Міністерство освіти і науки України
Національний авіаційний університет

Кваліфікаційна наукова праця
на правах рукопису

ГРІНЕНКО СЕРГІЙ АНАТОЛІЙОВИЧ

УДК 004.05

МЕТОДИ ТА ЗАСІБ ОЦІНЮВАННЯ ЗРІЛОСТІ ПРОГРАМНИХ ПРОДУКТІВ

01.05.03 – математичне та програмне забезпечення
обчислювальних машин та систем

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ Гріненко С.А.

Науковий керівник – **Козловський Валерій Валерійович**,
доктор технічних наук, професор

Київ – 2021

АНОТАЦІЯ

Гріненко С.А. Методи та засіб оцінювання зрілості програмних продуктів. – На правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 01.05.03. – Математичне та програмне забезпечення обчислювальних машин і систем. – Національний авіаційний університет, Київ, 2021.

Дисертаційна робота присвячена розробці методів і засобу оцінювання зрілості програмних продуктів. У роботі обґрунтовано необхідність оцінювання зрілості при створенні програмного продукту. Для оцінювання зрілості програмних продуктів в процесі його розробки створено методи, які ґрунтується на автоматизованій багатокритеріальній оцінці зрілості. Застосування методів дозволяє ґрунтовно управляти зрілістю програмних продуктів з урахуванням ресурсів розробки. Розроблено математичні моделі оцінки та забезпечення зрілості програмних продуктів. Модель оцінки зрілості використання є трирівневою і ґрунтується на методиці оцінки кожного рівня даної моделі. Для дослідження програмних продуктів на зрілість була розроблена класифікація властивостей, які досліджуються відповідними методами, а також розроблені та реалізовані алгоритми для дослідження зрілості програмних продуктів. На основі запропонованих моделей та методів створено програмну систему, яка підтримує вирішення задачі оцінювання зрілості програмних продуктів, ґрунтуючись на автоматизованій оцінці з врахуванням критеріїв та метрик програмних продуктів.

Ключові слова: програмне забезпечення**,** зрілість програмного продукту, оцінювання зрілості програмного продукту, управління зрілістю програмних продуктів.

АННОТАЦИЯ

Гриненко С.А. Методы и средство оценивания зрелости программных продуктов. – На правах рукописи.

Диссертация на соискание ученой степени кандидата технических наук по специальности 01.05.03. – Математическое и программное обеспечение вычислительных машин и систем. – Национальный авиационный университет, Киев, 2021.

Диссертационная работа посвящена разработке методов и средства оценки зрелости программных продуктов. В работе обоснована необходимость оценки зрелости при создании программного продукта. Для оценки зрелости программных продуктов в процессе его разработки созданы методы, которые основывается на автоматизированной многокритериальной оценке зрелости. Применение методов позволяет основательно управлять зрелостью программных продуктов с учетом ресурсов разработки. Разработаны математические модели оценки и обеспечения зрелости программных продуктов. Модель оценки зрелости использования является трехуровневой и основывается на методике оценки каждого уровня данной модели. Для исследования программных продуктов на зрелость была разработана классификация свойств, которые исследуются соответствующими методами, а также разработаны и реализованы алгоритмы для исследования зрелости программных продуктов. На основе предложенных моделей и методов создано программную систему, которая поддерживает решения задачи оценки зрелости программных продуктов, основываясь на автоматизированной оценке с учетом критериев и метрик программных продуктов.

Ключевые слова: программное обеспечение, зрелость программного продукта, оценивание зрелости программного продукта, управление зрелостью программных продуктов.

ANNOTATION

Grinenko S.A. Methods and tool for assessing the maturity of software products. – As a manuscript.

Dissertation for obtaining a candidate degree in technical sciences of the speciality 01.05.03. – Mathematical and software of computing machines and systems. – National Aviation University, Kyiv, 2021.

The dissertation is devoted to the development of methods and tools for assessing the maturity of software products. The paper substantiates the need to assess maturity when creating a software product. To assess the maturity of software products in the development process, methods have been created that are based on an automated multi-criteria assessment of maturity. The use of methods allows you to thoroughly manage the maturity of software products, taking into account development resources. Mathematical models for assessing and ensuring the maturity of software products have been developed. The use maturity assessment model is three-tier and is based on the assessment methodology for each level of the model. To study software products for maturity, a classification of properties was developed, which are investigated by appropriate methods, a and developed and implemented algorithms for studying the maturity of software products. On the basis of the proposed models and methods, a software system has been created that supports solutions to the problem of assessing the maturity of software products, based on an automated assessment taking into account the criteria and metrics of software products.

Keywords: software, software product maturity, software product maturity assessment, software product maturity management.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях України:

1. Гріненко О.О., Гріненко С.А. Концептуальні основи моделювання екосистем програмного забезпечення / О.О. Гріненко, С.А. Гріненко // Науковий журнал. Наукові записки Українського науково-дослідного інституту зв'язку. – 2017. – №1(45). – С. 94 – 103. ***Внесок автора***: запропоновано модельно-орієнтований підхід (MDA) для моделювання екосистем програмного забезпечення.

2. Гріненко С.А. Інформаційна технологія забезпечення сталого розвитку ІТ-підприємств / С.А. Гріненко // Науковий журнал. Технічні науки та технології. – 2019. – № 3(17). – С. 140 – 145.

3. Поперешняк С.В., Гріненко С.А. Модель оцінки ІТ підприємств, яка заснована на інноваційній зрілості персоналу. / С.В. Поперешняк, С.А. Гріненко // Науковий журнал. Вісник інженерної академії. – 2019. – № 3. – С. 162 – 168. ***Внесок автора:*** запропоновано методику оцінювання ІТ підприємств на основі інноваційної зрілості персоналу.

Статті у наукових фахових виданнях України, які входять до міжнародних наукометричних баз даних:

4. Kozlovskiy V.V., Grinenko S. A., Skalova V.A. Models of Evaluation of Software Maturity / V.V. Kozlovskiy, S.A. Grinenko, V.A. Skalova // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 3(27). – С. 38 – 43. ***Внесок автора***: встановлено переваги та недоліки існуючих моделей оцінювання зрілості програмного забезпечення.

5. Grinenko S. A. Mathematical model of providing and software maturity assessment / S.A. Grinenko // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 4(28). – С. 5 – 14.

6. Kozlovskiy V.V., Grinenko S.A. The Measurements of Software Product Maturity / V.V. Kozlovskiy, S.A. Grinenko // Науковий журнал. Інженерія програмного забезпечення. – 2017. – № 2(30). – С. 52 – 58. ***Внесок автора***: запропоновані методи оцінювання зрілості ПП.

7. Popereshnyak S., Grinenko S. Development of an Ontological Model for the Domain of IT Enterprise Sustainable Development / S. Popereshnyak, S. Grinenko // Науковий журнал. Технологічний аудит та резерви виробництва. – 2019. – №3/2(47). – С. 43 – 45. ***Внесок автора***: розроблена онтологія ПрО сталого розвитку ІТ підприємств.

Статті у закордонних виданнях, які входять до міжнародних наукометричних баз даних:

8. Popereshnyak S., Grinenko S. Model of Innovative Maturity of Personnel for Estimation of IT Enterprises / S. Popereshnyak, S. Grinenko // Науковий журнал. East european scientific journal. – 2019. – Vol.5. – №10 (50). – С. 33 – 40. ***Внесок автора:*** запропоновані моделі для оцінювання ІТ підприємств на основі інноваційної зрілості персоналу.

9. Grinenko O., Grinenko S. One approach to study software ecosystem properties / O.Grinenko, S. Grinenko // International Journal “Information Theories and Applications”, Vol. 26, Number 4, 2019. – С. 397 – 405. ***Внесок автора:*** запропоновані характеристики для дослідження екосистем програмного забезпечення.

10. Popereshnyak S., Grinenko S., Grinenko O. Kovalenko O., Radivilova T. The Methodology for Assessing the Maturity Level of Software Ecosystems / S. Popereshnyak, S. Grinenko, O. Grinenko, O. Kovalenko, T. Radivilova // Proceedings of the 1st International Workshop on Cyber Hygiene & Conflict Management in Global Information Networks. – 2019. – ceur Vol-2654. – pp. 251 - 261. ***Внесок автора:*** розроблено 3-рівневу модель для оцінювання екосистем програмного забезпечення.

Тези наукових конференцій:

11. Grinenko S.A. Ecological Approach for Research of Software / S.A. Grinenko // XIX Всеукраїнська наукова конференція молодих учених «Актуальні проблеми природничих і гуманітарних наук у дослідженнях молодих учених», Черкаси, 27-28 квітня 2017 р. – Черкаси: Черкаський національний університет ім. Б.Хмельницького, 2017. – С. 202.

12. Grinenko S.A. Theoretical Aspects of the origin and Development of Software Ecosystems / S.A. Grinenko // Міжнародна науково-технічна конференція аспірантів та студентів «Інженерія програмного забезпечення – 2017», Київ, 06-09 червня 2017 р. – Київ: Національний авіаційний університет, 2017. – С. 12.

13. Grinenko S.A. Information technology for assessment of software ecosystem maturity levels / S.A. Grinenko // XIV Міжнародна науково-технічна конференція «АВІА – 2019», Київ, 23-25 квітня 2019 р. – Київ: Національний авіаційний університет, 2019. – С. 6.27 – 6.29.

14. Grinenko S.A. Mathematical Model of IT Ecosystem Based on Transition Systems / S.A. Grinenko // XV Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту – 2019», Херсон – Залізний Порт, 21-25 травня 2019 р. – Херсон: Херсонський національний технічний університет, 2019. – С. 47 – 48.

15. Grinenko S.A. Principles of Sustainable Development in IT industry / S.A. Grinenko // Міжнародна науково-технічна конференція «Інформаційні технології в освіті та науці – 2019», Мелітополь, 13-14 червня 2019 р. – Мелітополь: Мелітопольський державний педагогічний університет ім. Б.Хмельницького, 2019. – С. 88 – 91.

16. Grinenko S.A. One Approach for Evaluation the Maturity of IT Enterprise / S.A. Grinenko // III Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві

– 2019», Київ, 30 вересня 2019 р. – Київ: Київський національний університет імені Тараса Шевченка, 2019. – С. 202.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

CMMI – Capability Maturity Model Integration

CMMI-SW – Capability Maturity Model Integration for Software Engineering

IDEF5 – Integrated Definition for Ontology Description Capture Method

OSMM – Open Source Maturity Model

SCMM – Software Component Maturity Model

SEER – Scenario Exploration, Elaboration and Review

SMmm – Software Maintenance Maturity Model

SMM – SCOPE Maturity Model

SPMM – Software Product Maturity Model

SPQMM – Software Product Quality Maturity Model

TMM – Testing Maturity Model

UMM – Usability Maturity Model

UML – Unified Modeling Language

БД – база даних

ПП – програмний продукт

ВСТУП

Актуальність теми. Зрілість (maturity) - це важлива характеристика будь-якого продукту, який використовує людина. Не виключенням є і зрілість програмних продуктів (ПП) (software product maturity). З однієї точки зору зрілість ПП може визначати затрачені ресурси для досягнення цілей, тобто впливає на продуктивність праці людини – і це вкрай важливо для користувача ПП, а з іншого боку, зрілість є одним із визначних факторів при виборі ПП, тобто пов'язана з його конкурентоспроможністю – і це важливо для розробника ПП. Тому забезпечення зрілості ПП є актуальним для всіх зацікавлених сторін. Важливість зрілості підкреслюється ще й тим, що вона є характеристикою моделі якості ПЗ, і тому представлена в стандартах ISO/IEC ISO/IEC 25010:11, ISO/IEC 25022, ISO/IEC 25023, ISO 12119, ISO/IEC 9126 та ISO 14598.

Дослідженням зрілості приділяють значну увагу зарубіжні вчені: B. Boehm, A. Cooper, Yo. Kuwata, R. Al Qutaish, J. McCall, J. Nielsen, , J. Scholtz, а також українські вчені П. Андон, Б. Конорєв, К. Лавріщева, І. Туркін, О. Харченко.

З огляду на важливість зрілості ПП її необхідно досягати та підтримувати, тому в процесі розробки та супроводу ПП виконуються дії, які забезпечують створення зрілого для використання ПП. Оскільки зрілість є характеристикою якості ПП, то управління зрілістю відбувається в рамках управління якістю ПП. При цьому досягнення зрілості ускладнюється протиріччями або повнотою уявлень розробників та користувачів про властивості зрілого ПП; необхідністю досягнення зрілості ПП у процесі його розробки; відсутністю методів для управління зрілості ПП у технологіях створення ПП.

Враховуючи істотній внесок науковців з предметної області (ПрО) в розробку теоретичних та положень та практичних рекомендацій, можна зробити висновок, що результати їх досліджень полягають в дослідженні

окремих складових зрілості ПП або зрілості процесів розробки ПП. При цьому дослідження зрілості ПП вцілому та оцінки її рівня відсутні.

Відсутність у керівництвах зі створення ПП моделей, методів, методик для забезпечення зрілості ПП вцілому призводить до труднощів у ефективній інтеграції процесів для досягнення, управління зрілістю та створення ПП. Крім цього, це не дозволяє відслідковувати проблеми зрілості ПП на початку їх створення, що призводить до збільшення витрат на пошук шляхів їх вирішення в майбутньому.

Існуючі методи досягнення зрілості ПЗ стосуються окремих процесів управління зрілістю – планування та контролю. Тому особливої актуальності набуває задача розробки методів та засобу управління зрілістю ПЗ, які б дозволили забезпечувати зрілість ПП у процесі їх створення та супроводження, беручи до уваги вищенаведені фактори. Саме в цьому полягає актуальність дисертаційної роботи.

Зв'язок роботи з науковими програмами, планами, темами. Дисертацію виконано в Національному авіаційному університеті на кафедрі інженерії програмного забезпечення в рамках таких науково-дослідних робіт, у яких Гріненко С.А. брав участь: №586–ДБ009 «Екологія програмного забезпечення» (держ. реєстр. № 0109U001769), «Методи та засоби інженерії програмного забезпечення» (№24 Ф4/К44), № 29/09.01.02 «Онтології в інженерії програмного забезпечення», 58/09.01.02 «Методологія підвищення ефективності процесів життєвого циклу розробки програмного забезпечення у гнучких підходах його розробки» (2019-2022р.).

Мета та задачі дослідження. Метою дисертаційного дослідження є вирішення важливого науково-технічного завдання оцінки зрілості ПП у процесі створення та супроводу ПП за допомогою розробки методів оцінювання зрілості ПП. Методи були реалізовані у вигляді програмного засобу.

Для досягнення згаданої мети в роботі поставлено та розв'язано такі задачі:

- вивчення ПрО та аналіз наявних моделей, методів та підходів до забезпечення зрілості ПП;
- розробка моделі для досягнення зрілості ПП;
- розробка методів оцінювання зрілості ПП, що забезпечує задану розробником зрілість відповідно до розробленої моделі;
- розробка методики дослідження узгодженості даних від користувачів та/або експертів ПрО щодо зрілості ПЗ;
- розробка методики визначення наявності і форми залежності між показниками зрілості ПП;
- розробка архітектури та програмного засобу для оцінювання зрілості, яка реалізує запропоновані методи;
- апробація запропонованих методів та програмного засобу за допомогою емпіричного дослідження.

Об'єкт дослідження – оцінювання зрілості програмних продуктів.

Предмет дослідження – моделі, методи і засоби оцінювання зрілості програмних продуктів.

Методи дослідження. Для досягнення поставленої мети в роботі використано такі методи: аналіз, аналогія, синтез та узагальнення – при дослідженні ПрО та підходів до формалізованого представлення понять; та моделювання; статистичні методи аналізу даних; об'єктно-орієнтований аналіз і програмування – при проектуванні та конструюванні програмного засобу оцінювання зрілості ПП; методи емпіричної інженерії програмного забезпечення – анкетування та експертне оцінювання на базі SEER – при розв'язанні задачі застосування запропонованих в роботі методів і засобу.

Наукова новизна роботи полягає в розробці нових, уточненні і доповненні наявних науково-методичних положень та практичних рекомендацій з оцінювання зрілості ПП:

вперше:

- на основі процесного підходу до управління якістю ПП створено модель зрілості ПП, яка, охоплюючи задачі планування, контролю,

забезпечення й управління зрілості ПП, спрямована на їх розв'язання за допомогою застосування формального апарату теорії прийняття рішень. Застосування методів дозволяє ґрунтовно управляти зрілістю ПП з урахуванням ресурсів розробки;

удосконалено:

- математичну модель оцінювання рівнів зрілості на основі ключових практик, цілей та областей процесів, тобто використовуються ієрархічні системи нечіткого логічного висновку. Застосування моделі дозволяє в разі невідповідності рівня зрілості встановленим вимогам здійснити коригуючі дії з досягнення відповідного рівня в результаті рішення оптимізаційної задачі;

отримало подальший розвиток:

- розв'язання задачі контролю зрілості шляхом створення математичної моделі оцінки зрілості ПП на кожному із рівнів моделі. Застосування моделі дозволяє контролювати відповідність зрілості ПП встановленим вимогам та створити на її основі математичну модель забезпечення зрілості ПП.

Практичне значення одержаних результатів роботи полягає в тому, що впровадження в процес розробки ПП запропонованих у дисертації математичних моделей, методів та програмного засобу сприяє підвищенню ефективності управління проектами за відповідними критеріями зрілості, а також конкурентоспроможності ПП.

На базі створених методів та моделі спроектовано та реалізовано ПС для оцінювання зрілості ПП.

Результати дисертації у вигляді моделі та методів та розробленого засобу оцінки зрілості ПП ІТ компаній впроваджені в таких організаціях: Національному авіаційному університеті – НДР №586–ДБ009 «Екологія програмного забезпечення» (держ. реєстр. № 0109U001769), «Методи та засоби інженерії програмного забезпечення» (№24 Ф4/К44), № 29/09.01.02 «Онтології в інженерії програмного забезпечення», 58/09.01.02

«Методологія підвищення ефективності процесів життєвого циклу розробки програмного забезпечення у гнучких підходах його розробки» (2019-2022р.) (акт впровадження від 23.12.2020) та в навчальному процесі при проведенні лабораторних робіт згідно з програмами навчальних дисциплін «Екологія програмного забезпечення», «Дослідження технологій розробки та супроводу програмного забезпечення систем», «Моделювання процесів зрілості в інженерії програмного забезпечення» за напрямом професійної підготовки 121 «Інженерія програмного забезпечення» (акт впровадження від 26.11.2020); ТОВ «Lime Systems» шляхом застосування рекомендацій щодо застосування принципів зрілості для компанії та окремих проектів при розробці ПП (акт впровадження від 06.11.2020).

Особистий внесок здобувача. Усі результати, які виносяться на захист, отримані здобувачем особисто й опубліковані в одноосібних підготовлених працях [2, 6, 8-13]. У роботах [1, 3-5, 7, 14-15], написаних у співавторстві, автору належить: модельно-орієнтований підхід (MDA) для моделювання екосистем програмного забезпечення, методика оцінювання ІТ підприємств на основі інноваційної зрілості персоналу, встановлення переваг та недоліків існуючих моделей оцінювання зрілості програмного забезпечення, пропозиція методів для оцінювання зрілості ПП, розроблена онтологія ПрО сталого розвитку ІТ підприємств, пропозиція моделей для оцінювання ІТ підприємств на основі інноваційної зрілості персоналу, запропоновані характеристики для дослідження екосистем програмного забезпечення, розроблена 3-рівнева модель для оцінювання екосистем програмного забезпечення.

Апробація роботи. Результати роботи доповідалось і обговорювались на XIX Всеукраїнській науковій конференції молодих учених «Актуальні проблеми природничих і гуманітарних наук у дослідженнях молодих учених» (Черкаський національний університет ім. Б.Хмельницького, м. Черкаси, 2017 р.), Міжнародній науково-технічній конференції аспірантів та студентів «Інженерія програмного забезпечення – 2017» (Національний

авіаційний університет, м. Київ, 2017 р.), XIV Міжнародній науково-технічній конференції «АВІА – 2019» (Національний авіаційний університет, м. Київ, 2019 р.), XV Міжнародній науковій конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту – 2019» (Херсонський національний технічний університет, м. Херсон – Залізний Порт, 2019 р.), Міжнародній науково-технічній конференції «Інформаційні технології в освіті та науці – 2019» (Мелітопольський державний педагогічний університет ім. Б.Хмельницького, м. Мелітополь, 2019 р.), III Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві – 2019» (Київський національний університет імені Тараса Шевченка, м. Київ, 2019 р.), 1st International Workshop on Cyber Hygiene & Conflict Management in Global Information Networks (Національний авіаційний університет, м. Київ, 2019 р.), а також на наукових семінарах Національного авіаційного університету.

Публікації. Основні результати дослідження опубліковано в шістнадцяти наукових роботах [1-16], десять з яких [1-10] – у наукових фахових виданнях і шість [11-16] – у збірниках тез доповідей наукових конференцій. З них одна [10] – у виданні, що індексується наукометричною БД Scopus, одна [7] – у виданні, що індексується наукометричною БД Index Copernicus та дві [8, 9] – у виданнях зарубіжних країн, що входять до ЄС, а також вісім [2, 5, 11-16] є одноосібними.

Структура і обсяг. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Обсяг роботи становить 152 сторінки, 195 джерел, містить 40 рисунків, 14 таблиць, 3 додатка.

РОЗДІЛ 1
ЗРІЛІСТЬ ПРОГРАМНИХ ПРОДУКТІВ

Розділ присвячено концептуалізації знань щодо зрілості програмних продуктів. Встановлені існуючі означення, метрики та моделі, які пов'язані з дослідження зрілості програмних продуктів. На основі проведеного аналізу зроблено висновок про необхідність створення моделей, методів та програмного засобу для оцінювання зрілості програмних продуктів.

1.1. Поняття «зрілості» в інженерії програмного забезпечення

В літературі з інженерії програмного забезпечення можна зустріти таке поняття, як «зрілість». В загальному значенні слова, під «зрілістю» розуміють [194]:

1. Стан організму, який досяг повного розвитку.

2. Високий ступінь розвитку, досконалості, майстерності.

Зрілість (maturity) - це важлива характеристика будь-якого продукту, який використовує людина. Не виключенням є і зрілість програмних продуктів (ПП) (software product maturity). З однієї точки зору зрілість ПП може визначати затрачені ресурси для досягнення цілей, тобто впливає на продуктивність праці людини – і це вкрай важливо для користувача ПП, а з іншого боку, зрілість є одним із визначних факторів при виборі ПП, тобто пов'язана з його конкурентоспроможністю – і це важливо для розробника ПП. Аналіз наукових джерел за допомогою Systematic Mapping Studies показав (рис. 1.1), що зрілість розглядається найчастіше з чотирьох точок зору: зрілість процесів розробки ПП, зрілість програмного коду, зрілість ПЗ, зрілість ПП. У [168] зрілість розглядається як підхарактеристика надійності ПЗ: «ступінь відповідності системи, продукту чи компонента потребам у надійності при нормальній роботі». При цьому в примітках до даної характеристики говориться: «Поняття зрілості може також застосовуватися до інших характеристик якості, щоб вказати ступінь, наскільки вони відповідають необхідним потребам при нормальній експлуатації». Щодо

зрілості процесів, то існують наступні моделі оцінювання зрілості процесів. Наприклад, модель CMMI (Capability Maturity Model Integration) – це модель вдосконалення ПЗ, яка спрямована на вдосконалення процесів розробки ПЗ. Крім цього, зрілість часто ототожнюються або пов’язують з якістю ПП.CMMI фокусується на “якості процесів” розробки ПП замість “якості ПП”. Дослідження показали, що орієнтація лише на “якість процесу” не гарантує якості розробленого ПП, тоді як однакова увага до якості ПП є важливою для забезпечення загальної якості ПЗ. Метою даної дисертаційної роботи є представлення основ для дослідження та оцінювання зрілості ПП. Вимірювання, яке використовується для зрілості ПП, є перш за все рівень відповідності ПП внутрішнім і зовнішнім атрибутам якості, визначеним у вимогах зацікавлених сторін, які зосереджуються на якості процесів розробки ПП. Запропонована структура допоможе оцінити якість ПП шляхом оцінки кінцевого результату ПЗ. Успішна реалізація запропонованої основи забезпечить кращий аналіз якості ПП, а отже, і його зрілості.

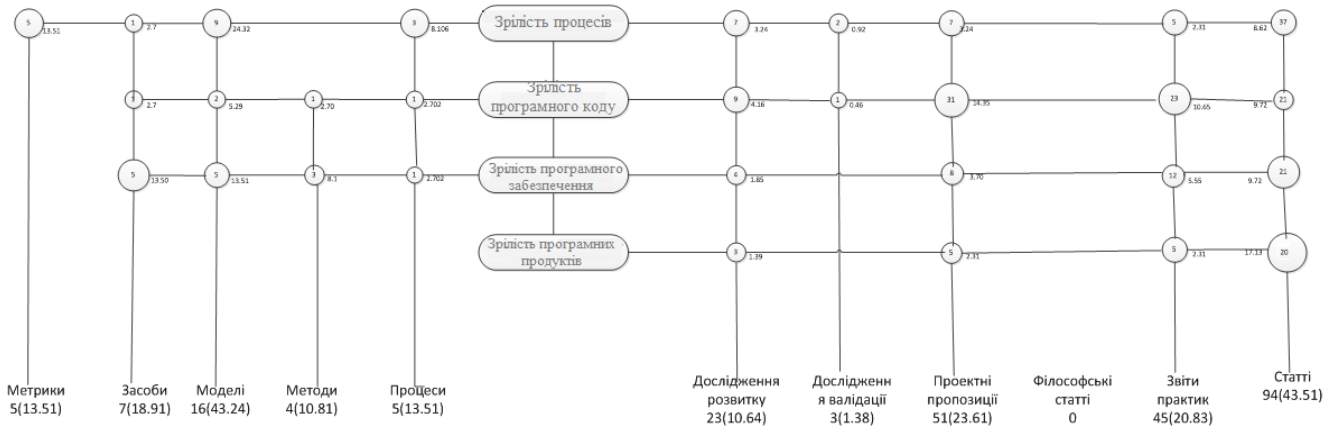

Рис. 1.1. Bubble Plot

Під моделлю зрілості розуміється структурована сукупність елементів, що описують характеристики ефективних процесів або продуктів. Крім того, модель зрілості може бути використана як еталон для оцінки різних процесів або продуктів для еквівалентного порівняння.

У літературі з програмного забезпечення є багато моделей зрілості. Ці моделі можливостей та зрілості можна класифікувати на моделі можливостей

процесу та зрілості, залежно від того, який аспект програмного забезпечення вони можуть застосовуватись.

Моделі зрілості процесів розробки програмного забезпечення є корисними, оскільки вони вказують на різні рівні продуктивності процесів і, отже, на напрямок, в якому повинен вдосконалюватися процес розробки ПЗ. До таких моделей зрілості відносяться:

1. Capability Maturity Model Integration for Software Engineering – CMMI-SW.
2. Testing Maturity Model – TMM.
3. ISO Process Assessment Model – ISO 15504.

Capability Maturity Model (CMM) була розроблена Інститутом програмної інженерії (SEI) Університету Карнегі-Меллона у відповідь на прохання надати Міністерству оборони США метод оцінювання своїх підрядників з розробки ПЗ. В оновлену версію Capability Maturity Model Integration (CMMI) на даний час входять чотири сукупності знань (дисциплін):

1. Системна інженерія.
2. Інженерія програмного забезпечення.
3. Інтегрована розробка продуктів і процесів.
4. Постачання продуктів.

В даній роботі розглядається лише Capability Maturity Model Integration for Software Engineering (CMMI-SW).

Рис. 1.2. Рівні зрілості згідно з CMMI-SW

Компонентами CMMI-SW є сфери процесів, конкретні цілі, конкретні практики, загальні цілі, загальні практики, типові робочі продукти, підпрактики, примітки, підсилення дисципліни, розробки загальної практики та посилання. CMMI-SW може організовувати процесні області у поетапному або постійному представництві. Поетапне представлення включає п'ять рівнів зрілості для підтримки та керівництва вдосконаленням процесів; крім того, згідно з цією моделлю відповідні процесні області групуються за рівнем зрілості, вказуючи, які області обробки застосовувати для досягнення кожного рівня зрілості. На рис. 1.2 представлена модель зрілості CMMI-SW.

Крім того, на рисунку 1.3 показані компоненти моделі CMMI-SW у поетапному поданні та проілюстровано взаємозв'язок між цими компонентами.

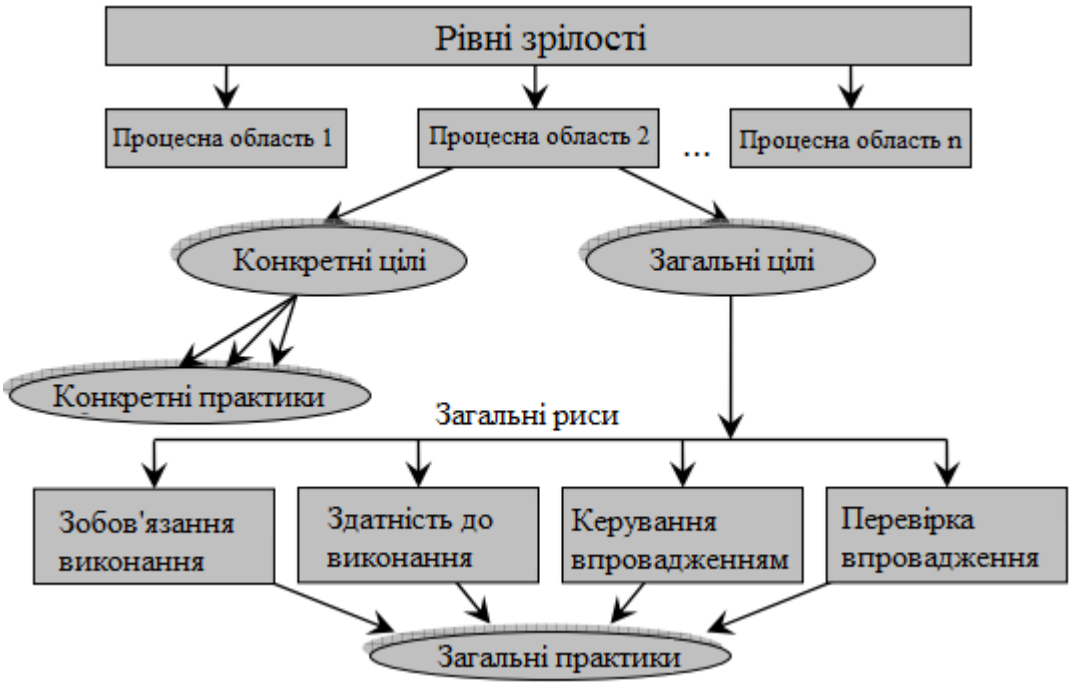

Рис. 1.3 Компоненти CMMI-SW

Testing Maturity Model

Модель Testing Maturity Model (TMM) пов'язана з процесом тестування ПЗ. Ця модель зрілості є кількісною, тобто базується на вимірюваннях. Ця модель була розроблена Burnstein et al. (1996 р.) в Іллінойському технологічному інституті. TMM складається з п'яти рівнів тестування на зрілість (рис. 1.4), кожен

з яких має цілі на зрілість, визначаючи вдосконалення тестування, які слід вирішити для досягнення наступного рівня зрілості.

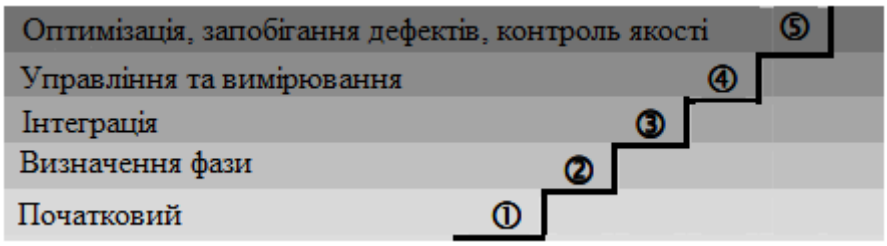

Рис. 1.4. Рівні моделі Testing Maturity Model

ISO 15504: оцінювання процесів ПЗ

ISO 15504 складається з сукупності документів, що стосуються оцінювання ПЗ, який був опублікований у 1998 році як серія з 9 технічних звітів. Протягом 2003-2005 років ISO повторно опублікував цей міжнародний стандарт, що складається з 5 частин:

1. ISO 15504-1: Concepts and Vocabulary.

2. ISO 15504-2: Performing an Assessment.

3. ISO 15504-3: Guidance on Performing an Assessment.

4. ISO 15504-4: Guidance on use for Process Improvement and Process Capability Determination.

5. ISO 15504-5: An Exemplar Process Assessment Model.

Перша частина – Concepts and Vocabulary – є вступом до ISO 15504. Вона дає вступ до понять цього міжнародного стандарту та визначає низку суміжних термінів. Крім того, у цій частині описано, як поєднуються інші чотири частини, та надано вказівки щодо їх вибору та використання. Рис. 1.5 показує потенційну дорожню карту для використання цього міжнародного стандарту.

Друга частина – Performing an Assessment – цього міжнародного стандарту містить нормативні вимоги до оцінювання процесу розробки, а також визначає межі вимірювання для оцінки можливостей процесу.

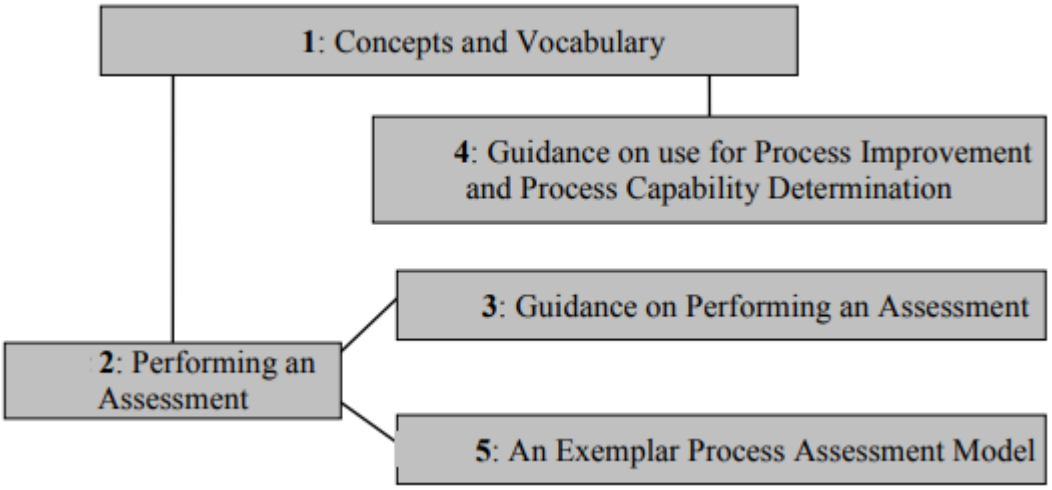

Рис. 1.5. Потенційна дорожня карта для використання ISO 15504

Система вимірювань визначає 9 атрибутів процесу, які згруповані у 6 рівнів здатності процесу, які визначають порядкову шкалу можливостей, застосовну до всіх вибраних процесів. Крім того, у цій частині описуються взаємозв'язки між компонентами моделі оцінки процесу, як на рис. 1.6.

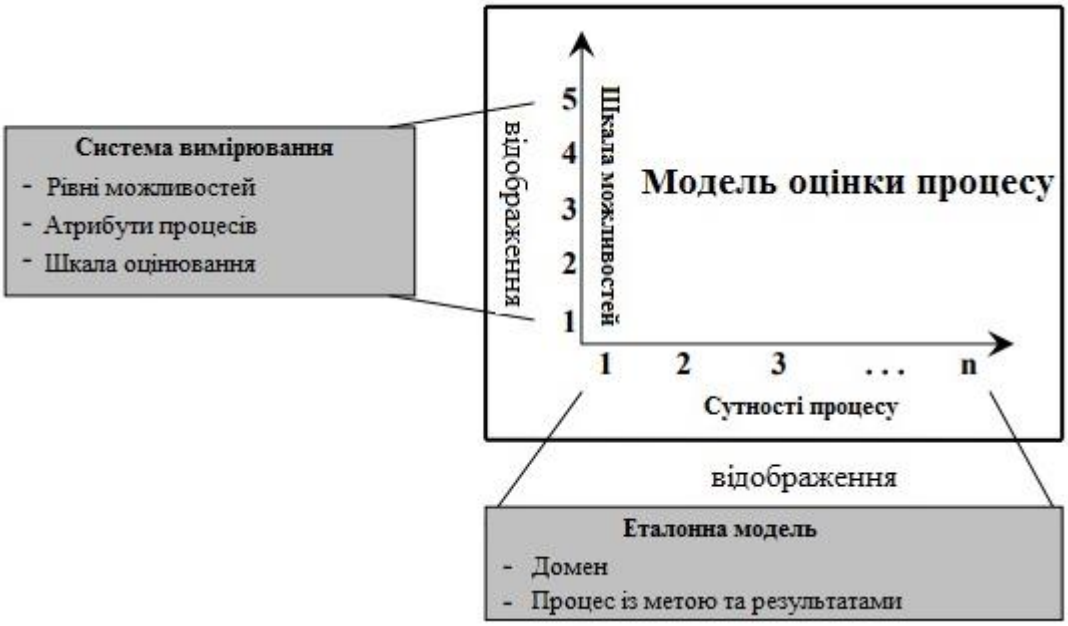

Рис. 1.6. Зв'язки моделі оцінки процесу

Рис. 1.7 ілюструє взаємозв'язок між атрибутами процесу та їх рейтингами та відповідними рівнями можливостей. На рис. 1.6 рівні можливостей починаються з

першого рівня, тобто виключається нульовий рівень, оскільки це вказує на те, що процес не реалізований або не може досягти своєї мети.

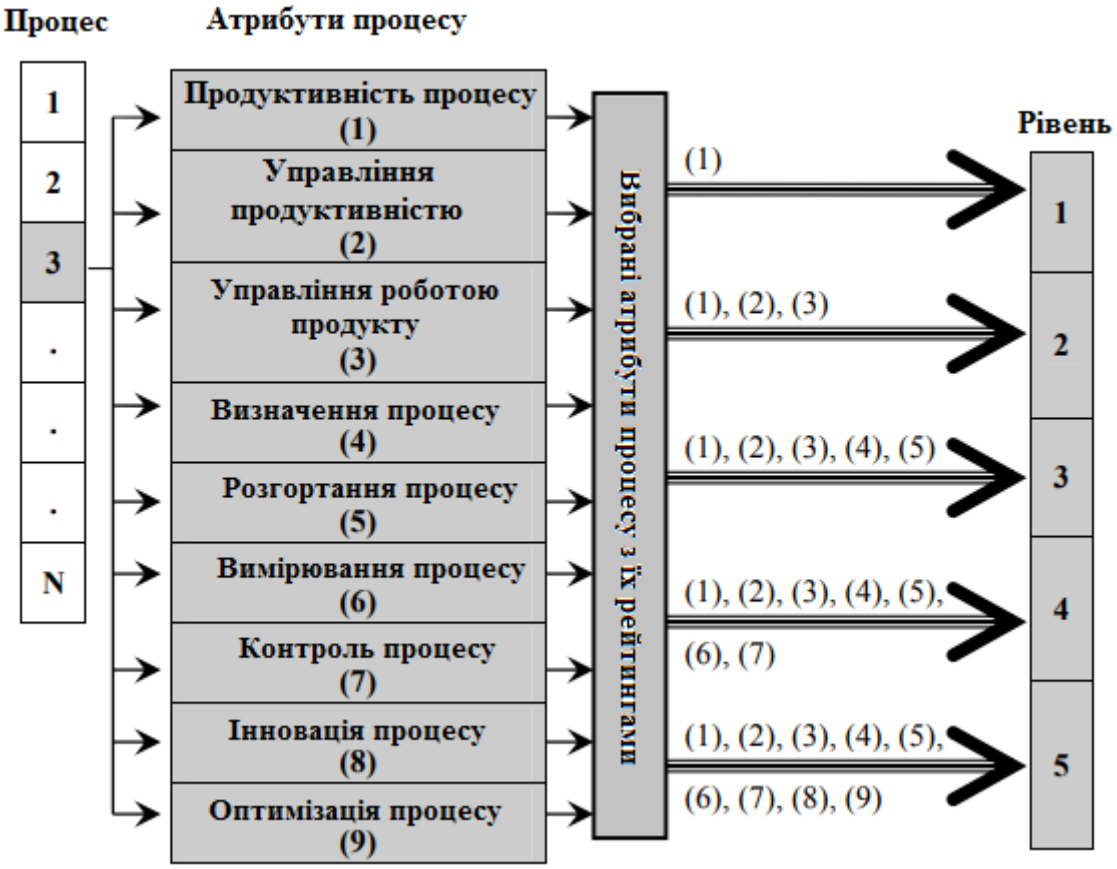

Рис. 1.7. Зв'язок між атрибутами процесу, їх рейтингами та відповідними рівнями можливостей

Крім того, дана частина – Performing an Assessment – ISO 15504 вводить наступні рейтингові категорії, які слід використовувати для оцінки кожного з атрибутів процесу:

• NA (Not achieved): Не досягнуто (досягнення від 0% до 15%).

• PA (Partially achieved): Частково досягнуто (досягнення 15% - 50%).

• LA (Largely achieved): Істотно досягнуто (досягнення 50% - 85%).

• FA (Fully achieved): Повністю досягнуто (85% - 100% досягнення).

Третя частина – Guidance on Performing an Assessment – містить вказівки щодо того, як задовольнити мінімальний набір вимог для проведення оцінювання, що міститься у другій частині – Performing an Assessment – цього стандарту. Вона

надає огляд оцінювання процесу та інтерпретує вимоги шляхом надання вказівок щодо:

1. Виконання оцінювання.
2. Структури вимірювання можливостей процесу.
3. Референтні моделі процесів та моделі оцінювання процесів.
4. Вибір та використання інструментів оцінювання.
5. Компетентність оцінювачів.
6. Перевірка відповідності.

Крім того, ця частина також забезпечує еталонний документований процес оцінювання.

Четверта частина – Guidance on use for Process Improvement and Process Capability Determination – містить вказівки щодо того, як використовувати оцінку відповідного процесу в рамках програми вдосконалення процесу або для визначення здатності процесу. У контексті вдосконалення процесу оцінка процесу забезпечує засіб, що характеризує організаційну одиницю з точки зору можливостей обраних процесів. Аналіз результатів оцінювання відповідності процесу бізнес-цілям організаційного підрозділу визначає сильні, слабкі сторони та ризики, пов'язані з процесами. Крім того, це може допомогти визначити, чи ефективні процеси у досягненні бізнес-цілей, і забезпечить рушій для вдосконалення. Визначення можливостей процесу пов'язане з аналізом результатів однієї або декількох відповідних оцінок процесу для виявлення сильних, слабких сторін та ризиків, пов'язаних із здійсненням конкретного проекту, використовуючи обрані процеси в рамках даної організаційної одиниці.

Нарешті, п’ята частина – An Exemplar Process Assessment Model – забезпечує еталонну модель для проведення оцінювання процесів, яка базується на та безпосередньо сумісна з еталонною моделлю процесу в Поправці 1 та Поправці 2 ISO 12207. Розмір процесу забезпечується зовнішньою еталонною моделлю процесу, яка визначає набір процесів, що характеризується твердженнями про мету процесу та результати процесу. Розмір можливостей базується на структурі вимірювань, визначеній у частині 2 – Performing an

Assessment – цього стандарту. Моделі оцінювання розширюють еталонну модель процесу та систему вимірювань шляхом включення множини показників ефективності та можливостей процесу.

Потенційними користувачами цієї сукупності стандартів є наступні (ISO, 2004a):

1- Оцінювачі (Assessors).

2- Набувачі (Acquirers).

3- Постачальники (Suppliers).

Наступні дві моделі для оцінювання зрілості пов’язані з управлінням проектами.

Згідно з **моделлю зрілості Берклі**, існує 5 рівнів розвитку організацій (рис. 1.8):

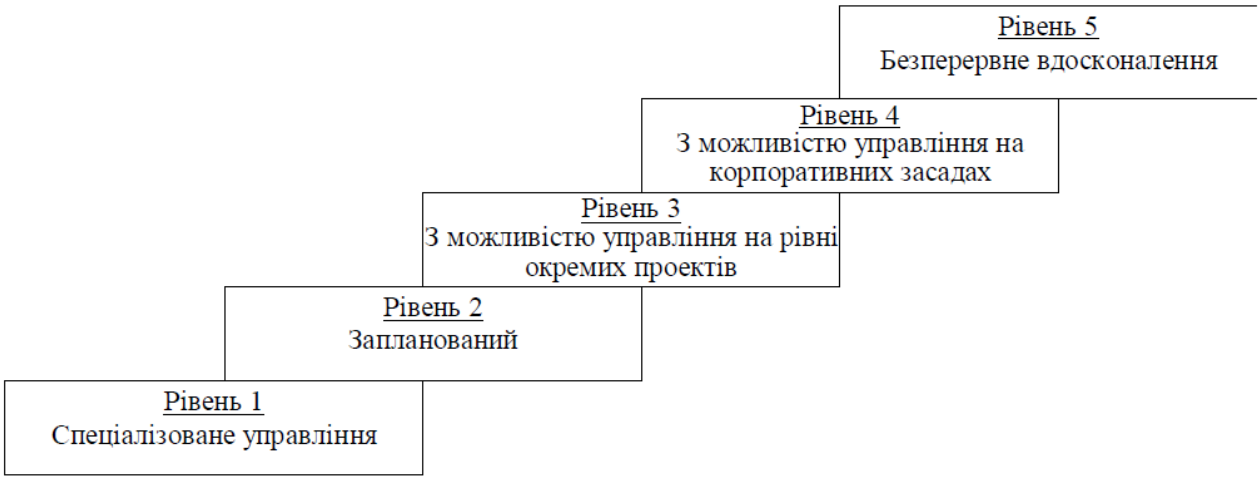

Рис. 1.8. Модель зрілості управління проектами Берклі

1. Спеціалізоване управління (відсутність розуміння з боку керівництва вигод, що дає управління проектами; нерегулярне, непослідовне або відсутнє застосування інструментів і методів управління проектами; поверхневі розрахунки за проектами в розрізі вартості робіт та термінів виконання);

2. Запланований рівень (володіння основами управління проектами на рівні проектної команди, часткова формалізація процесів управління проектами; процес планування проектів залежить від індивідуальних потреб менеджерів проектів; проекти підлягають перепланування, але причини змін, що зумовили

даний процес не аналізуються);

3. З можливістю управління на рівні окремих проектів (персонал підприємства володіє необхідними навиками і знаннями в галузі управління проектами; формалізовані процеси планування і контролювання проектів; проекти підлягають аналізуванню);

4. З можливістю управління на корпоративному рівні (системне навчання управлінню проектами на регулярній основі; здійснюється мультипроектне планування і контроль; повна формалізація і стандартизація процесів управління проектами; налагоджена взаємодія між проектними командами);

5. Безперервне вдосконалення (вдосконалення процесів управління проектами на постійній основі; використовується системний підхід до планування і контролювання проектів; впроваджуються інновації; проблеми, пов'язані з проектним управлінням усвідомлюються і вирішуються).

Перевагами моделей Берклі і Керцнера, на думку автора, є прозорість тлумачення кожного з етапів проектної зрілості підприємства, послідовне формулювання процесів вдосконалення; їх недоліком є те, що вони не враховують рівні управління програмами і портфелем проектів і програм.

Модель зрілості OPM3 містить такі рівні зрілості в управлінні проектами: управління проектами, управління програмами, управління портфелем проектів і програм [133].

Базисом моделі зрілості OPM3 є такі поняття:

– знання (кращі практики з управління проектами, що характеризують ті чи інші рівні організаційної зрілості);

– оцінка (інструмент для визначення зрілості управління проектами підприємства);

– покращення (інструмент формування системи управління проектами на засадах безперервного вдосконалення для забезпечення найефективнішого досягнення стратегічних цілей підприємства).

Позитивним моментом формування моделі зрілості OPM3 є врахування управління на рівні програм і портфелю проектів і програм як вищих етапів

розвитку проектного управління підприємства. Проте незіставність практик управління проектами іноземних підприємств для порівняння з вітчизняними значно звужує сфери застосування даної моделі.

Дослідження наукових джерел з інженерії ПЗ показали, що вищезгадані моделі відносяться до зрілості процесів розробки ПЗ або до зрілості управління проектами. Проте результат аналізу літератури показав, що на сьогодні існують наступні моделі зрілості, які стосуються ПП:

- Open Source Maturity Model – OSMM.
- Software Product Maturity Model.

Однак слід зазначити, що ці дві моделі зрілості ПП не стосуються якості цих продуктів.

Наведемо короткий опис цих двох моделей.

Open Source Maturity Model (OSMM) призначена допомогти організаціям успішно впровадити ПЗ з відкритим кодом. OSMM є трифазним процесом і виконує наступні завдання, як показано у табл. 1.1.

Перший етап складається з наступних етапів:

1. Визначення вимог.
2. Знаходження ресурсів.
3. Оцінювання зрілості елемента.
4. Визначення оцінки елемента.

Другий етап передбачає присвоєння об’єктивних вагових коефіцієнтів, які надаються як зважування за замовчуванням, які можуть бути змінені окремими організаціями, щоб відобразити їх особливі потреби.

Останній етап включає обчислення балу зрілості ПП шляхом множення балів кожного елемента на його вагу, а потім підсумовування результатів для отримання результату оцінки OSMM як числового балу від 0 до 100.

Таблиця 1.1

Процес трифазної оцінки OSMM

	Фаза 1: Оцінка зрілості елемента				Фаза 2	Фаза 3
	Визначити вимоги	Визначити ресурси	Оцінити зрілість елемента	Призначити оцінку елементу	Призначити ваговий коефіцієнт	Обрахувати оцінку зрілості ПП
ПП						
Підтримка						
Документування						
Навчання						
Інтеграція ПП						
Професійні послуги						

Цей бал можна порівняти з рекомендованими рівнями для різних цілей, які залежать від того, чи є організація першим, хто застосовує інформацію, або прагматичним користувачем інформаційних технологій (табл. 1.2) щодо рекомендованих мінімальних балів OSMM.

Таблиця 1.2

Рекомендовані мінімальні бали OSMM

	Тип користувача	
	Новачок	Прагматик
Експериментальна версія ПП	25	40
Дослідна версія ПП	40	60
Виробництво ПП	60	70

Використовуючи ключову концепцію зрілості ПЗ (тобто, на якій стадії життєвого циклу знаходиться розроблюване ПЗ, які зміни з ними відбуваються тощо), OSMM оцінює рівень зрілості таких ключових елементів ПП:

1. ПЗ.
2. Підтримка.
3. Документування.

4. Навчання.

5. Інтеграція ПП.

6. Професійні послуги.

OSMM розроблений як простий процес, який дозволяє оцінити зрілість ПП з відкритим кодом протягом двох тижнів або менше.

OSMM використовується для оцінювання зрілості ПП з відкритим кодом і враховує інші елементи, а не лише саме ПЗ, тобто: підтримку, документацію, навчання, інтеграцію продуктів та професійні послуги. Дійсно, ці елементи можуть вплинути на якість ПП, оскільки - наприклад, високоякісне ПЗ з низькою якістю документації може трактуватися як неякісний ПП з точки зору замовника.

OSMM спроектований для використання ПП з відкритим кодом після завершення їх розробки, тобто вони готові до випуску. Крім того, це переважно корисно, коли організації потрібно вибрати між різними ПП з відкритим кодом. Крім того, дана модель не грунтується на жодній моделі якості.

Software product maturity model (SPMM)

Ще однією моделлю зрілості ПП є модель, яка була запропонована Nastro у 1997р. Дана модель зрілості складається з трьох елементів ядра та двох піделементів, які можуть застосовуватися до конкретних ПП [136].

Елементами ядра моделі є:

1. Можливості ПП.

2. Стійкість ПП.

3. Супроводжуваність ПП.

А піделементами є:

1. Повторюваність ПП.

2. Сумісність ПП.

На основі підрахованого рівня зрілості кожного елемента з ядра та піделементів, Nastro запропонував таке рівняння для розрахунку рівня зрілості ПП вбудованої системи обробки сигналів у реальному часі:

$$PC * (PS + PR + PM) / 3 \qquad (1.1)$$

де:

PC – це рівень зрілості можливостей ПП,

PS – рівень зрілості стабільності ПП,

PR – рівень зрілості повторюваності ПП,

PM – це рівень зрілості супроводжуваності продукту.

У наведеному вище рівнянні PC має найбільшу вагу серед усіх елементів ядра та піделементів через свою критичність для цього застосування.

Слід зазначити, що дана модель має певні обмеження:

1. Модель застосовна лише до виконуваного ПП. Таким чином, вона може бути використана лише до ПП з інкрементним життєвим циклом, який забезпечує кілька релізів (версії) виконуваного ПП.

2. Модель грунтується не на довільній моделі якості, а лише на невеликій кількості характеристик якості продукту (5).

3. Модель розроблена для самого ПП, а не для якості ПП.

4. Для кожного елемента (елемента ядра або допоміжного) існує лише одна міра.

5. Модель розроблена для відслідковування та звітування про зусилля з розробки ПЗ протягом інкрементного життєвого циклу.

Software Product Quality Maturity Model (SPQMM)

Al-Qutaish та ін. [164] запропонували модель зрілості якості ПП (SPQMM) для оцінки якості ПП. Запропонована модель базується на ISO 9126, Six Sigma та ISO 15026. Модель використовує характеристики, підхарактеристики та вимірювання згідно з ISO 9126. Отримані виміряні значення об'єднуються в одне значення, яке перетворюється на шість сигм. Після цього обчислюється рівень цілісності ПП з використанням ISO 15026. Згодом, визначається рівень зрілості ПП. SPQMM обмежується атрибутами та показниками якості, визначеними стандартом ISO/IEC 9126.

SCOPE Maturity Model (SMM)

Консорціум EuroScope [147] запропонував SCOPE Maturity Model (SMM), модель зрілості оцінювання ПП. Модель має п’ять рівнів зрілості: початковий, повторюваний, визначений, керований та оптимізований. Рівні SMM 2, 3 та 4 використовують стандарти ISO 12119, ISO/IEC 9126 та ISO 14598. SMM - це показник якості з точки зору узгодження заявлених специфікацій або вимог; тести проводяться для оцінки ступеня, на якому ПП відповідає необхідним специфікаціям. SMM вимагає, щоб процес був задокументований, щоб забезпечити відповідність ПП специфікаціям. Таким чином, SMM не орієнтується на якість кінцевої продукції, а саме тобто код.

Software Maintenance Maturity Model (SMmm)

April та ін. [135] запропонував модель зрілості супроводження ПЗ (SMmm). Однак SMmm зосереджується лише на супроводжуваності.

Software Component Maturity Model (SCMM)

Alvaro та ін. [134] запропонував модель зрілості компонентів ПЗ (SCMM), яка базується на стандартах ISO/IEC9126 та ISO/IEC 14598. SCMM містить п'ять рівнів. SCMM залежить головним чином від моделі якості компонентів (CQM). SCMM вимірює лише зрілість компонентів і не може оцінити різні типи продуктів, такі як корпоративні програми, веб-послуги тощо.

Usability Maturity Model (UMM)

Дана модель грунтується на стандарті ISO TR 18529 [138] і застосовується до наступних процесів людино-орієнтованої розробки (Human-centered design processes, HCD-процеси), які необхідні для створення зручного у використанні ПП:

HCD.1 Забезпечення передумов для людино-орієнтованої розробки. Процес включає дослідження потреб користувачів, аналіз можливих змін в майбутніх потребах, визначення очікуваного контексту використання майбутнього ПП, оцінку вартості впровадження HCD-процесів.

HCD.2 Планування та управління. Основними діями є визначення та планування шляхів залучення користувачів до кожного етапу розробки ПП; вибір

методів людино-орієнтованої розробки та визначення їх місця в розробці ПП; управління участю користувачів в процесі розробки.

HCD.3 Визначення вимог зацікавлених сторін. Процес включає такі основні дії: визначення цілей, виконання яких буде забезпечувати майбутній ПП; аналіз важливості ПП для кожної з зацікавлених сторін; оцінка ризиків; визначення вимог до ПП з боку зацікавлених сторін; вибір критеріїв вимірювання ЗВ та відповідних підхарактеристик якості у використанні.

HCD.4 Визначення контексту використання. Процес складається з: визначення та документування характерних рис користувачів, їх завдань, особливостей соціального, технічного та фізичного середовища.

HCD.5 Розробка рішень зі створення зручного у використанні ПП. Основними діями є: розподіл функцій між людиною та ПП; опис варіантів використання ПП; прототипування; розробка сервісів ПП для підтримки користувача.

HCD.6 Оцінка отриманих результатів на відповідність вимогам. Метою є збір відгуків користувачів стосовно розроблених рішень. Вона досягається шляхом: визначення змісту оцінки; оцінки ранніх прототипів з метою коригування вимог до ПП; оцінки пізніх прототипів з метою їх покращення; оцінка ЗВ на відповідність вимогам зацікавлених сторін наприкінці розробки та в контексті використання.

HCD.7 Введення в експлуатацію та супровід з точки зору взаємодії користувача з ПП. Процес включає наступні дії: визначення впливу ПП на зацікавлених сторін; забезпечення навчання користувачів; підтримка користувачів.

У UMM описано шість рівнів зрілості організації-розробника, перебування на кожному з яких залежить від ступеня можливостей виконання HCD-процесів:

Рівень 0 Незавершений. Організація-розробник не підтримує людино-орієнтований процес розробки ПП.

Рівень 1 Виконуваний. Забезпечуються базові дії для досягнення зручності використання.

Рівень 2 Керований. HCD-процеси забезпечують прийнятні результати з врахуванням обмежень загальних ресурсів та часу.

Рівень 3 Встановлений. Процес людино-орієнтованої розробки та його ресурси чітко визначені організацією-розробником.

Рівень 4 Передбачуваний. HCD-процеси виконуються постійно в рамках визначених вимог до їх якості та ресурсів.

Рівень 5 Оптимізований. Процес людино-орієнтованої розробки може бути надійно адаптований до окремих вимог.

На виконуваному рівні UMM представлені усі HCD-процеси в початковому вигляді. Перехід організації на кожен наступний рівень зрілості з розробки зручного у використанні ПП відбувається через вдосконалення управлінських дій. Згідно таких дій, передбачених у UMM для п’ятого рівня зрілості, необхідно:

- визначати можливості покращень HCD-процесів;
- розробляти стратегію впровадження змін;
- підтверджувати ефективність виконаних змін.

Досягнення організацією оптимізованого рівня зрілості можливе лише при наявності процесного підходу до управління зручності використання, який має забезпечуватись відповідним методом і засобом (рис. 1.9). **HCD.6,** тобто процес оцінки отриманих результатів з забезпечення ЗВ на відповідність вимогам, на оптимізованому рівні викликає особливий інтерес.

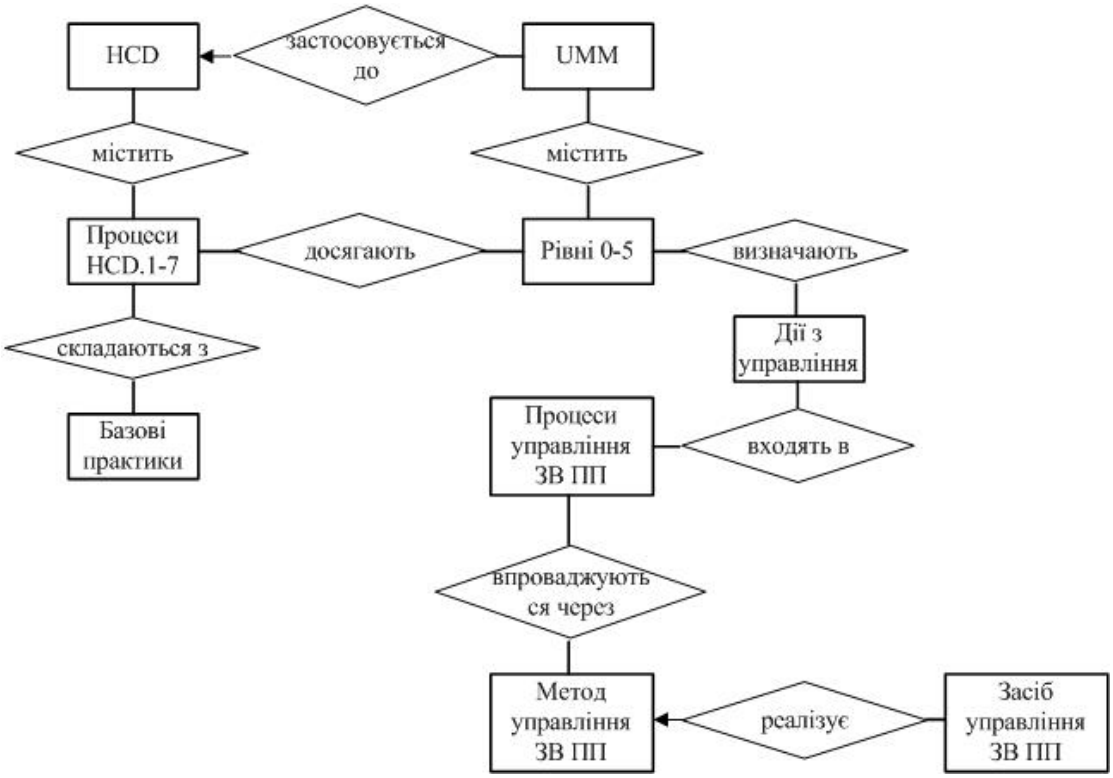

Рис. 1.9 Діаграма «сутність – зв’язок» для UMM (нотація Чена)

Підтвердження ефективності змін, враховуючи рекомендації ISO 9241-210 [167], вимагає забезпечення ітераційності оцінки зручності використання ПП і покращення процесу розробки шляхом оцінки, орієнтованої на користувачів. Тому для досягнення та підтримки зручності використання актуальним є розроблення методу та засобу управління зручності використання, які ґрунтуються на ітераційній оцінці зручності використання на основі відгуків користувачів. Управління зручністю використання певною мірою обмежено ресурсами організації або вимагає оптимального їх використання для досягнення позитивного економічного ефекту. Тому доцільно, щоб у методі також враховувалася оптимізація досягнення зручності використання.

1.2. Вимірювання та оцінювання зрілості програмних продуктів

Вимірювання та оцінювання зрілості ПП відіграють важливу роль в процесі розробки та супроводження ПЗ. Зрілість ПП можна розглядати як сукупність пов’язаних підхарактеристик (subcharacteristics), які утворюють базис для специфікації вимог до зрілості ПП та її оцінювання. Підхарактристикам

відповідають набори властивостей (properties) ПП, яким ставляться у відповідність міри (measures) [172]. Значення мір отримують у результаті застосування функції вимірювання (measurement function) до елементів мір (measure elements) (рис. 1.10).

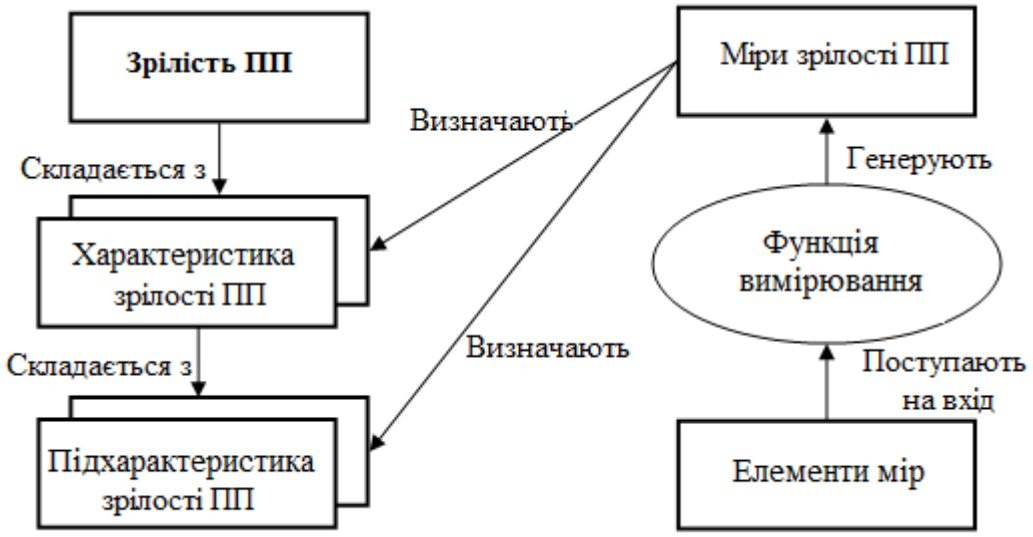

Рис. 1.10 Модель вимірювання зрілості

Вимірювання – це сукупність операцій, метою виконання яких є знаходження значення міри [163]. Функція вимірювання – це алгоритм, використовуваний для комбінації елементів мір. Елемент міри – це поєднання конкретного методу вимірювання та шкали вимірювання (засобу, використовуваного для структурування отримуваних значень); змінна, якій надається значення в результаті вимірювання. Метод вимірювання – це логічна послідовність операцій, що використовується для отримання значення властивостей згідно обраної шкали. Назви мір зазвичай однойменні назвам елементів мір.

Оцінювання (англ. evaluation) – це: 1) аналітичний інструмент або процедура, призначення якої – вимірювання прямих ефектів, результативності і довгострокових наслідків реалізації державних програм, галузевих політик, програм розвитку, проєктів некомерційного сектору, корпоративних програм; 2) міждисциплінарне дослідження, яке використовує економічні, соціологічні, політологічні методи у відповідності з визнаними стандартами національних і міжнародних спільнот у сфері оцінювання [178].

Для обробки даних, отриманих в результаті вимірювань, використовують різноманітні методи оцінки. У ISO/IEC 25040 [68] зазначається, що оцінка – це систематичне визначення ступеня відповідності сутності визначеним критеріям. Дамо наступне означення:

Оцінювання зрілості ПП – процес визначення реального стану ПП відносно бажаного стану з точки зору можливості використання певними користувачами для досягнення визначених цілей з продуктивністю, ефективністю та задоволеністю у певному контексті використання; а також результат такого процесу у числовій формі.

Під *станом* ПП розуміємо множину кількісно виражених властивостей, які визначають поведінку ПП.

Область оцінювання зрілості ПП узагальнюється як сукупність усіх форм методів, які можуть допомогти в розумінні шляхів досягнення зрілості ПП, проблем з використанням у всіх контекстах або навіть строків використання.

Залежно від джерела вхідних даних щодо зрілості ПП, методи оцінювання поділяються на 3 групи (рис. 1.11): оцінювання на основі відгуків кінцевих споживачів, експертне оцінювання та оцінювання на основі моделей [161]. Всі три методи пов’язані з діяльністю під час розробки та супроводу ПП та виконуються інженерами або фахівцями з предметної області.

Оцінка на основі відгуків кінцевих споживачів. Найстарішим джерелом оцінки зрілості ПП є відгуки користувачів. Вона здійснюється шляхом визначення типових користувачів, завдань та створення процедур для висвітлення проблем, які можуть виникати під час використання певного програмного ПП при виконанні тих чи інших завдань. Протягом етапів проектування/тестування/розробки програмного забезпечення проводиться два види оцінки: формуюча та підсумкова. Формуюча оцінка використовується для аналізу отриманої від користувачів інформації і подальшого її використання в проектуванні. Результати підсумкової оцінки використовуються для визначення ефективності, раціональності та рівня задоволеності користувача програмним

продуктом. Проведення обох видів оцінки є важливим для забезпечення людино-орієнтованої розробки [165].

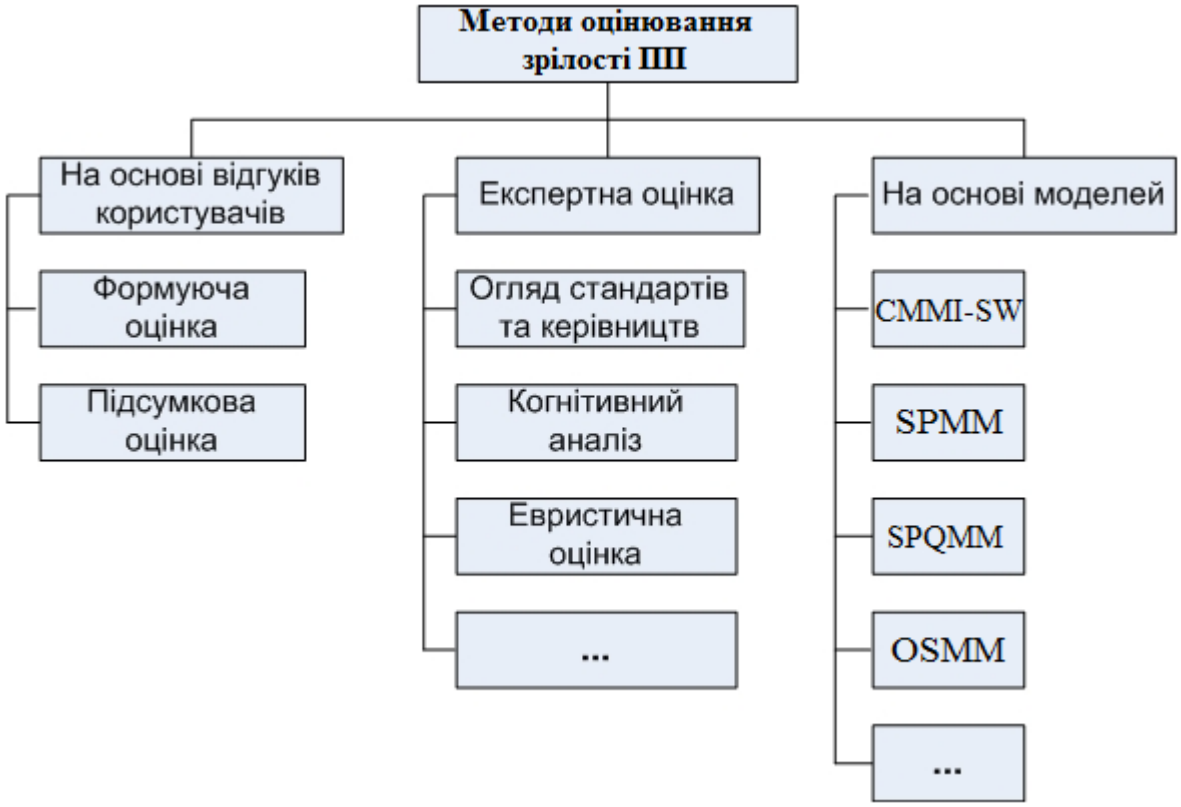

Рис. 1.11 Класифікація методів оцінювання зрілості ПП залежно від джерела вхідних даних

Формуюча оцінка. Направлена на отримання перших відгуків користувачів щодо ранніх концептів ПП та пов'язана здебільшого з користувацьким інтерфейсом. Основне джерело даних формуючої оцінки – словесні дані, отримані від користувачів. Для ранньої оцінки можуть використовуватися паперові прототипи або проекти початкових екранних форм. Пізня оцінка може здійснюватися на окремих прототипах або прототипах окремих компонентів інтерфейсу користувача. Якщо можливо, то використовується і спеціальне програмне забезпечення для реєстрації та відслідковування взаємодії з ПП. Опитування належить до одного з найкращих джерел отримання інформації для формуючої оцінки. Часто його використовують для визначення причин, які призводять до плутанини в інтерфейсі.

Підсумкова оцінка. Для неї характерна більша формалізація, ніж при формуючій оцінці. Проводиться для отримання числових значень під-характеристик зрілості ПП. Рекомендована кількість користувачів, які залучаються до оцінки, становить 5-7 осіб на один клас функціональних користувачів. Основою для ефективної підсумкової оцінки є докладно продуманий етап тестування: хто та за яких умов буде використовувати даний ПП. Завдання, які даються під час оцінки, як правило, спрямованні на випробування основних функціональних можливостей ПП, але можуть використовуватися і завдання для тестування нових або вдосконалених функцій. Вказівки та матеріали, які видаються користувачам, попередньо мають бути випробувані на пілотній версії оцінки. Це необхідно для того, щоб пересвідчитися, що матеріал викладено чітко та зрозуміло. Бажаний рівень ЗВ має визначатися на етапі проектування, а результати підсумкової оцінки порівнюються з ним.

Переваги та недоліки оцінки зрілості ПП на основі відгуків користувачів. Головна перевага такої оцінки – залучення користувачів до самого процесу, оскільки результат оцінки базується на підставі даних про вади, які призводять до проблем під час взаємодії типових користувачів з ПП. Таке залучення сприяє вирішенню протиріч представлень розробників та користувачів про властивості зручного у використанні ПП. Особливо ефективним є виконання підсумкової оцінки в процесі створення ПП, оскільки вона є основою для управління зрілістю ПП. Недоліки полягають в тому, що залучення користувачів витратна справа, як за коштами, так і за часом. Виникають труднощі з пошуком типових користувачів, які мають представляти окрему групу використання ПП. Проведення оцінки вимагає спеціального лабораторно-технічного обладнання та роботи експертів-аналітиків. Одним із складних є питання вибору завдань, що максимально відображають задачі, з якими зустрічається користувач повсякденно.

Експертне оцінювання. Методи експертного оцінювання почали використовуватися з 90-х років. Найбільш використовувані: огляд стандартів та керівництв з використання, когнітивний аналіз та евристичне оцінювання [170].

Ідея і зміст методу експертних оцінок – побудова раціональної процедури інтуїтивно-логічного мислення людини з кількісними методами оцінювання і обчислення отриманих результатів. При цьому висновки експертів, зроблені на базі науково-практичного досвіду, приймаються як розв'язання проблеми.

Задопомогою методів експертних оцінок вирішуються такі задачі:

- визначення цілей і задач управління із упорядкуванням їх за ступенем вагомосгі;
- оцінювання альтернативних варіантів прийняття рішень;
- перелік можливих подій у прогнозованому періоді;
- передбачення найбільш ймовірних моментів і періодів настання передбачуваних подій.

Серед методів експертних оцінок виділяють *методи індивідуальних (персональних)* та *колективних експертних оцінок.* Методи індивідуальних експертних оцінок полягають в отриманні незалежних оцінок від кожного експерта, тобто за умов відсутності впливу на експерта думок інших експертів, та обчисленні на їхній підставі інтегральної оцінки досліджуваного явища. Надійність інтегральної оцінки залежу від узгодженості висновків експертів.

Перевагу або відносну значущість прогнозованих варіантів розвитку систем можна встановити на підставі попарних порівнянь, безпосереднього оцінювання за допомогою ранжування.

Під час ранжування експерт для кожного з варіантів встановлює ранг – ставить у відповідність число з натурального ряду (кількість рангів відповідає кількості варіантів). Якщо експерт хоче привласнити декільком варіантам однаковий ранг, то кожному з них буде відповідати середній ранг – середнє значення з відповідних чисел натурального ряду. За міру узгодженості думок експертів вибирають *коефіцієнт конкордації,* який розраховують за такою формулою:

$$W = \frac{12 \sum_i^m S_i^2}{n^2(m^3 - m)} \tag{1.2}$$

де m – кількість варіантів;

n – кількість експертів;

$S_i = (d_i - \bar{d})$ – відхилення сумарного рангу i-го варіанта від середньої суми рангів $\bar{d}$.

R_{ij} – ранг, присвоєний i-му варіанту j-им експертом;

$d_i = \sum_{j=1}^{n} R_{ij}$ – сума рангів, присвоєних і-му варіанту (сумарний ранг і-го варіанта);

Коефіцієнт конкордації лежить у межах $0 < W < 1$. Відзначимо, що вищому рівню узгодженості відповідає більше значення W, яке наближається до 1. Дія перевірки істинності коефіцієнта конкордації використовують критерій χ^2 з $(m-1)$ ступенями вільності. Розрахункове значення критеріальної статистики $\chi^2_{\text{розр}} = Wn(m-1)$ порівнюють із табличним значенням для вибраного рівня ймовірності. Якщо розрахункове чинення перевищує табличне, то узгодженість думок експертів підтверджується.

Індивідуальні експертні оцінки можна отримати за допомогою інтерв'ю або аналітичних записок. У першому випадку відбуваються бесіди організатора експертизи (прогнозиста) з експертом згідно з розробленою заздалегідь програмою. У другому випадку передбачається можливість тривалої роботи експерта над поставленими перед ним запитаннями, а свої висновки він подає замовнику у вигляді аналітичної записки. При цьому у найбільшій мірі використовуються індивідуальні можливості експерт, глибоке усвідомлення процесу, який підлягає дослідженню.

Недоліком методу індивідуальних експертних оцінок вважають той факт, що далеко не кожен експерт може і бажає взяти на себе відповідальність за прогнозування складних явиш без урахування думок інших експертів. Цього недоліку позбавлені методи колективних експертних оцінок, які ґрунтуються на спільній роботі експертів. Серед найпоширеніших методів колективних експертних оцінок виділяють метоли сценарного розвитку, метод колективної генерації оцінок («мозкового штурму»), метод Дельфі, SEER тощо.

Методи *сценарного розвитку* базуються на визначенні логіки процесу

(явища) при різних умовах функціонування системи. Підготовку і узгодження передбачень тенденцій розвитку системи здійснюють на підставі написаних сценаріїв, які містять змістовні висловлювання, результати техніко-економічного і статистичного аналізів з відповідними висновками. У сценарії відображаються можливі варіанти розвитку подій , можливі труднощі і проблеми, послідовність розв'язання задач тощо. Завданням експертної групи є розроблення сценарію розвитку економічної системи – комплексу управлінських і організаційних заходів, пов'язаних єдиною програмою дій і спрямованих на вирішення великомасштабних проектів. Побудова сценарію ґрунтується на системному підході до дослідження проблеми і передбачає розчленування загальної проблеми (цілі) на окремі частини та виявлення можливих альтернатив їхнього розв'язання. Сценарний метод прогнозування передбачає розроблення прогнозів окремих частин (підцілей) з подальшим їхнім об'єднанням для отримання загального результату.

Метод *колективної генерації оцінок* ґрунтується на гіпотезі про те, що серед великої кількості ідей і суджень є хоча б декілька, які відповідають найімовірнішому варіанту розвитку системи. Реалізація методу ґрунтується на неформальному аналізі, на проведенні у відкритій формі дискусій щодо оцінок майбутнього стану системи, напрямків і засобів досягнення поставлених цілей, що дає змогу експертам впливати один на одного для уникнення помилок і вироблення спільного висновку. З метою розв'язання проблемної ситуації формується група експертів, яка спочатку генерує ідеї, а потім намагається зруйнувати (деструкгурувати) їх шляхом висунення контр ідей і вироблення узгодженої думки. Метод передбачає реалізацію таких етапів:

- формування експертної групи;
- складання проблемної записки;
- генерація ідей;
- деструктуризація ідей;
- оцінка критичних зауважень і складання списку ідей, які можуть мати практичну реалізацію;

- вироблення спільного висновку.

Метод *Дельфі* та його модифікації побудовані на реалізації принципів анонімності, регульованого зворотного зв'язку та узгодженості групової оцінки. Принцип анонімності полягає у тому, що кожний експерт формулює свій висновок з оцінкою у межах визначеної шкали, не спілкуючись з іншими членами експертної групи. Кожний експерт заповнює спеціально розроблені анкети, які містять Перелік запитань, а також інформацію про рівень узгодженості думок з відповідними доказами, і контактує дише з організатором експертизи.

Реалізація процедури методу Дельфі передбачає проведення декількох турів, прийму результати попереднього туру оголошують усім членам експертної групи (інформація міститься в анкетах, зміст яких від туру до туру змінюється). Принцип зворотнього зв'язку полягає у тому, що на підставі аналізу анкетної інформації експерт має змогу змінити попередній висновок без публічних пояснень. Зворотний зв'язок дає можливість кожному експерту ознайомитися з думками своїх анонімних колег і статистичний характеристиками групової відповіді, порівняти свої відповіді із загальними висновками, що забезпечує вироблення обґрунтованого розв'язання поставленої задачі. Вироблення групового висновку і оцінка його надійності потребує статистичного оброблення інформації. Для цього розраховують медіану і чверті статистичної сукупності, отриманої в результаті проведення туру, рангові коефіцієнти кореляції Спірмена і Кендалла, коефіцієнт конкордації тощо. Згідно з методом Дельфі після відповідної попередньої підготовки експертне опитування проводять в декілька турів (як правило, не більше від чотирьох) для того, щоб не обмежувати експертів жорсткими рамками відповіді, дати їм змогу вільно сформулювати відповіді стосовно досліджуваної проблеми, анкети першого турі певною мірою можуть бути безструктурними. Можливість неструктурованого подання висновків дає змогу ширше розглядати проблему, уточнювати незрозумілі моменти постановці задачі. Крім того, треба враховувати і той факт, що організатори експертизи не мають таких фундаментальних знань з досліджуваної проблематики, як члени експертної групи.

Прикладом модифікації методу Дельфі, що в деякій мірі позбавляє його недоліків, можна вважати метод *SEER* (Scenario Exploration, Elaboration and Review) – систему огляду та оцінювання подій. Згідно з методикою *SEER* передбачається проведення лише двох турів опитування, причому у кожному турі використовують різний склад експертів. Експертами першого туру є вчені з конкретної предметної галузі дослідження а експерти другого туру – це кваліфіковані спеціалісти-практики, що безпосередньо приймають рішення, а також спеціалісти з суміжних проблемних галузей. Відповіді кожного туру експерти переглядають, лише якщо вони виходять за межі наперед визначеного інтервалу, якому належить переважна більшість оцінок (наприклад, 80% усіх оцінок).

Деякі організації, окрім загальновизнаних стандартів в галузі якості ПЗ та забезпечення зрілості ПП, використовують власні керівництва. Вони розробляються шляхом аналізу експертами великої кількості результатів досліджень, які часом протирічать один одному. Написання рекомендацій щодо створення ПП залежить, зокрема, від цільового призначення, типу функціональних користувачів та середовища, в якому даний ПП буде використовуватися. З метою вирішення цієї проблеми було розроблено *метод когнітивного аналізу*. Для проведення такого аналізу необхідно мати детальний опис інтерфейсу користувача, завдання, дані щодо демографічного стану функціональних користувачів, дані щодо контексту використання ПП, дані щодо успішної послідовності дій, які мають виконуватися. Спочатку виконується аналіз ПП, потім, згідно з визначеними для завдання діями, оцінюється, наскільки користувач буде здатний вибрати потрібну дію, зможе визначити, що вибрана дія вірна і чи буде помітно прогрес у вирішенні завдання. Аналіз зрілості ПП ґрунтується на інформації про те, що має знати користувач перед виконанням завдання та що він має засвоїти під час його виконання.

Переваги та недоліки експертного оцінювання зрілості ПП. Евристична оцінка менш витратна порівняно з оцінкою на основі відгуків користувачів. Когнітивний аналіз ПП може проводитися з використанням лише текстового

опису інтерфейсу користувача і тому може здійснюватися ще на початкових етапах розробки ПП. Проте згадані методи оцінювання ПП не містять дії щодо вирішення проблем. Більше того, слід перевіряти експертні оцінки на узгодженість, оскільки досліджувані питання різняться між собою та мають різні рівні складності. Евристична оцінка ефективніша при виявленні можливих проблем, ніж огляд стандартів та когнітивний аналіз. За допомогою згаданих методів оцінки зрілості ПП можна виявити до 50% помилок.

Оцінювання на основі моделей. Даний вид оцінювання ґрунтується, як правило, на використанні програмних засобів, які реалізують розроблені моделі. Слід зазначити, що більшість з них дозволяють оцінити лише окремі властивості зрілості ПП. Серед відомих засобів виділяють наступні: MATURE, MESA Manufacturing Operation Management Maturity Assessment Tool, SESD.

1.3. Онтологія домену «зрілість програмного продукту»

В дисертації використовується означення онтології згідно зі стандартом IDEF5 [148]: онтологія, це словник термінів предметної області (ПрО) та їх формальних описів (аксіом), що обмежують зміст термінів в цьому словнику і забезпечують узгоджену інтерпретацію даних, які використовує словник. Формально онтологію ПрО можна задати як трійку:

$$O = (X, R, F),$$

де $X = \{x_i | \; x_i -$ поняття або терміни ПрО, $i = \overline{1,n}\}$, $R = \{r_1, r_2, \dots, r_m\}$, $R: x_1 \times x_2 \times \dots \times x_n$, $F: X \times R$ – скінченна множина функцій інтерпретації, заданих на концептах і/або відношеннях. Частковим випадком задання множини функцій інтерпретації F є глосарій, складений для множини понять X.

Перевагами застосування онтологій є:

– стандартизація термінології ПрО в єдину інформаційну основу для всіх учасників проекту;

– повторне використання при зміні вимог до проекту і продуктів розробки;

– управління розробкою шляхом поділу доступу до інформації різним учасникам проекту.

Процес побудови онтології, згідно з методологією IDEF5 складається з п'яти основних дій [4]:

1. *Вивчення та систематизування початкових умов.* Ця дія встановлює основні цілі і контексти проекту розробки онтології, а також розподіляє ролі між членами проекту.

2. *Збір і накопичення даних.* На цьому етапі відбувається збір і накопичення необхідних початкових даних для побудови онтології.

3. *Аналіз даних.* Ця стадія полягає в аналізі і угрупованні зібраних даних і призначена для полегшення побудови термінології.

4. *Початковий розвиток онтології.* На цьому етапі формується попередня онтологія, на основі відібраних даних.

5. *Уточнення і твердження онтології.* Заключна стадія процесу.

Онтологія може застосовуватися для вирішення наступних задач:

– узгодження процесів, документів, продуктів;

– проектування систем;

– моделювання та реінженерія бізнес-процесів;

– проектування концептуальної схеми.

Для дослідження домену були проаналізовані наступні джерела:

– Наукові дослідження вчених у періодичних виданнях [137-139, 141, 152-153, 157, 161-162, 164];

– Монографії [166-167, 170, 173-172];

– Нормативно-законодавча база (стандарти, закони тощо) [1-131];

– опитування доменних експертів.

В роботі наведена розроблена онтологія, що містить 31 термін з предметної області «зрілість ПП».

Процес розробки онтології складався з наступних етапів:

– визначення меж домену, цілей проекту і доменного контексту;

– збір даних та їх аналіз – дослідження джерел термінів, відбір термінів для онтології, встановлення відношень між ними, вербальний опис термінів;

– розробка онтології – створення схематичного (графічного) і аналітичного опису онтології;

– перевірка повноти і коректності онтології відповідно до вимог.

В даній роботі представлена розроблена онтологія за допомогою вербальних описів термінів, графічного представлення згідно зі схематичною мовою Schematic Language (SL) стандарту IDEF5 та аналітичним описом.

Схематична мова Schematic Language (SL) стандарту IDEF5 включає в себе кілька видів діаграм для побудови онтології. У рамкам проведеного дослідження була використана композиційна схема (Composition Schematics), яка є механізмом графічного представлення складу класів онтології і фактично являють собою інструменти онтологічного дослідження за принципом «Що з чого складається». Зокрема, композиційні схеми дозволяють наочно відображати склад об'єктів, що відносяться до того чи іншого класу [160].

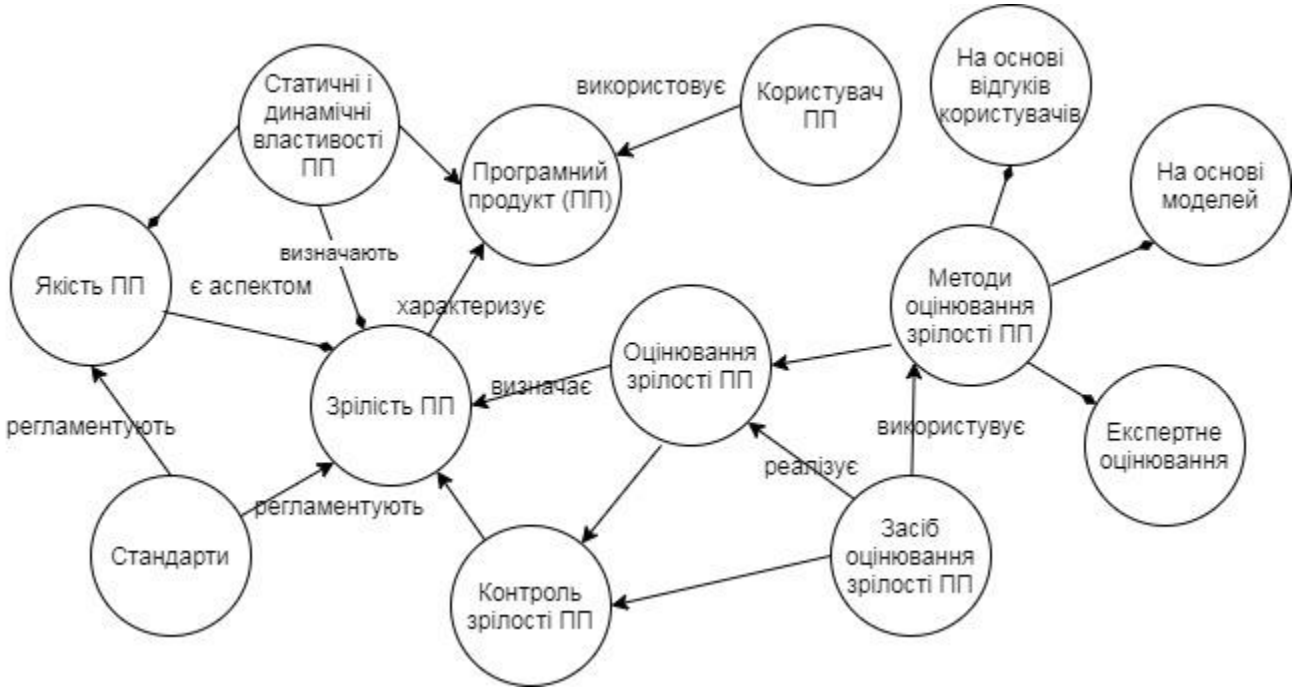

Рис. 1.12. Онтологічна модель домену «зрілість ПП»

Онтологія на рис. 1.12 побудована на основі понять, терміни яких складають словник досліджуваної предметної галузі, та зв’язків між ними. Відношення «part

of» застосовується в композиційній частині схеми та відображає склад об’єкту; «subkind of» – в класифікаційній частині схеми та вказує на різновид об’єкту.

Висновки по розділу 1

1. У дисертаційному дослідженні вперше зроблено спробу теоретично обґрунтувати поняття «зрілості ПП».

2. У роботі «зрілість ПП» (software product maturity) розглядається як ступінь, у якому ПП може бути використаний користувачами з метою досягнення визначених власних цілей з ефективністю\економічністю та\або задоволеністю у відповідному контексті застосування. Під терміном «користувач» (user) розуміється окрема особа або організація, яка використовує програмну систему для виконання визначених функцій.

3. Зрілість ПП, окрім класичних та загальноприйнятих аспектів, які, як правило розглядають зрілість у контексті процесів, що застосовуються при розробці ПП, розглядається також дослідження статичних та динамічних властивостей ПП та впливу на якість у використанні ПП.

4. Існуючі методи досягнення зрілості в основному містять кроки щодо вимірювання та оцінки зрілості процесів і не гарантують зрілість ПП вцілому.

5. Оскільки об'єктом дослідження є оцінювання зрілості ПП, то одним з найбільш проблемних місць є конкретизація предметної області з урахуванням її специфіки. Саме тому, перш за все, постає питання у представленні знань про предметну область. В результаті цього, актуальним є розробка моделі, яка б відображала об’єкти домену та зв’язки між ними. Таким засобом є онтології, які є формалізованим представленням знань про домен. У дисертації наводиться розроблена онтологія, яка має лише графічне представлення, що зумовлено обмеженістю обсягу дисертації.

РОЗДІЛ ІІ
МЕТОДИ ОЦІНЮВАННЯ ЗРІЛОСТІ ПРОГРАМНИХ ПРОДУКТІВ

У попередньому розділі була обґрунтована доцільність створення методів оцінювання зрілості ПП. Матеріали даного розділу присвячені опису методів, моделей та методиці оцінювання зрілості ПП. Розглядаються статистичні та оптимізаційні математичні методи, адаптовані для аналізу та розв'язування зазначених моделей та можливість програмної реалізації запропонованих методів.

2.1. Модель зрілості ПП

Для оцінювання рівнів зрілості ПП важливу роль відіграє модель, за допогою якою і буде здійснюватися оцінювання, а також встановлення елементів цієї моделі. У розділі представлено модель зрілості ПП (рис. 2.1), яка сформована з позицій 3-х ключових рівнів зрілості: генезису, росту та зрілості.

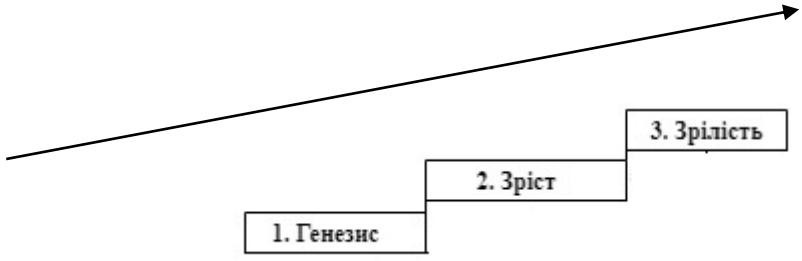

Рис. 2.1. 3-рівнева модель зрілості ПП

Перший рівень моделі характеризується оцінюванням рівня зрілості програмного коду. Оцінка вводиться за допомогою індексу зрілості програмного коду, яка обчислюється за допомогою значень обрахованих метрик коду.

Другий рівень моделі характеризується оцінювання рівня зрілості ПЗ. Під зрілістю ПЗ, як правило, розуміють якість ПЗ. Тому для оцінювання якості ПЗ необхідно дотримуватися відповідних характеристик, підхарактеристик, метрик та стандартів.

Рівень «Зрілість» характеризується двома напрямами: новими моделями організації ІТ-підприємств (екосистеми, технопарки, бізнес-інкубатори) та

оцінкою зрілості ІТ-підприємств. Цей етап пов'язаний з побудовою таких структур, як бізнес-інкубатори, технопарки та екосистеми, які забезпечуються наявним бізнес-потенціалом, який може бути головним рушієм у становленні стійкості. Таким чином, проектування та встановлення вищезазначених структур включає: оцінювання процесів ІТ-підприємств, перебудовою організаційної структури, підвищенням кваліфікації персоналу в цій галузі тощо. Цей крок включає: встановлення меж екосистем ПЗ, визначення експертів у цій конкретній галузі, вибрати характеристики та показники (продуктивність, різноманітність, гнучкість, продуктивність, стійкість (життєздатність), зрілість) для оцінювання, аналіз результатів та вибрати тип моделі для ІТ-підприємства, сформувати рекомендації. Для оцінювання зрілості ІТ-підприємств на сьогодні існує значна кількість розробок моделей зрілості управління, найпопулярнішими з яких є такі: модель зрілості Керцнера; модель зрілості Берклі; модель зрілості ОРМ3 (Organizational Project Management Maturity Model); модель CMMI (Capability Maturity Model Integration). Вибір моделі для оцінювання залежить від проектів та процесів ІТ-підприємств. Ці моделі мають багато спільних рис. Основним принципом їх побудови є перехід від нижчого рівня зрілості (загальне усвідомлення проектного управління) до вищого, що передбачає безперервне вдосконалення процесу управління.

2.2. Метод оцінювання зрілості ПП з відкритим кодом

Програмне забезпечення з відкритим кодом (Open Source Software, OSS) – це різновид ПЗ з доступним вихідним кодом, до якої будь-яка особа може отримати доступ, використовувати та копіювати за умови дотримання відповідних положень ліцензії. Більш вичерпне та офіційне визначення ПЗ з відкритим кодом можна знайти на веб-сайті [159]. Вибір відповідного ПЗ з відкритим кодом для оцінювання його зрілості або множини вимог може бути дуже складним завданням. Деякі труднощі пов’язані з тим, що не існує загальновизнаного набору критеріїв, які б використовувались при оцінюванні, і що зазвичай існує багато проектів ПЗ з відкритим кодом для вирішення певної

проблеми. У даному дослідженні пропонується набір критеріїв та методику для оцінки ПЗ з відкритим кодом на відповідність зрілості з використанням як функціональних, так і нефункціональних факторів. Ці критерії використовуються для вдосконаленого розв’язання завдання прийняття рішення за допомогою добре відомого методу аналітичної ієрархії.

Сьогодні доступні принаймні чотири моделі для оцінювання ПЗ з відкритим кодом.

1. Open Source Maturity Model (OSMM) був створений в 2003 році консалтинговою компанією Capgemini. Вона використовує 12 зважених критеріїв, які можна для оцінювання звести до табличного вигляду [167].

2. Open Source Maturity Model ($OSMM_2$), була створена Navica в 2004 році.$OSMM_2$ прагне оцінити зрілість 6 ключових елементів ПЗ та використовує таблиці для допомоги в оцінці [156].

3. Qualification and Selection of Open Source software (QSOS), була створена в 2004 році та спонсорована консалтинговою компанією Atos Origin. QSOS також використовує набір критеріїв оцінки та надає веб-інструменти для допомоги у процесі оцінювання. Веб-сайт також містить БД попередніх оцінок [154].

4. Business Readiness Rating (BRR) була створена в 2005 році за підтримки Карнегі Меллона Веста, SpikeSource, O'Reilly та Intel. BRR також надає критерії оцінки та електронні таблиці для оцінки [141].

Проаналізувавши вищезгадані моделі можна зробити висновок про те, що всі вони мають спільну структуру. На рис. 2.2 наведено модель взаємозв'язку OSS. Крім цього для дослідження зрілості кожна з них має свої недоліки. А також, лише одна з цих моделей включала оцінювання фактичного критерію вихідного коду. Оскільки оцінка якості коду є важливою частиною визначення загальної придатності ПЗ, тобто зрілості, необхідно до критерії також включати критерій якості вихідного коду. В результаті було виокремлено 9 критеріїв, які в свою чергу поділяються на функціональні (2 критерія) та нефункціональні (7 критеріїв).

У таблиці 2.1 перелічено ці 9 основних критеріїв, які використовуються у дослідженні з їх коротною характеристикою.

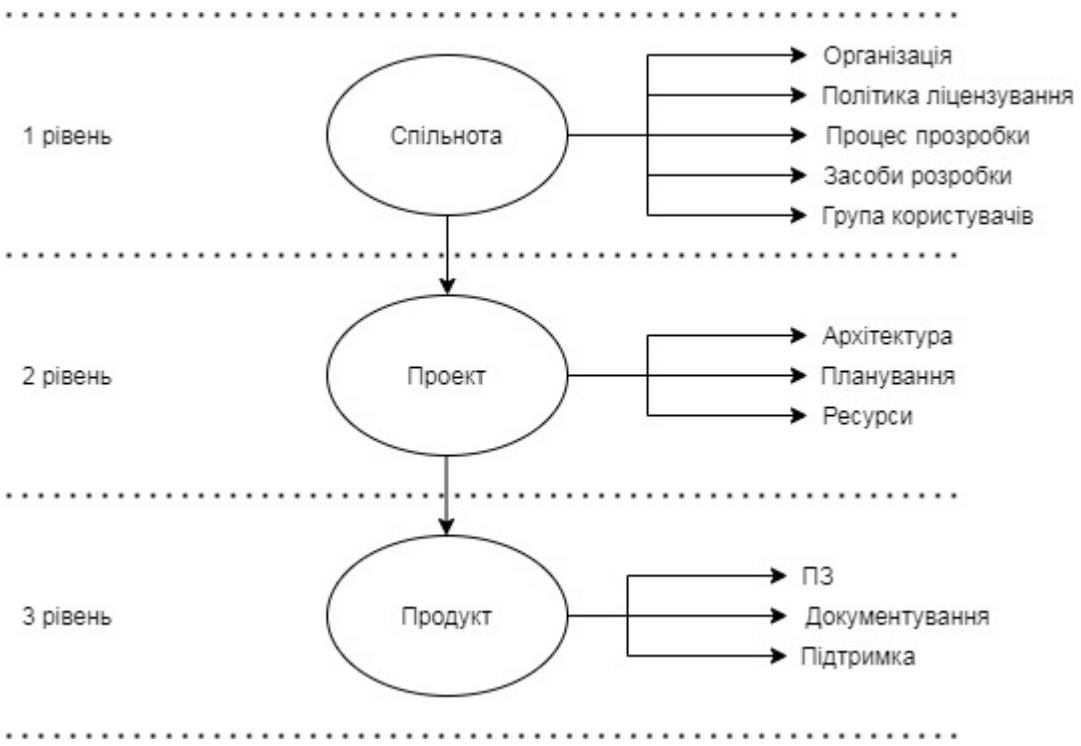

Рис. 2.2. Модель взаємозв'язку OSS

Таблиця 2.1

Критерії та їх короткий зміст критеріїв оцінювання ПЗ з відкритим кодом

№	Критерій	Опис
Функціональні		
1	Функціональність (k_1)	Критерій функціональності k_1, використовується, щоб вказати, чи має програмне забезпечення з відкритим кодом необхідні функції. Функціональність часто використовується в більшості традиційних моделей оцінки ПЗ і не є специфічною для оцінки ПЗ з відкритим кодом. Використовуючи один і той же стандарт для оцінки кількох продуктів, можна провести деякі об’єктивні порівняння.
2	Еволюція ПП (k_2)	Критерій еволюції проекту, k_2, вказує, чи існують

		або чи розроблені чіткі плани щодо функцій, які будуть додані або вдосконалені. Він також може бути використаний для оцінювання «здоров'я» проекту, оцінювання його зростання чи занепаду. «Здоровий» проект зазвичай має регулярні релізи ПЗ з доданою або вдосконаленою функціональністю та досить постійним темпом розвитку. Існує кілька джерел для збору даних про випуски проекту з відкритим кодом: сторінка завантаження на веб-сайті проекту, журнал змін або сховище вихідного коду.
Нефункціональні		
3	Ліцензування k_3	Використовується для позначення відповідності ліцензії на програмне забезпечення для передбачуваного використання
4	Тривалість (k_4)	Критерій тривалості k_4 використовується як показник якості лінії проекту.
5	Спільнота(k_5)	Критерій спільноти, k_5, розглядає спільноту розробників, тестувальників, користувачів ПЗ як непрямий показник загального стану «здоров'я» програмного проекту. Використовується як показник якості (зрілості) спільноти, яка бере участь у проекті з відкритим кодом.
6	Інтеграція (k_6)	Критерій k_6 використовується для оцінювання інтеграції ПЗ з відкритим кодом на ринку. Об'єктивні показники, такі як показники продажів або встановлена клієнтська база, які часто використовуються в інших галузях, недоступні для ПЗ з відкритим кодом, оскільки користувачі можуть

		просто анонімно завантажувати ПЗ. Однак існує кілька джерел, за допомогою яких можна побічно виміряти проникнення ПЗ з відкритим кодом: Ohloh (http://www.ohloh.net), Google Trends (http://trends.google.com) тощо.
7	Документування (k_7)	Критерій k_7 використовується для оцінювання якості документації
8	Підтримка (k_8)	Критерій k_8 використовується для оцінювання якості підтримки, як комерційної, так і спільноти, яка підтримує ПЗ.
9	Якість коду (k_9)	Критерій k_9 використовується для оцінювання якості вихідного коду.

Оцінювання відбувається за наступною методикою, як описано нижче:

– оцінювання ПП за кожним критерієм;

– визначення методу зважування до попередніх елементів на основі вимог,

– розрахунок загальної оцінки зрілості ПП,

– оцінювання кожного елемента та оцінювання їх зрілості за шкалою від 1 до 10 (табл..2.2).

Таблиц 2.2.

Елемент	Критерій	Вага
ПП	$k_1, k_2, k_3, k_4, k_5, k_9$	50
Підтримка	k_8	10
Документування	k_7	10
Навчання		10
Інтеграція ПП	k_6	10
Професійні послуги		10
Всього		100

2.3. Метод оцінки рівнів зрілості екосистем програмного забезпечення на базі моделі CMMI

Сьогодні розробка IT продуктів відбувається не лише класичними компаніями, а й з’являються нові форми комьюніті. Однією з таких форм є екосистеми ПЗ. Екосистеми ПЗ стали досить популярними за останні два десятиліття. Поняття екосистеми ПЗ широко використовується сьогодні, але на сьогоднішній день не існує офіційного визначення цього поняття. Існуючі доменні дослідження представляють лише словесне та описове поняття. Аналіз існуючих джерел показав, що концепцію програмних екосистем можна розглядати з різних точок зору, що породжує різні класифікації та характеристики екосистем. У роботі використовується означення екосистеми ПЗ, наведене в [160]: «…екосистема ПЗ є сукупність підприємств, що функціонують як цілісні об'єкти та взаємодіють із загальним ринком ПЗ та послуг, а також взаємозв'язками між ними. Ці відносини часто підтримуються загальною технологічною платформою або ринком і діють завдяки обміну інформацією, ресурсами та артефактами».

Для дослідження властивостей екосистем ПЗ, в тому числі і зрілості екосистем ПЗ, необхідні специфічні моделі та методи [140], що використовуються для дослідження екосистем, а також рівнів зрілості та їх якісної оцінки.

У роботі [164] була введена модель зрілості для стартапових екосистем ПЗ, заснована на систематичних якісних дослідженнях навколо багаторазового тематичного дослідження трьох екосистем. Дослідження проводилось протягом 4 років і включало широкий спектр механізмів збору даних, таких як огляди літератури, інтерв'ю експертів та спостереження в трьох відповідних екосистемах (Тель-Авів, Сан-Паулу та Нью-Йорк); всі зібрані дані були проаналізовані за допомогою методів, заснованих на обгрунтованій теорії (Grounded Theory), що призвело до концептуальної основи запуску екосистем ПЗ. Також була розроблена модель зрілості стартапових екосистем, яка допомагає зрозуміти їх еволюцію та динаміку.

У [153] представлена модель зрілості моделі управління екосистемою управління (SEG-M2), яка розроблена на основі принципів моделі зрілості зони

фокусування. SEG-M2 був розроблений для організацій, що розробляють програмне забезпечення, для оцінки своїх практик управління екосистемами, встановлення мети вдосконалення та виконання плану вдосконалення.

У [154] обговорювались проблеми, пов'язані з гнучкими програмними екосистемами. Для кожного виклику були визначені такі властивості: зрілість екосистеми, фаза в рухливому життєвому циклі, інші фактори впливу та причини відповідного виклику.

Результати цих досліджень стосуються кількісного аналізу зрілості екосистеми ПЗ та її рівнів. На жаль, якісного аналізу відповідних характеристик не існує. Таким чином, актуальність цього дослідження породжується необхідністю розробки методу якісної оцінки зрілості програмних екосистем.

Для оцінювання екосистем ПЗ пропонується використовувати модель і метод оцінки зрілості екосистем ПЗ на основі CMMI-SW.

CMMI-SW пропонує критерії, що дозволяють оцінити зрілість організацій-розробників. Ці критерії можуть використовуватися організаціями-розробниками для поліпшення процесів розробки і супроводження ПЗ, а також державними і комерційними організаціями-замовниками для оцінки ризиків укладення договорів на розробку програмних проектів з визначеними організаціями-виконавцями.

На базі CMM SEI розробив 2 методи оцінювання зрілості процесу:

– Метод SPA (Software Process Assessment) – оцінювання поточного стану процесу. Використовується для дослідження процесу програмної інженерії в організації, визначення його поточного стану, виявлення існуючих проблем, вибору високопріоритетних цілей для поліпшення процесу розробки, вироблення відповідної стратегії поліпшення і отримання підтримки з боку керівництва [21];

– Метод SCE (Software Capability Evaluation) – оцінка спроможностей бюджету організації-розробника ПЗ. Може використовуватися для ідентифікації ризиків замовника, пов’язанного з певним проектом чи контрактом та організацією-виконавцем на розробку високоякісного ПЗ відповідно до встановлених термінів і бюджету. Може використовуватися при визначенні

потенційних організацій-виконавців програмних проектів або для управління ефективністю процесу в організаціях-виконавцях, які мають певні ресурси розробки [190].

Методи SPA і SCE відрізняються мотивацією, цілями, структурою результуючих даних і способами інтерпретації результатів. А це, в свою чергу, визначає застосовувані процедури оцінювання, умови проведення дослідження, динаміку інтерв'ювання, спектр запитань, характер і обсяг інформації, що збирається, а також принципи підготовки фахівців для груп оцінювання.

Дослідження методом SPA з метою поліпшення процесу в організації виконується регулярно (з періодичністю 18 - 36 місяців) в умовах відкритості та співпраці з керівництвом і колективом розробників.

Оцінювання методом SCE виконується в умовах, наближених до умов проведення ревізій.

Рекомендації експертів допомагають вибрати найбільш надійних виконавців проектів.

Основні кроки виконання оцінок по CMM методами SPA і SCE.

Крок 1. Вибір групи експертів, навчених основам CMM і специфіки методів оцінювання поточного стану або потенційних можливостей організації. Члени групи повинні бути професіоналами в програмної інженерії та менеджменті. Для оцінювання рівня зрілості за допомогою запропонованої моделі передбачається виконання наступних кроків:

1. Вибір групи експертів, що є професіоналами в програмній інженерії та менеджменті.

2. Підготовка контрольного опитувального листа, що буде використовуватися під час оцінювання.

3. Опитування IT-спеціалістів, огляд документації та зіставлення результатів з результатами аналізу за опитувальним листом.

4. Підготовка звіту.

Результатом оцінювання є визначення рівня зрілості і підготовка розроблення рекомендацій щодо покращення рівня зрілості.

Обробка опитувальних листів для отримання оцінки зрілості передбачає виконання таких дій:

1. Кожній оцінці ставиться у відповідність числовий коефіцієнт (табл. 2.1)

Таблиця 2.1

Оцінка	Числовий коефіцієнт
Майже завжди	1
Часто	0,75
Іноді	0,5
Рідко	0,25
Ніколи	0

2. Обробка опитувальних листів: підраховується кількість відповідей за кожною оцінкою одного напряму (кількість «+» у стовпчиках). Ця кількість відповідей множиться на відповідний коефіцієнт і обчислюється їх сума. Потім ця сума ділиться на кількість питань, що стосуються даного напряму, і множиться на 100% (для отримання оцінки досяжності цілей напряму і відсотках). Приклад опитувального листа наведений у табл. 2.2.

Табл. 2.2.

Приклад опитувального листа

	Майже завжди	Часто	Іноді	Рідко	Ніколи
Питання 1	+				
Питання 2			+		
…					
Питання n				+	

3. В кінці оброблення всіх опитувальних листів оцінки за кожним напрямом усереднюються.

Усереднена оцінка обчислюється як середнє арифметичне часткових оцінок.

Отримані сумарні оцінки у відсотках за кожним напрямом записуються у підсумковий звіт (табл. 2.3). Для обчислення рівня зрілості екосистеми застосовується формула

$$L_{maturity} = \frac{2}{n}\sum_{i=1}^{n}\frac{SUM_i}{100}, \quad (2.1)$$

де SUM_i – отримані сумарні оцінки у відсотках, i – напрям оцінювання, n – кількість напрямів оцінювання.

Табл. 2.3

Оцінка рівня зрілості

Напрям оцінювання	Майже завжди >90% випадків	Часто 60-90% випадків	Майже порівну 40-60%	Інколи 10-40% випадків	Рідко <10% випадків
Управління вимогами					
Планування проекту					
Моніторинг проекту					
…					
Забезпечення гарантії якості					

У роботі пропонується використовувати теорію нечітких множин для побудови інструменту оцінки зрілості екосистем.

У моделі зрілості CMMI ключові практики групуються у спеціальні цілі, далі цілі групуються в області процесу CMMI. Таким чином, для визначення рівня зрілості доцільно використовувати ієрархічні системи нечіткого логічного висновку. Тоді зрілість екосистеми буде визначатися співвідношенням (2.2):

$$M = f\big(f_1(P_1, P_2, \dots, P_7), f_2(P_8, P_9, \dots, P_{18})\big), \quad (2.2)$$

де M – зрілість компанії, $f_1(), f_2()$ – функції виведення зрілості для 2 і 3 рівнів, $P_1, P_2, \dots, P_{18}$ – засвоєння процесних областей у моделі CMMI.

У свою чергу, засвоєння кожної процесної області буде визначатися співвідношенням (2.3):

$$P_i = h_i(g_{i1}(p_{i11}, \dots, p_{i1i}), \dots, g_{in}(p_{inm})), \quad (2.3)$$

де $h_i()$ – функція нечіткого виведення освоєності процесної області з індексом i, $g_{ij}()$ – функція нечіткого виведення досягнення IT-компаній j-ої спеціальної мети для i-ої процесної області, p_{ijk} – освоєння IT-компанією k-ої спеціальної практики моделі CMMI, що входить в j-у спеціальну мету для i-ої процесної області.

Визначимо оцінки NI, PI, LI, FI як нечіткі числа, задані нечіткими множинами N, P, L, F на множині дійсних чисел R. Носіями множин N, P, L, F визначимо інтервали на множині R – $[n_L, n_R], [p_L, p_R], [l_L, l_R], [f_L, f_R]$ відповідно. Для кожного числа NI, PI, LI, FI задамо на R функції належності $\mu_i(x)$ такі, що $\forall x \in R | \mu_i(x) \in [0,1]$, де $i \in \{N, P, L, F\}$.

$$\mu_i(x) = \begin{cases} left\left(\dfrac{x - a_1}{a_2 - a_1}\right), & \text{де } a_1 \le x < a_2 \text{ – функція лівого краю } \mu_{i\uparrow}(x), \\ 1, & a_2 \le x \le a_3 \\ right\left(\dfrac{a_4 - x}{a_4 - a_3}\right), & a_1 < x \le a_2 \text{ – функція правого краю } \mu_{i\downarrow}(x). \\ 0, & x > a_4 \end{cases} \quad (2.4)$$

Для таких нечітких чисел визначено поняття метрики між двома числами:

$$d^2(a, b) = \int_0^1 (A_L(a) - B_L(a))^2 da + \int_0^1 (A_U(a) - B_U(a))^2 da, \quad (2.5)$$

де нечітке число задано через поняття $\alpha -$ перетину такого, що $\forall a \in [0,1]\ \exists\ a_1 \le x \le a_4$ і $\mu_a(x) \ge a$ і $A_L(a) = \mu_{a\uparrow}^{inv}(x)$ – обернена функція $\mu_a(x)$ на інтервалі її зростання, а $A_U(a) = \mu_{a\downarrow}^{inv}(x)$ – обернена функція $\mu_a(x)$ на інтервалі її спадання.

Для визначення досяжності IT-компаній спеціальної мети і областей процесу CMMI розглянемо функцію $f(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$, де операція $\oplus$ буде операцією додавання нчітких чисел згідно з формулою (3.6):

$$a \oplus b = \int_{a_1 \otimes b_1}^{a_2 \otimes b_2} \frac{\min(\mu_a(x), \mu_b(y))}{x \otimes y}, \quad (2.6)$$

Де операція $\otimes$ – операція $+, -, \times, /$. Досягнення спеціальної мети G_i освоєння процесів оцінюваної IT-компанії опишемо у вигляді лінгвістичної характеристики: повністю досягнута (FR), частково досягнута (PR), не досягнута (NR).

У загальному випадку будемо вважати, що спеціальна мета G_i досягнута (FR), якщо сума відповідей експертів на питання про ступінь впровадження в компанії ключових практик, визначених моделлю зрілості CMMI, вираженого у вигляді нечіткого числа, рівного $f(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$, наближаються до нечіткого числа $FI * n = FI_1 \oplus \dots \oplus FI_n$.

Спеціальна мета G_i досягнута частково (PR), якщо $f(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$ наближається до нечіткого числа $PI * n = \frac{LI*n \oplus PI*n}{2}$.

Спеціальна мета G_i не досягнута (NR), якщо $f(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$, наближається до нечіткого числа $NI * n = NI_1 \oplus \dots \oplus NI_n$.

Освоєння області процесу виражається у вигляді суми оцінок G_i виду NR, PR, FR, отриманих на попередньому кроці для спеціальних цілей, що входять до конкретної області процесу CMMI. Освоєння областей процесу позначимо лінгвістичними змінними NF, FF, PF, що буде відповідати поняттям «область не освоєна», «освоєна частково», «повністю освоєна». Вони будуть обчислюватися в загальному вигляді за такими правилами:

процесна область $F_i = PF$, якщо $\sum_{i=1}^{n} G_i$ наближається до $\sum_{1}^{n} NR$;

процесна область $F_i = PF$, якщо наближається до $\sum_{1}^{n} PR$;

процесна область $F_i = PF$, якщо наближається до $\sum_{1}^{n} FR$;

Далі необхідно визначити освоєння IT-компанією рівнів зрілості. Для цього необхідно визначити функції $f(), f_1(), f_2()$ з рівності (4). Але якщо для функцій $f_1()$ і $f_2()$ можна застосовувати аналогічні міркування, виразивши їх результат через суму освоєних процесних областей, то для функції $f()$ такі міркування вже не підійдуть. У моделі CMMI вважається, що компанія досягає 3-го «повторюваного» рівня (R), якщо вона досягла 2-го «встановленого» рівня (D) і в

більшості освоїла процесні області, специфічні для 3-го рівня зрілості. Якщо значення функцій $f_1()$ і $f_2()$ визначаються терм-множиною $\{NM,\ PM,\ FM\} = \{\text{"генезис"}, \text{"розвиток"}, \text{"зрілість"}\}$, тоді можна сформулювати такий набір нечітких правил:

Якщо $f_1(P_1, \dots, P_7) = NM$, тоді $M = I$.

Якщо $f_1(P_1, \dots, P_7) = PM$, тоді $M = D$.

Якщо $f_1(P_1, \dots, P_7) = FM$, тоді $M = D$.

Якщо $f_1(P_1, \dots, P_7) = PM$ і $f_2(P_8, \dots, P_{18}) = NM$, тоді $M = D$.

Якщо $f_1(P_1, \dots, P_7) = PM$ і $f_2(P_8, \dots, P_{18}) = PM$, тоді $M = R$.

Якщо $f_1(P_1, \dots, P_7) = FM$ і $f_2(P_8, \dots, P_{18}) = PM$, тоді $M = R$.

Якщо $f_1(P_1, \dots, P_7) = FM$ і $f_2(P_8, \dots, P_{18}) = FM$, тоді $M = R$.

Отримані правила дозволяють побудувати програмний засіб для оцінювання зрілості екосистем.

Стадії зрілості екосистем програмного забезпечення пов'язані з оцінкою потенціалу регіонів для утворення екосистем. Зростання чисельності учасників екосистем програмного забезпечення і видів їх діяльності впливає на привабливість екосистем програмного забезпечення регіону і забезпечує зростання його потенціалу: поява і кооперація нових учасників приведуть до створення нових компаній, розробок, інноваційних бізнес-моделей і ін., Що створить базу для подальшого розвитку території.

В даний час не існує інструментів, що дозволяють оцінити зрілість екосистем програмного забезпечення з урахуванням потенціалу для здійснення інноваційної діяльності в регіоні. Використання існуючих методик оцінки розвитку екосистем програмного забезпечення проблематично через труднощі збору даних. До того ж вони орієнтовані тільки на аналіз числа зв'язків між учасниками, виявлення учасників-концентраторів (тобто мають найбільшу кількість зв'язків) і ін., При цьому не зачіпають проблем оцінки потенціалу.

Для стійкої роботи екосистем програмного забезпечення необхідно існування різноманітних видів діяльності. Зміна числа видів і їх складу визначає розвиток екосистеми і впливає на зміну стадій зрілості. Таким чином,

різноманітність є запорукою існування і розвитку екосистем програмного забезпечення.

Різноманітність екосистем програмного забезпечення характеризується наявністю рідкісних (мале число компаній) і масових (велике число компаній) видів діяльності. Якщо рідкісних видів багато, то з них, за певних умов, можуть розвинутися масові види діяльності. Отже, рідкісні види формують потенціал екосистем програмного забезпечення.

Таким чином, аналіз зрілості екосистем програмного забезпечення необхідно здійснювати з урахуванням зміни різноманітності видів діяльності, тому що вона впливає на потенціал екосистем програмного забезпечення регіону.

Кожна екосистема програмного забезпечення має свій унікальний видовий склад, тому різноманітність не може бути проаналізовано з використанням даних тільки про кількість видів і чисельності компаній в видах. Показником, що дозволяє порівнювати екосистеми програмного забезпечення різних регіонів по співвідношенню рідкісних і масових видів, є γ, який визначається на основі формули (2.7):

$$\Omega(x) = W_0 * x^{-\gamma}, \qquad (2.7)$$

де $\Omega(x)$ – кількість видів діяльності компаній розміру (численності) x,

W_i – кількість видів діяльності компаній, представлених i-им числом компаній,

$W_0 = \frac{V}{\sum_x x^{-\gamma}}$ – нормуючий множник,

V - загальна кількість видів,

$x \in [1, \infty]$ – число компаній виду і, $i = [x]$ (кількість компаній, що представлені визначеним числом видів),

γ – показник, що характеризує відношення рідкісних і масових видів, що визначається емпіричним способом.

В дисертаційному дослідженні пропонується наступна послідовність розрахунку показника γ:

1. Об’єднання видів діяльності комнаній, представлених рівною кількістю в групи.

2. Розташування отриманих груп в порядку зростання в них кількості (числа) видів діяльності.

3. Розрахунок показника γ.

Графічно розподіл компаній по рідкісним і масовим видам відображається гіперболою (рис. 2.2). На графіку по осі X буде відображатися кількість компаній певного виду (починаючи з найменшої кількості компаній виду), а по осі Y – кількість видів, представлених зростаючою кількістю компаній. Крива на графіку показує сукупність точок, кожна з яких відповідає видам екосистеми, які представлені зростаючою кількістю компаній. При зміні чисельності рідкісних і масових видів змінюється нахил гіперболи: чим більше «прогин» (крутизна), тим вище «розрив» в рідкісних і масових видах. Зміна співвідношення відбувається внаслідок корекції числа видів (їх зростання). На прикладі даних [165] візуально переконуємось, що залежність між двома факторами є гіперболою.

Оскільки екосистема программного забезпечення є динамічною системою, то поєднання рідкісних і масових видів діяльності постійно змінюється, отже, змінюється значення γ.

Рис. 2.2. Графічна інтерпретація показника γ

З огляду на те, що γ характеризує різноманітність, можна зробити висновок, що динаміка γ відображає динаміку різноманітності. Раніше відзначалося, що для кожної стадії розвитку екосистеми програмного забезпечення характерні різні

тенденції зміни видів і їх чисельності. Це дозволяє зробити висновок, що різноманітність визначає зрілість, а виявлені тенденції можуть бути трактовані як характеристики різних стадій зрілості екосистеми програмного забезпечення. Таким чином, позитивну динаміку γ відповідає стадія «генезис або розвиток», уповільнення росту – «зрілість», зниження – «занепад», а відсутності динаміки – «стагнація».

Крім показника γ, різноманітність також характеризується введенням автором коефіцієнтом різноманітності, який відображає «насиченість» видами діяльності компаній в екосистемі програмного забезпечення. Цей показник дозволяє виявляти потенціал регіонів: в екосистемах програмного забезпечення, що знаходяться на більш ранніх стадіях розвитку, різноманітність тільки наростає, отже, виникають рідкісні види і потенціал теж зростає. У таких регіонах коефіцієнт має більш високе значення. Коефіцієнт розраховується за формулою (2.8):

$$K_{\text{різноманітність}} = \frac{\text{число видів діяльності компаній регіону}}{\text{число компаній регіону}}, \qquad (2.8)$$

Розрахунки показали, що з часом коефіцієнт знижується, тому що екосистема програмного забезпечення стає зрілою і кількість видів діяльності не збільшується настільки інтенсивно, як на початкових стадіях.

У сукупності з іншими показниками, що впливають на зростання або скорочення різноманітності (обсяг венчурного фінансування, кількість компаній) за допомогою даного показника можна розрахувати інтегральну оцінку регіону на певну дату і порівняти з іншими територіями. Після ранжирування регіонів за зменшенням отриманої оцінки, робиться висновок про перспективи тих чи інших територій.

Для узагальненої характеристики регіону пропонується визначати інтегральний показник розвитку за формулою (2.9):

$$I = K_1 \times P_1 + K_2 \times P_2 + K_3 \times P_3, \qquad (2.9)$$

де I – інтегральний показник розвитку регіону,

K_1 – вага критерію «обсяг венчурного фінансування»;

K_2 – вага критерію «компаній, профінансованих венчурним капіталом»;

K_3 – вага критерію «коефіцієнт різноманітності»;

P_1 – рейтинг регіону в загальному обсязі фінансування;

P_2 – рейтинг регіону в сукупній кількості компаній;

P_3 – рейтинг регіону в сумарному коефіцієнті різноманітності.

Розрахунок інтегрального показника рекомендується проводити за зазначеними критеріями, з урахуванням ваг, присвоєних експертами.

Методика розрахунку тривалості прогнозованого періоду, необхідного для досягнення регіоном рівня розвитку екосистеми програмного забезпечення

Оцінка часових параметрів, необхідних для досягнення регіоном цільового рівня розвитку екосистеми програмного забезпечення, необхідна для виявлення періоду, після закінчення якого рівень різноманітності регіону наблизиться до цільового, який визначається з урахуванням особливостей і завдань розвитку регіону. Вибір даного критерію пояснюється тим, що різноманітність визначає динаміку розвитку екосистеми програмного забезпечення, а також має взаємозв'язок з потенціалом регіону.

Для розрахунку шуканого періоду пропонується наступна методика:

1. Визначення параметрів лінійного тренду $x(t) = a + b * t$, де t – період часу, необхідний для досягнення рівнем екосистеми програмного забезпечення показників передового регіону, взятого для порівняння, а $x(t)$ – значення коефіцієнта різноманітності.

2. Розрахунок прогнозного періоду (t) визначається виходячи з рівняння, з урахуванням зміни між рівнем цільового значення коефіцієнта різноманітності $x'(t)$ і фактичного $x(t)$ за формулою:

$\Delta t = \Delta x / b$, де b – параметр, що характеризує зміну коефіцієнта різноманітності за одиницю прийнятого тимчасового періоду.

2.4. Метод оцінювання зрілості на основі ієрархічної моделі

У межах дисертаційного дослідження зрілість ПП може розглядатися як відповідність певній ієрархічній моделі [183]. ПП декомпозується на атрибути,

кожен з яких – на відповідні показники, а ті в свою чергу – на метрики. Така декомпозиція ґрунтується на відповідних стандартах, керівництвах, експертних судженнях тощо. Будемо розглядати поняття «зрілість ПП» як сукупність складових, поступово деталізуючи її доти, доки не буде досягнуто рівня абстракції, прийнятного для формування кількісних значень $m_{ij1}, \ldots, m_{ijk}$ (k – кількість метрик, що відповідають j-му показнику i-го атрибуту).

Побудуємо формальну модель оцінювання зрілості ПП, спираючись на зведення значень окремих показників атрибутів в інтегральну оцінку Q [184]. Представимо зрілість ПП як 4-рівневу ієрархічну структуру. Кожен критерій вищого рівня ієрархії містить критерії нижчого рівня. Допускається введення додаткових критеріїв на кожному з рівнів (рис. 2.3).

Критерії рівнів ієрархії, окрім першого рівня, характеризується двома числовими параметрами – кількісним значенням та ваговим коефіцієнтом. Числові значення показників зрілості ПП розраховуються на основі відповідних метрик, значення яких обчислюється як середнє арифметичне оцінок користувачів.

Для вирішення задачі отримання оцінювання зрілості ПП, яка є основою для побудови моделі забезпечення зрілості ПП, найчастіше використовують наступні методи: метод зваженої суми; адитивна згортка; мультиплікативна згортка [184].

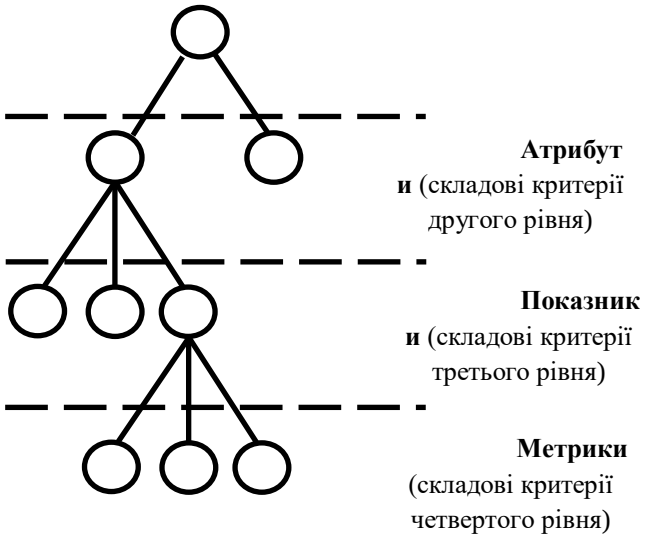

Рис. 2.3 Графічне зображення ієрархії складових критеріїв зрілості ПП

Розглядатимемо багатокритеріальні моделі, тобто такі, у яких кожна альтернатива (варіант) оцінюється множиною критеріїв. Найбільш відомими багатокритеріальними моделями є багатомірні функції корисності, моделі багатомірного шкалювання, модель (метод) аналізу ієрархій (Сааті). Найчастіше використовують адитивні та мультиплікативні функції корисності. Адитивна функція малочутлива до зміни показників з малою вагою; мультиплікативна, навпаки, сильно залежить від зміни показників з малими значеннями оцінок корисності [168]. Пропонується для вирішення задачі отримання інтегральної оцінки зрілості ПП визначати як адитивну функцію корисності та застосувати адитивну згортку відповідно. Узагальнена формула адитивної функції корисності має вигляд:

$$Q(x) = \sum_{i=1}^{n} p_i \cdot \hat{Q}_i(x),$$

де Q – функція корисності варіанта x; p_i – вага критерія i; $\hat{Q}_i(x)$ – оцінка корисності варіанта x за критерієм i.

Для аналітичної оцінки всієї ієрархічної структури зрілість ПП застосуємо метод вкладених скалярних згорток [168]. Тоді модель оцінки зрілість ПП можна описати функцією адитивної скалярної згортки:

$$\begin{aligned} Q(x) &= \sum_{i=1}^{n} p_i a_i(x) = \sum_{i=1}^{n} p_i \sum_{j=1}^{m} q_j a_{ij}(x) = \\ &= \sum_{i=1}^{n} p_i \sum_{j=1}^{m} q_j \sum_{k=1}^{l} r_k a_{ijk}(x), \end{aligned} \qquad (2.10)$$

де $Q(x)$ – загальний критерій для оцінок користувачів $x \in X$; $\{a_i(x)\}_1^n$, $\{a_{ij}(x)\}_1^m$, $\{a_{ijk}(x)\}_1^l$ – набори складових критеріїв відповідних рівнів ієрархії ; n, m, l – кількість критеріїв на рівнях; p_i, q_j, r_k - вага складових критеріїв a_i, a_{ij}, a_{ijk}. Для важливості (ваги) виконується умова нормування:

$$\sum_{i=1}^{n} p_i = \sum_{j=1}^{m} q_j = \sum_{k=1}^{l} r_k = 1.$$

Для критеріїв зрілості ПП на всіх рівнях використовується єдина шкала оцінки від 0 до 1. Виходячи з вищенаведеного, можемо оцінити складовий критерій третього рівня:

$$a_{ij} = \sum_{k=1}^{l} r_k a_{ijk},$$

$$\left.\begin{array}{l} 0 < a_{ijk} \le 1, \\ 0 < r_k < 1 \wedge \sum\limits_{k=1}^{l} r_k = 1 \end{array}\right| \Rightarrow 0 < r_k a_{ijk} < 1 \Rightarrow$$

$$\Rightarrow \left(\exists\, r_k, a_{ijk}, k = \overline{1,l}, k, l \in N\right)\left(0 < a_{ij} < 1\right) \wedge$$

$$\wedge \left(\exists\, r_k, a_{ijk}, k = \overline{1,l}, k, l \in N\right)\left(a_{ij} \ge 1\right).$$

Аналогічно a_i та підсумкова оцінка Q також можуть приймати значення більші 1 або в межах від 0 до 1. Достатність рівня зрілості ПП визначається шляхом порівняння отриманої загальної оцінки та розрахованих значень показників з відповідними аналогами, що приймаються за еталонний зразок. Аналогами обирають реально існуючий програмний продукт того самого функціонального призначення, з такими ж основними параметрами, подібної структури та умовами експлуатації.

Задачу багатокритеріальної оцінки даної ієрархічної структури (див. рис. 2.3), у якій Q виступає як функція оцінки зрілості ПП, будемо розглядати як базову при дискретній багатокритеріальній оптимізації, у якій Q виступає як цільова функція [154].

Введемо функцію оцінки трудомісткості забезпечення заданого рівня зрілості ПП як критерій на верхньому (першому) рівні розглядуваної ієрархії. У загальному вигляді така функція залежить від поточних значень показників зрілості ПП та величини їх зміни:

$$H = H(a_{1j}, \ldots, a_{nj}, \Delta a_{1j}, \ldots, \Delta a_{nj}).$$

Визнаним методом оцінки трудовитрат (в людино-місяцях) на розробку програмних продуктів є COCOMO II [55], головною особливістю якої є те, що для виконання розрахунків необхідно знати розмір програмного продукту (або

його компонентів) в тисячах рядків вихідного коду (KSLOC). Також досить відомим є метод PERT [156], в основі якого лежить оцінка мінімально/максимально можливої та найбільш ймовірної трудомісткості виконання кожного елементарного пакету робіт, які входять в розробку ПП. Використання вказаних методів не дає можливості виразити залежність між трудовитратами та величиною Δa_{ij} зміни показників зрілості ПП.

Нехай $\tilde{a}_{ij}$ – значення показника зрілості ПП, яке необхідно досягнути, $\tilde{a}_{ij} = a_{ij} + \Delta a_{ij}$, де a_{ij} - поточне значення показника, Δa_{ij} - величина зміни показника. Мають місце наступні твердження:

1. Чим більше значення $\tilde{a}_{ij}$, тим більше відповідне значення H_{ij}.
2. Існує верхня границя a_{max} для $\tilde{a}_{ij}$.
3. Трудомісткість покращення показника a_{ij} на величину Δa_{ij} при $a_{ij} \to a$ більша за трудомісткість його покращення на ту саму величину при $a_{ij} \to a_{max}$.

Враховуючи вищенаведене, доцільно описати залежність між зміною значення показника зрілості ПП та трудомісткістю її покращення за допомогою рівняння S-подібної кривої, а саме – кривої Гомперця (рис. 2.4) [157]:

$$\tilde{a}_{ij} = a_{max} \cdot b^{c^{H_{ij}}}, \qquad (2.11)$$

де b і c – параметри, $b>0$ і $0<c<1$.

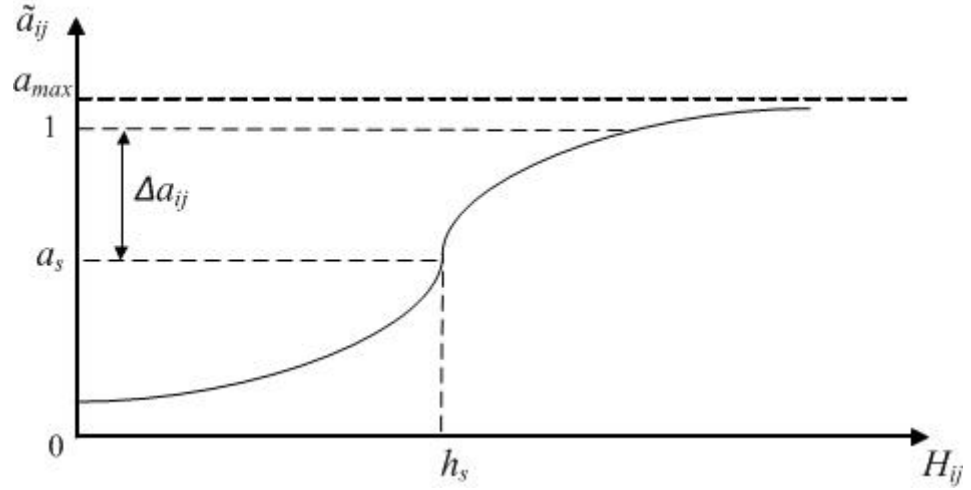

Рис. 2.4 Вид кривої Гомперця для $\tilde{a}_{ij} = \tilde{a}_{ij}(H_{ij})$

Точка $S(h_s; a_s)$ - точка перегину, а пряма $\tilde{a}_{ij} = a_{\max}$ - горизонтальна асимптота кривої Гомперца.

Оцінка параметрів b і c виконується за характерними точками кривої [58], наприклад, значенням функції в початковій точці ($H_{ij} = 0 \Rightarrow \tilde{a}_{ij} = a_{ij} = a_{max} \cdot b^{c^0} = a_{max} \cdot b \Rightarrow b = \frac{a_{ij}}{a_{max}}$), точці перегину (друга похідна прирівнюється до нуля) та кінцевій точці ($\tilde{a}_{ij} \to a_{max}$). Значення функції в згаданих точках залежать від особливостей розробки програмного продукту та мають визначатись із застосуванням відповідної статистики історичних даних з трудомісткості досягнення заданого рівня зрілості ПП.

З рівняння (2.11) виразимо H_{ij} (прийнявши $a_{max} = 1$):

$$H_{ij}(\tilde{a}_{ij}) = \log_c \log_b \tilde{a}_{ij}.$$

Ця функція є нелінійною, спадною ($0<c<1$) та неперервною на області визначення $\log_b \tilde{a}_{ij} > 0$.

Для сумарної трудомісткості виконується рівність (за аналогією з методом PERT):

$$H(\tilde{a}_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{m} H_{ij}(\tilde{a}_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{m} \log_c \log_b \tilde{a}_{ij}, \qquad (2.12)$$

при цьому що більше незалежні між собою заплановані роботи з покращення окремих показників зрілості ПП, то точніша сумарна оцінка трудовитрат.

Задачу забезпечення зрілості ПП_представимо як двокритеріальну (на верхньому рівні ієрархії) задачу оптимізації, при якій найкращий варіант визначатиметься як такий, для якого функція оцінки зрілості ПП буде максимальна, а функція оцінки трудомісткості досягатиме при цьому мінімально можливого значення. Для вирішення задачі доцільно використати метод головного критерію [154], а саме: мінімізувати критерій трудомісткості при заданих обмеженнях на значення функції оцінки зрілості ПП. У якості критеріїв, екстримізація яких має забезпечити необхідне значення функції оцінки зрілості ПП, розглядатимемо критерії третього рівня ієрархічної структури (див. рис. 2.3) – показники зрілості ПП. Це пояснюється тим, що інформація про відповідні

зміни показників є достатньою для формування необхідних управляючих дій з досягнення зрілості ПП, а кількість критеріїв на цьому рівні, як правило, прийнятна для використання методу перебору.

На основі проведеного аналізу пропонується наступна формальна постановка задачі забезпечення зрілості ПП.

Введемо позначення:

n – кількість показників зрілості;

k – значення, яке обмежує кількість показників, за якими одночасно ведеться пошук;

Q' – необхідний рівень зрілості;

$Q(a_{1j},\ldots,a_{nj})$ – функція оцінки зрілості;

$H(\tilde{a}_{ij})$ – функція оцінки трудомісткості, $\tilde{a}_{ij} = a_{ij} + \Delta a_{ij}$;

a_{ij} - поточні значення показників зрілості;

Δa_{ij} - величина зміни показників зрілості.

Необхідно знайти такі значення показників зрілості (обрати відповідний варіант), при яких досягатиметься рівень зрілості не нижче заданого, а цільова функція трудомісткості зміни показників буде досягати мінімуму

$$\min_{\delta_{ij},\tilde{a}_{ij}\in D} H(\delta_{ij},\tilde{a}_{ij}) = \min_{\delta_{ij},\tilde{a}_{ij}\in D} \sum_{i=1}^{n}\sum_{j=1}^{m} \delta_{ij}\cdot \log_c \log_b (a_{ij}+\Delta a_{ij}) = H^*$$

при області допустимих рішень D та обмеженнях:

$$Q\left(a_{1j}+\delta_{1j}\,\Delta a_{1j},\ldots,a_{nj}+\delta_{nj}\,\Delta a_{nj}\right)\geq Q', \tag{2.13}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\delta_{ij}\leq k,\ \delta_{ij}\in\{0,1\} \tag{2.14}$$

$$a_{ij},(\Delta a_{ij}+a_{ij})\in[0;1],$$

$$\Delta a_{ij}\geq 0,\ i=\overline{1,n},\ j=\overline{1,m}.$$

Вказана задача має розв’язки згідно умов (2.13) і (2.14), остання з яких

штучно обмежує кількість показників, за якими одночасно ведеться пошук, для зосередження зусиль на головних напрямках покращення показників

зрілості ПП. Таке обмеження є необхідним при використанні ітераційних технологій розробки ПП з огляду на коротку тривалість ітерацій.

Оскільки при обмеженні кількості показників зрілості ПП, які оптимізуються, кількість варіантів, з яких обирається найкращий розв'язок задачі забезпечення зрілості, є відносно невеликою, то використовуватимемо метод комбінаторної оптимізації – метод прямого перебору (з поверненням) [59]. Зауважимо, що остаточний вибір варіанту, який має оптимальне співвідношення отриманого рівня зрілості ПП та трудомісткості, виконується експертами. Розв'язком поставленої задачі буде набір показників a_{ij} (де $\delta_{ij} = 1$) на яких досягається мінімальне (або одне з мінімальних) значення функції трудомісткості при забезпеченні рівня зрілості ПП не нижче заданого. Також надається інформація про початкове значення показників та величину зміни, яка необхідна для досягнення заданого рівня зрілості ПП.

Згідно заданих умов, у якості моделі забезпечення зрілості ПП виступає *n*-вимірний простір (*n* – кількість показників). Значення показників змінюються від поточного до абсолютного (одиниці) з кроком Δ. У точках простору задано: рівень ЗВ в заданий момент часу – *Q*, значення трудомісткості забезпечення даного рівня у точці – *H*. Задача зводиться до відшукання точки у просторі, для якої виконується:

$$Q \geq Q';\ \forall (Q'', H'): H' < H = H^* \Rightarrow Q'' < Q'.$$

При застосуванні описаної моделі виникає запитання щодо впливу коригуючих дій із покращення певних показників на рівень зрілості ПП. Уникнути зниження останнього можна, встановивши існування зв'язку між показниками зрілості ПП за допомогою кореляційно-регресійного аналізу.

2.5. Кореляційно-регресійний аналіз показників зрілості ПП

У попередньому параграфі зазначалося, що важливим є питання визначення залежності між показниками зрілості ПП. Однак, не завжди експерт, який розробляє моделі оцінки та забезпечення ЗВ, може в явному вигляді виразити існуючі залежності між змінними. В рамках дисертаційного дослідження

пропонується використати алгоритм знаходження явного виду емпіричних залежностей для двох показників, який ґрунтується на кореляційно-регресійному аналізі.

До основних завдань кореляційно-регресійного аналізу належать такі:

- встановлення наявності зв’язку між досліджуваними ознаками;
- виявлення виду функції зв’язку (специфікація моделі);
- знаходження числових значень коефіцієнтів функції зв’язку (оцінювання параметрів моделі);
- оцінювання достовірності отриманих результатів (тестування моделі на адекватність).

Послідовність здійснення процедури кореляційно-регресійного аналізу зв’язку між статистичними показниками зображена на рис. 3.8. При цьому термін «кореляція» використовується для оцінювання щільності зв'язку між ознаками, а термін «регресія» – для опису вигляду і параметрів функції зв’язку (регресійної моделі). За числом факторних ознак, які входять в регресійну модель, будемо розрізняти однофакторні та багатофакторні моделі (моделі множинної регресії).

У випадку оцінки користувачами метрик зрілості ПП дані, якими є обчислені показники, задаються числовими рядами значень. Розглядаючи будь-яку пару показників, ми припускаємо, що один із них представлений числовим рядом значень залежної величини (y_k), а інший - незалежної (x_k), кожна з яких в загальному випадку крім певної регулярної (детермінованої) складової може містити й випадкові складові різної природи, зумовлені як статистичною природою досліджуваних процесів, так і зовнішніми факторами процесів вимірювань і перетворення даних.

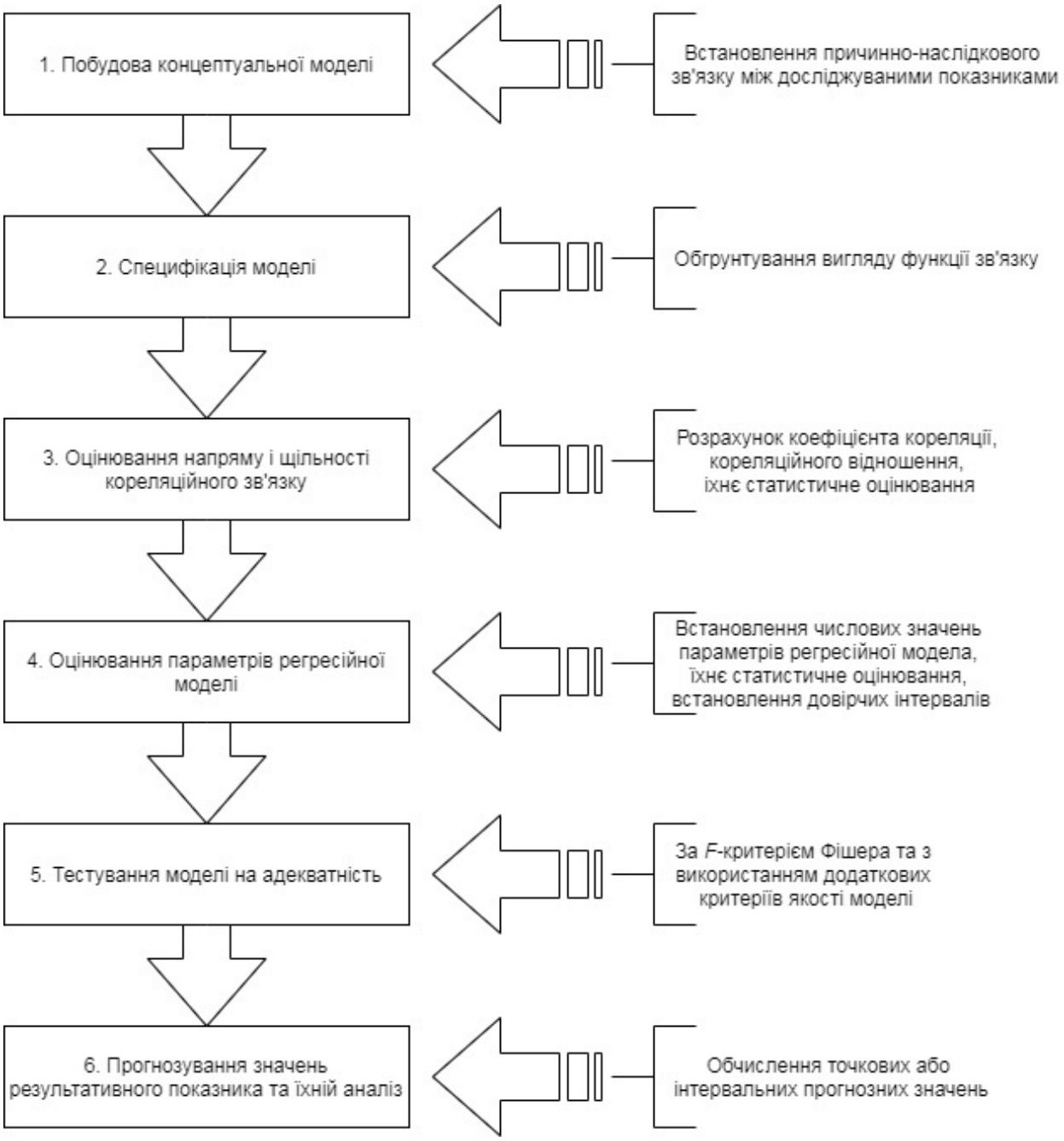

Рис. 2.5. Етапи побудови регресійної моделі

Незалежна змінна x_k зазвичай покладається детермінованою, а, отже, її випадкова складова "переноситься" на залежну змінну y_k. Припускається також, що значення випадкової складової залежної змінної (як власні, так і "сумарні") розподілені за деяким ймовірнісним законом (наприклад, нормальним). Залежність однієї випадкової величини від значень, які приймає інша випадкова величина (показник зрілості ПП), є регресією.

Процедура пошуку передбачуваної залежності між рядами значень показників зрілості ПП включає наступні етапи:

- Встановлення сили та значущості зв'язку між ними (кореляційній аналіз);

- Можливість подання цієї залежності у формі математичного виразу - рівняння регресії (регресійній аналіз).

Кореляційний аналіз. Для кількісної оцінки існування зв'язку між досліджуваними сукупностями випадкових величин використовується спеціальний статистичний показник - коефіцієнт кореляції r. Коефіцієнт r - це безрозмірна величина, вона може змінюватися від 0 до ± 1. Чим ближче значення коефіцієнта до одиниці (неважливо, з яким знаком), тим з більшою впевненістю можна стверджувати, що між двома розглянутими показниками зрілості існує лінійний зв'язок. Якщо виявиться, що r = 1 (або -1), то має місце класичний випадок чисто функціональної залежності (тобто реалізується ідеальний взаємозв'язок). Існують різні аналітичні прийоми визначення коефіцієнта кореляції. При дослідженні використовувались формули для лінійного та рангового коефіцієнта кореляції, розрахунок яких реалізовано в програмному комплексі.

Лінійний коефіцієнт кореляції (коефіцієнт кореляції Пірсона) розраховується таким чином, що у випадку лінійного зв’язку між показниками, він точно встановлює тісноту цього зв’язку. Використання даного коефіцієнту припускає, що показники мають нормальний розподіл. Формула для розрахунку має наступний вигляд [192]:

$$r_{XY} = \frac{cov_{XY}}{\sigma_X \sigma_Y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}},$$

де x_i - числові значення, які приймає показник X,

y_i - числові значення, які приймає показник Y ,

$\bar{x}$ - математичне сподівання по X,

$\bar{y}$ - математичне сподівання по Y.

З метою якісного тлумачення числових значень коефіцієнта кореляції Пірсона і кореляційного відношення використовуються таку шкалу (таблиця 2.1).

Оцінка значимості коефіцієнта парної кореляції проводиться шляхом порівняння його абсолютної величини з табличним (або критичним) показником $r_{крит}$, значення якого беруться зі спеціальної таблиці, наведеної, наприклад, в [192]. Якщо $\left|r_{розрах} \geq r_{крит}\right|$, то із заданою ймовірністю (зазвичай 95 %) можна стверджувати, що між розглядуваними показниками існує значимий лінійний зв'язок. У зворотному випадку робиться висновок про відсутність такого зв'язку.

Таблиця 2.4

Якісна оцінка щільності зв'язку між коефіцієнтом кореляції Пірсона і кореляційним відношенням

Значення	Якісна оцінка щільності зв’язку
0 – 0,1	Зв'язок відсутній
0,1 – 0,3	Зв'язок слабкий
0,3 – 0,5	Зв'язок помірний
0,5 – 0,7	Зв'язок відчутний
0,7 – 0,9	Зв'язок щільний
0,9 – 0,(9)	Зв'язок дуже щільний

Коефіцієнт кореляції Спірмена, заснований на використанні рангів, використовується найчастіше для оцінки зв'язку якісних ознак, вимірюваних в порядковій шкалі. Але також може застосовуватись і для кількісних ознак при умові ранжування їх значень. Відповідна формула має вигляд:

$$p = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\left(n^2 - 1\right)},$$

d_i – різниця між рангами факторної та результативної ознак,

n – число парних членів ряду значень показників.

Коефіцієнт рангової кореляції застосовується в дослідженні при перевірці узгодженості ранжувань експертів, про що йтиметься далі. Для визначення сили зв'язку між показниками застосовується лінійний коефіцієнт кореляції.

Для випадку, коли розподіл значень показників не є нормальним, передбачено наступні основні перетворення вхідних даних:

1. $x' = \lg x$. При логарифмуванні вхідних даних ліва гілка кривої розподілу сильно розтягується, і розподіл приймає наближено нормальний характер.
2. $x' = \lg(x \pm a)$. Має місце при асиметричному розподілі з однією вершиною.
3. $x' = \frac{1}{x}$. Перетворення «обернена величина» вважається найбільш потужним.
4. $x' = x^a$. Степеневе перетворення слугує для нормалізації зсунутого вправо розподілу. При цьому a = 1,5 при помірному і a = 2 при сильно вираженому правому зсуненні.

Алгоритм визначення сили та значимості зв'язку між обчисленими, на основі користувацьких оцінок метрик, значеннями показників зрілості представлено на рис. 2.6.

Регресійний аналіз. У випадку, коли на етапі кореляційного аналізу були визначені пари показників зрілості ПП, які мають значимий та сильний зв'язок, постає питання опису його форми, тобто рівняння регресії. Для точного опису рівняння регресії необхідно знати умовний закон розподілу залежного показника Y за умови, що показник X приймає значення *x*. На практиці отримати таку інформацію, як правило, не вдається, оскільки дослідник має лише вибірку пар значень (x_i, y_i) обмеженого об'єму *n*.

У даному випадку мова може йти лише про оцінку (апроксимацію) за вибіркою функції регресії. Такою оцінкою є вибіркова лінія (крива) регресії [62]:

$$\hat{y} = \hat{\varphi}(x, b_0, b_1, ..., b_p),$$

де $\hat{y}$ - умовна групова середня показника Y при фіксованому значенні показника X= *x*, $b_0, b_1, ..., b_p$ - параметри кривої.

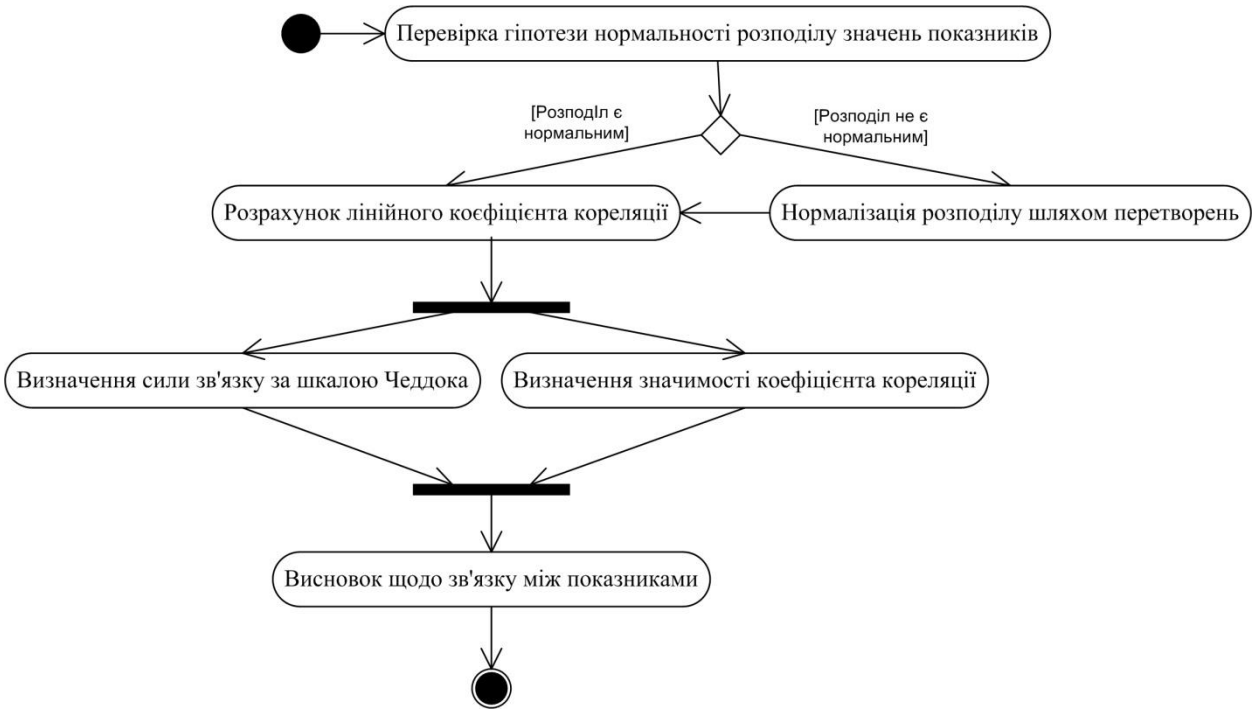

Рис. 2.6 Алгоритм визначення зв’язку між показниками зрілості ПП

Апроксимуючі рівняння (емпіричні формули) доцільно використовувати, оскільки:

1. Точний аналітичний вираз залежності між досліджуваними показниками зрілості ПП може залишатись невідомим і тому за необхідністю доводиться обмежуватись наближеними формулами емпіричного характеру.

2. Точна функціональна залежність виражається формулою настільки складною, що її безпосереднє застосування при обчисленнях є проблематичним.

Залежність між однією й тією ж самою парою показників можна виразити різними за видом рівняннями регресії. Найточніша апроксимація характеризується найбільшим співпадінням значень, обчислених за формулою, з дослідними даними.

Рівняння регресії перевіряється на значимість (адекватність моделі) за допомогою F-критерія (критерія Фішера):

$$F_{емп} = \frac{S_X^2}{S_Y^2},$$

де S_X^2 і S_Y^2 - відповідно дисперсії двох вибірок, членами яких є значення залежних показників.

Перед застосуванням критерію Фішера необхідно перевірити виконання наступних умов:

1. Значення показників вимірюються в шкалі інтервалів або відношень.

2. Значення досліджуваних залежних показників розподілені за нормальним законом.

3. Більша за величиною дисперсія знаходиться в чисельнику, а менша – в знаменнику.

Отже, якщо гіпотеза нормальності розподілу не підтверджується, то необхідно виконати відповідні математичні перетворення вхідних даних.

Для оцінки величини похибки в передбаченні емпіричних результатів використовуються табличні значення $F_{крит}$ [192]. Якщо $F_{емп} < F_{крит}$, то модель визнається адекватною, тобто з заданим ступенем достовірності вона вірно передбачає реальний результат. Якщо $F_{емп} > F_{крит}$, то робиться обернений висновок [173].

Результат виявлення форми зв'язку показників суттєво залежить від наступних основних факторів:

1. Вибору міри близькості залежної змінної (показника) до шуканої функції і методу побудови наближення (параметрів математичної моделі).

2. Вибору відповідного класу функції апроксимації (степеневої, логарифмічної тощо), який відповідає фізичній природі залежності між показниками зрілості ПП.

3. Методу оптимізації порядку модельної функції або числа членів ряду апроксимуючого виразу.

В рамках дисертаційного дослідження вищевказані фактори визначені наступним чином:

1. Міра наближення – квадратична. Реалізується в методі найменших квадратів (МНК) та забезпечує максимальну правдоподібність функції

наближення при нормальному розподілі випадкової складової залежного показника Y:

$$\sum_k [y_k - \varphi(x_k)]^2 \to \min,$$

де $\varphi(x_k)$ - значення апроксимуючої функції.

2. У роботі розглядаються та програмно реалізовані наступні функції апроксимації:

- лінійна: $y = ax + b$;
- експоненціальна: $y = a + b \cdot e^{cx}$;
- логарифмічна: $y = a + b \cdot \ln x$;
- гіпербола: $y = a + \frac{b}{x}$;
- квадратична: $y = a + bx + cx^2$;
- кубічна: $y = a + bx + cx^2 + dx^3$;
- поліном степеня 4: $y = a + bx + cx^2 + dx^3 + ex^4$;
- поліном степеня 5: $y = a + bx + cx^2 + dx^3 + ex^4 + fx^5$.

При інших рівних умовах доцільно використовувати функції з мінімальною кількістю параметрів, що забезпечує більше число степенів свободи, тобто менші значення дисперсії залишків.

3. Порядок моделі, як показали експериментальні дослідження на реальних оцінках користувачів, пропонується в більшості випадків використовувати не більше третього. При підвищенні порядку моделі в функцію апроксимації входить не лише регулярна складова даних, але все більша частка випадкових складових, яка з певного моменту не лише не наближує функцію апроксимації до регулярних складових, а навпаки – збільшує розходження.

Для оцінки якості наближення використовується коефіцієнт детермінації:

$$R^2 = \frac{D_\varphi}{D_Y}, \quad D = \frac{\left\{\sum_k [y_k - \varphi(x_k)]^2\right\}}{k - m},$$

де D_φ та D_Y – дисперсії функції наближення та даних відповідно, m – кількість параметрів функції наближення, $(k-m)$ – число степенів свободи.

Якість апроксимації тим більша, чим ближче до 1 значення коефіцієнта детермінації [173].

Алгоритм регресійного аналізу залежності між показниками зрілості ПП зображено на рис. 2.7. Алгоритми кореляційного та регресійного аналізу, представлені на рис. 2.6 і рис. 2.7, були реалізовані в рамках розробленої ПС.

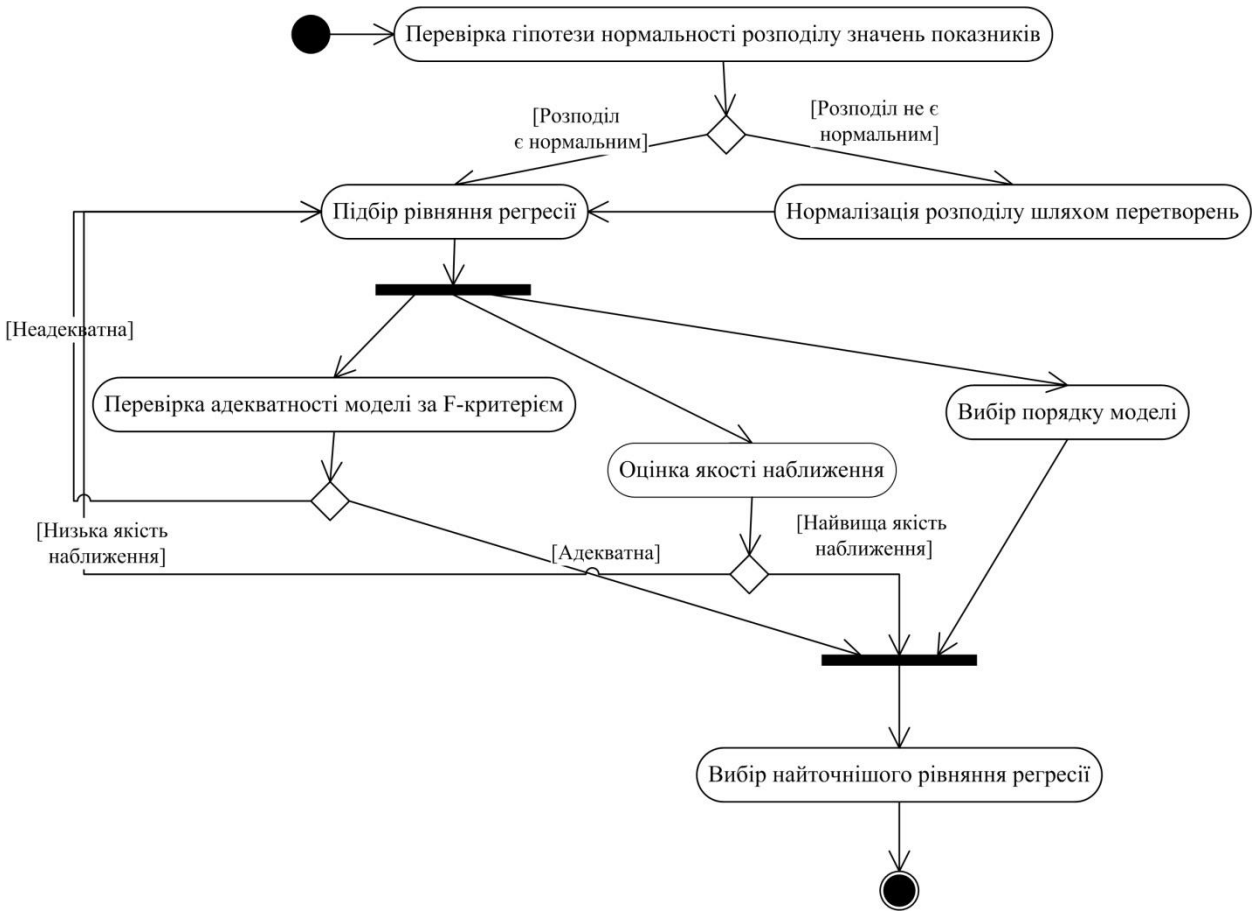

Рис. 2.7. Регресійний аналіз залежності між показниками зрілості ПП

Пропонується методика реалізації процесу управління змінами зрілості ПП, заснована на математичному апараті кореляційно-регресійного аналізу, що дозволяє встановити залежності між показниками зрілості ПП на основі оцінок користувачів. Встановлені залежності враховуються при аналізі впливу коригуючих дій на рівень зрілості ПП. Методика включає наступні етапи:

1. Встановлення сили та значущості зв'язку між показниками зрілості ПП. За умови ненормального розподілу значень показників зрілості ПП виконуються основні математичні перетворення вхідних даних.

2. Зображення залежності між показниками, за умови сильного і значущого зв'язку, у формі математичного виразу – рівняння регресії.

3. Аналіз варіанта забезпечення зрілості ПП на наявність показників, пов'язаних з іншими. Якщо такі показники є, то:

а) при прямій залежності показників варіант може бути використаний;

б) при оберненій залежності показників необхідно відкинути цей варіант зміни показників зрілості ПП.

Крім цього, рівень зрілості ПП можна визначити за допомогою методу сигм. Для цього необхідно визначити рівень якості ПЗ (QL), рівень цілісності ПЗ (SI) та застосувати функцію σ, яка повертає обернене значення нормального інтегрального розподілу для вказаного середнього та стандартного відхилення (табл. 2.5). Відповідно до значення σ були визначені наступні п’ять рівнів зрілості ПП:

• Гарантований ($\sigma \geq 5$)

• Сертифікований ($4 \leq \sigma < 5$)

• Нейтральний ($3 \leq \sigma < 4$)

• Задовільний ($2 \leq \sigma < 3$)

• Незадовільний ($\sigma < 2$)

Рівень цілісності ПЗ використовується для класифікації ПП. Тому ПП з високим рівнем цілісності дуже чутливі до якості. Для того, щоб визначити рівень цілісності ПП, спочатку необхідно визначити наслідки будь-якої несправності цього ПП. Таблиця 2.6 ілюструє визначення 6 наслідків. Крім того, настання цих наслідків є дуже важливим для визначення рівня цілісності. Частоту їх виникнення можна оцінити, використовуючи орієнтовну частоту (IFreq) попередніх та аналогічних ПП (табл. 2.7). Виходячи з наслідків та їх виникнення,

можна класифікувати рівні цілісності ПЗ за упорядкованою шкалою від 0 до 5 (табл. 2.8).

Таблиця 2.5.

Відповідність значень σ рівню якості та рівню цілісності

Рівні цілісності ПЗ (Software Integrity, SI)						
5	4	3	2	1	0	
Рівні якості ПЗ (Quality Levels, QL)						
0 σ зсув	1.5 σ зсув	2.0 σ зсув	2.5 σ зсув	3.0 σ зсув	3.5 σ зсув	σ
QL ≥ 99.99997%	QL ≥ 99.976%	QL ≥ 99.865%	QL ≥ 99.379%	QL ≥ 97.724%	QL ≥ 93.319%	**σ ≥ 5**
QL < 99.99997% і QL ≥ 99.996%	QL < 99.976% і QL ≥ 99.379%	QL < 99.865 і QL ≥ 97.724%	QL < 99.379% і QL ≥ 93.319%	QL < 97.724% і QL ≥ 84.134%	QL < 93.319% і QL ≥ 69.146%	**5 > σ ≥ 4**
QL < 99.996% і QL ≥ 99.865%	QL < 99.379% і QL ≥ 93.319%	QL < 97.724% і QL ≥ 84.134%	QL < 93.319% і QL ≥ 69.146%	QL < 84.134% і QL ≥ 50%	QL < 69.146% і QL ≥ 30.853%	**4 > σ ≥ 3**
QL < 99.865% і QL ≥ 97.724%	QL < 93.319% і QL ≥ 69.146%	QL < 84.134% і QL ≥ 50%	QL < 69.146% і QL ≥ 30.853%	QL < 50% і QL ≥ 15.865%	QL < 30.853% і QL ≥ 6.680%	**3 > σ ≥ 2**
QL < 97.724%	QL < 69.146%	QL < 50%	QL < 30.853%	QL < 15.865%	QL < 6.680%	**σ < 2**

Табл. 2.6.

Визначення наслідків

Наслідок	Визначення
Катастрофічний (Catastrophic)	Повна відмова роботи системи, втрата безпеки системи або великі фінансові чи соціальні втрати
Критичний (Critical)	Великий вплив, серйозні пошкодження системи або великі фінансові або соціальні втрати
Важкий (Severe)	Погіршення чи фінансові або соціальні втрати
Граничний (Marginal)	Стійкий вплив
Незначний (Minor)	Незначний вплив на продуктивність системи
Немає (None)	Впливу немає

Табл. 2.7

Орієнтовна частота для кожного випадку

Поява	Орієнтовна частота (IFreq) (на рік)
Часто (Frequent)	IFreq>1
Ймовірно (Proba ble)	$0{,}1 < \text{IFreq} \leq 1$
Можливо (Occasional)	$0{,}01 < \text{IFreq} \leq 0{,}1$
Зрідка (Remote)	$0{,}0001 < \text{IFreq} \leq 0{,}01$
Дуже рідко (Improbable)	$0{,}000001 < \text{IFreq} \leq 0{,}0001$
Майже ніколи (Incredible)	$\text{IFreq} \leq 0{,}000001$

Табл. 2.8

Визначення рівня цілісності програмного забезпечення з використанням наслідків та їх виникнення

Наслідок	Поява					
	Часто	Ймовірно	Можливо	Зрідка	Дуже рідко	Майже ніколи
Катастрофічний	5	5	5	4	3	2
Критичний	5	5	4	3	2	1
Важкий	4	4	3	2	1	1
Граничний	3	3	2	2	0	0
Незначний	2	2	1	1	0	0
Немає	0	0	0	0	0	0

Рівень зрілості ПП визначається за наступною методикою:

1. Визначити відповідний рівень якості ПЗ.
2. Визначити значення σ.
3. Визначити відповідний рівень цілісності.

4. Відповідно до визначених рівнів і використовуючи табл. 2.5 визначити зсув σ.

5. Знайти значення $\sigma + \text{зсув } \sigma$.

6. Порівняти отримане значення з визначеними рівнями для σ.

2.6. Метод оцінювання зрілості коду ПЗ

Основним методом досліджень програмного коду є вимірювання, а одним із головних інструментів для цих досліджень є метрики [157]. Існує три методи підбору метрик для властивостей програмного забезпечення:

- ціль–питання–метрика,
- модель відповідальної особи
- стандартизовані метрики [177].

Перший метод передбачає три етапи підбору метрик для властивості [179]:

– формулювання цілі, яка буде досягнута шляхом вимірювання программного забезпечення;

– формулювання питань на які потрібно отримати відповідь щоб зрозуміти, чи досягається ціль;

– визначаються метрики, які можна використати для відповіді на запитання, що визначені на другому етапі. Другий метод передбачає підбір необхідних метрик безпосередньо відповідальною особою, який пов'язується з задачами, які повинні бути вирішені та рішеннями, які потрібно приймати при вирішенні даних задач. Рішення про вимірювання метрики приймаються на основі досвіду даної особи або за рекомендаціями експертів. Третій метод пов'язаний з використанням рекомендацій або стандартів деяких організацій, які займаються дослідженнями в потрібній сфері [177]. Цей метод базується на проведених емпіричних досліджень. Перші два методи цілком не формалізовані, третій – напівформалізований, оскільки його використання залежить від кваліфікації та досвіду експертів, а також рекомендації з підбору метрик носять загальний характер. Усі три методи дають можливість вирішувати поставлену задачу, тобто підбирати метрики для дослідження властивостей программного забезпечення.

Проте вони не є формалізованими, що накладає певну долю ймовірності отримання неточних значень. Крім цього, в результаті використання цих методів можна отримати лише набір метрик, які можуть слугувати для оцінки властивості. Але вони не дозволяють визначити метрики, які найкраще підходять для оцінки властивості. Підбір метрик для дослідження тих чи інших властивостей програмного забезпечення пропонується можна за допомогою використання предметно-орієнтованого методу побудови залежностей між метриками програмного забезпечення [179].

Розрізняють прямі та непрямі метрики програмного коду. Прямі метрики піддаються вимірюванню, але їх недостатньо для оціювання більшості властивостей. А непрямі метрики формуються на основі прямих. Головне завдання полягає у визначенні виду та ступеня залежності непрямої метрики від прямої [178]. Для цього використовують два підходи – статистичний аналіз та нейронні мережі [177]. У рамках дослідження відбувається вимірювання прямих метрик програмного забезпечення, збір даних – результатів вимірювань, обробка даних та визначення залежностей непрямих метрик від прямих. Розробка методу визначення залежностей між метриками програмного коду здійснюється за допомогою статистичного аналізу з урахуванням особливостей програмного забезпечення. У загальному вигляді статистичний аналіз, який виконується з метою визначення залежностей, складається з трьох етапів: первинний статистичний аналіз, кореляційний аналіз та регресійний аналіз [4]. Залежності будуються на етапах кореляційного та регресійного аналізу. Для визначення наявності залежності непрямої метрики від прямої проводиться кореляційний аналіз. Загалом кореляційний аналіз можна проводити двома шляхами [192]: простий розрахунок коефіцієнтів парної кореляції та розрахунок парної рангової кореляції. Перший використовується тоді, коли досліджувані величини мають нормальний розподіл, другий – коли нормального закону розподілу немає. Під час емпіричних досліджень програмного забезпечення щодо визначення залежності непрямої метрики від прямої необхідно використати розрахунок парної рангової кореляції, що пов’язано з розподілом, відмінним від нормального. Для визначення

виду залежності непрямої метрики від прямої застосовується регресійний аналіз. Він полягає в побудові та розрахунках коефіцієнтів функції регресії, яка відображає залежність непрямої метрики від прямої.

Суть статистичного аналізу полягає в тому, що для визначення виду залежності непрямої метрики від прямої використовується розрахунок тільки парної рангової кореляції метрик без великої кількості перевірок та будуються тільки нелінійні функції регресії у зв'язку з відсутністю нормального розподілу метрик без проведення первинного статистичного аналізу. Запропонований метод дає змогу визначати залежність непрямої метрики від прямої без попереднього визначення їх законів розподілу та визначити лінію регресії без попереднього аналізу досліджуваних величин. Отже, запропонований метод статистичної обробки даних досліджень програмного забезпечення характеризується так (рис. 2.8): відсутність визначення закону розподілу метрики; обов'язкове велике значення вибірки; використання розрахунку тільки парної рангової кореляції; відсутність перевірки точності коефіцієнтів кореляції; відсутність перевірки спільного закону розподілу метрик; побудова регресії методом лінеаризації [177].

У деяких наукових джерелах під зрілістю програмного коду розуміють його «вік». Проте стан програмного коду (якість) за обумовленою шкалою та за відповідними метриками і буде характеризувати рівень зрілості. Поняття якості програмного коду є складовою якості ПП. Якість коду може визначатись різними критеріями. Деякі з них мають значення тільки з точки зору людини. Наприклад, форматування тексту програми – неважливо для комп'ютеру, але може мати велике значення для супроводу. Багато з існуючих стандартів кодування, що визначають специфічні для мови програмування угоди та задають низку правил, мають на меті полегшити супровід ПЗ в майбутньому. Також існують інші критерії, що визначають якість написання коду, наприклад, такі, як структурованість – ступінь логічного розділення коду на блоки [179].

Метрика – це кількісний масштаб і метод, який може використовуватися для вимірювання [170]. Введення і використання метрик необхідно для поліпшення контролю над процессом розробки, зокрема над процесом тестування [175].

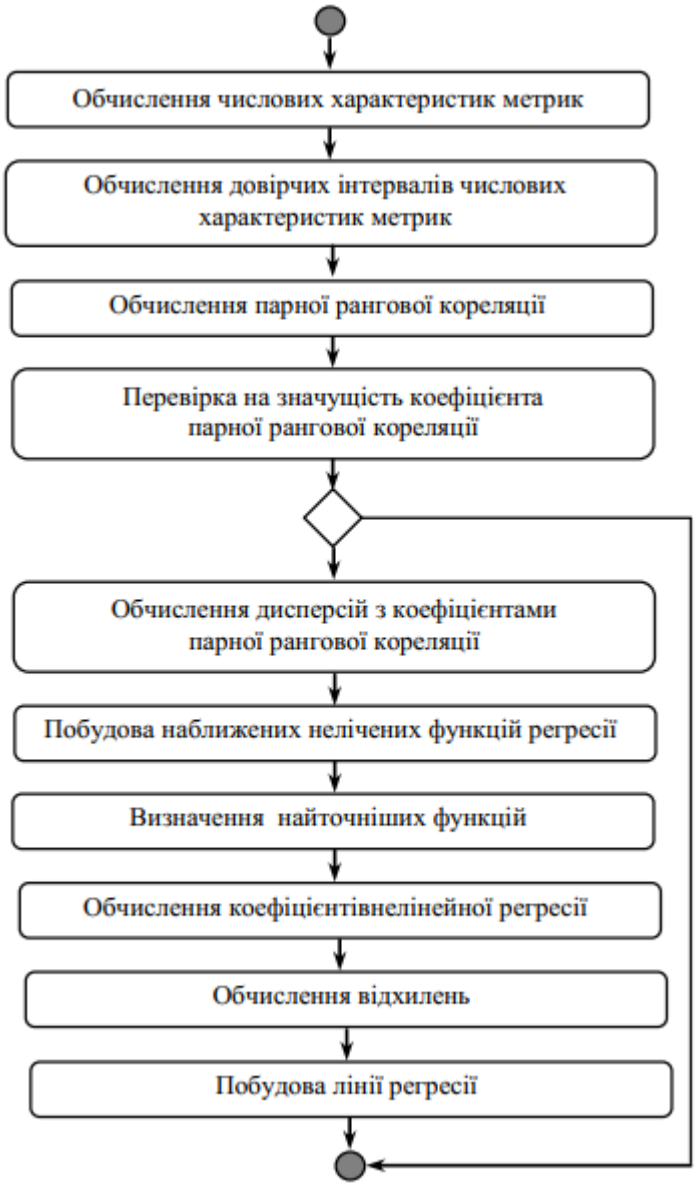

Рис. 2.8. Метод визначення залежностей між метриками програмного коду за допомогою статистичного аналізу

Метрики складності програм прийнято розділяти на 3 основні групи [175]:

- метрики розміру програм;
- метрики складності потоку управління програм;

метрики складності потоків даних програм. Метрики розміру програм базуються на визначенні кількісних характеристик, пов'язаних з розміром програми, і відрізняються відносною простотою. Метрики цієї групи орієнтовані на аналіз вихідного тексту програм, тому вони можуть використовуватись для

оцінки складності проміжних продуктів розробки. До найбільш відомих метрик даної групи відносять кількість операторів програми, кількість рядків вихідного тексту, набір метрик Холстеда [179].

Метрики складності потоку управління програм базуються на аналізі керуючого графу програми. Метрики другої групи теж можуть застосовуватись для оцінки складності проміжних продуктів розробки. Представником другої групи є метрика Маккейба [176].

Метрики складності потоків данпх програм базуються на оцінці використання, конфігурації та розташування даних в програмі. В першу чергу це стосується глобальних змінних. До даної групи відносять метрики Чепіна [156].

Розмірно-орієнтовані метрики (показники оцінки обсягу) прямо вимірюють програмний продукт і процес його розробки. Базуються такі метрики на LOC-оцінках (LOC – Lines of Code). Кількість рядків вихідного коду (Lines of Code – LOC, Source Lines of Code – SLOC) є найбільш простим і поширеним способом оцінки обсягу робіт по проекту. Ці метрики не універсальні, особливо це відноситься до показника LOC, який істотно залежить від використовуваної мови програмування.

В залежності від того, яким чином враховується вихідний код, виділяють 2 основних показники LOC [179]: кількість «фізичних» рядків коду (LOC, SLOC, KLOC, KSLOC, DSLOC) – визначається як загальна кількість рядків вихідного коду, включаючи коментарі і порожні рядки; кількість «логічних» рядків коду (LSI, DSI, KDSI, де SI-source instructions) – визначається як кількість команд і залежить від використовуваної мови програмування. В тому випадку, якщо мова не допускає розташування декількох команд на одному рядку, то кількість "логічних" LOC буде відповідати кількості "фізичних", за винятком кількості порожніх рядків і рядків коментарів.

Метрика Холстеда належить до метрик, які обчислюються на основі аналізу кількості рядків та синтаксичних елементів вихідного коду програми.

Основу метрики Холстеда складають 4 вимірювані характеристики програми [156]: $NUOprtr$ (Number of Unique Operators) – кількість унікальних

операторів програми, включаючи символи- роздільники, імена процедур і знаки операцій (словник операторів); $NUOprnd$ (Number of Unique Operands) – кількість унікальних операндів програми (словник операндів); $NOprtr$ (Number of Operators) – загальна кількість операторів в програмі; $NOpnd$ (Number of Operands) – загальна кількість операндів в програмі.

На основі цих характеристик розраховуються оцінки: словник програми, довжина програми, обсяг програми, оцінка її реалізації, складність її розуміння, трудомісткість кодування, інформаційний вміст, оптимальна модульність, складність програми

$$HDiff = (NUOprtr / 2) * (NOprnd / NUOprnd).$$

Показник цикломатичної складності Маккейба є одним з найбільш розповсюджених показників оцінки складності програмних проектів. Цикломатичне число Маккейба показує необхідну кількість проходів для покриття всіх контурів сильнозв'язаного графа або кількість тестових прогонів програми, необхідних для вичерпного тестування за принципом «працює кожна гілка програми». Показник цикломатичної складності розраховується для модуля, метода та інших структурних одиниць програми.

Показник цикломатичної складності дозволяє не лише провести оцінку трудомісткості реалізації окремих елементів програмного проекту і скоригувати загальні показники оцінки тривалості та вартості проекту, але й оцінити зв'язані ризики і прийняти необхідні управлінські рішення.

Існує декілька модифікацій метрик Чепіна. Суть найпростішого з точки зору практичного використання і досить ефективного методу полягає в оцінці інформаційної міцності окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку введення виведення. Вся множина змінних, яка складає список введення-виведення, розбивається на 4 функційні групи [156]: множина «P» – змінні, що вводяться для розрахунків та для забезпечення виведення; множина «M» – модифіковані або створювані всередині програми змінні; множина «C» – змінні, які приймають участь в управлінні

роботою програмного модуля (керуючі змінні); множина «T» – не використовувані в програмі («паразитні») змінні.

Значення метрики Чепіна:

$Q = P + 2* M + 3*C + 0.5* T$.

Для оцінки складності програми використовуються й інші метрики [158]:

метрики Джилба – кількість операторів циклу, кількість операторів умови, кількість модулів або підсистем, відношення кількості зв'язків між модулями до кількості модулів;

метрика Шнадевида – кількість шляхів в графі керування;

метрика Майєрса – інтервальна міра;

метрика Хансена – пара (цикломатична кількість, кількість операторів);

метрика Чена – топологічна міра;

метрика Вудворда – кількість вузлів передач управління;

метрика Кулика – кількість найпростіших циклів;

метрика Хура – цикломатична кількість мереж Петрі;

метрики Вітворфа, Зулевського – міра складності потоку керування, міра складності потоку даних;

метрика Петерсона – кількість багатовходових циклів;

метрики Харрісона, Мейджела – функційна кількість, функційне відношення, регулярні вирази;

метрика Пивоварського – модифікована цкломатична міра складності;

метрика Пратта – тестуюча міра;

метрика Кантоне – характеристичні числа поліномів графу програми;

метрика Мак-Клура – міра складності, заснована на кількості можливих шляхів виконання програми, кількості керуючих конструкцій та змінних;

метрика Кафура – міра на основі концепції інформаційних потоків;

метрика Схуттса, Моханті – ентропійні міри;

метрика Коллофело – міра логічної стабільності програм;

метрика Зольновського, Сімонса, Тейєра – зважена сума різних індикаторів;

метрика Берлінгера – інформаційна міра;

метрика Шумана – складність з позиції статистичної теорії мови;

метрика Янгера – логічна складність з врахуванням історії обчислень;

метрика Тая – покращення метрики Маккейба;

метрика Кокола – комплексна метрика, заснована на більш простих; метрики зв'зності (зчеплення) – ступінь залежності кожного модуля від кожного з інших модулів за даними, за зразком, за управлінням, за зовншініми посиланнями, за загальною областю, за змістом (кількісний показник – ступінь зчеплення).

Для оцінки складності програми на етапі розробки специфікації вимог до програми використовується метрика прогнозованої кількості операторів Nпрогн програми [156]: Nпрогн = NF* Nод , де NF – кількість функцій або вимог в специфікації вимог до розроблюваної програми;Nод – одиничне значення кількості операторів (середня кількість операторів, які доводяться на одну середню функцію або вимогу).

Оцінка складності програми на етапі визначення архітектури [158]:

$$C = \frac{NI}{NF * NI_{\text{од}} * K_{\text{скл}}},$$

де NI – загальна кількість змінних, які передаються по інте- рфейсах між компонентами програми (статистична); $NIод$ – одиничне значення кількості змінних, які передаються по інтерфейсах між компонентами (середня кількість змінних, які передаються по інтерфейсах, що доводяться на одну середню функцію або вимогу); Кскл – коефіцієнт складності розроблюваної програми, враховує ріст одиничної складності програми (складності, що доводиться на одну функцію або вимогу специфікації вимог до програми) для великих та складних програм в порівнянні з середнім показником складності.

В якості метрик процесу проектування використовуються:

– очікувана LOC-оцінка – за кожною функцією експерти надають краще, гірше та імовірне значення, тоді очікувана LOC-оцінка обчислюється за формулою:

$LOC_{\text{оч}_i} = (LOC_{\text{кращ}_i} + LOC_{\text{гірш}_i} + 4 * LOC_{\text{імов}_i}) / 6$;

Є вільно поширювані інструменти очікуваної LOC-оцінки [156]; ця метрика на етапі проектування має прогнозоване значення;

– метрики складності ПЗ – метрики Холстеда та Маккейба; з метрик Холстеда на етапі проектування використовуються [158]: міра довжини модуля ($N = n1\ log2\ (n1)\ +\ n2\ log2\ (n2\ )$, де n_1 – кількість різних операторів, n_2 – кількість різних операндів) та обсяг модуля як кількість символів для запису всіх операторів іоперандів тексту програми ($V = N * \log_2 (n_1 + n_2)$) ; для оцінки складності ПС, виходячи з топології внутрішніх зв'язків, Маккейб розробив метрику цикломатичної складності V(G) = E – N + 2 , де E – кількість дуг, а N – кількість вершин в керуючому графі ПЗ; ці метрики на етапі проектування мають прогнозоване значення;

– метрики Чепіна – аналізують характер використання змінних зі списку введення, тобто оброблювану інформацію; ці метрики вже на етапі проектування мають точне значення;

– метрики зв'язності (зчеплення) – чим вище зв'язність модуля з іншими модулями, тим вища чутливість до внесення змін, тобто високий ступінь зв'язності (зчеплення) модуля з іншими модулями – це суттєвий недолік; ці метрики вже на етапі проектування мають точне значення;

– метрика Джилба – на етапі проектування можна тільки підрахувати кількість модулів та відношення кількості зв'язків між модулями до кількості модулів; ці складові метрики вже на етапі проектування мають точне значення;

– метрика Хансена – ця метрика на етапі проектування має прогнозоване значення;

- метрика Мак-Клура – метрика, спрямована на вимірювання архітектури системи; ця метрика вже на етапі проектування має точне значення;

- метрика Кафура – метрика, заснована на врахуванні потоку даних; оскільки на етапі проектування вже є відомості про оброблювану інформацію,

то й метрика має точне значення вже на етапі проектування;

– метрика Зольновського, Сіммонса, Тейєра – складова метрики (структура, взаємодія, обсяг, дані) обчислюється вже на етапі проектування і має точне

значення;

– метрика звертання до глобальних змінних – пара (модуль, глобальна змінна); ця метрика вже на етапі проектування має точне значення;

– метрика Кокола – комплексна метрика, яка, як правило, враховує значення метрик Холстеда, Маккейба і LOC, отже, на етапі проектування має прогнозоване значення;

– метрика Вітворфа, Зулевського – складова метрики "міра складності потоку даних" має точне значення вже на етапі проектування;

– загальний час розробки і окремий час для кожної стадії – метрика має прогнозоване значення на етапі проектування;

– час модифікації моделей – метрика має точне значення на етапі проектування;

– час виконання робіт в процесі – метрика має прогнозоване значення на етапі проектування;

– кількість знайдених помилок при інспектуванні - метрика має точне значення на етапі проектування;

– прогнозована кількість операторів програми;

– прогнозована оцінка складності програми;

– очікувана вартість розробки кожної функції:

$ВАРТІСТЬ_i = LOC_{оч_i} * ВАРТІСТЬ_{рядка_i}$,

вартість рядка є константою і не змінюється від реалізації до реалізації;

– прогнозована продуктивність розробки ко- жної функції (на основі продуктивності аналогічних функцій в анлогічних програмних продуктах):

$ПРОДУКТ_i = ПРОДУКТ_{ан_i} * (LOC_{ан_i} / LOC_{оч_i})$

– прогнозовані витрати на розробку кожної функції: $ВИТРАТИ_i = (LOC_{оч_i} / ПРОДУКТ_i)$;

Важливим показником якості програмного коду є щільність дефектів. Щільність дефектів (defect density) – кількість дефектів на одиницю розміру продукту. Є різні різновиди цього показника [156].

Метрика Альбрехта. Метрика дефектів якості програмних засобів. Існують два різновиди цієї метрики:

- Заснована на рядках коду

- Заснована на функціональних покажчиках.

Метрика, заснована на функціональних покажчиках використовується для непрямої оцінки рівня якості процедурно-орієнтованих ПС. Перевага - легкість обчислення. Недолік – результати ґрунтуються на суб'єктивних даних. Використовуються не прямі, а непрямі вимірювання.

Формула метрики:

$DQ = \frac{кількість\ дефектів}{FP}$

$FP = F * (0{,}65 + 0{,}001 * \sum_{i=1}^{14} k_i)$

FP - функціональні показчики.

$F = \sum_{i=1}^{5} f_i$

F – загальна кількість функціональних покажчиків.

Значення коефіцієнтів регулювання складності k залежать від відповідей на запитання, що стосуються впливу певних факторів на виконання функцій ПЗ, на які відповідає людина. Саме через суб’єктивність значення даного виду метрики, була обрана метрика, заснована на рядках коду.

Дані на вхід:

- кількість виявлених помилок,

- кількість рядків коду для даної функціональності

Виконання розрахунків

DQ = Кількість помилок/Кількість рядків коду (DQ - щільність дефектів) [158].

SLOC = (кількість рядків коду), де

N – кількість структурних блоків у коді, Si – обсяг і- го блоку.

Обґрунтування метрики

За результатами вимірювань визначити, чи досягнута мета, чи вирішене завдання, чи отримані відповіді на питання підрахунку кількості дефектів.

Мета цієї діяльності - сформувати попередні оцінки, які дозволять:

- Пред'явити замовнику коректні вимоги за вартістю і витратами на розробку програмного продукту.

- Скласти план програмного проекту.

При виконанні оцінки можливі два варіанти використання LOC- і FP-даних:

- В якості оціночних змінних, що визначають розмір кожного елемента продукту.

- В якості метрик, зібраних за минулі проекти і входять до метричний базис фірми.

Універсальна метрика якості коду.

Дані на вхід:

- Кількість знайдених дефектів під час перегляду коду.

- Час.

Виконання розрахунків

CRD = Кількість дефектів/хвилина, де CRD - code review defects.

Обґрунтування метрики

Дана метрика вимірюється в ході процессу перегляду коду (code review). Review – діяльність, що проводиться для встановлення придатності, адекватності та ефективності, яку має досягнути об’єкт [156]. Для того, щоб переконатись, що код короткий, простий та відповідає діловій меті «переглядач» запускає секундомір і починає дивитись код. Кожен раз, коли він зустрічає в програмі помилку, додає до свого лічильника +1. Таким чином, розділивши показник лічильника на час, отримаємо метрику якості коду.

Alternative defect density (Альтернативна щільність помилок).

Дані на вхід:

- Кількість нових помилок після додання коду

- Кількість рядків доданого коду

Виконання розрахунків:

ADD = Кількість дефектів/ Кількість рядків доданого коду

Обґрунтування метрики:

Оцінка якості програмного продукту та ефективності доданого коду.

Дані на вихід:

Кількість помилок на 1000 рядків коду

Інтерпретація даних

Типовим вважається кількість дефектів 1-5 на 1000 рядків коду.

Defect density (для програміста). Щільність помилок для коду автора. Отримати дану метрику можна, наприклад, за допомогою MSR Tools.

Дані на вхід:

- Кількість помилок

- Кількість рядків коду певного автора

Виконання розрахунків

DDP = Кількість помилок/ Кількість рядків коду певного автора

Обґрунтування метрики

Якісний аналіз даної метрики допоможе визначити проблемні участки коду, ефективність роботи конкретного програміста, попередити програміста про потенційну небезпеку змін, розподілити ресурси тестування, оптимально обрати дату релізу.

Дані на вихід

Кількість помилок на 1000 рядків коду

Інтерпретація даних

Типовим вважається кількість дефектів 1-5 на 1000 рядків коду.

Статичний аналіз для оцінки, верифікації та аналізу вихідного коду набуває популярності завдяки широкому вибіру комерційних готових інструментів, які стають все більш зручними для розробників, недорогими та мають рішення, які добре інтегровані із середовищем розробки популярних мов програмування. Більш досконалі інструменти надають організації платформу, для якої використовуються різні інструменти, такі як аналізатори коду, плагіни і набори правил або бібліотеки для стандартів кодування та домовленостей, що можуть бути інтегровані та якості вихідного коду.

Процес оцінювання якості коду складається з набору найкращих практик, які включають перевірку відповідності коду, якість коду, накладення автоматизованого модульного тестування, який виграє від безперервного процесу інтеграції. Основна мета - оцінити рівень якості коду перед випуском. Рівень якості коду полягає не лише у складності коду, інструментів чи показників, але він включає автоматизовані методи управління якістю, забезпечуючи узгодженість практик розробника та відповідність запропонованим найкращим практикам. Керуючи процесами, очікується, що команди дадуть послідовні результати та оптимістично допоможуть зменшити кількість технічних проблем під час забезпечення гарантій якості. Це також полегшує відстеження якості ПЗ декількох випусків. Крім того, процес дасть переваги проектуванню, рефакторингу, враховуючи проблеми та пропозиції, проаналізовані інструментами. Важливими практиками, які доручаються проектним командам у нашому середовищі, є статичний аналіз, автоматизоване тестування модулів та рефакторинг.

Статичний аналіз допомагає аналізувати програмне забезпечення без фактичного запуску програми. Порівняно з динамічним аналізом, статичний аналіз аналізує точну лінію вихідного коду, тоді як динамічний аналіз аналізує програму під час її виконання; шляхом аналізу всіх видів тестів на об'єктні коди та виконувані файли. Таким чином, інструменти статичного аналізу допомагають:

– Виявити можливі помилки, які налагоджувач не може виявити;

– Надати інформацію про уразливість та виправлення, яка розглядається як гідне керівництво, особливо для нових інженерів в удосконаленні своїх навичок кодування;

– Дотримуватися стандарту кодування, який підтримує невідповідність між проектами.

Індекс зрілості програмного коду – це кортеж, який надає зворотний зв'язок користувачам щодо зрілості ПЗ [155].

$$\left(\frac{kINCL}{akINCL}, \frac{kTYPE}{akTYPE}\right) \quad (2.15)$$

де $\frac{kINCL}{akINCL}$ визначає знання розробників про домен, тобто розуміння бізнес логіки і її правильна реалізація, тобто на кожні 1000 рядків коду є велика кількість корисних команд, а отже немає непотрібних чи лишніх рядків коду,

$\frac{kTYPE}{akTYPE}$ визначає знання розробників самої мови програмування якщо показник високий, це свідчить про те, що код написано правильно, коротко та лаконічно.

Таблиця 2.9.

Змінна	Означення
$kINCL$	кількість команд на 1000 строк коду (KSLOC)
$akINCL$	середня кількість команд на 1000 коду із усього датасету
$kTYPE$	кількість визначених типів які зустрічаються на 1000 строк коду
$akTYPE$	середня кількість визначених типів на 1000 строк коду

Зрілими вважаються значення від 1,5 і більше. Наприклад, значення (2, 0.8) говорять про те, що код є зрілим з точки зору знання бізнес логіки, проте недостатньо зрілим з точки зору мови на якій він пишеться, а тому можна сказати, що він є недостатньо зрілим і потребує виправлення.

З іншого боку ***індекс зрілості коду*** можна визначити як метрику, яка забезпечує вказівку на стабільність ПП (на основі змін, що відбуваються для кожного випуску продукту) [2]. Індекс зрілості ПЗ обчислюється таким чином:

$$SMI = [MT - (Fa + Fc + Fd)] / MT \quad (2.16)$$

де MT – кількість модулів у поточному випуску;

Fc – кількість модулів у поточному випуску, які було змінено;

Fa – кількість доданих модулів у поточному випуску

Fd – кількість модулів попереднього випуску, які були видалені в поточному випуску.

Індекс зрілості використовується для оцінки готовності ПП до випуску, коли в існуючі програмні системи вносяться зміни, доповнення або видалення.

Якщо індекс зрілості наближається до 1, то продукт починає стабілізуватися. SMI може також використовуватися як метрика для планування заходів із супроводження ПЗ. Середній час випуску ПП може співвідноситися з індексом зрілості, і можуть бути розроблені емпіричні моделі для супроводження. Даний індекс ґрунтується на пакетах, а не на рівнях деталізації класу або методу.

Висновки по розділу 2

1. Розроблені методи оцінювання зрілості ПП, принциповою особливістю яких є те, що вони передбачають оцінювання не лише якихось складових зрілості ПП, а загальну оцінку вцілому.

2. Запропоновано математичну модель, яка базується на розробленій багатокритеріальній моделі оцінювання зрілості ПП. Застосування методів оцінювання та забезпечення зрілості ПП, які є формалізованим, забезпечує конструктивне рішення задачі досягнення зрілості ПП в процесі створення та супроводу ПП.

3. Показано можливість встановлення залежностей між показниками зрілості шляхом застосування кореляційно-регресійного аналізу, що дозволяє реалізувати процес управління змінами зрілості ПП. Визначені залежності враховуються для аналізу встановлення впливу коригувальних дій на рівень зрілості.

РОЗДІЛ 3
АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ ОЦІНЮВАННЯ ЗРІЛОСТІ ПРОГРАМНИХ ПРОДУКТІВ

Матеріали даного розділу присвячені опису та проектуванню програмного засобу, яка реалізує розглянуті методи. У роздалі представлено характеристику основних компонентів програмного засобу, їх призначення та особливостей програмної реалізації, а також основних прецендентів створеної в рамках дисертаційного дослідження реалізації програмного засобу.

3.1. Функціональні вимоги до засобу

Засіб оцінювання зрілості ПП призначений для отримання узагальненої оцінки зрілості ПП. В результаті використання даного продукту можна отримати оцінку або за певними показниками, або загальну оцінку зрілості. Вважається, що достатність рівня зрілості визначається шляхом порівняння отриманої загальної оцінки та розрахованих значень показників з відповідними аналогами, що приймаються за еталонний зразок. Аналогами обирають реально існуючі ПП того самого функціонального призначення, з такими ж основними параметрами, подібною структурою та умовами експлуатації.

Цілі створення засобу:

- автоматизація управління зрілістю ПП в процесі створення та супроводу ПП, використовуючи модель оцінювання зрілості ПП;
- зменшення часу на визначення оцінки зрілості ПП.

Дана модель дозволяє виміряти зрілість ПП, що базується на оцінці характеристик, підхарактеристик, які розкладаються на метрики.

Користувачем даної системи можуть бути експерти чи користувачі ПП, що зацікавлені у визначенні зрілості використовуваного ПП.

Вимоги до інтерфейсу користувача:

– інтерфейс користувача має бути зручним, простим та зрозумілим, щоб користувач міг швидко виконати бажані завдання.

– система має містити довідку для користувача та видавати повідомлення про некоректність його дій.

– має бути достатня комунікація між користувачем та системою.

Вхідними даними будуть дані, введені користувачем у систему, тип яких відповідає розширенню системи та відповідні запити на виконання певних функцій системою (зміна рангів атрибутів, редагування даних, видалення даних, виконання функцій розрахунку рівня зрілості) або програмний код відповідного ПП.

Вихідними даними будуть відповідно згенеровані системою оцінки підхарактеристик зрілості та інтегральна оцінка зрілості, представлені у текстовому та графічному вигляді.

На етапі проектування засобу оцінювання зрілості ПП необхідно визначити функціональні можливості програми та реакцію засобу на певні дії з боку користувача.

Функціональні можливості програмного азсобу:

- Система повинна надавати користувачу можливість вводити дані у визначеному форматі, а саме числові цілі значення, та забороняти введення даних іншого формату.
- Система повинна розраховувати вагові коефіцієнти атрибутів зрілості, виходячи з присвоєних даним атрибутам рангів.
- Система має надавати можливість зміни рангів атрибутів зрілості за бажанням користувача та використовувати введені значення для визначення вагових коефіцієнтів атрибутів зрілості.
- Система повинна обчислювати оцінку кожної підпідхарактеристиці зрілості на основі значень метрик, визначених для кожної підхарактеристики.
- Система повинна обчислювати інтегральну оцінку зрілості на основі значень підхарактеристик зрілості.
- Система повинна представляти користувачу можливість перегляду отриманих результатів, що для кожного проекту включають числові оцінки підхарактеристик.

- Система повинна зберігати розраховані дані на запит користувача. Дані мають зберігатися у .xls документ.
- Система повинна надавати можливість порівняння критеріїв якості різних проектів, для цього має бути можливість збереження розрахованих даних. Система повинна відображати результати у текстовому та графічному вигляді. Ця функція буде використовуватися, щоб дати можливість порівняти значення різних атрибутів.
- Система повинна надавати відомості про представлені метрики та підхарактеристики зрілості. Дана інформація має бути зрозумілою для користувача.
- Система повинна надавати можливість вибору та завантаження файлу для аналізу програмного коду;
- Система повинна надавати можливість відображення коду завантаженого файлу;
- Система повинна надавати можливість вибору метрик програмного коду для аналізу;
- Система повинна надавати можливість обчислення метрик за формулами;
- Система повинна надавати можливість обчислення математичного сподівання, коефіцієнтів асиметрії, коефіцієнтів кореляції, значущості залежності та визначення наявності чи відсутності залежності, визначення виду функції залежності однієї метрики від іншої;
- Система повинна надавати можливість вибіру лічильника, інтервалу, таймера;
- Система повинна надавати можливість відображення метрик за допомогою діаграм, а також виведення розподілів та графіків функцій залежності;
- Система повинна надавати можливість перегляд статистики.

Функціональні можливості розроблюваного ПП графічно можна представити, використовуючи діаграму варіантів використання (рис. 3.1), що відображає можливі варіанти використання засобу.

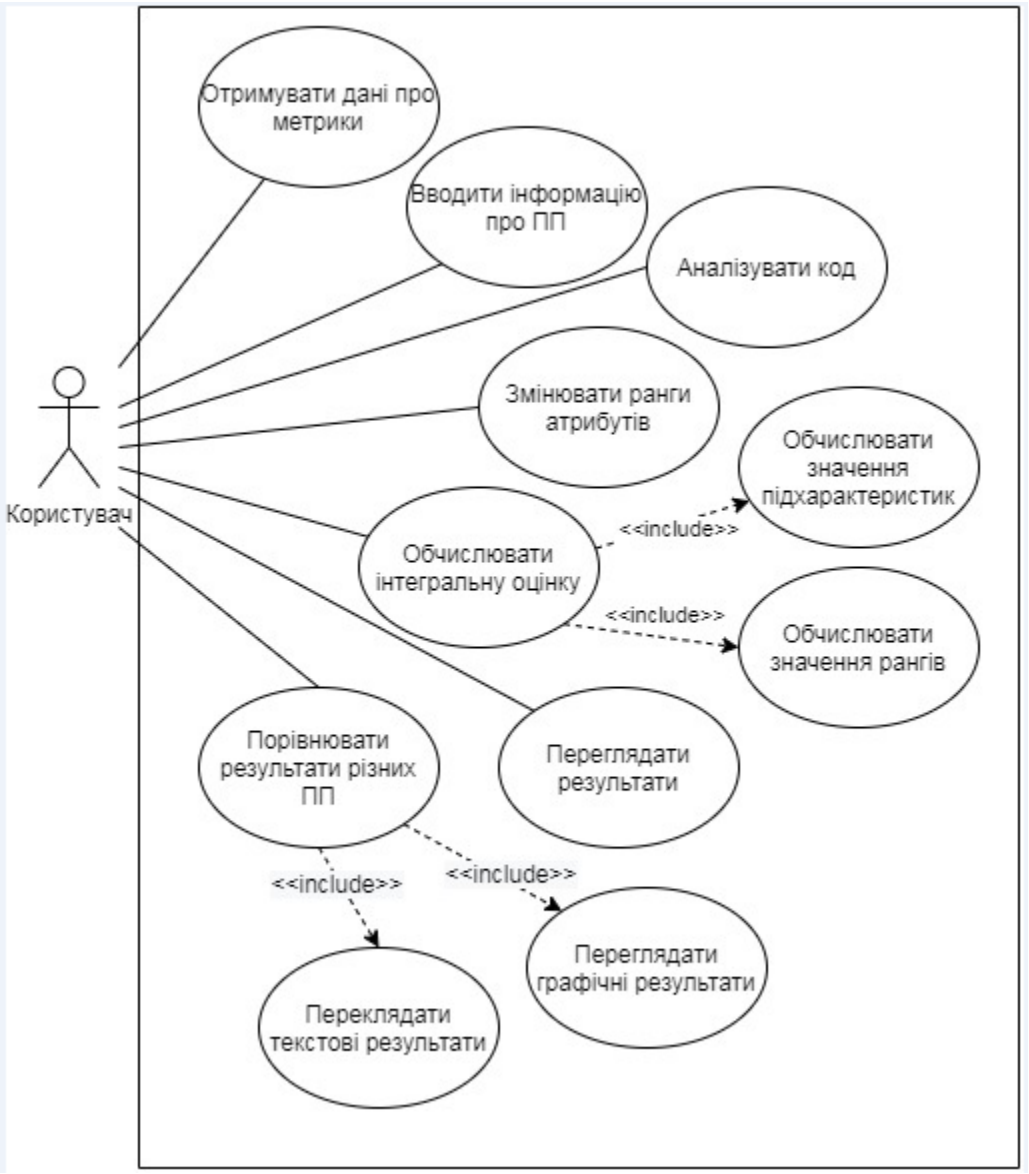

Рис. 3.1. Діаграма варіантів використання

Крім того, розроблюваній програмній системі повинні бути притаманними наступні властивості:

1) надійність – група властивостей (безвідмовність, відновлюваність), що обумовлює здатність ПЗ зберігати працездатність та перетворювати вихідні дані в очікуваний результат у заданих умовах за встановлений час. Для надійного функціонування розроблюваного засобу повинні бути дотримані наступні вимоги:

– забезпечення безперервного живлення для роботи програмного засобу;

– сумісність лише із ліцензійним ПЗ та використання для розробки засобу лише ліцензійного ПЗ;

– використання антивірусних систем.

2) зручність застосування – здатність ПЗ бути зрозумілим, придатним до вивчення і привабливим для користувача при його використанні в заданих умовах. Отже, розроблюваний програмний засіб повинен бути зрозуміли та зручним для користувача. Навігаційні елементи системи повинні бути розташовані в зручній для користувача формі, бути йому інтуїтивно зрозумілими, та відповідати загальноприйнятим правилам текстового та графічного відображення для забезпечення швидкого доступу до інформації. Інтерфейс повинен відповідати сучасним ергономічним вимогам, не має бути перевантажений графічними елементами і забезпечувати високу швидкість завантаження і відображення вкладень, а також всі сторінки користувальницького інтерфейсу повинні бути виконані в єдиному графічному дизайні, з однаковим розташуванням основних елементів управління та навігації. Переважає використання маніпулятора типу «Миша», а клавіатура використовується для введення\редагування даних.

3) супроводжуваність – здатність ПЗ до модифікації, що може містити у собі виправлення, поліпшення або адаптацію ПЗ до змін середовища, вимог або функціональних специфікацій;

4) переносимість – здатність ПЗ бути перенесеним з одного організаційного, апаратного або програмного середовища в інше.

3.2. Проектування програмного засобу

В даному розділі представлені основні компоненти, їх призначення та особливості, архітектуру системи (рис. 3.2); зображено компоненти складових програмної системи за допомогою діаграм компонентів, а також ER-діаграма схеми БД; діаграма класів та діаграма розгортання.

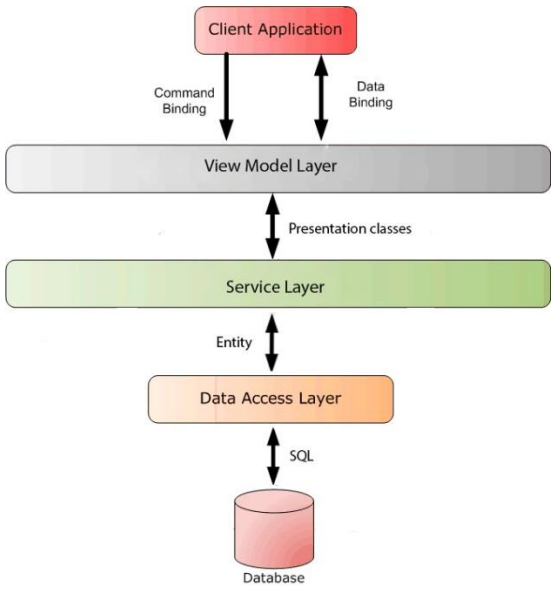

Рис. 3.2. Архітектура програмної системи

Система розділена на декілька компонентів, які представлені на діаграмі компонентів (рис. 3.3).

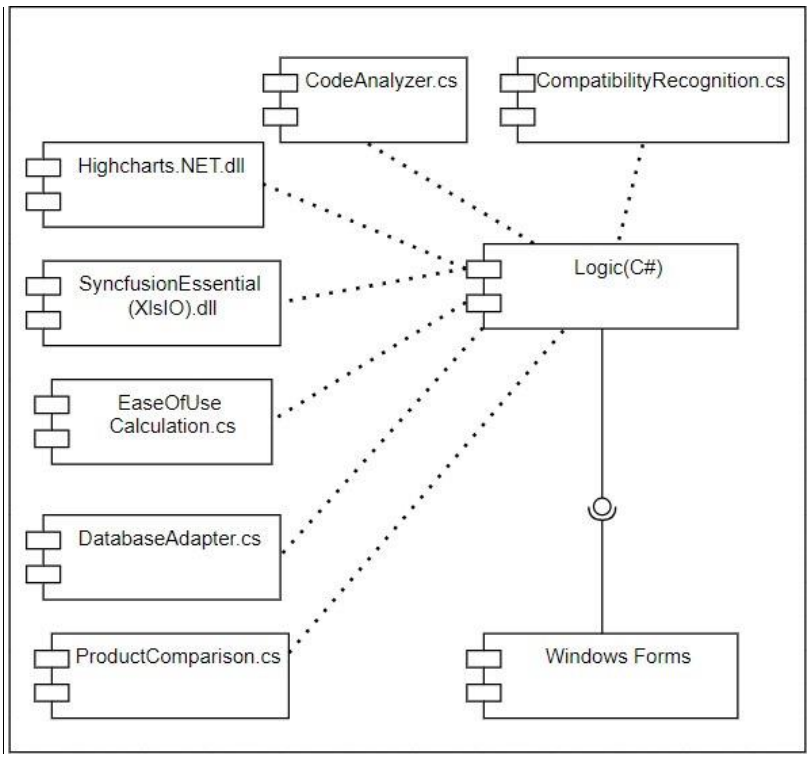

Рис. 3.3. Діаграма компонентів засобу

Щодо аналізатору коду, то він складається з наступних компонентів – завантаження даних, обчислень метрик, кореляційного аналізу, регресійного аналізу, побудови графіків, виведення та БД метрик. Архітектура складається з модуля введення даних – призначений для завантаження файлу з програмним кодом для аналізу. Також він призначений для вибору метрик для досліджень. Модуль попередніх обчислень призначений для обчислення відповідних метрик програмного коду, математичного сподівання, коефіцієнтів асиметрії. Модуль кореляційного аналізу призначений для розрахунків коефіцієнтів кореляції, значущості залежності та визначення наявності чи відсутності залежності. Модуль регресійного аналізу призначений для визначення виду функції залежності однієї метрики від іншої. Модуль побудови графіків призначений для побудови графіків для знайдених метрик та виведення розподілів та графіків функцій залежності. Модуль виведення призначений для виведення обчислених результатів та висновків за отриманими закономірностями. БД призначена для зберігання множин метрик та результатів вимірювань програмного коду. Аналізатор коду може функціонувати у двох режимах: вибір даних та аналіз. У режимі «вибір даних» є можливість вибрати метрики для дослідження. Аналізувати можна як одну, так і декілька метрик. У режимі «аналіз даних» можна побачити основні статистичні характеристики, коефіцієнти кореляції метрик. Також можна запустити графічний режим, у якому можна побачити графіків для знайдених метрик, функції розподілу метрик та лінії регресії. За потреби можна вивести коефіцієнти функції регресії та результати перевірок.

За допомогою діаграми послідовності відобразимо взаємодію об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

На рис. 3.4 представлена діаграма послідовності виконання для обчислення метрик коду, що описує дії вкладення «Аналізатор коду», а саме: відкриття вкладення, завантаження досліджуваного файлу з кодом та обчислення метрик з їх числовим відображенням.

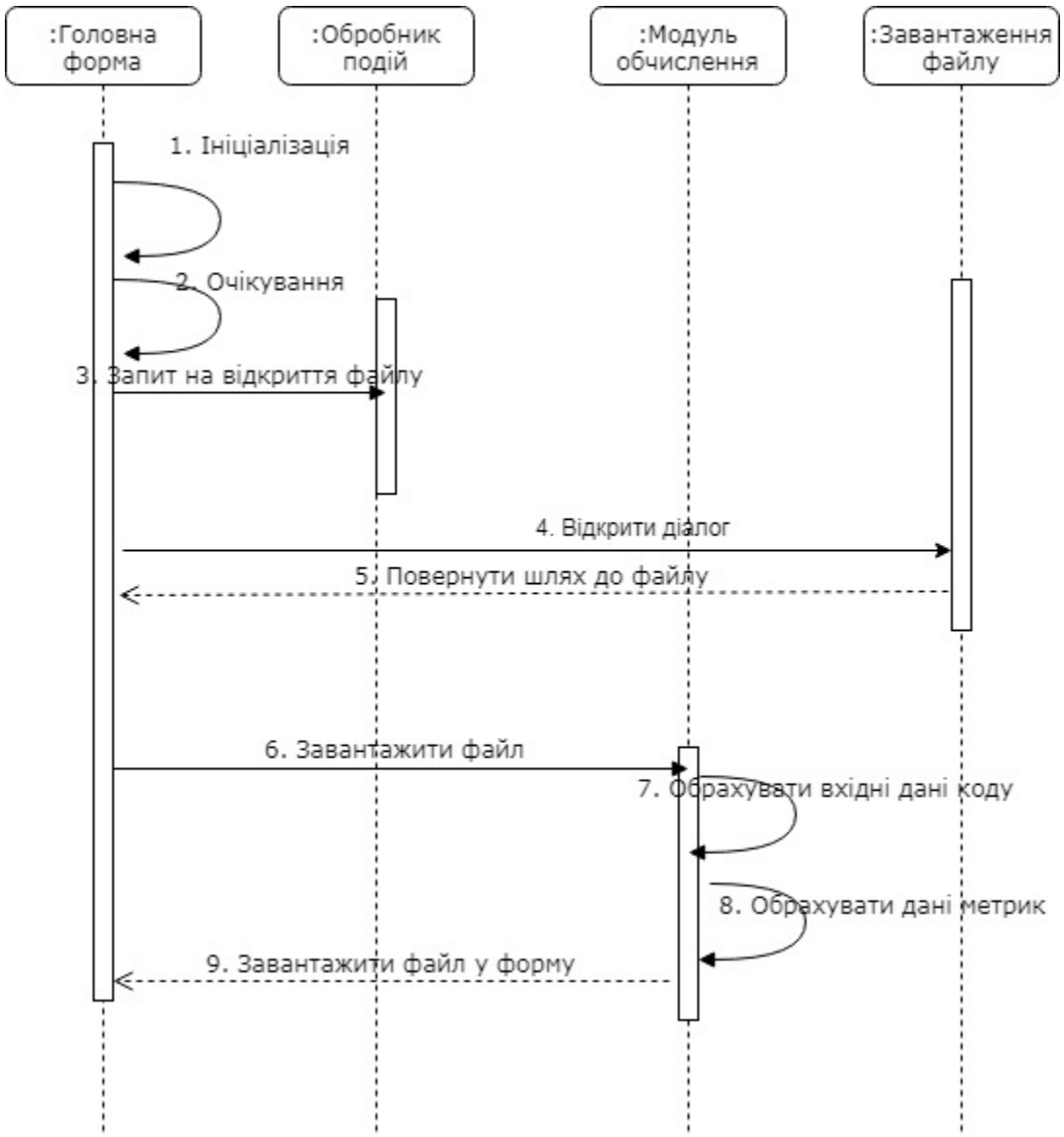

Рис. 3.4. Діаграма послідовності виконання для обчислення метрик коду

На рис. 3.5. представлена діаграма послідовності виконання для відображення діаграм, що описує дії у тому ж вкладенні, а саме: завантаження даних та їх графічне зображення у вигляді діаграм.

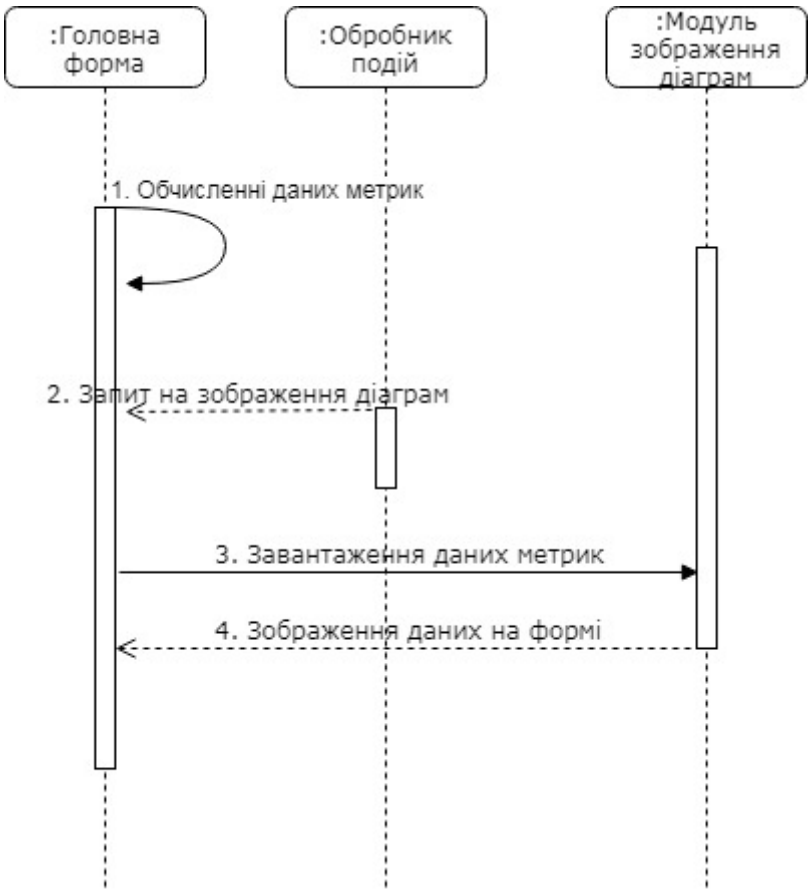

Рис. 3.5. Діаграма послідовності виконання для відображення діаграм

Засіб оцінювання зрілості ПП буде містити класи та методи. Найбільш якісно показати та зрозуміти структуру майже будь-якого ПЗ можна саме за допомого діаграми класів. Діаграма класів – діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язки між ними. У табл. 3.1 представлені методи для оцінювання зручності використання ПП.

Таблиця 3.1

Опис класу та методів розроблюваного засобу для оцінювання зручності використання ПП

Назва методу	**Опис**
Accessiability(data: Double[]):Double	метод обчислює метрики для підхарактеристики доступності та повертає значення суми цих даних з урахуванням ваги кожної метрики
Клас Formulas – реалізація формул для розрахунку	

AdivB(A:Double, B:Double): Double	реалізація формули для обрахунку метрики типу A/B, де A та B є вхідними значеннями
1minAdivB(A:Double, B:Double): Double	реалізація формули для обрахунку метрики типу 1-A/B, де A та B є вхідними значеннями
1minA(A:Double): Double	реалізація формули для обрахунку метрики типу 1-A, де A є вхідним значенням
Sum (values:Double[]): Double	реалізація формули для отримання суми метрик та підхарактеристик, вхідними значеннями є масив даних
Calculating – визначення значень підхарактеристики та інтегральної оцінки	
Appropriateness(data: Double[]):Double	метод обчислює метрики для підхарактеристики розпізнавання сумісності та повертає значення суми цих даних з урахуванням ваги кожної метрики
Learniability (data: Double[]):Double	метод обчислює метрики для підхарактеристики можливості навчання та повертає значення суми цих даних з урахуванням ваги кожної метрики
Operability(data: Double[]):Double	метод обчислює метрики для підхарактеристики операбельності та повертає значення суми цих даних з урахуванням ваги кожної метрики
ErrorProtection (data: Double[]):Double	метод обчислює метрики для підхарактеристики захисту від помилок користувача та повертає значення суми цих даних з урахуванням ваги кожної метрики
InterfaceAesthetics (data:Double[]):Double	метод обчислює метрики для підхарактеристики естетика користувацького інтерфейсу сумісності та повертає значення суми цих даних з урахуванням ваги метрик
Accessiability(data: Double[]):Double	метод обчислює метрики для підхарактеристики доступності та повертає значення суми цих даних з урахуванням ваги кожної метрики
Usability():Double	обчислює загальну оцінку ЗВ, що визначається як сума значень підхарактеристик ЗВ з урахуванням їх ваги
QualityModel – опис структури моделі зручності використання, значення необхідних атрибутів для отримання оцінки ЗВ, кожен з яких представлений у вигляді пари значень: назва - вага	
getWeight(): void	отримання значень ваги кожного з атрибутів
Клас Weight - розрахунок ваги кожного атрибуту ЗВ	
getRank(): Integer	отримання значень рангів атрибутів ЗВ

calculateWeight(rank: Integer,elenemtCount: Integer):Double	розрахунок ваги атрибутів, використовуючи їхні ранги
Клас Results - відображення збереження та імпорт отриманих результатів	
saveResults(): void	збереження отриманих результатів розрахунку у .xls файл
showResults(): string	відображення результатів обрахунку
exportData(): void	експорт збережених даних у .xls файл
getResults(): void	отримання результатів обрахунків
Клас CompareProducts – відображення результатів різних проектів у текстовому та графічному вигляді	
BuildDiagram(): void	побудова гістограми зі значеннями отриманих з використанням методу getResults(), стовпцями діаграми є значення підхарактеристик ЗВ
getResults(): void	отримання результатів для відображення у вигляді таблиці
importData(): void	зчитування результатів розрахунку проектів з .xls файлу
showResults(): void	виведення даних та представлення у текстовому вигляді
MainForm	
viewInfo(): void	відображення користувачу інформації про метрики

Відобразити відповідні класи, їх атрибути, методи та взаємозв’язки між ними можна за допомогою діаграми класів, що представлена на рис. 3.6

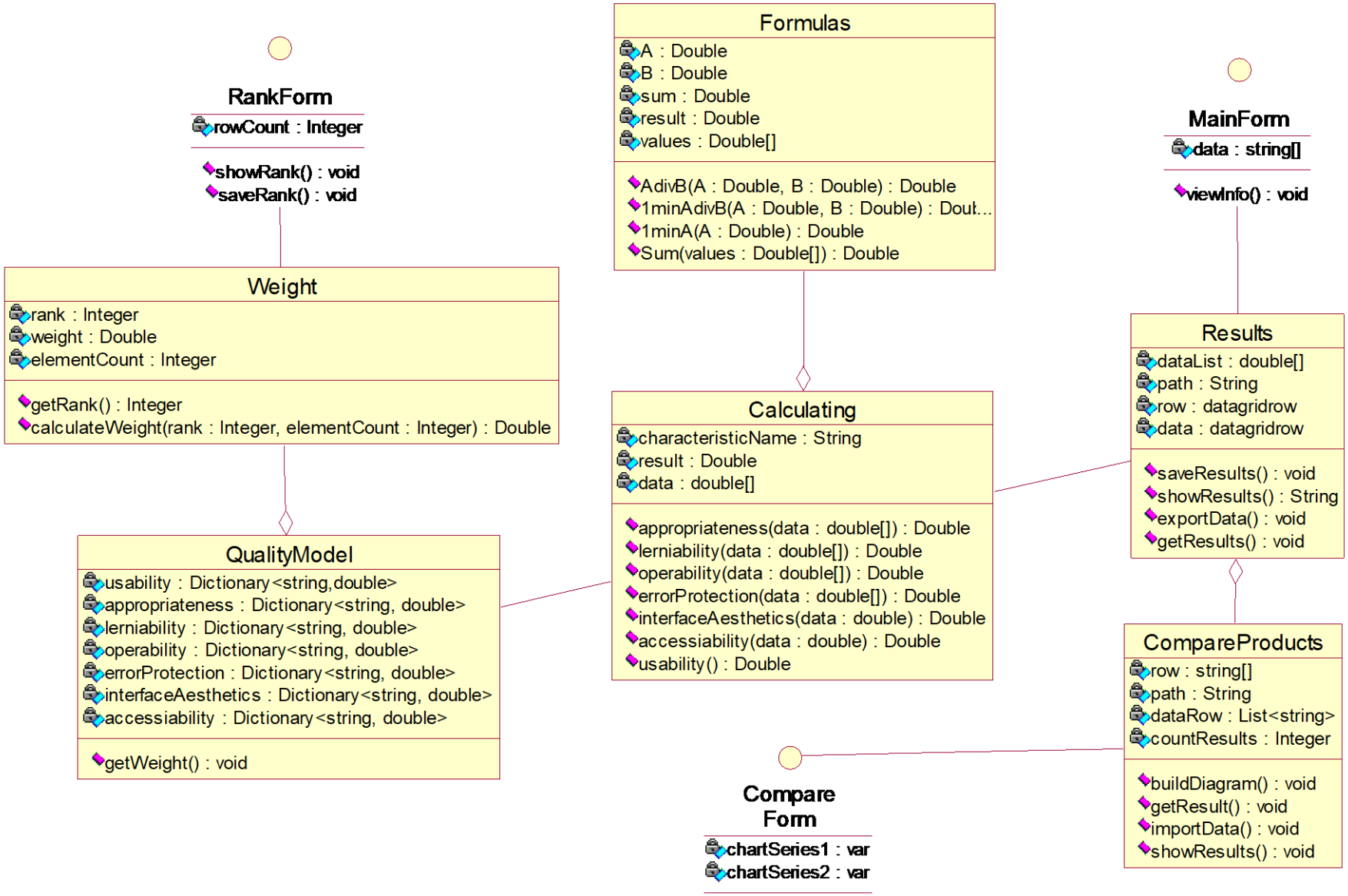

Рис. 3.6. Діаграма класів

Роботу програми можна представити за допомогою діаграми станів, що показана на рис. 3.7. Дана діаграма дозволяє показати поведінку системи та реакцію її на деякі події.

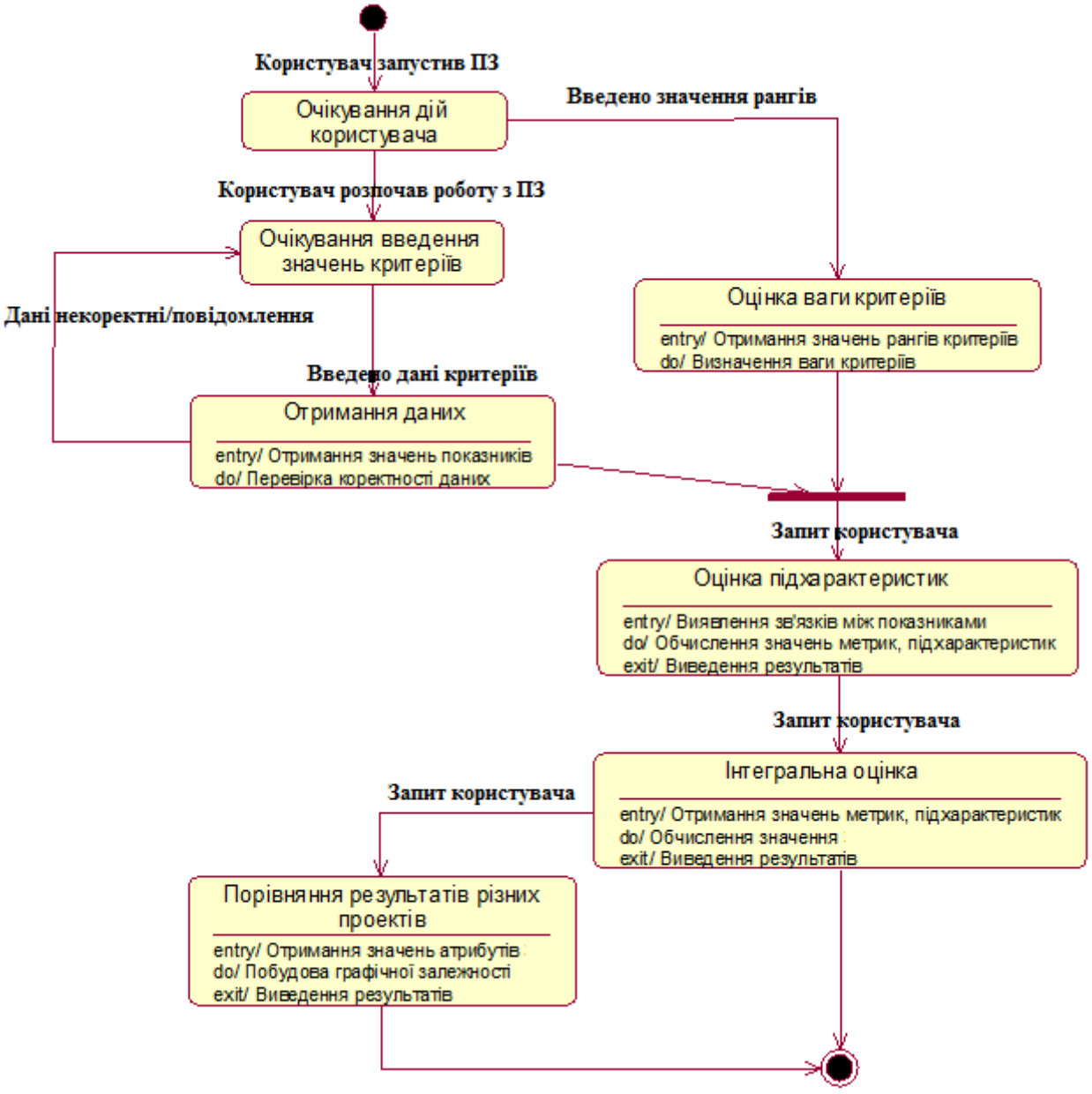

Рис. 3.7 Діаграма станів

Програмна система складається з трьох основних частин:

1. Web-сервер, що забезпечує збір даних. Використовувалась технологія ASP.NET MVC. Відповідні сервлети, на які перетворюються ASP-

документи перед використанням, розміщено у Web-контейнері IIS (Internet Information Server), що забезпечує обмін даними між сервлетом та користувачами, виконує функції створення програмного середовища для функціонуючого сервлета, ідентифікації та авторизації експертів, організації сесії для кожного з них. Статичні вихідні дані ASP-сторінки оформлено в HTML (Razor).

2. База даних, яка зберігає зібрану інформацію та налаштування системи. Управляється за допомогою MS SQL Server 2019.

3. Аналізатор, який забезпечує налаштування всього комплексу, вибірку даних з БД, їх аналіз у напівавтоматичному режимі і виведення результатів. Реалізований у вигляді віконної інтерактивної системи. При розробці застосовувалися мови C# і C++. Аналізатор містить такі підсистеми:

- підсистема завантаження та первинного аналізу даних. Використовуючи бібліотеки Highcharts .NET та Syncfusion Essential XlsIO, виконує розрахунки статистичних коефіцієнтів і будує двовимірну гістограму розподілу значень оцінок користувачів та рангів експертів, отриманих з БД. Результати виводяться у числовій і графічній формах відповідно. Для графічного відображення використано бібліотеку візуалізації на основі ILNumerics.;

- підсистема оцінки зрілості та узгодженості відповідей користувачів/експертів. Результати виконання відповідних функцій виводяться у формі звіту, у якому відображається значення коефіцієнта конкордації та розподіл експертних ранжувань за кластерами. Після цього підсистема виконує розрахунок вагових коефіцієнтів і поточного рівня зрілості, що аналізується. Результати в числовому вигляді відображаються у формі звіту.

– підсистема аналізу коду ПЗ, що дає можливість для автоматизації процесу обчислення метрик коду ПЗ та моніторингу еволюції ПЗ.

3.3. Інструментальні засоби для реалізації програмного засобу

Для створення програмної системи в роботі було використано середовище розробки Visual Studio 2017, платформу .NET Framework 4.7 та мову програмування C#.

Мінімальні системні вимоги:

- Operating System: Windows 7 Service Pack 1;
- CPU: Pentium 4 1.5 GHz or Athlon XP 1500+;
- RAM -512 MB;
- Hard drive - 1 GB of free space;
- Graphic device: compatible with DirectX 9.0c;
- Installed .NET Framework 4.7.

Рекомендовані системні вимоги:

- Operating system: Windows 10;
- CPU: Core 2 Duo or Athlon X2 2.4 GHz;
- Memory - 2 GB;
- Hard drive - 1 GB of free space;
- Graphic device: compatible with DirectX 10;
- Installed .NET Framework 4.7.

Висновки по розділу 3

1. Для реалізації запропонованих в дисертаційному дослідженні методів визначено функціональні вимоги та розроблені архітектура й прототип програмної системи, яка дозволяє автоматизувати оцінювання зрілості ПП в процесі створення та супроводу ПП. Архітектура поєднує в собі елементи клієнт-серверної архітектури.

2. Розглянуто варіанти використання програмної системи, її складові та їх компоненти, описано задіяні технологічні рішення. Система складається з трьох основних частин: web-сервер, що забезпечує збір даних; база даних, яка зберігає зібрану інформацію та налаштування системи;

аналізатор, що забезпечує управління системою, аналіз даних та виведення результатів.

3. Розроблено й описано підсистеми аналізатора, який входить в програмну систему, його призначення та особливості програмної реалізації.

РОЗДІЛ 4
ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАСОБУ ДЛЯ ОЦІНЮВАННЯ ЗРІЛОСТІ ПРОГРАМНИХ ПРОДУКТІВ

В попередньому розділі було розглянуто особливості архітектури та проектування програмного засобу для оцінювання зрілості ПП. Матеріали даного розділу присвячено оцінюванню працездатності розробленого засобу і, безпосередньо, самих запропонованих методів. Описується проведене емпіричне дослідження та отримані результати.

4.1. Керівництво для користувача по роботі з системою

Програмний засіб для оцінювання зрілості «SPMT» (Software Product Maturity Tool) призначена для знаходження оцінки зрілості відповідно до обраних методів.

Запуск програми в ОС сімейства Windows здійснюється запуском виконуючого файлу «SPMT».exe.

При запуску програми, користувачу представляється головне вікно програми, що представлений на рис. 4.1. Вверху вікна знаходиться головне меню, доступне для користувача. На лівій панелі відображені атрибути, що будуть використовуватися при оцінюванні ПП, вони представлені у вигляді структурного дерева. Знизу у вигляді таблиці будуть відображатися результати обрахунку атрибутів.

Користувачу доступне меню, що складається з трьох вкладень: вимірювання, аналізатора коду та довідки. Пункт довідка містить інформацію необхідну для користувача та описує метрики і атрибути, що використовуються при розрахунку зрілості ПП.

Пункт меню вимірювання (рис. 4.2) містить команди, що забезпечують такі функції:

- Вимірювання процесів – отримання інтегральної оцінки зрілості процесів.
- Вимірювання зрілості ПЗ – отримання інтегральної оцінки зрілості ПЗ.
- Порівняння продуктів – ця функція надає можливість порівняння отриманих оцінок підхарактеристик та інтегральної оцінки двох або більше програмних продуктів.

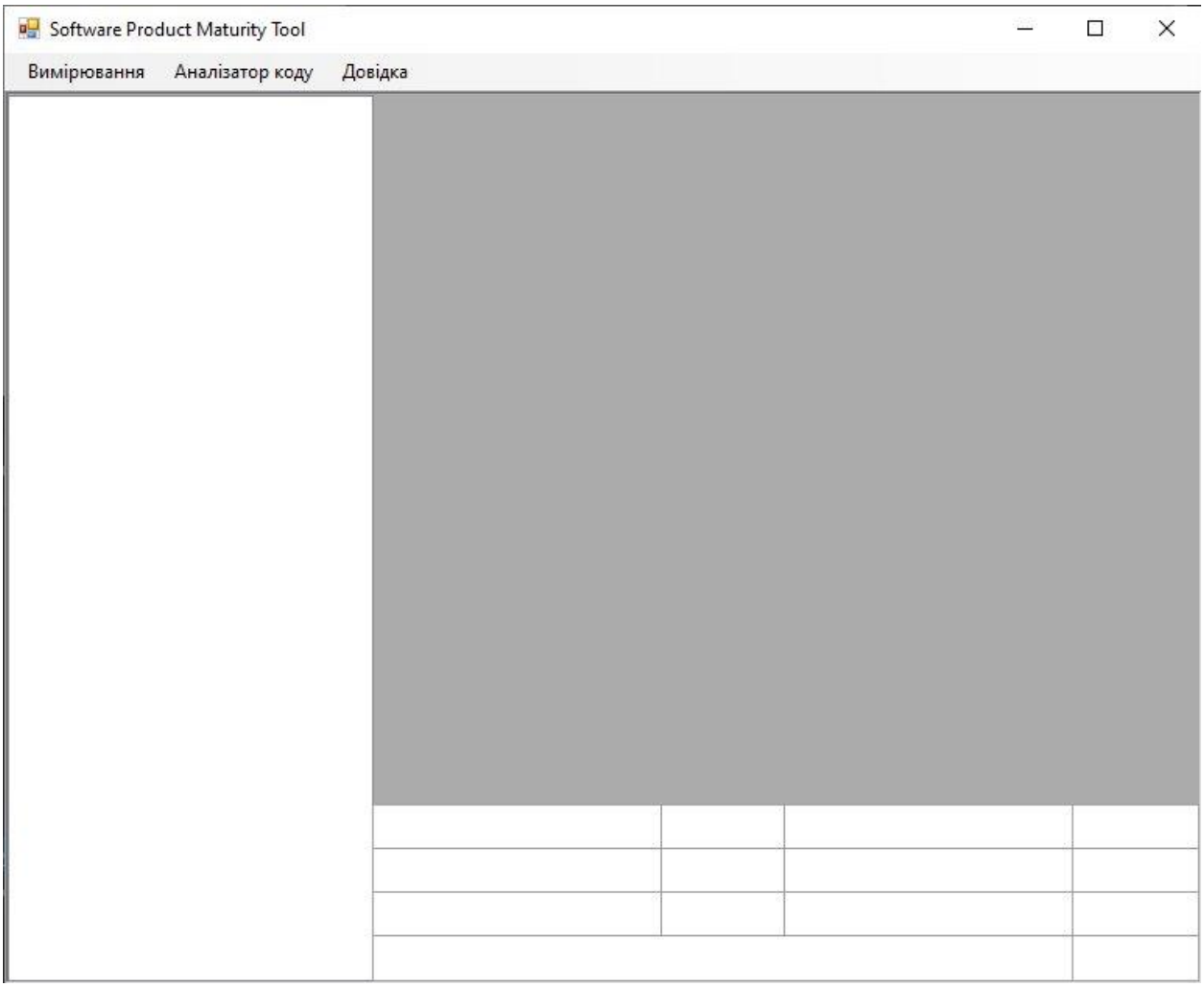

Рис. 4.1 Головне вікно програми

- Збереження даних для порівняння – при натисненні на даний пункт меню відбувається збереження результатів вимірювання певного ПП для подальшого порівняння.

- Відображення результату – ця функція дозволяє відображати результат у текстовому та графічному виді, розрахований для двох чи більше програмних продуктів.
- Зміна значень рангів – можливість зміни рангів кожного з атрибутів зрілості.

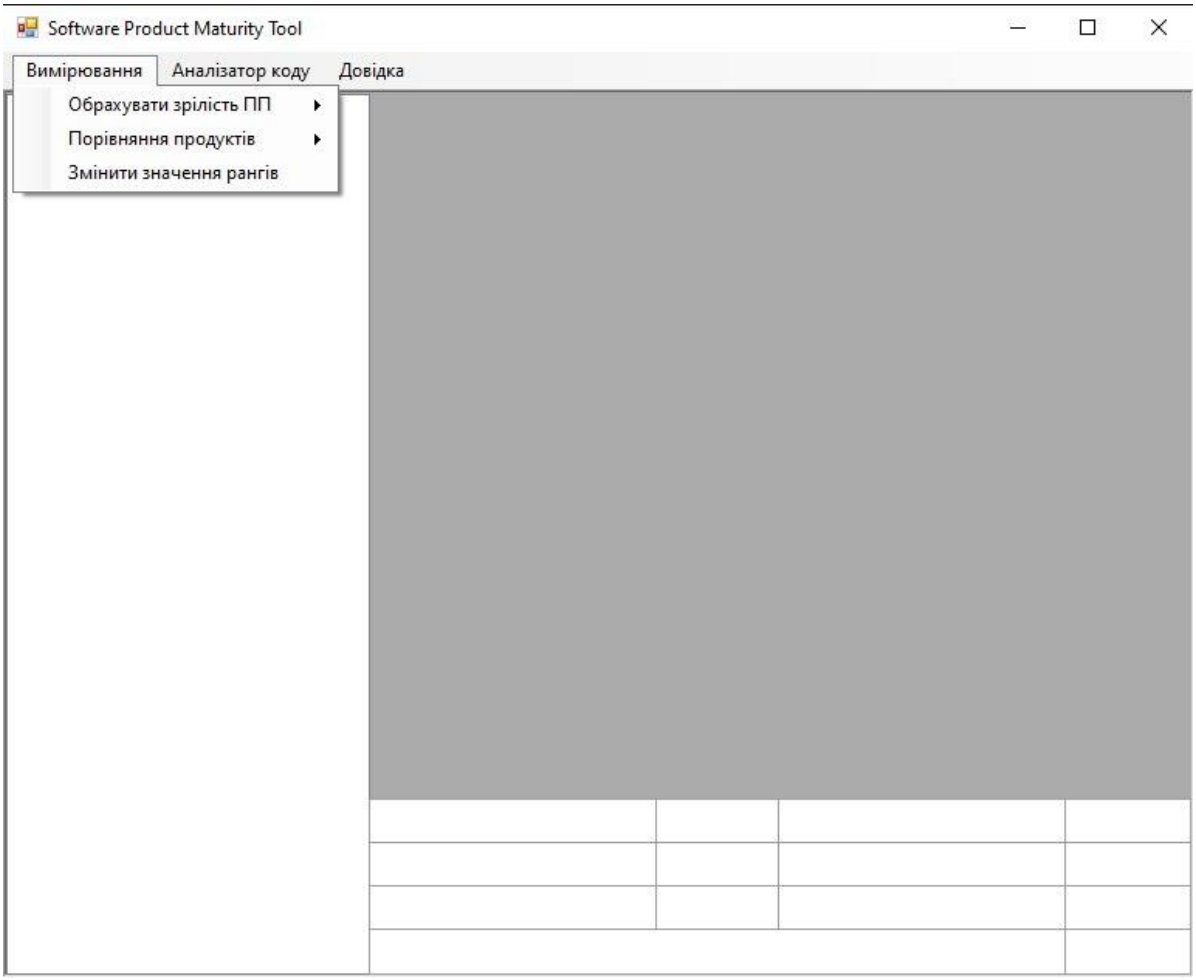

Рис. 4.2 Головне меню програми

При натисненні на певний елемент структурного дерева, що ієрархічно відображає усі атрибути, що використовують при оцінці зрілості ПП, буде відображена форма для введення відповідних даних. Приклад такої форми представлений на рис. 4.3.

У поля для вводу даної форми необхідно ввести дані про ПП, що мають бути визначеними для кожної метрики. При натисненні кнопки «Ok» відбувається обрахунок зазначеної підхарактеристики за визначеними метриками та відображення отриманих даних у таблиці результатів. При

натиснення кнопки «Cancel» відбувається закриття даної форми, а всі введені данні не використовуються.

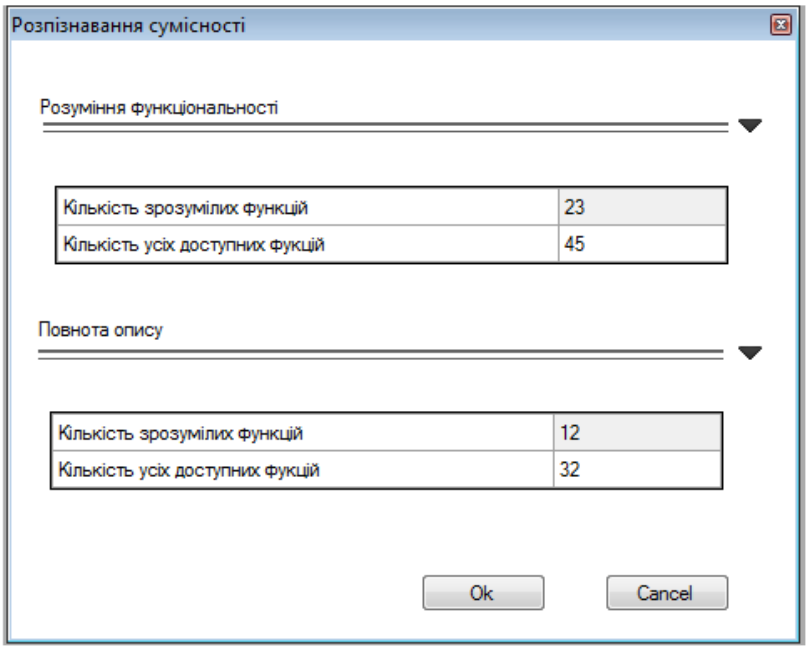

Рис. 4.3. Форма для введення даних

Користувач матиме можливість перегляду інформації про використовувані метрики, а саме пояснення даної метрики, формула розрахунку та значення, що будуть використовуватися. Для кожної метрики, при натисненні кнопки ▼, дана інформація буде відображена для користувача, як показано на рис. 4.4.

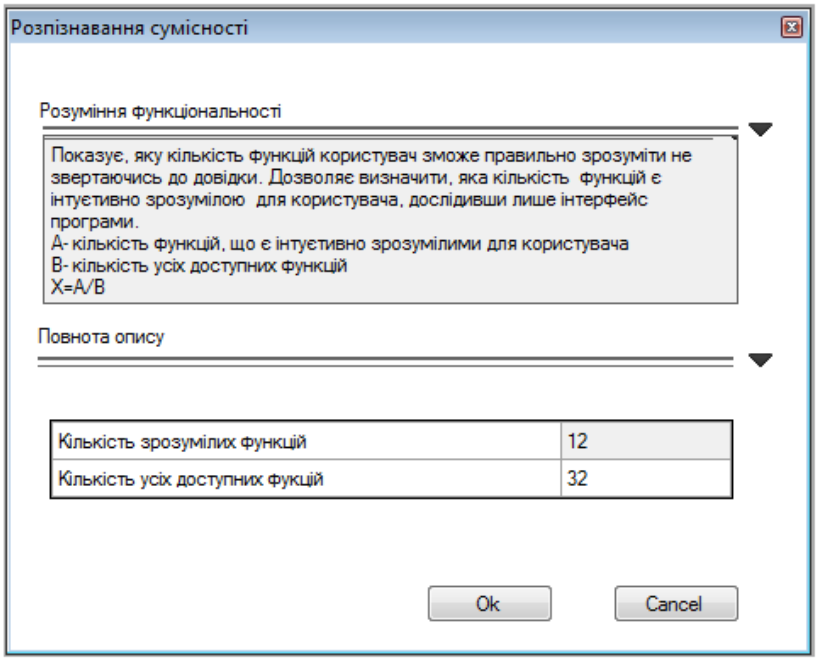

Рис. 3.4. Приклад відображення інформації про метрики

Після введення усіх даних, при натисненні команди «Обрахувати» в пункті головного меню «Вимірювання», у таблиці результатів буде виведена інтегральна оцінка, як показано на рис. 4.5.

Змінити значення рангів атрибутів можна при натисненні команди «Змінити значення рангів» в пункті головного меню «Вимірювання». У вікні зміни значень (рис. 4.6) буде відображена таблиця з назвами критеріїв та їх відповідними рангами, що можна змінити для визначених атрибутів за бажанням користувача. При натисненні кнопки «Ok» дані будуть збережені, а при натисненні кнопки «Відмінити» будуть використовуватися значення за замовчуванням.

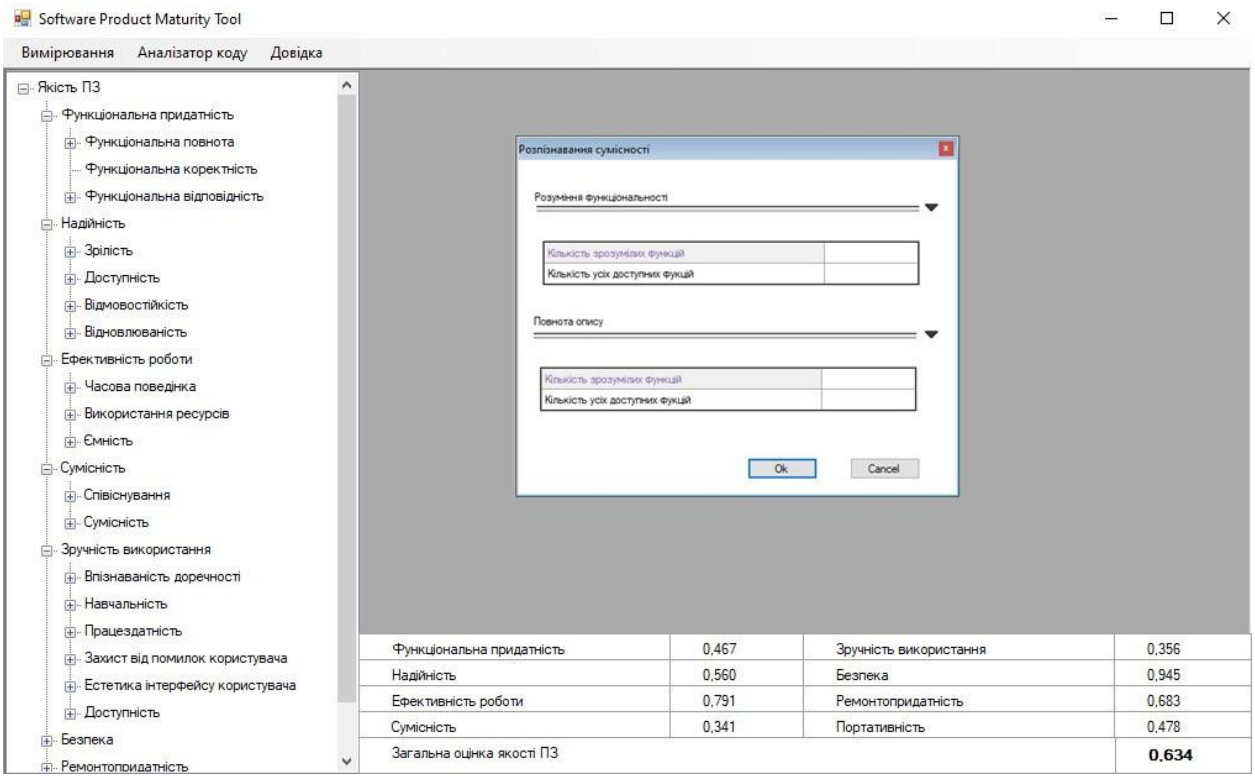

Рис. 4.5 Результат виконання команди «Обрахувати»

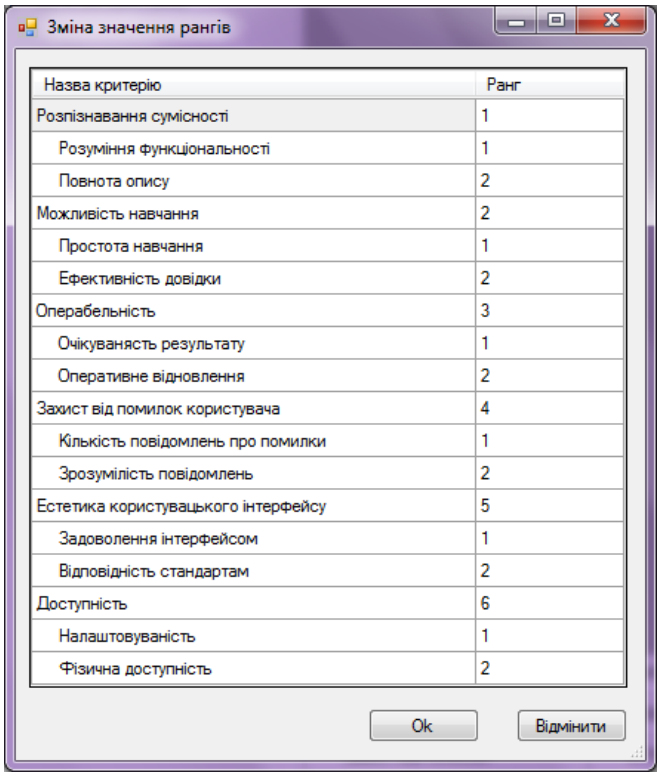

Рис. 4.6 Вікно зміни рангу атрибутів

При натисненні команди «Зберегти дані для порівняння», дані будуть збережені у документ Excel для подальшого їх використання. При натисненні команди «Відобразити результат» буде здійснюватися відкриття форми (рис. 4.7), що буде у текстовому та графічному вигляді відображати результати вимірювання різних проектів.

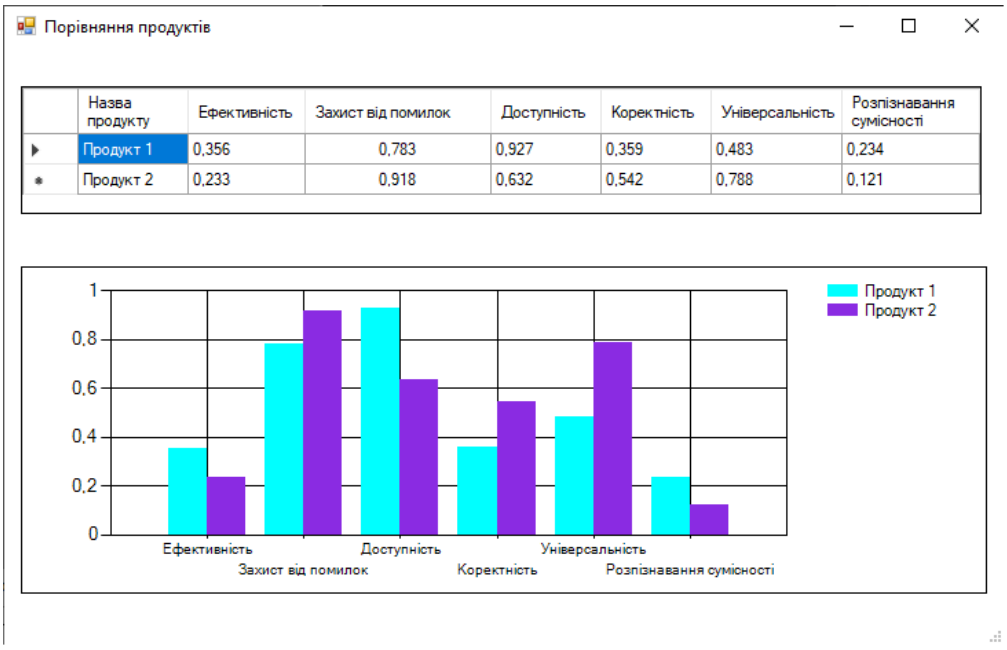

Рис. 4.7 Вікно порівняння результатів

При відкритті вкладення програми «Аналізатор коду» з’являються 2 вкладення: «Вибір данних», «Аналізувати».

При натисненні на вкладення «Вибір данних» відкривається вікно із наступними можливостями (рис. 4.8).

При натисненні на кнопку «Browse file…» відкривається вікно для пошуку відповідного файлу. Після того, як файл з кодом був знайдений, завантажуємо його. У верхній частині вікна відображається шлях до файлу, у лівому текстовому блоці – метрики, у правому – код. У режимі «Вибір даних» вибираються метрики, які необхідно дослідити. За потребою можна додати для аналізу як одну, так і кілька метрик.

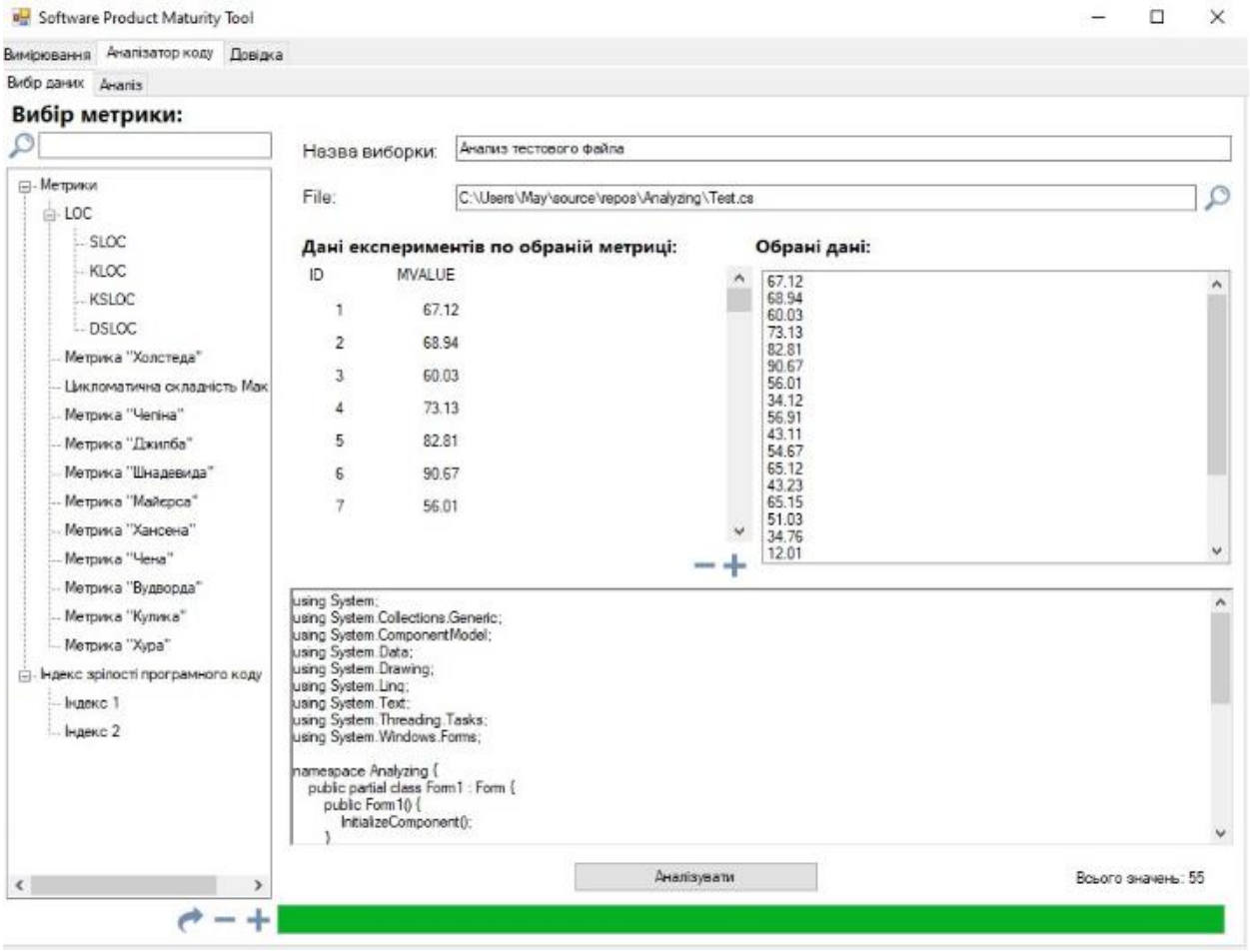

Рис. 4.8. Режим роботи «вибір даних»

У режимі «Аналіз даних» (рис. 4.9) можна побачити основні статистичні характеристики, коефіцієнти кореляції метрик. Доступно також запустити графічний режим, у якому можна побачити функції розподілу метрик та лінії регресії. За потреби можна вивести коефіцієнти функції регресії та результати перевірок.

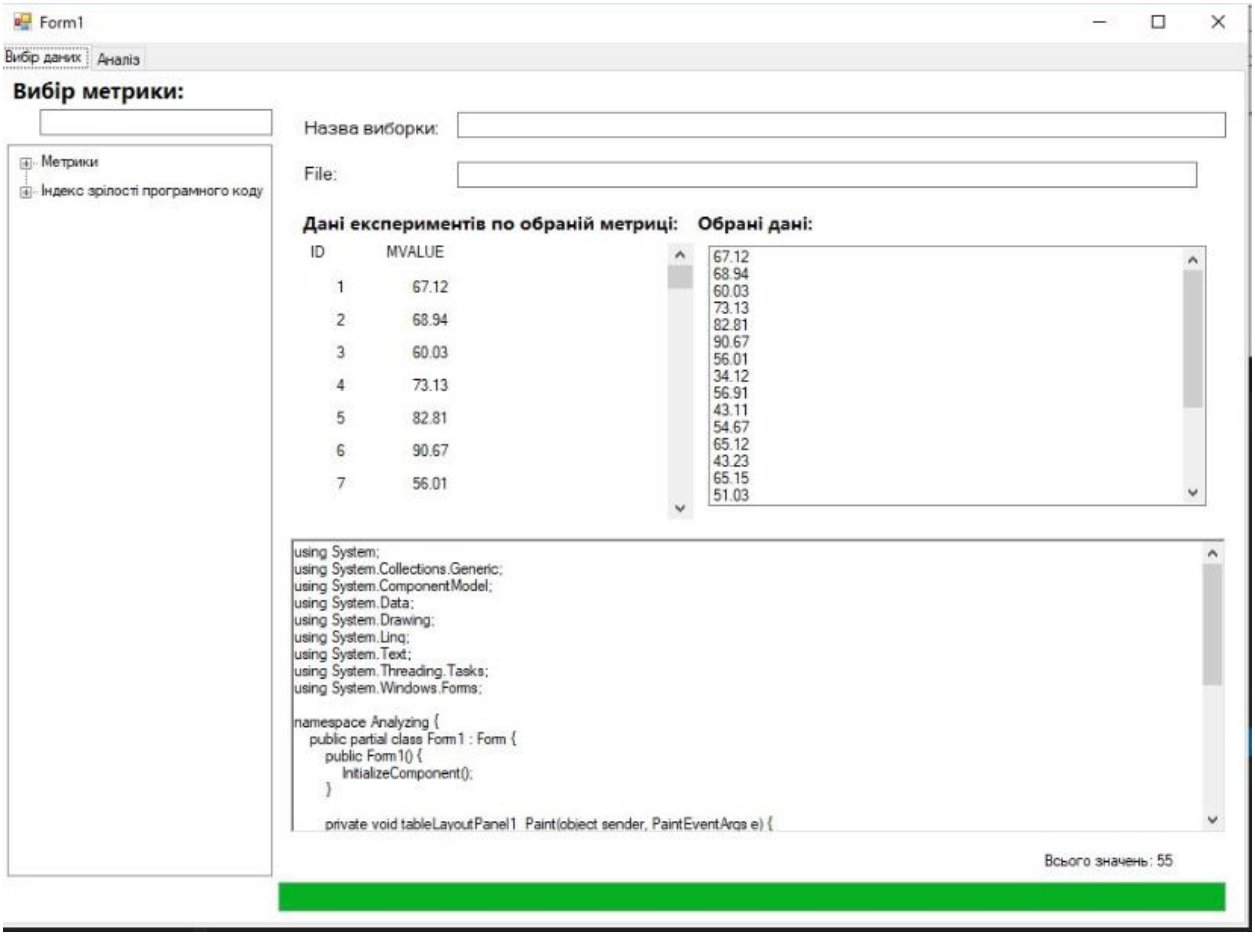

Рис. 4.9. Режим роботи «Аналіз даних»

Реалізований засіб для визначення залежностей між метриками, який набагато спрощує роботу щодо дослідження програмного коду. Тепер не потрібно додатково у статистичних середовищах писати програмний код для проведення необхідних розрахунків. Достатньо мати тільки набір метрик, які потрібно дослідити. Після занесення метрик до бази даних тепер достатньо натиснути кілька кнопок, щоб отримати результат як у числовому вигляді, так і у графічному вигляді.

4.2. Вибір критеріїв та практична оцінка зрілості ПП

Одним із важливих етапів оцінювання зрілості ПП є вибір моделі. У якості формальної моделі було взято ієрархічну модель, засновану на інтегрованій оцінці зважених показників зрілості ПП. Для формування поточних значень атрибутів, показників та метрик використовується опитування користувачів, які дають числову оцінку мір зрілості ПП, які визначаються відповідними метриками, а також ранжування експертів для визначення ваги критеріїв. Таким чином, вкрай важливим є визначення найбільш повного переліку критеріїв зрілості ПП, яким можуть дати оцінку користувачі та виставити ранги експерти. Цей список уточнюється експертами для кожного конкретного ПП.

Пропонується на другому рівні ієрархічної структури, якою представлена зрілості ПП, використати атрибути, що відповідають підхарактеристикам якості ПЗ в ISO/IEC 25010:2011 [3]. У подальшому атрибути декомпозуються на окремі показники.

У зв'язку великою кількістю характеристик, наведемо характеристику деяких з них:

Функціональна придатність (Functional suitability): група властивостей ПС, що обумовлює його здатність виконувати певний перелік функцій, які задовольняють потреби відповідно до призначення.

1. Функціональна повнота (Functional Completeness)
2. Функціональна коректність (Functional coverage)
3. Функціональна відповідність (Functional Appropriateness)

Надійність (Reliability): група властивостей, що обумовлює здатність ПС зберігати працездатність і перетворювати вихідні дані на результат за встановлений період часу, характер відмов якого є наслідком внутрішніх дефектів й умов його застосування.

4. Зрілість (Maturity)
5. Доступність (Availability)

6. Відмовостійкість (Fault tolerance)

7. Відновлюваність (Recoverability)

Ефективність роботи (Performance Efficiency): Група властивостей, що характеризується ступенем відповідності використовуваних ресурсів середовища до функціонування рівня якості (надійності), функціонування ПС за заданих умов застосування.

8. Часова характеристика (Time Behavior). Здатність програмного продукту при визначених умовах витрачати відповідну до виконуваної функції кількість часу.

9. Використання ресурсів (Resource Utilization)

10. Можливість (Capacity)

11. Співіснування (Co-existence)

12. Сумісність (Interoperability)

Зручність використання (Usability): сукупність властивостей ПС для передбачуваного кола користувачів й освоєння, що характеризують його простоту і адаптації до умов, що змінюються, експлуатації, стабільність роботи й підготовки даних, зрозумілість результатів, зручності внесення змін у програмну документацію й програми.

13. Впізнаваність доречності (Appropriateness recognisability)

14. Навчальність (Learnability)

15. Привабливість (Attractiveness). Це здатність програмного продукту бути привабливим для користувача. Цей субфактор більше пов’язаний із зовнішнім виглядом та відчуттям програмного продукту (використання кольору, природа графічного дизайну). Візуально приємний інтерфейс викликає більшу зацікавленість користувачів.

16. Приємність (Likeability). Чуттєве сприйняття, відчуття та відгуки користувача стосовно програмного продукту. Кожен користувач має власний рівень зацікавленості у продукті.

17. Гнучкість (Flexibility). Здатність інтерфейсу програмного продукту бути налаштованим згідно особистих вподобань користувача.

18. Мінімальна дія (Minimal Action). Здатність програмного продукту допомагати користувачам виконувати завдання за мінімальне число кроків.

19. Мінімальне навантаження пам'яті (Minimal Memory Load). Необхідність тримати в пам'яті користувача мінімальну кількість інформації для виконання певного завдання. Тут визначають три головні аспекти людської пам'яті: число елементів, що необхідно запам'ятати; часовий проміжок, на який треба запам'ятати ці елементи; подібність між запам'ятовуваними елементами.

20. Керування користувачем (User Guidance). Ступінь забезпечення інтерфейсом контекстно-залежної допомоги і значимого відгуку на помилки. Розглядається як складова частина дизайну інтерфейсу. Впливає на швидкість виконання завдань, кількість помилок, задоволеність користувача.

21. Узгодженість (Consistency). Узгодженість та гармонія частин або функцій одне з одним та в цілому. При цьому подібні дії користувача приводять до подібних результатів. Цей показник також спирається на загальні складові, компоновку (розташування), кольори, оформлення тощо. Розрізняють три види узгодженості інтерфейсу користувача:

- внутрішня узгодженість дизайну із самим собою;
- зовнішня узгодженість дизайну інтерфейсу з іншими дизайнами, знайомими користувачу;
- зовнішня метафорична або аналогова відповідність дизайну властивостям оточуючого світу.

22. Інформативність (Self-Descriptiveness). Здатність програмного продукту виражати своє призначення і надавати користувачу чітку підтримку під час функціонування.

23. Відгук (Feedback). Ефективність відповіді програмного продукту на дії користувача чи події.

24. Правильність (Accuracy). Здатність забезпечувати правильні результати або дії.

25. Стійкість до помилок (Fault-Tolerance). Здатність програмного продукту підтримувати визначений рівень продуктивності у випадках помилок програмного забезпечення або порушення встановленого інтерфейсу.

26. Зручність читання (Readability). Легкість розуміння візуального контенту. В основному стосується веб-сайтів.

27. Контрольованість (Controllability). Ступінь відчуття користувачем контролю над програмним продуктом.

28. Навігація (Navigability). Економічність переміщень користувача в додатку; здатність інтерфейсу фокусувати увагу на потрібному матеріалі та забезпечувати переміщення до цього матеріалу. Особливо важливо для веб-сайтів.

29. Простота (Simplicity). Чи усунені сторонні елементи з інтерфейсу користувача без втрати значущої інформації. Стосується трьох аспектів: зниження функціональності; зрозумілості і легкості використання програмного продукту. Розрізняють наступні види простоти:

- вербальна простота, яка передбачає використання зрозумілої, діючої, позитивної мови;
- візуальна простота, що досягається показом лише найбільш важливих об’єктів;
- простота завдання, що досягається, коли пов’язані завдання згруповані і в будь-який момент часу доступно лише декілька альтернатив;
- концептуальна простота вимагає використання природного відображення і семантики.

30. Знайомість (Familiarity). Ступінь представлення інтерфейсом знайомих користувачу елементів та зрозумілих дій. Базується на попередньому досвіді або вивченні того, як працювати з конкретним інтерфейсом.

31. Керівництво з використання. Зміст та ефективність керівництва з використання.

32. Демонстрації. Зміст демонстрацій.

33. Довідникова служба. Зміст та ефективність довідникової служби.

34. Конфіденційність (Confidentiality)

35. Цілісність (Integrity)

36. Підзвітність (Accountability)

37. Автентичність (Authenticity)

Супроводжуваність (Maintainability): група властивостей, що визначає зусилля, необхідні для виконання, пристосовність до діагностики відмов і наслідків внесення змін, модифікації й атестації ПС, що модифікується.

38. Модульність (Modularity)

39. Багаторазове використання (Reusability)

40. Випробуваність (Testability)

41. Модифікація (Modifiability)

42. Аналіз (Analyzability)

Портативність (Portability): група властивостей ПС, що забезпечує його пристосовність для перенесення з одного середовища функціонування в інші, зусилля для перенесення й адаптації ПС до нового середовища функціонування.

43. Адаптованість (Adaptability)

44. Встановлюваність (Installability)

Стосовно метрик, що розраховуються для кожного показника зрілості ПП, то в [3] наведено список зі 84 метрик, що обчислюються за формулами або простим підрахунком. Цей список є узагальненням (із деякими

доповненнями) більшості існуючих метрик. Так, наприклад, часова характеристика вимірюється трьома метриками: час виконання завдання, витрачений на помилки час і час вибору. Однією з метрик для вимірювання точності є частота помилок (чим ближче значення до нуля, тим краще) тощо. Метрики показників розпізнавання сумісності: ефективність довідникової служби, наприклад, вимірюється відношеннями ефективності (відсоток функціональних елементів, вірно використаних після перегляду довідки) та зрозумілості (відсоток функціональних елементів, які були вірно інтерпретовані після використання довідки).

Запропонований в другому розділі метод містить якісну оцінку атрибутів та показників експертами, а метрики слугують джерелом кількісної інформації.

Досліджуваний програмний продукт. У рамках дисертаційного дослідження було запропоновано метод управління зрілістю ПП, який ґрунтується на ітераційній оцінці поточного рівня зрілості ПП в процесі створення й супроводу ПП та формуванні оптимального варіанту забезпечення бажаного рівня зрілості ПП, який задається розробником на початку. Для перевірки методу та створеної програмної системи було обрано LibreOffice з відкритим вихідним кодом. LibreOffice - багатоплатформний, вільно розповсюджуваний офісний пакет з відкритим вихідним кодом, створений як відгалуження OpenOffice в 2011 році. Розробляється співтовариством з більш ніж 480 програмістів під егідою некомерційного фонду The Document Foundation за рахунок пожертвувань окремих осіб і організацій.

Офісний пакет містить в собі текстовий і табличний процесор, програму для підготовки і перегляду презентацій, векторний графічний редактор, систему управління базами даних і редактор формул. Основним форматом файлів, що використовується в додатку, є відкритий міжнародний формат OpenDocument (ODF, ISO / IEC 26300), але можлива робота і з

іншими популярними форматами, в тому числі Office Open XML, DOC, XLS, PPT, CDR.

Офісний пакет поширюється під громадської ліцензією MPL 2.0, тому може вільно встановлюватися і використовуватися в бюджетних і комерційних організаціях, а також на домашніх комп'ютерах і в навчальних закладах.

LibreOffice інтенсивно розвивається і з моменту своєї появи увібрав в себе безліч додаткових можливостей. Пакет представлений у версіях для різних операційних систем персональних комп'ютерів, а також мобільних пристроїв, а також в хмарної редакції LibreOffice Online.

Характеристики досліджуваної стабільної 7.03 версії наведено в таблиці 4.2.

Таблиця 4.2

Характеристики досліджуваного ПП

Виробник	The Document Foundation
Назва	LibreOffice
Тип	Офісний пакет
Дата випуску	25 січня 2011
ОС	Linux, Microsoft Windows, macOS, BSD, Android, iOS, Haiku
Апаратна платформа	IA-32, x86_64, ARM, PowerPC, MIPS и IBM System/390
Мова	C++
Остання версія	7.03 29 жовтня 2020 р.
Створювані формати файлів	OpenDocument і Office Open XML
Доступні формати файлів для читання	PDF и PowerPoint show[
Ліцензія	Mozilla Public License, version 2.0

4.2. Емпірична оцінка та варіанти забезпечення зрілості ПП

Повне практичне застосування методу управління зрілості ПП потребує виконання коригуючих дій зі зміни показників зрілості ПП на етапі

проектування ПП та проведення повторної оцінки зрілості ПП для підтвердження досягнення заданого розробником рівня зрілості ПП. При цьому необхідно, щоб організація-розробник мала відповідну статистику історичних даних із трудомісткості досягнення зрілості ПП. Оскільки доступ до таких даних обмежений, було виконано такі дії для апробації: обрано ПП із відкритим вихідним кодом та можливістю відслідковування змін у кожній наступній версії; розраховано рівень зрілості ПП на основі оцінок користувачів та експертних ранжувань; отримано варіанти зміни показників для покращення рівня зрілості ПП.

З метою отримання емпіричних даних було проведено опитування користувачів та експертів. У ролі користувачів виступила група спеціалісті ТОВ «Lime Systems», а також викладачі факультету кібербезпеки, комп'ютерної та програмної інженерії Національного авіаційного університету. Їм було запропоновано оцінити (у балах) міри зрілості ПП, які визначаються відповідними метриками, обраними для аналізу досліджуваного ПП. Функція введення користувацьких ранжувань показників, атрибутів та метрик зрілості ПП реалізована в програмній системі управління зрілості ПП, але її застосування не є обов'язковим, оскільки для визначення ваги критеріїв, як правило, використовують ранги експертів. Зауважимо, що в рамках дослідження не ставилася задача опитування кількості студентів достатньої для екстраполяції на всю генеральну сукупність з високою точністю. Це пояснюється тим, що основною метою є не виведення будь-яких статистичних закономірностей, а перевірка роботи методу управління зрілості ПП. Кількість опитаних дає змогу узагальнювати отримані результати з довірчою ймовірністю 85% та довірчим інтервалом (похибкою) в 10% за умови нормальності розподілу. Результати первинної обробки даних опитування показали, що розподіли оцінок в переважній більшості випадків не мають викидів та задовольняють подальшому аналізу.

Нижче подано коротку характеристику деяких отриманих метрик якості ПП для LibreOffice.

Для оцінювання даного програмного засобу використаємо метод SEER визначення експертами значимості атрибутів за допомогою безпосередньої оцінки, ранжування та парного порівняння. Доцільність використання того чи іншого метода визначається характером аналізованої інформації.

Перевага ранжування як метода експертного виміру – простота здійснення процедури оцінки. В процесі ранжування експерти мають встановити взаємо'язок вимог між усіма об'єктами, розглядаючи їх як єдину сукупність.

В даному методі експерту пред'являється весь набір об'єктів для оцінки та пропонується вказати найбільш пріоритетний серед них. Цей об'єкт виключається з подальшого розгляду, оскільки його ранг вважається визначеним, а далі обирається найбільш пріоритетний об'єкт з решти. В результаті порівняння всіх об'єктів створюється упорядкована послідовність $a1 > a2 > ... > aN$, де об'єкт з першим номером є найбільш пріоритетний з усіх об'єктів, другий є менш пріоритетним за перший, але важливіший за інші.

При неможливості розрізнити за важливістю два або декілька атрибутів використовується один і той самий ранг, значення якого є середнє суми місць, поділених цими атрибутами.

Вагу критеріїв можна визначити, використовуючи ранги критеріїв вимог, що були визначені експертами з використанням методу ранжування. Припустимо, що думки експертів є узгодженими, оскільки детальне дослідження даного питання виходить за межі роботи.

Виходячи з підсумкового ранжирування, для оцінки значимості кожного критерію нефункціональних вимог можна використати шкалу Фішберна [190]:

$$p_i = \frac{2(n-i+1)}{n(n+1)},$$

де p_i – коефіцієнт значимості i-го критерію;

i – ранг поточного критерію в підсумковому ранжуванні;

n – кількість критеріїв.

Серед запропонованих для оцінки метрик зрілості ПП більшість розподілів характеризується ненормальним законом, значенням медіани $Me = 8$ та дисперсією $0{,}37 < \sigma^2 < 2{,}8$, а отже, стандартним відхиленням $0{,}61 < \sigma < 1{,}67$. Таким чином похибка репрезентативності (середнього значення) наступна $0{,}085 < m = \frac{\sigma}{\sqrt{n}} < 0{,}232$, де n=50 – кількість елементів вибірки.

Наступним результатом дослідження було отримання експертних ранжувань для подальшого розрахунку ваги метрик, показників та атрибутів зрілості. У якості експертів виступили 7 фахівців ПрО. Коефіцієнт конкордації для виставлених рангів атрибутів становить W=0,88 та є значимим, що свідчить про сильну ступінь узгодженості за шкалою якісної оцінки щільності зв'язку між коефіцієнтом кореляції Пірсона і кореляційним відношенням. Отримані з підсумкового ранжування відповідні ваги критеріїв зрілості наведені в табл. 4.3.

Таблиця 4.3

Вага атрибутів та показників LibreOffice

Атрибут	Ранг	Вага	Показник	Ранг	Вага
Функціональна повнота (Functional Completeness)	3	0,19048	Функціональне покриття (Functional coverage)	3	0,14545
Функціональна коректність (Functional coverage)	2	0,12569	Функціональна коректність (Functional coverage)	4	0,09091
Функціональна відповідність (Functional Appropriateness)	5	0,04762	Функціональна доцільність цілі використання (Functional appropriateness of usage objective)	2	0,16364
			Функціональна доцільність системи (Functional	1	0,5

			appropriateness of the system)		
Зрілість (Maturity)	4	0,14286	Виправлення несправностей (Fault correction)	3	0,16667
			Середній час між відмовою (Mean time between failure, MTBF)	1	0,5
			Рівень відмов (Failure rate)	2	0,33333
			Покриття тесту (Test coverage)	4	0,13333
Доступність (Availability)	1	0,28571	Доступність системи (System availability)	1	0,16667
			Середній час простою (Mean down time)	2	0,33333
Відмовостійкість (Fault tolerance)	2	0,2381	Уникнення відмов (Failure avoidance)	3	0,16667
			Надлишковість компонентів (Redundancy of components)	1	0,5
			Середній час повідомлення про несправність (Mean fault notification time)	2	0,33333
Відновлюваність (Recoverability)	3	0,19048	Середній час відновлення (Mean recovery time)	2	0,17778
			Повнота резервного копіювання даних (Backup data completeness)	1	0,2
Часова поведінка (Time behavior)	9	0,14286	Середній час реакції (Mean response time)	2	0,26667
			Адекватність часу відгуку (Response time adequacy)	1	0,33333
			Середній час обороту (Mean turnaround time)	3	0,2
			Адекватність часу виконання (Turnaround time adequacy)	4	0,13333
			Середня пропускна здатність (Mean throughput)	5	0,06667
Використання ресурсів (Resource Utilization)	5	0,09524	Середнє використання процесора (Mean processor utilization)	2	0,26667
			Середнє використання пам’яті (Mean memory utilization)	1	0,33333
			Середнє використання пристроїв вводу-виводу (Mean I/O devices	3	0,2

			utilization)		
			Використання смуги пропускання (Bandwidth utilization)	4	0,13333
Ємність (Capacity)	8	0,19048	Ємність обробки транзакцій (Transaction processing capacity)	1	0,5
			Ємність доступу користувача (User access capacity)	2	0,33333
			Доступ користувачів збільшує адекватність (User access increase adequacy)	3	0,13333
Співіснування (Co-existence)	7	0,14286	Співіснування з іншими продуктами (Co-existence with other products)	1	0,5
Сумісність (Interoperability)	6	0,04678	Обмінність форматів даних (Data formats exchangeability)	1	0,18182
			Достатність протоколу обміну даними (Data exchange protocol sufficiency)	2	0,01818
			Адекватність зовнішнього інтерфейсу (External interface adequacy)	3	0,12727
Впізнаваність доречності (Appropriateness recognisability)	4	0,2762	Повнота опису (Description completeness)	2	0,16364
			Демонстраційне висвітлення (Demonstration coverage)	1	0,03636
			Точка входу самоописуваності (Entry point self-descriptiveness)	3	0,05455
Навчальність (Learnability)	5	0,14286	Повнота керівництва користувача (User guidance completeness)	3	0,14545
			За замовчуванням поля введення (Entry fields defaults)	4	0,12727
			Зрозумілість повідомлень про помилки (Error messages understandability)	2	0,10909
			Пояснювальний інтерфейс користувача (Self-explanatory user interface)	1	0,01818
Працездатність (Operability)	8		Послідовність роботи (Operational consistency)	4	0,13333
			Ясність повідомлення (Message clarity)	7	0,06667

		0,28571	Функціональна настроюваність (Functional customizability)	6	0,08889
			Настроюваність інтерфейсу користувача (User interface customizability)	5	0,11111
			Можливість моніторингу (Monitoring capability)	3	0,15556
			Скасувати можливість (Undo capability)	2	0,17778
			Зрозуміла категоризація інформації (Understandable categorization of information)	1	0,2
			Послідовність зовнішнього вигляду (Appearance consistency)	9	0,02222
			Підтримка пристрою введення (Input device suppor)	8	0,04444
Захист від помилок користувача (User error protection)	4	0,19048	Уникнення помилки роботи користувача (Avoidance of user operation error)	1	0,5
			Виправлення помилок входу користувача (User entry error correction)	3	0,16667
			Відновлення помилок користувача (User error recoverability)	2	0,33333
Естетика інтерфейсу користувача (User interface aesthetics)	8	0,14286	Естетика зовнішнього вигляду інтерфейсів користувача (Appearance aesthetics of user interfaces)	1	0,5
Доступність (Accessibility)	6	0,0456	Доступність для користувачів з обмеженими можливостями (Accessibility for users with disabilities)	1	0,5
			Адекватність підтримуваних мов (Supported languages adequacy)	2	0,5
Конфіденційність (Confidentiality)	7	0,2457	Керування доступом (Access controllability)	1	0,5
			Правильність шифрування даних (Data encryption correctness)	3	0,16667
			Сила криптографічних алгоритмів (Strength of	2	0,33333

			cryptographic algorithms)		
Цілісність (Integrity)	9	0,19346	Цілісність даних (Data integrity)	3	0,16667
			Запобігання корупції внутрішніх даних (Internal data corruption prevention)	1	0,5
			Запобігання переповненню буфера (Buffer overflow prevention)	2	0,33333
Аналіз (Analyzability)	8	0,14548	Повнота системного журналу (System log completeness)	1	0,5
			Ефективність функції діагностики (Diagnosis function effectiveness)	3	0,16667
			Достатність функції діагностики (Diagnosis function sufficiency)	2	0,33333
Випробуваність (Testability)	3	0,2381	Перевірка повноти функції (Test function completeness)	1	0,5
			Автономне тестування (Autonomous testability)	3	0,16667
			Перевірка перезавантажуваності (Test restartability)	2	0,33333
Адаптованість (Adaptability)	4	0,04762	Апаратна екологічна пристосованість (Hardware environmental adaptability)	1	0,5
			Адаптованість середовища системного програмного забезпечення (System software environmental adaptability)	3	0,16667
			Адаптованість до робочого середовища (Operational environment adaptability)	2	0,33333
Встановлюваність (Installability)	4	0,2561	Ефективність часу установки (Installation time efficiency)	1	0,16667
			Простота установки (Ease of installation)	2	0,33333
Замінність (Replaceability)	5	0,04361	Схожість використання (Usage similarity)	4	0,03636
			Еквівалентність якості продукції (Product quality equivalence)	1	0,18182
			Функціональна інклюзивність (Functional inclusiveness)	2	0,05455

			Можливість повторного використання / імпорту даних (Data reusability/import capability)	3	0,09091

Маючи оцінки користувачів та вагу критеріїв якості, можна розрахувати за формулою поточний рівень зрілості Q LibreOffice. Згідно отриманих даних $Q = 0{,}86$. Перед вирішенням задачі забезпечення бажаного рівня зрілості було встановлено, використовуючи підсистему визначення залежностей між показниками зрілості, наявність зв'язків між показниками шляхом виявлення емпіричних залежностей (рис. 4.10). Тіснота зв'язку між показниками визначалася за шкалою якісної оцінка щільності зв'язку між коефіцієнтом кореляції Пірсона і кореляційним відношенням: до уваги бралися лише ті пари показників, що мають сильний зв'язок. Визначені зв'язки дозволяють оцінити вплив зміни одного з показників на інший. Результати дослідження показують, що всі залежності є прямо пропорційними, тобто при збільшенні (зменшенні) незалежного показника, значення залежного також збільшується (зменшується) (табл.4.4).

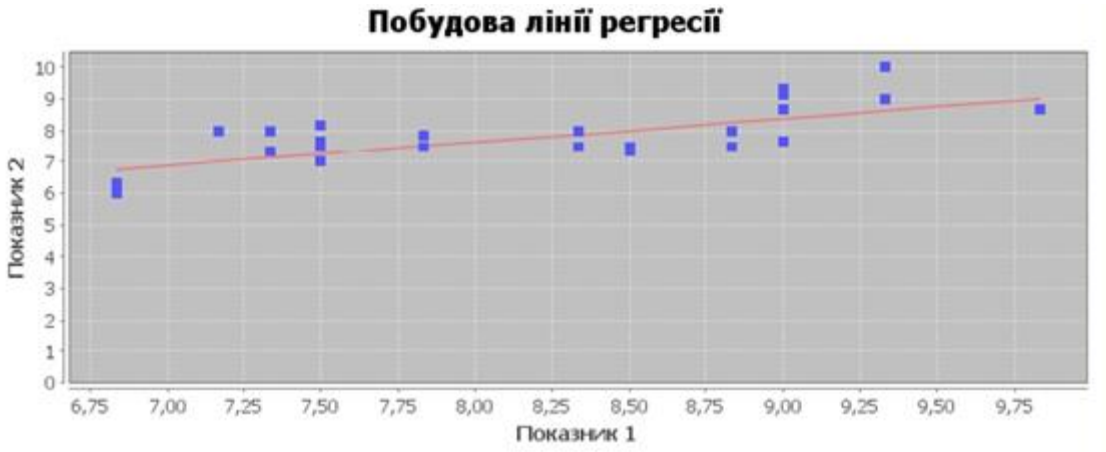

Рис. 4.10. Дослідження форми залежності між показниками якості ПЗ

Таблиця 4.4

Емпіричні залежності між показниками якості LibreOffice

Показник 1, (x)	**Показник 2, (y)**	**Коефіцієнт рангової кореляції,** r	**Формула залежності**	**Коефіцієнт детермінації,** R^2
Адекватність зовнішнього інтерфейсу	Пояснювальний інтерфейс користувача	0,71	$y = 6161{,}2 - 4301{,}2 \cdot x + +1177{,}8 \cdot x^2 - 158{,}5 \cdot x^3 + +10{,}5 \cdot x^4 - 0{,}3 \cdot x^5$	0,78
Повнота опису	Повнота керівництва користувача	0,8	$y = -47{,}3 + 20 \cdot x - 2{,}5 \cdot x^2 + + 0{,}1 \cdot x^3$	0,72
Співіснування з іншими продуктами	Адаптованість до робочого середовища	0,84	$y = 1963{,}2 - 1458{,}3 \cdot x + + 420 \cdot x^2 - 58{,}8 \cdot x^3 + 4 \cdot x^4 - - 0{,}1 \cdot x^5$	0,76

Згідно з табл. 2.5 – 2.8 можна визначити рівень зрілості LibreOffice. Відповідно до інтегральної оцінки якості $Q = 0{,}86$, отримуємо $\sigma = 1{,}17$. Відповідний рівень цілісності дорівнює 1. Відповідно до методики отримуємо $1{,}17\sigma + 3 = 4{,}17\sigma$, що відповідає сертифікованому рівню (4 рівень зрілості).

Висновки по розділу 4

1. З метою апробації працездатності розробленого засобу і, безпосередньо, самих запропонованих методів, наведені скріншоти роботи програмного засобу.

2. Для проведення емпіричного дослідження було зібрано та досліджено дані опитування користувачів та експертів на прикладі LibreOffice, а саме визначено перелік критеріїв зрілості ПП, яким можуть дати оцінку користувачі та виставити ранги експерти; проведено опитування, отримана оцінка поточного рівня зрілості вищезгаданого ПП.

ВИСНОВКИ

У дисертаційній роботі теоретично обґрунтовано й вирішено важливе науково-технічне завдання оцінювання зрілості ПП. Основні результати дисертаційної роботи:

1. За результатами аналізу предметної галузі розроблено онтологію зрілості ПП. Встановлено означення та моделі зрілості ПП, проаналізовано методи та засоби здійснення її оцінювання та забезпечення встановленого вимогами рівня. Показано, що наявні методи досягнення зрілості ПП не направлені на виявлення недоліків та визначення можливих покращень шляхом автоматизованого аналізу зрілості у процесі розробки ПП, а також направлені, як правило, лише на оцінювання зрілості процесів або окремих компонентів ПП, а не на ПП вцілому.

2. Вперше створено модель зрілості ПП з використанням покладено процесного підходу до управління якістю, а формалізоване вирішення відповідних задач планування, контролю, забезпечення та управління зрілістю ґрунтується на запропонованих математичних моделях оцінки та забезпечення зрілості, а також на встановленні залежностей між її показниками.

3. Удосконалено математичну модель оцінювання рівнів зрілості на основі ключових практик, цілей та областей процесів, коду. Застосування моделі дозволяє в разі невідповідності рівня зрілості встановленим вимогам здійснити коригуючі дії з досягнення заданого рівня в результаті рішення оптимізаційної задачі

4. Отримало подальший розвиток вирішення задачі контролю зрілості ПП шляхом створення математичної моделі оцінки, яка, на відміну від наявних, базується на методі вкладених скалярних згорток і описана функцією адитивної скалярної згортки. Застосування моделі дозволяє в процесі розробки ПП контролювати відповідність зрілості встановленим вимогам.

5. Отримано методику реалізації процесу управління змінами зрілості ПП, засновану на математичному апараті кореляційно-регресійного аналізу, що дозволяє встановити залежності між показниками зрілості на основі оцінок користувачів. Встановлені залежності враховуються при аналізі впливу коригуючих дій на рівень зрілості ПП.

6. Спроектовано та реалізовано програмний засіб, спрямований на вирішення завдання управління зрілістю шляхом застосування запропонованих методів. Використання програмного засобу забезпечує автоматизацію управління зрілістю із врахуванням відгуків користувачів.

7. За допомогою розроблених у дисертації методів та засобу оцінювання зрілості ПП проведено оцінку ПП з відкритим вихідним кодом – LibreOffice. Результати дисертаційної роботи впроваджено в навчальний процес Національного авіаційного університету та в процеси розробки програмних продуктів у ТОВ «Lime Systems».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ANSI/IEEE Std. 1008:1987. IEEE Standard for software Unit Testing. – Режим доступу: http://www.ieee.org.

2. IEEE 982.1-1988IEEE 982.1-1988 - IEEE Standard Dictionary of Measures to Produce Reliable Software. Режим доступу: http://www. ieee.org

3. IEEE Std. 610:12-1990. IEEE Standard Glossary of Software Engineering Terminology. – Режим доступу: http://www.ieee.org.

4. IEEE/EIA Std. 12207.0:1996. Software Life Cycle Processes. – Режим доступу: http://www. ieee.org.

5. IEEE/EU Std. 12207.1:1997. Software Life Cycle Processes Life Cycle Data. – Режим доступу: http://www.ieee.org.

6. IEEE Std 830-1993. Recommended Practice for Software Requirements Specification. – Режим доступу: http://www.ieee.org. 1993. – 36 p.

7. IEEE Std. 1233:1998. IEEE Guide for Developing System Requirements Specifications // – Режим доступу: http.7/www. ieee.org. – 1998. – 36 p.

8. IEEE Std. 1016:1998. IEEE Recommended Practice for Software Design Descriptions. – Режим доступу: http.7/www. ieee.org.

9. IEEE Std. 12207.2:1997. Software life Qcle Processes Implementation Consideration. – Режим доступу: http://www.ieee.org

10. IEEE Std. 1062:1998. IEEE Recommended Practice for Software Acquisition. – Режим доступу: http://www. ieee.org.

11. IEEE Std. 1219:1998. IEEE Standard for Software Maintenance.– Режим доступу: http://www. ieee.org.

12. IEEE Std. 828:1998. IEEE Standard for Software Configuration Management Plans. – Режим доступу: http://www. ieee.org.

13. IEEE Std. 730:1998. IEEE Standard for Software Quality Assurance Plans. – Режим доступу: http://www.ieee.org.

14. IEEE Std. 730-1:1995. IEEE Guide for Software Quality Assurance

Planning. – Режим доступу: http://www.ieee.org.

15. IEEE Std. 1059:1993. IEEE Guide for Software Verification and Validation Plans. – Режим доступу: http://www.ieee.org.

16. IEEE Std. 1012:1998. IEEE Standard for Software Verification and Validation. – Режим доступу: http://www.ieee.org.

17. IEEE Std. 1028:1997. IEEE Standard for Software Reviews. - Режим доступу: http://www. ieee.org.

18. IEEE Std. 829-1998. IEEE Standard for Software Test Documentation. – Режим доступу: http://www.ieee.org. – 52 p.

19. IEEE Computer Society, Software Engineering Body of Knowledge, IEEE CS, 2901. – Режим доступу: http://www.ieee.org.

20. ISO/IEC 2382-20:1990. Information Technology–Vocabulary - Part 2: System development. – Режим доступу: http.7/www. iso.org; “tp://www. iec. ch.

21. ISO/IEC 9126. Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their use / – Режим доступу: http://www.iso.org; http://www.iec.ch, 1991. – 14 p.

22. ISO/IEC12119. Information technologies – Software Packages - Quality Requirements and Testing. – Режим доступу: http.//www. iso.org; http://www. iec.ch, 1994. – 29 p.

23. ISO/IEC 2382-1:1993. Information Technology – Vocabulary – Part 1: Fundamental Terms. – Режим доступу: http.7/www. iso.org; http://www. iec.ch.

24. ISO/1EC 12207:1995. Information technologies – Software Life Cycle Processes. – 1995. – 61 p. – Режим доступу: http://www. iso.org; http://www. iec. ch.

25. ISO/IEC 12207:1995 / Amd 1:2002. Information Technology – Software life Cycle Processes. – Режим доступу: http://www.iso.org; http://www. iec.ch.

26. ISO/IEC 12207:1995 / Amd 2:2004. Information Technology - Software Life Cycle Processes. – Режим доступу: http://www.iso.org;

http://www. iec. ch.

27. ISO/IEC 15271:1998. Information Technologies – Guide for ISO/IEC 12207 Software Life Cycle Processes. – Режим доступу http://www.iso.org; http://www. iec. ch.

28. ISO 13407:1999. Human-centred Design Processes for hiterdv- tive Systems. – Режим доступу: http://www.iso.org: hup. www. iec. ch.

29. ISO/IEC 15288:2002. Systems Engineering – System life cycle processes. – Режим доступу: http://www.iso.org; http://www. iec. ch: http://www.gost.ru. – 2002. – 108 p.

30. ISO/IEC 15288:2008. Systems and Software Engineering – System Life Cycle Processes. – Режим доступу: http://www. iso.org; http://www. iec. ch.

31. ISO/IEC 14143-1. Information Technologies Software Requirement – Functional size measurement. – Режим доступу: http://www. iso.org; http://www.iec.ch.

32. ISO/IEC 15939:2002. Software Engineering – Software Requirement Process. – Режим доступу: http.7/www. iso.org; hup iec. ch; http://www. gost.ru.

33. ISO/IEC 14102:1995. Information Technologies – Guidelines for the Evaluation and Selection of CASE Tools. – Режим доступу: http://www. iso.org; http://www. iec. ch.

34. ISO/IEC 15846:1998. Information Technologies – Software Life Cycle Processes-Configuration Management. – Режим доступу: http://www.iso.org; http://www. iec. ch.

35. ISO/IEC 15026:1998. Information technologies – System and Software Integrity Levels. – Режим доступу: http://www.iso.org; http://www. iec. ch.

36. ISO/IEC 14471:1999. Information Technologies – Guidelines for the Adoption of CASE Tools. – Режим доступу: http://www. iso.org; http://www. iec. ch.

37. ISO/IEC 14143-1. Information Technologies Software *• urement – Functional size measurement. – Режим доступу: http://www. iso.org;

http://www.iec. ch.

38. ISO/IEC 15939:2002. Software Engineering – Software r urement Process. – Режим доступу: http.7/www. iso.org; http://www.iec.ch.

39. ISO/IEC 14102:1995. Information Technologies – Guidelines for the Evaluation and Selection of CASE Tools. – Режим доступу: http://www. iso.org; http://www. iec. ch.

40. ISO/IEC 14471:1999. Information Technologies – Guidelines for the Adoption of CASE Tools. – Режим доступу: http://www. iso.org; http://www. iec. ch.

41. ISO/IEC 14764:1999. Information technologies – Software Maintenance. – Режим доступу: http://www. iso.org; http://www. iec.ch.

42. ISO 8402. Quality Management and Quality Assurance – Vocabulary. – Режим доступу: http://www. iso.org.

43. ISO/IEC 15910:1999. Information Technologies – Software user Documentation Process. – Режим доступу: http://www. iso.org; http://www. iec. ch.

44. ISO/IEC 14598-1. Information technology – Software Product Evaluation. – Реяшм доступу: http://www.iso.org; http://www.iec.ch. – Part 1: General Overview, 1999. – 19 p.

45. ISO/IEC 14598-2. Software engineering – Product Evaluation. Part 2: Planning and Management. – Режим доступу: http://www iso.org; http://www.iec.ch, 2000. – 10 p.

46. ISO/IEC 14598-3. Software Engineering – Product Evaluation. Part 3: Process for Developers. – Режим доступу: http://www.iso.0r5 http://www. iec.ch, 2000. – 17 p.

47. ISO/IEC 14598-4. Software Engineering – Product Evaluation. - Part 4: Process for Acquirers. – Режим доступу: http://www.iso.org; http://www.iec.ch, 1999. – 36 p.

48. ISO/IEC 14598-5. Information Technologies – Software Product evaluation. – Режим доступу: http://www.iso.org; http://www.iec.di. – Part 5:

Process for Evaluators, 1998.

49. ISO/IEC 14598-6. Information Technology – Software Product Evaluation. – Part 6: Documentation of Evaluation Modules. – Режим доступу: http://www.iso.org; http://www.iec.ch, 2001. – 17 p.

50. ISO/IEC 6592:2000. Information Technologies – Guidelines for the Documentation of Computer-based Application Systems. – Режим доступу: http://www.iso.org; http://www.iec.ch.

51. ISO/IEC 9294:2005. Information Technologies – Guidelines for the Management of Software Documentation. – Режим доступу: http://www. iso.org; http://www. iec. ch.

52. ISO/IEC 9126-1. Software engineering – Product Quality –I Part 1: Quality model, 2001. – Режим доступу: http://www.iso.org//ISO/IEC. http://www.iec.ch.

53. ISO/IEC 9126-2. Software engineering – Product Quality. – Режим доступу: http://www.iso.org; http://www.iec.ch. – Part 2: External Metrics, 2003. – 86 p.

54. ISO/IEC 9126-3. Software engineering – Product Quality. –I Режим доступу: http://www.iso.org; http://www.iec.ch. – Part 3: Internal Metrics, 2003. – 66 p.

55. ISO/IEC 9126-4. Software engineering – Product quality. - Режим доступу: http://www.iso.org; http://www.iec.ch. – Part 4: Quality in use Metrics, 2004. –70 p.

56. ISO/IEC 17025:1999. General Requirements for the t'oinpe- tence of Testing and Calibration Laboratories. – Режим доступу http://www.iso.org; http://www.iec.ch.

57. ISO/IEC 25000. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2005. – 41 p.

58. ISO/IEC 25001. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Planning and Management, 2007. –

Режим доступу: http://www.iso.org; http://www.iec.ch.

59. ISO/IEC 25010. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Quality Model, 2010. – Режим доступу: http://www.iso.org; http://www.iec.ch.

60. ISO/IEC 25012. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Data quality model. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2008. – 13 p.

61. ISO/IEC 25020. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Measurement Reference Model and Guide. – Режим доступу: http://www.iso.org // ISO/IEC: http://www.iec.ch. – 2007. – 15 p.

62. ISO/IEC 25021. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Quality measure elements. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

63. ISO/IEC 25022. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Measurement of internal quality. – Режим доступу: http://www.iso.org; http://www. iec.ch. – 2007.

64. ISO/IEC 25023. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Measurement of external quality. – Режим доступу: http://www.iso.org; http://www. iec.ch. – 2007.

65. ISO/IEC 25024. Software Engineering – Software Product Quality Requirements arid Evaluation (SQuaRE) – Measurement of quality in use. – Режим доступу: http://www. iso.org; http://www.iec.ch. – 2007.

66. ISO/IEC 25025. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Documentation of evaluation modules. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

67. ISO/IEC 25030. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Quality requirements. – Режим доступу: http://www.iso.org; http://www.iec.ch, 2007. – 36 p.

68. ISO/IEC 25040. Software Engineering – Software Product Quality

Requirements and Evaluation (SQuaRE) – Evaluation Reference Model and Guide. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

69. ISO/IEC 25041. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Evaluation Modules. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

70. ISO/IEC 25042. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Evaluation Process for Developers. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

71. ISO/IEC 25043;. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Evaluation process for acquirers. – Режим доступу: http://www. iso.org; http://www.iec.ch. – 2007.

72. ISO/IEC 25044. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Evaluation process for evaluators. – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2007.

73. ISO/IEC 25051. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing, – Режим доступу: http://www.iso.org; http://www.iec.ch. – 2006. – 27 p.

74. ISO/IEC 25062:2006. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Common Industry Format (CIF) for usability test reports. – Режим доступу. http://www.iso.org; http://www.iec.ch.

75. ISO 9000:2005. Quality management systems – Fundamentals and Vocabulary. – Режим доступу: http://www.iso.org.

76. ISO 9001:2000. Quality Management Systems – Requirements. – Режим доступу: http://www. iso.org.

77. ISO 9004:2000. Quality Management Systems – Guidelines for Performance Improvements. – Режим доступу: http://www.iso.org.

78. ISO 9001:1994. Quality systems – Model for Quality Assurance in Design, Development, Production, Installation and Servicing. – Режим доступу:

http://www.iso.org.

79. ISO 10005:1995. Quality management -– Guidelines for Quality Plans. – Режим доступу: http://www.iso.org.

80. ISO 10006:1997. Quality management – Guidelines to Quality in Project Management. – Режим доступу: http://www.iso.org.

81. ISO 10007:1995. Quality management – Guidelines for Configuration Management. – Режим доступу: http://www.iso.org.

82. ISO 10011-1:1990. Guidelines for Auditing Quality Systems. – Part 1: Auditing. – Режим доступу: http://www.iso.org.

83. ISO 10011-2:1991. Guidelines for Auditing Quality Systems. – Part 2: Qualification Criteria for Quality Systems Auditors. – Режим доступу: http://www.iso.org.

84. ISO 10011-3:1991. Guidelines for Auditing Quality Systems. – Part 3: Management of Audit Programmes. – Режим доступу: http://www. iso.org.

85. ISO 9000-3:1997. Quality Management and Quality Assurance Standards. – Part 3: Guidelines for the Application of ISO 9001:1994 to the Development, Supply, Installation and Maintenance of Computer Software. (Revision of ISO 9000-3:1991). – Режим доступу: http://www. iso.org.

86. ISO/IEC 15504-1:2004. Information Technology – Process Assessment. – Режим доступу: http://www.iso.org; http:// www.iec.ch. – Part 1: Concepts and Vocabulary.

87. ISO/IEC 15504-2:2003. Information Technology – Process Assessment. – Режим доступу: http://www.iso.org; http:// www.iec.ch. – Part 2: Performing an Assessment.

88. ISO/IEC 15504-3:2004. Information Technology – Process Assessment. – Режим доступу: http://www.iso.org; http:// www.iec.ch. – Part 3: Guidance on Performing an Assessment.

89. ISO/IEC 15504-4:2004. Information Technology – Process Assessment. – Режим доступу: http://www.iso.org; http:// www.iec.ch. – Part 4: Guidance on use for Process Improvement and Process Capability Determination

90. IS0/IEC 15504-5:2006. Information Technology – Process Assessment. – Режим доступу: http://www.iso.org; http:// www.iec.ch. – Part 5: An Exemplar Process Assessment Model.

91. ISO/IEC 19759. Software Engineering – Guide to the Software Engineering Body of Knowledge – SWEBOK. – Режим доступу: http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf.

92. ISO 10014:1998. Guidelines for Managing the Economics of Quality. – Режим доступу: http://www. iso.org.

93. ISO 10015:1999. Quality Management – Guidelines for Training. – Режим доступу: http://www. iso.org.

94. ISO 10017:1999. Software Engineering – Guidelines on Statistical Techniques for ISO 9001:1994. – Режим доступу: http://www. iso.org.

95. ISO 19011:2002. Guidelines for Quality and/or Environmental Management Systems auditing. – Режим доступу: http://www. iso.org.

96. ISO 10013:1995. Guidelines for Developing Quality Manuals. – Режим доступу: http://www.iso.org.

97. ISO/IEC 15504-1:1998. Information Technologies Software Process Assessment. Part 1: Concepts and Introduction Guide. – Режим доступу: http://www.iso.org; http://www.iec.ch.

98. ISO/IEC 15504-3:1998. Information technologies – Software Process Assessment. Part 3: Performing an Assessment. – Режим доступу: http://www.iso.org // ISO/IEC. http://www. iec.ch.

99. Guide to Software Engineering Body of Knowledge (SWEBOK). IEEE Computer Society, 2004. – Режим доступу: http://www.swebok.org.

100. Guide to the Project Management Body of Knowledge PMBOK GUIDE. Third Edition. – Режим доступу: http://www.pmi.org/emeaelink/pmiE-linkl0-04.pdf.

101. NASA Std. 2202:1993. NASA Software Formal Inspections Standard. – Режим доступу: http://www.nasa.gov.

102. NASA GB A302.1993. NASA Software Formal Inspections

Guidebook. – Режим доступу: http://www.nasa.gov.

103. NASA Std. 2201:1993. NASA Software Assurance Standard. – Режим доступу: http://www.nasa.gov.

104. NASA GB A201:1995. NASA Software Assurance Guidebook. - Режим доступу: http://www.nasa.gov.

105. NASA Std. 8719.1 3A: 1997. NASA Software Safety Standard. - Режим доступу: http://wwwmasa.gov

106. ДСТУ 3919-1999. Інформаційні технології. Основні напрямки оцінювання та відбору CASE-інструментів. – [Чинний від 1999–11–29]. – К.: Держстандарт України. – 2000. – 39 с.

107. ДСТУ ISO/IEC 12182-2004. Інформаційні технології Класифікація програмних засобів:– [Чинний від 2000-07–01]. – К: Держстандарт України. – Режим доступу: http://www.dstu.mfo; http-//www. ukrndnc.org.ua, http://www. sips.gov.ua.

108. ДСТУ 2850-94. Програмні засоби ЕОМ. Показники та методи оцінювання якості. – [Чинний від 1996-01-01]. – К.: Держстандарт України, 1994. – 20 с.

109. ДСТУ 2844-94. Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення. – [Чинний від 1996-01-01]. – К.: Держстандарт України, 1995. – 15 с.

110. ДСТУ 2462-94. Сертифікація. Основні поняття. Терміни та визначення. – [Чинний від 1995-01-01]. – К.: Держстандарт України, 1994. – 27 с.

111. ДСТУ 2853–94. Програмні засоби ЕОМ. Підготовлення і проведення випробувань. – [Чинний від 1996-01-01]. – К: Держстандарт України, 1994. –17 с.

112. ДСТУ 2851–94. Програмні засоби ЕОМ. Документування результатів випробувань. – [Чинний від 1996–01–01]. – К.: Держстандарт України, 1994. – 11 с.

113. ДСТУ ISO/IEC 15288:2005. Інженерія систем. Процеси життєвого

циклу системи. – [Чинний від 2000-07–01]. – К.: Держстандарт України, 2005.

114. ДСТУ ISO 9000-2001. Системи управління якістю. Основні положення та словник. – [Чинний від 2001–07–01]. – К.: Держстандарт України.

115. ДСТУ ISO 9001–2001. Системи управління якістю. Вимоги: (ISO 9001:2000) – [Чинний від 2001–06–27]. – К.: Держстандарт України, 2001. – 23 с.

116. ДСТУ ISO 9007-95. Стандарти з управління якістю та забезпечення якості. Ч. 3. Настанови щодо застосування під час розроблення, постачання та супроводження програмних засобів. ДСТУ ISO 9000-3-98. – [Чинний від 2001-07-01]. – К.: Держстандарт України, 1998.

117. ДСТУ ISO/IEC 14764. Інформаційні технології – Супроводження програмного забезпечення. – [Чинний від 2000-07-01].–К: Держстандарт України, 2002. – 30 с.

118. ДСТУ 2941-94. Розроблення систем. Терміни та визначення. – [Чинний від 1996.01.01]. – К.: Держстандарт України.

119. ДСТУ ISO/IEC 12119. Інформаційні технології: Пакети програм – Тестування і вимоги до якості. – [Чинний від 2004--01 -01], К.: Держстандарт України. – 2003. – 20 с.

120. ДСТУ ISO/IEC 15504-1:2002. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 1. Концепції та вступна настанова. – [Чинний від 2003-07-01]. – К.: Держстандарт України.

121. ДСТУ ISO/IEC 15504-2:2002. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 2. Еталонна модель процесів та потужності процесу. – [Чинний від 2003-07-01]. – К.: Держстандарт України.

122. ДСТУ ISO/IEC 15504-3:2002. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 3. Виконання оцінювання. – [Чинний від 2003-07-01]. – К.: Держстандарт України.

123. ДСТУ ISO/IEC 15504-4:2002. Інформаційні технології.

Оцінювання процесів життєвого циклу програмних засобів. Ч. 4. Настанови з виконання оцінювання. – [Чинний від 2003-07-01]. – К.: Держстандарт України.

124. ДСТУ ISO/1EC 15504-5:2002. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 5. Модель оцінювання та настанови щодо показників. – [Чинний від 2003-07-01]. – К.: Держстандарт України.

125. ДСТУ ISO/IEC 15504-6:2003. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 6. Настанови з визначення компетенції оцінювачів. – [Чинний від 2004-08-01]. – К.: Держстандарт України.

126. ДСТУ ISO/IEC 15504-7:2003. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 7. Настанови з удосконалення процесу. – [Чинний від 2004-08–01]. – К.: Держстандарт України.

127. ДСТУ ISO/IEC 15504-8:2003. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 8. Настанови з визначання потужності процесу постачальника. – [Чинний від 2004-08-01] – К.: Держстандарт України.

128. ДСТУ ISO/IEC 15504-9:2003. Інформаційні технології. Оцінювання процесів життєвого циклу програмних засобів. Ч. 9. Словник термінів. – [Чинний від 2004-08-01]. – К.: Держстандарт України.

129. ДСТУ ISO 10011-1-97. Настанови щодо перевірки систем якості. Ч. 1. Перевірка. – [Чинний від 1998.06.01]. – К.: Держстандарт України (відповідає ISO 10011 -1:1990).

130. ДСТУ ISO 10011-3-97. Настанови щодо перевірки систем якості. Ч. 2. Кваліфікаційні вимоги до аудиторів з систем якості: ДСТУ ISO 10011-2-97. – [Чинний від 1998-06-01]. – К.: Держстандарт України (відповідає ISO 10011-2:1991).

131. ДСТУ ISO 10011-3-97. Настанови щодо перевірки систем якості,

Ч. 3, Управління програмами перевірок. [Чинний від 1998-01-01]. – К.: Держстандарт України (відповідає ISO 10011-3:1991).

132. Software Specification: A Framework. – Режим доступу: www.sei.cmu.edu/topics/publications/documents/cms/cm.011 .html.

133. Abran Alain. Software Metrics Need to Mature into Software Metrology (Recommendations) // Proceedings **of** the NIST Workshop on Advancing Measurements and Testing for Information Technology. (Gaithersburg (MD), USA, 26-27 Oct. 1998).

134. Alvaro A., de Almeida E. S., and Meira S. L., A Software Component Maturity Model (SCMM). // Proceedings of the 33rd EUROMICRO Conference, Software Engineering and Advanced Applications. – 2007. – pp. 83-92.

135. April A., Huffman Hayes J., Abran A., Dumke R. Software Maintenance Maturity Model (SMmm): the software maintenance process model. // Journal of Software Maintenance and Evolution: Research and Practice. – Vol. 17. – 2005. – pp. 197-223.

136. Alshayeb M., Abdellatif A., Zahran S., Niazi M. Towards a Framework for Software Product Maturity Measurement // ICSEA 2015 : The Tenth International Conference on Software Engineering Advances. – 2015. – pp. 7-11.

137. Brooks Frederick P. The Mythical Man-Month: Essays on Software Engineering, 1995. – 322 c. – (Addison-Wesley).

138. Gardler R. Software Sustainability Maturity Model. [Електронний ресурс]. – Режим доступу до ресурсу: http://oss-watch.ac.uk/resources/ssmm

139. GitLab Maturity [Електронний ресурс]. – Режим доступу до ресурсу: https://about.gitlab.com/direction/maturity/

140. Grinenko O., Grinenko S. One approach to study software ecosystem properties // International Journal “Information Theories and Applications”, Vol. 26, Number 4, 2019. – С. 397 – 405.

141. Grinenko S. A. Mathematical model of providing and software maturity assessment // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 4(28). – С. 5 – 14.

142. Grinenko S.A. Ecological Approach for Research of Software / S.A. Grinenko // XIX Всеукраїнська наукова конференція молодих учених «Актуальні проблеми природничих і гуманітарних наук у дослідженнях молодих учених», Черкаси, 27-28 квітня 2017 р. – Черкаси: Черкаський національний університет ім. Б.Хмельницького, 2017. – С. 202.

143. Grinenko S.A. Theoretical Aspects of the origin and Development of Software Ecosystems / S.A. Grinenko // Міжнародна науково-технічна конференція аспірантів та студентів «Інженерія програмного забезпечення – 2017», Київ, 06-09 червня 2017 р. – Київ: Національний авіаційний університет, 2017. – С. 12.

144. Grinenko S.A. Information technology for assessment of software ecosystem maturity levels / S.A. Grinenko // XIV Міжнародна науково-технічна конференція «АВІА – 2019», Київ, 23-25 квітня 2019 р. – Київ: Національний авіаційний університет, 2019. – С. 6.27 – 6.29.

145. Grinenko S.A. Mathematical Model of IT Ecosystem Based on Transition Systems / S.A. Grinenko // XV Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту – 2019», Херсон – Залізний Порт, 21-25 травня 2019 р. – Херсон: Херсонський національний технічний університет, 2019. – С. 47 – 48.

146. Grinenko S.A. Principles of Sustainable Development in IT industry / S.A. Grinenko // Міжнародна науково-технічна конференція «Інформаційні технології в освіті та науці – 2019», Мелітополь, 13-14 червня 2019 р. – Мелітополь: Мелітопольський державний педагогічний університет ім. Б.Хмельницького, 2019. – С. 88 – 91.

147. Grinenko S.A. One Approach for Evaluation the Maturity of IT Enterprise / S.A. Grinenko // III Міжнародна науково-практична

конференція «Прикладні системи та технології в інформаційному суспільстві – 2019», Київ, 30 вересня 2019 р. – Київ: Київський національний університет імені Тараса Шевченка, 2019. – С. 202.

148. IDEF5 Method Report. Режим доступу: http://www.idef.com/idef5-ontology-description-capture-method

149. Jakobsen A. B., O'Duffy M., Punter T. Towards a maturity model for software product evaluations // Proceedings of 10th european conference on software cost estimation (ESCOM'99), 1999.

150. Etheredge J. How Do We Measure Maturity In Software? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.simplethread.com/how-do-we-measure-maturity-in-software/

151. Houaich Y., Belaissaoui M.. Measuring the maturity of open source software // The 6th International Conference on Information Systems and Economic Intelligence (SIIE), 2015. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.researchgate.net/publication/290391086

152. Kozlovskiy V.V., Grinenko S. A., Skalova V.A. Methods of Evaluation of Software Maturity // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 3(27). – С. 38 – 43.

153. Kozlovskiy V.V., Grinenko S.A. The Measurements of Software Product Maturity // Науковий журнал. Інженерія програмного забезпечення. – 2017. – № 2(30). – С. 52 – 58.

154. Maturity Models [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/joelparkerhenderson/maturity_models

155. Motherudin F., Moksen N. E. Md. A Proposed Model for Code Quality Maturity Model // Journal of Software. – Vol. 10 (3). – 2015. – 374 – 383.

156. Lanza M., Marinescu R., Ducasse S. Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. – Springer-Verlag Berlin Heilderbeg, 2006. – 213 p.

157. Linda M. Laird, M. Carol Brennan Software Measurement and Estimation: a practical approach. John Wiley & Sons, Inc., Hoboken, New Jersey 2006. – 257 p.

158. McGary J., Card D., Jones C., Layman B., Clark W., Dean J., and Hall F. Practical Software Measurement, Objective Information for Decision Makers, Addison-Wesley, Boston, 2002. – 294 p.

159. OSS website [Електронний ресурс]. – Режим доступу до ресурсу:https://opensource.org/licenses

160. Popereshnyak S. Development of an Ontological Model for the Domain of IT Enterprise Sustainable Development / S. Popereshnyak, S. Grinenko // Науковий журнал. Технологічний аудит та резерви виробництва. – 2019. – №3/2(47). – С. 43 – 45.

161. Popereshnyak S. Development of IT Enterprise Infrastructure with Sustainable Principles/ S. Popereshnyak, S. Grinenko // Науковий журнал. East european scientific journal. – 2019. – №3/2(47). – С. 27 – 38 / S. Popereshnyak, S. Grinenko // Науковий журнал. East european scientific journal. – 2019. – №3/2(47). – С. 43 – 45.

162. Popereshnyak S. The Methodology for Assessing the Maturity Level of Software Ecosystems / S. Popereshnyak, S. Grinenko // Proceedings 1st International Workshop on Cyber Hygiene & Conflict Management in Global Information Networks, Kyiv, 29-30 November 2019. – Київ: Національний авіаційний університет, 2019. – С. 202.

163. Pronschinske M. A General Software Maturity Model. [Електронний ресурс]. – Режим доступу до ресурсу: https://dzone.com/articles/a-general-software-maturity-model

164. Qutaish R., Abran A. A maturity model of software product quality. // Journal of Research and Practice in Information Technology. – Vol. 43. – 2011. – pp. 307-327.

165. Sidorov N. A. Software Ecosystem Modeling / N. A. Sidorov, O. O. Grinenko // Інженерія програмного забезпечення. – 2013. – № 2. – С. 38-48.

166. Software as a Service vs Software as a Product [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.bynder.com/en/blog/software-as-a-product-vs-software-as-a-service/.

167. Stafford G. Infrastructure as Code Maturity Model [Електронний ресурс]. – Режим доступу до ресурсу: https://programmaticponderings.com/2016/11/25/infrastructure-as-code-maturity-model/

168. Systems and software engineering, Systems and software Quality Requirements and Evaluation (SQuaRE), System and software quality models: ISO/IEC 25010:2011. – Geneva: International Organization for Standardization /International Electrotechnical Commission, 2011. – 34 p.

169. Андон Ф.И. Основы инженерии качества программных систем / Ф. И. Андон, Г. И. Коваль, Т. М. Коротун [и др.]. – К.: Академпериодика, 2007. – 672 с.

170. Бабенко Л. П. Основи програмної інженерії: навч. посібник // Л. П. Бабенко, К. Лаврищева. – К.: Товариствово «Знання», КСХ, 2001. – 269 с.

171. Благун І.С. Математичні методи в економіці: навч. посібник / І.С. Благун, В.П. Кічор, Р.В. Фещук, С.Й. Воробець; за ред. В.П. Кічора. – Тернопіль: Навчальна книга – Богдан, 2011. – 264 с.

172. Брауде Э. Технология разработки программного обеспечения: пер. с англ. / Э. Брауде. – СПб.: Питер, 2004. – 655 с.

173. Брукс П. Метрики для управления ИТуслугами/ Пер.с англ. – М.: Альпина Бизнес Букс, 2008. – 283 с.

174. Гнеденко Б. В. Курс теории вероятностей / Б. В. Гнеденко. – М.: Наука, 1988. – 447 с.

175. Гріненко О.О. Концептуальні основи моделювання екосистем програмного забезпечення» / О.О. Гріненко, С.А. Гріненко // Науковий журнал. Наукові записки Українського науково-дослідного інституту зв’язку. – 2017. – №1(45). – С. 94 – 103.

176. Гріненко С.А. Інформаційна технологія забезпечення сталого

розвитку ІТ-підприємств // Науковий журнал. Технічні науки та технології. – 2019. – № 3(17). – С. 140 – 145.

177. Дишлевий О. П. Засіб моніторингу та аналіз метрик дефектів якості програмного коду / О. П. Дишлевий, Ю. В. Драпушко // Інженерія програмного забезпечення. – 2013. – № 2. – С. 49-54.

178. Дишлевий О.П. Предметно-орієнтований метод побудови залежностей між метриками програмного забезпечення // Науковий журнал «Вісник НАУ». – 2009. – №3. – С. 206-212.

179. Дишлевий О.П. Пакет статистичного аналізу для емпіричної інженерії програмного забезпечення // Наука і молодь. Прикладна серія. Збірник наукових праць. – 2009. – № 9. – С. 104-108.

180. Дишлевий О.П. Підбір метрик для властивостей програмного забезпечення / О.П. Дишлевий // Науковий журнал «Проблеми програмування». – 2010. – №2-3. – С. 237-242.

181. Зіатдінов Ю.К. Стандартизація та сертифікація інформаційних управляючих систем: навч. Посібник / Ю.К. Зіатдінов, І.Е. Райчев, О.Г. Харченко. – К.: НАУ, 2016. – 184 с.

182. Лаврищева Е: М. Области знаний программной инженерии SWEBOK и подход к обучению этой дисциплине / Е. М. Лаврищева, В. Н. Грищенко // УсиМ. – № 1. –2005. – С. 38-54.

183. Лаврищева Е. М. Концепция аналитической оценки характеристик качества программных компонентов / Е. М. Лаврищева, А. М. Рожнов // Проблемы программирования. – № 2–3. – 2004. – 180-187.

184. Липаев В. В. Выбор и оценивание характеристик качества программных средств. Методы и стандарты / В. В. Липаев. – М. : СИНТЕГ, 2001. – 228 с.

185. Липаев В. В. Оценка качества программных средств / В. Липаев // Сетевой журнал. – № 3. – 2002. – С. 52-56.

186. Марголин Л. Н. Общие вопросы программного обеспечения компьютера [Електронний ресурс] – Режим доступу: http://hi-edu.ru/e-

books/xbook709/01/part-002.htm.

187. Оцінювання [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9E%D1%86%D1%96%D0%BD%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

188. Поперешняк С.В. Модель оцінки ІТ підприємств, яка заснована на інноваційній зрілості персоналу. / С.В. Поперешняк, С.А. Гріненко // Науковий журнал. Вісник інженерної академії. – 2019. – № 3. – С. 162 – 168.

189. Постанова № 869 Кабінету Міністрів України «Про затвердження загальних вимог до програмних продуктів, які закуповуються та створюються на замовлення державних органів» [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/869-2009-%D0%BF#Text.

190. Програмний продукт // Інструкція про порядок складання звіту про використання програмних продуктів і комп'ютерних мереж, 1997 [Електронний ресурс]. – Режим доступу до ресурсу: https://ips.ligazakon.net/document/reg1419?an=1&ed=0000_00_00.

191. Райчев І. Е. Концепція побудови сертифікаційної моделі якості програмних систем // Е. Райчев, О. Г. Харченко // Проблемы программирования. – № 2-3. – 2006. – С. 275-281.

192. Райчев І. Е. Проблеми оцінювання якості критичних програмних систем при їх сертифікації / І. Е. Райчев, О. Г. Харченко // Проблемы программирования. – № 2-3. – 2004. – С. 198- 207.

193. Севастьянов Б. А. Курс теории вероятностей и математической статистики / Б. А. Севастьянов. – М.: Наука, 1982. – 256 с.

194. Словник української мови: в 11 тт. / АН УРСР. Інститут мовознавства; за ред. І. К. Білодіда. – К.: Наукова думка, 1970–1980. – Т. 3. – С. 709.

195. Стандарт онтологического исследования IDEF5. [Електронний ресурс] – Режим доступу: http://citforum.ck.ua/cfin/idef/idef5.shtml

ДОДАТОК А

Глосарій

Атрибут якості ПП – це характеристика / поведінка ПП, яку можна визначати, вимірювати та постійно контролювати для забезпечення того, щоб кінцевий продукт відповідав межам якості, визначеним зацікавленими сторонами.

Зрілість ПП – а) це повна розробка (досконалий стан продукту), вищі рівні можливостей продукту свідчать про вищу зрілість продукту (чим більші продукт має можливості, тим вищий рівень його зрілості).

б) зрілість товару відображається на рівні відповідності продукту необхідним ознакам якості, визначеним зацікавленою стороною продукту.

Еталонна модель – у системах та програмній інженерії – це абстрактна структура або специфічна для галузі онтологія, що складається із взаємопов’язаного набору чітко визначених концепцій, розроблених експертом чи експертною групою. Еталонна модель може представляти складові будь-якої послідовної ідеї, від ділових функцій до системних компонентів, якщо вона представляє повний набір. Потім цю систему відліку можна використовувати для чіткого передавання ідей членам тієї самої спільноти. Довідкові моделі часто ілюструються як сукупність понять із певним зазначенням взаємозв'язку між поняттями. Це забезпечує загальну основу для оцінювачів для оцінки програмного продукту. Довідкова модель описує рівні зрілості програмного продукту, яких він може досягти. Він також надає набір атрибутів та показників якості. Це абстрактна основа для розуміння значущих взаємозв’язків між сутностями певного середовища та для розробки послідовних стандартів або специфікацій, що підтримують це середовище.

Можливість ПП – властивість ПП, яка надає ПП «здатність досягти заздалегідь визначеної мети, яка пов’язана з певним рівнем зрілості». У PMMI можливості ПП на стадії розробки матеріалізуються завдяки

внутрішнім атрибутам якості, а рівень зрілості товару на цьому етапі визначається ступенем відповідності товару внутрішнім вимогам до якості. Результати випробувань (виражені через показники якості) є показниками рівня відповідності. Те саме стосується етапу випуску ПП, але основна увага приділяється зовнішнім атрибутам якості товару.

Програмний продукт (англ. *software product, software as a product (SaaP)*) – програмне забезпечення, розроблене для вирішення задачі масового попиту та призначене для постачання користувачам. Програмний продукт відрізняється від просто програмного забезпечення максимально узагальненим набором вхідних даних, ретельним тестуванням, наявністю документації, гарантії та технічної підтримки. На відміну від програмного забезпечення, яке надається як послуга (*SaaS*), програмні продукти зазвичай встановлюють на обладнанні користувача (власному чи орендованому). Термін ліцензії на продукт зазвичай не обмежується (хоча трапляються випадки, коли виробники обмежують час використання продукту).

Рівні зрілості ПП – для всього ПП - середнє значення всіх рівнів можливостей усіх атрибутів товару. Рівень зрілості ПП визначається за наступною шкалою:

FC: (понад 90%)

LC: (від 60% до 90%)

PC: (від 20% до 60%)

NC: (менше 20%)

Метрика атрибуту якості ПП – це числове значення очікуваного результату випробування конкретного атрибута якості ПП.

ДОДАТОК Б

Список опублікованих праць за темою дисертації

Статті у наукових фахових виданнях України:

17. Гріненко О.О., Гріненко С.А. Концептуальні основи моделювання екосистем програмного забезпечення / О.О. Гріненко, С.А. Гріненко // Науковий журнал. Наукові записки Українського науково-дослідного інституту зв'язку. – 2017. – №1(45). – С. 94 – 103. ***Внесок автора***: запропоновано модельно-орієнтований підхід (MDA) для моделювання екосистем програмного забезпечення.

18. Гріненко С.А. Інформаційна технологія забезпечення сталого розвитку ІТ-підприємств / С.А. Гріненко // Науковий журнал. Технічні науки та технології. – 2019. – № 3(17). – С. 140 – 145.

19. Поперешняк С.В., Гріненко С.А. Модель оцінки ІТ підприємств, яка заснована на інноваційній зрілості персоналу. / С.В. Поперешняк, С.А. Гріненко // Науковий журнал. Вісник інженерної академії. – 2019. – № 3. – С. 162 – 168. ***Внесок автора:*** запропоновано методику оцінювання ІТ підприємств на основі інноваційної зрілості персоналу.

Статті у наукових фахових виданнях України, які входять до міжнародних наукометричних баз даних:

20. Kozlovskiy V.V., Grinenko S. A., Skalova V.A. Models of Evaluation of Software Maturity / V.V. Kozlovskiy, S.A. Grinenko, V.A. Skalova // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 3(27). – С. 38 – 43. ***Внесок автора***: встановлено переваги та недоліки існуючих моделей оцінювання зрілості програмного забезпечення.

21. Grinenko S. A. Mathematical model of providing and software maturity assessment / S.A. Grinenko // Науковий журнал. Інженерія програмного забезпечення. – 2016. – № 4(28). – С. 5 – 14.

22. Kozlovskiy V.V., Grinenko S.A. The Measurements of Software Product Maturity / V.V. Kozlovskiy, S.A. Grinenko // Науковий журнал. Інженерія програмного забезпечення. – 2017. – № 2(30). – С. 52 – 58. ***Внесок автора***: запропоновані методи оцінювання зрілості ПП.

23. Popereshnyak S., Grinenko S. Development of an Ontological Model for the Domain of IT Enterprise Sustainable Development / S. Popereshnyak, S. Grinenko // Науковий журнал. Технологічний аудит та резерви виробництва. – 2019. – №3/2(47). – С. 43 – 45. ***Внесок автора***: розроблена онтологія ПрО сталого розвитку ІТ підприємств.

Статті у закордонних виданнях, які входять до міжнародних наукометричних баз даних:

24. Popereshnyak S., Grinenko S. Model of Innovative Maturity of Personnel for Estimation of IT Enterprises / S. Popereshnyak, S. Grinenko // Науковий журнал. East european scientific journal. – 2019. – Vol.5. – №10 (50). – С. 33 – 40. ***Внесок автора:*** запропоновані моделі для оцінювання ІТ підприємств на основі інноваційної зрілості персоналу.

25. Grinenko O., Grinenko S. One approach to study software ecosystem properties / O.Grinenko, S. Grinenko // International Journal "Information Theories and Applications", Vol. 26, Number 4, 2019. – С. 397 – 405. ***Внесок автора:*** запропоновані характеристики для дослідження екосистем програмного забезпечення.

26. Popereshnyak S., Grinenko S., Grinenko O. Kovalenko O., Radivilova T. The Methodology for Assessing the Maturity Level of Software Ecosystems / S. Popereshnyak, S. Grinenko, O. Grinenko, O. Kovalenko, T. Radivilova // Proceedings of the 1st International Workshop on Cyber Hygiene & Conflict

Management in Global Information Networks. – 2019. – ceur Vol-2654. – pp. 251 - 261. ***Внесок автора:*** розроблено 3-рівневу модель для оцінювання екосистем програмного забезпечення.

Тези наукових конференцій:

27. Grinenko S.A. Ecological Approach for Research of Software / S.A. Grinenko // XIX Всеукраїнська наукова конференція молодих учених «Актуальні проблеми природничих і гуманітарних наук у дослідженнях молодих учених», Черкаси, 27-28 квітня 2017 р. – Черкаси: Черкаський національний університет ім. Б.Хмельницького, 2017. – С. 202.

28. Grinenko S.A. Theoretical Aspects of the origin and Development of Software Ecosystems / S.A. Grinenko // Міжнародна науково-технічна конференція аспірантів та студентів «Інженерія програмного забезпечення – 2017», Київ, 06-09 червня 2017 р. – Київ: Національний авіаційний університет, 2017. – С. 12.

29. Grinenko S.A. Information technology for assessment of software ecosystem maturity levels / S.A. Grinenko // XIV Міжнародна науково-технічна конференція «АВІА – 2019», Київ, 23-25 квітня 2019 р. – Київ: Національний авіаційний університет, 2019. – С. 6.27 – 6.29.

30. Grinenko S.A. Mathematical Model of IT Ecosystem Based on Transition Systems / S.A. Grinenko // XV Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту – 2019», Херсон – Залізний Порт, 21-25 травня 2019 р. – Херсон: Херсонський національний технічний університет, 2019. – С. 47 – 48.

31. Grinenko S.A. Principles of Sustainable Development in IT industry / S.A. Grinenko // Міжнародна науково-технічна конференція «Інформаційні технології в освіті та науці – 2019», Мелітополь, 13-14 червня 2019 р. –

Мелітополь: Мелітопольський державний педагогічний університет ім. Б.Хмельницького, 2019. – С. 88 – 91.

32. Grinenko S.A. One Approach for Evaluation the Maturity of IT Enterprise / S.A. Grinenko // III Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві – 2019», Київ, 30 вересня 2019 р. – Київ: Київський національний університет імені Тараса Шевченка, 2019. – С. 202.

ДОДАТОК В

Відомості про апробацію результатів дисертації

ЗАТВЕРДЖУЮ

ГЕНЕРАЛЬНИЙ ДИРЕКТОР

ТОВ «ЛАЙМ СИСТЕМС»

Гладков Сергій Анатолійович

«06» листопада 2020 р.

АКТ

впровадження результатів дисертаційної роботи

Гріненка Сергія Анатолійовича

Результати дисертаційної роботи Гріненка Сергія Анатолійовича «Методи та засіб оцінювання зрілості програмних продуктів» успішно впроваджені в діяльності ТОВ «ЛАЙМ СИСТЕМС». Розроблені автором методи та моделі, застосовані в робочому процесі ТОВ «ЛАЙМ СИСТЕМС», показали значну ефективність, дозволили підвищити ефективність впровадження рішень для банків у зв'язку з покращенням зрілості програмних продуктів. Запропоновані дисертаційні положення мають практичну цінність і можуть бути використані при управлінні та моніторингу програмних проектів, спільнот та компаній з розробки програмного забезпечення.

ЗАТВЕРДЖУЮ
Проректор з навчальної роботи
_____ А. Полухін
"___" _____ 20__ р.

АКТ ВПРОВАДЖЕННЯ
результатів дисертаційної роботи в
навчальний процес
Національного авіаційного університету

Комісія, що є представниками Національного авіаційного університету, у складі декана факультету кібербезпеки, комп'ютерної та програмної інженерії д.т.н., професора Нестеренко К.С., завідувача кафедри інженерії програмного забезпечення д.т.н., доцента Зибіна С.В., к.т.н., доцента, доцента кафедри інженерії програмного забезпечення Гученко І.В., к.ф.-м.н, доцента, доцента кафедри інженерії програмного забезпечення Оленіна М.В.
склали дійсний акт у тому, що результати дисертаційної роботи Гріненка С.А. "Методи і засіб оцінювання зрілості програмних продуктів"
(назва дисертаційної роботи)
використовуються в навчальному процесі Національного авіаційного університету факультету кібербезпеки, комп'ютерної та програмної інженерії на кафедрі інженерії програмного забезпечення
(ВНЗ, факультет, кафедра)

Що впроваджено (ТЗ, ТТВ, прилад, технологія, програма, алгоритм, модель, методика, рекомендації, регламент, інструкція, винахід, посібник, нормативи та інше)	Форма впровадження (підручник, навчальний посібник, конспект лекцій, методичні розробки, лабораторний практикум, програма курсу, постановка лабораторної роботи, продовження розробки в курсовому та дипломному проекті)	Ефект від впровадження
1. Методи дослідження зрілості програмних продуктів. 2. Програмний засіб, який забезпечує автоматизацію дослідження зрілості програмних продуктів.	1. Використовуються в навчальному процесі при проведенні практичних робіт згідно з програмою навчальної дисципліни «Екологія програмного забезпечення» зі спеціальності 121 «Інженерія програмного забезпечення». 2. Використовується в навчальному процесі при проведенні лабораторних робіт згідно з програмою навчальної дисципліни «Дослідження програмих продуктів, технологій створення і супроводження» зі спеціальності 121 «Інженерія програмного забезпечення». 3. Використовується в навчальному процесі при проведенні лабораторних робіт згідно з програмою навчальної дисципліни «Моделювання процесів зрілості в інженерії програмного забезпечення» зі спеціальності 121 «Інженерія програмного забезпечення».	1. Підвищення якості навчального процесу шляхом використання запропонованих методів. 2. Підвищення ефективності навчання шляхом використання програмного засобу.

Голова комісії д.т.н., професор Нестеренко К.С.

Члени комісії: д.т.н., доцент Зибін С.В.
к.т.н., доцент Гученко І.В.
к.ф-м. н., доцент Оленін М.В.