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

бакалавр
(назва освітнього ступеня)

на тему: Розробка архітектури програмного продукту для управління змінами вимог в розподілених командах.

Виконав(ла): студент(ка) 4 курсу, групи СНз-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Маркушевський П.С. (прізвище та ініціали)

Керівник (підпис) Никитюк В.В. (прізвище та ініціали)

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

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

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2023

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)
«12» червня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Маркушевський Петро Степанович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розробка архітектури програмного продукту для управління змінами вимог в розподілених командах**

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

Затверджені наказом ректора від «*07*» *лютого* 2023 року № *4/7-135*

2. Термін подання студентом завершеної роботи 12 червня 2023 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП РОЗДІЛ 1. АНАІЗ ПРОБЛЕМИ ФОРМУВАННЯ ВИМОГ В РОЗПОДІЛЕНИХ КОМАНДАХ 1.1 Підхід до моделювання процесу формування вимог 1.2 Дерево функцій 1.3 Формування вимог у Distributed Agile 1.4 Фаза ініціалізації проекту РОЗДІЛ 2. ІНСТРУМЕНТ ПІДТРИМКИ УПРАВЛІННЯМ ЗМІНАМИ ВИМОГ 2.2 Оцінка інструменту підтримки 2.3 Критерії оцінювання 2.4 Обов'язки команд з оцінки 2.5 Резюме оцінювання РОЗДІЛ 3. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ ВИСНОВОК ПЕРЕЛІК ПОСИЛАНЬ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доц.		

7. Дата видачі завдання 24 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.23-27.01.23	*Виконано*
2.	Підбір джерел по темі роботи	28.01.23 – 01.04.23	*Виконано*
3.	Оформлення першого розділу	15.04.2023	*Виконано*
4.	Оформлення другого розділу	30.04.2023	*Виконано*
5.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи охорони праці»	15.05.2023	*Виконано*
6.	Оформлення кваліфікаційної роботи	07.06.2023	*Виконано*
7.	Перевірка на плагіат	07.06.2023	*Виконано*
8.	Нормоконтроль	09.06.2023	*Виконано*
9.	Попередній захист кваліфікаційної роботи	11.06.2023	*Виконано*
10.	Захист кваліфікаційної роботи	13.06.2023	

Студент _____ Маркушевський П.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка архітектури програмного продукту для управління змінами вимог в розподілених командах // Кваліфікаційна робота освітнього рівня "Бакалавр" // Маркушевський Петро Степанович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНз-41 // Тернопіль, 2023 // с. – 42, рис. – 5, табл. – 4, кресл. – 10, бібліогр. – 15.

Ключові слова: програмна архітектура, розподілені команди, управління вимогами, гнучка розробка.

Розробка програмного забезпечення на сьогодні широко використовує гнучкі підходи та процеси. Застосування таких підходів у розподіленому середовищі призведе до отримання багатьох переваг, таких як зниження витрат, вища ефективність розробки, а з іншого боку, вона зіткнеться з багатьма проблемами, наприклад робота в різних часових поясах, зміни вимог, особисті відбір і управління знаннями.

Однією з головних проблем є керування змінами вимог у процесі розподіленої гнучкої розробки програмного забезпечення. Більшість опублікованих досліджень у цьому контексті базуються на виробничому досвіді, що збільшує потребу в поєднанні виробництва з академічними колами в цій галузі. У цій роботі представлено підхід до управління змінами вимог у розподіленій гнучкій розробці. Цей підхід базується на моделі дерева функцій. Цей підхід пов’язаний із використанням допоміжного програмного інструменту, який керує змінами вимог у розподіленій гнучкій розробці.

ANNOTATION

Software Product Architecture Development for Management of Requirements Changes in Distributed Teams // Qualification work of the educational level "Bachelor" // Petro Markushevskyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group СН-41 // Ternopil, 2023 // p. – 42 , fig. – 5, tables – 4, references – 15, posters – 10.

Keywords: software architecture, distributed teams, requirements management, agile development.

Software development today widely uses agile approaches and processes. Applying such approaches in a distributed environment will give in the future many benefits, such as cost reduction and higher development efficiency, but on the other hand, it will face many challenges. For instance, a working in different time zones, requirements changes, team composing and others.

One of the main challenges is requirements management changes, especially in distributed agile projects. Most of the published research in this context is based on industrial experience, which increases the need to combine production with science in this field. This paper presents an approach to requirements change management in distributed agile development. This approach is based on the feature tree model. This approach involves the use of a supporting software tool that manages requirements changes in distributed agile development.

ЗМІСТ

ВСТУП

В останні роки розробка програмного забезпечення розповсюдилася географічно. Зараз індустрія програмного забезпечення має тенденцію переносити свої виробничі підрозділи в децентралізовані зони, де кваліфікована робоча сила є більш доступною, використовуючи таким чином політичні та економічні чинники. Основна мета цього полягає в оптимізації ресурсів для розробки продуктів вищої якості за нижчою ціною.

Розподілена розробка програмного забезпечення (DSD) дозволяє членам команди знаходитися на різних віддалених сайтах протягом життєвого циклу програмного забезпечення, таким чином створюючи мережу віддалених підгруп. У деяких випадках ці групи можуть бути членами однієї організації; в інших випадках може існувати співпраця або аутсорсинг із залученням різних організацій.

Таким чином, традиційні особисті зустрічі більше не є поширеними, а взаємодія між членами вимагає використання технологій для полегшення спілкування та координації. Незважаючи на те, що це явище почалося в 90-х роках, лише протягом останніх десяти років було визнано його стратегічне значення, а відповідні дослідження є зовсім недавніми.

Відстань між різними командами може коливатися від кількох метрів (коли команди працюють у суміжних будівлях) до різних континентів. Ситуація, в якій команди розподілені за межами нації, називається глобальною розробкою програмного забезпечення (GSD). Цей вид сценарію цікавий з кількох причин, головним чином тому, що він дозволяє організаціям абстрагуватися від географічної відстані, маючи при цьому кваліфіковані людські ресурси та мінімізуючи витрати, таким чином збільшуючи площу ринку за рахунок виробництва програмного забезпечення для віддалених клієнтів і отримання довшого робочого дня. використовуючи переваги різниці в часі. Однак необхідно вирішити низку проблем, викликаних головним чином відстанню,

часом і культурними відмінностями, і вони значною мірою залежать від конкретних характеристик кожної організації.

Вищезазначені практики розробки мають як спільний фактор проблеми, що виникають через відстань, які безпосередньо впливають на процеси комунікації та координації, так і контрольну діяльність. У цих середовищах комунікація є менш гнучкою, ніж у колокалізованих групах розробки, тому виникають проблеми, пов’язані з координацією, співпрацею чи груповою обізнаністю, що негативно впливає на продуктивність і, як наслідок, якість програмного забезпечення. Усі ці фактори впливають на те, як програмне забезпечення визначається, будується, тестується та доставляється клієнтам, таким чином впливаючи на відповідні етапи життєвого циклу програмного забезпечення.

Щоб пом’якшити ці наслідки та з метою досягнення вищих рівнів продуктивності, організаціям потрібні нові технології, процеси та методи шляхом удосконалення, пов’язаного з життєвим циклом програмного забезпечення, плануванням проекту, оцінками, управлінням ризиками, якістю. Ітераційні підходи зазвичай використовуються на відміну від традиційних каскадних або послідовних методів, але їх стає важче використовувати послідовно, коли команди територіально розподілені.

Наразі в цій галузі розвивається підхід, керований моделлю (MDD), який забезпечує повторне використання, зручність обслуговування, взаємодію та адаптивність за допомогою різних мов і платформ, а також покращує якість програмного забезпечення та продуктивність розробників. Архітектура, керована моделлю (MDA) є найбільш часто використовуваним стандартом MDD і забезпечує концепції поділу в окремих моделях і методах перетворення.

Існують також такі інструменти, як InterDOC, які служать прикладом потужності підходу для забезпечення авторського процесу, коли необхідна взаємодію між різними додатками для спільної роботи.

Однією з головних проблем, з якими стикається процес формулювання вимог до програмного забезпечення, є зв’язок між основним та офшорними (аутсорсними) частинами команди в розподілених групах. Щоб скоротити час ітерації на офшорній ділянці, потрібен більш ефективний зв’язок із основною платформою. Попередні знання в певній області дуже важливі для розуміння вимог до програмного забезпечення.

Гнучкі методи вважаються хорошим рішенням, коли вони працюють у невеликих ітераціях, щоб забезпечити повне програмне рішення в кінці цих ітерацій. Розробка Agile спрямована на підвищення ефективності команди, скорочуючи час комунікації та обміну інформацією між людьми. Іншим аспектом Agile є мінімізація наданої документації.

Деякі дослідження в цій області виконувались також в ТНТУ імені Івана Пулюя, зокрема в роботах [1, 2, 3].

Основною працею, з котрої також використовувались дані дослідження, є робота [4].

У розподілених Agile командах існує поділ на дві категорії: основна команда та офшорна команда. Одна з головних причин, чому проекти програмного забезпечення розгортаються в офшорах, полягає в тому, що вони використовують дешевшу робочу силу та інфраструктуру. Зменшення вартості розробки є однією з головних переваг, яких можна досягти за допомогою офшорної розробки (аутсорсу) . Розподіл процесів розробки між офшорними та основними частинами загальної команди, як у розподіленій гнучкій розробці, спричинить проблему комунікації. Проблема комунікації між клієнтами та основною командою, з одного боку, та командою розробників на офшорі, з другого боку, виникає через труднощі врегулювання зустрічей віч-на-віч та обговорення різних питань проекту відповідно до різниці у фізичному розташуванні та, отже, різних географічних районах та часові зони. Основними проблемами в розподіленій розробці є комунікація, управління знаннями, культурне розмаїття, різниця в часі та зміни вимог [5].

Ця робота представляє підхід до розподіленої гнучкої розробки для керування вимогами та їх змінами під час процесів розробки. Цей підхід заповнює прогалину між практичною розробкою та дослідженнями у розподіленій гнучкій розробці шляхом поєднання промислової практики та академічних методів. Запропонований підхід базується на моделі ознак [6] з використанням дерева властивостей. Функціональна модель спочатку введена для моделювання спільності та мінливості на етапі інженерної розробки лінійки програмних продуктів. Допоміжний інструмент (його архітектура) розроблений для допомоги в управлінні змінами вимог до програмного забезпечення. Його головна мета – пом’якшити деякі проблеми, з якими стикається розподілена гнучка команда.

РОЗДІЛ 1. АНАІЗ ПРОБЛЕМИ ФОРМУВАННЯ ВИМОГ В РОЗПОДІЛЕНИХ КОМАНДАХ

1.1 Підхід до моделювання процесу формування вимог

Моделювання функцій – це підхід до моделювання вимог до програмного забезпечення з високою складністю та вираженням кількох вимог у функціях та структурування їх ієрархічно на діаграмах функцій. Розробка сімейств програмного забезпечення або лінійки продуктів вимагає всебічного розуміння предметної області. Необхідно провести детальний і ретельний аналіз доменів, а результати мають бути задокументовані в узгодженій формі. У літературі існує ряд методологій аналізу, наприклад, моделювання домену організації [5], аналіз предметно-орієнтованого домену [6] та доменно-орієнтована архітектура програмного забезпечення [7] є найпопулярнішими з них.

На рисунку 1.1 представлено приклад моделі функцій для простої системи веб-реєстрації.

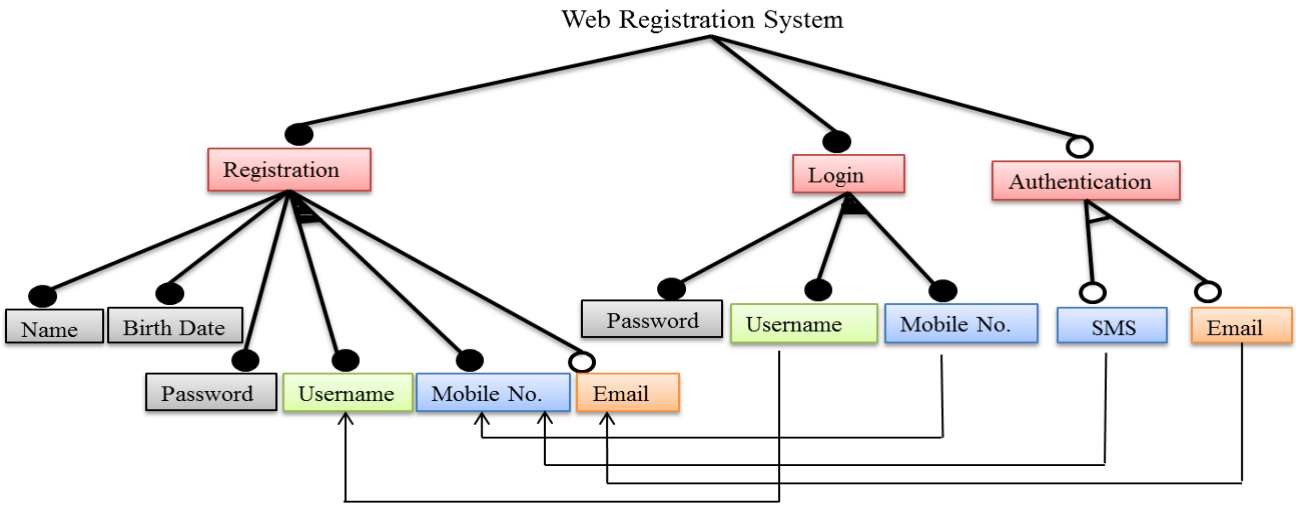


Рисунок 1.1 – Модель функції веб-реєстрації

Система веб-реєстрації складається з трьох основних функцій: двох обов'язкових функцій, які є функціями реєстрації та входу, і однієї додаткової функції, яка є функцією автентифікації. Функція реєстрації містить дві

обов’язкові функції: ім’я та дату народження, а також містить дві альтернативні функції: ім’я користувача та номер мобільного телефону, остання функція – електронна пошта, і це необов’язкова функція. Другою основною функцією є функція входу, яка містить один обов’язковий пароль і дві альтернативні функції – ім’я користувача та номер мобільного телефону.

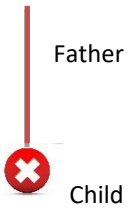
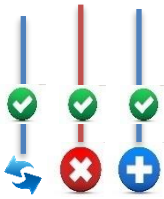
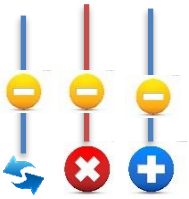
Для вибору функції «Ім’я користувача» у функції «Вхід» потрібно вибрати функцію «Ім’я користувача» в головній функції «Реєстрація», а для вибору функції «Електронна пошта» потрібно вибрати функцію «Електронна пошта» в основній функції «Реєстрація». Третя і остання основна функція – це функція автентифікації, і вона є додатковою. Функція автентифікації містить дві альтернативні функції: автентифікація електронної пошти та автентифікація через SMS. Для вибору функції SMS потрібно вибрати функцію «Номер мобільного телефону», а для вибору електронної пошти в автентифікації потрібно вибрати функцію електронної пошти під час реєстрації. Функціональна модель системи веб-реєстрації може задовольнити вимоги для більш ніж однієї програми в області веб-реєстрації та те, що називається вимогами формування для сімейства систем або лінійки продуктів.

Модель функцій використовується для моделювання мінливості для сімейства систем або лінійки продуктів у певній області. У нашому випадку ми цього робити не будемо. Метою нашого підходу є моделювання вимог до єдиного програмного додатку в розподіленому гнучкому середовищі. Вимоги до програмного забезпечення в цьому випадку можуть час від часу змінюватися в процесі розробки. Ми прагнемо прийняти модель функцій для керування змінами вимог і представити ці зміни як змінність для вивчення впливу цих змін на інші вимоги, що називається залежністю вимог. З іншого боку, зміни вимог до моделювання допоможуть команді розробників офшорних компаній зрозуміти нові вимоги та можливість розробки. У наступному розділі ми представимо наше запропоноване дерево функцій, яке містить деякі модифікації

моделі функцій для підтримки процесу керування змінами вимог у розподіленій гнучкій розробці.

У таблиці 1.1 представлені запропоновані позначення для підтримки та контролю процесів зміни вимог, які будуть представлені в дереві функцій у наступному розділі.

Таблиця 1.1– Нотації моделі розширених функцій

Тип	Семантичний	Позначення	Тип	Семантичний	Позначення
додати	Додати нову дочірню функцію.		Видалити	Видалити наявну дочірню функцію.	
оновлення	Оновити наявну дочірню функцію.		Схвалити модифікацію	Підтвердьте видалення, додавання та оновлення.	
Запобігання модифікації	Заборонити видалення, додавання та оновлення		Виключення	Вказує на те, що обидва функції не можуть бути обрані в одній конфігурації продукту, тому вони взаємовиключні	
Підтекст	необхідно також вибрати неявну функцію, ігноруючи її позицію в дереві функцій.				

Процес прийняття моделі функцій включатиме нові позначення. Усі вимоги будуть змодельовані як обов’язкова характеристика зі значною концентрацією на зв’язку між вимогами, який називається залежністю вимог у

формі зв’язку наслідків і виключень. Немає необхідності вказувати необов’язкові, обов’язкові чи альтернативні вимоги, які використовуються в лінійці продуктів. Щоб маніпулювати та керувати змінами вимог, пропонуються нові нотації для представлення процесів додавання, оновлення та видалення вимог.

1.2 Дерево ознак

Процес побудови дерева ознак складається з двох основних кроків. Перший крок спрямований на побудову початкової форми дерева ознак. Початкова форма дерева функцій міститиме всі вимоги до кандидатів. На рисунку 1.2 показано приклад початкового дерева ознак.

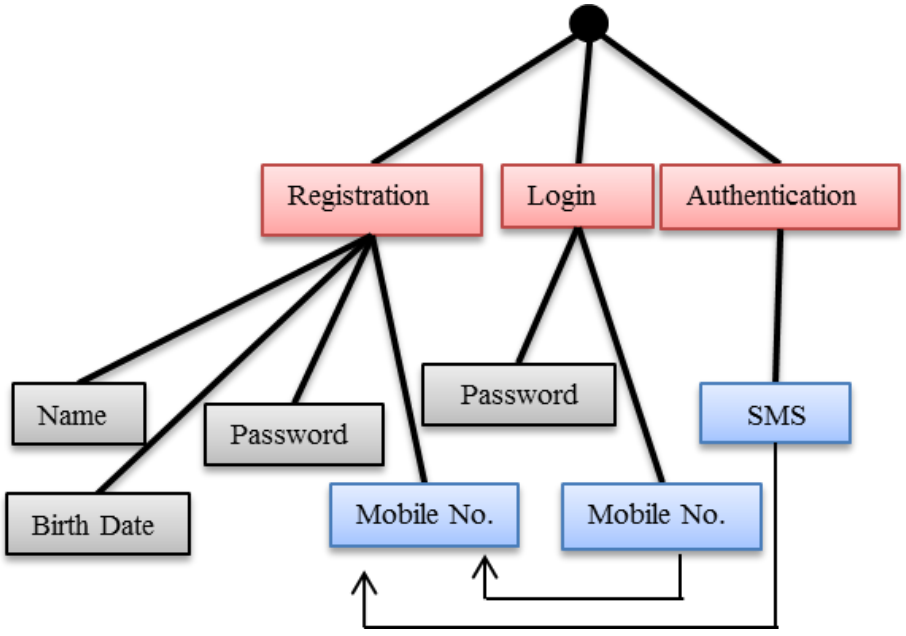


Рисунок 1.2 – Приклад дерева ознак

Другим кроком є керування змінами вимог. Після кожної ітерації команда розробників прийматиме рішення про схвалення нових змін або запобігання їм, використовуючи позначки схвалення та запобігання, наведені в таблиці 1.1. Процес управління змінами можна підсумувати трьома видами діяльності:

a) Модифікація вимог у формі додавання, видалення та оновлення.

b) Контроль змін у формі схвалення або запобігання.

c) Моніторинг за залежністю між вимогами у формі припущення та виключення.

1.3 Формування вимог у Distributed Agile

Щоб представити повний розподілений гнучкий підхід із керуванням змінами вимог, пропонується поєднати запропоноване дерево функцій із новим гнучким процесом для розподілених проектів (NAPDP) [8, 9]. NAPDP – це підхід до глобального гнучкого розвитку (GAD); він надає переваги обміну знаннями домену та пропонує краще розгалуження та керування релізами. Він також надає набір передових практик, допомагаючи зв’язати дизайн, підходи до тестування та інструменти, які допомагають підтримувати архітектуру програмного забезпечення.

NAPDP базується на вивченні різних підходів, які слідують гнучким моделям розробки та Agile-маніфесту. NAPDP об’єднав найбільш відповідні методи та найкращі практики з Rational Unified Process (RUP) [13], Extreme Programming (XP) [11, 12], Scrum і Rapid Object Oriented Process for Embedded System (ROPES) [10]. Запропонований підхід використав переваги NAPDP і поєднав ці переваги із запропонованим деревом функцій. Запропонований підхід складається з двох основних фаз: ініціалізації проекту та циклу розробки, кожна фаза містить кілька кроків.

1.4 Фаза ініціалізації проекту

На етапі ініціалізації проекту керівник проекту і інженери вимог фіксують вимоги замовника/зацікавлених сторін. Багато заходів з планування також проводяться на цьому етапі. Цей етап складається з наступних трьох кроків:

1. Планування проекту.

На попередніх нарадах щодо проекту виконується кілька видів діяльності: планування та розклад проекту, розподіл ресурсів, стратегії розвитку, включаючи план розвитку та керування конфігурацією.

2. Інженерія вимог.

На цьому кроці вимоги інженерів охоплюють вимоги від різних ресурсів, таких як експерти домену та зацікавлені сторони. Інженери вимог також відповідають за виявлення, аналіз, специфікацію, перевірку вимог і створення початкового дерева функцій .

3. Створення списків пріоритетних функцій.

На цьому кроці створюються списки пріоритетних функцій. Кожен список буде реалізовано в ітерації циклу розробки та інтегровано в систему. Керівник проекту, архітектор програмного забезпечення, менеджер з тестування та якості, дизайнер функцій, інженери вимог, менеджер конфігурації та виконавчі програмісти – усі вони залучені до цього кроку.

4. Фаза циклу розробки.

На етапі циклу розробки реалізується певна кількість функцій. Під час кожної ітерації циклу розробки зберігається детальне дерево функцій. Функції розроблені, реалізовані та перевірені. Швидкий прототип реалізованих функцій надається для оцінки замовником наприкінці кожної ітерації. У наступному розділі ми детальніше пояснимо кожен крок.

5. Аналіз.

Кожна ітерація циклу розробки починається з етапу аналізу. Цей крок включає аналіз вимог, аналіз архітектури та аналіз об’єктів.

6. Ведення дерева функцій.

Процес підтримки дерева функцій включає додавання нових функцій, зміну або оновлення існуючих функцій і видалення існуючих функцій . Процес обслуговування включає також затвердження зміни вимоги та відстеження зв’язків залежності між вимогами.

7. Дизайн.

Після завершення аналізу та збереження нових змін вимог у дереві функцій процес переходить до етапу проектування. Цей крок починається з архітектурного проектування.

8. Реалізація.

Етап реалізації перетворює функції на досяжне рішення. Розробники пишуть необхідний код для розробки функцій.

9. Тест.

Тестовий етап гарантує, що функції вже спроектовані та реалізовані правильно. Цей крок починається після того, як усі розробники завершать процес написання коду. Результати тестів оцінюються та надсилаються назад для повторного впровадження в разі невдачі тесту.

10.Просування продукту.

На цьому етапі клієнти залучаються до команди розробників для оцінки розроблених функцій. На рис.1.3 показано фази запропонованого підходу.

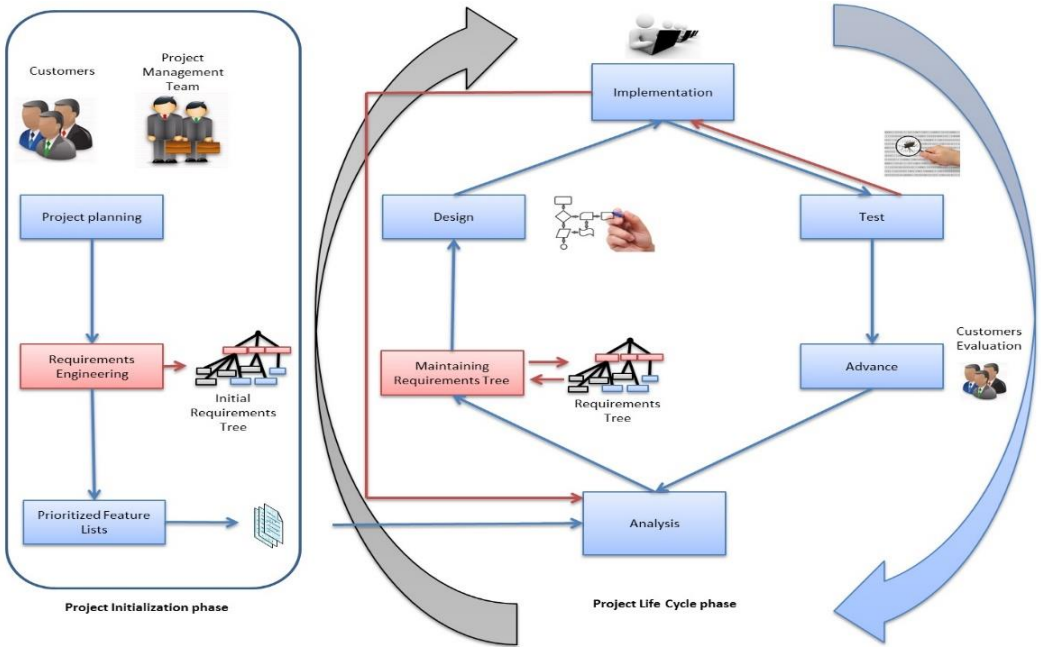


Рисунок 1.3 – Фази запропонованого підходу гнучкої розробки в розподілених командах

Команда розробників може згодом переглянути коди та відредагувати їх, якщо необхідно, на основі коментарів клієнтів.

РОЗДІЛ 2. ІНСТРУМЕНТ ПІДТРИМКИ УПРАВЛІННЯМ ЗМІНАМИ ВИМОГ

Для допомоги у застосуванні запропонованого підходу розроблено допоміжний інструмент. Він зосереджений на управлінні змінами вимог у розподіленій гнучкій розробці. Інструмент враховує природу розподіленої гнучкості, наприклад різні географічні розташування команд розробників, зв'язок між командою розробників і управління знаннями. Це веб-інструмент, призначений для полегшення та керування моделюванням вимог і змінами на всіх етапах.

Він починає фіксувати вимоги на етапі ініціації проекту та відстежує вимоги та їх зміни протягом усіх наступних кроків та ітерацій. Інструмент також дозволяє керівнику проекту відстежувати статус функцій на кожному кроці розробки. Рис. 2.1 представляє діаграму діяльності допоміжного програмного інструменту.

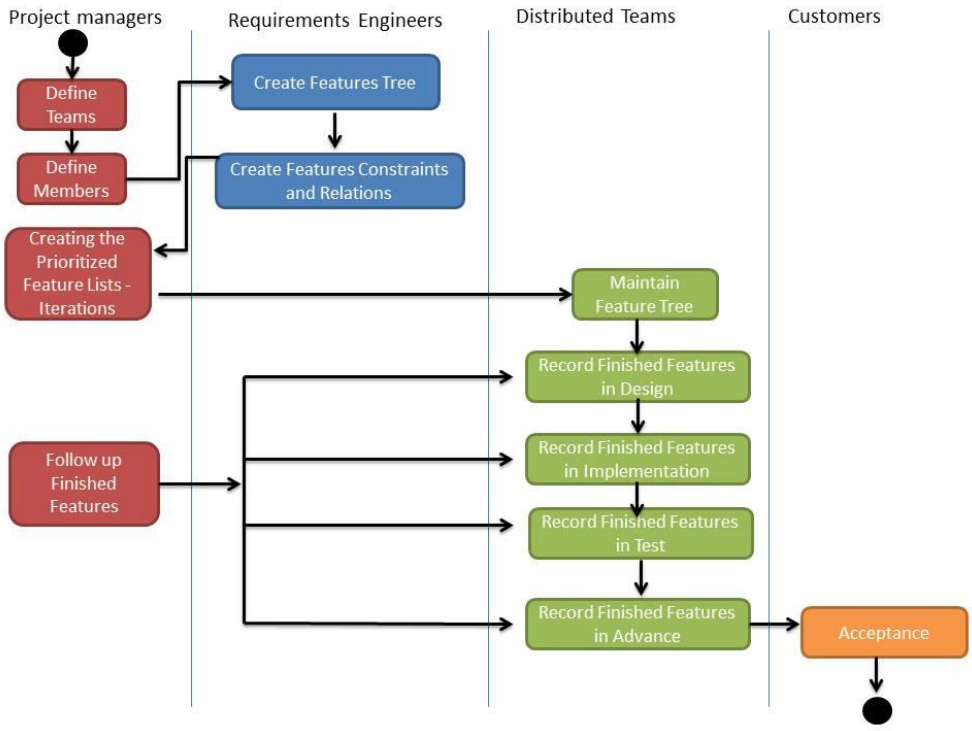


Рисунок 2.1 – Діаграма діяльності допоміжного програмного інструменту

Керівник проекту тісно співпрацює з різними зацікавленими сторонами та групами розробників, щоб визначити учасників розробки, а потім керівник проекту призначає кожного учасника до команди. Після цього інженери вимог створюють початкову форму дерева функцій, обмеження між функціями також визначаються ними. Керівник проекту знову бере участь у створенні пріоритетного списку функцій, кожен список функцій буде призначено одній команді. Розподілені групи будуть задіяні для запису кожної завершеної функції зі списку пріоритетів на кожному етапі циклу розробки, починаючи з кроку аналізу. Групи розробників також зберігають і записують зміни вимог у дерево функцій.

Команди розробників продовжують записувати готові функції на всіх етапах, і керівник проекту зможе контролювати та відстежувати готові функції на кожному кроці в ході проекту. Останнім кроком є прийняття, яке проводиться в тісній співпраці із замовником. Рисунок 2.2 демонструє, як допоміжний інструмент може підтримувати дерево функцій.

Рисунок 2.2 – Ведення дерева функцій у інструменті підтримки

2.2 Оцінка інструменту підтримки

Щоб оцінити цю роботу, ми зв’язалися з кількома організаціями з розробки програмного забезпечення в Єгипті, які можуть бути зацікавлені в області розподіленої гнучкої розробки. Однією з таких організацій є IT-відділ Національного дослідницького центру (NRC) в Єгипті. Основними обов’язками IT-відділу є розробка, підтримка, заміна та оновлення систем програмного забезпечення для NRC. NRC є найбільшим дослідницьким центром в Єгипті, який налічує понад 4000 дослідників, які працюють у різних напрямках досліджень. IT-відділ NRC виявив інтерес до оцінки та перевірки допоміжного інструменту, оскільки зростає потреба в роботі в розподіленому середовищі розробки. Вони збиралися розробити систему управління документами (DMS), метою якої є ініціювання, відстеження, керування та зберігання документів для зменшення паперової роботи. Групи розробників розташовані в різних будівлях і філіях NRC; кожна команда відповідає за охоплення та задоволення потреб у програмному забезпеченні певних секторів.

2.3 Критерії оцінювання

Після кількох зустрічей із керівником проекту та керівниками команд для визначення критеріїв оцінки вони погодилися щодо таких критеріїв оцінки: концепція, дизайн, зручність використання, можливості, безпека, відстежуваність, доступність, багатокористувацька функція, конфігурації, загальність і призначення завдань. У таблиці 2.1 представлені критерії оцінки з додатковими поясненнями.

Таблиця 2.1 – Критерії оцінювання

№	Критерії	Опис
1	Концепція	Чи забезпечує інструмент механізм керування змінами вимог під час процесу розподіленої розробки?
2	Дизайн	Члени команди повинні оцінити дизайн інструменту, щоб охопити такі аспекти: • Дивитися і відчувати, • навігація, • Зв'язок між вимогами.
3	Юзабіліті	Наскільки легко використовувати інструмент у різних проектах?
4	Доступність	Чи має інструмент необхідні можливості для підтримки етапів розробки на двох етапах розробки?
5	Безпека	Чи безпечний інструмент для багатокористувацького середовища?
6	Простежуваність	Чи може інструмент відстежувати статус проекту?
7	Доступність	Чи доступний інструмент з різних географічних зон?
8	Багатокористувацький	Чи підтримуватиме інструмент використання кількох користувачів одночасно?
9	Конфігурації	Яка конфігурація потрібна для встановлення та використання інструменту?
10	Загальність	Чи можна використовувати його в проектах програмного забезпечення різного типу та розміру?
11	Завдання	Чи надає інструмент можливість призначати витрати на розробку для кожного завдання в процесі розробки?

2.4 Обов'язки команд з оцінки

Обов’язки груп з оцінки:

– Виявляти та збирати вимоги від різних користувачів.

– Визначати особливості вимог або переведення вимог користувача у функції.

– Побудова дерева ознак.

– Ініціювання обмеження між функціями.

– Призначення функцій для іншої команди розробників; кожен набір функцій сформує окрему ітерацію.

– Оцінка часу, необхідного для кожної ітерації, відповідно до бачення керівника проекту та складності функцій.

– Перегляд та дотримання вимог на кожній ітерації та, якщо потрібно, оновлення дерева функцій.

– Підтримка зв’язку між функціями на етапі циклу розробки.

– Відслідковування за готовою функцій на кожному кроці на етапах розробки.

Таблиця 2.2 представляє оцінку інструменту згідно з думкою команд розробників.

Таблиця 2.2 – Оцінка інструменту

№	**Критерії**	**Опис**
1	**2**	**3**
1	Концепція	Концепція функції є хорошою ідеєю, яка надає групі розробників гнучкість для представлення вимог і відстеження функцій на етапах розробки. Відносини між вимогами представлені обмеженнями перехресного дерева, які зберігають залежність між вимогами. Інструмент дозволяє групі розробників мати огляд недостатнього розвитку проекту, зосереджуючись на призначеній роботі. Інструмент працює як єдиний контролер для управління та контролю прогресу розробки. Побудова дерева функцій вимагає багато зусиль на початку проекту, а підтримка дерева функцій на етапі циклу розробки також вимагає багато зусиль і часу. Витрачений час і зусилля на побудову та підтримку дерева функцій є прийнятними з огляду на керування змінами вимог.

Продовження таблиці 2.2

1	2	3
2	Дизайн	Інструмент розроблений як веб-інтерфейс із простим графічним інтерфейсом користувача. Має навігаційне меню для полегшення переходу від сторінки до іншої. Вимоги представлені ієрархічно в батьківських стосунках, що дає розумне та простежене представлення вимог. Перехресні зв'язки між вимогами представлені обмеженнями наслідків і виключень. Ці обмеження дозволяють підтримувати вимоги поза межами батьківських відносин.
3	Юзабіліті	Інструмент простий у використанні у великих проектах, і навігація між функціями також проста. Простота використання робить інструмент більш зручним.
4	Можливість	Інструмент надав зрозумілий і простий механізм керування вимогами розподіленої гнучкої розробки. Інструмент розділений на дві частини; кожна частина представляє фазу розвитку відповідно до запропонованого підходу.
5	Безпека	Інструмент безпечний; кожен член команди має власний пароль доступу та ім'я користувача. Керівник проекту, керівники команд і члени команди мають різний рівень доступу до інструменту.
6	Простежуваність	Інструмент забезпечує механізм відстеження всіх кроків на етапах розробки. Він також відстежує завершену та незавершену функцію для кожної ітерації.
7	Доступність	Інструмент є веб-додатком, який можна розмістити в Інтернеті та отримати доступ будь-де та будь-коли. Інструмент використовує переваги веб-додатків.
8	Багатокористувацький	Інструмент підтримує доступ кількох користувачів одночасно. Доступ до інструменту можна отримати з будь-якого місця.
9	Конфігурації	Інструменту потрібен веб-сервер для розміщення веб-програми; у цьому випадку потрібен Microsoft Internet Information Server (IIS). Інструмент зберігає дані в базі даних сервера Microsoft SQL, яку також потрібно встановити.
10	Загальність	Інструмент, який використовується для розробки системи управління документами, яка розглядається як система баз даних середнього розміру. Інструмент підтвердив свою здатність керувати змінами вимог у контексті розподіленої гнучкої розробки.
11	Завдання	Інструмент не надає можливість керівникам груп призначати завдання учаснику розробки. Керівники групи відповідають за введення даних розробки, тоді як учасник розробки може переглядати та отримувати ці дані.

У таблиці 2.3 представлені рекомендації щодо інструменту, які підвищать функціональність і зручність використання інструменту в майбутньому.

Таблиця 2.3– Рекомендації щодо покращення допоміжного інструменту

№	Критерії	Опис
1	Концепція	Інструменту необхідно розширити його функціональність, щоб розглянути можливість призначення завдань учаснику розробки. Інструменту потрібно додати кілька звітів для вимірювання продуктивності команди розробників.
2	Дизайн	Команди розробників вважають за краще показувати повідомлення веб-додатків у спливаючих вікнах, а не текст на веб-сторінці червоного кольору. Команди розробників попросили додати улюблене меню, яке буде містити часто відвідувані сторінки веб-додатку.
3	Юзабіліті	Інструмент не можна буде використовувати без знання підходу. Інструмент доступний лише англійською мовою, і його буде зручніше використовувати, якщо він підтримуватиме більше мов.
4	Можливості	Розширте функціональні можливості інструменту, щоб розглянути можливість призначення завдань учаснику розробки. Інструмент повинен повідомляти керівника проекту про зміни в дереві функцій.
5	Безпека	Інструмент повинен сповіщати користувачів про їхні рівні безпеки.
6	Поступливість	Інструмент не має графічних зображень для представлення прогресу проекту.
7	Загальність	Інструмент підтвердив свою здатність керувати змінами вимог у контексті розподіленої гнучкої розробки для програм баз даних.
8	Завдання	Інструмент не надає можливість керівникам груп призначати дублі учаснику розробки.

2.5 Резюме оцінювання

Концепція функції є хорошою ідеєю, і її можна налаштувати відповідно до природи проекту, вона забезпечує гнучкість представлення вимог. Відносини

між вимогами представлені обмеженнями перехресного дерева , які зберігають залежність між вимогами. Інструмент довів свою здатність керувати змінами вимог у розподіленій гнучкій розробці. Він простий у використанні та вимагає розумної конфігурації для встановлення. Це дозволяє команді розробників контактувати та співпрацювати в одному місці; він також забезпечує чітку структуру для роботи в розподіленій розробці. Запропонований підхід впорався зі змінами вимог і довів свою здатність підтримувати узгодженість вимог. Інструменту необхідно розширити його функціональність, щоб виконувати деякі завдання керування проектами, як-от призначення завдань і графічне спостереження. Набір представлених рекомендацій розширить функціональність інструменту та підвищить зручність його використання.

РОЗДІЛ 3. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 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].

3.2 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 вібромічність, 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.

ВИСНОВОК

У цій роботі ми застосували метод систематичного огляду, щоб проаналізувати літературу, пов’язану з темою DSD, основною метою якого є створення нової моделі DSD, за допомогою якої можна керувати взаємовідносинами між центром планування та проектування та створення архітектури допоміжного програмного інструменту зі створення програмного забезпечення. Ця робота служить відправною точкою для визначення проблем, на яких буде зосереджено подальше дослідження.

Результати, отримані в результаті цього систематичного огляду, дозволили нам отримати глобальне бачення відносно нової теми, яку слід детально дослідити. Проте кожна організація має конкретні потреби, які в основному залежать від її характеристик розподілу, її діяльності та інструментів, які вона використовує. Таким чином, ці фактори роблять цей предмет надзвичайно широким і призводять до необхідності адаптації як технічних, так і організаційних процедур відповідно до конкретних потреб кожної організації.

Пропозиції, знайдені в проаналізованих дослідженнях, загалом стосувалися вдосконалень, пов’язаних із використанням інструментів для співпраці, інтеграцією існуючих інструментів, керуванням вихідним кодом або використанням агентів для співпраці. Крім того, слід підкреслити, що оцінка результатів, отриманих від запропонованих покращень, часто базується на дослідженнях в одній організації, а іноді враховує лише суб’єктивне сприйняття розробників.

Важливими факторами успіху стають застосування гнучких методологій, заснованих на поетапній інтеграції та частих поставках, а також частий перегляд проблем для коригування процесу. Ми також виявили зростаючий інтерес до моделювання в розробці програмного забезпечення та підходів MDA як засобу підвищення продуктивності, якості та розуміння серед учасників, залучених до процесу розробки.

Нарешті, ми повинні підкреслити, що пошук було зведено до обмеженої кількості пошукових систем і виключено дослідження, які стосувалися теми DSD, але не внесли жодного значного методу чи покращення в цьому контексті дослідження. Однак, оскільки це така широка область, деякі з цих робіт представляють цікаві паралельні теми для розвитку цього дослідження, і тому їх вивчення буде важливим у майбутній роботі.

У цій роботі вивчалися різні підходи та практики щодо глобальної архітектурної розробки (GAD). Виявилось, що в GAD бракує практик і підходів до управління вимогами. Більшість існуючих досліджень у літературі представлені у формі звітів про промисловий досвід. Цей документ поєднав модель функцій із промисловим підходом, щоб забезпечити рішення, яке може керувати вимогами в GAD. Автори модифікували модель функцій для обробки процесу зміни вимог. Модифікована модель функцій використовується для моделювання вимог до однієї системи та керування її змінами, тоді як оригінальна модель функцій використовується в основному для моделювання сімейства систем або лінійки програмних продуктів.

Модифікована модель ознак тут називається деревом функцій. Дерево функцій інтегровано з NAPDP, який є підходом для розподіленої гнучкої розробки. NAPDP складається з двох фаз: фази ініціації проекту та фази циклу розробки. Управління змінами вимог відбувається у дві фази: починається з фази ініціації проекту та продовжується через фазу циклу розробки. Цей підхід пов’язаний із допоміжним інструментом, який допомагає його теоретичному баченню. Цей інструмент є веб-інструментом для підтримки розподіленої гнучкої розробки. Інструмент допомагає керувати змінами вимог і допомагає керівнику проекту відстежувати статус проекту на двох етапах. Інструмент підтримки оцінюється в реальному середовищі в NRC. Проект, який використовувався в оцінці, був системою управління документами, яка є системою бази даних середнього розміру. Результати оцінювання є

обнадійливими, оскільки вони базуються на вичерпному наборі критеріїв, розроблених професійними розробниками програмного забезпечення.

У майбутньому можна використовувати інструмент у різних проектах, щоб підвищити його продуктивність і надати йому більше функціональних можливостей. Застосування рекомендацій щодо оцінки інструменту підвищить зручність використання інструменту та дозволить використовувати його в багатьох проектах програмного забезпечення з різними розмірами та доменами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Волович, В., Береженко, Б. М., & Боднарчук, І. О. (2022). Задача проєктування програмної архітектури в процесах забезпечення якості. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 104-106.
2. Гузеляк, О., Шевчук, Ю., Береженко, Б. М., & Боднарчук, І. О. (2022). Програмна архітектура в розподілених командах гнучких проєктів. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 110-112.
3. Боднарчук, І., Харченко, О., Хоміцький, Б., & Шимчук, Г. (2019). Проектування архітектури програмних систем в проектах з гнучкими методами управління. Матеріали XXI наукової конференції Тернопільського національного технічного університету імені Івана Пулюя, 46-48.
4. Lloyd, Domia, Ramadan Moawad, and Mona Kadry. "A supporting tool for requirements change management in distributed agile development." Future Computing and Informatics Journal 2.1 (2017): 1– 9.
5. D. Creps, C. Klingler, L. Levine, and D. Allemang, "Organization domain modeling (ODM) guidebook version 2.0," Software Technology for Adaptable, Reliable Systems (STARS), 1996.
6. Jules White, Author Vitae, José A. Galindo, Author Vitae, Tripti Saxena Author Vitae, Brian Dougherty, Author Vitae, David Benavides, Author Vitae, Douglas C. Schmidt, "Evolving feature model configurations in software product lines", Journal of Systems and Software Volume 87, January 2014, Pages 119–136.
7. K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," DTIC Document1990.

8. Klaus Bergner, "DoSAM – Domain-Specific Software Architecture Comparison Model", DOI:10.1007/11558569_3· January 2005
9. M. M. S. Arefin and D. Korzun, "Improvement of the new agile process for distributed projects", 2010.
10. Mikko Korkala," Customer communication in distributed agile software development", VTT Science 80,2015.
11. K. Henrik, "Scrum and XP from the Trenches (Enterprise Software Development)," Lulu. com, second edition,2015.
12. K. Beck, Extreme programming explained: embrace change: addison-wesley professional, second edition, 2005.
13. Pedro Borges1, Paula Monteiro2, and Ricardo J. Machado2," Mapping RUP Roles to Small Software Development Teams", SWQD 2012, LNBIP 94, pp. 59–70, 2012.
14. Жидецький, В. Ц., Джигирей, В. С., & Мельников, О. В. (2000). Основи охорони праці. Львів: Афіша, 350, 132-136.
15. Навакатіян О.О., Кальниш В.В., Стрюков С.М. Охорона праці користувачів комп'ютерних відеодисплейних терміналів. - К.:1997. - 400с.