Міністерство освіти і науки України
Національний технічний університет “Харківський політехнічний інститут”

Кваліфікаційна наукова
праця на правах рукопису

МАРТІНКУС ІРИНА ОЛЕГІВНА

УДК 004.051

ДИСЕРТАЦІЯ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗРОБКИ ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ НА ОСНОВІ МЕТОДІВ ТА ЗАСОБІВ ДОМЕННОГО МОДЕЛЮВАННЯ

05.13.06 - інформаційні технології
технічні науки

Подається на здобуття наукового ступеня кандидата наук

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

Науковий керівник
Ткачук Микола Вячеславович, доктор технічних наук, професор

Харків-2017

АНОТАЦІЯ

Мартінкус І.О. Інформаційна технологія розробки лінійок програмних продуктів на основі методів та засобів доменного моделювання. - Кваліфікаційна наукова праця на правах рукопису

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 «Інформаційні технології». – Національний технічний університет “Харківський політехнічний інститут”, Міністерство освіти і науки України, Харків, 2017.

У дисертаційні роботі запропоновано рішення актуальної науково-практичної задачі створення інформаційної технології розробки лінійок програмних продуктів на основі методів та засобів доменного моделювання. Об’єктом дослідження в роботі виступає процес розробки сімейств програмних систем (СПС) та лінійок програмних продуктів (ЛПП) із використанням методів та інструментальних засобів побудови та використання доменних моделей (ДМ).

Методи дослідження основані на використанні принципів прикладного системного аналізу та теорій множин – для формалізації задачі визначення ефективності застосування технологій доменного моделювання та розробки алгоритмічної моделі; методів доменного моделювання та принципів генеруючого програмування - для дослідження та розробки підходів до розробки ЛПП на основі методів доменного моделювання; принципів об’єктно - орієнтованого аналізу – для розробки методу визначення структурно-функціональної складності доменних моделей; методів багатокритеріального аналізу, кількісних метрик оцінки структурних та функціональних характеристик програмних систем – для розробки методу визначення ступеня повторного використання програмного коду.

В роботі запропонована формалізація задачі визначення ефективності застосування методів доменного моделювання (МДМ) та їх інструментальних засобів при розробці лінійок програмних продуктів (ЛПП) за рахунок введення

евристичних тверджень. Також в роботі запропоновано 3-х вимірний інформаційний простір для моделювання ефективності застосування технологій доменного моделювання при розробці ЛПП.

Проаналізовано та узагальнено структурно-логічні взаємозв'язки між показниками якості ПЗ, метриками структурної складності коду та кількісними показниками повторного використання, що відображено у відповідних запропонованих «mind-mapping» схемах. На підставі цього аналізу було обрано відповідно множину метрик для визначення ступеня повторного використання коду для подальшого застосування у розробленому методі оцінки рівня ПВ. Це дозволяє враховувати при розробці ЛПП один із таких важливих показників якості ПЗ як супроводжуваність.

Запропоновано концептуальні схеми розробки ЛПП із використанням МДМ як з «нуля» так і на основі УПС. При цьому процес розробки ЛПП пропонується розглядати як функціонування системи автоматизованого управління з контуром зворотного зв'язку, в якому як критерій оцінки якості управління використовується коефіцієнт ефективності застосування технологій доменного моделювання. При розробці ЛПП на основі УПС виникає проблема відновлення початкових вимог до системи. Цю проблему автором пропонується вирішити за рахунок застосування адаптивної матриці трасування (ADTM), що дозволяє зв'язати вимоги до ПЗ з програмними артефактами (вихідним кодом) та надає можливість простежити відповідні зміни. В результаті із використанням матриці ADTM шляхом ретроспективної обробки каталогу завдань (Tasks), які вже були виконані для цієї УПС, виконується реконструкція повного набору бізнес-вимог користувачів (User Story).

Розроблено алгоритмічну модель (АМ) процесу визначення ефективності застосування МДМ та їх інструментальних засобів при розробці ЛПП, яка дозволяє представити в формалізованому вигляді компоненти, що необхідні для кількісного визначення коефіцієнту ефективності. Запропонована АМ відрізняється від існуючих підходів комплексним використанням багатовимірного інформаційного ресурсу, експертних методів та відповідних

метрик, що дозволяє кількісно оцінити ступінь ефективності застосування окремих МДМ як у разі розробки ЛПП з «нуля» так і у разі їх створення на основі використання успадкованих програмних систем (УПС). Представлена АМ уявляє собою кортеж, який складається із:

- інформаційного базису моделі, що представляє собою кортеж із множини методів доменного моделювання, множини технологій їх реалізації та множини відповідних доменних моделей;
- сукупності алгоритмів реалізації методів оцінки ефективності застосування технологій доменного моделювання, до яких належать метод визначення ступеня повторного використання програмного коду та метод визначення показника складності ДМ;
- колекція метрик, що застосовуються в відповідних алгоритмах моделі АМ, до яких належать метрики оцінки структурної складності програмного коду (застосовуються для аналізу каркасів, отриманих шляхом генерації на основі відповідної ДМ), метрики оцінки ступеня повторного використання коду, метрик оцінки структурно-функціональної складності ДМ, метрики визначення коефіцієнту ефективності застосування певної ДМ при розробці ЛПП.

В роботі отримали подальший розвиток моделі та інструментальні засоби дослідження технологічних особливостей використання МДМ в процесах розробки ЛПП шляхом розробки експертного методу аналізу структурно-функціональної складності ДМ, що дає можливість отримувати кількісні оцінки структурно-функціональної складності відповідних ДМ. Запропонований підхід , на відміну від існуючих, враховує не лише структурні елементи ДМ, але й її функціональне наповнення. Крім того перевага запропонованого методу полягає у тому, що він враховує основні типи зав’язків між елементами моделі та їх вплив на загальну складність ДМ.

Запропоновано метод визначення ступеня повторного використання вихідного коду, шляхом використання сукупності кількісних метрик і обчислювальних алгоритмів, що дозволяє враховувати структурну складність

програмних компонентів ЛПП. Завдяки комбінації у цьому метод декількох методів багатокритеріального аналізу (а саме попарних порівнянь та комплексної оцінки), можливо його застосування у різних проектних ситуаціях, а саме :

- у випадку, коли достатня кількість ООП-експертів;
- у випадку, коли кількість ООП-експертів мінімальна, але є можливість отримати тестові розрахунки за критеріями (метриками);
- у випадку, коле немає можливості застосувати експерті оцінки і є доступ лише до тестових розрахунків критеріїв (метрик).

Також у роботі представлена процедура комплексної оцінки ефективності використання методів та засобів доменного моделювання.

На підставі запропонованих методів, моделей та алгоритмів розроблено інформаційну технологію оцінки ефективності застосування окремих МДМ в процесах розробки ЛПП за рахунок використання експертних методів у поєднанні із кількісними метриками оцінки рівня повторного використання вихідного коду та метрик структурно-функціональної складності ДМ.

Запропоновано архітектуру інструментального CASE - засобу для автоматизації ряду процесів дослідження ефективності технологій доменного проектування. Розроблений інструментальний засіб (прототип) дозволяє автоматизовано проводити розрахунки щодо визначення кількісних параметрів для коефіцієнту ефективності – складність доменної моделі та ступень повторного використання коду. На аналіз програмному забезпеченню подається розроблена доменна модель (у XML-форматі) та згенерований каркас коду. Прототип програмного засобу розроблено як десктопне ПЗ із використанням мови програмування Java, та також додатково застосовувалася платформа JavaFX для реалізації графічного інтерфейсу та CKJM-бібліотека для розрахунку деяких метрик структурної складності.

Для експериментального дослідження розробленого підходу в роботі пропонується методика проведення експерименту, яка налічує ряд основних етапів. Ця методика визначає; як і у якій формі необхідно отримати вхідні

данні; побудову відповідних моделей та множин; проведення необхідних обчислень для визначення коефіцієнту ефективності.

Це забезпечує можливість автоматизації процесів попереднього аналізу та оцінки ефективності альтернативних варіантів розробки нових компонентів ЛПП для мотивованого вибору технології доменного проектування на ранньому етапі життєвого циклу ПЗ.

Для тестування запропонованого підходу до оцінки ефективності було проведено ряд експериментів. В якості тестової ПрО та ЛПП виступала система управління учбовими закладами (спільний україно-австрійський аутсорсинговий проект). В результаті проведених експериментів були отримані загальні висновки щодо працездатності та доцільності запропонованого підходу до визначення ефективності застосування альтернативних технологій доменного моделювання:

- для всіх експериментів з 2-ма обраними технологіями доменного моделювання ODM/EMF та JODA/Actifsorce зберігається стала тенденція переваги застосування саме технології ODM / EMF, що свідчить про відповідну достовірність отриманих експериментальних результатів;
- шляхом застосування запропонованого підходу є можливим забезпечити обрання технології доменного моделювання, яка забезпечує зростання значення коефіцієнту ефективності її використання в 1.4 - 1.9 рази;
- існує певна залежність між ступенем можливого зростання коефіцієнту ефективності використання обраної технології Keff та рівнем складності відповідної доменної моделі DMC, і визначення типу цієї залежності може бути предметом подальших досліджень.

Практична значимість отриманих результатів дисертаційної роботи визначається тим, що розроблені методи та інструментальні засоби можуть бути використані для підвищення ефективності процесів розробки та супроводу СПП та ЛПП із застосуванням методів доменного моделювання (ДМ), у тому числі і на основі успадкованих програмних систем.

Запропонований комплексний підхід до оцінки ефективності використання ДМ і розроблені для його підтримки інструментальні програмні засоби були успішно використані для вирішення цих завдань в розробках по прикладній держбюджетній НДР МОН України, при виконанні IT-проектів в компаніях «Інтерпак – Інформаційні системи» (м. Харків) та Bitmedia e-Learning Solution GmbH & Co KG (м. Зальцбург, Австрія), а також впроваджені в навчальному процесі кафедри програмної інженерії та інформаційних технологій управління НТУ «ХПІ» в дисциплінах «Аналіз вимог до ПЗ», «Основи проектування ПЗ», «Моделювання та аналіз проблемно-орієнтованих програмних систем».

За результатами дисертаційного дослідження опубліковано 11 наукових праць, у тому числі 5 статей у наукових фахових виданнях України з технічних наук, 2 статті – у наукових періодичних іноземних виданнях (наукометрична база «Scopus»), і 4 публікації у тезах та матеріалах міжнародних конференцій та семінарів.

Ключові слова: програмне забезпечення, проблемно-орієнтована розробка, доменна модель, лінійка програмних продуктів, повторне використання, алгоритмічна модель, структурна складність, ефективність, метрика, CASE-засіб.

Список публікацій здобувача

1. Ткачук Н.В., Нагорный К.А., Мартинкус И.О. Методика оценки динамической сложности требований в процессах сопровождения унаследованных программных систем. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”.2010. № 67. с.116-125.

2. Tkachuk M., Martinkus I. Models and Tools for Multi-dimensional Approach to Requirements Behavior Analysis / ed. H.C. Mayr et al. UNISCON 2012, LNBIP 137: Springer-Verlag Berlin Heidelberg, 2013. pp. 191-198.

3. Гамзаев Р. О., Ткачук Н. В., Мартинкус И. О. Мета-модель процесса трассировки требований при разработке программного обеспечения. Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. Х: НТУ «ХПІ» 2014. № 26 (1069). с.121-128

4. Tkachuk M.V., Gamzaev R.A., Martinkus I.O., Ianushkevych S.D. Methods and Tools for Dynamic Requirements Catalog Management in Agile Software Development. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”. 2015. № 52(1058). с.11-18 .

5. Tkachuk, M., Martinkus, I., Gamzayev, R., Tkachuk A. An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development / ed. Heinrich C. Mayr, Martin Pinzger. INFORMATIK 2016. Lecture Notes in Informatics (LNI). Vol. P-259: Kollen Druck+Verlag GmbH, Bonn, 2016. pp. 143-156.

6. Мартинкус І.О., Ткачук М.В., Гамзаєв Р.О. Конструювання лінійок програмних продуктів із застосуванням доменного моделювання та метрик повторного використання коду. Системи управління, навігації та зв’язку: зб. наук. пр. ЦНДІ НУ. К., 2017. Вип. 3(43). с. 93-97.

7. Ткачук М.В., Мартінкус І.О., Нагорний К.А., Гамзаев Р. О. Про один підхід до оцінки ефективності застосування методів доменного моделювання при розробці сімейств програмних систем. Збірник наукових праць Харківського національного університету Повітряних Сил. 2017. №5(54), с. 45-54

8. Мартинкус И.О., Гамзаев Р.А., Ткачук Н.В. Модели и технологи разработки линеек программных продуктов. Інформаційні технології: наука, техніка, технологія, освіта, здоров’я: Тези доповідей XXII міжнародної науково-практичної конференції, (15-17 жовтня 2014р., Харків) / за ред. проф. Товажнянського Л.Л. Харків, НТУ «ХПІ». С.13.

9. Martinkus I.O., Gamzayev R.A, Tkachuk, M. V. Towards Effectiveness Assessment of Domain Modeling Approaches to Software Development.

Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXIII міжнародної науково-практичної конференції, (21-22 травня 2016р., Харків) / за ред. проф. Сокола Є.І. Харків, НТУ «ХПІ». С.25.

10. Ткачук М.В., Гамзаев Р.О., Мартінкус І.О. Підхід до розробки лінійок програмних продуктів на основі успадкованих програмних систем із використанням методів доменного моделювання. Теоретичні та прикладні аспекти побудови програмних систем:матеріали міжнародної наукової конференції, м. Київ, 5-9 грудня 2016р. / редкол. М.С. Нікітченко, Л.Б. Буй та ін. Кіровоград, «Центр оперативної поліграфії «Авангард». 2016. с. 236-241.

11. Мартінкус І.О. Інформаційна технологія розробки та супроводу лінійок програмних продуктів на основі методів та засобів доменного моделювання. Сучасні інформаційні технології. Матеріали засідань школи-семінару за 2016-2017 н. р. Харків: ХНУ імені Каразіна, 2017. с.24-28

ABSTRACT

Martinkus I.O. Information technology for software products lines development based on methods and tools of domain modeling. – Qualifying scientific work on the rights of the manuscript.

A thesis for the candidate degree in technical sciences in the specialty 05.13.06 – information technology - National Technical University "Kharkiv Polytechnic Institute", Ministry of Education and Science of Ukraine, Kharkiv, 2017.

In the dissertation research it was proposed a solution of an actual scientific-practical issue concerning elaboration of an information technology for software product lines development based on methods and tools of the domain modelling. An object of the research in the dissertation is the process of software system families (SSF) and software product lines (SPL) design using methods and tools for domain models (DM) engineering and usage.

The research methods are based on usage of system analysis principles and the set theory – to formalize an issue of usage effectiveness assessment of domain modelling technologies and an algorithmic model elaboration; domain modelling methods and generative programming principles – to research and elaborate approaches for SPL design based on domain modelling; object-oriented analysis principles – to estimate the structural-functional complexity of domain models; multiple-criteria analysis methods, quantitative metrics for software systems' structural and functional characteristics estimation – to elaborate a method for the source code reusability extent (CRE).

In the dissertation research, it was proposed a formalization of the usage effectiveness assessment issue for domain modelling methods (DMM) and corresponding CASE-tools during software product lines development due to introduction of heuristic definitions. It was proposed a 3-D information space to model usage effectiveness of domain modelling technologies during SPL design, as well.

Were analyzed structural-logical interconnections between software quality, structural source code complexity metrics and quantitative indicators of reusability, which is represented at corresponding proposed “mid-mapping” schemes. On the basis of this analysis an appropriate metric set was selected for a code reuse extent determination, for further usage in the elaborated method for the reusability estimation. This gives a possibility to consider one of such important software quality indicators as the reusability with in SPL development process.

Were proposed conceptual schemes of the LSS development based on MDM, namely: “from scratch” and on the basis of LSS. At the same time it is proposed an SPL development process to consider as functioning of an automated control system with a feedback loop, where as a control quality assessment criterion the domain modelling usage effectiveness coefficient is used. During SPL development based on LSS a problem of reconstruction of initial requirements to a software system emerges. The author proposes to solve this problem with the adaptive trace matrix (ADTM) application, this permit to bind software system requirements to program artefacts (source code) and let to back-trace corresponding changes. As a result of ADTM application, using the retrospective task catalog (Tasks) processing, which already have been implemented for this LSS, a reconstruction of the full business-requirements collection (User Story) is performed.

The algorithmic model (AM) of the process of the MDM usage effectiveness and their CASE-tools assessment during SPL design was elaborated, this model gives a possibility to formalize components, required for effectiveness coefficient estimation. Proposed AM is being different from existent approaches because of the complex usage of multi-dimensional informational resource, expert methods and respective metrics, this permit to estimate quantitatively usage effectiveness degree of each MDM in both cases: a SPL development “from scratch” and a SPL development on the basis of a legacy software system (LSS). Presented AM represents a tuple, which consists of:

- model’s informational basis, which is a tuple of domain modelling methods set, a set of their realization technologies and a set of corresponding domain models;

- aggregate of algorithms of methods for domain modelling technologies usage effectiveness assessment, the source code reuse extent estimation method and the DM complexity indicator estimation method are their representatives.

- metric collection, which is used in a corresponding algorithms of the AM, among which are following: source code structural complexity estimation metrics (they are applied for frames analysis, have been got by a generation based on a respective DM), metrics of source code reuse extent estimation, metrics for DM structural-functional complexity estimation, metrics of a particular DM usage effectiveness coefficient estimation during SPL development.

In the dissertation research a further evolution have got models and CASE-tools for research of MDM usage technological peculiarities within SPL design process by elaboration of the expert method of DM structural-functional complexity analysis, which permits to obtain quantitative estimations of DM structural-functional complexity. The proposed approach, in contrast to the existing ones, considers not only DM structural element, but it's functional content as well. Moreover, an advantage of the proposed approach is in the consideration of main relation types between model elements and their impact on the overall DM complexity.

Was proposed the source code reuse extent method, which includes usage of metrics aggregate and estimation algorithms, which gives a possibility to take into account structural complexity of SPL components. Because of a combination of some multi-criteria analysis methods (namely pairwise comparisons and complex evaluation), it is possible to apply it for different project cases, viz:

- in case, if there is enough OOP-experts quantity;

- in case, if quantity of OOP-experts is minimal, but there is a possibility to obtain test calculations due to criteria (metrics);

- in case, if there is no possibility to use expert estimations and only test calculations of criteria (metrics) available.

Also in the research was introduced the procedure of the complex assessment of domain model methods and CASE-tools usage effectiveness.

On the basis of proposed methods, models and algorithms, there was the information technology designed for particular MDM usage effectiveness assessment in SPL development process, by expert methods application combined with quantitative metrics of source code reuse extent level estimation and DM structural-functional complexity metrics.

An architecture of CASE-tool was proposed – a CASE-tool for an optimization of a bunch of processes for effectiveness exploration of domain project technologies. Designed prototype of the CASE-tool permits to perform automated calculations concerning evaluation of quantitative parameters for the effectiveness coefficient – domain model complexity and source code reuse extent. For the analysis input is a domain model (in XML-format) and a generated source code frame. The CASE-tool prototype is designed as a desktop application based on Java programming language, platform JavaFX was used additionally for graphical user interface development and CKJM-library to calculate some metrics.

For an experimental examination of the elaborated approach in the dissertation research a methodic of experiment implementation is proposed, which consist of series of primary phases. This methodic defines: which form should be input data; appropriate models and sets construction; making required calculation to obtain effectiveness coefficient.

This provides a possibility of automation for the initial analysis and assessment of the effectiveness of alternative variants for SPL novel components to motivate a choice of domain project technology at the early software lifecycle phase.

To test the proposed approach for effectiveness assessment, a bunch of experiments were provided. As a test subject domain and SPL was used a system of education institutions management (a joint Ukrainian-Austrian outsource project). As a result of provided experiments general conclusions were done regarding operativeness and expediency of the proposed approach to usage effectiveness assessment of alternative domain modelling technologies:

- for all experiments with two selected domain modelling technologies ODM/EMF and JODA/Actifsource it is preserved a constant tendency of usage

superiority exactly ODM/EMF technology, this testify of trustworthiness of obtained results;

- with an application of the proposed approach it is possible to supply the selection of domain modelling technology, which guarantees increase of it's usage effectiveness coefficient in 1.4 – 1.9 times.

- a particular dependency exists between degree of possible increase of the selected technology usage effectiveness K_{Eff} and corresponding domain model complexity DMC and determination of this dependency level could be an object for the further research.

The practical significance of the dissertation research results is determined by the fact that elaborated methods and CASE-tools could be used to increase design and maintenance of SSF and SPL processes effectiveness with domain modelling (DM) usage, including design based on legacy software systems.

The proposed comprehensive approach to assess DM usage effectiveness and designed CASE-tool for it's support were successfully used to solve this objectives in context of applied state budget for research and development of the Ministry of Education and Science of Ukraine, in context of IT-projects in "Interpak – Information Systems" (Kharkiv, Ukraine) and Bitmedia e-Learning Solution GmbH & Co KG (Salzburg, Austria) companies, also in context of educational process of "Software Engineering and Management Information Technologies" department of National Technical University "Kharkiv Polytechnical institute" for "Software requirements analysis", "Essential principles of software project", "Problem-oriented systems modelling and analysis" disciplines.

According to the results of the dissertation research, 11 scientific works were published, including 5 articles in scientific professional editions of Ukraine on technical sciences, 2 articles in scientific periodical foreign publications (Scopus science and technology base), and 4 publications in theses and materials of international conferences and seminars.

Key words: software, problem-oriented development, domain model, software product line, reuse, algorithmic model, structural complexity, efficiency, metric, CASE-tool.

List of publications of the applicant

1. Ткачук Н.В., Нагорный К.А., Мартинкус И.О. Методика оценки динамической сложности требований в процессах сопровождения унаследованных программных систем. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”.2010. № 67. с.116-125.

2. Tkachuk M., Martinkus I. Models and Tools for Multi-dimensional Approach to Requirements Behavior Analysis / ed. H.C. Mayr et al. UNISCON 2012, LNBIP 137: Springer-Verlag Berlin Heidelberg, 2013. pp. 191-198.

3. Гамзаев Р. О., Ткачук Н. В., Мартинкус И. О. Мета-модель процесса трассировки требований при разработке программного обеспечения. Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. Х: НТУ «ХПІ» 2014. № 26 (1069). с.121-128

4. Tkachuk M.V., Gamzaev R.A., Martinkus I.O., Ianushkevych S.D. Methods and Tools for Dynamic Requirements Catalog Management in Agile Software Development. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”. 2015. № 52(1058). с.11-18 .

5. Tkachuk, M., Martinkus, I., Gamzayev, R., Tkachuk A. An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development / ed. Heinrich C. Mayr, Martin Pinzger. INFORMATIK 2016. Lecture Notes in Informatics (LNI). Vol. P-259: Kollen Druck+Verlag GmbH, Bonn, 2016. pp. 143-156.

6. Мартинкус І.О., Ткачук М.В., Гамзаєв Р.О. Конструювання лінійок програмних продуктів із застосуванням доменного моделювання та метрик

повторного використання коду. Системи управління, навігації та зв'язку: зб. наук. пр. ЦНДІ НУ. К., 2017. Вип. 3(43). с. 93-97.

7. Ткачук М.В., Мартінкус І.О., Нагорний К.А., Гамзаев Р. О. Про один підхід до оцінки ефективності застосування методів доменного моделювання при розробці сімейств програмних систем. Збірник наукових праць Харківського національного університету Повітряних Сил. 2017. №5(54), с. 45-54

8. Мартинкус И.О., Гамзаев Р.А., Ткачук Н.В. Модели и технологи разработки линеек программных продуктов. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXII міжнародної науково-практичної конференції, (15-17 жовтня 2014р., Харків) / за ред. проф. Товажнянського Л.Л. Харків, НТУ «ХПІ». С.13.

9. Martinkus I.O., Gamzayev R.A, Tkachuk, M. V. Towards Effectiveness Assessment of Domain Modeling Approaches to Software Development. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXIII міжнародної науково-практичної конференції, (21-22 травня 2016р., Харків) / за ред. проф. Сокола Є.І. Харків, НТУ «ХПІ». С.25.

10. Ткачук М.В., Гамзаев Р.О., Мартінкус І.О. Підхід до розробки лінійок програмних продуктів на основі успадкованих програмних систем із використанням методів доменного моделювання. Теоретичні та прикладні аспекти побудови програмних систем:матеріали міжнародної наукової конференції, м. Київ, 5-9 грудня 2016р. / редкол. М.С. Нікітченко, Л.Б. Буй та ін. Кіровоград, «Центр оперативної поліграфії «Авангард». 2016. с. 236-241.

11. Мартінкус І.О. Інформаційна технологія розробки та супроводу лінійок програмних продуктів на основі методів та засобів доменного моделювання. Сучасні інформаційні технології. Матеріали засідань школи-семінару за 2016-2017 н. р. Харків: ХНУ імені Каразіна, 2017. с.24-28

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АМ – алгоритмічна модель

БД - база даних;

ГП – генерувальне програмування

ДАМ – доменний аналіз та моделювання

ДМ – доменна модель

КПВ - компонент повторного використання

ЛПП – лінійка програмних продуктів

ПВ – повторне використання

СКБД – система керування БД

СПС – сімейство програмних систем

ПС – програмна система

ПЗ – програмне забезпечення

ПрО – предметна область

УПС – успадкована програмна система

DDD - domain-driven design

ODM - Organizational Data Modeling

DSM – domain specific modeling

EMF - Eclipse Modeling Framework

DMM – Domain Modeling Method

DMT – Domain Modeling Technology

DM – domain model

DMC – Domain Model Complexity

GCF - Generated Code Framework

CRE - code reusability extent

ВСТУП

Актуальність теми. Сучасний етап розвитку сфери інформаційних технологій (ІТ) в цілому та, зокрема, такої її важливої галузі як інженерія програмного забезпечення (ПЗ), характеризується зростанням питомої ваги інтелектуальних, знання-орієнтованих підходів до створення нових проектних рішень та реалізації відповідних програмних систем (ПС). Саме таким шляхом, за умов постійного ускладнення функціональних задач, які мають бути вирішені шляхом використання цих систем, та стрімкого розширення переліку предметних областей (ПрО) їх застосування, можуть бути зменшені витрати на розробку та супровід відповідного ПЗ. В свою чергу, це передбачає можливість побудови та подальшого ефективного використання вже не окремих ПС, а цілих сукупностей взаємопов’язаних складних програмних компонентів, які отримали назву лінійок програмних продуктів (ЛПП), або сімейств програмних систем (СПП) [1-3].

Одним з найбільш ефективних шляхів вирішення ц задачі побудови ЛПП та СПС є повторне використання (reuse) різних проектних активів (assets), а саме: специфікацій вимог (requirements) до ПЗ, еталонних програмних архітектур (reference software architectures), проектних патернів (design patterns) і, нарешті, вихідного коду (source code) ПС [2,3]. Для досягнення цієї мети в сучасній інженерії ПЗ широко застосовується концепція предметно-орієнтованого проектування (domain-driven design), в якій центральне місце займає поняття доменної моделі (domain model) як засобу для концептуалізації та повторного використання знань щодо предметної області (ПрО) розробки ПС [4-7]. Створення як СПС так і ЛПП передбачає необхідність побудови доменної моделі (ДМ) для заданої ПрО, на основі якої, із використанням відповідних інструментальних засобів (CASE-tools), можлива генерація каркасу програмного коду (source code framework), який потім має бути основою для створення програмних компонентів повторного використання (КПВ). Таким чином, з методологічної точки зору, розробка як СПС так і ЛПП використовує

загальну парадигму генерувального програмування (generative programming) [3], яка передбачає можливість комплексного застосування як різних методів побудови ДМ для заданої ПрО, так і відповідних технологічних середовищ створення, накопичення та контролю версій КПВ. Саме ці проблеми досліджуються та вирішуються в роботах таких відомих вітчизняних та закордонних вчених як І.В. Сергієнко, О.В. Палагін, М.С. Нікітченко, Н.Д. Панкратова, Г.М. Жолткевич, М.М. Глибовець, Д.В. Федасюк, І. Соммервіль (I. Sommerville), К. Поль (K. Pohl) та інших фахівців.

Слід зазначити, що практичне застосування методів та засобів доменного моделювання як в процесах розробки нових ПС так і при супроводі успадкованих ПС (УПС) пов’язано з додатковими витратами часу та інших проектних ресурсів. Тому, зважаючи на це, а також беручи до уваги існування на теперішній час значної кількості альтернативних методів та інструментальних засобів для побудови доменних моделей, виникає проблема визначення ефективності використання певної технології доменного моделювання у відповідному проекті. Таким чином, постановка задачі дослідження цієї дисертаційної роботи є достатньо актуальною та практично значущою.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалася на кафедрі програмної інженерії та інформаційних технологій управління НТУ «Харківський політехнічний інститут» відповідно до завдань прикладної держбюджетної НДР МОН України «Розробка інтелектуальних моделей та технологій для підвищення ефективності проектування та супроводу складних програмних систем» (ДР № 0111U002288) де здобувач брав участь як співвиконавець окремих розділів.

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності використання методів та інструментальних засобів побудови доменних моделей (ДМ) у процесі розробки лінійок програмних продуктів (ЛПП) шляхом розробки та застосування інформаційної технології, яка поєднує

в собі моделі, експертні методи, кількісні метрики та відповідні програмні рішення. Для досягнення цієї мети в роботі поставлені такі задачі:

- проаналізувати технологічні особливості процесів розробки ЛПП, у тому числі і таких, які створюються на основі успадкованих програмних систем (УПС), з урахуванням можливостей застосування в цих процесах методів та засобів побудови ДМ;
- запропонувати формалізований підхід та розробити відповідну алгоритмічну модель (АМ) для визначення ефективності застосування методів та засобів побудови ДМ в процесах розробки ЛПП;
- розробити метрики та метод для кількісної оцінки структурно-функціональної складності різних ДМ;
- розробити метод для кількісної оцінки ступеня повторного використання програмного коду, який отримано в процесі розробки ЛПП на основі побудови та опрацювання відповідної ДМ;
- запропонувати інформаційну технологію для реалізації розробленого формалізованого підходу до визначення ефективності застосування методів та інструментальних засобів побудови ДМ в процесах розробки ЛПП;
- реалізувати основні програмні компоненти інформаційної технології для практичного застосування запропонованого підходу в процесах розробки ЛПП;
- провести експериментальне дослідження працездатності запропонованого підходу та на основі аналізу отриманих результатів надати практичні рекомендації щодо підвищення ефективності використання ДМ в процесах розробки ЛПП.

Об'єктом дослідження є процеси розробки та супроводу лінійок програмних продуктів (ЛПП) із використанням методів та інструментальних засобів побудови та використання доменних моделей (ДМ).

Предметом дослідження є моделі, методи та програмні засоби для оцінки ефективності застосування ДМ в процесах розробки ЛПП.

Методи дослідження базуються на застосуванні принципів сучасної програмної інженерії, зокрема на використанні кількісних метрик якості ПЗ, об'єктно-орієнтованих методах аналізу та синтезу ПЗ, а також на використанні базових положень теорії автоматичного управління, математичного апарату теорії множин, експертних методів теорії прийняття рішень і застосуванні уніфікованої мови моделювання UML (Unified Modeling Language) та нотації побудови мап пам'яті (mind mapping) для аналізу вимог та синтезу проектних рішень.

Наукова новизна отриманих результатів:

1. *Вперше* запропоновано алгоритмічну модель (АМ) процесу вибору методів доменного моделювання (МДМ) та інструментальних засобів при розробці ЛПП. АМ використовує оригінальний критерій ефективності, який визначається як відношення ступеню повторного використання (ПВ) згенерованого програмного коду до рівня структурно-функціональної складності відповідної доменної моделі (ДМ). Це дозволяє кількісно оцінити ефективність застосування альтернативних технологій доменного моделювання як у разі розробки нових ЛПП, так і в процесі реінжинірингу успадкованих програмних систем (УПС).

2. *Отримали подальший розвиток* методи дослідження технологічних особливостей використання МДМ в процесах розробки ЛПП за рахунок використання запропонованого методу визначення структурно-функціональної складності ДМ, який на відміну від існуючих дозволяє врахувати не лише структурні елементи ДМ, але й її функціональність, а також різні типів зв'язків між структурними елементами ДМ.

3. *Отримали подальший розвиток* методи аналізу та визначення ступеня ПВ вихідного коду шляхом застосування сукупності метрик і обчислювальних алгоритмів, що дозволяє враховувати структурну складність програмних компонентів ЛПП ще на етапі їх проектування.

4. *Удосконалено* інформаційну технологію розробки ЛПП за рахунок розробки підходу до визначення ефективності застосування окремих методів та

засобів доменного моделювання в процесах розробки ЛПП, який забезпечує можливість автоматизації процесів попереднього аналізу та оцінки ефективності альтернативних варіантів розробки нових компонентів ЛПП шляхом використання експертних методів у поєднанні із кількісними метриками обчислення рівня ПВ вихідного коду та метриками структурно-функціональної складності ДМ.

Практичне значення. На основі запропонованих моделей та методів доведених до інженерних методик та розробленої інформаційної технології реалізовано інструментальний CASE – засіб для автоматизації ряду процесів дослідження ефективності технологій доменного проектування. Розроблений інструментальний засіб дозволяє автоматизовано проводити розрахунки щодо визначення кількісних параметрів для коефіцієнту ефективності – структурно-функціональну складність ДМ та рівень ПВ коду. На аналіз програмному забезпеченню подається розроблена доменна модель (у XML-форматі) та згенерований каркас програмного коду. Інструментальний CASE – засіб розроблено із використанням мови програмування Java, та також додатково застосовувалася платформа JavaFX для реалізації графічного інтерфейсу та CKJM-бібліотека для розрахунку деяких метрик структурної складності ПЗ.

Розроблені в дисертаційній роботі моделі, методи та інструментальні засоби можуть бути використані для підвищення ефективності процесів розробки ЛПП із застосуванням методів побудови ДМ, у тому числі і на основі УПС. Запропонований комплексний підхід до оцінки ефективності використання ДМ і розроблені для його підтримки інструментальні програмні засоби були успішно використані для вирішення цих завдань в розробках по прикладній держбюджетній НДР МОН України, при виконанні IT-проектів в компаніях «Інтерпак – Інформаційні системи» (м. Харків) та Bitmedia e-Learning Solution GmbH & Co KG (м. Зальцбург, Австрія), а також впроваджені в навчальному процесі кафедри програмної інженерії та інформаційних технологій управління НТУ «ХПІ» в дисциплінах «Аналіз вимог до ПЗ», «Основи проектування ПЗ», «Моделювання та аналіз проблемно-орієнтованих програмних систем».

Особистий внесок здобувача. Усі наукові результати, викладені в дисертації, отримані здобувачем особисто. У роботах, які написано у співавторстві, особистий внесок здобувача полягає у такому: розглянуті особливості аналізу та моделювання вимог в процесах супроводу успадкованих програмних систем (УПС) [8]; розроблено підхід до оцінки структурної складності УПС [9]; виконано аналіз існуючих підходів до трасування вимог в процесах розробки ПС на основі моделей предметних областей [10], розроблені інструментальні засоби для автоматизації процесів отримання експертних оцінок показників пріоритетності та складності вимог до ПЗ [11]; розроблено метод оцінки ступеня повторного використання програмного коду, який отримано шляхом генерування на основі доменних моделей [12]; запропонована технологічна схема розробки ЛПП із використанням методів доменного моделювання [13]; розроблена алгоритмічна модель процесу визначення ефективності технологій доменного моделювання та запропоновано метод оцінки структурно-функціональної складності ДМ [14]; проведено аналіз існуючих моделей і технологій побудови ЛПП [15]; подано опис інструментальних засобів та отриманих експериментальних даних щодо оцінки структурної складності програмного коду на основі доменних моделей [16]; запропонована технологічна схема розробки ЛПП на основі УПС із використанням методів доменного моделювання [17], представлені результати експериментального дослідження запропонованого підходу [18].

Апробація результатів дисертації. Основні положення та результати досліджень пройшли апробацію та одержали позитивну оцінку на 4th International United Information Systems Conference «UNISCON-2012» (Ялта, 2012); на XXI, XXIII Міжнародних науково-практичних конференціях «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я:», (Харків - 2014, 2016); на International conference INFORMATIK'2016 (Klagenfurt, Austria), на Міжнародній науковій конференції «Теоретичні та прикладні аспекти побудови програмних систем» (Київ, 2016), а також на наукових семінарах кафедри програмної інженерії та інформаційних технологій

управління НТУ «ХПІ», кафедри теоретичної та прикладної системотехніки Харківського національного університету імені В.Н. Каразіна та кафедри штучного інтелекту Харківського національного університету радіоелектроніки.

Публікації. За результатами дисертаційного дослідження опубліковано 11 наукових праць, у тому числі 5 статей у наукових фахових виданнях України з технічних наук, 2 статті – у наукових періодичних іноземних виданнях (наукометрична база «Scopus»), і 4 публікації у тезах та матеріалах міжнародних конференцій та семінарів.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів основної частини, висновків, списку використаних джерел, додатків.

У ***першому розділі*** проаналізовано причини виникнення сучасних методів та технологій розробки ЛПП, та наведена їх стисла характеристика. Розглянуто основні характеристики принципів та методів доменного моделювання, генерувального програмування та запропоновано узагальнюючу 3-х рівневу схему застосування ДМ. Проаналізовано основні аспекти проблеми повторного використання при розробці ПЗ, а також наведена постановка задачі дослідження.

У ***другому розділі*** детально проаналізовано існуючи методи доменного моделювання та інструментальні засоби їх реалізації. Запропонована формалізація задачі визначення ефективності застосування методів та засобів доменного моделювання у процесах розробки ЛПП. Досліджено структурно-логічні взаємозв'язки між показниками якості програмного забезпечення, метриками його структурної складності та ступенем повторного використання вихідного коду. Розроблено концептуальні схеми розробки ЛПП з використанням методів доменного моделювання як з «нуля», так і на основі УПС.

У ***третьому розділі*** запропонована алгоритмічна модель процесу оцінки ефективності застосування методів та засобів доменного моделювання.

Розроблено метод визначення структурно-функціональної складності доменних моделей та метод визначення ступеня повторного використання програмного коду. Запропоновано процедура комплексної оцінки ефективності використання методів та засобів доменного моделювання. Розроблені моделі, методи та процедури, що забезпечують вирішення задач дослідження, були інтегровані в єдину прикладну інформаційну технологію для автоматизації процесу отримання оцінки ефективності використання технологій доменного моделювання при розробці ЛПП.

У ***четвертому розділі*** описується предметна область дослідження. Наведена архітектура та основні програмні компонентів інтегрованого CASE-засобу для реалізації запропонованого підходу . Запропонована методика проведення експерименту, вхідні дані та приклади розрахунків чисельних експериментів. Проаналізовано отримані результати і наведено практичні рекомендації по використанню методів та засобів доменного моделювання.

У ***висновках*** приведені основні результати дисертаційної роботи.

У ***додатках*** до дисертації наведено акти впровадження результатів дослідження, список публікацій здобувача та приклади документації з розробки інструментального програмного засобу для оцінки ефективності застосування технологій доменного проектування.

Список джерел, які використано у даному розділі, наведено у повному списку інформаційних джерел під номерами: [1-18].

1 АНАЛІЗ СУЧАСНИХ ПРОБЛЕМ РОЗРОБКИ СІМЕЙСТВ ПРОГРАМНИХ СИСТЕМ ТА ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ ІЗ ЗАСТОСУВАННЯМ МЕТОДІВ ДОМЕННОГО МОДЕЛЮВАННЯ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз причини виникнення та стисла характеристика сучасних методів та технологій розробки сімейств програмних систем та лінійок програмних продуктів

Сучасна розробка програмного забезпечення (software development), згідно з визначенням актуального міжнародного стандарту для системної та програмної інженерії IEEE/ISO/IEC 15288-2015 [19], є комплексом складних міждисциплінарних процесів, що утворюють декілька взаємопов'язаних фаз, які можуть бути формалізовано представлені у вигляді відповідної моделі життєвого циклу (life cycle model), як традиційних: каскадна, спіральна, так і сучасних: V-подібна, інкрементна, еволюційна та ін. [3, 6].

В той же час слід зазначити, що переважна більшість сучасних методологій розробки ПЗ, і в першу чергу новітні так звані гнучкі (agile-) підходи [19], мають на меті досягнення двох основних цілей:

1) зменшення витрат на реалізацію відповідного ІТ-проекту з урахуванням необхідності реалізації всіх функціональних вимог та атрибутів якості [20] до цільової ПС,

2) скорочення часу, потрібного для виходу цього продукту на ринок користувачів подібних ПС [3, 4, 6].

Одним з найбільш ефективних шляхів вирішення цих двох взаємопов’язаних задач (1) – (2) є забезпечення принципу повторного використання (reuse) різних проектних активів (project assets) при розробці ПС, а саме [3,6,21-23]:

а) специфікацій вимог (requirements) до ПЗ,

b) еталонних архітектур (reference architectures) ПС,

c) проектних патернів (design patterns) і, нарешті,

d) вихідного коду (source code) ПС.

В свою чергу, для досягнення відповідного рівня повторного використання активів (a) – (d) в сучасній інженерії ПЗ широко застосовується концепція предметно-орієнтованого проектування (domain-driven design - DDD), в якій центральне місце займає поняття доменної моделі (domain model) як засобу для концептуалізації та повторного використання експертних знань (expert knowledge) щодо предметної області (ПрО) розробки ПС [3-5,24-27].

Підхід DDD застосовується для аналізу інформаційних об'єктів та структурування знань в складних ПрО (див. більш докладно в п.1.2), що в свою чергу дозволяє перейти від розробки окремих ПС до створення сімейств програмних систем (software system families) та лінійок програмних продуктів (software product lines), що також є однією з характерних тенденцій в сучасній програмній інженерії [2-3,28-30]. Слід зазначити, що ці два терміни досить часто використовуються як синоніми, у тому сенсі, що обидва вони позначають особливу сукупність компонентів ПЗ, які мають як певні загальні, так і деякі специфічні функціональні властивості, а також спеціальні механізми забезпечення варіабельності (variability) [3,31], за рахунок чого вони можуть бути налаштовані для багаторазового використання при розв'язанні різних задач у відповідній ПрО. Але, як зазначається у деяких роботах [2, 3, 32], певна різниця між сімейством програмних систем (СПС) та лінійкою програмних продуктів (ЛПП) полягає в тому, що члени СПС, як правило, застосовуються разом для вирішення проблемних задач у відповідній ПрО, в той час коли окремі компоненти ЛПП можуть бути використані автономно у певній проектній ситуації. Типовим прикладом СПС може слугувати сімейство сервіс-орієнтованих програмних рішень на хмарній платформі корпорації Oracle SOA Suite 11g [33], а у якості такого прикладу ЛПП можна навести добре відомий пакет офісних програм MS Office [34].

В подальшому в цій дисертаційній роботі розглядаються саме проблеми побудови ЛПП, більш точне визначення яких наведено в документах такої провідної міжнародної організації в сфері ІТ-технологій як Інститут програмної інженерії університету Карнегі - Мелона (США) [35], а саме:

«*… software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way / ...лінійка програмних продуктів є сукупністю програмних систем, які мають загальний керований набір властивостей, що задовольняють специфічні потреби певного сегменту ринку або мають певне призначення, і які розроблені з єдиного набору базових активів у заздалегідь визначений спосіб…* ».

За рахунок цих своїх властивостей застосування ЛПП забезпечує, зокрема, такі переваги у порівнянні з процесом виготовлення окремих ПС, а саме:

- зниження витрат на розробку (до 60%),
- зменшення часу виходу програмного продукту на ринок (в окремих випадках до 90%),
- суттєве скорочення потреби у кількості розробників ІТ- проекту (до 80%)

та деякі інші [35].

Розглядаючи типові структури ЛПП (див. , напр. в [28]), зазвичай виділяють 3 основні групи її програмних компонентів (див. рис. 1.1), а саме:

i. постійні компоненти, які утворюють так зване ядро (core) відповідної ЛПП,
ii. варіабельні компоненти, тобто такі, що вже існують і можуть бути налаштовані на специфічні особливості застосування того чи іншого продукту у складі цієї ЛПП,
iii. нові компоненти, а саме такі, що мають бути розроблені додатково для даної ЛПП, із урахуванням нових функціональних вимог користувачів і.т.д.

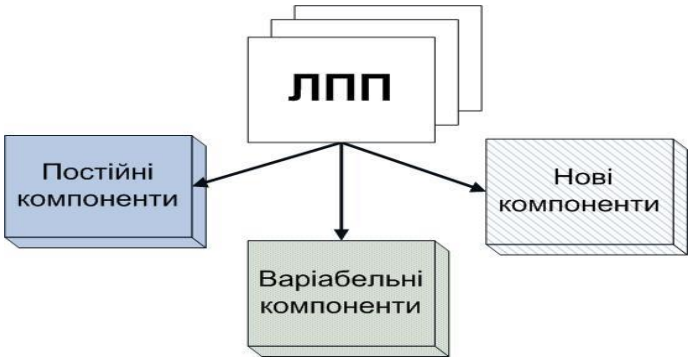

Рисунок 1.1 - Типова структура компонентів ЛПП

Узагальнюючи дослідження, що представлені у [21], компоненти будь-якої ЛПП можна розділити на їх основні типи (i) – (iii) за наступною ознакою:

- постійні компоненти з групи (i) – це компоненти, в функціональності яких кількість змін не перевищує 25%;
- варіабельні компоненти з групи (ii) – це компоненти, в функціональності яких кількість змін знаходиться в межах 25% - 75%;
- нові компоненти з групи (iii) – це компоненти, в функціональності яких кількість змін перевищує 75%.

Конкретним прикладом подібної ЛПП, що в подальшому розглядається в цій роботі, є сукупність ПС, які застосовуються для управління організаційними процесами в навчальних закладах різного профілю підготовки учнів. Компоненти цієї ЛПП схематично представлені на рисунку 1.2:

- прямокутниками позначені постійні компоненти: «Stable» компоненти;
- овалами позначено варіабельні компоненти: «Var» компоненти;
- трикутниками позначено новій компоненти: «New» компоненти.

Приклад формування програмних систем (PS) для даної ЛПП (SPL):

$$PS_1 = \{Stable_1, Stable_2, Var_1, New_1, New_N\}$$
$$PS_2 = \{Stable_1, Stable_N, Stable_3, Var_1, New_2\}$$
$$PS_3 = \{Stable, Stable_3, Var_2, Var_4, New_1, New_N\} \quad (1.1)$$
$$PS_4 = \{Stable, Stable_N, Var_2, Var_N\ New_3, New_N\}$$
$$PS_5 = \{Stable_1, Stable_3, Var_3, New_4\}.$$

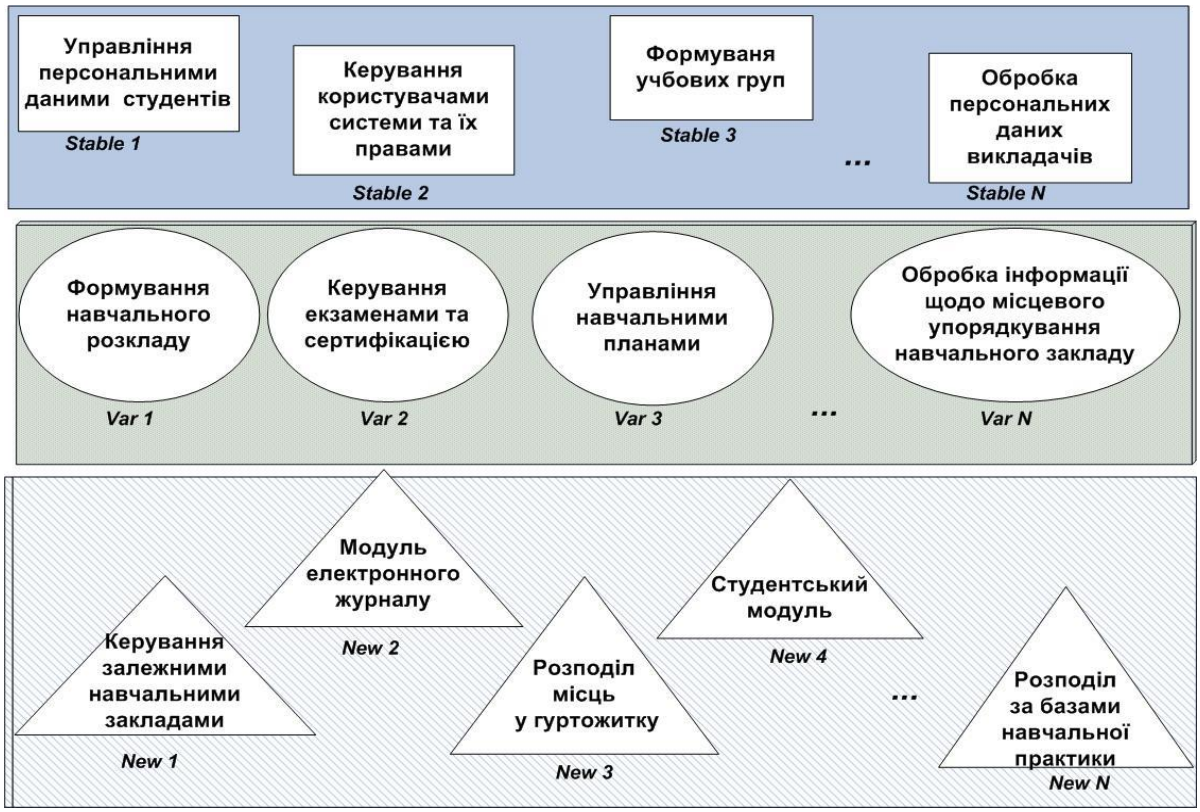

Рисунок 1.2 - Структура ЛПП для управління навчальними закладами

І, відповідно, ЛПП, що може бути побудована на основі цих ПС, має наступний вигляд:

$$SPL = \{PS_1, PS_2, PS_3, PS_4, PS_5, \}. \quad (1.2)$$

Проблеми розробки та ефективної реалізації ЛПП розглядаються у працях багатьох фахівців [28-30, 36], і про їх актуальність говорить те, що на протязі вже більш ніж 20 років щорічно проводиться представницька міжнародна науково-практична конференція Software Product Line Conference (SPLC) [36]. Так, зокрема, серед цих проблем можуть бути визначені наступні:

- розробка еталонних архітектур ЛПП [29],
- створення фреймворків для розробки та супроводу ЛПП [28],
- управління вимогами при розробці ЛПП.[27],
- трансформація УПС у ЛПП [37, 38]

та деякі інші.

В той же час більш детальний аналіз отриманих при цьому результатів показує, що до сих пір недостатньо опрацьованими залишаються наступні досить важливі питання, а саме

- аналіз властивостей ЛПП, які впливають на ефективність процесів їх розробки та супроводу;
- дослідження показників структурної складності програмних компонентів ЛПП які впливають на підвищення ступеня їх повторного використання;
- розробка підходів структурно-функціональної складності ДМ на основі яких створюється відповідні ЛПП;
- розробка підходів до визначення ефективності застосування альтернативних доменних моделей з метою підвищення показників якості побудови ЛПП.

Наведені вище підходи до розробки ЛПП, вже отримані при цьому результати, а також ще невирішені питання стосуються, насамперед, процесів створення таких систем «з нуля», де ресурсом для побудови відповідної доменної моделі є початкові вимоги користувачів до окремих ПС у складі майбутньої ЛПП. Але в той же час, існують проектні ситуації, коли має бути вирішена задача трансформації вже існуючих успадкованих програмних систем (legacy software system) у відповідну ЛПП і саме такі питання розглядаються в роботах [17, 37-38].

1.2 Доменне моделювання як концептуальна основа для автоматизації процесів створення ЛПП із використанням парадигми генерувального програмування

Розглядаючи методологічні основи концепції доменного моделювання в сфері програмної інженерії, слід зазначити, що її появу слід розглядати з урахуванням наступних чинників:

1) по-перше, в інженерії програмного забезпечення вже досить давно використовуються формалізовані моделі бізнес-процесів та інформаційних об’єктів, таких як: моделі даних різного типу для створення баз даних [39,40], різноманітні моделі компонентних програмних систем і технологій [6,41], моделі процесів проектування та реалізації ПЗ із застосуванням уніфікованої мови об’єктно-орієнтованого моделювання UML [42], архітектурні моделі різного рівня в концепції MDA (Model-Driven Architecture) [43] та деякі інші;

2) по-друге, взагалі в сфері розробки IT-технологій вже також на протязі останніх 10-15 років успішно застосовуються знання-орієнтовані методи аналізу та проектування складних інформаційних систем, зокрема, із застосуванням доменних онтологій (domain ontology) [44-48], категорних моделей (див., напр. в [49]) та інших засобів формалізованого опису слабко-структурованих об’єктів та процесів.

Таким чином, можна стверджувати, що концепція доменного моделювання в програмній інженерії виникла як логічне узагальнення і продовження вже існуючих модельних абстракцій, що застосовуються на різних етапах життєвого циклу ПЗ, а особливе значення вона набувала з появою таких нових технологій розробки ПЗ як необхідність побудови лінійок програмних продуктів (ЛПП) та сімейств програмних систем (СПС) [1-3,24,28-31].

Основними принципами доменного моделювання в контексті його застосування для предметно-орієнтованого проектування ПЗ (або DDD-підходу) є наступні [4,6,25]:

- аналіз предметної області, визначення об'єктів і зв’язків між ними;
- визначення області дій об'єктів домену;
- визначення загальних функціональних і варіабельних характеристик, побудова моделі характеристик;
- створення базису для реалізації конкретних екземплярів програмних систем у ЛПП з механізмами їх створення;
- підбір і підготовка готових компонентів багаторазового застосування;
- опис аспектів виконання завдань ПрО;

- генерація окремого домену, члена сімейства і ПС в цілому.

В даний час DDD підхід вважається визнаною методологією для побудови складних ПС в різних областях застосування для досягнення наступної мети: забезпечити високий рівень багаторазового використання проектних активів у при створені нових, або модифікації вже існуючих ПС [4, 24, 25].

Хоча основні суттєві переваги та деякі обмеження DDD вже активно обговорюються в багатьох сучасних публікаціях, представляється доцільним відзначити наступну позитивну його властивість, а саме: існує важлива аналогія між DDD-підходом до розробки ПЗ та добре відомим 3-рівневим представленням даних у процесах розробки БД [40], а також Model Driven Architecture, яка представлена схемою на рис.1.3.

Рисунок 1.3 - 3-х рівнева схема застосування принципів доменного моделювання при розробці ПС

Суть цієї аналогії полягає у тому, що:

- для однієї і тієї ж моделі предметної області, що представляє собою концептуальний рівень моделювання ПЗ, можливо побудувати декілька її реалізацій за допомогою відповідних методів доменного моделювання, що в свою чергу, представляє логічний рівень моделювання ПЗ;

- для кожної реалізації доменної моделі можливо, використовуючи відповідні CASE-засоби, можливо згенерувати цільовий каркас програмного коду (generated code framework), який представляє собою фізичний рівень моделювання ПЗ.

В той же час створення ЛПП невід'ємно пов'язане із процесом генерувального програмування (generative programming). Вивченням проблем застосування генерувального програмування (ГП) займаються провідні фахівці у програмній інженерії [3,50-53], і зокрема, в Україні, наукова група в Інституті програмних систем НАН України під керівництвом проф. К.М.Лаврищевої [3,53]. ГП об'єднує в собі основи усіх сучасних парадигм програмування, і тому його інколи називають мультипарадигмальним програмуванням (див. рис. 1.4).

Рисунок 1.4 - Структурна схема середовища генеруючого програмування з розроблення СПС та ЛПП [3]

Згідно з визначення в роботі [3], ГП - це методи і інструментальні засоби побудови сімейств програмних систем на основі, компонентів повторного використання (КПВ). Модельним базисом цього процесу є об’єктно-орієнтований підхід, доповнений механізмами опису специфічних властивостей КПВ, з можливістю подальшої автоматичної генерації їх вихідного коду (напр., на Java, C#, C++ тощо).

Як і в інших підходах до розробки СПС, у парадигмі ГП також відокремлюють 2 основні рівня розробки: інженерія домену (Domain Engineering) та розробка ПЗ (Application Engineering). У парадигмі ГП рівень Domain Engineering розглядається як збір, класифікація і визначення задач, подання вимог до КПВ і системи в цілому, наявність понять і термінів, що використовують групи розробників, а також знання про способи конструювання СПС для деякої предметної області. Application Engineering у парадигмі ГП розглядається як складальний процес розроблення ПС з окремих компонентів програмних повторного використання, за концепцією збирального конвеєра [3].

Аналізуючи ці аспекти було визначено, що для ефективного застосування основних концепцій ГП необхідні методи та метрики для кількісної оцінки показників ефективності розробки та застосування компонентів ГП при розробці СПС та ЛПП. Узагальнений процес побудови ЛПП із використанням методів доменного моделювання є взаємодією двох основних операційних рівнів: рівень моделювання домену та рівень розробки ПС, і він представлено у вигляді концептуальної схеми на рис. 1.5. Цей процес складається із наступних дій:

1) Кінцеві користувачі майбутньої ЛПП є основним джерелом вимог. Таким чином, користувачі є основними доменними експертами в даному процесі. На основі вимог, що надійшли від користувачів формуються Історії користувача (User Stories) із якої формується загальна специфікація домену. Ця активність схематично зображена стрілкою «1».

2) Згідно з тим, що доменна модель (ДМ) відображає лише функціональність домену (без урахування нефункціональних вимог), на основі загальної специфікації домену формується функціональність домену. Ця активність схематично зображена стрілкою «2».

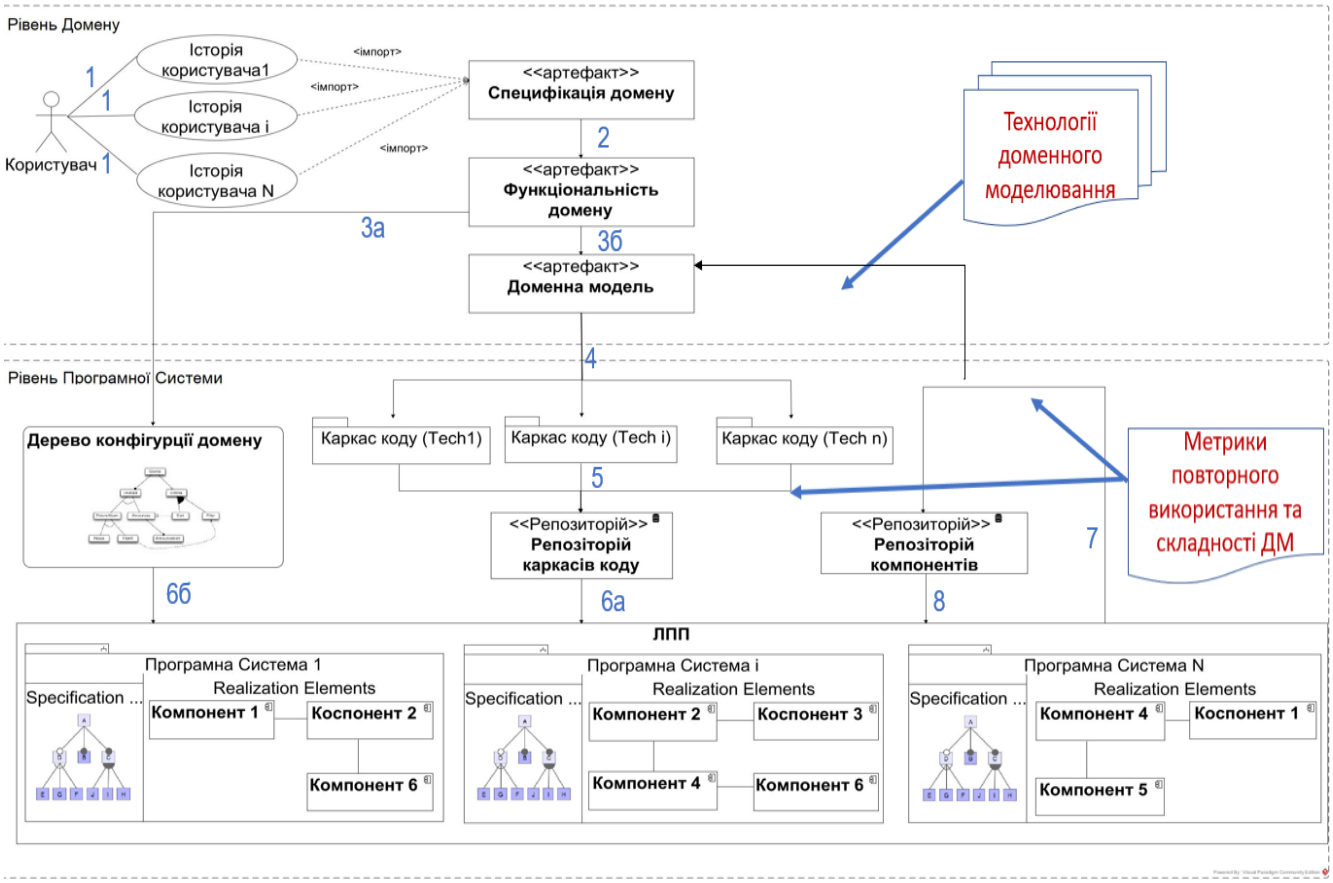

Рисунок 1.5 - Концептуальна схема розробки ЛПП та проблеми вибору технологій доменного моделювання

3) Враховуючи концептуальний опис об’єктів, їх відношень та необхідну функціональність, зазначених у попередньому артефакті, створюється відповідна ДМ із застосуванням обраної технології доменного моделювання. Ця активність схематично зображена стрілкою «3а». Також на цьому етапі паралельно формується дерево конфігурації домену, за допомогою якого визначається які компоненти входять в кожний екземпляр ЛПП. Ця активність схематично зображена стрілкою «3б». Таким чином вже на цьому етапі

створення ЛПП необхідно приймати рішення щодо вибору найбільш ефективної технології доменного моделювання для поточної ПрО.

4) Наступним кроком є генерація каркасу похідного коду у відповідному CASE-засобі доменного моделювання, що зображено стрілкою «4». Цей етап є перехідним між рівнем проектування домену та рівнем розробки ПЗ.

5) Отримані каркаси коду заносяться до репозиторію каркасів коду. Цей процес позначено стрілкою номер «5».

6) На основі репозиторію каркасів коду (стрілка «6а») та дерева конфігурацій (стрілка «6б») створюються екземпляри програмних систем для ЛПП.

7) Використовуючи усі сформовані екземпляри ПС відокремлюються унікальні компоненти із яких формується репозиторій компонентів для повторного використання. Ця активність схематично зображена стрілкою «7». На цьому етапі закінчується ітерація розробки поточної версії ЛПП.

8) На наступній ітерації циклу розробки ЛПП при формування екземплярів ПС можливо застосовувати вже розроблені компоненти із репозиторію. Ця активність схематично зображена стрілкою «8».

Аналізуючи це процес в ньому визначаються дві основні критичні точки та відповідні задачі: «Яким чином мотивовано обрати технологію доменного проектування?» та «Необхідні метрики для оцінки коду на доменної моделі».

1.3 Аналіз актуальності проблеми повторного використання проектних активів як засіб підвищення ефективності процесів розробки ЛПП

Процес розробки ЛПП є достатньо актуальним для сучасної програмної інженерії, що підтверджується численними публікаціями та конференціями у цьому напрямку [2, 24, 36, 54]. В цих роботах розглядаються типові підходи до розробки ЛПП, особливості технологічного процесу їх створення та типові

проблеми та задачі що виникають у ЖЦ ЛПП. Узагальнений процес розробки нової ЛПП представлений на рисунку 1.6.

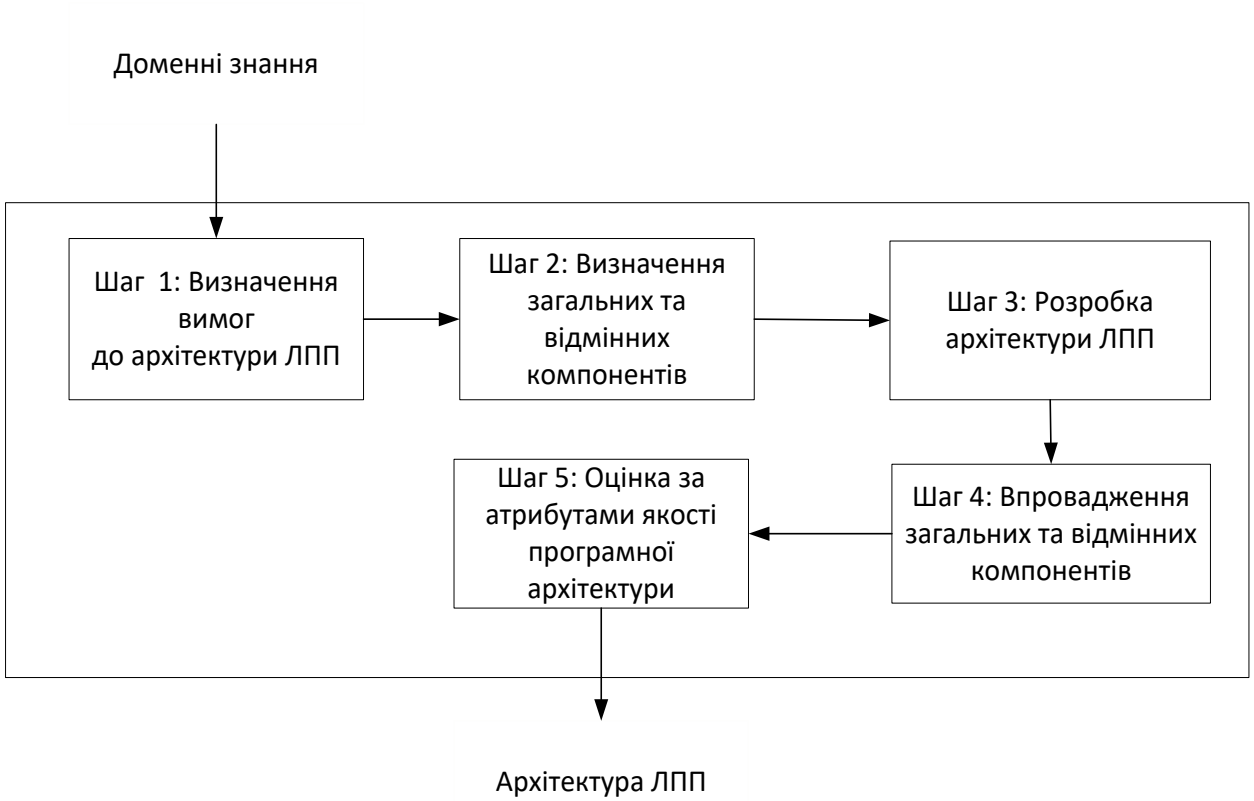

Рисунок 1.6 - Процес розробки нової ЛПП [55]

Такий процес складається з таких основних етапів:

1) Визначення вимог до архітектури ЛПП;
2) Визначення загальних та відмінних компонентів;
3) Розробка архітектури ЛПП;
4) Впровадження загальних та відмінних компонентів;
5) Оцінка за атрибутами якості програмної архітектури.

При такому процесі розробки ЛПП виникають рід проблем, а саме:

- Як коректно врахувати певні властивості та задачі у ПрО?
- Як зменшити проектні витрати?
- Як обрати найбільш ефективну технологію для підтримки процесу розробки ЛПП?

В розділі 1.1 даної роботи також зазначалося, що однією із актуальних науково-технічних проблем у розробці ЛПП є трансформація успадкованих програмних систем у ЛПП [56]. Загальна схема такого процесу представлена на рисунку 1.7.

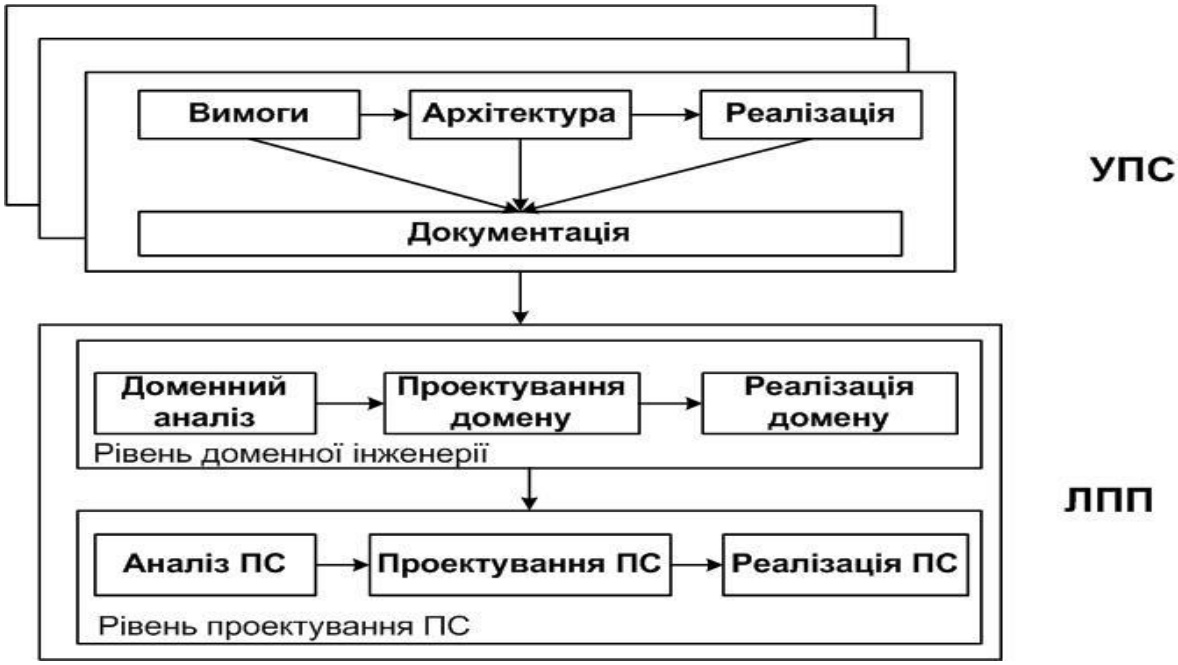

Рисунок 1.7 – Загальний процес розробки ЛПП на основі УПС [56].

Аналізуючи цей процес розробки ЛПП, визначається основна проблема (задача): для ефективної розробки ЛПП необхідні повторно використовувані рішення\артефакти.

Повторне використання ПЗ (software reuse) [22] представляє собою процес створення ПЗ із використанням вже існуючих проектних активів (project asset): специфікацій вимог, еталонних архітектур, патернів, програмного коду та ін.

Повторне використання (ПВ) як окремий об’єкт для вивчення вперше простежується в роботах M. D. McIlroy [57]. В його роботах це розглядалося в контексті ідеї «масового виробництва компонентів ПЗ» (mass produced software components) як основа для розробки прикладних програмних систем. Девід Парнас [58] в своїх роботах одним із перших запропонував ідею сімейств

програмних систем, яка стала інженерною основою для розробки багаторазових компонентів та розробки ПЗ на основі повторного використання. Такий науковець як James Neighbors запропонував концепт домену та доменного аналізу, а також був автором такого методу доменного моделювання як Draco [59, 60]. В подальшому ці питання розглядалися в роботах таких вчених як Atkinson [61] , Frakes [62], Prieto-Diaz[63], Kang Lee [64], Donohoe [65].

На рисунку 1.8 показано розвиток концепції повторного використання в розробці ПЗ від повторного використання коду до концепції ЛПП шляхом повторного використання, наприклад, шаблонів дизайну та архітектурних моделей.

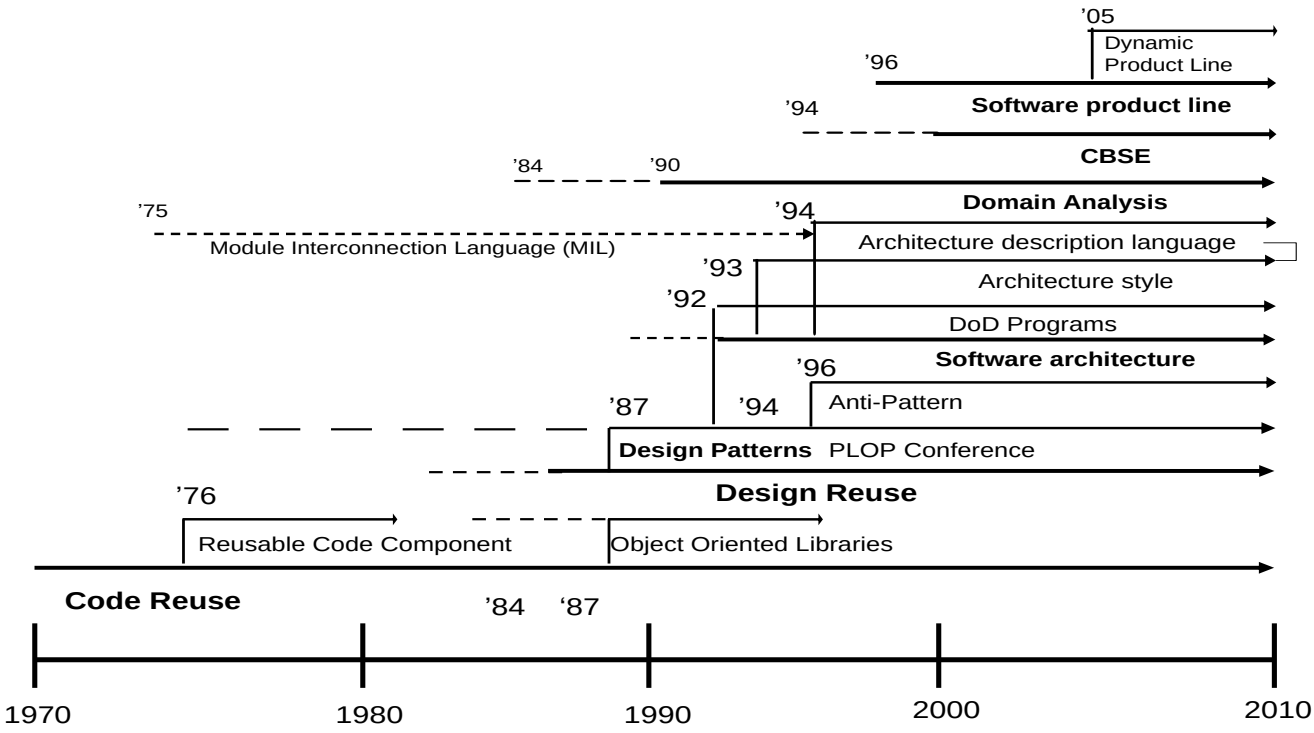

Рисунок 1.8 – Еволюція концепції повторного використання в розробці ПЗ [66]

Ключовими концепціями, що виникають і розвиваються в той період є систематичне повторне використання, аналіз варіабельності, точки варіабельності, компонентно - орієнтована розробка ПЗ, генерувальне програмування та різні підходи до доменно-орієнтованих генераторів.

Найбільш сучасною серед них є концепція динамічних ЛПП з можливістю автоматичного внесення змін згідно точок варіабельності, що забезпечує динамічну підтримку процесу розробки ЛПП.

В літературних джерелах наводяться різні класифікації типів ПВ. Зокрема у роботі [22] розрізняють наступні характеристики ПВ: Development scope, Modification, Approach reuse, Domain scope, Management scope.

Область розробки (Development scope) – визначає звідки отримані повторно використовувані компоненти (з того самого проекту або з іншого). За цією ознакою розрізняють Internal Scope та External Scope як типи у повторному використанні. Internal Scope визначає повторне використання компонентів програми у тій самій програмі, як ПВ коду з одного модуля програмного забезпечення в інший модуль. External Scope визначає ПВ при якому компонент, що повторно використовується застосовується у іншій програмній системі [66].

Модифікація (Modification) – визначає наскільки компонент повторного використання потребує змін. За цією ознакою розрізняють Black Box та White Box як типи у повторному використанні. Black Box визначає ПВ при якому компонент, що повторно використовується не піддається змінам у коді, дизайні чи архітектурі. White Box визначає ПВ при якому компонент потребує змін чи доробки відповідно до потреб замовника програмного продукту [66].

Підхід (Approach Reuse) – визначає які саме технічні методи будуть застосовані для реалізації ПВ. За цією ознакою розрізняють Compositional Reuse, In-the-small та In-the-large як типи у повторному використанні. In-the-small – визначає ПВ невеликих об’єктів, таких як класів, пакетів і застосування відповідних технік для їх повторного використання. In-the-large визначає ПВ великих об’єктів, таких як підсистеми, дизайн, архітектура із застосуванням відповідних технік. Compositional Reuse уявляє собою використання існуючих компонентів як будівельних блоків для нової системи. Його можна розглядати як підхід до створення архітектури ПЗ, який на основі існуючих архітектурних ресурсів, що дозволяє створити ще більшу за масштабами архітектуру [66].

Область домену (Domain scope) [68] визначає де відбувається ПВ - у рамках одного сімейства програмних систем або між декількома сімействами. За цією ознакою розрізняють Vertical Reuse та Horizontal Reuse. Vertical Reuse передбачає ПВ в рамках одного сімейства ПС. Horizontal Reuse передбачає ПВ між декількома сімействами програмних систем.

Управління (Management) [69] визначає наскільки систематично проводиться процес ПВ. За цією ознакою розрізняють Systematic Reuse та Ad-hoc Reuse. При Systematic Reuse ПВ планується на початку створення компоненту. При Ad-hoc Reuse компонент, що повторно використовується стає таким вже коли його повністю розроблено.

Також у роботі [70] наводиться наступна, більш спрощена класифікація повторного використання, яка відокремлює наступні типи ПВ: White-Box Reusability, Black-Box Reusability, Open Issues.

Для кількісної оцінки повторного використання у розробці застосовуються різноманітні підходи та концепції. Одним із найпоширеніших шляхів кількісної оцінки ПВ є аналіз відношення нового написаного коду в конкретному програмному продукті та довжини (розміру, об’єму) коду, який був повторно використаний. Крім того існує багато модифікацій зазначеного підходу на основі оцінки повторного використання компонентів різного рівня (строки коду, модулі, класи та ін) [70].

Узагальнюючи існуючи підходи до оцінки ПВ, можна розділити їх на такі групи: аналіз відношення частки повторно використаного коду в програмному продукті, дослідження впливу на повторне використання певних об’єктно-орієнтованих метрик, побудова інтегрованої оцінки на основі емпіричних або експертних оцінок з урахуванням показників якості програмних систем та їх впливу на ПВ [67, 71].

До першої групи відносяться наступні показники:

- процент ПВ (Reuse percent - RP): визначається як співвідношення кількості повторно використаних рядків коду до загальної кількості рядків коду;

- рівень ПВ (Reuse Level - RL): визначається як співвідношення кількості повторно використаних об’єктів до загальної кількості об’єктів;
- частота ПВ (Reuse Frequency - RF): визначається як співвідношення посилань на повторне використаний об’єкт до загальної кількості посилань;
- розмір ПВ та частота (Reuse size & Frequency - RSF): є подібним до RF, але також бере до уваги розмір елементів у кількості рядків коду;
- коефіцієнт ПВ (Reuse Ratio - RR): є подібним до RP, але також розглядає частково змінені об’єкти як повторно використані;
- щільність ПВ (Reuse Density – RD): розглядається як співвідношення кількості повторно використаних частин до загальної кількості рядків коду;

та деякі інші.

До другої групи відносяться різноманітні підходи [72,73] вивчають вплив різноманітних метрик структурної складності на показник повторного використання коду. Серед таких метрик є : Weighted methods per class, Depth of inheritance tree, Number of children, Coupling between object classes, Responses for a class, Number of Template Children, Depth of Template Tree, Method Template Inheritance Factor, Attribute Template Inheritance Factor та ін.

До третьої групи належать підходи що базуються на основі оцінки атрибутів якості ПЗ та їх впливу на ПВ. У ряді джерел досліджується цей взаємозв’язок показника повторного використання та різних атрибутів якості ПЗ [21, 22, 70]. Серед них найбільш вагомими факторами відокремлюють супроводжуваність, зрозумілість, адаптивність, портабельність . Крім того в [70] визначається інтегрований показник ПВ, що уявляє собою суму показників Availability, Documentation, Complexity, Quality, Maintainability,Price, Adaptability, Reuse (в контексті частоти ПВ) із відповідними ваговими коефіцієнтами.

Недоліком першою групи показників є те, що їх можливо застосовувати лише коли система вже повністю розроблена. У то же час підходи до оцінки ПВ із другої групи можливо застосовувати і на ранньому етапі проектування ПС.

Показники з третьої групи на даний час хоча і запропоновані концептуально, але мають слабку формалізацію і реалізацію.

1.4 Постановка задачі розробки та дослідження модельно-технологічного інструментарію для оцінки ефективності застосування методів доменного моделювання при розробці та супроводі СПС та ЛПП

Метою дисертаційної роботи є підвищення ефективності використання методів та інструментальних засобів доменного моделювання у процесі розробки ЛПП шляхом розробки та застосування інформаційної технології, яка поєднує в собі моделі, експертні методи, кількісні метрики та відповідні програмні рішення. Для досягнення цієї мети в роботі необхідно вирішити такі задачі:

- проаналізувати технологічні особливості процесів розробки ЛПП, у тому числі і таких, які створюються на основі успадкованих програмних систем, з урахуванням можливостей застосування в цих процесах методів та засобів побудови ДМ;
- запропонувати формалізований підхід та розробити відповідну алгоритмічну модель для визначення ефективності застосування методів та засобів побудови ДМ в процесах розробки ЛПП;
- розробити метрики та метод для кількісної оцінки структурно-функціональної складності різних ДМ;
- розробити метод для кількісної оцінки ступеня повторного використання програмного коду, який отримано в процесі розробки ЛПП на основі побудови та опрацювання відповідної ДМ;
- запропонувати інформаційну технологію для реалізації розробленого формалізованого підходу до визначення ефективності застосування методів та інструментальних засобів побудови ДМ в процесах розробки ЛПП;

- реалізувати основні програмні компоненти інформаційної технології для практичного застосування запропонованого підходу в процесах розробки ЛПП;

- провести експериментальне дослідження працездатності запропонованого підходу та на основі аналізу отриманих результатів надати практичні рекомендації щодо підвищення ефективності використання ДМ в процесах розробки ЛПП.

1.5 Висновки по розділу

У цьому розділі дисертаційної роботи:

1) проаналізовано причини виникнення та наведена стисла характеристика сучасних методів та технологій розробки сімейств програмних систем та лінійок програмних продуктів;

2) розглянуто доменне моделювання як концептуальну основу для автоматизації процесів створення ЛПП із використанням парадигми генерувального програмування;

3) мотивована актуальність проблеми повторного використання проектних активів як засіб підвищення ефективності процесів розробки ЛПП;

4) на основі результатів проведеного аналізу сформульовані мета та задачі дисертаційного дослідження.

Список джерел, які використано у даному розділі, наведено у повному списку інформаційних джерел під номерами: [1-6, 17-73].

2 МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ МОДЕЛЬНО-ТЕХНОЛОГІЧНОГО ІНСТРУМЕНТАРІЮ ДЛЯ ОІІНКИ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ МЕТОДІВ ДОМЕННОГО МОДЕЛЮВАННЯ

2.1 Класифікація та порівняльний аналіз сучасних методів та інструментальних засобів доменного моделювання

За останні два десятиліття з’являлися і розвивалися різноманітні методи доменного моделювання (МДМ). На базі старих методів з’являлися нові, деякі методи розділялися на інші підвиди. На рис. 2.1 наведена одна з можливих класифікацій МДМ, що побудована на основі аналізу [74-75]. Основним критерієм цієї класифікації є артефакт, який повторно використовується. Згідно з цим критерієм МДМ розподіляються наступним чином:

- методи повторного використання компонентів ПЗ;
- методи повторного використання вимог до ПЗ;
- методи повторного використання архітектури ПЗ;
- методи повторного використання ресурсів/активів.

До цих методів належать [75]:

- ODM (Organization Domain Modeling) метод. Був розроблений, щоб забезпечити спільну основу для життєвого циклу доменної інженерії;
- FODA (Feature Oriented Domain Analysis) метод. Він заснований на виявленні, аналізу і документуванні основних особливостей програмних систем в області, не тільки загальних аспектів, але й відмінностей з іншими доменами;
- Synthesis представляє собою системний підхід до виявлення подібності та відмінностей між бізнес-вимогами, для їх застосування подібних властивостей у сімействах програмних систем;
- STARS Цей метод також відомий як "STARS Reuse Library Process Mode SRLPMI". Він базується на розробці компонентів, які можуть бути реалізовані та введені до бібліотеки для подальшого повторного використання. Моделі

домену виробляються у вигляді загальної архітектури або стандартних конструкцій. Повторне використання гарантується тим, що компонент низького рівня виступає в якості будівельних блоків для складання архітектури;

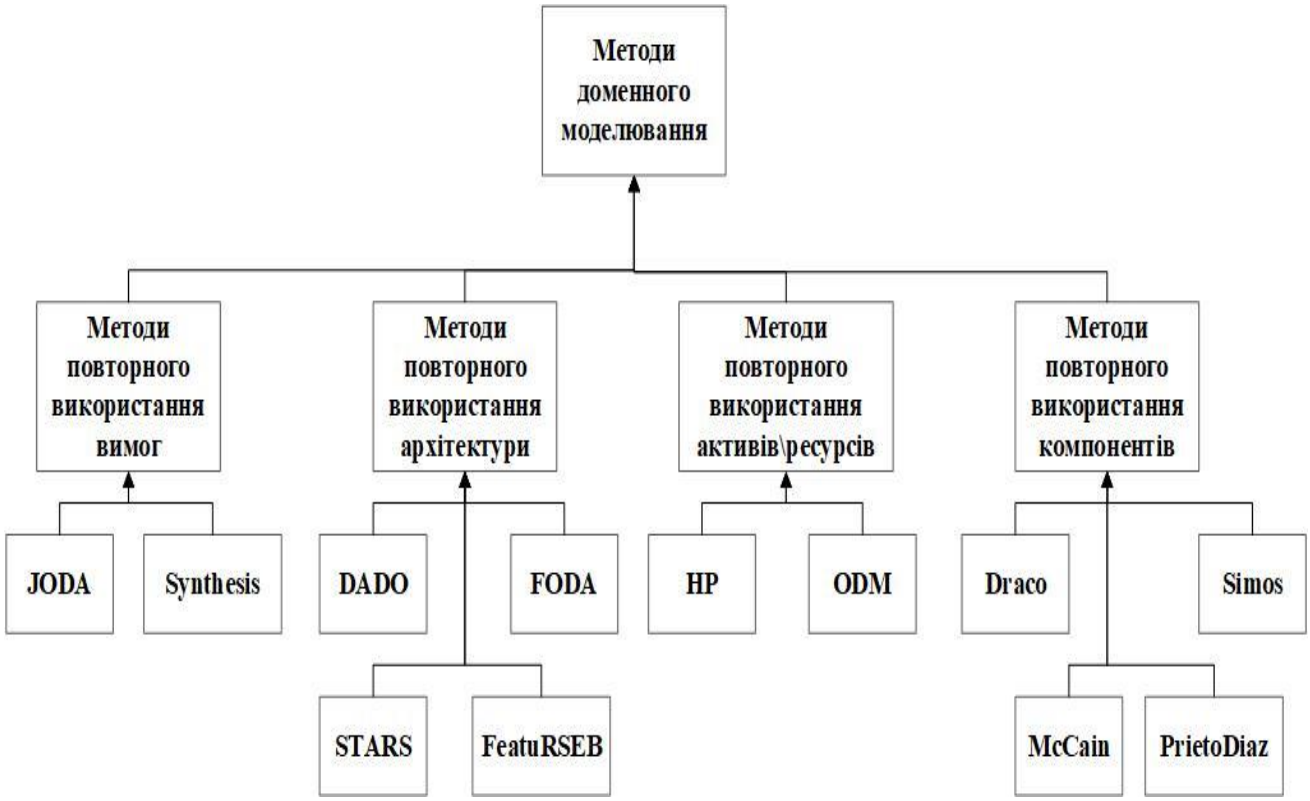

Рисунок 2.1 - Класифікація методів доменного моделювання

- FeatureRSEB поєднує концепції FODA та ODM, але створені функціональні моделі простіші, ніж ті, що містяться в оригінальному FODA, зосереджуючись на функціонально орієнтованих частинах для цілей інжинірингу домену. Архітектура та багаторазові підсистеми описуються з використанням діаграми використання, і останнім часом вони перетворені в об'єктні моделі, які можуть бути порівняно з діаграмами використання;
- HP метод розглядає збір обмежень і вимог виробників, замовників і очікуваних користувачів результатів аналізу домену та пов'язані з ним активи;
- DRACO метод базується на використанні предметно-орієнтованих мов, а саме опис специфіки домену за допомогою предметно-орієнтованої мови, опис

основних функціональностей домену за допомогою предметно орієнтованої мови та реалізація базової функціональності;

- PrietoDiaz – цей метод став основою для появи базових 3х етапів у доменному моделюванні, таких як підготовка домену, відокремлення спільностей та розбіжностей у домені та моделювання домену;

- JODA (Join Integrated Avionics Working Group Object-Oriented Domain Analysis). JODA використовує OOA / D замість функціональних методів для покриття фази аналізу предметної області.

Крім того ще існують методи доменного моделювання, які базуються на таких аспектах ПВ , як повторне використання технологій, досвіду та процесів. До таких методів відносяться:

- SHERLOCK. Це практика постачальницької лінії, яка використовує OOA для аналізу. Цей метод використовує діаграми прецедентів і діаграми класів для моделювання;

- SODA (Strategic Options Design And Assessment). Метод забезпечує підхід до проектування довгостроковій живої архітектури системи;

- DSSA (Domain-Specific Software Architecture). Спеціалізується на конкретному типі завдань (домені), узагальнених для ефективного використання в цій області, складається в стандартизовану структуру (топологію) ефективну для побудови успішного ПЗ;

- FORM (Feature-Oriented Reuse Method). Це систематичний метод, який шукає і охоплює загальні риси і відмінності ПЗ в домені для розробки архітектури і компонентів;

- RSEB (The Reuse-Driven Software Engineering Business). Це систематичний, модель-орієнтований підхід до об’ємного потворного використання.

У дисертаційній роботі в подальшому більш детально досліджуються методи ODM та JODA. Ці методи на даний час є досить популярними у доменній інженерії, мають добре розвинутий технологічний інструментарій та є типовими представниками своїх підгруп.

2.1.1 Метод ODM (Organizational Data Modeling)

Метод ODM вперше був формалізований Mark Simos з Organon Motives Inc. і був фондований програмою ARPA STARS. ODM є найбільш ефективним, коли його використовують для підтримки проектів доменної інженерії для предметних областей, які є цілком розвинутими, стабільними й економічно життєздатними [74,76].

Основна мета ODM – систематичне перетворення програмних артефактів з успадкованих систем в повторно використовувані активи, які можуть бути корисними для майбутніх розробок. ODM можна використовувати як і для сімейства систем (вертикальні домени) так і для частин систем (горизонтальні домени). Тому метод може бути корисним як для реінжинірингу частин успадкованих систем так і для розробки нових систем.

Метод складається з трьох окремих фаз: простий доменний інжиніринг, моделювання домену та інжинірингу бази активів. Кожна фаза поділена на три підфази, кожна з яких вимагає виконання трьох завдань.

Фаза простого доменного інжинірингу сфокусована на взаєморозумінні зацікавлених осіб та визначенні меж домену. Вона складається з трьох підфаз: встановлення завдань, встановлення меж домену, визначення домену:

- підфаза встановлення завдань поєднує у собі визначення індивідуальності кожної зацікавленої особи, розуміння їх завдань, а також завдань проекту в цілому та визначення пріоритетів серед зацікавлених осіб та їх завдань;
- встановлення меж домену включає такі завдання як визначення і характеристика потенціальних доменів, визначення критеріїв вибору домену та насамкінець вибір домену для подальшої роботи;
- підфаза визначення домену складається з визначення меж домену через правила та приклади систем, які будуть включені, визначення основних властивостей домену та аналіз зв’язків між цим та іншими доменами.

Фаза доменного моделювання стосується збору і документування інформації домену, що нас цікавить. Вона складається з трьох підфаз: отримання інформації домену, опис домену, удосконалення домену:

- отримання інформації домену супроводжується спочатку плануванням завдань, а потім збору інформації від експертів домену, користувачів системи, існуючої успадкованої документації тощо. Ця підфаза завершується, коли інформація інтегрована і найбільш необхідні властивості системи були визначені;

- підфаза опису домену спочатку передбачає розробку глосарію домену, що визначає конкретну мову домену. Наступними кроками є моделювання семантики ключових понять домену, а потім моделювання різниці цих понять через визначення та представлення властивостей;

- удосконалення домену включає інтеграцію існуючих моделей в узгоджену загальну модель, моделювання компромісів (чому певні властивості використовуються або не використовуються) та насамкінець кластеризації і експериментування з різними комбінаціями властивостей.

Фаза інжинірингу бази активів – фінальна фаза методу ODM, що складається з визначення меж, створення архітектури та реалізації бази активів для домену, що є цільовим:

- визначення меж бази активів складається з установлення співвідношення властивостей з клієнтами, потім встановлення пріоритетів між ними та вибору властивостей, що будуть реалізовані;

- створення архітектури бази активів супроводжується визначенням зовнішніх і внутрішніх архітектурних обмежень і визначення архітектури базуючись на цьому;

- реалізація бази активів складається з планування реалізації, реалізації активів та наприкінці реалізації інфраструктури, що включає механізми вибори і кваліфікації активів.

2.1.2 Метод JODA (Joint Integrated Avionics Working Group Object-Oriented Domain Analysis)

Метод JODA застосовує об’єктно-орієнтований аналіз та проектування замість функціональних методів для виконання фази доменного аналізу. Об’єктно – орієнтовані техніки аналізу використовуються для визначення структури та фіксації вимог [74,77].

До переваг методу можна віднести те, що об'єкти є більш постійними по своїй суті та більш стійкими до змін, аніж функції (у відповідних методах), тобто нейтралізується недолік пов’язаний з обмеженнями на характер домену.

Фаза аналізу домену визначає, що конкретно буде повторно використане, як це може бути структуровано та як це може бути використане. Вона включає в себе 3 фази нижчого рівня.

Фаза 1: Підготовка домену: ідентифікація та збір відповідних джерел даних, посилань та програмних артефактів відповідних домену. Результатом є структуровані джерела даних домену та підтримка від експертів домену.

Фаза 2: Визначення домену: на цій фазі створюються діаграми вищого рівня сутностей, генералізації – спеціалізації, сервісів та залежностей домену, глосарій домену та текстовий опис.

Фаза 3: Моделювання домену: єдина фаза, яка використовує ООА та складається з наступних етапів:

– визначення властивостей та сервісів;

– ідентифікації об'єктів та їх зв'язків / залежностей;

– ідентифікація, визначення та моделювання сценаріїв домену;

– абстракція та групування об'єктів таким чином, щоб модель була пристосована до наступного використання в процесі доменної інженерії.

Загальний процес розробки ПЗ за цим методом складається з таких кроків: бізнес планування, методологічне планування, доменний інжиніринг та інжиніринг застосування. Мета бізнес планування – визначити і обрати високорівневі домени, що будуть розглянуті при доменному аналізі. Критерії

визначення домену наступні: домен повинен бути добре зрозумілим та технологічно передбачуваний. Ризик проведення доменного аналізу теж має бути врахований. Метою методологічного планування є визначення набору методів доменного інжинірингу, що сумісні з методами інжинірингу застосувань. Якщо ці методи не є сумісними, то знання домену і програмні об’єкти не зможуть бути повторно використані.

2.1.3 Аналіз інструментальних засобів предметно-орієнтованого проектування

За останній час інструментальні засоби візуального моделювання значно еволюціонували. Раніше домінували засоби, що підтримують стандарт UML, але зараз акцент робиться на розробку і використання DSM-рішень [26,78]. Як вже зазначалося вище, предметно-орієнтоване моделювання є підходом до розробки ПЗ, який підвищує рівень абстракції використовуваних концепцій. Цей підхід дозволяє створювати ПЗ у термінах цільової ПрО, при цьому програмний код генерується автоматично. Останнє є можливим, оскільки і мова моделювання, і генератор коду створені спеціально для даної області.

Більшість існуючих технології підтримки DSM-підходу призначені для розробки повноцінних систем візуального моделювання і дозволяють визначати абстрактний і конкретний синтаксиси нових метамоделей, автоматично створювати для цих метамоделей редактори та інші інструменти підтримки.

Для того щоб провести класифікацію засобів доменного моделювання було обрано ряд критеріїв, які є найбільш вагомими у процесі розробки ЛПП з використанням проблемно-орієнтованого проектування. Ці критерії визначаються наступними характеристиками засобів:

- підтримувані мови (як моделювання, так і програмного коду);
- наявність обов'язкової оплачуваної ліцензії;
- можливість генерувати програмний код на основі діаграм;

- можливість генерації діаграм, маючи програмний код.

Результати аналізу засобів доменного проектування представлені в таблиці 2.1. Для подальшого дослідження та експериментальних дослідів було обрано інструментальний засіб EMF для імплементації методу ODM, та відповідно засіб Actifsource для імплементації методу JODA.

Таблиця 2.1 – Порівняння деяких інструментальних засобів доменного моделювання

	Підтримувані мови							*Обов'язкова платна ліцензія*	*Підтримка генерації коду за моделлю*	*Підтримка генерації моделі по коду*
	Моделювання				Програмний код					
	Feature	UML	ER	XML	Java	C++	C#			
Eclipse Modeling Framework	+	+	-	+	+	-	-	-	+	-
Rational Rose	-	+	+	+	+	+	-	+	+	+
Visual Paradigm	-	+	-	-	+	+	-	+/-	+	+
CA ERwin	-	-	+	+	-	-	-	+/-	-	-
FeatureIDE	+	+	-	+	+	+	-	-	+	+
Actifsource	-	+	-	+	+	+	+	-	+	+

Позначення в Табл. 2.1: «-» - властивість відсутня; «+» - доступна у вільному користуванні, або існує; «+/-» - платна ліцензія, або знаходиться в стадії розробки.

Проект EMF [79,80] – це фреймворк для моделювання ПЗ, а також засіб для генерації програмного коду для побудови застосувань, що базуються на моделі даних. Зі специфікації моделі описаної в форматі XMI, EMF надає засоби і підтримку часу виконання для того, щоб отримати набір Java-класів для моделі разом з набором класів-адаптерів, що дозволяють представляти і редагувати модель на основі команд і базовий редактор.

Ядро EMF побудовано на підставі загального стандарту для моделей даних, на якому базуються багато технологій і фреймворків. Це включає серверні рішення, persistence-фреймворки, фреймворки для розробки інтерфейсу користувача та підтримку трансформацій. Підтримуються три рівні генерації коду:

- модель – надає Java-інтерфейси та класи-реалізації для всіх класів моделі, а також класи-фабрики і мета-клас реалізації пакету;
- адаптери – генерують класи-реалізації (ItemProviders), що адаптують класи моделі для редагування та відображення;
- редактор – надає правильно структурований редактор, що відповідає рекомендованому стилю для редакторів моделей Eclipse EMF і служить відправною точкою для налаштування.

Усі генератори підтримують регенерацію коду у той час як зберігають зміни зроблені користувачем. Генератори можуть бути запущені або засобами графічного інтерфейсу користувача або через командний рядок.

Eclipse Modeling Project (EMP) [79] складається більш ніж з десятка різноманітних проектів для візуального моделювання (рис.2.4). Центральним проектом у Eclipse Modeling Project є EMF, один із основних призначень якого є реалізація мета-моделей візуальних мов. Інші проекти наслідують властивості EMF для різноманітної обробки мета-моделей.

Нижче перераховані проекти, що входять до складу Eclipse Modeling Project [79] (рисунок 2.2). Приклади діаграм Eclipse Modeling Project приведені на рисунках 2.3 - 2.4.

Проект GMF (Graphical Modeling Framework) слугує для реалізації графічного редактора нової мови. Але при цьому немає необхідності від розробника у написанні додаткового коду, використовуючи EMF як засіб для роботи та зберігання моделей.

Проекти M2M та M2T забезпечують можливості трансформації моделі. При цьому M2M раціє із моделями створеними засобами EMF. У той час коли M2T трансформує моделі, що задані текстовим описом чи документацією.

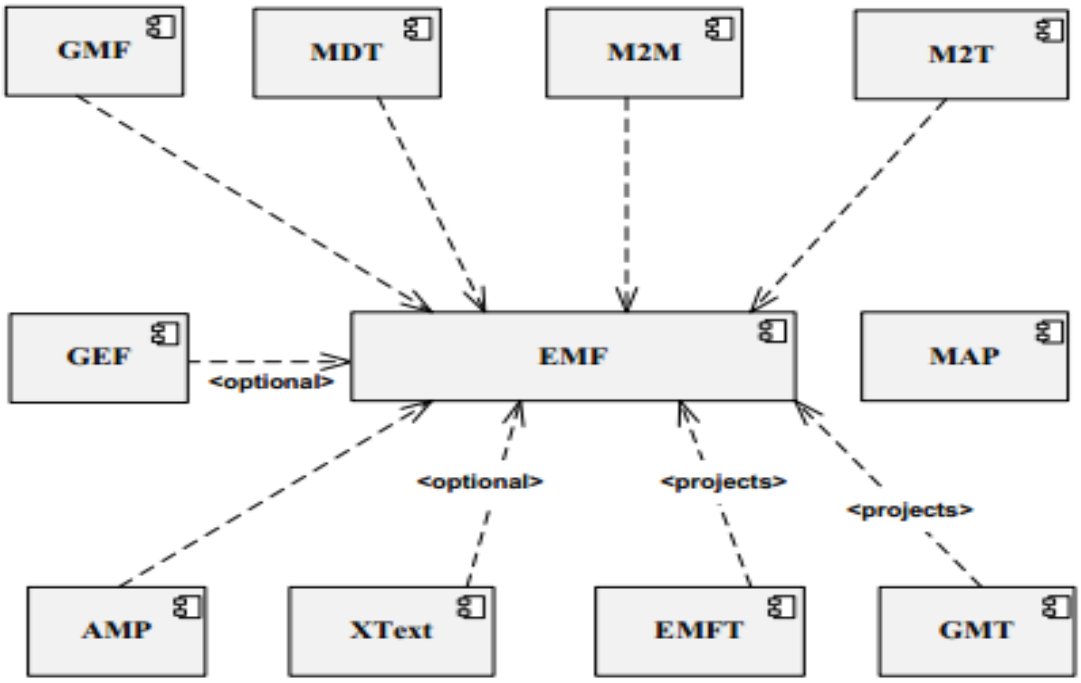

Рисунок 2.2 - Схема Eclipse Modeling Project [79]

Проект AMP (Agent Modeling Platform) забезпечує розширенні структури та інструменти для представлення, редагування, створення, виконання та візуалізації моделей на основі агентно-орієнтовани моделей (agent-based models - ABM) та будь-якого іншого домену, що вимагає просторових, поведінкових та функціональних функцій(завдання агентних мереж, симуляцію, генерацію цільового коду).

Проект MDT (Model Development Tools) забезпечує реалізацію галузевих стандартних метамоделей і надає інструменти для розробки моделей на основі цих метамоделей(UML та ін.).

Проект XText присвячений розробці текстових предметно-орієнтованих мов. моделями, і може як використовувати EMF, так і обходитися без нього (на рисунку це позначено стереотипом <optional> на зв'язку з EMF).

Проект GEF (Graphical Editing Framework) забезпечує реалізацію графічних елементів в Eclipse Modeling Project. Він більш поширений як самостійний засіб, але також він може інтегруватися із Eclipse Modeling Project для взаємодії з іншими проектами Eclipse Modeling Project.

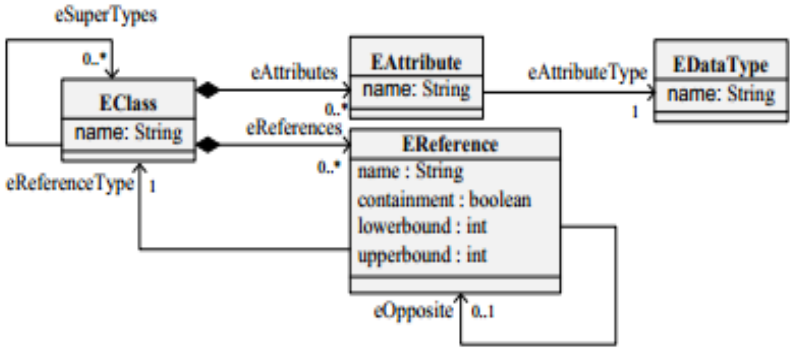

Рисунок 2.3 – Приклад Ecore метамоделі [79]

MFT (Model Focusing Tools) є додатковим проектом для забезпечення при моделювання функцію фокусування на конкретному і потрібному саме в цей час, об’єкті.

GMT (Generative Modeling Technologies) є додатковим проектом Eclipse, що забезпечує створення прототипів для Model Driven Engineering (MDE).

Проект MAP є службовим і призначений для більш зручного розгортання компонент проектів, що входять до складу Eclipse Modeling Project.

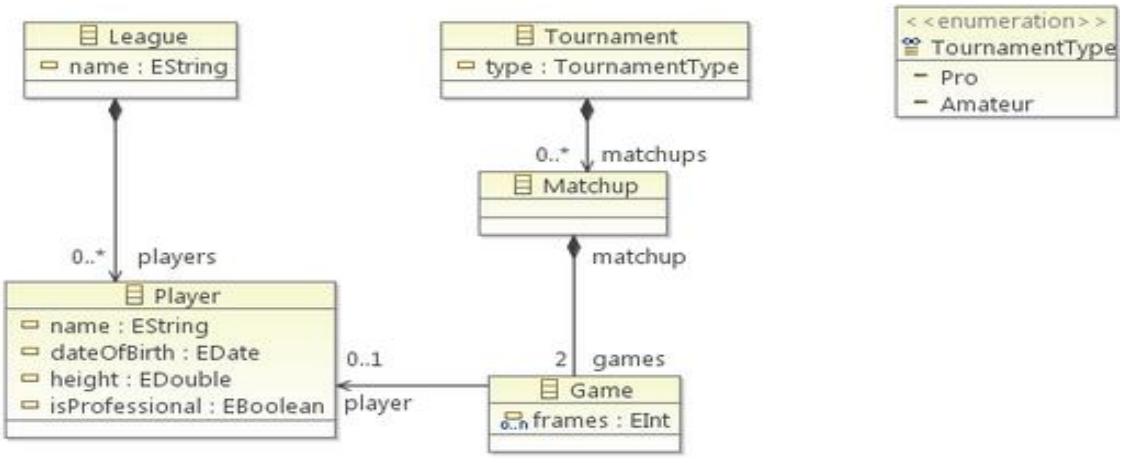

Рисунок 2.4 – Приклад доменної моделі в EMF

CASE - засіб Actifsource [81, 82] є програмним середовищем для дизайну ПЗ та генерації вихідного коду, який дозволяє виконувати аналіз домену розробки включаючи дизайн моделей, тестування, рефакторинг та супровід ПЗ

(див. рисунок 2.5). Приклад його візуального інтерфейсу наведений на рисунках 2.6– 2.7.

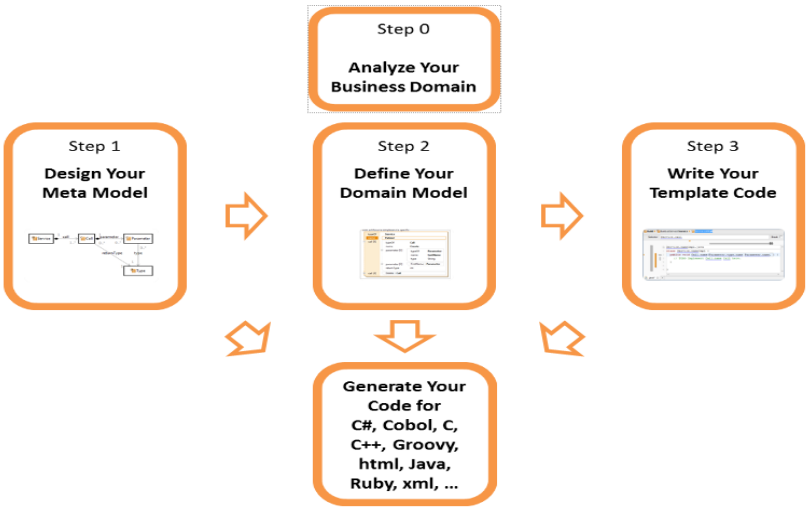

Рисунок 2.5 – Схема можливостей засобу Actifsource [81]

Так як все залежить від метамоделі, розробка повинна починатися з аналізу бізнес-домену. Коли метамодель вже спроектована, є можливим задавати специфікацію. Також разом з цим створюється шаблонний код. З цих трьох моделей (метамоделі, доменної моделі та шаблону коду) стає можлива генерація коду генератором Actifsource [82].

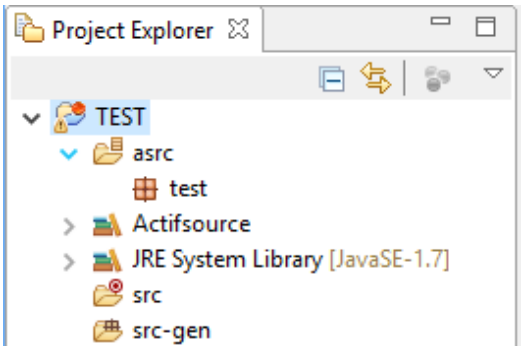

Рисунок 2.6 – Фрагмента дерева Actifsource-проекту [81]

Actifsource реалізується як плагін для середовища розробки програмного забезпечення Eclipse. Він підтримує створення декількох доменних моделей, які можуть бути пов'язані одна з одною.

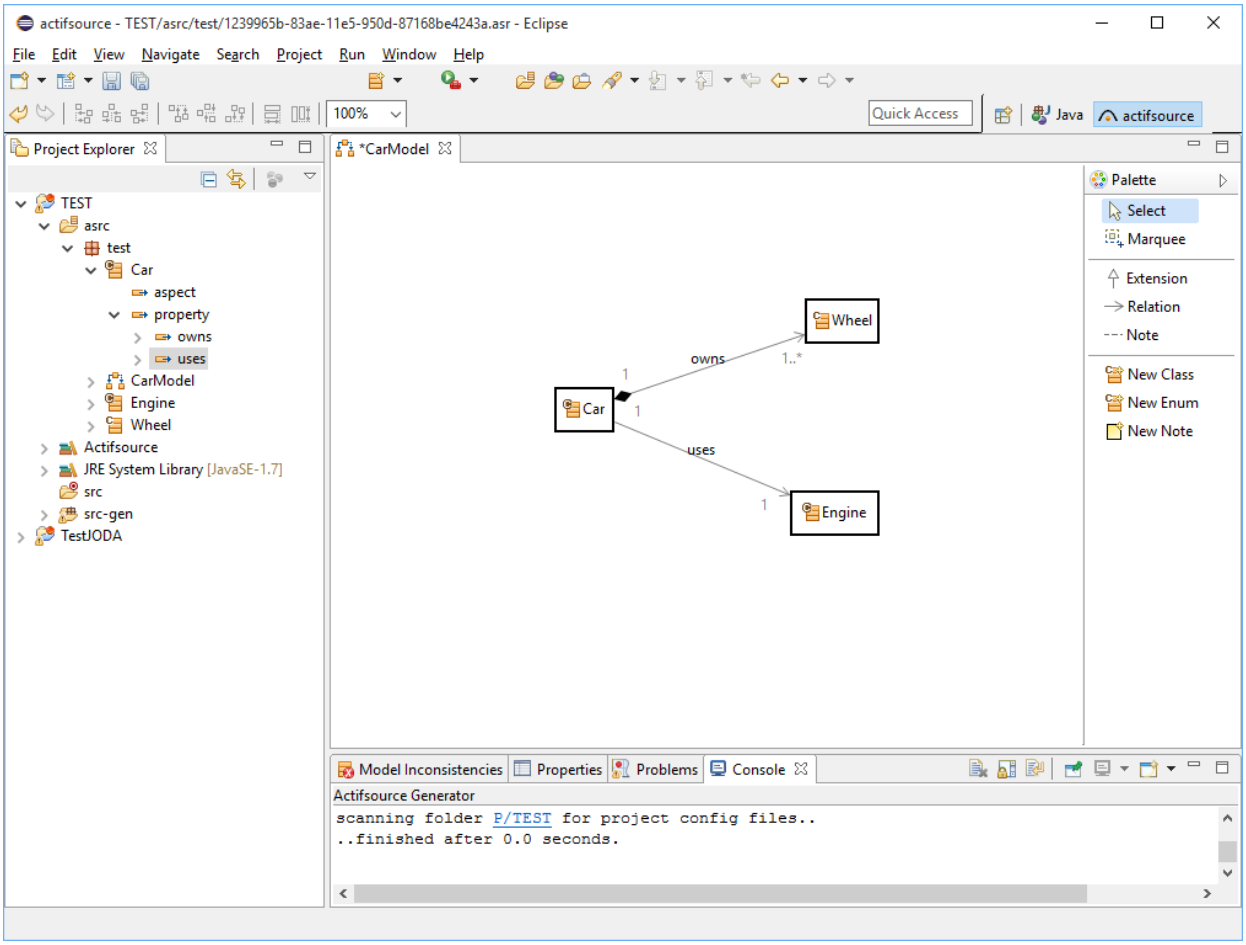

Рисунок 2.7 – Приклад доменної моделі в Actifsource

Він поставляється з графічним редактором UML-типу для створення доменних мов і загального графічного редактора для редагування структур на створених мовами. Він підтримує генерацію коду з використанням шаблонів визначених користувачем, які безпосередньо пов'язані з моделями домену. Генерація коду інтегрована в процес інкрементной збірки Eclipse.

Actifsource забезпечує складну підтримку багатомодельності. Можна почати з маленьких моделей в уже існуючому проекті і розвивати його зв’язуючи моделі крок за кроком. Actifsource представляє модельно-орієнтовану розробку у контексті доменного моделювання розробку програмного забезпечення у проектах з мінімальним ризиком та швидкою вигодою.

2.2 Формалізація задачі визначення ефективності застосування методів та засобів доменного моделювання у процесах супроводу СПС та ЛПП

Для комплексного та ефективного опрацювання окремих задач дисертаційного дослідження, які були сформульовані у п 1.4 попереднього розділу, необхідно провести формалізацію загальної постановки задач визначення ефективності застосування технологій доменного проектування при розробці ЛПП та запропонувати певний методологічний підхід для її вирішення.

З метою формалізації основної задач дослідження пропонується ввести наступні припущення та визначення.

Припущення 1. Існує певна множина методів доменного моделювання (множина M):

$$(DMM)_i \in M, i = 1,2,3..., \quad (2.1)$$

де DMM (DomainModelingMethod) - це ідентифікатор, що позначає окремий метод, який може бути застосовано для побудови ДМ.

Також існує множина відповідних технологій T:

$$(DMT)_j \in T, j = 1,2,3..., \quad (2.2)$$

де DMT (DomainModelingTechnology) – це ідентифікатор, що визначає певну технологію реалізації відповідного DMM.

В результаті застосування певного DMM із відповідним DMT, для обраної ПрО можуть бути побудовані відповідні ДМ з множини D:

$$(DM)_{i,j} \in D, \quad (2.3)$$

де $DM_{i,j}$ (DomainModel) – це ДМ, яка була отримана в результаті застосування і-го методу доменного моделювання та j-ї технології його реалізації.

Припущення 2. Доменні моделі з множини D мають різний рівень складності, тобто існує таке відображення ρ:

$$\rho : D \to DMC, \tag{2.4}$$

де DMC (DomainModelComplexity) - це множина можливих значень кількісного рівня структурно-функціональної складності доменних моделей.

Припущення 3. На основі кожної ДМ з множини D може бути отриманий відповідний каркас програмного коду, тобто існує таке відображення φ:

$$\varphi : D \to GCF, \tag{2.5}$$

де GCF (GeneratedCodeFramework) - це множина каркасів програмного коду, який може бути використаний для побудови СПС або ЛПП.

Припущення 4. Каркаси коду з множини GCF мають різний показник ступеню повторного використання програмного коду (code reusability extent - CRE) , тобто існує відображення σ:

$$\sigma : GCF \to CRE, \tag{2.6}$$

де CRE це множина можливих значень ступеню повторного використання програмного коду.

Припущення 5. Коефіцієнт ефективності застосування певної ДМ при розробці або супроводі ПЗ може бути визначений як відношення ступеню повторного використання програмного коду, який отримано із застосуванням цієї ДМ, до рівня її структурної складності, тобто:

$$K_{Eff}\left[(DM)_{i,j},(GCF)_{i,j}\right]=\frac{(CRE)_{i,j}}{(DMC)_{i,j}}, \qquad (2.7)$$

де K_{Eff} – це коефіцієнт ефективності, а змінні $(CRE)_{i,j}$ та $(DMC)_{i,j}$ визначаються за виразами (2.6) та (2.4) відповідно.

Зважаючи на доволі високу складність отримання аналітичних виразів для визначення кількісних параметрів функціональної залежності (2.7), в подальшому для створення відповідного модельно-технологічного інструментарію для оцінки ефективності застосування технологій доменного проектування в процесах розробки ЛПП доцільно використати підхід, який передбачає розробку алгоритмічних моделей (див., напр.. в [83]) та експертних методів для структурування та обробки гетерогенних інформаційних ресурсів, які є необхідними для визначення змінних, що входять до виразу (2.7).

Слід зазначити, що саме такий знання-орієнтований підхід до розробки інтелектуальних інформаційних технологій вже досить успішно застосовувався для вирішення таких слабо-структурованих задач як, наприклад, трасування вимог в гнучких процесах розробки програмних систем [84], підвищення ефективності функціонування систем управління IT-інфраструктурою організацій [85] та оцінка ефективності застосування пост об'єктно-орієнтованих технологій розробки та супроводу успадкованих програмних систем [86].

Беручи до уваги вирази (2.1) – (2.7), методологічний підхід до вирішення задачі визначення ефективності застосування технологій доменного моделювання в процесі розробки ЛПП, можна представити за допомогою метафори багатовимірного інформаційного простору (БІП). Побудова цього БІП розглянута в [8,9], а графічна інтерпретація представлено на рисунку 2.8.

У цій інтерпретації кожна вісь БІП представляє собою одну з трьох комплексних інформаційних характеристик, які є необхідними для кінцевої оцінки ефективності застосування МДМ, а саме:

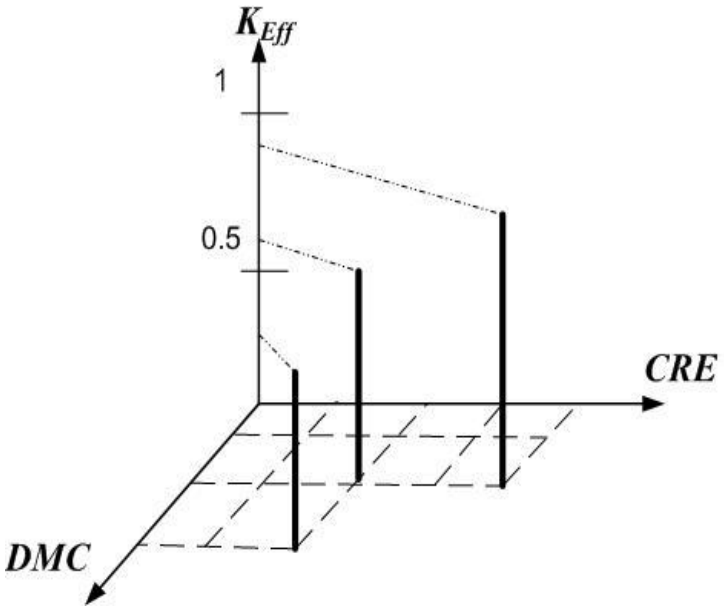

Рис. 2.8 - Геометрична інтерпретація БІП
для визначення ефективності застосування МДМ

- вісь абсцис містить показники структурно-функціональної складності доменних моделей *DMC* з їх множини D, яка визначена формулою (2.4));

- вісь ординат містить показники ступеня повторного використання *CRE* для згенерованих каркасів коду із множини *GCF* - див. вираз (2.6);

- і, нарешті, вісь аплікат репрезентує значення коефіцієнту ефективності використання відповідної технології доменного моделювання для відповідної доменної моделі - це є коефіцієнт K_{Eff}, (див. вираз (2.7)).

Використовуючи метафору БІП і запропонований знання-орієнтований методологічний підхід, в подальшому пропонується розробити сукупність моделей, експертних методів та інструментальних засобів, які дозволять отримати відповідні кількісні оцінки показників, що входять до виразу (2.7), шляхом послідовного вирішення наступних взаємопов'язаних комплексних задач [14].

Задача 1. Розробити алгоритмічну модель процесу оцінки ефективності застосування методів доменного моделювання в процесах розробки ЛПП за критерієм **Keff**.

Задача 2. Розробити метрики та процедуру для кількісної оцінки складності окремих доменних моделей $\boldsymbol{DMC}$.

Задача 3. Розробити алгоритм методу оцінки ступеня повторного використання $\boldsymbol{CRE}$.

Задача 4. Реалізувати основні програмні компоненти прикладної інформаційної технології для практичного застосування запропонованих моделей та методів доменного моделювання в процесах розробки ЛПП. Провести експериментальне дослідження запропонованого підходу.

В подальшому у цьому розділі розглянуто підходи до розробки моделей та методів які мають забезпечити ефективне вирішення наведених вище задач.

2.3 Дослідження структурно-логічних взаємозв'язків між показниками якості програмного забезпечення, метриками його структурної складності та ступенем повторного використання вихідного коду

Як вже було зазначено вище, саме проблемно-орієнтований підхід до розробки ПЗ передбачає повторне використання різноманітних проектних артефактів, що в кінцевому рахунку має на меті забезпечити достатньо високі показники якості для ПС, що розробляється. Різноманітність видів цих проектних активів та досить складний, а також слабо формалізований характер зв'язків між ними ускладнюють та практично унеможливлюють їх кількісний аналіз, що є само по собі досить складною та актуальною проблемою програмної інженерії (див., напр. в [4]).

У розділі 1.3 розглянуті чинники і фактори, які впливають на ПВ, а також їх додаткові характеристики. Серед них є такі, що більш вагомо впливають на безпосередньо процес розробки ЛПП, що є темою цього дослідження. Ці артефакти більш детально розглянуто нижче.

Для структурування релевантних артефактів у процесах повторного використання ПЗ, з можливістю подальшого якісного аналізу певних

взаємозв'язків між ними, в роботі пропонується застосувати мапи пам'яті (mind map), які на відміну від більш формалізованих нотацій, таких, як UML, IDEF0 та інші, дозволяють для будь-якої ПрО представити певні концептуальні сутності та їх семантичні зв'язки довільної природи [87]. Саме тому мапи пам'яті розглядаються деякими авторами як так звані ментальні моделі (mental model), див., наприклад, в [88], які також застосовуються для аналізу процесів програмної інженерії [20]. Така ментальна модель у вигляді мапи пам'яті для загальної класифікації та аналізу основних чинників впливу у процесі повторного використання ПЗ може бути побудована на підставі узагальнення результатів досліджень в [22,23], і вона наведена на рисунку 2.9.

Ця мапа містить концепти та їх взаємозв'язки, що якісно описують будь-який процес повторного використання (Reuse), а саме:

- концепт *область розробки* (Development scope) - визначає, звідки отримані компоненти повторно використання (КПВ): з того самого проекту або з іншого;

- концепт *підхід* (Approach) - визначає, які саме технічні методи будуть застосовані для реалізації КПВ;

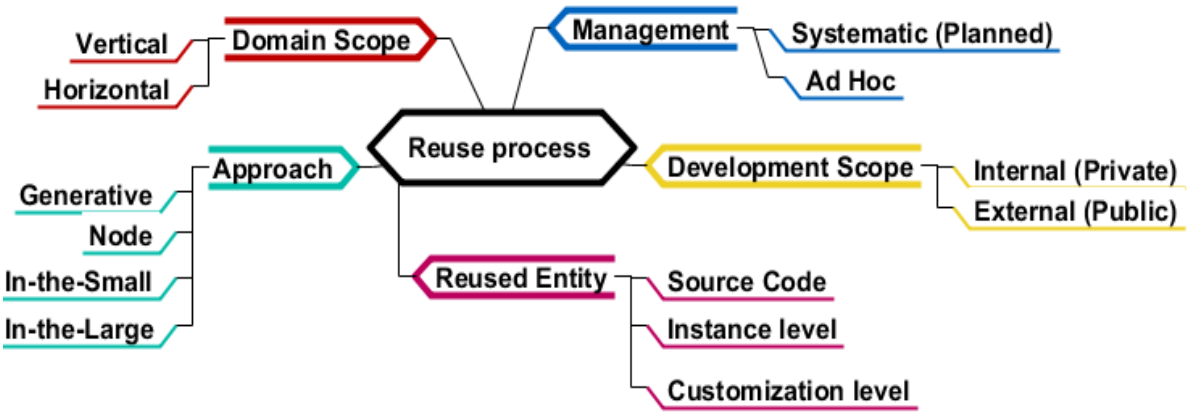

Рисунок 2.9 - Структурна схема концептів процесу повторного використання ПЗ

- концепт *область домену* (Domain scope) – визначає, де саме відбувається ПВ: у межах КПВ однієї ЛПП або між декількома ЛПП;

- концепт *управління* (Management) визначає наскільки систематично проводиться процес ПВ;

- концепт *повторно використана сутність* (Reused entity) – визначає рівень та тип КПВ, серед яких для подальшого дослідження найбільш важливим є КПВ на рівні вихідного коду (Source Code).

Аналіз сучасних публікацій, присвячених дослідженню проблеми оцінки та підвищення ступеня CRE при розробці ПС показує, що досить складно знайти зв'язок між певним рівнем CRE та різними факторами впливу в розробці ПЗ. Слід зазначити, що на теперішній час існує певна множина альтернативних підходів до кількісного визначення показника ПВ.

Систематизуючи наявні підходи та метрики для кількісного визначення рівня ПВ коду, їх можливо умовно розділити на 2 групи:

- метрики та методи на основі аналізу повного обсягу коду;

- метрики та методи на основі метрик структурної складності коду (ООП – метрики).

Так, зокрема, в [23] представлені результати емпіричного дослідження рівня CRE в ПС з відкритим кодом і наведено набір метрик щодо його оцінки. В роботах [22, 71] розглядаються різні підходи щодо оцінки CRE. Ряд із них розглядають CRE як відношення повторно використаних компонентів (коду) до загального обсягу компонентів (коду) та їх модифікацій (наприклад Reuse Percentage, Reuse level, Reuse size та ін). Але їх недолік полягає у тому, що вони можуть бути застосовані лише етапі супроводу (коли вже певні артефакти були повторно використані), і не має можливості застосувати їх на етапі проектування. Низка інших авторів [72,73] розглядає оцінку CRE із застосуванням ООП-метрик (та їх модифікацій). Особливо впливовими для оцінки ступеня CRE визнаються метрики таких груп як зв’язність, зчеплення та глибина дерева успадкування.

В даному дослідженні для визначення рівня повторного використання застосовано саме ООП-метрики. Їх основна перевага полягає у тому, шо їх можливо застосовувати на ранній стадії проектування ЛПП.

Наступним кроком у цьому дослідженні є якісний аналіз взаємозв’язків між здатністю ПЗ до повторного використання (Reusability), такими показниками якості ПЗ як супроводжуваність (Maintainability), адаптивність (Adaptability) та зрозумілість (Understandability), а також показниками його структурної складності. Відповідна мапа пам’яті для їх якісного аналізу представлена на рис.2.9. З неї можна зробити однозначний висновок відносно того, що вищезазначені показники якості ПЗ, а таким чином, і його здатність до повторного використання, залежать від рівня структурної складності відповідної ПС. Як відомо, цей показник для ПЗ, що розробляється на основі об’єктно-орієнтованого підходу, визначається за допомогою добре відомих метрик [21], а саме (див. рис. 2.10).

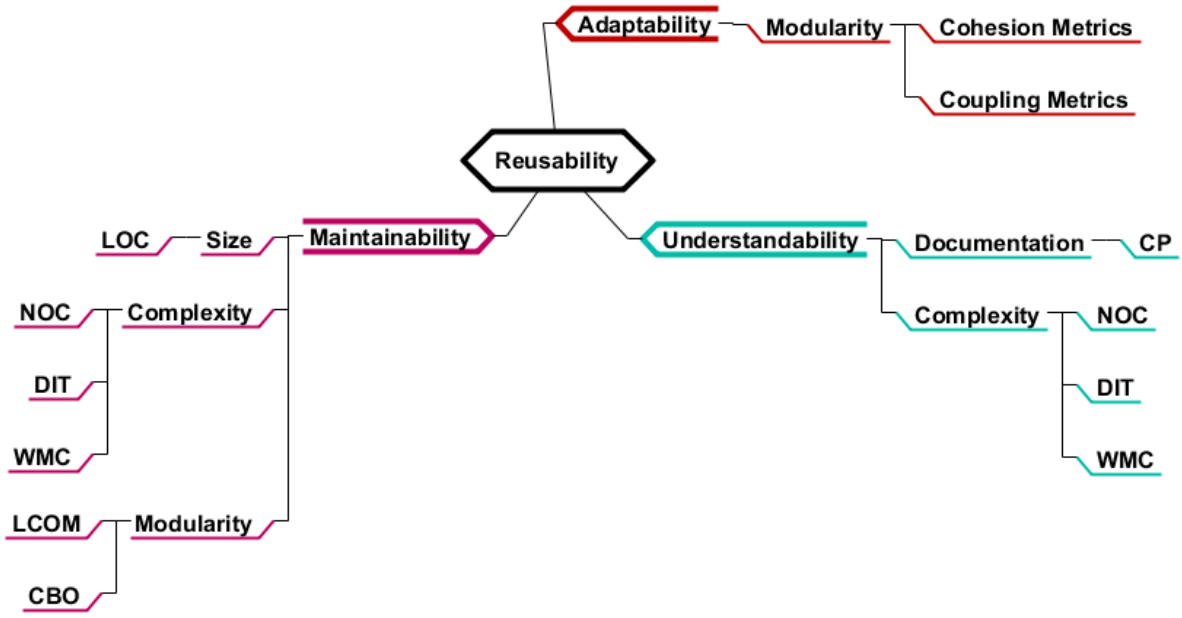

Рисунок 2.10 - Структурно-функціональний взаємозв'язок показників супроводжуваності та метрик структурної складності коду

При цьому слід окремо зазначити [12], що при розробці ПЗ із застосуванням підходу на основі DDD, на теперішній час ще недостатньо проаналізовано вплив окремих методів доменного моделювання на складність відповідного програмного коду, який генерується на основі відповідної

доменної моделі. Тому є досить важливим завданням визначити цю кореляцію, щоб зменшити витрати на реалізацію DDD - орієнтованих програмних проектів, і зокрема, на створення ЛПП.

2.4 Концептуальні схеми розробки та реінжинірингу СПС та ЛПП з використанням методів доменного моделювання

Для концептуалізації процесу розробки ЛПП на основі методів та засобів доменного моделювання запропоновано наступну схему (див. рис. 2.11). Процес розробки ЛПП пропонується розглядати як функціонування системи автоматизованого управління з контуром зворотного зв'язку, в якому як критерій оцінки якості управління використовується коефіцієнт ефективності застосування технологій доменного моделювання.

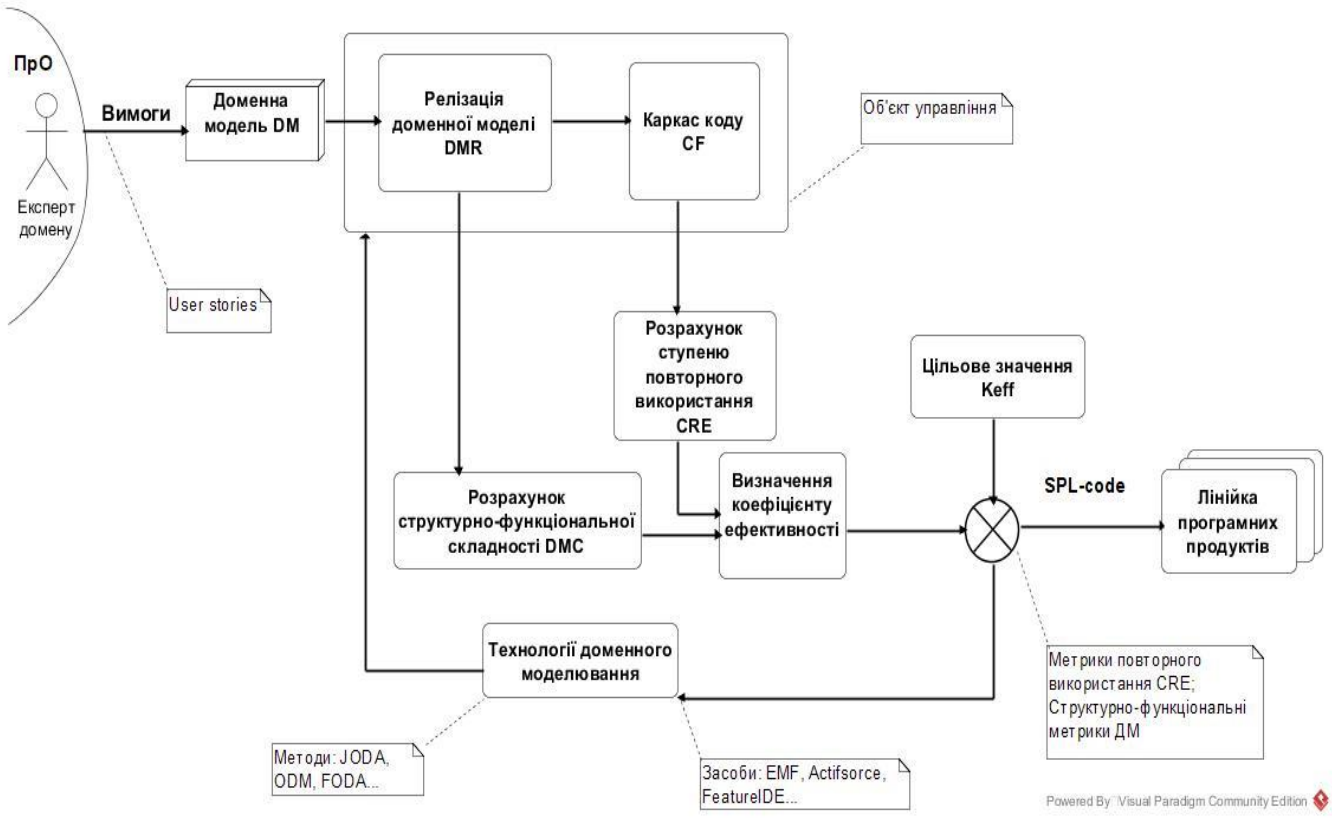

Рисунок 2.11 - Концептуальна схема розробки ЛПП «з нуля» із використанням технологій доменного моделювання

Основні її функціональні блоки взаємодіють у наступний спосіб:

- первинний опис певної ПрО тобто бізнес-вимоги користувачів (User stories) до функціональності майбутньої ПС слугує інформаційним базисом для побудови доменної моделі (DM) на концептуальному рівні;

- за допомогою відповідних (наявних) методів доменного моделювання (domain modeling method – DMM), напр.: JODA, ODM, FODA та ін. та відповідних CASE-засобів їх інструментальної підтримки (DMT – Domain Modeling Technology), таких як EMF, FeatureIDE, Actifsource, та ін., створюється реалізація доменної моделі (domain model realization – DMR) та генерується відповідний каркас програмного коду;

- каркас програмного коду GCF після певних доробок (напр., із застосуванням відповідних патернів кодування), може бути використано для побудови компонентів цільової лінійки програмних продуктів;

- побудована реалізація доменної моделі DMR є основними вхідними даними для визначення кількісної оцінки структурно-функціональної складності доменної моделі DMC;

- за каркасом програмного коду GCF визначається ступень повторного використання коду CRE;

- кількісні результати CRE та DMC є основними параметрами для визначення коефіцієнту ефективності застосування технологій доменного моделювання Keff;

- після отримання кількісного значення Keff приймається рішення щодо його відповідності цільовому значенню. Слід зазначити, що цільове значення Keff може бути отримане двома шляхами – як постійне значення , визначене експертами; шляхом порівняння із попередніми значеннями Keff від інших технологій і обрання найбільшого. В другому випадку прийняття рішення можливе лише після другого циклу функціонування системи управління;

- механізм зворотного зв’язку представлено застосуванням альтернативної технології доменного моделювання. Тобто до об’єкту

управління (шо представляє собою DMR та GCF) застосовуються альтернативний метод та відповідний CASE- засіб доменного моделювання;

- в той час, коли критерій Keff досягає необхідного значення поточна технологія доменного моделювання обирається як найбільш ефективна для поточно ПрО. Обрана технологія використовується для отримання коду ЛПП.

Слід зазначити, що концептуальна схема на рис. 2.11 не виключає можливості застосування інших метрик якості програмного коду, - для аналізу ефективності різних варіантів побудови ЛПП, і в цьому сенсі вона може розглядатися як удосконалення вже існуючих підходів до вирішення загальної проблеми варіабельності (variability) при розробці ЛПП.

Але на даний час окрім розробки ЛПП «з нуля», іншою важливою технічною задачею є розробка ЛПП на основі успадкованої програмної системи (УПС). У [37] розглядається процес отримання ЛПП на основі УПС, але при цьому не беруться до уваги деякі важливі чинники, що мають істотний вплив на структуру та якість компонентів майбутньої ЛПП. Одним з таких чинників, який має суттєвий позитивний вплив на ефективність процесу побудови ЛПП, є ступінь повторного використання програмного коду CRE в окремих компонентах ЛПП та структурно-функціональна складність доменної моделі УПС.

Створення ЛПП представляє собою не тільки технологічну проблему, а також складну організаційну задачу, яка пов’язана з необхідністю брати до уваги такі фактори як різні методи моделювання ПрО, механізми аналізу та відновлення попередніх вимог та деякі інші. В [89, 90] було запропоновано механізм управління вимогами з використанням адаптивної матриці трасування (ADTM), що дозволяє зв’язати вимоги до ПЗ з програмними артефактами (вихідним кодом) та надає можливість простежити відповідні зміни.

Враховуючи висновки, зроблені вище, можна запропонувати керований процес трансформації вже існуючої УПС в відповідну ЛПП шляхом застосування методів доменного моделювання для подальшого генеруванням програмних компонентів ЛПП із певним рівнем повторного використання коду

CRE. Концептуальна схема цього підходу наведена на рис. 2.12, який виконано без застосування будь-якої стандартної нотації (UML, IDEF0 або інших) для того, щоб підкреслити досить високий рівень абстракції такого представлення.

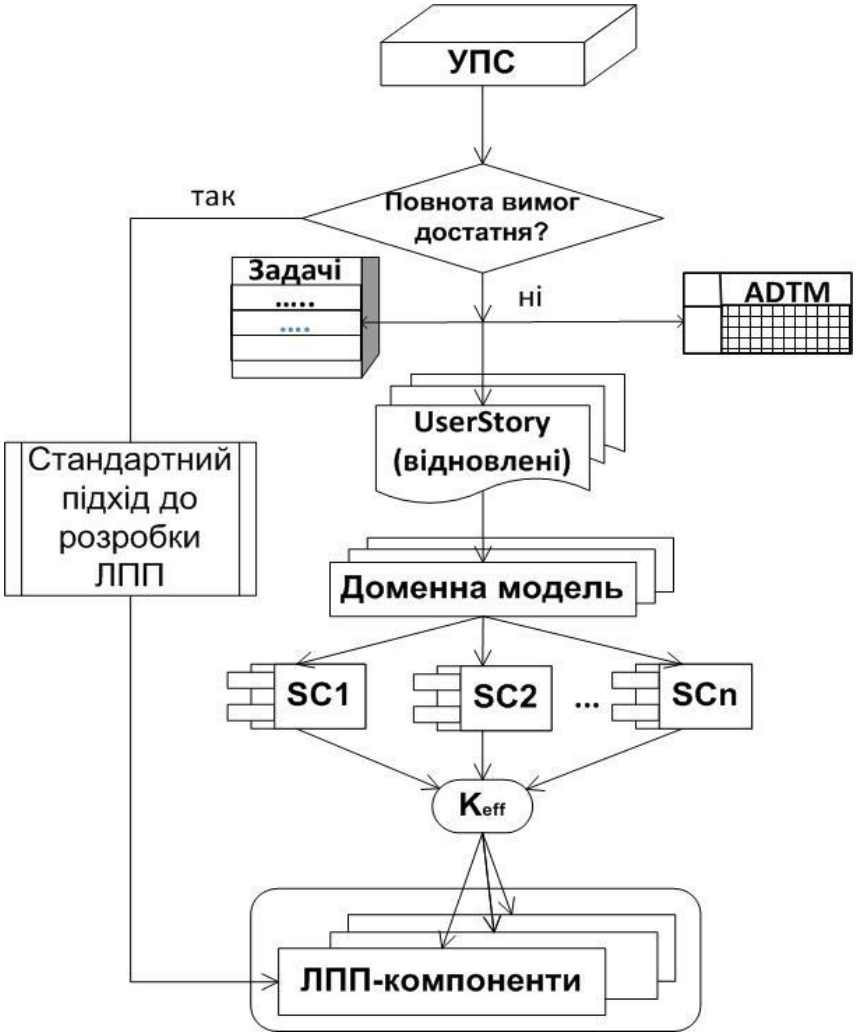

Рисунок 2.12 - Концептуальна схема розробки ЛПП на основі УПС

Початковим артефактом процесу побудови ЛПП є існуюча УПС, для якої по-перше визначається повнота наявних вимог. Якщо повнота вимог є достатньо високою, то можливо безпосередньо перейти до побудови ЛПП із застосуванням стандартних процедур, у протилежному випадку пропонується виконати наступну послідовність дій відповідно до рисунку 2.12 :

- із використанням матриці ADTM шляхом ретроспективної обробки каталогу завдань (Tasks), які вже були виконані для цієї LSS, виконується реконструкція повного набору бізнес-вимог користувачів (User Story);

- за допомогою відповідних методів та інструментальних засобів створюються альтернативні варіанти доменної моделі ДМ для ПрО застосування майбутньої ЛПП;

- для кожної отриманої ДМ генерується вихідний програмний код (SorceCode - SC), для якого шляхом обчислення ООП-метрик може бути отримана оцінка для очікуваного рівня повторного використання коду CRE, також визначається відповідна структурно-функціональна складність DMC та як фінальний критерій обчислюється коефіцієнт ефективності Keff;

- на основі аналізу альтернативних варіантів DM за критерієм максимального рівня Keff. можливо обрати найбільш ефективний варіант реалізації програмних компонентів ЛПП, яка має бути побудована для відповідної ПрО.

Запропонована концептуальна схем розробки ЛПП на основі УПС дозволяє забезпечити вибір такого методу та засобу доменного моделювання, який будуть найбільш ефективним для ПрО цієї УПС із врахуванням підвищення рівня повторного використання коду її компонентів.

2.5 Висновки по розділу

У даному розділі дисертаційної роботи:

1) Наведена класифікація та проведено порівняльний аналіз сучасних методів та інструментальних засобів доменного моделювання. Розглянуто більш детально методи ODM (Organizational Data Modeling) та JODA (Joint Integrated Avionics Working Group Object-Oriented Domain Analysis);

2) Проведено порівняльний аналіз існуючих інструментальних засобів доменного моделювання з урахуванням ключових для розробки ЛПП характеристик ;

3) Для оцінки ефективності застосування технологій доменного моделювання у процесах розробки ЛПП запропоновано підхід до формалізації цієї задачі;

4) Досліджено структурно-логічні взаємозв’язки між показниками якості програмного забезпечення, метриками його структурної складності та ступенем повторного використання вихідного коду;

5) Запропоновано концептуальні схеми розробки та реінжинірингу ЛПП з використанням технологій доменного моделювання. Вони можуть бути застосовані як у разі розробки ЛПП з «нуля» так і у разі їх створення на основі використання УПС.

Список джерел, які використано у даному розділі, наведено у повному списку інформаційних джерел під номерами: [4, 8-9, 12, 14, 21-23, 26, 37, 67, 71, 74-90].

3 РОЗРОБКА МОДЕЛЕЙ, МЕТОДІВ ТА ТЕХНОЛОГІЧНИХ ПРОЦЕДУР ДЛЯ ОЦІНКИ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ МЕТОДІВ ТА ЗАСОБІВ ДОМЕННОГО МОДЕЛЮВАННЯ

3.1 Алгоритмічна модель процесу оцінки ефективності застосування методів та засобів доменного моделювання

Запропонований у другому розділі дисертації (див. п. 2.2) знання-орієнтований підхід до побудови модельно-технологічного інструментарію для дослідження ефективності застосування технологій DDD передбачає, що при розгляді особливостей процесів їх розробки та подальшого використання при розробці ЛПП необхідно враховувати взаємодію великої кількості чинників, які в свою чергу можуть бути визначені шляхом аналізу досить складних та слабоформалізованих інформаційних об’єктів. У формалізованому вигляді взаємодія цих чинників подана шляхом формулювання *Припущень 1-5* (див. п.п. 2.2 попереднього розділу), а їх геометрична інтерпретація на концептуальному рівні подана у вигляді метафори багатовимірного інформаційного простору, що представлена на рис. 2.8. Таким чином, зважаючи на ці особливості процесу визначення ефективності застосування технологій доменного проектування при розробці ЛПП, для вирішення цієї задачі може бути запропонована побудова відповідної алгоритмічної моделі (АМ), поняття про яку ефективно застосовується в таких роботах, як, наприклад, [83-86]). Ця модель з методологічної точки зору поєднує в собі окремі методи, алгоритми та метрики, що були визначені у п.п. 2.2 – 2.5 другого розділу дисертації для вирішення переліку комплексних *Задач 1-4*, а також слугує основою для розробки відповідних інструментальних засобів, які потім мають бути інтегровані у прикладну інформаційну технологію.

Функціональна залежність (2.7) – розділ 2.2 – не може бути отримана аналітично, тому для її дослідження необхідно розробити сукупність

інформаційних моделей, методів та метрик. Відповідна алгоритмічна модель (АМ) дозволяє в формалізованому вигляді представити всі ці компоненти, и вона бути подана у наступному вигляді[14]:

$$AM = \langle InfoBase.WorkFlow(Methods), Metrics \rangle, \quad (3.1)$$

де: InfoBase – інформаційний базис моделі; WorkFlow(Methods) – сукупність алгоритмів (WorkFlow) реалізації методів (Methods) оцінки ефективності застосування ДМ; Metrics - колекція метрик, що застосовуються в відповідних алгоритмах моделі АМ.

Інформаційний базис алгоритмічної моделі АМ може бути поданий у вигляді наступного кортежу:

$$InfoBase = \langle M, T, D \rangle, \quad (3.2)$$

де M – це множина методів побудови ДМ за виразом (2.1);

T – множина технологій їх реалізації за виразом (2.2);

D – множина ДМ за виразом (2.3).

До методів оцінки ефективності Methods з виразу (3.2) належать:

- метод визначення ступеня повторного використання програмного коду, який запропоновано в [12, 13];
- метод визначення показника складності ДМ, який подано за виразом (2.4) [14].

В свою чергу, колекція метрик Metrics з виразу (3.2) складається з:

a) групи метрик оцінки структурної складності програмного коду [21], які застосовуються для аналізу каркасів, отриманих шляхом генерації на основі відповідної ДМ за виразом (5);

b) множини метрик оцінки ступеня повторного використання коду [23], який потім має бути використаним для побудови відповідних СПС або ЛПП;

c) сукупності метрик оцінки складності ДМ [26,91] з множини D за виразом (2.4);

d) і, нарешті, метрики визначення коефіцієнту ефективності застосування певної ДМ при розробці або супроводі ЛПП, яка подана за виразом (2.7).

Наведені вище питання більш детально розглянуто у наступних підрозділах тексту дисертаційної роботи.

3.2 Розробка методу визначення структурно-функціональної складності доменних моделей

Основою для побудови ДМ є вимоги користувача, які у даній роботі пропонується розглядати у форматі User Stories, які застосовуються для визначення та формалізації початкових вимог домену. Історії користувача (User Stories) [92] – це високорівневий опис загальних вимог до ПС, що містить достатньо інформації для того, щоб розробники могли початково оцінити обсяг зусиль, необхідних для їх реалізації. Для формалізації представлення User Stories використовують наступний синтаксичний шаблон:

As a <type of user>, I want <some goal> so that <some reason>

Як <роль користувача>, мені необхідна <дія>, для того щоб <мета>

User stories є одним з основних артефактів розробки ПЗ для таких гнучких (agile) методологій як Scrum або Extreme Programming (XP) [93]. Фактично, це вимоги до ПЗ, що визначаються на концептуальному рівні опису функціональності майбутньої ПС.

Розглядаючи процес декомпозиції (трансформації) User stories у функціональні вимоги слід зазначити, що в залежності від масштабів проекту та проектних ситуацій, він може відбуватись на декількох рівнях. Більш детально цей процес розглянуто на рисунку 3.1.

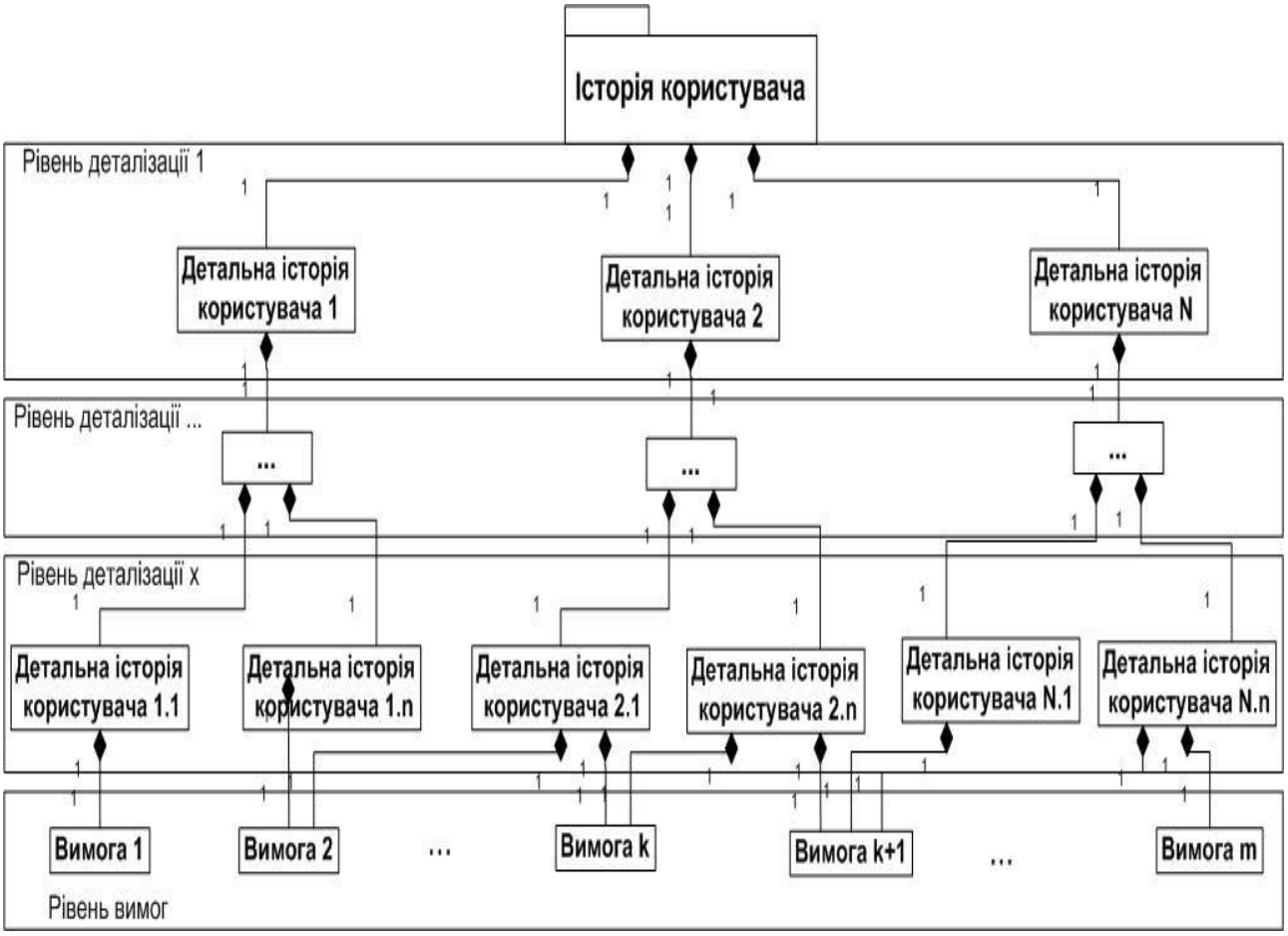

Рисунок 3.1 - Схема декомпозиції користувацьких історій у відповідні вимоги [94]

Такі артефакти User Stories як «*дія*» та «*мета*» визначають елементи доменної моделі $\underline{E}$, їх властивості $\underline{P}$, функціональні властивості $\underline{F}$, зв'язки між ними елементами моделі $\underline{R}$.

Аналізуючи сучасні публікації з цієї проблеми, можна зробити висновок, що задача оцінки структурної складності ДМ є досить актуальною як для розробки окремих складних ПС, так і для проектування ЛПП.

Так, в [26] наведено оцінку складності ДМ, яка враховує усі основні структурні компоненти, а саме: типи об’єктів ДМ (#OT), зв’язки між цими об’єктами (#ED) та операції над ними (#DO). Загальна структурна складність CD такої ДМ розглядається як відношення наявних асоціацій до кількості типів об’єктів: CD= (#ED) / (#OT), але при цьому кількість операцій (#DO) не враховується при розрахунку CD , і є окремим показником складності моделі.

Таким чином, цей підхід не дозволяє комплексно оцінити складність відповідної ДМ.

В [91] проблема оцінки складності ДМ розглядається вже більш детально. Базуючись на методології GOPPRR (Graph-Object-Property-Port-Role-Relationship) для оцінки окремих структурних одиниць моделі, автори пропонують їх наступні кількісні оцінки:

$$\begin{aligned} C_{interface} &= \#Relationships + \#Roles + \#Constrains \\ C_{element} &= \#Objects + \#Ports \\ C_{property} &= \#Properties, \end{aligned} \tag{3.3}$$

де $C_{interface}$, $C_{element}$, $C_{property}$ є відповідно кількісними показниками складності структурних елементів ДМ. Для кінцевої оцінки складності ДМ пропонується застосовувати наступний вираз:

$$C_{Overall} = C_{interface} + C_{element} + C_{property}, \tag{3.4}$$

Але при цьому цей метод також не враховує функціональну складність ДМ, а оцінює лише складність її структури.

Враховуючи вищезазначені недоліки деяких існуючих методів оцінки складності ДМ, необхідно розробити комплексний підхід до оцінки їх структурно-функціональної складності, що дозволить в подальшому отримати єдиний інтегральний показник для визначення ефективності застосування окремих методів та засобів доменного моделювання в процесах розробки ЛПП .

В роботі [95] запропоновано підхід для оцінки складності архітектурних моделей ПС, які розробляються із застосуванням пост об’єктно-орієнтованих технологій (ПООТ). Він передбачає можливість отримання аналітичних виразів для оцінки питомої складності застосування окремих ПООТ на підставі

врахування їх специфічних компонентів та відповідних вагових коефіцієнтів, які можуть бути отримані експертним шляхом. Подібний підхід можна застосувати й для оцінки складності ДМ. Для цього необхідно ввести формалізовані визначення для всіх основних артефактів в контексті побудови відповідної ДМ, приклад фрагменту якої наведено на рисунку 3.2.

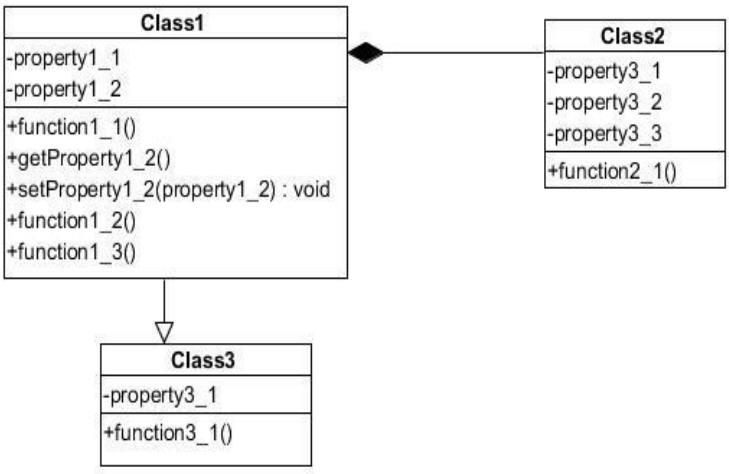

Рис.3.2 - Приклад фрагменту опису ДМ

Визначення 1: Доменна модель (Domain Model – DM) – це кортеж:

$$DM = \langle \underline{C}, \underline{R} \rangle, \tag{3.5}$$

до складу якого входить множина класів (елементів) моделі $\underline{C}$ та множина зв'язків між ними $\underline{R}$.

Визначення 2: Клас (class – C) – це кортеж:

$$C = \langle \underline{P}, \underline{F} \rangle, \tag{3.6}$$

де $\underline{P} = \{p_i\}, i = \overline{1,n}$ - множина властивостей класу, яка дозволяє зберігати стан сутності домену;

$\underline{F} = \{f_j\}, j = \overline{1,m}$ - множина функцій (або методів) класу, яка представляє його функціональне наповнення.

Визначення 3: Множина зв'язків між елементами моделі:

$$\underline{R} = \{r_{i,j}\}, \tag{3.7}$$

де $r_{i,j}$– це зв'язок між i-тим та j-тим елементом ДМ. В той же час кожен з елементів множини R відноситься до одного з елементів множини RT.

Визначення 4: Множина типів зв'язків між елементами:

$$\underline{RT} = \{As, Ag, Cm, In\}, \tag{3.8}$$

де множина RT включає 4 основних типу зав'язків в доменній моделі, а саме [96]:

- асоціація (Association - As),
- агрегація (Aggregation - Ag),
- композиція (Composition - Cm),
- успадкування (Inheritance - In).

В [97] проведено кількісний аналіз впливу кожного з цих типів зв'язків на поведінку об'єктів (класів) у відповідній ДМ. Аналітично цю залежність можна виразити наступним шляхом:

$$H_{As} < H_{Ag} < H_{Cm} < H_{In}, \tag{3.9}$$

де параметри $H_{As}, H_{Ag}, H_{Cm}, H_{In}$ є коефіцієнтами ступеню впливу відповідного типу зв'язку на загальну складність ДМ.

Метод визначення структурно-функціональної складності ДМ реалізує відображення ρ (див. вираз 2.4), що передбачає послідовне виконання наступних етапів:

1) розрахунок складності окремого класу ДМ;

2) розрахунок складності зв'язків між класами;

3) безпосередньо визначення загальної складності моделі.

При цьому вагові коефіцієнти для кожного типу компонентів ДМ визначаються експертним шляхом, наприклад, із застосуванням методу аналізу ієрархій чи методом парних порівнянь [98].

Крок 1. Розрахунок складності окремого класу.

Враховуючи те, що вплив функціональних властивостей класу є більш вагомим на загальну складність ДМ, складність кожного класу ДМ можна отримати за допомогою наступного виразу:

$$CC = K_{PC} * PC + K_{FPC} * FPC, \tag{3.10}$$

де CC (ClassComplexity) – це параметр, що кількісно визначає загальну структурно-функціональну складність відповідного класу;

PC (PropertyComplexity) – параметр, що визначає складність властивостей класу (складність його структури);

FPC (FuncPropertyComlexity) – параметр, що визначає складність функціональних властивостей класу (тобто складність поведінки класу).

Аналогічно, кількісно вплив на складність класу його «простих» (тобто атомарних) атрибутів та їх колекцій може бути представлена наступним виразом:

$$PC = K_{STP} * (\#STP) + K_{CTP} * (\#CTP), \tag{3.11}$$

де #STP (SipmleTypeProp) – це кількість властивостей будь-яких «простих» типів (не колекцій);

#CTP (CollectionTypePrperty) – кількість властивостей будь-яких типів колекцій.

Згідно з тим, що визначення ДМ передбачає лише сигнатуру її функцій, то для визначення їх впливу на загальну складність окремого класу пропонується наступний вираз:

$$FPC = \#FP, \tag{3.12}$$

де #FP **(**FunctionalProperty) – це кількість функціональних властивостей (операцій) відповідного класу.

Відповідні вагові конфіденти для кожної із формул визначаються експертним шляхом (наприклад за допомогою методу попарних порівнянь чи за допомогою методу MAI) :

$$K_{PC} = 0.3 \quad K_{FPC} = 0.7$$
$$K_{STP} = 0.4 \; K_{CTP} = 0.6 \; . \tag{3.13}$$

Крок 2. Розрахунок складності зав'язків.

Відповідно до виразу (3.9) можна визначити наступні вагові коефіцієнти для кожного типу зв'язків між класами ДМ:

$$W_{As} = 0.1, W_{Ag} = 0.2, W_{Cm} = 0.3, W_{In} = 0.4 . \tag{3.14}$$

З урахуванням коефіцієнтів з виразу (3.14) загальну складність зв'язків у певній ДМ пропонується отримати за допомогою наступного виразу:

$$RC = W_{As} * (\#As) + W_{Ag} * (\#Ag) + W_{Cm} * (\#Cm) + W_{In} * (\#In) \; , \tag{3.15}$$

де RC (RelationalComplexity) – це параметр, який визначає загальну складність зв'язків у ДМ;

#As (AssociationRelation) – це кількість зв'язків типу асоціації;

#Ag (AggregationRelation) – кількість зв'язків типу агрегації;

#Cm (CompoistionRelation) – кількість зв'язків типу композиції;

#IR (InharitanceRelation) – це кількість зв'язків типу наслідування.

Крок 3. Розрахунок загальної структурно-функціональної складності ДМ.

Для фінальної оцінки структурно-функціональної складності ДМ пропонується наступний вираз:

$$DMC = K_C \times (\#Class) \times CC + K_{RC} \times RC \tag{3.16}$$

$$K_{RC} = 0.3\ K_C = 0.7,$$

де DMC (DomainModelComplexity) – це параметр, що визначає загальну структурно-функціональну складність ДМ, #Class – кількість класів у цій ДМ.

Таким чином, сукупність аналітичних виразів (3.5) – (3.16) визначає алгоритм реалізації методу визначення структурно - функціональної складності відповідної ДМ, який є необхідною складовою частиною алгоритмічної моделі АМ, яка подана за виразом (3.1).

3.3 Розробка методу визначення ступеня повторного використання програмного коду

Як вже було зазначено у п 2.3 дисертаційної роботи для оцінки згенерованого каркасу програмного коду були використані метрики структурної складності програмного коду. Усі метрики цього типу поділяються на три категорії [20, 99]:

- метрики структурної складності класу;

- метрики взаємодії між класами;

- метрики взаємозв’язків класів в пакетах.

В розділі 2.3 дисертаційної роботи розглядалися метрики структурної складності за допомогою яких можливо визначати ступень повторного використання програмного коду. Серед них безпосередньо для методу визначення ступеня повторного використання програмного коду, шо пропонується у даному пункті дисертаційної роботи, було обрано наступні метрики [20, 99].

Зважена насиченість класу (Weighted Methods per Class) рахує суму цикломатичних складностей в класі K з методами $M_1 \dots M_n$, які визначені в цьому класі, але не успадковані від предків. $c_1 \dots c_n$– цикломатична складність методів $M_1 \dots M_n$.

$$WMC = \sum_{i=1}^{n} c_i \,, \tag{3.17}$$

Зчеплення між об’єктами (Coupling Between Objects) показує взаємодію об’єктів класу і визначає кількість сторонніх класів, з якими зв’язаний даний клас окрім успадкованих класів.

$$CBO = C_a + C_e, \tag{3.18}$$

де C_a – аферентне зчеплення (afferent coupling),

C_e – еферентне зчеплення (efferent coupling).

Глибина дерева успадкування (Depth of inheritance tree, DIT) - забезпечує для кожного класу міру рівнів успадкування від вершини ієрархії об'єктів. У мові програмування Java, де всі класи успадковують «Object», його мінімальне значення дорівнює 1.

Відклик для класу (Response for a class, RFC) визначає кількість різних методів, що можуть бути викликані, коли об’єкт класу отримує повідомлення.

Кількість нащадків (Number of children, NOC) визначається як кількість нащадків класу [100].

В роботі [23] визначається кореляція між рівнем повторного використання коду та деякими метриками структурної складності. Також у [23] наведено статистичні данні цієї залежності, що були отримані на декількох реальних проектах. Фрагмент цих статистичних даних наведено нижче у Табл. 3.1.

Таблиця 3.1 Фрагмент статистичних даних із експериментального прикладу (співвідношення між значеннями метрик та рівнем повторного використання[23])

Metric	Value	AVG (Cr)	Cr/Value
DIT	1	10.46	16,60
	2	25.21	
	…….	…….	
	6	99.11	
RFC	5	21.21	0,46
	10	19.44	
	…….	……..	
	200	95.34	
NOC	0	0.33	43,30
	1	66.67	
	……..	……	
	6	258.33	
CBO	1	22.22	3,79
	3	21.11	
	……..	…….	
	24	91.73	
WMC	3	40.38	1,05
	5	30.00	
	……..	……	
	100	105.86	

де Value – значення відповідних метрик структурної складності;

AVG (Cr) - середнє значення рівня повторного використання.

На основі цих статистичних даних представлених є можливим визначити значення щодо ступеня впливу кожної метрики на рівень CRE (визначити відносну «вагу» кожної «одиниці значення» кожної метрики) - Cr/Value. Як

результат ми маємо можливість визначити важливість однієї метрики відносно іншої.

Головним етапом у вирішенні «задачі 3» дисертаційного дослідження є розробка комплексної метрики для визначення ступеня повторно використання коду. Для цього необхідно визначити вагові коефіцієнти для кожної із обраних метрик для подальшого створення згортки цих метрик. Можливим шляхом визначення вагових коефіцієнтів є застосування одного із методів багатокритеріального аналізу (МБА) [101] для багатокритеріальних задач. Класифікація цих методів наведена на рисунку 3.3.

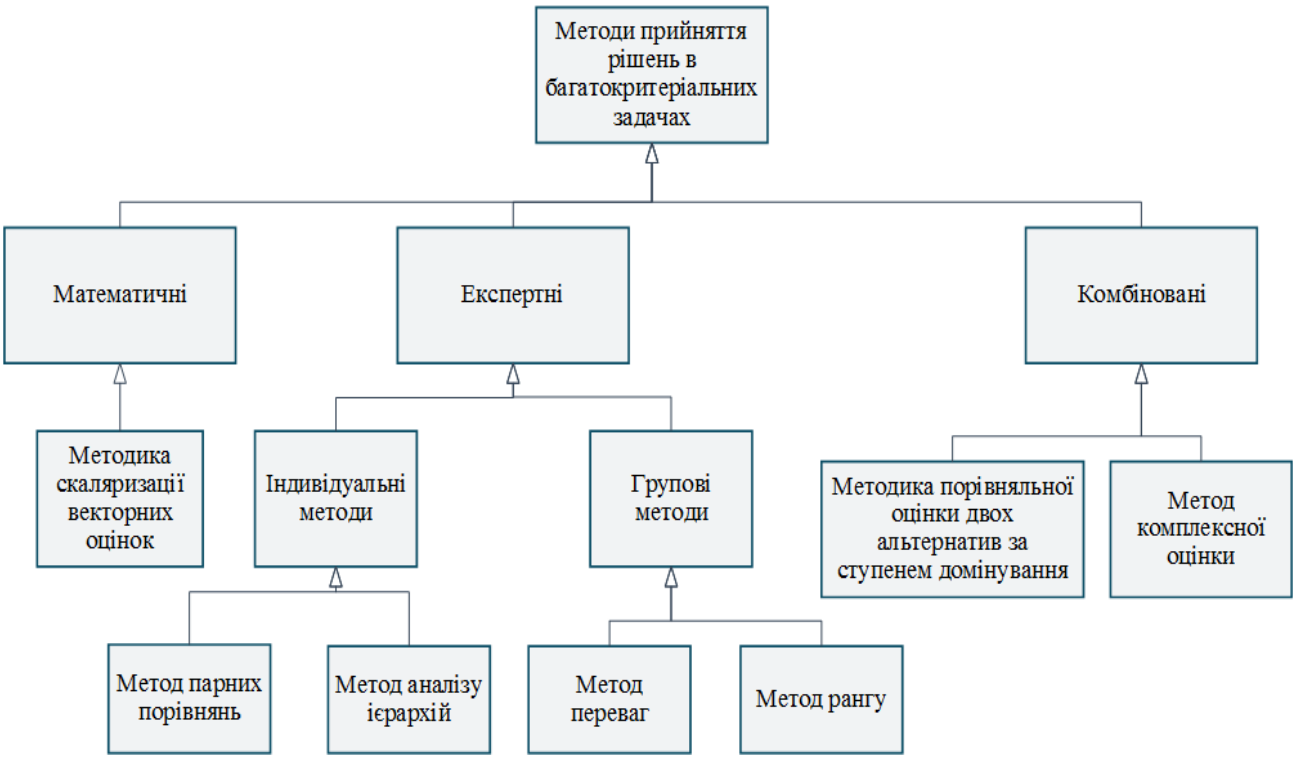

Рисунок 3.3 - Методи прийняття рішень у багатокритеріальних задачах[101]

З метою вибору найбільш доцільного методу для вирішення задачі визначення коефіцієнтів метрик структурної складності розглянемо зазначені методи більш детально.

1. Метод парних порівнянь. Цей метод базується на попарному порівнянні можливих альтернатив. Для кожної пари альтернатив експертом визначається, яка з альтернатив краща чи важливіша. Наразі цей метод реалізовано рядом

існуючих підходів, які розрізняються за кількістю залучених експертів і за шкалами порівняння альтернатив. Найбільш відомий серед них і частіше застосовуваний це метод Сааті. Він базується на порівнянні альтернатив, що виконується лише одним експертом. Експерт визначає для кожної пари альтернатив у якій мірі одна з них краща чи важливіша відносно іншої альтернативи.

2. Метод переваг. Цей метод базується на ранжуванні альтернатив і виконується групою експертів. В цій групі експерти незалежно один від одного, проводять ранжування альтернатив і визначають, яка з альтернатив є важливішою, яка є на наступному місці по важливості і т.д. Базуючись на цих даних обчислюються ваги альтернатив. В цьому методів також необхідно проведення узгодження експертних оцінок, що є необхідним для того, щоб ідентифікації різких відхилень в судженнях експертів. Якщо думки експертів суттєво відрізняються одна від одної, то необхідно знайти причини таких розходжень і, якщо це можливо уточнити деякі оцінки.

3. Метод рангу. Цей метод засновано на бальних оцінках альтернатив, що визначаються декількома експертами, в якому експерти, незалежно один від одного, оцінюють альтернативи за певною шкалою (частіше ця шкала 10ти бальна). Чим більш важливішою є альтернатива, тим більш високий бал визначається для неї. Як і в методі переваг, необхідно провести узгодження експертних оцінок і, якщо це можливо уточнити деякі оцінки.

4. Метод аналізу ієрархій (MAI, також відомий методом Сааті, але не є методом парних порівнянь Сааті). Першим етапом цього методу є аналіз всіх елементів, що впливають на її вирішенні (як альтернативи так і їх критерії) та створення їх ієрархічного представлення. На першому рівні ієрархії у будь якому випадку зазначається лише один елемент, який відображає мету. На другому рівні ієрархії зазначаються критерії, на третьому – альтернативи, з яких проводиться вибір з врахуванням критеріїв. Наступним етапом цього методу є виконання попарного порівняння усіх елементів, що впливають на вирішенні задачі. Для цього порівнюються критерії щодо їх важливості та

впливовості. Далі порівнюються альтернативи за кожним критерієм, що потребує побудови матриці парних порівнянь. На цих матриць парних порівнянь визначаються оцінки важливості критеріїв, оцінки переваги альтернатив по кожному з критеріїв і узагальнені оцінки переваги альтернатив.

5. Метод скаляризації векторних оцінок. Це метод призначено для вибору доцільних альтернативи з множини альтернатив, що визначаються за рядом критеріїв. Метод спрямований на вирішення задач, в яких рішення приймається на основі числових критеріїв (чи можливий перехід до таких числових критеріїв). Головна перевага методу скалярізаціїї векторних оцінок – це мінімальний обсяг інформації, яку необхідно отримати від особи, що приймає рішення або експерта для вибору рішення, що майже повністю формалізує рішення задачі. Недоліком методу є недостатнє врахування суб'єктивних суджень особи, що приймає рішення. Метод базується на визначенні узагальненої оцінки кожної і порівняні цих оцінок.

6. Метод порівняльної оцінки двох альтернатив за ступенем домінування. Цей метод використовується для вирішення задач, в яких необхідно обрати кращу з двох альтернатив. Такі задачі найчастіше виникають при проектуванні різноманітних технічних систем, коли основною задачою є обрання кращої з двох варіантів системи: базовий або новий. Для кожної з двох альтернатив, що порівнюються, визначається узагальнена оцінка за усіма критеріями, за якими вона переважає над іншою альтернативою. При цьому враховується ступінь переваги, а також важливість кожного критерію. Визначені узагальнені оцінки порівнюються між собою та обирається альтернатива, яка має найбільшу оцінку.

7. Метод комплексної оцінки. Метод дозволяє усунути недолік методу скаляризації векторних оцінок, за рахунок інтегрування експертних оцінок в процес розрахунку.

Найчастіше рішення в багатокритеріальних задачах приймаються з урахуванням декількох критеріїв (наприклад цілей або показників якості). Інша назва таких задач це завдання векторної оптимізації. І саме тому, що рішення в

них приймається з урахуванням набору з множини критеріїв, які називаються вектором критеріїв. Якщо при виборі рішення враховується тільки один з цих критеріїв, то така задача відноситься типу скалярної оптимізації [101]. Найчастіше такі задачі є слабоструктурованими. Головною причиною їх відношення до слабоструктурованих задач є те, що в постановці таких задач є як об'єктивні дані, так і суб'єктивні, тобто оцінки або вимоги, визначені особою, яка вирішує це завдання.

Як правило, одна альтернатива не може бути кращою за всіма критеріями. Крім того, критерії можуть бути різні за важливістю, тобто при виборі рішення потрібно більшою мірою звертати увагу на одні критерії, в меншою – на інші. Все це ускладнює рішення таких задач. Для їх рішення застосовуються методи, що представляють собою комбінацію математичних методів і методів експертного аналізу

Для вирішення задачі отримання вагових коефіцієнтів для комплексної метрики визначення повторного використання може використовуватись наприклад метод аналізу ієрархій. Але використання метод аналізу ієрархій в реальних проектних ситуаціях вимагає великого обсягу експертної інформації. Це передбачає, що експерту необхідно виконати порівняння усіх критеріїв, а також всіх альтернатив за кожним критерієм. Таким чином, метод аналізу ієрархій виявляється більш складним для використання. Крім того, застосування методу аналізу ієрархій краще використовувати лише тільки за умови, коли в прийнятті рішень у поставленій проблемі задіяні висококваліфіковані фахівці та експерти, що є не завжди можливим.

Саме тому основою для вирішення задачі визначення впливу однієї метрики на рівень ПВ відносно іншої, та для побудови комплексної метрики, у дисертаційній роботі було обрано метод комплексної оцінки [101], оскільки він простіше практичному застосування, та може не потребувати залучення експертів. А при залученні експертів дає змогу обрати будь-який з існуючих методів експертної оцінки. Тим самим він усуває головний недолік методики скаляризації векторних оцінок.

Нижче наведено алгоритм запропонованого методу оцінки ступеня повторного використання згенерованого програмного коду, який складається з 7 основних етапів :

Етап 1. Визначення множини метрик для визначення рівня повторного використання коду CRE

$$CRM = [WMC,\ RFC, DIT,\ NOC, CBO]. \quad (3.19)$$

Етап 2. Розрахунок кількісних значень обраних метрик CRM

На основі отриманих значень метрик формується множина значень

$$E = [E_1 \dots E_i]\ . \quad (3.20)$$

Етап 3. Визначення вагового коефіцієнту для кожної метрики методом парних порівнянь (pairwise comparison method - PCM):

$$K=[K_1 \dots K_i]\ . \quad (3.21)$$

Для реалізації PCM необхідно виконати наступні етапи:

Етап 3.1 Формування матриці з кількісною оцінкою переваги критерія.

$$A = [a_{ij}],\quad i,j = 1..n, \quad (3.22)$$

де a_{ij} – кількісна оцінка переваги критерія,

i, j – номера рядків і стовпців в матриці порівнянь,

n – кількість критеріїв.

Для представлення результатів оцінок у кількісному виразі рекомендовано використовувати шкалу парних порівнянь Т.Сааті (таблиця 3.2). Згідно з цією шкалою нас не цікавитиме відсутність фізичних чи об'єктивних одиниць

виміру. Основною перевагою цього методу є те, що він є безрозмірним і не виникає проблем при приведенні до однакових одиниць виміру.

Таблиця 3.2 – Відносна шкала важливості Т. Сааті [98]

Значення переваги	Визначення	Коментарі
1	Перевага відсутні (альтернативи рівні)	Альтернативи рівнозначні
3	Середня перевага	Одна альтернатива трохи важливіша за іншу
5	Помітна перевага	Одна альтернатива важливіша за іншу
7	Сильна перевага	Одна альтернатива значно важливіша за іншу
9	Абсолютна перевага	Одна альтернатива абсолютно важливіша за іншу
2,4,6,8	проміжні оцінки між сусідніми твердженнями	компромісне рішення
Обернені числа	Якщо альтернатива А, має менший пріоритет ніж В, то значення А буде дорівнювати обраному числу.	Значення $A = \frac{1}{n}$, де n значення встановлене експертом

Етап 3.2 Після розрахунку матриці парних порівнянь необхідно обчислити по ній вагові коефіцієнти:

$$\lambda_i = \sum_{j=1}^{n} a_{ij}. \tag{3.23}$$

Етап 3.3. Для нормалізації значень використовується формула:

$$V_i = \lambda_i^{\text{норм}} = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}. \tag{3.24}$$

В результаті обробки матриці парних порівнянь знаходяться локальні пріоритети (оцінки важливості) критеріїв. Чим більше локальний пріоритет, тим важливіше критерій

Етап 4. Застосування методу комплексної оцінки (Integrated assessment method - IAS) для отримання коефіцієнтів для кожної метрики:

$$W=[W_1 \dots W_i]. \quad (3.25)$$

Етап 4.1. Подання оцінок альтернатив у безрозмірному вигляді. Це перетворення виконується по-різному в залежності від виду і спрямованості критерію:

- критерії, що підлягають максимізації, всі оцінки об'єктів за даним критерієм поділяються на максимальну оцінку.

- критеріїв, які підлягають мінімізації, з оцінок за даним критерієм вибирається мінімальна, і вона ділиться на всі оцінки за даним критерієм.

- для змістовних (словесних) критеріїв, відбувається перехід до числових оцінок, в діапазоні [0,1].

Отримані значення формують матрицю

$$P = [P_{ij}], \quad i = 1..n,\ j = 1..m, \quad (3.26)$$

де, n – кількість критеріїв,

m – кількість альтернатив.

Етап 4.2. Визначення ваг критеріїв, які відображають розкид критеріїв.

Етап 4.2.1 *Визначення середньої оцінки за кожним критерієм*

$$P_i = \frac{1}{N} \cdot \sum_{j=1}^{N} p_{ij},\ i = 1..M, \quad (3.27)$$

де, M – кількість критеріїв,

N – кількість альтернатив,

p_{ij} – безрозмірні оцінки.

Етап 4.2.2 .*Визначення величини розкиду за кожним критерієм*

$$R_i = \frac{1}{N \cdot P_i} \cdot \sum_{j=1}^{N} |P_{ij} - P_i|, \quad i = 1..M \quad (3.28)$$

Етап 4.2.3. Визначення суми величини розкиду

$$R = \sum_{i=1}^{M} R_i \,. \tag{3.29}$$

Етап 4.2.4. Визначення ваг критеріїв, які відображають розкид оцінок

$$Z_i = \frac{R_i}{R}, \quad i = 1..M \,. \tag{3.30}$$

Етап 5. Визначення узагальнених ваг критеріїв

$$W_i = \frac{K_i + Z_i}{2}, \quad i = 1..M \,. \tag{3.31}$$

Етап 6. Конструювання комплексних метрики для визначення рівня повторного використання

1. У разі, якщо достатня кількість експертів:

$$CRE_{PCM} = \sum_{i=1}^{m} K_i \times E_i \,. \tag{3.32}$$

2. У разі, якщо кількість експертів мінімальна, але є можливість отримати тестові розрахунки за критеріями:

$$CRE_{IAS} = \sum_{i=1}^{m} W_i \times E_i \,. \tag{3.33}$$

3. Уразі, якщо немає можливості застосувати експерті оцінки і доступ лише до тестових розрахунків критеріїв:

$$CRE_{MATH} = \sum_{i=1}^{m} Z_i \times E_i \,. \tag{3.34}$$

Таким чином, запропонований алгоритм визначення ступеня повторного використання коду CRE дозволяє проводити розрахунки для різни проектних ситуацій, в залежності від того – чи є достане кількість експертів та чи є можливість отримати тестові розрахунки за метриками.

3.4 Процедура комплексної оцінки ефективності використання методів та засобів доменного моделювання

На основі алгоритмічної моделі, поданої за виразом (3.1), з урахуванням складу її компонентів, які визначаються за формулами (2.1) – (2.6) та (3.2), і з застосуванням алгоритму методу оцінки структурно-функціональної складності ДМ, який представлено формулами (3.10) – (3.16) та методу оцінки ступеня повторного використання який представлено формулами (3.19) – (3.34), можливо запропонувати процедуру визначення коефіцієнту ефективності застосування методів доменного моделювання за формулою (2.7). Ця процедура у вигляді UML-діаграми дій [14] представлена на рисунку 3.4.

Етап побудови доменної моделі передбачає формування множини D (див. вираз (2.3)), тобто створюється множина ДМ для одного домену за допомогою наявних технологій доменного моделювання, для яких необхідно визначити ефективність їх застосування.

Етапи щодо визначення складності доменної моделі DMC («Розрахунок складності окремих класів», «Розрахунок складності зав’язків», «Розрахунок загально структурно-функціональної складності ДМС») детально розглянуто у попередньому розділі 3.2. Для здійснення цього етапу необхідно сформувати статистику щодо структури та функціональності ДМ. Ці статистичні данні є вхідними параметрами для розрахунку структурно-функціональної складності ДМ. Цей етап реалізує відображення ρ (див. вираз 2.4 розділ 2.2) та більш детально показаний на рисунку В.1 у додатку В.

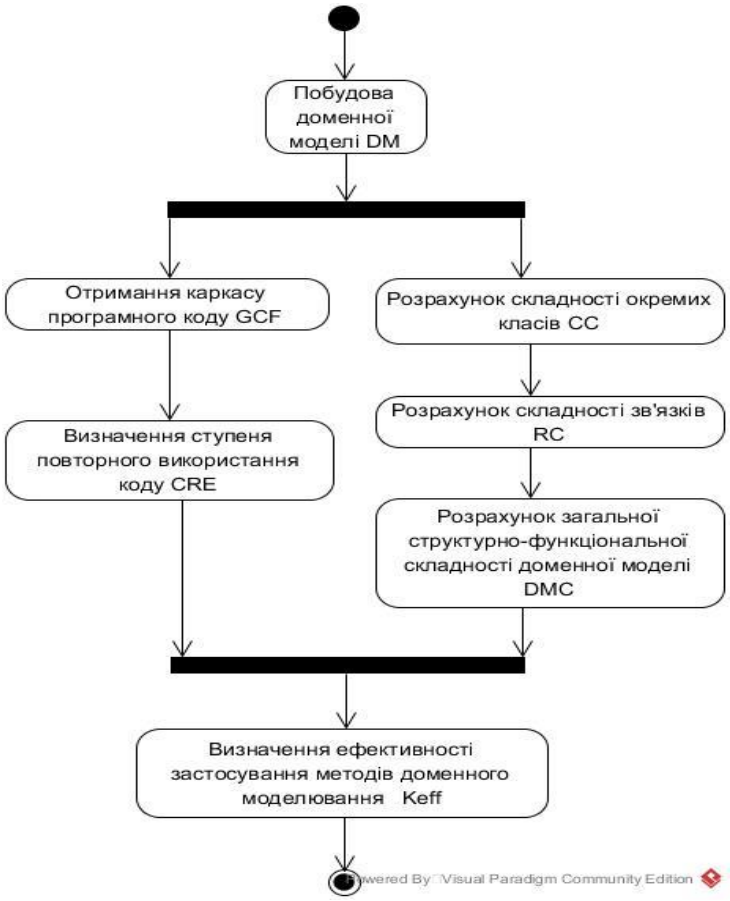

Рис. 3.4 - Процедура визначення коефіцієнту ефективності застосування методів доменного моделювання

Етап «Отримання каркасу програмного коду GCF» реалізує відображення φ (див. вираз 2.5), що можливо завдяки застосуванню відповідних інструментальних засобів доменного проектування. Ці інструментальні засоби більш детально розглянуто у розділі 2.1.3 даної роботи.

Слід зазначити, що вищезгадані етапи запропонованої процедури можуть виконуватися одночасно, що показано на діаграмі дій на рис. 3.4 шляхом застосування утворення 2-х паралельних логічних потоків з подальшою їх синхронізацією. У такий спосіб на практиці для оцінки великих за розміром ДМ може бути значно скорочено час, потрібний для отримання кінцевого результату цього процесу.

Етап «Визначення ступеня повторного використання коду CRE» (див. вираз 2.6) передбачає реалізацію відображення σ, застосовуючи відповідні

метрики. Реалізація цього відображення запропонована у розділі 3.3, а також більш детально показана на рисунках В.2 – В.5 у додатку В.

Для визначення K_{Eff} згідно з виразом (2.7), для визначення необхідно визначити дві змінні: CRE та DMC. Це забезпечується виконанням попередніх етапів даної процедури.

3.5 Загальна схема прикладної інформаційної технології для реалізації запропонованого підходу

Запропоновані вище методи, моделі та процедури, які забезпечують вирішення *Задач 1 – 4*, які були сформульовані у п. 2.2 попереднього розділу, мають бути інтегровані в єдину прикладну інформаційну технологію (ІТ), концептуальна схема якої в нотації IDEF0 [47] представлена на рисунку 3.5.

Запропонована процедура складається із сьомих функціональних блоків (ФБ), відповідно до правил нотації IDEF0 кожний такий ФБ *A1, A2 ....A7* має чотири типи інтерфейсів (”Вхід”, ”Вихід”, ”Управління”, ”Механізм реалізації”). Це дозволяє відобразити зв’язок усіх необхідних ресурсів для функціонування цієї інформаційної технології: вхідних даних, моделей, методів та алгоритмів, а також експертів та інструментальних CASE-засобів. Це є необхідним для автоматизації процесу отримання оцінки ефективності використання технологій доменного моделювання при розробці ЛПП.

Основним джерелом даних для визначення ефективності використання технологій доменного моделювання є безпосередня доменна модель. Процесу її конструювання відповідає один функціональний блок :

- A1: «Конструювання ДМ». Для конструювання ДМ, функціональні вимоги ПрО мають бути описані та подані у формі Історій Користувача (UserStories) у відповідних CASE – засобах доменного моделювання та для цього повинен бути залучений експерт ПрО. . Ці два елементи в термінах IDEF0 є для цього функціонального блока механізмами реалізації. Елементом

управління в цьому випадку служать методи доменного моделювання. Результатом роботи функціонального блоку є безпосередньо сама доменна модель та її структурно-функціональна статистика;

Наступні процеси для отримання ефективності (крім фінального А7) відбуваються незалежно та паралельно, а саме визначення ступеня ступеню повторного використання коду та структурно-функціональної складності ДМ.

Процесу визначення ступеня повторного використання коду відповідають наступні три блока:

- А2: «Генерація вихідного коду». Для генерації каркасу похідного коду джерелом вихідних даних є ДМ. Генерація каркасу коду проводиться у відповідності з методом ДМ та формальною граматикою мови програмування. Ці два артефакти є для даного функціонального блоку елементами управління. Механізмами реалізації даного функціонального блоку є ООП-експерт та відповідний інструментальний засіб доменного моделювання, що підтримує генерацію коду. Результатом виконання цього ФБ є згенерований каркас програмного коду.

- А3: «Аналіз структурної складності вихідного коду». Отриманий як результат ФБ А2 каркас програмного коду є вхідними даними для поточного ФБ. Для розрахунку структурної складності (напр. згенерованого на мові програмування JAVA) каркасу коду автоматично розбирається відповідним парсером JAVA-коду, що є складовим компонентом розробленого CASE-засобу. Також на цьому етапі визначається які саме метрики будуть обрані для визначення рівня повторного використання. Зокрема цей вибір здійснює ООП-експерт. Ці два елементи - ООП-експерт та розроблений CASE-засіб - в термінах IDEF0 є для цього функціонального блока механізмами реалізації. Елементами управління для цього ФБ є безпосередньо метрики складності. Результатом виконання цього ФБ є кількісні показники метрик ООП;

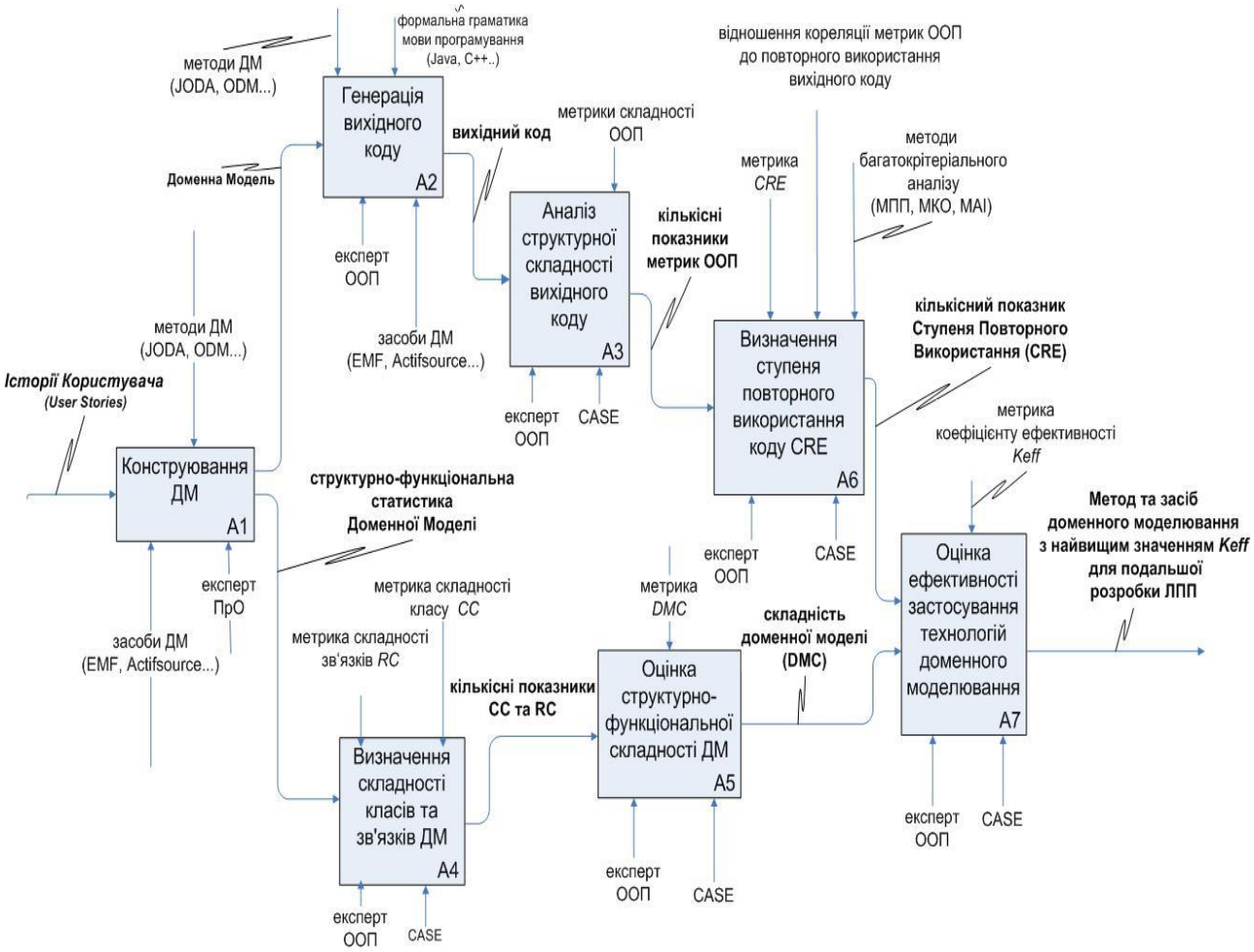

Рисунок 3.5 - Прикладна інформаційна технологія для реалізації запропонованого підходу.

- A6: «Визначення ступеня повторного використання коду CRE». Для ФБ визначення ступеня повторного використання коду CRE вхідними даними є кількісні показники метрик ООП. Але як було зазначено у 2му розділі даної роботи різні метрики мають різній вплив на загальну ступень ПВ. Для кількісного визначення коефіцієнтів впливу кожної метрики застосовуються методи багатокритеріального аналізу (МБА). Загальна метрика CRE визначається з врахуванням відношення кореляції метрик ООП до ПВ коду. Зазначені вище артефакти для даного ФБ є елементами управління. Процес розрахунку CRE автоматизовано у розробленому CASE-засіб, але на початковому етапі розрахунку ООП-експерту необхідно ввести вхідні дані для

МБА. Для цього ФБ ООП-експерт та CASE-засіб є механізмами реалізації. Результатом виконання цього ФБ є кількісний показник CRE.

Таким чином сукупність наведених вище функціональних блоків А2, А3, А6 дає можливість в результаті їх виконання отримати кількісне значення очікуваного ступеня повторного використання коду CRE.

Процесу визначення структурно-функціональної складності ДМ відповідають наступні три блока:

- А4: «Визначення складності класів та зав'язків ДМ ». Вхідними даними для цього ФБ є структурно-функціональна статистика ДМ. Для кількісного визначення складності зав'язків та складності класів застосовуються відповідно метрики (3.12) та (3.8) які є для даного ФБ елементами управління. Для визначення вагових коефіцієнтів для кожного елементу з метрик необхідна оцінка ООП-експерта. Процес визначення складності класів та зав'язків ДМ автоматизовано в розробленому CASE-засіб. Для цього ФБ ООП-експерт та CASE-засіб є механізмами реалізації. Результатом виконання цього ФБ є кількісні показники CC та RC;
- А5: «Оцінка структурно-функціональної складності ДМ». Вхідними даними для цього ФБ є кількісні показники CC та RC. Для визначення загальної структурно-функціональної складності ДМ застосовується відповідна метрика DMC (3.13), яка є для цього ФБ елементом управління. Для визначення вагових коефіцієнтів для кожного елементу з метрики DMC необхідна оцінка ООП-експерта. Процес розрахунку DMC автоматизовано у запропонованому CASE-засобі. Для цього ФБ ООП-експерт та CASE-засіб є механізмами реалізації. Результатом виконання цього ФБ є кількісне значення DMC.

Таким чином сукупність наведених вище функціональних блоків А2, А3, А6 дає можливість в результаті їх виконання отримати кількісне значення структурно - функціональної складності DMC.

Фінальним ФБ є блок для остаточної оцінки ефективності застосування технологій доменного моделювання:

- А7: «Оцінка ефективності застосування технологій доменного моделювання» . Це комплексний показник, який визначає наскільки конкретна технологія ефективна саме для поточного домену, для якого розроблялась ДМ. Вхідними даними для цього ФБ є визначені на попередніх етапах структурно - функціональна складності DMC та рівень повторного використання коду CRE для відповідних методів та засобів доменного моделювання. Ефективність визначається за допомогою метрики коефіцієнту ефективності застосування технологій доменного моделювання (2.7), що є для даного ФБ елементом управління. Процес визначення Keff автоматизовано у запропонованому CASE-засобі, що разом із ООП-експертом є для цього ФБ механізмами реалізації. Результатом виконання цього ФБ є метод та засіб доменного моделювання з найвищим значенням коефіцієнту ефективності.

Таким чином, запропонована прикладна інформаційна технологія агрегує в собі всі локальні рішення відповідних задач (із списку *Задача 1 – Задача 4*, див. п. 2.2) та дає можливість, на базі кількісних та експертних оцінок, отримати остаточну комплексну кількісну оцінку ефективності застосування технологій доменного моделювання при розробці ЛПП.

3.6. Висновки по розділу

У даному розділі дисертаційної роботи:

1) Розроблена алгоритмічна модель процесу оцінки ефективності застосування методів та засобів доменного моделювання;
2) Запропоновано метод визначення структурно-функціональної складності доменних моделей;
3) Розроблено метод визначення ступеня повторного використання програмного коду;
4) Запропонована процедура комплексної оцінки ефективності використання методів та засобів доменного моделювання;

5) Представлена загальна схема прикладної інформаційної технології для реалізації запропонованого підходу.

Список джерел, які використано у даному розділі, наведено у повному списку інформаційних джерел під номерами: [14, 20, 21, 23, 26, 29, 47, 83-101].

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ МОДЕЛЕЙ, АЛГОРИТМІВ ТА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

4.1. Опис предметної області дослідження

Для проведення експериментального дослідження в дисертаційні роботі використовувалась інформаційно-аналітична ПС адміністрування навчального процесу «Обробка інформації в системі автоматизації навчального закладу» . Це веб-орієнтована система, яка призначена для автоматизації процесу управління навчальними. Вона призначена для вирішення завдань, пов’язаних з накопиченням та опрацюванням інформації щодо:

- навчальних закладів, їх організаційної структури та підпорядкованості;
- навчальних груп та розподілу за ними студентів;
- персональної інформації студентів,
- персональної інформації викладачів;
- розробки та управління навчальними планами;
- екзаменів та сертифікації;

та ін.

Також система дозволяє генерувати різних типів звіти, які надають інформацію користувачам різних категорій у компактній і зручній формі.

На рис. 4.1 представлена архітектура цієї ПС. Середня кількість файлів програмного коду складає: у серверній частині приблизно 10^3, в клієнтській частині - 10^2 одиниць.

При розробці подібних систем, як правило, вимоги користувачів до функціональності постійно змінюються, і це вимагає проведення постійних відповідних змін у програмному коді окремих модулів ПС, або навіть випуск нової версії продукту. Ці зміни можуть бути викликані:

- зміною законодавства, що регулює навчальний процес;

- необхідністю розширення базового функціоналу системи додатковими модулями на вимогу окремих груп користувачів;

- необхідністю переходу на нові технології та засоби програмування системи та ін.

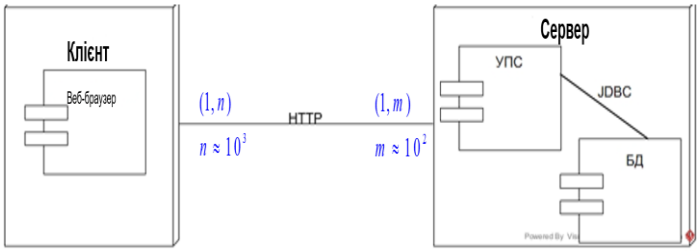

Рисунок 4.1 – Архітектура інформаційно-аналітичної ПС адміністрування навчального процесу

При реалізації нових модулів у таких ПС досить часто виникає проблема з дублювання вихідного коду. Причому це може відображатися як в копіюванні функціональності окремих функцій, так і необхідність дублювання функціоналу цілих модулів.

Наприклад, вже реалізована ПС з наведеною вище базовою функціональністю складається з наступних проблемно-орієнтованих модулів:

a) «Управління персоналом навчального закладу»;

b) «Вибір / управління навчальними курсами»;

c) «Автоматизація документообігу»;

d) «Структура навчального закладу».

Цільовими користувачами такої системи є адміністрація та вчителі, начального закладу, а також - працівники департаменту освіти регіонального рівня, які працюють з системою в режимі віддаленого доступі за рахунок Веб-базованої технології її реалізації на платформі JEE (Java Enterprise Edition) [102].

У реальному проекті по розробці та подальшому супроводу цієї ПС виникла ситуація, коли декілька інших навчальних закладів, які збиралися встановити таку ПС, висунули вимогу на розробку в її складі нового

проблемно-орієнтованого модуля, користувачами якого мають бути студенти. При цьому цей модуль повинен бути реалізований на новій технологічній платформі, а саме, мати мобільні клієнтські застосування (mobile client application), які розробляються із застосуванням JME (Java Micro Edition) [103]. Функціональні можливості такого модуля мають бути наступними:

e) «Персональна інформація студентів»;

f) «Вибір / управління курсами для навчання»

g) «Повідомлення для студентів та їх батьків (спонсорів навчання)»;

h) «Отримання документів та звітів».

В той же час функціональність «Вибір / управління курсами для навчання» має бути таким же як і у вже існуючий ПС. Таким чином, в термінах опису стандартної структури ЛПП (див. Розділ 1, п.п 1.1) у цьому випадку існують 2 продукти у складі такої лінійки, серед яких є наступні групи компонентів:

(i)компоненти (b) та (f) мають однакову функціональність, тобто вони можуть бути представлені у вигляді одного компоненту (напр., саме компонент (в)), а також компонент (d) відносяться до *постійних* компонентів такої ЛПП (див. рис. 1.1);

(ii)сукупності компонентів {(a),(e)} і {(c),(h)} мають подібну функціональність, але у новому модулі їх реалізація потребує певним доробок, і тому вони можуть бути віднесені до групи *варіабельних* у цій ЛПП;

(iii)компонент (g), а саме: «Повідомлення для студентів та їх батьків (спонсорів навчання)» є потрібним лише у 2-му продукті (другій версії ПС) модулі, і тому він є так званим *новим* компонентом ЛПП (див. рис. 1.1).

При стандартному підході до створення такої ЛПП компоненти груп (ii) - (iii) мали б створюватися з «нуля», з необхідністю дублювання вихідного коду і т.п. У разі проблемно-орієнтованого підходу (див. Розділ 1, п.п. 1.2) шляхом розробки відповідної ДМ з деталізованою функціональністю кожної її структурної частини можливо автоматизувати процес створення каркаса коду незалежно від платформи.

Наведений приклад є лише однією з таких ситуацій для подібного класу систем, коли доцільно застосовувати проблемно - орієнтованого проектування із використанням методів ДМ.

Для тестування запропонованого в дисертаційні роботі підходу було обрано такі функціональні фрагменти домену «Обробка інформації в системі автоматизації навчального закладу» як:

- «Обробка персональної інформації студентів»,
- «Обробка персональної інформації викладачів»,
- «Вибір / управління курсами для навчання»

Для обраних частин домену було сформульовано Історії користувачів (ІК) [92], фрагмент списку яких представлено у таблиці 4.1.

Таблиця 4.1 – Фрагмент опису історій користувача для тестового домену.

…	……
4	*Як <куратору>,мені необхідно <зберегти в системі ім'я, вік, дату народження, місце народження студента>, для того щоб <створити запис про нового студента>*
5	*Як <куратору>, мені необхідно <зберігати в системі основну та додаткову адресу студента>, для того щоб <визначати поточний адрес проживання студента>*
7	*Як <куратору>, мені необхідно <зберігати данні щодо попередніх місць навчання студента із урахуванням профілю попереднього навчального закладу>, для того щоб <визначити академічну різницю, яку необхідно скласти студенту >*
…	……
13	*Як <керівнику гуртожитка>, мені необхідно <визначати в системі період проживання студента у гуртожитку>, для того щоб <мати інформацію щодо наявності вільних місць у гуртожитку на певний період >*
…	……
34	*Як <працівнику учбової частини>, мені необхідно <формувати навчальні групи із зазначенням назви, періоду дії>, для того щоб <розподілити студентів за навчальними групами на початку їх навчання>*
35	*Як <працівнику деканату>, мені необхідно <задавати формат ідентифікаційного коду>, для того щоб <данні студента відповідали державним стандартам>*
…	……

На основі наведених вище ІК та принципу їх декомпозиції (див. Розділ 3, рисунок 3.1), було сформовано функціональні вимоги щодо обраних фрагментів домену, а саме:

1. Обробка персональної інформації студентів

1.1 Система повинна надавати можливість зберегти, оновити та видалити основну персональну інформацію студентів (а саме ім’я, прізвище, дату народження, місце народження, унікальний ідентифікаційний номер, стать).

1.2 Для студента необхідно мати можливість в системі зберігати та керувати інформацією щодо адреси його проживання. Студент обов’язково має одну основну адресу проживання та опціонально декілька додаткових.

1.3 Для студента необхідно зберігати його контактну інформацію (електронна пошту, телефон тощо).

1.4 При створенні нової облікової записи студента необхідно зазначати основні відомості про батьків (опікунів або довірених осіб), а саме ім’я, прізвище, контактний телефон та адреса проживання.

1.5 В режимі керування учбовою історією студента необхідно зберігати інформацію щодо попередніх місць навчання (роки, навчальний заклад, профіль).

1.6 Система повинна дозволяти переглядати поточну завантаженість гуртожитків та здійснювати відповідний розподіл студентів за гуртожитками.

2. Обробка персональної інформації викладачів.

2.1 Система повинна надавати можливість зберегти, оновити та видалити основну персональну інформацію викладачів, а саме ім’я, прізвище, дата народження, місце народження, унікальний ідентифікаційний номер, стать. Також необхідно зберігати основну контактну інформацію викладача (електронна пошта. телефон).

2.2 Для режиму редагування облікової записи викладача необхідно мати можливість в системі зберігати та керувати інформацією щодо адреси його проживання.

2.3 У режимі управління викладачем необхідно зазначати його освіту за фахом (місце навчання, номер диплому, кваліфікація, вчене звання) та інформацію щодо попередніх місць роботи (назва установи, період роботи, посада, особливі відзнаки).

3. Керування навчальними планами

3.1Система повинна забезпечувати можливість формування навчальних група із зазначенням назва групи, номеру, кафедри, факультету, період навчання, профіль.

3.2В системі необхідно здійснювати розподіл студентів за навчальними групами та підгрупами (напр. іноземна мова, фізичне виховання та ін. - навчання проводиться у підгрупах) .

3.3Для визначення навчальних профілів (спеціалізацій) навчального закладу в системі необхідно зазначати назву профілю навчального закладу, його характеристику, міністерський код навчального профілю.

3.4Для створення в системі навчальних дисциплін необхідно мати можливість зберігати їх назву, характеристику, абревіатуру, міністерський код. Також необхідно групування навчальних дисциплін за профілем.

3.5Система повинна дозволяти формувати навчальні плани на основі існуючих в системі дисциплін. При створення навчального плану необхідно зазначати його назву, унікальний код, профіль, дисципліни, період дії.

3.6Для кожної навчальної дисципліни в системі необхідно визначення часового навантаження відповідно до навчального плану.

3.7В системі необхідно здійснювати розподіл навчальних дисциплін за викладачами.

На основі зазначених вище ІК далі створюються доменні моделі для тестування запропонованого в дисертаційній роботі підходу, що розглянуто у наступних підрозділах.

4.2. Розробка архітектури та основних програмних компонентів інтегрованого CASE-засобу для реалізації запропонованого підходу.

Для автоматизації комплексної процедури оцінки ефективності використання технологій доменного моделювання, що була запропонована у п.п. 3.5 третього розділу, був розроблений прототип відповідного CASE-засобу [12] (див. рис. 4.2).

Діаграма на рис. 4.2 представляє основні функціональні можливості розробленого інструментарію у вигляді UML-діаграми сценаріїв (use-case diagram) [96]. Даний CASE-засіб передбачає одного користувача. Відповідно до зазначеної вище комплексної процедури (див. рис. 3.3), розроблений інструментарій передбачає основні функції:

- «Імпорт даних для аналізу» - це функція CASE-засобу, яка передбачає отримання необхідних даних для аналізу, що включає імпорт каркасів згенерованого коду та структури відповідних доменних моделей;

- «Імпорт каркасу коду» - це функція яка забезпечує імпорт каркасів згенерованого JAVA - коду для аналізу;

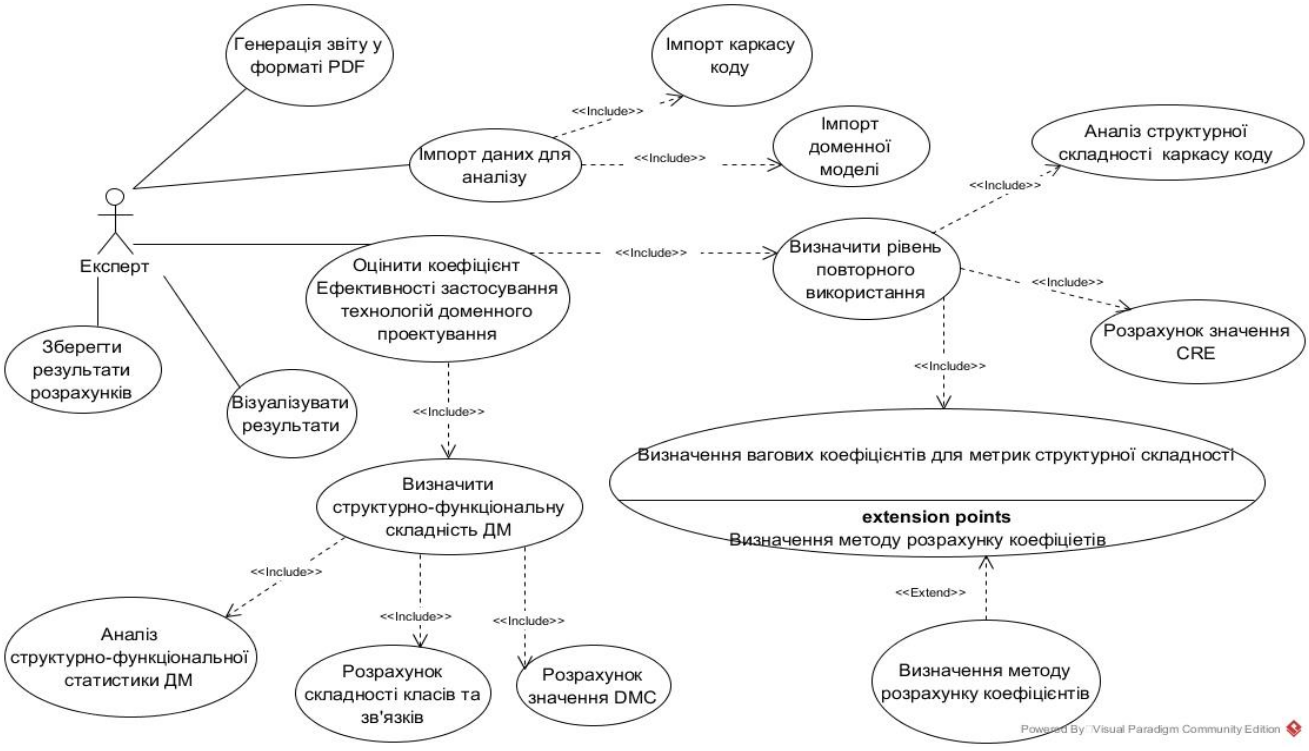

Рисунок 4.2 - Варіанти використання розробленого CASE-засобу

- «Імпорт доменної моделі» - це функція CASE-засобу, яка забезпечує завантаження структури доменних моделей для аналізу у XML-форматі;

- «Оцінити коефіцієнт Ефективності застосування технологій доменного проектування» - це центральна функція CASE-засобу, яка включає в себе виконання таких функцій як «Визначити структурно-функціональну складність ДМ» та «Визначити рівень повторного використання»;

- «Визначити рівень повторного використання» - одна з двох основних функцій CASE-засобу яка визначає рівень повторного використання коду на основі раніше завантаженого каркасу JAVA - коду. Вона включає в себе «Аналіз структурної складності каркасу коду», «Розрахунок значення CRE», «Визначення вагових коефіцієнтів для метрик структурної складності». Остання функція також розширяється можливістю вибору методу розрахунку вагових коефіцієнтів;

- «Визначити структурно-функціональну складність ДМ» - друга основна функція CASE-засобу яка забезпечує кількісне визначення структурно-функціональної складності ДМ. Ця функція включає «Аналіз структурно-функціональної статистики ДМ», «Розрахунок складності класів та зв'язків» та «Розрахунок значення DMC».

Крім основних обчислювальних та аналітичних функцій, опис яких наведено вище, розроблений CASE-засіб має ряд додаткових можливостей, таких як :

- «Візуалізувати результати» - це функція, яка дозволяє побудувати стовпчикову діаграму за отриманими кількісними розрахунками;

- «Зберегти результати розрахунків» - це функція, яка надає можливість збереження даних про зазначені розрахунки у відповідну базу даних;

- «Генерація звіту у форматі PDF» - це допоміжна функція, яка дозволяю отримати звіт про проведені розрахунки у PDF-форматі, який включає в себе результати кількісних розрахунків, а також відповідні діаграми.

Розроблений CASE-засіб має основну функціональність яка необхідна для оцінки ефективності застосування технологій доменного моделювання, а

також дозволяє отримати звіт з отриманими розрахунками у форматі PDF та зберегти відповідні данні у БД.

Відповідно до розроблених функціональних можливостей CASE-засобу була розроблена концептуальна модель даних (МД) яка наведена у нотації IDEF1X [42]. Фрагмент цієї БД із основними сутностями та їх атрибутами та зв’язками наведений на рисунку 4.3.

Основними сутностями цього фрагменту МД є такі:

- «DM» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані щодо доменних моделей розроблених із застосуванням відповідної технології доменного моделювання для відповідного домену;
- «DMT» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані про наявні технології доменного моделювання;
- «TestSystemDomain» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані щодо тестових доменів;
- «Efficience_values» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані про отримані показники ефективності для конкретної технології у відповідному домені;
- «CRE_Values» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані щодо отриманих значень ступеня повторного використання коду для конкретної доменної моделі;
- «DM_Statistic» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані про кількість відповідних структурних елементів доменної моделі;
- «DMC_Values» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані про отримані показники структурно-функціональної складності відповідної доменної моделі;
- «DM_Elements» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає загальні дані про структурні елементи доменної моделі;

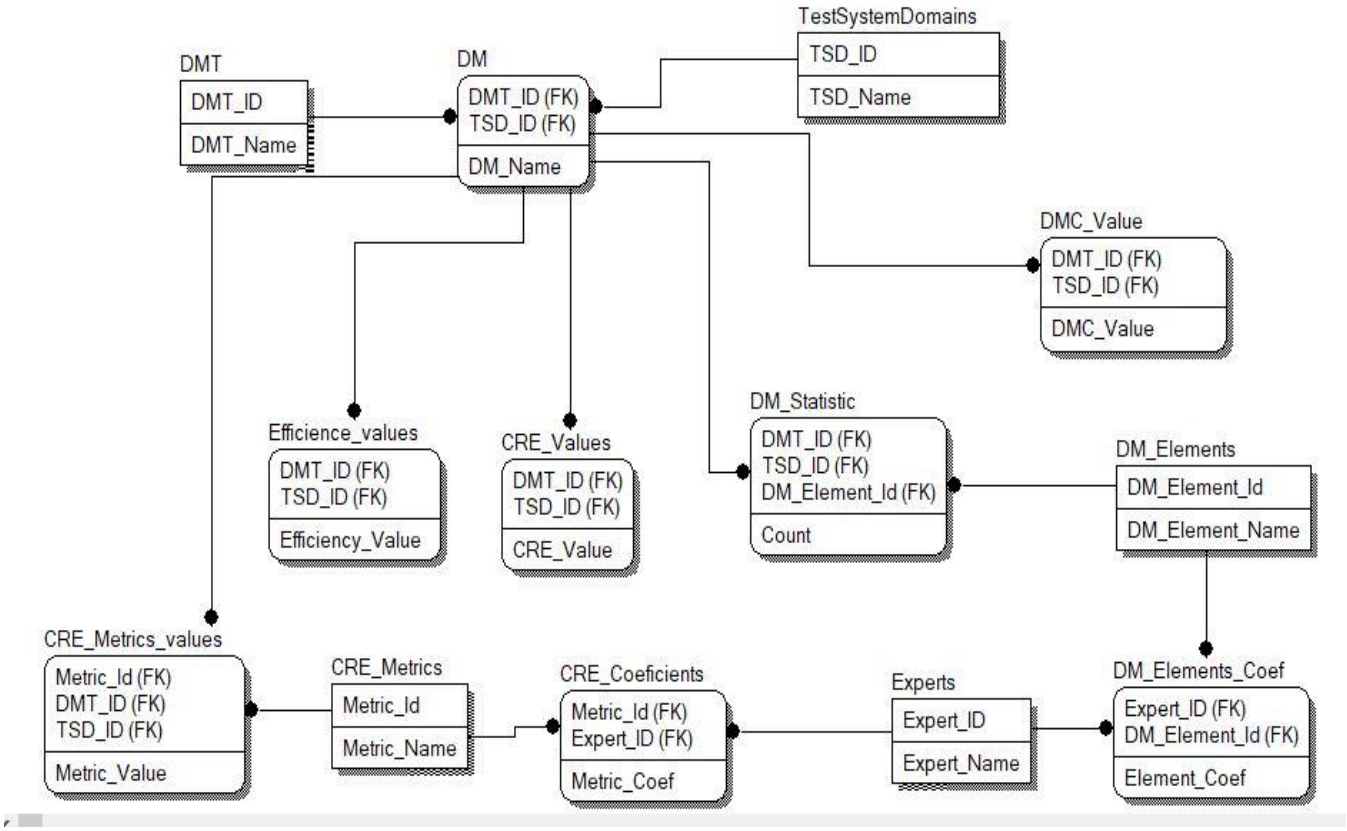

Рисунок 4.3 – Фрагмент моделі даних

- «CRE_Metrics_Values» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані щодо отриманих значень метрик повторного використання для згенерованого каркасу коду відповідної ДМ;

- «CRE_Metrics» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані що описують метрики для визначення ступеня повторного використання коду;

- «CRE_Coeficients» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані про вагові коефіцієнти метрик для визначення ступеня повторного використання коду.

- «Experts» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані щодо експертів які визначають данні для розрахунку вагових коефіцієнтів;

- «DM_Elements_Coef» - це сутність, що представляє відповідну таблицю фізичного рівня та зберігає дані вагові коефіцієнти структурних елементів доменної моделі.

Для реалізації функціональних можливостей, опис яких наведений вище (рис. 4.2), були розроблені програмні компоненти, які схематично показані на компонентній UML-діаграмі (component diagram) [42], рисунку 4.4. Основними компонентами розробленого прототипу CASE-засобу є такі:

1) «Компонент управління» - основний компонент, який забезпечує обмін даними між іншими компонентами CASE-засобу. До його складу входить множина необхідних для надання інтерфейсів, які необхідні для імпорту вхідних даних про розроблену ДМ, а саме:

- IGCFParsable – інтерфейс який необхідний для парсингу (розбору) згенерованого каркасу коду GCF (в даному випадку Java-коду) для його подальшого аналізу.

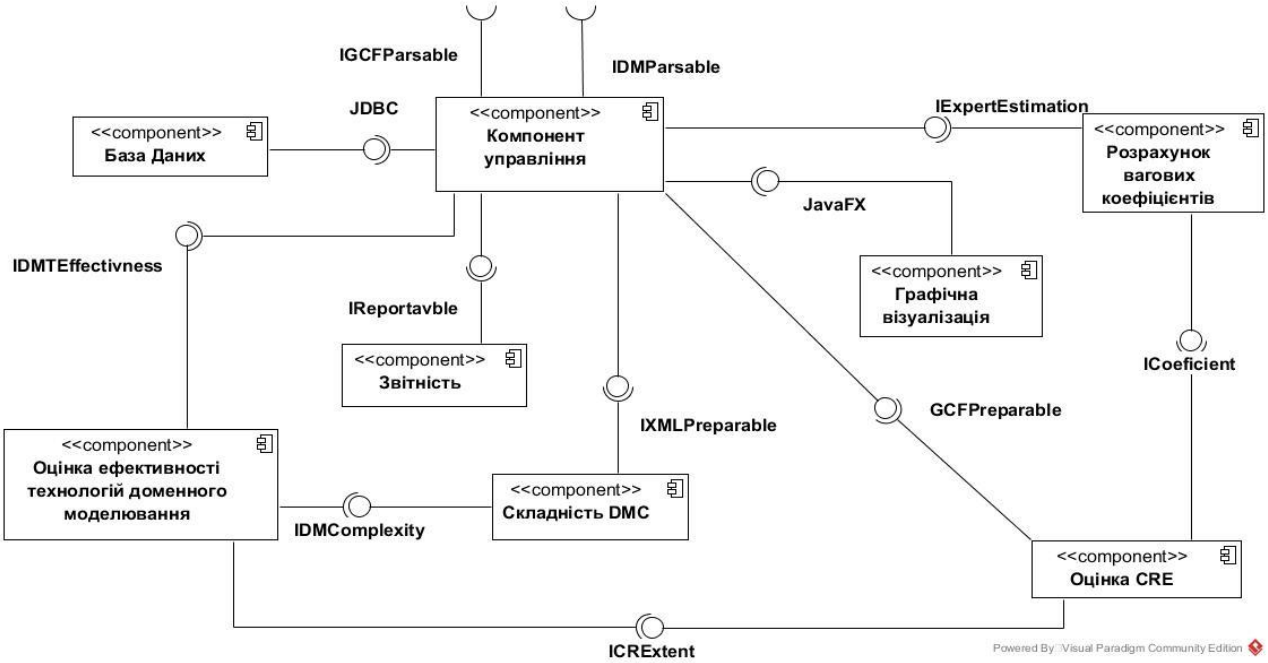

Рисунок 4.4 - Компонентна діаграма розробленого CASE-засобу

- IDMParsable – інтерфейс, який служить для розбору мета-коду доменних моделей поданих у форматі XML.

2) «Складність DMC» - компонент розробленого прототипу CASE-засобу, який виконує операцію з розрахунку структурно-функціональної складності ДМ. Він взаємодіє з компонентами «Компонент управління» та «Оцінка

ефективності технологій доменного моделювання» за допомогою наступних інтерфейсів :

- IXMLPreparable required interface, який подає статистичну інформацію щодо ДМ із розібраного XML файлу;

- IDMComplexity provided interface, який реалізує даний компонент, для надання даних про розрахунки щодо структурно-функціональної складності ДМ;

3) «Оцінка CRE» - компонент CASE-засобу, який забезпечує розрахунок ступень повторного використання коду. Він взаємодіє з компонентами «Компонент управління», «Розрахунок вагових коефіцієнтів» та «Оцінка ефективності технологій доменного моделювання» за допомогою наступних інтерфейсів:

- ICoeficient required interface, який дозволяє отримати розраховані коефіцієнти для кожної метрики;

- GCFPreparable required interface, який дозволяє отримати попередньо розібраний JAVA – код для подальшого аналізу;

- ICRExtent provided interface, який реалізує компонент для надання даних про розраховану ступень повторного використання.

4) «Розрахунок вагових коефіцієнтів» - компонент CASE-засобу, який визначає вагові коефіцієнти для метрик повторного використання коду. Він взаємодіє з компонентами «Компонент управління» та «Оцінка CRE» за допомогою наступних інтерфейсів:

- ICoeficient provided interface, який реалізує можливість отримати розраховані коефіцієнти для кожної метрики; ;

- IExpertEstimation required interface, який дозволяє обробити експертні оцінки для визначення вагових коефіцієнтів метрик;

5) «Оцінка ефективності технологій доменного моделювання» - компонент розробленого прототипу CASE-засобу, який виконує операцію розрахунку ефективності використання технологій доменного моделювання

при розробці ЛПП. Він взаємодіє з компонентами «Компонент управління», «Складність DMC» та «Оцінка CRE» за допомогою наступних інтерфейсів:

- ICRExtent required interface, який дозволяє надати данні про розраховану ступень повторного використання;
- IDMComplexity required interface, який дозволяє отримати данні щодо результатів оцінки структурно-функціональної складності ДМ;
- IDMTEffectivness provided interface, який реалізує компонент для визначення ефективності застосування технологій доменного проектування;

6) «Графічна візуалізація» - компонент розробленого прототипу CASE-засобу, який виконує операцію графічного відображення даних розрахунків, які отримує «Компонент управління» від компонентів «Складність DMC», «Оцінка CRE» та «Оцінка ефективності технологій доменного моделювання». Він має наступні інтерфейси:

- JavaFX provided interface, реалізує графічний інтерфейс CASE-засобу,

7) «Звітність» - компонент розробленого прототипу CASE-засобу, який дозволяє отримати звіт щодо отриманих результатів розрахунків у PDF-форматі. Має наступний інтерфейс:

- IReportable required interface, який надає можливість отримати дані для формування звіту,

8) «База Даних» - компонент розробленого прототипу CASE-засобу, який на діаграмі компонентів (рис. 4.5) відображає зовнішню базу даних, і відповідає за зберігання даних розрахунків. Обмін даними з БД виконується з використанням інтерфейсу з'єднання з базою даних JDBC (Java Database Connectivity).

- JDBC provided interface, який забезпечує обмін даними з базою даних.

Прототип реалізовувався як десктопне застосування . Для його реалізації застосувалася мова програмування Java [104], додаткова платформа Java FX для реалізації графічного інтерфейсу та додаткові бібліотеки для розрахунку метрик структурної складності.

Java FX [105] – набір графічних і медіа пакетів, що дозволяють розробникам проектувати, створювати, тестувати, налагоджувати та розгортати насичені клієнтські застосування, що оперують у кросплатформному середовищі. Завдяки тому, що Java FX створена як частина Java API, програмний код застосування може посилатися на будь-яку Java бібліотеку. Наприклад Java FX застосування може використовувати бібліотеки Java API для доступу до притаманних системних можливостей і під ’єднатися до серверних. Зовнішній вигляд Java FX застосувань може бути налаштований та стилізований за допомогою каскадних таблиць стилів (CSS), що дозволяє відокремити стилізацію від реалізації. Для відокремлення розробки графічного інтерфейсу від бізнес логіки може бути використана мова розмітки інтерфейсу користувача FXML. Програмний інтерфейс Java FX доступний як компонент повністю інтегрований з JRE та JDK .

На сьогоднішній день існує велика кількість інструментів для підрахунку значень метрик структурної складності програмного коду. Усі вони відрізняються одне від одного наступними критеріями:

a) метриками, які можуть бути визначені;

b) підтримуваними мовами програмування;

c) типом файлів, що подаються на вхід (.java, .class тощо);

d) представленням вихідних результатів (у графічному вигляді, текстовому тощо).

Найбільш відомі та використовувані засоби визначення метрик структурної складності наведені нижче.

1. Analyst4j [106] – базується на платформі Eclipse, доступний як автономне застосування, а також як плагін для Eclipse. Він виконує пошук та аналіз метрик, якості та генерацію звітів для Java застосувань.

2. CCCC [107] – засіб для командного рядку з відкритим вихідним кодом. Він аналізує C++ та Java файли та генерує звіти по різноманітним метрикам.

3. Chidamber & Kemerer Java Metrics [100] - засіб для командного рядку з відкритим вихідним кодом. Він підраховує значення об’єктно-орієнтованих C&K метрик працюючи з байт кодом скомпільованих Java класів.

4. Dependency Finder [108] – засіб з відкритим вихідним кодом. Він виконує аналіз залежностей в Java застосуванні.

5. Eclipse Metric Plug-in 1.3.6 [109] – Eclipse пагін для підрахунку значень метрик та аналізу залежностей з відкритим вихідним кодом.

Таблиця 4.2 показує метрики, можливість розрахунку яких надають наведені вище засоби [110].

Таблиця 4.2 – Засоби та метрики

Засоби	Метрики								
Назва	CBO	DIT	LCOM-CK	LCOM-HS	NOC	NOM	RFC	TCC	WMC
Analyst4j	+	+	+		+	+	+		+
CCCC	+	+			+	+			
CKJM	+	+	+		+	+	+		
Dependency Finder		+			+	+			
Eclipse Metric Plugin 1.3.6		+		+	+	+			+

Для розрахунку метрик структурної складності був використаний засіб Chidamber & Kemerer Java Metrics [100] . CKJM – засіб для командного рядку з відкритим вихідним кодом. Він підраховує значення об’єктно-орієнтованих C&K метрик працюючи з байт кодом скомпільованих Java класів. Також він підтримує розрахунок більшості метрик, які необхідні в даному дослідженні. Розрахунок решти метрик було реалізовано самостійно у відповідних класах.

На рисунках 4.5 - 4.7 наведено фрагменти інтерфейсу користувача CASE-засобу.

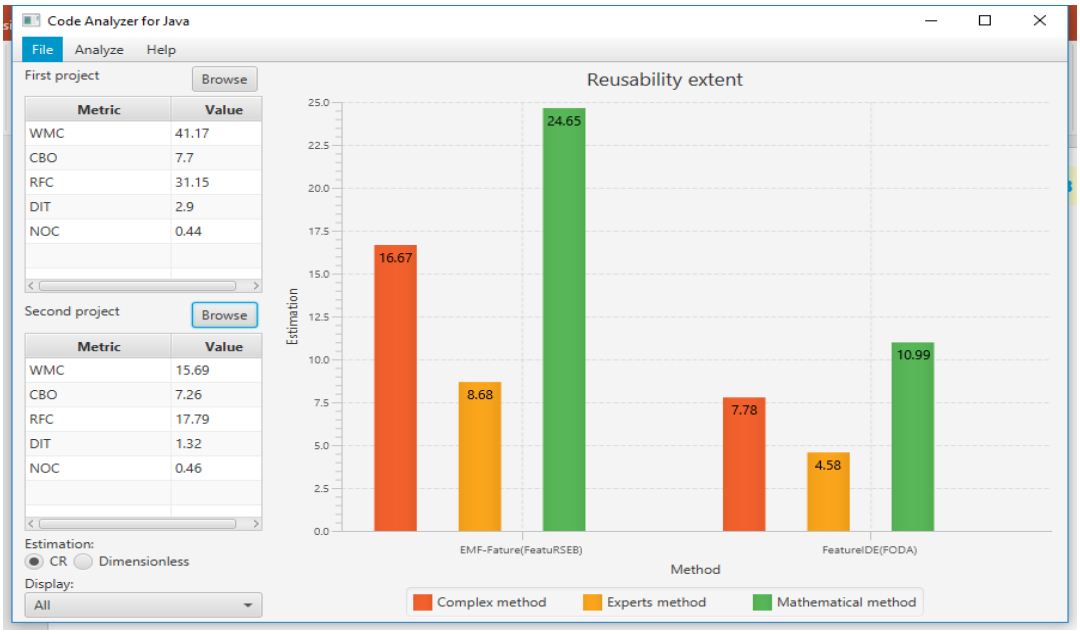

Рисунок 4.5 - Фрагмент інтерфейсу користувача «Розрахунок CRE»

Ця частина системи дозволяє:

- завантажити каркас коду для аналізу та розрахунку метрик складності коду;

- здійснити перехід до вікна визначення вагових коефіцієнтів;

- обрати метод розрахунку CRE;

- вибрати режим відображення графічної інтерпретації результатів;

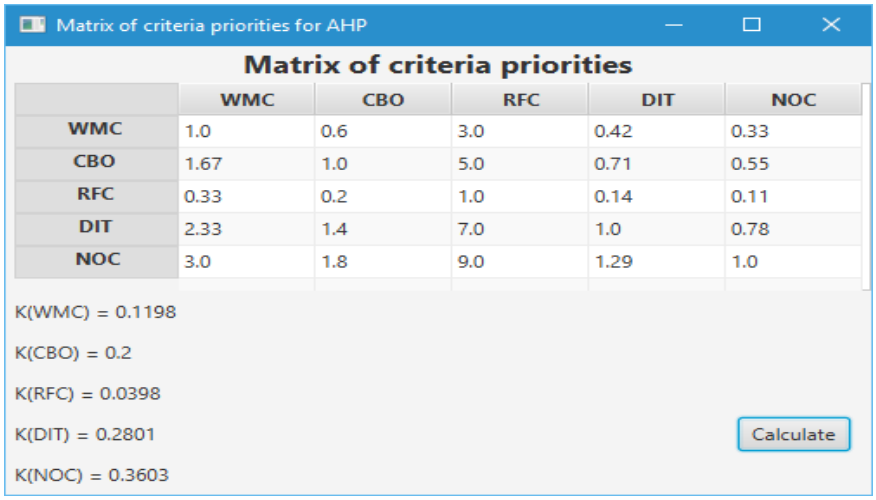

Рисунок 4.6 - Фрагмент інтерфейсу користувача «Визначення вагових коефіцієнтів метрик»

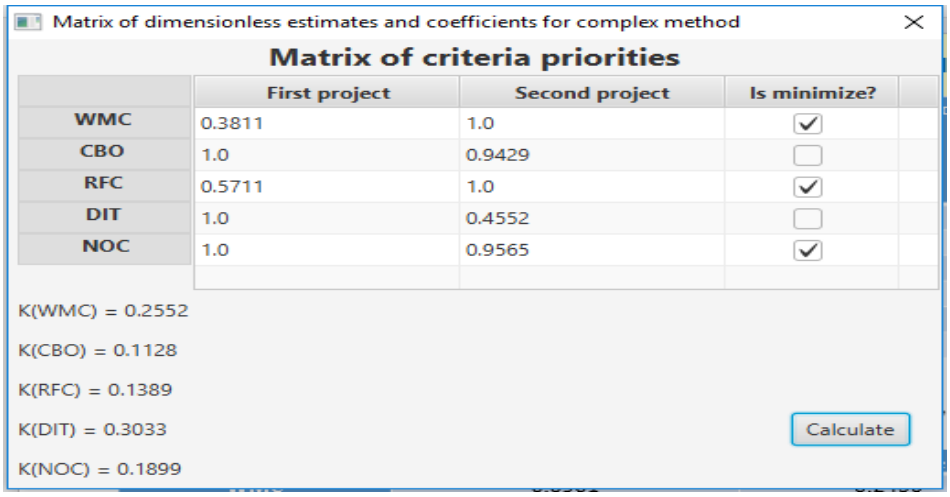

Рисунок 4.7 - Фрагмент інтерфейсу користувача «Визначення мінімізації\максимізації критеріїв»

Також даний CASE-засіб дозволяє отримати звіт щодо розрахунків у PDF-форматі. Варіант такого звіту наведено на рисунку 4.8.

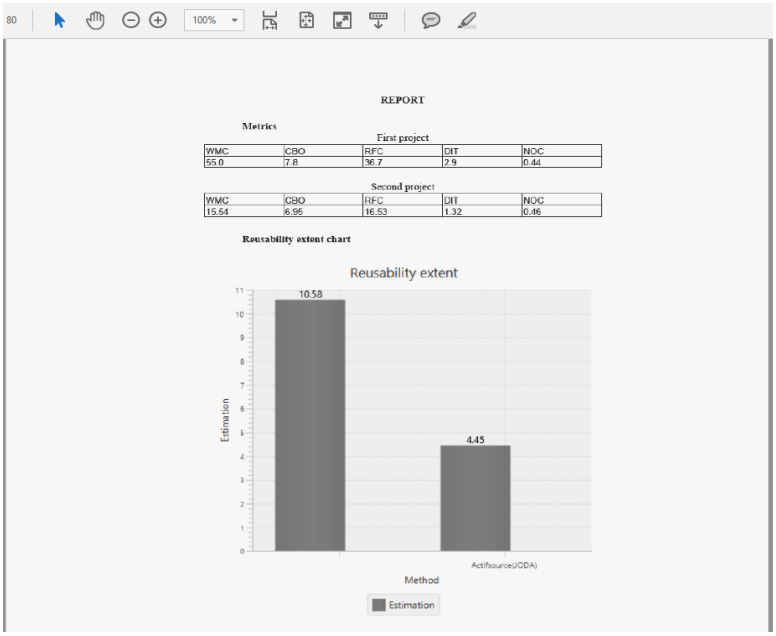

Рисунок 4.8 – Приклад фрагменту PDF-звіту CASE-засобу.

Запропонований CASE-засіб дозволяє автоматизувати усі етапи процедури визначення ефективності застосування технологій доменного моделювання.

4.3. Методика проведення експерименту, вхідні дані та приклади розрахунків чисельних експериментів

Відповідно до розроблених у п.п. 3.1 – 3.4 загальної алгоритмічної моделі (див. у [14]) та окремих методів і процедур для оцінки ефективності застосування технологій доменного моделювання при розробці ЛПП, які об'єднуються у схемі запропонованої інформаційної технології (див. рис. 3.4), проведення обчислювального експерименту передбачає наявність:

a) Бізнес-вимоги домену;

b) Доменних моделей, реалізованих у обраних технологіях доменного моделювання;

c) Згенерованого каркасу похідного коду.

Саме на підставі обробки цих артефактів можуть бути отримані початкові експериментальні дані, які потім мають бути структуровані у вигляді інформаційного базису алгоритмічної моделі процесу оцінки ефективності застосування технологій доменного моделювання при розробці ЛПП (див. вирази (3.1) – (3.2)). Відповідно до інформаційної технології, описаної у п. 3.4 попереднього розділу, обчислювальний експеримент складається з 10 основних етапів:

1. Отримати формалізовані вимоги у форматі User Stories.

2. Визначити множину доступних технологій доменного моделювання DMT.

3. Побудувати множину доменних моделей D із застосуванням обраних технологій.

4. Згенерувати множину GCF каркасів програмного коду для кожної із моделей.

5. Розрахувати за отриманими каркасами коду кількісні значення метрик структурної складності із множини CRM.

6. Визначити ступень повторного використання CRE для кожного із каркасів коду.

7. Визначення складності відповідних елементів доменних моделей.

8. Для кожної побудованої доменної моделі визначити її структурну складність DMC.

9. Обчислити значення коефіцієнту ефективності K_{Eff} технологій доменного моделювання для тестових реалізацій.

10. Визначити діапазон можливого зростання коефіцієнту ефективності K_{Eff} за допомого запропонованого підходу.

Для тестування запропонованого підходу, як вже зазначалося у пункті 4.1 , було обрано інформаційно-аналітичну ПС адміністрування навчального процесу.

На 1-му етапі експерименту визначалися бізнес-вимоги домену і формалізувалися у форматі Історій Користувача (див. табл. 4.2). На основі цих даних визначалися визначають елементи доменної моделі $\underline{E}$, їх властивості $\underline{P}$, функціональні властивості $\underline{F}$, зв'язки між ними елементами моделі $\underline{R}$. Ці данні необхідні для побудови ДМ.

На 2-му етапі експерименту визначаються технології доменного проектування, у яких буде побудовані тестові ДМ та згенерований каркас похідного коду. Як зазначалося у пункті 2.1 даної роботи, , для поточного дослідження було обрано методи ODM (Organizational Domain Modeling) та JODA (Joint integrated avionics Object oriented Domain Analysis), які програмно можуть бути реалізовані за допомогою таких інструментальних CASE-засобів як EMF (Eclipse Modeling Framework) та Actifsource відповідно. Ці засоби є плагінами для середовища розробки Eclipse та розповсюджуються безкоштовно.

На 3му етапі експерименту у відповідності із отриманими на 1му етапі даними, було побудовано 2 тестові ДМ за допомогою обраних на 2му етапів засобів. Ці ДМ наведені на рис 4.9 та 4.10 відповідно.

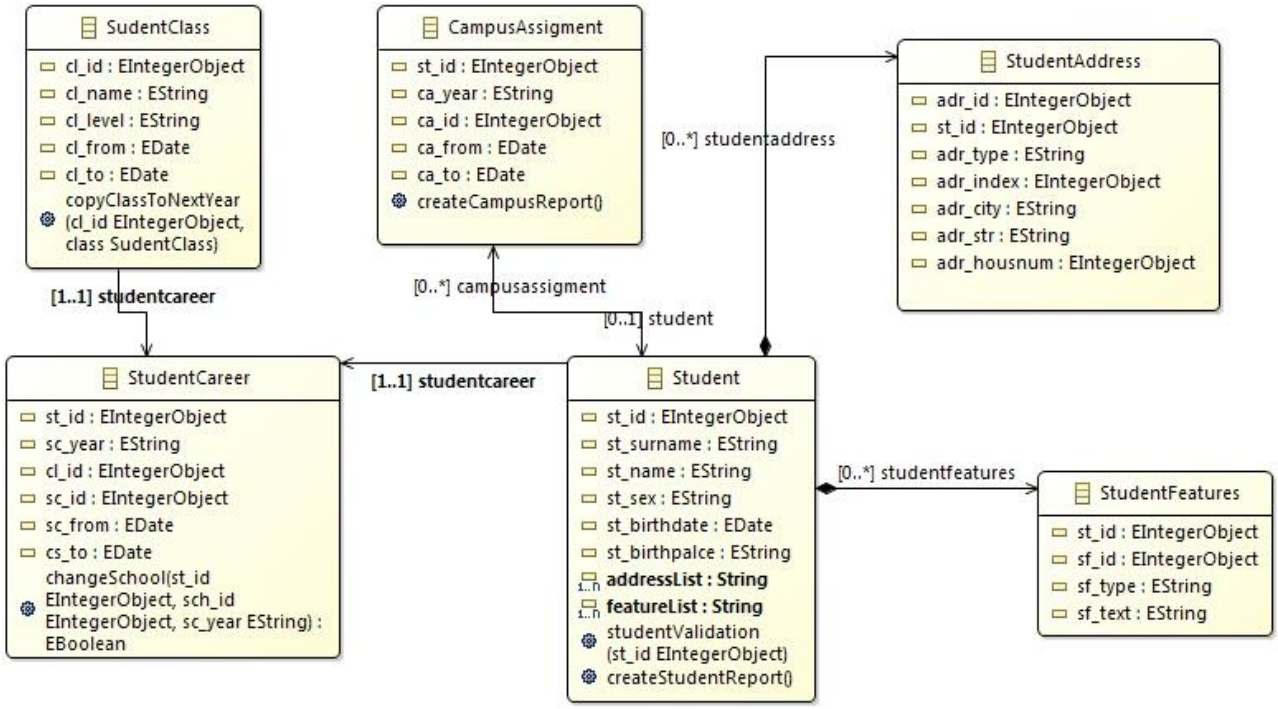

Рисунок 4.9 - Доменна модель для тестового домену у EMF CASE – tool

Наведені доменні моделі є фрагментами для обраної ПрО - інформаційно-аналітичної ПС адміністрування навчального процесу.

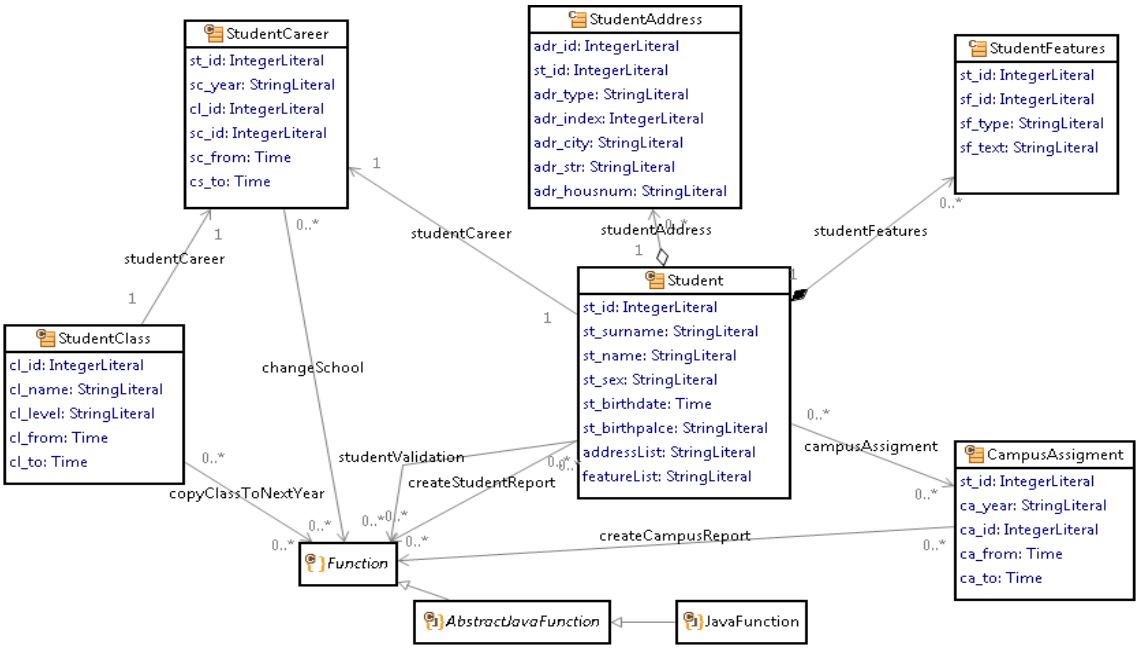

Рисунок 4.10 - Доменна модель для тестового домену у Actifsource CASE – tool

На 4му етапі експерименту для побудованих ДМ генерується каркас похідного коду у відповідних засобах.

Код отриманий у засобі EMF містить 3 пакета класів: : sTD (містить інтерфейси), sTD .impl (містить реалізацію), sTD .util (службові класи) - див рис 4.11 та 4.12.

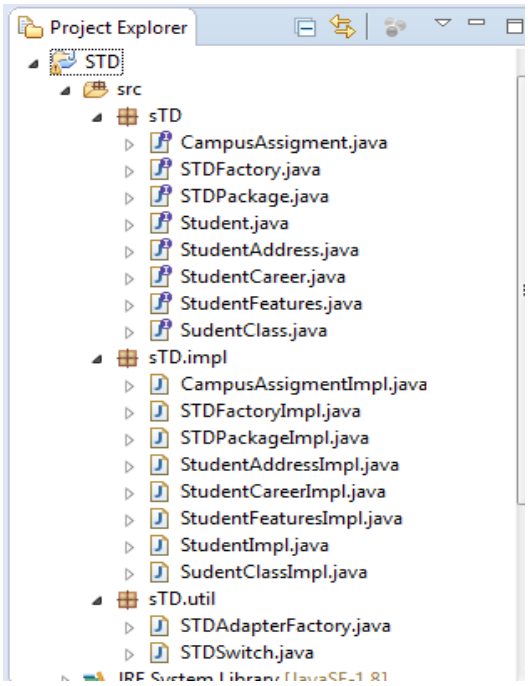

Рисунок 4.11 – Дерево згенерованого коду для ODM/EMF

```
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public Long getST_ID() {
    return sT_ID;
}

/**
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public void setST_ID(Long newST_ID) {
    Long oldST_ID = sT_ID;
    sT_ID = newST_ID;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET, TestPackage.STUDENT__ST_ID, oldST_ID, sT_
}
```

Рисунок 4.12 – Фрагмент згенерованого коду (ODM/EMF)

Код отриманий у засобі Actifsource містить 2 пакета класів: student (містить додаткові ресурсні класи) та student.javamodel (містить реалізацію та інтерфейси) – див. рис 4.13 та 4.14.

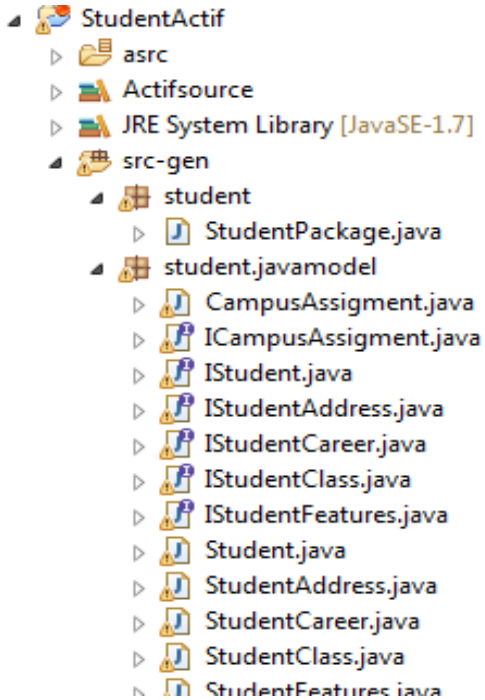

Рисунок 4.13 – Дерево згенерованого коду для JODA/Actifsource

```
@Override
public java.lang.String selectST_ID() {
  return _getSingleAttribute(java.lang.String.class, schuelerjoda.Test.TestPackage.Student_ST_aE_ID);
}

public void setST_ID(java.lang.String sT_ID) {
   _setSingleAttribute(schuelerjoda.Test.TestPackage.Student_ST_aE_ID, sT_ID);
}
```

Рисунок 4.14 - Фрагмент згенерованого коду (Actifsource)

На 5му етапі експерименту визначаються кількісні значення метрик складності за згенерованим каркасом коду (див. рис. 4.15 та 4.16).

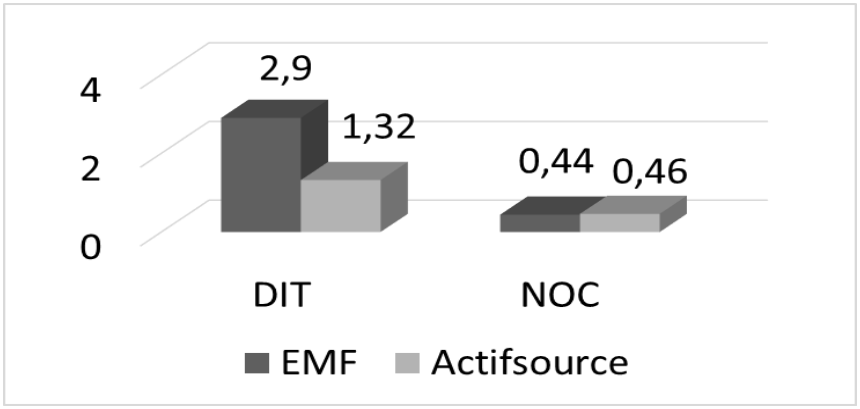

Рисунок 4.15 - Порівняння результатів для метрик DIT та NOC

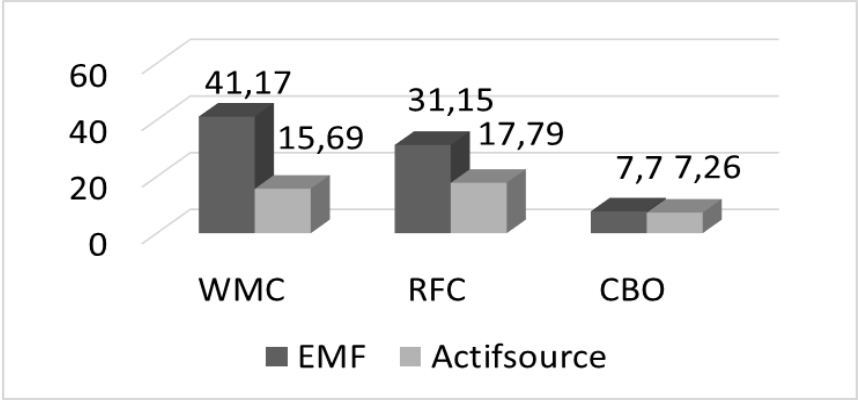

Рисунок 4.16 – Порівняння результатів для метрик WMC, RFC та CBO

На 6му етапі експерименту визначається ступень повторного використання CRE. Для цього необхідно визначити вагові коефіцієнти для кожної із метрик. Як зазначалося у пункті 3.3 даної роботи, визначення вагових коефіцієнтів залежить від проектної ситуації і наявності ООП-експертів. Можливо три випадки:

a) У випадку коли достатня кількість експертів;

b) У випадку коли кількість експертів мінімальна, але є можливість отримати тестові розрахунки за критеріями;

c) У випадку, коле немає можливості застосувати експерті оцінки і доступ лише до тестових розрахунків критеріїв.

Для даного проекту більш характерна ситуація «b». Тобто е можливість отримати розрахункові данні за метриками складності, але у наявності є лише невелика кількість експертів. Згідно с алгоритмом розрахунку CRE наведеному

у пункті 3.3 застосовується метод комплексної оцінки для визначення вагових коефіцієнтів критеріїв .

В пункті зазначалося 3.3 даної роботи зазначалося, що на основі статистичних даних представлених у [23] можливо визначити значення щодо ступеня впливу кожної метрики на рівень CRE (визначити відносну «вагу» кожної «одиниці значення» кожної метрики). Як результат ми маємо можливість визначити важливість однієї метрики відносно іншої. Ці данні в поточному експерименті використовувалися як вихідні для методу комплексної оцінки у якості експертних оцінок. Для його використання необхідно визначити відносну шкалу важливості метрик (див. таблицю 4.3)

Таблиця 4.3 - Відносна шкала важливості метрик

Значення	Опис
1	Перевага відсутні
3	Середня перевага
5	Помітна перевага
7	Сильна перевага
9	Абсолютна перевага

Далі за визначеною шкалою відносних оцінок та (як зазначалося вище) на основі статистичних даних виконується попарне порівняння важливості метрик, що зазначено у таблиці 4.4.

Таблиця 4.4 - Попарне порівняння важливості метрик

	WMC	RFC	DIT	NOC	CBO
WMC	3/3	3/1	3/7	3/9	3/5
RFC	1/3	1/1	1/7	1/9	1/5
DIT	7/3	7/1	7/7	7/9	7/5
NOC	9/3	9/1	9/7	9/9	9/5
CBO	5/3	5/1	5/7	5/9	5/5

Далі отримані значення нормалізуються та визначаються базові експертні вагові коефіцієнти для метрик, що зазначено у таблиці 4.5.

Таблиця 4.5 - Базові експертні вагові коефіцієнти для метрик

	WMC	RFC	DIT	NOC	CBO	Сума за строкою	Вагові коефіцієнти
WMC	1	3	0,42	0,33	0,6	5,35	0,1198
RFC	0,33	1	0,14	0,11	0,2	1,78	0,0398
DIT	2,33	7	1	0,78	1,4	12,51	0,2801
NOC	3	9	1,29	1	1,8	16,09	0,3603
CBO	1,67	5	0,71	0,55	1	8,93	0,2000
Разом						44,66	= 1,0000

Далі в методі необхідно застосувати оцінки тестових моделей за метриками структурної складності та привести їх до безрозмірного вигляду. Також необхідно розрахувати ваги метрик, що відображають розкид оцінок. Після цього визначаються узагальнені ваги метрик. Результати занесено в таблицю в таблицю 4.6.

Для цього, по-перше, визначається який критерій максимізується а який мінімізується:

- WMC – мінімізується;
- RFC – мінімізується;
- DIT – максимізується
- NOC – мінімізується;
- CBO – максимізується.

Таблиця 4.6 – Фінальні розрахунки коефіцієнтів

Метрика	EMF	Actifsource	Середня оцінка за метрикою (P_i)	Величини розкиду по кожній метриці (R_i)	Ваги метрик (Z_i)	Узагальнені ваги метрик (W_i)
WMC	0.3811	1.0000	0.6906	0.4481	0.3906	0.2552
RFC	0.5711	1.0000	0.7856	0.2730	0.2380	0.1389
DIT	1.0000	0.4552	0.7276	0.3744	0.3264	0.3032
NOC	1.0000	0.9565	0.9783	0.0222	0.0194	0.1899
CBO	1.0000	0.9429	0.9715	0.0294	0.0256	0.1128
Сума				1.1471	1.0000	1.0000

Згідно з тим, що у даному тестовому прикладі визначається проектна ситуація «b», тобто є можливість отримати розрахункові данні за метриками складності, але у наявності є лише невелика кількість експертів, для даного прикладу буде використано відповідну формулу (3.33).

$$\mathrm{CRE_{IAS}} = 0{,}2552 * \mathrm{WMC} + 0{,}1389 * \mathrm{RFC} + 0{,}3032 * \mathrm{DIT} + +0{,}1899 * \mathrm{NOC} + 0{,}1128 * \mathrm{CBO}$$

Відповідно результатам розрахунку метрик складності, отриманих на 5му етапі експерименту, та обраної вище формули, визначається рівень повторного використання коду CRE для двох тестових реалізацій ДМ. Результати цього розрахунку наведені нижче у таблиці 4.7.

Таблиця 4.7 – Результати розрахунку рівня повторного використання

$(CRE)_{i,j}$	
Actifsorce	EMF
7,78	16,67

На 7му етапі експерименту визначається складності відповідних елементів доменних моделей побудованих на 3му етапі. Результати цих розрахунків наведено нижче (див. табл. 4.8 – 4.11).

По перше визначається структурно-функціональна статистика ДМ для обох тестових реалізацій, а саме JODA /Actifsource (див. табл. 4.8 та 4.9)

Таблиця 4.8 Структурна статистика ДМ для реалізації JODA /Actifsource

	Class	#STP	#CTP	#FP
1	Student	6	2	2
2	StudentAddress	7	0	0
3	SturdentFeature	4	0	0
4	StudentCareer	6	0	1
5	StudentClass	5	0	1
6	CampusAsignement	5	0	1
7	Function	0	0	0
8	AbstractFunction	0	0	0
9	JavaFunction	0	0	0
	∑	**33**	**2**	**5**

Таблиця 4.9 - Функціональна статистика ДМ для реалізації JODA /Actifsource

Association	8
Aggregation	1
Composition	1
Inheritance	2

Також визначається структурно-функціональна статистика ДМ для реалізації ODM / EMF (див. табл. 4.10 та 4.11).

Таблиця 4.10 Оцінка складності класів ДМ для реалізації ODM / EMF

	Class	**#STP**	**#CTP**	**#FP**
1	Student	6	2	2
2	StudentAddress	7	0	0
3	SturdentFeature	4	0	0
4	StudentCareer	6	0	1
5	StudentClass	5	0	1
6	CampusAsignement	5	0	1
	∑	33	2	5

Таблиця 4.11 Оцінка складності відношень ДМ для реалізації ODM / EMF

Association	3
Aggregation	0
Composition	2
Inheritance	0

На 8му етапі експерименту визначається структурно-функціональна складність тестових доменних моделей на основі статистики отриманої на 7му етапі. Враховуючи дані з таблиць4.7-4.11 та із застосуванням формул (16) – (21), були отримані наступні параметри відповідних ДМ.

Для реалізації JODA /Actifsource:

*PC(JODA / Actifsource)=0.4*33+0.6*2 = 14.4*

FPC(JODA/Actifsource)=5

*CC(JODA/Actifsource)=0.3*14.4+0.7*5=7.82*

*RC(JODA /Actifsource)= 0,1*8+0.2*1 +0,3*1+0,4*2=2,1*

*DMC(JODA / Actifsource)=0.7*9*7.82+ 0.3*2.1=49,9*

Для реалізації ODM / EMF:

*PC(ODM / EMF)=0.4*33+0.6*2=13.2+1.2=14.4*

FPC(ODM / EMF)=5

*CC(ODM/EMF)=0.3*14.4+0.7*5=4.32+3.5=7.82*

*RC(ODM/EMF)= 0,1*3+0.2*0+0,3*2+0,4*0=0,9*

*DMC(ODM/EMF)=0.7*6*7.82+0.3*0.9=33.11*

На 9му етапі експерименту визначається коефіцієнт ефективності K_{Eff} технологій доменного моделювання для тестових реалізацій. Згідно с даними отриманими на попередніх етапах експерименту, а саме : CRE із 6го етапу та DMC з 8го етапу визначається коефіцієнт ефективності застосування технологій доменного моделювання згідно формули (2.7) – див таблицю 4.12.

Таблиця 4.12 – Результати обчислення коефіцієнту ефективності застосування технологій доменного моделювання

	CRE	**DMC**	K_{Eff}
JODA / Actifsource - **JA**	7,78	49.90	0.156
ODM / EMF - **OE**	16,67	33.11	0.503

На 10му етапі визначається діапазон можливого зростання коефіцієнту ефективності за допомого запропонованого підходу (див табл. 4.13).

Таблиця 4.13 - діапазон можливого зростання коефіцієнту ефективності

K_{Eff}**(OE)**	K_{Eff}**(JA)**	$\Delta=(K_{Eff}(\mathbf{OE})-K_{Eff}(\mathbf{JA}))$	$\frac{\Delta(K_{Eff}(OE),K_{Eff}(JA))}{K_{Eff}(JA)}$
0.503	0.156	0.347	2.22

Як видно таблиць 4.12 та 4.13 для тестового домену технологія ODM / EMF – OE є більш ефективною порівняно з технологією JODA / Actifsource - JA, що підтверджується відповідними значеннями K_{Eff}**.** Крім того застосування технології OE у порівнянні із технологіє JA визначає діапазон можливого зростання коефіцієнту ефективності у 2.22 раз.

4.4. Аналіз отриманих результатів і практичні рекомендації по використанню методів та засобів доменного моделювання

На основі обробки повного об’єму експериментальних даних, приклад розрахунків яких був приведений у попередньому параграфі, були отримані остаточні результати застосування запропонованої інформаційної технології визначення ефективності застосування технологій доменного проектування у процесі розробки ЛПП [14]. Для цього були обрані це ряд фрагментів домену «Обробка інформації в системі автоматизації навчального закладу».

Для кожного фрагменту домену було побудовано відповідні домені моделі у засобах Actifsource та EMF. Загальна статистична інформація за цими моделями наведена нижче у таблиці 4.14:

Таблиця 4.14 – Загальна статистична інформація досліджуваних фрагментів домену

№	Кількість об’єктів домену
Test 1	11
Test 2	10
Test 3	19
Test 4	15
Test 5	12

Згідно запропонованої методики експерименту були отримані наступні кількісні показники за результатами розрахунку CRE, DMC та Keff (дивись у таблиці 4.15).

Ці результати дозволяють проаналізувати залежності між показниками структурно-функціональної складності, рівня повторного використання згенерованого коду та кінцевими показниками коефіцієнту ефективності.

Таблиця 4.15 – Тестові результати

	ODM\EMF - OE			JODA\Actifsorce - JA		
	$(DMC)_{i,j}$	$(CRE)_{i,j}$	K_{Eff}	$(DMC)_{i,j}$	$(CRE)_{i,j}$	K_{Eff}
Test 1	82.52	20.52	0.2487	107.53	9.08	0.0844
Test 2	64.04	15.34	0.2395	87.09	7.22	0.0829
Test 3	30.91	11.78	0.3811	30.46	4.87	0.1599
Test 4	56.51	12.26	0.2170	91.05	7.84	0.0861
Test 5	127.79	16.83	0.1317	162.50	7.64	0.0470

За результатами кількісних показників коефіцієнтів ефективності можна зробити висновок, що для усіх тестових фрагментів домену більш ефективною буде застосування технології ODM\EMF - OE (див. рис 4.17).

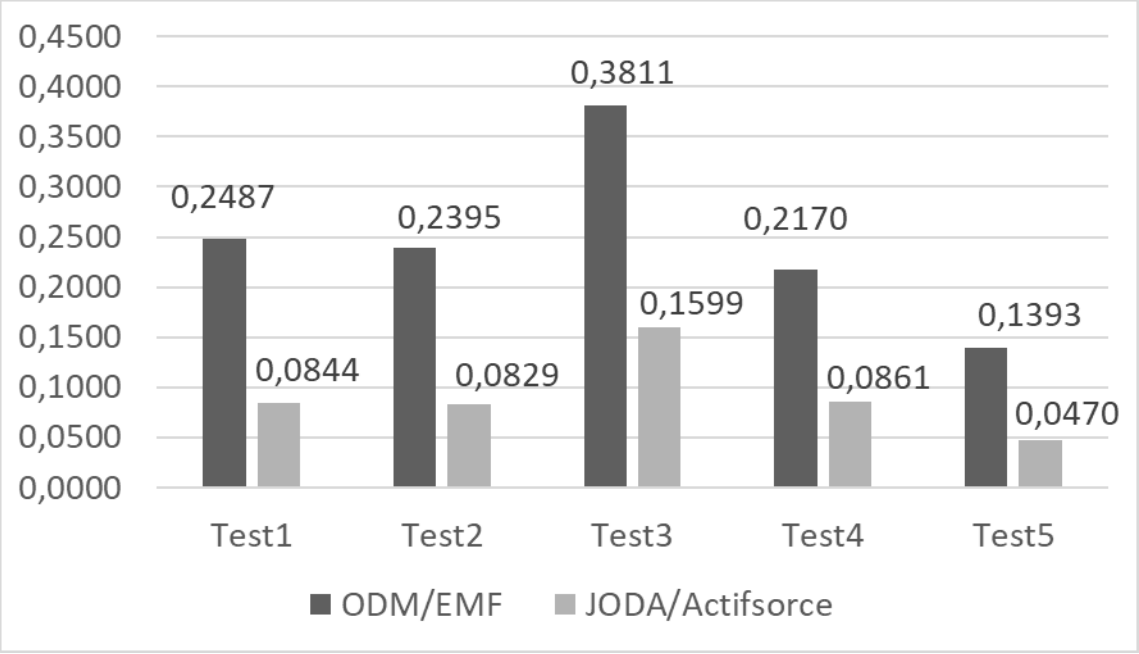

Рисунок 4.17 – Результати обчислення коефіцієнту ефективності застосування МДМ

За отриманими результатами коефіцієнту ефективності визначається діапазон можливого зростання коефіцієнту ефективності за допомого запропонованого підходу (див. табл. 4.16).

Таблиця 4.16 – Діапазон можливого зростання коефіцієнту ефективності

	K_{Eff}(OE)	K_{Eff}(JA)	Δ=(K_{Eff}(OE)−K_{Eff}(JA))	$\frac{\Delta(K_{Eff}(OE), K_{Eff}(JA))}{K_{Eff}(JA)}$
Test 1	0.2487	0.0844	0.1642	1.9
Test 2	0.2395	0.0829	0.1566	1.9
Test 3	0.3811	0.1599	0.2212	1.4
Test 4	0.2170	0.0861	0.1308	1.5
Test 5	0.1317	0.0470	0.0847	1.8

Аналіз отриманих експериментальних результатів показав, що для всіх експериментів з 2-ма обраними технологіями доменного моделювання ODM/EMF та JODA/Actifsorce зберігається стала тенденція переваги застосування саме технології ODM / EMF, що свідчить про достатню валідність запропонованого підходу до визначення коефіцієнту ефективності альтернативних методів доменного моделювання за критерієм ступеня повторного використання отриманого програмного коду. Крім того, визначено, що шляхом застосування цього підходу можливо забезпечити обрання такої технології доменного моделювання, яка забезпечує зростання значення коефіцієнту ефективності її використання в 1.4 - 1.9 рази, в залежності від структурно-функціональної складності доменної моделі, що була побудована для обраної предметної області.

4.5. Висновки по розділу

1. Наведено опис тестової предметної області, для якої в подальшому проводиться дослідження працездатності запропонованого підходу.

2. Наведено опис розробленого CASE-засобу для реалізації запропонованої інформаційної технології. Розроблені основні варіанти сценаріїв використання

та компонентна архітектура засобу. Надано схему бази даних з описом основних сутностей та основні візуальні інтерфейси користувача інструментального CASE-засобу.

3. Представлена методика проведення обчислювальних експериментів, вхідні дані та приклади виконання розрахунків окремих етапів.

4. Наведені результати аналізу проведених експериментальних досліджень та сформульовані практичні рекомендації щодо ефективного застосування існуючих методів та засобів доменного моделювання.

Список джерел, які використано у даному розділі, наведено у повному списку інформаційних джерел під номерами: [12, 42, 92, 102-110].

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

У дисертаційній роботі поставлена та вирішена актуальна науково-практична задача розробки інформаційної технології, яка складається з моделей, методів та інструментальних засобів для підвищення ефективності використання методів доменного моделювання у процесах розробки лінійок програмних продуктів (ЛПП). Внаслідок виконаних досліджень отримано наступні наукові та практичні результати:

1. Розглянуто методологічні принципи та технологічні особливості процесів розробки ЛПП, у тому числі і тих, які створюються на основі УПС, з урахуванням можливостей застосування в цих процесах методів та засобів побудови доменних моделей (ДМ).

2. Побудована класифікація та виконано порівняльний аналіз існуючих методів розробки ДМ та відповідних інструментальних засобів для їх реалізації.

3. Досліджено методологічний взаємозв'язок таких чинників як показник супроводжуваності програмних компонентів ЛПП, рівень їх структурної складності та ступінь повторного використання вихідного коду.

4. Запропоновано формалізований підхід до визначення ефективності застосування методів та засобів побудови ДМ в процесах розробки ЛПП і розроблені метрики та процедура для кількісної оцінки структурно-функціональної складності різних ДМ.

5. Вперше запропоновано алгоритмічну модель (АМ) процесу вибору методів доменного моделювання (МДМ) та інструментальних засобів при розробці ЛПП. АМ використовує оригінальний критерій ефективності, який визначається як відношення ступеню повторного використання (ПВ) згенерованого програмного коду до рівня структурно-функціональної складності відповідної доменної моделі (ДМ). Це дозволяє кількісно оцінити ефективність застосування альтернативних технологій доменного моделювання як у разі

розробки нових ЛПП, так і в процесі реінжинірингу успадкованих програмних систем (УПС).

6. Отримали подальший розвиток методи дослідження технологічних особливостей використання МДМ в процесах розробки ЛПП за рахунок використання запропонованого методу визначення структурно-функціональної складності ДМ, який на відміну від існуючих дозволяє врахувати не лише структурні елементи ДМ, але й її функціональність, а також різні типів зв'язків між структурними елементами ДМ.

7. Отримали подальший розвиток методи аналізу та визначення ступеня ПВ вихідного коду шляхом застосування сукупності метрик і обчислювальних алгоритмів, що дозволяє враховувати структурну складність програмних компонентів ЛПП ще на етапі їх проектування.

8. Удосконалено інформаційну технологію розробки ЛПП за рахунок розробки підходу до визначення ефективності застосування окремих методів та засобів доменного моделювання в процесах розробки ЛПП, який забезпечує можливість автоматизації процесів попереднього аналізу та оцінки ефективності альтернативних варіантів розробки нових компонентів ЛПП шляхом використання експертних методів у поєднанні із кількісними метриками обчислення рівня ПВ вихідного коду та метриками структурно-функціональної складності ДМ.

9. Із використанням вільно поширюваних програмних засобів і технологій таких як Java, JavaFX, Spring MVC, MySQL, Castor JDO та XML реалізовані основні програмні компоненти інтегрованого інструментального CASE-засобу для реалізації запропонованого підходу.

10. Проведено експериментальне дослідження розробленого підходу та на основі аналізу отриманих результатів зроблено мотивований висновок, що шляхом застосування запропонованого підходу є можливим забезпечити обрання альтернативних технологій доменного моделювання при розробці ЛПП, що в кінцевому рахунку забезпечує зростання коефіцієнту ефективності використання доменних моделей в 1.4 – 1.9 рази.

11. Одержані в роботі теоретичні та практичні результати, були використані при виконанні держбюджетної теми у НТУ «Харківський політехнічний інститут», а також при виконанні IT-проектів в компаніях «Інтерпак – Інформаційні системи», ТОВ (м. Харків) та Bitmedia e-Learning Solution GmbH & Co KG (м. Зальцбург, Австрія).

12. Отримані в дисертаційній роботі результати також були запроваджені в навчальному процесі кафедри програмної інженерії та інформаційних технологій управління НТУ «ХПІ» в дисциплінах «Аналіз вимог до ПЗ», «Основи проектування ПЗ», «Моделювання та аналіз проблемно-орієнтованих програмних систем».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pohl K. et al. Learning and Evolution in Dynamic Software Product Lines. 11th Int'l Symposium on Software Engineering for Adaptive and Self-Managing Systems, May 2016, Austin, Texas, USA, ACM, 2016. P. 158-164.

2. Lee S.U. An Effective Methodology with Automated Product Configuration for Software Product Line Development. Mathematical Problems in Engineering Volume 2015, Article ID 435316, 11p.

3. Лавріщева К.М., Коваль Г.І., Бабенко Л.П., Слабоспицька О.О., Ігнатенко П.П. Монографія. Ін-т програмних систем. Київ. 2011. 277с.

4. Эванс Э. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем: М.: ООО "И.Д. Вильямс", 2011. 448 с

5. Буров Є.В. Концептуальне моделювання інтелектуальних програмних систем: монографія. Монографія. Львів: Видавництво Львівської політехніки, 2012. 432 с.

6. Sommerville I. Software Engineering. Pearson, 2011. 773 p.

7. Палагін О.В., Малахов К.С., Величко В.Ю., Щуров О.С. Проектування та програмна реалізація підсистеми створення та використання онтологічної бази знань публікацій наукового дослідника. Проблеми програмування. 2017. №2, с. 72-81.

8. Ткачук Н.В., Нагорный К.А., Мартинкус И.О. Методика оценки динамической сложности требований в процессах сопровождения унаследованных программных систем. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”.2010. № 67. с.116-125.

9. Tkachuk M., Martinkus I. Models and Tools for Multi-dimensional Approach to Requirements Behavior Analysis / ed. H.C. Mayr et al. UNISCON 2012, LNBIP 137: Springer-Verlag Berlin Heidelberg, 2013. pp. 191-198.

10. Гамзаев Р. О., Ткачук Н. В., Мартинкус И. О. Мета-модель процесса трассировки требований при разработке программного обеспечения. Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. Х: НТУ «ХПІ» 2014. № 26 (1069). с.121-128

11. Tkachuk M.V., Gamzaev R.A., Martinkus I.O., Ianushkevych S.D. Methods and Tools for Dynamic Requirements Catalog Management in Agile Software Development. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”. 2015. № 52(1058). с.11-18 .

12. Tkachuk, M., Martinkus, I., Gamzayev, R., Tkachuk A. An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development / ed. Heinrich C. Mayr, Martin Pinzger. INFORMATIK 2016. Lecture Notes in Informatics (LNI). Vol. P-259: Kollen Druck+Verlag GmbH, Bonn, 2016. pp. 143-156.

13. Мартинкус І.О., Ткачук М.В., Гамзаєв Р.О. Конструювання лінійок програмних продуктів із застосуванням доменного моделювання та метрик повторного використання коду. Системи управління, навігації та зв’язку: зб. наук. пр. ЦНДІ НУ. К., 2017. Вип. 3(43). с. 93-97.

14. Ткачук М.В., Мартінкус І.О., Нагорний К.А., Гамзаев Р. О. Про один підхід до оцінки ефективності застосування методів доменного моделювання при розробці сімейств програмних систем. Збірник наукових праць Харківського національного університету Повітряних Сил. 2017. №5(54), с. 45-54

15. Мартинкус И.О., Гамзаев Р.А., Ткачук Н.В. Модели и технологи разработки линеек программных продуктов. Інформаційні технології: наука, техніка, технологія, освіта, здоров’я: Тези доповідей XXII міжнародної науково-практичної конференції, (15-17 жовтня 2014р., Харків) / за ред. проф. Товажнянського Л.Л. Харків, НТУ «ХПІ». С.13.

16. Martinkus I.O., Gamzayev R.A, Tkachuk, M. V. Towards Effectiveness Assessment of Domain Modeling Approaches to Software Development.

Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXIII міжнародної науково-практичної конференції, (21-22 травня 2016р., Харків) / за ред. проф. Сокола Є.І. Харків, НТУ «ХПІ». С.25.

17. Ткачук М.В., Гамзаев Р.О., Мартінкус І.О. Підхід до розробки лінійок програмних продуктів на основі успадкованих програмних систем із використанням методів доменного моделювання. Теоретичні та прикладні аспекти побудови програмних систем:матеріали міжнародної наукової конференції, м. Київ, 5-9 грудня 2016р. / редкол. М.С. Нікітченко, Л.Б. Буй та ін. Кіровоград, «Центр оперативної поліграфії «Авангард». 2016. с. 236-241.

18. Мартінкус І.О. Інформаційна технологія розробки та супроводу лінійок програмних продуктів на основі методів та засобів доменного моделювання. Сучасні інформаційні технології. Матеріали засідань школи-семінару за 2016-2017 н. р. Харків: ХНУ імені Каразіна, 2017. с.24-28

19. IEEE/ISO/IEC 15288-2015. ISO/IEC/IEEE International Standard - Systems and software engineering - System life cycle processes URL: https://www.techstreet.com/standards/ieee-iso-iec-15288-2015?product_id=1895213

20. Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю. Основы инженерии качества программных систем. 2-е изд. К.: Академпериодика, 2007. 672с.

21. Paliwal N, Shrivastava V., Tiwari K. An Approach to Find Reusability of Software Using Object Oriented Metrics. International Journal of Innovative Research in Science, Engineering and Technology Vol. 3, Issue 3, March 2014

22. Dubey A., Kaur H. Reusability Types and Reuse Metrics: A Survey. International Journal of Computer Applications Volume 131. No.2, December 2015. pp 12-16

23. Nandakumar A.N. Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design. International Journal of Advanced Computer Science and Applications, Vol. 7, No. 2, 2016. pp. 429-438

24. Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. Domain Engineering: Product Lines, Languages, and Conceptual Models. Heidelberg, Springer, 2013. 422 p.

25. Abel, A.; Floyd, M. Domain Driven Design Quickly. Lulu.com, 2007. 104 p.

26. Preschern C., Kajtazovic N., Kreiner C. Evaluation of Domain Modeling Decisions for two identical Domain Specific Languages. Lecture Notes on Software Engineering, Vol. 2, No. 1, February 2014. p 37-41

27. Broy M. Domain Modeling and Domain Engineering: Key Tasks in Requirements Engineering. In: Perspectives on the Future of Software Engineering. / ed. Munch J., Schmid K. Springer-Verlag. Berlin 2013. pp. 15-30.

28. Pohl K., Böckle G., Linden F. Software product line engineering: Foundations, Principles, and Techniques. Springer, 2005. 467p.

29. Perovich D., Rossel P.O., Bastarrica M.C. Feature model to product architectures: Applying MDE to Software Product Lines. Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. Helsinki. Finland. 2009. pp 201-210

30. Bayer J., Kettemann S., MuthigD. Principles of software product lines and process variants. Process Family Engineering in Service-Oriented Applications. BMBF-Project. PESOA-Report No. 03/2004. February 06, 2004. 43p.

31. Capilla R., Bosch J., Kang, K. Systems and Software Variability Management, Springer, 2013. 317p.

32. Kittlaus H.B., Clough P. Software Products: Terms and Characteristics. In: Software Product Management and Pricing. Springer, Berlin, Heidelberg, 2009. 231p.

33. Oracle SOA Suite. URL: http://www.oracle.com/technetwork/middleware/soasuite/overview/index.html

34. Microsoft Office. URL: https://www.microsoft.com/uk-ua/download/office.aspx

35. Software Engineering Institute. Carnegie Mellon University. Software Product Lines. URL: https://www.sei.cmu.edu/productlines

36. Software Product Line Conference URL: http://splc.net/

37. Bosch J., Bosch-Sijtsema P. M. Introducing agile customer-centered development in a legacy software product line. Software: Practice and Experience, 2011. pp. 871-882

38. Laguna M.A., Crespo Y. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. Science of Computer Programming Volume 78, Issue 8, 1 August 2013, pp. 1010-1034

39. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. М. «Вильямс», 2001. С.1120

40. Elmasri R., Navathe S.B. Fundamentals of Database Systems 6th ed. Addison-Wesley, 2011. 1201p.

41. Zholtkevych G. N. , Nosov K. V., Bespalov Yu. G. Descriptive Models of System Dynamics. Proceedings of the ICTERI-2016: 12th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer, Kyiv, Ukraine, June 21-24, 2016, CEUR-WS.org/Vol-1614, pp. 57-72.

42. Ларман К. Применение UML 2.0 и шаблонов проектирования: практическое руководство. Третье издание М.: Вильямс, 2013. 736 с.

43. Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture – Practice and Promise. Addison-Wesley, 2003. 170p.

44. Pansa I., Reichle M., Leist C., Abeck S. A Domain Ontology for Designing Management Services. In: 3-rd Int. Conf. on Advanced Service Computing, 2011. pp. 11-18.

45. Valiente, M.C., Barriocanal-Garcia E., Sicilia, M.A. Applying an Ontology Approach to IT Service Management for Business-IT Integration. J. on Knowledge-Based Systems. vol., 28, 2012, pp. 76-87.

46. Палагін О.В., Петренко М.Г. Архітектурно-онтологічні принципи розбудови інтелектуальних інформаційних систем. Математичні машини і системи. 2006. №4. С. 15-20.

47. Палагин А.В., Кривой С.Л., Петренко Н.Г. Знание-ориентированные системы с обработкой естественно-языковых объектов:

основы методологии и архитектурно-структурные организация. УСиМ. 2009, №3. С. 42-55.

48. Палагин А.В., Петренко Н.Г., Малахов К.С. Методика проектирования онтологии предметной области. Комп'ютерні засоби, мережі та системи, 2011, № 10. С. 5-12.

49. Левыкин В.М., Гниденко О.С. Разработка категорной модели реализации жизненного цикла процесса выбора мероприятий. Вестник Национального технического университета «ХПИ». Сборник научных трудов. Серия: Новые решения в современных технологиях. Харьков: НТУ «ХПИ». 2013. № 38(1011). С.109-120.

50. Чарнецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение. СПб.: Питер., 2005. 876 с.

51. 12th International Conference on Generative Programming: Concepts & Experiences (GPCE'13). URL: http://program-transformation.org/GPCE13

52. Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences. GPCE 2016. /ed. Fischer B. Amsterdam. The Netherlands. October 31 - November 1, 2016. 212p.

53. Лаврищева К.М. Генерирующее и сборочное программирование: Аспекты разработки семейств программных систем. Кибернетика и системный анализ, 2013, №1. С.129-144

54. International Conference on Software Reuse URL: https://icsr2018.wordpress.com

55. Breivold H.P, Larsson S., Land R. Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies. SEAA '08 Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications. 2008. pp 232-239

56. John, I. Building Domain Models from Legacy Documentation Assets. Net.ObjectDays 2005. Tagungsband : Offizielle Nachfolge-Veranstaltung der JavaDays, 10. - 13. September 2005

57. McIlroy M. D. Mass produced software components. Proc. NATO Conf. on Software Engineering, Garmisch, Germany. 1968. 136p.

58. Parnas D. L. On the Design and Development of Program Families. IEEE Transactions on Software Engineering Volume 2 Issue 1, January 1976 pp. 1-9

59. Neighbors J.M. Software Construction using Components. Technical Report. Department of Information and Computer Sciences, University of California, Irvine, 1980. 75p.

60. Neighbors, J.M. The Draco Approach to Constructing Software from Reusable Components. IEEE Transactions on Software Engineering, Volume: SE-10, Issue: 5, Sept. 1984. pp. 564 - 574

61. Atkinson C., Muthig D. Enhancing Component Reusability through Product Line Technology. ICSR 2002: Software Reuse: Methods, Techniques, and Tools. LNCS. Volume 2319. 2002 pp 93-108.

62. Frakes W.B, Kang K. Software reuse research: Status and future. IEEE Transactions on Software Engineering. Volume 31. Issue 7. July 2005. pp 529-536.

63. Prieto-Diaz R. Reuse as a New Paradigm for Software Development. Systematic Reuse: Issues in Initiating and Improving a Reuse Program. Springer-Verlag London Limited. 1996. pp 1-13.

64. Kang K.C., Lee H., Lee J. Variability in the software product line life cycle. Systems and Software Variability Management Springer Berlin Heidelberg 2013. pp 119-137

65. Kang K.C., Lee J.P. Donohoe: Feature-Oriented Product Line Engineering. IEEE Software vol. 19. No. 4. 2002. pp 58-65

66. Kang K.C., Sugumaran V., Park S. Applied Software Product Line Engineering. CRC-press. 2009. 561p.

67. Frakes W.B., Tech V., Terrys C. Software Reuse: Metrics and Models. ACM Computing Surveys (CSUR) Volume 28 Issue 2, June 1996. pp 415-435

68. Thakral S., Sagar S., Vinay. Reusability in Component Based Software Development - A Review. World Applied Sciences Journal 31 (12). 2014. pp. 2068-2072

69. Bieman J.M. Deriving Measures of software reuse in Object Oriented System. Formal Aspects of Measurement. Proc. BCS-FACS Workshop on Formal Aspects of Measurement. 1992. pp. 63-83

70. Hristov D., Hummel O., Huq M., Janjic W. Structuring Software reusability Metrics for Component-Based Software Development. ICSEA 2012: The Seventh International Conference on Software Engineering Advances.2012. pp. 421-429

71. Suri P. K., Garg N. Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse. IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2009. pp. 237-248

72. Parul G., Kumar B.P. Reusability Metrics for Object-Oriented System: An Alternative Approach. International Journal of Software Engineering (IJSE), Malaysia, V.1, I.4, 2010. pp. 62-73

73. Gui G., Scott P.D. Measuring Software Component Reusability by Coupling and Cohesion Metrics. Journal of Computers, vol. 4, no. 9, September 2009. pp. 797-805.

74. Ferré X., Vegas S. An Evaluation of Domain Analysis Methods. In 4th CAiSE/IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design. 1999. pp.1-13

75. Rodriguez A. I. Domain Engineering Methodologies Survey. GMV. Madrid, 2007. 38p.

76. Organizational Domain Modeling (ODM) Guidebook. Informal technical report for software technology for adaptable, reliable systems. Version 2.0, 1996. 491p.

77. Holibaugh R. Joint Integrated Avionics Working Group (JIAWG) Object-Oriented Domain Analysis Method (JODA), Version 3.1 .1993. 92p.

78. Gronback R. C. Eclipse modeling project. A Domain-Specific Language Toolkit. Addison-Wesley Professional. 2009. 736p.

79. Сорокин А.Н., Кознов Д.В. Обзор проекта Eclipse Modeling Project // Системное программирование, 5:1. 2010. сс.6-31

80. Eclipse Modeling Framework(EMF) – плагин для Eclipse// URL: http://www.eclipse.org/emf

81. Actifsource. URL: http://www.actifsource.com/

82. Actifsource User Manual. https://www.actifsource.com/_downloads/ActifsourceManual_ActifsourceUserManual.pdf

83. Сергиенко А. М., Симоненко В. П. Алгоритмические модели обработки потоков данных. Электронное моделирование. Т. 30, №6. 2008. С. 49–60.

84. Гамзаєв Р.О. Моделі та інформаційна технологія трасування вимог в гнучких процесах розробки програмного забезпечення: дис. … канд. техн. наук: 05.13.06 / Національний технічний університет «Харківський політехнічний інститут». Харків, 2013. 173 с.

85. Сокол В.Є. Моделі та інструментальні засоби систем управління IT-інфраструктурою : дис. … канд. техн. наук: 05.13.06 / Національний технічний університет «Харківський політехнічний інститут». 2014. – 179 с.

86. Нагорний, К.А. Моделі та інструментальні засоби супроводу програмних систем на основі пост об'єктно-орієнтованих технологій: дис. … канд. техн. наук: 05.13.06 / Національний технічний університет «Харківський політехнічний інститут», Харків, Україна, 2016. 182 с.

87. Guerrero JM, Ramos P. Mind Mapping for Reading and Understanding Scientific Literature. International Journal of Current Advanced Research 4(11). 2015. pp 485-487.

88. Davies M. Concept Mapping, Mind Mapping and Argument Mapping: What are the Differences and Do they Matter?. Higher Education, Vol. 62, Issue 3, 2011. pp. 279-301

89. Tkachuk M.V., Gamzayev R.O., Mayr H.C. Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development. Problems in Programming. Kyev. Ukrainian National Academy of Science. No 2- 3 (special issue). 2012. pp.160-167.

90. Ткачук М.В. Гамзаев Р.О. Модель та інформаційна технологія побудови адаптивної матриці трасування вимог в гнучких процесах розробки

програмного забезпечення. Вісник НТУ «ХПІ». Харків. № 2 (976). 2013. С. 49 – 60.

91. Poels G., Dedene G., Viane S. A Quantitative Assessment Of The Complexity Of Static Conceptual Schemata For Reference Types Of Front-office. In Proceedings of the Fifth International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE2001). Budapest, Hungary. 2001. pp. 1-12

92. Ambler “Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling”. URL: http://agiledata.org/essays/agileDataModeling.html

93. Амблер С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. Библиотека программиста. СПб.: Питер, 2005. 412 с.

94. Leffingwell D. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional. 2010. 560p.

95. Литвинчук М.М., Нагорний К.А., Ткачук М.В. Архітектурні моделі оцінки складності пост об’єктна-орієнтованих технологій розробки програмних систем. Вісник ХНУ імені В.Н. Каразіна, Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». № 1000, 2012. С. 225-235.

96. Unified Modeling Language (UML) Resource Page //Офіційний вєб-ресурс уніфікованої мови моделювання UML, що розробляється консорціумом OMG – Object Management Group. URL: http://www.uml.org

97. Kang D., Xu B., Lu J. A Complexity Measure for Ontology based on UML. Distributed Computing Systems. FTDCS 2004. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. pp. 222 – 228

98. Саати Т.Л. Принятие решений. Метод анализа иерархий. пер. с англ. М.: Радио и связь, 1993. 278 с.

99. Gronback R. C. Software Remodeling: Improving Design and Implementation Quality. Using audits, metrics and refactoring in Borland Together Control Center. A Borland White Paper. 2003. pp. 18–23.

100. Chidamber and Kemerer Java Metrics.URL: http://www.spinellis.gr/sw/ckjm/doc/metric.html/

101. Волошин О.Ф., Мащенко С.О. Моделі та методи прийняття рішень: навч. посіб. для студ. вищ. навч. закл. 2- ге вид., перероб. та допов. К. : ВПЦ "Київський університет", 2010. 336 с

102. Java™ EE at a Glance URL: http://www.oracle.com/technetwork/java/javaee/overview/index.html

103. Java Platform, Micro Edition (Java Me) URL: http://www.oracle.com/technetwork/java/embedded/javame/index.html

104. Java official Website URL: https://www.java.com

105. JavaFX Overview. URL: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784

106. Analyst4j. URL: https://codeswat.com/cswat/index

107. CCCC - C and C++ Code Counter URL: http://cccc.sourceforge.net/

108. Dependency Finder URL: http://depfind.sourceforge.net/

109. Metrics 1.3.6 URL: http://metrics.sourceforge.net/

110. Lincke R., Lundberg J., Löwe W. Comparing Software Metrics Tools ISSTA '08 Proceedings of the 2008 international symposium on Software testing and analysis. pp. 131-142 2008. 11 с.

ДОДАТОК А.
АКТИ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного технічного університету
«Харківський політехнічний інститут»

_____ проф. Марченко А.П.

"25" листопада 2017р.

АКТ
про використання результатів дисертаційної роботи
асистента кафедри програмної інженерії та інформаційних технологій управління
Мартінкус Ірини Олегівни
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗРОБКИ ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ НА ОСНОВІ МЕТОДІВ ТА ЗАСОБІВ ДОМЕННОГО МОДЕЛЮВАННЯ»

Ми, що нижче підписалися, декан факультету комп'ютерних наук та програмної інженерії (КН) проф. І.П. Гамаюн, завідувач кафедри програмної інженерії та інформаційних технологій управління проф. Годлевський М.Д., професор кафедри програмної інженерії та інформаційних технологій управління проф. Ткачук М.В. склали цей акт у тому, що дисертаційна робота Мартінкус І.О. виконувалася на кафедрі програмної інженерії та інформаційних технологій управління (ПІТУ) у межах держбюджетної теми МОН України "Розробка інтелектуальних моделей та технологій для підвищення ефективності проектування та супроводу складних програмних систем" (ДР № 0111U002288), де Мартінкус І.О. брала участь як співвиконавець окремих етапів. В результаті виконання наукової роботи були отримані, зокрема, такі результати:

1. Запропоновані моделі та методи для визначення ефективності застосування методів та засобів доменного моделювання при розробці лінійок програмних продуктів (ЛПП), що дозволяє кількісно оцінити ступінь ефективності застосування окремих технологій доменного моделювання залежно від складності початкових вимог до функціональності ЛПП.
2. Реалізовано прикладну інформаційну технологію та методику оцінки ефективності застосування методів та засобів доменного моделювання, що дозволило провести експериментальне дослідження запропонованого підходу, оцінити його працездатність та надати рекомендації щодо його практичного впровадження.

Декан КН-факультету проф. І.П. Гамаюн

Зав. кафедрою ПІТУ проф. М.Д. Годлевський

Науковий керівник теми ДР № 0111U002288 проф. М.В.Ткачук

bit media
member of eee group

bit media e-solutions GmbH
Kaerntner Strasse 337, 8054 Graz, Austria
phone +43 316 / 28 66 60
fax +43 316 / 28 66 60 – 50
mail office@bitmedia.at
web www.bitmedia.at

To the Vice-rector for scientific and research issues
at the National Technical University
"Kharkiv Polytechnic Institute"
Prof. Dr. A. Marchenko

Graz, 20.10.2017

CONFIRMATION LETTER

The company Bit media e-learning solution GmbH ascertains with this letter that the results achieved within the PhD-thesis activities performed by Mrs. Iryna Martinkus were used in maintenance process for several software projects performed during 2014-2017, especially the following points:

1. The method for an estimation of a source code reusability extend;
2. The approach to decision making support to choose the most effective domain modeling methods within a domain-driven software development as a prospective way to design software product lines in future.

These results actually can be recognized as the useful tools for complex software systems development, because they allow to make reasonable decision which domain modeling technology is more effective for a given application area. The performed test-cases have shown that depend on a complexity of initial requirements (given as user stories) the effectiveness coefficient can be evaluated between 1,4 and 1.9 times accordingly.

Kind regards,

bit media
member of eee group
bit media e-solutions GmbH
Kaerntner Strasse 337, 8054 Graz
Tel +43 316/28660-0 Fax DW 50
office@bitmedia.at

CEO
Manfred Brandner

ЗАТВЕРДЖУЮ
Директор ТОВ «ІНТЕРПАК
ІНФОРМАЦІЙНІ СИСТЕМИ»

_____ Гамзаєв О. К.

"01" 09 2017 г.

АКТ

про використання результатів дисертаційної роботи
Мартінкус Ірини Олегівни
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗРОБКИ ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ НА ОСНОВІ МЕТОДІВ ТА ЗАСОБІВ ДОМЕННОГО МОДЕЛЮВАННЯ»,
виконаної в НТУ «Харківський політехнічний інститут», що подана
на здобуття наукового ступеня кандидата технічних наук

Комісія у складі:
голови - Марчука Артема Володимировича, керівника групи розробки програмних систем, к.т.н.; членів комісії - Сагайдачного Дениса Олексійовича, провідного інженера групи розробки програмних систем; Бабака Олександра Юрійовича, старшого інженера-розробника програмного забезпечення, встановила, що у період 2015 - 2017 рр., при виконанні проектів з розробки програмних систем для автоматизації організаційних процесів та документообігу в мережі навчальних закладів, були використані такі результати наукових досліджень Мартінкус І.О.:

- запропоновані методи та інструментальні засоби аналізу та визначення ступеня повторного використання вихідного коду шляхом застосування кількісних метрик і обчислювальних алгоритмів, що дозволяє враховувати структурну складність програмних модулів у складі лінійки програмних продуктів(ЛППП);
- розроблені компоненти прикладної інформаційної технології для автоматизації процесів попереднього аналізу та оцінки ефективності альтернативних варіантів розробки нових модулів ЛППП.

Використання результатів, отриманих в дисертаційній роботі Мартінкус І.О., дозволило підвищити якість процесу розробки окремих модулів для перспективної лінійки програмних продуктів за рахунок підвищення ступеню повторного використання вихідного коду, що забезпечило зменшення трудовитрат в окремих проектних завданнях.

Даний акт не є підставою для проведення будь-яких фінансових рохрахунків або отримання винагороди.

Голова комісії Марчук А.В

Члени комісії: Сагайдачний Д.О

Бабак О.Ю.

Проректор з науково-педагогічної роботи
Національного технічного університету
"Харківський політехнічний інститут"

_____ проф. Мигущенко Р.П.

"24" 10 2017 р.

АКТ

про впровадження результатів дисертаційної роботи
асистента кафедри програмної інженерії та інформаційних технологій управління
Мартінкус Ірини Олегівни
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗРОБКИ ЛІНІЙОК ПРОГРАМНИХ ПРОДУКТІВ НА ОСНОВІ МЕТОДІВ ТА ЗАСОБІВ ДОМЕННОГО МОДЕЛЮВАННЯ»

Ми, що нижче підписалися, декан факультету комп'ютерних наук та програмної інженерії (КН) проф. Гамаюн І.П., завідувач кафедри програмної інженерії та інформаційних технологій управління проф. Годлевський М.Д., професор кафедри програмної інженерії та інформаційних технологій управління проф. Ткачук М.В., доцент кафедри програмної інженерії та інформаційних технологій управління, к.т.н. Гамзаєв Р.О. склали наступний акт у тому, що результати дисертаційної роботи Мартінкус І.О. впроваджені в навчальний процес на кафедрі програмної інженерії та інформаційних технологій управління (ПІТУ).

Запропоновані у роботі моделі та процедури оцінки для визначення ефективності застосування методів та засобів доменного моделювання при розробці лінійок програмних продуктів (ЛПП) застосовані в лекційному матеріалі та в лабораторних практикумах до дисциплін «Аналіз вимог до програмного забезпечення», «Основи проектування програмного забезпечення» та «Аналіз та моделювання проблемно-орієнтованих програмних систем».

Декан КН-факультету
НТУ «ХПІ» д.т.н., проф. _____ Гамаюн І.П.

Зав. кафедри ПІТУ
НТУ "ХПІ", д.т.н., проф. _____ Годлевський М.Д.

Професор кафедри ПІТУ
НТУ "ХПІ", д.т.н., проф. _____ Ткачук М.В.

Доцент кафедри ПІТУ
НТУ "ХПІ", к.т.н. _____ Гамзаєв Р.О.

ДОДАТОК Б.
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

1. Ткачук Н.В., Нагорный К.А., Мартинкус И.О. Методика оценки динамической сложности требований в процессах сопровождения унаследованных программных систем. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”.2010. № 67. с.116-125.

2. Tkachuk M., Martinkus I. Models and Tools for Multi-dimensional Approach to Requirements Behavior Analysis / ed. H.C. Mayr et al. UNISCON 2012, LNBIP 137: Springer-Verlag Berlin Heidelberg, 2013. pp. 191-198.

3. Гамзаев Р. О., Ткачук Н. В., Мартинкус И. О. Мета-модель процесса трассировки требований при разработке программного обеспечения. Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. Х: НТУ «ХПІ» 2014. № 26 (1069). с.121-128

4. Tkachuk M.V., Gamzaev R.A., Martinkus I.O., Ianushkevych S.D. Methods and Tools for Dynamic Requirements Catalog Management in Agile Software Development. Вісник Національного технічного університету "ХПІ" Серія: Системний аналіз, управління та інформаційні технології. Харків: НТУ ”ХПІ”. 2015. № 52(1058). с.11-18 .

5. Tkachuk, M., Martinkus, I., Gamzayev, R., Tkachuk A. An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development / ed. Heinrich C. Mayr, Martin Pinzger. INFORMATIK 2016. Lecture Notes in Informatics (LNI). Vol. P-259: Kollen Druck+Verlag GmbH, Bonn, 2016. pp. 143-156.

6. Мартинкус І.О., Ткачук М.В., Гамзаєв Р.О. Конструювання лінійок програмних продуктів із застосуванням доменного моделювання та метрик повторного використання коду. Системи управління, навігації та зв’язку: зб. наук. пр. ЦНДІ НУ. К., 2017. Вип. 3(43). с. 93-97.

7. Ткачук М.В., Мартінкус І.О., Нагорний К.А., Гамзаев Р. О. Про один підхід до оцінки ефективності застосування методів доменного моделювання при розробці сімейств програмних систем. Збірник наукових праць Харківського національного університету Повітряних Сил. 2017. №5(54), с. 45-54

8. Мартинкус И.О., Гамзаев Р.А., Ткачук Н.В. Модели и технологи разработки линеек программных продуктов. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXII міжнародної науково-практичної конференції, (15-17 жовтня 2014р., Харків) / за ред. проф. Товажнянського Л.Л. Харків, НТУ «ХПІ». С.13.

9. Martinkus I.O., Gamzayev R.A, Tkachuk, M. V. Towards Effectiveness Assessment of Domain Modeling Approaches to Software Development. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XXIII міжнародної науково-практичної конференції, (21-22 травня 2016р., Харків) / за ред. проф. Сокола Є.І. Харків, НТУ «ХПІ». С.25.

10. Ткачук М.В., Гамзаев Р.О., Мартінкус І.О. Підхід до розробки лінійок програмних продуктів на основі успадкованих програмних систем із використанням методів доменного моделювання. Теоретичні та прикладні аспекти побудови програмних систем:матеріали міжнародної наукової конференції, м. Київ, 5-9 грудня 2016р. / редкол. М.С. Нікітченко, Л.Б. Буй та ін. Кіровоград, «Центр оперативної поліграфії «Авангард». 2016. с. 236-241.

11. Мартінкус І.О. Інформаційна технологія розробки та супроводу лінійок програмних продуктів на основі методів та засобів доменного моделювання. Сучасні інформаційні технології. Матеріали засідань школи-семінару за 2016-2017 н. р. Харків: ХНУ імені Каразіна, 2017. с.24-28

ДОДАТОК В.
ПРИКЛАДИ ДОКУМЕНТАЦІЇ ПО РОЗРОБЦІ ІНСТРУМЕНТАЛЬНОГО ПРОГРАМНОГО ЗАСОБУ ДЛЯ ОЦІНКИ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ДОМЕННОГО ПРОЕКТУВАННЯ

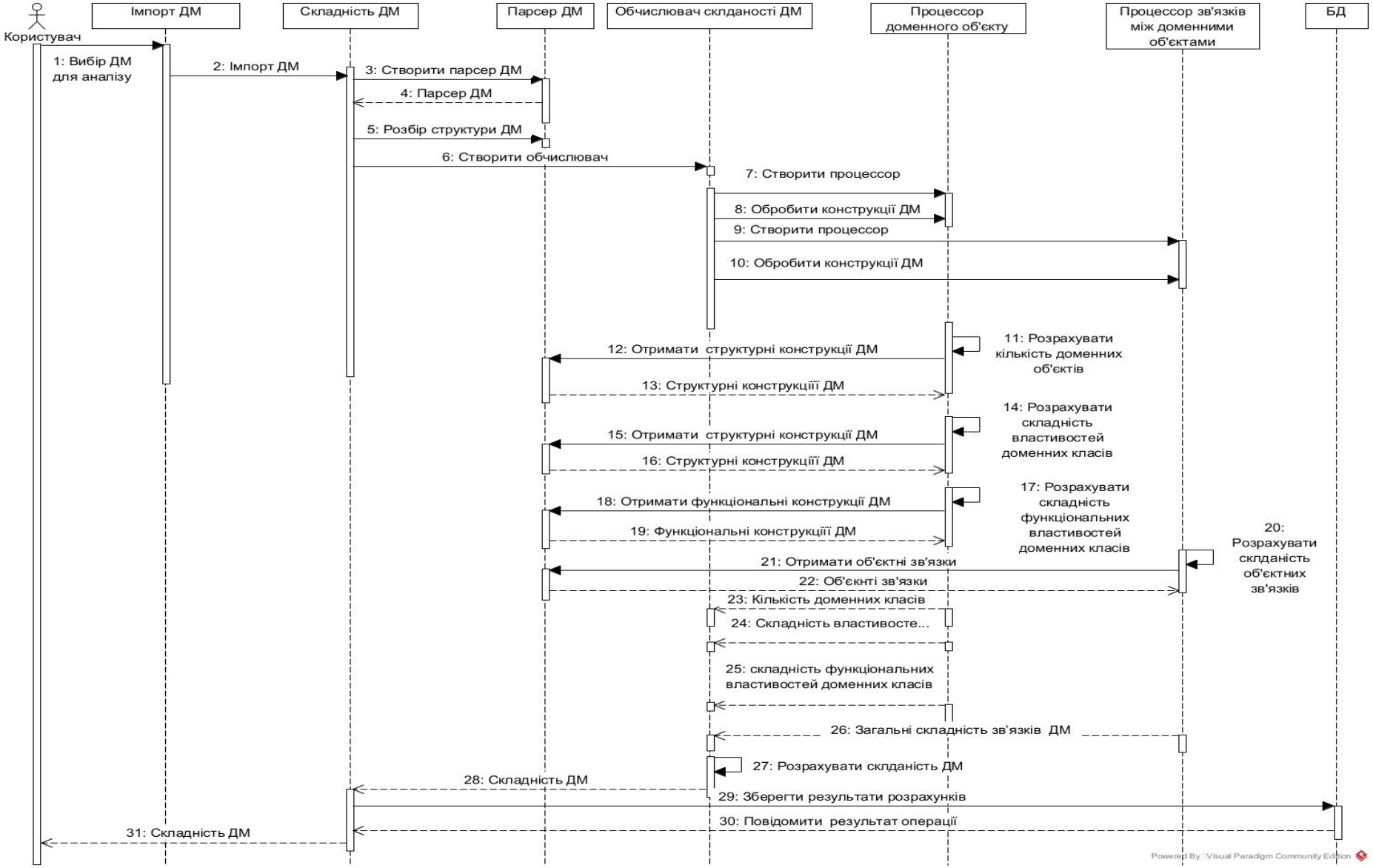

Рисунок В.1 – Діаграма послідовності для визначення структурно-функціональної складності ДМ

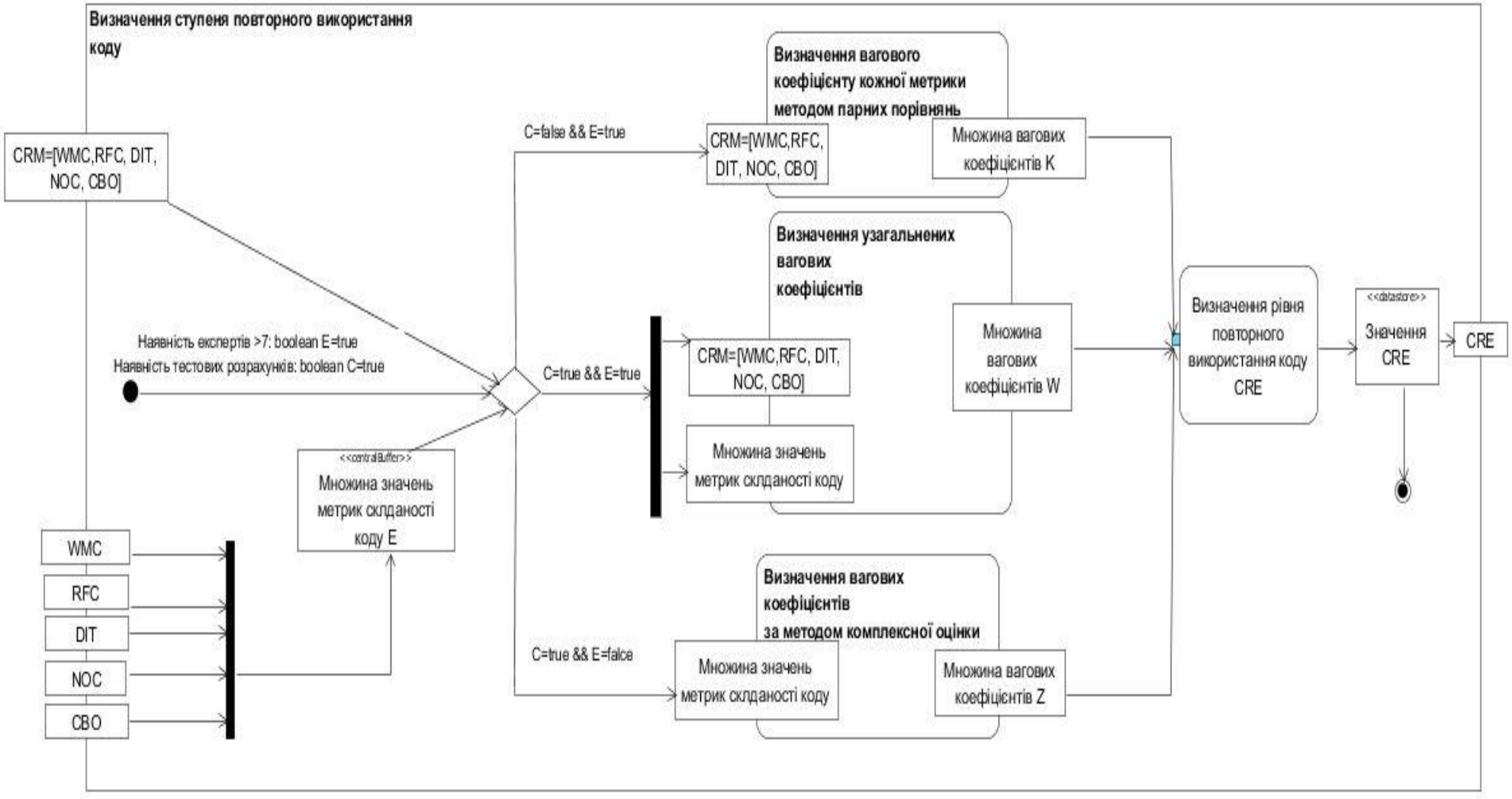

Рисунок В.2 – Діаграма активностей для визначення ступеня повторного використання коду

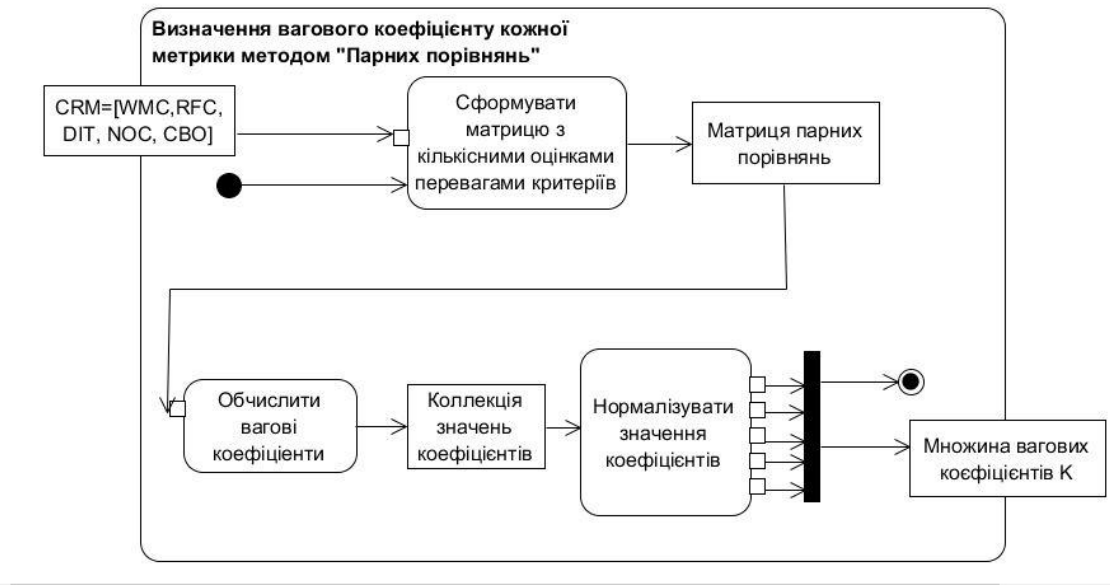

Рисунок В.3 – Діаграма активностей для визначення вагових коефіцієнтів методом парних порівнянь

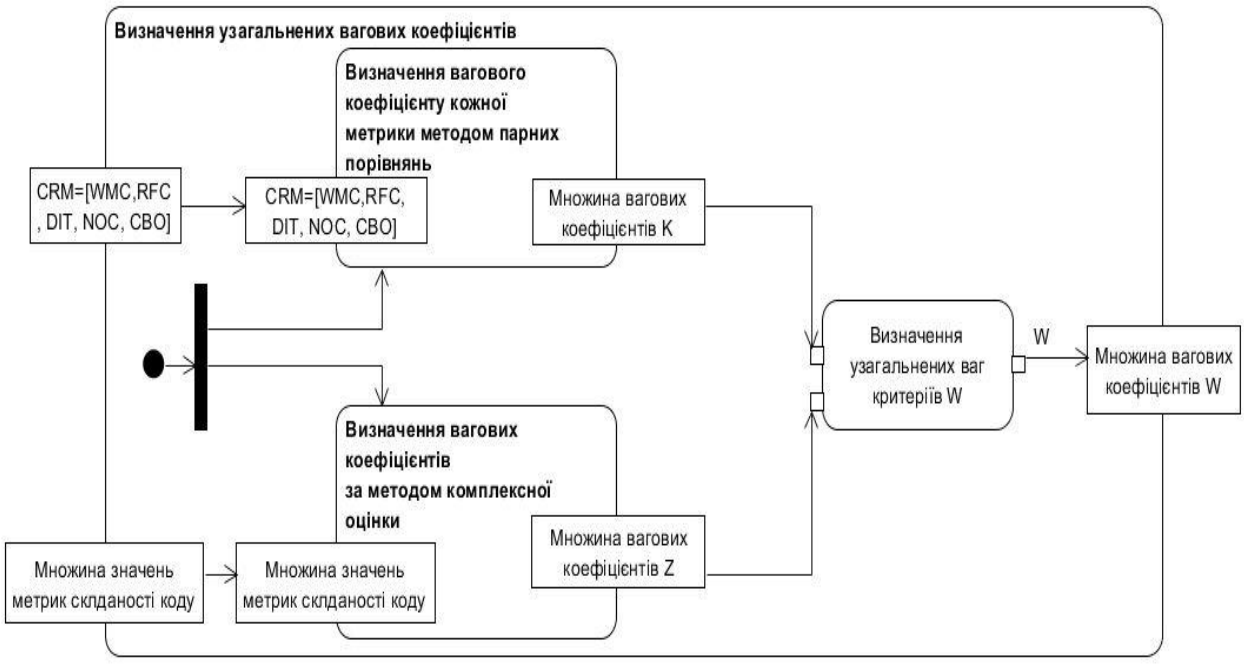

Рисунок В.4 – Діаграма активностей для визначення узагальнених вагових коефіцієнтів

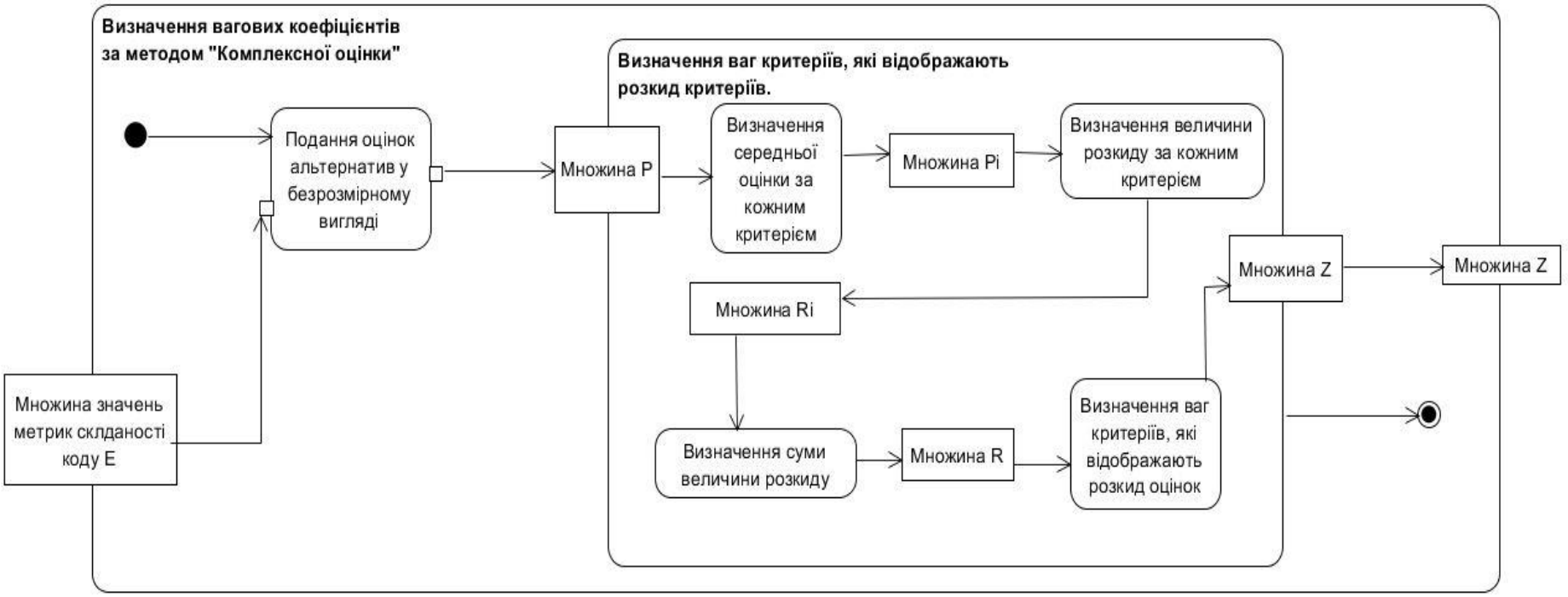

Рисунок В.5 – Діаграма активностей для визначення узагальнених коефіцієнтів методом «Комплексної оцінки»