env_data.
risk_indicator[ ] = набори порогових значень для кожного типу даних середовища.
//Між кожним типом env_data та індикатором ризику існує зв’язок 1:N. Кожен індикатор ризику являє собою вимірювання, яке можна відчути в середовищі для цього елемента даних середовища.
risk_rules[ ] = Визначений користувачем набір правил, який визначає для кожного індикатора ризику, як реагувати, коли значення env_ data відповідає/перевищує/падає нижче risk_indicator. Існує зв’язок 1:N між індикатором_ризику та правилом_ризику[i].
Вивід даних
risk_register[ ] = набір ризиків, де ризик – це текстова заява та концептуальна цінність (висока, середня, низька).
Панель інструментів = відображення ризиків
```

Функції.

env_data_onChange(): ця функція існує для кожного елемента env_data і запускається, коли відчутне значення змінилося;

checkRiskCondition(env_data) – це функція, яка порівнює значення env_data з відповідним індикатором і повертає набір застосовних правил із набору правил, risk_rules[ ];

applyRule() викликає будь-які необхідні дії (наприклад, сповіщення електронною поштою);

updateRegister() додає або оновлює ризик у реєстрі ризиків[ ];

updateDashboard() повідомляє інформаційну панель про нові/змінені/видалені ризики.

2 ТЕМАТИЧНІ ДОСЛІДЖЕННЯ

Для того, щоб показати, що запропонований підхід можливий, були проведені тематичні дослідження з використанням артефактів навчальних проєктів студентів.

2.1 Методологія дослідження проєктів

Ця робота має на меті відповісти на такі дослідницькі запитання (RQs):

RQ1: Чи можна підтримувати управління ризиками в гнучких проєктах, використовуючи наявні дані середовища SE для подолання бар'єрів у застосуванні управління ризиками?

RQ2: Чи можна збирати дані з мінімальним втручанням і зусиль?

RQ3: Чи можуть програмні агенти в поєднанні з механізмом правил надати засіб для автоматизації управління ризиками у гнучких проєктах з використанням даних із середовища розробки програмного забезпечення?

Для вирішення цих проблем були використані тематичні дослідження з використанням запропонованого підходу до вирішення та підтримки інструментів. Вибір тематичного дослідження базувався на тому, що розробка підходу була досить новою та дослідницькою, що включала велику кількість змінних; тому потрібен був гнучкий і природний метод. Автор роботи [17] оголосив тематичні дослідження придатними для попередніх досліджень деяких явищ з метою розробки нових гіпотез і побудови теорій, а також підтверджуючих тематичних досліджень для оцінки існуючих теорій. Тематичні дослідження можуть забезпечити більш глибоке розуміння досліджуваних предметів, і отримані результати є цінними, а також сприяють набору знань [23].

Робота [17] сприяє дослідженню тематичних досліджень, де використовується цілеспрямована вибірка, таким чином залежно від характеру цілей дослідження. Ситуаційне дослідження підходить, коли необхідні дослідження, щоб отримати глибоке уявлення про ланцюги причин і наслідків. Крім того, дослідницькі дослідження є доречними там, де є незначний контроль над змінними в дослідженні.

Все варто починати з дослідження проблем і проблем управління ризиками в Agile проєктах, щоб отримати глибше розуміння явищ, а потім запропонований підхід до вирішення. Пізніше підхід до вирішення перевіряється за допомогою розробленого інструменту-прототипу з використанням тематичних досліджень з позначками CSA і CSB. Перший приклад, Case Study Alpha (CSA), вважався необхідним для вивчення проблемної області та мав попередній характер, в основному призначений для розробки дослідницьких питань вище.

Другий приклад, Case Study Beta (CSB), був проведений як підтверджувальний приклад і використовувався для оцінки існуючих теорій та результатів, розроблених на основі CSA. Проте обидва тематичні дослідження були спрямовані на підтвердження підходу до рішення та підтримки інструментів із вдосконаленням CSB на основі уроків, отриманих від CSA. Були використані змішані методи, включаючи

1) неформальне інтерв'ю з власником продукту для підтвердження результатів інструменту-прототипу та

2) аналіз артефактів або архівів, щоб зрозуміти дотримання практичних методів у команді, а також продемонструвати результати, отримані від інструмент-прототип.

Аналіз артефактів або архівів відноситься до дослідження даних проєкту, яке включає електронну таблицю інструменту Agile Project Management, який вони використовували для збору даних, протоколів SVN, щоб визначити закономірності поведінки команди розробників.

Кейс-дослідження було проведено на групах студентів, яким було доручено розробити програмний проєкт і використали їхні дані як вхідні дані для запропонованого підходу. Багато фахівців-практиків, які займаються гнучкими проєктами, стверджують, що гнучкі методи по суті знижують ризик, але, як відомо авторам [7], було проведено дуже мало досліджень, щоб підтвердити це або зв'язати ці дві області. Проте впроваджувати щось нове в організації важко і, ймовірно, буде дорого коштувати. Автори [29] використовують термін «посуха даних» для опису ситуації, коли організації не бажають ділитися своїми оперативними даними через, серед інших факторів, чутливість бізнесу, пов'язану з даними. З огляду на труднощі отримання промислових даних у поєднанні з доступністю даних студентських

проєктів в університетських умовах, проєкти студентських груп використовувалися для демонстрації та підтвердження підходу. З іншого боку, використання даних студентських проєктів посилило підхід завдяки доступу до унікального набору даних, який недоступний у жодному іншому середовищі. Це включає доступ до даних для набору паралельних Agile команд, які виконують ідентичні набори історій користувачів. Цей сценарій було б дуже важко знайти або створити в подібному дослідженні в іншому місці, і практично неможливо в промислових умовах.

2.2 Вибір даних для ілюстрації методу

У дослідженні були використані дані з курсу з відзнакою останнього курсу бакалаврату з Agile Methods, який викладався в одному з університетів Європи в 2011 та 2012 роках. У теоретичній частині курсу студенти читали лекції про загальні практики Agile-розробки з акцентом на Scrum. Під час курсу студенти повинні були створити великий програмний артефакт з використанням технологій Microsoft.NET, використовуючи промислове потужне середовище, використовуючи як гнучкі проєкти, так і методи розробки програмного забезпечення. Це включає в себе застосування важливих методів управління проєктами Scrum, таких як парне програмування, розробка на основі тестування, планування випуску та ітерації та рефакторинг у своєму програмному проєкті. Перший кейс був розроблений у 2011 році за участю 38 студентів бакалавра, розподілених на шість груп по шість або сім розробників у кожній. Усі групи повинні були розробити однакові вимоги до продукції. Перше дослідження було розроблено в 2011 році (CSA) з групами, позначеними від Alpha1 до Alpha6 (Alp1–6). Другий кейс був розроблений у 2012 році (CSB) і охоплював загалом 56 студентів бакалаврату з восьми груп, від Beta1 до Beta8 (Bet1–8), кожна група складалася з п’яти до восьми розробників. Усім групам були надані ті самі вимоги до продукту, що й у першому дослідженні.

Однак через відсутність деяких даних у групі Bet8 цю групу виключили з дослідження, залишивши лише 48 студентів, які взяли участь у цьому дослідженні. Проєкти в обох тематичних дослідженнях включали два спринти, SP1 та SP2, які мали

відповідно принаймні 10-15 робочих днів. Перед початком проєкту Власник продукту, роль якого відіграє член академічного персоналу, представив елементи відставання продукту, які складаються зі списку пріоритетних історій користувачів. Згодом кожна група була відповідальною за доставку їх до задоволення Власника продукту. Власник продукту та студенти мали змогу спілкуватися щодо будь-яких проблем, що виникають під час обох спринтів; тому групи контролювалися протягом усього процесу розробки. Як описано вище, приклади були використані для демонстрації потоку процесу ART (ДИВ. рис. 1.2), який містить основні етапи на етапі входу, процесу та виходу. Найважливішою частиною процесу є визначення даних про навколишнє середовище та правил ризику для проєкту. Це детально розглянуто в наступних розділах.

2.2.1 Дані середовища

На початку цієї роботи було вивчено два інструменти управління проєктами; Extreme Manager та програмний продукт Rally з метою визначення можливих даних про середовище, які можуть бути доступні в реальному сценарії та можуть бути використані в цій роботі. Дані про навколишнє середовище були класифіковані таким чином:

– Дані проєкту – містить інформацію про проєкт, тобто назву проєкту, дату початку та дату завершення.

– Дані команди – містять інформацію про членів команди, тобто навички та досвід.

– Дані про завдання – містить інформацію про історії користувачів і розбивку завдань, пов’язаних із приблизними годинами.

– Дані про виконання – містять інформацію про звіти команди про хід виконання завдання.

Після того, як ця інформація була отримана з досліджуваних інструментів, було сфокусовано чотири категорії даних, описаних вище. Після запуску першого тематичного дослідження було створено шаблон ART для збору даних з цих

категорій. Зібрані дані студентських проєктів були, наскільки це можливо, відібрані та узгоджені, щоб відповідати загальним випадкам із досліджуваних інструментів.

Таблиця 2.1 – Дані про середовище від досліджуваних проєктів

№	Атрибути даних про середовище	Значення	Використо-вується	Репозиторій
1	Ідентифікатор проєкту	Унікальний номер проєкту	✓	Дані проєкту
2	Статус проєкту	Не розпочато, триває, завершено	✓	Дані проєкту
3	Назва команди/ідентифікатор	Ім'я члена команди або унікальний номер	✓	Дані команди
4	Роль	ScrumMaster /розробник	✓	Дані команди
5	Загальна кількість призначених ролей	1 або 2 ролі	✓	Дані команди
6	Загальна кількість залучених проєктів	1 або більше проєктів	✗	Дані команди
7	Командні навички	Програмування (C#)	✗	Дані команди
8	Agile досвід	Правда/неправда	✓	Дані команди
9	Agile рівень	Дуже добре, добре, середньо, погано, дуже погано	✓	Дані команди
10	Рівень майстерності	5 (найвищий навик) до 1 (найнижчий навик)	✓	Дані команди
11	Назва/ідентифікатор завдання	Назва завдання або ідентифікатор	✓	Дані завдання
12	Пріоритет завдання	Високий, Середній, Низький	✓	Дані завдання
13	У парі	У парі чи без пари [“]	✓	Дані завдання
14	Повністю у власності	1, 2 або більше розробників	✓	Дані завдання
15	Розрахункові години	Кількість годин	✓	Дані завдання
16	Відвідував щоденну зустріч	Так ні	✓	Дані про хід

Незважаючи на те, що студентські проєкти не використовували жодного з досліджуваних інструментів, ті самі дані середовища були доступні в їхньому середовищі іншими засобами. Резюме можливих даних про середовище, які можна зібрати, спрощено в наступній таблиці 2.1.

Зауважте, що є два доступні елементи даних, які не були використані в цьому дослідженні; Загальна кількість залучених проєктів, а також командні навички, оскільки студенти в тематичних дослідженнях були задіяні лише в одному проєкті за раз, і всі студенти виконали необхідні навички програмування, які в даному випадку були необхідністю досвіду C#. Передбачається, що в промисловому проєкті ці дані також використовуватимуться для визначення ризиків.

2.2.2 Правила ідентифікації ризику

У попередніх дослідженнях дослідницькі проблеми та проблеми в Agile-проєктах обговорювалися та перетворювалися на набір проблемних сценаріїв, які потім були представлені на рис. 1.1. Кожен сценарій проблеми представляє можливу подію ризику, яка пов’язана з метою спринту для проєкту. Ціль Sprint є важливою, оскільки її можна використовувати для розгляду того, як значення даних середовища можуть бути використані як індикатори загроз для цих цілей, тобто тригерів для ризиків. Тому пропонується сформулювати правила ризику з використанням індикаторів ризику для визначення подій, які спричиняють збитки (затримка/додаткові витрати/втрата вартості), тобто ризики, що призведе до ситуації, коли ідентифікація ризику може бути автоматизована.

У таблиці 2.2 нижче показано набори правил ризику та індикаторів ризику для проблеми володіння завданням. Ця проблема є загрозою успіху досягнення мети.

Решта цілі проєкту; G2: Навички та досвід, G3: Ресурси та G4: Прогрес, далі в цій роботі не обговорюються. Однак створені правила ризику, пов’язані з цілями, описаними раніше на рис.1.2, були зіставлені з цілями наступним чином (Таблиця 4). У кожному випадку ризик представляє собою висновок з питань ризику, пов'язаних з людьми, і за допомогою шаблону правила, представленого в таблиці 1.

Раніше була розроблена модель ART GSRM, яка показує зв'язок між метою, ризиком-перешкодою та оцінкою. Використовуючи цю модель, наступна таблиця шаблонів правил (таблиця 2.2) показує один із наборів цілей, сценаріїв проблем, правил та індикаторів ризику, які використовуються в цій роботі.

Таблиця 2.2 – Шаблон правила для володіння завданням

Ціль	G1: У спринті X завдання Y слід призначити відповідній кількості розробників після запуску Sprint X
Проблемний сценарій	PB1: під час спринтів розробник не має жодної пари або має занадто багато партнерів з програмування для обраного завдання
Наслідки	Уникайте «один експерт» або занадто багато розробників, які обмінюються кодом
Показники	IN1.1: проєкт розпочато, і коли завдання вибрано в спринті, «завдання в парі» порожнє – вказує на високий ризик IN1.2: проєкт запущено, а вибране завдання належить більше ніж 2 розробникам – вказує на низький ризик
Репозиторій/дані	Дані проєкту Дані завдання
Правило(а)	RL1.1: якщо Project.Project_Status = [Виконується] І Якщо Task.Paired_By = [“”] RL1.2: Якщо Project.Project_Status = [Виконується] І Якщо Task.Total_Owned > 2
Ідентифікатор та назва ризику	RN1.1: парне програмування R0001 RN1.2: право власності на завдання R0002

Таблиця 2.3 – Зіставлення цілей для зв’язування ідентифікатора ризику, що використовується в прикладі альфа (CSA) і прикладі бета-версії (CSB)

Цілі	**Ідентифікатор ризику**
G1: володіння завданням	R0001, R0002
G2: навички та досвід	R0003, R0004, R0009, R0019
G3: ресурси	R0005, R0006
G4: прогрес	R0007, R0008

Таблиця 2.2 показує набір правил та індикаторів ризику для цілі G1, де, коли починається спринт, відповідна кількість розробників має бути призначена для конкретних завдань. Використання індикаторів, як правило, залежить від того, як керівник проєкту хоче позначити, що стан є ризикованим. Для цілі було обрано два індикатори на основі проблем ризику, зазначених раніше. Індикатор IN1.1 вказує, що як тільки статус проєкту був [У виконанні], а вибране завдання не має пари [“ ”], тобто нульовий або порожній рядок, то запускається ризик 0001 – «Парне програмування».

IN1.2 стверджує, що як тільки статус проєкту [В розробці] і Завдання належить більше ніж двом розробникам [>2], то запускається ризик 0002, «Власність завдання». Ці індикатори потім перетворюються на правила RL1.1 і RL1.2, які містять об'єкт, атрибути та значення атрибутів, які будуть підібрані агентами. Розділ сховища показує, де задіяні дані середовища для конкретних ризиків. Як згадувалося раніше, набір індикаторів і правил можна час від часу оновлювати залежно від того, як керівник проєкту вирішить визначити ризик. Одним із таких випадків є зміна правила парного програмування, що відбулася між CSA та CSB.

2.3 Приклад використання методу

Враховуючи, що підхід є новим і все ще знаходиться на стадії дослідження, вибране дослідження мало дослідницький характер, де воно спирається на збір наявних даних, які використовуються в проєкті, а не на поточний збір даних в ідеальній ситуації. Там, де це було можливо, використовувалися численні джерела доказів (тріангуляція), що означає, що для підтвердження висновків використовувалися архівні артефакти та неформальні інтерв'ю з власником продукту. Наприклад, після кожного збору даних проводилося неформальне інтерв'ю з власником продукту, щоб перевірити інтерпретацію на основі зібраних даних. Як згадувалося раніше, студенти не знали, що їхні дані проєкту оцінюються на предмет ризику, що усуває будь-яку можливу упередженість у цьому відношенні. Скоріше вони були мотивовані демонстрацією того, що вони дотримувалися практичних методів управління проєктами, наприклад, парного програмування, як навчали на заняттях, та створення високоякісного робочого програмного забезпечення. На основі даних, зібраних у CSA, було виявлено деякі проблеми при прийнятті підходу. Це було відзначено разом із висновками подальшого обговорення з власником продукту. Результати цього розслідування були зафіксовані у слідчих записках. Крім того, в одне правило, R0001, було внесено одну модифікацію, яке буде використано в наступному дослідженні.

У наступному розділі детально розглянуто перебіг процесу ART (див. рис. 1.2), як було проведено в двох тематичних дослідженнях. Обидва тематичні дослідження складалися з двох спринтів, так що процес повторювався ітераційно в чотирьох випадках.

2.3.1 Альфа версія прикладного дослідження (CSA)

Як описано раніше, потік процесу ART складається з трьох основних етапів: введення, процес і вихід. У цьому розділі цей процес детально описано.

Етап введення починається зі збору всіх необхідних даних з артефактів студентського проєкту та перекладу даних у відповідність до шаблону ART. Під час етапу введення були необхідні два вхідні дані – визначення переліку даних про навколишнє середовище та визначення правил ризику, використаних у цьому дослідженні. Для визначення списку даних середовища було виконано два кроки.

1. Збір даних

Для визначення переліку даних про середовище були використані архівні артефакти, отримані зі студентських проєктів. У цьому дослідженні було використано п’ять основних артефактів, узагальнених нижче.

– Таблиця Hartmann-Orona – добре відома електронна таблиця, яка переважно містить дані про відставання спринту, включаючи розбивку кожної історії користувача на завдання, приблизні години та власника кожного завдання.

– Sprint Backlog Target—містить список історій користувачів і пов’язаних точок і залежностей.

– Протокол наради Scrum – містить інформацію про відвідуваність команди та оновлення завдань.

Репозиторії коду – система контролю версій Subversion, яка використовувалася студентами для керування проєктом. Кожна група повинна була зареєструвати свою діяльність і перевірити всі документи та вихідний код.

– Вихідний код – студенти повинні були використовувати мови програмування C# та VB.Net

У таблиці 2.4 нижче наведено список витягнутих даних з архівованих артефактів даних про середовище.

Таблиця 2.4 – Дані середовища, отримані з артефактів студентського проєкту

№	Ім'я файлу	Дані доступні/зібрані
1	Електронна таблиця Хартмана-Орони	Інформація про відставання спринту Основна область завдань/історії користувачів Назва завдання Власник завдання Статус завдання (завершено/виконується/не розпочато) Розрахункові години Виділяє дні та години для кожного завдання Інформація про члена команди Ім'я та ініціали члена команди Робочі дні для цього спринту Робочий час для цього спринту Дата початку Кінцева дата Кількість календарних днів Діаграма щоденної активності члена команди Sprint Діаграма вигорання членів команди
2	Ціль відставання спринту	Список історій користувачів Бали за кожну історію користувача Залежності
3	Протокол зборів Scrum (щодня)	Ім'я скрам-майстра на день Хід роботи для кожного члена команди Робота виконана з учорашнього дня Роботи заплановані на сьогодні Проблеми
4	Репозиторії TortoiseSVN	Версійність каталогів і файлів Фіксувати файли/код Відстежувати зміни
5	Вихідний код (C#) інтегрується з resharper	Вихідний код групового проєкту Аналіз якості коду

Проте наявні в архіві артефакти не надали стільки даних, скільки досліджувані комерційні інструменти. Тим не менш, заархівовані артефакти містили достатньо корисної інформації, зокрема, пов’язаної з відставанням спринту та історіями користувачів, розбивкою завдань, деталями розробника, відповідального за завдання, тощо. Крім того, метою дослідження було продемонструвати підхід та підтримку інструментів, а не застосовність до кожного елемента даних, зібраного в основних інструментах.

Однією з проблем, яку потрібно було подолати, було відсутність даних у сховищі SVN. Хоча всі студентські групи повинні були зареєструвати свою роботу в репозиторії, деякі групи не зробили цього. Наприклад, усі групи повинні були щоденно записувати протоколи зустрічей, але деякі з них були відсутні в сховищах. Отже, власник продукту повинен був відстежити цей запис іншими методами, такими як архів електронної пошти, щоб отримати відсутні дані. Так само були певні невідповідності у форматі протоколів засідань. Наприклад, кожна група повинна вказати ім'я Scrum Master для кожної зустрічі. Проте в деяких групах ця інформація була відсутня. Знову власнику продукту довелося отримати цю інформацію через архіви електронної пошти. Усі виявлені проблеми були зафіксовані та записані в записках розслідування, щоб у майбутньому процес міг бути покращений.

Після того, як усі дані були визначені та впорядковані за категоріями, наступним кроком було перекласти ці дані вручну, щоб вони відповідали шаблону ART. На рисунку 2.1 показано приклад перекладу даних з архівованих артефактів у шаблон ART.

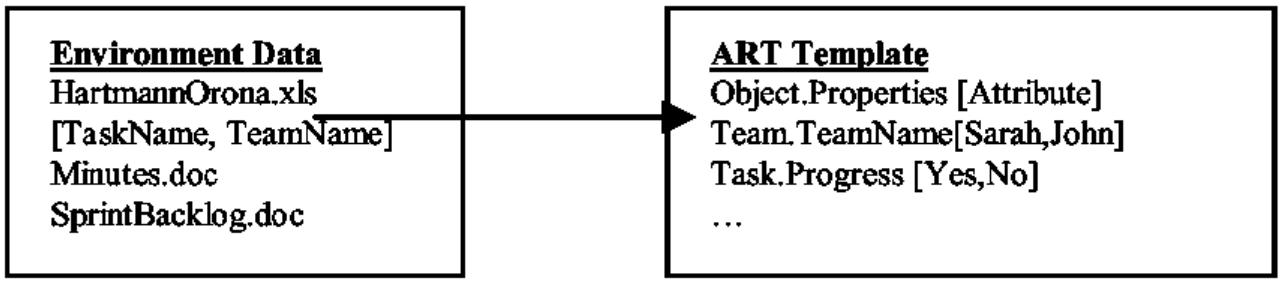


Рисунок 2.1 – Переведення даних середовища в шаблон ART

Це включає перетворення вихідних даних, отриманих з артефактів, у форму об'єктно-орієнтованих концепцій.

Це важливо для того, щоб агенти ART мали змогу збирати дані та узгоджувати їх із правилами, вбудованими в інструмент. Оскільки це дослідження проводилося після завершення проєкту, отримані дані були вичерпними, починаючи з першого дня спринту SP1 до 15 дня спринту SP2, і були готові до перекладу на шаблон. У випадку, якщо проєкт є новим або «свіжим», дані можна вводити або додавати безпосередньо

через інтерфейс користувача. Аналогічно, під час виконання цього кроку були також висвітлені проблеми. Основна виявлена проблема була пов’язана з процесом перекладу вихідних даних у шаблон. Оскільки це робилося вручну, це включало виснажливі та дуже трудомісткі завдання. У цьому дослідженні було шість студентських груп із шістьма різними наборами архівних даних. Незважаючи на те, що вони могли мати той самий формат або майже однаковий формат документа, інтерпретація та пояснення даних були різними. Щоб подолати це, часто доводилося ретельно переглядати всі документи в сховищах, щоб зрозуміти, як вони реалізували свій проєкт.

Крім того, проблема посилюється у випадку відстеження завдання, призначеного члену команди. Наприклад, у електронній таблиці Hartmann-Orona команда визначила історії користувачів і розбивку завдань. Сама електронна таблиця не передавала унікального ідентифікатора для кожного із завдань, хоча містила ім’я особи, відповідальної за завдання. Хоча в протоколі кожної зустрічі кожен член команди повідомляв оновлення щодо завдання, яке йому було призначено, це було описано лише коротко. У випадку, коли потрібно було визначити, хто з членів команди, який прийняв конкретне завдання, і був узгоджений з електронною таблицею, потрібно пройти через сховища та переглянути код, який член команди зробив. Час, витрачений на переклад даних однієї групи, становив 2–3 години, без урахування часу, витраченого на отримання відсутніх даних.

Далі були визначені правила ризику для цього прикладу. У таблиці 2.5 нижче наведено перелік назв ризику та правил, які обговорюються на початку цього розділу, а також оцінка ймовірності та впливу, визначена для кожного ризику. Зауважте, що правила були вбудовані в механізм Rule під час розробки інструменту прототипу ART, і на цьому етапі потрібно було лише вибрати існуючі правила з бази даних Rule engine. Однак, коли це було потрібно, нові правила були додані в механізм правил або існуючі правила відредаговані за допомогою наданого інтерфейсу користувача. Аналогічно, при введенні ймовірності та впливу параметри можуть бути відкориговані пізніше.

Таблиця 2.5– Перелік назв ризику разом із відповідними правилами та оцінкою ймовірності та впливу

Ідентифікатор ризику	Назва ризику	Правила [Object.Attribute] = [Value]	Пробний бал	Оцінка Imp
R0001	Парне програмування	PROJECT.PROJECT_STATUS = Виконано TASK.PAIRED_BY = “”	3	5
R0002	Власність завдання	PROJECT.PROJECT_ СТАТУС = Завершено TASK.TOTAL_OWNED > 2	1	1
R0003	Завдання високого пріоритету призначено невідповідному члену команди неможливо завершити вчасно	ЗАВДАННЯ.ПРІОРИТЕТ = Вищий КОМАНДИ.РІВЕНЬ НАВІНЦІВ = 1	5	4
R0005	Перевантажені завдання можуть спричинити труднощі в управлінні часом	PROJECT.PROJECT_STATUS = Виконано TEAM.TOTAL_NO_ROLE > 1	5	1
R0007	Розробник відсутній на засіданні можливої плинності працівників	PROJECT.PROJECT_STATUS = Завершено PROGRESS.DAILY_MEETING = N	1	3
R0008	Немає звіту про прогрес	PROJECT.PROJECT_STATUS = Завершено PROGRESS.PROGRESS_DETAILS = N	1	3
R0009	Неможливо зрозуміти Agile процес і досягти мети	PROJECT.PROJECT_STATUS = Завершено TEAM.AGILE = хибно	3	3
R0019	Неможливо дотримуватись Agile процесу	TEAM.AGILE = вірно TEAM.EXPERIENCE = Дуже погано	1	1

Для цього випадку значення ймовірності та оцінки впливу для кожного ризику було перехресно перевірено з Власником продукту. Оскільки це був студентський проєкт, фактичного впливу на вартість цього проєкту не було. Таким чином, імпакт-фактор був заснований на наслідках того, що студент не завершив проєкт і не виробив якісний кінцевий продукт, як того вимагає Власник продукту. У реальному проєкті визначений ризик буде більш специфічним для проєкту, іншими словами, ризики

оцінюються індивідуально для конкретної проєктної ситуації або середовища. Значний ризик проєкту може бути результатом певних характеристик у середовищі проєкту. Наприклад, розробник, який, як вважають, має дуже низькі навички, але йому призначено завдання з високим пріоритетом, може призвести до більш високого ризику, ніж у випадку, коли цьому розробнику призначено завдання з нижчим пріоритетом. Коротше кажучи, керівник проєкту визначає, який ризик є значним і наскільки серйозним.

Етап процесу – це етап, на якому автоматично відбулася оцінка ризику. На основі визначених вхідних даних, описаних раніше, агенти ART взаємодіють між шаблоном ART і механізмом правил. На цьому етапі, як тільки проєкт завантажено в прототип ART, зміни можна внести за допомогою наданого інтерфейсу користувача. Після того, як інструмент «запущено», агенти ART відреагують, якщо спрацює будь-яке з правил, а потім повідомить про виявлений ризик. Будь-які зміни у вхідних даних також призведуть до змін результатів ініційованого ризику. Це пояснюється тим, що ідентифікований ризик можна спостерігати на етапі вихідних даних, а проблеми ігнорування ризику можна уникнути.

На етапі вихідних даних після визначення ризику результат ризику відображається в Реєстрі ризиків. Тут також має бути представлений огляд усіх ініційованих ризиків. Сюди входить назва ризику, місце розташування ризику, пов’язаного із завданням, на яке впливає, і власника ризику, за дефолтом власника завдання. У реєстрі ризиків також відображався результат ризику відповідно до пріоритету, починаючи від ризику високого до низького ступеня тяжкості. Після завершення одного спринту результат ризику зберігається в сховищі даних ризиків.

Після завершення кожного спринту в CSA створювалися звіти. Представлені звіти надають корисну інформацію про загальну оцінку ризику щодня та за один спринт. Сюди входить інформація про розподіл ризиків, що виявляються щодня.

2.3.2 Бета-версія прикладного дослідження (CSB)

На основі досвіду CSA, нотатки розслідування виявили такі проблеми:

1. Дизайн проєкту: Оскільки проєкт був розроблений для студентів як частина університетського курсу, справжньою метою на практиці було те, щоб студенти застосували те, що вони навчилися під час курсу. Тому змінити структуру проєкту було неможливо. Крім того, через обмеження часу на завершення їх проєкту неможливо додати більше завдань управління. Тим не менш, збір даних мав бути простішим.

2. Формат документа: Для уникнення втрати даних були встановлені стандартні формати документів, які використовуються для збору даних у проєкті; тобто протокол засідання.

3. Умови найменування/відстежуваність: було вирішено, що для легкого відстеження даних між завданням та його власником потрібен унікальний ідентифікатор для кожного завдання та кожного члена команди. Наприклад, усі завдання повинні починатися з унікального ідентифікатора, який починається з «TS», наприклад TS001, а всі члени команди можуть мати унікальний ідентифікатор, який починається з «TM», наприклад, TM001.

4. Розподіл та оцінка завдань: на основі підсумку даних, зібраних про кожну групу проєкту, було виявлено, що деякі групи не змогли рівномірно розподілити завдання кожному члену команди. Деякі члени команди виконали занадто багато завдань, що призводило до того, що деякі завдання не були виконані. Крім того, було помітно, що деякі з передбачуваних розмірів завдань були занадто великими і їх слід було розділити на менші завдання майже однакового розміру. Також наголошувалося, що члени команди повинні практикувати парне програмування кожного разу, коли доручається більш велике завдання.

Визначення рівня навичок студента та досвіду спритності: під час спринту SP1 було запропоновано враховувати навички студента та досвід спритності. На цьому етапі визначити навички студента було відносно легко. Вказати Agile досвід є більш проблематичним. У SP1 передбачається, що всі студенти не мали жодного досвіду

маневрування, але їх досвід у швидкій роботі був виміряний у SP2 на основі оцінки викладача під час першого спринту.

Таблиця 2.6 – Назва ризику разом з пов'язаними з ним правилами та оцінкою ймовірності та впливу

Ідентифікатор ризику	Назва ризику	Правила [Object.Attribute] = [Value]	Пробний бал	Оцінка
R0001	Парне програмува-ння	PROJECT.PROJECT_ STATUS = Завершено TASK.PAIRED_BY = “” TASK.ESTIIMATE > 5	3	5

5. Правило ризику для парного програмування: результати ризиків були представлені власнику продукту, і один із найпоширеніших знайдених ризиків був пов’язаний з відсутністю парного програмування. З отриманих висновків було очевидно, що більшість студентів не дотримувалися цієї практики. Проте аргумент полягав у тому, що деякі завдання були замалі й не підходили для роботи в парах. Таким чином, важливо запропонувати змінити правило, де ця зміна описана далі в наступному прикладі – CSB.

На основі засвоєних уроків було зроблено покращення збору даних та покращення ефективності студентів у застосуванні методів, засвоєних під час курсу. Про це повідомили студентів. На цьому етапі передбачається, що обізнаність власника продукту про результати діяльності студентів зросла на основі уроків, отриманих від CSA. Крім того, очікується, що зміниться результативність студентської групи в цьому прикладі.

Під час етапу введення було дотримано тих самих кроків, які були включені в попередній приклад. Як описано раніше, через характер проєкту ми не змогли змінити структуру проєкту, тому були використані ті самі артефакти та методи збору даних із цих артефактів.

На етапі збору даних виявилося, що процес набагато легший, ніж у попередньому дослідженні. Наприклад, деякі студентські групи використовували нову конвенцію щодо імен, називаючи свої звіти, а одна група використовувала

формат запропонованого протоколу засідання. Як повідомлялося раніше, щодо деяких відсутніх даних з попередніх студентських проєктів і для цього прикладу була одна група, яка не зареєструвала свій основний документ. Цей документ не було відстежено, тому цю групу необхідно вилучити з цього дослідження. Хоча сприйняття зміни методу роботи в даному випадку досить погане, це можна поступово покращувати. Аналогічно, у реальному проєкті цілком нормально, що деякі організації можуть відмовлятися або відчувати себе викликом, коли їх просять змінити свої методи. Однак через обмеження, про які йшлося раніше, неможливо змусити студентів дотримуватися стандартів через обмеження, що це також є вправою для оцінювання.

Як обговорювалося раніше, труднощі у здійсненні процесу перетворення вихідних даних у шаблон під час етапу трансляції даних ART. Хоча виконання цього кроку все ще досить інтенсивно, час на переклад даних однієї групи було скорочено до менш ніж години. Особливо це стосується тих випадків, коли у звітах для деяких груп використовувалися стандартні правила найменування. Крім того, згодні, що реалізувати цей підхід з першого разу було досить складно. Це було значно покращено, і процесом було б легше управляти, якщо його застосовувати повторно.

Далі були застосовані правила ризику для цього прикладу. У попередньому прикладі було узагальнено перелік ризиків та пов’язаних з ними правил (Таблиця 2.6). На основі результатів та уроків, винесених із попереднього прикладного дослідження, правило парного програмування було змінено, щоб забезпечити більш реалістичне правило ризику. Можливість змінювати правила за потреби демонструє, що підхід до вирішення та підтримка інструментів динамічно реагує на зміни, таким чином, як це потрібно в гнучких проєктах. Це описано в таблиці 2.6, де показано змінене виділене правило ризику. Решта правил ризику залишилися без змін.

Після завершення етапу процесу та результату для двох спринтів знову були створені та представлені у вигляді діаграм звіти щодо інформації, зібраної щодо даних про ризик. Результати обох тематичних досліджень дали результати загального показника ризику (TRS).

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Дані про ризики, отримані за допомогою інструменту та відображені в Реєстрі ризиків, були записані та збережені в сховищі даних про ризики. Пізніше результати ризику були оцінені та представлені за допомогою графіків.

3.1 Узагальнення даних проєкту CSA

На етапі планування кожній команді були надані елементи відставання продукту, які складалися зі списку історій користувачів. Їх попросили оцінити робочі зусилля, необхідні для кожної історії користувача в годинах, і розбити це на набір завдань.

Отже, кожна команда мала різний діапазон загальної кількості завдань у кожному спринті. На цьому етапі, на основі лекцій, прочитаних у класі, студентам поставили виклик у їх здатності планувати та оцінювати робочі зусилля, необхідні для кожного спринту.

Вихідні дані інструменту використовуються як засіб оцінки загального ризику в проєкті в будь-який момент або в огляді після спринту, як показано на рисунках 3.1 і 3.2. Цей графік містить інформацію про розподіл ризику, що визначається щодня за допомогою загального показника ризику (TRS).

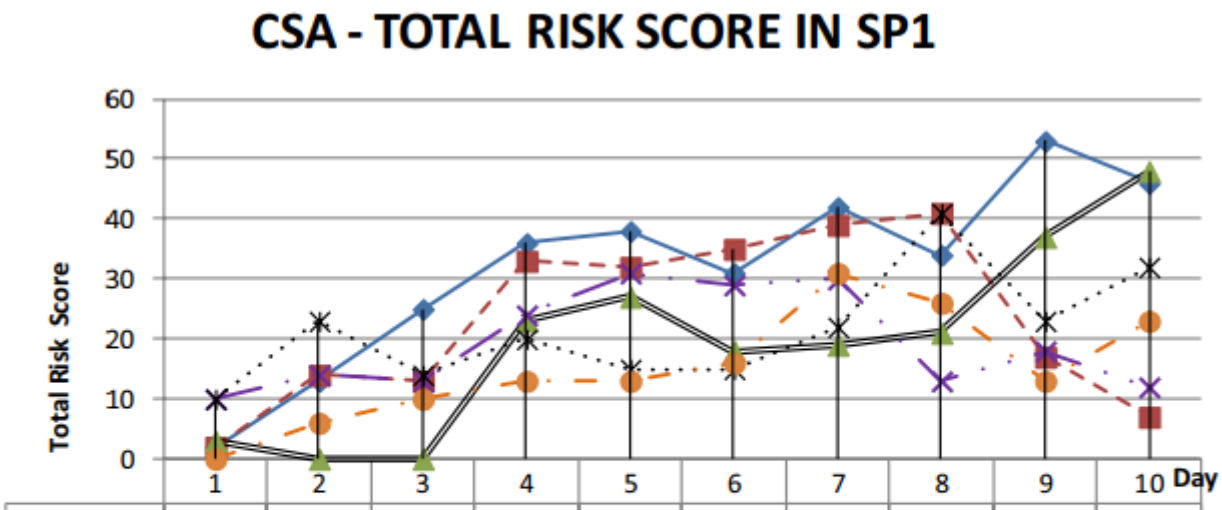
TRS базується на загальній оцінці серйозності елемента ризику для завдання, з яким він пов’язаний. Метрика дає результати щодо щоденного та кумулятивного підрахунку ризику підрахунок ризиків у спринті для цілей постійного управління ризиками. TRS розраховується, як показано нижче.

Вважається, що в проєкті на певний день d є набір завдань T:

$T_d = \{t_1 \ldots t_n\}$, де n = загальна кількість завдань у день d .

Існує також набір попередньо визначених загальних ризиків R, які потенційно можуть бути ідентифіковані в проєкті:

$R = \{r_1 \ldots r_m\}$, де m = загальна кількість попередньо визначених ризиків.



	1	2	3	4	5	6	7	8	9	10
Alp1	2	13	25	36	38	31	42	34	53	46
Alp2	2	14	13	33	32	35	39	41	17	7
Alp3	3	0	0	23	27	18	19	21	37	48
Alp4	10	14	13	24	31	29	30	13	18	12
Alp5	10	23	14	20	15	15	22	41	23	32
Alp6	0	6	10	13	13	16	31	26	13	23

Рисунок 3.1 – Графік загальної оцінки ризику альфа-версії прикладного дослідження (Спринт 1)

Таким чином, ризики, пов’язані з кожним завданням, t на певний день d є $R_{td} \subset R$.

Отже, TRS для завдання *t* дня d дорівнює:

TRS_{td} = карта (R_{td}), де карта – це потужність набору.

Ці ризики можуть бути пов’язані з будь-яким із завдань t в T. Ризики присутні під час виконання завдання в спринті.

Отже, для даного дня d TRS дорівнює t = n

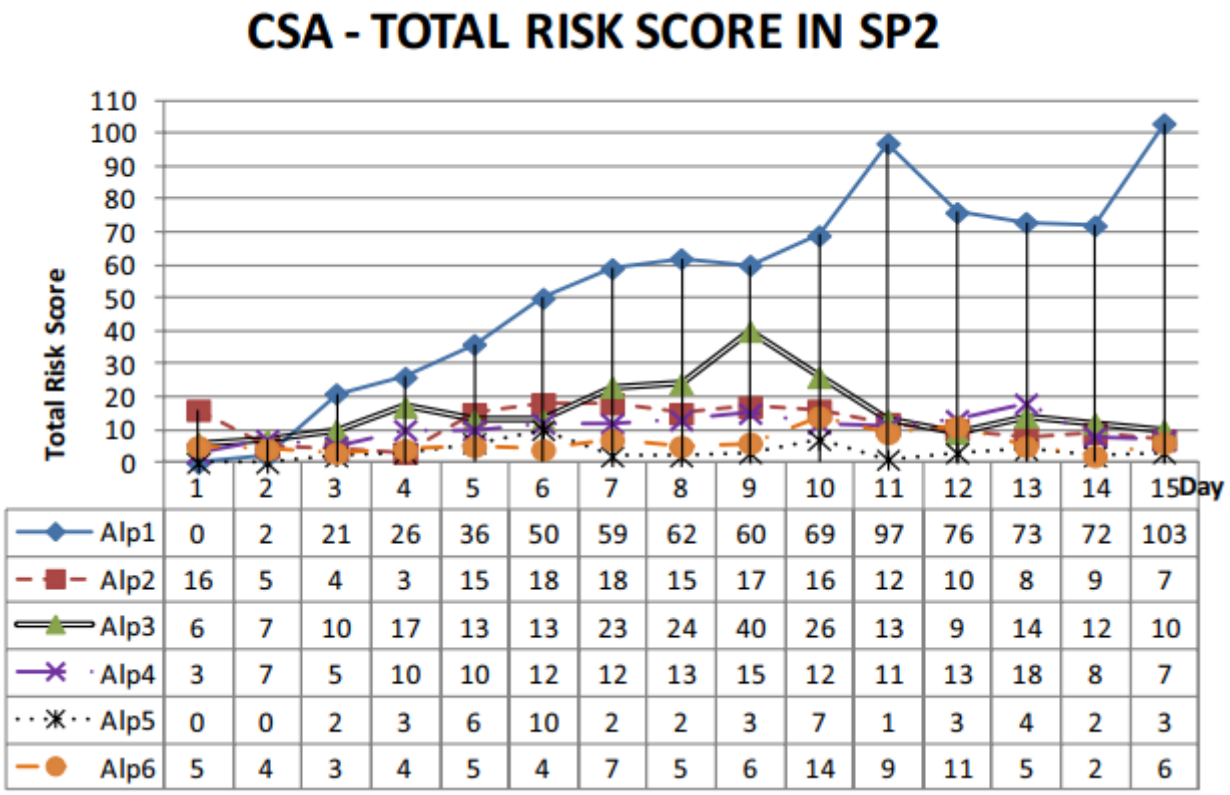
$$TRS_d = \sum TRS_{td}, t = 1, \quad (3.1)$$

де загальна кількість ініційованих ризиків буде розрахована для всіх завдань, виконаних у цей конкретний день.

Застосування TRS можна використовувати в поточному або поточному проєкті або в минулому проєкті. У поточному проєкті TRS розраховується щодня. Це коли керівник проєкту може побачити тригери ризику з першого дня, і якщо це буде

вирішено, ризик більше не з'являється. З іншого боку, TRS можна застосувати до минулого проєкту як засіб для перевірки. Отримані дані про ризики можуть бути використані як вхідні дані з використанням запропонованого підходу або для прогнозування ризику для майбутнього проєкту.

На рисунках 3.2 і 3.3 нижче показано графік TRS для обох спринтів. У SP1 кількість оцінок ризиків коливалася від нуля (або відсутність ризику, або відсутність поточного завдання) до найвищого з 53 ризиків, знайдених у команді Alp1, день 9 спринту. У SP2 оцінка ризику коливалася від нуля до найвищої з 103, також виявлена в команді Alp1, день 15. Обидва графіки показують тенденцію зростання та зменшення ризику, виявленого для кожної команди, з деякими піками в певні дні, що дає кращу візуалізацію ризику.



Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alp1	0	2	21	26	36	50	59	62	60	69	97	76	73	72	103
Alp2	16	5	4	3	15	18	18	15	17	16	12	10	8	9	7
Alp3	6	7	10	17	13	13	23	24	40	26	13	9	14	12	10
Alp4	3	7	5	10	10	12	12	13	15	12	11	13	18	8	7
Alp5	0	0	2	3	6	10	2	2	3	7	1	3	4	2	3
Alp6	5	4	3	4	5	4	7	5	6	14	9	11	5	2	6

Рисунок 3.2 – Графік загальної оцінки ризику альфа-версії прикладного дослідження (Спринт 2)

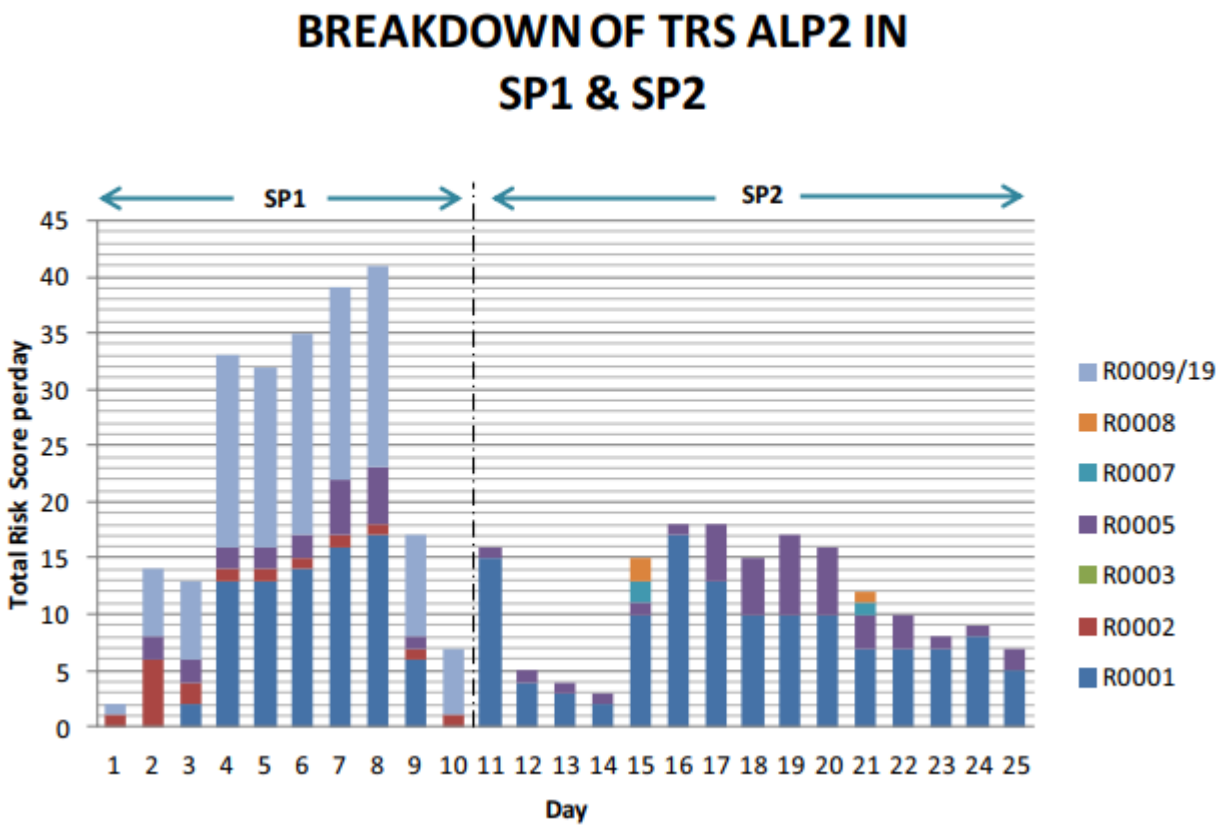
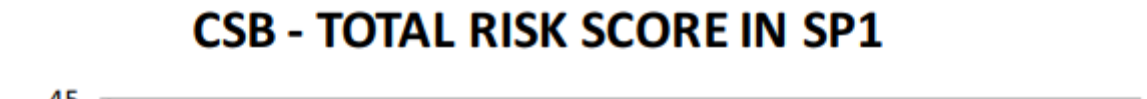


Рисунок 3.3 – Графік, що показує розбивку загального показника ризику для команди Alp2 у SP1 та SP2

Вже сформована діаграма вигоряння ризиків [13] зазвичай призводить до зниження рівня ризику, розрахованого на основі зміни ймовірності та розміру потенційних втрат у кожному спринті. Однак ця методика не показує і не візуалізує конкретно, який ризик ідентифікується, оцінюється та контролюється протягом усього проєкту. З іншого боку, наведений нижче графік можна використовувати для розробки тенденції ризику для типу або категорії для спритної команди. Наприклад, команда, яка новачок у проєктах із гнучкістю, може помітити, що ризики можуть поступово зростати до піку та почати зменшуватися до кінця спринту, коли ризики вирішуються або завдання виконуються. З іншого боку, більш сформована команда може не мати жодного ризику щодня в їхньому спринті; замість цього показують пік у певний час стосовно типу ризику, який вони хочуть відстежувати.

Дані, показані на рисунку 3.4, забезпечують кращу візуалізацію типу ризику, що ініціюється щодня в команді Alp2, для обох спринтів SP1 і SP2.



	1	2	3	4	5	6	7	8	9	10
Bet1	5	6	13	3	4	13	9	14	6	4
Bet2	1	11	22	29	32	22	18	18	20	13
Bet3	4	9	16	20	19	23	32	21	30	31
Bet4	0	6	11	12	15	26	19	17	17	1
Bet5	8	7	8	15	12	15	13	19	18	8
Bet6	4	14	12	9	9	19	10	11	11	2
Bet7	9	22	29	23	19	41	28	43	28	24

Рисунок 3.4 – Графік загальної оцінки ризику бета-версії прикладного дослідження (Спринт 1)

Перевага цього методу була подвійною; (1) можна було б переглянути категорію ризиків, ініційованих для конкретного дня, а також (2) можна було б побачити, який ризик ініціюється найбільше і потребує уваги з боку менеджера проєкту, щоб допомогти у прийнятті рішень наприклад, чи слід вжити негайних заходів щодо ризику, що виникла послідовно. Наприклад, у SP1 команда Alp2 створювала ризики в основному через невиконання парного програмування та через наявність членів команди без досвіду адаптації. Однак у SP2, оскільки студенти вже розробили використання Agile у SP1, ризик відсутності досвіду роботи з Agile більше не існував, але ризик невиконання парного програмування все ще міг виникнути. У студентських проєктах ці ризики, якщо вони реалізовані, можуть мати дуже незначний вплив на успіх проєкту, але в реальному проєкті, і ці ризики спрацьовують без їх усунення, загроза для проєкту може виникнути. З іншого боку, моделі виникнення ризиків для конкретного члена команди можуть бути виведені та

використані для оцінки майбутніх ризиків проєкту в подібних проєктах з тими ж членами команди.

3.2 Резюме даних проєкту CSB

У порівнянні з TRS, представленим у CSA раніше (див. рис. 3.3 , 3.4) для обох спринтів, TRS для CSB був набагато нижчим, ніж TRS в CSA. Це підтвердило висловлене раніше припущення, що вказівка на можливу перевірку щодо ризику може вплинути на поведінку розробника або студента під час проєкту.

Незважаючи на модифікацію правила парного програмування для CSB, TRS, створений у цьому прикладі, був більш реалістичним, тому парне програмування було критичним лише для більших завдань. Це може бути пов’язано з тим, що студентські проєкти були використані в тематичному дослідженні та меншою мотивацією виконувати парне програмування, ніж якщо б це сказали на робочому місці. Таким чином, через порушення цього правила виникла велика кількість ризиків. На практиці керівник проєкту міг час від часу змінювати це правило. На рисунках 3.5 і 3.6 показано розрахований TRS для обох спринтів. У SP1 оцінка ризику коливалася від нуля (або відсутність ризику, або відсутність поточних завдань) до найвищої оцінки ризику 43 у команді Bet7, день 8 спринту. Тоді як у SP2 оцінка ризику коливалася від нуля до найвищої оцінки ризику 38, яка також була знайдена в команді Bet7, день 9 спринту. Обидва графіки показують тенденцію зростання та зменшення ризику, виявленого для кожної команди, з деякими піками в певні дні, що дає нам кращу візуалізацію ризиків, що виникають.

Зазвичай можна очікувати, що ризик зменшиться (згорить) під час спринту. Однак, посилаючись на обидві цифри огляду спринту, для команди Bet1 на обидва спринти чітко продемонстрували б корисний результат використання інструменту, що продуктивність команди тут була проблематичною. Подібно до CSA, у світлі перегляду конкретних ризиків, які спрацьовували в певний день, дані, показані на рисунку 3.6, забезпечують кращу візуалізацію типу ризику, що ініціюється щодня для команди Bet5, для обох спринтів.

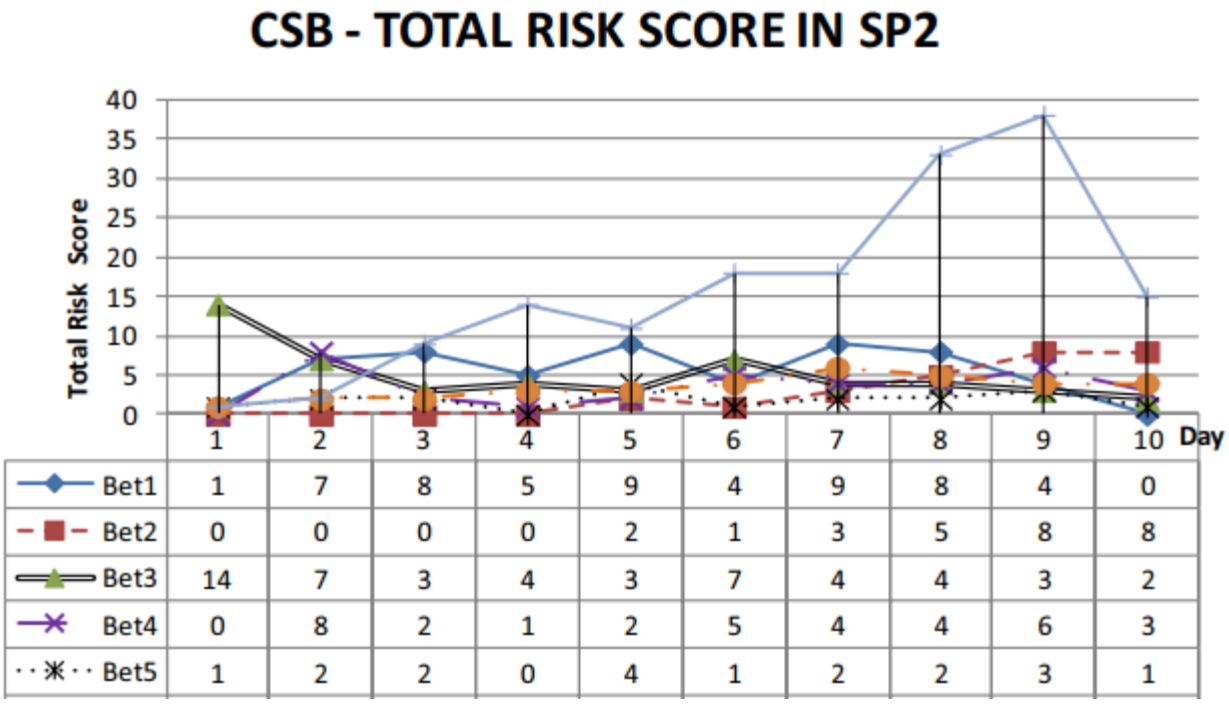


Рисунок 3.5 – Графік загальної оцінки ризику бета-версії прикладного дослідження (Спринт 2)

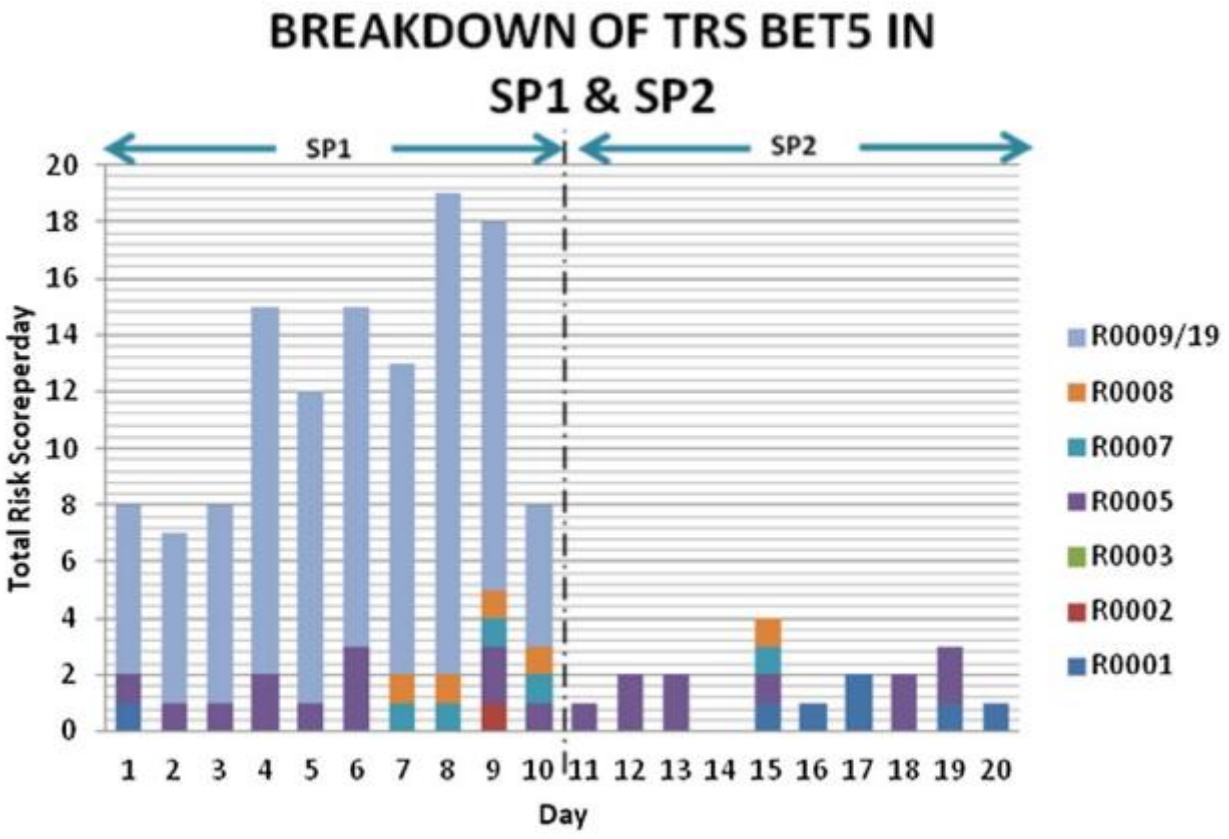


Рисунок 3.6 – Графік, що показує розбивку загальної оцінки ризику для команди Bet5 у SP1 та SP2

Перевага цього методу полягає в тому, що ми зможемо переглянути тип ризику, який ініціюється на певний день, а також побачити, який ризик ініціюється найбільше, а отже, якому слід протидіяти як пріоритет.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці та її актуальність в ІТ-сфері

Для підвищення ефективності системи управління охорони праці (СУОП) дуже важлива роль належить формуванню і розвитку інформаційної культури фахівців ІТ-технологій, яка впливає на удосконалення інформаційного контуру сучасних підприємств, дозволяє створювати надійні прогнози щодо стану умов праці, показників здоров'я та працездатності, виробничого травматизму і професійної захворюваності, визначати політику розвитку підприємств, установ та організацій на основі різноманітних стратегій охорони праці (інноваційні, маркетингові, інвестиційні, фінансові, технологічні, диверсифікаційні). Поряд з інформаційною культурою важливо використовувати в рамках СУОП «трикутник» її складових: правову, організаційну, управлінську.

В управлінні охороною праці потрібно реалізувати основні положення, окремі теоретико-методологічні підходи інформаційного менеджменту. Головну роль та відповідальність за стан СУОП мають нести фахівці служби охорони праці сучасного підприємства.

Сучасне суспільство називають постіндустріальним, постекономічним, інформаційним, оскільки йдеться про багатосторонні і кардинальні зміни у розвитку цивілізації.

Інформаційне суспільство передбачає докорінну зміну, яка полягає у перетворенні інформації і знань у головний професійно-виробничий потенціал особистості, соціуму і держави.

На постіндустріальному етапі розвитку суспільства вирішальним фактором стає інформація. Її домінування ініціювала науково-технічна революція, яку ще іменують інформаційною, оскільки нею охоплена будь-яка інтелектуальна діяльність, починаючи з інформаційних образів штучного інтелекту у нових технологіях, економіки, і продовжуючи інформатизацією суспільства в умова світової глобалізації науки й освіти тощо.

Інформаційні технології розглядаються як потужний важіль економічного зростання України. Для цього необхідні значні стратегічні інвестиції у комп'ютерну та комунікаційну інфраструктуру, програми досліджень і розробок, освітню галузь [42].

Під інформаційною культурою розуміють сукупність, складову НІТ (новітні інформаційні технології), технологічну, правову, психологічну, соціологічну та ергономічну підсистеми, що сприяють спрямованому впливу на протікання соціальних процесів у суспільстві, колективі і вихованню свідомого відношення людини до праці, виконання прав та обов'язків [43].

Поняття інформаційної культури виникло в процесі активізації дослідницької уваги до механізмів інформаційного обміну у зв'язку зі значним підвищення ролі інформації в соціокультурних процесах суспільства, яке розглядають як інформаційне суспільство знань, де в центрі знаходяться інформаційні технології.

Робота з інформацією та інформаційна культура в цілому є одним з найважливіших компонентів спроб компанії управляти змінами. Є три принципові причини, в силу яких сьогодні необхідно дбати про інформаційну культуру компанії.

По-перше, вона все більше і більше стає найважливішою частиною загальної організаційної (корпоративної) культури компанії. Все більше компаній розуміють необхідність перетворень, орієнтованих на задоволення очікувань споживача. Щоб сьогодні впливати на майбутнє, потрібно уявляти собі на що вона буде схожа. А для цього потрібно працювати з різноманітною діловою, професійною, технологічною, соціальною, ринковою та політичною інформацією.

По-друге, інформаційні технології роблять можливим створення в компаніях комп'ютерних мереж, за допомогою яких йде спілкування між менеджерами, але важливо знати, як люди використовують цю інформацію. Саме по собі створення такої мережі з усіма її робочими станціями і мультимедійними можливостями не гарантує того, що інформація буде використовуватися більш розумно і більш ефективно.

По-третє, для різних функціональних служб, підрозділів та робочих груп сучасних підприємств в сфері охорони праці інформаційна культура різна, а це

означає відмінність методологічних підходів до процесів усвідомлення, збору, організації, обробки, поширення і використання інформації. Тому багато менеджерів погодяться з тим, що корпоративна інформаційна культура важлива для вироблення різних стратегій охорони праці та запровадження відповідних заходів з її вдосконалення.

Для деяких галузей, таких як розробка програмного забезпечення, інформаційна культура є необхідною умовою виживання, тому що зміна технологій в розробці програмного забезпечення відбувається кожні 6-8 місяців, а інвестиції на підготовку персоналу і освоєння нової технології величезні і у великих компаніях варіюються від 1,5 до 2 млрд. доларів на рік [46].

Аналіз свідчить, що інформатизація та інтеграція комунікаційного простору України сприяє різкому підвищенню інформаційної та професійної компетентності, ділової активності, стимулюванню конкуренції, створенню інноваційних підприємств та організацій, нових робочих місць, зниженню витрат на утримання управлінського апарату [45].

Поряд із задачами і здобутками окреслилися негативи використання інформаційних технологій:

1) надмірне інформаційне навантаження, суть якого полягає у тому, що кількість корисної інформації, яка надходить до мережі, перевищує психофізіологічні можливості її сприйняття людиною;

2) велика кількість інформації, яка сприймається, але не є корисною для фахівців в даний момент;

3) інформаційний голод, причиною якого є саме надлишок інформації, викликаний інформаційним перенавантаженням;

4) «інформоманія» як хвороба людини, яка робить останню знеособленою, залежною від перебування в інформаційному просторі і роботи з комп’ютером і чому вона віддає перевагу, уникаючи «живого» спілкування з людьми;

5) поява «кіберспільнот», що за своїми соціокультурними характеристиками набагато ближчі до представників інших культур у глобальному інформаційному

просторі, ніж до своєї етнонаціональної спільноти чи решти населення, не охопленого Інтернетом;

6) індивідуалізм і дегуманізація способу життя «мешканців» Інтернету – відсутність готовності ділитися своїми знаннями.

Слід розуміти, що комп'ютерні технології, а особливо їх мережі істотно впливають на життєдіяльність людини, припускаючи глобалізацію і технократизацію суспільства. Але в ще більшій мірі цей вплив поширюється безпосередньо на центральну нервову систему, яка звикає працювати в дуже інтенсивному режимі багатозадачності, де вже переважають не тривалі логічні роздуми, а інтуїтивно-реактивні ланцюжки розумових формулювань у зв'язку з величезним обсягом оброблюваної щодня інформації, кількість якої зростає за експоненціальною швидкістю. Виникає припущення, що саме збільшення обсягу інформації та прискорення її обробки людиною може згубно вплинути на розвиток розумових здібностей людини.

Аналіз продуктивності розумової праці в найбільших за чисельністю фахівців IT-фірм показав, що велике значення з точки зору впливу на її результати має організаційна (корпоративна) культура. В цьому напрямі влаштовуються різні тімбілдинги, заходи, тренінги для розвитку персоналу. Також кожен керівник повинен добре розуміти свого співробітника, що саме для нього важливо, що його мотивує. Важливо відвести потрібну роль відповідному співробітнику, щоб він виконував ті завдання, які йому цікаві.

На подібних тренінгах в тому числі повинна розглядатися інформаційна культура працівника, в освоєнні, володінні, мотивуванні, застосуванні, перетворенні інформації із застосуванням сучасних інформаційних технологій і використанням цих умінь в навчанні з охорони праці і в подальшій професійній діяльності. Особливо вони будуть корисні, як доповнення до існуючих інструктажів з охорони праці на підприємстві, або як контроль психологічного стану та взаємовідносин у колективі.

Інформаційна культура як інтегративне утворення абсолютно не зводиться до розрізнених знань, вмінь та навичок роботи за комп'ютером. Вона передбачає інформативну спрямованість цілісної особистості, яка володіє мотивацією до

застосування і засвоєння нових даних. Інформаційну культуру можна розглядати, як одну з граней особистісного розвитку промислових робітників. Це шлях універсалізації якостей людини.

Оволодіння інформаційною культурою сприяє реальному розумінню особистістю свого місця, себе і своєї ролі у виробничому колективі. Вона має сприяти формуванню нового покоління фахівців інформаційного суспільства, який повинен володіти наступними навичками: виділення релевантної, значущої інформації, диференціації вихідних даних, розробки інформативних критеріїв її оцінки інформації, вміло використовувати її в рамках СУОП.

Сьогодні продовжує діяти стратегічне правило «Можливості комп'ютерної техніки обмежені тільки нашими уявленнями» [44].

4.2 Шкідлива дія шуту та вібрації і захист від неї

Для запобігання шкідливої дії шуму і вібрації на організм працюючих проводяться технічні, організаційні і медикопрофілактичні заходи.

Одним з основних технічних заходів є зменшення при експлуатації та на стадії проектування, конструювання обладнання причин шуму і вібрації в самому джерелі утворення. Досягають цього завдяки використанню раціональної конструкції обладнання, заміни ударної дії деталей і машин коливальною, з'єднання елементів гнучкими зв'язками, врівноважування обертових частин механізмів, заміни металевих деталей пластмасовими, забезпечення різних власних частот коливань механізму з частотою збуджуючої сили. Аеродинамічний шум може бути зменшений застосуванням глушників та повітропроводів зі змінним перерізом. Шум трансформаторів (електромагнітний шум) знижується, якщо застосувати листи заліза як складових осердя трансформатора з малою магнітострикцією, серцевини.

Якщо неможливо ізолювати чи знизити шум і вібрацію самого джерела, потрібно:

– ізолювати джерело шуму або вібрації від навколишнього середовища засобами вібро- та звукоізоляції

– раціонально планувати виробничі приміщення, що мають інтенсивні джерела шуму;

– збільшувати звукопоглинання внутрішніх поверхонь приміщення шляхом звукопоглинальних покриттів.

Принцип роботи звукоізоляційних екранів оснований на відбиванні звукової хвилі від різних екранів, стін, кожухів обладнання. Шумливі агрегати слід закривати звукоізоляційними кожухами з виводом назовні органів керування та контрольних приладів. Звукоізоляційні екрани виготовляють з металу, деревини, пластмаси та інших щільних матеріалів. Екрани зсередини покривають звукопоглинаючими матеріалами (скловатою пінополіуретаном), а по периметру кожуха – віброізоляційними підкладками (гума).

Вихідними даними для розрахунку параметрів необхідного екрану є спектр шуму, який необхідно ослабити, кількість екранів, через які проходить шум, їх площа, акустичні характеристики приміщення.

За розрахованими значеннями необхідної звукової ізоляційної здатності екрану підбирається матеріал конструкції й екрану.

Принцип звукопоглинання оснований на явищі трансформації коливальної енергії звуку в теплову через втрати при терті. Найбільші втрати при терті мають пористі, волокнисті і перфоровані матеріали: поролон, пемзолітові і деревоволокнисті плити тощо.

Енергія звукової хвилі переходить у теплову енергію, причому, ефект звукоізоляції збільшується з ростом частоти звукової хвилі. Звукопоглинаючими матеріалами оббивають стелі, стіни. Щоб одержати ефективну звукоізоляцію, найбільш доцільно застосовувати багатошарові огородження з м’якими прошарками (мінеральна вата).

Важливим технічним рішенням у забезпеченні виробничих умов є вдосконалення ручних віброінструментів. Для цього використовують віброгасіння, змінюють ударний вузол, проводять балансування частин, що обертаються.

Послаблення локальної вібрації і передачі вібрації на підлогу і сидіння досягається засобами віброізоляції і вібропоглинання, застосуванням пружинних і гумових амортизаторів, прокладок тощо. Для обмеження поширення вібрацій через ґрунт, між фундаментом і ґрунтом залишають повітряні проміжки, які називаються акустичними розривами.

В останні роки знаходять застосування динамічні віброгасники, в яких створюються вібрації, що співпадають по частоті і протилежні по фазі вібрації машини, коливання якої необхідно зменшити.

До організаційних заходів по боротьбі з шумом та вібрацією на виробництві відносяться: впровадження раціонального режиму праці і відпочинку, обмеження часу роботи при використанні ручного інструменту, який створює вібрацію.

Глушники звуку застосовуються для зменшення шуму аеродинамічних установок (вентиляторів, пневмоінструментів, газотурбінних, дизельних, компресорних установок). Вони поділяються на активні, які поглинають звукову енергію, що на них поступила, і реактивні, які відбивають цю енергію. Потужні джерела шуму як правило розміщують в окремих приміщеннях, які віддалені від постійних робочих місць.

Ізоляційні кабіни або екрани застосовують як екрани робочих місць для зменшення зовнішніх шумів.

Якщо не вдається зменшити рівень шуму і вібрації на робочому місці до нормативних значень та необхідно використовувати засоби індивідуального захисту: рукавиці, взуття, навушники, м’які шоломи, які зменшують рівень звукового тиску на 40-50 дБ.

У процесі виробництв, експлуатації і зберігання радіоелектронних засобів можуть виникати механічні і динамічні дії, що характеризуються широким діапазоном частот коливань, а також амплітудою, прискоренням і часом дії. Рівень механічних дій визначається умовами транспортування й експлуатації.

Необхідно розрізняти два види механічних дій: удари і вібрації. Удар виникає, коли апаратура отримує швидку зміну прискорення (піддаються удару входи кабелів, джгути, резистори, конденсатори, напівпровідникові діоди і тріоди, силові

трансформатори, дроселі тощо). Вібрації – довготривалі знакозмінні процеси, які впливають на роботу апаратури при безпосередньому контакті з джерелом коливань або через повітряне середовище.

У результаті дії вібрацій і удару можуть бути наступні ушкодження апаратури: порушення герметичності через псування паяльних, зварних і клеєних швів і появи тріщин у метало-скляних спаях; повне руйнування корпусів або окремих їх частин через механічний резонанс або циклічну втому; обривання монтажних зв'язків, відшарування багатошарових друкованих плат, руйнування підставок; вихід з ладу електричних контактів; модуляція розмірів хвилеводних трактів; коаксіальних кабелів, конденсаторів змінної ємності, коливальних контурів, електровакуумних приладів, зміщення положення органів настроювання і управління.

Під впливом вібрацій може статись зміна параметрів напівпровідникових приладів, вольт амперних характеристик діодів, транзисторів. Все це призводить до руйнування конструкцій за рахунок явищ втоми.

Радіоелектронна апаратура (РЕА) повинна мати віброміцність, вібростійкість, ударостійкість.

Захист РЕА здійснюється наступними групами методів:

– зменшується інтенсивність джерел вібрації шляхом балансування, зменшення зазорів, віброізоляції джерела вібрацій;

– зменшується величина дій, що передається апаратом шляхом віброізоляції, демпфірування, виключення резонансів, активного віброзахисту за допомогою ексцентриків, маятників, гіроскопів;

– використання найбільш добротні і жорсткі компоненти і вузли;

– застосовуються амортизатори.

Захист часом, захист віддалю, усунення джерела тепловиділення, теплоізоляція, охолодження гарячої поверхні, забезпечення тепловіддачі тіла людини та індивідуальні засоби захисту.

Захист часом передбачає обмеження часу перебування робітника в зоні дії інфрачервоного випромінювання. Потужність випромінювання можна знизити за

рахунок конструкторських і технологічних рішень (зміною нагрівання виробів у нагрівальних пічках індукційним нагріванням та ін.) і за рахунок покриття поверхні, яка нагрівається, тепло ізолювальним матеріалом.

Якщо теплоізоляція неможлива, тоді захист від прямої дії інфрачервоного випромінювання здійснюється екрануванням.

Екрани можуть бути прозорими, напівпрозорими і непрозорими.

У свою чергу вони поділяються на тепловідбивальні, тепловідвідні та теплопоглинальні; стаціонарні і нестаціонарні.

Застосовують також прозору водяну завісу у вигляді суцільної тонкої водяної плівки. Вода є активним поглиначем інфрачервоного випромінювання.

Перегрівання людини попереджують раціональним режимом пиття, режимом праці та гідро процедурами. Спецодяг виготовляється з незаймистого, стійкого до інфрачервоного випромінювання, м’якого і повітронепроникного матеріалу (тканина з металевим покриттям відбиває 90 % інфрачервоного випромінювання).

Для захисту очей застосовують світлофільтри зі спеціального жовто-зеленого або синього скла.

Першочергові заходи – це конструкторські і технологічні рішення, які виключають генерацію або понижують інтенсивність випромінювання. Спеціальні засоби захисту (екранування джерел випромінювання, фарбування стін у світлі кольори) попереджують розповсюдження і знижують інтенсивність цих випромінювань у виробничих приміщеннях. Очі захищають окулярами або щитками зі склом – світлофільтром. Для захисту шкіри використовують мазі з речовинами – світлофільтрами для цих променів (салол, саліцилово-метиловий ефір та ін.), а також спецодяг з бавовняних тканин і грубововняного сукна. Руки захищають рукавицями.

ВИСНОВКИ

Результати, отримані в ході вивчення прикладу, засвідчили, що, крім новизни запропонованого підходу та інструментальної підтримки, цей підхід надає корисні дані. Графіки TRS, створені в обох тематичних дослідженнях, забезпечують метод візуалізації, який підтримує ідентифікацію та моніторинг ризиків у динамічному Agile проєкті. Вони показують кількість ризиків, які виявляють агенти ART, і забезпечують реалістичний та інтерактивний спосіб моніторингу ризиків. Дослідження показало, що можна використовувати наявні дані середовища SE для підтримки управління ризиками. Однак існує багато елементів даних середовища, які можна використовувати для виявлення ризику, залучаючи менеджера проєкту, щоб визначити, які з них пов’язані з їхнім проєктом. Крім того, наші емпіричні дані також показали, що дані можна зібрати з мінімальним втручанням і зусиллями.

Три дослідницькі питання (RQ) були встановлені раніше як вираження цілей цієї дослідницької роботи. У наступних параграфах узагальнено, як на них реагували в тематичних дослідженнях.

RQ1: Висновок надає докази того, що можна використовувати дані середовища проєкту для визначення ризиків і таким чином подолати основні бар'єри у застосуванні управління ризиками. Це підтверджує думку про те, що в гнучких процесах потрібен легкий підхід до управління ризиками, який автоматизує деякі процеси ризику. В результаті цього було розроблено рішення з використанням програмних агентів для реагування на середовище проєкту на основі визначених правил для управління ризиками.

RQ2: Метод, використаний для обох тематичних досліджень, що свідчить про те, що збір даних, проведений з обох тематичних досліджень, передбачав мінімальне втручання та зусилля, а також без жодних витрат. У використаних тематичних дослідженнях використовувані дані середовища включають дані студентського проєкту, у цьому випадку з архівних даних. Дані зберігалися в репозиторіях SVN і були отримані викладачем для використання в тематичному дослідженні. Обидва тематичні дослідження не включали жодної взаємодії з учасниками в умовах

дослідження для досягнення мети дослідження. Це включає в себе виявлення ризиків у своєму проєкті та дослідження дотримання членом команди методів Agile. Зібрані дані були перевірені викладачем/власником продукту студентського проєкту на основі дискусійних сесій до та після впровадження проєкту.

RQ3: Інструмент-прототип був розроблений, щоб перевірити взаємодію між агентами, відповідність агентам призначеним правилам і те, як агенти реагують на зміни в даних середовища проєкту. Це обговорюється вище, починаючи з визначення вхідних даних, обробки вхідних даних і створення вихідних даних. Пізніше в обох тематичних дослідженнях, які підтримуються інструментом-прототипом, буде використано покрокове керівництво процесу ART. Це демонструє, що програмні агенти в поєднанні з механізмом правил можуть автоматизувати управління ризиками за допомогою даних із середовища проєкту.

Оскільки представлене дослідження запроваджує новий підхід до управління ризиками в Agile-проєкті, основні питання зосереджені на внутрішніх загрозах. Перша внутрішня загроза – це точність виміряних даних, особливо тому, що використані дані були засновані на історичних артефактах. Крім того, підтвердження цих даних було неможливим, оскільки на момент аналізу проєкт уже був завершений. По-друге, використаний підхід передбачав ручний збір та переклад даних з архівних артефактів в інструмент ART. Ці людські зусилля були необхідні, перш ніж агенти ART могли почати реагувати на дані про навколишнє середовище, оскільки вони були розроблені для роботи. Ці зусилля можна було б звести до мінімуму, вибравши відповідну людину в команді для проведення цього процесу, наприклад, Scrum Master у Scrum-проєкті. .

Один із кроків, зроблених для забезпечення якості дослідження, полягав у тому, що час від часу проводилася перехресна перевірка з власником продукту, щоб підтвердити сприйняття на основі його спостережень. Враховуючи зовнішні загрози чинності, підхід до управління ризиками та підтримка інструментів були розроблені так, щоб вони були максимально загальними, щоб це було загалом застосовно до гнучких проєктних середовищ. Це включає в себе врахування двох популярних інструментів керування проєктами, які вивчаються для цієї роботи, щоб підхід був

максимально застосовним до інших контекстів, але також був легким і ненав'язливим для щоденної діяльності команди. Тим не менш, не можна претендувати на хорошу підгонку з невивченими інструментами. Крім того, у дослідженні використовувалися дані студентських проєктів разом із рекомендаціями щодо виконання кейсів, а не промислові дані. Отже, будуть аргументи, чи це застосовно до реального середовища.

Таким чином, у цій роботі ми представили новий підхід до управління ризиками в Agile проєктах. Робота пропонує внески у двох областях:

1) щодо використання тематичних досліджень для оцінки нових методів та інструментів і надає приклад того, як студентські команди можуть бути використані для збору інформації, нездійсненної в промислових умовах;

2) про використання агентів для напівавтономного управління ризиками програмного забезпечення.

У цій роботі наведено кілька важливих досліджень проблем і проблем управління ризиками, зокрема в Agile проєктах. Було продемонстровано, що розробка моделі ART та підтримки інструментів допомагає принаймні зменшити проблеми, які раніше були виявлені під час управління ризиками. Цей підхід обов'язково підтримується інструментом-прототипом, який, як було показано, керує ризиками в прикладі гнучких проєктів. Роль управління ризиками в ітеративних і гнучких процесах на сьогоднішній день нехтувала, але ця модель інтегрує модель управління ризиками з гнучкими методами таким чином, що не роздуває гнучкий процес.

Однак цей підхід, до знань і розуміння авторів, ніколи не застосовувався в управлінні ризиками, особливо з конкретною метою зменшення людських зусиль. Крім того, отриманий в результаті процес управління ризиками, природно, є легким, оскільки кожен програмний агент розроблений для досягнення визначеної мети, тобто для визначення, оцінки, визначення пріоритетів або моніторингу ризику. Ця стаття привела до використання призначених програмних агентів для полегшення процесу управління ризиками. Таким чином, ця робота демонструє потенціал автономних обчислень, які застосовуються для управління ризиками, коли програмні агенти використовуються для допомоги орієнтованому на людину та складному процесу управління ризиками. У майбутньому ця робота мала на меті осягнути

фізичну реалізацію моделі ART та підтримки інструментів, де є необхідність інтегрувати це з існуючими інструментами Agile Project Management, можливо, як плагін, щоб можна було повністю реалізувати автоматизоване управління ризиками. Це дозволить більш практично керувати ризиками, поки проєкт виконується на передньому плані, програмні агенти знаходяться на задньому плані, готові керувати ризиками, що виникають.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahmed A, Kayis B, Amornsawadwatana S (2007) A review of techniques for risk management in projects. Benchmarking: Int J 14(1):22–36

2. Bandyopadhyay K, Mykytyn PP, Mykytyn K (1999) A framework for integrated risk management in information technology. Management Decision 37(5):437–445

3. Bannerman PL (2008) Risk and risk management in software projects: a reassessment. J Syst Softw 81(12):2118–2133.

4. Bellifemine FL, Caire G, Greenwood D (2007) Developing multiagent systems with JADE, Wiley.com.

5. Boehm BW (1989) Tutorial: software risk management. IEEE Computer Society Press, Washington.

6. Boehm B, Turner R (2003) Using risk to balance Agile and plandriven methods. Computer 36(6):57–66.

7. Boehm B, Turner R (2005) Management challenges to implementing Agile processes in traditional development organizations. Softw IEEE 22(5): 30–39.

8. Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2002) Modeling early requirements in Tropos: a transformation based approach. In: Agent-Oriented Software Engineering II. Springer, Berlin, pp 151–168.

9. Chakradhar K (2009) Risk management in Agile development. White paper edn. Polaris Software Lab Limited.

10. Cho J (2008) Issues and Challenges of Agile software development with SCRUM. Issues Inf Syst 9(2):188–195.

11. Cockburn A, Highsmith J (2001) Agile software development, the people factor. Computer 34(11):131–133.

12. Cohn M (2005) Agile estimating and planning. Pearson Education, London

13. Cohn M (2010) Managing Risk on Agile Projects with the Risk Burndown Chart.

14. Conboy K, Coyle S, Wang X, Pikkarainen M (2010) People over process: key people challenges in Agile development. IEEE Software.

15. Dardenne A, Van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Programm (20)1: 3–50.

16. Deemer P, Benefield G, Larman C, Vodde B (2010) The scrum primer. Scrum primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, vol. 1285931497. http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf

17. Easterbrook S, Singer J, Storey M, Damian D (2008) Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, Springer, Berlin, pp 285–311.

18. Fowler M, Highsmith J (2001) The Agile manifesto. Softw Dev 9(8):28–35

19. Hossain E, Babar MA, Paik H, Verner J (2009) Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework. Software Engineering . Conference, 2009. APSEC'09, Asia-Pacific, IEEE, p. 457.

20. Ibbs CW, Kwak YH (2000) Assessing project management maturity. Project Management Journal 31(1):32–43.

21. Islam, Shareeful. "Software development risk management model: a goal driven approach." Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium. 2009.

22. Kitchenham B, Pickard L, Pfleeger SL (1995) Case studies for method and tool evaluation. Softw IEEE 12(4):52–62.

23. Kontio J (1997) The RISKIT method for software risk management, version 1.00. Computer Science Technical Reports, University of Maryland, College Park, MD, USA.

24. Layman L, Williams L, Cunningham L (2006) Motivations and measurements in an Agile case study. J Syst Archit 52(11):654–667.

25. Lindvall M, Basili V, Boehm B, Costa P, Dangle K, Shull F, Tesoriero R, Williams L, Zelkowitz M (2002) Empirical findings in Agile methods. In: Extreme Programming and Agile Methods – XP/Agile Springer, pp 197–207.

26. Melnik G, Maurer F (2006) Comparative analysis of job satisfaction in Agile and non-Agile software development teams. In: Extreme Programming and Agile Processes in Software Engineering . Springer, pp 32–42.

27. Melo C, Cruzes DS, Kon F, Conradi R (2011) Agile team perceptions of productivity factors. Agile Conference (AGILE), 2011 IEEE, pp 57.

28. Menzies T, Williams S, Elrawas O, Baker D, Boehm B, Hihn J, LumK, Madachy R (2009) Accurate estimates without local data? Softw Process Improv Pract 14(4):213–225.

29. Nelson C.R., Taran G., de Lascurain Hinojosa L. (2008) Explicit risk management In: International Conference on Agile Processes and Extreme Programming in Software Engineering. Springer, pp. 190–201.

30. Nerur S, Mahapatra R, Mangalaraj G (2005) Challenges of migrating to Agile methodologies. Commun ACM 48(5): 72–78.

31. Nyfjord J, Kajko-Mattsson M (2008) Integrating risk management with software development: State of practice. In: Proceedings, IAENG International Conference on Software Engineering.

32. BrownWalker Press, Boca Raton Citeseer, Odzaly EE, Greer D, Sage P (2009) Software risk management barriers: An empirical study. Empirical Software Engineering and Measurement. ESEM 2009 pp 418–421.

33. Odzaly EE, Greer D, Stewart D (2014) Lightweight Risk Management in Agile Projects. In: SEKE (pp 576–581).

34. Patterson FD, Neailey K (2002) A risk register database system to aid the management of project risk. Int J Project Manage 20(5):365–374.

35. Pfleeger SL (2000) Risky business: what we have yet to learn about risk management. J Syst Softw 53(3):265–273.

36. Ropponen J, Lyytinen K (1997) Can software risk management improve system development: an exploratory study. Eur J Inf Syst 6(1):41–50.

37. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131–164

38. Willams TM (1994) Using a risk register to integrate risk management in project definition. Int J Project Manage12(1):17–22.

39. Williams RC, Walker JA, Dorofee AJ (1997) Putting risk management into practice. IEEE Soft 14(3):75–82.

40. Психология безопасности труда / Укладач Кальянов А.В. // Донецкий областной совет профсоюза, 2008. – 32 с.

41. Сьогодні UA [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.segodnya.ua/lifestyle/fun/pochti-kak-u-google-chemudivlyayut-ofisy-ukrainskih-it-kompaniy–764025.html – відкритий.

42. Конспект лекцій з курсу «Охорона праці в галузі» / Укладачі: Яскілка В.Я., Олійник М.З. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 56 с.

43. Шконда В.В., Кальянов А.В. Культорологічні засади становлення майбутніх фахівців: Монографія. – Донецьк, 2012. – 262 с.

44. Шконда В.В., Кальянов А.В., Давыдов П.Г. Феномен синергетики: наука – общество – образование: Монография / Ред. Шконда В.В. – Донецк: Норд-Пресс, 2009. – 156 с.

45. Информационная культура предприятий, виды информационной культуры, информационное поведение [Електронний ресурс]: [Веб-сайт]. – Електронні дані (Лекції). – Режим доступу: https://lektsii.com/1-78900.html – відкритий.

46. Пивоваров М.Г., Медко Д.А. Перспективы создания и развития информационно-коммуникационной системы Украины // Економіка: проблеми теорії та практики: Зб. наук. праць. – Вип. 49. – Дніпропетровськ: Дніпропетр. Нац. Ун-т, 2000. – С.56-61.

ДОДАТКИ

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

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

Р. Войтюк, Ю. Тарасовський
ЗАДАЧА ВИБОРУ ПРОГРАМНОЇ АРХІТЕКТУРИ ПРИ ЗМІНІ ВИМОГ
R. Voitiuk, Yu. Tarasovskyi
SOFTWARE ARCHITECTURE CHOOSING FOR CHANGING REQUIREMENTS 148

А.М. Долінський
СТВОРЕННЯ ЗАСОБІВ ДЛЯ ОПТИМІЗАЦІЇ РОБОТИ З КОНТЕНТОМ В MAGENTO 2
A.M. Dolinskiy
CREATION OF TOOLS FOR OPTIMIZATION OF WORK WITH CONTENT IN MAGENTO 2 149

А.М. Долінський
РОЗРОБКА ПЕРСОНАЛІЗОВАНОЇ ТЕМИ MAGENTO 2 НА ОСНОВІ LUMA
A.M. Dolinskiy
DEVELOPMENT OF A PERSONALIZED THEME MAGENTO 2 ON THE BASIS OF LUMA 150

М.Р. Петрик, В.В. Борейко
ЗАСТОСУВАННЯ C# LIBRARY FOR MARKDOWN В ПРОЕКТУВАННІ ПРОГРАМНИХ СИСТЕМ ФОРМУВАННЯ ДОКУМЕНТАЦІЇ
M.R. Petrik, V.V. Boreiko
APLICATION OF C# LIBRARY FOR MARKDOWN IN DESIGN OF SOFTWARE SYSTEMS OF DOCUMENTATION FORMATION 151

В.О. Босяк, М.Р. Петрик
ПРОЕКТУВАННЯ ТА РОЗРОБКА РОЗПОДІЛЕНОЇ СИСТЕМИ НА ОСНОВІ МОДЕЛІ КЛІТИННОГО АВТОМАТУ
V.O. Bosiak, M.R. Petryk
DESIGN AND DEVELOPMENT OF DISTRIBUTED SYSTEM BASED ON CELLULAR AUTOMATA MODEL 152

Н.О. Голуб, Г.Б. Цуприк
РОЗРОБКА ЄДИНОЇ СИСТЕМИ ДОСТУПУ ДО ПУБЛІЧНОЇ ІНФОРМАЦІЇ З ВИКОРИТАННЯМ СУЧАСНИХ IT-ТЕХНОЛОГІЙ
N.O. Holub, H.B. Tsupryk
DEVELOPMENT OF A UNIFORM SYSTEM OF ACCESS TO PUBLIC INFORMATION USING MODERN IT TECHNOLOGIES 154

Ю.М. Громик, І.В. Бойко
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗДІЙСНЕННЯ АНАЛІЗУ ТОНАЛЬНОСТІ ТЕКСТУ ІЗ ЗАСТОСУВАННЯМ ТЕХНОЛОГІЇ ГЛИБОКОГО МАШИННОГО НАВЧАННЯ ТА МОВИ PYTHON
Y.M. Gromyk, I.V. Boyko
DEVELOPMENT OF AN INFORMATION SYSTEM FOR SENTIMENT ANALYSIS USING DEEP MACHINE LEARNING AND PYTHON 155

Н.Т. Дзись, Г.Б. Цуприк
РОЗРОБКА НОВИХ МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВРАХУВАННЯМ РАЦІОНАЛЬНО УНІФІКОВАНОГО ПІДХОДУ
N.T. Dzys, H.B. Tsupryk
DEVELOPMENT OF NEW SOFTWARE MODULES TAKING INTO ACCOUNT A RATIONAL UNIFIED PROCESS 156

УДК 004.42
Р. Войтюк, Ю. Тарасовський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ЗАДАЧА ВИБОРУ ПРОГРАМНОЇ АРХІТЕКТУРИ ПРИ ЗМІНІ ВИМОГ

UDC 004.42
R. Voitiuk, Yu. Tarasovskyi

SOFTWARE ARCHITECTURE CHOOSING FOR CHANGING REQUIREMENTS

Сучасні гнучкі технології програмування по суті є ітераційними. При виконанні поточної ітерації можуть вноситися зміни у вимоги, або обмеження, що потребуватиме внесення відповідних змін у розділи проєкту, в тому числі і в розділ архітектури. Також в процесі експлуатації програмної системи (ПС) відбуваються зміни вимог предметної області, що викликає необхідність внесення змін в ПС, а тому в першу чергу необхідні зміни програмної архітектури (ПА), оскільки вона визначальним чином впливає на якість ПС. Вибір архітектури здійснюється з множини альтернатив, які конструюються на основі функціональних вимог із стандартних компонентів. Для вибору найкращого варіанта архітектури обчислюються їх оцінки по окремих критеріях якості, а потім на основі отриманих оцінок виконується багатокритерійний вибір архітектури. Задача оцінювання альтернатив по окремим критеріям якості найбільш ефективно розв'язується використанням методу аналізу ієрархій Сааті (МАІ) або його модифікованого варіанта (ММАІ). Суттєвим недоліком застосування МАІ є обмежена кількість альтернатив, які можна оцінювати одночасно ($n \leq 7 \pm 2$).

В роботах [1], [2] розглянуті питання застосування модифікованого МАІ до задачі оцінювання альтернативних варіантів архітектури програмних систем при великій кількості альтернатив. В цих методах відносна оцінка альтернатив визначається з використанням експертної інформації і при зміні вимог до ПС на черговій ітерації проєктування потрібно повторно проводити експертне оцінювання та розрахунки оцінок критеріїв альтернатив. Оскільки в ітераційних технологіях проєктування ПС процеси можуть виконуватись одночасно на декількох стадіях життєвого циклу з використанням базового варіанта архітектури, то його зміна потребуватиме внесення коректив в декілька розділів проєкту.

Для зменшення об'єму необхідних змін в проєкті, пов'язаних із зміною вимог до ПС, пропонується використати процедуру корекції критеріїв якості базової архітектури та оптимізації цієї корекції. Оптимізація заміщення є задачею багатокритерійної оптимізації. В якості критерія, який оптимізується, пропонується використати нелінійну скалярну згортку. В ній оптимізується цільова функція, яка залежить від міри «напруженості ситуації», котра визначається близькістю значень критеріїв до своїх обмежень. Для формалізації процесу визначення ваг критеріїв використовується ітераційна процедура симплекс-планування. Отримані оптимальні значення корекції критеріїв використовуються для модифікації архітектури ПС.

Література

1. Харченко О.Г. Метод багатокритеріальної оптимізації програмної архітектури на основі аналізу компромісів [Текст] / Харченко О.Г., Боднарчук І.О., Галай І.О. // Інженерія програмного забезпечення. – К.: НАУ.-2012. – № 3–4 (11–12). – с. 5 – 11.
2. Kharchenko A. The method for comparative evaluation of software architecture with accounting of trade-offs/ Alexander Kharchenko, Ihor Bodnarchuk, Vasyl Yatcyshyn // American Journal of Information Systems. V. 2, No. 1. 2014. – P. 20-25. Available online at http://pubs.sciepub.com/ajis/2/1/5