С. С. Великодний

МОДЕЛІ ТА МЕТОДИ ПРОАКТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ З РОЗВИТКУ ПРОГРАМНИХ СИСТЕМ І ПРОДУКТІВ

Монографія

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

С. С. Великодний

МОДЕЛІ ТА МЕТОДИ ПРОАКТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ З РОЗВИТКУ ПРОГРАМНИХ СИСТЕМ І ПРОДУКТІВ

Монографія

Одеса
Одеський державний екологічний університет
2021

УДК 005.8: 001.51: 004.4'22
В 27

Великодний Станіслав Сергійович

В 27 Моделі та методи проактивного управління проєктами з розвитку програмних систем і продуктів: монографія. Одеса: Одеський державний екологічний університет, 2021. 322 с.

ISBN 978-966-186-182-3

Монографія розв'язує наукову проблему формування нових моделей та розробки нових методів проактивного управління проєктами з розвитку життєвого циклу програмних продуктів. Отримані результати забезпечують закладання методологічних основ проактивного управління проєктами з реінжинірингу програмних систем. Запропоновано методи проактивного управління проєктами розвитку наскрізного проєктування програмних систем. Розроблено моделі проактивного управління розвитком лінгвістичного та інформаційного забезпечення проєктів. Запропоновано моделі проактивного управління розвитком поведінкової та структурної частин проєктів з реінжинірингу програмних систем. Удосконалено метод розрахунку показників проєкту за проєктними точками Карнера. Сформовані ідеалізовані моделі управління процесом оцінки проєкту реінжинірингу. Розроблено метод представлення оцінки проєкту з реінжинірингу програмних систем. Підтверджено практичне використання запропонованих моделей та методів у вигляді низки впроваджень для вирішення реальних задач.

Монографія стане у нагоді магістрам і аспірантам, що навчаються за галуззю знань 12 – «Інформаційні технології», а також фахівцям у галузі програмної інженерії та комп'ютерних наук: системним архітекторам, аналітикам, експертам з аналізу даних, тестувальникам та проєктним менеджерам.

Velykodniy Stanislav

Models and methods of proactive project management for development of software systems and products: monograph. Odesa: Odessa State Environmental University, 2021. 322 p.

The monograph solves the scientific problem of formation of new models and design of new methods of proactive project management in the development of software product lifecycle. Systematization of leading project management methodologies has been further developed, it has been expanded by forming the concept of a proactive project management for the development of the life cycle of software systems and products. Models of the proactive management of projects linguistic and information support development have been formed. Models for proactive management of the development of behavioral and structural parts of software systems reengineering projects have been improved. The method of calculating project indicators by Karner project points has been improved. Idealized models for managing the process of reengineering project evaluation have been developed. The method for presenting the software systems reengineering project assessment has been developed. The practical use of the proposed models and methods in the form of a number of implementations to solve real problems is confirmed.

The monograph will be useful for masters and postgraduate students studying in the field of knowledge 12 – "Information Technology", as well as specialists in software engineering and computer science: systems architects, analysts, data analysis experts, testers and project managers.

УДК 005.8: 001.51: 004.4'22

Р е ц е н з е н т и :

О. Б. Зачко, д-р техн. наук, професор, Львівський державний університет безпеки життєдіяльності Державної служби України з надзвичайних ситуацій;

І. В. Гребеннік, д-р техн. наук, професор, Харківський національний університет радіоелектроніки;

В. В. Любченко, д-р техн. наук, професор, Одеський національний політехнічний університет

Рекомендовано до друку рішенням вченої ради Одеського державного екологічного університету Міністерства освіти і науки України (протокол №6 від 30.06.2021 р.)

ISBN 978-966-186-182-3

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АРМ – автоматизоване робоче місце.

АСКОЕ – автоматизована система комерційного обліку електроенергії.

БД – база даних.

ВВ – варіант використання.

ГБД – графічні бази даних.

ДМА – державна метрологічна атестація.

ДМД – динамічна модель діяльності.

ДО – діюча особа.

ДПРД – доріжка розділення проєктної діяльності.

ЖЦ – життєвий цикл.

ІМУПР – ідеалізована модель управління проєктом реінжинірингу.

ІП – інваріантний постпроцесор.

КП – керуюча програма.

КПВ – компоненти повторного використання.

ЛЗ – лінгвістичне забезпечення.

МВВ – модель варіантів використання.

МГ – мережевий графік.

МДРВ – масив даних ручного введення.

МЗД – масив звітних даних.

МКР – модель компонентних рішень.

МНД – масив необроблених даних.

МООО – модель опису оточення об’єктів.

МОПК – модель операцій із проєктними класами.

МОС – модель опису станів.

МП – мова програмування.

МПР – модель послідовності розвитку.

МУІР – модель управління інформаційним розгортанням.

МУП – модель управління пакетами.

НДР – науково-дослідної роботи.

ННЦ – національний науковий центр.

НУ «ОМА» – національний університет «Одеська морська академія».

ОДЕКУ – Одеський державний екологічний університет.

ОМ – об'єктна модель.

ООП – об’єктно-орієнований підхід.

ПЗ – програмне забезпечення.
ПМК – програмно-методичний комплекс.
ПП – програмний продукт.
ПС – програмна система.
САП – система автоматизованого проєктування.
САПР – система автоматизації проєктувальних робіт.
СУБД – система управління базами даних.
СЧПУ – система числового програмного управління.
УП – управління проєктами.
ЦППР – цільова програма проактивного розвитку.

BRL – Ballistic Research Laboratory.
CASE – Computer-Aided Software Engineering.
CBD – Component-Based Development.
CSG – Constructive Solid Geometry.
DHR – Default Hourly Rate.
DR – Duration.
DSL – Domain Specific Language.
EC – ECF Constant.
ECF – Environment Complexity Factor.
ERP – Enterprise Resource Planning.
EWE – Estimated Work Effort.
EWF – ECF Weight Factor.
GDM – Generative Domain Model.
GNU – General Public License.
GUI – Graphic User Interface.
GUI – Graphical User Interface.
IDL – Interface Definition Language.
PERT – Program (Project) Evaluation and Review Technique.
RTF – Rich Text Files.
SCADA – Supervisory Control And Data Acquisition.
TC – TCF Constant.
TCF – Technical Complexity Factor.
TWF – TCF Weight Factor.
UCP – Use Case Point.
UEV – Unadjusted ECF Value.
UG – Unigraphics.

UML – Unified Modeling Language.
UTV – Unadjusted TCF Value.
UUCP – Unadjusted Use Case Points.
VLDB – Very Large DataBase.

ПЕРЕДМОВА

Залишені без втручання, ~~події~~
[програмні системи – *авт.*]
мають тенденцію розвиватися від поганого до гіршого.
5-й висновок із закону Е. Мерфі
(*Edward A. Murphy, Jr.*)

Наукова робота автора над цією монографією тривала дванадцять років: з 2009 по 2021 рр. Усі складові компоненти монографії представлялися та обговорювалися на різноманітних наукових заходах, а саме на: 14-му Міжнародному молодіжному форумі «Радиоэлектроника и молодёжь в XXI веке» (Харків, 2010); 20-й Міжнародній конференції «New leading technologies in machine building» (с. Рибаче, АРК, 2010); 10-й, 11-й Всеукраїнській науково-технічній конференції «Математичне моделювання та інформаційні технології» (Одеса, 2011, 2012); 5-й Всеукраїнській науково-практичній конференції «Информационные технологии и автоматизация» (Одеса, 2012); 9-й та 12-й Міжнародній конференції «Strategy of Quality in Industry and Education» (Болгарія, м. Варна, 2013, 2016); 1-й, 5-й – 8-й Міжнародних науково-практичних конференціях «Information Control Systems and Technologies» (Одеса, 2013, 2016 – 2019); науково-технічній конференції «Енергетика судна: Експлуатація та ремонт» (Одеса, 2014); 14-й науковій конференції молодих вчених ОДЕКУ (Одеса, 2015); 16-й Міжнародній науково-практичній конференції «Современные информационные и электронные технологии» (Одеса, 2015); 22-й міжнародній конференції з управління «Автоматика 2015» (Одеса, 2015); міжнародній науково-практичній конференції «Новітні технології в автомобілебудівництві та транспорті» (Харків, 2015); 1-й та 3-й Всеукраїнській науково-практичній конференції «Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук» (Одеса, 2016, 2018); Всеукраїнській науково-методичній конференції «Управління якістю підготовки фахівців» (Одеса, 2017); 18-й Міжнародній конференції «Dynamical system modeling and stability investigation», (Київ, 2017); 1-му – 3-му Всеукраїнських пленерах з питань природничих наук (Одеса, 2017 – 2019); Міжнародній конференції «Advances in Quantum Systems in Chemistry, Physics and Mathematics»

(Харків, 2017); 35-й міжнародній науковій інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (Тернопіль, 2019); 4-й Всеукраїнській науково-практичній конференції «Інформаційні технології в моделюванні» (Миколаїв, 2019); 16-й Всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» (Одеса, 2019); Міжнародній науковій конференції «Економіко-екологічні проблеми сучасності у дослідженнях науковців» (Одеса, 2019); міжнародній науково-практичній конференції «Інтелектуальні системи та інформаційні технології ISIT-2019» (Одеса, 2019).

Окремо хочу висловити подяку своїй дружині – Зайцевій-Великодній Світлані Сергіївні, яка завжди підтримувала мене та надихала на плідну наукову працю, що стосується не тільки цієї монографії, а всієї наукової діяльності в цілому.

Доктор технічних наук,
професор кафедри
автоматизованих систем моніторингу
навколишнього середовища
Одеського державного
екологічного університету
Великодний Станіслав Сергійович

ВСТУП

Управління проєктами (УП) настільки щільно увійшло до нашого життя, що неможливо уявити собі існування та розвиток людства без елементів проєктної діяльності. Методи УП застосовуються у різноманітних галузях життя й діяльності людини, але найбільше поширення вони отримали у галузях, де необхідною є робота із сукупністю багатьох та не завжди послідовних розгалужених операцій – це стосується промисловості, виробництва, транспорту, навчання, інфокомунікацій та інформаційних технологій.

УП у цих галузях відбувається за допомогою програмних систем (ПС), що є результатами завершених проєктів з їх створення. УП зі створення потужної галузевої ПС – складна і трудомістка задача, виконання якої під силу тільки великому висококваліфікованому і злагодженому колективу розробників, оскільки тільки сам процес створення вже містить у собі понад десяток стадій: передпроєктні (у тому числі соціологічні) дослідження, технічне завдання, технічна пропозиція, ескізний проєкт, технічний проєкт, робочий проєкт, виготовлення (кодування), налагодження, випробування (тестування), уведення в дію (інсталяція та конфігурування) та подальший технічний супровід та підтримка проєкту.

Завжди перед розробником ПС поставало питання: у якому вигляді передати завершений програмний продукт (ПП). У деяких випадках замовник сам звужував рамки цієї дилеми та вимагав інсталяційний пакет або вже зібрані його компоненти, також певні обмеження, як обов’язкова умова, накладає технічне завдання до ПП. Проте все це – є етапами вже налагодженого процесу УП, участь у якому беруть менеджери проєкту, системні архітектори, прикладні програмісти, проєктні аналітики та ін.

Зараз все частіше перед великою корпорацією (а саме такі створюють ПП, які дійсно популярні) постає проблема розширення ринку за рахунок підключення принципів крос-платформності та мультимовності.

Перший являє собою сумісність різноманітних операційних систем, що стало особливо актуальним у зв’язку із вільним розповсюдженням UNIX-подібних систем, і саме при застосуванні цього принципу випливає неготовність багатьох виробників ПС підтримувати відкриті платформи.

Другий – це, взагалі, принцип, із яким пов’язана одна з найважливіших **проблем** УП зі створення ПС та ПП, а саме уніфікація або універсальність.

Цю проблему, можна віднести до «проблеми розвитку», під якою слід розуміти, що всі сучасні ПП (різного галузевого призначення), на превеликий жаль, сучасними не є. Це пов’язано, перш за все, з тим, що створювались вони на тих мовах програмування (МП), які були актуальні у самому початку їх розробки. Більшість з них через 2 – 3 роки не витримує підвищених вимог щодо швидкості роботи з відтвореним графічним зображенням та його обчислювальним відновленням (рендерінгом), а трансформація вихідного коду з однієї МП в іншу, виходячи з того, що сучасні ПП можуть складатися з декількох мільйонів рядків коду, може зайняти багато місяців.

Крім того, звичайно, що у процесі експлуатації ПС, під впливом чинників технічної еволюції проєкту, відбувається старіння видів забезпечення й така тенденція призводе до погіршення швидкісних, інформаційно-комунікаційних, графічних, часових та інших характеристик, аж до повної відмови ПС.

Таким чином при розгляді таких проєктів, виникає **протиріччя**: з одного боку необхідне прискорення темпів розробки ПС з урахуванням зростаючого рівня науково-технічних досягнень, з другого – якісне та повне наслідування методів та технічних засобів УП з розвитку ПП, що знаходиться у кількарічній експлуатації.

Для подолання цього протиріччя у нагоді стануть проактивні моделі та методи реінжинірингу ПП, тобто докорінної переробки ПС із наслідуванням позитивних якостей та відділенням й відмовою від негативних. Реінжиніринг дозволяє виконати розвиток (еволюціонування) ПС, шляхом внесення позитивних змін до її структури з метою покращення характеристик експлуатації та технічного супроводу.

Подібні процеси управління еволюційним удосконаленням ПС та ПП, що починаються з перепроєктування, вимагають обов’язкової **достовірної** аналітичної оцінки показників УП, адже існують випадки, коли проєкт реінжинірингу нерентабельний стосовно конкретної ПС.

Таким чином, тема дослідження, що передбачає створення моделей та методів проактивного УП та програмами з удосконалення ПС, розробку комплексного достовірного інструменту оцінки ПП, що здатен прогнозувати рентабельність перепрограмування ПС, у сполученні із

іншими показниками проєктної діяльності, які визначають складність еволюційного розвитку ПС та ПП є актуальною.

Зв'язок роботи з науковими програмами, планами темами.

Монографія виконувалась згідно із:

– основними завданнями Державної програми «Інформаційні та комунікаційні технології в освіті і науці», затв. постановою Кабінету Міністрів України №1153 від 07.12.2005 р.;

– технічним завданням до держбюджетної науково-дослідної роботи (НДР), що виконувалась у Одеському державному екологічному університеті (ОДЕКУ), «Вдосконалення методів інформаційних технологій з метою їх використання в дослідженні об'єктів довкілля та у процесі підготовки фахівців», № ДР 0114U000627; дата супровідного листа 06-13/295 від 19.03.14; код за ЄДРПОУ 26134086, у якій автор монографії виступав у ролі відповідального виконавця;

– технічним завданням до НДР, що виконувалась у національному університеті «Одеська морська академія» (НУ «ОМА») «Акустична система моніторингу терористичних погроз на водному транспорті», № ДР 0115U003576, науковий керівник – д. т. н., проф. Вишневський Л. В., у якій автор монографії на посаді старшого наукового співробітника виступав у ролі відповідального виконавця розділів «Створення алгоритмічного забезпечення інформаційної технології ідентифікації сигналів» та «Створення ПЗ програмно-апаратних комплексів напівнатурного моделювання об’єктів»);

– технічним завданням до держбюджетної НДР, що виконувалась у НУ «ОМА», «Удосконалення систем моніторингу та дистанційного управління рухом судна», УДК 629.5.0613:004, ДР № 0116U002387, підстава для проведення робіт: 34, лист 10/424; 19.02.2016 р., у якій автор монографії був науковим керівником.

Мета і завдання дослідження.

Мета роботи – сформувати моделі та розробити методи, які забезпечать методологічні основи розв’язання проблем проактивного управління виконанням проєктів з реінжинірингу ПП, що дозволять працювати з багаторівневими ПС, за допомогою комплексного інструментарію проєктної оцінки.

Для досягнення мети роботи, необхідно вирішити низку встановлених *завдань*:

а) провести аналіз методологій УП зі створення відкритих, вільних та комерційних ПП; розглянути забезпечення та умови розвитку проєктів з реінжинірингу ПС та ПП, відповідно до змісту стандарту ISO/DIS 21500, для з'ясування проблем проактивного УП з розвитку ПС та ПП;

б) систематизувати провідні методології УП у концепцію проактивного УП з розвитку життєвого циклу (ЖЦ) ПС та ПП;

в) формалізувати методи проактивного УП розвитку наскрізного проєктування ПС за допомогою парадигми системної трансформації та інтеграції ПП у проєкт;

д) сформувати моделі проактивного управління розвитком:

1) лінгвістичного та інформаційного забезпечення проєктів за допомогою динамічного оточення породжувальних граматик;

2) поведінкової та структурної частин проєктів з реінжинірингу ПС;

е) сформувати моделі управління процесом комплексної оцінки проєкту реінжинірингу ПС на підставі ідеалізованого подання їх розвитку та із уведенням ресурсних та часових обмежень на виконання проєкту;

ж) розробити комплекс методів оцінки проєкту реінжинірингу ПС у вигляді:

1) методу розрахунку показників проєкту;

2) методу представлення оцінки проєкту реінжинірингу ПС за допомогою проєктних коефіцієнтів;

и) узагальнити результати комплексних експериментальних досліджень з проактивного УП з розвитку ПС та сформульовано рекомендації із розробки системної архітектури ПП, що планується розвивати;

к) розробити програмно-методичний комплекс (ПМК) управління плануванням реінжинірингу ПС та ПП для команди проєктного менеджменту офісу УП.

Об'єктом дослідження є процес проактивного управління проєктами в умовах обмеження на час їх розвитку та ресурсів для їх реалізації.

Предметом дослідження – моделі та методи управління проєктами реінжинірингу програмних систем, який необхідний для еволюції життєвого циклу унікального програмного продукту.

Методи дослідження. У монографії використовуються наступні наукові методи. Метод Боема (Boehm), що знайшов своє продовження у аналітичних моделях із запропонованими змінами та пропозиціями щодо гнучкості побудови на етапі передпроєктного аналізу. Метод проєктних

точок Карнера (Karner) та метод проєктних коефіцієнтів, що дозволяють формувати оцінку проєктних витрат. Метод констант Якобсона (Jacobson) із доданими доповненнями та розширеннями, що відображені у методі розрахунку проєктних показників. Метод проєктних діаграм Ґанта (Gantt), за якими здійснюється облік запланованого часу виконання проєкту. Метод PERT (Project Evaluation and Review Technique) – техніка оцінки та аналізу проєктів, що використовується при УП. Методи побудови спіральних моделей Архімеда та векторного представлення годографів Гамільтона та Михайлова. Також у монографії використовуються методи конкретизуючого, синтезуючого, композиційного програмування; методи дедукційного (згори-униз) та індукційного (знизу-догори) програмування; методи складального програмування (побудова програмних інтерфейсів) та управління модульними структурами як алгебраїчними системами.

Наукова новизна отриманих результатів полягає у створенні нових моделей та методів проактивного УП з розвитку ЖЦ ПП та закладанні методологічних основ розв’язання проблем проактивного УП з реінжинірингу ПС, а саме:

– вперше формалізовано методи проактивного УП розвитку наскрізного проєктування ПС, які відрізняються формуванням парадигми системної трансформації, що дозволяє інтегрувати ПП у оновлені проєктні комплекси;

– вперше сформовано моделі проактивного управління розвитком лінгвістичного та інформаційного забезпечення проєктів, які відрізняються застосуванням динамічного оточення породжувальних граматик, що дозволяє позбавитися виведення тупикових ланцюжків при переведенні з однієї МП до іншої;

– вперше розроблено метод представлення оцінки проєкту з реінжинірингу ПС, який відрізняється застосуванням інструментарію проєктних коефіцієнтів, що дозволяє управляти якісними змінами конфігурації графічних моделей перепроєктування, у вигляді відповідних годографів проєктів реінжинірингу ПС;

– вперше сформовані ідеалізовані моделі управління процесом оцінки проєкту реінжинірингу, які відрізняються уведенням ресурсних та часових обмежень на виконання проєкту, що дозволяє підвищити точність оцінки проєктних витрат з перепроєктування ПС;

– удосконалено моделі проактивного управління розвитком поведінкової та структурної частин проєктів з реінжинірингу ПС, які

відрізняються уведенням оновлених структур уніфікованих діаграмних моделей, що дозволяє зберігати позитивні властивості відношень між проєктними класами, компонентами та сутностями ПС незалежно від МП;

– удосконалено метод розрахунку показників проєкту за проєктними точками Карнера, який відрізняється внесенням доповнень та розширень щодо процесів реінжинірингу ПС, що дозволяє зменшити похибку в оцінюванні прогнозованого переліку витрат на реалізацію проєкту у порівнянні із методом Карнера;

– дістала подальшого розвитку систематизація провідних методологій УП, яка розширена формуванням концепції проактивного УП з розвитку ЖЦ ПС та ПП, що дозволяє на етапі оцінки проєкту визначити доцільність цільової програми реінжинірингу.

Практичне значення одержаних результатів. Підтверджена можливість практичного використання розроблених моделей та методів проактивного УП з розвитку ПС та ПП у вигляді низки впроваджень наукових результатів, що засвідчено відповідними актами й показало свою достовірність.

Експериментальні дослідження спіральної ідеалізованої моделі управління процесом оцінки проєкту реінжинірингу за даними 7 виконаних проєктів показали прогнозування витрат на проєкт реінжинірингу ПС із максимальною похибкою 30%, що для первинної оцінки порядку витрат (при відсутності інших даних на етапі прийняття рішення про застосування проєкту реінжинірингу), цілком придатне.

При використанні методу розрахунку показників проєкту за проєктними точками та методу представлення оцінки проєкту з реінжинірингу ПС за допомогою інструментарію проєктних коефіцієнтів, виконано оцінку кожного етапу проєкту з удосконалення ПС у Одеському ім. А. Міцкевича відділенні Спілки поляків в Україні із отриманням економічного показника впровадження у вигляді оцінки вартості кожного окремого етапу проєкту із удосконалення ПС, які не відрізняються від фактичних більш ніж на 7 %.

Розроблений ПМК управління плануванням реінжинірингу проєктів містить у собі сукупність програмного, інформаційного, математичного, організаційного та методичного забезпечення для виконання робіт із планування та організації розвитку ПС та ПП.

При застосуванні у ТОВ «Люксстройпроєкт» (м. Харків) ПМК стосовно до УП, досягнуто скорочення строків проєктування виробничого

процесу за методологією мережевого планування за рахунок прискорення складання мережевих графіків організації виробництва, внаслідок використання інтерактивних методів управління мережевою моделлю. Зафіксовано скорочення затраченого часу при проєктуванні мережевих графіків у межах: 16% ÷ 24%; при трансформації робіт у події та навпаки: 87% ÷ 96%.

При плануванні та управлінні життєвим циклом рекламної продукції ТОВ «Рекламне агентство «ТРАСТ МЕДІА», що займається її виготовленням, розміщенням, аналізом та просуванням, знайшли своє застосування розроблені методи та засоби управління мережевим плануванням проєкту реінжинірингу ПП.

Застосування ПМК управління мережевим плануванням проєкту реінжинірингу, як складової інформаційної технології із організації створення та просування рекламного продукту, скоротило: на 22% процес складання мережевого графіка у порівнянні із його складанням до застосування ПМК; на 17% процес підрахунку часу, що закладається у базову траєкторію ЖЦ проєкту; на 8% процес аналізу ефективності рекламного продукту для прийняття рішення щодо його подальшого реінжинірингу та застосування конкретних методів розвитку й просування.

Розроблені моделі та методи стануть у нагоді керівникам проєктів, проєктним менеджерам, системним архітекторам та інженерам-програмістам, які задіяні у перепроєктуванні ПС та ПП, що вже знаходяться у кількарічній експлуатації.

Запропоновані моделі та методи, а також ПМК проактивного УП з розвитку ПС та ПП, впроваджено у НДР та навчальний процес у: Одеському державному екологічному університеті, національному університеті «Одеська морська академія» та Одеській національній академії зв'язку ім. О. С. Попова, що підтверджено відповідними актами (додаток Б).

Особистий внесок здобувача. Усі отримані наукові результати, що внесено до монографії, опубліковані у роботах [1] – [73] та отримані автором особисто. У наукових працях, що виконано у співавторстві, авторові належить: у [3] – складання структурних моделей проєкту; у [4] –

---

1. Velykodniy S. Reengineering of open software system of 3D modeling BRL-CAD. *Innovative Technologies and Scientific Solutions for Industries.* 2019. No. 3 (9), P. 62–71. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.9.062.

моделювання архітектури програмного засобу для управління мережевим плануванням; у [5] – моделювання програмного забезпечення (ПЗ) для управління проєктуванням мережевих графіків; у [6] – складання поведінкових моделей проєкту; у [9] – формування методу розрахунку показників оцінки проєкту; у [10] – виконання порівняльного аналізу властивостей програмних продуктів; у [11] – формування та складання

---

2. Velykodniy S. S. Analysis and synthesis of the results of complex experimental research on reengineering of open CAD systems. *Applied Aspects of Information Technology.* 2019. Vol. 2. No 3. P. 186–205. (кат. «Б») DOI: 10.15276/aait.03.2019.2.

3. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проектування BRL-CAD. Моделювання структурної частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського.* 2019. Вип. 3 (116). С. 130–139. (кат. «Б») DOI: 10.30929/1995-0519.2019.3.130-139.

4. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Розробка архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. *Сучасний стан наукових досліджень та технологій в промисловості.* 2019. № 2 (8). С. 25–35. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.8.025.

5. Velykodniy S., Burlachenko Zh., Zaitseva-Velykodna S. Software for automated design of network graphics of software systems reengineering. *Scientific Journal Herald of Advanced Information Technology.* 2019. No 2 (03). P. 20–32. (кат. «Б») DOI://10.15276/hait.02.2019.2.

6. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проектування BRL-CAD. Моделювання поведінкової частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського*. 2019. Вип. 2 (115). С. 117–126. (кат. «Б») DOI: 10.30929/1995-0519.2019.2.117-126.

7. Великодний С. С. Метод представлення оцінки реінжинірингу програмних систем за допомогою проектних коефіцієнтів. *Сучасний стан наукових досліджень та технологій в промисловості.* 2019. № 1 (7). С. 34–42. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.7.034.

8. Великодний С. С. Ідеалізовані моделі реінжинірингу програмних систем. *Радіоелектроніка, інформатика, управління.* 2019. № 1. С. 150–156. DOI: 10.15588/1607-3274-2019-1-14.

9. Великодний С. С., Тимофєєва О. С., Зайцева-Великодна С. С. Метод розрахунку показників оцінки проекту при виконанні реінжинірингу програмних систем. *Радіоелектроніка, інформатика, управління.* 2018. № 4. С. 135–142. DOI: 10.15588/1607-3274-2018-4-13.

10. Великодний С. С., Тимофєєва О. С., Зайцева-Великодна С. С., Нямцу К. Є. Порівняльний аналіз властивостей відкритого, вільного та комерційного програмного забезпечення. *Інформаційні технології та комп'ютерна інженерія*. 2018. № 1 (41). С. 21–27.

11. Великодний С. С., Тимофєєва О. С. Спосіб мультилінгвістичного перекодування програмного забезпечення складних інформаційних систем та технологій. *Наукові праці ОНАЗ ім. О. С. Попова*. 2017. № 2. С. 153–159.

способу мультилінгвістичного перекодування ПС; у [12] – модель представлення лінгвістичного забезпечення ПС; у [13] – постановка задачі та математичний виклад лінгвістичного забезпечення ПС; у [14] – математичне представлення та виведення граматик; у [15] – формування метода та екстраполяція його на SCADA-системи; у [19] – характеристика програмних комплексів та дослідження властивостей ПС; у [21] – дослідження та класифікаційне порівняння алгоритмів адаптивного управління щодо ПС; у [23] – дослідження властивостей та проєктування годографів Михайлова за допомогою програмних продуктів; у [24] –

---

12. Velykodniy S., Tymofieieva O. The paradigm of linguistic supply submission by generative grammar assistance. *American scientific journal*. 2017. № 17. P. 4–7.

13. Великодний С. С., Тимофєєва О. С. Парадигма подання лінгвістичного забезпечення за допомогою породжувальних граматик (Част. 2). *Automation of technological and business-processes*. 2017. Vol. 9, Iss. 3. С. 46–50.

14. Великодний С. С., Тимофєєва О. С. Парадигма подання лінгвістичного забезпечення за допомогою породжувальних граматик (Част. 1). *Розвиток транспорту*. 2017. № 1. С. 128–135.

15. Velykodniy S., Tymofieieva O. Multilingual recording method designed for SCADA-system's software upgrade. *Automation of technological and business-processes.* 2017. Vol. 9, Iss. 1. P. 17–22.

16. Великодний С. С. Методи реінжинірингу програмних систем. *Технологии приборостроения*. 2014. Спец. вып. С. 65–68.

17. Великодный С. С. Методологические основы реинжиниринга систем автоматизированного проектирования. *Управляющие системы и машины*. 2014. № 2. С. 39–43.

18. Великодный С. С. Проблема реинжиниринга видов обеспечения систем автоматизированного проектирования. *Управляющие системы и машины*. 2014. № 1. С. 57–61, 76.

19. Зайцева-Великодна С. С., Великодний С. С. Проблеми атестації обчислювальної компоненти програмних комплексів автоматизованих систем комерційного обліку електроенергії. *Системы обработки информации*. Вып. 99. 2012. С. 152–156.

20. Великодный С. С. Реализация процесса «сквозного» проектирования с помощью CAD/CAM «ADEM». *Холодильна техніка і технологія.* 2011. № 1 (129). С. 56–59.

21. Великодный С. С., Котенко Е. В. Разработка системы реального времени распознавания лиц с использованием примитивов Хаара и алгоритма ADABOOST. *Холодильна техніка і технологія*. 2010. №5 (127). С. 67–70.

22. Великодний С. С. Проектування траєкторії руху технологічного обладнання при виготовленні елементів суднових конструкцій. *Автоматизация судовых технических средств*. Вып. 16. 2010. С. 10–18.

23. Невлюдов И. Ш., Великодный С. С., Фомовская Е. В. Построение годографов Михайлова с помощью пакета «MathCAD». *Вопросы проектирования и производства конструкций летательных аппаратов.* Вып. 3 (63). 2010. С. 186–193.

24. Невлюдов І. Ш., Великодний С. С., Фомовська О. В. Моделювання електромеханічної частини маніпулятора промислового робота. *Вопросы*

моделювання точності програмного управління; у [25] – дослідження інтерфейсів програмних продуктів, виявлення та опис проблеми управління масивами звітних даних; у [26] – виконання аналізу використання CAD/CAM/CAE-систем; у [27] – складання структурних моделей проєкту; у [28] – діаграмні моделі управління проєктуванням мережевих графіків; у [29] – моделювання системної архітектури програмної системи; у [30] – складання поведінкових діаграмних моделей проєкту; у [31] – складання структурних діаграмних моделей проєкту; у [32] – аналіз продуктивності ПС; у [33] – формування методу

проектирования и производства конструкций летательных аппаратов. Спец. вып. 2010. С. 181–185.

25. Великодний С. С., Стрілець В. О. Деякі питання атестації автоматизованих систем комерційного обліку електроенергії. *Український метрологічний журнал.* 2010. № 2. С. 40–44.

26. Невлюдов И. Ш., Великодный С. С., Омаров М. А. Использование CAD/CAM/CAE/CAPP при формировании управляющих программ для станков с ЧПУ. *Восточно-Европейский журнал передовых технологий*. 2010. № 2/2 (44). С. 37–44.

27. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Graphic databases reengineering in BRL-CAD open source computer-aided design environment. Modeling of the structural part. Information control systems and technologies: VIII international scientific-practical conference: materials. Odessa. Ukraine, 23 – 25 sep. 2019. Odessa: «Ecologia», 2019. P. 222–224.

28. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Software for automated design of network graphics of software systems reengineering. Матер. міжн. наук.-практ. конф. «Інтелектуальні системи та інформаційні технології ISIT-2019». Одеса. Україна. 19 – 24 серпня 2019 р. Одеса: ТЕС, 2019. С. 253–257.

29. Burlachenko T., Великодний С. С. Розробка системної архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. Міжн. наук. конф. «Економіко-екологічні проблеми сучасності у дослідженнях науковців». м. Одеса. Україна. 25 – 26 черв. 2019 р. Одеса: ТЕС, 2019.С. 20–26.

30. Burlachenko Zh. V., Velykodniy S. S. Graphic databases reengineering in BRL-CAD open source computer-aided design environment. Modeling of the behavior part. Матеріали III-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 22 чер. 2019. С. 7–9.

31. Зайцева-Великодна С. С., Великодний С. С. Реінжиніринг графічних баз даних у середовищі відкритої САПР BRL-CAD. Моделювання структурної частини. Матеріали III-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 22 чер. 2019. С. 27–29.

32. Великодний С. С., Нямцу К. Є. Сучасна інформаційна система для підготовки LaTeX-документів «TeXstudio». Інформатика, інформаційні системи та технології: тези доп. 16-ї Всеукр. конф. студ. і мол. наук. Одеса, 19 квітня 2019 р. Одеса, 2019. С. 11–13.

33. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Method of presenting the assessment for reengineering of software systems with the project coefficients help. Інформаційні технології в моделюванні: матер. IV-ої всеукр. наук.-практ. конф.

представлення оцінки проєкту за допомогою проєктних коефіцієнтів; у [34] – аналіз ПС розмітки наукових текстів; у [35] – моделі реляційних відношень для задач автоматизованого планування; у [36] – формування методу розрахунку показників оцінки проєкту; у [37] – моделювання реінжинірингу ПС; у [39] – формування методу оцінки за допомогою проєктних показників; у [40] – моделі представлення реінжинірингу лінгвістичного забезпечення ПС; у [41] – представлення та виведення формальних граматик; у [42] – формування способу перекодування

студ., аспірантів та мол. вчених, 21–22 березня 2019 р., м. Миколаїв. Миколаїв: МНУ ім. В. О. Сухомлинського, 2019. С. 10–11.

34. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. LaTeX-орієнтовані системи підготовки наукових текстів. Міжн. наук. Інтернет-конф. «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (вип. 35)». Зб. тез доп.: вип. 35, м. Тернопіль, 5 лют. 2019 р. Тернопіль. 2019. С. 6–8.

35. Петухін Д. О., Великодний С. С., Козловська В. П. Застосування методів реляційної алгебри для задачі автоматизованого складання розкладу занять. Information Control Systems and Technologies: VII International scientific-practical conference: materials, Odessa, Ukraine, 17 – 18 sep. 2018. Odessa: «Astroprint», 2018. P. 80–83.

36. Тимофєєва О. С., Великодний С. С., Зайцева-Великодна С. С. Метод розрахунку показників оцінки проекту при виконанні реінжинірингу програмних систем. Матеріали ІІ-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 26 – 28 лип. 2018. С. 18–19.

37. Великодний С. С., Зайцева-Великодна С. С. Ідеалізовані моделі реінжинірингу програмних систем. Матеріали ІІ-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 26 – 28 лип. 2018. С. 15–17.

38. Великодний С. С. Ідеалізовані моделі реінжинірингу програмних систем. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: ІІІ Міжн. наук.-практ. конф., 6 – 8 черв. 2018 р. Одеса, 2018. С. 80–83.

39. Tymofieieva O., Velykodniy S., Zaitseva-Velykodna S. Method of calculating project evaluation indicators for the implementation of software system reengineering. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: ІІІ Міжн. наук.-практ. конф., 6 – 8 черв. 2018 р. Одеса, 2018. С. 68–71.

40. Velykodniy S., Tymofieieva O. Reengineering Models of Linguistic Providing Software Systems. Advances in Quantum Systems in Chemistry, Physics and Mathematics, Ser.: Progress in Applied Mathematics and Quantum Optics, Eds. A. Glushkov, O. Khetselius, V. Buyadzhi. Kharkiv: FOP Panov, 2017. P. 385–388.

41. Великодний С. С., Тимофєєва О. С., Нямцу К. Є. Парадигма подання лінгвістичного забезпечення за допомогою формальних граматик. Information Control Systems and Technologies: VI International scientific-practical conference: materials, Odessa, Ukraine, 20 – 22 sep. 2017. Одеса: «ВидавІнформ НУ «ОМА», 2017. С. 266–268.

42. Великодний С. С., Тимофєєва О. С. Спосіб мультилінгвістичного перекодування програмного забезпечення SCADA-систем. Матеріали І-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 23 лип. 2017. С. 7–8.

SCADA-систем; у [43] – огляд проблеми рефакторингу інтерфейсів ПС; у [44] – модель реінжинірингу видів забезпечення ПС; у [45] – порівняльний огляд підходів до наукової діяльності та методології управління науковим проєктом; у [46] – аналіз та систематизація методів реінжинірингу ПС; у [47] – метод лінгвістичного транслювання; у [51] – моделювання інформаційного забезпечення ПС; у [52] – управління напрямком

43. Великодний С. С., Сирчин Д. О. Аналітика та рефакторинг інтерфейсів CRM-систем Dynamical system modeling and stability investigation: XVIII International Conference: Abstract of Conf. Reports, Kyiv, Ukraine, 24 – 26 may 2017. Київ, ДП Інформ.-аналіт. агентство, 2017. С. 194.

44. Великодний С. С., Тимофєєва О. С. Ідеалізовані моделі реінжинірингу видів забезпечення програмних систем. Dynamical system modeling and stability investigation: XVIII International Conference: Abstract of Conf. Reports, Kyiv, Ukraine, 24 – 26 may 2017. Київ: ДП Інформ.-аналіт. агентство. 2017. С. 195.

45. Великодний С. С., Тимофєєва О. С. Порівняльний огляд європейської та вітчизняної наукової діяльності у галузі інформаційних технологій. Управління якістю підготовки фахівців: зб. тез доп. Всеукр. наук.-мет. конф., 21 – 22 лют. 2017 р. Одеса, 2017. С. 9–10.

46. Великодний С. С., Тимофєєва О. С. Базові методи реінжинірингу програмних компонентів. V International scientific-practical conference “Information Control Systems and Technologies”: materials ICST-Odessa, 20 – 22 sep. 2016. Odessa, 2016. С. 293–295.

47. Сырчин Д. А., Великодный С. С. Методы лингвистического транслирования языковых конструкций. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: Всеукр. наук-практ. конф., 20 – 22 квіт. 2016 р. Одеса, 2016. С. 100–101.

48. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров для систем автоматизированного проектирования формообразования деталей. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: Всеукр. наук-практ. конф., 20 – 22 квіт. 2016 р. Одеса, 2016. С. 32–33.

49. Великодний С. С. Модель реінжинірингу програмного забезпечення SCADA-систем, що застосовуються на транспорті. Новітні технології в автомобілебудівництві та транспорті: наук. праці міжнар. наук-практ. конф. до 85-річ. ХНАДУ, 15–16 жов. 2015 р. Харків, 2015. С. 104–105.

50. Великодний С. С. Реінжиніринг систем моніторингу та дистанційного управління судновими енергетичними установками. Матер. XXII міжн. конф. з автом. управл. «Автоматика 2015», 10–11 вер. 2015, Одеса, 2015. С. 133–134.

51. Великодний С. С., Тимофєєва О. С., Онищенко С. М. Моделювання інформаційного сполучення компонентів SCADA-систем. XII International Conference “Strategy of Quality in Industry and Education”: proceedings, 30 May – 2 June 2016, Varna, Bulgaria. С. 511–518.

52. Орлов В. В., Великодный С. С., Бережной К. Ю. Информационная технология совершенствования судовых систем приёма внешних звуковых сигналов. Современные информационные и электронные технологии: труды XVI междун. науч.-практ. конф., 25–29 мая 2015 г. Одесса, 2015. С. 107–108.

удосконалення інформаційної технології; у [55] – спосіб лінгвістичного транслювання мовних конструкцій ПС; у [60] – дослідження властивостей та проєктування годографів Михайлова за допомогою програмних продуктів; у [61] – моделювання точності програмного управління; у [62] – оптимізація проєктування виробничої мережі; у

---

53. Великодний С. С. Методологічні засади реінжинірингу SCADA-систем. Матер. конф. молод. вчених ОДЕКУ, 11 – 15 трав. 2015 р. Одеса, 2015. С. 95–97.

54. Великодный С. С. Специализированные САПР для моделирования и исследования судовых компьютерных сетей. Енергетика судна: Експлуатація та ремонт: матер. наук.-техн. конф., 26 – 28 бер. 2014 р. Ч. 2. Одеса, 2014. С. 116–117.

55. Великодный С. С., Сырчин Д. А. Способы лингвистического транслирования языковых конструкций. International scientific-practical conference “Information Control Systems and Technologies”: materials ICST-Odessa, 8 – 10 oct. 2013. Odessa, 2013. P. 318–321.

56. Великодный С. С. Особенности реинжиниринга программного обеспечения открытых САПР. IX International Conference “Strategy of Quality in Industry and Education”: proceedings. Vol. 1, Varna, Bulgaria, 31 May – 7 June 2013. Dnipropetrovsk-Varna, 2013. P. 304–307.

57. Великодний С. С. Передумови реінжинірингу систем автоматизованого проектування. Математичне моделювання та інформаційні технології: тези доп. XI Всеукр. наук.-техн. конф., 21 – 23 лист. 2012 р. Одеса, 2012. С. 23.

58. Великодний С. С. Методологія зворотного проектування програмного забезпечення САПР. Информационные технологии и автоматизация – 2012: тез. докл. V Всеукр. науч.-практ. конф., 10 – 11 окт. 2012 г. Одесса, 2012. С. 10–11.

59. Великодный С. С. Технико-экономический анализ внедрения системы автоматизированного производства. Математичне моделювання та інформаційні технології: тези доп. X Всеукр. наук.-техн. конф., 23 – 25 лист. 2011 р. Одеса, 2011. С. 24–25.

60. Невлюдов І. Ш., Великодний С. С., Фомовська О. В. Побудова годографів Михайлова за допомогою пакета «MathCAD». XX International conference “New leading technologies in machine building”: collect. of the scient. papers, Rybachie, 3 – 8 sept. 2010. Kharkov, 2010. P. 13.

61. Невлюдов И. Ш., Великодный С. С. Фомовская Е. В. Моделирование электромеханической части манипулятора промышленного робота. XX International conference “New leading technologies in machine building”: collect. of the scient. papers, Rybachie, 3 – 8 sept. 2010. Kharkov, 2010. P. 13.

62. Лалашкова А. И., Великодный С. С. Выбор технологии организации связи для рационального функционирования производственной сети. Радиоэлектроника и молодёжь в XXI веке: матер. XIV Межд. молод. форума 18 – 20 мар. 2010 г. Ч. 1. Харьков, 2010. С. 370.

63. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров для систем автоматизированного проектирования формообразования деталей. Монографія. Харків: ФОП Панов А.Н., 2017. 232 с.

64. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров. Для систем автоматизированного проектирования формообразования деталей. Монография. 2-е изд. Эрфурт: LAMBERT Academic Publishing, 2019. 236 с.

[71], [72], [73] – постановка задачі та імітаційні моделі програмної реалізації для кожної роботи.

---

65. Великодний С. С. Реінжиніринг програмних систем та продуктів. Управління IT-проектами. Монографія. Нордерштедт: LAMBERT Academic Publishing, 2019. 160 с.

66. Великодний С. С. Спосіб виявлення прихованого кабелю між поверхнями. Патент на кор. мод. № 119764, зар. в Держ. реєстрі пат. України на кор. мод. 10.10.2017 р. 8 с.

67. Великодний С. С. Спосіб розмітки під отвори. Патент на кор. мод. №109897, зар. в Держ. реєстрі пат. України на кор. мод. 12.09.2016 р. 10 с.

68. Великодний С. С. Спосіб мультилінгвістичного перекодування компонентів систем автоматизованого проектування (Спосіб МЛП КСАП). Свід. про реєстр. авт. пр. на тв. № 48350; заявл. у Держ. службу інт. власн. 17.01.2013, заява № 48624; зареєстр. 18.03.2013. 16 с.

69. Великодний С. С. Імітаційне моделювання. Навчальний посібник. Одеська державна академія холоду. Одеса: Вид. центр ОДАХ, 2011. 188 с.

70. Великодний С. С. Дипломне проектування. Організація, вимоги щодо структури та оформлення пояснювальної записки, порядок захисту. Посібник для виконання дипломної роботи спеціаліста студентами денної та заочної форм навч. спец. 7.05010102 «Інформаційні технології проектування». Одеська державна академія холоду. Одеса: Вид. центр ОДАХ, 2011. 36 с.

71. Glushkov O. V., Velykodniy S. S., Buyadzhi V. V. Methodical instructions for laboratory work on discipline. “Mathematical Methods of Operations Research”. Odessa: Odessa State Environmental University, 2016. 24 p.

72. Глушков О. В., Великодний С. С., Буяджи В. В. Методичні вказівки для самостійної роботи студентів і виконання контрольної роботи з дисципліни «Математичні методи дослідження операцій». Одеса: ОДЕКУ, 2016. 21 с.

73. Великодний С. С., Онищенко С. М. Математичні методи дослідження операцій. Методичні вказівки по виконанню лабораторних робіт Одеський державний екологічний університет. Одеса, 2015. 96 с.

1 ОГЛЯД ЗАБЕЗПЕЧЕННЯ ТА УМОВ РОЗВИТКУ ПРОЄКТІВ З РЕІНЖИНІРИНГУ ПРОГРАМНИХ СИСТЕМ ТА ПРОДУКТІВ

ІТ-компанії, які створюють ПП, функціонують в середовищі, що постійно змінюється під впливом внутрішніх та зовнішніх факторів. Зміни стосуються не тільки технологій, які використовуються в процесі реалізації ІТ-проєктів, але й підходів до УП [74]. У проєктах створення та впровадження інформаційних технологій присутній досить високий відсоток невизначеностей, які можуть впливати на параметри проєкту та хід його виконання [75]. Для успішної реалізації ІТ-проєктів керівники чи відповідальні особи повинні своєчасно і адекватно реагувати на можливі несприятливі ситуації [76].

Проєктне управління розвивалося спочатку як методологія управління діяльністю, що спрямована на певний результат, але потреби практики поширили застосування підходів проєктного управління на весь ЖЦ продукту [77]. У сучасних ПС передбачається зберігання, обробка і передача інформації в комп'ютерних середовищах, оперативний доступ до даних.

Відповідно до стандарту ISO/DIS 21500 [78], [79] ПС слід розглядати як систему, що виконує низку процедур, певним чином тих, що логічно пов'язані між собою і виконавцями для прийняття проєктних рішень. ПС можуть бути використані в різних галузях науки, техніки, у тому числі й виробництва: для проєктування окремих деталей, предметів, вузлів, механізмів, машин, комплексів, агрегатів, виробничих ліній цілих виробничих підприємств і комплексів.

В розвиток та розробку сучасної теорії УП та програмами суттєвий внесок внесли вітчизняні вчені: С. Д. Бушуєв, Н. С. Бушуєва,

---

74. Бушуева Н. С. Модели и методы проактивного управления программами организационного развития: монография. Киев: Науковий світ, 2007. 200 с.

75. Онищенко І. І. Аналіз ризиків в процесі управління ІТ-проектами. *Вісник НТУ «ХПІ»*. 2014. № 2. С. 95–100.

76. Чечет А.М. Управління проектами на етапах життєвого циклу проекту. *Управління проектами, системний аналіз і логістика*. 2012. Вип. 10. С. 602–606.

77. Бушуев С. Д., Гогунський В. Д., Кошкін К. В. Напрями дисертаційних наукових досліджень зі спеціальності «Управління проектами та програмами». *Управління розвитком складних систем.* 2012. № 12. С. 5–7.

78. ГОСТ Р ИСО 21500–2014. Руководство по проектному менеджменту. Москва: Стандартинформ, 2015. 50 с.

79. ISO/DIS 21500. Guidance on project management. Geneva: International Organization for Standartization, 2011. 44 p.

І. В. Кононенко, В. Д. Гогунський, К. В. Кошкін, К. В. Колеснікова, А. В. Шахов, С. К. Чернов, В. В. Іванов, Є. А. Дружинін та ін., а також провідні закордонні вчені: У. Шухарт, А. Фейгенбаум, Ф. Кросбі, У. Демінг, Х. Танака, К. Ісікава, Р. Каплан, Д. Нортон, Ю. Адлер та ін.

1.1 Управління забезпеченням проєктів зі створення програмних систем та продуктів

Ідеалізований ІТ-проєкт – спеціалізована система з максимальним використанням уніфікованих модулів. Вимоги високої ефективності і універсальності, як правило, суперечливі. Стосовно до ПС це положення зберігає свою силу.

Модель, що запропонована Бушуєвим С. Д. та Бушуєвою Н. С. [80] конкретизує знайдену ідею при виконанні трьох стадій: бізнесової, технічної і організаційної. Реалізація цієї моделі можлива за допомогою концепції «Управління та аналізу проєктів», що запропонована у статті [81].

Технічне забезпечення ПС являє собою сукупність пов’язаних між собою технічних засобів (компьютерів, периферійних пристроїв, мережевого обладнання, ліній зв’язку, вимірювальних пристроїв тощо). Технічні пристрої, багато в чому, визначають ефективність систем, що створюються, тому, їм надається особливе значення.

У даний час і у найближчі роки – створення систем автоматичного проєктування не передбачається, і ніщо не загрожує монополії людини на прийняття вузлових проєктних рішень у процесі проєктування.

Стейкхолдери проєкту [82] із розвитку ПС [83] повинні вирішувати, по-перше: всі завдання, які не формалізовані, по-друге: завдання, вирішення яких людина здійснює на основі своїх евристичних здібностей більш ефективно, ніж сучасний комп’ютер на основі своїх

---

80. Бушуев С. Д., Бушуева Н. С. Модели и методы стратегического развития организаций от видения к реальности. *Управління проектами та розвиток виробництва.* 2005. № 4 (16). Луганськ: вид-во СНУ ім. В. Даля. С. 5–13.

81. Tanaka H., Bushuyev S. Innovative development and meta program management of a new generation of mega projects in the oil & gas and infrastructure sectors. *Управління розвитком складних систем.* 2013. № 16.

82. Draft international standard ISO / DIS 21500. Guidance on project management. Voting begins on 2011-04-04. International Organization for Standardization, 2011. 44 p.

83. ISO/IEC/IEEE 42010:2011. Systems and software engineering – Architecture description. Publication date: 2011-12. 37 p.

обчислювальних можливостей. Тісна взаємодія оператора та електронних засобів проєктування у процесі проєктування – один з принципів побудови та експлуатації ПС.

Високої ефективності ІТ-проєкту, яка виражається, перш за все, через мінімізацією часових, а значить і матеріальних витрат при вирішенні проєктних завдань, домагаються за рахунок удосконалення технічного забезпечення.

Вочевидь, що при цьому зростає число різноманітних технічних пристроїв. Щоб знизити витрати на розробку багатьох спеціалізованих технічних засобів, доцільно будувати їх на основі максимального використання уніфікованих складових частин, де необхідною умовою уніфікації – стає пошук спільних рис і здатностей у різнорідних технічних об'єктах, що планується використовувати під час проактивного розвитку ПС.

Процес проєктування та подальшого розвитку нової ПС триває, при встановленому порядку, кілька років [84]. За цей час, у більшості галузей з’являються нові наукові ідеї та рішення, які виводять виробництво на новий рівень і породжують нове покоління систем, приладів та установок. Реінжиніринг технічного забезпечення ПС дозволить, значною мірою, подолати протиріччя між темпами розвитку науки і техніки та процесів проєктування, підвищити ефективність проєкту, скоротити терміни його реалізації.

В результаті реалізації ІТ-проєкту – від технічного завдання, послідовно, проходячи низку проєктних стадій, замовник одержує робочий проєкт об'єкта проєктування (робочі креслення, технічний опис тощо).

Проте проєктування та звітні креслення – це тільки одна сторона проєкту. Другою, не менш важливою частиною проєктування – є моделювання. З розвитком ПС почали уходити від практики макетування та натурного тестування об’єктів у реальності. Нішу практичного випробування зайняло комп’ютерне моделювання – це стало великим проривом тому, що у комп’ютерній системі можливо спроєктувати деталь з будь-якого матеріалу та набагато легше зробити чисельний аналіз показників. Також було розв’язано проблему середовища, у якій проводяться випробування, тепер її можливо спроєктувати будь-якою.

---

84. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. Publication date: 2018-11. Ed. 2. 92 p.

Лінгвістичне забезпечення (ЛЗ) – сукупність мов, що використовуються у ПС, для представлення інформації про об'єкти проєктування, процеси та засоби проєктування. ЛЗ використовується для: здійснення діалогу проєктувальник – електронний інструментарій проєктування, обміну даними між технічними засобами ПС. ЛЗ включає терміни, визначення, правила формалізації природної мови, методи стиснення та розгортання.

В наш час є велика кількість мов проєктування, моделювання та програмування, які виконують велику кількість спеціалізованих завдань. Деякі з них зав’язані лише на якусь одну галузь промисловості, інші – застосовуються у великому розмаїтті, але тенденція йде шляхом спеціалізації мов в цілому.

Безумовно, для ПС, може бути сформульовано низку причин, що підкреслює багатобічність та велику складність проблеми реінжинірингу ПС.

Інформаційне забезпечення. У ПС забезпечується зручність її використання за рахунок застосування: засобів оперативного зв’язку проєктувальника з електронними засобами проєктування, спеціальних проблемно-орієнтованих мов і наявності інформаційно-довідкової бази.

Структурними складовими ПС є підсистеми, що володіють всіма властивостями систем та які створюються як самостійні структури. Це виділені, за деякими ознаками, частини ПС, що забезпечують виконання деяких закінчених проєктних завдань, з отриманням відповідних проєктних рішень і проєктних документів.

При побудові ПС – головна увага приділяється сукупності інформаційно-узгоджених підсистем. Цей дуже важливий принцип повинен ставитися не тільки стосовно до зв'язків між великими підсистемами, але і до зв'язків між більш дрібними частинами підсистем.

Інформаційна узгодженість означає, що всі або більшість можливих послідовностей завдань проєктування обслуговуються інформаційно-узгодженими базами.

Дві бази є інформаційно-узгодженими, якщо всі ті дані, які являють собою об'єкт переробки в обох базах, входять в числові масиви, які не потребують змін при переході від одного програмного модуля до іншого (перекодування форматів). Так інформаційні зв'язки можуть проявлятися у тому, що результати розв'язання однієї задачі будуть вихідними даними для іншої задачі.

Якщо для узгодження баз потрібна істотна переробка загального масиву за участю оператора, який додає відсутні параметри, вручну перекомпонує масиви або змінює числові значення окремих параметрів, то такі бази є інформаційно не узгодженими.

Ручне перекомпонування масиву веде до істотних часових затримок, зростання кількості помилок і тому зменшує попит на послуги ПП. Інформаційна неузгодженість перетворює ПС у сукупність автономних програм, при цьому, через неурахування в підсистемах багатьох факторів, що оцінюються у інших підсистемах – знижується якість проєктних рішень, що є основною проблемою застосування ПС.

Іншими словами, у цьому випадку для виконання принципу інформаційної узгодженості необхідна переробка конверторів проміжних форматів, що є ознакою реінжинірингу інформаційного забезпечення ПС.

Основною задачею, що ставиться перед реінжинірингом інформаційного забезпечення – є створення банку моделей реінжинірингу, який включає у себе функціонально-повний ЖЦ ПП.

Програмне забезпечення. Основною проблемою із якою стикаються розробники ПС є проблема асимптотичної недосяжності початкових вимог замовника. На кожному новому етапі розробки до ПС додається новий модуль чи навіть підсистема, проте питанням сумісності із вже існуючими та оптимізації компонентів (як фізичних модулів коду) на рівні зв’язків, класів, залежностей тощо – приділяється увага досить поверхнево або не приділяється зовсім.

На жаль, у такому разі кожна наступна версія майже готового продукту за своїми системними показниками стає гіршою за попередні. В кінцевому результаті, приходить момент, коли майже готова «зведена» ПС – не виконує більшості функцій, що повинна була виконувати, хоча на етапах розробки кожного окремого модуля – ці задачі виконувались.

На практиці, часто проблема вибору між реінжинірингом та повторною розробкою вирішується у бік останньої – і це прикро. Причини такого рішення – криються у людському факторі: у ситуації з оновленням «проблемної» ПС, як правило, повторна розробка доручається новій команді конструкторів та програмістів, яким, на їх думку, легше виконати розробку «з нуля», чим аналізувати та виправляти існуючи помилки. Наслідком цього, як правило, стають повторні схожі помилки та все ті ж структурні недоробки.

Таким чином, у якості передумов реінжинірингу, як форми проактивного розвитку проєкту з удосконалення ПС можна відзначити:

а) оновлення платформи;

б) нестача персоналу із знанням застарілої МП;

в) зміна структури ПС.

До операцій реінжинірингу ПС належать:

а) ідентифікація компонентів;

б) розширення функцій;

в) переклад мови програмування;

г) реструктуризація;

д) модифікація опису.

Головною проблемою перед реінжинірингом ПС – стає управління адаптацією різноманітних ПС під розповсюджені операційні системи, зокрема, під так звані «відкриті» платформи.

В результаті аналізу, що було проведено автором монографії [85] заради перевірки гіпотези про помилковість деяких уявлень з організації проєктування – можна сформувати головну передумову реінжинірингу ПС. Ця передумова криється у «вимогах замовників, які з часом неможливо задовольнити, оскільки користувачі ПП «зростають» разом з проєктом, що вже не має «кінцевої точки» у своєму розвитку» [85].

1.2 Проблеми управління обліковою інформацією у проєктах програмних продуктів

Згідно зі стандартами ISO 9000, ISO 10006, ISO/DIS 21500, проєкт повинен мати розвиток. У роботах Іванова В. В. [86] розглянуто фази проєктування продукту, які складаються із стадій: технічна пропозиція, ескізний проєкт, технічний проєкт, а також виготовлення. У роботах Шахова А. В. [87] розглядаються життєві цикли технічних систем. Розглянемо досвід вивчення проблем управління інформацією у проєктах

---

85. Тимченко А.А. Основи системного проектування та системного аналізу складних об'єктів: Основи САПР та системного проектування складних об'єктів. 2-ге вид. Київ: Либідь, 2003. 272 с.

86. Іванов В. В. Моделі проекту зворотного інжинірингу. *Вісник НТУ «ХПІ».* 2017. № 2 (1224) С. 52–57. DOI: 10.20998/2413-3000.2017.1224.9

87. Шахов А. В. Проектирование жизненного цикла ремонтопригодных технических систем. Одесса: Феникс, 2005. 164 с.

на прикладі програмних продуктів, призначених для виконання обліку промислового споживання електричної енергії.

У сучасних умовах існування промисловості України особливого значення набувають питання достовірного обліку електричної енергії на усіх ділянках та рівнях її виробництва, передачі та споживання. Незважаючи на суттєвий вплив кризових явищ в економіці, теперішні атестаційні вимоги щодо автоматизованих систем комерційного обліку електроенергії (АСКОЕ), примушують виконувати ці роботи на найсучаснішому технологічному рівні, враховуючі тенденції, що особливо швидко змінюються стосовно до ПС.

До 2000-х років в України були відсутні підприємства з виробництва необхідної номенклатури засобів вимірювальної техніки, збору, передачі та обробки інформації. Недостатня також нормативна база і концепція створення зазначених засобів. З 2000 року підприємства України та зарубіжних фірм пропонують інформаційно-вимірювальні системи для всіх рівнів та засоби вимірювальної техніки різних типів, тому після прийняття рішення про розробку галузевої програми і концепції розвитку АСКОЕ в умовах енергоринку [88], постало питання щодо їх державної метрологічної атестації (ДМА), зокрема обчислювальної інформаційної компоненти.

Таким чином, необхідно було визначити проблеми, що виникають при управлінні обчислювальною компонентою у проєктах АСКОЕ та внести конкретні пропозиції щодо їх усунення.

Національний науковий центр (ННЦ) «Інститут метрології» проводить роботи з обов’язкової ДМА АСКОЕ, до складу яких входять вимірювальні канали, в тому числі телемеханічні, що складаються із: вимірювальних трансформаторів струму й напруги, багатофункціональних електронних лічильників, маршрутизаторів, засобів систем комунікацій та зв'язку, серверів збирання баз даних.

Також ННЦ «Інститут метрології» виконує роботи з ДМА автоматизованих робочих місць (АРМ) з обліку електроенергії, адміністрування та контролю стану системи. АРМ призначено для виконання збору та обробки даних комерційного обліку електричної

88. Концепція побудови автоматизованих систем комерційного обліку електроенергії в умовах енергоринку: затв. Мінпаливенерго України від 17.04.2000 р., нак. № 32/28/28/276/75/54. К.: Держспоживстандарт, 2008. 25 с.

енергії, а також для забезпечення проведення невід'ємних проєктних процедур при роботі АСКОЕ:

– автоматизації функцій обліку електроенергії на промислових об'єктах;

– автоматизації функцій складання балансів споживання електроенергії;

– побудови фактичних графіків навантаження на добовому, місячному та річному інтервалах часу при різних системах тарифів;

– підготовки звітних документів з обліку електроенергії.

Згідно з основними принципами організації збору і обробки інформації [88], основною вимогою – є загально-інформаційний простір для всіх суб'єктів енергетичного ринку. На практиці це положення реалізується у вигляді єдиної інтегрованої мережі управління (збір, накопичення та обробка) інформації про вироблення та споживання енергії.

Досвід роботи закордонних енергетичних систем, особливо тих, що працюють в умовах ринку, доводить необхідність введення процедур перевірки точності і достовірності інформації на всіх рівнях і в усіх точках системи обліку, де здійснюється облік і обробка даних. Це важливо не тільки з технічної точки зору, але і з точки зору економічних та правових взаємовідносин виробника, постачальника і споживача.

Як апаратний базис інтеграції пристроїв обробки даних на рівнях регіонального устаткування збору даних рекомендується використовувати високонадійні вимірювальні засоби, які відповідають сучасним промисловим стандартам. Це дозволяє поєднувати їх високі експлуатаційні характеристики з доступністю ПС для базового операційного середовища. До таких можна віднести продукцію виробників, із якими співробітничає (через заявників та замовників) ННЦ «Інститут метрології» при атестації АРМ АСКОЕ, а саме: компанія «ЕМН Україна», «Енергоцентр», концерном «Енергоміра», корпорацією «Облік» та ін.

Компанія «ЕМН Україна», що була створена на базі відомого німецького концерну EMH Elektrizitätszähler GmbH & Co KG, розробила ПП EMH-Mobile, призначене для використання у PDA (кишеньковий комп'ютер), за допомогою якого (разом із Blutooth-адаптером) можливий бездротовий зв'язок із лічильниками у багатоквартирному будинку. Крім зчитування та передачі даних наведений ПП дозволяє перевіряти правильність підключення лічильника.

ПП EMH-Combi Master 2000 призначений для параметризації лічильників, зчитування їх показань та додаткової інформації. Це ПЗ дозволяє реалізувати можливість конфігурації лічильників ED 2500, ITZ, LZQJ-XC; встановлювати ідентифікатори та коефіцієнти трансформації, тарифний розклад й автозчитування швидкості передачі між інтерфейсами, зчитування профілю навантаження та параметрів мережі з послідовним конвертуванням у текстовий файл, перевірку вірності підключення лічильника та багато іншого.

ПП, що розроблені корпорацією «Облік», крім наведених функцій попередніх ПП, дозволяє також забезпечити прогнозування енерговитрат із досить якісним ймовірним довірчим інтервалом.

Основною компонентою АРМ є ПС, що призначені для збирання, зберігання та обробки показань лічильників електричної енергії об'єктів контролю (підстанцій, електричних станцій, споживачів), які складають нижній рівень з наступною передачею на верхній (сервер).

Атестація обчислювальної компоненти ПС виконується експериментально-розрахунковим методом, згідно затвердженої методики атестації за якою знаходяться відповідні метрологічні характеристики.

Інтерфейс оператора ПС АСКОЕ, повинен забезпечувати індикацію усіх параметрів, що обробляються, серед яких дані від лічильника із міткою часу та відповідним, вибраним користувачем, інтервалом виміру (як правило 30-ти хвилинним). Первинні дані в необробленому вигляді підлягають архівації і збереженню без будь-якого корегування.

За кожною точкою обліку обчислюються та зберігаються наступні параметри активної та реактивної енергії:

– усереднена потужність, відповідно до заданого періоду інтеграції із реєстрацією витрат електроенергії, диференційованої за часом доби, сезоном, вихідними та святковими днями (рис. 1.1);

– графік активних та реактивних навантажень за агрегатами й енергосистемою в цілому за останні десять діб;

– енергія та потужність за поточні й минулі облікові періоди;

– відображення в реальному масштабі часу миттєвих навантажень та показників якості електроенергії, що відповідають показам лічильників (рис. 1.2).

На основі цих параметрів будується інформаційна база даних (БД), логічна структура якої містить наступні розділи:

– масив необроблених даних (МНД);

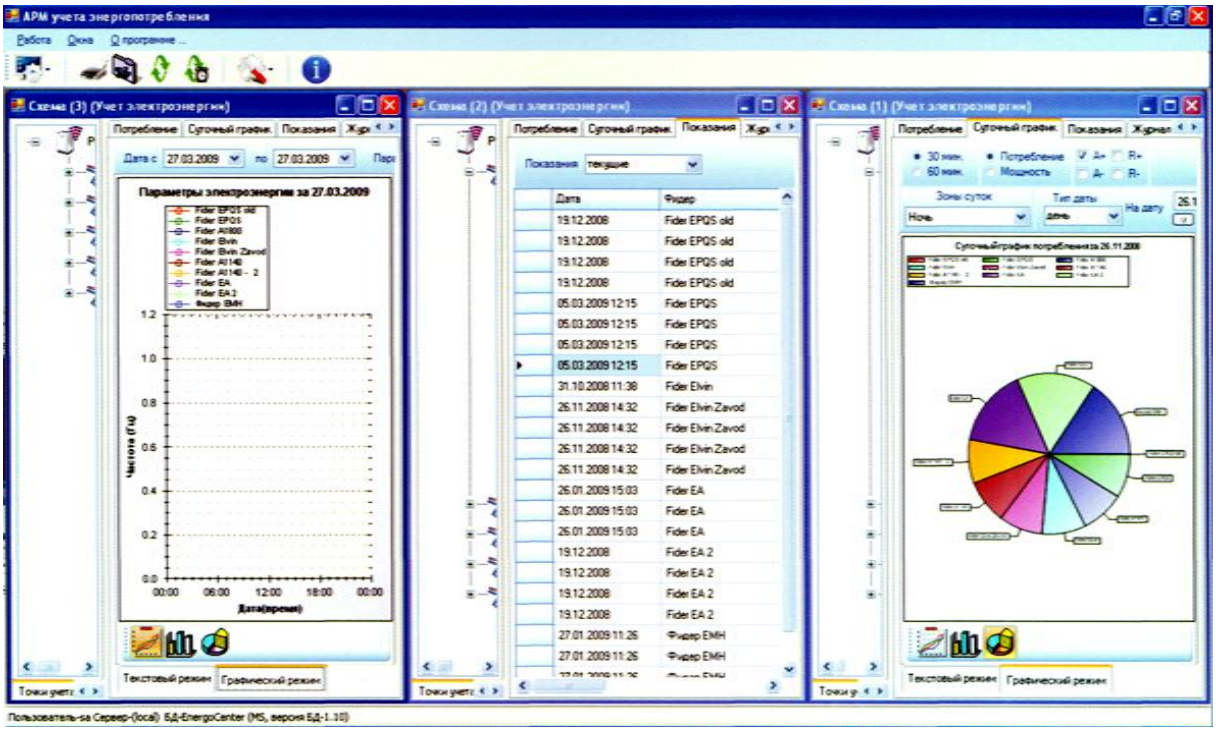


Рисунок 1.1 – Загальний інтерфейс управління обліковою інформацією проєкту ПС АСКОЕ

– масив даних ручного введення (МДРВ) та розрахункових величин;
– масив звітних даних (МЗД);
– масив нормативно-довідкової інформації.

У МНД зберігається початкова інформація, яка збирається з об’єктів обліку програмою автоматичного та ручного збору даних, причому процедура збереження даних та їх подальше відновлення, повинна повністю виключати можливість зміни у порівнянні з оригіналом. МДРВ служить для зберігання інформації, що вводиться оператором системи обліку вручну або розраховується на основі МНД. МЗД служить для складання необхідних вихідних документів.

Виходячи з розглянутих вимог, можна сформулювати наступні загальні вимоги до проєктування ПС обліку споживання електроенергії:

– інтерфейс оператора повинен забезпечувати індикацію всіх параметрів, що оброблюються, уведення паролів і даних, адаптацію алгоритмів обробки, конфігурування та настроювання каналів, протоколів зв’язку й тестування;

– передбачати можливість застосування стандартних МП і графічних інтерфейсів користувачів;

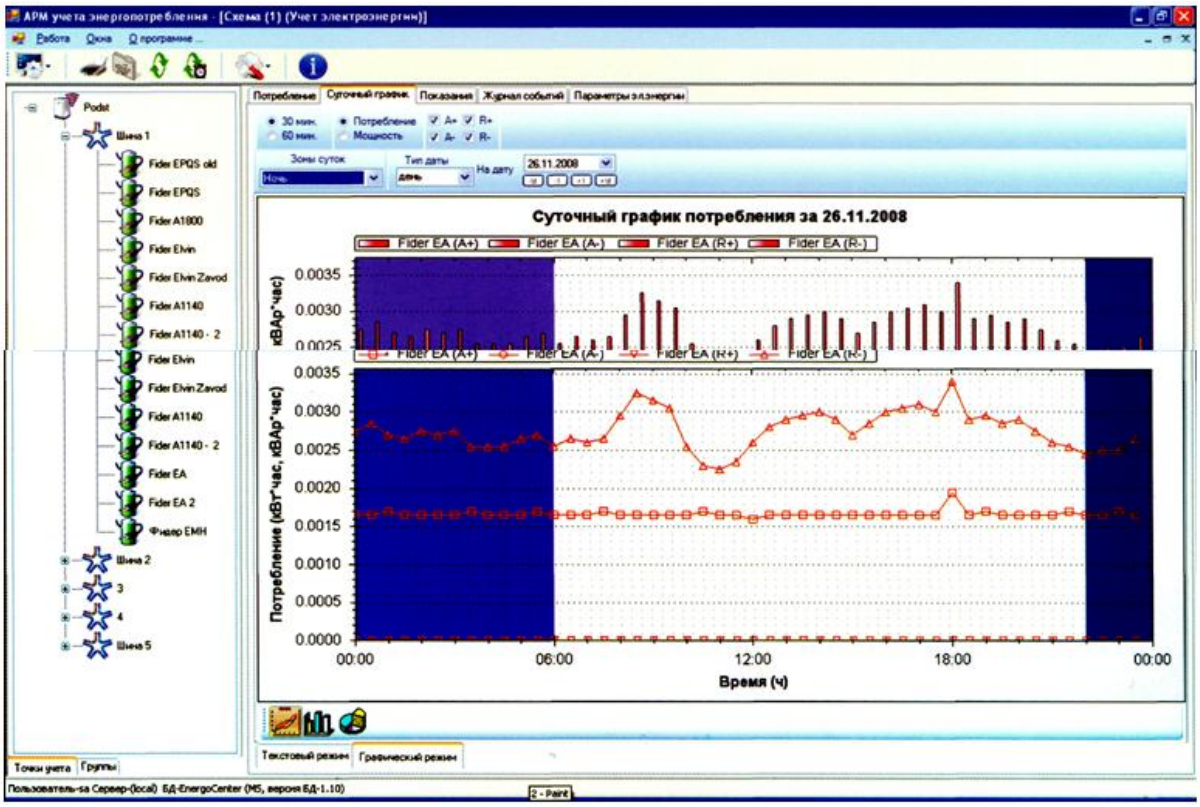


Рисунок 1.2 – Управління показниками миттєвих навантажень за допомогою проєкту ПС АСКОЕ

– облік електроенергії від будь-яких типів лічильників;

– облік енергоресурсів (суматори імпульсів, СИНЭТ-1);

– підтримка систем управління БД ORACLE, MS SQL, PostgreSQL;

– автоматичне приймання та відправка макетів;

– зберігання первинної БД;

– генератор звітів на базі MS Excel;

– підтримку ручного уводу;

– підтримку комунікаційних протоколів передачі, що відповідають вимогам архітектури відкритих систем;

– доступність оновлень на сайті розробника.

Також додатково бажано реалізувати підтримку наступних функцій:

– настроювання «наскрізного» каналу опитування приладів сервісними програмами;

– відправлення файлів-макетів за GPRS-каналом на завдані e-mail адреси;

– опитування пристроїв із сервера користувача;

– конфігурування та перегляд накопиченої інформації через Web-браузер;

– керування та сигналізація за допомогою SMS- повідомлень;

– синхронізація часу за NTP, GPS.

Загалом нарікань на роботу ПП АСКОЕ з боку користувачів не помічено, а ось що стосується атестації обчислювальної компоненти проєктів, то тут існують деякі проблеми.

По-перше, згідно з [88], розробники інтерфейсу програмування прикладного рівня при атестації ПП для АСКОЕ повинні надавати декларативні та алгоритмічні описи, проте деякі фірми-розробники так переймаються питаннями захисту інтелектуальної власності, що співробітники випробувальної лабораторії вимушені витрачати час на відкриття вихідного програмного коду.

По-друге, при атестації ПС великими труднощами є моделювання (імітація) аварійних режимів роботи АРМ, ліній зв’язку або сервера, при якому уся інформація повинна зберігатися у енергонезалежній буферній пам’яті лічильників (з розбивкою за обліковими зонами: ніч, пік, напівпік та ін.), а після відновлення каналів зв’язку – база даних повинна актуалізуватися. При відсутності зовнішнього живлення – буферна пам'ять має забезпечити фіксацію часу зникнення живлення, зберігання даних протягом не менш 30 діб (що впливає на тривалість випробування), хід часу, календарну дату та фіксацію часу відновлення живлення.

Ще одна вагома проблема при атестації обчислювальної компоненти ПС АСКОЕ, полягає у розрахунках абсолютної похибки обліку витрат активної та реактивної енергії за МЗД, зведених до інформаційної БД.

При формуванні звітних даних обліку, інтерактивне вікно оператора виводить наступні поля запиту:

– звітна дата (у форматі: ДД-ММ-РРРР);

– початкова точка формування МЗД (у форматі: ГГ-ХХ);

– інтервал обліку (півгодини / година);

– обліковий період (доба / місяць).

Суть проблеми полягає в тому, що при потребі сформувати МЗД за визначений період, наприклад за добу: з 09 год. 00 хв. 25.01.2010 р. до 09 год. 00 хв. 26.01.2010 р. із півгодинним інтервалом виміру; замість 48 контрольних точок – формується 49, якраз 49-ю є точка: 09 год. 00 хв. 26.01.2010 р. (рис. 1.3).

Виключити при компіляції вихідного коду формування кінцевої точки шляхом закритого інтервалу – неможливо, тому, що саме у цій точці потрібно знати остаточне (для цього облікового періоду) показання лічильника, за яким буде підраховане значення розбіжності між фактичними витратами та показаннями.

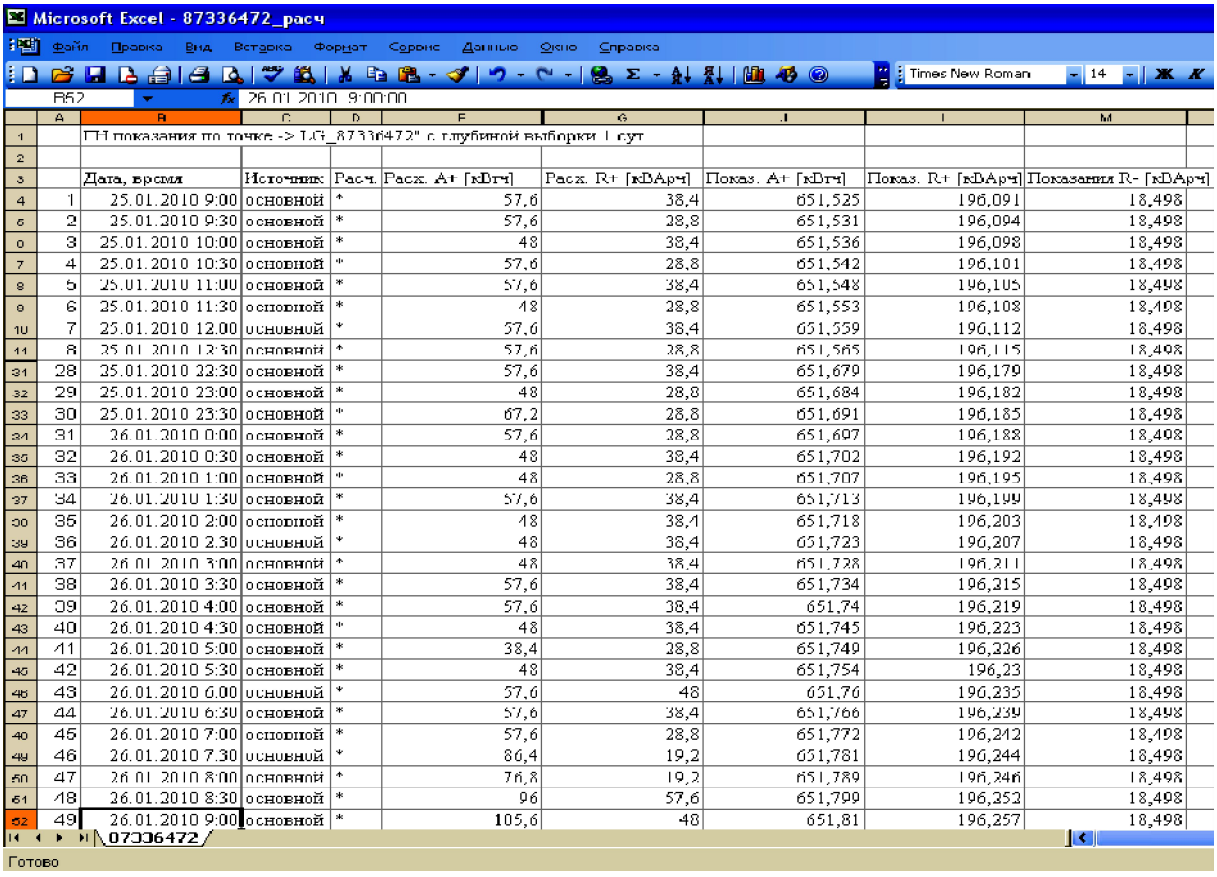


Рисунок 1.3 – Управління інформацією з масива звітних даних на базі MS Excel

Виключити першу точку – також неможливо тому, що саме вона несе інформацію щодо показання лічильника, яке є початковим для періоду обліку, що цікавить.

Особливо слід зауважити, що параметр «ВИТРАТИ» (кВт•г) формується з МНД, які збираються з об’єктів обліку, виходячи виключно із значення миттєвої потужності, тобто значення витрат на фактично нульовій обліковій точці (09 год. 00 хв. 25.01.2010 р.) дорівнює витратам за попередні півгодини (з 08 год. 30 хв. 25.01.2010 р. до 09 год.00 хв. 25.01.2010 р.).

В процесі розрахунків подальших облікових сум за параметрами «ВИТРАТИ» та «ПОКАЗАННЯ», виходить розбіжність показань якраз у розмірі значення витрат у нульовій точці облікового періоду, яке тягне за собою виникнення абсолютної похибки обчислювальної компоненти.

Таким чином, із усього наведеного можна сформувати конкретні пропозиції, адресовані розробникам ПС АСКОЕ, щодо усунення штучних причин виникнення небажаного погіршення метрологічних характеристик проєкту, а саме: розробити програмну процедуру встановлення плаваючого нуля в початкову (згідно з періодом обліку) адресну комірку параметру «ВИТРАТИ».

Крім того, досвід реалізації подібних проєктів дозволяє рекомендувати розробникам ПС необхідність дотримуватись орієнтації на підтримку відкритих уніфікованих протоколів зв’язку з робочими станціями та серверами, завдяки цьому буде можлива інтеграція з різними операційними платформами і пристроями, що використовуються на верхніх рівнях систем обліку. Для передачі даних можливе сумісне використання каналів зв’язку ПС АСКОЕ та інших систем з метою резервування та зменшення витрат на устаткування. З метою скорочення часу на технічне супроводження та навчання стейкхолдерів по роботі із ПП, необхідно виконувати розробку ПЗ модулів адаптації приладів обліку до вимог замовника та виявлених проблем до впровадження проєктів ПС.

1.3 Використання CAD/CAM/CAE-систем в задачах управління проєктами

Однією з головних умов, що ставиться перед УП з розвитку CAD/CAM/CAE-систем або систем автоматизованого проєктування (САП) – це вирішення проблеми якості векторизації у проєкті. Інакше кажучи, відчувається потреба у використанні інтерактивних методів управління одержанням векторної моделі, при застосуванні яких користувач задає на екрані монітора набір точок зображення і ряд інших параметрів, що дозволяють керувати процесом розпізнавання і, як наслідок цього, впливати на точність проєктного результату.

При розгляді САП з цієї позиції, очевидний висновок, що найбільш складні проблеми при проактивному УП з розвитку САП пов’язані саме з тими питаннями, які було сформульовано як завдання дослідження.

Було проведено аналіз предметної галузі дослідження, яка включає огляд існуючих проєктів розв'язання траєкторних задач та розгляд різноманітних САП для створення складних геометричних поверхонь. Насамперед, були піддані аналізу ПС та засоби автоматизованого проєктування формоутворення, що використовуються на профільних підприємствах. Це ПП від провідних пострадянських і західних розробників: CATIA, Unigraphics, Pro/Engineer, Duct, PowerMill, ProCAM, CADDS, Euclid, Anvill, КРЕДО, ГЕМА-3D, САП-УФА, АРТ, БАПТ, MODAPT, а також пакети Anvil, AutoCAD, «КОМПАС» і багато інших проєктів.

Корпорації, що управляють проєктами з розвитку САП – проєктують багато спеціалізованих ПП, наприклад – AutoDesk. В них є цілі портфелі програм для роботи з машинобудівним профілем (Inventor), архітектурою (ArchiCad), дизайном (3dMAx) та проєктуванням у широкому сенсі (AutoCad).

Із проведеного аналізу ПС можна виділити, що сучасні САП дозволяють реалізувати будь-які закони управляння, однак, властивий цифровим обчислювальним пристроям послідовний характер реалізації алгоритмів, ставить вимоги розробки проблемно-орієнтованого ПЗ, заснованого на ефективних обчислювальних методах, при застосуванні яких забезпечуються висока швидкодія у проєкті та найвища точність опису траєкторій.

Згідно із загальноприйнятою методологією – САП класифіковано за трьома рівнями проєктів:

– прості – це більша частина ПП виробника AutoDesk, ADEM, КОМПАС та інші;

– середні – ПП рівня SolidWorks;

– високі – спеціалізовані ПП високої потужності (CATIA).

Широкий спектр проєктних робіт, складність конструкції, стислість термінів виконання замовлень, посилення конкурентної боротьби на ринку змушують сьогодні проводити роботи на найсучаснішому технічному рівні, тому конструювання, аналіз, технологічна підготовка виробництва здійснюється з використанням так званих «інтегрованих» САП, вибір і організація роботи яких, проходять в рамках єдиної концепції розвитку

засобів проєктування [89] і технологічної підготовки виробництва, здатних вирішувати поставлені задачі нелінійної візуалізації.

При написанні даного підрозділу, перш за все, були піддані аналізу методи та засоби автоматизованого проєктування для систем числового програмного управління (СЧПУ), що використовуються на профільних підприємствах. Це пакети і системи від провідних наукових інститутів країни та західних розробників.

Першою з ретельно проаналізованих систем, став проєкт EDS Unigraphics (UG). На цей вибір вплинуло відповідність UG наступним, з моєї точки зору, важливим проєктним критеріям: після освоєння ПС, проведення авторами серії абсолютно необхідних організаційних заходів, написання ряду сполучних програм було створено програмно-апаратне середовище, що відповідає цілям проєктування [90].

У цьому середовищі UG є тією віссю, на яку спираються інші засоби проєктування. Багато з цих ПП за своєю вартістю в кілька разів дорожче UG, але їх робота без використання функціональності CAD/CAM – неефективна. [91].

Практичне застосування ПС ведеться в напрямках технологічної підготовки виробництва, а саме: проєктування найбільш складних елементів ливарного оснащення для турбінних лопаток; генерація необхідних постпроцесорів для СЧПУ (із застосуванням модуля GPM); створення на мовах C++ і GRIP необхідних сервісних додатків; розробка програм управляння для СЧПУ [92].

Однак, незважаючи на всі переваги UG, залишаються невирішеними ряд складних і не менш важливих технічних завдань формоутворення – проєктування та виготовлення різних тонкостінних конструкцій, що складаються з великого набору криволінійних поверхонь, які повинні дуже гладко сполучатися один з одним і мати плавну зміну похідної.

Наступним розглянутим проєктом була САП Anvil. У даному середовищі, для адекватного скорочення часу УП, був застосований

---

89. Писаренко Д. М. Инструментальные средства проектирования многофункциональных самоорганизующихся мобильных роботов. *Искусственный интеллект.* 2005. №1. С. 86–92.

90. Unigraphics Direct Interface: Reference Manual. Southampton: ICEM Ltd., 2004. 39 p.

91. Бормалев С., Червонных С. Практическое применение EDS Unigraphics в авиастроении. *Открытые системы*. 1997. №2. С. 43–46.

92. Краснов М., Чигишев Ю. Unigraphics для профессионалов. Москва: ЛОРИ, 2004. 320 с.

принцип типізації конструкції [93] і створені типові креслення сполучених поверхонь. При цьому було використано метод побудови цих поверхонь з типових елементів. Такий підхід дозволив: з одного боку, заздалегідь створити необхідний набір скетчів та програм для параметризації елементів деталі і її оснащення, розробити асоціативно пов'язані креслення, а з іншого, – прискорити формування маршрутних карт на базі типової технології, прискорити розрахунок міжопераційних розмірів і проєктування оснащення другого порядку.

Система Anvil має непогані можливості для проєктування і виготовлення деталей підвищеного рівня складності. Інтегрована оболонка позбавляє технолога від необхідності вивчати апаратне і ПЗ, тим самим прискорюючи проєкт введення СЧПУ в експлуатацію і практично позбавляючи технологічні служби підприємств від обслуговування системними програмістами [94].

Інтегрована оболонка має розвинуті засоби для налаштувань численних параметрів САПР-ЧПУ індивідуально для кожного користувача, відновлюючи значення при повторних сеансах [95]. Оболонка концентрує управління викликами всіх модулів САПР-ЧПУ від розрахунку контуру деталі та каркасного моделювання до формування управляючої програми.

Модуль «Графічний Процесор «Фенікс»» включений до складу базового проєкту ПС САПР-ЧПУ. Модуль «Фенікс», використовується для графічної верифікації управляючих програм, тобто для графічного представлення контуру деталі, елементів роз'єднаної геометрії, а також імітації переміщень інструменту в процесі обробки деталі [96]. Графічний процесор включає не тільки розвинені засоби візуалізації на екрані, а й потужні засоби для зняття копії проєкту на будь-який Windows-сумісний

---

93. Горитов А. Н., Кориков А. М. Оптимальность в задачах проектирования и управления роботами. *Автоматика и телемеханика.* 2001. №7. С. 82–90.

94. Фаголь И. Об одном подходе к проектированию паспортов в САПР-ЧПУ/2000. Пермь: ООО «Евразия Лимитед», 2002. 52 с.

95. Филиппович К. В. Некоторые аспекты настройки пользовательских предпочтений в САПР-ЧПУ/2005. Пермь: ООО «Евразия Лимитед», 2005. 48 с.

96. Филиппович К., Попович И. ToolStore – среда для ведения библиотеки инструментов в верификаторе CNC-Verify системы САПР-ЧПУ/2005. Пермь: ООО «Евразия Лимитед», 2005. 36 с.

принтер для подальшої роботи або зробити простий проєкт для передачі в цех та подальшого управляння ним [97].

Роз'єднана геометрія, що використана технологом для побудови контуру деталі, відображається як сукупність точок, прямих і кривих разом зі своїми ідентифікаторами. Це спрощує візуальний аналіз проєкту, а ортогональні і аксонометричні проекції і сприяють більш швидкому знаходженню можливих помилок [98].

Верифікація траєкторії руху інструменту здійснюється у декількох режимах: прискореному, уповільненому, покадровому та відладному [99]. В останньому режимі на екран видається гамма геометричних та технологічних параметрів поточного переміщення різального інструменту в кожному кадрі управляючої програми. Є засоби візуалізації тільки певної частини траєкторії руху інструменту. Це досягається вибором певної зміни інструмента, або використанням потужного інструментарію розстановки контрольних точок проєкту [100].

Однак, що стосується недоліків, то у даному проєкті не передбачено засобів для проведення вимірювань (кути нахилу, визначення приналежності, знаходження точок дотику або перетину елементів), автовизначення канонічних параметрів роз'єднаної геометрії, а також немає реалізації графічного моделювання формотворної інструменту для токарної і фрезерної обробки і відсутні бібліотеки графічних образів інструментів.

У зв'язку з цим, у якості графічного верифікатору, було розглянуто модуль CutViewer (від Tudor I / E Ltd), повністю інтегрований з NCVerify [101].

У проєкт інтегрованого ПП САПР-ЧПУ вбудований «Коректор управляючих програм». Його призначення полягає у введенні з носія раніше підготовленої програми, її перекодування з інформаційного коду

---

97. Wakeford L. How Your Design Can Affect The Cost, Quality And Time Required To Manufacture Parts. *MCADVision Magazine*. 2001. July, Part 1. P. 68–71.

98. Diehl B. CAD/CAM a la Carte: A modular approach to choosing machining software. *CNC Machining Magazine*. 2001. Vol.5, #16. P. 54–57.

99. Lynch M. The Key Concepts Of Computer Numerical Control. New Mexico: CNC Concepts Inc., 2004. 60 p.

100. Калачёв О. Н. Моделирование в CAD/CAM Cimatron механообработки на станке с ЧПУ. Ярославль: ЯГТУ, 2003. 28 с.

101. Калачёв О. Н., Рехтер А. Д. Моделирование размеров механообработки в среде AutoCAD 200x на основе использования приложения GRAKON7. *САПР и графика.* 2002. №2. С. 100–104.

верстата (ISO, EIA) у текстовий вигляд, подальше коректування за допомогою екранного редактору і, нарешті, зворотна операція виведення на носій в необхідному коді СЧПУ [102], [103].

APTIPP (універсальний постпроцесорний проєкт) – дозволяє використовувати єдину бібліотеку з 300 постпроцесорів для будь-якої із систем: ADEM, CATIA, UG, Pro/Engineer, Duct, PowerMill, ProCAM, CADDS, Euclid, Anvill, КРЕДО, ГЕММА–3D, САП-УФА, АРТ, БАПТ, MODAPT [104]. APTIPP генерує управляючу програму на основі CL-файлів, сформованих CAD/CAM-системами [105]. За допомогою APTIPP технолог може розробляти постпроцесори для будь-якого проєкту «верстат-СЧПУ» [106].

«Інваріантний постпроцесор» (ІП) – унікальний проєкт, розроблений у 1975 р. та який пройшов довгий шлях еволюції, до цих пір не має повних аналогів. У складі системи САП СМ4 він впроваджений на 500 підприємствах колишнього СРСР, а у складі САПР-ЧПУ – на 220 підприємствах Росії та СНД [107]. Модуль ІП призначений для перетворення файлу траєкторії руху інструменту та техкоманд у файл управляючої програми, що адаптована до конкретного проєкту «верстат-система ЧПУ».

ІП є доповненням до будь-якої імпортної і вітчизняної CAD/CAM-системі, так як вже має готові паспорти практично на будь-яке ЧПУ-обладнання. Однак, якщо ж обладнання унікальне, то використання ІП не дає значного виграшу у вартості, термінах створення та зручності

---

102. Калачёв О.Н. Документация по программным продуктам: «KON7 Расчет технологических размерных цепей»; «GRAKON7 Автоматизированное построение в среде AutoCAD 2000 размерной схемы технологического процесса механообработки». Ярославль: ЯГТУ, 2000. 94 с.

103. Грунина Е. В., Калачёв О. Н. Методика проектирования в CAD/CAM Cimatron УП для гравирования профиля на юбилейной медали. Ярославль: ЯГТУ, 2008. 15 с.

104. Митрофанов В. Г., Калачев О. Н., Схиртладзе А. Г. САПР в технологии машиностроения: учеб. пособие. Ярославль: ЯГТУ, 2001. 298 с.

105. Зильбербург Л. И., Марьяновский С. М., Молочник В. И., Яблочников Е. И. Компьютерное проектирование и производство: моногр.; под общ. ред. С. М. Марьяновского. Санкт-Петербург: КПЦ «МиР», 1998. 166 с.

106. Филиппович К.В. Идеология постпроцессирования в современных CAD/CAM-системах. Пермь: ООО «Евразия Лимитед», 2000. 60 с.

107. Zelinski P. A Better Process From Better Posts. *Serving the Metalworking Industries*. 2001. №2. P. 28–31.

модифікації паспортів у порівнянні із технологіями індивідуального постпроцесування або генерування постпроцесорів.

Проєкт GrafCAM – це надбудова верхнього рівня над системою САПР-ЧПУ. GrafCAM повністю орієнтований на 2.5D-візуальні графічні засоби параметричного проєктування G-кодів. Завдяки засобам імпорту DXF-файлів, GrafCAM легко інтегрується з будь-якою конструкторською CAD-системою, забезпечуючи наскрізний цикл конструювання-проєктування [108]. Візуальні засоби редагування геометрії контурів або технологічних переходів, повна параметризація і можливість нарощування функціональності – роблять GrafCAM потужним інструментом для управляння технологічним проєктом. Починаючи з 1992 р., GrafCAM впроваджено на більш ніж 70 підприємствах.

Модуль GrafCAM призначений для проєктування управляючих програм до СЧПУ, використовуючи принцип графічної побудови і редагування контуру деталі, траєкторії руху інструменту і технологічних команд [109]. GrafCAM орієнтований на управління технологічними проєктами і може бути налаштований на будь-який рівень їх кваліфікації [110]. Модуль дозволяє будувати елементи роз'єднаної геометрії і об'єднувати їх в контури, на цій основі технолог проєктує траєкторію руху інструменту. Для проєктування роз'єднаної геометрії, використовуються 60 способів завдання точок, прямих та кіл.

GrafCAM ідеальний проєкт для графічного формування траєкторії руху інструменту: користувач вказує елементи роз'єднаної геометрії, пов'язуючи їх у ланцюг. У будь-який момент часу можна ввести технологічну команду управління верстатом. Модуль GrafCAM підтримує бібліотеки параметризованих елементів (макропроцедури) і допускає поповнення бібліотек та меню макропроцедури самим користувачем: доповнення контурів деталі або траєкторій руху інструменту [111]. Також користувачеві доступні різноманітні видові операції.

---

108. Филиппович К. В. Импорт DXF-файлов в GrafCAM v.6.20. Новые возможности для технолога. Пермь: ООО «Евразия Лимитед», 2005. 52 с.

109. Филиппович К. В. Редактирование геометрии и технологических команд в GrafCAM v.7. Пермь: ООО «Евразия Лимитед», 2005. 46 с.

110. Воскобойников Ю. С., Филиппович К. В. Внешний редактор в Grafcam v.7.08 – решение, проверенное временем. Пермь: ООО «Евразия Лимитед», 2006. 50 с.

111. Трухнин Н. М., Филиппович К. В. Макрорасширения в GrafCAM – путь к наращиванию функциональности.Пермь: ООО «Евразия Лимитед», 2006. 72 с.

Відмінним принципом GrafCAM є автоматичне формування тексту програм на вхідній мові своєї базової системи, тобто САПР-ЧПУ. Крім того, GrafCAM є 100% графічним інтерпретатором та відладчиком текстів програм управляння, створених за допомогою систем САП-32, САП-ЄС, САП-СМ4, САП-ПК, ACAD-SAP, що дозволяє забезпечувати режим наскрізного проєктування за допомогою простої макромови.

Після виконаного аналізу CAD/CAM/CAE-систем слід зазначити, що найбільшими труднощами при реалізації проєктів інтерактивних ПС є:

– використання методів аналітичної геометрії для опису поверхонь;

– застосування сучасних комп'ютерних бібліотек, що дозволяють маніпулювати із зображенням у проєкті;

– розробка методів проактивного розвитку архітектури програм і графічного діалогу;

– реалізація спеціальних проєктних вимог, висунутих до ПП.

На підставі даних аналізу, можна сформулювати головний недолік в УП САП, який є найвагомішою вимогою до їх проактивного розвитку – це необхідність удосконалення проєктів з оновлення платформ, під які вони були розроблені та переведення МП до сьогоденних вимог щодо ПС, тобто виконання УП реінжинірингу CAD/CAM/CAE-систем як ПП. Цей процес лежить у векторі мети роботи, досягненню якої присвячено подані дослідження.

1.4 Аналіз методологій управління проєктами зі створення відкритих, вільних та комерційних програмних продуктів

Розглядаючи питання проактивного УП з розвитку ПС та ПП, необхідно розглянути методології створення ПП с точку зору доступу та використання вихідного коду ПС.

Відкрите ПЗ (англ. Open-Source Software) – ПЗ з відкритим вихідним кодом. Вихідний код у таких ПС доступний для перегляду, вивчення та зміни [112]. На основі такого ПЗ можна створювати модифікації, виправляти помилки, створювати нове ПЗ.

Дуже часто використовується ще один термін для надання характеристики ПС – вільне ПЗ. Визначення відкритого і вільного ПЗ не цілком збігаються один з одним, але близькі.

---

112. Goodbye, «free software»; hello, «open source». URL: http://www.webcitation.org/617oVjlKk (дата звернення: 10.01.2019).

Відмінності між відкритим і вільним ПЗ полягають, в основному, у точці зору їх розробників. Ті, хто підтримує методологію «Open-Source» дивляться на ПС з точки зору ефективності їх відкритих початкових кодів як методу розробки, модернізації та супроводу. Прихильники методології «Free-Software», як правило, вважають, що найголовнішим у такому ПП є його безоплатне розповсюдження [113]. Як ми бачимо, відмінності не дуже істотні, але у своїх рамках їх утримують певні ліцензійні угоди, які й визначають статус продукту.

Комерційне ПЗ з відкритим вихідним кодом (синонім англ. Open-Core) являє собою ПП, який містить деякі елементи вільного і відкритого ПЗ для того, щоб законно претендувати на статус «Open-Source». Іноді у відкритій, безкоштовній версії виключаються деякі можливості, присутні у комерційній версії цього ж продукту, яка поширюються за пропрієтарною ліцензією. Відкриття частини вихідного коду, створеного раніше під пропрієтарною ліцензією, залишає потенційну можливість прив'язки такого рішення до одного єдиного постачальника.

На просторах глобальної мережі можна знайти ПП, які є платними, але при цьому мають відкритий програмний код. Прикладом може бути архіватор UnRAR. Такі ПП – цілий окремий клас. Вони використовують термін «Open-Source» щодо платного ПЗ. Найчастіше зустрічаються 2 варіанти таких програм:

а) умовно-безкоштовна версія з відкритим кодом та обмеженими можливостями (3 – 4 основних або дуже корисних модулі є платними);

б) платне ПП із відкритим кодом: таке ПЗ розвиває власним темпом компанія, код відкривається для загального розвитку або ж для пошуку нових програмістів до компанії.

Випуск ПП під подвійною ліцензією – це зовсім інший спосіб створення методології «Open-Source» рішення на основі пропрієтарного коду.

Вихідні коди відкритих програм випускаються або як суспільне надбання, або на умовах «вільних» ліцензій як, наприклад, методологія General Public License (GNU) чи BSD License. Вільна ліцензія дозволяє використовувати вихідний код програми для своїх потреб із мінімальними обмеженнями, що не суперечать визначенню «Open-Source» [113]. Таким обмеженням може бути вимога посилатися на попередніх творців або

113. Categories of free and nonfree software. URL: http://www.gnu.org/philosophy/categories.en.html (дата звернення: 10.01.2019).

вимога зберігати властивість відкритості при подальшому поширенні тієї ж самої або модифікованої відкритої програми – методологія копілефт [114]. У деяких випадках (наприклад, Apache або FreeBSD) ці обмеження дуже малі, в інших (наприклад, GNU) досить поширювати ПЗ разом із вихідним кодом і текстом ліцензії, не змінюючи її.

САП з відкритим вихідним кодом так само мають значне поширення. Наприклад, якщо взяти Linux, то майже всі CAD-системи, у яких можна комфортно працювати, є безкоштовними і вільно поширюваними, тобто модифікуються тільки силами добровольців-ентузіастів. Звичайно, у порівнянні з тими ж AutoCad або SolidWorks їх можливості поки виглядають не так стабільно, але чи багато хто використовує ці САП на повну потужність? Знову ж таки: зручністю відкритих систем є можливість роботи з усіма простими, але потрібними функціями, а чого не вистачає – завжди можна дописати чи знайти того, хто вже дописав, тому будь-які модифікації з часом з'являються в Інтернеті. Існує досить багато перспективних проєктів з відкритим кодом, які, з часом, переростуть у щось більше [115].

Отже, суть методології «Open-Source» (систем з відкритим кодом) є можливість змінювати та створювати ПП під свої потреби [116]. У розвитку проєктів зі створення таких ПС існує низка переваг та недоліків, що зведено до табл. 1.1.

На цей момент більше половини компаній включили ПЗ з відкритим кодом у свої IT-стратегії, як заявляють в Gartner. При цьому майже третина респондентів назвала переваги такого ПЗ:

а) гнучкість;

б) швидкість впровадження;

в) скорочення часу розробки.

---

114. What is free software? URL: http://www.gnu.org/philosophy/free-sw.en.html (дата звернення: 10.01.2019).

115. High Priority Free Software Projects by Free Software Foundation. URL: https://www.fsf.org/ campaigns/priority-projects/ (дата звернення: 10.01.2019).

116. Open Source Paradigm Shift by Tim O'Reilly. URL: http://archive.oreilly.com/ pub/a/oreilly/tim/ articles/paradigmshift_0504.html (дата звернення: 10.01.2019).

Таблиця 1.1 – Переваги та недоліки розвитку проєктів зі створення ПП з відкритим кодом

Відкрите ПЗ (методологія «Open-Source»)	
Переваги:	Недоліки:
повна або часткова безкоштовність для користувачів, а звідси й поширеність	безкоштовне ПЗ не має такого функціоналу, як у платного ПП
систему піддано декомпозиції на модулі, кожен з яких відповідає за власну задачу, у наслідок чого розробка, удосконалення та налагодження виконуються набагато швидше і легше	вихідний код доступний будь-якому користувачеві, тобто будь-хто може знайти уразливості й помилки ПП і використовувати їх зі своєю метою (але цей варіант досить рідкісний)
виробник ПЗ не може шпигувати, обманювати і створювати спеціальні незручності – backdoors (поняттям «backdoors» фахівці позначають спеціально залишені або розроблені вразливості у захисті ПС, які можна використовувати для крадіжки та зміни даних користувача)	у безкоштовного ПЗ відсутні підготовчі курси, системи допомоги у програмах, сертифіковані навчальні посібники та служби технічної підтримки, тобто будь-які питання з помилками, нестикуваннями тощо доводиться вирішувати самотужки
можливість поліпшення ПП власноруч	метод розробки – дуже часто подальша розробка ПЗ виконується недосвідченими фахівцями, результатом чого стають недопрацьовані модулі.
чим більше поширений ПП, тим легше знайти тестерів і вільних програмістів, готових працювати над поліпшенням продукту	
проста інтеграція компонентів від різних розробників	розвиток одного ПП в принципово різних напрямках – випадки, коли розробки настільки різні, що поєднувати модулі неможливо (найвідоміший прикладом є дистрибутиви Linux, де існують розробки компаній та вільних користувачів і, як результат, не повна сумісність)
при виникненні помилки виправляти треба тільки модуль, у якому вона виникла	

ПЗ з відкритим вихідним кодом використовується в багатьох галузях, включаючи управління бізнес-процесами, проєктування, IT-безпека, управління ризиками тощо. Як відзначають аналітики [115], конкурентні переваги у галузі IT відіграють все більшу роль і використання рішень «Open-Source» може дати їх компаніям значні переваги перед конкурентами. Наприклад, якщо компанія змінює код додатку, зробивши його унікальним, вона отримує певну перевагу. Найчастіше відкриті рішення використовуються компаніями у комплексі з власними розробками.

Розвиток проєктів зі створення платних ПП так само має низку переваг та недоліків, що зведено у табл. 1.2.

Таблиця 1.2 – Переваги і недоліки розвитку проєктів зі створення комерційного ПП

<table>
<tr><td colspan="2">Платне (ліцензійне) ПЗ</td></tr>
<tr><td>Переваги:</td><td>Недоліки:</td></tr>
<tr><td>легке впровадження в організації: у будь-яких фірм-розробників, що займаються ПП є план з швидкого і якісного впровадження</td><td>комерційне ПЗ закрито, його розробники зберігають таємниці своїх рішень та не розкривають: внутрішньої архітектури, форматів представлення даних, інтерфейсів</td></tr>
<tr><td>краща продуктивність і оптимізація</td><td rowspan="2">будь-які комерційні рішення випускаються у формі великих модулів, дрібні модифікації не проводяться, що призводить до того, що людина отримує разом з однією потрібною функцією декілька функцій, які не використовуються</td></tr>
<tr><td>система підтримки: питаннями та виправленням помилок займається спеціальна команда</td></tr>
<tr><td>захищеність ПП: кожен комерційний ПП має свою систему захисту (що постійно розвивається) від шпигунства</td><td>у 60% випадків не передбачена заміна компонентів, тобто не розвинена система підключення модулів;</td></tr>
<tr><td>автоматичне оновлення – ліцензійний пакет містить не тільки покупку продукту та виправлення помилок, але й можливість поліпшення ПС з часом</td><td>не існує єдиного стандарту – можливість інтеграції з продуктами третіх фірм практично не передбачена</td></tr>
</table>

Кінець таблиці 1.2

Платне (ліцензійне) ПЗ	
Переваги:	Недоліки:
організована система продажу – сервіси надаються постачальниками комерційного ПЗ	якщо ПП не влаштовує клієнта у можливостях, то доводиться купувати нову версію
	якщо побудоване рішення (у новій версії) не влаштовує клієнта або виникають проблеми, то залишається чекати (наступної версії), а при урахуванні того, що випускаються нові рішення тільки у формі великих модулів – це займає великий проміжок часу

За результатами аналізу попиту розробок 228 компаній було виділено переваги проєктів зі створення відкритих ПП над комерційними, які знайшли відображення у відповідних процентних відношеннях гістограми, наведеної на рис. 1.4. Само опитування тривало 2,5 місяця, відбувалося у шести містах України (Київ, Харків, Одеса, Львів, Запоріжжя, Миколаїв) та чотирьох містах Польщі (Варшава, Торунь, Люблін, Вроцлав), причому подекуди опитування проводилося через спеціалізовані форуми.

Анкета опитування була складена за допомогою Google-форм (додаток А). Вона містить 242 можливих варіанти відповідей. Результати опитування, статистика, гістограми та сама анкета зберігається на Google-диску автора монографії.

Серед респондентів були присутні різні класи користувачів ПЗ, які погодилися надати відповіді на запитання будь-якому одному з чотирьох авторів статті [10]. Опитування являє собою модель управління джерелом знань в умовах динамічного оточення. Модель охоплює науковців, студентів, аспірантів, співробітників бюджетних організацій, приватних підприємців, держслужбовців. Серед усіх названих ПП, була підрахована кількість розробників – 228.

Першою і, безумовно, найголовнішою перевагою відкритого ПЗ виявилася ціна. Звичайно відкрите ПЗ – це набагато більше ніж просто безкоштовна програма на просторах Інтернету, але при цьому замовники і

власники фірм на перше місце ставлять саме цей фактор. Досить часто трапляються випадки, коли корпоративний замовник звертає увагу тільки на ціну, але при цьому не враховує функціональність, що призводить до сумних наслідків.

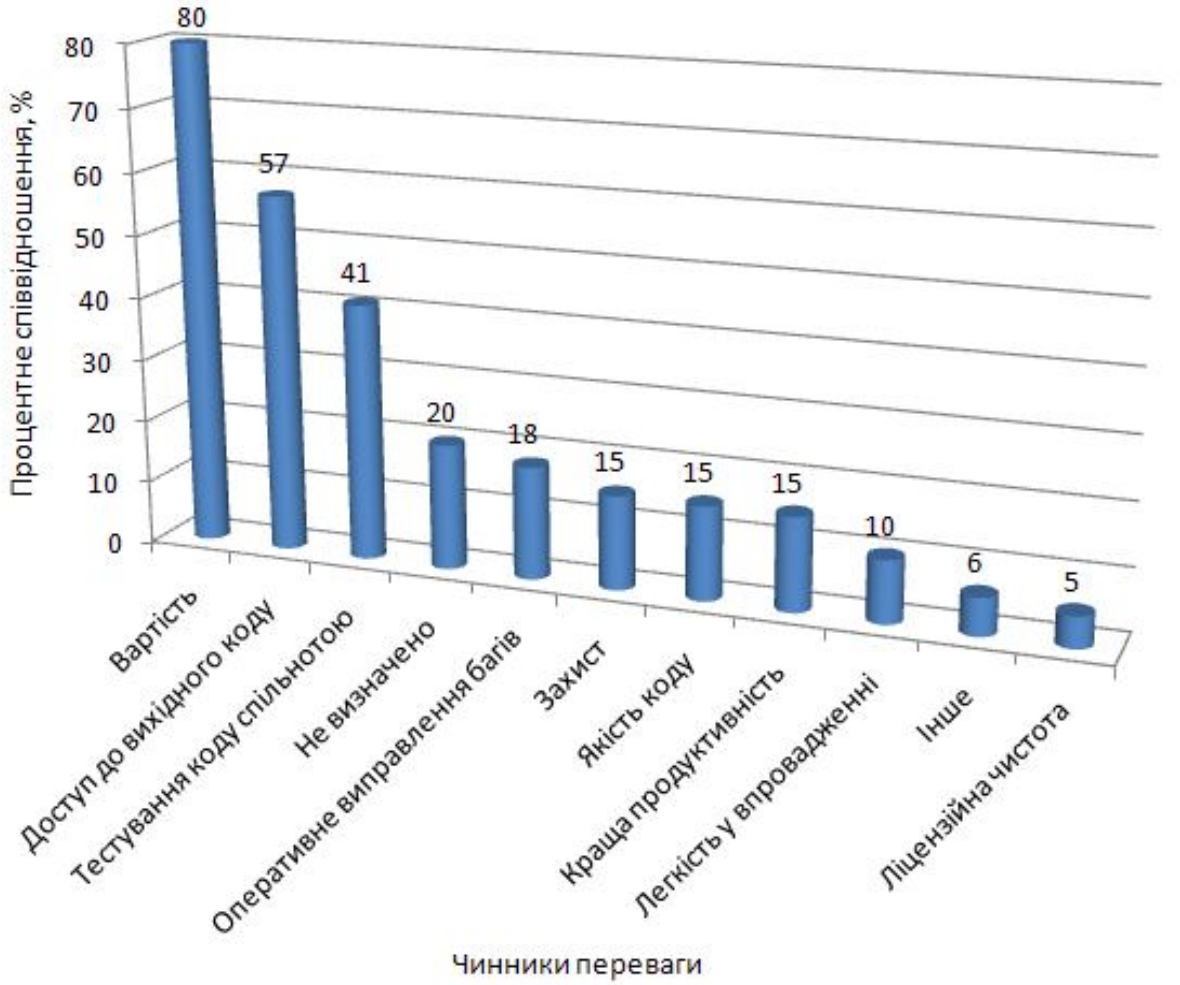


Рисунок 1.4 – Переваги проєктів зі створення відкритих ПП над комерційними

Другим пунктом є доступ до вихідного коду. 57% респондентів сказали, що це є одним з найбільш важливих факторів. По суті – це так і є, адже якщо є професійна команда програмістів, то можливо створити будь-який потрібний додаток або знайти його. Все одно: витрати будуть менші, а результати швидші ніж у комерційній сфері, але далеко не кожному споживачеві, як виявилося, потрібна можливість працювати з вихідним кодом – досить часто вистачає й стандартних функцій.

Оперативне виправлення багів та захист стоять на наступному місці, тому і те й інше у ПП з відкритим кодом часто страждає або взагалі відсутнє. Як показує статистика – потреби у складному захисті немає, адже відкрите ПЗ найчастіше використовують малі та середні компанії, у яких по-справжньому серйозні атаки відбуваються дуже рідко.

Якість коду і краща продуктивність за статистикою займають одне з останніх місць. Це не дивно тому, що модифікації ПП та модулів часто проводяться недосвідченими програмістами (найчастіше є першими проєктами початківців), в наслідок чого «чистота» коду й продуктивність залишають бажати кращого.

Легкість у впровадженні в організацію та ліцензійну чистоту відзначили лише 10% та 5% респондентів. З причини безкоштовності продукту впровадженням, зазвичай, доводиться займатися самому споживачеві, що у випадку невеликих фірм не спричиняє великих проблем. Середні фірми, зазвичай, мають складнощі з впровадженням, які доводиться вирішувати своїми силами, що займає чимало часу ще й з причини поганої оптимізації продукту під потреби великої кількості користувачів.

Переваги проєктів зі створення комерційних ПП над відкритими знайшли відображення у відповідних процентних відношеннях гістограми, поданої на рис. 1.5.

У комерційного ПП є одна дуже велика перевага, відзначили 65% респондентів – це організована система продажу. В Інтернеті, в магазинах можна знайти повний список властивостей, книг-помічників, консультантів тощо. Великий відсоток людей, які купують ПЗ, звертають на це увагу, адже набагато простіше використовувати інформацію, яка вже сформульована.

Друга велика перевага комерційного ПЗ – легкість у впровадженні. Політика впровадження комерційного ПЗ має на увазі подальшу співпрацю підприємства з виробником протягом кількох років, після чого проводиться аналіз ефективності та приймається рішення про подальшу співпрацю із компанією або перехід на інший ПП.

Важливу роль відіграють автоматичне оновлення та система підтримки. Багато фірм вважають за краще залишати вирішення питань і виправлення помилок спеціалістам, які створили ПП. Хоча, з іншого боку, вони не мають вибору – адже код ПП закритий. Що ж до автоматичного оновлення – багато залежить від договору із клієнтом. Великі фірми,

зазвичай, домовляються про створення нової версії під потреби та фіксовану мету; малі компанії часто задовольняються й стандартними оновленнями, що зафіксовано у ліцензійній угоді.

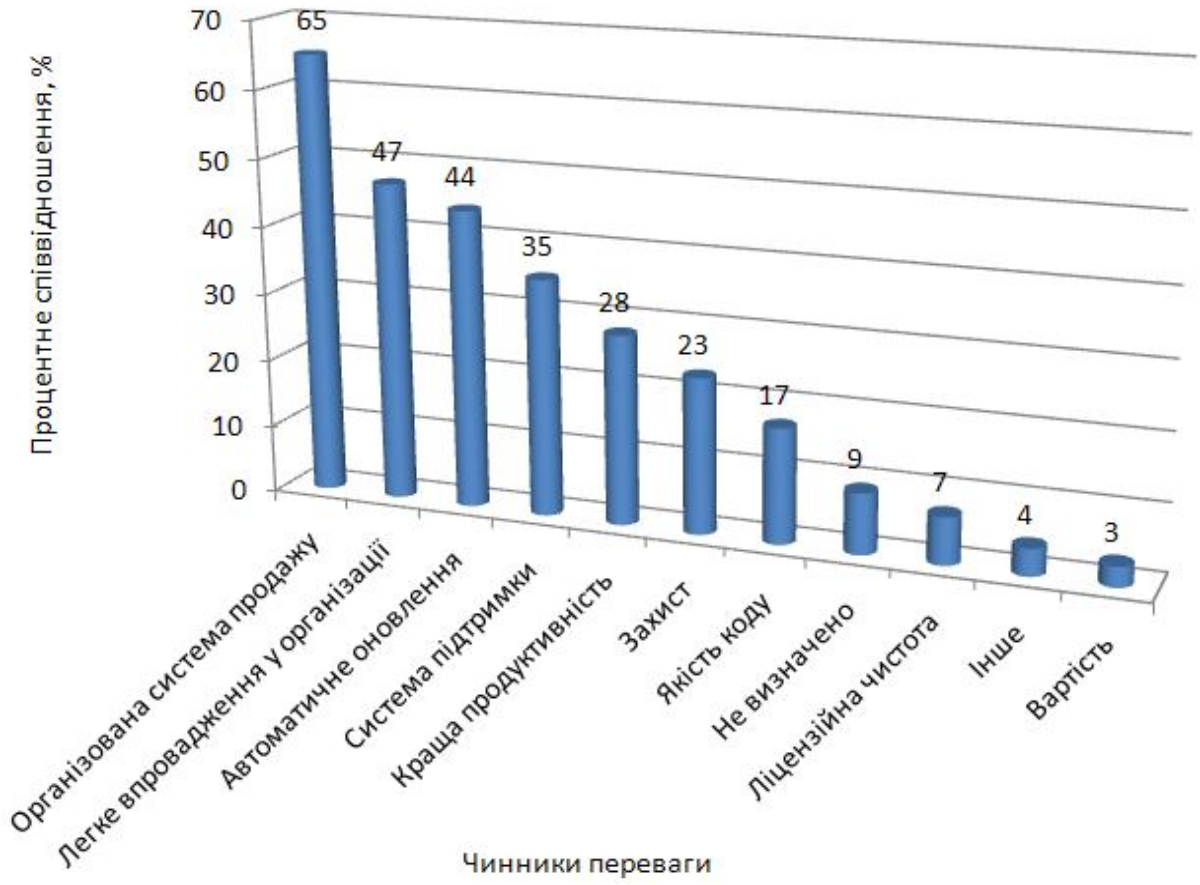


Рисунок 1.5 – Переваги проєктів зі створення комерційних ПП над відкритими

За переваги захисту проголосувало всього лише 23% опитаних. Проаналізованих причин для такого результату дві:

а) захист не потрібен тому, що не проводиться масивних атак та витоків інформації (ситуація звичайна для дрібних фірм);

б) захист, який існує – занадто слабкий для відбиття серйозних атак (ситуація звичайна для великих фірм).

Якість коду зацікавила тільки 17%. Звичайно це впливає на продуктивність, але при цьому код є закритою частиною проєкту в комерційному ПП, тому не особливо зрозуміло: чим зумовлено таке зацікавлення компаній.

Ліцензійна чистота і вартість відіграють не значну роль у перевагах комерційних ПП тому, що сплачувати за продукт не є багато бажаючих, крім того, модифікації виконує тільки одна компанія, а це є великим мінусом.

Переваги та недоліки УП з різною методологією розповсюдження ПП інтегровано у табл. 1.3.

Досить цікавий факт, що користувачів не особливо хвилює ліцензійна чистота продуктів: відкриті вони чи загальнодоступні, чи є комерційними проєктами. Таке саме становище із захистом та якістю коду.

Таблиця 1.3 – Наявність проєктних показників у комерційному та вільному ПП

Властивості:	Комерційний ПП	Відкритий ПП
вартість	+	–
служба підтримки	+	–
автоматичне оновлення	+	–
захист	+	–
організована система продажу	+	–
відкритий вихідний код	–	+
робота з великими модулями	+	–
робота з дрібними модулями	–	+
єдиний стандарт проєктів	–	–
легкість впровадження	+	–
підготовчі курси, навчальні посібники	+	–
виробниче шпигунство розробниками	+	–
розробка і поліпшення модулів власними силами	–	+
зручність виправлення помилок	–	+
декомпозиція на модулі	–	+

Отже, на цей час багато західних аналітиків схиляються до використання відкритого ПЗ, тому вже з'явилися корпорації та фірми, які забезпечують розвиток й підтримку для безкоштовних ПП, при цьому є можливість управляти проєктом створення коду. Проте, ні допомогу спільноти, ні низькі ціни на ПП комерційний розробник забезпечити не в змозі.

Що ж стосується ситуації із Україною – наші розробники не активні в УП з розвитку відкритого ПЗ. Однією з причин є те, що такі проєкти не так давно з'явилося на наших ринках та стейкхолдери поки «не влилися» у світовій потік модифікацій ПП. Так само: багато власників компаній та менеджерів ще не досконалі у знаннях про можливості проєктів із відкритим ПЗ й обирають недорогі аналоги за критерієм мінімізації вартості створення проєкту ПС. Слід додати, що в останні роки зростає тенденція відмови від УП за допомогою російських галузевих ПП (проєктування, облік та аудит, фінанси, системи управління підприємствами та проєктами тощо).

1.5 Аналіз розвитку систем управління базами проєктних даних, методи подання та класифікація графічних баз даних

У літературі пропонується множина визначень поняття «база даних», що відображають скоріше суб'єктивну думку тих чи інших авторів, однак єдине загальновизнане формулювання відсутнє. Слід зазначити, що багато фахівців вказують на поширену помилку, що складається в некоректному використанні терміна «база даних» замість терміна «система управління базами даних» (СУБД), та вказують на необхідність розрізнення цих понять [117].

Сам термін «database» (база даних) з'явився на початку 1960-х років і був введений у вживання на симпозіумах, організованих фірмою SDC (System Development Corporation) у 1964 – 1965 роках, хоча розумівся спочатку в досить вузькому сенсі, у контексті систем штучного інтелекту. У широке вживання, у сучасному розумінні, термін увійшов лише в 1970-ті роки [118].

Наступний важливий етап пов'язаний з появою на початку 1970-х реляційної моделі даних, завдяки роботам Едгара Ф. Кодда. Роботи Кодда відкрили шлях до тісного зв'язку прикладної технології БД з математикою

117. Дейт К. Дж. Введение в системы баз данных. Изд. 8-е. Москва: Вильямс, 2005. 1328 с.

118. Haigh T. How Data Got its Base: Information Storage Software in the 1950s and 1960s. IEEE Annals of the History of Computing, 2010. Vol. 31. Iss. 4. P. 6–25. DOI: 10.1109/MAHC.2009.123.

і логікою. За свій внесок в теорію і практику Едгар Ф. Коддом також отримав премію Тюрінга [119].

Одночасно, вживаються визначення БД з міжнародних стандартів, що подано нижче. БД – сукупність даних, що зберігаються у відповідності зі схемою даних, маніпулювання якими виконують відповідно до правил засобів моделювання даних [120]. БД – сукупність даних, організованих відповідно до концептуальної структури, яка описує характеристики цих даних і взаємовідносини між ними, причому таке зібрання даних, яке підтримує одну або більше областей застосування [121]. Приклад такої концептуальної структури із визначеними зв'язками між даними, що спроєктовано автором поданої монографії стосовно до графічних БД (ГБД), наведено на структурній моделі (рис. 1.6).

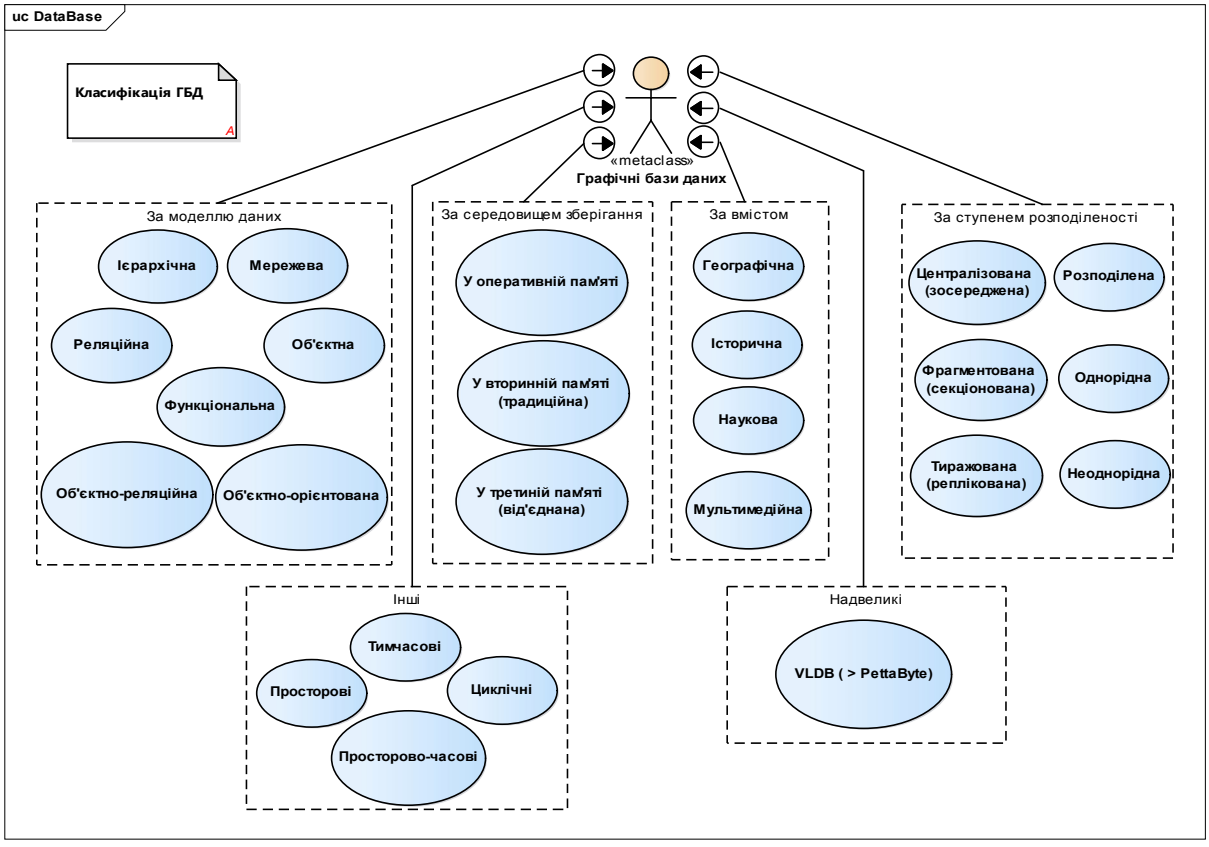


Рисунок 1.6 – Структурна модель класифікації ГБД

119. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. Москва: Вильямс, 2003. 1088 с.

120. C. J. Date. Date on Database: Writings 2000–2006. New York City: Apress, 2006. 566 p.

121. Когаловский М. Р. Перспективные технологии информационных систем. Москва: ДМК Пресс; Компания АйТи, 2003. 288 с.

Для повноцінної роботи із ГБД, приведемо нижче визначення БД з декількох авторитетних монографій. БД – організована відповідно до певних правил і підтримувана в пам'яті комп'ютера сукупність даних, що характеризує актуальний стан деякої предметної області і використовувана для задоволення інформаційних потреб користувачів [122]. БД – деякий набір перманентних (постійно збережених) даних, що використовуються прикладними програмними системами управління підприємства. БД – спільно використовуваний набір логічно зв'язаних даних (та опис цих даних), призначений для задоволення інформаційних потреб організації [123].

При аналізі усіх наведених визначень БД (додавши ще й [124]), слід виділити найбільш часто (явно або неявно) присутні наступні ознаки:

а) БД зберігається та обробляється у обчислювальній системі (таким чином, будь-які позакомп’ютерні сховища інформації (архіви, бібліотеки, картотеки тощо) БД не є);

б) дані у БД логічно структуровані (систематизовані) з метою забезпечення можливості їх ефективного пошуку та обробки в обчислювальній системі (структурованість передбачає явне виділення складових частин (елементів), зв'язків між ними, а також типізацію елементів і зв'язків, за якої з типом елемента (зв'язком) співвідноситься певна семантика й припустимі операції);

в) БД включає схему або метадані, що описують логічну структуру БД у формальному вигляді (відповідно до деякої метамоделі, наприклад, відповідно до [122]: «постійні дані в середовищі БД включають в себе схему і БД. Схема включає в себе описи змісту, структури і обмежень цілісності, використовувані для створення і підтримки БД. БД включає в себе набір постійних даних, визначених за допомогою схеми. Система управління даними використовує визначення даних у схемі для забезпечення доступу і управління доступом до даних у БД»).

З перерахованих ознак лише перша є суворою, а інші допускають різні трактування й різні ступені оцінки. Можна лише встановити деяку ступінь відповідності вимогам до ГБД.

---

122. Когаловский М. Р. Энциклопедия технологий баз данных. Москва: Финансы и статистика, 2012. 460 с.

123. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е изд. М.: Вильямс, 2003. 1436 с.

124. Мирошниченко Е. А. К формальному определению понятия «база даных». Проблемы информатики. 2011. № 2. С. 83–87.

У такій ситуації не останню роль відіграє загальноприйнята практика. Відповідно до неї, наприклад, не називають БД файлові архіви, Інтернет-портали або електронні таблиці, незважаючи на те, що вони, в деякій мірі, володіють ознаками БД. Прийнято вважати, що ця ступінь, в більшості випадків, недостатня (хоча можуть бути винятки).

При виконанні аналізу досліджень і публікацій, особливу увагу, у науковому сенсі, було приділено методам, що застосовуються при роботі із складними графічними об’єктами та базами проєктних даних.

Нижче розглянуто основні наукові геометричні методи, що використовуються при графічному моделюванні об'єктів, процесів, явищ в техніці та тенденції їхнього розвитку. За напрямок наукового дослідження обирається твердотільне моделювання об'єктів, що утворюються (та змінюються в часі) під впливом різних зовнішніх чинників.

Комп'ютерне моделювання є одним з ефективних методів вивчення будь-яких складних систем, що піддаються візуалізації. Комп'ютерні моделі простіше і зручніше досліджувати у силу їх можливості проводити так звані обчислювальні експерименти, у тих випадках, коли реальні експерименти ускладнені через фінансові або фізичні перешкоди або можуть дати непередбачуваний результат [125].

Тривимірний опис об’єкта (англ.: 3D) – це представлення об’єкта у трьох просторових вимірах. Як правило, ці виміри представлені у вигляді координат *X*, *Y*, та *Z*. Можливо мати дані з ідентичними координатами *X* та *Y* при відмінній координаті *Z*. Наприклад, для цифрового представлення океанічних потоків, використовують 3D [126].

Твердотільне моделювання – це найдосконаліший і достовірний метод створення копії реального об'єкта, природний спосіб вираження сутності виробу [127].

Рендеринг (англ.: rendering – відтворення, відрисовування) в комп'ютерній графіці – це процес отримання зображення за моделлю за допомогою комп'ютерної програми [128]. Тут модель – це опис

---

125. Норенков И. П. Автоматизированное проектирование. Москва: МГТУ им. Н. Э. Баумана, 2000. 188 с.

126. Ли Дж., Уэр Б. Трёхмерная графика и анимация. Изд. 2-е. Москва: Вильямс, 2002. 640 с.

127. Концевич В. Г. Твердотельное моделирование машиностроительных изделий в Autodesk Inventor. Киев, Москва: ДиаСофтЮП, ДМК Пресс, 2007. 672 с.

128. Херн Д., Бейкер М. П. Компьютерная графика и стандарт OpenGL. Изд. 3-е. Москва: Вильямс, 2005. 1168 с.

тривимірних об'єктів (3D) суворо визначеною мовою або у вигляді структури даних. Такий опис може містити геометричні дані, положення точки спостерігача, інформацію про освітлення. Зображення – це цифрове растрове зображення. Зазвичай під рендерингом розуміють накладення текстури на уже готову твердотільну модель (solid-works) у машинобудівних проєктах [128] та на каркас (framework) в інженерній графіці [129].

Трасування променів (англ.: ray tracing) у комп'ютерній графіці є способом створення зображення тривимірних об'єктів чи сцени за допомогою відстеження ходу променя світла крізь точку екрану і симуляції взаємодії цього променя з уявними об'єктами, що підлягають відображенню [129]. Цей спосіб дозволяє створювати надзвичайно реалістичні зображення, зазвичай значно вищої якості, ніж дає типовий алгоритм Scanline або ж метод відбивання променів (англ.: Ray casting), проте має значно вищу обчислювальну складність. Із цієї причини алгоритми трасування променів використовуються там, де немає суттєвих обмежень часу рендерингу.

Граничне подання – це опис меж об'єкту або абсолютного аналітичного завдання граней, що описують тіло [130]. Цей метод дозволяє створити якісне зображення геометричного твердого тіла, щоб встановити взаємну відповідність, потрібно задати кордони або контури об'єктів, а також ескізи різних видів об'єктів, і вказати лінії зв'язків між даними видами. Методи визначення складних контурів та векторизації растрових моделей було розглянуто у [131]. Для створення ГБД існують методи граничного (B-Rep) та конструктивного (C-Rep) їх представлення. В системі з B-Rep поданням моделі будуються з твердотільних примітивів. Ці примітиви визначаються розмірами, орієнтацією, формою та точкою прив'язки. Інструментами побудови C-Rep є булеві операції, вони базуються на алгебраїчної теорії множин. Найчастіше використовуються операції: різниця, перетин та об'єднання.

---

129. Энджел Э. Интерактивная компьютерная графика. Вводный курс на базе OpenGL. Изд. 2-е. Москва: Вильямс, 2001. 592 с.

130. Снук Г. 3D-ландшафты в реальном времени на C++ и DirectX 9. Изд. 2-е. Москва: Кудиц-пресс, 2007. 368 с.

131. Козир А. Є., Славко Г. В. Алгоритми і методи визначення складних контурів динамічних об’єктів з використанням технологій web-графіки. Вісник Кременчуцького національного університету імені Михайла Остроградського. 2016. Вип. 3 (98). Ч. 1. С. 20–26.

У кожного з цих методів управління об'ємними моделями є плюси і мінуси у порівнянні з іншими. У системи з C-Rep поданням, перевага полягає в первинному формуванні моделі. Крім того, дане представлення забезпечує більш зручний опис моделей у ГБД. B-Rep метод актуальний в утворенні складних структур, які дуже складно відтворити за допомогою C-Rep методу.

Перевагою систем з B-Rep є простіша зміна граничного подання у каркасну модель та зворотною її зміною. Причиною є те, що опис меж аналогічний опису каркасної моделі. Наприклад, проєктування литтєвого оснащення та ливарних форм є традиційною областю суцільного, об'ємного моделювання із імітацією руху. Найбільшою очевидною відмінністю від двовимірного креслення є точне створення об'ємної комп'ютерної моделі.

Існує величезна кількість різновидів БД, що відрізняються за різними критеріями. Наприклад, у [132], визначаються понад 50 видів БД. Основні класифікації ГБД наведені нижче, а їх графічну структуру вже подано на у вигляді структурної моделі на рис. 1.6. Деякі деталізовані пояснення щодо ГБД зведемо до табл. 1.4, 1.5.

Таблиця 1.4 – Класифікації ГБД за середовищем зберігання

Середовище зберігання:	Характеристика управління
оперативна пам'ять (англ.: in-memory database, memory-resident database, main memory database)	всі дані на стадії виконання знаходяться у оперативній пам'яті
вторинна пам'ять або традиційна (англ.: conventional database)	периферійна енергонезалежна пам'ять – як правило жорсткий диск (в оперативну пам'ять система управління БД (СУБД) поміщає лише кеш і дані для поточної обробки)
третинна пам'ять (англ.: tertiary database)	пристрій масового зберігання, що від'єднується від сервера (флеш-накопичувач або оптичний диск, віртуальне або хмарне сховище

132. Когаловский М. Р. Энциклопедия технологий баз данных. Москва: Финансы и статистика, 2012. 460 с.

Таблиця 1.5 – Класифікація ГБД за ступенем розподіленості

Вид:	Характеристика управління
централізована, або зосереджена (англ.: centralized database)	ГБД, що повністю підтримується на одному комп'ютері
розподілена (англ.: distributed database)	ГБД, складові частини якої розміщуються в різних вузлах комп'ютерної мережі
неоднорідна (англ.: heterogeneous distributed database)	фрагменти розподіленої ГБД в різних вузлах мережі підтримуються засобами більше однієї СУБД
однорідна (англ.: homogeneous distributed database)	фрагменти розподіленої ГБД в різних вузлах мережі підтримуються засобами однієї і тієї ж СУБД
фрагментована, або секціонована (англ.: partitioned database)	методом розподілу даних є фрагментованість (секціонування), вертикальне чи горизонтальне
тиражована (англ.: replicated database)	методом розподілу даних – є тиражування (реплікація)

Проте, не усі види ГБД можна класифікувати за загальними ознаками БД. Існують деякі ГБД, які традиційно відносяться до так званих інших видів (табл. 1.6).

Таблиця 1.6 – Інші види ГБД

Вид:	Характеристика управління
просторова (англ.: spatial database)	ГБД, в якій підтримуються просторові властивості сутностей предметної області, такі ГБД широко використовуються в геоінформаційних системах [133]
тимчасова, чи темпоральна (англ.: temporal database)	ГБД, в якій підтримується будь-який аспект часу, не враховуючи часу, обумовленого користувачем

133. Вамболь В. В. Идентификация источников формирования экологической опасности в местах несанкционированного скопления отходов. Вісник Кременчуцького національного університету імені Михайла Остроградського. 2016. Вип. 1 (96) С. 122–128.

Кінець таблиці 1.6

Вид:	Характеристика управління
просторово-часова (англ.: spatial-temporal database)	ГБД, в якій одночасно підтримується одне або більше вимірів в аспектах як простору, так і часу
циклічна (англ.: round-robin database)	ГБД, обсяг збережених даних якої не змінюється з часом, оскільки в процесі збереження даних одні й ті ж дані використовуються циклічно
надвелика (англ.: Very Large DataBase – VLDB)	ГБД, яка займає надзвичайно великий обсяг на пристрої фізичного зберігання; термін передбачає максимально можливі обсяги ГБД, які визначаються останніми досягненнями у технологіях фізичного зберігання даних і в технологіях програмного оперування даними

Кількісне визначення поняття «надзвичайно великий обсяг» змінюється із часом. У даний час вважається, що це обсяг, який вимірюється щонайменше ПБ (10^{15} байт). Для порівняння: у 2010 р. – найбільшими в світі вважалися БД з об'ємом сховища близько 100 ТБ [134].

Фахівці відзначають необхідність особливих підходів до УП, що містять надвеликі ГБД. Для їх створення нерідко виконуються спеціальні проєкти з метою пошуку таких системотехнічних рішень, які дозволили б хоч якось працювати з такими великими обсягами даних. Як правило, необхідні спеціальні рішення для дискової підсистеми, спеціальні версії операційної системи і спеціальні механізми звернення СУБД до даних [135].

Дослідження в галузі управління VLDB-проєктами завжди знаходяться на вістрі теорії і практики БД. Зокрема, з 1975 року проходить щорічна конференція International Conference on VLDB (Міжнародна конференція з надвеликих баз даних). Більшість досліджень проводиться під егідою некомерційної організації VLDB Endowment (Фонд цільового

134. C. J. Date. Date on Database: Writings 2000–2006. New York City: Apress, 2006. 566 p.

135. Когаловский М. Р. Перспективные технологии информационных систем. Москва: ДМК Пресс; Компания АйТи, 2003. 288 с.

капіталу «VLDB»), яка забезпечує просування наукових робіт та управління інформацією у галузі надвеликих БД і суміжних областях, зокрема ГБД.

1.6 Мережеві методи та моделі планування в управлінні проєктами

1.6.1 Мережевий графік

Автоматизовані системи планування ресурсів виробництва (англ.: Enterprise Resource Planning – ERP), зазвичай включають комп'ютерні програми, які у тій, чи іншій мірі автоматизують деякі етапи складання та коригування мережевих графіків, проте подібні програми мають досить високу ліцензійну ціну та непід'ємні для вітчизняного виробника.

Мережевий графік (МГ) – це динамічна модель планування виробничого процесу, яка відображає послідовність виконання процесів та погоджує їх завершення у часі із урахуванням витрат часу, ресурсів та вартості робіт із визначенням при цьому критичних місць [136].

Основні елементи структури МГ – робота та подія, перша відображає процес, у якому беруть участь як виконавці так й обладнання та матеріальні ресурси, що використовуються при: проєктуванні, постачанні, розв’язанні задач на персональному комп’ютері (ПК) та технологічному очікуванні [137].

Кожна робота мережевого графіка має конкретний зміст. Робота як процес управління вимагає витрат часу та ресурсів. Для наочного відображення порядку проведення робіт при побудові МГ, використовують зображувані штриховими гілками графів додаткові дуги – фіктивні роботи. Вони не вимагають витрати ані часу, ані ресурсів, а лише вказують, що початок однієї роботи залежить від закінчення іншої [137].

Подія позначує факт завершення однієї або декількох робіт, що безпосередньо передують події. Подія, що стоїть на початку МГ, називається початковою (вхідною), у кінці – кінцевою (завершальною). На відміну від робіт, події відбуваються миттєво без споживання ресурсів

---

136. Bondy J. A., Murty U. S. R. Graph Theory with Applications. New York: North-Holland, 1986. 270 p.

137. Березина Л. Ю. Графы и их применение: Пособие для учителей. Москва: Высшая школа, 1989. 326 с.

[138]. Шлях найбільшої довжини між вихідними і завершальними подіями називається критичним (L_m). [139]. Якщо критичний час не відповідає заданому проєктному нормативу, то скорочення термінів виробничого процесу необхідно починати зі скорочення тривалості критичних робіт [140].

МГ заснований на використанні математичної моделі – графа [141], [142]. Найбільш поширений тип МГ у вигляді робіт являє систему кіл та з'єднуючих їх спрямованих відрізків (ребер) (рис. 1.7), що відображають самі роботи, а кола на їх кінцях («події») – початок чи закінчення цих робіт [143].

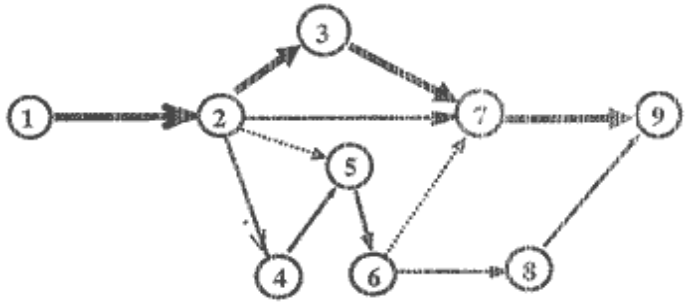


Рисунок 1.7 – Загальний вигляд типового мережевого графіка [144]

Наведений рисунок спрощено показує лише одну з можливих конфігурацій МГ, без даних, що характеризують самі заплановані роботи. Фактично, на МГ наводиться множина відомостей про виконані роботи. Над кожним ребром пишеться назва роботи, під ребром – тривалість, цієї роботи (зазвичай у днях чи годинах). У самих подіях, що розділені на сектори, також міститься інформація, зміст якої буде наведений у розд. 6. Фрагмент можливого МГ з такими даними щодо УП представлений на рис. 1.8.

---

138. Михеева В. С. Математические методы в экономической географии. Ч. 2. Приложение теории графов: Курс лекций. Москва: Математика, 1983. 316 с.

139. Голиков А. П., Трофимов А. М., Черванёв И. Г. Математические методы в географии. Харьков: Наука, 1986. 208 с.

140. Глазжан І. М., Новиков В. Г. Основи мережевого планування та управління. Харків : Вид-во ХГУ, 1996. 198 с.

141. Оре О. Теория графов. Москва: Наука, 1968. – 336 с.

142. Уилсон Р. Введение в теорию графов: пер. с англ. Москва: Мир, 1977. 208 с.

143. Спекторський І. Я. Дискретна математика. Київ: «Політехніка», 2004. 220 с.

144. Мала гірнича енциклопедія. У 3-х т. За ред. В. С. Білецького. Донецьк: «Донбас», 2004. 518 с.

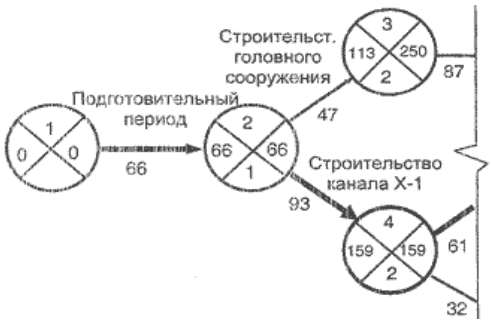


Рисунок 1.8 – Фрагмент мережевого графіка із зазначеними даними [144]

1.6.2 Діаграми Ґанта

Діаграми Ґанта (англ. *Gantt chart*, стрічкова діаграма, графік Ґанта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проєктом [145]. Є одним з методів планування та УП [146].

Перший формат діаграми був розроблений Генрі Л. Ґантом (Henry L. Gantt, 1861 – 1919) у 1910 році. Діаграма Ґанта представляє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу (рис. 1.9). Кожен відрізок відповідає окремому завданню або під задачі проєкту. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання [147].

Діаграма може використовуватися для представлення поточного стану проєкту: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні» (рис. 1.9) [145]. Часто діаграма Ґанта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці – містять додаткову інформацію про задачу (рис. 1.10) [146].

---

145. Jalote P. Software project management in practice. Indianapolis: Addison-Wisley, 2005. 242 p.

146. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Eight Edition. New Jersey: John Wiley & Sons, 2003. 914 p.

147. Schulte-Zurhausen Manfred: Organisation. 3-Auflage. Verlag Franz Vahlen: München, 2002. 284 p.

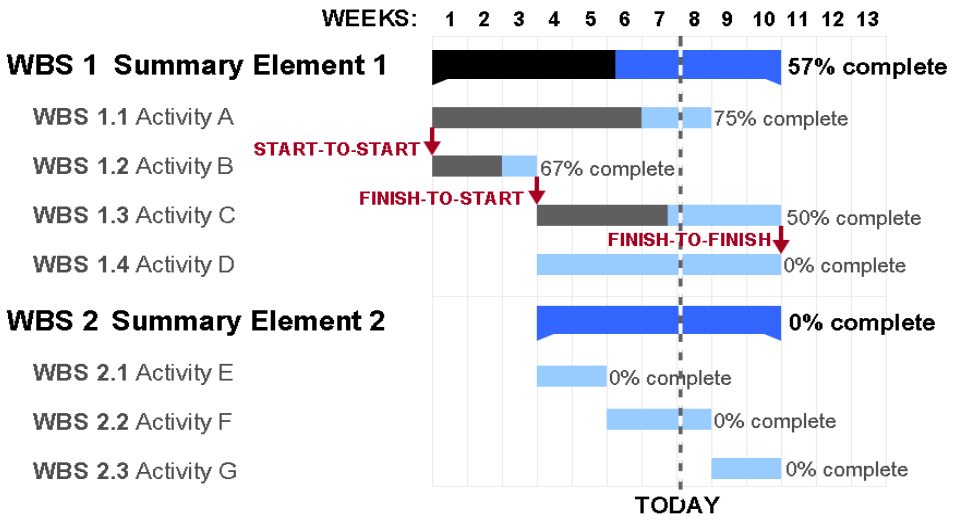


Рисунок 1.9 – Загальний вигляд діаграми Ґанта [145]

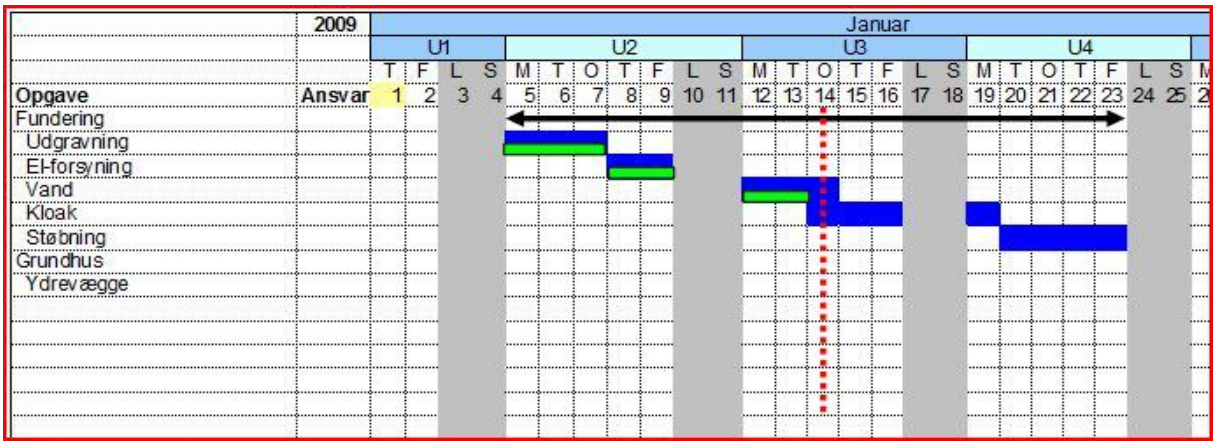


Рисунок 1.10 – Приклад діаграми Ґанта [146]

1.6.3 Методологія PERT

PERT – Program (Project) Evaluation and Review Technique (англ.) – техніка оцінки та аналізу програм (проєктів), яка використовується при УП [148]. PERT – це спосіб аналізу завдань, необхідних для виконання проєкту, особливо, аналізу часу, який потрібен для виконання кожної

148. Program Evaluation and Review Technique (PERT). URL: https://www.inc.com/encyclopedia/program-evaluation-and-review-technique-pert.html (дата звернення: 01.05.2019).

окремої задачі, а також визначення мінімального необхідного часу для виконання всього проєкту [149] .

PERT був розроблений головним чином для спрощення планування на папері та створення графіків великих і складних проєктів [148]. PERT призначений для масштабних, унікальних, складних, нерутинних проєктів [149]. Метод передбачав наявність невизначеності, даючи можливість розробити робочий графік проєкту без точного знання деталей і необхідного часу для всіх його складових.

Найпопулярнішою частиною PERT – є метод критичного шляху, що спирається на побудову МГ (мережевої діаграми PERT) [136]. Метод критичного шляху – ефективний інструмент планування розкладу та управління термінами проєкту [140]. В основі методу лежить визначення найбільш тривалої послідовності завдань від початку проєкту до його закінчення з урахуванням їх взаємозв'язку [148].

У зв'язку з цим при виконанні проєкту критичні завдання вимагають більш ретельного контролю, зокрема, своєчасного виявлення проблем та ризиків, що впливають на терміни їх виконання і, отже, на строки виконання проєкту в цілому. У процесі виконання проєкту критичний шлях проєкту може змінюватися тому, що при зміні тривалості задач деякі з них можуть опинитися на критичному шляху.

Найвідоміша частина PERT – це діаграми взаємозв'язків робіт і подій (рис. 1.11) [149]. PERT пропонує використовувати діаграми-графи з роботами на вузлах, з роботами на стрілках (МГ), а також діаграми Ґанта [146].

Зазвичай, початок і кінець реалізації проєкту – пов'язані множиною шляхів, довжини яких розрізняються [136]. Найбільша – визначає тривалість усього проєкту, мінімально можливу при зафіксованих характеристиках дуг графа [140]. Відповідний шлях – критичний, тобто саме від тривалості складових його робіт залежить загальна тривалість проєкту, хоча при зміні тривалості будь-яких робіт проєкту критичним може стати й інший шлях [145].

149. Punmia B. C., Khandelwal K. K. *Project Planning and Control with PERT and CPM.* New Delhi: Laxmi Publications, 2016. 258 p.

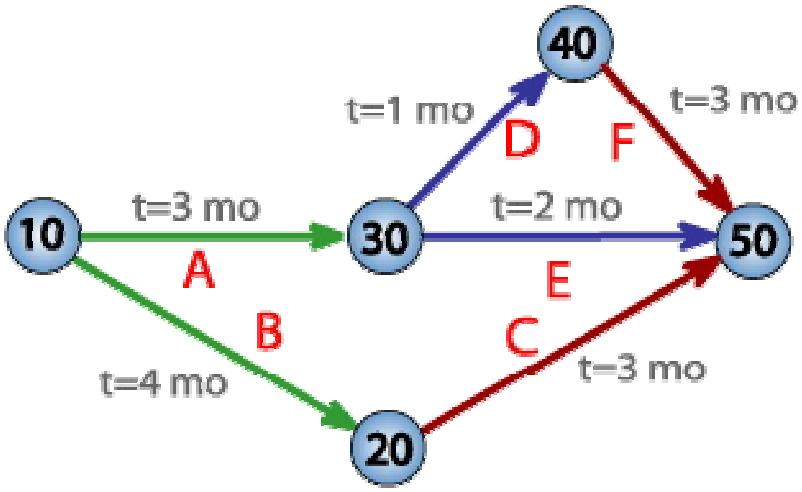


Рисунок 1.11 – Приклад мережевої діаграми PERT для проєкту тривалістю у сім місяців із п'ятьма проміжними точками (від 10 до 50) і шістьма діяльностями (від A до F) [149]

1.7 Постановка проблеми проактивного управління проєктами з розвитку програмних систем та продуктів у загальному вигляді

В результаті написання розділу, було визначено переваги і недоліки УП ПП із різною методологією її створення. Причиною обрання такої теми стало постійне зростання та популяризація відкритих ПП у всьому світі. У найпрогресивніших країнах вже давно проєкти зі створення нових ПП не розроблюється з нуля, для них використовуються системні дослідження, які допомагають набагато швидше й ефективніше створити будь-які потрібні структуровані дані.

Методологія «Open-Source» набула активного використання у всьому світі та швидко інтегрувалась у загальну структуру УП із розробки ПС. Її зручність у тому, що вона не прив'язана до жодного з методів розробки та є дуже гнучкою у використанні. Розвиток означеної методології притаманні для країн Заходу та Європи. Україна сильно відстала в цьому плані та тільки у 2010 р. почалися проєкти із активної експлуатації відкритих ПС. Під час проведення аналізу цієї галузі проблемною стала відсутність навчальних матеріалів щодо УП зі створення відкритих ПП.

Головна перевага проєктів зі створення відкритих та вільних ПП полягає в тому, що поширюються вони за вільною ліцензією та не виникає жодних юридичних проблем через: копіювання, модифікацію та інші дії із кодом. Так само слід зазначити, що оскільки код виконано відкритим, то

існує тенденція, що розробники додають коментарі, щоб цей код був зрозумілий.

Одним із головних недоліків УП по створенню відкритих систем є відсутність правильного та прийнятного інтерфейсу ПП, втім розробкою інтерфейсу дизайнери, зазвичай, займаються із половини, а то й ближче до закінчення проєкту – коли вже точно відомий повний функціонал та принципи роботи ПС.

З економічної точки зору використання відкритого ПП доцільніше, адже це суттєва економія бюджету компанії та розвиток професіоналізму проєктної команди, а наведений у розділі результат аналізу стане у пригоді при визначенні переваг чи недоліків по кожному подібному проєкту. Цей набір властивостей забезпечить однозначність обрання стейкхолдерами методології розробки ПП для задоволення конкретного функціонального галузевого набору.

Проактивний розвиток ПС та ПП, на відміну від реактивного являє собою величезну науково-технічну проблему, а його впровадження вимагає значних капіталовкладень [150], оскільки необхідно управляти проєктом з упередженням [151]. Усі ПС, вже створені та ті, що створюються за допомогою сформованої методології, містять у собі результати багаторічних досліджень тисяч стейкхолдерів: науковців, інженерів, конструкторів та програмістів, які брали участь у розробці проєктних рішень.

За сучасними світовими тенденціями УП, ПП повинні бути такими, що розвиваються [152]. Існує, принаймні, дві вагомі причини, за якими ПП повинні змінюватись за часом. По-перше: розробка такого складного об'єкта як ПС займає тривалий час та економічно вигідно вводити в експлуатацію частини системи за еволюційним механізмом [153]: введений

---

150. Blum B. Software engineering: a holistic view. URL: https://dl.acm.org/ citation.cfm?id=SERIES9569.128915 (дата звернення: 12.01.2019).

151. Бушуєв С. Д., Гогунський В. Д., Кононенко І. В. Формула та напрями наукових досліджень зі спеціальності «Управління проектами та програмами». Управління проектами: стан та перспективи : VIII Міжнар. наук.-практ. конф. Миколаїв: НУК, 2012. С. 28 – 31.

152. Klein M. Reengineering methodologies and tools. A Prescription for Enhancing Success. URL: https://www.tandfonline.com/doi/abs/10.1080/ 10580539408964633 (дата звернення: 12.01.2019). DOI: 10.1080/10580539408964633.

153. Link D., Behnam P., Moazeni R., Boehm B. The Value of Software Architecture Recovery for Maintenance (Submitted on 23 Jan 2019 in Cornell University). URL: https://arxiv.org/abs/1901.07700 (дата звернення: 27.06.2019).

в експлуатацію базовий варіант – надалі розширюється [154]. По-друге: постійний прогрес об'єктів проєктування, технологій управління інформацією, обчислювальної техніки та обчислювальної математики зумовлює появу нових, більш досконалих математичних моделей і методів, які повинні замінювати старі менш вдалі аналоги систем УП [155].

У зв'язку з цим, ПС повинні мати властивість зручності використання та можливості розширення за допомогою підключення розроблених та / або удосконалених видів забезпечення [156]. Ось тут постає питання подальшого вирішення цієї проблеми – це може бути нова розробка або реінжиніринг (reengineering) [157], тобто проактивний розвиток.

З комерційної точки зору, реінжиніринг часто вважають єдиним засобом збереження успадкованих модулів у експлуатації ПП. На думку провідних фахівців [158], [159], нову розробку проєкту такої системи не рекомендується розглядати не тільки з точки зору дефіциту часу, що вже було затрачено на первинну розробку, та як наслідок – збільшення

154. Manganelli R., Klein M. The Reengineering Handbook: A Step-by-Step Guide to Business Transformation. URL: https://www.sciencedirect.com/science/article/pii/S00https://journals.lww.com/jhqonline / Citation/1995/03000/ The_Reengineering_Handbook_A_Step_by_Step_Guide.11.aspx (дата звернення: 12.01.2019). DOI: 10.1097/01445442-199503000-00011.

155. Grover V., Malhotra M. Business process reengineering: A tutorial on the concept, evolution, method, technology and application. URL: https://www.sciencedirect.com/science/article/abs/pii/S0272696396001040 (дата звернення: 12.01.2019). DOI: 10.1016/S0272-6963(96)00104-0.

156. Boehm B. A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 1986. Vol. 11, Iss. 4. P. 14–24. DOI: 10.1145/12944.12948.

157. Hammer M., Champy J. Reengineering the corporation: A manifesto for business revolution. URL: https://www.sciencedirect.com/science/article/pii/S0007681305800643?via%3Dihub (дата звернення: 12.01.2019). DOI: 10.1016/S0007-6813(05)80064-3.

158. Норенков И. П. Основы автоматизированного проектирования: учеб. для вузов. 4-е изд., перераб. и доп. М.: Изд-во МГТУ им. Н. Э. Баумана, 2009. 430 с.

159. Тимченко А. А. Основи системного проектування та системного аналізу складних об'єктів. Кн. 1. Основи САПР та системного проектування складних об’єктів. Київ: Либідь, 2003. 272 с.

економічних витрат [160], а й з точки зору ризику виникнення структурних помилок проєкту [161].

Відсутність у більшості сучасних ПП логічно-об'єднаної системи задач управління ЖЦ продукції, особливо на ранніх стадіях, або взагалі відсутність методологічних ознак системного УП, що тягне за собою перенесення традиційних та / або застарілих засобів проєктування – змушує вдаватися до виконання реінжинірингу видів забезпечення ПП. У цей же час реінжиніринг дає змогу виконати еволюціонування ПС шляхом позитивних змін видів забезпечення з метою підвищення зручності експлуатації та супроводу ПП [162].

Також слід зупинитись на одній із найголовніших проблем УП зі створення ПС, а саме: уніфікація або універсальність [163]. Цю проблему можна віднести до «проблеми початку», під якою слід розуміти, що всі сучасні проєкти зі створення ПС (різного галузевого призначення), на превеликий жаль, сучасними не є – це пов'язано, перш за все, з тим, що створювались вони в тих мовах, які були актуальні на самому початку їх розробки [164]. Якщо проєкт не містить механізму проактивного розвитку (з упередженням), то більшість з них через 1,5 – 2 роки не витримує підвищених вимог щодо швидкості роботи з відтвореним графічним зображенням та його обчислювальним відновленням (рендерінгом), а системна трансформація вихідного коду з однієї мови в іншу, виходячи з того, що сучасні ПС можуть складатися з декількох мільйонів рядків коду, може зайняти місяці та навіть роки [165].

Головним результатом при розробці цільової програми реінжинірингу ПС стане формування основ його методології, що утворить

---

160. Subriadi A. P., Muqtadiroh F. A., Dewi R. S. A model of owner estimate cost for software development project in Indonesia. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2175 (дата звернення: 27.06.2019).

161. Boehm B. Software Risk Management. URL: https://link.springer.com /chapter/10.1007%2F3-540-51635-2_29 (дата звернення: 13.11.2018). DOI: 10.1007/3-540-51635-2_29.

162. Пантелеймонов А. А. Аспекты реинженерии приложений с графическим интерфейсом пользователя. *Проблемы программирования.* 2001. № 1–2. С. 53–62.

163. Фаулер М. Рефакторинг: улучшение соответствующего кода. Санкт-Петербург: Символ-Плюс, 2003. 432 с.

164. Boehm B. Spiral Development: Experience, Principles and Refinements: special Report. CMU. SEI-2000-SR-008. 2000. 37 p.

165. Selby R. W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management and Research. New Jersey: John Wiley & Sons, 2007. 818 p.

фундамент наукового потенціалу, який забезпечить подальший успіх в управлінні усіма проєктами з проактивного розвитку ПС.

Основні власні наукові дослідження, подані автором у першому розділі монографії, опубліковано у наукових працях автора: [3], [6], [10], [16] – [22], [25], [26], [32], [34], [45], [48], [54], [56], [57], [59], [62] – [65], [69].

2 ДОСЛІДЖЕННЯ ТА РОЗВИТОК ПРОВІДНИХ МЕТОДОЛОГІЙ УПРАВЛІННЯ ПРОЄКТАМИ З ПРОАКТИВНОГО РОЗВИТКУ ПРОГРАМНИХ СИСТЕМ ТА ПРОДУКТІВ

Основними вхідними критеріями, що встановлені перед проєктуванням ПС будь-якого призначення – це скорочення строків її проєктування і персоналу, який необхідний для здійснення цього, та як наслідок – собівартості готового ПП.

У методологічному плані розвиток ПС можна визначити як комплекс наукових методів дослідження діяльності у галузі проєктування з метою розробки системи методик, математичних моделей і алгоритмів, що забезпечують найкращий розподіл функцій між проєктувальником та комп'ютером, при пошуку оптимальних проєктних рішень.

У результаті всебічного вивчення об'єктів проєктування, розробляються моделі, методи та алгоритми, які з максимально можливою точністю фіксують майбутні закономірності формоутворення об'єктів та логіки, що відбуваються при проєктуванні та виготовлені ПП.

Відповідно до принципу «людина-машина», що покладено в основу побудови ПС, в технічному аспекті, систему можна визначити як комплекс інформаційного, програмного, технічного забезпечення системи підпорядкованої єдиної методології машинного проєктування.

Для завдання методологічних основ реінжинірингу, необхідно увести низку основоположних понять предметної області [166] дослідження.

Реінжиніринг (reengineering) ПС – це проактивний розвиток системи шляхом її докорінної зміни з метою підвищення зручності її експлуатації, супроводу або еволюційної зміни її функцій.

Метод реінженерії ПС – цільовий засіб отримання нового компонента шляхом виконання послідовності операцій внесення змін, модернізації або модифікації, а також перепрограмування окремих компонентів ПС. Реалізується сукупністю моделей, методів і процесів, що змінюють структуру і можливості компонентів з метою отримання компонента з новими можливостями. Нові компоненти ідентифікуються

166. Любченко В. В. Правила декомпозиції при формуванні моделі предметної області для навчальних курсів. *Восточно-Европейский журнал передовых технологий.* 2009. Т. 1, № 2(37). С. 42–44.

іменами, які використовуються при створенні компонентних конфігурацій і каркасів ПС [167].

Проєкт реінжинірингу містить у собі процеси реорганізації та реструктуризації ПС, переведення окремих компонентів системи на іншу, сучаснішу мову програмування, а також процеси модифікації або модернізації структури і системи даних [167], при цьому архітектура системи може залишатися незмінною [168].

З технічної точки зору реінженерія – це вирішення проблеми еволюції ПС шляхом зміни її компонентів та адаптації архітектури до нового середовища, в якому компоненти розміщуються згідно конфігурації операційної системи. Причиною еволюції може бути зміна МП ПС (наприклад: Fortran, Cobol або навіть С тощо) з переходом на сучасні об'єктно-орієнтовані мови (Java, C#, Python тощо).

Проте з комерційної точки зору реінженерію часто вважають єдиним способом збереження успадкованих систем в експлуатації, оскільки повна («із нуля») еволюція системи є дорогою та ризикованою процедурою продовження часу існування ПС.

2.1 Концепція управління проєктом з розвитку програмних систем та продуктів

2.1.1 Критерії доцільності реінжинірингу та управління вартістю проєкту

Головна відмінність між реінжинірингом і новою розробкою ПС з точки зору УП полягає в тому, що опис системної специфікації починається не з «нуля», а з розгляду можливостей старої успадкованої системи, за рахунок цього повторного використання компонентів. Згідно з даними [167] повторне використання в 4 рази дешевше, ніж нова розробка ПС. Проте, слід помітити, що у деяких випадках, все ж таки дійсно вигідніше застосувати повторну розробку.

У зв’язку з цим, одним з підзавдань, що ставляться для розв’язання у поданому підрозділі монографії – є задача формалізації критеріїв

---

167. Лаврищева Е. М., Грищенко В. Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. доп. и перераб. Киев: Наук. думка, 2009. 372 с.

168. Jacobson I., Ericsson M., Jacobson A. The Object Advantage: Business Process Reengineering with Object Technology. ACM Press. URL: http://eaststemcell.com/files/storage.cloud.php?id=MDIwMTQyMjg5MQ== (дата звернення: 12.01.2019).

доцільності реінжинірингу ПС та управління вартістю проєкту з розвитку ПС, за якими, після побудови визначених порівняльних характеристик буде прийматись однозначне рішення щодо застосування або відхилення реінжинірингу.

Проте, слід зауважити, що ні в якому разі не можна будувати порівняльні характеристики лише на критеріях, що охоплюють лише інструментарій розвитку ПС. Подані критерії повинні базуватися на всебічному аналізі усіх видів забезпечення проєкту з розвитку ПС.

2.1.2 Цільова програма проактивного розвитку проєкту

Наступним підзавданням, що ставиться у монографії, вже після прийняття рішення щодо застосування реінжинірингу – є задача розробки цільової програми проактивного розвитку (ЦППР) ПС, що є похідною від логічної схеми проєктування [159], яка включає в себе алгоритмічний вибір компонентів реінжинірингу.

ЦППР представляє – цільовий засіб отримання нового компонента ПС шляхом виконання послідовності процесів внесення змін, модернізації або модифікації, а також управління перепрограмування окремих компонентів ПС. ЦППР починається з експерименту, об’єднуючи зовнішню (формування цілей) та внутрішню (моделювання) частини [169].

Екстраполюємо ідеї С. Д. Бушуєва [80], [81] на забезпечення проєктів зі створення ПС, сформуємо модель варіантів використання (ВВ) видів забезпечення ПС (рис. 2.1) й проаналізуємо кожен з видів забезпечення.

ЦППР складається із сукупності моделей, методів і процесів УП, що змінюють структуру й можливості компонентів ПС з метою отримання продукту з новими можливостями. Нові компоненти ідентифікуються іменами, які використовуються при створенні компонентних конфігурацій і каркасів ПС.

Спираючись на стандарти ISO 9000, ISO 10006, ISO/DIS 21500 та ISO/IEC/IEEE 29148, ЖЦ проєктів повинні мати розвиток управління стадіями проєкту. Наукова парадигма управління фазами проєктування кінцевого продукту (технічної системи), що закладена у роботах [86], [87],

---

169. Тимченко А. А. Основи системного проектування та системного аналізу складних об'єктів. Кн. 2. Основи системного підходу та системного аналізу об’єктів нової техніки. Київ: Либідь, 2004. 288 с.

містить стадії: технічна пропозиція, ескізний проєкт, технічний проєкт, а також виготовлення.

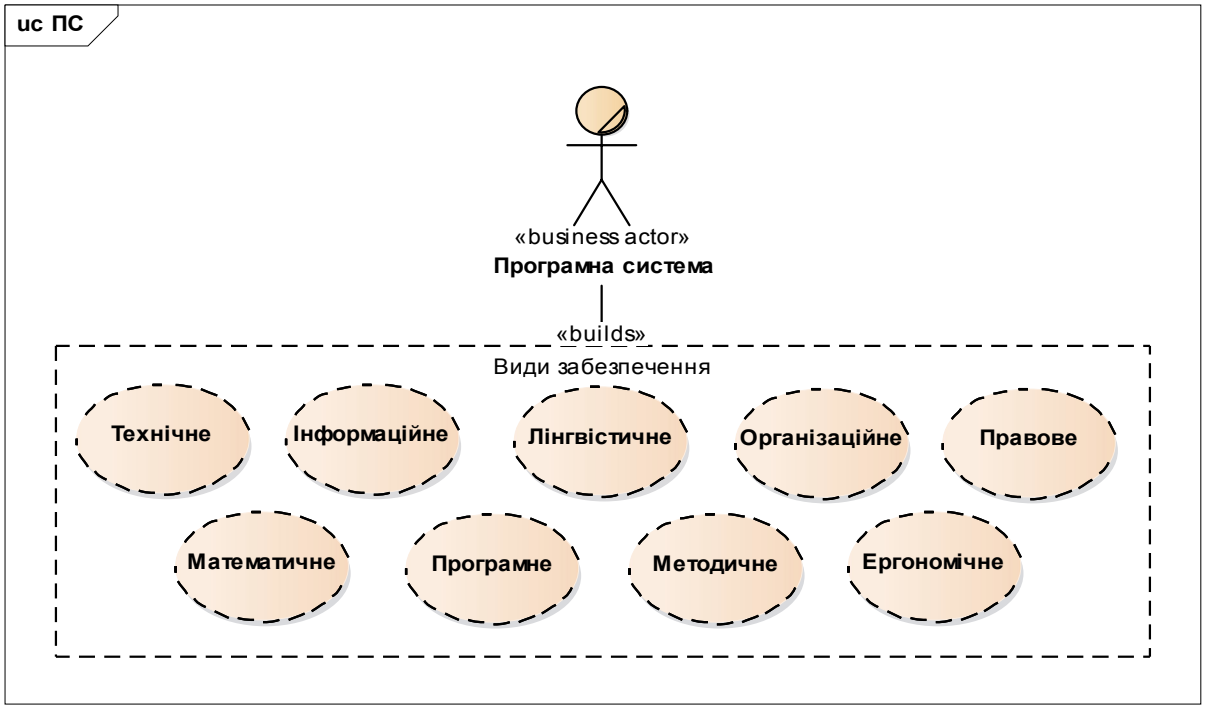


Рисунок 2.1 – Модель ВВ видів забезпечення ПС

Розглянута парадигма була розширена автором монографії на життєві фази (стадії) розвитку проєкту зі створення ПП (із урахуванням стандарту ISO/IEC/IEEE 29148). На рис. 2.2 приведена розроблена модель станів, що містить фази (стадії) розвитку проєкту зі створення ПП у квадратних дужках наведені «обмежуючі умови», тобто: якщо умова невиконана – перехід до наступної стадії не відбувається).

При використанні ЦППР ПС можна виділити наступні основні етапи, до яких можна віднести:

– переклад початкового коду у застарілій МП на сучасну версію цієї мови або в іншу мову;

– аналіз ПС, згідно з документованою структурою і функціональними можливостями системи;

– модифікація структури ПС для нарощування нових властивостей і можливостей;

– розбиття ПС на модулі для їхнього групування та усунення надмірності;

– зміна даних, з якими працює ПС.

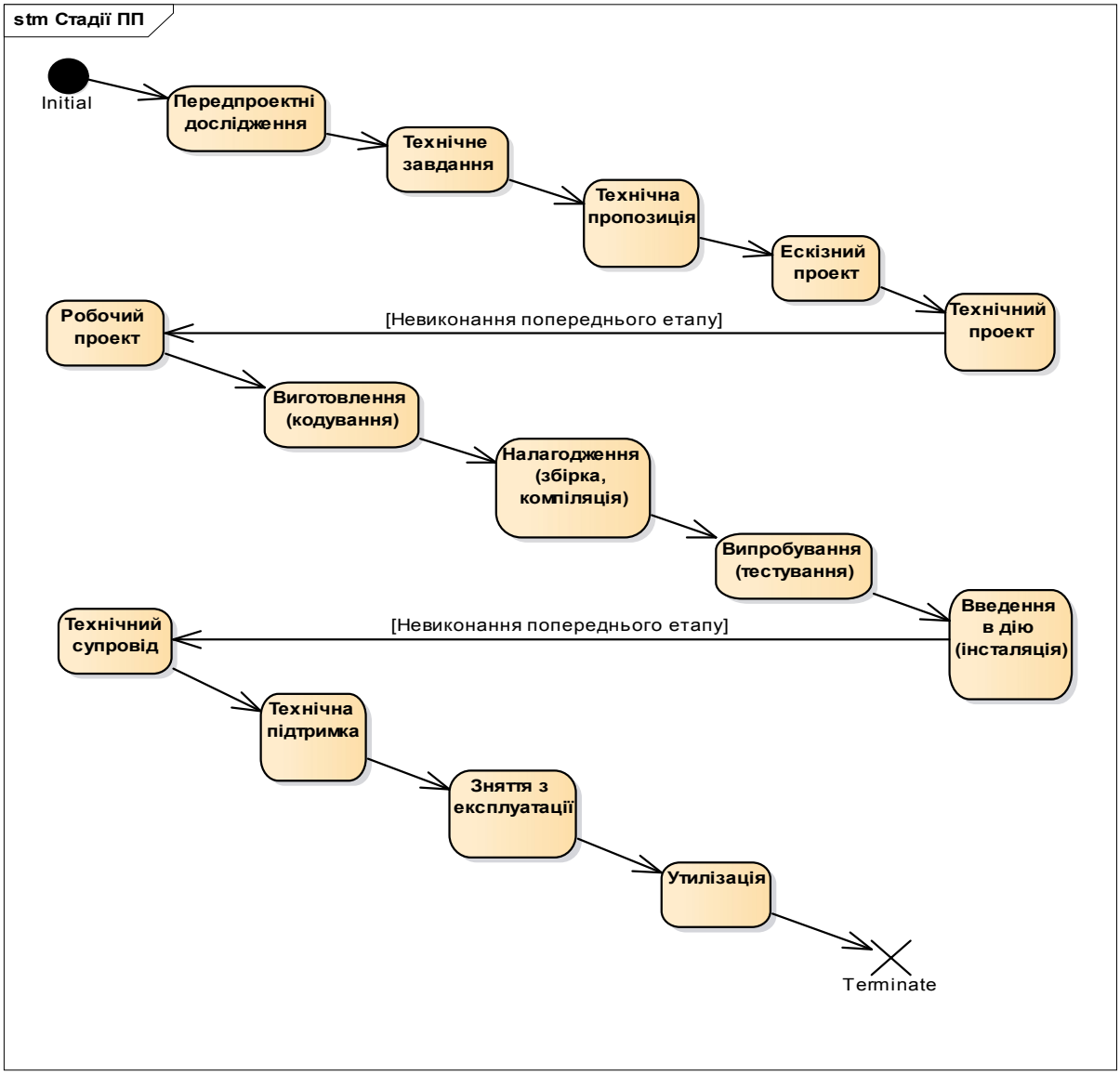


Рисунок 2.2 – Модель станів, що містить фази (стадії) розвитку проєкту зі створення ПП

У цьому разі, ПС, до якої було застосовано ЦППР – набуде ознаки переходу на новий якісний рівень – системне проєктування, що складається з процесів із неформальними процедурами системних досліджень, методологічно пов’язаних із процесами імітаційного та математичного моделювання УП на сучасному технічному забезпеченні.

Вірно виконана ЦППР ПС обов’язково характеризується досягненням таких результатів:

– зниження ризику виникнення помилок при майбутньому оновленні ПС;

– зниження собівартості ПП за рахунок повторного використання компонентів при розробці нової ПС та скорочення працемісткості за рахунок майже повного виключення рутинних операцій програмування багатьох вже ідентифікованих компонентів.

Планується, що використання ЦППР ПС дасть значне підвищення ефективності УП з розвитку ПС в усіх сферах їх використання.

2.1.3 Принцип декомпозиції проєкту на робочі пакети

Робочі пакети визначають зону діяльності в системі проєктування з вирішенням всіх завдань, що пов'язані з розробкою проєктів для кожної з виділених груп об'єктів проєктування. Кожний робочий пакет поєднує в собі діяльність з проєктування всіх об'єктів одного виду та відносяться до однієї певної галузі УП.

У свою чергу, робочі пакети мають свій поділ на взаємопов'язані зони, що спеціалізовані за проєктуванням об'єктів різного призначення. Межі між робочими пакетами відокремлюють замовлення на проєктування (вхід до підсистеми) від завдання на проєктування, яке є початковою стадією у проєктному процесі і в результаті якої, може народжуватись замовлення на УП.

На рис. 2.3 представлено розроблену модель композитної структури процесу уведення до експлуатації ПС та подальше проактивне УП розвитку ПП.

Завдяки потужним обчислювальним підсистемам, що містять банки і БД готових проєктно-конструкторських рішень, можливо швидко внести корегування у необхідні параметри розвитку об’єктів проєктування (розміри, форма, порядок обробки тощо), що виготовляються; а також у послідовність операцій, тобто переорієнтувати увесь виробничий процес.

Проте використання принципу декомпозиції проєкту на робочі пакети дає значну ефективність в тому випадку, коли результати УП використовуються у виробництві ПС. Ефективність, в даному випадку, обумовлюється тим, що за сучасних темпах розвитку науки і техніки, виникає протиріччя між зростаючим рівнем науково-технічних досягнень та існуючими методами й технічними засобами УП.

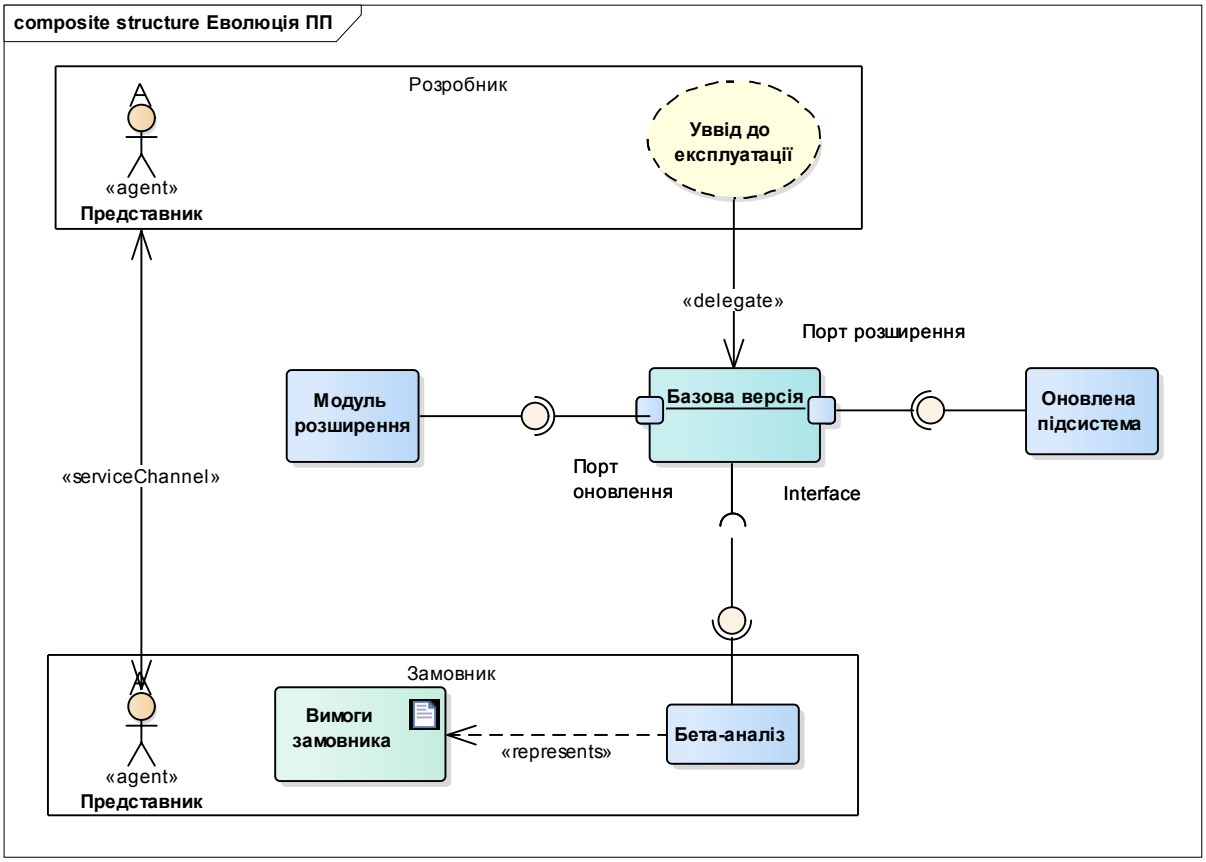


Рисунок 2.3 – Модель композитної структури процесу уведення до експлуатації ПС та подальше проактивне УП розвитку ПП

2.1.4 Перевизначення життєвого циклу програмного продукту

ЖЦ ПП – це період часу між початком розробки ПС і завершенням використання ПП. Він характеризується кількома процесами, що відображають закономірності при розробці і застосуванні програмних засобів. Різні підходи, методи, інструментальні засоби УП можуть вносити відмітні особливості в окремі процеси, однак сам зміст процесів ЖЦ ПП цілком визначений. Найбільш загальними є наступні процеси, що зведені до вигляду розробленої моделі профілів (рис. 2.4), також до моделі зведено методи перевизначення ЖЦ ПП.

Постановка завдання. На даному процесі формулюється завдання, для вирішення якого необхідно розробити ПП. Визначаються: доцільність розробки, її вартість, сфера застосування, ефект від впровадження. Умова розв'язуваної задачі оформлюється у вигляді її специфікації: технічного завдання.

Проєктування. Цей процес може ділитися на окремі підпроцеси, що відображають ієрархічну структуру ПП, включаючи загальне проєктування, проєктування підсистем, програм і окремих модулів. Проєктування складається з розробки алгоритмів розв'язання поставлених завдань, абстрактних структур даних, операційного середовища функціонування. Результат проєктування: специфікації програмних об'єктів засобами МП і опис інформаційних структур. Тут вирішується також питання про підбір готових програм, що виконують необхідні функції та підфункції деякої задачі предметної області [170].

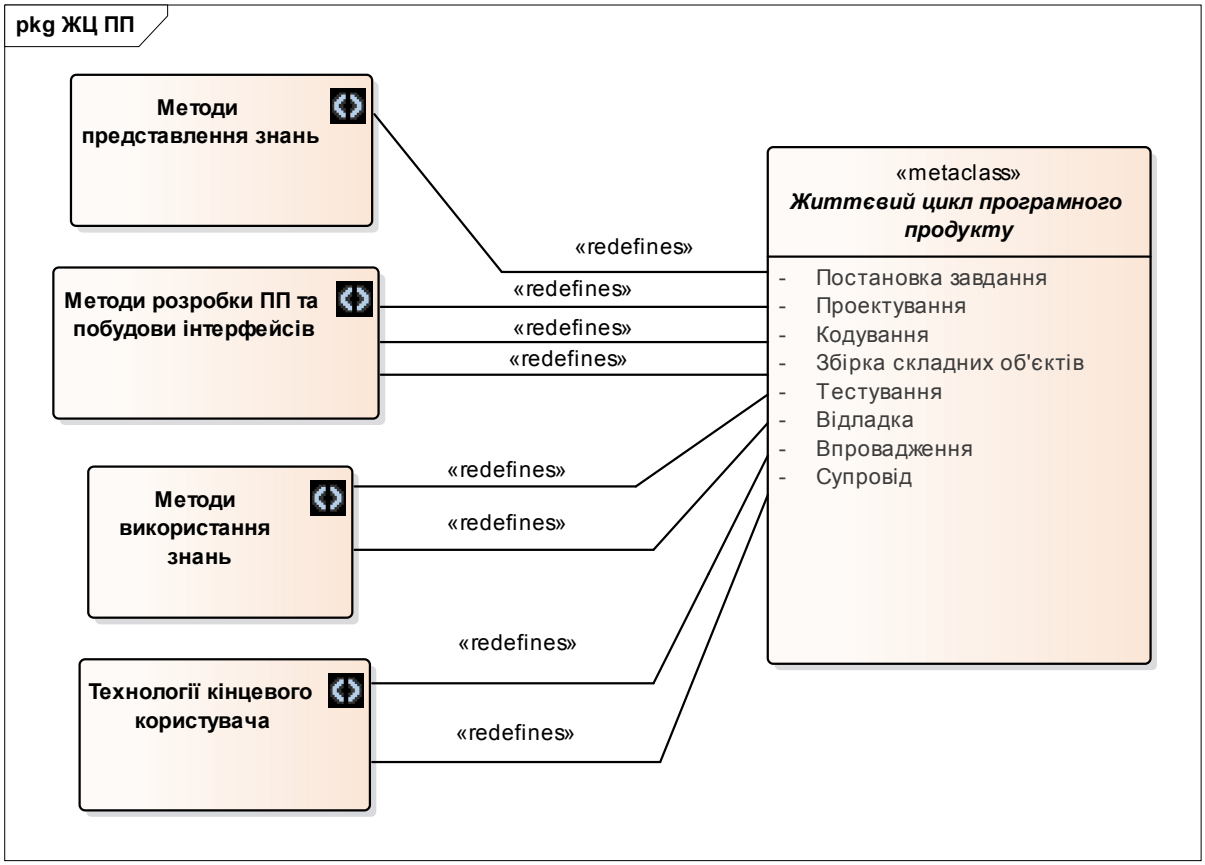


Рисунок 2.4 – Модель профілів із представленням методів перевизначення ЖЦ ПП

Кодування. Специфікації проєктування для знову розроблювальних компонентів ПП переводяться в тексти на МП. Структури даних

170. Денисюк А. В., Кавицкая В. С., Любченко В. В. Определение адекватной модели для представления знаний в современных информационных системах. *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка.* 2014. № 61. 114–119

конкретизуються та відображаються в реальну пам'ять. Вихідні тексти переносяться на зовнішні носії інформації. Готові компоненти перевіряються на виконання функцій в наборі вихідних даних.

Збірка складних програмних об'єктів. Здійснюється складання готових ПП з числа повторно використовуваних і знову розроблювальних методом комплексування або інтеграції. У результаті створюється інтерфейсне середовище, яке включає в себе програми перетворення переданих даних і передач управління відповідним компонентам об'єкта збірки.

Тестування. Об’єкт проходить тести, що перевіряють створені інтерфейси (правильність передач даних між об'єктами) і функціонування всього об'єкту. Незадовільні результати тестування призводять до заміни окремих елементів або до виправлення в них типів і структур даних, що передаються. Ці дії викликають виконання вказаних вище процесів ЖЦ. Результат цього процесу – зразок.

Налагодження ПП. Цей процес поділяється на декілька підпроцесів, пов'язаних з тестуванням і налагодженням різних видів програмних об'єктів, що входять до складу ПП, а також з усуненням різних типів помилок, виявлених при кодуванні або проєктуванні. Помилкові результати даного процесу призводять до виконання попередніх дій і необхідних корегуючих змін в ПП.

Впровадження ПП. Складається технологічна документація, розробляються контрольні приклади, проводиться дослідна експлуатація ПП. Після завершення дослідної експлуатації та виправлення виявлених помилок дослідний зразок ПП вважається готовим до промислової експлуатації (ПС набує властивостей ПП).

Супровід. Цей процес – останній у ЖЦ і триває до завершення використання ПП. Закінчення застосування пов'язано з:

а) розробкою більш якісного і досконалого ПП;

б) завершенням розв'язуваної задачі і моральним старінням самого ПП або операційного середовища його функціонування.

У процесі супроводу відбувається усунення раніше не виявлених помилок та зміна ПП, що пов'язані з удосконаленням технології його застосування.

Використання конкретного методу програмування позначається на тривалості кожного процесу ЖЦ, його складності та вживаних інструментальних засобах. У розглянутих вище процесах ЖЦ

створюваного ПП явно не відображені процеси його документування та виготовлення. Це пов'язано з тим, що управління розвитком ПП регламентує поступову розробку документації, її коригування у всіх процесах проєктування і виготовлення ПП. Тому до моменту виготовлення і впровадження значна частина документації вже підготовлена.

В процесі виготовлення здійснюється копіювання носіїв програм, доробка документації та підготовка носіїв ПП. Застосування інструментальних засобів підтримки розробки спрощує процес управління документацією, і тому відпадає необхідність у виділенні його як окремого процесу в рамках управління складальним програмуванням.

Методи представлення знань. ПС – це фактично один із способів представлення знань про вирішувані завдання, конкретних предметних областях. З даної позиції розглянемо більш докладно процес розробки та використання ПП.

При розробці баз знань необхідно розглянути два ключових питання: уявлення знань та їх використання. Методами представлення знань в розглянутій області по суті є методи і засоби розробки ПП (рис. 2.4) як знання фахівців-розробників. При цьому мовами подання знань служать: специфікацій, програмування, опису даних, управління завданнями тощо. Самі розроблені ПС представляються як частина знань про предметну область, до якої належить розв'язувана задача.

Методи використання знань. Використання знань пов'язано з процесами впровадження (частково) та супроводу ЖЦ ПП, тобто його застосування. Метод використання ПП – технологія його застосування. Результати розглянутого підходу до уявленню знань про повторно використовуваних ПС відображені на рис. 2.3.

Головна відмінність реінжинірингу від інших методів повторного використання полягає в наявності «стійкого зворотного зв'язку» (перевизначення (redefines) – на рис. 2.4). Цей «зворотній зв'язок» означає, що до технології використання розробленого ПП входять методи його застосування при проєктуванні інших ПП, де перевизначення входить як складовий елемент. Термін «стійкий» підкреслює, що методи застосування розроблених підсистем у подальшій розробці ПС є необхідною умовою використання методу реінжинірингу.

Слід зазначити, що методи реінжинірингу є узагальненням традиційних методів проактивного розвитку ПС, але як вони додатково містять у собі і методи побудови інтерфейсів між окремо розробленими

підсистемами. Процес проєктування включає вибір готових ПП, які повністю або частково вирішують поставлену задачу, і рішення проблем інтерфейсів між компонентами. Якщо потрібна додаткова розробка нових і складних ПС, то формується завдання, що розв'язується із застосуванням методів програмування. Процес кодування спрощується за наявності готових декількох або всіх необхідних компонентів. Основна увага при цьому приділяється комплексному налагодженню ПС, створених з готових компонентів.

Головні напрямки процесу об'єднання накопичених у цих базах знань різнопланових програмних ресурсів – розвиваються сучасні методи управління виготовленням ПС, отже й перевизначається ЖЦ ПП.

2.1.5 Планування управління складанням програмних об’єктів

Під плануванням управління складанням програмних об'єктів розуміється схема їх взаємодії, що визначається безпосередніми зверненнями до компонентів або послідовністю їх виконання. При цьому взаємодія кожної пари об'єктів залежить від спільного використання проєктних даних. У загальному випадку, схема управління складається із сукупності моделей, що відображають різні типи зв'язків між компонентами:

а) передача управління;

б) обмін інформацією;

в) умови спільного функціонування тощо.

Операції складання виконуються згідно із паспортами об'єктів та правилами сполучення. Інформація у паспортах повинна бути систематизована та виділена у такі окремі групи управління:

а) дані, що передаються;

б) типи цих даних;

в) об'єкти, що викликаються;

г) файли, що спільно використовуються тощо.

Правила сполучення визначають сумісність поєднуваних об'єктів та містять опис функцій управління, необхідних для узгодження різних характеристик, наведених у паспортах.

Процес складання ПС може проводитися ручним, автоматизованим та автоматичним способами. Як правило, останній спосіб практично нездійсненний, що пов'язано із недостатньо формальним визначенням

програмних об'єктів та їх інтерфейсів. Ручний спосіб недоцільний тому, що складання готових компонентів являє собою великий обсяг дій, які носять скоріше рутинний, ніж творчий характер. Найбільш прийнятний – автоматизований спосіб збірки у середовищі системи, яка за заданими специфікаціями (моделями) програм здійснює управління складанням за допомогою стандартних правил сполучення під контролем людини.

Засоби, що підтримують даний спосіб збірки, називаються інструментальними засобами управління складальним програмуванням.

До них відносяться засоби:

– комплексування (об'єднання компонентів у більш складний об'єкт);

– інтерфейсів (реалізація сполучення об'єктів згідно їх паспортів та стандартними правилами зв'язку);

– опису та використання моделей складального програмування (сукупність моделей складання різних програмних об'єктів).

Із розглянутої схеми виділимо завдання управління складального програмування та умови його застосування:

а) вибір компонентів із числа готових програмних об'єктів для забезпечення процесу вирішення класу задач з певної предметної області;

б) проєктування структури (моделей) створюваного об'єкта, елементами якого є готові програмні елементи, визначені на множині даних обраної предметної області;

в) складання (згідно з моделями) ПП, що реалізує функції, які відповідають цілям та задачам автоматизації предметної області.

Необхідні умови застосування даного плану включають у себе:

а) наявність великої кількості різноманітних ПП, як об'єктів складання;

б) паспортизація програмних об'єктів складання;

в) наявність достатньо повного набору стандартних правил сполучення й алгоритмів їх реалізації, засобів автоматизації процесу складання;

г) створення технологій застосування розроблених об'єктів для використання у більш складних ПП.

Остання умова означає, що повинні існувати певні форми уявлення ПС як знань про предметні області [171], універсальні з точки зору проєктування та розробки ПС.

Основне питання управління складанням – це реалізація інтерфейсів між окремими модулями та / або компонентами, що забезпечують їх «стикування» або зв'язок.

2.2 Методи проактивного розвитку програмних систем

Проактивне УП з розвитку ПС містить умови підвищення продуктивності її функціонування за рахунок використання готових виробничих ресурсів, а саме компонентів повторного використання (КПВ) зі стандартними або уніфікованими інтерфейсами ПП, згідно із заданими вимогами замовників. Складальний процес управління розвитком проєкту ПП апріорі ґрунтується на контролі діяльності виконавців та оцінці показників якості як окремих КПВ, так і ПП в цілому. Іншими словами, новими закономірностями у сучасному проактивному УП з розвитку ПС та ПП є:

а) поширення на сферу програмування промислових методів УП (планування працевитрат, визначення працемісткості, облік, контроль результатів та якості праці тощо) при розвитку ПС;

б) перенесення акценту з процесу програмування ПС на більш ранні процеси проєктування та моделювання, які забезпечують аналіз предметної області та формування вимог до створюваного ПП;

в) введення в практику УП розвитку ПП таких понять як модель ЖЦ, технологічна карта, маршрут та ін., які є основним фундаментом організації проєкту розвитку ПС.

Таким чином, мета поданого підрозділу – систематизація методів інтеграції у нові програмні структури КПВ, програм та готових ресурсів, що накопичено людством за визначений час, тобто методів проактивного розвитку ПС та ПП.

---

171. Любченко В. В. Модели знаний для предметных областей учебных курсов. *Искусственный интеллект.* 2008. №4. С. 458–462.

2.2.1 Базові методи реінжинірингу програмних систем

Інтеграція програмних структур спочатку виконувалася за допомогою готових підпрограм бібліотек різного призначення шляхом їх вставки до ПС, що інтегрується. Згодом з'являються різні методи реінжинірингу у вигляді нового знання (конкретизуюче, синтезуюче, композиційне тощо), які вирішували проблему комплексування програмних об'єктів методами, близькими до складання різнорідних об'єктів (табл. 2.1).

Методи реінжинірингу сприяють поступовому розвитку індустріальних основ виготовлення ПС: КПВ, ЖЦ, вимір та оцінювання робіт на стадіях ЖЦ й самого ПП на окремі показники якості. Пізніше сформувалися й інші стилі програмування, які використовують готові компоненти в процесі створення ПС методами складального типу – комплексування, інтеграція, композиція. До них можна віднести: об'єктно-орієнтоване, компонентне, генеруюче, аспектне, агентне тощо.

Таблиця 2.1 – Базові методи реінжинірингу ПС

Нове знання	Механізм	Метод управління	Засіб адаптації
Конкретизуюче програмування (походить від наявності деякого універсального ПЗ)	Універсальна багатопараметрична структура ПС (СУБД, пакети прикладних програм тощо)	Метод адаптації з технікою змішаних обчислень, оптимізаційних перетворень і генерації ПС, як конкретизація більш загальної програми до умов її застосування (до структур даних, функцій конкретної предметної області)	Спеціалізована технологія для адаптації СУБД до умов застосування за допомогою: мови опису, типів даних і засобів маніпулювання; розбиття ПС на дрібні програми; нестандартні функції перетворення

Продовження таблиці 2.1

Синтезуюче програмування (походить від постановки завдання, яке складається у вигляді моделі обчислень або специфікації програми вирішення задачі)	Модель обчислень, що відображає поняття і відносини предметних областей у вигляді моделі програми, що містить опис даних і макроозначень. Специфікація програми – опис задачі в термінах математичних понять, що використовується як завдання при розробці програми	Метод доказової генерації на основі керуючої програми, заданої послідовністю операторів-викликів, що реалізують відношення між об'єктами моделі обчислень. Метод покрокового уточнення (символічний метод вирішення задачі) – доказове судження (програма) про суть обчислення задачі	Система, у якій за описом моделі обчислень вибирається шлях і будується програма рішення задачі, шляхом синтезу окремих програм-задач. Система із вбудованими типами даних для створення БД. Система містить специфікації задач і збирача програм.
Композиційне програмування (композиція функцій і даних в програми)	Функції та операції композиції в логіко-математичній системі	Процес композиції включає у себе: дані, функції, операції, композиції та десріптори (вирази, терми, формули)	Система побудови композиційних програм

Кінець таблиці 2.1

Нове знання	Механізм	Метод управління	Засіб адаптації
Складальне програмування (походить від банку модулів, програм, ПС, КПВ тощо)	Схема (модель) зборки – це орієнтований навантажений граф, у вершинах якого знаходяться елементи банку, а дуги завдають зв'язки між ними та режимом функціонування	Метод комплексування різномовних модулів за схемою, яка задає їх взаємозв'язок за управлінням та за даними, що перетворені до релевантних типів даних. Інтеграція ПС за схемою з об'єктів банку модулів в середовищі інтеграції та перетворення різних типів даних і передач управління	Система реалізує зв'язок різномовних модулів у МП, через модулі посередники, що забезпечують необхідне перетворення типів і структур незбіжних даних. Система встановлює зв'язки між програмними компонентами за маршрутною схемою

Розглянемо базові методи реінжинірингу ПС (з табл. 2.1) з елементами інтеграції, комплексування та синтезу.

Конкретизуюче програмування базується на виділенні з деякої універсальної програми окремої її частини, налаштованої на особливі, певні умови виконання. Можна відзначити два типи такого виділення. Перший характеризується формуванням конкретної програми та аналогічний процесу макрогенерації. Другий тип пов'язаний із конкретизацією інформаційних структур у програмному засобі, що використовується.

При *синтезуючому* програмуванні будується модель програми за специфікацією завдання, за якою буде синтезована програма її вирішення. Специфікація задається у термінах деякої формальної мови. На її основі та правил побудови алгоритмів опису конкретної предметної області відбувається формування необхідної програми.

Композиційне програмування базується на принципах функціональності та композиційності, які розглядають програми як набір функцій, що будуються з інших функцій за допомогою спеціальних операцій, названих композиціями. На основі композиційного уточнення (експлікації – explication) створюється логіко-математична система композиційної побудови програм, яка об'єднує сучасні парадигми програмування (структурне, функціональне, об'єктно-орієнтоване тощо) у рамках єдиної концептуальної, експлікативної платформи. Її основу складають три типи об'єктів: самі об'єкти, методи композиції та засоби побудови з одних об'єктів інших об'єктів або функцій.

Складальне програмування характеризується складальною побудовою програм із готових «деталей», якими є програмні об'єкти різного ступеня складності. Елементи процесу складання присутні у багатьох методах програмування: згори-униз, знизу-догори тощо.

Проєктні менеджери, розробляючи програми без застосування будь-яких методів реінжинірингу, виділяють повторно використовувані оператори та оформляють їх у вигляді окремих, самостійних фрагментів або підпрограм для подальшого використання.

Виникають питання: у чому суть складального програмування та що дозволяє виділити його у нове знання з реінжинірингу ПС. Для відповіді на поставлене питання перш за все відзначимо, що таке складальне програмування.

Складальне програмування є одним з методів програмування та підкоряється його загальним закономірностям та представляє одну із форм повторного використання ПС.

Під методом складання розуміється спосіб сполучення різномовних програмних об'єктів у МП, заснований на теорії специфікації й відображення (Mapping) типів та структур даних МП, представлених алгебраїчною системою. Основу алгебраїчного формалізму складають типи даних, операції над ними та функції релевантного, еквівалентного перетворення одних типів у інші.

Результат відображення – оператори релевантного перетворення типів даних у спеціальному модулі-посереднику для кожної пари об'єктів, що сполучаються.

Методом близьким до збірки – є генерація різних об'єктів до одного загального вихідного коду й середовища функціонування. Поняття генерації програм виникло майже одночасно із поняттям збірки та, на

сьогоднішній день, воно отримало новий розвиток у зв'язку із орієнтацією на опис моделі предметної області (домену) засобами мови DSL (Domain Specific Language), що відображає специфіку цієї галузі.

Такий новий напрям ще не має стандартних рішень щодо управління самою проблемою поступової трансформації опису та виконання інструментів генерації, налагодження та інтеграції для отримання кінцевої ПС.

2.2.2 Системна трансформація програмних систем

Розглянута трансформація орієнтована на створення сімейств ПС шляхом перетворення опису членів сімейства специфікаціями високого рівня типу мови DSL з відображенням специфіки предметної області в термінах відповідної базової термінології УП. Результат послідовної трансформації генерується і збирається у вихідний код [172].

Основа трансформації – загальна породжуюча доменна модель GDM (Generative Domain Model) сімейства систем. У ній містяться:

а) опис членів сімейства в мовах (DSL, RSL, CSP, Clear тощо);

б) компоненти реалізації функцій членів сімейства і їх характеристики;

в) знання про конфігурацію (Configuration knowledge) у вигляді правил конструювання, варіантів оптимізації, поєднання і залежності між членами сімейства тощо;

г) модель загальних, змінних характеристик членів сімейства та їх взаємодії;

д) механізми забезпечення мінливості, синхронізації і безпеки як компонентів, так і членів сімейства засобами опису зв'язують аспектів.

Знання про конфігурацію упаковано у вигляді абстракцій загального та спеціального призначення і зберігаються в активній бібліотеці багаторазового використання. Там же зберігаються і виробничі знання про засоби управління тестуванням, вимірюванням, плануванням та оцінкою окремих компонентів і членів сімейства ПС. У наборі цих знань визначаються завдання і підходи до реалізації ПС.

---

172. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. Москва, Сантк-Петербург: Издательский дом Питер. 2005. 730 с.

Кожна модель члена сімейства ПС представляється в деякому конкретному синтаксисі (наприклад: UML, OCL тощо). Ця проєктна модель трансформується від вихідного до цільового подання за наступною схемою (рис. 2.5):

а) трансформація вихідної моделі до проміжного (об'єктного) вигляду;

б) трансформація вихідної моделі у цільову;

в) трансформація цільової моделі в конкретний код ПС.

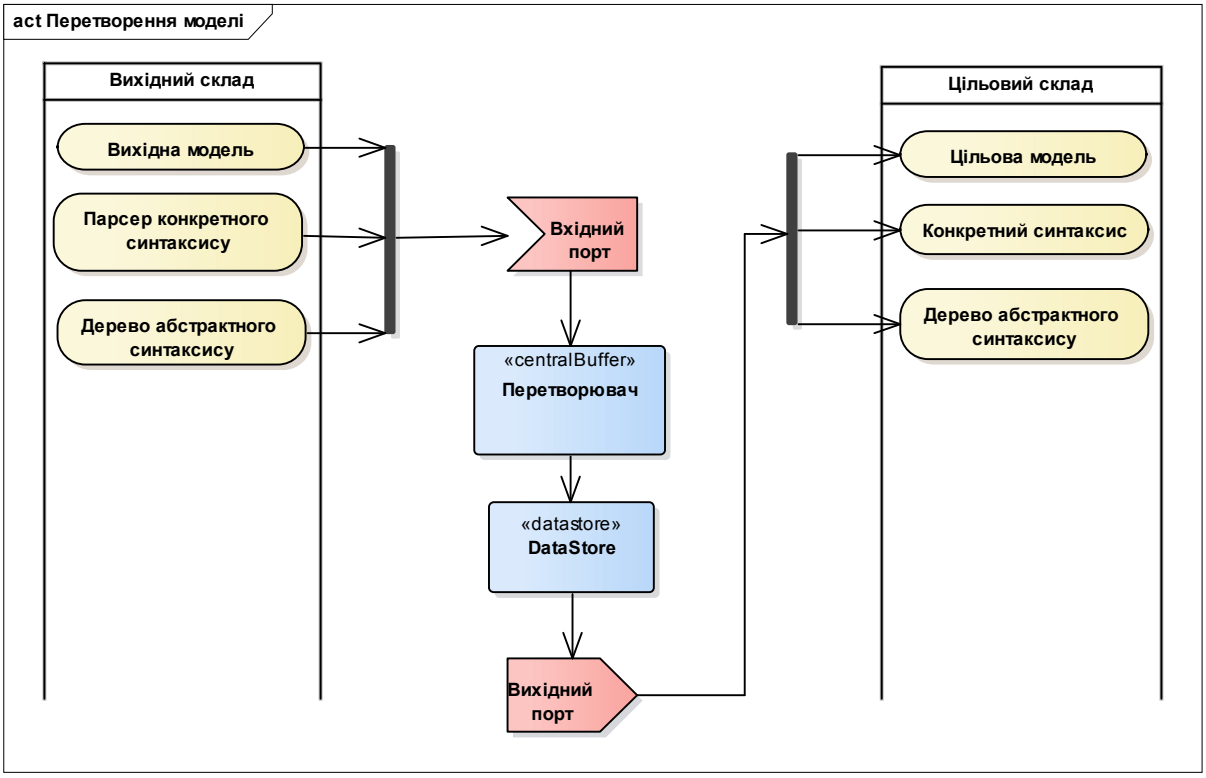


Рисунок 2.5 – Управління трансформацією проєктної моделі ПС від вихідного до цільового представлення

Мета генерації коду – забезпечення можливості його налаштування на цільову платформу для отримання вихідного коду здійснюється в два етапи. Перший етап – забезпечує читання конфігурації, по якій створюється відповідний генератор поточного перетворення, другий – використовує створений генератор для виконання необхідних перетворень моделі.

Інтеграція (збірка) згенерованого коду та того, що створено програмістом, полягає у додаванні програм для незаповнених ділянок шаблону шляхом вставки.

Шаблон – це деякий регулярний вираз концептуальної моделі ПС. Для його опису створюється спеціальна мова шаблону, заснована на вихідній мові програмування та включає у себе критерії пошуку складних збігів у вихідній трансформованій моделі.

Генерація вихідного коду здійснюється такими способами:

а) ітеративним, при якому перераховуються всі вузли вихідної моделі і вони перетворюються в відповідне подання цільової моделі;

б) шаблонним, що дозволяє вставляти макроси, тобто ділянки коду в будь-яку частину заданого шаблону;

в) пошуковим, що складається в знаходженні місця у вихідній моделі та її перетворення.

Наступне управляння інтеграцією виконуваного коду ПС здійснюється за допомогою:

а) використання шаблонів проєктування;

б) збереження згенерованого і виправленого коду в окремих файлах;

в) дизайну архітектури ПС, в якій визначені згенеровані артефакти;

г) використання гнучких методик інтеграції згенерованих та базових частин коду.

2.2.3 Комплектуючі методи складального програмування

Механізм складання будь-яких виробів характеризується: комплектуючими деталями і вузлами, схемою складання (взаємозв'язками окремих компонентів і правилами взаємодії), складальним конвеєром (технологією складання). Конкретизуємо ці поняття з точки зору методу складального програмування стосовно до розвитку ПС. При цьому будемо припускати, що використовуються тільки готові ПП [173].

Комплектуючими в методі складального програмування є прості програмні елементи: модулі, об'єкти, компоненти, сервіси та ін. (табл. 2.1).

З них збираються програмні структури різної складності і ступеня готовності: програми, комплекси, пакети прикладних програм, ПС тощо.

Результатом складання може бути будь-який програмний об'єкт, за винятком елементарного елемента: модуля, компонента, сервісу тощо. Однак для практичного застосування вибирається декілька базових типів

---

173. Лаврищева Е. М. Методы программирования. Теория, инженерия, практика. Киев: Наук. думка, 2006. 451 с.

програмних об'єктів, які розглядаються як готові компоненти та результат використання методу складального програмування.

Таблиця 2.1 – Комплектуючі методи складального програмування

Програмний елемент	Засіб	Механізм	Подання, зберігання	Результат
Процедура, підпрограма, функція	Ідентифікатор	Безпосереднє звернення, оператор виклику	Бібліотеки підпрограм та функцій	Програма
Модуль	Паспорт модуля зв'язку	Виклик модулів, інтеграція модулів	Банк, бібліотеки модулів	Програма з модульною структурою
Об'єкт	Опис класу	Створення екземплярів класів, виклик методів	Бібліотеки класів	Об'єктно-орієнтована програма
Компонент	Опис бізнес-логіки, інтерфейсів (APL, IDL), схеми розгортки	Віддалений виклик у компонентних моделях (COM, CORBA, OSF тощо)	Репозитарій компонентів, сервери і контейнери компонентів	Розподілений компонентно-орієнтований додаток
Сервіс	Опис бізнес-логіки та сервісу (XML, WSDL тощо)	Віддалений виклик (RPS, HTTP, SOAR тощо)	Індексація та каталогізація сервісів (XML, UDDI)	Розподілений сервісно-орієнтований додаток

Для технологічності механізму складання – всі об'єкти повинні мати паспорти, що містять дані, які необхідні для інформаційного сполучення та організації спільного функціонування в рамках програмного об'єкта більш складної структури. Важлива умова складання полягає у наявності великої кількості різноманітних комплектуючих, які забезпечують вирішення широкого спектру завдань з різних предметних областей [174].

174. Любченко В. В. Інформаційна технологія розробки та аналізу моделей предметних областей для їх вивчення. *Проблеми програмування.* 2012. № 2–3. С. 315–321.

2.2.4 Методи інтеграції програмних комплексів

Поряд із розглянутими методами, сформувалися і методи інтеграційного типу, які використовують готові компоненти в процесі створення ПС шляхом комплексування, інтеграції, композиції тощо. До них відносяться сучасні підходи управління інтеграцією, що зведено у табл. 2.3.

Таблиця 2.3 – Підходи до управління інтеграцією ПС

Підхід	Об’єкт	Модель	Метод	Інтерфейс	Середовище	Результат
Об’єктно-орієнтований	Об’єкт, клас, інтерфейс	Об’єктна модель	Інтеграції, взаємодії	IDL, RPC, RMI, C++, Visual C++, Java	CORBA, COM, RR, MS.NET, MSF	Об’єктно-орієнтоване ПЗ, додаток
Компонентний	Компонент, каркас, контейнер, додаток	Компонентна модель, модель додатка	Інтеграції, композиції, взаємодії	IDL, RPC, PDL, JNI, RMI, DOM. Платформо-орієнтований інтерфейс, трансформація в C++ брокер	CORBA, COM, Widows API, API Viewer, MatLab, RMI	Компонентна система
Породжуючий	Об'єкт, клас, компонент, каркас, контейнер, додаток	Породжуюча доменна модель, об'єктна чи компонентна модель	Генерація, взаємодія, інтеграція, збірка, вбудовування	IDL, XML, RSL, DSL, MX, UML, повідомлення	Windows API, API Viewer, Demral, DLL, IDE Eclipse, RR	Додаток, родина ПС

2.2.4.1 Об'єктно-орієнтований підхід

Об'єктно-орієнтований підхід (ООП) включає в себе стратегію розвитку, в рамках якої розробники системи замість операцій і функцій оперують об'єктами. Багато сучасних МП (C++, C#, Java, Python, PHP, Delphi тощо) включили у себе поняття класу та інші принципи ООП. ПС в ООП – це взаємодіючі об'єкти, що мають власний локальний стан і набір операцій для визначення станів об'єктів. Об'єкти можуть інкапсулювати інформацію про подання станів і обмежують до них доступ. Проєктування ПС полягає в проєктуванні класів об'єктів і взаємовідносин між ними.

У такому разі, проактивний розвиток ПС виконується за допомогою наступних механізмів:

а) аналіз – створення об'єктної моделі (ОМ) ПС, у якій об'єкти відображають реальні її сутності та операції над ними;

б) проєктування – уточнення ОМ з урахуванням опису вимог для реалізації конкретних завдань системи;

в) програмування – реалізація ОМ засобами C++, Java тощо;

д) супровід – використання, внесення змін, як до складу об'єктів, так і в методи їх реалізації;

е) модифікація – зміна системи в процесі її супроводу шляхом додавання нових функціональних можливостей, інтерфейсів та операцій.

Наведені процеси можуть виконуватися ітераційно один за одним і з поверненням до попереднього. На кожному з них може використовуватися одна і та ж система нотацій. Перехід до наступного процесу призводить до вдосконалення результатів попереднього процесу шляхом більш детальної реалізації раніше визначених класів об'єктів і додавання нових.

Результат процесу аналізу ЖЦ – модель ПС та набір інших моделей (модель архітектури, модель оточення та використання), отриманих на процесах ЖЦ. Вони відображають зв'язки між об'єктами, їх стану та набір операцій для динамічної зміни стану інших об'єктів, а також взаємини з середовищем. Об'єкти інкапсулюють інформацію про їх стан і обмежують до них доступ.

Модель оточення і використання системи – це дві взаємно доповнюючі один одного моделі зв'язку системи із середовищем. Модель оточення системи – це статична модель, яка описує інші системи з простору розробляється ПС, а модель використання системи – це динамічна модель, яка визначає взаємодію системи зі своїм середовищем.

Взаємодія визначається послідовністю запитів до сервісів об'єктів і реакцією системи після виконання запиту. Коли взаємодія між об'єктами ПС та її оточенням визначені, ці дані використовуються для розробки архітектури системи, яка може розроблятися за допомогою об'єктів, створених у попередніх проєктах.

Основні засоби системної архітектури такі:

а) статична модель описує структуру системи в термінах класів, об'єктів і взаємин між ними (узагальнення, розширення, використання та ін.);

б) динамічна модель являє динамічну структуру системи та взаємодії між об'єктами системи у процесі виконання.

Результат – це удосконалена ПС, в якій всі необхідні об'єкти визначені статично (за допомогою класів) або динамічно – відповідних методів їх реалізації. Отримана об'єктно-орієнтована система перевіряється на показники якості на основі результатів тестування та збору даних про помилки. Таку систему можна розглядати як сукупність автономних і незалежних об'єктів. Зміна методу реалізації об'єкта або додавання нових функцій не впливає на інші об'єкти системи, вони можуть бути такими, що повторно використовуються.

2.2.4.2 Компонентно-орієнтована розробка

Компонентно-орієнтована розробка (Component-Based Development – CBD). Це самостійна методологія, поява якої поряд з модульною технологією, пов'язана з тенденціями в програмуванні [175], [176]. Нижчеподані тенденції ґрунтуються на матеріалах досліджень [177], а також уніфікації програмних компонентів, їх вибору й побудови з них складних ПП.

Отже, особливість компонентної розробки – застосування КПВ на всіх процесах ЖЦ. Іншими словами: всі елементи і об'єкти компонентної розробки – це модулі, компоненти, програми, дані тощо, що подаються програмними КПВ.

Використання КПВ у розвитку проєкту ПС має низку переваг:

а) зниження витрат на розробку ПС;

б) скорочення часу розробки окремих компонентів;

в) поліпшення показників якості та надійності програм з компонентів;

д) спрощення процедур та зниження витрат на підтримку й супровід ПП;

---

175. Лаврищева Е. М. Интерфейс в программировании. *Проблеми програмування.* 2007. № 2. С. 126–139.

176. Лаврищева Е. М. Методы программирования. Теория, инженерия, практика. Киев: Наукова думка, 2006. 451с.

177. Лаврищева Е. М., Грищенко В. Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. Киев: Наукова думка, 2009. 372 с.

е) створення певної інфраструктури в розробці ПС, заснованої на уніфікації та стандартизації компонентів з метою практичного застосування;

ж) створення ринкових компонентів за рахунок їх стандартизації і орієнтації на повторне використання.

З формальної точки зору програмний компонент (component) – це незалежний від МП самостійно реалізований програмний об'єкт, доступ до якого можливий лише за допомогою інтерфейсів, що визначають його функції і порядок звернення до його операціях. У компонентній розробці забезпечений механізм взаємодії компонентів через контракт, в якому визначені правила взаємодії з іншими компонентами, що входять до контейнеру або каркасу.

Інтеграція (збірка) компонентів і їх розгортання з середовища не залежать від ЖЦ розробки ПС. Заміна будь-якого компонента новим компонентом не повинна призводити до перекомпіляції або перенастроювання зв'язків у ПС. Так як інтерфейс не надає реалізацію операцій, то можна змінювати реалізацію без зміни інтерфейсу і таким чином покращувати виконавчі або функціональні властивості компонента без перебудови ПС в цілому, а також додавати нові інтерфейси без зміни існуючої реалізації ПС.

Інтеграція компонентів у архітектуру цільової ПС в CASE складається з декількох типових методів об'єднання, серед яких найбільшого поширення набули патерни, каркаси та контейнери [178], [179].

Патерн – абстракція, яка містить опис взаємодій сукупності об'єктів, ролей учасників і їх відповідальності. Він практично визначає повторюване рішення в проблемі об'єднання КПВ в програмну структуру. Для кожного об'єднання визначається взаємодія об'єктів при спільної кооперативної діяльності із завданням абстрактних учасників, їх ролей, розподілу повноважень (або обов'язків).

Крім патернів, в складальної стратегії широко використовується каркас – типова повторно виникаюча ситуація на рівні моделі ПС, яка визначає структуру проєкту і має невизначені елементи, позначені

178. Андон П. І., Суслов В. Ю., Коротун Т. М., Коваль Г. І. та ін. Визначення витрат на створення ПЗ автоматизованих систем. *Проблемы программирования.* 1998. № 3. С. 23–34.

179. Баховец О. Б., Грінченко Т. О., Гуляєв К. Д. та ін. Передумови становлення інформаційного суспільства в Україні. Київ: Азимут України, 2008. 287 с.

порожніми слотами для розташування в них нових певних компонентів. В цьому сенсі каркас стає КПВ з властивістю екземплярізації і представлений високорівневу абстракцію проєкту ПС, в якій відокремлені функції компонентів від завдань управління ними. В бізнес-функціях каркас задає надійне управління ними. Каркас об'єднує множина взаємодіючих між собою об'єктів у деяке інтегроване середовище, що призначено для рішення заданої кінцевої мети типу «білий» або «чорний ящик».

Каркас типу «білий ящик» включає в себе абстрактні класи для уявлення мети об'єкта і його інтерфейсів. При реалізації ці класи успадковуються в конкретні класи із зазначенням відповідних методів реалізації, як це прийнято в ООП. Каркас типу «чорний ящик» характеризується тим, що у видимій, інформаційної, частини мається опис точок входу і виходу. Через ці точки можна входити і виходити з компонента не обов'язково через кінцевий оператор.

Таким чином, каркас визначає контекст інтеграції окремо збудованих програмних частин. Фізично він реалізується за допомогою одного або більше патернів, які в свою чергу виступають в ролі інструкцій по реалізації проєктних рішень. Створення компонентної системи починається з побудови компонентної моделі, що включає в себе проєктні рішення по композиції програмних компонентів, різні типи патернів, зв'язки між ними, способи взаємодії (інтерфейси) та операції управління ПС у середовищі функціонування. Самі програмні компоненти у даній моделі можуть бути виготовлені за допомогою будь-якої методології УП з розробки ПС, але необхідно враховувати вимоги та умови її застосування при подальшому проактивному розвитку ПС.

2.3 Методологія наскрізного проєктування

Сучасні методи управління проєктуванням припускають застосування CAD/CAM систем, що підтримують, так званий, «наскрізний» процес розвитку ПС. Проте поширена точка зору, що «наскрізний» процес розвитку пов'язаний з абсолютними знаннями складних побудов, створенням математичних моделей і завданням технологічних параметрів, що вимагають додаткових навичок і досвіду роботи в тій чи іншій ПС.

У [20] на прикладі розвитку CAD/CAM ADEM показана реалізація принципів «наскрізного» проєктування та підготовки керуючих програм (КП) для різного устаткування. Розглянемо наскрізний процес розвитку проєкту на двох прикладах: плаский ескіз (креслення) та 2.5D обробка; тривимірна модель та 3D обробка. На кожному прикладі простежимо послідовність дій зі створення моделі розвитку.

2.3.1 Проєктний опис операцій із пласкою моделлю

Розглянемо процес УП плаского ескізу деталі «зірочка» та створення управляючої програми. Етапи розглянемо за відповідними процесами.

Пласке моделювання модуля ADEMCAD призначене для проєктно-конструкторських та креслярських робіт. Результатом його роботи є пласка геометрична модель, яка може бути виведена на друк як креслярсько-конструкторської документації, або використана у роботі інших модулів системи.

Використовуючи методи пласких побудов у системі ADEMCAD створимо пласку модель зірочки. Для створення управляючої програми виготовлення даної деталі досить мати вид згори. Створимо лише один із зубців зірочки, а для створення контуру зірочки застосуємо операцію «Кутове копіювання» до створеного елементу (рис. 2.6).

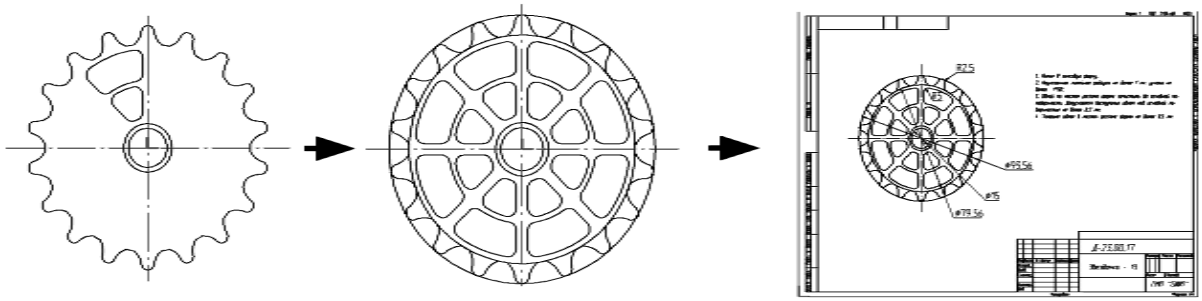


Рисунок 2.6 – Процес створення проєктних ескізу та креслення

Далі створимо контур вікон та застосуємо до них аналогічну операцію. Отриманий проєктний ескіз можемо використовувати для створення креслення деталі, при оформленні технологічного процесу, а також для створення управляючої програми.

Для створення управляючої програми перейдемо до модуля ADEMNC, що є модулем підготовки управляючих програм для різних

видів технологічного обладнання. Результатом роботи буде готова програма.

Модулі ADEMCAD та ADEMNC працюють у одному вікні. При запуску модуля ADEMNC геометрія, створена в ADEM2D, вже відома. Панелі модуля ADEMCAD замінюються на панелі ADEM NC. Усі режими й установки модуля ADEMCAD (одиниці вимірювання, масштаб, формат аркуша тощо) у ADEMNC залишаються за замовчуванням. Функціональні клавіші виконують аналогічні операції. Наведемо у проєкті маршрут обробки деталі. В даному випадку маршрут обробки складається із шести технологічних об'єктів (рис. 2.7).

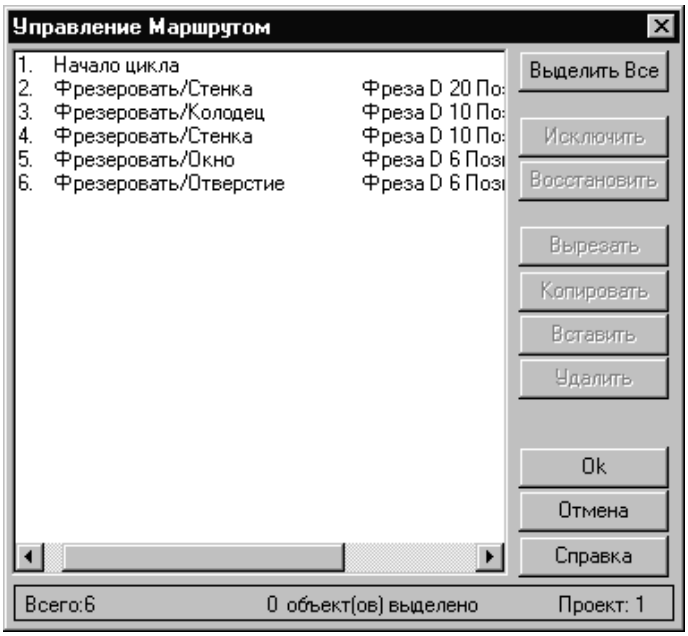


Рисунок 2.7 – Проєктний діалог «Управління маршрутом»

Після завдання проєктного маршруту обробки, розрахуємо траєкторію руху інструменту. Для цього виконаємо команду «Процесор». На екрані з'явиться траєкторія руху усіх інструментів.

Для відстеження помилок (зрізання тощо) змоделюємо процес обробки до виходу на верстат. Моделювання обробки може проходити, як у пласкому, так і у об'ємному режимі. Для об'ємного моделювання може служити модель, що підготовлена у форматі STL (рис. 2.8).

Для того, щоб отримати безпосередньо управляючу програму у кодах обладнання, слід вибрати відповідний постпроцесор та виконати команду «Адаптер». Результатом виконаних дій буде файл із послідовністю команд.

Асоціативність геометричної та технологічної моделі – одна з найважливіших особливостей CAD/CAM ADEM. Властивість асоціативності полягає в тому, що при внесенні змін до геометричної моделі, тобто її розвитку, не потрібно заново задавати процес обробки – необхідно тільки перерахувати траєкторію руху інструменту. Після створення маршруту обробки, моделювання процесу обробки, скорегуємо отримані результати, із урахуванням допущених помилок, а також зміни у геометричній моделі.

Рисунок 2.8 – Проєктування траєкторії руху та моделювання обробки

2.3.2 Проєктний опис операцій з об'ємною моделлю

Процес створення об'ємної моделі та управляючої програми для трикоординатного обладнання розглянемо на прикладі проєктування пресформи для виготовлення виробу «викрутка».

Робота зі створення тривимірної моделі виробу починається зі створення контурів (в якості контурів можуть бути використані елементи креслення).

Етапи створення об’ємної моделі виробу (рис. 2.9):

– використовуючи команду «Зсув із заданою висотою» зміщуємо зовнішній контур плити, яка є базовим тілом для створення об'ємної моделі пресформи;

– за допомогою команди «Обертання» побудуємо тіло обертання, яке описує ручку викрутки та за допомогою якої створимо активну частину пресформи;

– за допомогою булевої операції «Віднімання» видаляємо обсяг тіла обертання із плити;

– використовуючи команди «Зсув із заданою висотою», «Округлення» та «Віднімання» побудуємо поверхні, що описують виступи на ручці викрутки (рис. 2.10).

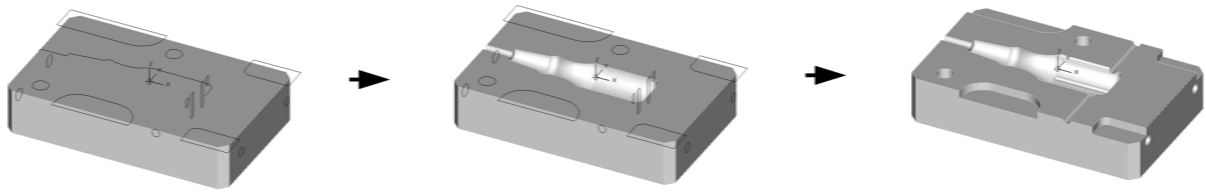


Рисунок 2.9 – Етапи створення об’ємної моделі виробу

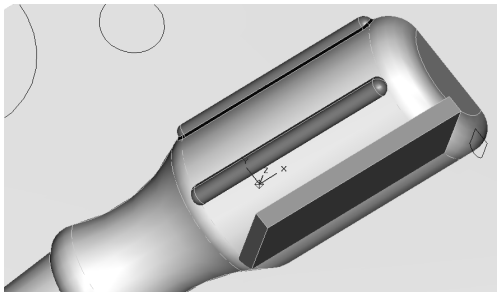


Рисунок 2.10 – Створення виступів

Моделювання активної частини прес-форми повністю завершено. Далі у проєкті: створення низки допоміжних елементів (уступи, направляючі отвори, кріпильні отвори тощо). Згодом перейдемо до створення управляючої програми для виготовлення даної пресформи.

При описі маршруту обробки можна використовувати об'ємний конструктивний елемент «Поверхня», але це далеко не завжди технологічно правильно, тому у ADEM реалізована можливість комбінації пласких та об'ємних конструктивних елементів у одному маршруті обробки.

Після завдання маршруту обробки, аналогічно діям у першому прикладі, розрахуємо траєкторію руху інструменту. При цьому ми отримаємо об'ємну траєкторію руху (у даному випадку 3 координати) та управляючу програму трикоординатного обладнання.

Зміни у геометрії об'ємної моделі, як і у пласкому прикладі, тягнуть за собою зміни у технологічній обробці даної деталі. На рис. 2.11 можна побачити розвиток моделювання етапів обробки прес-форми для виготовлення викрутки.

Таким чином, розглянуто два приклади, що ілюструють застосування сучасних засобів управління виробництвом, які дозволяють прискорити роботу із розвитку «наскрізного» проєктування та управляючих програм. Наведені етапи проєктування показують: наскільки легко та логічно, на відміну від традиційної постановки задачі, може відбуватися процес «наскрізного» проєктування. Традиційний «окремий» розгляд завдань конструктора та виробництва, на сьогодні, не може гарантувати ані високої якості розвитку проєктів, ані належного рівня організації управління виробничими процесами, що забезпечують скорочення виробничого циклу із досягненням висунутих вимог щодо якості продукції.

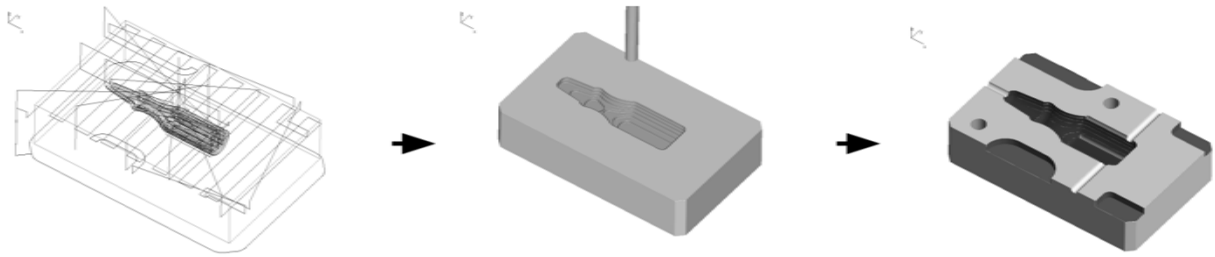


Рисунок 2.11 – Розвиток моделювання етапів 3D-обробки

Проблеми, що виникли, можуть бути успішно вирішено тільки за допомогою інтегрованого розгляду питань УП разом із питаннями підготовки виробничого циклу, які бездоганно підтримуються ПС ADEM. Крім того, методологія «наскрізного» проєктування не тільки забезпечує можливість розвитку спеціальних технологічних знань, а й одночасно виконує управління розвитком відповідних компетенцій персоналу.

2.4 Висновки за розділом

У поданому розділі монографії виконано встановлене завдання систематизації провідних методологій УП у концепцію проактивного УП з розвитку життєвого циклу (ЖЦ) ПС та ПП. Однією з головних проблем проактивного УП з розвитку ПС є створення теоретичних і прикладних основ швидкої та якісної побудови складних систем з більш простих програмних елементів, які виконано у сучасних МП. Фактично рішення цієї проблеми здійснюється шляхом збирання, об’єднання або інтеграції різнорідних програмних ресурсів та КПВ, що включають модулі, бібліотеки та програмні реалізації деякої складної ПС.

Створення та проактивний розвиток ПС та ПП, як проєкту, повинно здійснюватись провідною проєктною організацією з обов'язковим залученням інших стейкхолдерів проєкту, у тому числі науково-дослідних інститутів, профільних закладів вищої освіти, а також відомих учених, науковий потенціал і накопичений досвід яких, виражається в сотнях наукових публікацій і десятках захищених дисертацій під їх керівництвом.

У поданому розділі розроблено методологічні основи УП з проактивного розвитку ПС та ПП. Дістала подальшого розвитку систематизація провідних методологій УП, яка розширена формуванням концепції проактивного УП з розвитку ЖЦ ПС та ПП, що дозволяє на етапі оцінки проєкту визначити доцільність цільової програми реінжинірингу. Заради об’єктивності слід зазначити, що у деяких випадках, все ж таки раціональніше застосувати нову розробку ПС.

Ідею УП складанням ПС пропонували В. М. Глушков та А. П. Єршов. Згодом формування нового виду УП зі складального програмування відбувалося за участю К. Л. Ющенко. Засоби УП складання модулів, програм та компонентів розвиваються й постійно удосконалюються. Базова концепція УП організації складання різних видів і типів програмних об'єктів – це інтерфейс, як сполучна ланка різномовних ПП у будь-яких середовищах – отримала повну формалізацію у МП опису інтерфейсу IDL (Interface Definition Language) і стандарту опису загальних типів даних ISO / IEC 11404-1996, 2007 незалежно від МП.

Науково-технічною підтримкою парадигми УП складального програмування є результати робіт В. Н. Гріщенка та К. М. Лавріщевої. До них відносяться ключові основоположні концепції теорії УП складання з КПВ ПП, інженерії тестування та якості, оцінювання процесів та ПП, а також моделей подання знань про КПВ.

УП реінжинірингу виконується за допомогою комплексу засобів, у тому числі за рахунок застосування КПВ та CASE-систем. Згідно з оптимістичними прогнозами застосування КПВ здатне у 4 рази знизити вартість ПС у порівнянні з новою розробкою.

Також у розділі виконано наступне встановлене завдання формалізації методів проактивного УП розвитку наскрізного проєктування ПС, які відрізняються формуванням парадигми системної трансформації, що дозволяє інтегрувати ПП у оновлені проєктні комплекси.

Згідно із загальною методологією УП, задача проактивного розвитку полягає не тільки у з’ясуванні шляхів, а у визначенні характеру конкретної

перебудови процесів проєктування ПС. У той же час, ПС об’єднує у собі різні види забезпечення: технічне, математичне, програмне, інформаційне, лінгвістичне, методичне, організаційне, ергономічне та правове. Кожний з цих видів забезпечення має параметри та бажані характеристики управління, які обирають проєктувальники із максимальним урахуванням особливостей завдань інженерного проєктування, конструювання, технологій кодування, виготовлення та інтеграції ПС.

Рішення цієї проблеми здійснюється шляхом УП збирання, об'єднання або інтеграції різнорідних програмних ресурсів та КПВ, що включають модулі, бібліотеки та програмні реалізації проєкту деякої складної ПС.

До перспектив дослідження вже після прийняття рішення щодо застосування проєкту реінжинірингу, слід віднести наукові задачі розробки логічної схеми управління реінжинірингом, що є похідною від логічної схеми проєктування, яка починається цільовою програмою та включає в себе вибір компонентів УП реінжинірингу. Ця логічна схема представляє цільовий засіб отримання нової ПС шляхом виконання послідовності операцій внесення змін, модернізації або модифікації, а також перепрограмування окремих її компонентів. Логічна схема УП реінжинірингу повинна починатися цільовою програмою, об’єднуючи зовнішню та внутрішню частини, відповідно до формування цілей УП.

Основні власні наукові дослідження, подані автором у другому розділі монографії, опубліковано у наукових працях автора: [16 – 18], [20], [21], [26], [35], [45], [46], [51] – [54], [58], [59], [62], [65], [70] – [73].

3 МОДЕЛІ ПРОАКТИВНОГО УПРАВЛІННЯ РОЗВИТКОМ ЛІНГВІСТИЧНОГО ТА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ПРОЄКТІВ ПРОГРАМНИХ СИСТЕМ

Основою будь-якого суспільства є інформація. Без обміну та аналізу інформацією неможливий не тільки розвиток живого організму, але й його існування. Існує багато засобів передачі інформації між істотами, здебільшого за допомогою органів відчуттів. Щодо людини, то одним із основних засобів передачі інформації є мова, причому не тільки її звукова компонента (адже інформація може передаватись на писемною мовою, мовою жестів, шрифтом Брайля тощо).

Сучасною основою прискорення управлінням інформацією між людством є інформаційні технології, технічне забезпечення яких складається із багатьох різноманітних пристроїв, що, по суті, є модифікаціями комп'ютера. Ці пристрої також обмінюються інформацією у вигляді даних (не тільки між собою, але й з людиною – користувачем або оператором), якими необхідно правильно управляти. В залежності від рівнів представлення даних, інформація може подаватися у такому вигляді: двійковий, вісімковий, десятковий, шістнадцятковий коди; машинний код; низько- та високорівневі мови програмування тощо.

3.1 Динамічне моделювання лінгвістичного забезпечення проєктів за допомогою породжувальних граматик

Наукову основу будь-якої мови (в тому числі й мови програмування) становить лінгвістика, що вивчає закони, моделі та правила мови. Особливою галуззю лінгвістики, яку варто застосовувати до будови проєктів ПС, є генеративна лінгвістика, засновником якої був Аврам Ноам Чомські (Avram Noam Chomsky, у радянські часи зустрічалася інтерпретація «А. Н. Хомський»), який створив революцію у мовознавстві.

За способом завдання правильних ланцюжків розвитку формальні граматики поділяють на породжувальні та розпізнавальні. До породжувальних відносяться ті граматики, за допомогою яких можна побудувати будь-який правильний ланцюжок з вказівкою його структури і не можна побудувати жодного неправильного ланцюжка. Вперше, поняття породжувальної (генеративної) граматики запропонував А. Н. Чомські. Розпізнавальна граматика – це граматика, що дозволяє встановити

правильність довільно обраного ланцюжка та, у разі якщо він правильний, з’ясувати його будову. До формальних мов відносяться зокрема штучні мови для спілкування оператора з комп’ютером.

Лінгвістичне забезпечення проєктів ПС розглядає УП побудови ПС за допомогою однієї або декількох (узгоджених між собою) МП, кожна з яких ґрунтується на правилах конкретної граматики. До інформаційного забезпечення проєктів ПС віднесено питання інформаційної узгодженості між собою різноманітних банків та БД, у тому числі й графічних, які об’єднані використанням загальної бази знань проєкту та є обов’язковими для використання у ПС.

3.1.1 Методи управління проєктом реінжинірингу програмних систем в динамічному оточенні

Проєкти з інтеграції програмних структур спочатку виконувалася за допомогою готових підпрограм бібліотек різного призначення шляхом їх вставки до ПС, що інтегруються [180]. Згодом, з'являлися різні методи проактивного розвитку ПС, засновані на конкретизуючому, синтезуючому, композиційному програмуванні тощо, які вирішували проблему комплексування програмних об'єктів методами, близькими до складання різнорідних об'єктів.

Розглянемо методи реінжинірингу проєктів з елементами інтеграції, комплексування та синтезу.

Конкретизуючий метод базується на виділенні з деякої універсальної програми окремої її частини, налаштованої на особливі, певні умови виконання. Можна відзначити два типи такого виділення [181]. Перший характеризується формуванням конкретної програми управління та аналогічний процесу макрогенерації. Другий тип пов'язаний із конкретизацією інформаційних структур у ПС, що використовується.

При синтезуючому методі будується динамічна модель програми за специфікацією завдання, за якою буде синтезована програма її вирішення. Специфікація задається у термінах деякої формальної мови [182]. На її

---

180. Resource Estimation for Objectory Projects: project report. Objective Systems. SF AB, 1993. 9 p.

181. Chomsky N. Three Models for the Description of Language. IRE Transactions on Information Theory. Sep., 1956. P. 113–124.

182. Chomsky N. Logical Syntax and Semantics: Their Linguistic Relevance. Language. Vol. 31. No. 1. P. 36–45.

основі та правил побудови алгоритмів опису конкретної предметної області відбувається формування необхідної програми.

Композиційний метод базується на принципах функціональності та композиційності, які розглядають програми як набір функцій, що будуються з інших функцій за допомогою спеціальних операцій, названих композиціями. На основі композиційного уточнення (експлікації – explication) створюється логіко-математична система композиційної побудови програм, яка об'єднує сучасні парадигми програмування (структурне, функціональне, об'єктно-орієнтоване тощо) у рамках єдиної концептуальної, експлікативної платформи.

Складальний метод характеризується побудовою програм із готових «деталей», якими є програмні об'єкти різного ступеня складності. Елементи процесу складання присутні у багатьох методах програмування: згори-униз, знизу-догори тощо.

Архітектори, при розробці ПС без застосування динамічних методів управління, виділяють повторно використовувані оператори та оформляють їх у вигляді окремих, самостійних фрагментів або підпрограм для подальшого використання.

Складальний метод управління реінжинірингом ПП:

а) є одним з методів програмування та підлягає загальним закономірностям;

б) представляє одну із форм повторного використання ПС;

в) якісно відрізняється від процесів складання у інших методах.

Під складальним методом управління розуміється сполучення різномовних програмних об'єктів у МП, який ґрунтується на теорії специфікації й відображення (Mapping) типів і структур даних МП, представлених алгебраїчною системою. Основу алгебраїчного формалізму складають типи даних, операції над ними та функції релевантного, еквівалентного перетворення одних типів на інші [183].

Методом близьким до складального є генерація різних об'єктів до одного загального вихідного коду й середовища функціонування. Поняття генерації програм виникло майже одночасно із поняттям складання та, на сьогоднішній день, воно отримало новий розвиток у зв'язку із орієнтацією на управління моделлю предметної області (домену) засобами мови DSL (Domain Specific Language), що відображає специфіку цієї галузі.

---

183. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра. Языки. Программирование. Киев: Наук. думка, 1974. 328 с.

Такий новий напрям ще не має стандартних рішень щодо самої проблеми управління трансформацією опису МП та виконанням генерації, налагодження та інтеграції для отримання кінцевої ПС.

Таким чином, завдання поданого розділу – систематизація методів УП реінжинірингу ПС у нові програмні структури, ПС та готові інформаційні ресурси, що працюють в умовах динамічного оточення, та накопичені людством за визначений час.

3.1.2 Математичне представлення теоретичних основ управління породжувальними граматиками

Для завдання опису формальної мови проєкту ПС необхідно, по-перше, вказати алфавіт, тобто сукупність об'єктів, що називаються символами (або літерами), кожен з яких можна відтворювати у необмеженій кількості примірників [184], і, по-друге, завдати формальну граматику мови, тобто перерахувати правила, що являють собою набір процесів, за якими з символів вибудовуються їх унікальні послідовності, що належать до визначеної мови.

Будь-яка МП являє собою множину ланцюжків у деякому кінцевому алфавіті. У лінгвістиці замість терміну «алфавіт» використовується термін «словник» тому, що елементи, з яких він складений, являють собою словоформи [185]. У той же час, ланцюжок над словником розглядається як словосполучення або речення.

Зауважимо, що кожен символ алфавіту розглядається як нероздільний у тому сенсі, що при побудові ланцюжків ніколи не використовуються його графічні елементи (частини символів) та будь-яка послідовність символів однозначно являє деякий ланцюжок [186]. Практично ця вимога досягається, наприклад, шляхом встановлення «пробілу» (проміжку стандартної довжини) між символами. Цей «пробіл»

---

184. Горелов С. Евристичний аналіз грамматики. Евристичний аналіз будь-якої мови. URL: http://www.grammcheck.org/ (дата звернення: 13.01.2019).

185. Федоренко О. Ф., Сухорольська С. М., Руда О. В. Основи лінгвістичних досліджень = Fundamentals of Linguistic Research: підруч. для вузів. Львів.: «Центр» Львів. нац. ун-ту ім. І. Франка, 2009. 296 с.

186. Руднев В. П. Генеративная лингвистика. Словарь культуры XX века. URL: http://www.gumer.info/bibliotek_Buks/Culture/Rudnev/Dict/_04.php (дата звернення: 13.01.2019).

перевищує довжину будь-якого з проміжків, що зустрічається всередині символів алфавіту.

Правила формальної граматики необхідно розглядати як «продукції» (правила виходу) – елементарні операції управління, які у разі застосування у визначеній послідовності до вихідного ланцюжка (аксіоми) породжують лише правильні ланцюжки результату. Сама ж послідовність правил, що використовується у процесі управління породженням деякого ланцюжка, є її виведенням. Визначена таким чином мова являє собою унікальну формальну систему. Відомими прикладами формальних систем служать логічні обчислення (висловлювання, предикати), що відносяться до розділів математичної логіки.

Для здійснення управління породжувальною граматикою [187] або, коротко, граматикою задамо впорядкований *набір*:

$$G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle,$$

де $\mathrm{A} = \{a_1, a_2, \ldots, a_m\}$ – основний *термінальний* алфавіт;

Ψ – кінцевий допоміжний (позатермінальний) алфавіт, символи якого позначуються малими грецькими літерами;

ε $(\varepsilon \in \Psi)$ – початковий (позатермінальний) символ;

Z – кінцева система підстановок:

$$\mathrm{Z} = \{u_i \to v_i \mid i = 1, 2, \ldots, k\},$$

де u_i – ланцюжок;

$v_i \in \delta(v)$, де $\delta(v)$ – *вільна напівгрупа* над об'єднаним алфавітом Θ:

$$\Theta = (\mathrm{A} \cup \Psi).$$

Інакше кажучи, символи основного алфавіту A є елементарними одиницями мови, що визначається; символи алфавіту Ψ – *метазмінні*, що використовуються при управлінні правильними ланцюжками (у природних мовах для управління такими метазмінними є граматичні класи: іменник,

187. What is Generative Grammar? wiseGEEK. URL: http://www.wisegeek.com/what-is-generative-grammar.htm (дата звернення: 13.01.2019).

дієслово тощо); ε – *метазмінна аксіома*, з якої вибудовуються усі правильні ланцюжки (у природних мовах аксіомі відповідає граматичний клас «речення»); Z – *схема граматики*, що складається з продукцій (правила управління виведенням – граматичні правила визначеної мови).

Наприклад, породжувальною граматикою є граматика:

$$G_0 = \langle \{a, b, c\}, \{\alpha, \beta, \chi\}, \varepsilon, Z_0 \rangle,$$

причому Z_0 має набір процесів:

$$Z_0 = \begin{cases} \varepsilon \to abc, \\ \varepsilon \to b, \\ \varepsilon \to \alpha\alpha, \\ abc \to c. \end{cases}$$

Визначення мови $L(G)$, завданої породжувальною граматикою G, пов'язане з поняттям «*виведення*».

Нехай x, y – ланцюжки, що належать до вільної напівгрупи $\delta(v)$. Ланцюжок y безпосередньо управляється ланцюжком x у граматиці G:

$$x \underset{G}{\Rightarrow} y \text{ або } x \Rightarrow y \text{ (коли } G \text{ мається на увазі)}, \quad (3.1)$$

якщо у схемі Z поданої граматики знайдеться продукція $u \to v$ така, що

$$\begin{cases} x = x_1 u x_2, \\ y = x_1 v x_2; \end{cases}$$

де $x_1, x_2 \in \delta(v)$. Тобто ланцюжок y отримуємо як результат застосування до ланцюжка x продукції управління $u \to v \in Z$, що значить заміну у ланцюжку x виділеного входження лівої частини u поданої продукції її правою частиною v. Наприклад, у граматиці G_0:

$$b\varepsilon c \Rightarrow b\alpha\alpha c, \quad abcba \Rightarrow cba, \quad \ldots$$

Ланцюжок y виводиться з ланцюжка x у граматиці G, $x \overset{*}{\underset{G}{\Rightarrow}} y$ або $x \overset{*}{\Rightarrow} y$, схоже з (3.1), якщо ланцюжки x, y співпадають або існує послідовність ланцюжків $z_0, z_1, \ldots, z_k$ така, що:

$$z_0 = x, \quad z_k = y \quad \wedge \quad \forall i \left(1 \le i \le k\right) \quad z_{i-1} \Rightarrow z_i .$$

Послідовність ланцюжків $Q = (z_0, z_1, \ldots, z_k)$ має назву «управління виведенням» ланцюжка y з ланцюжка x у граматиці G. Наприклад, у граматиці G_0 $\varepsilon \overset{*}{\Rightarrow} acc$, причому послідовність є управлінням виведенням ланцюжка *acc* з ланцюжка ε:

$$\left\langle \varepsilon \Rightarrow abc;\ abc \Rightarrow a\varepsilon c \,\middle|\, b \to \varepsilon;\ a\varepsilon c \Rightarrow aabcc \,\middle|\, \varepsilon \to abc;\right.$$
$$\left. aabcc \Rightarrow acc \,\middle|\, abc \to c \right\rangle . \quad (3.2)$$

Слід додати, що на кожному кроці управління виведенням можна обрати будь-яку з продукцій, яку можна застосувати у поточний момент. А це означає, що послідовність застосування продукцій у граматиці довільна і будь-яку продукцію дозволено застосовувати після іншої, але у межах системи правил управління.

Таким чином, поняття породжувальної граматики докорінно відрізняється від поняття «нормальний алгоритм», у якому підстановки носять визначений характер і суворо виконуються у завчасно зазначеній послідовності управління.

Виведення $x \overset{*}{\underset{G}{\Rightarrow}} y$ є повним, якщо $y \in \delta(\mathrm{A})$, тобто ланцюжок y складається з термінальних символів. Будь-яке повне виведення закінчується застосуванням продукцій, якщо їхні праві частини являють собою термінальні ланцюжки. Вказане управління продукціями назвемо «*кінцевими результатами продукції*» поданої граматики.

Якщо $x \overset{*}{\underset{G}{\Rightarrow}} y$ та $y \notin \delta(\mathrm{A})$, причому у системі Z не існує правил управління, що застосовуються до ланцюжка y, то виведення ланцюжка y з ланцюжка x у граматиці G називається *тупиковим*. Наприклад:

виведення, що наведене у (3.2) є повним виведенням у граматиці G_0, acc – кінцевий результат продукції управління граматики G_0. А, наприклад, виведення:

$$\langle aabcabc \Rightarrow a\varepsilon abc \,|abc \to \varepsilon; a\varepsilon abc \Rightarrow a\varepsilon c \,|abc \to c;$$
$$a\varepsilon c \Rightarrow a\alpha\alpha c \,|\varepsilon \to \alpha\alpha\rangle$$

є тупиковим виведенням ланцюжка $a\alpha\alpha c$ з ланцюжка $aabcabc$ у граматиці G_0.

Далі розглянемо, як саме породжувальна граматика $G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle$ визначає мову, що відповідає їй. Ланцюжок $x \in \delta(\mathrm{A})$ буде правильним, якщо існує принаймні одне повне виведення ланцюжка з аксіоми ε в граматиці G. Інакше кажучи, ланцюжок x правильний, якщо:

$x \in \delta(\mathrm{A})$ – ланцюжок x складається з термінальних символів;

$\varepsilon \overset{*}{\underset{G}{\Rightarrow}} x$ – існує управління виведенням ланцюжка x з аксіоми ε.

Множина усіх правильних ланцюжків у граматиці G створює мову $L(G)$, що породжується граматикою G. Наприклад, граматика G_0 породжує мову:

$$L(G_0) = \left\{ x^n y x^n \cup y^m x y^m \,|n, m = 0, 1, 2, \ldots \right\}.$$

Отже, кожній граматиці $G = \langle \mathrm{A}, \Psi, \varepsilon, \mathrm{Z} \rangle$ однозначно відповідає мова $L(G)$, породжена цією граматикою. Проте, ця відповідність не ізоморфна: та ж сама мова може породжуватись різноманітними граматиками. Це дозволяє внести відношення еквівалентності на управління множиною граматик.

Граматики G та G' – еквівалентні $(G \Leftrightarrow G')$, якщо $L(G) = L(G')$, тобто граматики G та G' породжують одну мову. Наприклад, граматика:

$$G_1 = \langle \{a, b, c\}, \{\varepsilon\}, \varepsilon, \mathrm{Z}_1 \rangle,$$

схема якої має набір процесів:

$$Z_1 = \begin{cases} \varepsilon \to abc, \\ \varepsilon \to b, \\ \varepsilon \to c, \end{cases}$$

породжує мову $L(G_1)$, яка співпадає із мовою $L(G_0)$ та, відповідно, означає, що $G_0 \Leftrightarrow G_1$.

3.2 Моделі управління проєктом реінжинірингу графічних баз даних

В наш час є велика кількість програмних засобів, які виконують значну кількість спеціалізованих задач. Деякі з них прив'язані лише на одну галузь промисловості, інші – застосовуються у великій кількості галузей, але тенденція йде шляхом спеціалізації ПП у цілому.

Однією з важливих складових частин САП (як окремого класу ПС) є комп'ютерна графіка, що являє собою сукупність засобів та прийомів, за допомогою яких здійснюється управління (введення, перетворення та виведення) спеціалізованими середовищами графічної інформації.

Комп'ютерна графіка – актуальна галузь проєктування та застосування ПС, що інтенсивно розвиваються у останній час. Термін «комп'ютерна графіка» означає управління обчислювальною обробкою інформації, а також виведення результатів у вигляді різних графічних зображень. Дані, необхідні для управління відображенням результатів у графічному форматі, створюються на підставі графічної інформації. Особливий інтерес до комп'ютерної графіки став проявлятись у зв'язку з інтенсивним розробленням та впровадженням у даний час САП не тільки у машинобудуванні, приладобудуванні, радіоелектроніці, дизайні інтер'єрів, але і в інших галузях виробництва та навчання.

Відмінною складовою завдань комп'ютерної графіки є управління графічними базами даних (ГБД), які, по суті являють собою «звичайні» БД, але в основу управління якими закладено математичні алгоритми відновлення зображення за сформованими статистичними координаційними даними. Такі можливості є далеко не у кожній САП, але сучасні тенденції вимагають цього. Велика кількість ПП розроблюється з широким спектром моделюючих характеристик, BRL-CAD – це один з таких ПП.

Мета поданого підрозділу полягає в управлінні (створенні, підключенні, еволюційному удосконаленні тощо) проєктом реінжинірингу ГБД, як композиційного компоненту відкритого ПП BRL-CAD.

3.2.1 Відкритий проєкт BRL-CAD

BRL-CAD – це спеціалізована крос-платформова ПС з відкритим кодом. Вона являє собою потужну 3D САП для моделювання об`ємних тіл методами Constructive Solid Geometry (CSG). Ця ПС включає в себе інтерактивний геометричний редактор, паралельне трасування променів, рендеринг та геометричний аналіз.

BRL-CAD розроблялася близька 40 років та набула застосування у збройних силах США. Продукт працює із вихідного коду, а тому його можна використовувати на будь-яких платформах: GNU/Linux, MacOS, Solaris та Windows. Наведемо визначальні характеристики, що стосуються відкритого ПП та технологій проєктування ГБД.

Сирцевий код (зазвичай просто «сирці», також «вихідний код», «програмний код», «джерельний код», «текст програми», англ.: «source code») – будь-який набір інструкцій або оголошень, написаних мовою програмування і у формі, що її може прочитати людина. Сирцевий код дозволяє програмісту спілкуватися з комп'ютером за допомогою обмеженого набору інструкцій.

Сирцевий код програми – це набір файлів, потрібних для перетворення з форми, доступної для читання людині, на деякі види комп'ютерного виконуваного коду. Можливі два напрямки виконання сирцевого коду: транслювання у машинний код за допомогою компілятора (призначений для певної комп'ютерної архітектури) або виконання безпосередньо з тексту за допомогою інтерпретатора.

Відкрите ПЗ (англ.: open-source software) – це забезпечення, для якого є доступним (вихідний) програмний код, що забезпечує найкращі умови для вивчення такого ПЗ та можливого подальшого внесення змін (удосконалень тощо) до нього [114].

Досить часто останнє поняття вважають тотожнім вільному ПЗ, що не є абсолютно правильним. Найістотніша відмінність полягає в тому, що ліцензії на вільне ПЗ обумовлюють, що усі подальші модифіковані версії такого ПЗ теж повинні розповсюджуватись як вільні, в той час як більшість ліцензій на ПЗ з відкритими кодами надають повну свободу

авторам модифікованих версій. В результаті вільне ПЗ завжди є ПЗ з відкритими вихідними кодами, але зворотне є вірним далеко не завжди.

Одна з перших ПС з характеристиками відкритого проєкту з'явилась тому, що у 1979 році балістична науково-дослідна лабораторія армії США (U. S. Army Ballistic Research Laboratory (BRL), зараз – United States Army Research Laboratory, висловила гостру потребу в інструментах та засобах, які могли б управляти комп'ютерним моделюванням та інженерним аналізом бойових систем озброєння (танків, ракет, літаків тощо) та їх умовами роботи.

Коли жодна з ПС, які існували на той час, виявились неготовими для досягнення цієї мети, розробники з BRL почали систематизовувати набір утиліт, що здатні на управління інтерактивним переглядянням та редагуванням дерев геометричних моделей. Архітекори приступили до розробки власного пакету додатків, що призначені для відображення, редагування та суміщення геометричних моделей. Результатом став створений проєкт BRL-CAD – пакет додатків для управління твердотільним моделюванням.

Перший публічний реліз був зроблений у 1984 р. У грудні 2004 р. BRL-CAD став проєктом із відкритим кодом. Важливо, що проєкт BRL-CAD впроваджується на умовах ліцензій *BSD та GNU LGPL.

З того часу проєкт постійно розвивається, з'являються нові можливості, проте зараз вже само лінгвістичне забезпечення подання ГБД (мова «С») у ПП BRL-CAD потребує операцій переходу (реінжинірингу) на високорівневі мови («С++» чи «С#»). Сьогодні, завдяки приблизно мільйону рядків С-коду, відкритий проєкт став найпотужнішим пакетом графічного моделювання, що набув застосування більш ніж у 2 тис. організацій по всьому світу.

Проєкт BRL-CAD підтримує одночасно два способи взаємодії з користувачем: за допомогою командного рядка та графічного інтерфейсу користувача (GUI). Також ПП підтримує управління різноманітними геометричними засобами роботи з графічною інформацією: великий набір традиційних CSG-примітивних твердих речовин (еліпсоїди, конуси, тори), а також явні тверді (із закритих колекцій) уніформи, β-сплайнові поверхні, нерівномірні раціональні β-сплайни (NURBS), n-різноманітну геометрію (НРГ), грановані сітки тощо. Всі геометричні об'єкти можуть бути об'єднані із використанням логічних теоретико-множинних операцій, включно із CSG-об'єднаннями та перетинами.

Найактуальніша властивість проєкту полягає в здатності конструювати та аналізувати реалістичні моделі на основі складних об'єктів, які складаються з великого набору графічних примітивів (primitive shapes). Для управління складними об'єктами використовуються булеві операції: об'єднання, віднімання та перетину. Ще один потужний бік ПП BRL-CAD – швидкість засобів управління візуалізацією та трасувальником променів, який є одним із найшвидших серед існуючих. Нарешті, користувачі BRL-CAD можуть проєктувати моделі із великою точністю, від субатомних до галактичних масштабів за принципом «бачимо всі деталі увесь час».

3.2.2 Моделі управління розвитком поведінкової частини проєкту

Проєктування об'єктів машинобудування, промислового, цивільного будівництва та радіоелектроніки вступає в новий етап свого розвитку, коли разом зі зростанням складності проєктів мають забезпечуватися скорочення термінів проєктування й зменшення числа проєктувальників, значною мірою за рахунок моделювання управління проєктуванням та комп'ютеризації інженерної праці.

Об’єктом моделювання стане управління поведінковою частиною ГБД відкритого проєкту BRL-CAD. Виконане моделювання важливе, оскільки дозволяє удосконалити відкритий проєкт BRL-CAD шляхом високорівневого мовного оновлення.

3.2.2.1 Опис процесу управління ГБД в загальному вигляді

Розв’язання проблеми неможливе без глибокого проникнення в фізичну сутність досліджуваних явищ, розробки та вдосконалення відповідних теоретичних положень, впровадження досягнутих результатів у виробництво. Геометричні методи давно та успішно використовуються в багатьох галузях промисловості. Велику роль тут мають відіграти нові методи геометричного моделювання та їх реалізація в системах комп'ютерної графіки, що дозволить розв'язувати задачі управління ГБД.

Графічна інформація – це найбільш ємне і наочне уявлення великого обсягу інформації, однак, практичне застосування машинної графіки

довгий час стримувалось відсутністю відповідного обладнання та математичного забезпечення для його управління.

Логічність та формалізованість комп'ютерних моделей управління ГБД дозволяють виявити основні фактори, що визначають властивості досліджуваного об'єкта-оригіналу (або цілого класу об'єктів), зокрема, досліджувати відгук фізичної системи, що моделюється на зміни параметрів управління та початкових умов.

Побудова комп'ютерної моделі управління ГБД базується на абстрагуванні від конкретної природи явищ або досліджуваного об'єкта-оригіналу і складається з двох етапів: спочатку створення якісної, а потім і кількісної моделі. Само комп'ютерне моделювання полягає у проведенні серії експериментів на ПК, метою яких є аналіз, інтерпретація та зіставлення результатів моделювання з реальною поведінкою об'єкта моделювання та, за необхідністю, подальше уточнення моделі.

Одним з головних переваг тривимірного моделювання є швидке управління кресленнями. Використовувати результати моделювання можна і на подальших стадіях ЖЦ продукту – це є ще однією перевагою твердотільного моделювання [188].

Задачам створення програмних комплексів на базі відкритого вихідного коду, мовами «С» та «С++», що цікавлять з погляду реінжинірингу ГБД присвячено публікацію [189].

Для наочності сприйняття аналітиками та системними архітекторами, представимо моделі УП реінжинірингу ГБД відкритого ПП BRL-CAD із використанням методології UML із розширеною нотацією 2.5.

3.2.2.2 Модель варіантів використання програмного продукту

При побудові моделі ВВ (МВВ) розглянемо проєкт BRL-CAD. Побудована МВВ для ПП BRL-CAD наведена на рис. 3.1. Специфікація МВВ наведена у табл. 1.

---

188. Потемкин А. В. Трехмерное твердотельное моделирование. Москва: Компьютер-Пресс, 2002. 296 с.

189. Жуковський В. В. Про деякі підходи до створення програмних комплексів комп’ютерного моделювання підземних процесів. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. 2017. Вип. 2 (103). Ч. 1. С. 64–73.

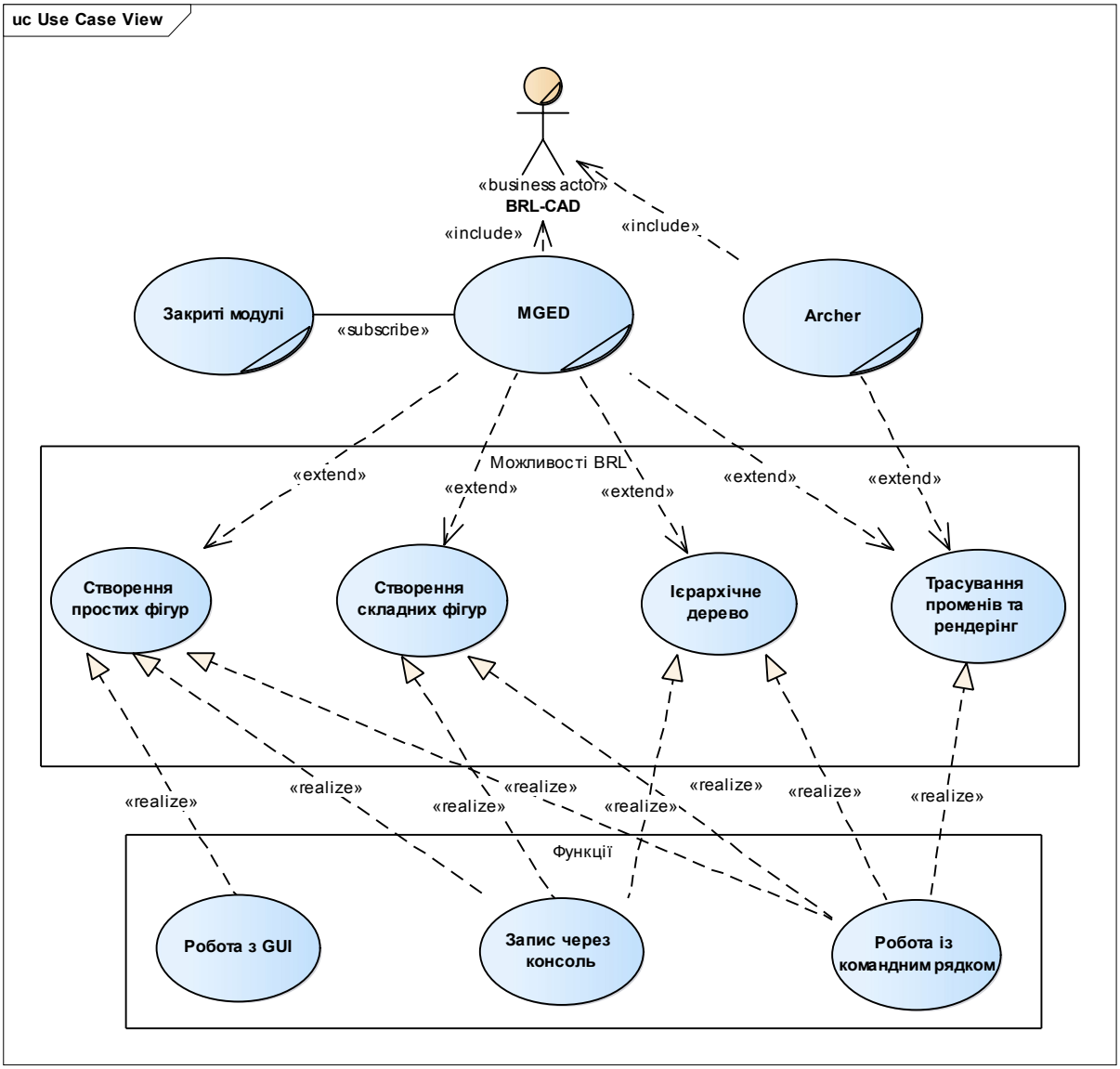


Рисунок 3.1 – Модель ВВ для проєкту BRL-CAD

Таблиця 3.1 – Специфікація МВВ

Тип	Різновид	Кількість, шт.
сутності	бізнес-актор (business actor)	1
	бізнес-прецеденти (business use case)	3
	варіанти використання (use case)	7
	границі використання (boundary)	2
зв’язки	включення (include)	2
	підключення (subscribe)	1
	розширення (extend)	5
	реалізація (realize)	8

Проєкт BRL-CAD складається з 2-х основних модулів – бізнес-прецедентів (business use case) «Archer» та «MGED». У зв'язку із тим, що відштовхуватися будемо від того, що реінжиніринг відбувається із повним списком модулів, то за бізнес-актора (business actor) візьмемо сам «BRL-CAD» – це відразу вказується у проєктних специфікаціях, тобто: бізнес-актор – це актор (стейкхолдер), який безпосередньо задіяний у роботі з клієнтом. Модуль «MGED» – це бізнес-прецедент (business use case) є основним для цього ПП. У ньому відбувається створення твердотільних моделей та управління трасуванням променів. Так саме і модуль «Archer», мають у собі стереотип бізнес типу (business use case) тому, що за допомогою цих двох модулів саме й виконується управління створенням ГБД (рис. 3.1).

Першим рівнем прецедентів є прецеденти з умовною назвою «Можливості», що відокремлені відповідною границею використання проєкту (boundary). Кожен стереотип першого рівня реалізований в якомусь вигляді і завдання полягає в тому, щоб правильно показати цей вигляд. При запуску «MGED.exe» відображаються два інтерфейси управління: консольний (console) і графічний (GUI). У кожному з цих вікон знаходиться панель команд, з якої виконуються команди управління трасуванням.

Основні можливості (рис. 3.1) або прецеденти першого рівня (use case) зведено у табл. 3.2. Всі інші модулі підключаються тільки за допомогою модулів з сайту програми, але їх немає у вільному доступі. Прецеденти другого рівня – це функції проєкту (рис. 3.1), що відокремлені відповідною границею використання (boundary), які реалізують функціональні можливості (табл. 3.2).

Таблиця 3.2 – Ієрархія проєктних прецедентів у ПП BRL-CAD

Прецеденти:	
першого рівня (можливості):	другого рівня (функції):
«Створення простих фігур»	«Робота з GUI» (графічний інтерфейс)
«Створення складних фігур»	«Запис через консоль» (Console)
«Ієрархічне дерево»	«Робота із командним рядком» (Command Line)
«Трасування променів та рендеринг»	

Через консоль можлива реалізація простих фігур, створення з них складних і подальша робота з ієрархічним деревом. Прості фігури задаються за допомогою команд, можливо вказати їх чітке місце розташування. Складні фігури так само задаються за допомогою команд, але за деяких умов: мінімум дві прості фігури відображено на екрані та ці фігури стикаються. Після цього можливий перегляд ієрархічного дерева і робота з ним.

У проєкті можливості управління графічним інтерфейсом досить обмежені: можливо тільки розміщення простих фігур так би мовити «на око», зміни кута перегляду і можливість переглянути результат трасування променів.

Усіма функціями проєкту, які було описано в перших двох прецедентах, так само можливо управляти через командний рядок, але він створювався не для цього. Основною його функцією є підготовка проєктної ієрархії до рендерингу, управління трасуванням променів та загальною обробкою графіки. У налаштуваннях командного рядка можна управляти щільністю тіла, кутом променів, що падають на фігуру, та більше ніж 10-ма константами, які використовуються у класичному твердотільному моделюванні.

Результати роботи командного рядка відображаються у графічному інтерфейсі. Так саме до «MGED» підходить один прецедент (зв'язок-підключення (subscribe)) – «Закриті модулі», який теж має стереотип бізнес-прецеденти (business use case). Це спеціальні модулі для розширення можливостей управління, які не можна отримати у вільному режимі, тому не будемо їх детально розглядати.

Крім «MGED» у проєкті існує також модуль «Archer», який створений для більш зручної роботи з графічним відображенням та переглядом твердотільної моделі. Від нього йде тільки один прецедент (use case) з пакету «Можливості» – це «Трасування променів та рендеринг». Також у «Archer» знаходиться графічний інтерфейс, консоль та командний рядок, які є прецедентами з пакету «Функції».

Далі опишемо зв'язки. Почнемо із гори (рис. 3.1): «BRL-CAD» – це загальна назва ПП, а модулі «Archer» та «MGED» є її складовими, тому між ними присутній зв'язок типу «включення» (include). Всі прецеденти першого рівня – це можливості, вони є розширенням роботи модулів і тому між ними стоїть тип зв'язку «розширення» (extend). Між першим і другим рівнем прецедентів, можливостями і функціями може знаходиться тільки

один рівень зв'язку – «реалізація» (realize), оскільки функції реалізують можливості.

У ході виконання поданого пункту – була розроблена МВВ для ПП BRL-CAD, що є концептуальною основою для виконання реінжинірингу ГБД.

3.2.2.3 Модель опису станів для проактивного управління проєктом

Розробимо модель опису станів (МОС) для проактивного УП реінжинірингу ГБД у середовищі ПП BRL-CAD (рис. 3.2). На МОС відображаються стани, в яких може знаходитись ГБД ПП. Основним логічним ланцюжком цієї моделі є те, що існує «Початкова точка» (initial state) та «Кінцева точка» (final state), які символізують початок та завершення роботи з управління середовищем.

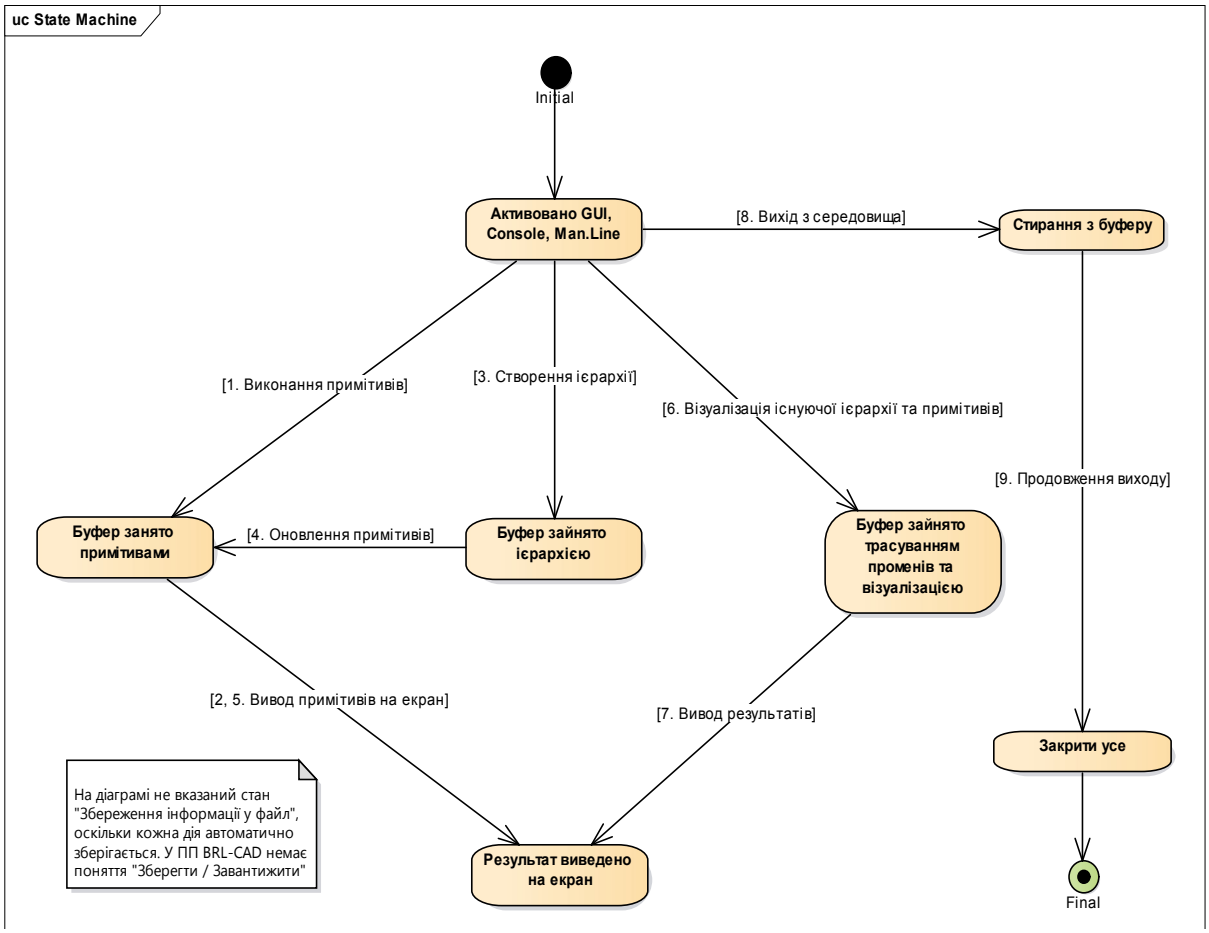


Рисунок 3.2 – Модель опису станів для проактивного УП реінжинірингу ГБД у середовищі ПП BRL-CAD

Відразу додамо, що початком роботи з ПП буде його «активація», тобто запуск, а закінчення роботи – це деактивація або закриття продукту.

При активації роботи запускається буфер і компілятор. Буфер – це стек для команд обробки фігур, робота з якими відбувається у даний момент. Компілятор – це спеціальний модуль, що займається обробкою уводів команд та побудовою графічного результату на екрані після обробки цих команд.

Якщо дотримуватися стандартної методології виконання проєктних операцій: першою дією буде створення примітиву при застосуванні можливості вивантажити у оновлений проєкт сформовані раніше ГБД.

Для цього з консолі уводиться команда управління створенням певного виду примітиву: його розмір, положення (якщо воно не зазначено, то примітив створюється на початку координат). Далі відбувається наступний процес: буфер отримує певну управляючу інформацію, яку зберігає до тих пір, поки її обробляє компілятор. Після того як компілятор обробив усю інформацію за поточною командою, при правильному результаті він видає графічні дані на екран і записує їх у постійну пам'ять. Коли створено кілька примітивних фігур, приходить час будувати ієрархію з них, тобто створювати більш складні фігури.

Умовами створення більш складних фігур, як було вказано раніше є їх кількість (більше двох) та зіткнення (перетини). Як тільки команда про побудову складної фігури надходить до буферу, відбувається обробка компілятором, але, тепер відбувається не тільки виведення на екран результату, а й заміна та оновлення даних з постійної пам'яті. Буфер після завершення команд автоматично очищується.

Наступним проєктним кроком після створення складних фігур є їх візуалізація. Для цього необхідно виконати управління трасуванням променів: або з консолі, або з командного рядка задати фактуру (з чого складається об'єкт) та ще десяток стандартних фізичних констант, після чого – зайти до спеціальної панелі командного рядка та виконати трасування. Процедура – аналогічна: «буфер-компілятор-відображення». Єдиною відмінністю буде те, що результат не зберігається у постійну пам'ять, а тільки відображається на екрані. Для запису візуалізації використовується інший модуль під назвою «Archer». На візуалізації завершується робота з даним ПП і тому наступною дією є «деактивація» (final) (рис. 3.2).

Слід особливо виділити той факт, що у ПП BRL-CAD є деякі особливості, які дозволяють виділяти його серед інших. У ПП немає такого поняття як «Зберегти проєкт / Завантажити проєкт». Будь-яка дія оброблюється компілятором та автоматично зберігається у постійну пам'ять. У такому підході є і як переваги так і недоліки. Переваги: при випадках екстреного закриття середовища вся інформація зберігається. Недоліки: якщо зроблена помилка, то виправляти її доводиться за допомогою видалення всієї спроєктованої фігури.

Після побудови МОС та виходячи з аналізу її змісту, можна зробити проміжні, але досить несподівані висновки, а саме: оскільки проєкт BRL-CAD, що згодом набув статусу відкритого, створювався для потреб американських збройних сил, то випливає припущення, що їх уподобанням була саме така оригінальна реалізація принципів управління ГБД.

3.2.2.4 Модель послідовності розвитку графічного представлення проєкту

Модель послідовності розвитку (МПР) спроєктуємо для того, щоб показати, які послідовні дії виконує програміст при роботі із середовищем BRL-CAD при виконанні розвитку графічного представлення проєкту. На МПР згори до низу розставляються повідомлення (рис. 3.3), кожне може бути позначено відповідним ім'ям [190]. За бажанням на МПР можна показати аргументи та деяку керуючу інформацію, також можна показати рефлексивні повідомлення, які об'єкт посилає самому собі, при цьому стрілка повідомлення вказує на ту ж саму лінію життя.

Виконаємо системний опису процесів, що представлені у МПР. Об'єкт або учасник (Object, Participant) позначається прямокутником, у якому зазначається інформація про учасника дій. Це, як правило, назва об'єкта та його клас, розділені двокрапкою. Наприклад: «saveButton» або «saveButton: JButton» або «: JButton». Тобто, назву класу можна опустити або навпаки – не вказувати назву об'єкта, але щось одне з двох (об'єкт або клас) необхідно вказати.

---

190. Lavagno L., Martin G., Selic B.V. UML for Real: Design of Embedded Real-Time. Systems Kluwer Academic Publishers, 2003. 388 p. ISBN: 1402075014, 9781402075018.

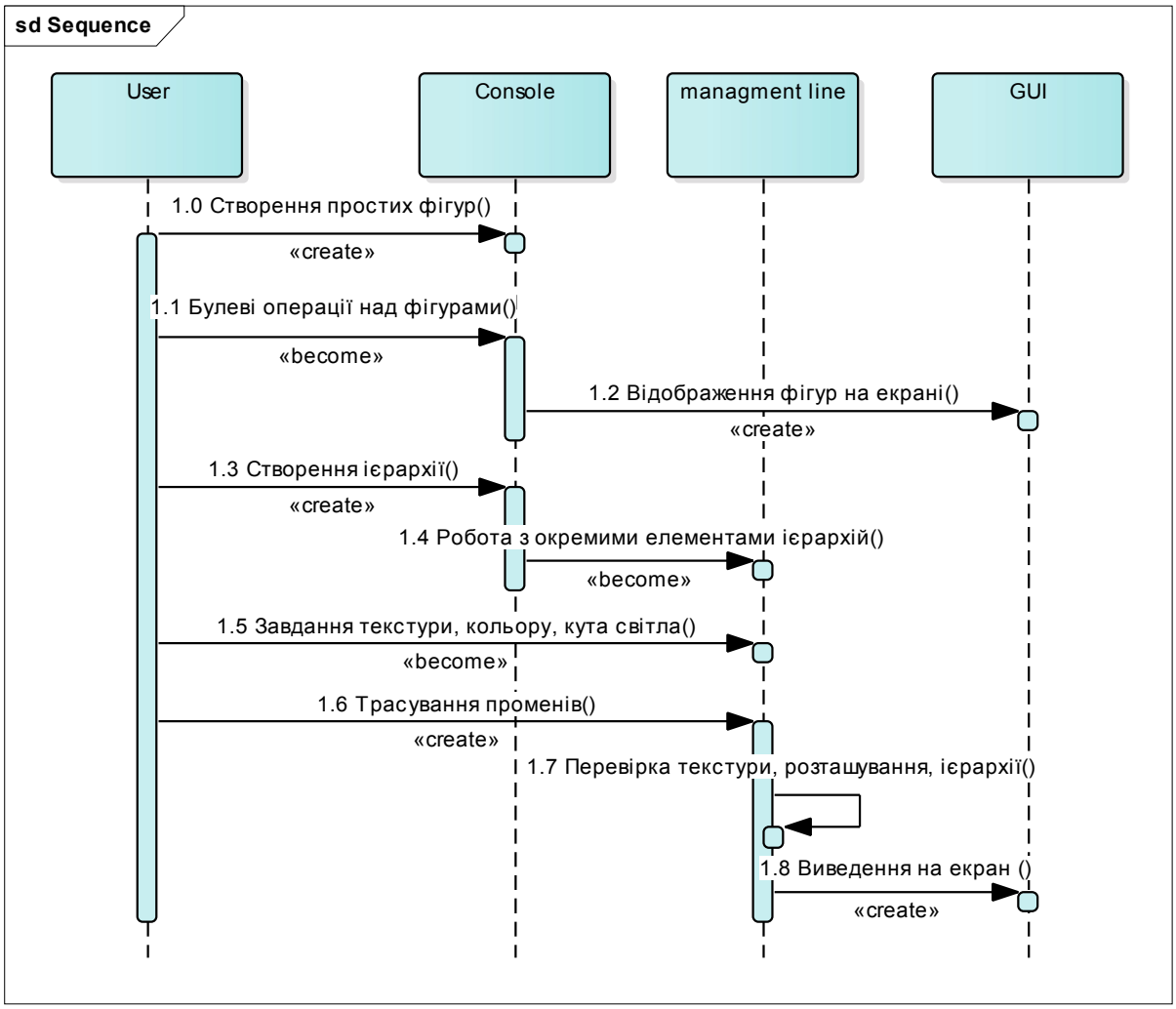


Рисунок 3.3 – Модель послідовності розвитку графічного представлення проєкту у середовищі BRL-CAD

Розташовуються об'єкти (як правило) вздовж верхнього краю діаграми. Від прямокутника вниз опускається «Лінія ЖЦ» (life line) – відрізок, який позначає відведений об'єкту час ЖЦ, позначається вона пунктирною лінією.

У розробленій МПР, стикаємося із розглянутими раніше модулями інтерфейсу: користувач «User»; консоль «Consol»; командний рядок «Management line»; графічний інтерфейс «GUI». У деяких випадках, лінії життя обривають (для цього існує спеціальний знак), що означає закінчення ЖЦ об’єкту. В нашому випадку – це не потрібно.

Активація або фрагмент виконання (Activation Bar, Execution Occurrences) позначається вузьким прямокутником, що розташовується на лінії ЖЦ (рис. 3.3). Він вказує на початок та завершення дії, у якій бере

участь об'єкт. Оскільки лінія ЖЦ – це метафора часу, то прямокутник на лінії ЖЦ вказує на активізацію об'єкта продовж відносного часу.

Повідомлення інформаційної підтримки (Message) – це стрілка від однієї лінії ЖЦ до іншої, що показує взаємодію об'єктів. Повідомлення інформаційної підтримки бувають різні й відрізняються [191]. Синхронне повідомлення позначається зафарбованої стрілкою, асинхронне – штриховою. Повернення показується пунктирною стрілкою, у зворотному напрямку.

У наведеній МПР (рис. 3.3), спроєктовані фундаментальні кроки реінжинірингу, які необхідно пройти при роботі із ГБД. Оскільки спроєктована МПР багатою мірою розповідає сама за себе у описаній методології послідовності, то не будемо описувати її у відповідному підпункті, що б не викликати дублювання проілюстрованої інформації. Отже, у виконаному підпункті було розроблено МПР графічного представлення проєкту у середовищі BRL-CAD, яка містить у собі поведінкову сценарну послідовність реінжинірингу ГБД.

3.2.2.5 Динамічна модель діяльності реінжинірингу графічної структури проєкту

Динамічна модель діяльності (ДМД) – це спеціальна діаграма, на якій показано декомпозицію реінжинірингу графічної структури проєкту на складові частини робіт. Під діяльністю (англ.: activity) розуміється специфікація поведінки, що виконується, у вигляді координованого послідовного чи паралельного виконання підлеглих елементів [192]: вкладених видів діяльності та окремих дій (англ.: action), що з'єднані між собою потоками інформаційної підтримки, які йдуть від виходу одного вузла до входу іншого [193].

У нашому випадку спроєктовано ДМД для розвитку структури ГБД у ПП BRL-CAD, яку представлено на рис. 3.4. Інформаційні сутності, що містить ДМД, зведено у проєктну специфікацію (табл. 3.3).

---

191. Miles R., Hamilton K. Learning UML 2.0. O'Reilly Media, 2006. 288 p.

192. Иванов Д. Ю., Новиков Ф. А. Унифицированный язык моделирования UML. Учебное пособие. Санкт-Петербург: Изд-во Политехн. ун-та, 2010. 249 с.

193. Hay D. C. UML and Data Modeling: A Reconciliation Technics Publications, 2011. 242 p.

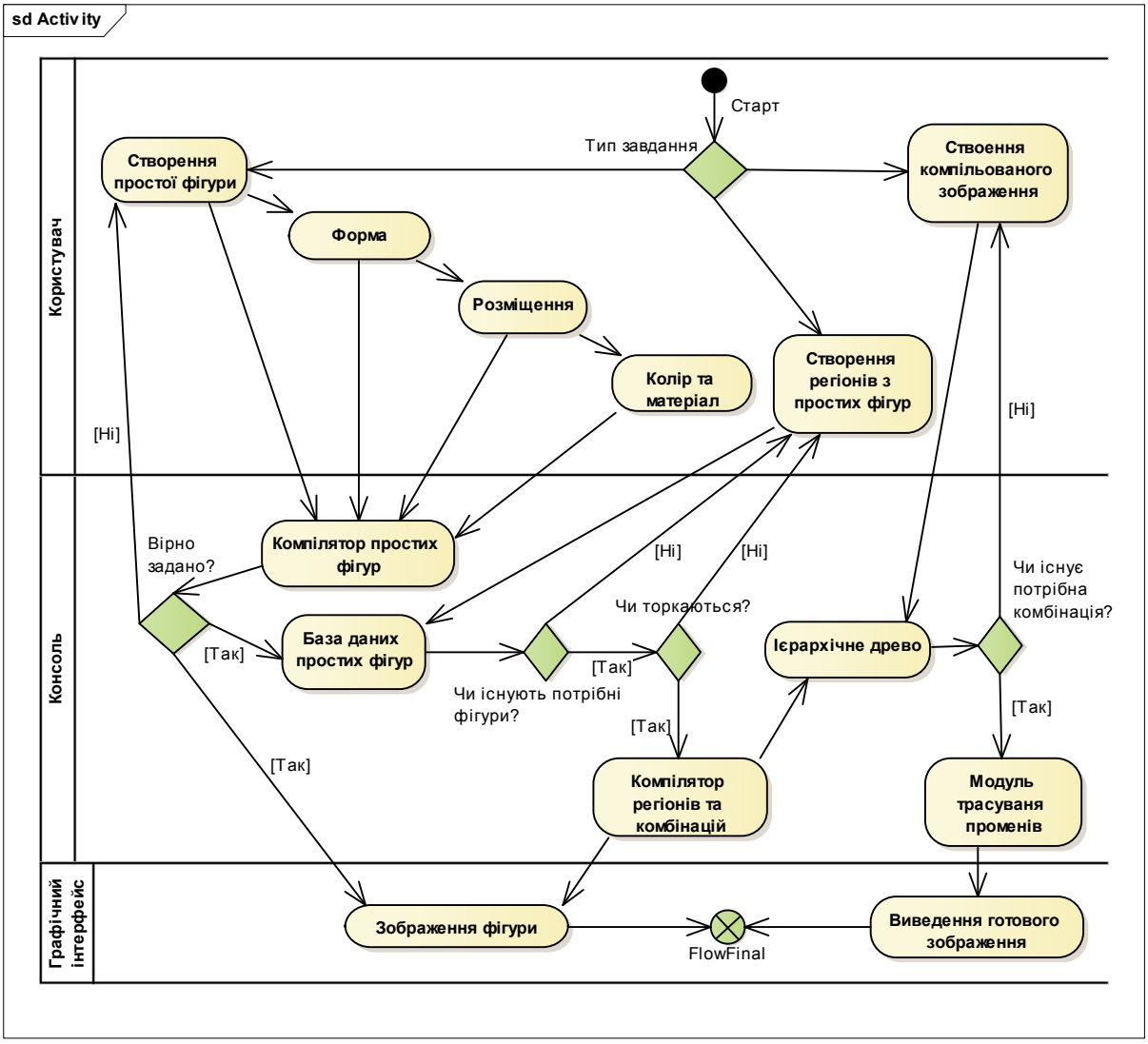


Рисунок 3.4 – Динамічна модель діяльності реінжинірингу графічної структури проєкту у середовищі BRL-CAD

Таблиця 3.3 – Проєктна специфікація сутностей динамічної моделі діяльності

Різновид сутності	Кількість, шт.
початкова активність (initial)	1
блоки-вирішення (decision)	5
стани активності (activity)	13
кінцева активність (flow final)	1

У поданій ДМД, крім минулих задіяних, створено нові проєктні сутності-діяльності (activity), що розміщено на доріжках розділення

проєктної діяльності (partition) або ДРПД, що занесені до проєктної специфікації (табл. 3.4).

Таблиця 3.4 – Специфікація сутностей, що розміщено на доріжках розділення проєктної діяльності

Назва ДРПД	Проєктна діяльність
«Користувач»	«Старт» (початкова активність)
	«Тип завдання» (блок-вирішення)
	«Створення компільованого зображення»
	«Створення простої фігури»
	«Форма»
	«Розміщення»
	«Колір та матеріал»
	«Створення регіонів з простих фігур»
«Консоль»	«Компілятор простих фігур»
	«База даних простих фігур»
	«Вірно задано?» (блок-вирішення)
	«Чи існують потрібні фігури?» (блок-вирішення)
	«Чи торкаються?» (блок-вирішення)
	«Чи існує потрібна комбінація?» (блок-вирішення)
	«Ієрархічне дерево»
	«Модуль трасування променів»
	«Компілятор регіонів та комбінацій»
«Графічний інтерфейс»	«Зображення фігури»
	«Виведення готового зображення»;
	Кінець потоку активності (flow final).

Опишемо структуру потоків управління інформацією у ДМД (рис. 3.4).

За базовий візьмемо звичайний початок процесу – створення простої фігури. У наведеній динамічній модельній нотації, щоб створити просту фігуру, потрібно не тільки ввести команду назви, але й вказати «Форму», «Розміщення», «Колір та матеріал», як спроєктовано у ДМД.

Після того, як указано дані параметри: вся інформація управління передається до компілятору простих фігур (на рівні фізичного модуля коду, компілятор один, але для наочності розділимо його на підрівні).

Особливість цієї моделі – це блоки-вирішення (decision), які позначують перевірку виконання умови, і, у разі виконання цієї умови, проводять операції далі; в іншому випадку – повертає туди, куди вказує стрілка негативного потоку управління.

Отже, після того як інформація надходить до компілятору, відбувається саме така перевірка умови. Припустимо, що умови не виконуються, і тоді на екран видається напис за типом «Shape is not create» (фігуру – не створено) і відбувається відсилання до початку управляючої команди або переривається поточна. У разі якщо перевірку пройдено успішно, то виконується відразу дві основних дії: графічне відображення та надходження управляючої інформації до основного файлу, з усіма даними проєкту.

Наступною операцією, після створення пари простих фігур, є створення складних фігур («регіонів»). При спробі створення регіонів відбувається процес перевірки умови, яка необхідна для створення складних фігур. Якщо всі вони виконані, то виконується вже не дві, а три основних дії управління: запис до пам'яті, відображення на екрані та, найголовніше для регіонів, – оновлення ієрархічного дерева внесення складних фігур.

Далі, після створення складних ієрархічних фігур, можна виконувати трасування зображення. У розглянутому прикладі все відбувається у програмному модулі «MGED», оскільки на даному етапі реінжинірингу цікавить найшвидший, а не презентаційний варіант.

Управління трасуванням променів також відбувається після перевірки виконання деяких умов. Хоч можна трасувати й примітиви, але це не практикується у силу марності такої дії, оскільки крім складових фігур, потрібна перевірка за фізичними константами. Для правильного управління трасуванням необхідно вказати: кут падіння світла, рівень світла, що пропускається, матеріал з чого складається фігура тощо. Коли всі проєктні параметри зазначено вірно: отримаємо цілком придатне для використання зображення (складова ГБД проєкту).

Отже, у даному підпункті було розроблено ДМД реінжинірингу графічної структури проєкту у ПП BRL-CAD. Ця модель дозволяє управляти: станами, шляхами та циклами проходження процесу реінжинірингу ГБД.

3.2.3 Моделі управління розвитком структурної частини проєкту

Предметом моделювання є структурна частина проєкту з реінжинірингу ГБД. Результатом стане побудований проєктний каркас (системна архітектура), на основі якого виконується моделювання структурної частини (моделі: об’єктів, класів, компонентів та розгортання), що всебічно характеризує процес реінжинірингу ГБД для відкритого проєкту BRL-CAD.

Оскільки ретельну постановку задачі реінжинірингу ГБД було вже виконано, перейдемо відразу до моделювання управління розвитком структурної частини. При проєктуванні архітектури використовується розширена проєктна нотація UML 2.5 та CASE-інструментарій Enterprise Architect 14.0.

3.2.3.1 Модель опису оточення об’єктів

Модель опису оточення об’єктів (МООО) показує те, як абстрактні функціональні пристрої та ПЗ працюють один з одним (встановлюють комунікаційні сполучення між об’єктами) та їхні можливості.

У спроєктованій узагальненій МООО використовуються п’ять типів об’єктів, які зображено на рис. 3.5, специфікацію проєктного оточення яких наведено у табл. 3.5.

Опишемо проєктну топологію МООО (рис. 3.5). При відкритті ПП відразу ж починають роботу перераховані вище об'єкти. Тут стрілки позначають системні повідомлення, обмін якими здійснюється у рамках даного варіанту оточення. Їх часова послідовність – зазначається шляхом нумерації повідомлень. У МООО можна використовувати один з декількох варіантів нумерації. У проєктній методології UML 2.5 застосовується десяткова схема нумерації, оскільки у цьому випадку зрозуміло: яка операція викликає яку, хоча може здаватися, що важче розгледіти загальну послідовність

Нумерація системних повідомлень робить сприйняття послідовності більш важким, ніж у випадку розташування ліній на сторінці згори до низу. З іншого боку, таке просторове розташування дозволяє більш легко відбити інші моменти, наприклад, можна показати взаємозв'язок об'єктів, що перетинаються, компоненти або іншу інформацію, яка стосується проєктної топології.

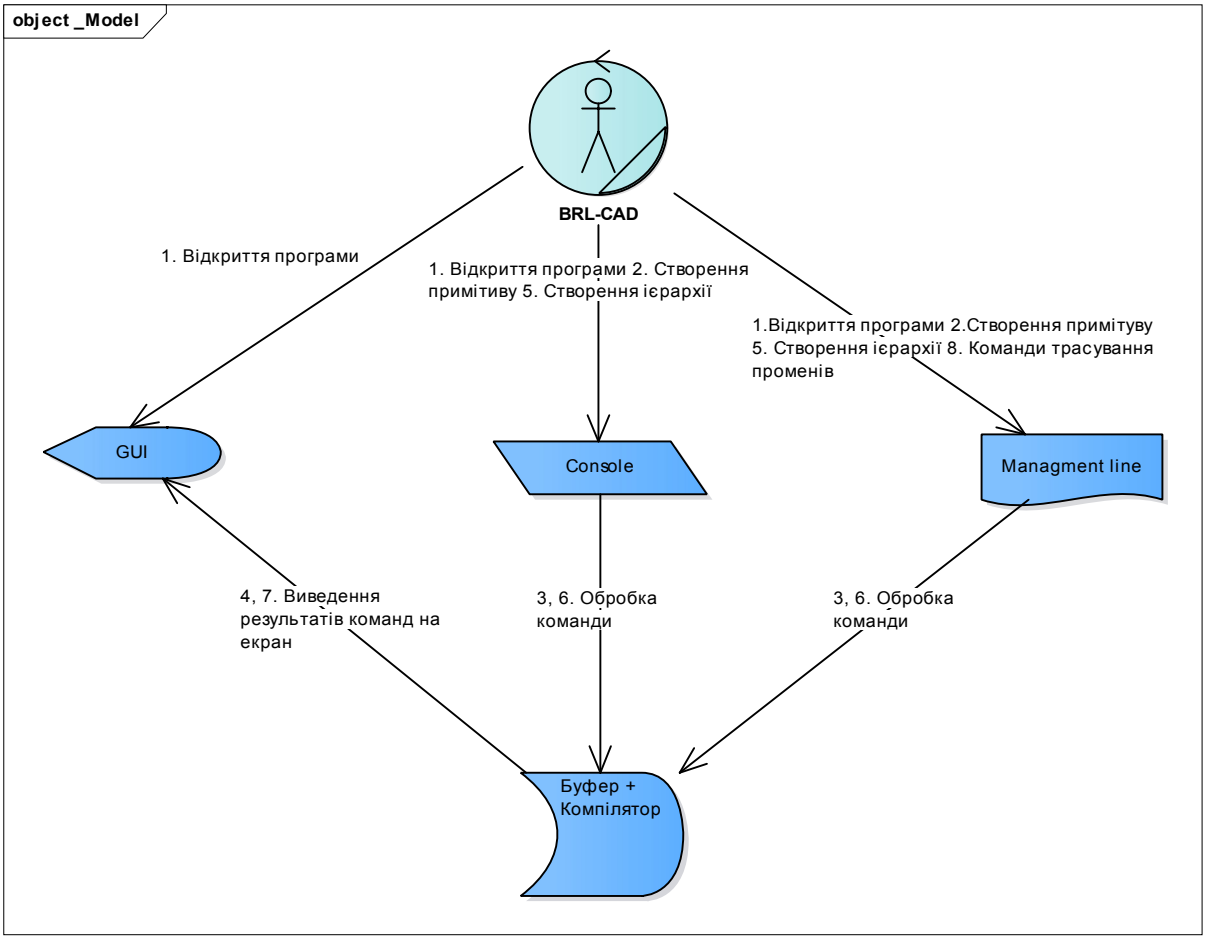


Рисунок 3.5 – Узагальнена модель опису оточення об’єктів для відкритого проєкту BRL-CAD

Таблиця 3.5 – Специфікація проєктного оточення моделі опису об’єктів

Об’єкт-сутність	Фактичний об’єкт	Проєктна функція (опис призначення)
Business worker	відкритий проєкт BRL-CAD	бізнес-виконавець
Display	графічний інтерфейс (GUI)	екран (інтерфейс)
Input / Output	консоль (Consol)	пристрій уводу або виводу
Report	командний рядок (Management line)	пристрій системного повідомлення
Storage	буфер та компілятор	пристрій накопичення та обробки

Як бачимо з рис. 3.5, МООО описує стандартні кроки, які проходить користувач при роботі із ПП:

а) «Відкриття програми»;

б) «Створення примітиву»;

в) «Обробка команди»;

д) «Виведення результатів команди на екран»;

е) «Створення ієрархії»;

ж) «Команди трасування променів».

МООО будуть корисні у тих випадках, коли потрібно оцінити наслідки зроблених змін, вони показують: які об'єкти взаємодіють один з одним. При внесенні змін до об'єкту, відразу зрозуміло: на які інші об'єкти, що оточують, це вплине. Помітимо, що першою дією у поданій моделі, відбувається відкриття програми, яка зачіпає відразу три створених об'єкти (рис. 3.5):

а) графічний інтерфейс «GUI»;

б) консоль «Consol»;

в) командний рядок «Management line».

Друга дія – це створення примітивів. Тут точно так саме можуть бути задіяні всі три елементи, що відображені раніше. Все це плавно перетікає у третю дію – обробку даних й команд буфером та компілятором. Після того, як створено декілька примітивних фігур, приходить час будувати ієрархію із них, тобто створювати більш складні графічні об’єкти. Стандартним кроком після створення графічних об’єктів є їх візуалізація. Для цього треба виконати розглянуту операцію управління трасуванням променів.

Під час проєктування МООО для проєкту BRL-CAD було визначено опис ключових об’єктів із оточенням яких необхідно встановлювати комунікаційне сполучення та розглянуто їхні функціональні можливості.

3.2.3.2 Модель операцій із проєктними класами

Призначенням моделі операцій із проєктними класами (МОПК) є можливість показати та генерувати класи ПП. Генерація класів може проводитися різними мовами з обраними користувачем специфікаціями – це одна з основних функцій для якої використовується МОПК. Особливої позитивної риси набувають МОПК при використанні у процесах проактивного розвитку, оскільки за їх допомогою відбувається подальше перекодування класів у сучасні (оновлені) МП.

У розробленій МОПК за основу оберемо мову C++, як найпоширенішу для створення легких та середньої складності ПП. Відкритий проєкт BRL-CAD не вимагає складних засобів розробки та низькорівневих МП (хоча створювалася ПС мовою «С» – фактично низькорівневою мовою: причиною була відсутність на той час розвинених високорівневих МП).

Розроблену МОПК для реінжинірингу ГБД у BRL-CAD наведено на рис. 3.6. При створенні МОПК яких-небудь складнощів не спостерігалося, оскільки для її побудови було залучено стандартний набір атрибутів та операцій, з яких зазвичай складаються ці моделі.

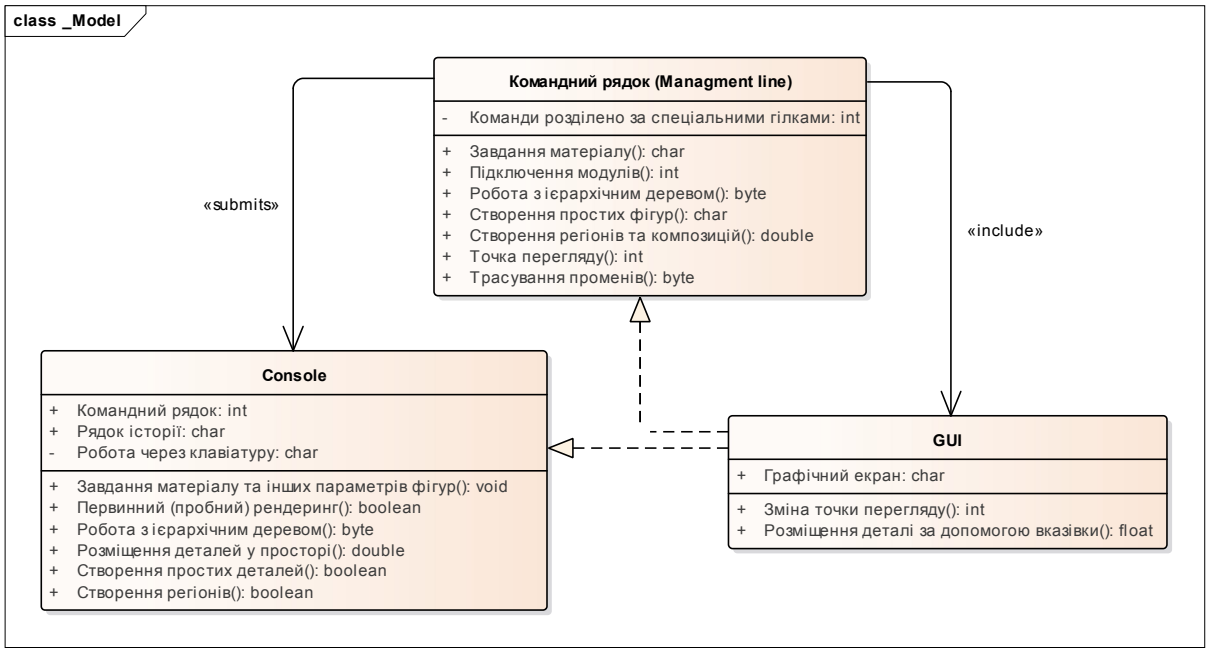


Рисунок 3.6 – Модель операцій із проєктними класами для виконання реінжинірингу ГБД у проєкті BRL-CAD

За основу взято 3 основних класи – це об’єкти, із якими взаємодіє користувач: консоль, графічний інтерфейс та командний рядок. Такий вибір обумовлений ще й тим, що основні специфікації та команди містяться саме в цих трьох класах.

Перший клас «GUI» загальний для всіх ПС схожого типу, виконує стандартні операції та має стандартні атрибути графічного екрану. При виконанні реінжинірингу ГБД, є можливість тільки змінювати вид та міняти розташування геометричних фігур, що й стало основними операціями, що виконуються цим класом.

Другий клас «Management line», також властивий майже всім ПС, його атрибутом є можливість роботи із різними гілками (відкривати та використовувати команди), а операціями даного класу стали саме ті команди, які розташовані у гілках: завдання матеріалу, фізичних констант, підготовка рендерингу, підключення модулів тощо.

Третій клас – це «Console» (рис. 3.6). Іноді консоль дає більш широкі можливості у роботі, але є деякі складності у її створенні – потрібно створювати новий набір команд (лексику), з якою буде працювати проєктувальник. В атрибутах даного класу знаходяться «Робота через клавіатуру», «Командний рядок» та «Рядок історії», який відображає вже виконані команди. Операції містять саме той «набір команд», що використовується поданою консоллю (у МОПК показано тільки основні команди): «Завдання матеріалу», «Первинний рендеринг», «Створення та розташування фігур» тощо.

Розглянуті три класи – пов'язані між собою трьома типами зв'язків – включення, залежність та підпорядкованість. «Командний рядок» підпорядковується «Консолі», тому обрано відповідний тип зв'язку. Також «Графічний інтерфейс» включає у себе «Командний рядок», тому він з’єднується асоціацією зі включенням «include». До графічного інтерфейсу «GUI» – підходять два відношення-реалізації (realize) тому, що класи «Консоль» та «Командний рядок» реалізують показ на екрані результатів оброблення графічної інформації. Таким чином було розроблено МОПК для реінжинірингу ГБД у відкритому проєкті BRL-CAD.

3.2.3.3 Модель компонентних рішень

Модель компонентних рішень (МКР) створюється для того, щоб ілюструвати розташування програмних модулів у комп'ютерному середовищі. МКР – одна з найголовніших, які використовуються у проєктній методології UML. Розроблену МКР наведено на рис. 3.7, її представлено стосовно мови програмування C++. У такому разі, кожний клас має свій власний заголовний файл (розширення «*.h») та файл тіла класу (розширення «*.cpp»).

У випадку з ПП BRL-CAD цей проєкт поділено на модулі. Таку МКР, безпосередньо, можна пов'язати із МОПК – це зручно тому, що кожне «Тіло підпрограми» (Subprogramm Body) – є окремим пакетом класів (рис. 3.7). Кожен з пакетів у МКР має свою відповідну специфікацію

(Package Specification), зазвичай властиву тільки йому. Проєктну специфікацію елементів, з яких складається МКР, зведено до табл. 3.6.

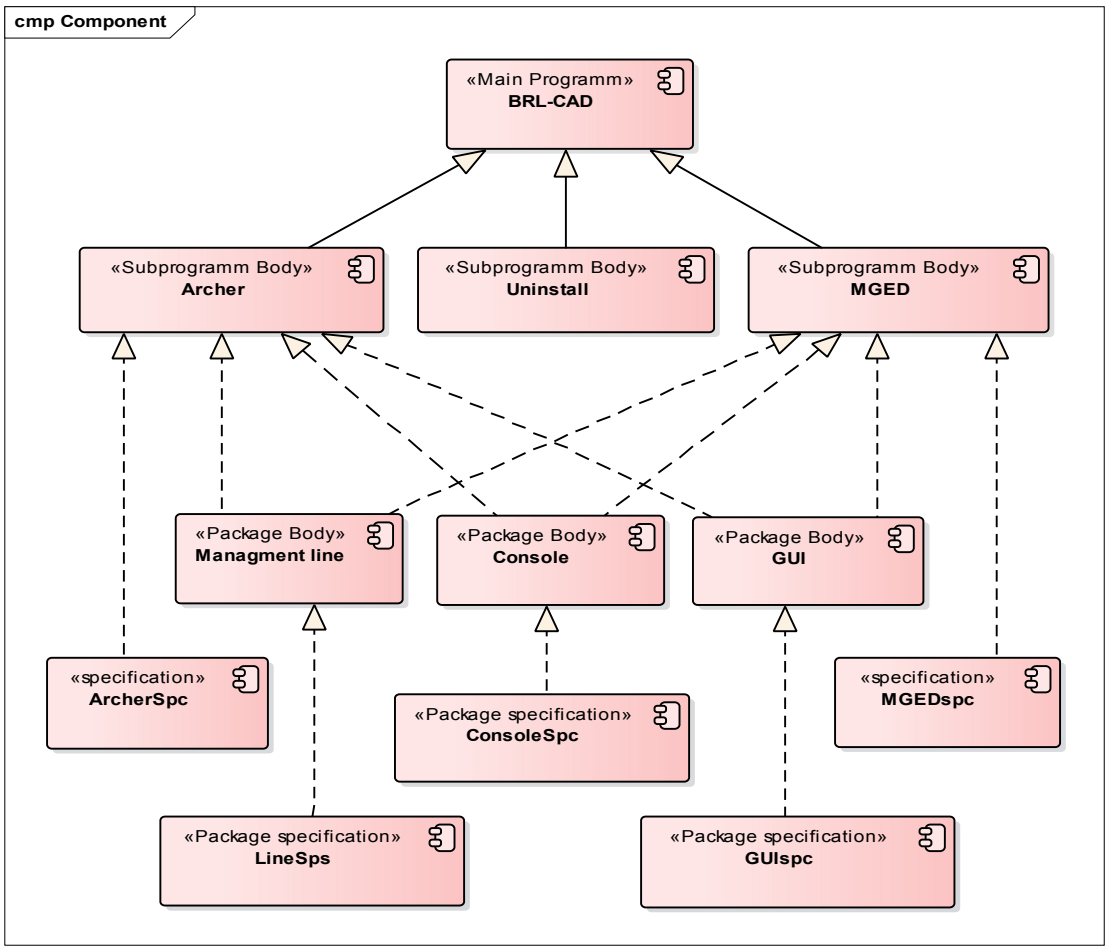


Рисунок 3.7 – Модель компонентних рішень для виконання реінжинірингу ГБД проєкта BRL-CAD

Таблиця 3.6 – Проєктна специфікація елементів МКР

Компонент	Назва	Набір специфікацій (spc)
модуль або «Тіло підпрограми» (Subprogram Body)	«Archer»	ArcherSpc
	«Unistall»	–
	«MGED»	MGEDspc
пакет (Package Body)	«Консоль»	ConsoleSpc
	«Графічний інтерфейс»	GUIspc
	«Командний рядок»	LineSpc

Залежності між компонентами повинні збігатися із залежностями між пакетами. Залежності показують: яким чином одні компоненти взаємодіють з іншими. Напрямок залежності показує рівень підпорядкування комунікації (стрілка стоїть з боку головного компонента).

При проєктуванні надвеликих систем може виявитися, що ПС повинна бути розкладена на кілька сотень або навіть тисяч компонентів. У такому разі будують вкладені ієрархічні МКР, які можуть бути об’єднані у відповідні пакети. Такий тип моделей дозволяє контролювати велику кількість модулів та їх зв'язків завдяки чітко вибудуваній ієрархії проєктних сутностей-компонетів.

3.2.3.4 Модель управління інформаційним розгортанням проєкту

Модель управління інформаційним розгортанням (МУІР) проєкту створюється для того, щоб показати фізичне розміщення ПС на системних носіях, накопичувачах та у БД. Існують деякі ПС, які використовують або просто підтримують спеціальні інтерфейси типу: Midi-клавіатури, графічні планшети, дігітайзери тощо.

МУІР рідко використовуються відносно простих ПС тому, що стандартне ПЗ розміщується на жорсткому диску комп’ютера та має стандартні периферійні пристрої уводу / виводу. Проте, для повноти складання системної архітектури проєкту, розробимо МУІР для розвитку ГБД ПП BRL-CAD, хоча вона буде виглядати дещо замалою, проте відбивати реальне розгортання ГБД (рис. 3.8).

У розробленій МУІР присутні наступні проєктні елементи, що зведено у проєктну специфікацію (табл. 3.7).

МУІР показує стандартне розміщення інстальованої ПС: ПП розташовано на платформі «ПК», що безпосередньо пов'язана із «Центральною БД» (зв’язком «вкладення» (nesting)) через «Глобальну мережу», за допомогою якої можна (у разі потреби) завантажити модулі та інше необхідне ПЗ.

На практиці МУІР може використовуватись нечасто з причин примарної простоти моделі, але багато розробників багатоланкових розподілених ПС можуть користуватися нею, як розгалуженою моделлю ПС. Що ж стосується проєкту розвитку ГБД, то застосування МУІР цілком виправдане, оскільки великі обсяги графічної інформації та

розпаралелювання розрахунків, що пов’язані із рендерингом, раціонально проводити на декількох вузлах обчислення та накопичування.

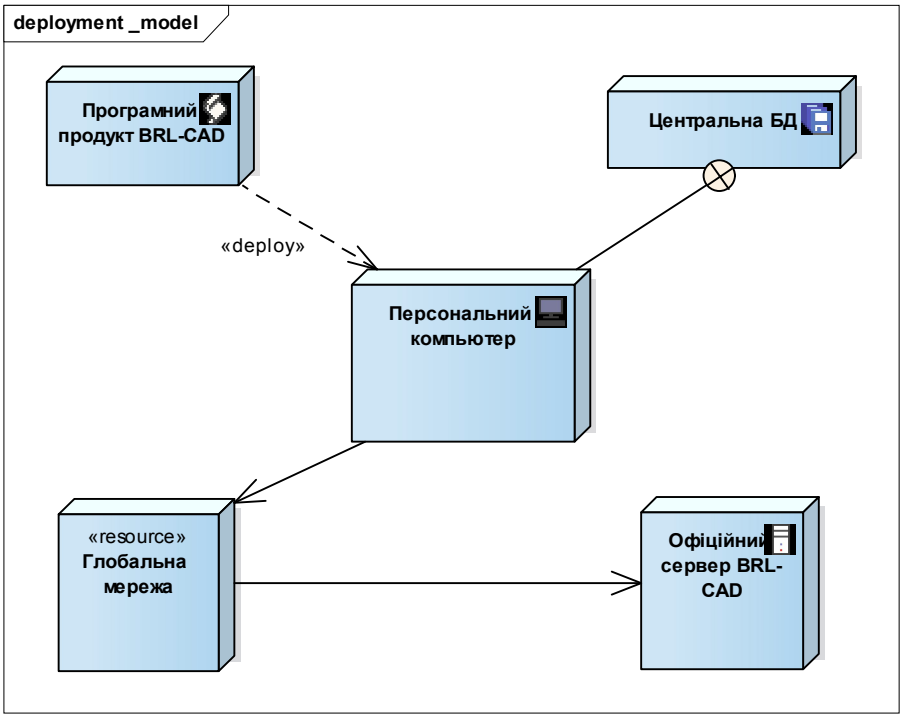


Рисунок 3.8 – Модель управління інформаційним розгортанням для проєкту розвитку ГБД у ПП BRL-CAD

Таблиця 3.7 – Проєктна специфікація моделі управління інформаційним розгортанням ГБД

Елемент	Назва	Стереотип
Вузол (node)	«Персональний комп’ютер»	PC
	«Програмний продукт BRL-CAD»	script library
	«Центральна БД»	disk array
	«Глобальна мережа»	resource
	«Офіційний сервер BRL-CAD»	server
Зв’язок (communication)		deploy
		nesting
		communication path

В ході розробки МУІР для проєкту розвитку ГБД у ПП BRL-CAD було сформовано модель, яка відображає майбутнє фізичне розміщення ГБД на розподілених системних вузлах, накопичувачах та у спеціалізованих БД.

3.3 Висновки за розділом

У розділі вирішено встановлене завдання формування моделей проактивного управління розвитком проєктів з реінжинірингу ПС, яке складається з двох підзавдань. Відмінною складовою завдань проєктів комп'ютерної графіки є обробка ГБД, які, по суті являють собою «звичайні» БД, але в основу яких закладено математичні алгоритми управління зображенням за сформованими статистичними координаційними даними проєкту. Такі можливості є далеко не у кожному проєкті ПС, але сучасні тенденції вимагають цього. Велика кількість ПП розроблюється з широким спектром моделюючих характеристик, BRL-CAD – один з таких ПП.

Відкритий проєкт BRL-CAD є прийнятним у застосуванні для досвідченого проєктувальника, проте для початківця або студента процес її застосування виявиться дуже ускладненим. У глобальній мережі немає жодних матеріалів українською або російською мовами, які хоча б поверхнево описували проєктувальнику роботу із системою у режимі «інструкція користувача». Матеріали англійською мовою – поверхневі та містять тільки декілька десятків консольних команд. При детальному аналізі середовища було виявлено наявність двох модулів, що містяться у структурі ПС, які допомагають потенційному користувачеві системою швидко конструювати необхідні ГБД.

Також фундаментальною властивістю проєкту можна назвати здатність підтримувати конструювання та аналіз візуальних моделей на основі складних об'єктів, що складаються з великого набору графічних примітивів.

Після проведених досліджень можна зробити висновок: потужний бік проєкту BRL-CAD – це надзвичайна швидкість засобів візуалізації, трасувальника променів та рендерингу. Після порівняння проєкту з аналогами, можна стверджувати, що процес візуалізації є одним із найшвидших серед існуючих. Остання перевага надає широкі перспективи для застосування ПП BRL-CAD у різноманітних галузях: військових,

промислових чи навчальних застосуваннях, таких як системи проєктування та аналізу у машинобудуванні, механічні вузли, архітектурні споруди, будова молекул тощо.

При вирішенні першого підзавдання, було сформовано моделі проактивного управління розвитком лінгвістичного та інформаційного забезпечення проєктів, які відрізняються застосуванням динамічного оточення породжувальних граматик, що дозволяє позбавитися виведення тупикових ланцюжків при переведенні з однієї МП до іншої.

При вирішенні другого підзавдання, було удосконалено моделі проактивного управління розвитком поведінкової та структурної частин проєктів з реінжинірингу ПС, які відрізняються уведенням оновлених структур уніфікованих діаграмних моделей, що дозволяє зберігати позитивні властивості відношень між проєктними класами, компонентами та сутностями ПС незалежно від МП.

Передумовою виконання УП реінжинірингу було моделювання проєктного каркасу (системної архітектури проєкту) на основі якої буде складено майбутні засади проактивного розвитку ГБД для композиційного підключення до оновленого відкритого проєкту BRL-CAD.

Результатом моделювання стало створення моделей проактивного розвитку ГБД для композиційного підключення до оновленого відкритого проєкту, а саме поведінкових:

- модель варіантів використання ПП;
- модель опису станів для проактивного УП;
- модель послідовності розвитку графічного представлення проєкту;
- динамічна модель діяльності реінжинірингу графічної структури проєкту;

та структурних моделей управління розвитком проєкту:

- модель опису оточення об’єктів;
- модель операцій із проєктними класами;
- модель компонентних рішень;
- модель управління інформаційним розгортанням проєкту;

як незамінних при УП з реінжинірингу ПС та ПП. При формуванні проєктної архітектури було використано розширену нотацію UML 2.5 та CASE-інструментарій Enterprise Architect 14.0.

Перспективи досліджень полягають у УП реінжинірингу: інструментальної групи роботи з утилітами відкритого ПП BRL-CAD, графічних бібліотек проєкту, системи використання команд та

можливостей, домовленостей про іменування файлів та геометрії. Також потребують удосконалення: проєктні процеси створення простих тіл, логічні операції, операції із комбінованими тілами, організація рендерингу та трасування променів.

Автор монографії хоче висловити велику подяку корпорації BRL-CAD за можливість відкритого користування і тестування вихідних файлів, збірок та систем, а також за підтримку крос-платформної методології відкритого проєкту.

Основні власні наукові дослідження, що подані автором у третьому розділі монографії, опубліковано у наукових працях автора: [1], [3], [6], [10], [12] – [14], [27], [30], [31], [40], [41], [43], [47], [55], [56], [65], [69], [70].

4 МОДЕЛІ ТА МЕТОДИ УПРАВЛІННЯ КОМПЛЕКСНОЮ ОЦІНКОЮ ПРОЄКТУ РЕІНЖИНІРИНГУ ПРОГРАМНИХ СИСТЕМ ТА ПРОДУКТІВ

ПС застосовуються у різноманітних галузях життя й діяльності людини, але найбільше поширення вони отримали у галузях, де необхідною є робота із багатьма рутинними операціями, великим обсягом інформації, яку необхідно одночасно оброблювати, змінювати, доповнювати та, звісно, управляти нею – це стосується промисловості, виробництва, транспорту, навчання та інфокомунікацій. При чому подальший розвиток цивілізації зменшує сегмент життя людини, що не охоплено ПС. Експлуатація, звичніше – використання, ПС у кожній окремій галузі (інформаційні технології, УП, виробництвом, транспортом, побудова телекомунікацій, наукова діяльність, освіта тощо) має свої принципові відмінності.

Спільної рисою для усіх ПС залишається те, що під впливом часу та інших невід'ємних факторів інформатизації (оновлення: операційних систем, МП, принципів дії розподілених систем обробки даних, що особливо важливо у сфері ІТ) відбувається еволюційне старіння ПС. Така тенденція призводе до погіршення швидкісних, інформаційно-комунікаційних, графічних, часових та інших характеристик аж до повної відмови ПС. Ось тут постає питання необхідності виконання удосконалення, покращення й переробки ПС, тобто проактивного розвитку, а всі ці процеси управління є складовими комплексного поняття «реінжиніринг».

Завданням розділу є розробка комплексу методів оцінки проєкту з реінжинірингу ПС у вигляді: методу розрахунку показників проєкту та методу представлення оцінки проєкту реінжинірингу ПС за допомогою проєктних коефіцієнтів.

4.1 Метод розрахунку показників оцінки проєкту при виконанні реінжинірингу програмних систем

Головною метою, що ставиться перед реалізацією методів УП створення ПС, є скорочення часу, собівартості та мінімізація ризиків, пов'язаних із розробкою об'єкта проєктування, який, до речі, може бути

будь-якого галузевого призначення. У представленому дослідженні наводиться розроблені методи оцінки проактивного розвитку ПС.

Управління скороченням собівартості проєктування, головним чином, відбувається за рахунок зменшення строків та персоналу, необхідного для здійснення проєктування ПС.

Завданням поданого підрозділу є розробка методу розрахунку показників проєкту, яка необхідна для прийняття остаточного рішення щодо його доцільності.

4.1.1 Опис вхідних даних та механізмів управління проєктними показниками

Виконання реінжинірингу потребує докорінної переробки (перепроєктування) ПС з метою покращення технічних характеристик, при виконанні якої треба унаслідуванням позитивні та відмовитися від впливу негативних якостей первинного об'єкту.

Для оцінки показників проєкту з реінжинірингу ПС необхідно мати набір даних із конкретизованих проєктних чинників, а саме: трьох видів ваги ВВ, 13 показників технічної складності, 8 показників кваліфікації персоналу, які отримано в результаті аналізу 5 – 7 вже закінчених схожих проєктів. Необхідно отримати розрахункові показники за якими буде відбуватись прогнозований фінансовий перелік витрат. Також треба формалізувати критерії оцінки та визначення складності реінжинірингу вже готової ПС, але яка потребує зміни (розвитку) із плином часу.

Уявлення про складність проєкту необхідні для подальшого оцінювання та розрахунку приблизного терміну реалізації реінжинірингу, до якого включаються час на: перепроєктування, тестування, переналагодження та деякі інші стадії ЖЦ ПП. Відповідь на це запитання дає метод Карнера (Karner's Use Case Points Method).

Метод, що орієнтовано на застосування показників Use Case Point (UCP), належить Густаву Карнеру [194]. Багато спеціалістів вважають, що облік видатків [195] на основі методу Карнера дає змогу отримати оцінку

194. Karner G. Resource Estimation for Objectory Projects: project report. Objective Systems. San Francisco: AB, 1993. 9 p.

195. Anda B. Effort Estimation of Use Cases for Incremental Large-Scale Software Development. 27-th International Conference on Software Engineering, St. Louis, MO, 15 – 21 May 2005, P. 303–311.

із похибкою у 20%, відносно реальних видатків [196], [197], тому саме цей метод візьмемо за основу при виконанні реінжинірингу ПС, з отриманням подальшого розвитку його стосовно до проєктних процедур проактивного розвитку.

Згідно з пропозиціями методу Карнера, пункти ВВ є функціями наступних аргументів:

а) кількість та складність ВВ в ПС;

б) кількість та складність акторів у ПС;

в) різні нефункціональні вимоги (такі як продуктивність, переносимість тощо, які були не описані у ВВ);

г) середовище розробки (МП, мотивація учасників тощо).

Згідно з [198] методика Карнера пропонує загальну оцінку витрат на розробку проєкту, але вона не дозволяє виділити який-небудь його етап. Більш того, методика не може бути використана, доти, поки всі діаграми ВВ не будуть спроєктовані [199]. Чинник технічної складності проєкту розраховується на підставі відповідних показників технічної складності проєкту [200].

Перед тим, як оцінювати обсяги проєкту, необхідно налаштувати управління технічними чинниками та чинниками середовища. Для визначення технічного чинника складності (англ. – Technical Complexity Factor (TCF)) та чинника складності оточуючого середовища (англ. – Environment Complexity Factor (ECF)) треба заповнити перелік чинників, що вплинуть на показники проєкту.

Першим з таких є чинник ваги, що визначається методом використання проєктних точок Карнера, проте вони можуть корегуватись відповідно до конкретних вимог проєкту.

---

196. Carroll E. R. Estimating Software Based on Use Case Point. OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, San Diego, 2005. P. 257–265. DOI: 10.1145/1094855.1094960

197. Clemmons R. Project Estimation with Use Case Points.*Cross Talk.*2016. Vol. 2, Iss. February. P. 18–22.

198. Дідковська М. В. Тестування: Критерії та методи. Київ: НТУУ «КПІ», 2010. 96 с.

199. Cohn M. Agile Estimating and Planning. New York City: Prentice Hall, 2005. 368 p.

200. Орлов С. А. Программная инженерия: уч. для вузов. 5-е изд. обновл. и доп. Санкт-Петербург: Питер, 2016. 640 с.

Наступний чинник – значення, що вказує на ступінь впливу визначеного чинника на проєкт. Індикацією значення є варіювання від «0» до «5», що означає діапазон управління від «відсутності» до «сильного» із можливими проміжними станами.

Перед тим, як оцінювати ПС за допомогою показників використання функцій, що будуть піддані реінжинірингу, необхідно призначити вагу для кожного з ВВ майбутньої ПС, спираючись на проєктні чинники (табл. 4.1).

Таблиця 4.1 – Проєктні чинники для управління вагою ВВ

Ранг	Показник складності	Інтерфейс користувача	Кількість сутностей БД	Кількість кроків сценарію	Кількість проєктних класів	Вага, W
1	Простий	Запропонований із шаблону	1	≤ 3	< 5	5
2	Середній	З елементами дизайну	≥ 2	4 – 7	5 – 10	10
3	Складний	Індивідуальна графічна розробка дизайну	≥ 3	> 7	> 10	15

Виходячи із табл. 4.1, слід призначати відповідну вагу ВВ у тому випадку, коли присутній хоча б один із чинників з максимальним показником для відповідного рангу. Сума показників складності ВВ знаходить своє відображення у показнику не скорегованих точок ВВ (англ. – Unadjusted Use Case Points (UUCP)). TCF обраховується виходячи із декількох показників.

Значення Unadjusted TCF Value (UTV), що розраховується, спираючись на перелік показників технічної складності проєкту (табл. 4.2), яким (окрім вагового коефіцієнту) призначується кількісне значення впливу показника на складність проєкту: від «0» – вплив не оказує до «5» – сильний вплив.

Таблиця 4.2 – Показники технічної складності УП проактивного розвитку ПС

Показник	Ваговий коефіцієнт	Кількісне значення	Результат
Розподілена система	2	5	10
Висока продуктивність	1	4	4
Ефективність роботи кінцевого користувача	1	2	2
Складна обробка даних	1	4	4
Повторне використання коду	1	2	2
Легкість інсталяції	0,5	5	2,5
Легкість використання	0,5	3	1,5
Портативність	2	3	6
Легкість корегування змін	1	3	3
Паралельність	1	2	2
Наявність спеціальних функцій безпеки	1	2	2
Доступ з боку зовнішніх користувачів	1	5	5
Вимоги до попереднього навчання користувачів	1	3	3

4.1.2 Методика організації розрахунків

Накопичене значення *UTV* розрахуємо як:

$$UTV = \sum_{i=1}^{13} [V_i \times k_i],$$

де V_i – ваговий коефіцієнт кожного з 13-ти показників технічної складності проєкту;

k_i – кількісне значення впливу кожного з 13-ти показників технічної складності проєкту.

Ваговий коефіцієнт TCF Weight Factor (TWF) – за замовчуванням встановлюється як 0,01; хоча теоретично може корегуватись. Постійна

проєкту TCF Constant (TC) – гарантований мінімум показника TC = 0,6. Таким чином TCF для кожного конкретного проєкту приймає значення:

$$TCF = TC + TWF \times UTV ,$$

тобто від теоретичного мінімуму: 0,6 до теоретичного максимуму: 1,25; чим ілюструє як позитивний вплив на витрати (зменшуючи на 40%), так і негативний – збільшуючи на 25%.

Оскільки проактивний розвиток ПС – це комплексний проєкт, що включає також стейкхолдерів, то необхідно при розрахунках показників проєкту розглянути проєктну команду, тобто включити чинники розробників – діючих осіб (ДО) до оцінки складності реінжинірингу ПС.

Розрахунковий чинник, що стосуються ДО, носить назву чинника складності оточуючого середовища (ECF) та обчислюється виходячи із декількох показників. Значення Unadjusted ECF Value (UEV), що розраховується спираючись на перелік показників кваліфікації проєктної команди, яка задіяна у реінжинірингу (табл. 4.3). Показнику призначується кількісне значення наявності конкретного набору вмінь у виконавця проєкту: від «0» – показнику немає до «5» – сильно виражений показник.

Таблиця 4.3 – Показники кваліфікації проєктної команди

Показник	Ваговий коефіцієнт	Кількісне значення	Результат
Знайомство з UML	1,5	4	6
Досвід роботи із конкретним середовищем	0,5	3	1,5
Досвід використання об’єктно-орієнованого програмування (ООП)	1	4	4
Кваліфікація системного аналітика	0,5	4	2
Мотивація	1	3	3
Стабільність вимог	2	4	8
Неповний робочий день	-1	0	0
Складність МП	-1	3	-3

Накопичене значення UEV розраховується як:

$$UEV = \sum_{i=1}^{8} [W_i \times m_i],$$

де W_i – ваговий коефіцієнт кожного з 8-ми показників кваліфікації проєктної команди;

m_i – кількісне значення наявності кожного з 8-ми показників проєктної команди.

Ваговий коефіцієнт ECF Weight Factor (EWF) – за замовчуванням встановлюється як (-0,03), хоча теоретично може корегуватись. Постійна проєкту ECF Constant (EC) – абсолютний максимум показника EC = 1,4.

Таким чином ECF для кожного конкретного проєкту приймає значення:

$$ECF = EC + EWF \times UEV ,$$

тобто від теоретичного мінімуму: 0,4 до теоретичного максимуму: 1,4.

ECF ілюструє як позитивний вплив на витрати (зменшуючи на 60%), так і негативний – збільшуючи на 40%.

Розрахункове значення для BB UCP – це число, що отримаємо для управління простим (див. табл. 4.1) BB. Значення вимірюється у так званих «проєктних точках» та обчислюється як:

$$UCP = UUCP \times TCF \times ECF .$$

Далі прогнозується загальний час УП Estimated Work Effort (EWE) у годинах:

$$EWE = UCP \times DR,$$

де DR (Duration) – значення часу тривання розробки однієї «проєктної точки» USP, яке краще за все обирати, спираючись на використання досвіду проєктів-аналогів.

DR є критичним чинником, і хоча його значення за замовчуванням дорівнює 10 год., воно легко може перевищувати 30 год., якщо значення UEV незбалансоване. При обранні DR необхідно брати до уваги кваліфікацію розробників, наприклад для нової команди слід обрати DR = 20 год.

Таким чином, приходимо до оцінки вартості проєкту EC:

$$EC = EWE \times DHR,$$

де DHR (Default Hourly Rate) – усереднена за статистичними даними погодинна ставка персоналу (програміста).

4.1.3 Достовірність експериментальних результатів з реалізації методу комплексної оцінки проєкту

Під час експериментальної перевірки реалізації методу розрахунку показників оцінки проєкту при управлінні реінжинірингом ПС, будемо використовувати безкоштовну 30-ти денну версію CASE-засобу Enterprise Architect 14, що узято із офіціального ресурсу розробника – Sparx Systems.

Наприклад, для реінжинірингу ПС, що представлена на рис. 4.1. у вигляді моделі ВВ, що розроблена на підставі наукових досліджень, запропонованих у розд. 3; необхідно, відповідно до табл. 4.1 налаштувати кожний із ВВ. Приклад управління показниками складності у середовищі Enterprise Architect 14 наведено на рис. 4.2.

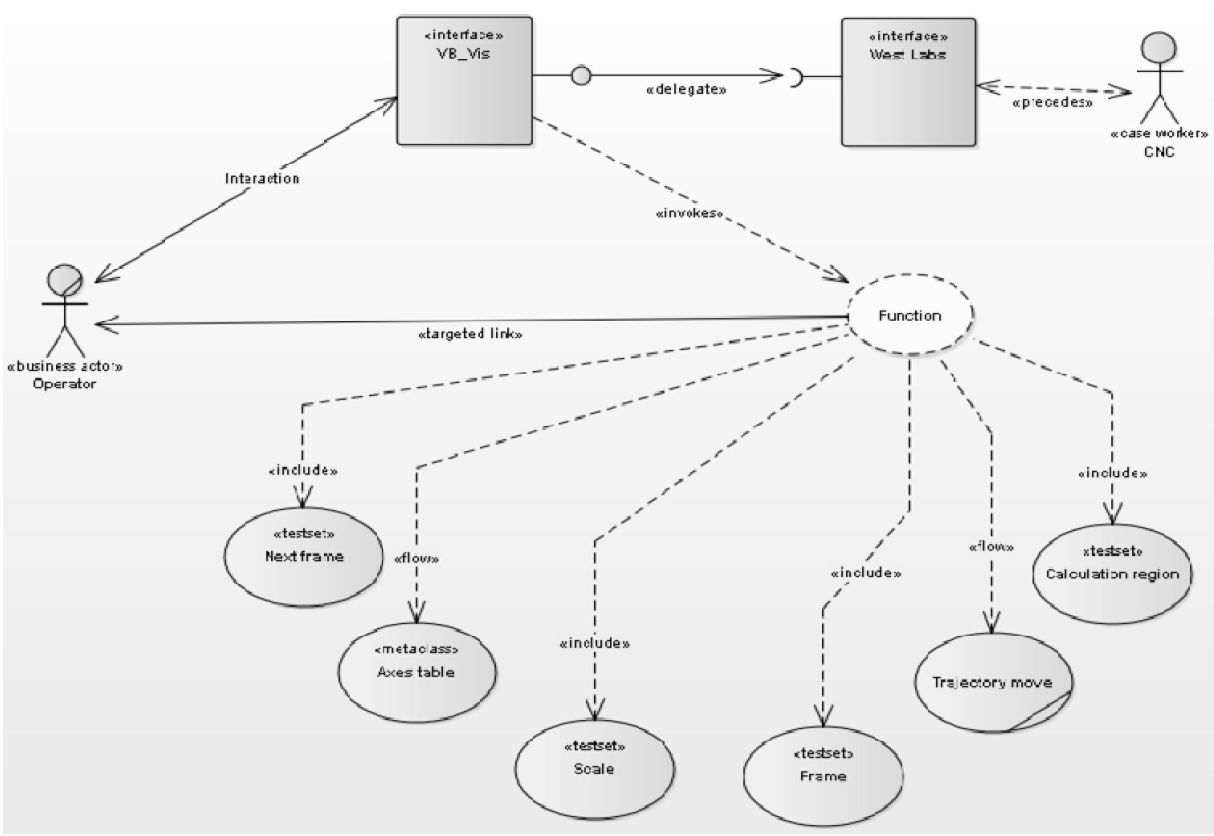


Рисунок 4.1 – Модель варіантів використання ПС, яку обрано для проведення оцінки проєктних показників реінжинірингу

Після настроювання ВВ, розрахунків чинників, вагових коефіцієнтів та обмежуючих констант цих чинників, можна переходити до наочної оцінки проєкту. Сукупність результатів розрахунку чинників оцінки та показників проєкту, перед виконанням проєкту реінжинірингу ПС, наведено на рис. 4.3.

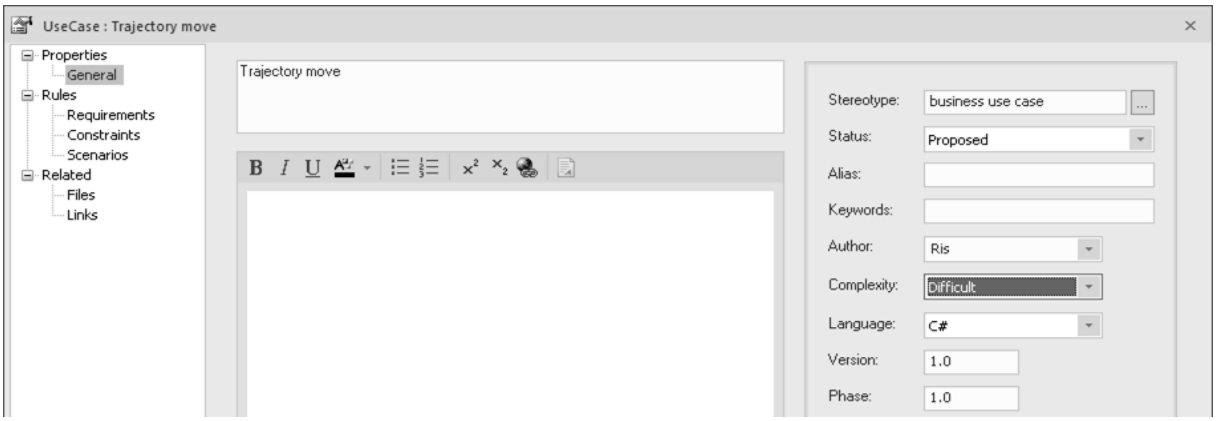


Рисунок 4.2 – Управління показниками складності для варіантів використання

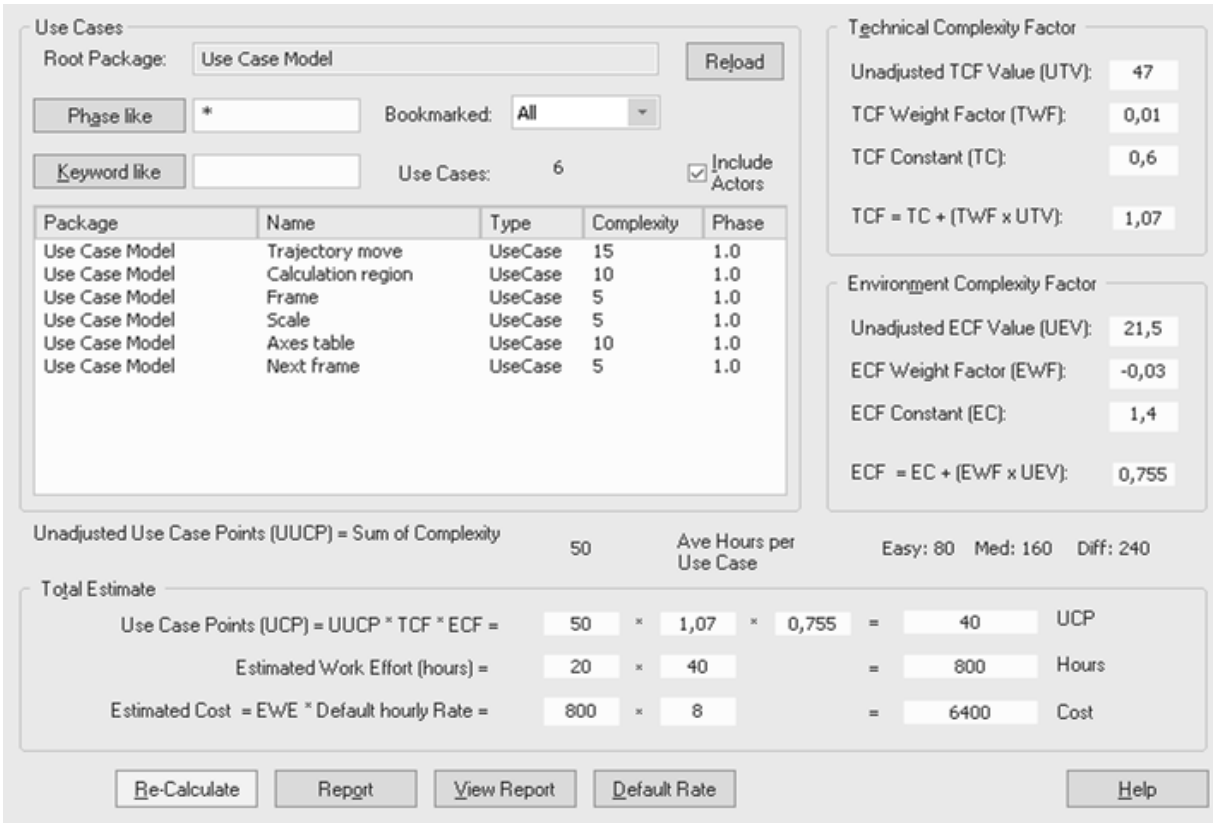


Рисунок 4.3 – Результати розрахунків чинників та значень, що впливають на оцінку проєктних показників управління реінжинірингом ПС

Для більшості розрахованих чинників та показників коментарі щодо їх одержання вичерпно наведено у п. 4.1.1, 4.1.2.

Пояснимо тільки призначення елементів керування, що подано у нижній частині рис. 4.3. Кнопка «Re-Calculate» – виконує перерахунок проєктних чинників, спираючись на змінені показники. Кнопка «Report» – генерує запит на створення та вказівку шляху збереження звіту з оцінки проєкту у форматі «RTF». Кнопка «View Report» – виконує перегляд встановлених показників оцінки проєкту із викликом ПЗ, асоційованого із переглядом Rich Text Files (RTF-файлів), наприклад: Microsoft Word. Кнопка «Default rate» – викликає діалогове вікно зміни показників оцінки проєкту, що завдані у Enterprise Architect 14 за замовчуванням.

4.1.4 Обговорення результатів застосування методу та їх валідація

Під час роботи з предметною галуззю монографії, дискусій та обговорень розглянутої теми на конференціях та спеціалізованих форумах, автор прийшов до наступних рекомендацій.

1) Метод Карнера рекомендує виключати розширені ВВ при розрахунку проєктних точок. Автор не згоден із цим та рекомендує розглядати усі ВВ при оцінки проєкту, якщо ці ВВ вимагають перепроєктування функціоналу, то існують зусилля на їх переробку, що повинні враховуватися за наведеною методикою.

2) Об’єктивний спосіб точно налаштувати при удосконалені проєкт – це розглянути ВВ 3-х – 7-ми вже успішно завершених (у свій час) проєктів, які вимагали реінжинірингу. Після аналізу завершеного проєкту і вивчення звіту про показники, доступні чинники можуть бути точно відкориговані, щоб дати оцінку фактичним годинам реінжинірингу. Згодом, можна використовувати ці дані у якості базової траєкторії ЖЦ ПП.

3) Достатня перевірка працездатності полягає в тому, щоб спираючись на показник «Ave Hours per Use Case» (рис. 4.3) проаналізувати: чи можна управляти розвитком простого, середнього чи складного ВВ у відведений час (включаючи усі стадії ЖЦ ПП).

4) Не слід очікувати вичерпної відповіді на питання «скільки коштує реінжиніринг?» або «як довго він триватиме?» – необхідно оцінювати отримані статистичні дані, управляти показниками та аналізувати досвід втілення показників успішних проєктів. Розглянута методика забезпечує на

17 – 19% меншу похибку, ніж метод Карнера, відносно фактичних видатків.

Відмінність від аналогів: при складанні методу комплексної оцінки показників ПС, яка буде піддана реінжинірингу, було виділено чинники, що впливають на планування ресурсів ПС:

а) дослідження моделі управління вимогами;

б) кількість кроків для виконання реінжинірингу елементу ПС;

в) технічна складність проєкту;

д) рівень кваліфікації команди програмістів.

Кожна змінна, що використовується для розрахунків у рамках реінжинірингу визначається та обчислюється окремо із використанням:

а) вимірювань характерних параметрів;

б) вагових коефіцієнтів;

в) обмежуючих констант.

Встановлення коефіцієнтів та обрання значення констант засновані на багато чисельній статистиці найбільш схожих проєктів, що виконані за технологією А. Якобсона. Управління параметрами проводяться керівником проєкту, який, спираючись на досвід 3-х – 7-ми схожих проєктів, виходить із власних уявлень про технічну складність проєкту та можливостях команди, яка буде виконувати реінжиніринг.

4.2 Моделі управління процесом оцінки проєкту з реінжинірингу програмних систем

УП з розвитку ПС, у кожній окремій галузі їх застосування (будівництво, телекомунікації та зв'язок, сфера послуг, освіта та ін.), має свої принципові відмінності. Як самостійний приклад можна навести розвиток ергатичної ПС, а саме: системи моніторингу та дистанційного управління (англ. – Supervisory Control And Data Acquisition (SCADA)). Їх призначено для спостерігання та управління віддаленим об’єктом, який, до речі, може бути небезпечним для здоров’я оператора; крім того, за допомогою SCADA-систем виконується аналіз, накопичення та необхідне сортування робочих даних.

Вочевидь, SCADA-система, як часний випадок ПС – повинна бути такою, що розвивається. За світовими тенденціями УП розвитку ПП, її експлуатація триває 3 – 4 роки. Звісно, що при оновленні об’єкту – оновлюється й ПС, за допомогою якої об’єкт обслуговується. На цьому

етапі виникає питання: що робити, коли ПС жорстко прив’язано до об’єкту експлуатації? Наприклад: суднова SCADA-система, що являє собою ПС, яку жорстко прив’язано до суднової енергетичної установки та до вимірювальних каналів, при цьому експлуатація судна складає 12 – 15 років.

Відповідь на це питання – одна: необхідно застосовувати УП реінжинірингу щодо ПС, при цьому архітектура системи може залишатися незмінною, тобто фактично виконується перепроєктування.

Завданням поданого підрозділу є формування моделей управління процесом оцінки реінжинірингу ПС на підставі ідеалізованого подання їх розвитку та із уведенням ресурсних та часових обмежень на виконання проєкту реінжинірингу.

4.2.1 Опис вхідних даних та механізмів управління моделюванням

Виконання реінжинірингу (перепроєктування) ПС неможливе без використання сучасних методів УП [154]. Розробка, впровадження та подальший проактивний розвиток ПС та ПП у різних галузях економіки країни є однією з важливих задач із просування цієї галузі на сучасний рівень світового промислового виробництва [159].

Розроблені ПС дають значну ефективність в тому випадку, коли результати проєктів по їх створенню використовуються у технічному виробництві [169]. Ефективність, в даному випадку, обумовлюється тим, що за сучасних темпах розвитку науки і техніки, виникає протиріччя між зростаючим рівнем науково-технічних досягнень та існуючими методами й технічними засобами проактивного розвитку ПС та ПП [201].

На подолання цих протиріч спрямована задача, яка вирішується у поданому підрозділі: пропонуються ідеалізовані моделі управління проєктом реінжинірингу (ІМУПР), що здатні оцінити час та витрати, необхідні для виконання реінжинірингу ПС.

Пакети робіт визначають зону діяльності в системі проєктування з управління всіма завданнями, що пов'язані з розробкою проєктів для кожної з виділених груп об'єктів. Кожен пакет ПС поєднує в собі

---

201. Reza Afshari A. R., Brtka V., Ćoćkalo-Hronjec M. Project risk management in Iranian software projects. *Journal of Engineering Management and Competitiveness (JEMC)*. 2018. Vol. 8. No. 2. P. 81–88.

діяльність з проєктування всіх об'єктів одного виду та відносяться до однієї певної області проєктування.

Корпорації, що розроблюють ПС – проєктують багато спеціалізованих ПП, наприклад – AutoDesk. В них є цілий комплект програм для роботи с машинобудівним профілем (Inventor), архітектурою (ArchiCad), дизайном (3dMAx) та УП у широкому сенсі (AutoCad) [150].

Завдяки потужним обчислювальним засобам у ПС за допомогою пошукових систем, що містять банки і БД готових проєктно-конструкторських рішень, можливо досить швидко внести корегування у необхідні параметри виробів (розміри, форма, порядок обробки тощо), що виготовляються; а також у послідовність технологічних операцій, тобто виконувати УП стосовно до виробничого процесу [152].

Подібне переорієнтування (у широкому сенсі УП), можна визначити як реінжиніринг ПС та ПП [155]. Проблему УП реінжинірингу ПС було розглянуто у [161]. Методологічні засади реінжинірингу було закладено у [157] та розширено стосовно до ПС та ПП у [156]. Методологія, що викладена у [164], стала у нагоді у якості базової траєкторії досліджень. Принципи та дослідження у [165], підказали практичні аспекти застосування ІМУПР.

4.2.2 Полярна ідеалізована модель управління проєктом реінжинірингу

Початкова ідея створення ІМУПР була спрямована до побудови моделі у полярній системі координат. Побудова запропонованої ІМУПР відбувалася наступним чином: візьмемо у просторі довільну точку O (полюс), яку назвемо полюсом реінжинірингу та побудуємо проміні OX (абсциса) та OY (ордината) (рис. 4.4) – це будуть вісі проєктних витрат (у євро).

Нехай M – довільна точка площини, M_x та M_y – її прямокутні координати (проекції на вісі OX та OY). Вектор OM – радіус-вектор проєктних витрат (модуль якого збільшується). У подальшому позначимо цей радіус-вектор як ρ (рис. 4.4). Приймемо будь-який додатній відрізок за одиницю витрат OA. Кут повороту φ – це час, протягом якого відбувається проєкт реінжинірингу (зростає за годинниковою стрілкою та, надалі, збільшується кількість повних обертів).

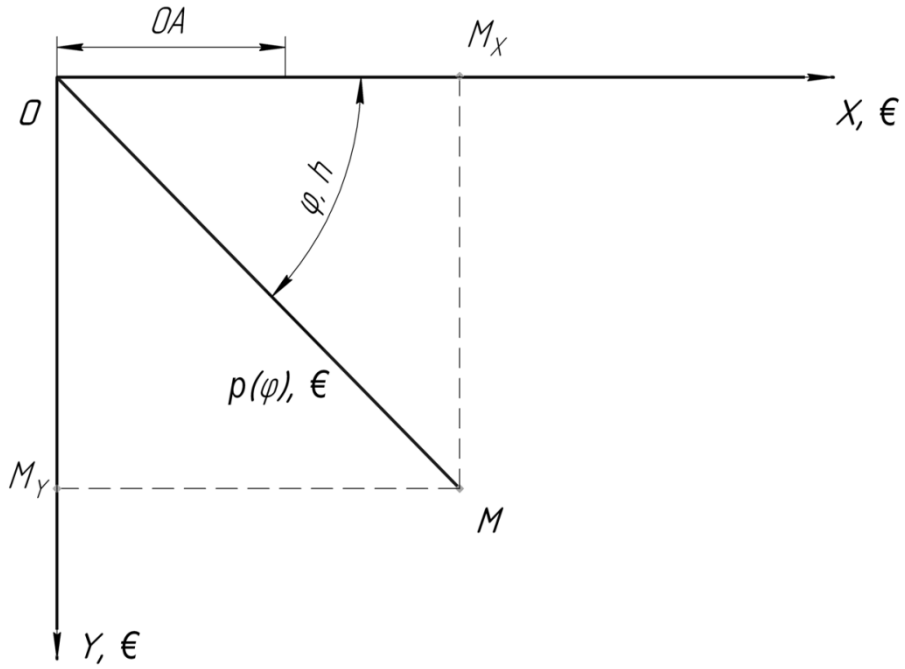


Рисунок 4.4 – ІМУПР, що побудована у полярній системі координат

Таким чином, ρ та φ – полярні координати. Тоді:

$$M_x = \rho \cos \varphi;$$
$$M_y = \rho \sin \varphi;$$

а радіус-вектор витрат:

$$\rho = \sqrt{(M_x)^2 + (M_y)^2}.$$

При оцінки полярної моделі проєкту реінжинірингу, експертами у галузі УП та математичного моделювання було знайдено недоліки, що значною мірою перешкоджають та ускладнюють її практичне застосуванню системними аналітиками та менеджерами проєкту:

1) жодної з формул з п. 4.2.2 недостатньо для розрахунку часу проєкту реінжинірингу φ, що потім було математично проаналізовано та уточнено [202, стор. 111], у зв’язку із чим проєктний час було фіксовано за фактом;

---

202. Выгодский М. Я. Справочник по высшей математике. Москва: ГИТТЛ, 1957. 784 с.

2) відсутня можливість обліку кількості ідентифікованих програмних компонентів (фізичних модулів коду), що необхідно наводити у лінійному масштабі, або кількості верифікованих рядків програмного коду – у логарифмічному масштабі;

3) подання ІМУПР тільки у проекції часу та витрат й неможливість реалізувати модель у декількох інших проекціях, що цікавлять системного архітектора, а саме: у ізометричній проекції програмних компонентів; у логарифмічній проекції рядків програмного коду.

З аналізу та обговорення цих недоліків було сформовано обмеження на застосування полярної моделі проєкту реінжинірингу: полярна система координат не відповідає висунутим вимогам подання проекцій з боку УП, а значить необхідно шукати шляхи побудови ІМУПР у інших системах координат.

4.2.3 Тривимірна спіральна модель управління проєктом реінжинірингу

Подальші дослідження ІМУПР, що наведені у підрозділі, узагальнили досвід роботи із системами координат у яких можливе застосування спіральної організації відліку.

Метод Боема (Boehm) [164], [156] та дослідження [152], [155], [165], призвели до думки об’єднати ідеї побудови ІМУПР у вигляді спіралі Архімеда та перенести її до циліндричної системи координат. В основу моделі закладено спіральний принцип організації відліку.

Початок побудови спіральної ІМУПР дещо схожий із вище розглянутою полярною ІМУПР. Однак, після завдання нульової точки реінжинірингу наступають нові етапи:

1) завдання осі *OZ* (рис. 4.5), яка відображує число ідентифікованих програмних компонентів (*i*) у лінійному масштабі або число верифікованих рядків програмного коду (*j*) у логарифмічному масштабі;

2) побудова послідовності точок $M_{i \vee j}$, які, власне, складають криву спіралі (рис. 4.6) – це компоненти програмного коду (для зручності, нижче, $M_{i \vee j}$ будемо записувати як M_i);

3) побудова вектору витрат (OM_i), який з’єднує полюс реінжинірингу (*O*) з поточною точкою спіралі M_i (рис. 4.6) та перевизначення його як ρ:

$$\rho = OM_{i \vee j}.$$

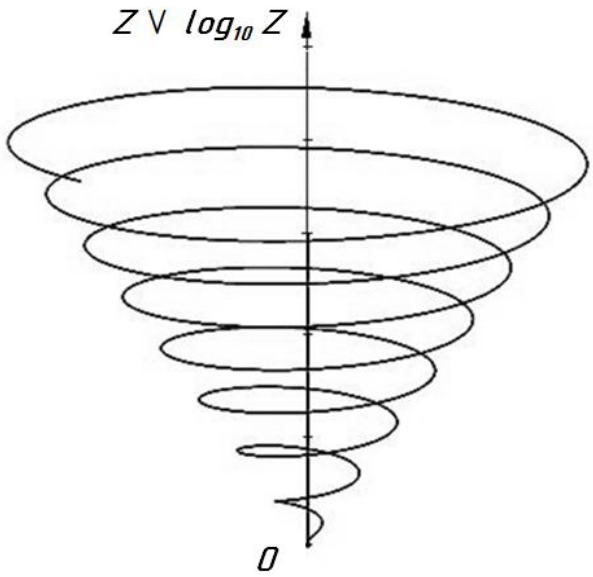


Рисунок 4.5 – Ескіз побудови спіральної ІМУПР

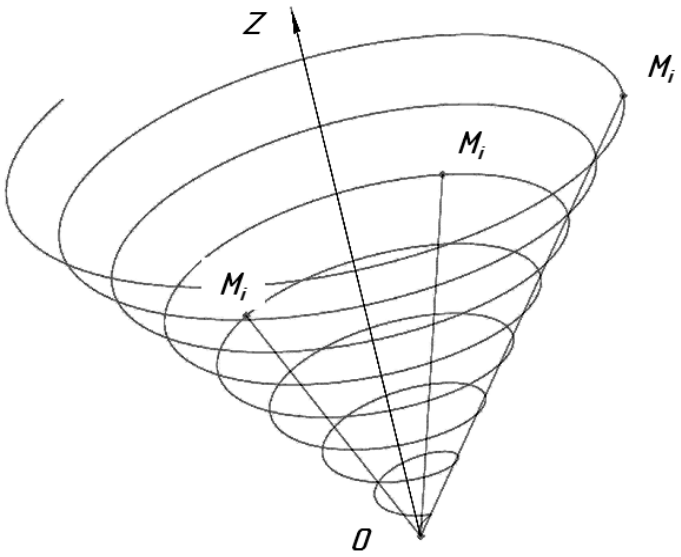


Рисунок 4.6 – Побудова послідовності точок M_i та вектору витрат OM_i

4.2.4 Достовірність експериментальних результатів із реалізації моделі управління проєктом

Для проведення експериментальної перевірки досліджень було обрано сім проєктів з реінжинірингу ПП, що написано однією МП до іншої мови. За домовленістю, виконавцями проєктів, було надано початкові умови (кількість рядків програмного коду) та фактичний час повного виконання, який було зафіксовано у проєктних діаграмах Ґанта.

За цими даними були побудовані ІМУПР, числовий матеріал яких подано у табл. 4.4. При виконанні проєктів вимірювання часу відбувається з округленням до півгодини.

Таблиця 4.4 – Час виконання реінжинірингу проєкту

Номер проєкту	Час, год.	
	Модельний	Фактичний
1	157	185
2	247	190
3	70	58,5
4	432,5	606
5	104,5	102,5
6	807	623,5
7	565	730,5

Кут оберту φ між полюсом реінжинірингу та поточною точкою M_i спіралі (кут повороту вектору OM_i) – це час, який необхідно затратити на виконання реінжинірингу, причому, чим далі точка M_i від O, тим більше значення φ (рис. 4.7).

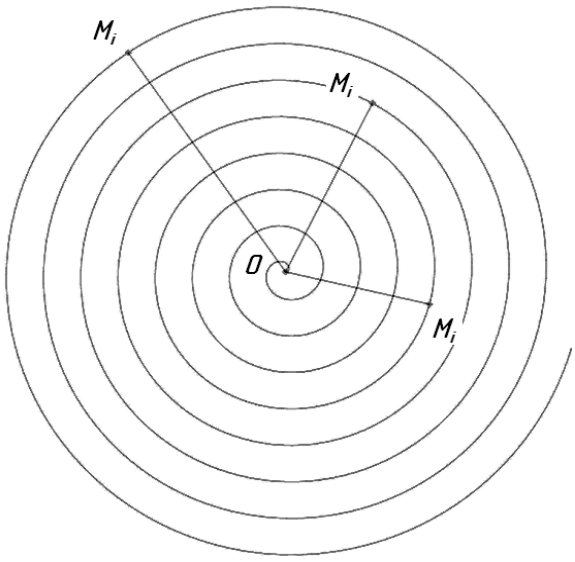


Рисунок 4.7 – Подання ІМУПР у проекції часу

Тобто кожний новий виток спіралі (n) додає 2π часу, а час тоді дорівнює:

$$t_i = \varphi_i(n+1), [\text{ум. од. часу}].$$

Проекція представлення ІМУПР може бути різна, наприклад, якщо подати проекцію уздовж вісі ідентифікованих програмних компонентів (*OZ*), як показано на рис. 4.7, то ІМУПР буде являти собою Архімедову спіраль, що, у нашому випадку, описується рівнянням:

$$\frac{\rho}{M_{i \vee j}} = \frac{t_{i \vee j}}{2\pi}$$

або

$$\rho = \xi\, t_{i \vee j},$$

де

$$\xi = \frac{M_{i \vee j}}{2\pi}.$$

Загалом, конфігурація спіралі може бути різна та залежить від багатьох чинників, що закладено у математичній моделі:

$$\Phi(r, \varphi, Z)\begin{Bmatrix} r(i \vee j) = \delta \times (i \vee j),\ \delta \forall [0 \ldots \infty] \\ \varphi(i \vee j) = \theta \times (i \vee j),\ \theta \forall [0 \ldots \infty] \\ Z(i \vee j) = \varepsilon \times (i \vee j),\ \varepsilon \forall [0 \ldots \infty] \end{Bmatrix} \wedge (i \vee j) \forall [1 \ldots \infty],$$

де Ф – функція конічної спіралі;

Z – число ідентифікованих програмних компонентів (у лінійному масштабі) або рядків програмного коду (у логарифмічному масштабі);

δ – коефіцієнт автоматизації реінжинірингу;

θ – коефіцієнт схожості компонентів;

ε – верхня межа граничних витрат.

За даними табл. 4.4 побудована діаграма порівняння прогнозуємого за ІМУПР часу та фактичного часу виконання проєкту реінжинірингу (рис. 4.8).

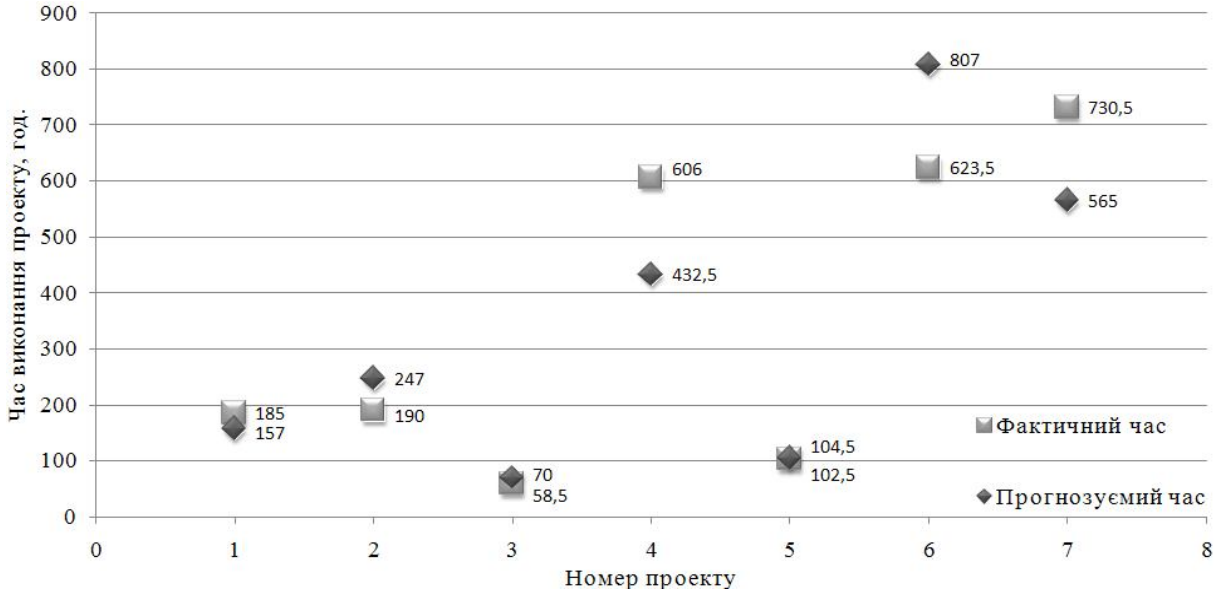


Рисунок 4.8 – Діаграма порівняння розрахованого за ІМУПР прогнозуємого часу та часу фактичного виконання проєкту реінжинірингу ПП

Статистичні результати порівняння зведено до табл. 4.5.

Таблиця 4.5 – Розбіжність прогнозуємого з фактичним часом виконання проєкту реінжинірингу

Номер проєкту	Абсолютна похибка, год.	Відносна похибка, %
1	28	15
2	57	30
3	11,5	20
4	173,5	29
5	2	2
6	183,5	30
7	165,5	23
Середнє значення відносної похибки, %:		21

З наведених графічних побудов функцій Ф (r, φ, Z) (рис. 4.6) – можна зробити висновки, що OM_i є утворюючою. Поверхня ІМУПР, для якої функція Ф (r, φ, Z) є направляючою, являє собою лінійчату конічну поверхню, що може бути зворотньовідновлена рухом утворюючої OM_i.

4.2.5 Результати застосування ідеалізованих моделей управління проєктом реінжинірингу

Запропоновані ІМУПР видів забезпечення ПС являють собою еволюційні спіралі, які побудовані у циліндричній системі координат. Операції з ІМУПР можуть відбуватися у наступних проекціях: у проекції часу та витрат; у ізометричній проекції програмних компонентів; у логарифмічній проекції рядків програмного коду.

Сформовані ІМУПР дозволяють підвищити точність оцінки проєктних витрат з перепроєктування ПС, за рахунок уведення ресурсних та часових обмежень на виконання проєкту.

Експериментальні дослідження спіральної ІМУПР за даними 7 виконаних проєктів показали прогнозування витрат на проєкт реінжинірингу ПС із максимальною похибкою 30% (табл. 4.5). Може здаватися, що діапазон похибки занадто великий, проте для первинної оцінки порядку витрат (при тому, що жодних інших даних на етапі прийняття рішення не існує), спіральна ІМУПР – цілком придатна.

Реалізація та застосування подальших удосконалень моделі дозволять зменшити середнє значення відносної похибки прогнозування проєктних витрат з теперішнього значення 21% до перспективного 7 ÷ 10%.

Прогнозується, що реінжиніринг, який буде виконано за допомогою розроблених ІМУПР дозволить не тільки скоротити проєктні витрати на перепроєктування ПС, але й підвищити ефективність технічного супроводу, збільшити ЖЦ ПС, що вже знаходяться у експлуатації та подолати протиріччя між швидкими темпами розвитку теорії УП та реалізації процесів проєктування нових ПС та ПП.

4.3 Метод представлення оцінки проєкту реінжинірингу програмних систем за допомогою проєктних коефіцієнтів

Таким чином, з наведених матеріалів, зрозуміло, що реінжиніринг дозволяє виконати еволюціонування ПС, шляхом внесення позитивних змін до її структури, з метою підвищення зручності її ж експлуатації та технічного супроводу.

Завдання підрозділу: сформувати метод представлення оцінки реінжинірингу ПС за допомогою проєктних коефіцієнтів, на підставі якого

приймається остаточне рішення щодо доцільності виконання реінжинірингу.

4.3.1 Виділення частин загальної проблеми та опис вхідних даних

Проєкт реінжинірингу дає змогу виконати еволюціонування ПП шляхом позитивних змін видів його забезпечення з метою підвищення зручності її експлуатації та супроводу.

Звісно, що не будь-яку ПС можна піддати реінжинірингу: іноді зміни у програмному коді тягнуть більші витрати ніж нова розробка, оскільки необхідно не тільки знайти спеціалістів у визначеній МП, але й оплатити їхній час на аналіз та розібрання старої версії проєкту ПС, що цікавить.

Таким чином, у якості вхідних даних розв’язуваної проблеми уводяться:

– вартість перепроєктування ПС із уведеним обмеженням, а саме: мінімізація вартості;

– швидкість уведення до експлуатації оновленої ПС із уведеним обмежуванням на затягування, тобто максимізація швидкості;

– виявлення помилок у застарілій версії ПС, які необхідно намагатись виключати у оновленій, тобто: мінімізація помилок.

Бажані результати розв’язання підзавдання цього розділу: формалізація критеріїв рентабельності реінжинірингу ПС, за якими після моделювання визначених порівняльних характеристик буде прийматись однозначне рішення щодо застосування або відхилення реінжинірингу.

Проте, необхідно зауважити, що ні в якому разі не можна остаточно оцінювати порівняльні характеристики тільки за критеріями, що охоплюють лише аналітику програмного коду реалізації ПС. Критерії оцінювання повинні базуватися на всебічному аналізі усіх видів забезпечення проєкту ПС, що підлягає удосконаленню.

4.3.2 Дослідження та матеріали, на підставі яких сформовано метод

Застосування реінжинірингу щодо УП з розвитку ПС та ПП, як початковий етап розвитку наукової думки з перепроєктування ПС розпочато у [150], [198]. Завдяки уведеній методології та обчислювальному інструментарію [152] можна виконати оцінку проєктних

ресурсів, спираючись на ідею Карнера [193], що закладено у основу розробленого метода розрахунку показників оцінки проєкту при виконанні реінжинірингу ПС.

Саме ж поняття «реінжиніринг», спочатку з'явилося у представленні оптимізації бізнес-процесів корпорації [155], [157] як переорієнтування та реорганізація структури забезпечення (у широкому сенсі) [154]. У роботі [169] було застосовано об'єктний підхід до управління бізнес-процесами у формі реструктуризації, та переведення окремих компонентів системи до іншої, сучаснішої, а також розглянуто процеси модифікації та модернізації відкритих структур і систем даних [203].

Методологія, що викладена у [165] стала у нагоді у якості базової траєкторії досліджень. Принципи, розглянуті у [194] та дослідження у [195], підказали практичні аспекти оцінки проєктів ПС та ПП [196], що підлягають реінжинірингу.

Головним завданням підрозділу є дослідження впливу проєктних коефіцієнтів, що уводяться до моделі реінжинірингу ПС, на відповідність до фактичних статистичних даних вже реально виконаних проєктів з реінжинірингу ПС та ПП.

Теоретичні відомості з матеріалів досліджень (підрозд. 4.2) для практичного застосування потребують розширення управління конфігурацією при моделюванні, для чого необхідно увести до моделі інструменти проєктних коефіцієнтів.

4.3.3 Управління годографами проєкту реінжинірингу

Оскільки моделі управління процесом оцінки проєкту реінжинірингу ПС та схеми їх побудови наведено автором у підрозд. 4.2, то відразу перейдемо до уведення проєктних коефіцієнтів до моделі та управління побудовою годографів реінжинірингу.

4.3.3.1 Коефіцієнт автоматизації

Системи автоматизації, стосовно до задач УП – це комплекс засобів автоматизації проєктування, що взаємопов'язано із підрозділами

---

203. O'Reily T. Open Source Paradigm Shift by Tim O'Reilly. URL : http://archive.oreilly.com/pub/a/oreilly/tim/ articles/paradigmshift_0504.html (дата звернення : 23.02.2019).

організації, яка саме й виконує управління проєктуванням. В такому разі, під автоматизацією проєкту реінжинірингу слід розуміти такий спосіб, при якому всі проєктні операції, процедури або їх частини здійснюється при взаємодії проєктувальника та електронного інструментарію УП (робоча станція, настільний комп'ютер, модифікації ноутбука, планшетний комп'ютер, різноманітні пристрої уведення та виведення інформації.

Чим більше процес реінжинірингу автоматизований, тим більш пласкої форми набирає модель (рис. 4.9), а це значить, що вектор витрат (OM_i) із зростанням кількості компонентів програмного коду (i) чи верифікованих рядків програмного коду (j) зростає не значно, а витрачений час на виконання реінжинірингу t_i – визначається машинним часом.

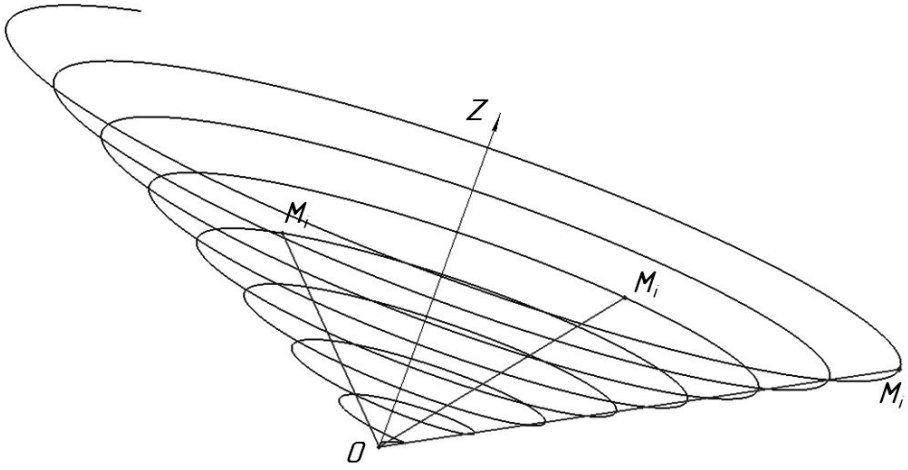


Рисунок 4.9 – Годограф з високим коефіцієнтом автоматизації реінжинірингу ПС: O – полюс реінжинірингу; Z – число ідентифікованих програмних компонентів(у лінійному масштабі) або рядків програмного коду (у логарифмічному масштабі); OZ – вісь компонентів (апліката)

Іншими словами – якщо узяти ПС із визначеною фіксованою i, у яку помістимо довільну точку площини (M_i), та збільшимо у моделі коефіцієнт автоматизації реінжинірингу (δ), то OM_i буде зменшуватись:

$$\text{if } \delta \to \max \therefore OM_{i \vee j} \to \min;$$

$$\lim_{\delta \to \max} \left\langle OM_{i \vee j}(\delta) \right\rangle = \min,$$

де M_{ivj} – побудована послідовність точок, які складають криву спіралі.

Навпаки, якщо проєкт реінжинірингу відбувається, значною мірою, у ручному режимі, то форма спіралі – витягнута (рис. 4.10), що ілюструє здороження вартості за рахунок вагомої складової інтелектуальної праці програмістів:

$$\text{if } \delta \to \min \therefore OM_{i \vee j} \to \max;$$

$$\lim_{\delta \to \min} \left\langle OM_{i \vee j}(\delta) \right\rangle = \max.$$

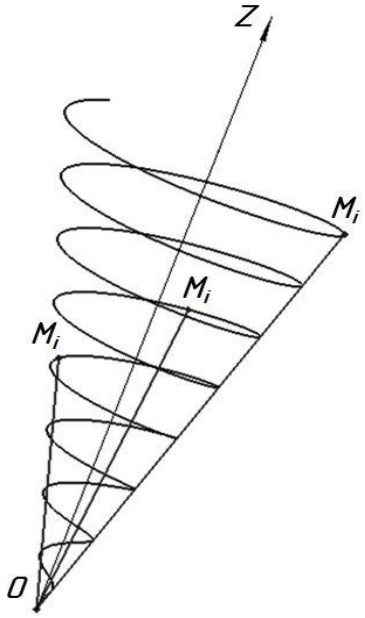


Рисунок 4.10 – Годограф із низьким коефіцієнтом автоматизації реінжинірингу ПС

4.3.3.2 Коефіцієнт схожості компонентів

Цей коефіцієнт залежить від ступеня впровадження КПВ. Для проведення порівняльного експерименту, візьмемо годограф (рис. 4.6): для нього коефіцієнт схожості компонентів (θ) буде базовим.

Чим більше в проєкті реінжинірингу застосовуються КПВ, тим густіше розміщуються витки спіралі (*n*) (рис. 4.11).

З представлення годографа бачимо, що:

$$\text{if } \theta \to \max \therefore OM_{i \vee j} \to \min \left| \left\{ M_{i \vee j} \middle| \theta \equiv M_{i \vee j} \middle| \theta_{\max} \right\} \right.;$$

$$\lim_{\theta \to \max} \left\langle OM_{i \vee j}(\theta) \right\rangle = \min.$$

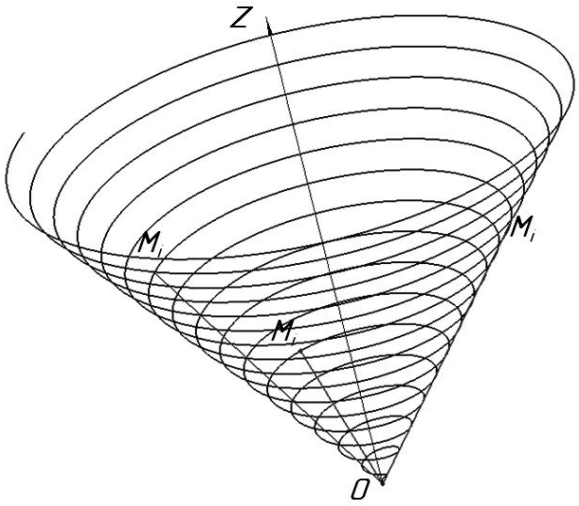


Рисунок 4.11 – Годограф проєкту реінжинірингу ПС із високим коефіцієнтом схожості компонентів

Тобто при збільшенні θ – зменшується вектор проєктних витрат OM_i при однаково узятих M_i, відповідно до випадків з θ та θ_{max} та навпаки: при використанні унікальних програмних компонентів – витки спіралі розміщуються нещільно (рис. 4.12), тобто збільшується OM_i при однаково узятих M_i, відповідно до випадків з θ та θ_{min}:

$$\text{if } \theta \to \min \therefore OM_{i \vee j} \to \max \left| \left\{ M_{i \vee j} \middle| \theta \equiv M_{i \vee j} \middle| \theta_{\min} \right\} \right.;$$

$$\lim_{\theta \to \min} \left\langle OM_{i \vee j}(\theta) \right\rangle = \max .$$

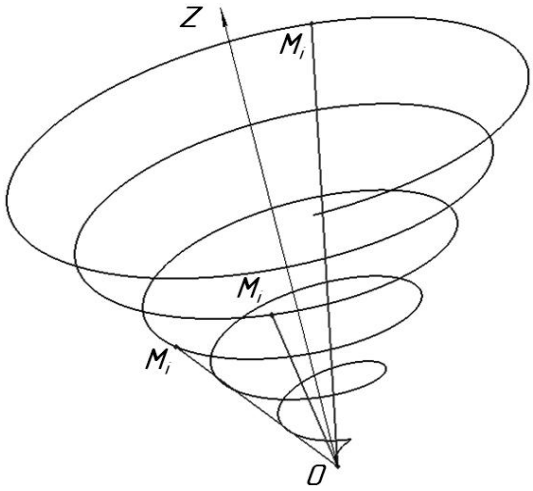


Рисунок 4.12 – Годограф проєкту реінжинірингу ПС із низьким коефіцієнтом схожості компонентів

4.3.4 Достовірність експериментальних результатів із дослідження методу представлення оцінки проєкту

Експеримент із побудови годографів проєкту реінжинірингу проводився із дванадцятьма ПС, деякі з яких узято з п. 4.2.4. За домовленістю, виконавцями проєктів, було надано початкові умови щодо кількості рядків програмного коду та фактичного часу повного виконання ними реінжинірингу ПС, який було зафіксовано у проєктних діаграмах Ґанта. Також цей час було ретельно перевірено за протоколами виконання та актами виконаних робіт.

За отриманими моделями було побудовано годографи проєкту реінжинірингу ПС, числовий матеріал яких подано у табл. 4.6. У таблиці наведено результати моделювання без використання проєктних коефіцієнтів – за ІМУПР, отриманими у підрозд. 4.2, та із уведенням описаних у цьому підрозділі коефіцієнтів. При виконанні проєктів обчислення вимірювання часу відбувається з округленням до півгодини.

Таблиця 4.6 – Час виконання проєктів реінжинірингу ПС

Номер проєкту	Час, год.		
	Модельний за ІМУПР	Модельний із проєктними коефіцієнтами	Фактичний
1	157	174	185
2	247	187,5	190
3	70	66,5	58,5
4	432,5	575	606
5	104,5	108,5	102,5
6	807	726	623,5
7	565	686	730,5
8	118	129	134
9	226	229,5	270
10	90	84	76,5
11	789	811	878
12	652	516	511

За даними табл. 4.6 побудована діаграма (рис. 4.13) порівняння фактичного часу (actual time) виконання проєкту реінжинірингу ПС та

змодельованих часів: за ІМУПР (idealized model) та із уведеними проєктними коефіцієнтами (with project coefficients).

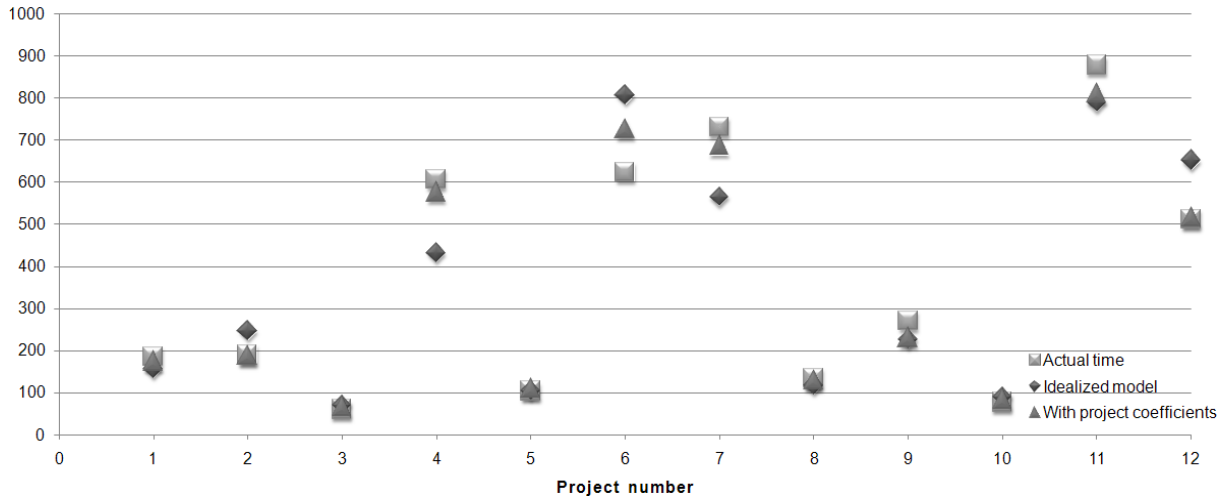


Рисунок 4.13 – Діаграма порівняння фактичного часу та змодельованих часів виконання проєкту реінжинірингу ПС

Статистичні результати порівняння зведено до табл. 4.7.

Таблиця 4.7 – Розбіжність порівняння за часом проєктів реінжинірингу ПС

Номер проєкту	Оцінка за ІМУПР		Оцінка за методом проєктних коефіцієнтів	
	Абсолютна похибка, год.	Відносна похибка, %	Абсолютна похибка, год.	Відносна похибка, %
1	28	15	11	6
2	57	30	2,5	1
3	11,5	20	8	14
4	173,5	29	31	5
5	2	2	6	6
6	183,5	29	102,5	16
7	165,5	23	44,5	6
8	16	12	5	4
9	44	16	40,5	15
10	13,5	18	7,5	10
11	89	10	67	8
12	141	28	5	1
Середнє значення, %:		19		8

4.3.5 Аналіз, обговорення та порівняння експериментальних результатів застосування методу

При аналізі та обговоренні результатів підрозділу, можна побачити, що метод представлення оцінки проєкту реінжинірингу ПС за допомогою проєктних коефіцієнтів, виходячи з табл. 4.7 та рис. 4.13, має значно меншу розбіжність з фактичним часом виконання проєкту реінжинірингу ПС ніж оцінка часу за ІМУПР.

Максимальне значення відносної похибки оцінки проєкту реінжинірингу ПС за методом проєктних коефіцієнтів не перевищує 16% (табл. 4.7), проти 30% при оцінці за ІМУПР. Середнє значення відносної похибки 8% за методом проєктних коефіцієнтів, проти 19% – за ІМУПР.

Для справедливості, необхідно обговорити, що п’ятий проєкт (табл. 4.6, 4.7) має оцінку ближчу до фактичної за ІМУПР ніж за методом проєктних коефіцієнтів (104,5 год. проти 108,5 год. при фактичному часі 102,5 год.) та відповідно меншу відносну похибку оцінки (2% проти 6%).

На думку автора, це пов’язано із помилковими даними (заниженням приблизно на 6 – 8 год.) фактичного часу виконання проєкту, який було надано розробниками, проте автор працював із вже готовими показниками виконання у вигляді діаграм Ґанта та не проводив корегування цих годин.

Що ж до наведених графічних побудов годографів проєкту реінжинірингу (рис. 4.9 – 4.12), то можна зробити висновки, що поверхня, для якою годограф $\Phi(r, \varphi, Z)$ є направляючою, являє собою лінійчату конічну поверхню, що може бути зворотно відновлена рухом OM_i, як утворюючої.

Крім того, у подальших дослідженнях, варто проаналізувати отримані лінійчату конічну поверхню на предмет практичного застосування наступної висунутої гіпотези: якщо обчислити її площу, то можна дістати кількісну характеристику виконаної роботи щодо здійснення проєкту реінжинірингу ПС:

$$S = \int_{0}^{\varphi_i(n+1)} \Phi\left(\rho, \varphi\right) d\varphi,$$

де S – площа лінійчатої конічної поверхні;

φ – час, протягом якого відбувається реінжиніринг;

ρ – радіус-вектор витрат;

Ф – функція конічної спіралі.

Якщо обчислити об’єм лінійчатої конічної поверхні,то можна визначити обсяг витрачених ресурсів:

$$V = \int_0^Z \Phi\left(\rho, \varphi, Z\right) dZ,$$

де V – об’єм лінійчатої конічної поверхні.

4.3.6 Узагальнення застосування та перспективи розвитку методу

Проєкт реінжинірингу виконується за допомогою комплексу засобів, у тому числі за рахунок застосування КПВ та CASE-систем. Згідно з оптимістичними прогнозами застосування КПВ здатне у 4 рази знизити вартість ПС у порівнянні з новою розробкою. Проте заради об’єктивності слід помітити, що у деяких випадках, все ж таки раціональніше застосувати нову розробку.

Для того, щоб прийняти мотивоване рішення щодо застосування або відмови від реінжинірингу, згідно із задачею підрозділу, було представлено метод проєктної оцінки реінжинірингу ПС, на підставі якого формується остаточний висновок щодо доцільності виконання проєкту реінжинірингу.

Згідно з цим методом до ІМУПР ПС, що були сформовані у попередньому підрозділі, уводяться проєктні коефіцієнти, що дозволяють підвищити точність оцінки реінжинірингу, яка виражена у зменшенні середнього значення відносної похибки щодо фактичного часу виконання реінжинірингу з 19% (при побудові за ІМУПР) до 8%.

4.4 Висновки за розділом

У розділі вирішено три встановлених завдання, які включали розробку комплексу моделей та методів оцінки проєкту реінжинірингу ПС. УП реінжинірингу, які буде виконано за допомогою розроблених моделей та методів дозволить не тільки скоротити витрати на перепроєктування ПС, але й підвищити ефективність технічного супроводу проєкту, збільшити ЖЦ ПП, що вже знаходяться у експлуатації та подолати

протиріччя між швидкими темпами розвитку науки, техніки та УП з розвитку ПС та ПП.

За результатами одного з завдань: удосконалено метод розрахунку показників проєкту за проєктними точками Карнера, який відрізняється внесенням доповнень та розширень щодо процесів реінжинірингу ПС, що дозволяє зменшити похибку у оцінюванні прогнозованого переліку витрат на реалізацію проєкту на 17 – 19%, у порівнянні із методом Карнера. Встановлення коефіцієнтів та обрання значення констант засновані на методі Якобсона та перевірені статистикою найбільш схожих проєктів.

Після аналізу завершеного проєкту і вивчення звіту про показники, доступні чинники можуть бути точно відкориговані, щоб дати оцінку часу, необхідного на виконання проєкту реінжинірингу. Згодом, можна використовувати ці дані у якості базової траєкторії ЖЦ проєкту. Вимірювання виконуються командою досвідчених системних аналітиків, що спираються на власні уявлення про технічну складність проєкту та можливості команди програмістів. Коефіцієнти та константи визначаються, виходячи із статистичних даних 3-х – 7-ми вже оцінених аналітиками виконаних проєктів із близьким ступенем схожості.

Крім того, у розділі монографії вирішено наступне завдання: сформовано ідеалізовані моделі управління процесом комплексної оцінки проєкту реінжинірингу ПС, які відрізняються уведенням ресурсних та часових обмежень на виконання проєкту, що дозволяє підвищити точність оцінки проєктних витрат з перепроєктування ПС. Виконано формування представлення оцінки параметрів витрат ресурсів на виконання проєкту реінжинірингу ПС за допомогою математичного інструментарію опису моделей проєктування. Запропоновані ІМУПР видів забезпечення ПС являють собою еволюційні спіралі, які побудовані у циліндричній системі координат.

Експериментальні дослідження спіральної ІМУПР за даними 7 виконаних проєктів показали прогнозування витрат із максимальною похибкою 30%. Середнє значення відносної похибки – 19%. Операції із поданими моделями можуть відбуватися у проекціях часу та витрат, у ізометричній проекції програмних компонентів, у логарифмічній проекції рядків програмного коду.

Також у розділі вирішено завдання розробки методу представлення оцінки проєкту реінжинірингу ПС, який відрізняється застосуванням інструментарію проєктних коефіцієнтів, що дозволяє управляти якісними

змінами конфігурації графічних моделей перепроєктування, у вигляді відповідних годографів проєктів реінжинірингу ПС. Максимальне значення відносної похибки оцінки проєкту реінжинірингу ПС за методом проєктних коефіцієнтів не перевищує 16%. Середнє значення відносної похибки – 8%.

Практичні результати розділу монографії: формалізовано критерії оцінки та визначення складності проєкту реінжинірингу вже готової ПС, але яка потребує проактивного розвитку із плином часу. На підставі розрахованих показників оцінювання проєкту формується звіт, який містить аналіз оцінки часу розробки, наявності визначених вмінь у стейкхолдерів та вартості реінжинірингу ПС.

При використанні методу розрахунку показників проєкту за проєктними точками та методу представлення оцінки проєкту з реінжинірингу ПС за допомогою інструментарію проєктних коефіцієнтів, виконано оцінку кожного етапу проєкту з удосконалення ПС у Одеському ім. А. Міцкевича відділенні Спілки поляків в Україні із отриманням економічного показника впровадження у вигляді оцінки вартості кожного окремого етапу проєкту із удосконалення ПС, які не відрізняються від фактичних більш ніж на 7 %.

Сформовані у розділі результати, з наукової точки зору, увійдуть до основ методології проактивного УП з розвитку ПС та ПП, а з практичної – стануть у нагоді системним програмістам, які задіяні у перепроєктуванні й переробці ПС та ПП, а також системним архітекторам, які працюють із мультимовними надбудовами ПС, що вже знаходяться у кількарічній експлуатації і набувають еволюційного розвитку із плином часу проєкту та удосконалення у процесі використання.

Подальші наукові дослідження у цьому напрямку потребують всебічного дослідження висунутої автором гіпотези щодо представлення кількісних характеристик виконаної роботи з проєкту реінжинірингу ПС та визначення обсягів витрачених ресурсів у формі обчислення площин та об’ємів лінійчатої конічної поверхні, що утворюються годографами проєкту реінжинірингу.

Подальший розвиток з досліджень метода розрахунку показників проєкту за проєктними точками: управління автоматизованим формуванням звіту з оцінки ПС, який включався би у вихідну проєктну документацію, необхідну для управління виконанням реінжинірингу кожного з видів забезпечення проєкту: технічного, математичного,

інформаційного, лінгвістичного, методичного, організаційного тощо. Ця проєктна документація є обов'язковою складовою частиною видів забезпечення ПС.

Основні власні наукові дослідження, що подані автором у четвертому розділі монографії, опубліковано у наукових працях автора: [7] – [9], [23], [24], [33], [34] – [39], [44], [49], [50], [60] – [61], [66], [67], [69].

5 АНАЛІЗ ДОСТОВІРНОСТІ НОВИХ ЗНАНЬ ТА УЗАГАЛЬНЕННЯ ПРАКТИЧНО-ОРІЄНТОВАНИХ РЕЗУЛЬТАТІВ КОМПЛЕКСНИХ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ З ПРОАКТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ З РОЗВИТКУ ПРОГРАМНИХ СИСТЕМ

У розділі розглядається достовірність нових знань з перепроєктування ПС та ПП різного галузевого призначення. Відмінною особливістю отриманих практично-орієнтованих результатів є можливість підтримки роботи більше десяти найпопулярніших МП. При застосуванні способу вдається автоматизувати процес перекодування компонентів ПЗ та, за рахунок цього, вивільнити робочий час програмістів від рутинного перепрограмування, зменшити вірогідність виникнення структурних помилок, що успадковуються від попередньої ПС, та знизити вартість проєкту реінжинірингу ПС.

5.1 Спосіб перекодування програмної системи за умов невизначеності розвитку проєкту

Актуальність задачі виходить з перспектив розробки способу реінжинірингу ПС, як достовірного засобу отримання нового компонента шляхом виконання послідовності операцій внесення змін, модернізації або модифікації, а також перепрограмування окремих компонентів ПС. Цей спосіб повинен містити у собі процеси реорганізації та реструктуризації системи, переведення окремих компонентів системи в іншу, сучаснішу мову програмування, а також процеси модифікації або модернізації структури і системи даних.

До причин, що перешкоджають найскорішому використанню способу, можна віднести відсутність опису у проєкті шляхів відновлення та успадкування програмної архітектури системи, що перепроєктується.

Високої ефективності в УП, яка виражається, перш за все, через мінімізацією часових, а, відповідно, й матеріальних витрат при експлуатації ПС, можна домогтися за рахунок удосконалення її ПЗ.

В основу розділу поставлено задачу розробки способу реінжинірингу ПС шляхом еволюції позитивних змін її ПЗ, що забезпечить підвищення зручності експлуатації, супроводу та використання, надасть можливості

розширення за допомогою підключення розроблених та / або удосконалених видів забезпечення ПС.

5.1.1 Формулювання способу у межах зовнішнього оточення об’єкта

UML, як мова графічного опису для об'єктного моделювання [204], окрім простого проєктування, підтримує ще й функцію генерації та реінжинірингу коду на основі даних моделей. Найважливішою дією для створення скелетної структури та реінжинірингу є створення двох структурних моделей [195]: моделі операцій із проєктними класами та моделі компонентних рішень. Саме на їх основі генерується код майбутнього ПП [198]. Всі інші моделі (див. розд. 3.2) мають допоміжну роль в УП з перекодування ПС [196] і використовувати їх потрібно тільки на свій розсуд.

Перевірка достовірності встановленої задачі досягається тим, що спосіб перекодування ПС можна звести до визначеної наведеної послідовності, яка включає виконання нижченаведених сформульованих проєктних дій.

1) Обрання кінцевої мови програмування.

Виконання цієї дії залежить від технічного завдання або сучасних вимог ринку, зведених у так званий топ-список найпопулярніших МП, до яких належать: C; C++; C#; Java; PHP; Delphi; Python; Visual Basic тощо.

2) Вибір оптимального CASE-засобу.

Вибір CASE-засобу залежить від уподобань користувача. На думку автора монографії, найоптимальнішим CASE-засобом, що підтримує імпортування та генерацію коду, написаного мовами, наведеними вище, є Enterprise Architect (EA). Саме EA (версія 14.0) будемо розглядати як ефективний інструмент перекодування.

Виходячи із поставленої задачі, в обраному середовищі моделювання повинні були бути присутніми багато можливостей, крім стандартного набору діаграмних моделей (15 штук), необхідно проводити аналіз ефективності, а це бізнес-діаграми, діаграми часу тощо.

Наприклад, у одному із найвідоміших середовищ, Rational Rose, бізнес діаграми та всі наступні економічні діаграми реалізовані не

---

204. Робинсон С., Корнз О., Глинн Дж. C# для профессионалов (комплект из 2 книг). Москва: Лори, 2005. 1000 с.

ефективно [205]. На даний момент існує обмежена кількість систем моделювання, які підтримують коректну генерацію коду багатьма мовами, особливо якщо врахувати не самі використовувані мови, хоча у поданій монографії ці мови не знадобляться, але ПП, який розвивається у даному напрямку має найкращі перспективи для досліджень. Так само важливі були зручності в роботі і простота інтерфейсу. Звичайно в плані простоти інтерфейсу EA відстає від своїх аналогів, але це перекривається його ефективністю [206].

3) Аналіз ПП.

Для підтвердження достовірності повноцінного розбору на компоненти обрано вже відомий (розд. 3) відкритий ПП BRL-CAD із конкретними перевагами [207]. Загальна тенденція в усе більш широкому використанні ПП з відкритим кодом сьогодні не викликає сумнівів. Проте, спочатку, стало дивним, що збройні сили США відкрили свою, на той час, секретну розробку [208] і пізніше, через 20 років, зробили його Open Source.

Виявилося, що переведення проєкту BRL-CAD у вільний режим було економічно вигідно. Весь розвиток і тестування ведеться за рахунок розробників по всій країні та за її межами, за відхиленнями і модифікаціями стежать збройні сили США, а у разі коли необхідна секретність, завжди є можливість підключити додатковий модуль, який буде захищати розробку.

Цей приклад американських військових показує, що підхід «Open Source» дає можливість, практично не вкладаючи коштів, контролювати розвиток проєкту у потрібному напрямку складних програмних комплексів, по суті, диктуючи правила «гри» іншим стейкхолдерам. Достовірність цієї концепції перевірено та вона може бути взята на озброєння й вітчизняними держзамовниками [209]. Із причин, створення

---

205. Боггс У., Боггс М. UML и Rational Rose. Санкт-Петербург: Лори, 2008. 600 с.

206. Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. Москва: Символ-Плюс, 2011. 192 с.

207. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4. Москва: Вильямс, 2011. 1392 с.

208. Колесов Ю. Б., Сениченков Ю. Б. Моделирование систем. Объектно-ориентированный подход. Санкт-Петербург: БХВ-Петербург, 2006. 192 с.

209. Воройский Ф. С. Основы проектирования автоматизированных библиотечно-информационных систем. Санкт-Петербург: ФИЗМАТЛИТ, 2002. 384 с.

вітчизняних «Open Source» проєктів та підтримка команд й розробників, є вигідною.

4) Ідентифікація компонентів.

Далі необхідно визначитись, який саме компонент ПС необхідно піддати перекодуванню. Для прикладу розглянемо обрання будь-якого відкритого компонента, написаного мовою С (рис. 5.1). Нехай це буде компонент «type.h» (див. рис. 5.1).

5) Імпорт компонентів.

Імпорт компонентів вже виконаємо за допомогою ЕА, де послідовно після встановлення необхідних параметрів, оберемо:

Code → Source Code → Import Code → C Files…

Так само можна виконати імпорт будь-якого компонента, написаного будь-якою мовою із переліку мов, що випадає зі списку, які включено до ЕА. Якщо імпорт компонентів було виконано правильно, то після зворотного інжинірингу імпортованого коду з’явиться відновлена структура первісного компонента (рис. 5.2).

6) Генерація коду нової структури.

Для виконання генерації коду після встановлення необхідних параметрів ЕА, до яких входить виділення потрібних структурних елементів, необхідно виконати наступні дії із сформованою структурою:

Source Code Engineering → Generate Current Element…

Після чого з’явиться запит – «Генерація коду» (рис. 5.3), у якому треба вказати місце для збереження нового компонента, перекодованого вже новою мовою.

Далі вже обираємо мову, в яку буде перекодований компонент, наприклад: Java (рис. 5.3) та натискаємо кнопку «Generate». Після цього виконується процес перекодування, за яким можна спостерігати за допомогою повідомлень декількох системних вікон (рис. 5.4). Якщо все виконано правильно, у вказаному місці з’явиться перекодований необхідною мовою компонент ПС.

7) Редагування перекодованого компонента.

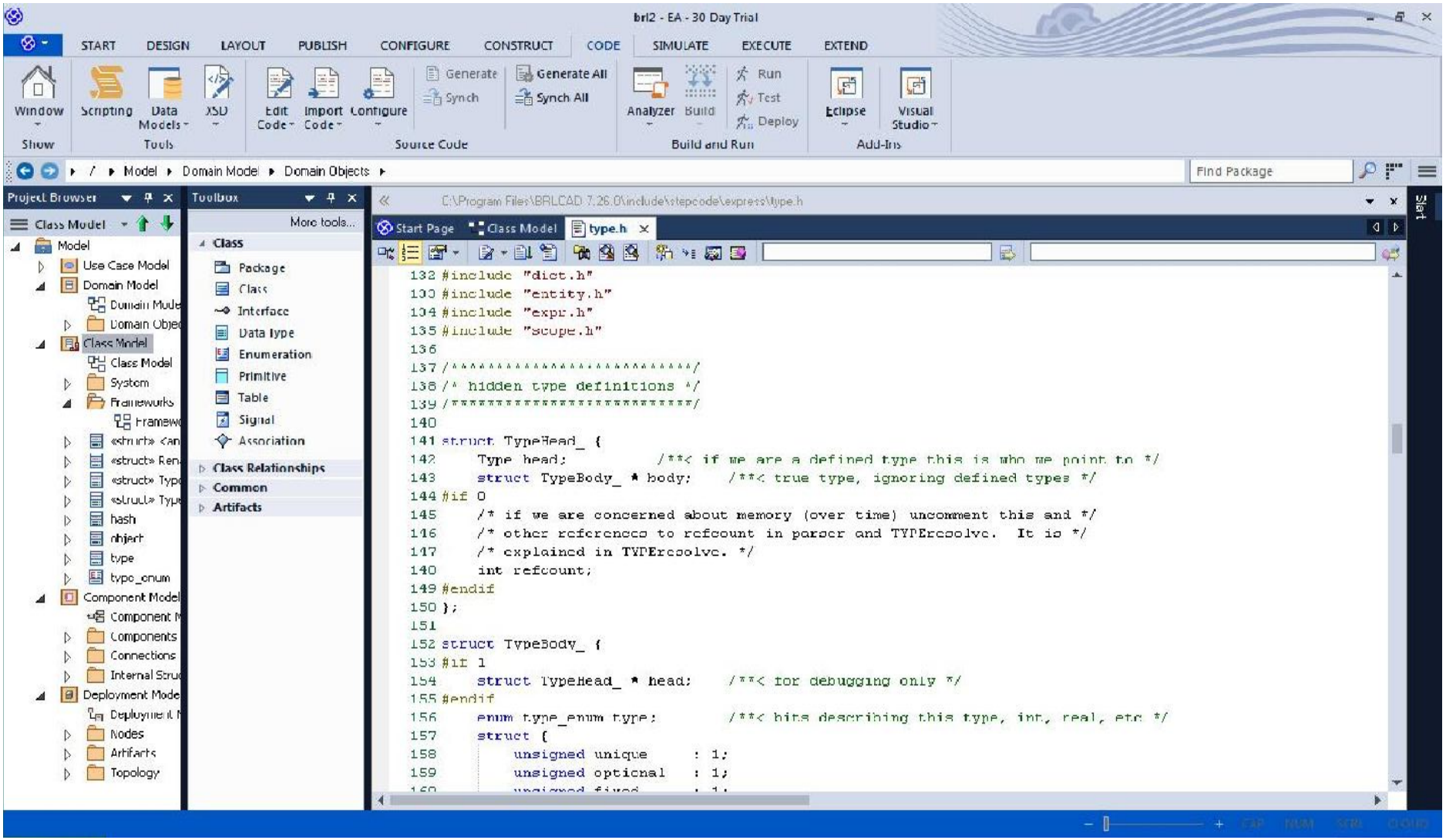


Рисунок 5.1 – Представлення компонента за допомогою вбудованого до EA редактора коду

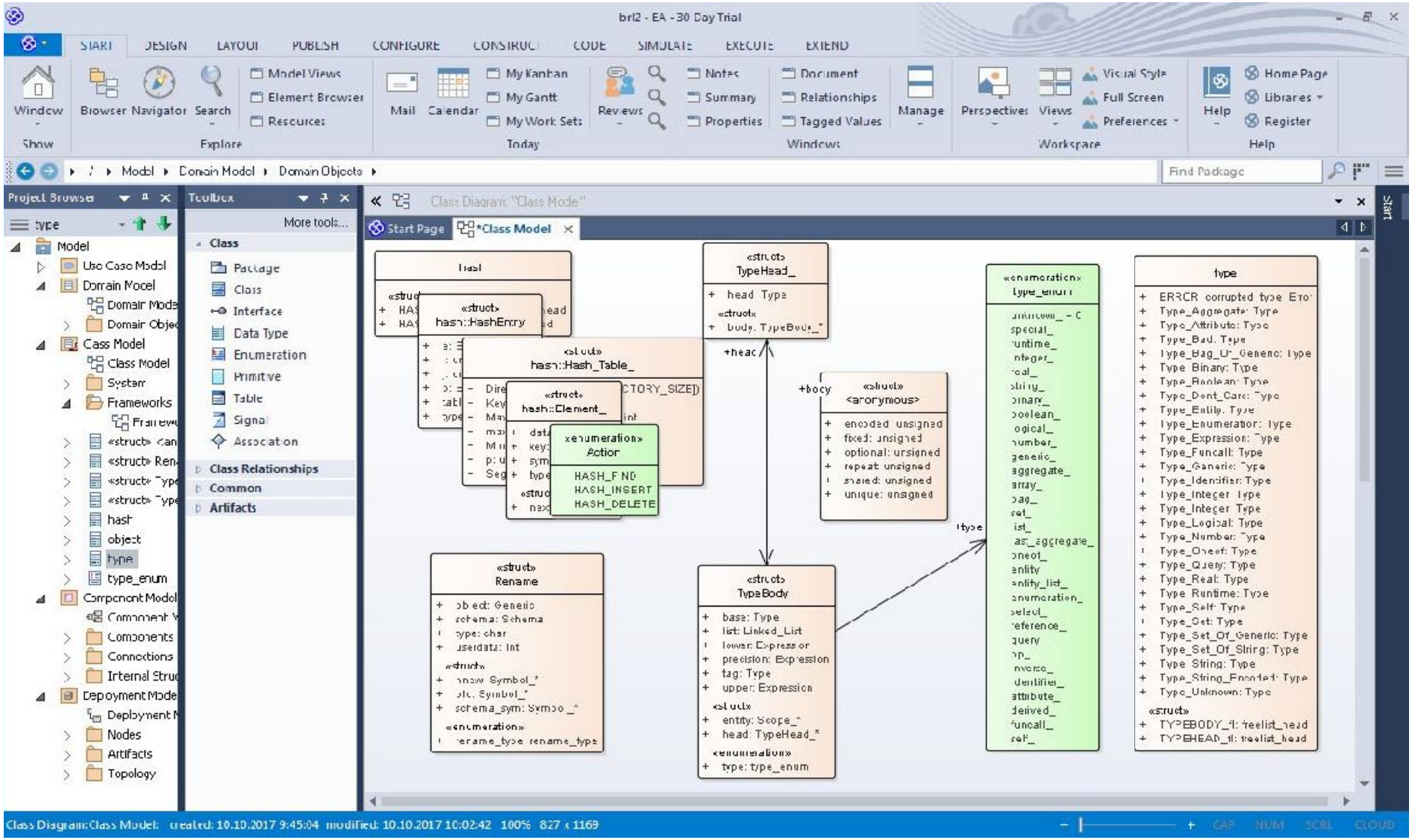


Рисунок 5.2 – Відновлена структура імпортованого компонента

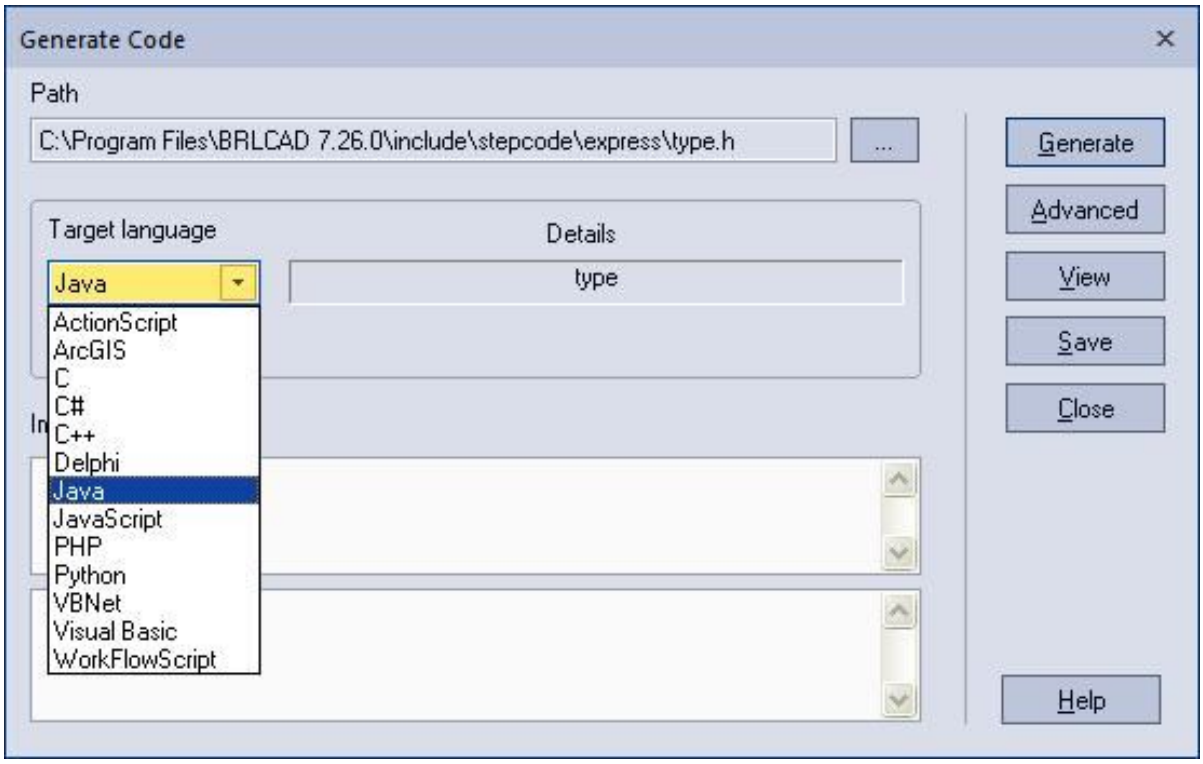


Рисунок 5.3 – Приклад запиту «Генерація коду»

Цей компонент ПС може бути доступний для подальшого редагування або підключення до нового проєкту як за допомогою EA, так і за допомогою будь-якого редактора, орієнтованого на вже нову перекодовану мову.

5.2 Аналіз імпортованої моделі операцій із проєктними класами

Поняття «клас» є основним будівельним блоком для більшої частини сучасних програмних засобів [197]. Клас є основною ООП [210] та уніфікованої мови моделювання – UML [199] і оскільки між ними існує відповідність, то генерація коду і реінжиніринг відбуваються саме завдяки цій моделі. Якщо у кожній МП зовнішній вигляд класу змінюється і структура може зазнавати незначних змін [211], то в моделях UML клас має фіксовану структуру – це прямокутник поділений на три області (рис

210. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. Москва: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2006. 320 с.

211. Воройский Ф. С. Информатика. Новый систематизированный толковый словарь. Москва: ФИЗМАТЛИТ, 2003. 760 с.

5.5). У верхній міститься назва класу, в середній – опис атрибутів (властивостей), в нижній – назви операцій – послуг, надаваних об'єктами цього класу.

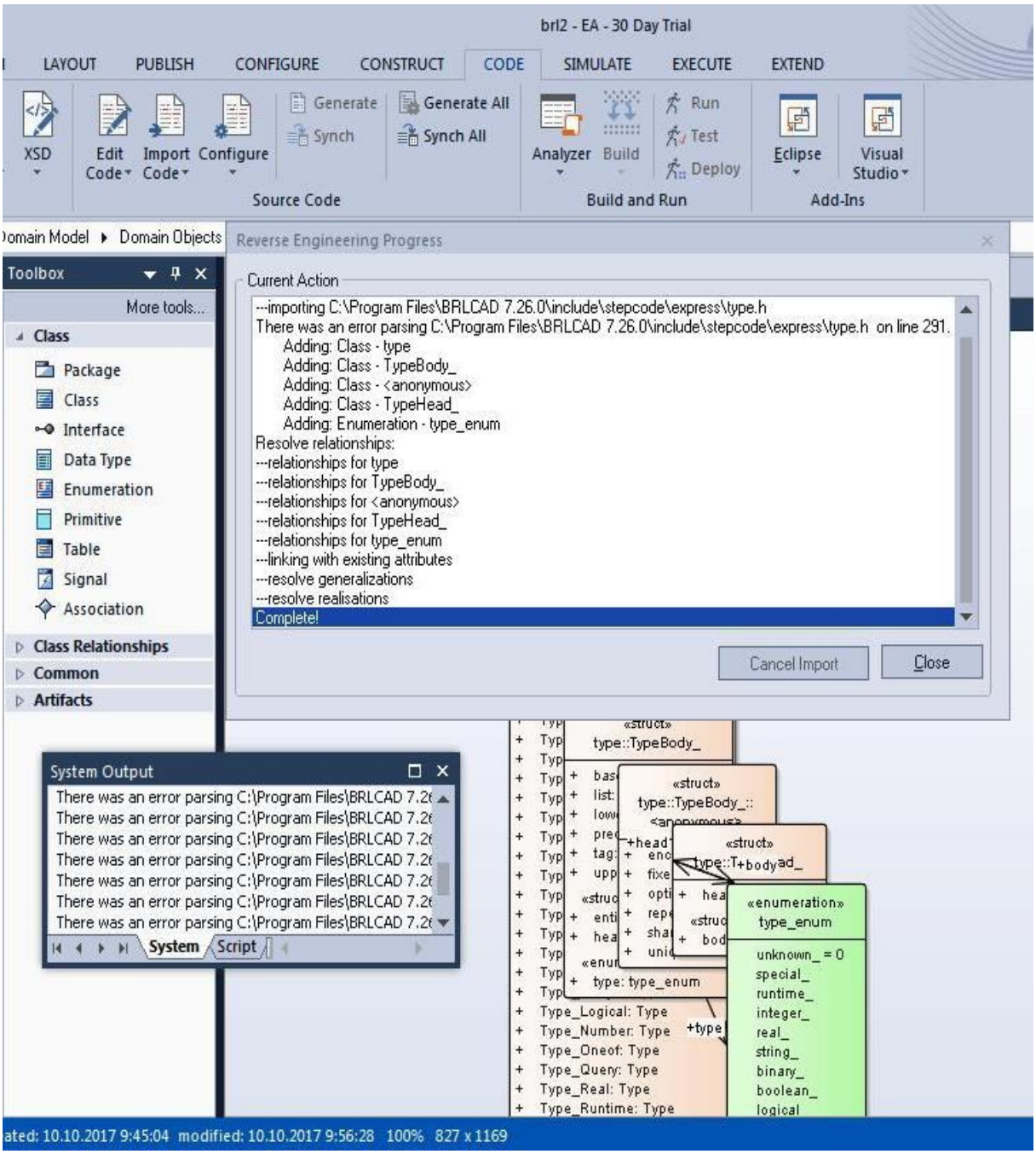


Рисунок 5.4 – Представлення процесу перекодування за допомогою системних повідомлень

Атрибути класу показують склад і структуру даних, які зберігаються в даному об'єкті [212]. Кожен з цих атрибутів має ім'я і тип даних, які він надає. Коли об'єкт реалізується, то під нього виділяється місце для зберігання всіх його атрибутів. Кожен атрибут буде мати певне значення в будь-який момент виконання програми.

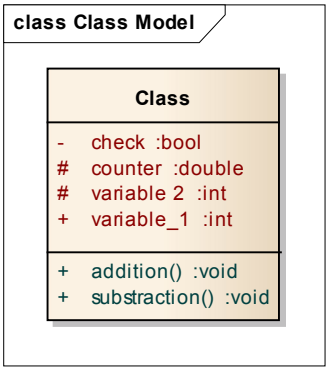


Рисунок 5.5 – Компонент моделі класів в нотації UML

Для кожного атрибута класу можна задати його видимість [213]. Цей параметр визначає видимість атрибута класу для інших класів. У нотації UML використовують 3 типи видимості об'єктів:

а) відкритий (публічний) – атрибут доступний і може використовуватися іншими класами МОПК;

б) захищений – атрибут доступний тільки нащадкам даного класу, нащадки призначаються за допомогою зв'язків, про що буде розказано нижче;

в) закритий (приватний) – атрибут може використовуватися тільки класом, в якому знаходиться – зовнішні об'єкти не бачать даний атрибут.

Зручність закритих атрибутів полягає в тому, що це гарантія відсутності несанкціонованого доступу до даних атрибута. Унаслідок цього зменшується кількість можливих помилок і ПС працює стабільно.

У МОПК зазвичай використовують два основних типи відношення – асоціація і узагальнення. Кожна асоціація несе інформацію про зв'язки між

---

212. Новиков Ф. А., Иванов Д. Ю. Моделирование на UML. Теория, практика, видеокурс (+2 DVD-ROM). Москва: Профессиональная литература; Наука и техника, 2010. 640 с.

213. Путилин А. Б., Юрагов Е. А. Компонентное моделирование и программирование на языке UML. Практическое руководство по проектированию информационно-измерительных систем. Москва: НТ Пресс, 2005. 664 с.

об'єктами всередині ПЗ. Найбільш часто використовуються бінарні асоціації, що зв'язують два класи. Асоціація може мати назву, яка має виражати суть відображуваного зв'язку (рис. 5.6).

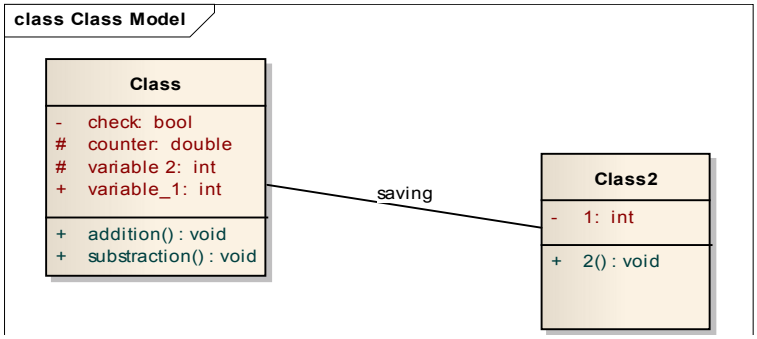


Рисунок 5.6 – Компоненти МОПК з асоціативним типом відносин

Узагальнення у МОПК використовується, щоб показати зв'язок між батьківським класом та класом-нащадком [214]. Воно вводиться до МОПК, коли виникає різновид якогось класу, а також у тих випадках, коли в системі виявляються кілька класів, що мають схожу поведінку [215] – в цьому випадку загальні елементи поведінки виносяться на більш високий рівень, утворюючи батьківські класи. На рис. 5.7 ми бачимо три незалежних один від одного класи.

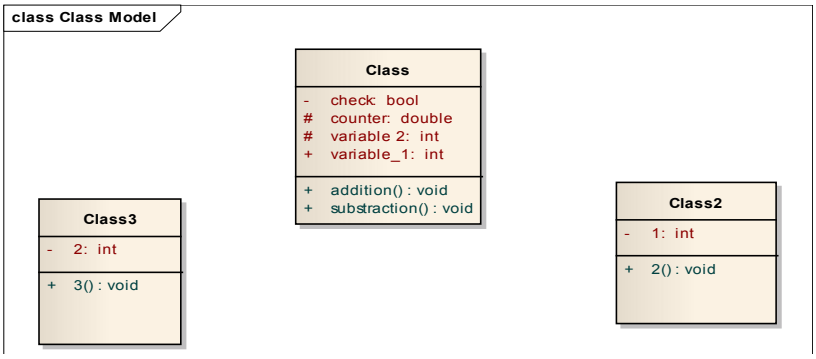


Рисунок 5.7 – Незалежні класи

214. Киммел П. UML. Универсальный язык программирования. Москва: НТ Пресс, 2008. 272 с.

215. Леоненков А. В. Самоучитель UML 2. Москва: БХВ-Петербург, 2007. 576 с.

Як тільки ми додаємо узагальнюючі відношення між класами «Class» і «Class 2» (роль батьківського класу виконує «Class»), то отримаємо повідомлення (рис. 5.8).

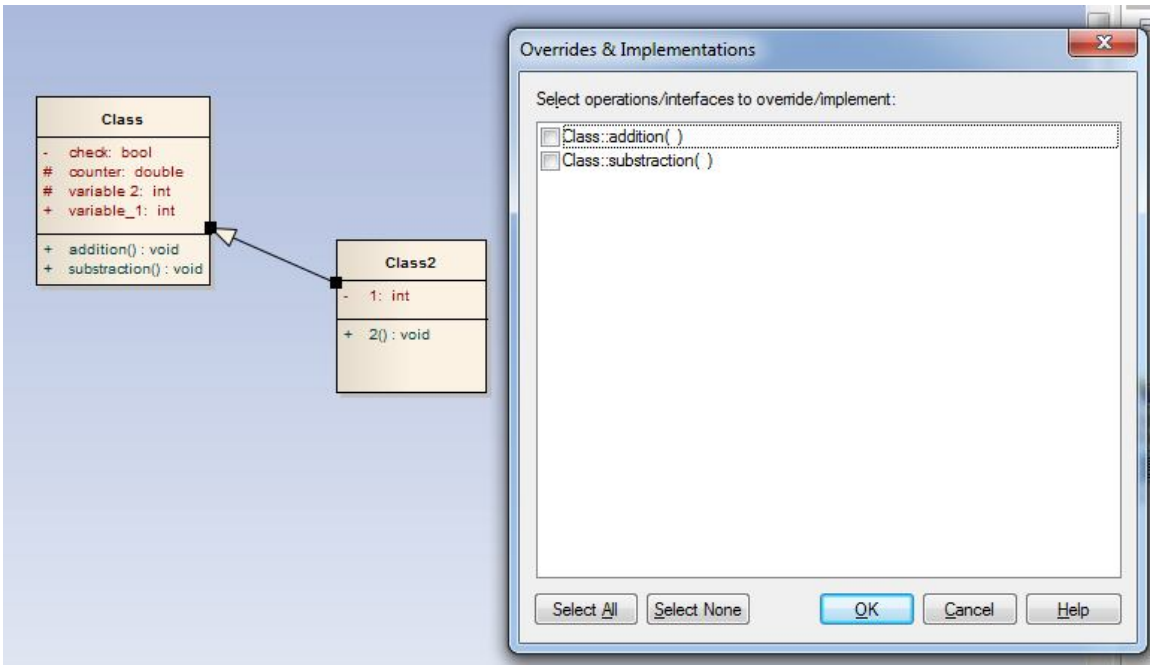


Рисунок 5.8 – Налаштування відношення між класами

Пропонується вибрати атрибут, який буде прив'язаний (виконуватися) дочірнім класом. Виберемо атрибут з назвою «Addition» (рис. 5.9).

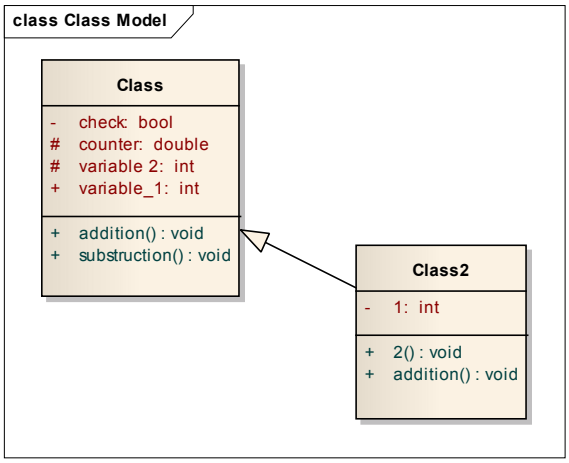


Рисунок 5.9 – Результат налаштування

З рис. 5.9 бачимо, що дочірній клас тепер виконує операцію, яку виконує і батьківський клас. Це був прямий приклад наслідування властивостей, що є одним із важливих для роботи з МОПК. Точно так само можуть успадковуватися й закриті (приватні) операції.

5.3 Аналіз імпортованої моделі компонентних рішень

МКР дозволяє нам описати особливості структури фізичного розташування елементів в ПП, що розроблюється [216]. Фізичне розташування допомагає розібратися в архітектурі системи, встановити зв'язок між фізичними і логічними компонентами системи, програмними компонентами [217]. Два основних види використовуваних елементів у МКР – компоненти й інтерфейси. Для розробки МКР, крім архітекторів, залучаються також програмісти й аналітики – це потрібно для того, щоб перехід між логічною моделлю та конкретним програмним кодом відбувався плавно й забезпечував меншу кількість виправлень в роботі.

МКР формується для таких цілей:

а) візуалізація загальної структури вихідного коду ПС;

б) специфікація ВВ ПС;

в) забезпечення багаторазового використання окремих фрагментів програмного коду;

д) представлення концептуальної та фізичної схем БД.

Деякі елементи МКР можуть бути активні тільки у певний період часу, наприклад, під час компіляції програмного коду.

Компоненти є фізичним представленням сутностей, вони є реалізацією деякого набору інтерфейсів та служать узагальненням для логічної робочої системи. Важлива особливість для розуміння МКР полягає у найменуванні компонентів, яке складається з двох частин: ім'я компонента (може складатися з букв та цифр) та представлення компонента на рівні типу або екземпляра (рис. 5.10), який визначає, чим може бути компонент:

а) динамічна бібліотека (розширення *.dll);

б) Web-сторінка (розширення *.html);

---

216. Бабич А. В. UML. Первое знакомство. Пособие для подготовки к сдаче теста UMO-100 (OMG Certified UML Professional Fundamental) (+CD-ROM). Санкт-Петербург: Бином. Лаборатория базовых знаний, 2008. 176 с.

217. Гросс К. C# 2008 и платформа .NET 3.5 Framework. Санкт-Петербург: Вильямс, 2009. 480 с.

в) текстовий файл (розширення *.txt або *.doc);

д) файл довідки (*.hip);

е) файл БД (*.db);

ж) файл з початковими текстами програм (*.h, *.cpp – для МП C++; *.java для МП Java);

и) скріптами (*.pi, *.asp) тощо.

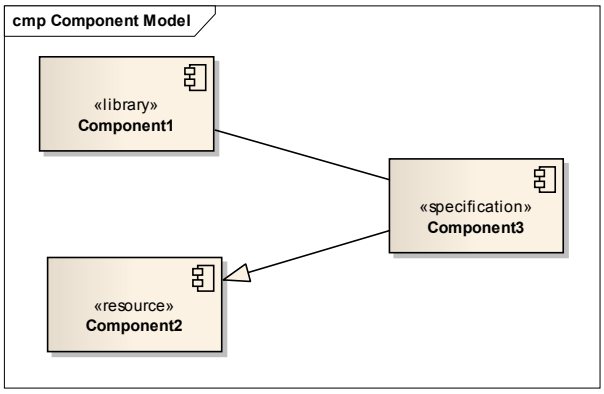


Рисунок 5.10 – Приклад типізованих компонентів

Оскільки компоненти являють собою фізичний модуль реалізації якоїсь частини коду, то для зручності часто використовують додаткове коментування, ілюструють конкретні особливості реалізації даного модуля. Коментарі не специфіковані в UML, але їх використання все одно конкретно спрощує роботу для персоналу з різних галузей.

Існує два види специфікації компонентів у МКР. Перший поділяється на три підвиди:

а) розгортання, які забезпечують безпосереднє виконання ПС своїх функцій: такими компонентами можуть бути спільні бібліотеки з розширенням *.dll, Web-сторінки мовою розмітки гіпертексту з розширенням *.html та файли довідки з розширенням *.hlp;

б) файли, що містять вихідний код продукту – можуть бути написані будь-якою МП, приклад розширення: *.dll, Web-сторінки мовою розмітки гіпертексту з розширенням *.html та файли довідки з розширенням *.hlp;

в) файли виконання – мають розширення *.exe та є інтерфейсом ПП.

Перелічені вище елементи називають артефактами [218]. У це поняття вкладається той сенс, що є закінчений інформаційний зміст, який запротокольовано та залежить від конкретної технології реалізації.

Іншим способом специфікації різних видів компонентів є явна вказівка його стереотипу компоненту перед ім'ям. Для компонентів визначені такі стереотипи:

а) бібліотека (library) – визначає перший різновид компонента, який представляється у формі динамічної або статичної бібліотеки;

б) таблиця (table) – також визначає перший різновид компонента, який представляється у формі таблиці бази даних;

в) файл (file) – визначає другий різновид компонента, який представляється у вигляді файлів з вихідними текстами програм;

д) документ (document) – визначає другий різновид компонента, який представляється у формі документа;

е) той, що виконується (executable) – визначає третій вид компонента, який може виконуватися у вузлі.

Наступним важливим компонентом МКР є інтерфейс [219]. Він представляється у МКР у вигляді прямокутника із написом «Interface» над ім'ям інтерфейсу і має секції, що призначено під атрибути та операції. Це зручно для представлення його внутрішньої структури та подальшої реалізації.

Зв'язок між фізичними і логічними компонентами у ПС здійснюється через з'єднання двох видів моделей – МОПК та МКР (рис. 5.11).

Після завершення побудови МОПК, необхідно пов'язати кожен клас з компонентом або інтерфейсом на МКР. Один елемент МКР може містити в собі множина елементів МОПК; класи як такі звичайно належать лише до одного компонента, а зв'язок з іншими відбувається за допомогою передачі інформаційних пакетів, що передаються відповідно до МОПК. Зв'язки у МКР існують, щоб показати тільки загальне розташування і тип зв'язку, хоча навіть у самій моделі типи зв'язків можна уточнювати.

При побудові великих і складних моделей зручно використовувати такий тип модулів як «Пакет». По суті «Пакет» – це лише умовне позначення, що показує: на яких рівнях або у яких папках знаходиться

---

218. Максимчук Р. А., Нейбург Э. Дж. UML для простых смертных. Москва: Лори, 2008. 304 с.

219. Дей Н., Мандел Л., Райман А. Eclipse. Платформа Web-инструментов. Санкт-Петербург: КУДИЦ-Пресс, 2008. 688 с.

потрібний клас або інший вид елементів. Так само вони спрощують роботу при складанні ієрархічних схем. Пакет зображується у вигляді великого прямокутника з невеликим прямокутником, приєднаним до лівої частини верхньої сторони першого (рис. 5.12).

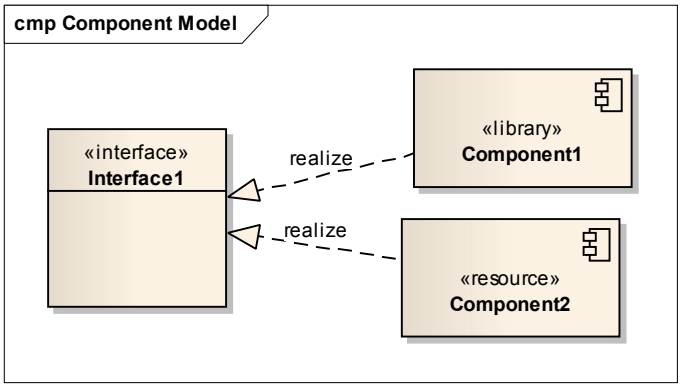


Рисунок 5.11 – Інтерфейс і компоненти

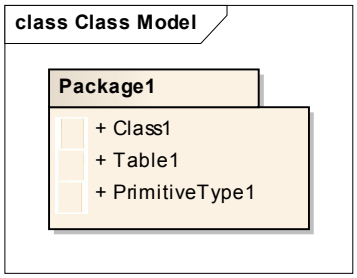


Рисунок 5.12 – Приклад пакета з елементами

Зручність управління за допомогою пакетів очевидна – візуалізація дозволяє переглянути кількість і тип елементів, що включені до пакету, але так само існують і недоліки, що пов'язані з проєктуванням.

На рис. 5.13 наведено два класи, що пов'язані між собою, поруч знаходиться пакет із третім класом. З практики ООП [199] відомо, що можна задавати параметри успадкування із будь-яким класом, який знаходиться поруч. Якщо прибрати клас-спадкоємець з кореневого каталогу, де знаходиться батьківський клас, то всі успадковані властивості зникнуть. Точно так само і при роботі з пакетами – задавати зв'язки між елементами МОПК можна тільки в межах одного пакета. Задавати спадкування між повноцінними пакетами теж не можна, окрім деяких видів зв'язків за типом «використання» або «передачі інформації».

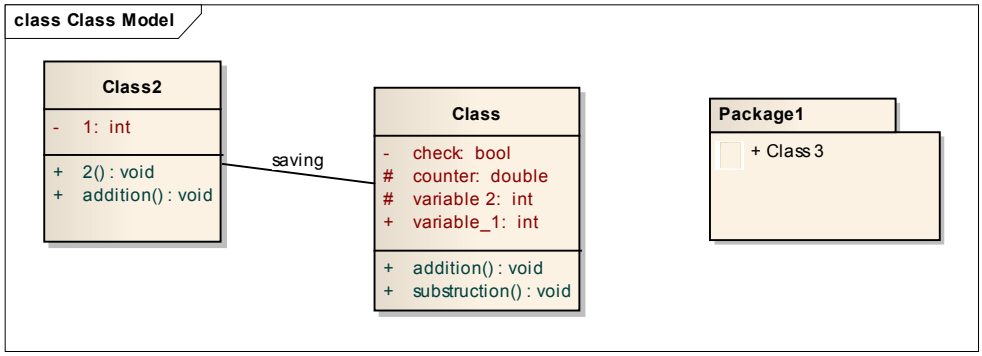


Рисунок 5.13 – Приклад пов’язаних класів та пакета із класом

Виходячи з наведеного вище стає зрозуміло, що пакети зручно використовувати при управлінні широкомасштабними проєктами, де є багато практично незалежних один від одного модулів або ж для візуалізації ієрархії процесу або структури ПС.

5.4 Засоби управління процесом генерації коду

МКР служить для нас зручною ланкою при з'єднанні класів і цілих пакетів, які складаються з подібних модулів [220]. У CASE-засобі Enterprise Architect МКР не має прямого впливу на відтворення коду з моделі, вона виконує лише допоміжні функції. Дуже показово саме те, що при створенні складної ПС відтворювати окрему МОПК не зручно, тому подальша прив'язка до МКР складається саме з перенесення модулів типу «Клас» в цю модель.

У ПП EA існує такий зручний тип компонентів, який називається «Packaging Component» – цей компонент має широку внутрішню структуру у вигляді ще однієї діаграмної моделі. Ця внутрішня діаграма створювалася саме для зручності роботи з модулями типу «Клас», але можливості EA дозволяють нам створювати там діаграми будь-якого типу (при детальному розборі методології – це виявилося зручно тому, що можна, наприклад, на основі блокнота показати структуру або методологію бізнес-моделі).

При створенні даного компонента всередині нього автоматично з'являється проєкт нової МКР, якою зручно управляти та завантажувати до

220. Ларман К. Применение UML и шаблонов проектирования. 2-е изд.: пер. с англ. Москва: Издательский дом «Вильямс», 2004. 624 с.

неї модулі класів. Усі модулі, які будуть створені або перенесені до цієї моделі, автоматично прив'язуються до компонента, у якому розташовуються.

Для генерації коду відкриємо фізичне розташування модулів типу «Клас» на екрані та після цього обираємо усі потрібні модулі, які необхідно відтворити. Далі вказується місце розташування для кожного створюваного елемента та покроково виробляється генерація (рис. 5.14).

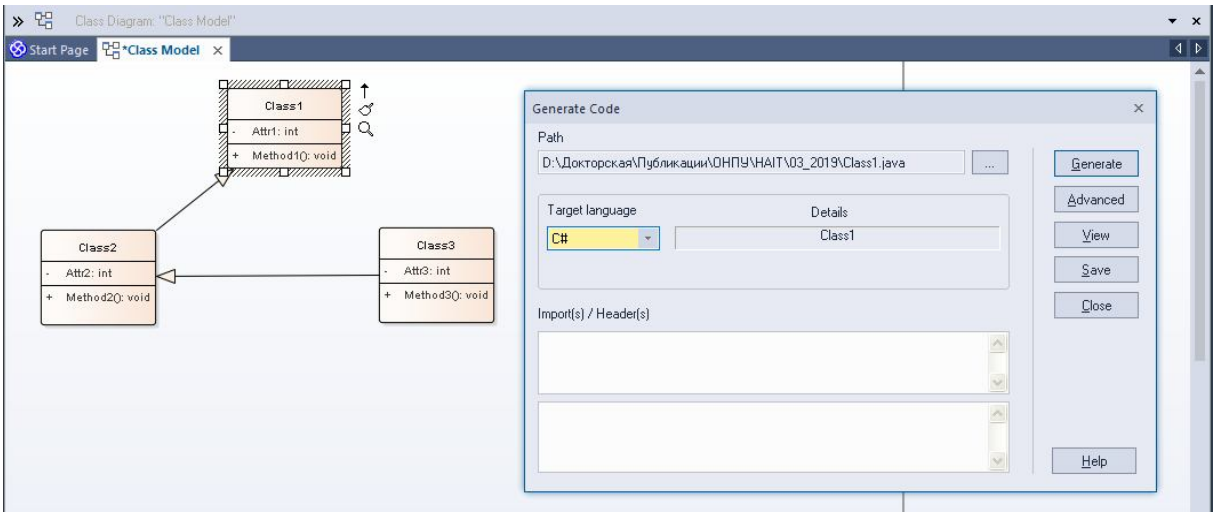


Рисунок 5.14 – Процес генерації коду

По суті на цьому процес генерації коду завершений. Проте, як пов'язана скелетна структура нашої програми, буде залежати саме від того, які типи зв'язку та які змінні, операції та атрибути вказати у МОПК. Не слід забувати, що це всього лише загальна основа (шаблон) під код. Увесь основний код, процеси та інші речі все одно повинні прописуватися програмістами (рис. 5.15). Зручністю такої генерації є саме структурування загального вигляду, допомога в розподілі завдань між програмістами і практично повне виключення проблематики несумісності модулів тому, що вся структура вже пов'язана спочатку.

Для того, щоб створити новий ПП на основі існуючого, необхідно розібратися у структурі первинного ПП. Структура BRL-CAD представлена у вигляді програмного коду мовою С і розбита на велику кількість модулів, в кожному з яких знаходиться один або декілька класів, взаємопов'язаних між собою або пов'язаних з іншими модулями. Крім розуміння кожного модуля, окремим завданням стало розібратися у зв'язках між класами і скласти цілісне уявлення. Оскільки це пряма робота

із кодом, то представляти структуру слід за допомогою моделей, що спеціально створені для цього – МОПК та МКР.

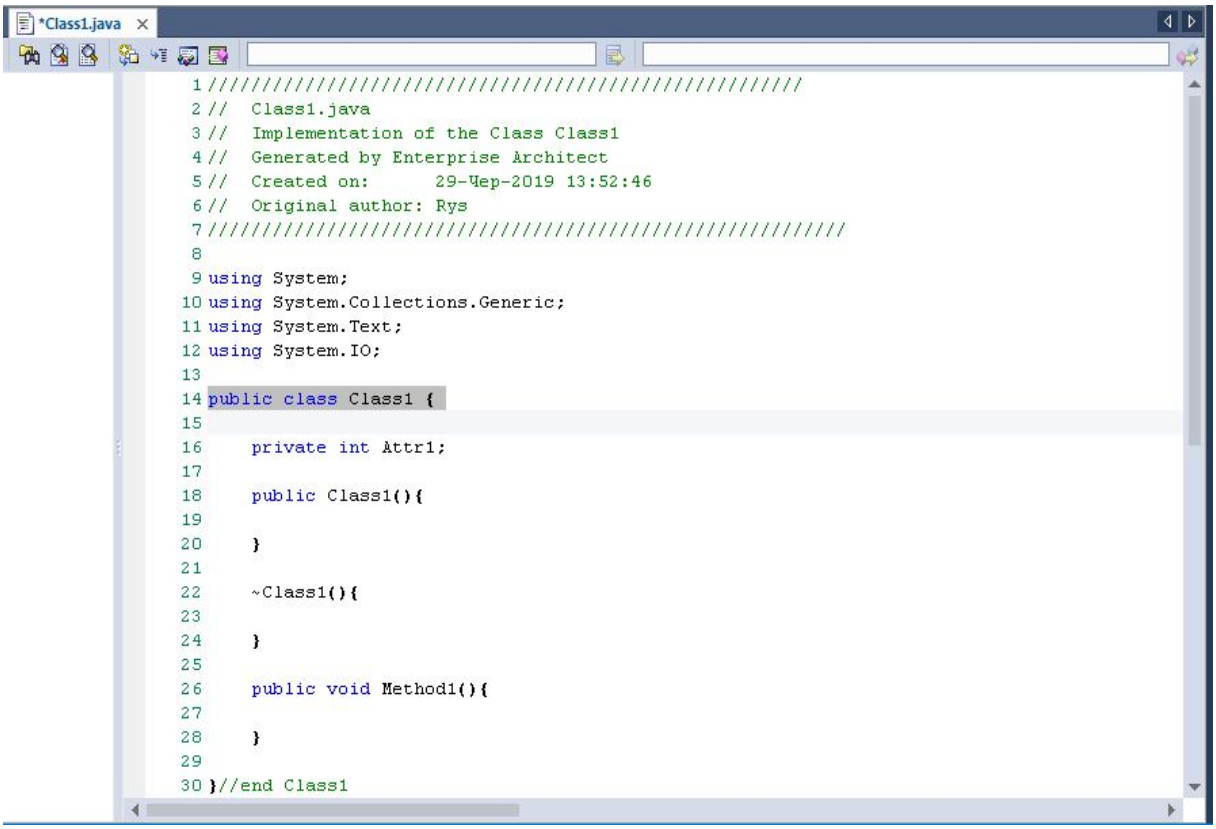


Рисунок 5.15 – Результат генерації

Першою належить скласти саме МОПК щодо первинного ПП тому, що складати відразу об’єднані МОПК та МКР недоцільно у плані трудомісткості роботи, при цьому ефективність такої дії практично не перевищує окремих моделей. Тобто при розборі будь-якого ПП кращим рішенням буде декомпозиція на окремі дрібні складові, у той час як при генерації нового ПП краще створювати точний й складний взаємозв'язок.

Для початку потрібно знайти «узагальнюючий» модуль (якщо він присутній). Узагальнюючим називається модуль, в якому представлені усі основні робочі підмодулі. Такий тип представлення здебільшого використовується в роботі ПП з відкритим кодом для більш зрозумілого й швидкого розбору та подальшої модифікації коду.

Було прийнято проєктне рішення: не відступати від цього правила та, оскільки точно так само будується САП з відкритим кодом, то необхідно код структурувати як можна краще, тобто намагатися зробити його «чистим».

У ПП BRL-CAD вся програмна структура мовою С знаходиться у папці «incude». Після відкриття файлу «brlcad.h», який є «узагальнюючим» модулем, отримаємо його набір управління підключенням:

```
#ifndef __BRLCAD_H__
#define __BRLCAD_H__

#include "common.h"

/* system headers presumed to be available */
#include <stdio.h>
#include <math.h>

/* basic utilities */
#include "bu.h"

/* vector mathematics */
#include "vmath.h"

/* non-manifold geometry */
#include "nmg.h"

/* basic numerics */
#include "bn.h"
/* database format storage types */

#include "db.h"
/* raytrace interface constructs */
#include "raytrace.h"
/* trimmed nurb routines */
#include "nurb.h"

/* the write-only database library interface */
#include "wdb.h"
/* in-memory representations of the database geometry
objects.  these
 * are subject to change and should not be relied upon.
 */
#include "rtgeom.h"
/* database object functions
 */
#include "rtfunc.h"

#endif  /* __BRLCAD_H__ */
```

Тут відбувається управління підключеннями підмодулів, що відповідальні за певні операції. Причому, передбачені навіть коментарі із поясненнями для підвищення зручності роботи з кодом. Отже, саме звідси

почнеться управління генерацією нової структури. Кожен підмодуль являє собою цілий набір файлів із включеними до них класами, тому всі ці файли найлегше буде представити у вигляді «Пакетів» взаємодіючих між собою у моделі пакетів.

При створенні першого ж пакета, який називається «include» (на ім’я папки, у якій знаходиться весь виконуваний код), пропонується обрати тип шаблону проєктування, який буде використано у подальшій внутрішній, моделі управління пакетами (МУП) (рис. 5.16).

Оскільки всі наступні модулі будуть являти собою окремі зв'язки у вигляді класів, що взаємодіють тільки на рівні управління передачі пакетних даних, то необхідно створити їх у вигляді таких же пакетів. При створенні цих пакетів знову пропонується обрати тип моделі, який буде використовуватися всередині. Вже далі слід додати МОПК, тому всередині цих модулів взаємодія відбувається у вигляді класів з їх операціями та атрибутами. Після додавання всіх основних пакетів, оглядач проєкту набуває такого вигляду, як зображено на рис. 5.17.

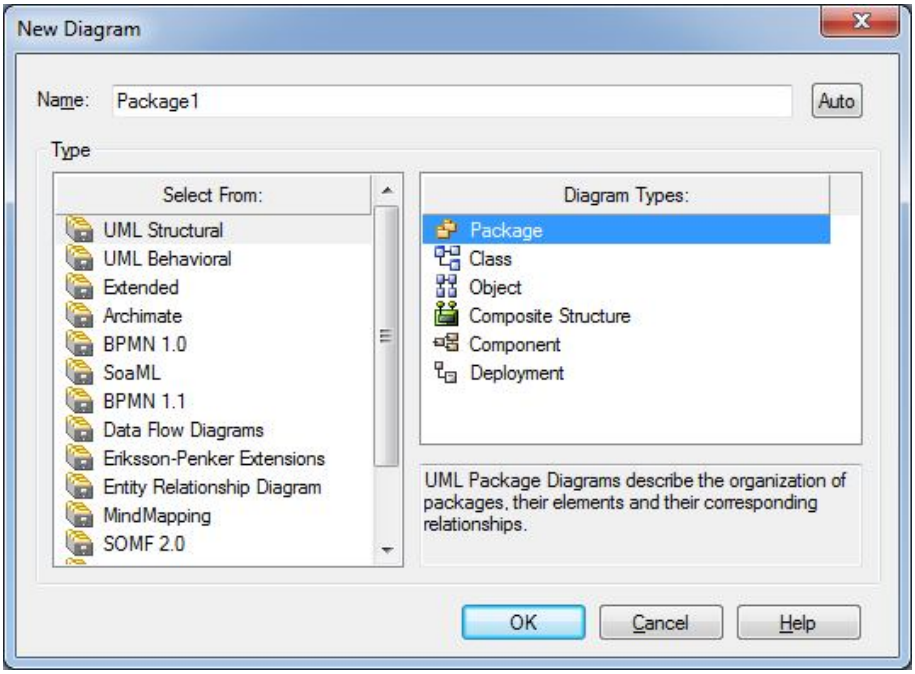


Рисунок 5.16 – Вибір типу діаграми всередині МУП

Коли основна структура завершена, приступимо до кожного пакета із розбивкою його за класами та взаємозв'язками між пакетами, що будуть розглянуті пізніше, після визначення їх внутрішньої структури.

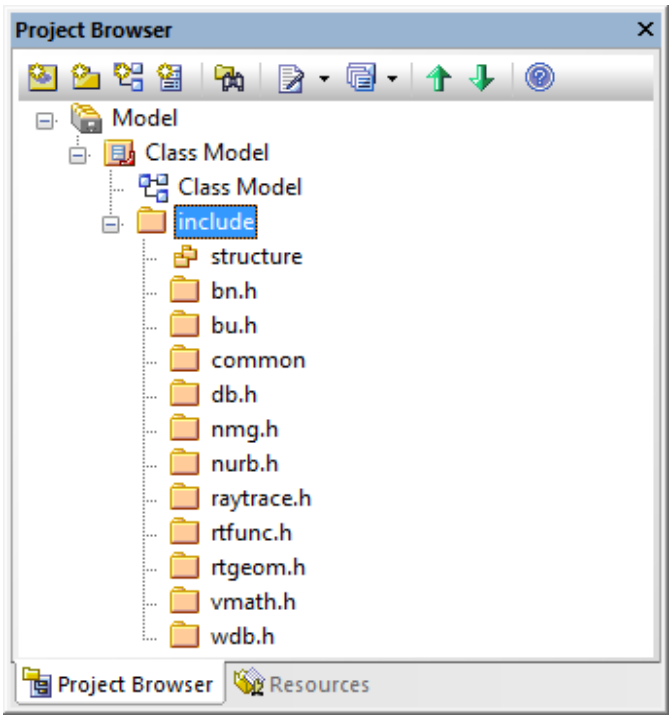


Рисунок 5.17 – Оглядач проєкту

Приступимо до роботи з моделлю пакетів. Перший пакет називається «bn.h» і на жорсткому диску він представлений єдиним файлом з аналогічною назвою. Перше, що необхідно зробити – це відновити його структуру за допомогою команди «Import C File» з підменю «Source Code Engineering». У підсумку отримаємо результат, з якого видно, що відновився всього лише один клас, що є дивним. Файл, який ми спробували відновити не був повноцінно прочитаний, про що свідчить повідомлення про помилки. Далі було відкрите вікно звіту про зворотній інжиніринг (рис. 5.18).

У ньому ми бачимо, що крім відтворення даного класу, так само відтворюються всі можливі види зв'язків та відновлення всіх типів відношень. Але для нас цього не достатньо, тому коли ми відкриваємо вікно самого фізичного файла розташування коду, то бачимо, що класів там набагато більше. Найперша відмінність, яка впадає в очі, говорить про те, що відтворений на екрані клас єдиний задає якісь змінній, в той час, як всі наступні користуються командою #define, яка служить для оголошення будь-якої константи. Ця константа може братися з інших модулів.

Файл, що нас цікавить, складається в основному з конструкцій типу:

а) Struct bn_tol – оголошення класу;

б) BN_EXPORT BU_EXTERN (void anim_tran, (mat_t m)) – оголошення процесу;

в) #Define bn_cx_add (ap, bp) {(ap) -> re + = (bp) -> re; (ap) -> im + = (bp) -> im;} – оголошені константи.

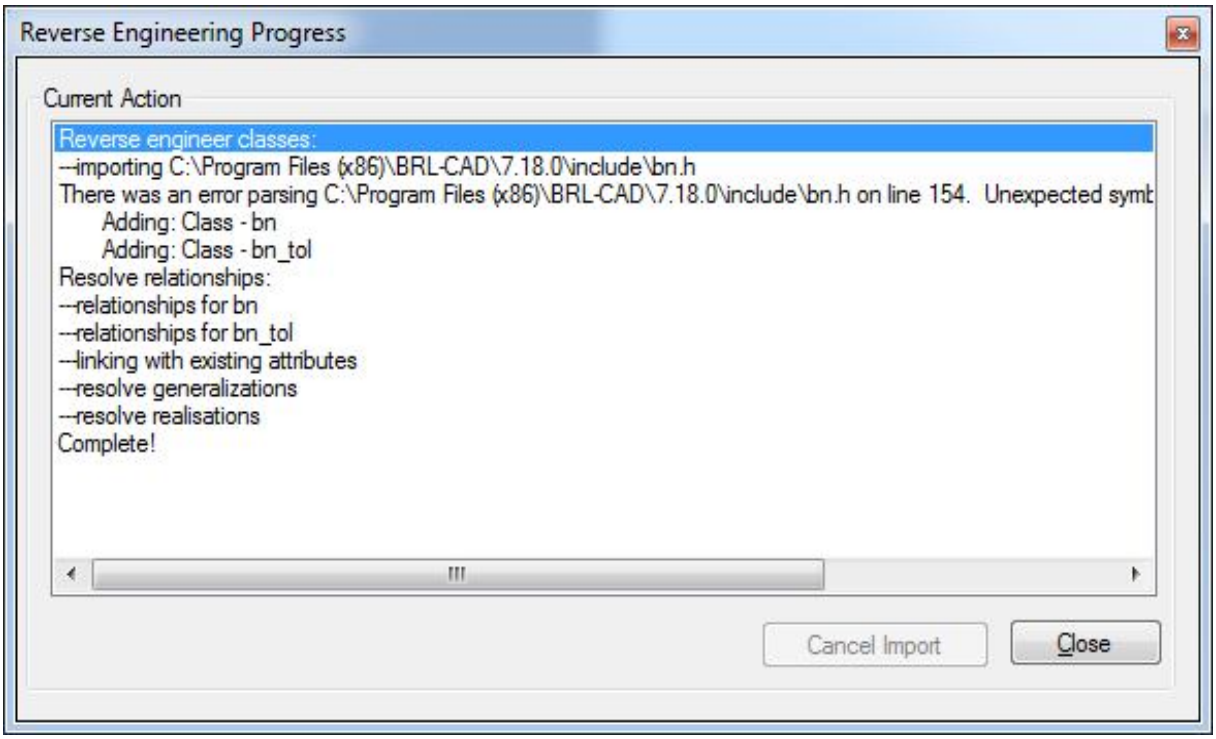


Рисунок 5.18 – Звіт про зворотній інжиніринг

Оскільки зворотний інжиніринг файлу не приніс якісного результату, то дороблювати розбір на діаграми доведеться вручну. Для цього необхідно розуміти, що ми будемо шукати в файлах. В першу чергу, нас звичайно ж цікавлять оголошені класи та їх змінні, від цього залежить загальна структура і вид діаграми. Після того як відтворено класи і змінні, то необхідно братися за процеси, що задіють дані класи.

Процес відтворення повноцінної структури аж до кожної змінної – не потрібно наводити, у реінжинірингу важливо зрозуміти процес роботи програмних модулів та скласти якомога більш зрозумілу схему, за якою буде зручно працювати не тільки самостійно, але і пояснювати принципи програмістам, які будуть відтворювати нові модулі. Тому багато дрібних класів та процесів не будемо вказувати або ж будемо об'єднувати їх у більш великі діаграмні процеси.

Приклад схематичного об'єднання процесів. У коді знаходиться багато дрібних процесів з поясненнями типу:

```
BN_EXPORT BU_EXTERN(void anim_dy_p_r2mat,
                (mat_t m,
                 double y,
                 double p,
                 double r));
/**
 * @brief Make a view rotation matrix, given desired yaw,
pitch and
 * roll. (Note that the matrix is a permutation of the
object rotation
 * matrix).
 */
BN_EXPORT BU_EXTERN(void anim_dy_p_r2vmat,
                (mat_t m,
                 double yaw,
                 double pch,
                 double rll));
/**
 * @brief Make a rotation matrix corresponding to a
rotation of "x"
 * radians about the x-axis, "y" radians about the y-
axis, and then
 * "z" radians about the z-axis.
 */
```

У коментарях чітко сказано: за що відповідає кожен процес (ще раз згадуємо, що це одна із зручностей роботи з ПС із відкритим кодом, хоча й ця зручність має низку недоліків). У нашому прикладі ці процеси відповідають за обертання обраного об'єкта у поданій системі координат: *XYZ*, за координатами якої відстежується, будь-яке відхилення. У моделі загалом назвемо цей процес Rotation (обертання) та не будемо вдаватися до деталей, оскільки все одно пізніше, при роботі з програмістами доведеться розробляти нову модель з урахуванням специфіки мови програмування. Цим ми полегшимо і скоротимо процес розробки моделей тому, що в одному тільки цьому файлі «bn.h» таких процесів більше тридцяти.

Починаємо побудову МОПК, враховуючи об'єднання несуттєвих класів. Покажемо докладно етапи по одному з класів (рис. 5.19, 5.20):

```
struct bn_unif {
    unsigned long magic;
    long msr_seed;
    int msr_double_ptr;
    double *msr_doubles;
    int msr_long_ptr;
    long *msr_longs;};
```

```
    #define BN_CK_UNIF(_p)  BU_CKMAG(_p, BN_UNIF_MAGIC,
"bn_unif")
    #define BN_CK_GAUSS(_p) BU_CKMAG(_p, BN_GAUSS_MAGIC,
"bn_gauss")
```

Остаточне подання МОПК для файла «bn.h» представлено на рис. 5.21.

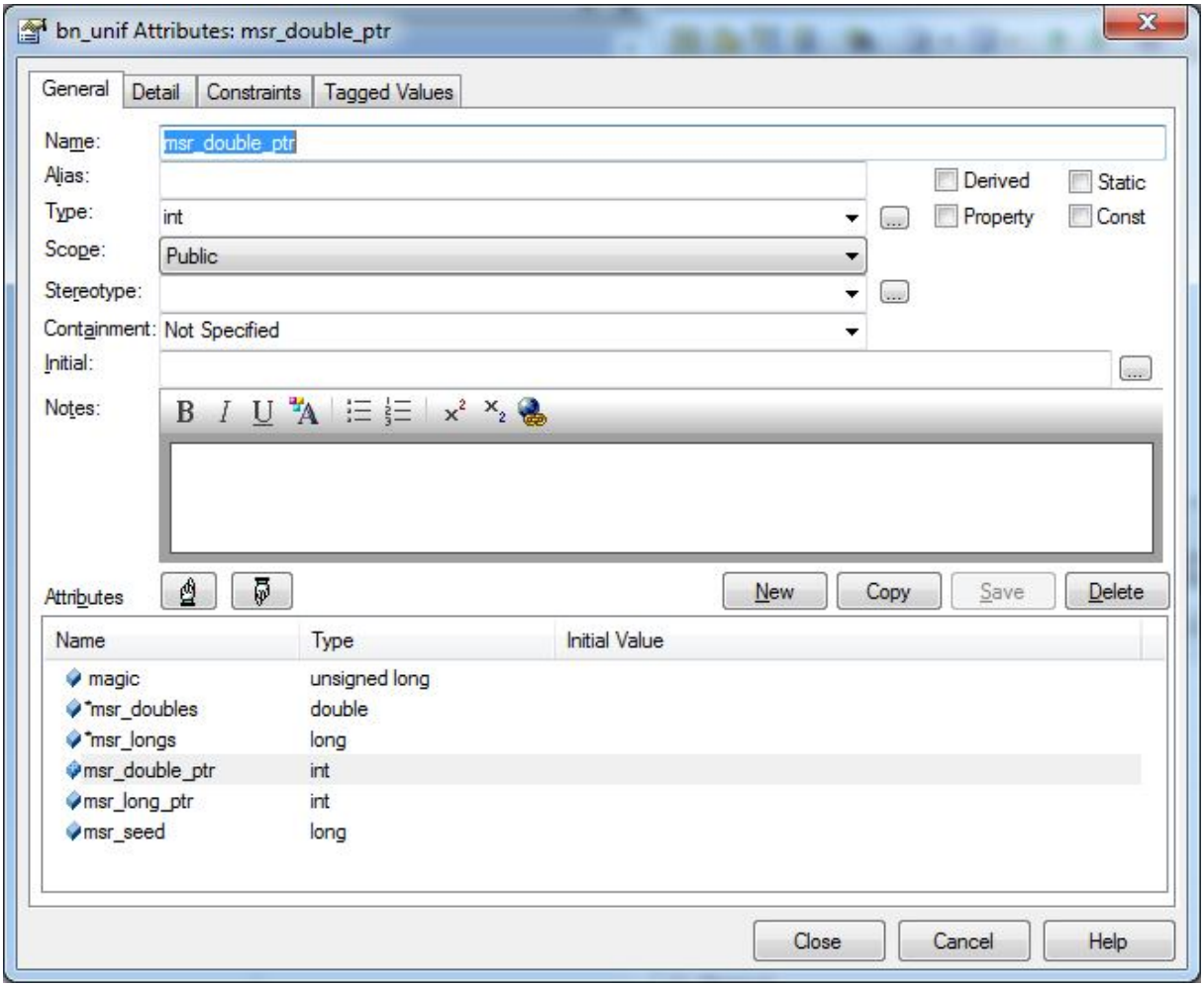


Рисунок 5.19 – Заповнення атрибутів класу bn_unif

Слід врахувати, що наведена МОПК є неповним відображенням всього файла, тому менш значні класи займають багато місця, але не є принциповими у відображенні. Те ж саме стосується і процесів – їх більше двох сотень, при цьому вони всього лише оголошують або структурують дані, тому їх відображаємо схематично, але при цьому через те, що кожен клас схильний до впливу на будь-які процеси, необхідно показати це на МОПК у вигляді оглядача (рис. 5.22).

У якості зв’язків обрано «асоціацію» тому, що цей зв'язок несе інформацію про відношення між об'єктами всередині програмного засобу. Асоціації можуть бути уточнені та показувати: який клас та як пов’язаний із іншими.

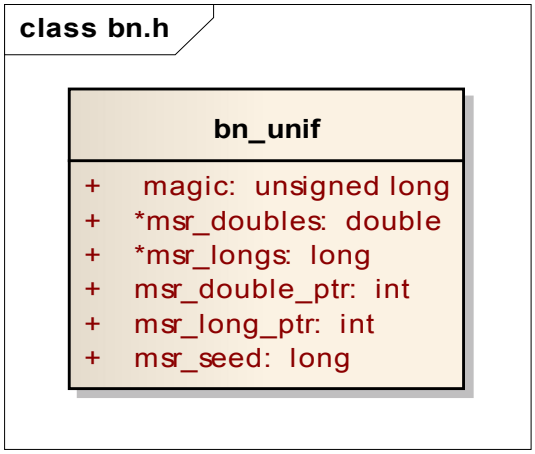


Рисунок 5.20 – Остаточне подання класу bn_unif

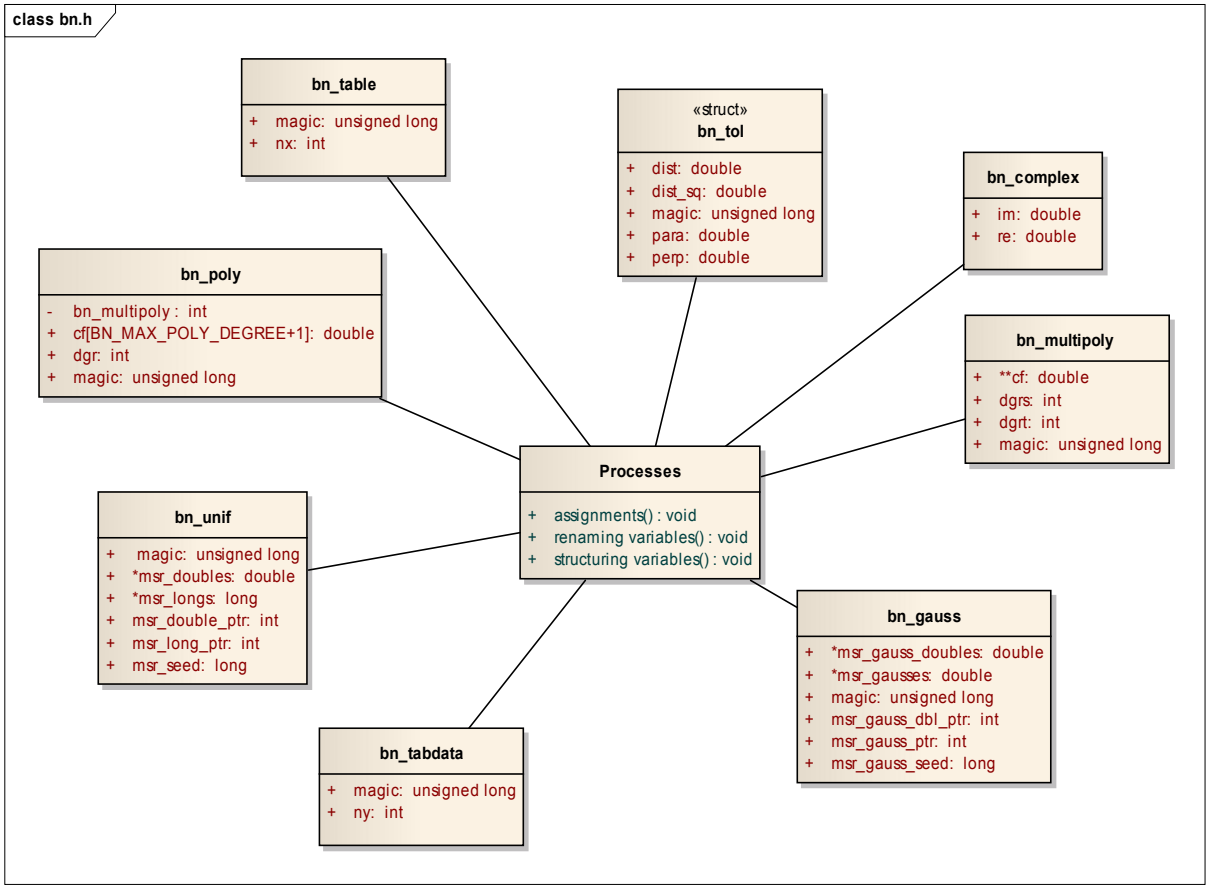


Рисунок 5.21 – МОПК для «bn.h»

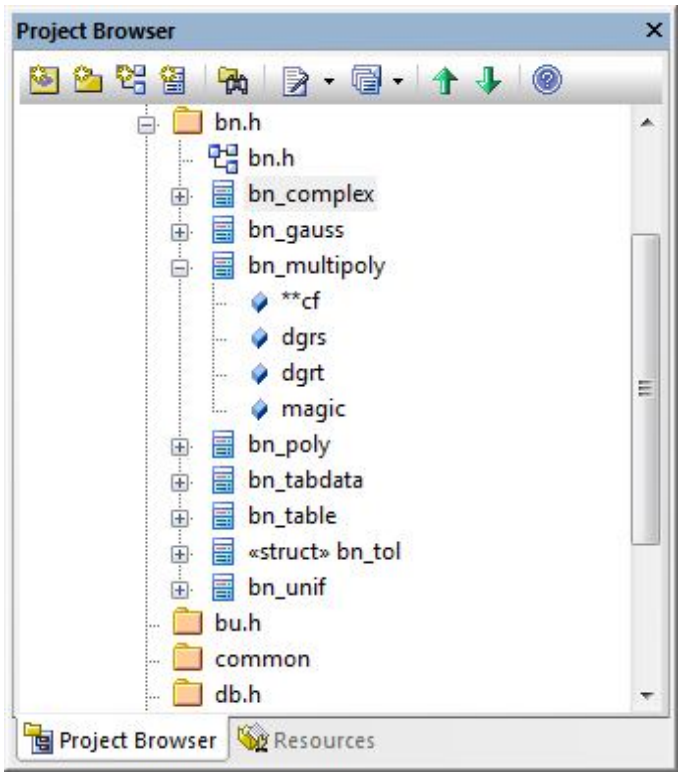


Рисунок 5.22 – МОПК у вигляді оглядача

Після завершення аналізу першого файлу, під назвою «bn.h», останнім кроком у роботі із ним необхідно показати зв’язок цього «пакета» з іншими. На початку файлу є такі рядки:

```
/* interface headers */
#include "bu.h"
/* required for BU_EXTERN, BU_CKMAG */
#include "vmath.h"
/* required for mat_t, vect_t */
```

Цей запис свідчить про те, що є взаємозв'язок з іншими пакетами, включеними в цю модель, тому цей зв'язок треба теж відобразити. Необхідно використовувати тип зв'язку «залежність» (dependency), тому виконання математичних та інших функцій в файлі залежить саме від цих двох підключених складових. У підсумку МУП буде виглядати так, як на рис. 5.23.

У деяких випадках зв'язками між класами можна знехтувати, оскільки робота з ними йде на рівні наших файлів (пакетів), а це означає що звернення до класу з файлу «bmath.h» відбуватиметься не всередині файлу, а ззовні, наприклад від файлу «bn.h». Так само в деякому випадку

зв'язки будуть відображатися в назві, наприклад відношення між батьківським і дочірніми класами (рис. 5.24). Тобто клас «rt_revolve_internal» є спадкоємцем класу «rtgeom».

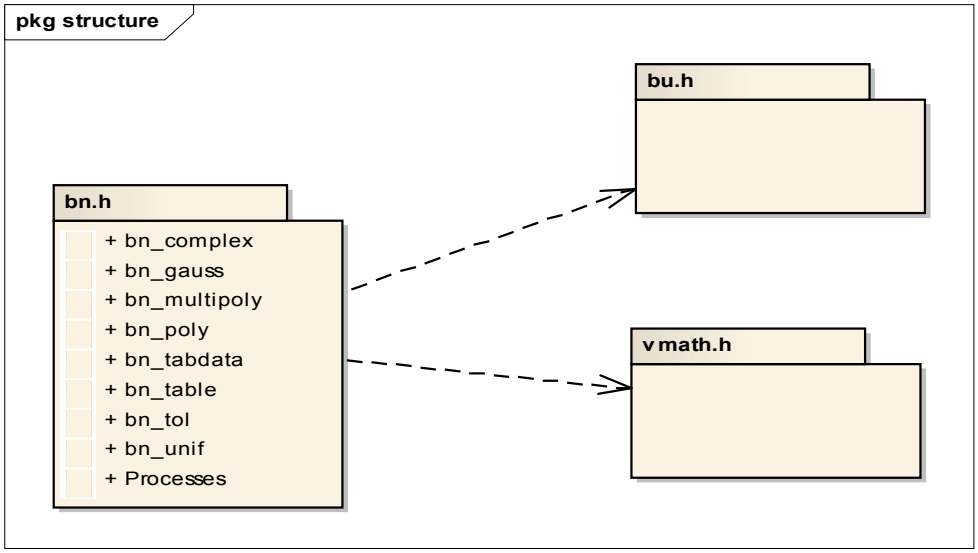


Рисунок 5.23 – Відносини «залежності» між пакетами

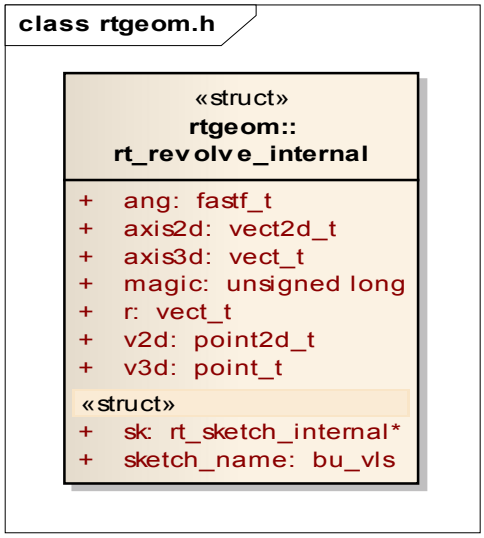


Рисунок 5.24 – Клас-нащадок

На даному етапі виконання роботи пакети «bu.h» і «vmath.h» ще не заповнені класами і функціями, тому їх відображення поки що є суто схематичним.

Таким чином, було розібрано, як відновлювати вручну файли з мови С. Для відображення повної МУП (рис. 5.25) аналогічним чином було розглянуто усі файли, що знаходяться в даному ПП.

На жаль, кількість пакетів та, відповідно, обсяг моделі настільки великий, що автор не в змозі навести її у повному представленні у межах аркушу А4. Якщо виконати декомпозицію МУП, то втрачається цілісне подання управління, тому наводимо тільки її ескізне представлення (рис. 5.25). Основне призначення ескізу моделі: не показати наповнення пакетів, а проілюструвати модель відношень об’єднань пакетів нового ПП.

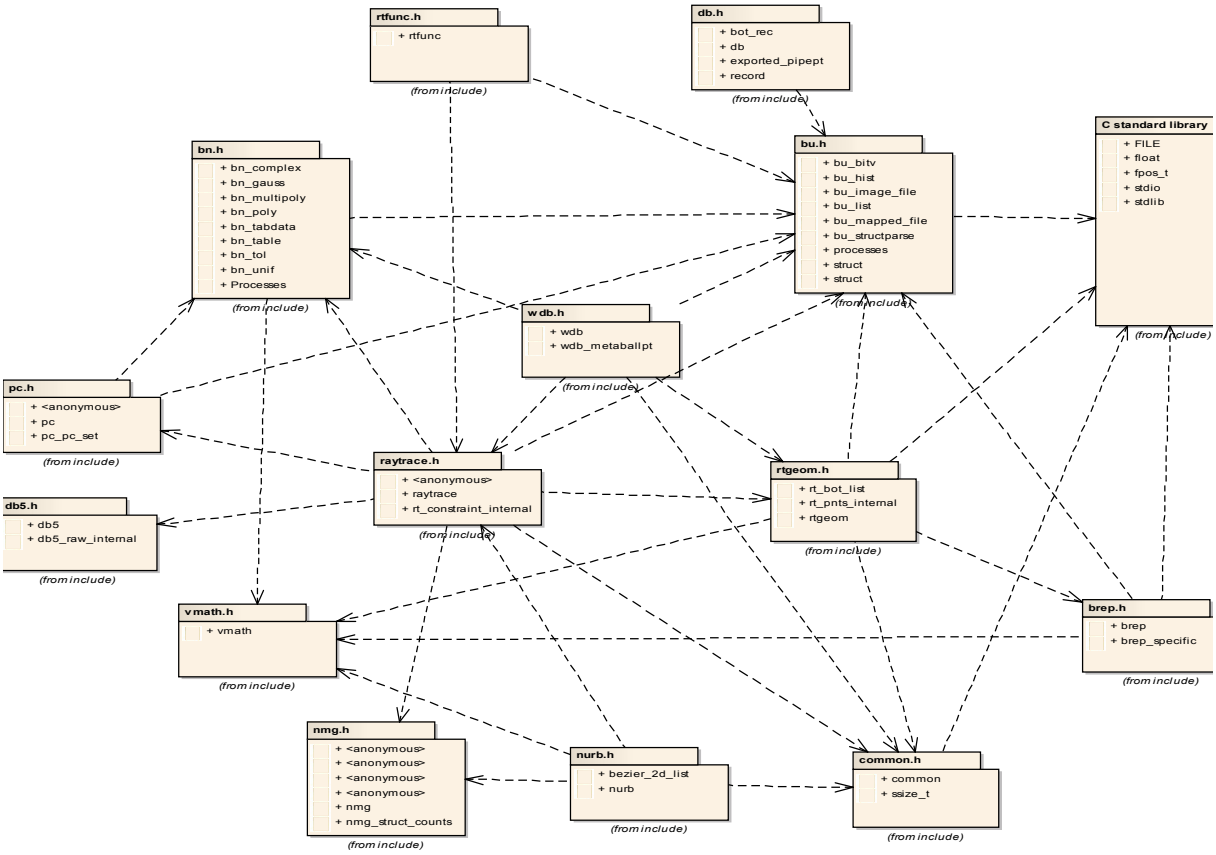


Рисунок 5.25 – Повна модель пакетів проєкту

5.5 Узагальнюючі кроки проактивного управління проєктами з розвитку програмних систем та продуктів

Для розробки архітектури оновленого ПП використаємо три основні моделі: МВВ, МОПК та МКР.

Крок 1. У першу чергу необхідно визначити функціонал майбутньої ПС, наскільки сильно він відрізнятиметься від оригінального продукту, які

зміни будуть внесені. Тому за вже використаним принципом побудови моделей [221] складаємо МВВ.

Для того, щоб розібрати і створити МВВ, необхідно визначитися з акторами та прецедентами моделі, тому почнемо з перегляду основних «exe»-файлів ПП. Їх усього два: «Archer» та «MGED». Це й будуть актори. Настроюванням їх специфікацій займемося пізніше, а зараз важливо визначити прецеденти, які відповідають кожному з них. Для цього запускаємо кожен виконавчий файл та дивимося на їх можливості. «MGED» – це основний програмний модуль, який відповідає за проєктування, зміну й трасування променів. Виглядає він так, як на рис. 5.26.

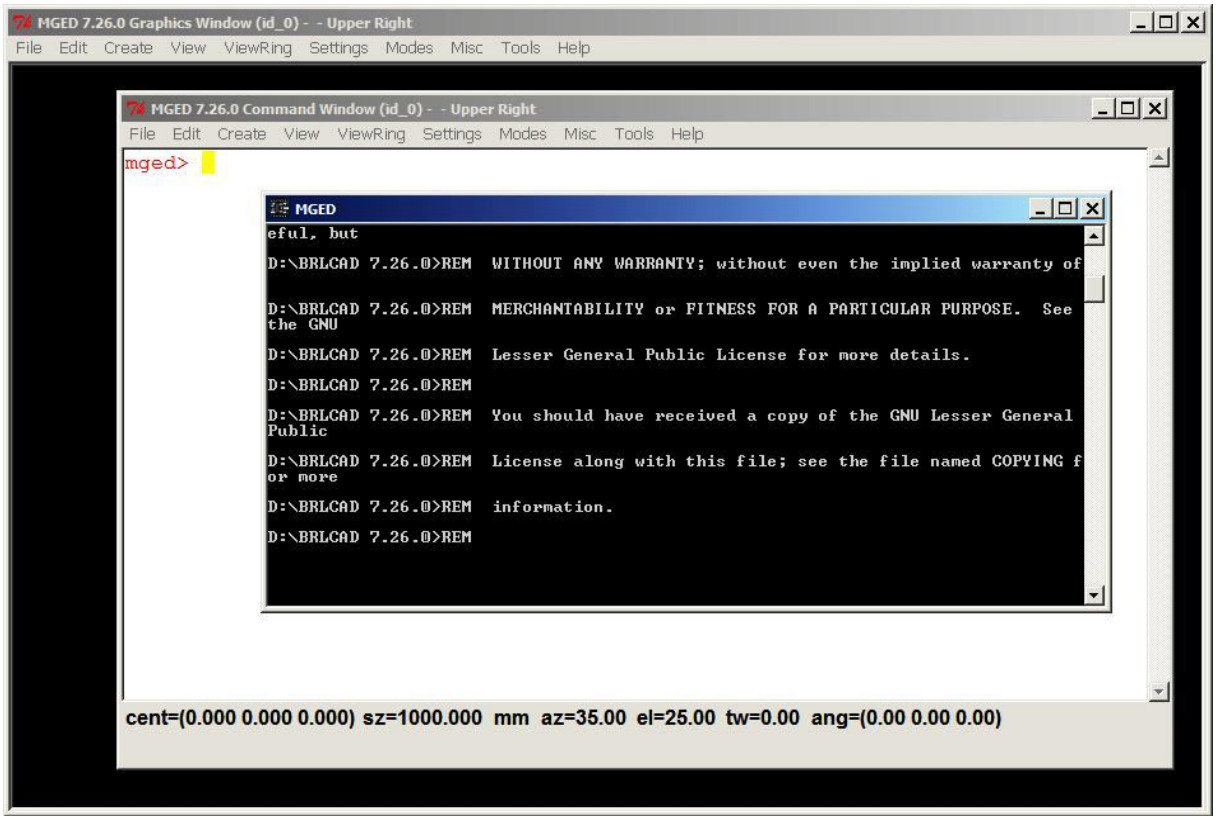


Рисунок 5.26 – Інтерфейс «MGED»

Крім консолі та GUI (графічного інтерфейсу користувача) є додаткові функції у верхньому меню, перерахуємо їх:

а) File – основні команди, включаючи трасування променів;

---

221. Yang H. Advances In UML And XML-based Software Evolution. Idea Group Publishing, 2005. 362 p.

б) Edit – відповідає за зміну простих та складних фігур за допомогою різних методів;

в) Create – створення простих фігур та складних ієрархій з простих фігур;

д) View – зміна кута й точки огляду;

е) Settings – загальні налаштування для роботи;

ж) Modes – модулі (одна з неофіційних обов'язкових властивостей вільно поширюваного ПЗ), що бувають платними та безкоштовними;

и) Tools – інструменти для роботи з фігурами та графікою;

к) Help – допоміжні файли та інструкції.

Це всі функції, які будуть першим рівнем прецедентів МВВ, тому відразу створимо їх (рис. 5.27) (МВВ, що виконана англ. мовою, із якою безпосередньо працювали програмісти наведена у дод. Б).

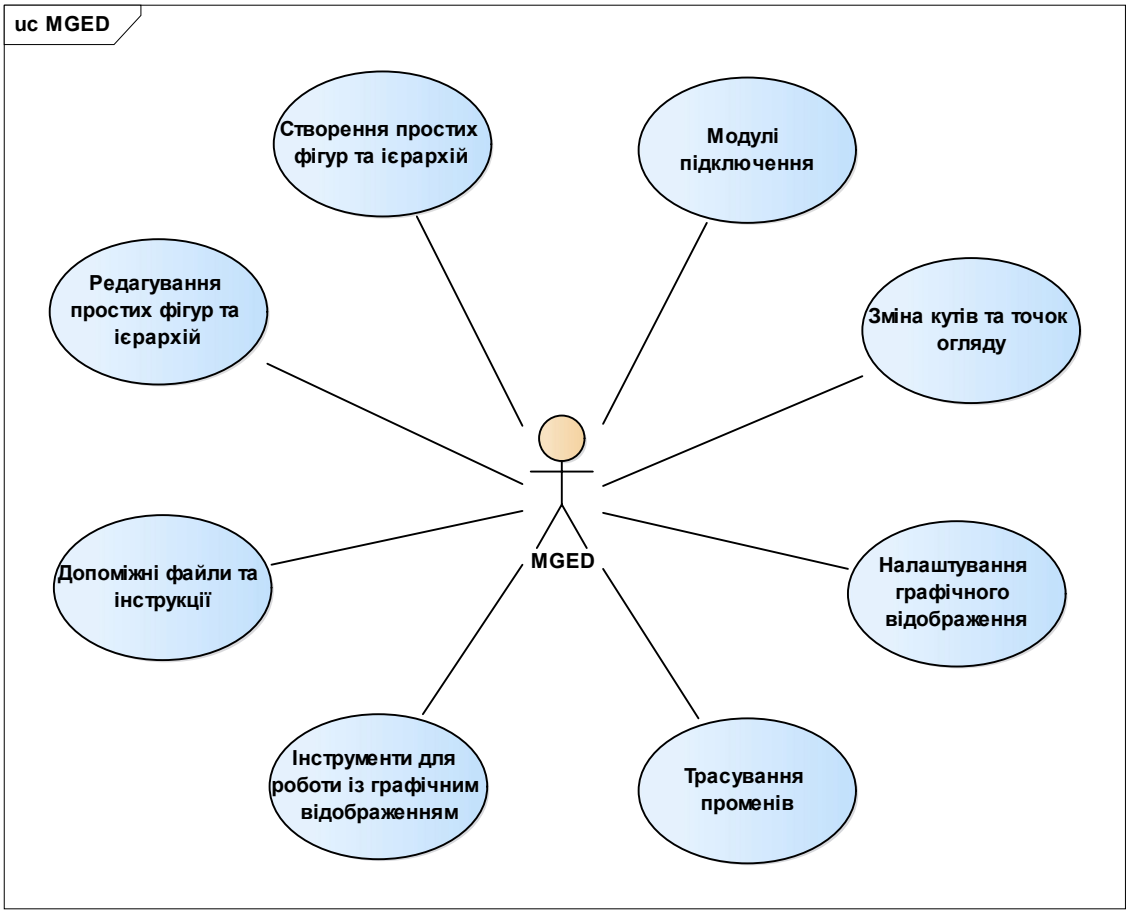


Рисунок 5.27 – Перший рівень прецедентів для «MGED»

На цьому етапі роботи важливо визначити тип актора й прецедентів та їх зв'язки між собою [222]. Всі об’єкти є модулями, що взаємодіють із користувачем, та для них є спеціальні стереотипи, які називаються «business use case» для прецеденту й «business actor» для моделі. У цьому конкретному випадку покажемо бізнес стереотип тільки для актора, тому буде логічно і зрозуміло із зв'язків, що всі наступні прецеденти пов'язані із ним. Далі, візьмемося за зв'язки.

Прецеденти першого рівня – це оновлені можливості користувача оновленого ПП. У зв'язках МВВ є спеціальний тип «use», сутність якого полягає в тому, що він показує використання можливостей будь-якого актора чи прецеденту [223]. Він найбільш детально описує цей рівень, тому показує, що прецеденти типу «створення простих фігур» є однією з розширених можливостей актора «MGED».

Єдиною відмінністю прецедентів будуть «Зовнішні модулі» – вони не обов'язкові для роботи, тому для них будемо використовувати інший тип зв'язку, який називається «subscribe», що означає «опис». По суті він й допомагає описати будь-який модуль та у в нашому випадку цим модулем є актор «MGED». Після розстановки зв'язків й стереотипів отримаємо такий результат як на рис. 5.27.

Крок 2. Другим рівнем МВВ буде створення МУП (рис. 5.28) (модель, що виконана англ. мовою, із якою безпосередньо працювали програмісти подається у дод. Б), яка містить ці ВВ, що реалізують можливості першого рівня [224]. Для цього відкриваємо специфікацію кожної з функцій у графі команд та докладно розписуємо команди як прецеденти. Тип зв'язків, які використовуються у МУП:

– «nesting» – показує вміст та додаткове розширення можливостей акторів;

– «compose» – композиційне підключення;

– «realize» – реалізація функцій.

---

222. Weilkiens T., Oestereich B. UML 2 Certification Guide: Fundamental & Intermediate. Exams Morgan Kaufmann. The MK/OMG Press, 2006. 320 p.

223. Бабич А. В. Введение в UML. Москва: НОУ ИНТУИТ, 2016. 209 с.

224. Samek M. Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems Newnes: 2nd edition, 2008. 728 p.

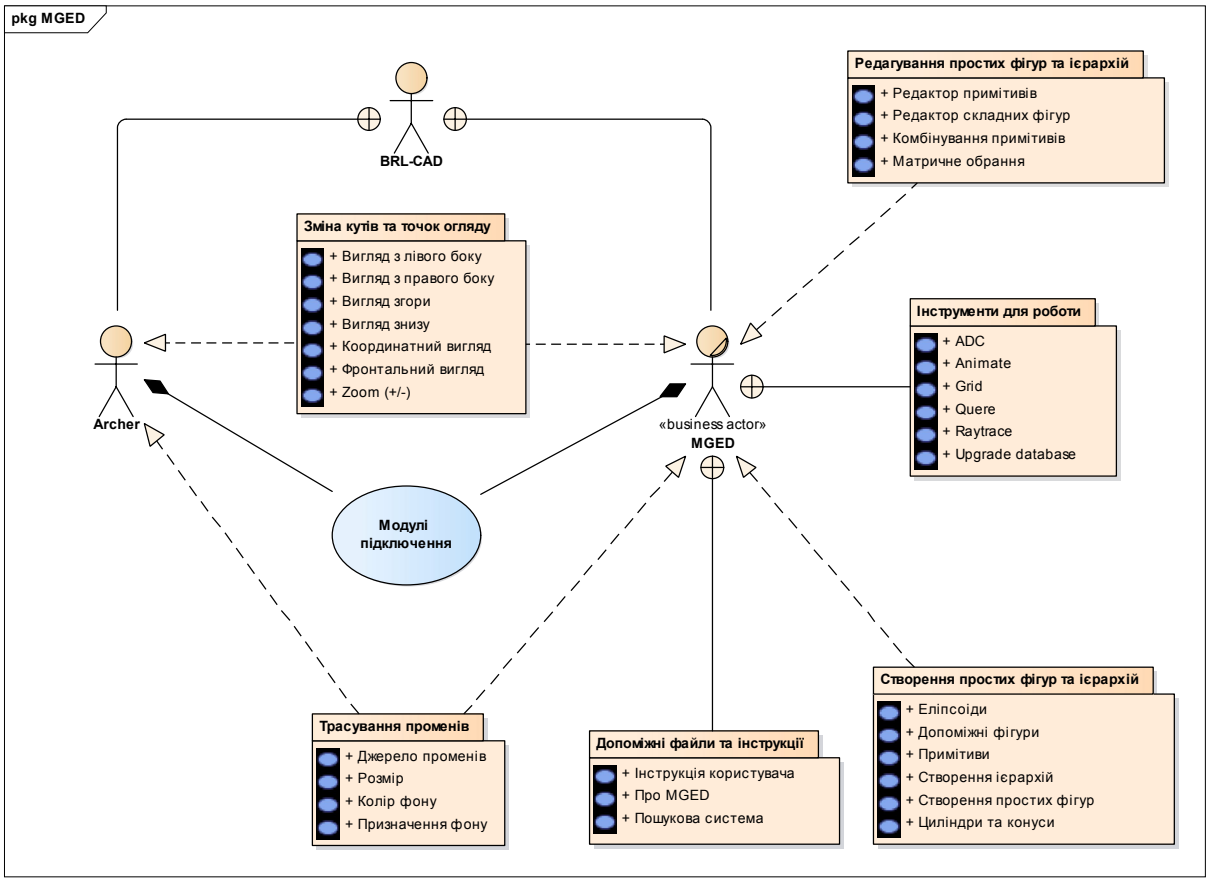


Рисунок 5.28 – Модель управління пакетами для MGED

Проте, якщо реалізовувати усі прецеденти у поданій моделі, то вийде перевантаження елементами, що є неправильним, тому кожен прецедент першого рівня розглянемо як об’єднані за функціоналом пакети (рис. 5.29) (МУП, що виконана англ. мовою, із якою безпосередньо працювали програмісти подається у дод. Б).

Процес розробки таких діаграм вже був описаний раніше. Було прийнято рішення, що функціонал центрального модуля («New_Product») слід залишити приблизно таким же, яким він і був, а всі додаткові складні функції включати за допомогою зовнішніх модулів. Це дозволяє не перенавантажувати ПС зайвими командами, що добре позначиться на рівні її продуктивності.

Пакети, що відповідають за виконання функцій, мають відповідні назви: «Зміна робочих кутів», «Редагування», «Створення фігур та ієрархій», «Трасування променів». Вони відносяться до центрального модуля «New_Product» композиційними залежностями. Кожен з цих пакетів має у своєму складі ВВ, що безпосередньо пов’язані із функцією

конкретного пакету (рис. 5.29). Пакети, що відповідають за підключення додаткового інструментарію також мають відповідні інструментарію назви: «Інструментарій», «Модулі підключення» та відношення до центрального модуля «New_Product» у вигляді агрегацій. Також пакет «Інструментарій» має у своєму складі ВВ, за допомогою яких виконуються графічні, логічні, інформаційні та інші перетворення й зміни.

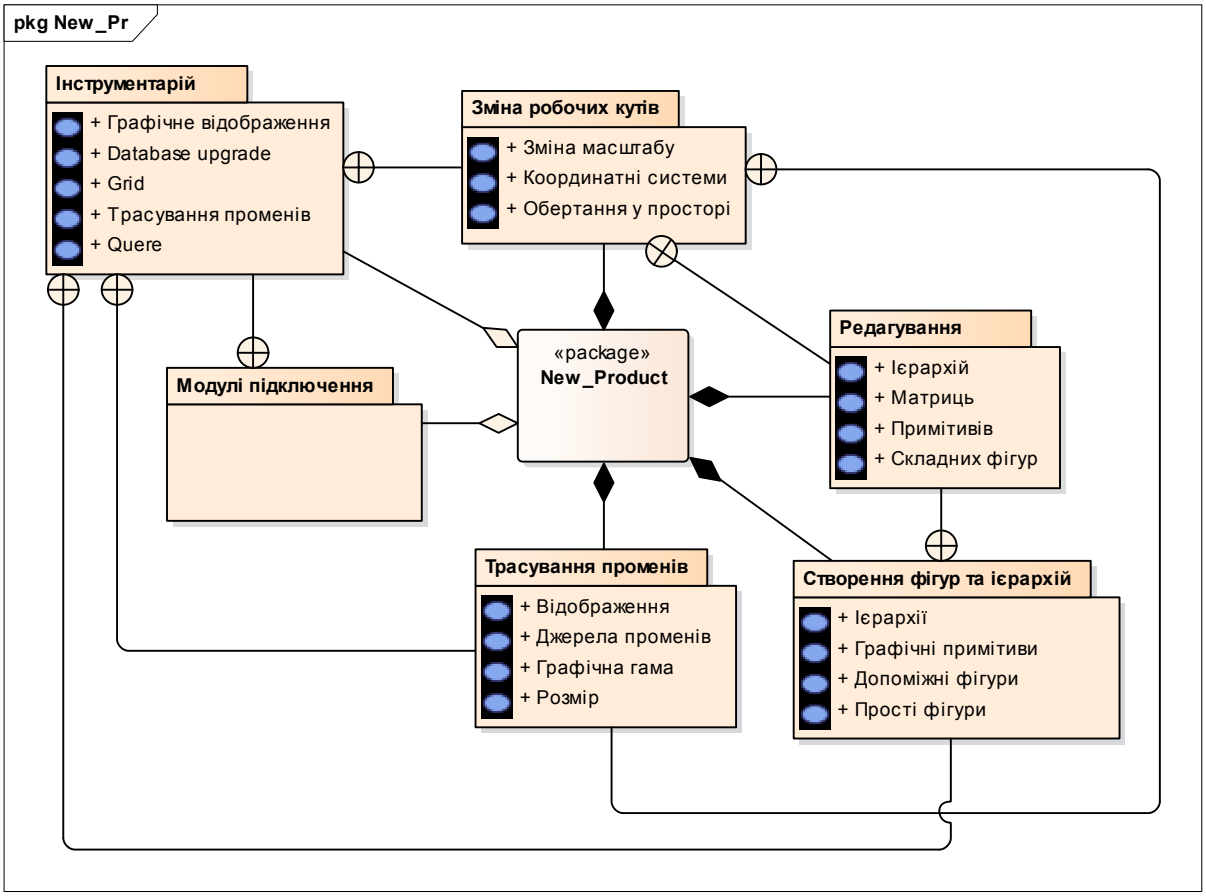


Рисунок 5.29 – МУП, що містить ВВ оновленого продукту

Розглянемо у МУП використання відношення типу «Nesting» (англ.: гніздування), воно означає ієрархічне розміщення у межах одного пакету виклику функцій іншого. Тобто, якщо брати за основу останню модель (рис. 5.29), то, наприклад, функція «Редагування» викликається тільки після створення відповідної фігури або ієрархії (пакет «Створення фігур та ієрархій») або теж «Редагування» доступне також як складова зміни після активізації пакету «Зміна робочих кутів». Також виклик функцій пакету «Трасування променів» доступний після активізації пакету «Зміна робочих кутів», який, в свою чергу, є візуалізованим результатом роботи

(наслідком) пакету «Інструментарій», що являє собою сукупність програмних модулів (пакет «Модулі підключення») до якого йде відповідне відношення «Nesting».

Якщо аналізувати далі МУП, то слід додати, що до пакету «Інструментарій» за допомогою відношень «Nesting» підключено пакети «Трасування променів» та «Створення фігур та ієрархій», що означає виконання функцій цих двох пакетів можливе лише за допомогою відповідного інструментарію.

Останній крок, невеликий, але не менш важливий – розставити специфікації для акторів – всі актори – це модулі ПП, які беруть участь у роботі з користувачем, тобто являють собою інтерфейс. Для такого типу модулів є спеціальна специфікація під назвою «business», її будемо використовувати. На цьому побудови МВВ та МУП – закінчено.

Крок 3. Після визначення функціоналу ПП наступним кроком буде складання робочої МОПК, яка складається відповідно до специфікації нової мови та вдалих рішень первинного ПП. В підсумку, ескізна МОПК оновленого ПП набуває такого вигляду, як на рис. 5.30. На жаль, кількість класів та, відповідно, обсяг МОПК настільки великий, що автор не в змозі навести її у повному представленні у межах аркушу А4. Якщо виконати декомпозицію моделі, то втрачається цілісне подання МОПК, тому наводимо тільки її ескізне представлення. Основне призначення ескізу МОПК: не показати наповнення класів, а проілюструвати модель ієрархічних відношень граничних об’єднань класів оновленого ПП (рис. 5.30).

Архітектурні зміни, які проведено у цьому ескізі, почалися з повноцінної візуальної розбивки модулів. Кожен унікальний модуль зібраний в свій окремий прямокутник – граничне об’єднання («boundary»), у якому й буде відбуватися увесь процес обчислення.

Наступна особливість: є центральний клас «Main», який відповідає за взаємодію між модулями. У кожного модуля є свій граничний клас, який взаємодіє з усіма іншими частинами ПС та закритими обчислювальними системами й БД. Такий спосіб було обрано, щоб мінімізувати кількість помилок у зв'язках та допомогти програмістам у поліпшенні ПП.

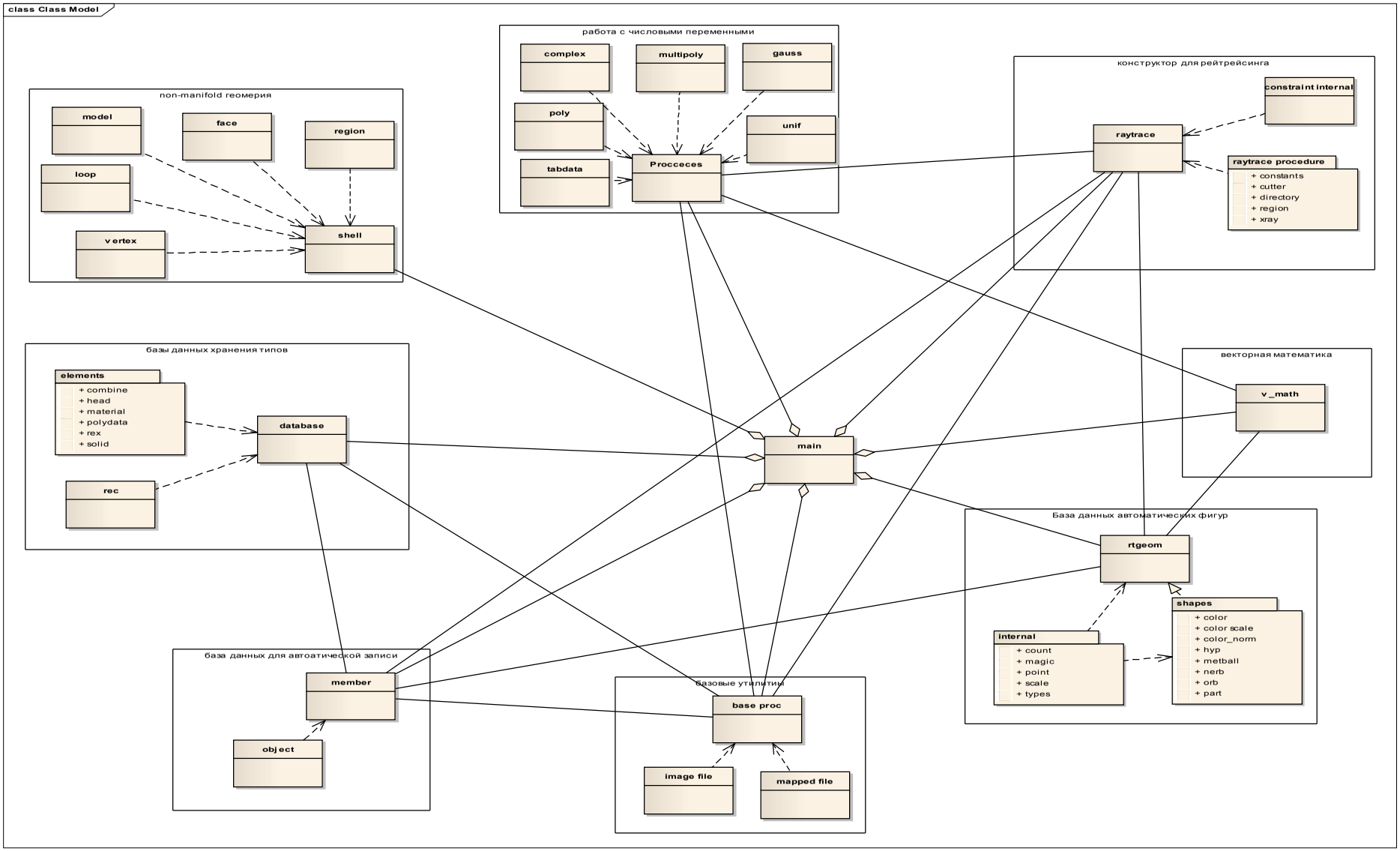


Рисунок 5.30 – Повна модель ієрархії класів (ескізне представлення) оновленого ПП

Крок 4. Як вже говорилося раніше, МКР – це друга модель, яка безпосередньо бере участь у створенні коду майбутнього ПП і для цього вона повинна бути пов'язана з першою – МОПК. Розберемо МКР для первинного ПП за класичними канонами [225] , з розбивкою на 3 складові частини:

– розгортання, які забезпечують безпосереднє виконання системою своїх функцій – такими компонентами можуть бути спільні бібліотеки з розширенням «* . dll»;

– робочі продукти, як правило, це файли з вихідними текстами програм, наприклад, з розширенням «* . h» з для мови С;

– виконавчі, що представляють собою модулі із розширенням «* . exe».

На першому етапі потрібен не стільки програмний код, скільки само фізичне розташування та загальна архітектура ПП, тому у загальній папці знаходиться встановлений додаток (рис. 5.31).

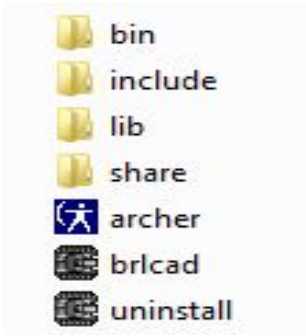


Рисунок 5.31 – Фізичне розташування додатка

Крім виконавчих файлів («archer» та «brlcad») в даній структурі знаходяться 4 папки. Папка «bin» містить основні та допоміжні «exe»-файли, які запускають окремі модулі (специфіка ПП: кожен модуль не інтегрується в інтерфейс, а підключається окремим вікном).

Папка «include» містить робочий продукт – файли з текстами ПС мовою С. Папка «lib» – бібліотечні файли, а папка «share» – текстові документи з описами, ліцензійні угоди, сторінки в Інтернет, загалом допоміжні файли.

---

225. Object Management Group. OMG Unified Modeling Language (OMG UML). Version 2.5 Object Management Group, 2013. 831 p.

Розглянувши цю структуру, вже можна побудувати «скелет» МКР (рис. 5.32).

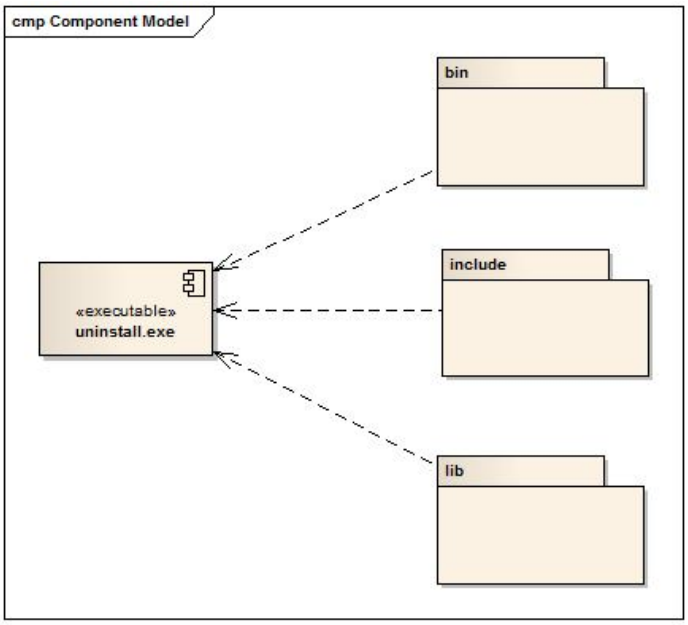


Рисунок 5.32 – Первинна структура

Наступним етапом роботи над МКР буде заповнення пакетів даними. Почнемо з самого важливого пакета – «include», у якому і містяться виконавчі файли.

Відтворимо структуру всередині папки і прив'язуємо до неї МОПК. Оскільки всі виконавчі файли написані мовою С, тобто вони одного формату, то створювати велику кількість компонентів не будемо, обмежимося одним. Задамо йому загальне ім'я «Source code», у специфікації вкажемо, що це мова С, й надалі прив'яжемо МОПК, яка містить весь код даного компонента. Загальна послідовність така:

– у менеджері проєкту обираємо модель, яка містить необхідні класи;

– знаходимо компонент, до якого буде приєднана МОПК;

– переносимо МКР до пакетного компонента.

У момент перенесення відбувається такий процес: сама МОПК переходить до ієрархії компонента, а усі класи залишаються на старому місці, у своєму підрозділі МОПК (рис. 5.33).

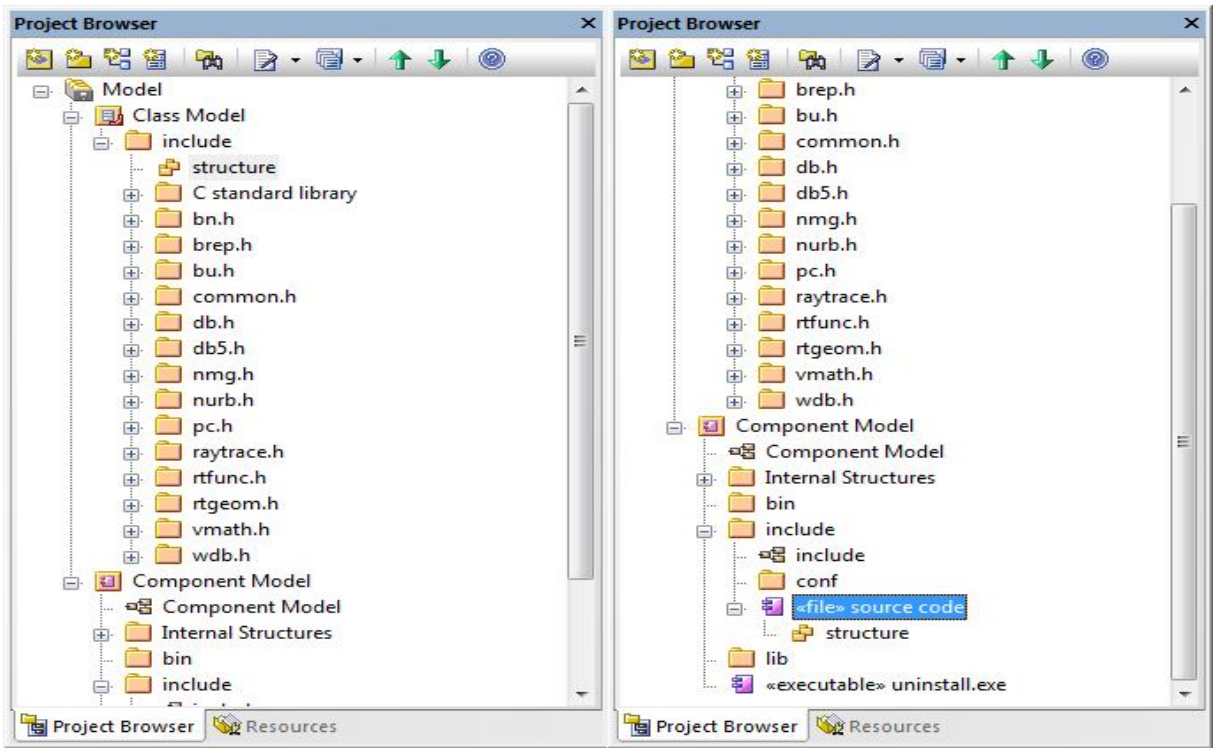


Рисунок 5.33 – Зміни у менеджерові проєкту

Зручність такої структури полягає в тому, що було пов'язано моделі, тобто перехід на МКР можна здійснити тільки через виділений компонент, проте самі класи не перемістилися, чим суттєво полегшили читаність, а у майбутньому й зміну структури.

Всі основні модулі перенесені й залишилося завершити лише деякі деталі, тому у папці, крім коду, міститься ще одна папка з конфігураційними файлами, які безпосередньо впливають на код. Створимо папку під назвою «config», заповнимо її компонентами відповідними файлу й надалі розставимо зв'язки у пакеті даних (рис. 5.34).

У папці «config» файли між собою не взаємодіють, тому внутрішні зв'язки не потрібні, але, в цілому, кожен з цих файлів виконує взаємодію з компонентом «Source code», що означає їх зв'язок. Залишається тільки визначити, який тип зв'язку.

В будь-якому з файлів папки «config» знаходиться набір чисел, до якого звертаються виконавчі файли, тобто при зміні числа змінюється і сам код. Для такого типу взаємодії існує спеціальний тип зв'язків, який називається «dependency» (залежність). Тому саме його помістимо до МКР, вказавши, що пакетний компонент «Source code» залежить від папки «config» (рис. 5.35).

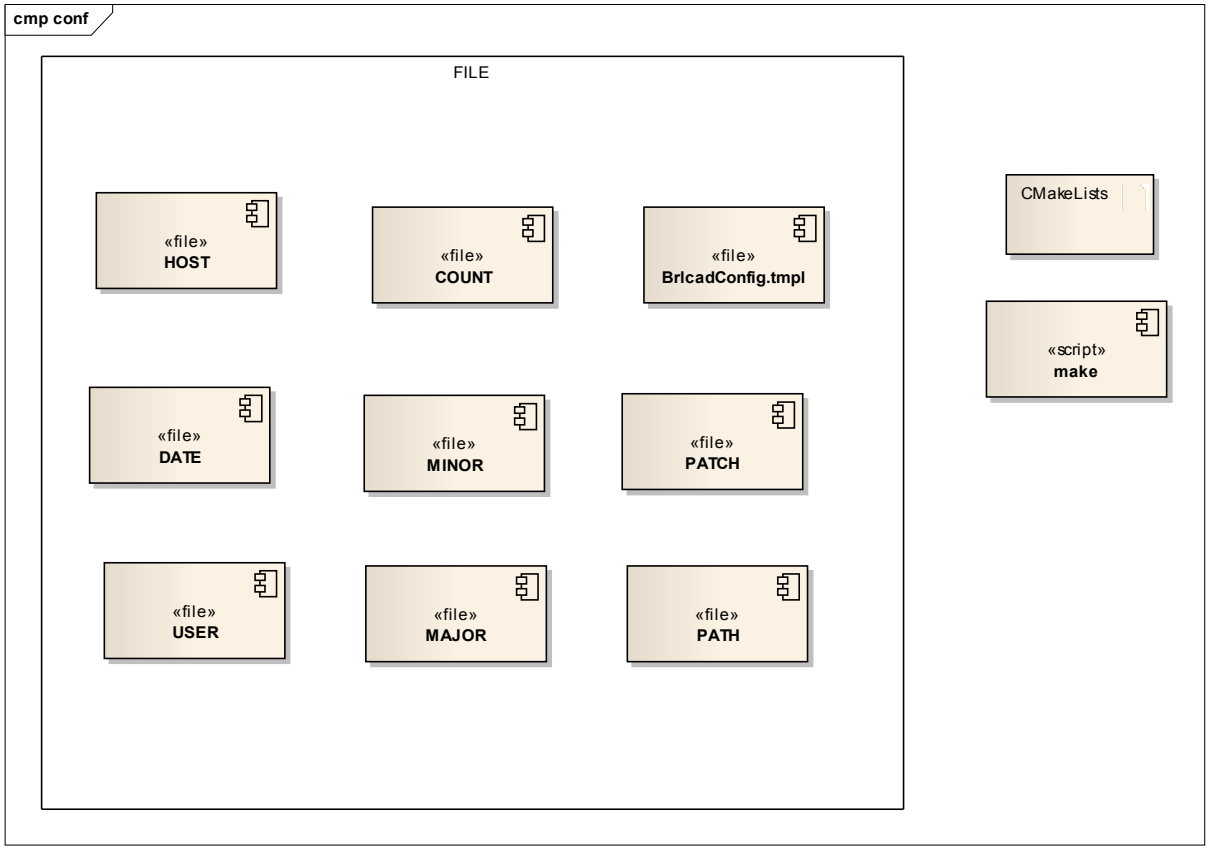


Рисунок 5.34 – МКР представлення папки «config»

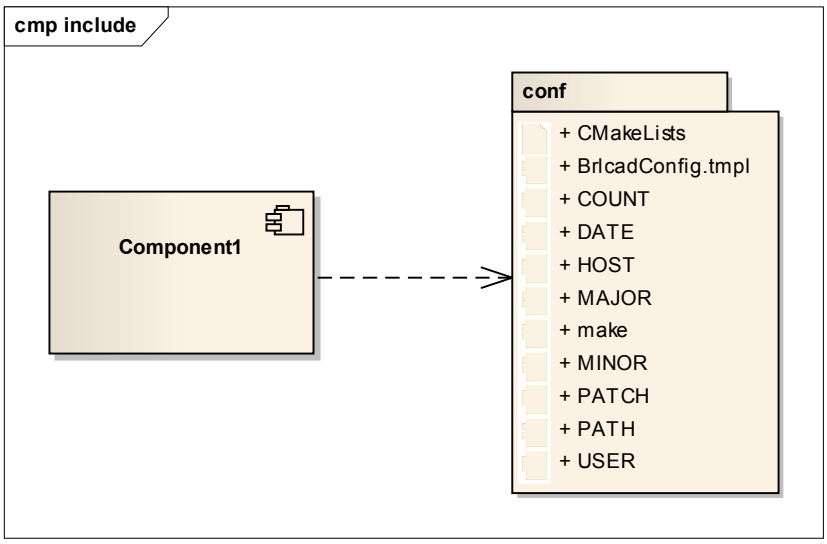


Рисунок 5.35 – Взаємодія всередині папки «include»

Таким же чином реалізовуємо у МКР інші 3 папки: «lib», «bin» та «share». Однак є деякі відмінності. У папці «lib» знаходиться величезна кількість бібліотек, кожну з яких можна представити як самостійний

компонент, але зробити це можна тільки вручну і займає це багато часу, тому підемо шляхом найменшого опору – перенесемо всі бібліотеки до МКР як «артефакти».

Артефакт – це будь-який штучно створений елемент в системі. Артефактами можуть бути які завгодно типи файлів, навіть елементарні блокноти, не кажучи вже про файли з кодом і бібліотеки. У даному випадку буде цілком логічно застосувати структурування саме у такий спосіб, але навіть при тому, що такий хід полегшує роботу, потрібно вказати тип або як він називається у UML – стереотип файлу (рис. 5.36). Відтворюючи кожен артефакт – задаємо його специфікацію.

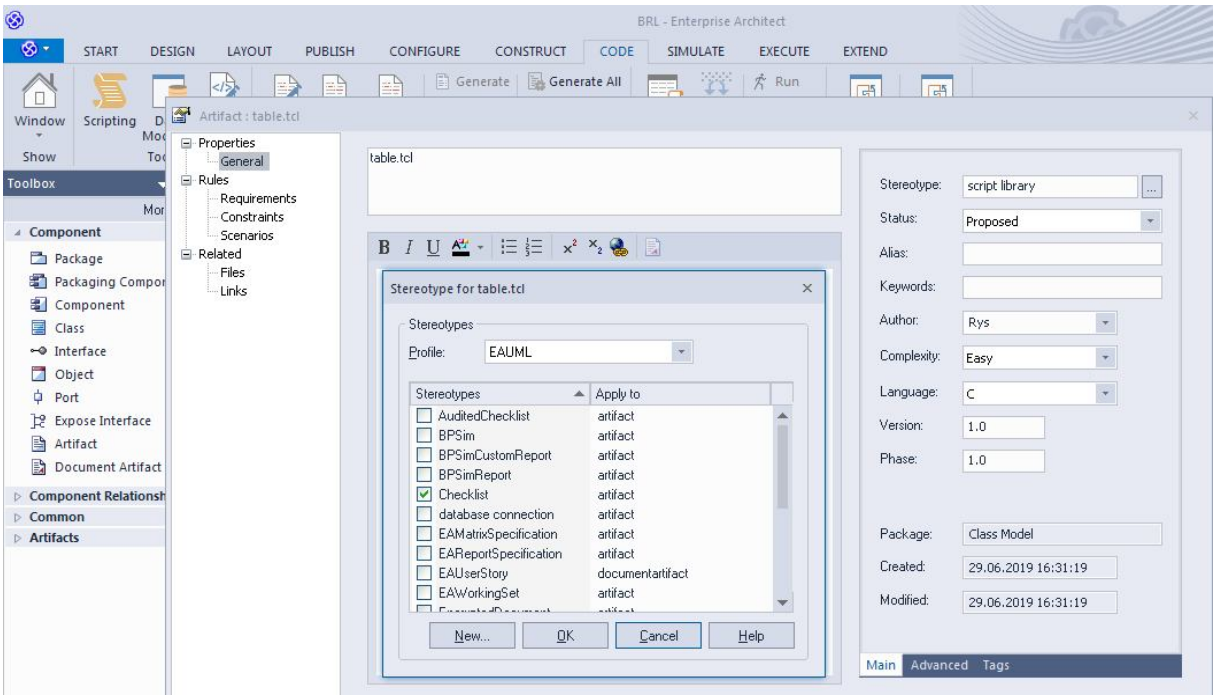


Рисунок 5.36 – Вибір стереотипу і програмної мови

Після відтворення модулів у МКР налаштовуємо зв'язки між ними. Як було видно з рис. 5.32, папка містить файл «Uninstall.exe». Якщо його активізувати, то відбувається деінсталяція САП, тобто цей файл впливає на все інше фізичне (та не тільки) розташування ПС.

В наслідок цього, всі інші папки та файли пов’язані із даним компонентом у вигляді зв'язку «dependency» (залежності). Залишилося тільки пов'язати наші основні пакети. Отже пакет «bin» є папкою з основними та допоміжними «exe»-файлами, а два інших пакети – «lib» та

«include» (бібліотеки та виконавчі файли з кодом) «реалізують» пакет «bin». Для цього є спеціальний тип зв'язку з аналогічною назвою «realize».

Пакети відтворено, зв'язки теж, прив'язка до МОПК виконана. Результат роботи в загальному вигляді наведено на рис. 5.37.

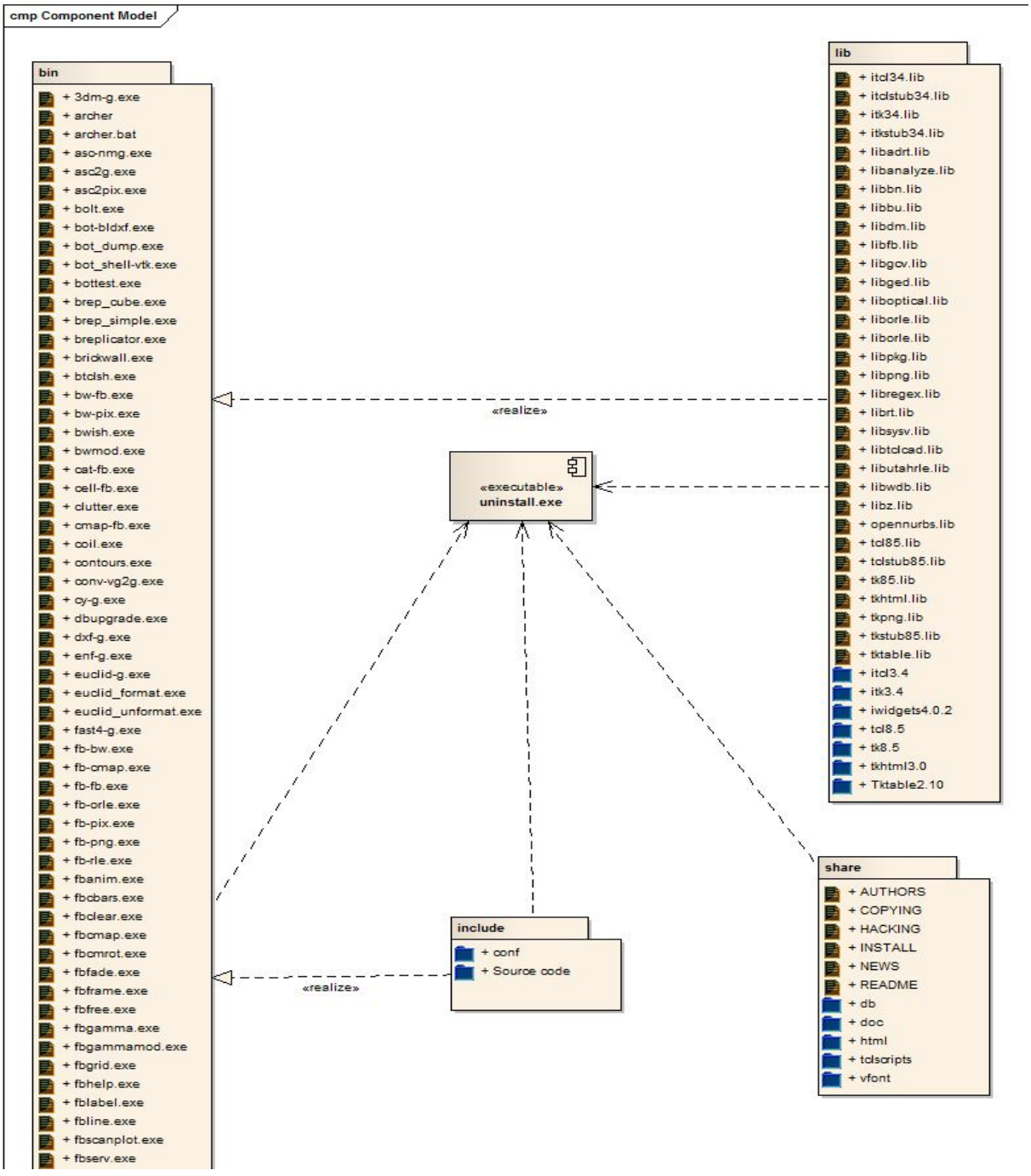


Рисунок 5.37 – Повна МКР BRL-CAD

Крок 5. Переходимо до процесу генерації коду. Зайвим буде зупинятися на подробицях технічного виконання процесу генерації,

оскільки спеціальність, за якою представляється монографія, не включає результати і дослідження закономірностей, що пов’язані з технологічними процесами створення продукту проєкту в будь-якій сфері управління цими процесами, тому проілюструємо тільки кінцевий результат (рис. 5.38). Після генерації коду візьмемо, наприклад, клас «main» (рис. 5.39). Для того, щоб зрозуміти: наскільки правильно згенеровано код, проаналізуємо кількість зв'язків із додатковими модулями. Порівнявши їх із МОПК, приходимо до висновку, що усе згенеровано правильно. На цьому розробку архітектури оновленого ПП можна вважати завершеною.

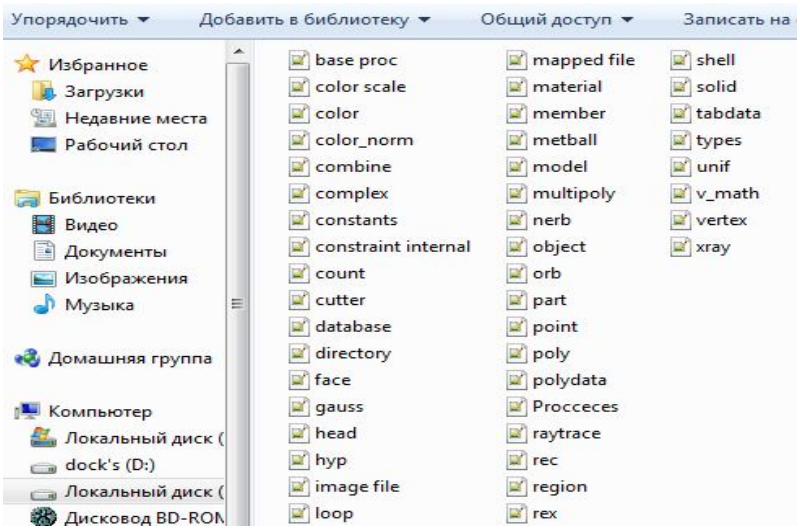


Рисунок 5.38 – Розташування коду

```
12  public class main {
13
14      public rtgeom m_rtgeom;
15      public database m_database;
16      public raytrace  m_raytrace ;
17      public Procceces m_Procceces;
18      public raytrace  m_raytrace ;
19      public shell m_shell;
20      public v_math m_v_math;
21      public base proc m_base proc;
22      public member m_member;
23
24      public main(){
25
26      }
27
28      ~main(){
29
30      }
31
32      public virtual void Dispose(){
33
34      }
35
36  }//end main
```

C# source file | length : 674 lines : 36 | Ln : 21 Col : 34 Sel : 0 | 0 | Dos\Windows | ANSI | INS

Рисунок 5.39 – Програмний код класу «main»

5.6 Висновки за розділом

У поданому розділі монографії вирішено встановлене завдання узагальнення результатів комплексних експериментальних досліджень з проактивного УП з розвитку ПС та формулювання рекомендацій із розробки системної архітектури ПП, що планується розвивати. Дослідження були виконані та змодельовані їх результати за допомогою проєктної методології UML 2.5 із використанням CASE-засобу Enterprise Architect. Методологія UML є досить об'ємною та в УП було розглянуто декілька удосконалених моделей, які використовуються для управління проєктуванням оновленого ПП.

Сьогодні, процес проєктування нових ПП є не ефективним без використання UML-методології, але при її використанні – швидкість розробки підвищується у рази. У розділі монографії спосіб мультилінгвістичного перекодування проєктів ПС був не тільки наведений, але і проаналізований. Проблематикою цієї галузі для нашої країни є відсутність будь-яких навчальних матеріалів, така ситуація хоч і дозволяє користуватися ПП та будувати чи відновлювати код, але приховує деякі цікаві і корисні можливості від користувачів, тому відмінною особливістю отриманих практично-орієнтованих результатів стала систематизація процесу та узагальнення логіки проактивного розвитку УП ПП, що складає основу наукової парадигми, за якою системний архітектор може з'ясувати принципи проєкту реінжинірингу ПС.

Проєкт BRL-CAD послужив відмінним експериментальним зразком для виконання досліджень із притаманними для відкритих проєктів перевагами копіювання, модифікації та інших дій, що пов'язані із внесенням змін до програмного коду. Так само слід зазначити, що адже проєкт зроблено відкритим, то розробники намагалися, щоб УП та його розвиток був зрозумілим – цьому сприяє велика кількість коментарів у програмному коді.

У найпрогресивніших країнах світу вже давно проєкти нових ПП не розроблюються «із нуля», для цього використовуються інструментальні засоби, які допомагають набагато швидше та ефективніше створити будь-яку потрібну структуру. Методологія UML та супутні CASE-засоби служать саме для цієї мети – підвищити ефективність розробки та структурувати проєктні дані. Активно використовуватись ця методологія почала досить недавно (не більше 10 років), але дуже швидко

інтегрувалася у загальну структуру розробки ПП. Зручність же сформованої методології реінжинірингу ПС в тому, що вона жорстко не прив’язана до жодного із методів розробки та є гнучкою у використанні та УП.

Розвиток методології UML для проєктування ПС характерний для Заходу та частини Європи. На початку проведення досліджень (тобто у 2009 р.), спеціалісти нашої країни тільки розпочали роботи із активної експлуатації цієї методології у тому вигляді, в якому її представлено зараз. Основний акцент поставлено на чотири моделі: МВВ, МУП, МОПК та МКР. Це пов'язано із тим, що безпосередньо на підставі цих моделей, відбувається УП, генерація коду та подальша робота проєктної команди, у той час як інші (допоміжні) моделі призначено для пояснення деяких складних специфікацій проєкту, що втім не зменшує їх значущість у рамках УП.

Підсумовуючи досягнуті результати, можна констатувати, що у поданому розділі монографії:

– виконано узагальнення результатів експериментальних досліджень на рівні представлення проєктних МВВ, МУП, МОПК та МКР, які подано за допомогою уніфікованої мови моделювання – UML із обробкою та інтерпретацією результатів на рівні CASE-засобів;

– проаналізовано та узагальнено результати перекладу вихідного коду, головним з яких стало скорочення працемісткості створення проєкту відкритої ПС на конкретному прикладі.

Одним із великих недоліків проєкту BRL-CAD є відсутність зрозумілого графічного інтерфейсу ПП, виправлення та удосконалення якого передбачається у перспективах проєкту реінжинірингу цієї ПС. Проте проєктуванням інтерфейсу розробники та дизайнери зазвичай займаються посередині, а то й ближче до закінчення проєкту, коли вже точно відомий повний функціонал та принципи роботи ПС.

Розроблені рекомендації стануть у нагоді керівникам проєктів, проєктним менеджерам, системним архітекторам та інженерам-програмістам, які задіяні у перепроєктуванні ПС та ПП, що позитивно зарекомендували себе впродовж ЖЦ їх експлуатації.

Перспективи розвитку наведених досліджень полягають у створенні моделей проєктів реінжинірингу для кожного з інших видів забезпечення ПС, що будуть перепроєктовані. УП реінжинірингу ПС дозволить подолати протиріччя між темпами розвитку науки і техніки та процесів

проєктування, підвищити ефективність технічного супроводу проєктів ПП, скоротити експлуатаційні витрати.

Основні власні наукові дослідження, що подані автором у п’ятому розділі монографії, опубліковано у наукових працях автора: [1] – [3], [6], [11], [15], [30], [31], [42], [68].

6 ПРОГРАМНО-МЕТОДИЧНИЙ КОМПЛЕКС УПРАВЛІННЯ ПЛАНУВАННЯМ РЕІНЖИНІРИНГУ ПРОГРАМНИХ СИСТЕМ ТА ПРОДУКТІВ ДЛЯ КОМАНДИ ПРОЄКТНОГО МЕНЕДЖМЕНТУ У СКЛАДІ ОФІСУ УПРАВЛІННЯ ПРОЄКТАМИ

Як відомо, проєктування – це процес створення проєкту, прототипу, прообразу майбутнього об'єкта, стану та способів його виготовлення. У техніці під проєктуванням розуміють розробку проєктної, конструкторської та іншої технічної документації, призначеної для забезпечення створення нових видів та зразків. В УП застосовують системний підхід, який полягає у встановлені структури системи, типу зв’язків, визначені атрибутів, аналізуванні впливів зовнішнього середовища. В процесі УП виконуються технічні та економічні розрахунки, схеми, графіки, пояснювальні записки, кошториси, калькуляції та описи.

Однією з головних складових проєктування – є планування, що визначається як заздалегідь намічений порядок дій або оптимальний розподіл ресурсів, необхідних для досягнення поставленої мети. Функції планування та контролю проєктних операцій виконує команда проєктного менеджменту, що входить до складу офісу УП.

6.1 Складання словника предметної галузі мережевого планування

Для розробки ПМК управління плануванням проєктами реінжинірингу ПС та ПП, необхідно однозначно визначити поняття та, на підставі цих понять, скласти словник предметної галузі.

Основні поняття, що використовуються у мережевому плануванні [138], [140], [143], [146], [149] зведено до табл. 6.1.

Таблиця 6.1 – Поняття та визначення мережевого планування

Поняття	Визначення
Види робіт:	
Дійсна робота	Робота, що вимагає витрат праці, матеріальних ресурсів та часу (наприклад: підготовка траси змагань)
Очікування	Робота, що не вимагає витрат праці і матеріальних ресурсів, але займає деякий час

Продовження табл. 6.1

Поняття	Визначення
Фіктивна робота	Зв'язок між двома або більше подіями, який не вимагає витрат праці, матеріальних ресурсів і часу, але вказує, що можливість початку однієї операції безпосередньо залежить від виконання іншої; тривалість такої роботи: 0
Ранній початок роботи	Термін, раніше якого не можна почати цю роботу, не порушивши прийнятої технологічної послідовності; визначається найбільш довгим шляхом від вихідної події до початку даної роботи
Пізнє закінчення роботи	Найпізніший термін закінчення роботи, при якому не збільшується загальна тривалість робіт; визначається найкоротшим шляхом від даної події до завершення всіх робіт
Раннє закінчення	Термін, раніше якого не можна закінчити цю роботу, дорівнює ранньому початку плюс тривалість даної роботи
Пізній початок	Термін, пізніше якого не можна починати цю роботу, не збільшивши загальну тривалість проєкту, дорівнює пізньому закінченню мінус тривалість даної роботи
Види подій:	Якщо подія є закінченням лише однієї роботи (тобто до неї спрямована тільки одна стрілка), то раннє закінчення цієї роботи збігається з раннім початком наступної
Початкова подія	Початок виконання комплексу робіт
Завершальна подія	Кінцева подія, що означає досягнення кінцевої мети комплексу робіт
Проміжна подія	Результат однієї або декількох робіт, що представляють можливість почати одну або декілька безпосередньо наступних робіт; тривалість проміжної події у часі завжди: 0
Види шляхів:	
Повний шлях	Початок якого збігається з вихідною подією мережі, а кінець – із завершальною

Кінець табл. 6.1

Поняття	Визначення
Шлях, що передує події	Шлях від вихідної події мережі до даної події
Шлях, наступний за подією	Шлях, що з'єднує подію із завершальною подією
Шлях між подіями *i* та *j*	Шлях, що з'єднує будь-які дві події *i* та *j*, з яких жодна – не є вихідною або завершальною подією мережевого графіка
Критичний шлях	Шлях, який має найбільшу тривалість від вихідної події до завершальної
Критична задача	Будь-яка задача на критичному шляху
Загальний (повний) резерв	Найбільший час, на який можна затримати виконання даної роботи, не збільшуючи загальну тривалість робіт; визначається різницею між пізнім і раннім початком (або пізнім і раннім закінченням, що те ж саме).
Вільний резерв	Найбільший час, на який можна затримати виконання даної роботи, не змінюючи раннього початку наступної; можливий тільки тоді, коли до події входять дві або більше роботи (залежності), тобто на неї спрямовані дві або більше стрілки. У цьому випадку, раннє закінчення збігатиметься із раннім початком наступної роботи, для інших же це будуть різні значення; різниця у кожної роботи і буде її вільним резервом.

На відміну від мережевого планування, теорія графів не має стійкої термінології. В різних статтях під одними й тими ж термінами розуміють різні поняття. Наведені нижче визначення є одними з найбільш розповсюджених.

Граф або *неорієнтований граф G* – це впорядкована пара $G := (V, E)$, для якої виконуються наступні умови [226]:

V – множина вершин або вузлів;

---

226. Фляйшнер Г. Эйлеровы графы и смежные вопросы. Москва: Мир, 2002. 176 с.

E – множина пар (у випадку неорієнтованого графу – невпорядкованих) вершин, які називають ребрами.

V (і так само *E*) зазвичай вважаються *скінченними* множинами. Велика кількість результатів, отриманих для *скінченних графів*, невірна (або інша) для *нескінченних графів*. Це пов'язано з тим, що певний набір ідей стає хибним у випадку нескінченних множин.

Граф (*геометричний граф*) — це фігура на площині, яка складається з непорожньої скінченної множини *V* точок (вершин) і скінченної множини *E* орієнтованих чи неорієнтованих ліній (*ребер*), що з'єднують деякі пари вершин [227].

Граф – це сукупність об'єктів із зв'язками між ними [136]. Об'єкти розглядаються як вершини, або вузли графу, а зв'язки – як дуги, або ребра (рис. 6.1).

Для різних областей використання види графів можуть відрізнятися орієнтованістю, обмеженнями на кількість зв'язків і додатковими даними про вершини або ребра. Велика кількість структур, які мають практичну цінність в математиці та інформатиці, можуть бути представлені графами [228]. Наприклад, будову Вікіпедії можна змоделювати за допомогою орієнтованого графу, в якому вершини – це статті, а дуги (орієнтовані ребра) – посилання на інші статті.

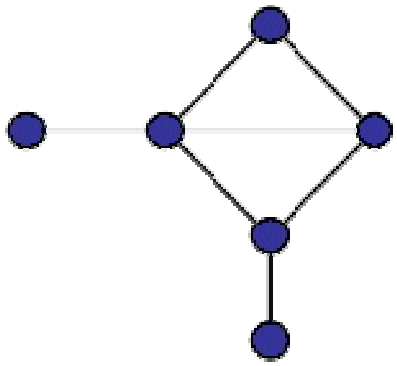


Рисунок 6.1 – Приклад графа із 6-ма вершинами та 7-ма ребрами

227. Татт У. Теория графов. Москва: Мир, 1988. 424 с.

228. Емеличев В. А., Мельников О. И., Сарванов В. И. Лекции по теории графов. Изд. 2, испр. Москва: Наука, 2009. 392 с.

Першою працею з теорії графів як математичної дисципліни вважають статтю Леонарда Ейлера [229], у якій розглядалася задача про Кенігсбергські мости [230]. Наступний імпульс теорія графів отримала близько 100 років потому з розвитком досліджень по електричним мережам, кристалографії, органічній хімії та іншим наукам [231].

Граф, який містить тільки ребра називається *неорієнтованим*, який містить тільки дуги – *орієнтованим*. Граф, що має як ребра так і дуги, називається *мішаним*. Якщо пара вершин сполучається кількома ребрами чи дугами одного напрямку, то ребра (дуги) називають *кратними* (*паралельними*). Дуга чи ребро що сполучає вершину саму із собою називається *петлею*. Граф без кратних дуг і петель називається *простим* [232].

Вершини сполучені ребром чи дугою називають *суміжними*, також називають суміжними ребра, що мають спільну вершину. Ребро (чи дуга) і її вершина називаються *інцидентними* [233]. Ребро (*u, v*) з'єднує вершини *u* та *v*, дуга (*u, v*) починається у вершині *u* й закінчується у вершині *v*.

Кожен граф можна відобразити в евклідовому просторі множиною точок, які відповідають вершинам, сполучених лініями, що відповідають ребрам (дугам). Іноді є потреба пару вершин з'єднати більше, ніж одним ребром. *Мультиграфом* називають пару $G = (V, E)$, де V – множина, елементи якої називають вершинами. E – родина ребер, кожне з яких – це пара вершин із V. Мультиграф, який може мати петлі, іноді називають *псевдографом* [234].

Графи за їх типом й характеристиками за ребрами та їх кратністю наведені у табл. 6.2.

---

229. Eulero L. Solvtio Problematis Ad Geometiam Sitvs. Pertinentis. Avctore. URL: http://eulerarchive.maa.org//docs/originals/E053.pdf (дата звернення: 17.06.2019).

230. Paoletti T. Leonard Euler's Solution to the Konigsberg Bridge Problem. URL: http://mathdl.maa.org/convergence/1/?pa=content&sa=viewDocument&nodeId=1310&bodyId=1452 (дата звернення: 17.06.2019).

231. Харари Ф. Теория графов. Москва: Мир, 1973. 316 с.

232. Jungnickel D. Graphs, Networks and Algorithms. Fourth Edition. Berlin : Springer, 2013. 677 p. DOI: https://doi.org/10.1007/978-3-642-32278-5.

233. Салий В. Н., Богомолов А. М. Алгебраические основы теории дискретных систем. Москва: Физ.-мат. литература, 1997. 424 с.

234. Ловас Л., Пламмер М. Прикладные задачи теории графов. Москва: Мир, 1998. 656 с.

Таблиця 6.2 – Типи графів

Тип графу	Ребра	Кратні ребра
Простий граф	Неорієнтовані	Ні
Мультиграф	Неорієнтовані	Так
Орієнтований граф	Орієнтовані	Ні
Орієнтований мультиграф	Орієнтовані	Так

Якщо мережу автомобільних доріг, ліній комунікацій чи будь-який граф взагалі треба використати в комп'ютерній програмі, то постає проблема збереження інформації про цей граф у пам'яті комп'ютера. Вибір структури даних для збереження графа в пам'яті має визначальне значення у процесі розробки ефективних алгоритмів [235].

У подальшому будемо припускати, що вершини графа мають номери від 1 до n, а ребра – від 1 до m [236]. Кожному ребру і кожній вершині зіставлена вага – ціле позитивне число.

Матриця суміжності – це таблиця, де як стовпці, так і рядки відповідають вершинам графа. У кожній клітинці цієї матриці записується число, що визначає наявність зв'язку від вершини-рядка до вершині-колонки (або навпаки) [237].

Недоліком є вимоги до пам'яті, прямо пропорційні квадрату кількості вершин.

Матриця інцидентності – матриця, де кожен рядок відповідає певній вершині графа, а стовпці відповідають зв'язкам графа [238]. У комірку на перетині i-го рядка з j-м стовпцем матриці записуються:

– «1» – у випадку, якщо зв'язок «виходить» з вершини;

– «-1» – якщо зв'язок «входить» у вершину;

– «0» – у всіх інших випадках (тобто якщо зв'язок є петлею або зв'язок не інцидентна вершині).

Даний спосіб – є найвимогливішим, з точки зору місткості для зберігання, але полегшує знаходження циклів у графі.

---

235. Кормен Т. М. Алгоритмы для работы с графами. Ч. VI. Алгоритмы: построение и анализ. 2-е изд. Москва: Вильямс, 2006. 1296 с.

236. Сик Дж., Ли Л., Ламсдэйн Э. C++ Boost Graph Library. Санкт-Петербург: Питер, 2006. 304 с.

237. Sedgewick R. Algorithms in Java, Third Edition, Part 5: Graph Algorithms. Boston : Addison Wesley, 2003. 528 p.

238. Кирсанов М. Н. Графы в Maple. Москва: Физматлит, 2007. 168 с.

Список ребер – це тип подання графа у пам'яті комп'ютерної програми, який передбачає, що кожне ребро представляється двома числами – номерами вершин цього ребра [239]. Список ребер більш зручний для реалізації різних алгоритмів на графах, у порівнянні з матрицею суміжності.

6.2 Спеціалізовані мови опису й побудови графів у програмних системах та формалізація принципів проєктування мережевих графіків

Для опису графів в цілях, придатних для машинної обробки і, одночасно, зручному для людського сприйняття – використовується кілька стандартизованих мов, серед яких: DOT (мова); GraphML; Trivial Graph Format; GML; GXL; XGMML; DGML.

Відзначимо спеціалізовані програми для побудови графів [240]. До найбільш вдалих відносяться комерційні ILOG, GoView, Lassalle AddFlow, LEDA. З безкоштовних можна відзначити Boost Graph Library. Для візуалізації графів можна використовувати Graphviz (на думку експертів, вона добре працює для орграфів) або LION Graph Visualizer [241]. Особливо відзначимо програму «Графоаналізатор» – це російськомовна програма, з вельми простим для користувача інтерфейсом [242].

Виходячи з аналізу використаних джерел та спираючись на предметну галузь роботи, ретельно сформулюємо правила складання мережевих графіків:

– подія визначає стан, а не процес;

– будь-яка робота в мережі з'єднує дві події: попередня (що є для неї початковою) і наступна за нею (кінцева);

– у мережі не може бути робіт, що мають однакові коди;

– у мережі не повинно бути подій, з яких не виходить жодної роботи, якщо тільки ця подія не є для даного графіка завершальною,

---

239. Глушков В. М., Амосов Н. М., Артёменко И. А. Энциклопедия кибернетики. Т. 2. Киев: Главная редакция УСЭ, 1974. 624 с.

240. Коммерческие специализированные программы для построения графов. URL: http://www.boost.org/ (дата звернення: 02.04.2019)

241. LION Graph Visualizer. URL: http://lion.disi.unitn.it/intelligent-optimization/visualizer.html (дата звернення : 03.04.2019).

242. Графоанализатор – среда визуализации графов. URL : http://grafoanalizator.unick-soft.ru/ (дата звернення: 03.04.2019).

відповідно, у мережі не повинно бути події, до якої не входить жодної роботи, якщо тільки ця подія не є вихідною;

– у МГ не повинно бути замкнутих контурів;

– у МГ не повинно бути тупикових ділянок, кожна подія має з'єднуватися суцільною або пунктирною стрілкою (або стрілками) з будь-якою попередньою (одною або декількома) та подальшою (одною або декількома) подією;

– нумерація подій виконується, приблизно, у тій послідовності, у якій вони відбуватимуться;

– початкова подія розташовується, зазвичай, з лівого боку графіка, кінцева – з правого;

– послідовність стрілок, у якій початок кожної наступної стрілки збігається із кінцем попередньої – називається шляхом;

– шлях позначається у вигляді послідовності номерів подій;

– у МГ між початковою і кінцевою подіями може бути кілька шляхів;

– шлях, що має найбільшу тривалість – називається критичним;

– критичний шлях визначає загальну тривалість робіт;

– всі інші шляхи мають меншу тривалість, і тому – роботи, що у них виконуються, мають резерви часу;

– критичний шлях позначається на МГ потовщеними або подвійними лініями (стрілками).

Крім описаного типу мережевих графіків, в якому вершини графа («кола») відображають події, а стрілки – роботи, існує інший тип, у якому вершинами є роботи. Різниця між цими типами непринципова – всі основні поняття (ранній початок, пізнє закінчення, загальні і вільні резерви, критичний шлях та ін.) зберігаються незмінними, відрізняються лише способи їх запису.

Побудова мережевого графіка цього типу ґрунтується на тому, що ранній початок подальшої роботи дорівнює ранньому закінченню попередньої. Якщо цій роботі передує кілька робіт, її раннє начало має дорівнювати максимальному ранньому закінченню попередніх робіт.

Розрахунок пізніх термінів ведеться у зворотному порядку – від завершальної до початкової, як і у МГ «вершини – події». У завершальній роботі пізніше і раннє закінчення збігаються і відображають тривалість критичного шляху. Пізніший початок подальшої роботи дорівнює

пізньому закінчення попередньої. Якщо за даною роботою слідує кілька робіт, то визначальним є мінімальне значення з пізніх початків.

Коригування мережевих графіків проводиться як на етапі їх складання, так і використання. Вона полягає в оптимізації робіт за часом та за ресурсами (зокрема, за рухом робочої сили). Якщо, наприклад, МГ не забезпечує виконання робіт у необхідні терміни (нормативні або встановлені контрактом) – проводиться його корегування за часом, тобто скорочується тривалість критичного шляху.

Зазвичай це робиться:

– за рахунок резервів часу некритичних робіт і відповідного перерозподілу ресурсів;

– за рахунок залучення додаткових ресурсів;

– за рахунок зміни організаційно-технологічної послідовності і взаємозв'язку робіт.

В останньому випадку у графіків «вершини – події» доводиться міняти їх конфігурацію (топологію).

Коригування за ресурсами проводиться шляхом побудови лінійних календарних графіків за ранніми початками, відповідними тому або іншому варіанту мережевого графіка, і коригування цього варіанту.

Мережеві графіки «вершини – роботи» з'явилися пізніше графіків «вершини – події», тому вони дещо менш відомі і, порівняно, рідше описуються у навчальній і довідковій літературі. Тим не менш, вони мають свої переваги, зокрема їх легше будувати і легше коригувати. При коригування графіків «вершини – роботи» їх конфігурація не змінюється, проте у графіків «вершини – події» такі зміни виключити не вдається.

Тому у поданій монографії, одним з завдань є розробити програмно-методичний комплекс автоматизованого проєктування, складання та коригування мережевих графіків. Створюється він для користувача, якому важливо знати лише послідовність робіт та їх резерви часу, і не має особливого значення, яким способом сформований графік, тобто якого він типу.

Слід зазначити, що хоча у сучасних платних спеціалізованих пакетах комп'ютерних програм планування і оперативного управління, в основному, використовується тип графіків «вершини – роботи», розроблений програмний засіб зможе працювати з усіма типами мережевих графіків із можливостями їх всебічної трансформації.

6.3 Проєктування системної архітектури програмного засобу управління мережевим плануванням

У проєктному підрозділі монографії розглядаються рішення, що запропоновані розробником. Зміст проєктної частини визначається, по-перше, специфікою теми монографії, по-друге, особливостями конкретних технічних пропозицій до роботи.

У поданому підрозділі проєктується архітектура (проєктний «каркас») програмного засобу, що розроблюється, у вигляді декількох діаграм різної природи, виконаних із дотриманням нотації UML [243].

6.3.1 Модель варіантів використання

Розробку системної архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проєкту (далі – ПЗ) розпочнемо із проєктування МВВ.

МВВ використовуються для надання аналітику детальної уяви про галузь застосування ПЗ, що розроблено [244]. З МВВ стає зрозуміло для чого призначений ПЗ, які підсистеми та модулі він має, якими зв’язками поєднані елементи та сутності у ПЗ.

Центральним елементом МВВ – є МГ, який зображено у вигляді актору зі стереотипом «business actor» (рис. 6.2). Від актору «МГ» відходять різноманітні зв’язки, більшість з яких асоціації зі стереотипом «uses» (використання), проте також присутні залежності.

Розглянемо, спочатку, сутності, об’єднанні з МГ першим типом зв’язків. Таких на МВВ міститься 9 – це:

– кооперація використання «Біологія» з асоціаціями «extended» (розширення) на: «Ланцюги харчування», «Екосистеми» та «Генетичні послідовності»;

– сукупність кооперацій використання «Археологія» та «Геологія», об’єднаних межею «Порядок вивчення пластів»;

– кооперація використання «Генеалогічні дерева»;

---

243. Ларман К. Применение UML 2.0 и шаблонов проектирования. 3-е изд. Москва: Вильямс, 2006. 736 с.

244. Шмуллер Дж. Освой самостоятельно UML 2.0 за 24 часа. Практическое руководство. Москва: Вильямс, 2005. 416 с.

– кооперація «Хімічні речовини» з асоціаціями «extended» (розширення) на: «Послідовність з’єднання» та «Будова молекул»;

– кооперація використання «Соціальні мережі» з асоціаціями «extended» (розширення) на: «Соціальні групи» та «Зв’язки»;

– кооперація використання «Карта шляхів» з асоціаціями «extended» (розширення) на: «Залізничні» та «Автомобільні»;

– кооперація використання «Файлова система комп’ютера» з асоціаціями «extended» (розширення) на: «Операційну систему» та «Порядок виконання файлів»;

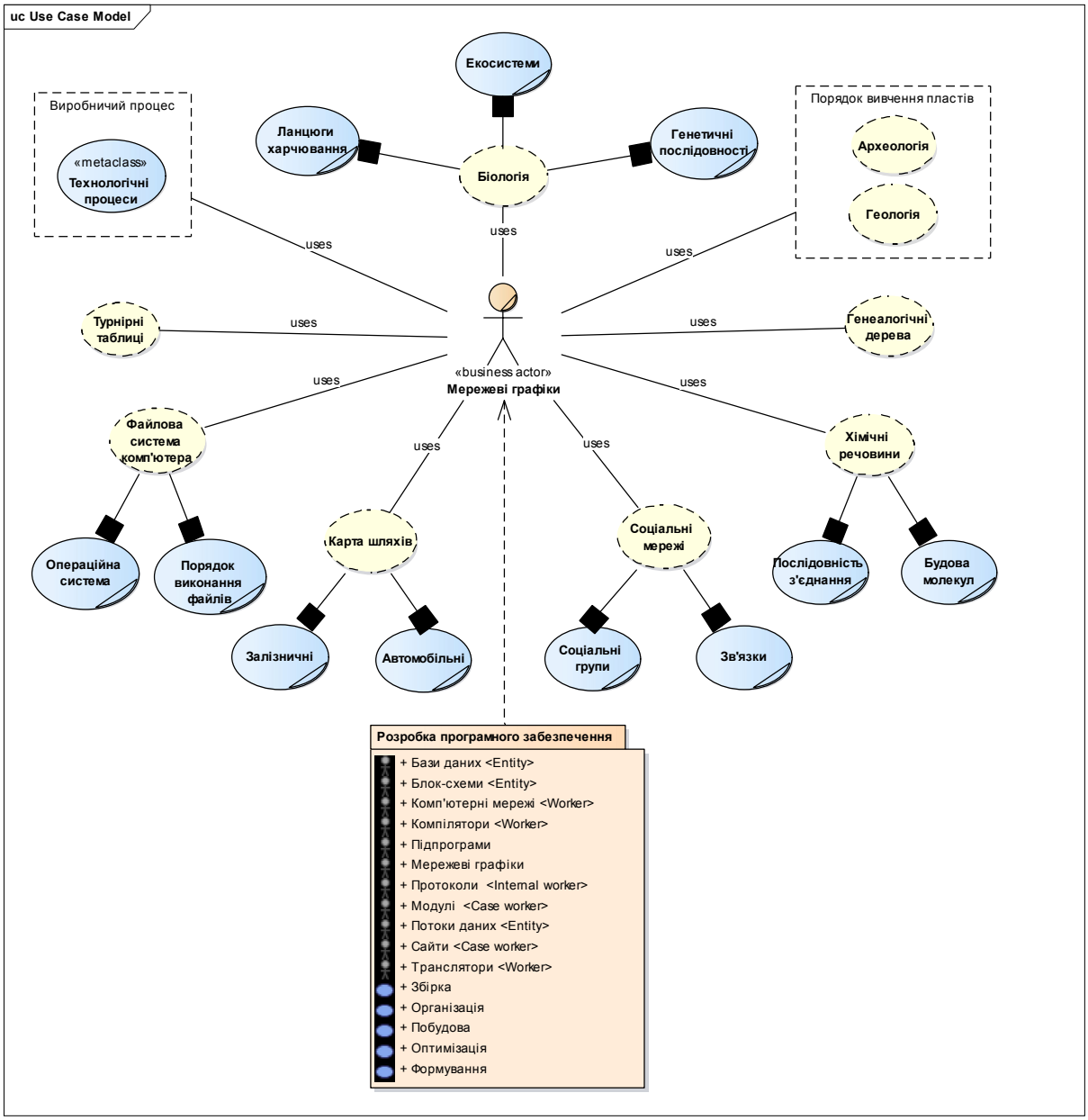


Рисунок 6.2 – Модель варіантів використання МГ

– кооперація використання «Турнірні таблиці»;

– метаклас «Технологічні процеси», у структурі межі «Виробничий процес».

З цього переліку видно, що МГ використовуються у всіх галузях застосування, пов’язаних із наведеним переліком. Далі розглянемо кожну з перелічених сутностей детальніше із розкриттям сутностей другого рівня деталізації (все стосовно до рис. 6.2).

МГ використовуються у біології та екології (кооперація використання «Біологія») досить давно; прикладами використання можуть бути (асоціація «extended»): ланцюги харчування, екосистеми, генетичні послідовності (ВВ зі стереотипом «business use case»), генетичні карти, таксономічна ієрархія живих організмів тощо.

МГ знаходять своє використання у археології та геології (сукупність кооперацій використання «Археологія» та «Геологія») для аналізу та вивчення стратиграфії (на МВВ – межа «Порядок вивчення пластів»).

МГ застосовуються у подіях, які виконуються протягом років – прикладом може стати послідовність складання генеалогічних дерев (кооперація використання «Генеалогічні дерева»).

МГ знаходять своє застосування у моделюванні хімічних речовин (кооперація використання «Хімічні речовини»), за допомогою яких вивчаються (асоціації «extended») їх послідовності з’єднання та будова молекул (ВВ зі стереотипами «business use case»).

МГ використовуються у побудові соціальних мереж (кооперація використання «Соціальні мережі»), де кожне перебування людини чи соціальної групи (ВВ зі стереотипом «business use case») – є вершиною, а зв’язки – ребрами (асоціації «extended»).

Карта проходження автомобільних чи будь-яких інших шляхів (кооперація використання «Карта шляхів») також є МГ, причому кожна дорога (залізнична чи автомобільна – ВВ зі стереотипами «business use case») може мати певне значення «ваги» (асоціації «extended»), наприклад: щільність транспортного потоку.

Файлова система комп'ютера (кооперація використання «Файлова система комп’ютера») може бути зображена у вигляді МГ, що показує порядок виконання файлів та тек у багатьох операційних системах (ВВ зі стереотипом «business use case»); такий МГ має вигляд дерева з асоціаціями «extended».

Турнірні таблиці спортивних чемпіонатів (кооперація використання «Турнірні таблиці») також можуть бути зображені у вигляді МГ.

Будь-який виробничий процес також може бути зображений за допомогою МГ (див. рис. 6.2, де усе існуюче розмаїття технологічних процесів зображено як метаклас («metaclass») «Технологічні процеси», у пакетній структурі-межі «Виробничий процес».

Перейдемо до розгляду сутності, що об’єднана з МГ типом зв’язку залежність «dependency». На МВВ це пакет «Розробка ПЗ», виконання якого дійсно базується (залежить) від МГ. До складу пакету входять багато сутностей, об’єднаних зв’язками, сукупність яких являє вкладену модель ВВ у вигляді пакету графу (рис. 6.3).

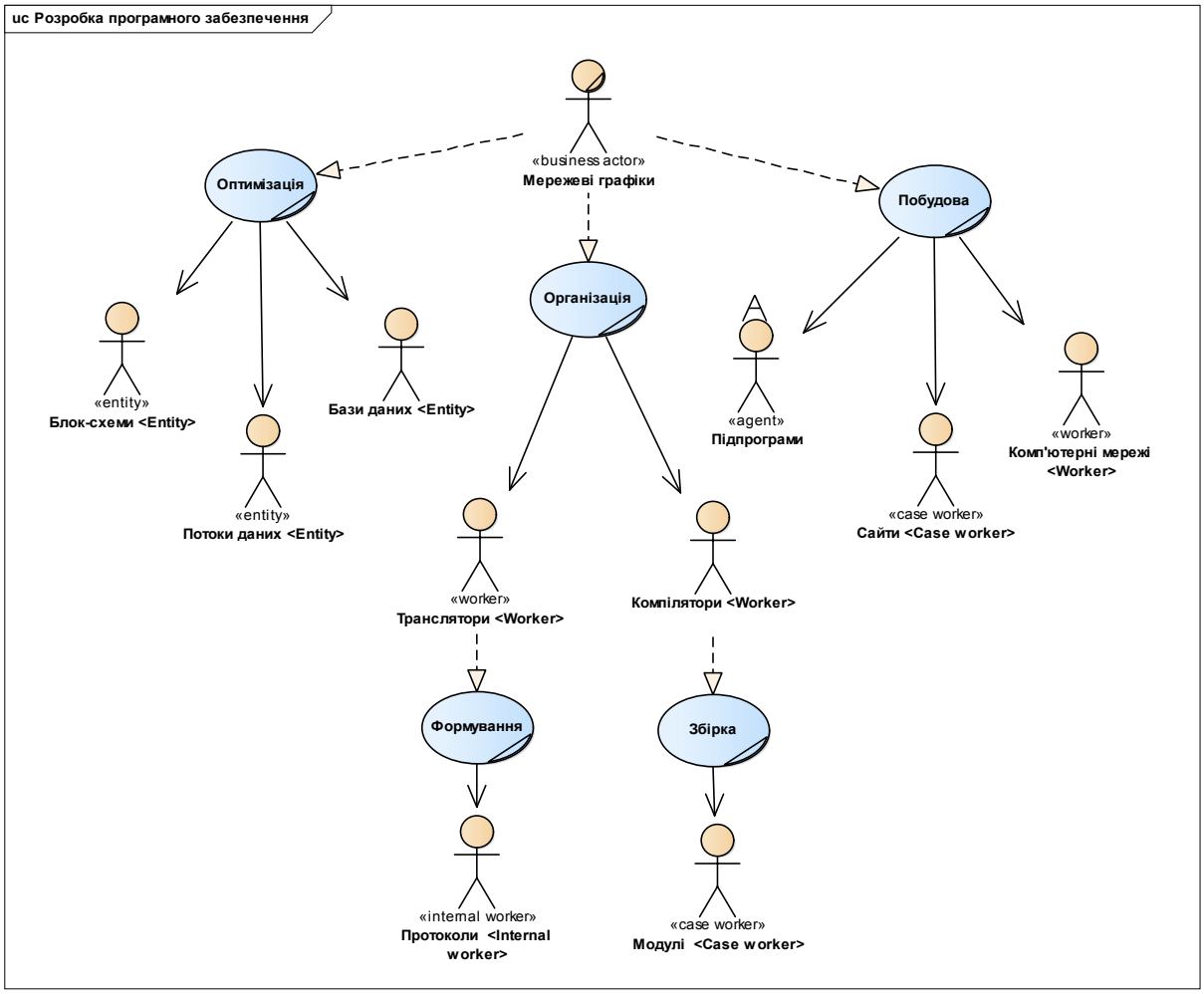


Рисунок 6.3 – Вкладена модель ВВ у вигляді пакету графу «Розробка ПЗ»

Розробка ПЗ та комп'ютерні науки взагалі є однією з тих галузей, де МГ застосовуються найчастіше, тому поданий пакет й винесено у вигляді

окремої залежної структури (рис. 6.3). Складність та велика кількість модулів і протоколів у сучасних програмних продуктах сильно ускладнює розуміння їх роботи, керування нею та її оптимізацію. Тому дуже часто складаються МГ програм, причому найчастіше це робиться автоматично трансляторами чи компіляторами.

МГ також є зручними для зображення структур даних, блок-схем, потоків даних, схем баз даних та баз знань, скінченних автоматів, схем комп'ютерних мереж та окремих сайтів, схем викликів підпрограм тощо. Також МГ широко використовуються у багатьох алгоритмах пошуку та сортування.

Крім того, одним з головних напрямків сучасних досліджень у галузі глобальних мереж – є задане консорціумом W3C – завдання побудови семантичної мережі (один з видів МГ) на базі існуючої мережі Інтернет.

В рамках ПМК – була виконана спроба об’єднати усе вищезазначене у діаграму-пакет «Розробка ПЗ», що наведена на рис. 6.3. Розглянемо детальніше спроєктовану структуру.

МГ (актор зі стереотипом «business actor») реалізує (за допомогою зв’язків «realization») наступні ВВ (зі стереотипами «business use case»), стосовно до ПЗ:

- оптимізація;
- організація;
- побудова.

В свою чергу, кожен з ВВ виконує дії, спрямовані на конкретні сутності, а саме відбувається:

а) оптимізація (з асоціативними зв’язками «association») блок-схем, потоків даних, баз даних (актори зі стереотипами «entity»);

б) організація (з асоціативними зв’язками «association») трансляторів та компіляторів (актори зі стереотипами «worker»), кожен з яких реалізує (за допомогою зв’язків «realization»):

1) формування (ВВ зі стереотипом «business use case») протоколів за внутрішніми принципами (актор зі стереотипом «internal worker», об’єднаний асоціативним зв’язком («association») з ВВ «формування»;

2) збірку (ВВ зі стереотипом «business use case») модулів за допомогою інструментальних засобів (актор зі стереотипом «case worker», об’єднаний асоціативним зв’язком («association») з ВВ «збірка»;

в) побудова (з асоціативними зв’язками «association») підпрограм, як вбудованої структури (актор зі стереотипом «agent»), сайтів та комп’ютерних мереж (актори зі стереотипами «worker»).

6.3.2 Модель послідовності розвитку проєкту

МПР проєкту призначені для формування уяви програміста про порядок виконання дій при роботі з майбутнім ПЗ, які сформовані системним архітектором [245]. Існує МПР двох основних типів: МПР для відображення дій програміста при розробці ПЗ та МПР для відображення дій користувача при подальшій роботі з майбутнім ПЗ. МПР саме другого типу використовуються для створення інструкцій користувача, які є складовою частиною ПМК.

Саме такий тип МПР спроєктовано у поданій роботі. Перейдемо до детального опису сформованої структури (рис. 6.4).

Проєктування МПР починається із визначення об’єктів – сутностей, які розташовані у верхній частині МПР. Критерієм для відбору сутностей є майбутні класи або компоненти із якими взаємодіє користувач (або розробник для першого типу МПР). Двокрапка у структурі назви об’єкту означає майбутній клас.

Кожен об’єкт має так звану «лінію життя» («life line») – пунктирна лінія, що простягається на усю довжину полотна МПР. У початковий момент «лінія життя» – є досить короткою, проте, з додаванням повідомлень до ДП – лінія автоматично збільшується.

Вздовж «лінії життя» ковзає індикатор активності – суцільний прямокутник, що показує довжину процесу активності кожного об’єкта. Саме до індикатора активності спрямовуються усі повідомлення. Цих індикаторів на лінії життя може бути множина, проте, вони мають різну довжину, наприклад індикатор активності користувача (див. рис. 6.4) має найбільшу довжину, що означає його активність протягом дії повідомлень, що розглядаються.

---

245. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. 2-е изд. Москва: ДМК-Пресс, 2004. 432 с.

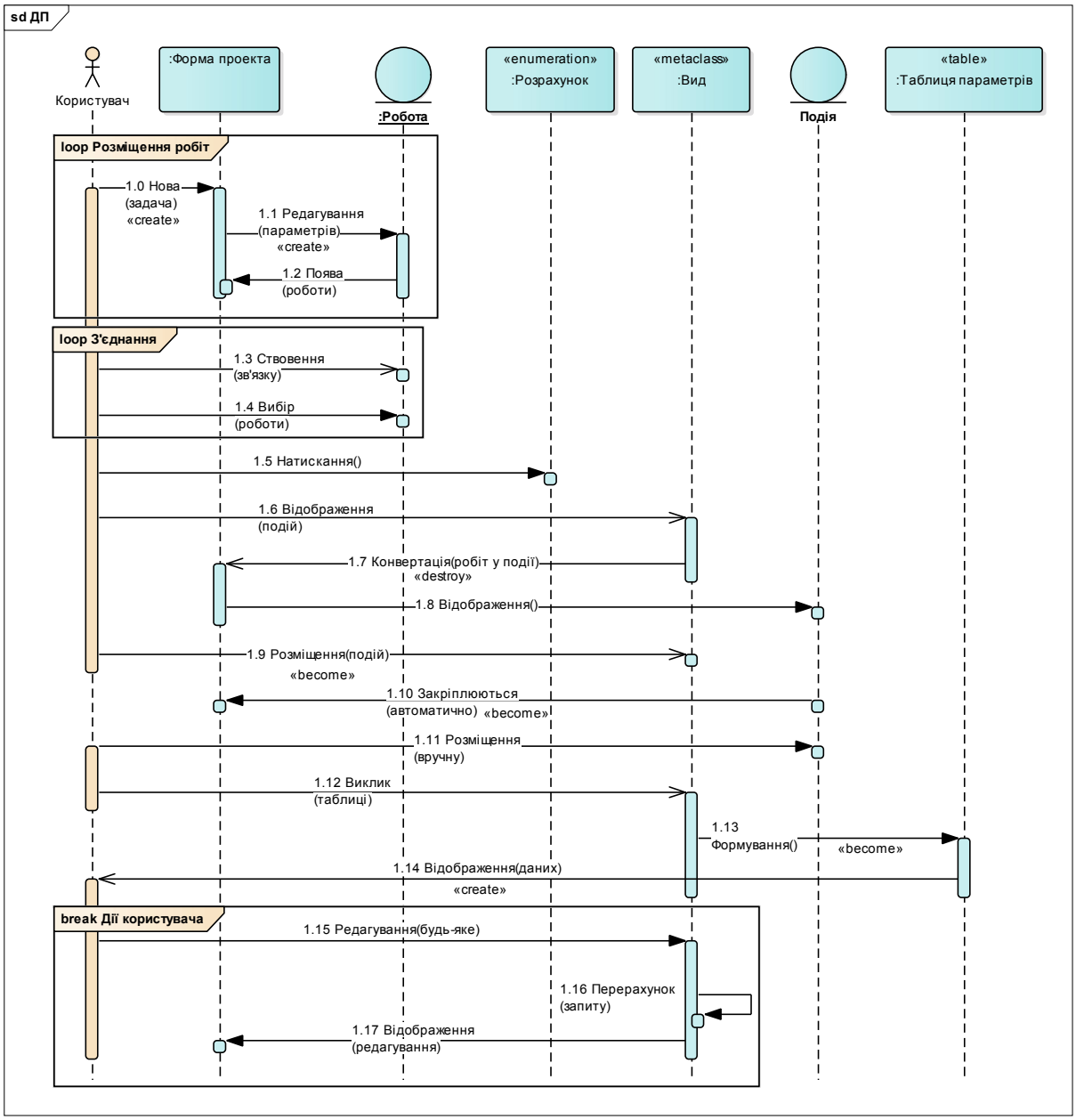


Рисунок 6.4 – Модель послідовності розвитку проєкту

На сформованій МПР присутні наступні об’єкти:

- користувач (актор);
- форма проєкту (клас);
- робота (клас-сутність «entity»);
- розрахунок (обчислювальний клас «enumeration»);
- вид (метаклас «metaclass»);
- подія (клас-сутність «entity»);
- таблиця параметрів (табличний клас «table»)

Перші повідомлення у складі МПР об’єднані у складову частину («combined fragment») «Розміщення робот», що організована за замкненим циклом («loop»). До цього циклу включено повідомлення:

– нова (задача);
– редагування (параметрів);
– поява (роботи).

Розглянемо детальніше ці повідомлення у МПР.

Перше повідомлення «Нова (задача)» має назву «Нова» та аргумент «задача», що вказується у дужках (тут і далі усі аргументи повідомлень будуть відразу подаватися у дужках, без подальшої вказівки, що це є аргументом) зі стереотипом «create» – спрямовано від «Користувача» до «Форми проєкту»; воно показує, що для створення нової задачі користувачеві необхідно звернутися до відповідного класу-форми, представленої інтерфейсом у вигляді робочої області МГ.

Друге повідомлення «Редагування (параметрів)» зі стереотипом «create» – спрямовано від «Форми проєкту» до класу-сутності «Робота»; воно показує, що відразу після створення нової задачі на формі проєкту – необхідно спочатку створити, а потім (за потреби) відредагувати параметри роботи.

Третє повідомлення «Поява (роботи)» з простим стереотипом – спрямовано у зворотному напрямку від «Роботи» до «Форми проєкту»; воно показує, що відразу після внесення змін у параметри роботи – відповідна робота з’явиться на формі проєкту та стане видною для користувача, про що свідчить індикатор активності на лінії життя «Користувач», який не переривається.

На цьому повідомленні завершується складова частина («combined fragment») «Розміщення робот» – розглянутий цикл буде виконано стільки разів, скільки знадобиться користувачеві, виходячи з необхідної кількості створення робіт або його власних задумів.

Наступні повідомлення у МПР об’єднані у наступну складову частину («combined fragment») «З’єднання», що також організована за замкненим циклом («loop»). Ця частина (за повтореннями) схожа з попередньою, також – цикл буде виконано стільки разів, скільки знадобиться користувачеві, тільки вихідною кількістю циклів буде необхідна кількість зв’язків, що створюються, а не робіт. До цього циклу включено повідомлення:

– створення (зв’язку);

– вибір (роботи).

Перше повідомлення «Створення (зв’язку)» з асинхронним («asynchronous») типом керування потоку («Control Flow Type») – спрямовано від «Користувача» до класу-сутності «Робота»; воно показує, що користувач обирає команду «Створити зв'язок» – асинхронність показує відсутність відповіді з боку об’єкту «Робота».

Друге повідомлення «Вибір (роботи)» з простим стереотипом – спрямовано від «Користувача» до класу-сутності «Робота»; воно означає, що користувач створює зв'язок із тією роботою, яку необхідно включити до МГ – відповідна робота з’явиться на формі проєкту та стане видною для користувача, про що свідчить синхронність повідомлення. Це повідомлення є останнім у межах циклу складової частини «З’єднання».

Два наступних повідомлення спрямовано від «Користувача» (рис. 5) Одне – «Натискання» – до обчислювального класу «Розрахунок» – вказує, що користувач може виконати необхідний розрахунок за допомогою виконання обчислення. Друге повідомлення «Відображення (подій)» – до метакласу «Вид» – показує звертання до панелі меню «Вид», що планується з метою відображення топології МГ у вигляді подій. Саме це повідомлення виконано з асинхронним («asynchronous») типом керування потоку («Control Flow Type»).

Наступне повідомлення «Конвертація (робіт у події)» також є асинхронним. Спрямовано від метакласу «Вид» до «Форми проєкту», має руйнуючий стереотип «destroy», що значить руйнування топології МГ з подальшою її перебудовою та за мету якого є виконання конвертації сутностей «Робота» у «Подію».

Слідом, від «Форми проєкту» до «Події», йде службове повідомлення «Відображення», що як раз й виконує графічне представлення спроєктованих робіт у формі нових сутностей у «Подія».

Помітимо, що протягом створення всіх цих повідомлень – індикатор активності «Користувача» – є безперервним (див. рис. 5), що свідчить про обов’язкову участь користувача у всій послідовності подій, що розглянуто.

Далі асинхронне повідомлення «Розміщення (подій)» зі стереотипом звертання («become»), що спрямовано від «Користувача» до метакласу «Вид» – показує завдання на виконання команди розміщення подій, що віддається користувачем.

Відразу від класу-сутності «Подія» до «Форма проєкту» йде синхронне повідомлення «Закріплюються (автоматично)» зі стереотипом

звертання («become»), що означає автоматичне закріплення сформованих сутностей-подій за сіткою форми проєкту.

У разі, якщо топологія виконаного закріплення не задовольняє користувача – він може розмістити події за формою вручну. На це вказує повідомлення «Розміщення (вручну)», спрямоване від «Користувача» до «Події».

Далі (за потребою) користувач може викликати побудову спеціальної таблиці, що містить усі розраховані параметри у числовій формі – асинхронне повідомлення «Виклик (таблиці)», що спрямоване від актора «Користувач» до метакласу «Вид».

Слідом відбувається створення нового об’єкту «Таблиця параметрів», що являє табличний клас «table» – про це свідчить повідомлення «Формування» зі стереотипом звертання («become»), яке спрямовано від метакласу «Вид» до «Таблиці параметрів».

Наступне асинхронне повідомлення «Відображення (даних)» зі стереотипом «create» – виконує відображення створених табличних чисельних даних й спрямовано від «Таблиці параметрів» до «Користувача» тому, що саме йому відображаються сформовані табличні структури.

На завершення створення МПР ПЗ проєктування МГ залишається внести етап роботи користувача, пов'язаний з повним або частковим редагуванням відображення МГ (масштабування, переміщення тощо). При виконанні цього етапу, кожний раз йде перерахунок та перебудова відображення всієї топології МГ, тому організація цього вбудованого фрагменту («combined fragment») буде мати тип «break» (переривання або розлом), а назву – «Дії користувача».

До складової частини («combined fragment») «Дії користувача» – включено наступні повідомлення:

– редагування (будь-яке);
– перерахунок (запиту);
– відображення (редагування).

Перше повідомлення фрагменту, що розглядається – «Редагування (будь-яке)» спрямовано від «Користувача» до метакласу «Вид» – означає виконання користувачем будь-яких змін у редагуванні зображення МГ.

Друге повідомлення фрагменту – «Перерахунок (запиту)» є рефлексивним («self-message»), тобто спрямовано від метакласу «Вид» само до себе – означає виконання перерахунку сформованого користувачем запиту всередині самого метакласу.

Третє повідомлення – «Відображення (редагування)» спрямовано від метакласу «Вид» до «Форми проєкту» – означає виконання відображення наслідків редагування МГ. Протягом створення всіх повідомлень складової частини «Дії користувача» – індикатор активності користувача – не переривається, що свідчить про фіксацію користувачем усієї послідовності подій складової частини та, у разі необхідності, багаторазового повторення редагування.

6.3.3 Модель опису станів

МОС необхідна для наочного подання тих станів ПЗ, у яких він може знаходитися в різні моменти часу. Іншими словами, можна сказати, що стани описують поведінку ПЗ, яка, в свою чергу, може залежати як від вказівок користувача, так і від обчислювальних процедур самого ПЗ.

Як видно з самої назви діаграми – ключовими моментами у ній – є стани (state), які представляють собою, у своєму роді, точки зупинки роботи ПЗ або збору статистичної та технічної інформації (протоколювання). Кожен стан має точки входу та виходу у цей стан, причому, таких точок може бути декілька (як для входів так і для виходів) – саме для фіксування цих позицій у межах стану – існує таке поняття, як «історія стану», яка опційно вмикається у специфікації кожного стану та має позначення: літера «H» (history), що подається у колі.

Крім того, важливими на МОС, також, є й переходи (transfer), що являють собою зв’язки між станами. Переходи зображуються у вигляді стрілок, які спрямовані від стану до іншого стану. Кожен перехід має синтаксис, який розкриває математичний або фізичний процес, що викликав цей перехід. Синтаксис, у свою чергу, крім назви переходу, може містити обмежуючу умову (подається у прямокутних дужках [ ]) – при невиконанні якої – перехід неможливий.

Спроєктовану МОС ПЗ проєктування МГ приведено на рис. 6.5. Розкриємо детально її трактування.

МОС, як видно з рисунку, являє собою каскадну модель задуму створення ПЗ, де у якості поступового каскаду реалізації станів, виступає головна діагональ діаграми з багатьма розгалуженнями (у залежності від точок виходу з конкретних станів) у різні боки.

Початковий стан («Initial state») на МОС зображено у вигляді чорного кола (рис. 6.5), саме з нього починається робота з ПЗ, можна

сказати, що він пов'язаний із завантаженням ПЗ та його готовності до роботи. Від початкового стану виходить перехід «Виклик», що має обмежуючу умову «[Правий щиглик]» – цей стан показує, що появу наступного стану необхідно викликати, причому виконати це можливо тільки за умови виконання користувачем щиглика правою кнопкою миші.

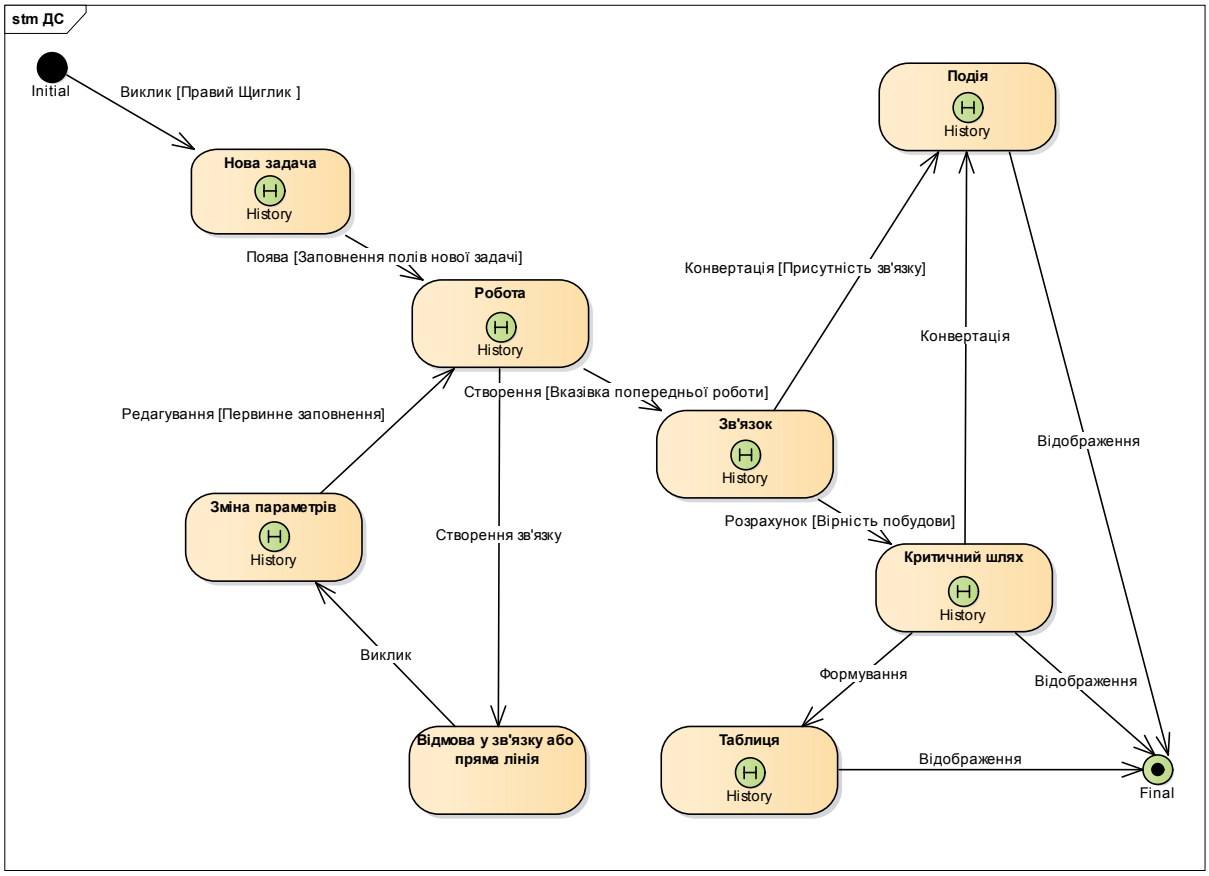


Рисунок 6.5 – Модель опису станів

Другий стан – «Нова задача» – містить у собі активну історію стану («state history»). Стан означає момент створення нової задачі на МГ.

Третій стан – «Робота» – вже містить у собі чотири переходи (два вхідних та два вихідних), саме для протоколювання схожих входів-виходів необхідно увімкнення історії стану. Проте, головними у стані є переходи головної діагоналі: так головним вхідним переходом – є «Поява» цієї роботи з обмежуючою умовою «[Заповнення полів нової задачі]», що значить – у разі незаповнення обов’язкових полів – стан «Робота» не з’явиться.

Розглянемо далі усі елементи головної діагоналі МОС, а після цього – повернемося до подання розгалужень.

Перехід «Створення» з обмежуючою умовою «[Вказівка попередньої роботи]» веде зі стану «Робота» до стану «Зв'язок» та означає створення зв’язку на МГ тільки якщо користувачем буде вказана попередня робота.

Стан «Зв'язок» породжується переходом «Створення» та означає виникнення зв’язків у МГ, що відображені у вигляді стану на МОС. Подальше існування зв’язків забезпечується активною історією стану («state history») «Створення». Також історія стану відповідає за протоколювання точок виходу стану, що розглядається, а таких у стані – дві.

Перехід «Розрахунок» з обмежуючою умовою «[Вірність побудови]» веде зі стану «Зв'язок» до стану «Критичний шлях». Цей перехід означає виконання розрахунку критичного шляху МГ, якщо МГ було вірно збудовано користувачем ПЗ.

Стан «Критичний шлях» породжується переходом «Розрахунок» та означає зображення критичного шляху у топології МГ. Цей стан містить історією стану («state history»), що відповідає за протоколювання точок виходу зі стану «Критичний шлях», а таких у стані – три, у той час, як вхідних – усього одна.

Перехід «Відображення» веде зі стану «Критичний шлях» до кінцевого стану «Final State» та означає виділення критичного шляху МГ, що буде реалізовано у майбутньому ПЗ – кольоровим поданням зв’язків МГ.

До стану «Final State» веде три переходи (див. рис. 6.5), цей стан має три точки входу та жодної – виходу. Саме це означає, що цей стан є останнім на МОС, причому до нього, в кінцевому випадку, стікаються усі розгалуження МОС. Кінцевий стан не містить історії стану тому, що при потраплянні до нього попереднє накопичення історії побудови МГ – не важливе.

Подальший опис МОС почнемо з подання розгалужень.

Першим розгалуження на головній діагоналі МОС відбувається у стані «Робота» (див. рис. 6.5). Як вже було наведено вище: стан має чотири переходи (два вхідних та два вихідних). Другим вихідним переходом, крім головного – є перехід «Створення зв’язку», що говорить сам за себе. Приводе цей перехід зі стану «Робота» до стану «Відмова у зв’язку або пряма лінія».

Потрапляння у стан «Відмова у зв’язку або пряма лінія» відбувається тоді, коли створення зв’язку між обраними роботами – неможливе. На це

може бути багато причин (невірне: обрання роботи, її розміщення, завдання параметрів тощо), проте стан до якого ведуть ці причини – є один. Поведінка цього стану проявляється у наслідках, які видно з самої назви стану: відбувається відмова у зв’язку або зв'язок зображується у вигляді прямої лінії. Цей стан на містить історії стану тому, що немає потреби у протоколюванні зв’язків, що невірно відображені.

Перехід «Виклик» веде зі стану «Відмова у зв’язку або пряма лінія» до стану «Зміна параметрів» та означає виклик діалогу зміни параметрів для усунення помилок зв’язків та подальшого їх вірного відображення.

Стан «Зміна параметрів» породжується переходом «Виклик» та означає редагування невірних параметрів зв’язків МГ. Цей стан містить історією стану («state history»), що відповідає за фіксування редагування.

Перехід «Редагування» з обмежуючою умовою «[Первинне заповнення]» веде зі стану «Зміна параметрів» до стану «Робота». Цей перехід означає завершення редагування параметрів та формування оновленого стану «Робота». Якщо всі параметри, необхідні для створення зв’язку – вірні – відбувається рух за станами головної діагоналі до розглянутого кінцевого стану «Final State». У противному випадку – відбувається повторне виконання циклу станів «Відмова у зв’язку або пряма лінія» – «Зміна параметрів» – «Робота» доти, доки параметри зв’язку не будуть вірними для його побудови.

Другим розгалуження на головній діагоналі МОС відбувається у стані «Зв'язок» (див. рис. 6.5). Як вже було наведено вище: стан має три переходи (один вхідний та два вихідних).

Другим вихідним переходом з цього стану, крім головного – є перехід «Конвертація» з обмежуючою умовою «[Присутність зв’язку]». Приводе цей перехід зі стану «Зв'язок» до стану «Подія». Призначення цього переходу – конвертація МГ з вершинами-роботами до МГ з вершинами-подіями, для виконання конвертації необхідна присутність встановлених зв’язків.

Стан «Подія» породжується, як бачимо, двома переходами «Конвертація», проте спрямованими з різних станів, а саме: «Зв'язок» та «Критичний шлях». Стан «Подія» означає відображення вершин МГ у формі подій, стан містить історією стану («state history»), що відповідає за фіксування моментів конвертації зображення.

Ще один перехід «Конвертація», який веде до стану «Подія» зі стану «Критичний шлях» має аналогічне призначення як і той, що веде до цього

стану зі стану «Зв'язок». Ці два аналогічні переходи наочно показують можливість конвертації з різних станів МГ (з моментів: встановлення зв’язків або розрахунку критичного шляху).

Перехід «Відображення», що веде до стану «Final State» – означає кінцеве відображення подій МГ для подання користувачеві.

Далі розглянемо останній (третій) вихідний перехід «Формування», що веде зі стану «Критичний шлях» до стану «Таблиця». Він призводить до формування чисельних даних параметрів МГ у вигляді табличної структури.

Стан «Таблиця» породжується переходом «Формування» та означає формування табличного відображення параметрів МГ, стан містить історією стану («state history»), що відповідає за накопичення та редагування параметрів, що розглянуто.

Останній перехід на МОС – це перехід «Відображення», що веде зі стану «Таблиця» до кінцевого стану «Final State» – означає відображення чисельних даних МГ для подання користувачу.

Таким чином, у сформованій МОС присутні: 10 станів та 13 переходів, до складних станів, слід віднести стани з вмістом більш ніж двох переходів, а це на розглянутій МОС:

а) стани з трьома переходами:

1) «Зв'язок» з – «Створення», «Розрахунок» та «Конвертація»;

2) «Подія» з – двома «Конвертаціями» та «Відображенням»;

б) стани з чотирма переходами:

1) «Робота» – з «Поява», «Створення», «Редагування» та «Створення зв’язку»;

2) «Критичний шлях» з – «Розрахунок», двома «Відображеннями» та «Формування».

6.3.4 Динамічна модель діяльності

ДМД призначена для відображення діяльності конкретного об’єкту, протягом аналізу дії, що цікавлять [246]. За допомогою ДМД можна виконувати дослідження та протоколювання спрямування інформаційних потоків, які необхідні для створення подальшого ПЗ. ДМД наслідують у собі деякі аспекти з МПР та МОС, проте тільки у ДМД є можливість

246. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд.; пер. с англ.; под общей редакцией О. С. Орлова. Санкт-Петербург: Питер, 2006. 736 с.

спроєктувати повноцінні алгоритмічні цикли, в основі яких будуть стояти блоки «Вирішення» («Decision»).

Головною сутністю ДМД є так звані діяльності (Activity), які являють собою сутність схожу зі станом на МОС. Діяльність показує конкретні дії процесу роботи ПЗ. Також на ДМД присутні переходи, що мають таку ж природу, як і переходи на МОС, з тими ж такі обмежуючими умовами. Крім того, на ДМД присутні доріжки діяльності, що означають виділення умовних меж та зон діяльності кожного з об’єктів.

Спроєктовану ДМД для роботи головного інформаційного потоку («I-Flow») ПЗ проєктування МГ приведено на рис. 6.6.

Розглянемо детальніше її структуру. У якості доріжок діяльності виступають «Користувач», «Інтерфейс» та «Область обчислень та пам'яті». Їх назви подано у верхній частині ДМД, а межі діяльності – відокремлені суцільними лініями. Початкова діяльність («ActivityInitial») знаходиться на доріжці «Користувач» – означає, що саме об’єкт «Користувач» починає поданий інформаційний потік. Ця діяльність зображена у вигляді чорного кола. Від початкової діяльності до доріжки об’єкту «Інтерфейс» – веде перехід до діяльності «Створення нової задачі», яка означає виконання задачі створення роботи на МГ. Далі від діяльності «Створення нової задачі» до доріжки об’єкту «Область обчислень та пам'яті» – веде перехід, що закінчується діяльністю «Запам’ятовування позиції», що означає запис до пам’яті комп’ютера координат роботи.

Від цієї діяльності до доріжки «Інтерфейс» веде перехід до наступної діяльності «Відображення запиту параметрів», що говорить сам за себе. Від цієї діяльності до доріжки «Користувач» йде двобічний перехід, на другому боці якого розташовуються діяльність «Заповнення параметрів». Специфіка цього двобічного переходу відзначається багаторазовим відображенням запиту параметрів та також багаторазовим їх заповненням (в залежності від необхідної кількості параметрів), тобто відбувається взаємодія сусідніх доріжок «Користувач» та «Інтерфейс».

Далі, оскільки було залучено подвійний перехід – з діяльності «Відображення запиту параметрів» відбувається перехід до діяльності «Запам’ятовування», що розташоване на доріжці «Область обчислень та пам'яті». З цієї діяльності веде перехід до блоку-вирішення (decision) «Достатнє заповнення?», що виражена через запитання. Ця умова має два виходи. Перший вихід, що має обмежуючу умову [«Ні»], веде до діяльності «Заповнення параметрів», що розташована на доріжці

«Користувач». У цьому випадку – відбувається подальше заповнення необхідних параметрів та проходження діяльностей: «Відображення запиту параметрів» та «Запам’ятовування», після цього перевірка виконання умови достатності заповнення та, у разі позитивної відповіді – перехід за обмежуючою умовою [«Так»] до діяльності «Відображення роботи» на доріжці «Інтерфейс».

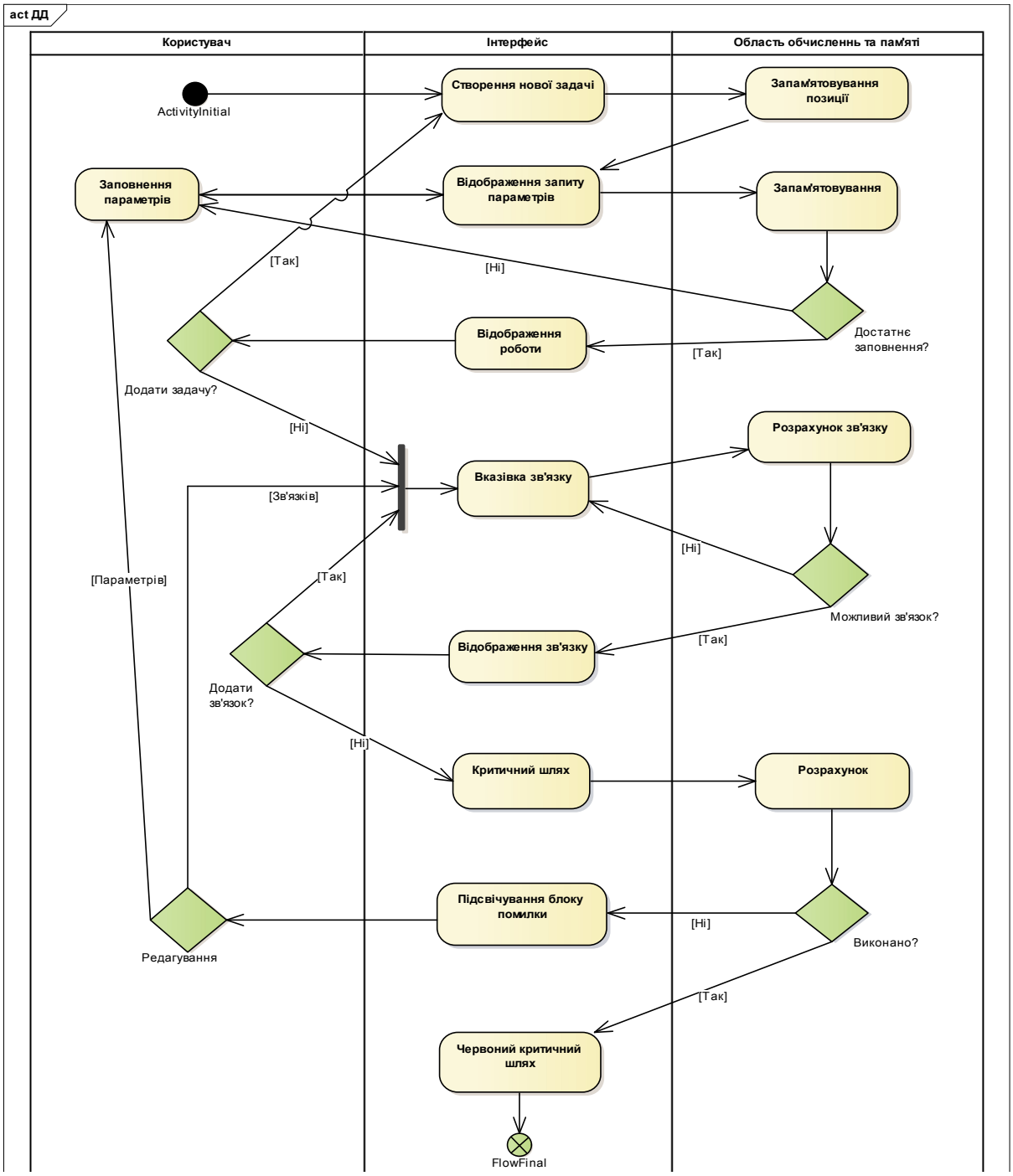


Рисунок 6.6 – Динамічна модель діяльності

З цієї діяльності перехід веде до наступного блоку вирішення «Додати задачу» (рис. 6.6) та у разі позитивного виконання обмежуючої умови – виконуються перехід за посиланням [«Так»] до блоку «Створення нової задачі» на доріжці «Інтерфейс». Далі після виконання цієї задачі створюється перехід до проходження вже описаного циклу, доки не буде необхідності у додаванні нової задачі. Коли немає потреби у додаванні нової задачі – відбувається перехід за обмежуючою умовою [«Ні»], який веде до вертикального синхронізатора, що з'єднує («join synchronization»), розміщеного на доріжці «Інтерфейс», що має три входи та один вихід.

Виходом з цього синхронізатору є перехід на діяльність «Вказівка зв’язку», після якої перехід спрямовується на доріжку «Область обчислень та пам'яті» у створену діяльність «Розрахунок зв’язку», з якої перехід йде на блок вирішення «Можливий зв'язок?», що розташовано на тій саме доріжці. У разі негативної відповіді – відбувається перехід за обмежуючою умовою [«Ні»] до діяльності «Вказівка зв’язку» доріжки «Інтерфейс», а далі – за описаним циклом «Розрахунок зв’язку» – блок «Можливий зв'язок?». Після того, як умова «Можливий зв'язок?» виконана – відбувається перехід за обмежуючою умовою [«Так»] до діяльності «Відображення зв’язку» доріжки «Інтерфейс», а далі – за допомогою переходу до блоку умови «Додати зв'язок?».

У позитивному випадку – відбувається перехід за обмежуючою умовою [«Так»] до третього входу вертикального синхронізатора, що з'єднує, а далі – за вже описаним циклом (див. рис. 6.6). У випадку негативної відповіді щодо умови додавання зв’язку – перехід відбувається за обмежуючою умовою [«Ні»] до діяльності «Критичний шлях» доріжки «Інтерфейс», і далі, за допомогою переходу – до діяльності «Розрахунок» доріжки «Область обчислень та пам'яті». Саме ця діяльність виконує розрахунок критичного шляху МГ. На цій же доріжці виконується перехід до блоку вирішення «Виконано?».

У разі негативної відповіді – відбувається перехід за обмежуючою умовою [«Ні»] до діяльності «Підсвічування блоку помилки» доріжки «Інтерфейс» та подальший перехід у блок вирішення «Редагування» з двома виходами, поданими у вигляді обмежуючих умов редагування: [«параметрів»] та [«зв’язків»].

У випадку «Редагування параметрів» – відбувається перехід до діяльності «Заповнення параметрів», що розташована на цій же доріжці та подальшого виконання потрібних діяльностей, що було розглянуто вище.

У випадку редагування [«зв’язків»] – відбувається перехід до другого входу вертикального синхронізатора («join synchronization»), що розташовано на доріжці «Інтерфейс», для подальшого проходження розглянутих діяльностей.

Усі описані цикли виконуються доти, доки після вирішення «Виконано?» не буде отримано позитивну відповідь та виконано перехід за обмежуючою умовою [«Так»] до діяльності «Червоний критичний шлях» доріжки «Інтерфейс». Ця діяльність означає підсвічування кольором зв’язків, які є складовими критичного шляху МГ. Далі на цій же доріжці «Інтерфейс» відбувається перехід до кінцевої діяльності розглянутого інформаційного потоку «Flow Final», на якій закінчується проєктування ДМД цієї гілки.

Таким чином, якщо виконати аналіз кількості діяльностей та блоків-вирішень на спроєктованій ДМД – отримаємо наступне розподілення за доріжками:

– «Користувач» – 2 діяльності та 3 вирішення;
– «Інтерфейс» – 9 діяльностей та 1 синхронізація;
– «Область обчислень та пам'яті» – 4 діяльності та 3 вирішення.

6.3.5 Модель операцій із проєктними класами

МОПК використовуються для проєктування основного формового наповнення майбутнього ПЗ [247]. Класом зветься деяка сутність, що інкапсулює дані [248]. Стосовно до нотації UML, класи містять атрибути (власно інкапсульовані дані різного походження) та операції (дії на цими або іншими даними) [249].

Кожні конкретні атрибути та операції класів, що спроєктовані у поданому розділі монографії буде розглянуто нижче. Крім класів на МОПК також важливими для аналізу є зв’язки або відношення (це більш точне визначення стосовно до МОПК). Взагалі, відношень, щодо методології МОПК – існує багате розмаїття, проте зупинимося лише на тих, що присутні у спроєктованій МОПК проєктування МГ (рис. 6.7).

---

247. Шмуллер Дж. Освой самостоятельно UML 2.0 за 24 часа. Практическое руководство. Москва: Вильямс, 2005. 416 с.

248. Ларман К. Применение UML 2.0 и шаблонов проектирования. 3-е изд. Москва: Вильямс, 2006. 736 с.

249. Буч Г., Рамбо Дж., Джекобсон А.. Язык UML. Руководство пользователя. 2-е изд. Москва: ДМК Пресс, 2004. 432 с.

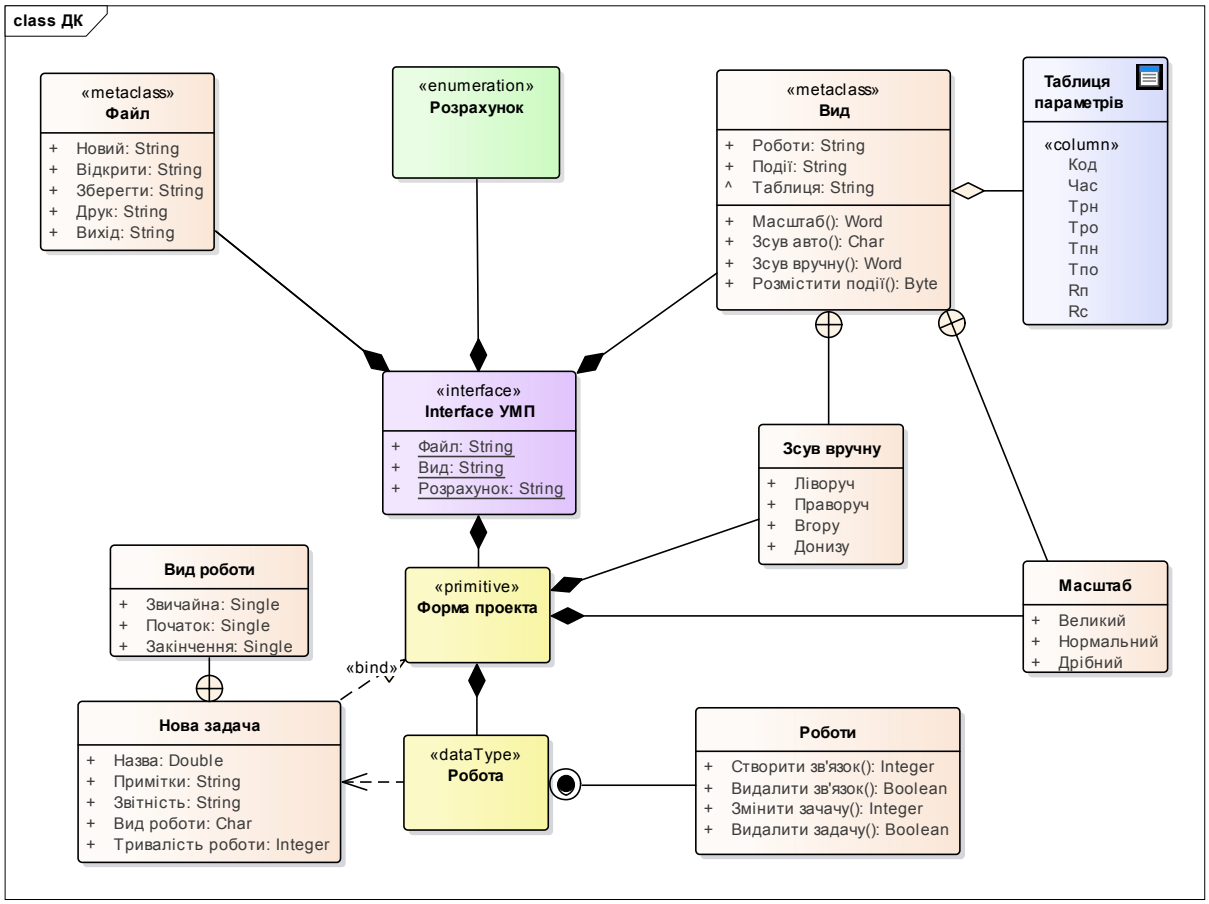


Рисунок 6.7 – Модель операцій із проєктними класами

На діаграмі присутні наступні відношення:

– агрегація (aggregation) – це одна з форм асоціації, яка відрізняється деякою незалежністю складових частин так, що у випадку знищення цілого – часткове залишається, на МОПК позначується у вигляді незафарбованого ромбу з боку цілого;

– композиція (composition) – також одна із форм асоціації, яка відрізняється від агрегації присутністю залежного зв’язку між цілим та частковим, при знищенні цілого – часткове також знищується, на МОПК позначується стрілкою у вигляді зафарбованого ромбу з боку цілого;

– вкладення (nesting) – особлива форма зв’язку (відношення), яке означає безпосереднє доповнення деяких функцій об’єкту-цілого окремими об’єктами-частками, що виконані, здебільшого, у вигляді вкладених переліків, на МОПК позначується у вигляді знаку «+» у колі з боку цілого;

– включення (contained) – форма зв’язку, що має на увазі особливу форму відношення об’єкту-частки, яка не наведена у загальному представленні об’єкта-цілого, а надається тільки після її окремого виклику, у МОПК позначується у вигляді зафарбованого кола у ще одному колі з боку цілого;

– залежність (dependency) – форма відношення, що показує безпосередню залежність одного класу від іншого (головного), зображується у вигляді пунктирної стрілки зі вказівником у бік головної частини;

– прив’язка (bind) – форма відношення, яка визначає закріплення одного класу за іншим (головним), позначується пунктирною стрілкою, спрямованою у бік головного класу з відповідним написом «bind» вздовж стрілки.

Центральним класом МОПК – є клас-інтерфейс («interface») (рис. 6.7), що являє собою графічний інтерфейс користувача ПЗ проєктування МГ. Цей клас першим постає перед користувачем у вигляді завантажувального примітивного класу (primitive) «Форма проєкту», що має з класом «Інтерфейс» композиційну залежність та визначає первинні геометричні розміри вікна інтерфейсу. Також перед користувачем з’явиться панель меню, елементи якої – є атрибутами класу «Інтерфейс».

Отже, атрибутами класу «Інтерфейс» є:

– «Файл», що являє собою окремий метаклас (metaclass) та надає доступ для роботи з різними файлами проєктів;

– «Вид», що також являє окремий метаклас (metaclass) та надає різноманітне представлення сутностей проєкту;

– «Розрахунок» – являє клас підрахунків (enumeration) та виконує розрахунки параметрів МГ.

Всі атрибути класу-інтерфейсу мають рядковий тип представлення даних («String»).

Метаклас (metaclass) «Файл», пов'язаний композиційною залежністю з класом «Інтерфейс» містить у собі наступні атрибути:

– «Вихід» – надає вихід з ПЗ;

– «Відкрити» – здійснює відкривання збереженого проєкту;

– «Друк» – відповідає за виклик настроювань друку;

– «Зберегти» – виконує запит на шлях зберігання проєкту;

– «Новий» – створює шаблон нового проєкту.

Всі атрибути метакласу «Файл» мають рядковий тип представлення даних («String»).

Клас підрахунків (enumeration) «Розрахунок», пов'язаний композиційною залежністю з класом «Інтерфейс» (рис. 6.7), виконує реалізацію чисельних (табличні параметри представлення) та графічних (критичний шлях) параметрів МГ.

Метаклас (metaclass) «Вид», пов'язаний композиційною залежністю з класом «Інтерфейс» містить у собі наступні атрибути:

– «Роботи» – відображення задач у вигляді робіт;

– «Події» – відображення задач у вигляді подій;

– «Таблиця» – подання табличних даних.

Всі атрибути метакласу «Вид» мають рядковий тип представлення даних («String»). Крім атрибутів, метаклас «Вид» має у своєму складі операції:

– «Масштаб» – виконує масштабування МГ, має вкладений (nesting) клас «Масштаб» та word-type формат представлення даних;

– «Зсув авто» – виконує автоматичний зсув (на основі оптимального розміщення задач) МГ за формою, має символьний формат (Char) представлення даних;

– «Зсув вручну» – виконує зсув МГ за визначеними користувачем розмірами зсуву, має вкладений (nesting) клас «Зсув вручну» та word-type формат представлення даних;

– «Розмістити події» – виконує автоматичне розміщення подій МГ за формою, має байтовий формат (Byte) представлення даних.

До метакласу «Вид» за допомогою агрегаційного відношення приєднано табличний клас (table) «Таблиця параметрів», що містить стовбцеві атрибути (column):

– Код – цифровий код задачі;

– Час – тривалість задачі;

– Трн – ранній строк початку задачі;

– Тро – ранній строк закінчення задачі;

– Тпн – пізній строк початку задачі;

– Тпо – пізній строк закінчення задачі;

– Rп – повний резерв часу;

– Rс – вільний резерв часу.

Вкладені класи (nesting) «Зсув вручну» та «Масштаб» – приєднані за допомогою відношення-вкладення до метакласу «Вид» та композиції – до

примітивного класу (primitive) «Форма проєкту», мають атрибути, що не потребують детальних пояснень, а говорять самі за себе. Так вкладений клас «Зсув вручну» має атрибути: «Ліворуч»; «Праворуч»; «Вгору»; «Донизу». А вкладений клас «Масштаб»: «Великий»; «Нормальний»; «Дрібний».

До класу «Форма проєкту» ще приєднано такі сутності (рис. 6): тип даних (DataType) «Робота» (відношення-композиція) та клас «Нова задача» (відношення-прив'язка). Також ці дві сутності поєднані між собою за допомогою залежності, спрямованої у бік класу «Нова задача».

Клас «Нова задача», що прив'язано (відношення «Bind») до «Форми проєкту», призначено для створення нового та наступного редагування існуючого типу даних (DataType) «Робота», що являє собою графічне зображення сутності «Робота». Клас «Нова задача» містить у собі наступні атрибути:

– «Назва» – формує назву поточної роботи, має подвійний цифровий формат (double) представлення даних, що містить цифру попередньої та поточної роботи;

– «Примітки» – необхідний для створення додаткових пояснень, має рядковий тип («String») представлення даних;

– «Звітність» – формує визначену звітність по роботі, має рядковий тип («String») представлення даних;

– «Вид роботи» – відповідає за наявність або відсутність вхідних та вихідних зв'язків, має вкладений (nesting) клас «Вид роботи» та символьний формат (Char) представлення даних;

– «Тривалість роботи» – створює число, що означає тривалість роботи (в умовних часових одиницях), має цілочисельний (integer) формат представлення даних.

Вкладений (nesting) клас «Вид роботи» – вкладено до класу «Нова задача». Він формує види робіт, що відрізняються наявністю або відсутністю вхідних та вихідних зв'язків, містить у собі наступні атрибути виду:

– «Звичайна» – атрибут за замовченням, який формує вид роботи з можливістю додавання вхідних та вихідних зв'язків;

– «Початок» – атрибут, який формує вид роботи без можливості додавання вхідних зв'язків, але з можливістю – вихідних.

– «Закінчення» – атрибут, який формує вид роботи без можливості додавання вихідних зв'язків, але з можливістю – вхідних.

Усі атрибути вкладеного (nesting) класу «Вид роботи» мають одинично-розрядовий (Single) формат представлення даних.

Крім того, до типу даних (DataType) «Робота» включено (відношення «contained») операційний клас «Роботи», що викликається опційно (наприклад правим щигликом миші). Цей клас містить операції, що виконують визначені їхньою назвою дії над сутностями МГ та не потребують детального пояснення, проте їх перелік наступний:

– «Створити зв'язок» – має цілочисельний (Integer) формат операційного подання даних, що містить номер роботи від якої прокладається зв'язок;

– «Видалити зв'язок» – має формат операційного подання у вигляді логічного виразу (Boolean), ця операція видаляє або залишає виділений зв’язок;

– «Змінити задачу» – має цілочисельний (Integer) формат операційного подання даних, що вказує номер роботи над якою необхідно виконати редакційні виправлення;

– «Видалити задачу» – має формат операційного подання у вигляді логічного виразу (Boolean), ця операція видаляє або залишає виділену задачу.

Таким чином, спроєктована ДК має 12 класів-сутностей та 14 відношень, з яких: 5 звичайних класів; 2 метакласи; 1 табличний клас; 1 клас-інтерфейс; 1 клас-примітив; 1 тип даних; 1 клас підрахунків; 7 композицій; 1 агрегація; 3 вкладення; 1 включення; 1 залежність; 1 прив’язка.

6.3.6 Модель компонентних рішень

МКР призначена для вивчення складу компонентів майбутнього ПЗ та вказівки послідовності компіляції та збірки окремих модулів. Компонентом зветься фізичний модуль коду та фактично містить його у собі. Компонент є найменшою, неподільною складовою частиною ПЗ. Єдиний тип зв’язку, що присутній на МКР – це залежності, що спрямовані від компонента, що породжується у бік головного компонента.

Головною вимогою до МКР – стандартно висунуто – відсутність циклів, тобто послідовність компонентів повинна бути чіткою та прозорою. Користувач працює із компонентами, які можуть бути йому досяжні – у зворотному порядку.

Спроєктована МКР управління плануванням МГ приведена на рис. 6.8.

Розглянемо детальніше її склад. Першим компонентом, із яким співпрацює користувач – є засіб ідентифікації (artifact) або ярлик виклику «Ярлик для spu . lnk», що прикріплено до виконавчого файлу (executable) «spu2-2 . exe», що запускає новий проєкт, через відкритий компонент інтерфейсу (exposed interface) «Видимість інтерфейсу».

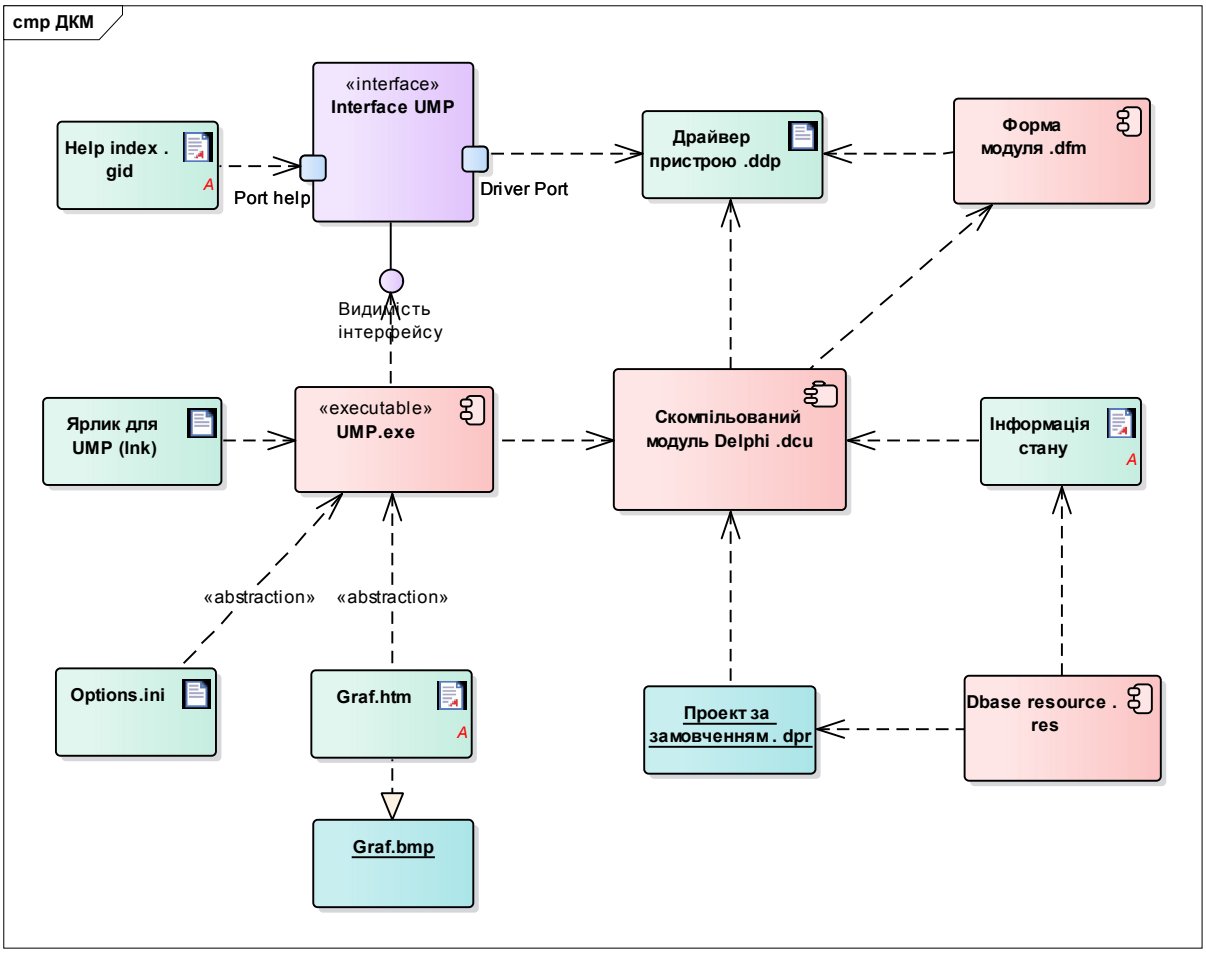


Рисунок 6.8 – Модель компонентних рішень

Цей компонент служить входом до самого інтерфейсу (interface) «Interface spu», що містить порти допомоги (Port help) та драйверу (Port Driver). До порту допомоги (Port help) підключено контекстну допомогу (document artifact) «Help index . gid», яка може бути опційно викликана користувачем при виниканні складної ситуації із проєктом. Через порт драйвера (Port Driver) – підключено об’єкт (artifact), що містить спеціальні дані – драйвер пристрою (Device Driver Profile) «Драйвер пристрою . ddp».

Від цього драйверу й залежить вигляд та підтримка розрішувальної здатності усього інтерфейсу ПЗ.

Також від «Драйвер пристрою . ddp» залежать компонент (component) форми Delphi-модуля «Форма модуля . dfm» та пакетний компонент (packaging component) скомпільованого модуля Delphi «Скомпільований модуль . dcu», що також залежить від «Форма модуля . dfm». Таким чином, пакетний компонент «Скомпільований модуль . dcu» має подвійну впорядкованість (рис. 6.8).

Пакетний компонент «Скомпільований модуль . dcu» виступає головним для цілої низки компонентів, а саме для:

– виконавчого файлу (executable) «spu2-2 . exe», який описано вище;

– службового протоколу (document artifact) «Інформація стану», що містить технічну інформацію щодо поточного та останнього стану проєкту;

– об’єкту (object) «Проєкт за замовченням . dpr», який завантажується у вигляді шаблону при створенні нового проєкту та який (при необхідності) може бути змінено та доповнено.

Компонент (component) накопичувальної бази даних «Dbase resource . res» має подвійну впорядкованість та компілюється кожного разу при додаванні або зміні інформації у службовому протоколі (document artifact) «Інформація стану» та об’єкту (object) «Проєкт за замовченням . dpr».

Виконавчий файл (executable) «spu2-2 . exe» є головним для:

– засобу ідентифікації (artifact) або ярлика виклику «Ярлик для spu . lnk», що описано вище;

– службового протоколу (document artifact) опцій та параметрів ПЗ «Options . ini», що містить технічну інформацію щодо останнього геометричного розміру форми проєкту, встановленої користувачем;

– браузерної сторінки (Web-document artifact) «Graf . htm», яка відображує остаточну сформовану топологію розміщення задач на МГ; причому браузерна сторінка (Web-document artifact) «Graf . htm» формує (реалізує) само графічне зображення (object) «Graf . bmp» у вигляді самостійного (автономного) bmp-файлу, що може бути як збережено (надруковано, відправлено тощо), так і відредаговано.

Причому, службовий протокол (document artifact) «Options . ini» та браузерна сторінка (Web-document artifact) «Graf . htm» впорядковані абстрактними (abstraction) залежностями (dependency), що означають створення них самих, як сутностей, тільки за потреби користувача та із

неодмінною його участю (внесення змін розміру, команди друку чи Web-конвертації.

Таким чином, спроєктована МКР ПЗ проєктування МГ містить 16 компонентів, об'єднаних 15 зв'язками.

6.4 Складання інструкції користувача

ПМК, що створюється у поданому розділі монографії, призначено для створення, розрахунків та оптимізації мережевих графіків УП реінжинірингу ПС. Для його створення став у нагоді матеріал з [250] – [253].

Методика роботи з мережевими графіками заснована на редагуванні параметрів завдань (робіт). Будь-яка зміна параметрів проводиться правою кнопкою миші, після чого визначається: на якому елементі був зроблений щиглик і пропонується відповідне меню. Ліва кнопка миші призначена для переміщення зображень за формою. Події (віхи) змінюються автоматично при зміні завдань, також присутня можливість змінювати назву подій (права кнопка миші на події).

6.4.1 Основні позначення, що прийняті у ПЗ

Для роботи із ПЗ, користувачеві необхідно володіти спеціальною термінологією, основні поняття якої зведено до словника, що подано вище.

Критичні задачі на графіку виділяються червоними кольорами. Особливість критичних задач полягає в тому, що кожна з них повинна починатися точно у момент часу, коли закінчилася попередня, крім того, тривати вона повинна не більше того часу, що відведено їй за планом. У противному випадку, критичний шлях збільшиться. Отже, критичний шлях повинен бути завжди під контролем керівників робіт тому, що від

---

250. Пашеку Х. Программирование в Borland Delphi 2006 для профессионалов. Москва : Вильямс, 2006. 944 с.

251. Культин Н. Основы программирования в Delphi XE. Санкт-Петербург: БХВ-Петербург, 2011. 416 с.

252. Осипов Д. Базы данных и Delphi. Теория и практика. Санкт-Петербург: БХВ-Петербург, 2011. 752 с.

253. Вальвачев А. Н., Сурков К. А., Сурков Д. А. Программирование на языке Delphi: учеб. пос. Киев: НТУ, 2005. 436 с.

виконання критичних задач цілком залежить виконання усього плану. Критичні задачі мають нульові резерви часу.

До часових параметрів подій відносяться:

T_p (i) – ранній строк настання події і (час, необхідний для виконання всіх задач, що передують даній події);

T_n (i) – пізній строк настання події i (час настання події і, перевищення якого викличе аналогічну затримку настання завершальної події мережі);

R (i) – резерв часу настання події i (це такий проміжок часів, на який може бути відстрочене настання цієї події без порушення строків завершення розробки в цілому).

У ПЗ подія позначується колом, задачі – у вигляді стрілок, цифри над якими показують тривалість цих задач (рис. 3.1). Значення часових параметрів на графіку подій записуються у спосіб, що наведено на рис. 6.9.

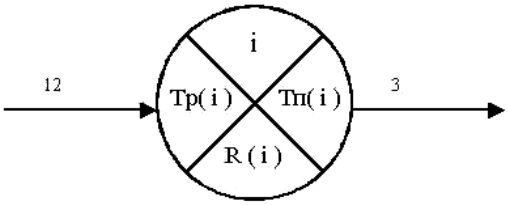


Рисунок 6.9 – Ескіз зображення подій із позначеннями у розробленому ПЗ

Задачі на мережевих графіках у спроєктованому ПЗ мають ідентичне позначення із роботами (рис. 6.10). Параметри робот записуються всередині квадрату, а стрілки – позначують зв’язки між роботами чи задачами (рис. 6.10). Усі параметри можна побачити та проаналізувати їх після генерації спеціальної таблиці (пп. 6.4.3.1). Також, у відповідному пункті монографії, наведено принципи розрахунків цих параметрів.

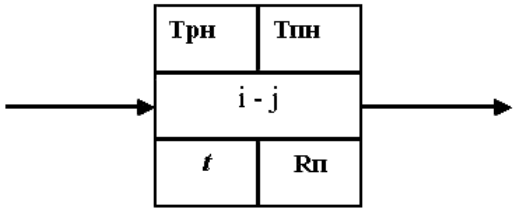


Рисунок 6.10 – Ескіз зображення робіт (задач) із позначеннями у розробленому ПЗ

6.4.2 Порядок роботи з ПМК

6.4.2.1 Створення нової задачі

Для створення нової задачі необхідно виконати щиглик правою кнопкою миші по формі (рис. 6.11), після чого відкриється вікно редагування параметрів задачі, що зображено на (рис. 6.12).

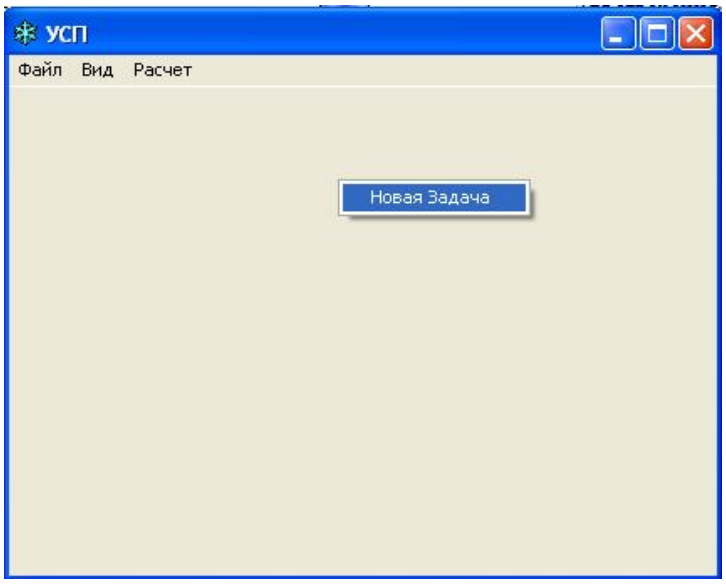


Рисунок 6.11 – Вікно створення нової задачі

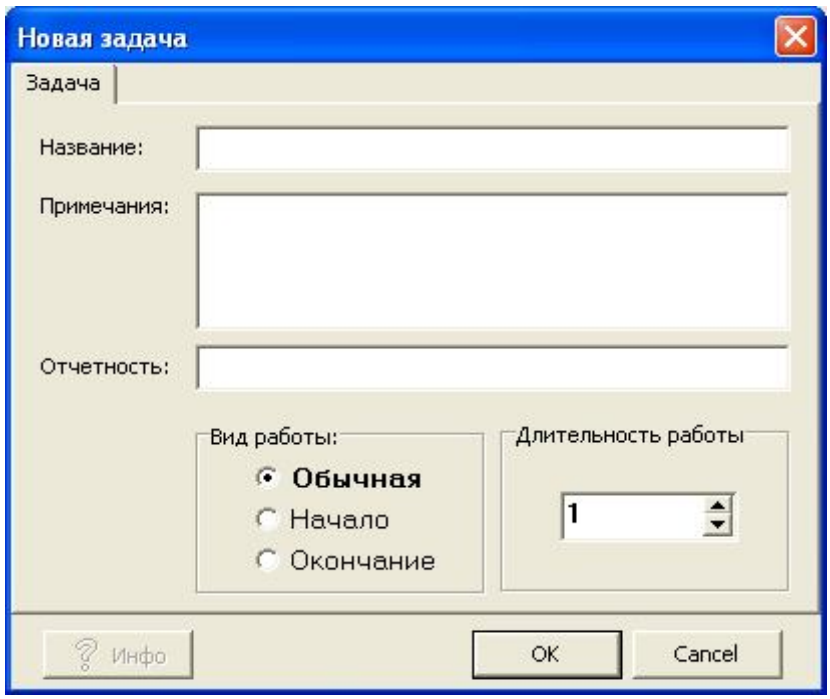


Рисунок 6.12 – Вікно редагування параметрів задачі

Усі задачі, що створюються, поділяються на три види: початкові, кінцеві та звичайні, причому кількість початкових та кінцевих задач може бути більше одиниці. У початкових задачах немає попередників, у кінцевих немає послідовників.

Необхідно звернути увагу на те, що призначення виду роботи «Початок» або «Закінчення» – викликає автоматичне видалення всіх зв'язків задачі, що редагується, таким чином: їх необхідно призначати заново.

Поле «Назва» використовується для швидкого пошуку потрібної задачі на МГ (при наведенні курсору на задачу – висвітлюється підказка).

«Примітки» – більш докладні пояснення (не є необхідними для заповнення), а «Звітність» – назва події, якою завершується подана задача (ця фраза буде використана при автоматичному формуванні назви подій).

Досвід побудови мережевих графіків показав, що графік зручніше будувати у два етапи. Спочатку створити всі задачі, заповнивши графи «Назва» та «Тривалість». Як назву зручно використовувати позначення задачі «1-2» або «1, 2» за бажанням).

При розміщенні зв'язків слід використовувати спливаючу підказку, що з'являється при наведенні курсору на задачі. Коли графік вже побудований – необхідно виконати «Розрахунок». При необхідності ліквідувати помилки, цикли тощо. Тільки коли критичний шлях буде розрахований – можна приступати до заповнення задач додатковою інформацією: змінювати «Назву», додавати «Примітки» та «Звітність»).

6.4.2.2 Зміна параметрів задачі

Виконавши щиглик правою кнопкою мишки по задачі, одержимо можливість виконати одну із чотирьох операцій (рис. 6.13):

– створити зв'язок (указати попередню задачу);

– видалити зв'язок (скасувати попередню задачу);

– змінити задачу (відкрити вікно редагування параметрів задачі);

– видалити задачу (разом із задачею віддаляються усі зв'язки, у яких вона бере участь).

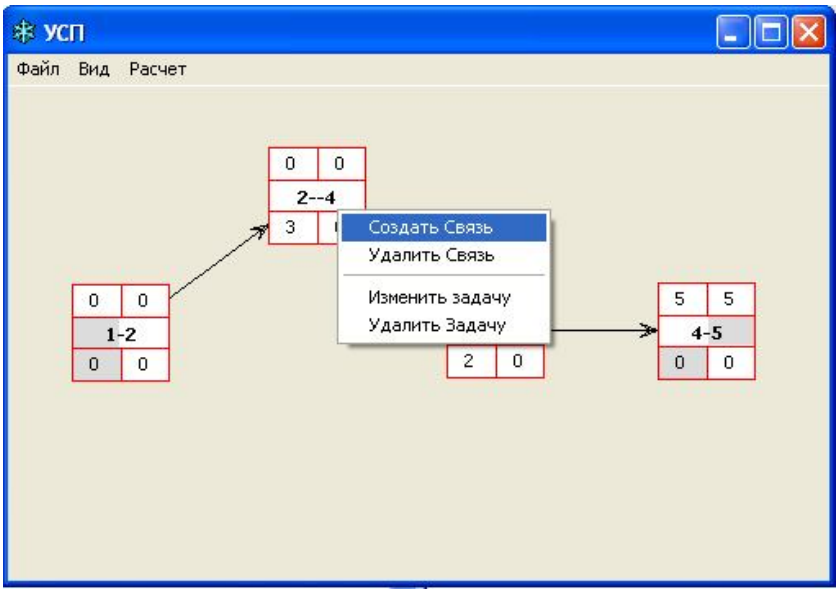


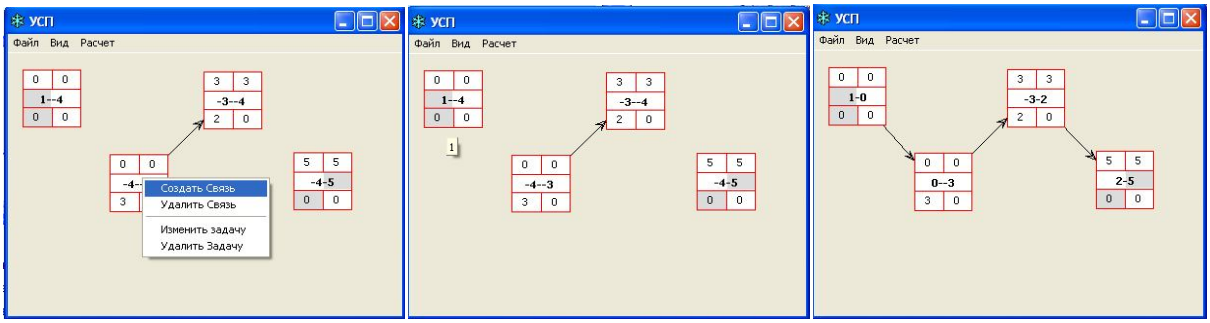
Рисунок 6.13 – Зміна параметрів задачі

6.4.2.3 Створення зв’язків між задачами

У ПЗ використовується тільки один вид зв'язку між задачами – «Попередник». Порядок зв'язування задач наступний: спочатку натискаємо правою кнопкою задачу, вибираємо «Створити Зв'язок», потім курсором натискаємо задачі, що є попередником. Між двома задачами встановлюється стрілка, що спрямована від попередника до послідовника, яка означає: наступна задача не може початися поки не закінчиться попередня. Цей порядок у наочній графічній формі наведено на рис. 6.14.

Правила побудови мережевих графіків вимагають, щоб більш ранні події перебували на графіку ліворуч, а більш пізні – праворуч. Дотримання такого правила, дозволяє краще проаналізувати виникнення циклів на МГ. Якщо у конкретному випадку замість стрілки відображується пряма лінія – це значить, що висунута умова не виконується. Тоді, необхідно пересунути, за допомогою лівої кнопки миші, задачу або подію у потрібний бік.

У випадку, якщо ПЗ відмовляється з'єднувати дві задачі – це означає, що користувач намагається порушити встановлені правила складання мережевих графіків. Тоді слід уважно перевірити правильність операції, що проєктується. Якщо вірність усіх операцій не викликає сумнівів – необхідно створити додаткові «Фіктивні» задачі із нульовою тривалістю та вставити їх між задачами у тому місці, де планувався зв'язок.



а) б) в)

Рисунок 6.14 – Порядок створення зв’язків між задачами:
а) обрання дії; б) обрання задачі-попередника; в) відображення зв’язків

У деяких випадках, допомагає видалення декількох зв'язків (або задач) з наступним їхнім відновленням, проведеному в іншому порядку. Також, у деяких визначених випадках, у ПЗ закладена можливість запропонувати створити «Фіктивну» задачу автоматично. При цьому можна погодитися із цією процедурою, натиснувши «ОК» або відмовитися та створити додаткову («Фіктивну») задачу самостійно. У особо складних випадках, коли все із вище переліченого не допомогло – слід закрити ПЗ та почати все заново (із завантаження останнього варіанта збереженої моделі мережевого графіка).

Існує велике розмаїття зв'язків між задачами. Наприклад: наступна задача може починатися, коли попередня задача виконана на 50%. У цьому випадку попередню слід розбити на дві прості задачі: першу (обсягом 50%) та другу (також 50%), після чого виконуємо їх зв'язок у звичайному порядку. При встановленні одного зв'язку, ПЗ автоматично прораховує всі інші зв'язки так, що після однієї операції по зв'язуванню задач, раптом, може з'явитися ще кілька зв'язків – це одна з особливостей мережевого графіка, побудованого у вигляді «задачі-зв'язки». До речі, саме складність сприйняття користувачем цих численних зв'язків, змушує у звітах використовувати мережеві графіки у вигляді «подія-задача». Однак, процес створення та редагування зручніше виконувати на мережевих графіках, що виконані у вигляді «задачі-зв'язки».

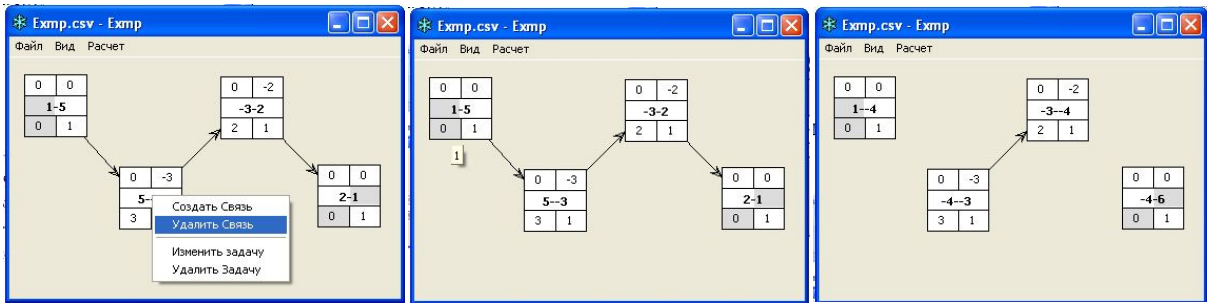
6.4.2.4 Видалення зв’язків між задачами

Видалення зв’язків відбувається у два етапи:

– спочатку виконується натискання правою кнопкою миші на першій задачі та обирання меню «Видалити Зв'язок»;

– далі курсором (у вигляді закресленого кола) натискаємо на попередню задачу (квадрат, з якого виходить стрілка зв'язку), після чого стрілка зникає.

Цей порядок у наочній формі наведено на рис. 6.15.



а) б) в)

Рисунок 6.15 – Порядок знищення зв’язків між задачами:
а) обрання дії; б) обрання задачі-попередника; в) відображення видалення

6.4.2.5 Робота з подіями

Виконати конвертацію мережевого графіка на графік, що побудовано із подій (цей графік не редагується) можна використавши пункт меню «Події» або одночасним натисканням клавіш «Alt + Q» на клавіатурі.

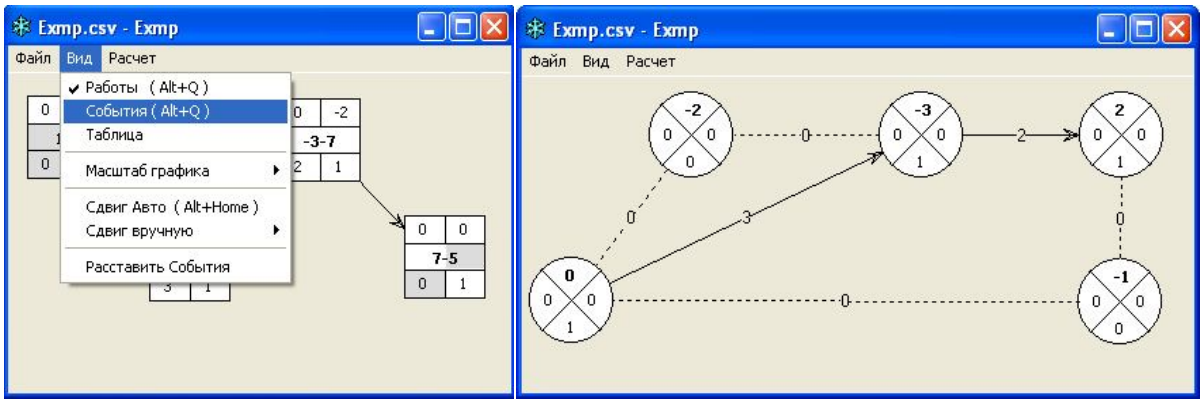
Для автоматичного розташування подій на формі, можна скористатися пунктом меню «Розставити події». Остаточне розміщення подій за формою виконується вручну за допомогою лівої кнопки миші. Відповідно до встановлених правил, фіктивні задачі (задачі з нульовою тривалістю) на графіку позначаються пунктиром. Порядок конвертації робіт у події – наведено на рис. 6.16.

Екранні знімки спроєктованих мережевих графіків виконання реінжинірингу однієї з ПС, узятої як приклад, наведено на рис. 6.17 (для випадку «вершини-роботи») та на рис. 6.18 (для випадку «вершини-події»).

Слід зауважити, що поки не виконано розрахунок критичного шляху (натисканням пункту меню «Розрахунок»), вписані у задачах та подіях цифри не мають ніякого значення.

У свою чергу, розрахунок виконується тільки у тому випадку, якщо всі задачі пов'язані між собою та присутні початкові й кінцеві задачі. У

противному випадку, розрахунок буде зупинено, а задача, на якій припинено розрахунок, буде підсвічена жовтими кольором.



а) б)

Рисунок 6.16 – Порядок конвертації робіт у події:
а) обрання відображення; б) відображення подій

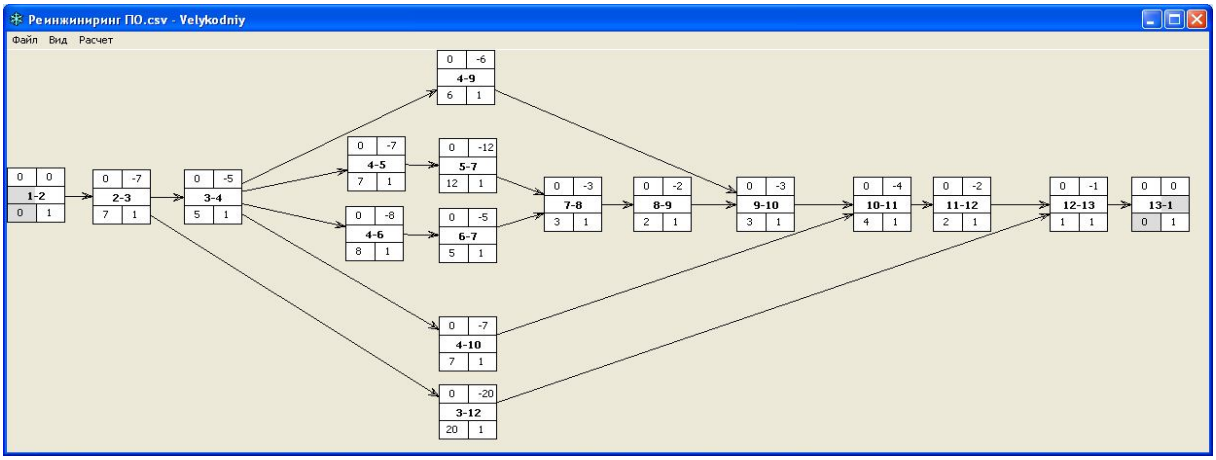


Рисунок 6.17 – Екранний знімок мережевого графіка виконання реінжинірингу програмної системи у вигляді «вершини-роботи»

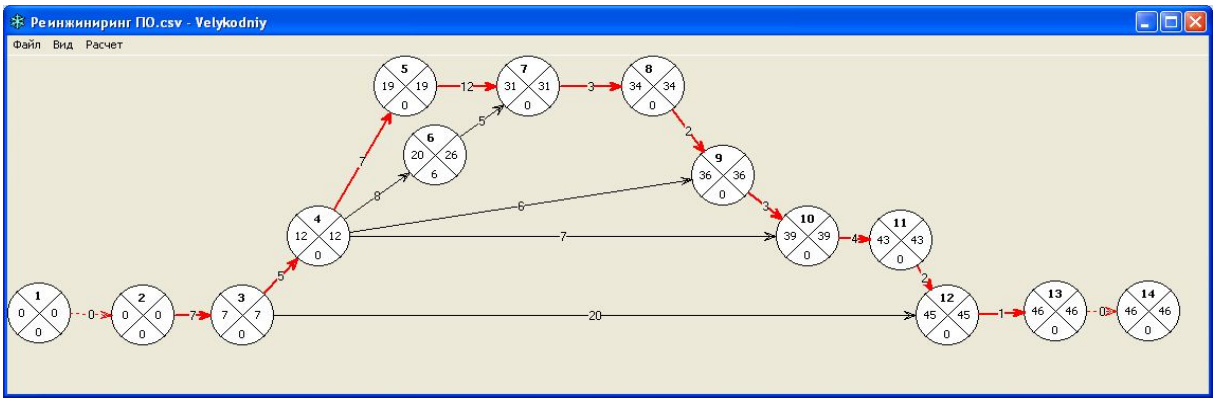


Рисунок 6.18 – Екранний знімок мережевого графіка виконання реінжинірингу програмної системи у вигляді «вершини-події»

Екранний знімок розрахунку критичного шляху мережевого графіка виконання реінжинірингу програмної системи наведено на рис. 6.19.

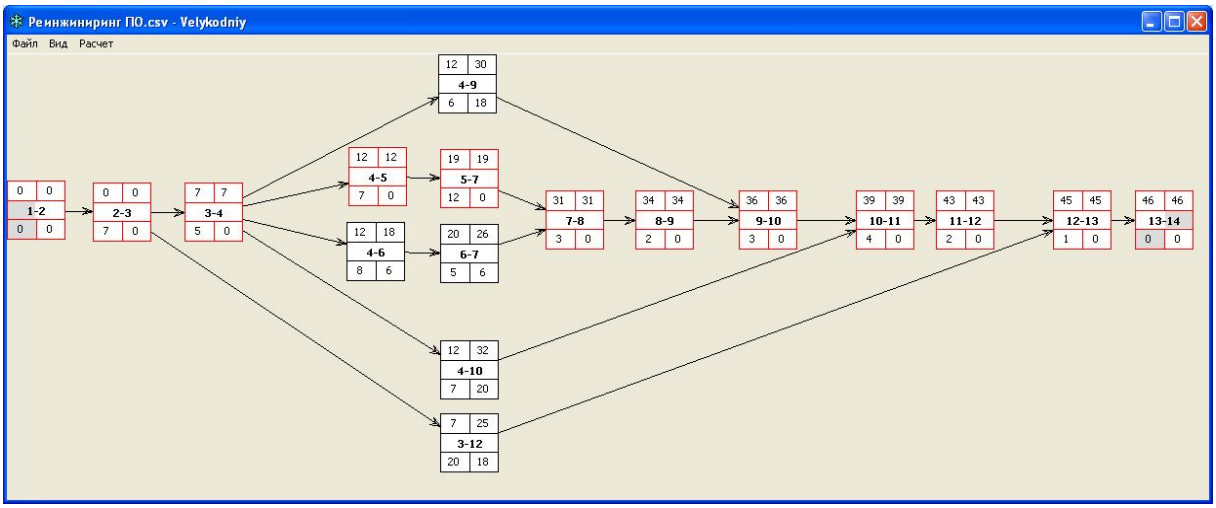


Рисунок 6.19 – Екранний знімок розрахованого критичного шляху мережевого графіка виконання реінжинірингу програмної системи

6.4.3 Додаткові можливості ПЗ

6.4.3.1 Генерування таблиць

Для того, щоб виконати генерацію таблиці, що містить усі проєктні розрахунки, необхідно обрати пункт головного меню «Вид», а далі натиснути «Таблиця» рис. 6.20.

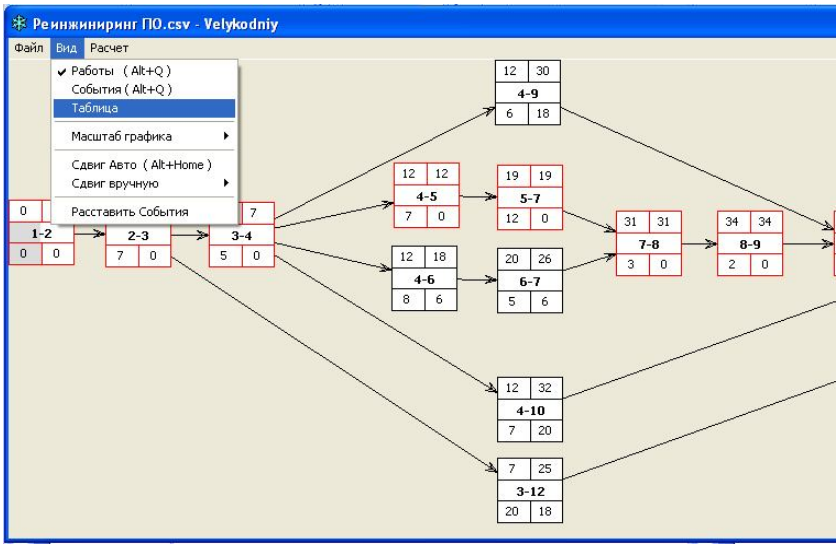
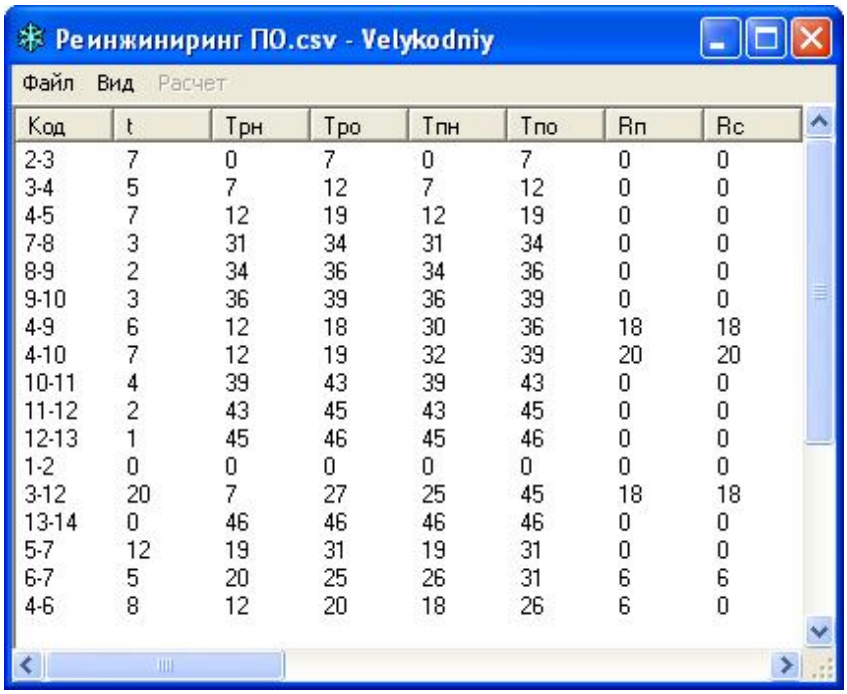


Рисунок 6.20 – Початкові дії для генерації таблиці

Згенеровану таблицю із розрахунками усіх параметрів мережевого графіка виконання реінжинірингу наведено на рис. 6.21.



Код	t	Трн	Тро	Тпн	Тпо	Rп	Rс
2-3	7	0	7	0	7	0	0
3-4	5	7	12	7	12	0	0
4-5	7	12	19	12	19	0	0
7-8	3	31	34	31	34	0	0
8-9	2	34	36	34	36	0	0
9-10	3	36	39	36	39	0	0
4-9	6	12	18	30	36	18	18
4-10	7	12	19	32	39	20	20
10-11	4	39	43	39	43	0	0
11-12	2	43	45	43	45	0	0
12-13	1	45	46	45	46	0	0
1-2	0	0	0	0	0	0	0
3-12	20	7	27	25	45	18	18
13-14	0	46	46	46	46	0	0
5-7	12	19	31	19	31	0	0
6-7	5	20	25	26	31	6	6
4-6	8	12	20	18	26	6	0

Рисунок 6.21 – Згенерована таблиця із розрахунками параметрів мережевого графіка УП реінжинірингу

До основних позначень та їх скорочень (скорочення позначень у ПЗ виконано російською мовою), що зведено до даної таблиці відносять наведені нижче змінні:

t – тривалість задачі (умовні одиниці часу – у нашому випадку – дні).

$T_{\text{рн}}$ – ранній строк початку задачі, що дорівнює:

$$T_{рн}(i,j) = T_p(i).$$

$T_{\text{ро}}$ – ранній строк закінчення задачі:

$$T_{ро}(i,j) = T_p(i) + t(i,j).$$

$T_{\text{пн}}$ – пізній строк початку задачі:

$$T_{пн}(i,j) = T_п(i) - t(i,j).$$

$T_{\text{по}}$ – пізній строк закінчення задачі:

$$T_{no}(i,j) = T_n(i).$$

$R_{\text{п}}$ – повний резерв часу:

$$R_n(i,j) = T_n(j) - Tp(i) - t(i,j).$$

$R_{\text{с}}$ – вільний резерв часу:

$$R_c(i,j) = T_p(j) - Tp(i) - t(i,j).$$

6.4.3.2 Перегляд, друк та збереження проєкту

При виборі пункту меню «Друк» (рис. 6.22), у папці із проєктом створюється файл «graf.htm», що автоматично проглядається за допомогою Internet-браузера, що встановлено на комп'ютері користувача (рис. 6.23).

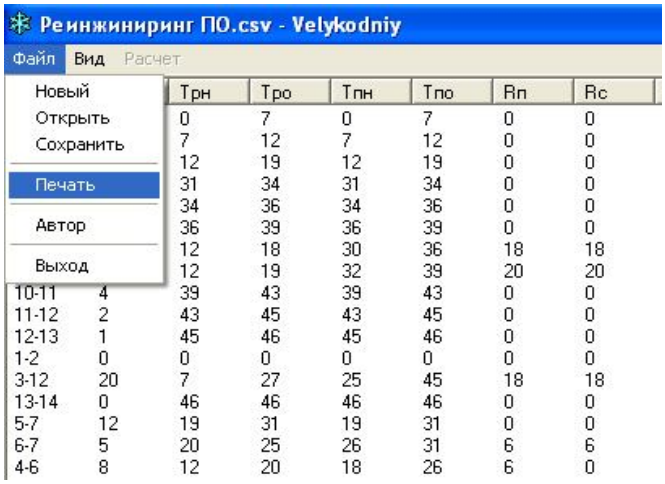


Рисунок 6.22 – Початкові дії для виконання конвертації у html-файл

Виконавши натиснення правою кнопкою миші по вільному від картинок полю браузера, можна викликати контекстне меню, у якому є пункт «Друк». Вибір цього пункту виводить МГ на принтер. Слід зауважити, що друкується та картинка, яку видно у цей момент на екрані. При цьому сіре підсвічування особливих параметрів задач – відключається.

Також можливо (знову ж за допомогою правої кнопки миші) копіювання отриманих картинок і таблиць у буфер обміну із наступною вставкою до інших будь-яких редакторів документів.

Крім того, у тій саме папці, де зберігся файл «graf.htm» – автоматично створюється файл «graf.bmp», переглянути який можна не за допомогою Internet-браузера, а за допомогою будь-якої програми роботи із зображеннями. Цей файл можна переміщати в інші папки, вставляти у звітні документи, змінювати в графічних редакторах.

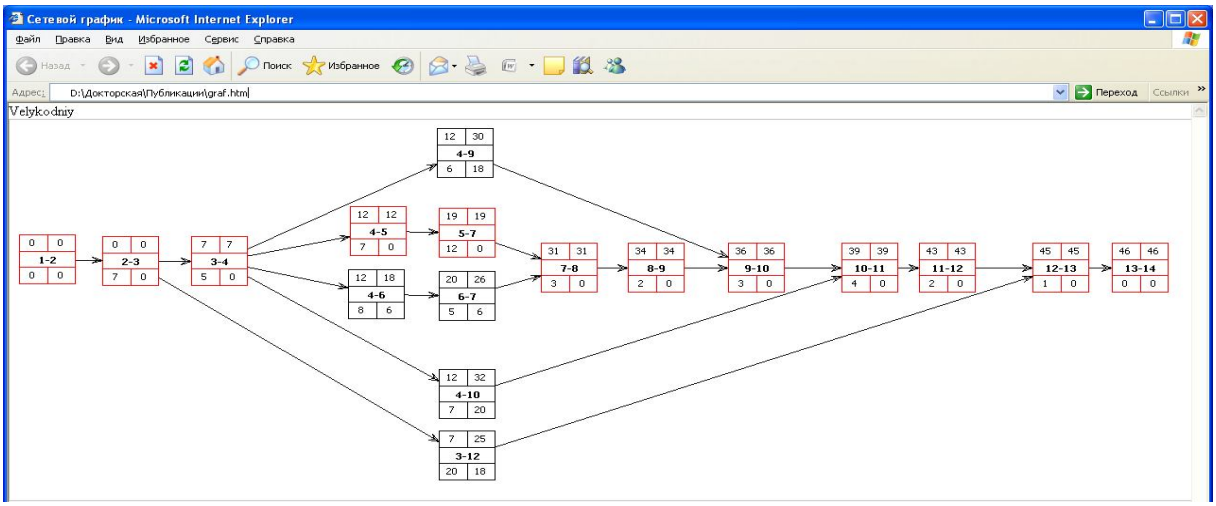


Рисунок 6.23 – Екранна форма топології мережевого графіку, у вигляді html-файлу

Екранний знімок МГ виконання проєкту реінжинірингу поданого ПМК, збереженого у форматі «bmp» та відкритого за допомогою Microsoft Office Picture Manager, наведено на рис. 6.24.

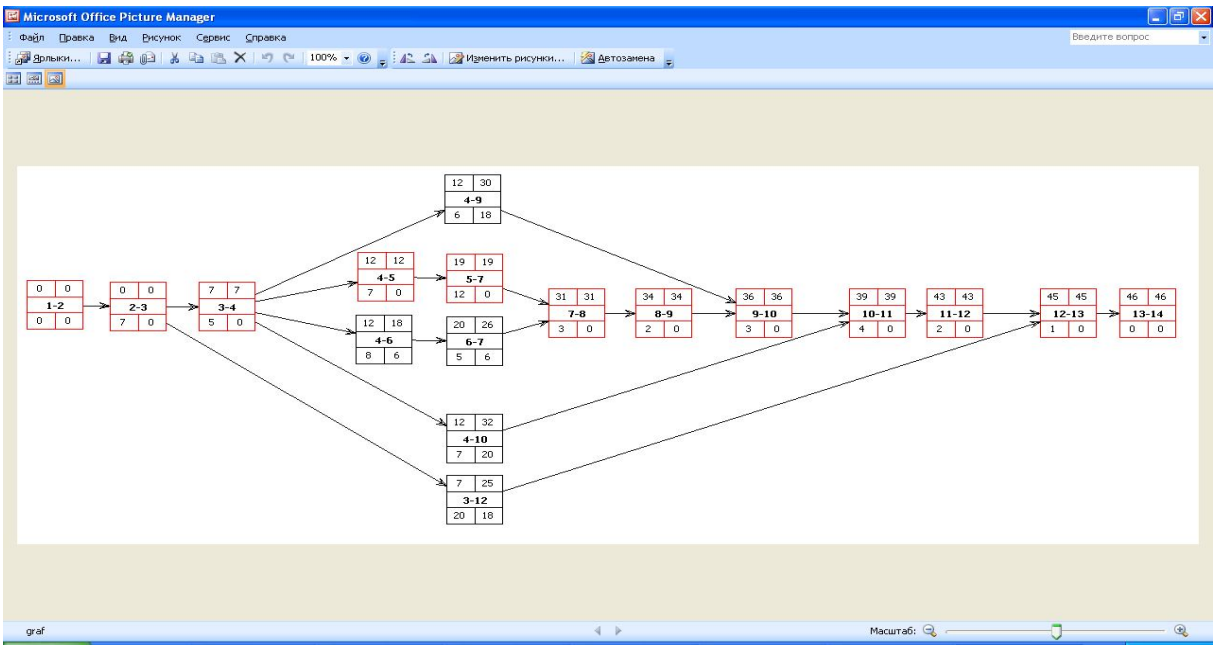


Рисунок 6.24 – Екранний знімок мережевого графіку виконання реінжинірингу роботи, відкритого за допомогою Microsoft Picture Manager

Для збереження спроєктованого МГ, необхідно, обравши відповідний пункт панелі меню, у діалоговому вікні, що з’явилося – увести назву проєкту (рис. 6.25).

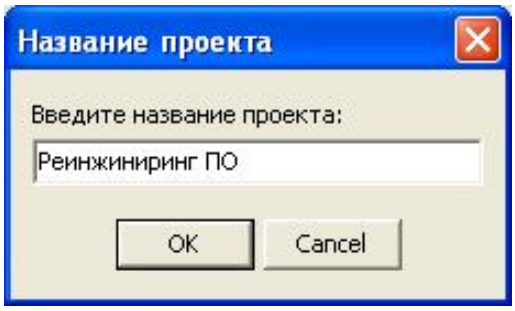


Рисунок 6.25 – Діалогове вікно збереження мережевого проєкту

6.4.3.3 Переміщення та зміни розміру структур

Для прискорення переміщень мережевих графіків за формою, у спроєктованому ПЗ передбачено кілька комбінацій «гарячих» клавіш. Всі вони зазначені у пункті «Зсув рисунку» та приведені на рис. 6.26.

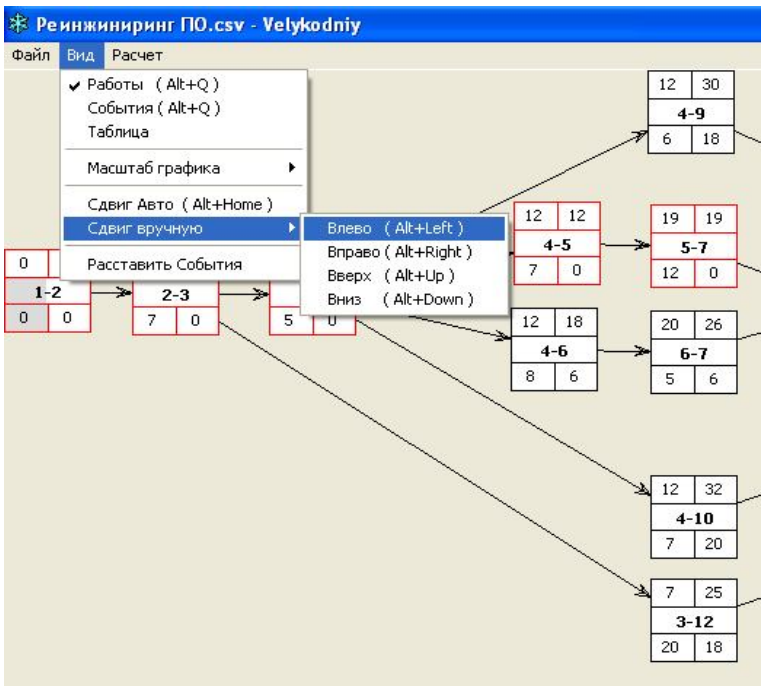


Рисунок 6.26 – Комбінації «гарячих» клавіш для переміщення сформованого мережевого графіку

Змінити розмір мережевого графіка можна двома способами (рис. 6.27):

– вибором відповідного пункту меню «Розмір задач»;
– за допомогою комбінацій клавіш («Alt + 1», «Alt + 2», «Alt + 3»).

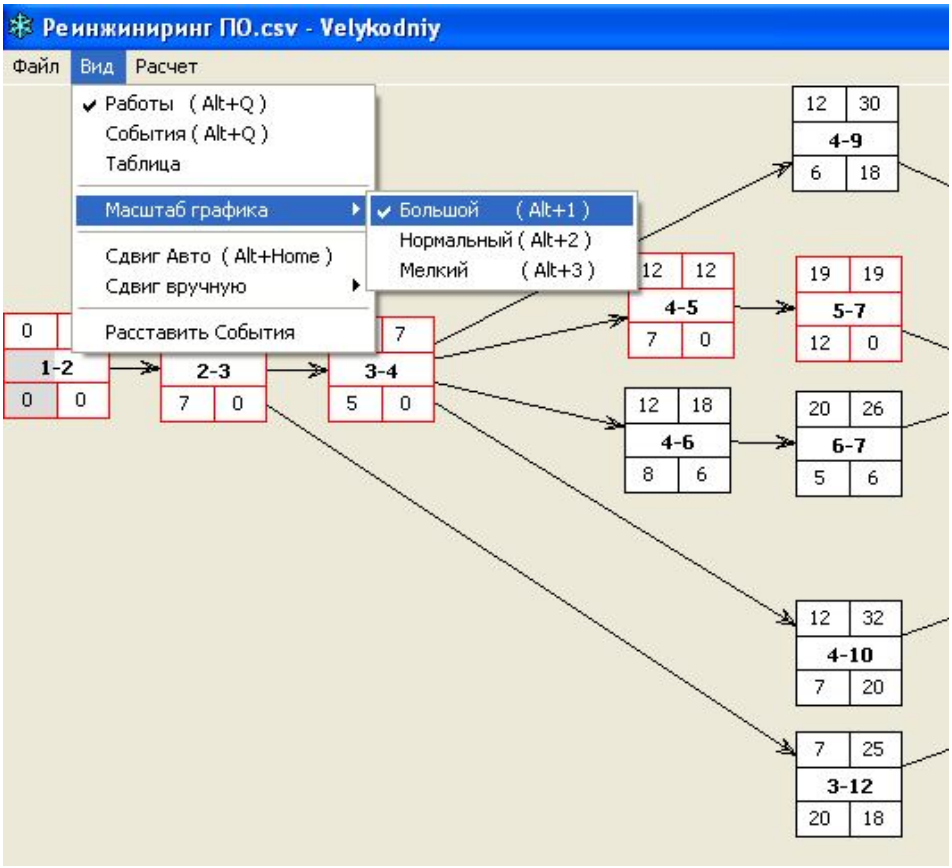
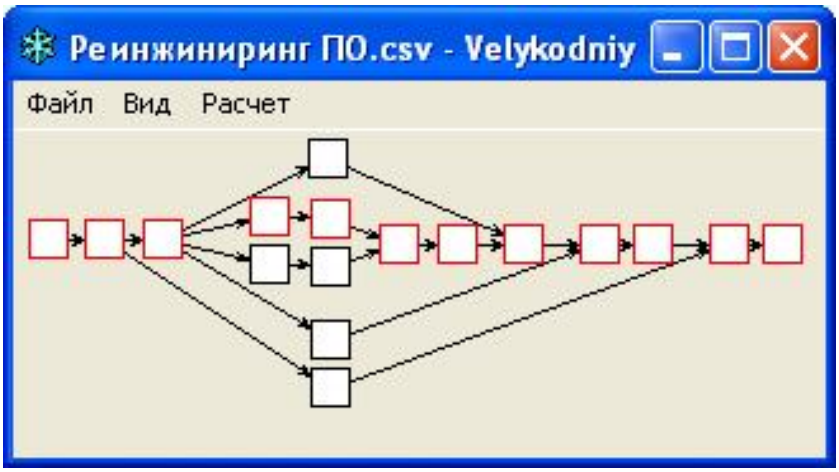


Рисунок 6.27 – Зміна розміру представлення мережевого графіка

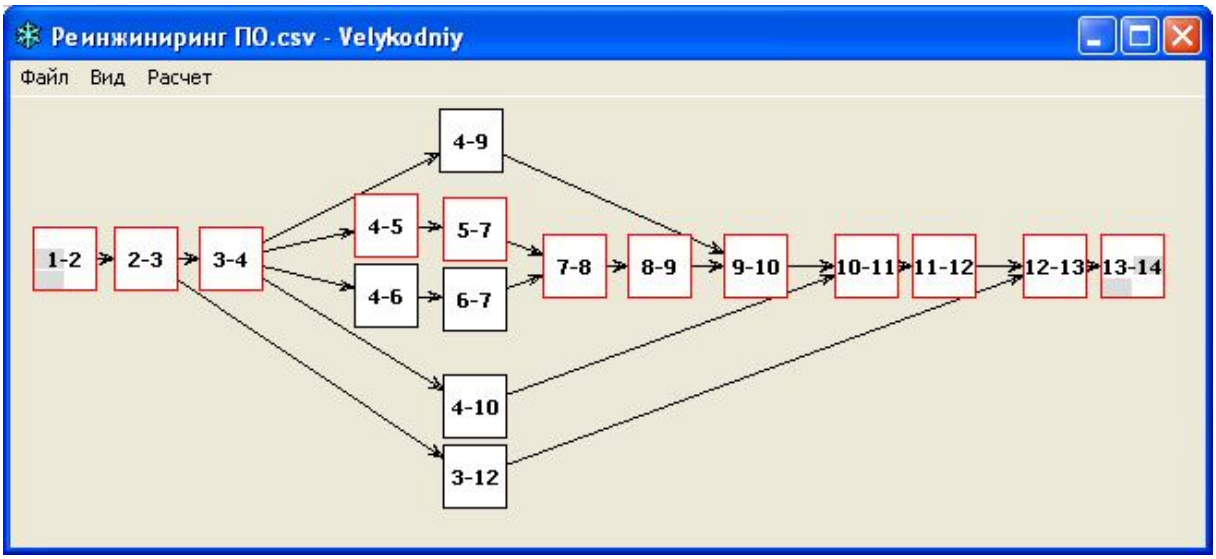
Приклади зміни розміру подання мережевих графіків виконання реінжинірингу ПС для дрібного та середнього розмірів подання задач наведено на рис. 6.28, 6.29; для великого розміру – на рис. 6.17, 6.18.

6.4.3.4 Додаткові файли

Для повноцінної роботи готового ПЗ – досить одного файлу, що виконується («exe»), проте, в момент завершення роботи, у папці із проєктом створюється службовий файл «Options.ini», у якому зберігаються останні обрані користувачем положення й розміри вікна.



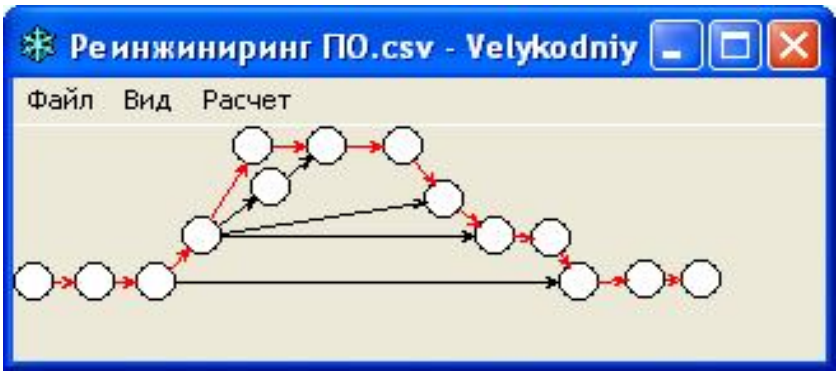
а)



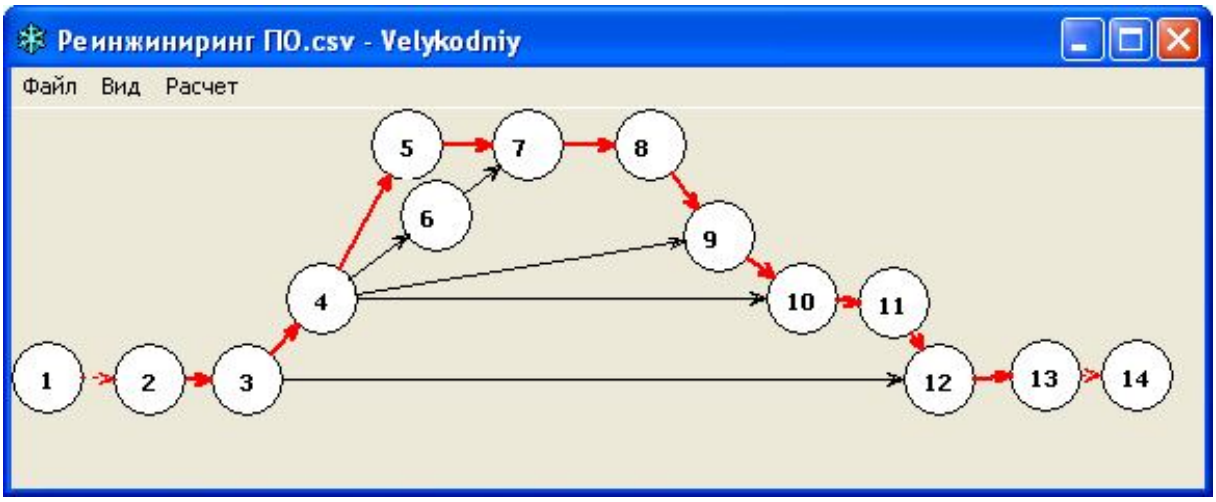
б)

Рисунок 6.28 – Екранні знімки подання мережевого графіка реінжинірингу з:
а) дрібним розміром представлення робіт; б) середнім розміром представлення робіт

Також папка із ПЗ містить усі збережені проєкти мережевих графіків у вигляді файлів з розширенням «* . csv», що складені за певними правилами. При цьому присутня можливість редагування проєктів без входу до ПЗ, що може знадобитись у деяких випадках (призначення завданням ресурсів та ін.) та хоча вся інформація про використані ресурси у файлі проєкту зберігається, для практичного використання цих даних потрібен вже інший ПЗ.



а)



б)

Рисунок 6.29 – Екранні знімки подання мережевого графіка реінжинірингу з:
а) дрібним розміром представлення подій; б) середнім розміром представлення подій

Приклад вмісту збереженого файлу «Реинжиниринг ПО.csv» з останніми змінами параметрів задач мережевих графіків наведено на рис. 6.30.

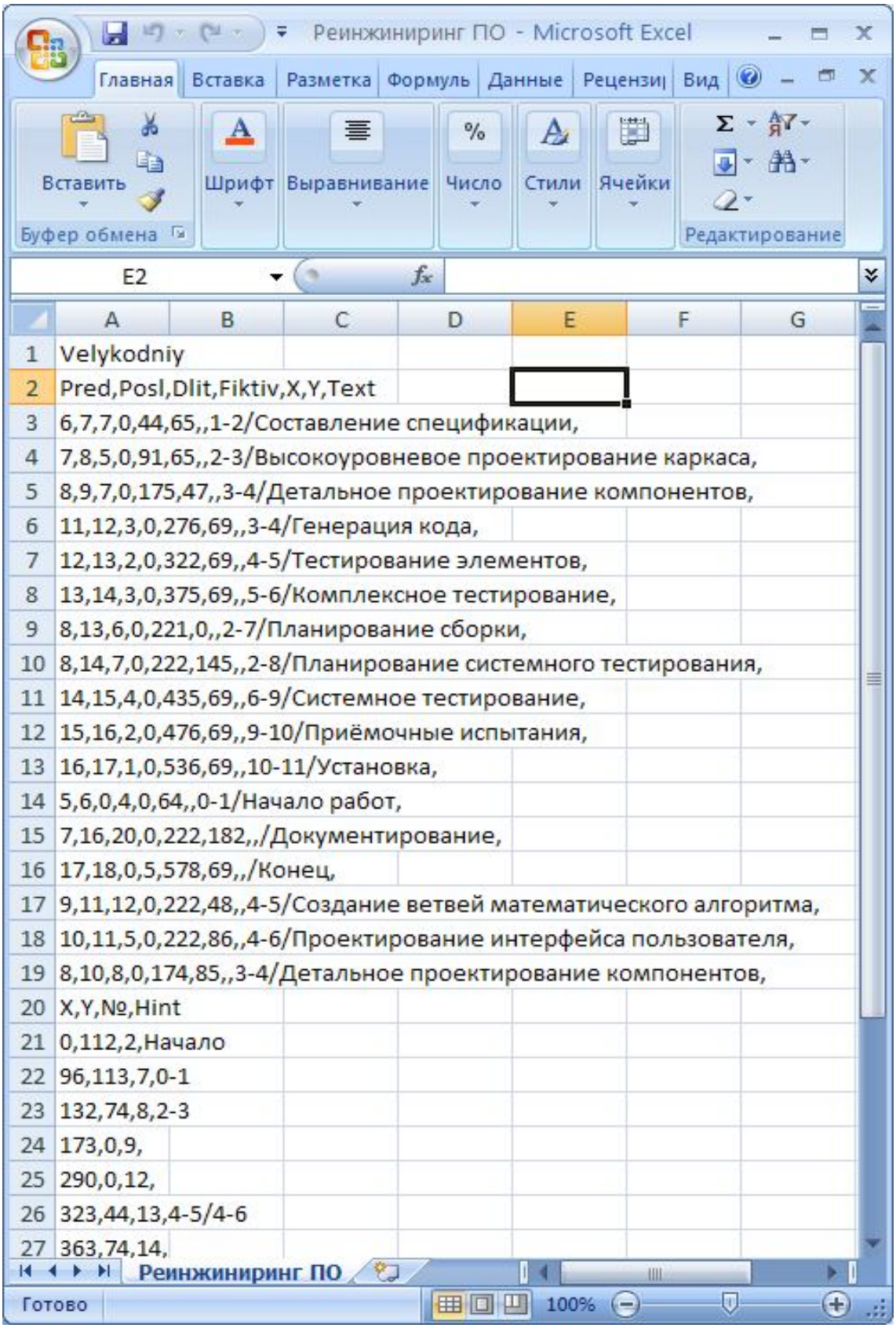


Рисунок 6.30 – Приклад вмісту збереженого csv-файлу з останніми змінами параметрів задач мережевих графіків УП реінжинірингу

6.5 Висновки за розділом

У поданому розділі монографії вирішується встановлене завдання розробки ПМК управління плануванням реінжинірингу ПС та ПП для команди проєктного менеджменту офісу УП. Зміст розділу визначається, по-перше, специфікою теми монографії, по-друге, особливостями конкретних проєктних рішень та технічних пропозицій до роботи. ПМК містить у собі сукупність програмного, інформаційного, математичного, організаційного та методичного забезпечення для виконання робіт із планування та організації розвитку ПС та ПП.

Змодельована та розроблена системна архітектура (проєктний «каркас») ПМК управління плануванням проєктів реінжинірингу ПС, у вигляді поведінкових та структурних моделей УП, виконаних із дотриманням нотації UML 2.5, а саме сформовано:

– модель варіантів використання із вкладеною моделлю управління пакетами;

– модель послідовності розвитку проєкту;

– модель опису станів;

– динамічну модель діяльності;

– модель опису проєктних класів;

– модель компонентних рішень.

Повний діаграмний комплекс УП, що виконано англійською мовою, та адаптований під безпосередню розробку програмного засобу, та наведений у дод. В.

Результатом виконання розділу монографії став розроблений ПМК проєктування мережевих графіків проєкту реінжинірингу ПЗ, який включає:

– опис сфери призначення;

– виділення понять у сформований словник предметної галузі;

– визначення порядку роботи із ПМК, що складається зі створення нової задачі, зміни параметрів задачі, створення зв’язків між задачами, видалення зв’язків між задачами, роботи з подіями;

– генерування спеціальних таблиць;

– розкриття додаткових можливостей;

– реалізація перегляду, друку та збереження інформації, переміщення та зміни розміру структур;

– складання та підключення додаткових файлів.

ПМК створено для використання командою проєктного менеджменту, що входить до складу офісу УП, якій важливо знати лише послідовність робіт та тривалість кожної з них і не має особливого значення, яким способом сформований графік проєкту, тобто якого він типу. У складі ПМК розроблено інструкцію користувача із застосування необхідного ПЗ, що доповнені коментарями, відносно роботи створеного ПЗ. Для роботи з ПМК, вистачить базових знань у галузі мережевого планування. Сам процес створення МГ нескладний, тому навіть його освоєння займе небагато часу.

Слід зазначити, що хоча у сучасних ліцензійних спеціалізованих пакетах програм планування та УП, в основному, використовується тип графіків «вершини-роботи» – створений ПМК здатен працювати з усіма типами мережевих графіків із можливостями їх всебічної трансформації.

При застосуванні у ТОВ «Люксстройпроєкт» (м. Харків) ПМК стосовно до УП, досягнуто скорочення строків проєктування виробничого процесу за методологією мережевого планування за рахунок прискорення складання мережевих графіків організації виробництва, внаслідок використання інтерактивних методів управління мережевою моделлю. Зафіксовано скорочення затраченого часу при проєктуванні мережевих графіків у межах: 16% ÷ 24%; при трансформації робіт у події та навпаки: 87% ÷ 96%.

При плануванні та управлінні ЖЦ рекламної продукції ТОВ «Рекламне агентство «ТРАСТ МЕДІА», що займається її виготовленням, розміщенням, аналізом та просуванням, знайшли своє застосування розроблені методи та засоби управління мережевим плануванням проєкту реінжинірингу ПП. Застосування ПМК управління мережевим плануванням проєкту реінжинірингу, як складової інформаційної технології із організації створення та просування рекламного продукту, скоротило: на 22% процес складання мережевого графіка у порівнянні із його складанням до застосування ПМК; на 17% процес підрахунку часу, що закладається у базову траєкторію ЖЦ проєкту; на 8% процес аналізу ефективності рекламного продукту для прийняття рішення щодо його подальшого реінжинірингу та застосування конкретних методів розвитку й просування.

Основні власні наукові дослідження, що подані автором у шостому розділі монографії, опубліковано у наукових працях автора: [4], [5], [28], [29], [69], [70].

ВИСНОВКИ

Проведене дослідження розв'язує наукову проблему формування нових моделей та розробки нових методів проактивного УП з розвитку ЖЦ ПП, які забезпечують закладання методологічних основ проактивного УП з реінжинірингу ПС, що дозволяє працювати з багаторівневими ПС, за допомогою комплексного інструментарію проєктної оцінки.

1. В результаті проведення аналізу методологій УП зі створення відкритих, вільних та комерційних ПП було з’ясовано, що проактивний розвиток ПС, на відміну від реактивного являє собою величезну науково-технічну проблему, а його впровадження вимагає значних капіталовкладень, оскільки необхідно управляти проєктом з упередженням. За сучасними світовими тенденціями УП ПП повинні бути такими, що розвиваються. Існує, принаймні, дві вагомі причини, за якими ПП повинні змінюватись за часом. По-перше, розробка такого складного об'єкта як ПС займає тривалий час та економічно вигідно вводити в експлуатацію частини системи за еволюційним механізмом: введений в експлуатацію базовий варіант – надалі розширюється. По-друге, постійний прогрес об'єктів проєктування, технологій управління інформацією, обчислювальної техніки та обчислювальної математики зумовлює появу нових, більш досконалих математичних моделей і методів, які повинні замінювати старі менш вдалі аналоги систем УП.

2. Дістала подальшого розвитку систематизація провідних методологій УП, яка розширена формуванням концепції проактивного УП з розвитку ЖЦ ПС та ПП, що дозволяє на етапі оцінки проєкту визначити доцільність цільової програми реінжинірингу. УП реінжинірингу виконується за допомогою комплексу засобів, у тому числі за рахунок застосування КПВ та CASE-систем. Згідно з оптимістичними прогнозами застосування КПВ здатне у 4 рази знизити вартість ПС у порівнянні з новою розробкою. До перспектив дослідження вже після прийняття рішення щодо застосування проєкту реінжинірингу, слід віднести наукові задачі розробки логічної схеми управління реінжинірингом, що є похідною від логічної схеми проєктування, яка починається цільовою програмою та включає в себе вибір компонентів УП реінжинірингу. Ця логічна схема представляє цільовий засіб отримання нової ПС шляхом виконання послідовності операцій внесення змін, модернізації або модифікації, а також перепрограмування окремих її компонентів. Логічна схема УП

реінжинірингу повинна починатися цільовою програмою, об’єднуючи зовнішню та внутрішню частини, відповідно до формування цілей УП.

3. Формалізовано методи проактивного УП розвитку наскрізного проєктування ПС, які відрізняються формуванням парадигми системної трансформації, що дозволяє інтегрувати ПП у оновлені проєктні комплекси. Згідно із загальною методологією УП, задача проактивного розвитку полягає не тільки у з’ясуванні шляхів, а у визначенні характеру конкретної перебудови процесів проєктування ПС, яка об'єднує у собі різні види забезпечення: технічне, математичне, програмне, інформаційне, лінгвістичне, методичне, організаційне, ергономічне та правове. Кожний з цих видів забезпечення має параметри та бажані характеристики управління, які обирають проєктувальники із максимальним урахуванням особливостей завдань інженерного проєктування, конструювання, технологій кодування, виготовлення та інтеграції ПС. Рішення цієї проблеми здійснюється шляхом УП збирання, об'єднання або інтеграції різнорідних програмних ресурсів та КПВ, що включають модулі, бібліотеки та програмні реалізації проєкту деякої складної ПС.

4. Сформовано моделі проактивного управління розвитком лінгвістичного та інформаційного забезпечення проєктів, які відрізняються застосуванням динамічного оточення породжувальних граматик, що дозволяє позбавитися виведення тупикових ланцюжків при переведенні з однієї МП до іншої. Досягнуті результати, з наукової точки зору, увійдуть до основ методології проактивного УП з розвитку ПС та ПП, а з практичної – стануть у нагоді системним програмістам, які задіяні у перепроєктуванні й переробці ПС та ПП, а також системним архітекторам, які працюють із мультимовними надбудовами ПС, що вже знаходяться у кількарічній експлуатації і набувають еволюційного розвитку із плином часу та удосконалення у процесі використання.

5. Удосконалено моделі проактивного управління розвитком поведінкової та структурної частин проєктів з реінжинірингу ПС, які відрізняються уведенням оновлених структур уніфікованих діаграмних моделей, що дозволяє зберігати позитивні властивості відношень між проєктними класами, компонентами та сутностями ПС незалежно від МП. На основі управління системною архітектурою проєкту складено моделі проактивного розвитку ГБД для композиційного підключення до оновленого відкритого проєкту, а саме моделі: варіантів використання; опису станів для проактивного УП; послідовності розвитку графічного

представлення проєкту; діяльності реінжинірингу графічної структури проєкту; опису оточення об'єктів; операцій із проєктними класами; компонентних рішень; управління інформаційним розгортанням проєкту.

6. Удосконалено метод розрахунку показників проєкту за проєктними точками Карнера, який відрізняється внесенням доповнень та розширень щодо процесів реінжинірингу ПС, що дозволяє зменшити похибку у оцінюванні прогнозованого переліку витрат на реалізацію проєкту на 17 – 19%, у порівнянні із методом Карнера. Встановлення коефіцієнтів та обрання значення констант засновані на методі Якобсона та перевірені статистикою найбільш схожих проєктів. Після аналізу завершеного проєкту і вивчення звіту про показники, доступні чинники можуть бути точно відкориговані, щоб дати оцінку часу, необхідного на виконання проєкту реінжинірингу. Згодом, можна використовувати ці дані у якості базової траєкторії ЖЦ проєкту. Вимірювання виконуються командою досвідчених системних аналітиків, що спираються на власні уявлення про технічну складність проєкту та можливості команди програмістів. Коефіцієнти та константи визначаються, виходячи із статистичних даних вже оцінених аналітиками виконаних проєктів із близьким ступенем схожості. Подальший розвиток з досліджень метода: управління автоматизованим формуванням звіту з оцінки ПС, який включався би у вихідну проєктну документацію, необхідну для управління виконанням реінжинірингу кожного з видів забезпечення проєкту: технічного, математичного, інформаційного, лінгвістичного, методичного, організаційного тощо. Ця проєктна документація є обов'язковою складовою частиною видів забезпечення ПС.

7. Сформовані ІМУПР, які відрізняються уведенням ресурсних та часових обмежень на виконання проєкту, що дозволяє підвищити точність оцінки проєктних витрат з перепроєктування ПС. Сформовано представлення оцінки параметрів витрат ресурсів на виконання проєкту реінжинірингу ПС за допомогою математичного інструментарію опису моделей проєктування. Запропоновані ІМУПР видів забезпечення ПС являють собою еволюційні спіралі, які побудовані у циліндричній системі координат. Експериментальні дослідження спіральної ІМУПР за даними 7 виконаних проєктів показали прогнозування витрат із максимальною похибкою 30%. Середнє значення відносної похибки – 19%. Операції із поданими моделями можуть відбуватися у проекціях часу та витрат, у ізометричній проекції програмних компонентів, у логарифмічній проекції

рядків програмного коду. Подальші наукові дослідження у цьому напрямку потребують всебічного дослідження висунутої автором гіпотези щодо представлення кількісних характеристик виконаної роботи з проєкту реінжинірингу ПС та визначення обсягів витрачених ресурсів у формі обчислення площин та об'ємів лінійчатої конічної поверхні, що утворюються годографами проєкту реінжинірингу.

8. Розроблено метод представлення оцінки проєкту з реінжинірингу ПС, який відрізняється застосуванням інструментарію проєктних коефіцієнтів, що дозволяє управляти якісними змінами конфігурації графічних моделей перепроєктування, у вигляді відповідних годографів проєктів реінжинірингу ПС. Максимальне значення відносної похибки оцінки проєкту реінжинірингу ПС за методом проєктних коефіцієнтів за даними 12 виконаних проєктів не перевищує 16%. Середнє значення відносної похибки – 8%. При використанні методу розрахунку показників проєкту за проєктними точками та методу представлення оцінки проєкту з реінжинірингу ПС за допомогою інструментарію проєктних коефіцієнтів, виконано оцінку кожного етапу проєкту з удосконалення ПС у Одеському ім. А. Міцкевича відділенні Спілки поляків в Україні із отриманням економічного показника впровадження у вигляді оцінки вартості кожного окремого етапу проєкту із удосконалення ПС, які не відрізняються від фактичних більш ніж на 7 %.

9. Узагальнено практично-орієнтовані результати комплексних експериментальних досліджень з проактивного УП з розвитку ПС та сформовано рекомендації із розробки системної архітектури програмних продуктів, що планується розвивати. Розроблені рекомендації стануть у нагоді керівникам проєктів, проєктним менеджерам, системним архітекторам та інженерам-програмістам, які задіяні у перепроєктуванні ПС та ПП, що позитивно зарекомендували себе впродовж ЖЦ їх експлуатації.

10. Розроблений ПМК управління плануванням реінжинірингу проєктів для використання командою проєктного менеджменту у складі офісу УП. ПМК містить у собі сукупність програмного, інформаційного, математичного, організаційного та методичного забезпечення для виконання робіт із планування та організації розвитку ПС та ПП. При застосуванні у ТОВ «Люксстройпроєкт» (м. Харків) ПМК стосовно до УП, досягнуто скорочення строків проєктування виробничого процесу за методологією мережевого планування за рахунок прискорення складання

мережевих графіків організації виробництва, внаслідок використання інтерактивних методів управління мережевою моделлю. Зафіксовано скорочення затраченого часу при проєктуванні мережевих графіків у межах: 16% ÷ 24%; при трансформації робіт у події та навпаки: 87% ÷ 96%. При плануванні та управлінні ЖЦ рекламної продукції ТОВ «Рекламне агентство «ТРАСТ МЕДІА», що займається її виготовленням, розміщенням, аналізом та просуванням, знайшли своє застосування розроблені методи та засоби управління мережевим плануванням проєкту реінжинірингу ПП. Застосування ПМК управління мережевим плануванням проєкту реінжинірингу, як складової інформаційної технології із організації створення та просування рекламного продукту, скоротило: на 22% процес складання мережевого графіка у порівнянні із його складанням до застосування ПМК; на 17% процес підрахунку часу, що закладається у базову траєкторію ЖЦ проєкту; на 8% процес аналізу ефективності рекламного продукту для прийняття рішення щодо його подальшого реінжинірингу та застосування конкретних методів розвитку й просування.

Запропоновані моделі та методи, а також ПМК проактивного УП з розвитку ПС та ПП, було впроваджено у НДР та навчальний процес у: Одеському державному екологічному університеті, національному університеті «Одеська морська академія» та Одеській національній академії зв’язку ім. О. С. Попова.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Velykodniy S. Reengineering of open software system of 3D modeling BRL-CAD. *Innovative Technologies and Scientific Solutions for Industries.* 2019. No. 3 (9), P. 62–71. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.9.062.
2. Velykodniy S. S. Analysis and synthesis of the results of complex experimental research on reengineering of open CAD systems. *Applied Aspects of Information Technology.* 2019. Vol. 2. No 3. P. 186–205. (кат. «Б») DOI: 10.15276/aait.03.2019.2.
3. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проектування BRL-CAD. Моделювання структурної частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського*. 2019. Вип. 3 (116). С. 130–139. (кат. «Б») DOI: 10.30929/1995-0519.2019.3.130-139.
4. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Розробка архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. *Сучасний стан наукових досліджень та технологій в промисловості.* 2019. № 2 (8). С. 25–35. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.8.025.
5. Velykodniy S., Burlachenko Zh., Zaitseva-Velykodna S. Software for automated design of network graphics of software systems reengineering. *Scientific Journal Herald of Advanced Information Technology.* 2019. No 2 (03). P. 20–32. (кат. «Б») DOI://10.15276/hait.02.2019.2.
6. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. Реінжиніринг графічних баз даних у середовищі відкритої системи автоматизованого проектування BRL-CAD. Моделювання поведінкової частини. *Вісник Кременчуцького національного університету ім. Михайла Остроградського*. 2019. Вип. 2 (115). С. 117–126. (кат. «Б») DOI: 10.30929/1995-0519.2019.2.117-126.
7. Великодний С. С. Метод представлення оцінки реінжинірингу програмних систем за допомогою проектних коефіцієнтів. *Сучасний стан наукових досліджень та технологій в промисловості.* 2019. № 1 (7). С. 34–42. (кат. «Б») DOI: https://doi.org/10.30837/2522-9818.2019.7.034.

8. Великодний С. С. Ідеалізовані моделі реінжинірингу програмних систем. *Радіоелектроніка, інформатика, управління.* 2019. № 1. С. 150–156. DOI: 10.15588/1607-3274-2019-1-14.
9. Великодний С. С., Тимофєєва О. С., Зайцева-Великодна С. С. Метод розрахунку показників оцінки проекту при виконанні реінжинірингу програмних систем. *Радіоелектроніка, інформатика, управління.* 2018. № 4. С. 135–142. DOI: 10.15588/1607-3274-2018-4-13.
10. Великодний С. С., Тимофєєва О. С., Зайцева-Великодна С. С., Нямцу К. Є. Порівняльний аналіз властивостей відкритого, вільного та комерційного програмного забезпечення. *Інформаційні технології та комп'ютерна інженерія*. 2018. № 1 (41). С. 21–27.
11. Великодний С. С., Тимофєєва О. С. Спосіб мультилінгвістичного перекодування програмного забезпечення складних інформаційних систем та технологій. *Наукові праці ОНАЗ ім. О. С. Попова*. 2017. № 2. С. 153–159.
12. Velykodniy S., Tymofieieva O. The paradigm of linguistic supply submission by generative grammar assistance. *American scientific journal*. 2017. № 17. P. 4–7. *Закордонне видання (New York, USA)*.
13. Великодний С. С., Тимофєєва О. С. Парадигма подання лінгвістичного забезпечення за допомогою породжувальних граматик (Част. 2). *Automation of technological and business-processes*. 2017. Vol. 9, Iss. 3. С. 46–50.
14. Великодний С. С., Тимофєєва О. С. Парадигма подання лінгвістичного забезпечення за допомогою породжувальних граматик (Част. 1). *Розвиток транспорту*. 2017. № 1. С. 128–135.
15. Velykodniy S., Tymofieieva O. Multilingual recording method designed for SCADA-system's software upgrade. *Automation of technological and business-processes.* 2017. Vol. 9, Iss. 1. P. 17–22.
16. Великодний С. С. Методи реінжинірингу програмних систем. *Технологии приборостроения*. 2014. Спец. вып. С. 65–68.
17. Великодный С. С. Методологические основы реинжиниринга систем автоматизированного проектирования. *Управляющие системы и машины*. 2014. № 2. С. 39–43.
18. Великодный С. С. Проблема реинжиниринга видов обеспечения систем автоматизированного проектирования. *Управляющие системы и машины*. 2014. № 1. С. 57–61, 76.

19. Зайцева-Великодна С. С., Великодний С. С. Проблеми атестації обчислювальної компоненти програмних комплексів автоматизованих систем комерційного обліку електроенергії. *Системы обработки информации*. Вып. 99. 2012. С. 152–156.
20. Великодный С. С. Реализация процесса «сквозного» проектирования с помощью CAD/CAM «ADEM». *Холодильна техніка і технологія.* 2011. № 1 (129). С. 56–59.
21. Великодный С. С., Котенко Е. В. Разработка системы реального времени распознавания лиц с использованием примитивов Хаара и алгоритма ADABOOST. *Холодильна техніка і технологія*. 2010. №5 (127). С. 67–70.
22. Великодний С. С. Проектування траєкторії руху технологічного обладнання при виготовленні елементів суднових конструкцій. *Автоматизация судовых технических средств*. Вып. 16. 2010. С. 10–18.
23. Невлюдов И. Ш., Великодный С. С., Фомовская Е. В. Построение годографов Михайлова с помощью пакета «MathCAD». *Вопросы проектирования и производства конструкций летательных аппаратов.* Вып. 3 (63). 2010. С. 186–193.
24. Невлюдов І. Ш., Великодний С. С., Фомовська О. В. Моделювання електромеханічної частини маніпулятора промислового робота. *Вопросы проектирования и производства конструкций летательных аппаратов.* Спец. вып. 2010. С. 181–185.
25. Великодний С. С., Стрілець В. О. Деякі питання атестації автоматизованих систем комерційного обліку електроенергії. *Український метрологічний журнал.* 2010. № 2. С. 40–44.
26. Невлюдов И. Ш., Великодный С. С., Омаров М. А. Использование CAD/CAM/CAE/CAPP при формировании управляющих программ для станков с ЧПУ. *Восточно-Европейский журнал передовых технологий*. 2010. № 2/2 (44). С. 37–44.
27. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Graphic databases reengineering in BRL-CAD open source computer-aided design environment. Modeling of the structural part. Information control systems and technologies: VIII international scientific-practical conference: materials. Odessa. Ukraine, 23 – 25 sep. 2019. Odessa: «Ecologia», 2019. P. 222–224.

28. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Software for automated design of network graphics of software systems reengineering. Матер. міжн. наук.-практ. конф. «Інтелектуальні системи та інформаційні технології ISIT-2019». Одеса. Україна. 19 – 24 серпня 2019 р. Одеса: ТЕС, 2019. С. 253–257.
29. Burlachenko T., Великодний С. С. Розробка системної архітектури програмного засобу для управління мережевим плануванням реінжинірингу програмного проекту. Міжн. наук. конф. «Економіко-екологічні проблеми сучасності у дослідженнях науковців». м. Одеса. Україна. 25 – 26 черв. 2019 р. Одеса: ТЕС, 2019.С. 20–26.
30. Burlachenko Zh. V., Velykodniy S. S. Graphic databases reengineering in BRL-CAD open source computer-aided design environment. Modeling of the behavior part. Матеріали III-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 22 чер. 2019. С. 7–9.
31. Зайцева-Великодна С. С., Великодний С. С. Реінжиніринг графічних баз даних у середовищі відкритої САПР BRL-CAD. Моделювання структурної частини. Матеріали III-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 22 чер. 2019. С. 27–29.
32. Великодний С. С., Нямцу К. Є. Сучасна інформаційна система для підготовки LaTeX-документів «TeXstudio». Інформатика, інформаційні системи та технології: тези доп. 16-ї Всеукр. конф. студ. і мол. наук. Одеса, 19 квітня 2019 р. Одеса, 2019. С. 11–13.
33. Velykodniy S. S., Burlachenko Zh. V., Zaitseva-Velykodna S. S. Method of presenting the assessment for reengineering of software systems with the project coefficients help. Інформаційні технології в моделюванні: матер. IV-ої всеукр. наук.-практ. конф. студ., аспірантів та мол. вчених, 21–22 березня 2019 р., м. Миколаїв. Миколаїв: МНУ ім. В. О. Сухомлинського, 2019. С. 10–11.
34. Великодний С. С., Бурлаченко Ж. В., Зайцева-Великодна С. С. LaTeX-орієнтовані системи підготовки наукових текстів. Міжн. наук. Інтернет-конф. «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (вип. 35)». Зб. тез доп.: вип. 35, м. Тернопіль, 5 лют. 2019 р. Тернопіль. 2019. С. 6–8.
35. Петухін Д. О., Великодний С. С., Козловська В. П. Застосування методів реляційної алгебри для задачі автоматизованого складання розкладу занять. Information Control Systems and Technologies: VII

International scientific-practical conference: materials, Odessa, Ukraine, 17 – 18 sep. 2018. Odessa: «Astroprint», 2018. P. 80–83.
36. Тимофєєва О. С., Великодний С. С., Зайцева-Великодна С. С. Метод розрахунку показників оцінки проекту при виконанні реінжинірингу програмних систем. Матеріали ІІ-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 26 – 28 лип. 2018. С. 18–19.
37. Великодний С. С., Зайцева-Великодна С. С. Ідеалізовані моделі реінжинірингу програмних систем. Матеріали ІІ-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 26 – 28 лип. 2018. С. 15–17.
38. Великодний С. С. Ідеалізовані моделі реінжинірингу програмних систем. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: ІІІ Міжн. наук.-практ. конф., 6 – 8 черв. 2018 р. Одеса, 2018. С. 80–83.
39. Tymofieieva O., Velykodniy S., Zaitseva-Velykodna S. Method of calculating project evaluation indicators for the implementation of software system reengineering. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: ІІІ Міжн. наук.-практ. конф., 6 – 8 черв. 2018 р. Одеса, 2018. С. 68–71.
40. Velykodniy S., Tymofieieva O. Reengineering Models of Linguistic Providing Software Systems. Advances in Quantum Systems in Chemistry, Physics and Mathematics, Ser.: Progress in Applied Mathematics and Quantum Optics, Eds. A. Glushkov, O. Khetselius, V. Buyadzhi. Kharkiv: FOP Panov, 2017. P. 385–388.
41. Великодний С. С., Тимофєєва О. С., Нямцу К. Є. Парадигма подання лінгвістичного забезпечення за допомогою формальних граматик. Information Control Systems and Technologies: VI International scientific-practical conference: materials, Odessa, Ukraine, 20 – 22 sep. 2017. Одеса: «ВидавІнформ НУ «ОМА», 2017. С. 266–268.
42. Великодний С. С., Тимофєєва О. С. Спосіб мультилінгвістичного перекодування програмного забезпечення SCADA-систем. Матеріали І-го Всеукраїнського пленера з питань природничих наук, Одеса, Україна, 20 – 23 лип. 2017. С. 7–8.
43. Великодний С. С., Сирчин Д. О. Аналітика та рефакторинг інтерфейсів CRM-систем Dynamical system modeling and stability investigation: XVIII International Conference: Abstract of Conf. Reports,

Kyiv, Ukraine, 24 – 26 may 2017. Київ, ДП Інформ.-аналіт. агентство, 2017. С. 194.
44. Великодний С. С., Тимофєєва О. С. Ідеалізовані моделі реінжинірингу видів забезпечення програмних систем. Dynamical system modeling and stability investigation: XVIII International Conference: Abstract of Conf. Reports, Kyiv, Ukraine, 24 – 26 may 2017. Київ: ДП Інформ.-аналіт. агентство. 2017. С. 195.
45. Великодний С. С., Тимофєєва О. С. Порівняльний огляд європейської та вітчизняної наукової діяльності у галузі інформаційних технологій. Управління якістю підготовки фахівців: зб. тез доп. Всеукр. наук.-мет. конф., 21 – 22 лют. 2017 р. Одеса, 2017. С. 9–10.
46. Великодний С. С., Тимофєєва О. С. Базові методи реінжинірингу програмних компонентів. V International scientific-practical conference “Information Control Systems and Technologies”: materials ICST-Odessa, 20 – 22 sep. 2016. Odessa, 2016. С. 293–295.
47. Сырчин Д. А., Великодный С. С. Методы лингвистического транслирования языковых конструкций. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: Всеукр. наук-практ. конф., 20 – 22 квіт. 2016 р. Одеса, 2016. С. 100–101.
48. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров для систем автоматизированного проектирования формообразования деталей. Теоретичні та прикладні аспекти застосування інформаційних технологій в галузі природничих наук: Всеукр. наук-практ. конф., 20 – 22 квіт. 2016 р. Одеса, 2016. С. 32–33.
49. Великодний С. С. Модель реінжинірингу програмного забезпечення SCADA-систем, що застосовуються на транспорті. Новітні технології в автомобілебудівництві та транспорті: наук. праці міжнар. наук-практ. конф. до 85-річ. ХНАДУ, 15–16 жов. 2015 р. Харків, 2015. С. 104–105.
50. Великодний С. С. Реінжиніринг систем моніторингу та дистанційного управління судновими енергетичними установками. Матер. XXII міжн. конф. з автом. управл. «Автоматика 2015», 10–11 вер. 2015, Одеса, 2015. С. 133–134.
51. Великодний С. С., Тимофєєва О. С., Онищенко С. М. Моделювання інформаційного сполучення компонентів SCADA-систем. XII

International Conference “Strategy of Quality in Industry and Education”: proceedings, 30 May – 2 June 2016, Varna, Bulgaria. С. 511–518.
52. Орлов В. В., Великодный С. С., Бережной К. Ю. Информационная технология совершенствования судовых систем приёма внешних звуковых сигналов. Современные информационные и электронные технологии: труды XVI междун. науч.-практ. конф., 25–29 мая 2015 г. Одесса, 2015. С. 107–108.
53. Великодний С. С. Методологічні засади реінжинірингу SCADA-систем. Матер. конф. молод. вчених ОДЕКУ, 11 – 15 трав. 2015 р. Одеса, 2015. С. 95–97.
54. Великодный С. С. Специализированные САПР для моделирования и исследования судовых компьютерных сетей. Енергетика судна: Експлуатація та ремонт: матер. наук.-техн. конф., 26 – 28 бер. 2014 р. Ч. 2. Одеса, 2014. С. 116–117.
55. Великодный С. С., Сырчин Д. А. Способы лингвистического транслирования языковых конструкций. International scientific-practical conference “Information Control Systems and Technologies”: materials ICST-Odessa, 8 – 10 oct. 2013. Odessa, 2013. P. 318–321.
56. Великодный С. С. Особенности реинжиниринга программного обеспечения открытых САПР. IX International Conference “Strategy of Quality in Industry and Education”: proceedings. Vol. 1, Varna, Bulgaria, 31 May – 7 June 2013. Dnipropetrovsk-Varna, 2013. P. 304–307.
57. Великодний С. С. Передумови реінжинірингу систем автоматизованого проектування. Математичне моделювання та інформаційні технології: тези доп. XI Всеукр. наук.-техн. конф., 21 – 23 лист. 2012 р. Одеса, 2012. С. 23.
58. Великодний С. С. Методологія зворотного проектування програмного забезпечення САПР. Информационные технологии и автоматизация – 2012: тез. докл. V Всеукр. науч.-практ. конф., 10 – 11 окт. 2012 г. Одесса, 2012. С. 10–11.
59. Великодный С. С. Технико-экономический анализ внедрения системы автоматизированного производства. Математичне моделювання та інформаційні технології: тези доп. X Всеукр. наук.-техн. конф., 23 – 25 лист. 2011 р. Одеса, 2011. С. 24–25.
60. Невлюдов І. Ш., Великодний С. С., Фомовська О. В. Побудова годографів Михайлова за допомогою пакета «MathCAD». XX International conference “New leading technologies in machine building”:

collect. of the scient. papers, Rybachie, 3 – 8 sept. 2010. Kharkov, 2010. P. 13.

61. Невлюдов И. Ш., Великодный С. С. Фомовская Е. В. Моделирование электромеханической части манипулятора промышленного робота. XX International conference “New leading technologies in machine building”: collect. of the scient. papers, Rybachie, 3 – 8 sept. 2010. Kharkov, 2010. P. 13.
62. Лалашкова А. И., Великодный С. С. Выбор технологии организации связи для рационального функционирования производственной сети. Радиоэлектроника и молодёжь в XXI веке: матер. XIV Межд. молод. форума 18 – 20 мар. 2010 г. Ч. 1. Харьков, 2010. С. 370.
63. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров для систем автоматизированного проектирования формообразования деталей. Монографія. Харків: ФОП Панов А.Н., 2017. 232 с.
64. Великодный С. С. Модели и методы интерполяции сложных геометрических контуров. Для систем автоматизированного проектирования формообразования деталей. Монография. 2-е изд. Эрфурт: LAMBERT Academic Publishing, 2019. 236 с.
65. Великодний С. С. Реінжиніринг програмних систем та продуктів. Управління IT-проектами. Монографія. Нордерштедт: LAMBERT Academic Publishing, 2019. 160 с.
66. Великодний С. С. Спосіб виявлення прихованого кабелю між поверхнями. Патент на кор. мод. № 119764, зар. в Держ. реєстрі пат. України на кор. мод. 10.10.2017 р. 8 с.
67. Великодний С. С. Спосіб розмітки під отвори. Патент на кор. мод. №109897, зар. в Держ. реєстрі пат. України на кор. мод. 12.09.2016 р. 10 с.
68. Великодний С. С. Спосіб мультилінгвістичного перекодування компонентів систем автоматизованого проектування (Спосіб МЛП КСАП). Свід. про реєстр. авт. пр. на тв. № 48350; заявл. у Держ. службу інт. власн. 17.01.2013, заява № 48624; зареєстр. 18.03.2013. 16 с.
69. Великодний С. С. Імітаційне моделювання. Навчальний посібник. Одеська державна академія холоду. Одеса: Вид. центр ОДАХ, 2011. 188 с.

70. Великодний С. С. Дипломне проектування. Організація, вимоги щодо структури та оформлення пояснювальної записки, порядок захисту. Посібник для виконання дипломної роботи спеціаліста студентами денної та заочної форм навч. спец. 7.05010102 «Інформаційні технології проектування». Одеська державна академія холоду. Одеса: Вид. центр ОДАХ, 2011. 36 с.
71. Glushkov O. V., Velykodniy S. S., Buyadzhi V. V. Methodical instructions for laboratory work on discipline. “Mathematical Methods of Operations Research”. Odessa: Odessa State Environmental University, 2016. 24 p.
72. Глушков О. В., Великодний С. С., Буяджи В. В. Методичні вказівки для самостійної роботи студентів і виконання контрольної роботи з дисципліни «Математичні методи дослідження операцій». Одеса: ОДЕКУ, 2016. 21 с.
73. Великодний С. С., Онищенко С. М. Математичні методи дослідження операцій. Методичні вказівки по виконанню лабораторних робіт Одеський державний екологічний університет. Одеса, 2015. 96 с.
74. Бушуева Н. С. Модели и методы проактивного управления программами организационного развития: монография. Киев: Науковий світ, 2007. 200 с.
75. Онищенко І. І. Аналіз ризиків в процесі управління ІТ-проектами. *Вісник НТУ «ХПІ»*. 2014. № 2. С. 95–100.
76. Чечет А.М. Управління проектами на етапах життєвого циклу проекту. *Управління проектами, системний аналіз і логістика*. 2012. Вип. 10. С. 602–606.
77. Бушуев С. Д., Гогунський В. Д., Кошкін К. В. Напрями дисертаційних наукових досліджень зі спеціальності «Управління проектами та програмами». *Управління розвитком складних систем.* 2012. № 12. С. 5–7.
78. ГОСТ Р ИСО 21500–2014. Руководство по проектному менеджменту. Москва: Стандартинформ, 2015. 50 с.
79. ISO/DIS 21500. Guidance on project management. Geneva: International Organization for Standartization, 2011. 44 p.
80. Бушуев С. Д., Бушуева Н. С. Модели и методы стратегического развития организаций от видения к реальности. *Управління проектами та розвиток виробництва.* 2005. № 4 (16). Луганськ: вид-во СНУ ім. В. Даля. С. 5–13.

81. Tanaka H., Bushuyev S. Innovative development and meta program management of a new generation of mega projects in the oil & gas and infrastructure sectors. *Управління розвитком складних систем.* 2013. № 16.
82. Draft international standard ISO / DIS 21500. Guidance on project management. Voting begins on 2011-04-04. International Organization for Standardization, 2011. 44 p.
83. ISO/IEC/IEEE 42010:2011. Systems and software engineering – Architecture description. Publication date: 2011-12. 37 p.
84. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. Publication date: 2018-11. Ed. 2. 92 p.
85. Тимченко А.А. Основи системного проектування та системного аналізу складних об'єктів: Основи САПР та системного проектування складних об'єктів. 2-ге вид. Київ: Либідь, 2003. 272 с.
86. Іванов В. В. Моделі проекту зворотного інжинірингу. *Вісник НТУ «ХПІ».* 2017. № 2 (1224) С. 52–57. DOI: 10.20998/2413-3000.2017.1224.9
87. Шахов А. В. Проектирование жизненного цикла ремонтопригодных технических систем. Одесса: Феникс, 2005. 164 с.
88. Концепція побудови автоматизованих систем комерційного обліку електроенергії в умовах енергоринку: затв. Мінпаливенерго України від 17.04.2000 р., нак. № 32/28/28/276/75/54. К.: Держспоживстандарт, 2008. 25 с.
89. Писаренко Д. М. Инструментальные средства проектирования многофункциональных самоорганизующихся мобильных роботов. *Искусственный интеллект.* 2005. №1. С. 86–92.
90. Unigraphics Direct Interface: Reference Manual. Southampton: ICEM Ltd., 2004. 39 p.
91. Бормалев С., Червонных С. Практическое применение EDS Unigraphics в авиастроении. *Открытые системы*. 1997. №2. С. 43–46.
92. Краснов М., Чигишев Ю. Unigraphics для профессионалов. Москва: ЛОРИ, 2004. 320 с.
93. Горитов А. Н., Кориков А. М. Оптимальность в задачах проектирования и управления роботами. *Автоматика и телемеханика.* 2001. №7. С. 82–90.

94. Фаголь И. Об одном подходе к проектированию паспортов в САПР-ЧПУ/2000. Пермь: ООО «Евразия Лимитед», 2002. 52 с.
95. Филиппович К. В. Некоторые аспекты настройки пользовательских предпочтений в САПР-ЧПУ/2005. Пермь: ООО «Евразия Лимитед», 2005. 48 с.
96. Филиппович К., Попович И. ToolStore – среда для ведения библиотеки инструментов в верификаторе CNC-Verify системы САПР-ЧПУ/2005. Пермь: ООО «Евразия Лимитед», 2005. 36 с.
97. Wakeford L. How Your Design Can Affect The Cost, Quality And Time Required To Manufacture Parts. *MCADVision Magazine*. 2001. July, Part 1. P. 68–71.
98. Diehl B. CAD/CAM a la Carte: A modular approach to choosing machining software. *CNC Machining Magazine*. 2001. Vol.5, #16. P. 54–57.
99. Lynch M. The Key Concepts Of Computer Numerical Control. New Mexico: CNC Concepts Inc., 2004. 60 p.
100. Калачёв О. Н. Моделирование в CAD/CAM Cimatron механообработки на станке с ЧПУ. Ярославль: ЯГТУ, 2003. 28 с.
101. Калачёв О. Н., Рехтер А. Д. Моделирование размеров механообработки в среде AutoCAD 200x на основе использования приложения GRAKON7. *САПР и графика.* 2002. №2. С. 100–104.
102. Калачёв О.Н. Документация по программным продуктам: «KON7 Расчет технологических размерных цепей»; «GRAKON7 Автоматизированное построение в среде AutoCAD 2000 размерной схемы технологического процесса механообработки». Ярославль: ЯГТУ, 2000. 94 с.
103. Грунина Е. В., Калачёв О. Н. Методика проектирования в CAD/CAM Cimatron УП для гравирования профиля на юбилейной медали. Ярославль: ЯГТУ, 2008. 15 с.
104. Митрофанов В. Г., Калачев О. Н., Схиртладзе А. Г. САПР в технологии машиностроения: учеб. пособие. Ярославль: ЯГТУ, 2001. 298 с.
105. Зильбербург Л. И., Марьяновский С. М., Молочник В. И., Яблочников Е. И. Компьютерное проектирование и производство: моногр.; под общ. ред. С. М. Марьяновского. Санкт-Петербург: КПЦ «МиР», 1998. 166 с.

106. Филиппович К.В. Идеология постпроцессирования в современных CAD/CAM-системах. Пермь: ООО «Евразия Лимитед», 2000. 60 с.
107. Zelinski P. A Better Process From Better Posts. *Serving the Metalworking Industries*. 2001. №2. P. 28–31.
108. Филиппович К. В. Импорт DXF-файлов в GrafCAM v.6.20. Новые возможности для технолога. Пермь: ООО «Евразия Лимитед», 2005. 52 с.
109. Филиппович К. В. Редактирование геометрии и технологических команд в GrafCAM v.7. Пермь: ООО «Евразия Лимитед», 2005. 46 с.
110. Воскобойников Ю. С., Филиппович К. В. Внешний редактор в Grafcam v.7.08 – решение, проверенное временем. Пермь: ООО «Евразия Лимитед», 2006. 50 с.
111. Трухнин Н. М., Филиппович К. В. Макрорасширения в GrafCAM – путь к наращиванию функциональности.Пермь: ООО «Евразия Лимитед», 2006. 72 с.
112. Goodbye, «free software»; hello, «open source». URL: http://www.webcitation.org/617oVjlKk (дата звернення: 10.01.2019).
113. Categories of free and nonfree software. URL: http://www.gnu.org/philosophy/ categories.en.html (дата звернення: 10.01.2019).
114. What is free software? URL: http://www.gnu.org/philosophy/free-sw.en.html (дата звернення: 10.01.2019).
115. High Priority Free Software Projects by Free Software Foundation. URL: https://www.fsf.org/ campaigns/priority-projects/ (дата звернення: 10.01.2019).
116. Open Source Paradigm Shift by Tim O'Reilly. URL: http://archive.oreilly.com/ pub/a/oreilly/tim/ articles/paradigmshift_0504.html (дата звернення: 10.01.2019).
117. Дейт К. Дж. Введение в системы баз данных. Изд. 8-е. Москва: Вильямс, 2005. 1328 с.
118. Haigh T. How Data Got its Base: Information Storage Software in the 1950s and 1960s. IEEE Annals of the History of Computing, 2010. Vol. 31. Iss. 4. P. 6–25. DOI: 10.1109/MAHC.2009.123.
119. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. Москва: Вильямс, 2003. 1088 с.
120. Date C. J. Date on Database: Writings 2000–2006. New York City: Apress, 2006. 566 p.

121. Когаловский М. Р. Перспективные технологии информационных систем. Москва: ДМК Пресс; Компания АйТи, 2003. 288 с.
122. Когаловский М. Р. Энциклопедия технологий баз данных. Москва: Финансы и статистика, 2012. 460 с.
123. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е изд. М.: Вильямс, 2003. 1436 с.
124. Мирошниченко Е. А. К формальному определению понятия «база даных». Проблемы информатики. 2011. № 2. С. 83–87.
125. Норенков И. П. Автоматизированное проектирование. Москва: МГТУ им. Н. Э. Баумана, 2000. 188 с.
126. Ли Дж., Уэр Б. Трёхмерная графика и анимация. Изд. 2-е. Москва: Вильямс, 2002. 640 с.
127. Концевич В. Г. Твердотельное моделирование машиностроительных изделий в Autodesk Inventor. Киев, Москва: ДиаСофтЮП, ДМК Пресс, 2007. 672 с.
128. Херн Д., Бейкер М. П. Компьютерная графика и стандарт OpenGL. Изд. 3-е. Москва: Вильямс, 2005. 1168 с.
129. Энджел Э. Интерактивная компьютерная графика. Вводный курс на базе OpenGL. Изд. 2-е. Москва: Вильямс, 2001. 592 с.
130. Снук Г. 3D-ландшафты в реальном времени на C++ и DirectX 9. Изд. 2-е. Москва: Кудиц-пресс, 2007. 368 с.
131. Козир А. Є., Славко Г. В. Алгоритми і методи визначення складних контурів динамічних об’єктів з використанням технологій web-графіки. Вісник Кременчуцького національного університету імені Михайла Остроградського. 2016. Вип. 3 (98). Ч. 1. С. 20–26.
132. Когаловский М. Р. Энциклопедия технологий баз данных. Москва: Финансы и статистика, 2012. 460 с.
133. Вамболь В. В. Идентификация источников формирования экологической опасности в местах несанкционированного скопления отходов. Вісник Кременчуцького національного університету імені Михайла Остроградського. 2016. Вип. 1 (96) С. 122–128.
134. Date C. J. Date on Database: Writings 2000–2006. New York City: Apress, 2006. 566 p.
135. Когаловский М. Р. Перспективные технологии информационных систем. Москва: ДМК Пресс; Компания АйТи, 2003. 288 с.
136. Bondy J. A., Murty U. S. R. Graph Theory with Applications. New York: North-Holland, 1986. 270 p.

137. Березина Л. Ю. Графы и их применение: Пособие для учителей. Москва: Высшая школа, 1989. 326 с.
138. Михеева В. С. Математические методы в экономической географии. Ч. 2. Приложение теории графов: Курс лекций. Москва: Математика, 1983. 316 с.
139. Голиков А. П., Трофимов А. М., Черванёв И. Г. Математические методы в географии. Харьков: Наука, 1986. 208 с.
140. Глазжан І. М., Новиков В. Г. Основи мережевого планування та управління. Харків : Вид-во ХГУ, 1996. 198 с.
141. Оре О. Теория графов. Москва: Наука, 1968. – 336 с.
142. Уилсон Р. Введение в теорию графов: пер. с англ. Москва: Мир, 1977. 208 с.
143. Спекторський І. Я. Дискретна математика. Київ: «Політехніка», 2004. 220 с.
144. Мала гірнича енциклопедія. У 3-х т. За ред. В. С. Білецького. Донецьк: «Донбас», 2004. 518 с.
145. Jalote P. Software project management in practice. Indianapolis: Addison-Wisley, 2005. 242 p.
146. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Eight Edition. New Jersey: John Wiley & Sons, 2003. 914 p.
147. Schulte-Zurhausen Manfred: Organisation. 3-Auflage. Verlag Franz Vahlen: München, 2002. 284 p.
148. Program Evaluation and Review Technique (PERT). URL: https://www.inc.com/encyclopedia/program-evaluation-and-review-technique-pert.html (дата звернення: 01.05.2019).
149. Punmia B. C., Khandelwal K. K. *Project Planning and Control with PERT and CPM.* New Delhi: Laxmi Publications, 2016. 258 p.
150. Blum B. Software engineering: a holistic view. URL: https://dl.acm.org/citation.cfm?id=SERIES9569.128915 (дата звернення: 12.01.2019).
151. Бушуєв С. Д., Гогунський В. Д., Кононенко І. В. Формула та напрями наукових досліджень зі спеціальності «Управління проектами та програмами». Управління проектами: стан та перспективи : VIII Міжнар. наук.-практ. конф. Миколаїв: НУК, 2012. С. 28 – 31.
152. Klein M. Reengineering methodologies and tools. A Prescription for Enhancing Success. URL: https://www.tandfonline.com/doi/abs/10.1080/

10580539408964633 (дата звернення: 12.01.2019). DOI: 10.1080/10580539408964633.
153. Link D., Behnam P., Moazeni R., Boehm B. The Value of Software Architecture Recovery for Maintenance (Submitted on 23 Jan 2019 in Cornell University). URL: https://arxiv.org/abs/1901.07700 (дата звернення: 27.06.2019).
154. Manganelli R., Klein M. The Reengineering Handbook: A Step-by-Step Guide to Business Transformation. URL: https://www.sciencedirect.com/science/article/pii/ S00https://journals.lww.com/jhqonline / Citation/1995/03000/ The_Reengineering_Handbook _A_Step_by_Step_Guide.11.aspx (дата звернення: 12.01.2019). DOI: 10.1097/01445442-199503000-00011.
155. Grover V., Malhotra M. Business process reengineering: A tutorial on the concept, evolution, method, technology and application. URL: https:// www.sciencedirect.com/science/article/abs/pii/S0272696396001040 (дата звернення: 12.01.2019). DOI: 10.1016/S0272-6963(96)00104-0.
156. Boehm B. A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 1986. Vol. 11, Iss. 4. P. 14–24. DOI: 10.1145/12944.12948.
157. Hammer M., Champy J. Reengineering the corporation: A manifesto for business revolution. URL: https://www.sciencedirect.com/science/article/pii/ S0007681305800643?via%3Dihub (дата звернення: 12.01.2019). DOI: 10.1016/S0007-6813(05)80064-3.
158. Норенков И. П. Основы автоматизированного проектирования: учеб. для вузов. 4-е изд., перераб. и доп. М.: Изд-во МГТУ им. Н. Э. Баумана, 2009. 430 с.
159. Тимченко А. А. Основи системного проектування та системного аналізу складних об'єктів. Кн. 1. Основи САПР та системного проектування складних об’єктів. Київ: Либідь, 2003. 272 с.
160. Subriadi A. P., Muqtadiroh F. A., Dewi R. S. A model of owner estimate cost for software development project in Indonesia. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2175 (дата звернення: 27.06.2019).
161. Boehm B. Software Risk Management. URL: https://link.springer.com /chapter/10.1007%2F3-540-51635-2_29 (дата звернення: 13.11.2018). DOI: 10.1007/3-540-51635-2_29.

162. Пантелеймонов А. А. Аспекты реинженерии приложений с графическим интерфейсом пользователя. *Проблемы программирования.* 2001. № 1–2. С. 53–62.
163. Фаулер М. Рефакторинг: улучшение соответствующего кода. Санкт-Петербург: Символ-Плюс, 2003. 432 с.
164. Boehm B. Spiral Development: Experience, Principles and Refinements: special Report. CMU. SEI-2000-SR-008. 2000. 37 p.
165. Selby R. W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management and Research. New Jersey: John Wiley & Sons, 2007. 818 p.
166. Любченко В. В. Правила декомпозиції при формуванні моделі предметної області для навчальних курсів. *Восточно-Европейский журнал передовых технологий.* 2009. Т. 1, № 2(37). С. 42–44.
167. Лаврищева Е. М., Грищенко В. Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. доп. и перераб. Киев: Наук. думка, 2009. 372 с.
168. Jacobson I., Ericsson M., Jacobson A. The Object Advantage: Business Process Reengineering with Object Technology. ACM Press. URL: http://eaststemcell.com/ files/storage.cloud.php?id=MDIwMTQyMjg5MQ== (дата звернення: 12.01.2019).
169. Тимченко А. А. Основи системного проектування та системного аналізу складних об'єктів. Кн. 2. Основи системного підходу та системного аналізу об’єктів нової техніки. Київ: Либідь, 2004. 288 с.
170. Денисюк А. В., Кавицкая В. С., Любченко В. В. Определение адекватной модели для представления знаний в современных информационных системах. *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка.* 2014. № 61. 114–119
171. Любченко В. В. Модели знаний для предметных областей учебных курсов. *Искусственный интеллект.* 2008. №4. С. 458–462.
172. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. Москва, Сантк-Петербург: Издательский дом Питер. 2005. 730 с.
173. Лаврищева Е. М. Методы программирования. Теория, инженерия, практика. Киев: Наук. думка, 2006. 451 с.

174. Любченко В. В. Інформаційна технологія розробки та аналізу моделей предметних областей для їх вивчення. *Проблеми програмування.* 2012. № 2–3. С. 315–321.
175. Лаврищева Е. М. Интерфейс в программировании. *Проблеми програмування.* 2007. № 2. С. 126–139.
176. Лаврищева Е. М. Методы программирования. Теория, инженерия, практика. Киев: Наукова думка, 2006. 451с.
177. Лаврищева Е. М., Грищенко В. Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. Киев: Наукова думка, 2009. 372 с.
178. Андон П. І., Суслов В. Ю., Коротун Т. М., Коваль Г. І. та ін. Визначення витрат на створення ПЗ автоматизованих систем. *Проблемы программирования.* 1998. № 3. С. 23–34.
179. Баховец О. Б., Грінченко Т. О., Гуляєв К. Д. та ін. Передумови становлення інформаційного суспільства в Україні. Київ: Азимут України, 2008. 287 с.
180. Resource Estimation for Objectory Projects: project report. Objective Systems. SF AB, 1993. 9 p.
181. Chomsky N. Three Models for the Description of Language. IRE Transactions on Information Theory. Sep., 1956. P. 113–124.
182. Chomsky N. Logical Syntax and Semantics: Their Linguistic Relevance. Language. Vol. 31. No. 1. P. 36–45.
183. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра. Языки. Программирование. Киев: Наук. думка, 1974. 328 с.
184. Горелов С. Евристичний аналіз грамматики. Евристичний аналіз будь-якої мови. URL: http://www.grammcheck.org/ (дата звернення: 13.01.2019).
185. Федоренко О. Ф., Сухорольська С. М., Руда О. В. Основи лінгвістичних досліджень = Fundamentals of Linguistic Research: підруч. для вузів. Львів.: «Центр» Львів. нац. ун-ту ім. І. Франка, 2009. 296 с.
186. Руднев В. П. Генеративная лингвистика. Словарь культуры XX века. URL: http://www.gumer.info/bibliotek_Buks/Culture/Rudnev/Dict/_04.php (дата звернення: 13.01.2019).
187. What is Generative Grammar? wiseGEEK. URL: http://www.wisegeek.com/ what-is-generative-grammar.htm (дата звернення: 13.01.2019).

188. Потемкин А. В. Трехмерное твердотельное моделирование. Москва: Компьютер-Пресс, 2002. 296 с.
189. Жуковський В. В. Про деякі підходи до створення програмних комплексів комп'ютерного моделювання підземних процесів. *Вісник Кременчуцького національного університету імені Михайла Остроградського*. 2017. Вип. 2 (103). Ч. 1. С. 64–73.
190. Lavagno L., Martin G., Selic B.V. UML for Real: Design of Embedded Real-Time. Systems Kluwer Academic Publishers, 2003. 388 p. ISBN: 1402075014, 9781402075018.
191. Miles R., Hamilton K. Learning UML 2.0. O'Reilly Media, 2006. 288 p.
192. Иванов Д. Ю., Новиков Ф. А. Унифицированный язык моделирования UML. Учебное пособие. Санкт-Петербург: Изд-во Политехн. ун-та, 2010. 249 с.
193. Hay D. C. UML and Data Modeling: A Reconciliation Technics Publications, 2011. 242 p.
194. Karner G. Resource Estimation for Objectory Projects: project report. Objective Systems. San Francisco: AB, 1993. 9 p.
195. Anda B. Effort Estimation of Use Cases for Incremental Large-Scale Software Development. 27-th International Conference on Software Engineering, St. Louis, MO, 15 – 21 May 2005, P. 303–311.
196. Carroll E. R. Estimating Software Based on Use Case Point. OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, San Diego, 2005. P. 257–265. DOI: 10.1145/1094855.1094960
197. Clemmons R. Project Estimation with Use Case Points.*Cross Talk.*2016. Vol. 2, Iss. February. P. 18–22.
198. Дідковська М. В. Тестування: Критерії та методи. Київ: НТУУ «КПІ», 2010. 96 с.
199. Cohn M. Agile Estimating and Planning. New York City: Prentice Hall, 2005. 368 p.
200. Орлов С. А. Программная инженерия: уч. для вузов. 5-е изд. обновл. и доп. Санкт-Петербург: Питер, 2016. 640 с.
201. Reza Afshari A. R., Brtka V., Ćoćkalo-Hronjec M. Project risk management in Iranian software projects. *Journal of Engineering Management and Competitiveness (JEMC)*. 2018. Vol. 8. No. 2. P. 81–88.
202. Выгодский М. Я. Справочник по высшей математике. Москва: ГИТТЛ, 1957. 784 с.

203. O’Reily T. Open Source Paradigm Shift by Tim O’Reilly. URL : http://archive.oreilly.com/pub/a/oreilly/tim/ articles/paradigmshift_0504.html (дата звернення : 23.02.2019).
204. Робинсон С., Корнз О., Глинн Дж. C# для профессионалов (комплект из 2 книг). Москва: Лори, 2005. 1000 с.
205. Боггс У., Боггс М. UML и Rational Rose. Санкт-Петербург: Лори, 2008. 600 с.
206. Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. Москва: Символ-Плюс, 2011. 192 с.
207. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4. Москва: Вильямс, 2011. 1392 с.
208. Колесов Ю. Б., Сениченков Ю. Б. Моделирование систем. Объектно-ориентированный подход. Санкт-Петербург: БХВ-Петербург, 2006. 192 с.
209. Воройский Ф. С. Основы проектирования автоматизированных библиотечно-информационных систем. Санкт-Петербург: ФИЗМАТЛИТ, 2002. 384 с.
210. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. Москва: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2006. 320 с.
211. Воройский Ф. С. Информатика. Новый систематизированный толковый словарь. Москва: ФИЗМАТЛИТ, 2003. 760 с.
212. Новиков Ф. А., Иванов Д. Ю. Моделирование на UML. Теория, практика, видеокурс (+2 DVD-ROM). Москва: Профессиональная литература; Наука и техника, 2010. 640 с.
213. Путилин А. Б., Юрагов Е. А. Компонентное моделирование и программирование на языке UML. Практическое руководство по проектированию информационно-измерительных систем. Москва: НТ Пресс, 2005. 664 с.
214. Киммел П. UML. Универсальный язык программирования. Москва: НТ Пресс, 2008. 272 с.
215. Леоненков А. В. Самоучитель UML 2. Москва: БХВ-Петербург, 2007. 576 с.
216. Бабич А. В. UML. Первое знакомство. Пособие для подготовки к сдаче теста UMO-100 (OMG Certified UML Professional Fundamental)

(+CD-ROM). Санкт-Петербург: Бином. Лаборатория базовых знаний, 2008. 176 с.
217. Гросс К. C# 2008 и платформа .NET 3.5 Framework. Санкт-Петербург: Вильямс, 2009. 480 с.
218. Максимчук Р. А., Нейбург Э. Дж. UML для простых смертных. Москва: Лори, 2008. 304 с.
219. Дей Н., Мандел Л., Райман А. Eclipse. Платформа Web-инструментов. Санкт-Петербург: КУДИЦ-Пресс, 2008. 688 с.
220. Ларман К. Применение UML и шаблонов проектирования. 2-е изд.: пер. с англ. Москва: Издательский дом «Вильямс», 2004. 624 с.
221. Yang H. Advances In UML And XML-based Software Evolution. Idea Group Publishing, 2005. 362 p.
222. Weilkiens T., Oestereich B. UML 2 Certification Guide: Fundamental & Intermediate. Exams Morgan Kaufmann. The MK/OMG Press, 2006. 320 p.
223. Бабич А. В. Введение в UML. Москва: НОУ ИНТУИТ, 2016. 209 с.
224. Samek M. Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems Newnes: 2nd edition, 2008. 728 p.
225. Object Management Group. OMG Unified Modeling Language (OMG UML). Version 2.5 Object Management Group, 2013. 831 p.
226. Фляйшнер Г. Эйлеровы графы и смежные вопросы. Москва: Мир, 2002. 176 с.
227. Татт У. Теория графов. Москва: Мир, 1988. 424 с.
228. Емеличев В. А., Мельников О. И., Сарванов В. И. Лекции по теории графов. Изд. 2, испр. Москва: Наука, 2009. 392 с.
229. Eulero L. Solvtio Problematis Ad Geometiam Sitvs. Pertinentis. Avctore. URL: http://eulerarchive.maa.org//docs/originals/E053.pdf (дата звернення: 17.06.2019).
230. Paoletti T. Leonard Euler's Solution to the Konigsberg Bridge Problem. URL: http://mathdl.maa.org/convergence/1/?pa=content&sa=viewDocument&nodeId=1310&bodyId=1452 (дата звернення: 17.06.2019).
231. Харари Ф. Теория графов. Москва: Мир, 1973. 316 с.
232. Jungnickel D. Graphs, Networks and Algorithms. Fourth Edition. Berlin : Springer, 2013. 677 p. DOI: https://doi.org/10.1007/978-3-642-32278-5.
233. Салий В. Н., Богомолов А. М. Алгебраические основы теории дискретных систем. Москва: Физ.-мат. литература, 1997. 424 с.

234. Ловас Л., Пламмер М. Прикладные задачи теории графов. Москва: Мир, 1998. 656 с.
235. Кормен Т. М. Алгоритмы для работы с графами. Ч. VI. Алгоритмы: построение и анализ. 2-е изд. Москва: Вильямс, 2006. 1296 с.
236. Сик Дж., Ли Л., Ламсдэйн Э. C++ Boost Graph Library. Санкт-Петербург: Питер, 2006. 304 с.
237. Sedgewick R. Algorithms in Java, Third Edition, Part 5: Graph Algorithms. Boston : Addison Wesley, 2003. 528 p.
238. Кирсанов М. Н. Графы в Maple. Москва: Физматлит, 2007. 168 с.
239. Глушков В. М., Амосов Н. М., Артёменко И. А. Энциклопедия кибернетики. Т. 2. Киев: Главная редакция УСЭ, 1974. 624 с.
240. Коммерческие специализированные программы для построения графов. URL: http://www.boost.org/ (дата звернення: 02.04.2019)
241. LION Graph Visualizer. URL: http://lion.disi.unitn.it/intelligent-optimization/visualizer.html (дата звернення : 03.04.2019).
242. Графоанализатор – среда визуализации графов. URL : http://grafoanalizator.unick-soft.ru/ (дата звернення: 03.04.2019).
243. Ларман К. Применение UML 2.0 и шаблонов проектирования. 3-е изд. Москва: Вильямс, 2006. 736 с.
244. Шмуллер Дж. Освой самостоятельно UML 2.0 за 24 часа. Практическое руководство. Москва: Вильямс, 2005. 416 с.
245. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. 2-е изд. Москва: ДМК-Пресс, 2004. 432 с.
246. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд.; пер. с англ.; под общей редакцией О. С. Орлова. Санкт-Петербург: Питер, 2006. 736 с.
247. Шмуллер Дж. Освой самостоятельно UML 2.0 за 24 часа. Практическое руководство. Москва: Вильямс, 2005. 416 с.
248. Ларман К. Применение UML 2.0 и шаблонов проектирования. 3-е изд. Москва: Вильямс, 2006. 736 с.
249. Буч Г., Рамбо Дж., Джекобсон А.. Язык UML. Руководство пользователя. 2-е изд. Москва: ДМК Пресс, 2004. 432 с.
250. Пашеку Х. Программирование в Borland Delphi 2006 для профессионалов. Москва : Вильямс, 2006. 944 с.
251. Культин Н. Основы программирования в Delphi XE. Санкт-Петербург: БХВ-Петербург, 2011. 416 с.

252. Осипов Д. Базы данных и Delphi. Теория и практика. Санкт-Петербург: БХВ-Петербург, 2011. 752 с.
253. Вальвачев А. Н., Сурков К. А., Сурков Д. А. Программирование на языке Delphi: учеб. пос. Киев: НТУ, 2005. 436 с.

ДОДАТОК А
ПРИКЛАДИ СТВОРЕНОГО АНКЕТУВАННЯ КОРИСТУВАЧІВ

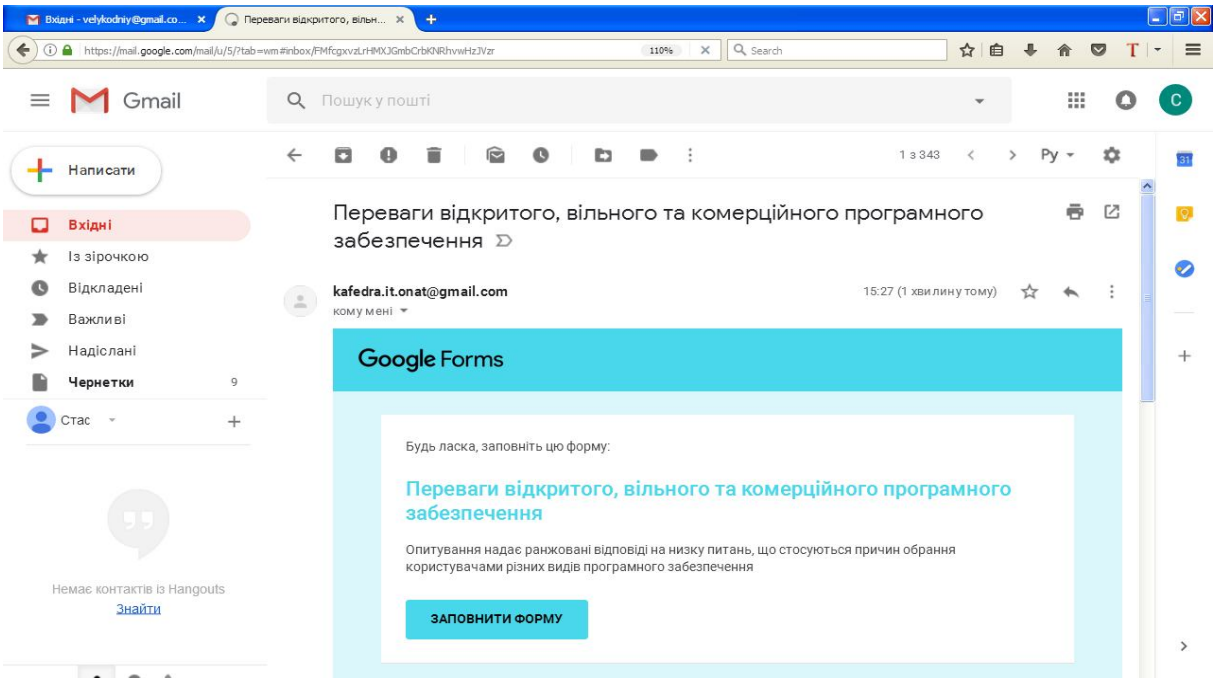


Рисунок А.1 – Приклад запрошення, що надсилається на пошту або роздається у вигляді відкритого посилання учаснику опитування

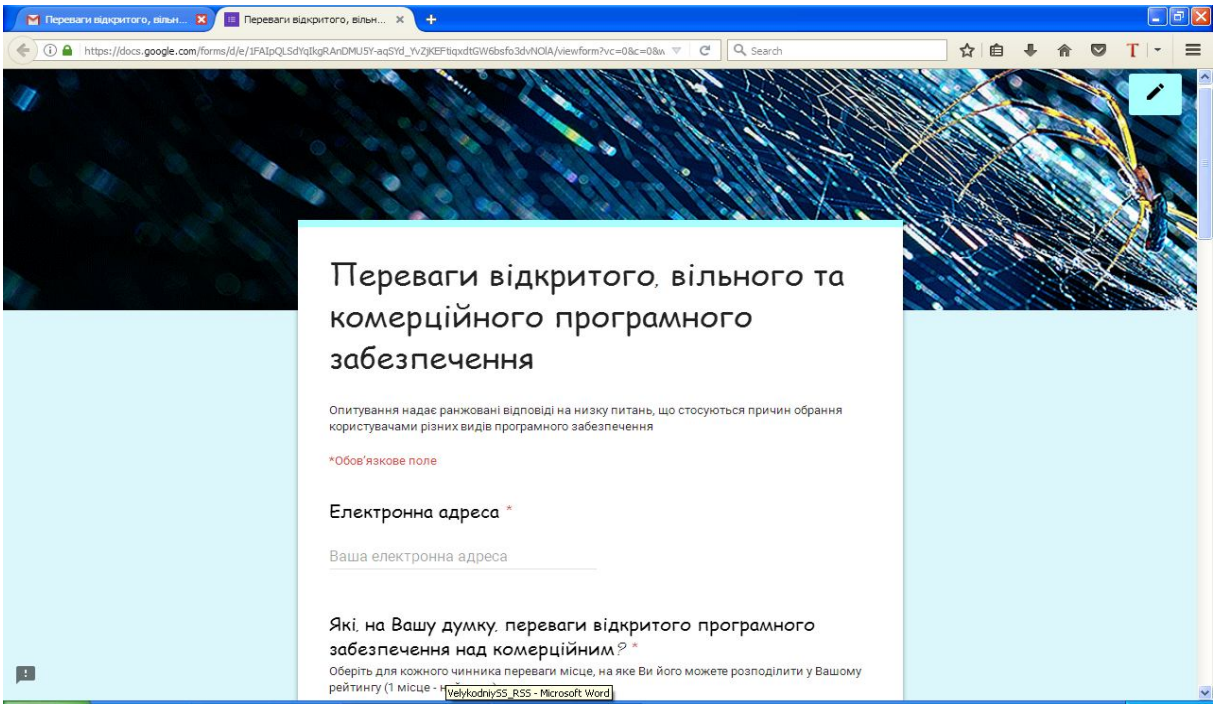


Рисунок А.2 – Початок анкети опитування

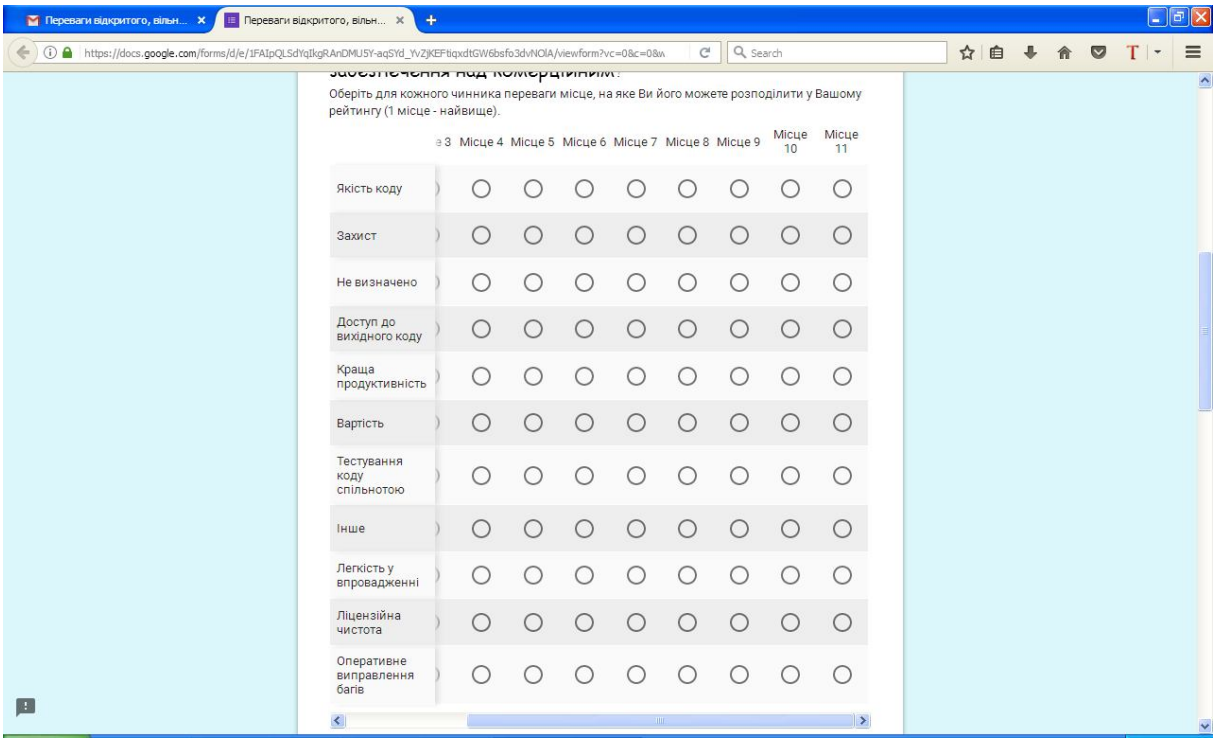


Рисунок А.3 – Фрагмент першої анкетної матриці, що містить 121 можливий варіант відповідей

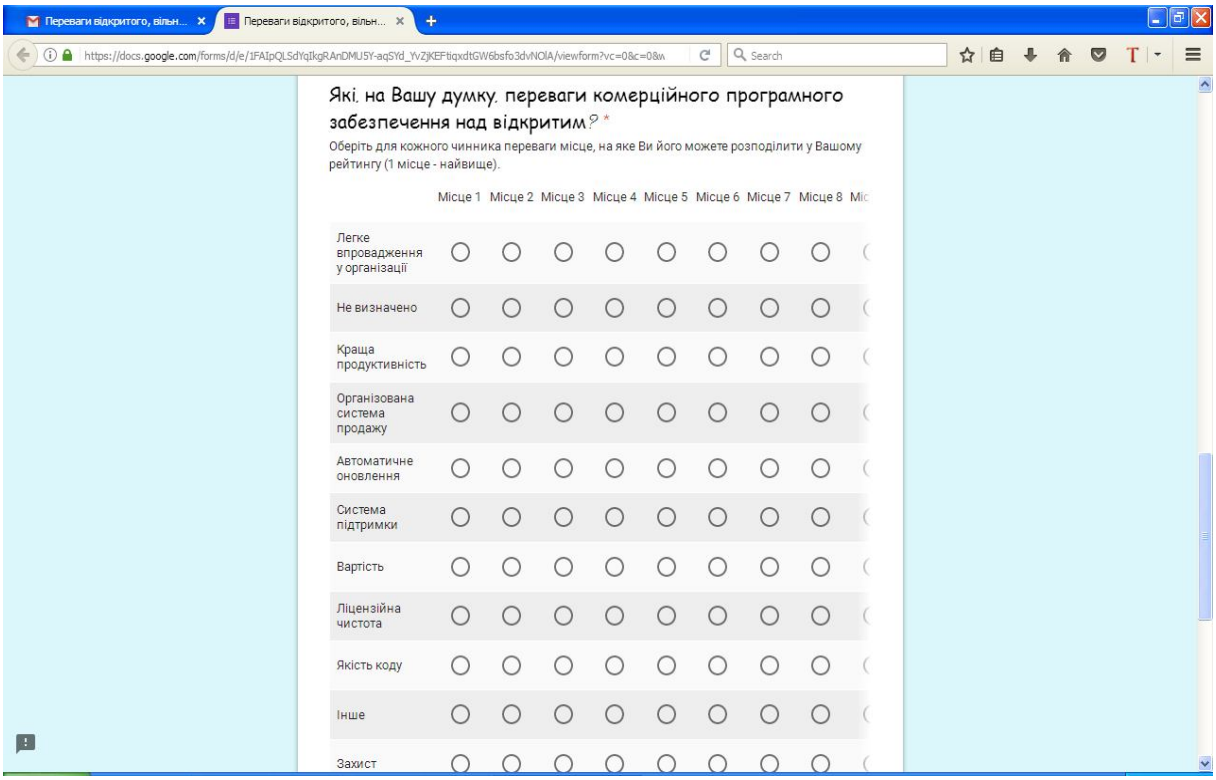


Рисунок А.4 – Фрагмент другої анкетної матриці, що також містить 121 можливий варіант відповідей

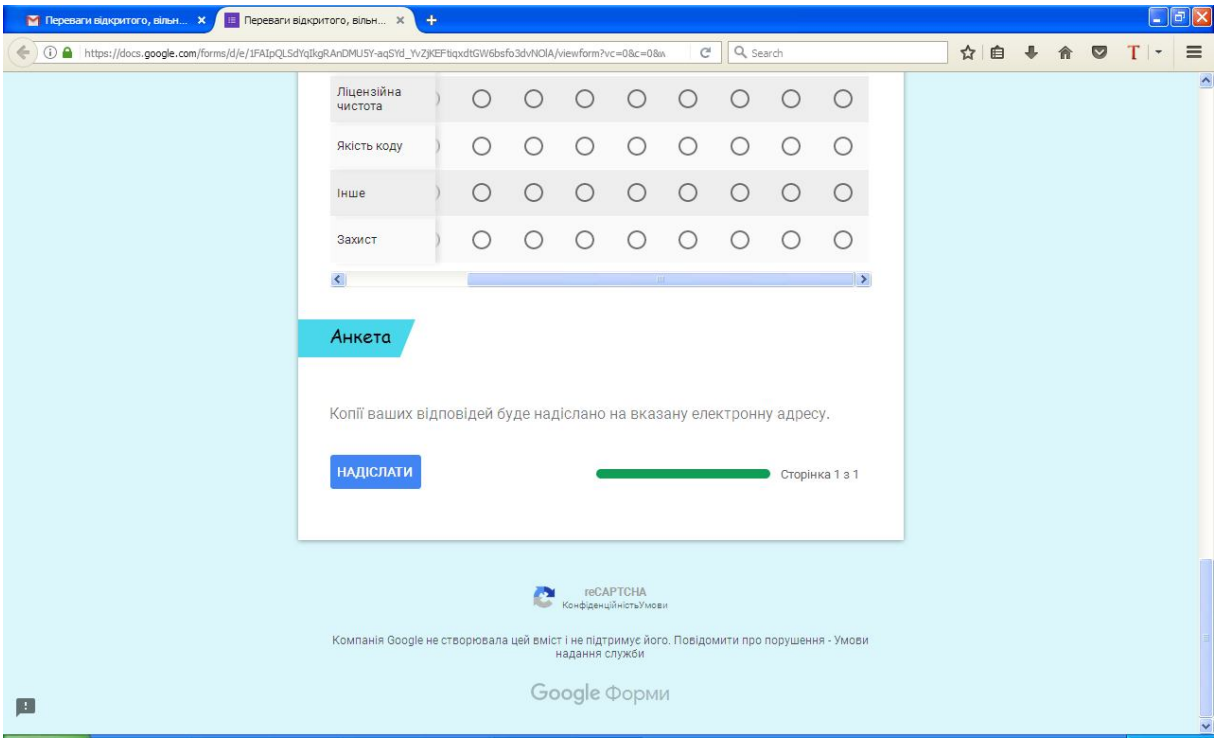


Рисунок А.5 – Завершення анкети опитування

ДОДАТОК Б
РЕІНЖИНІРИНГ ВІДКРИТОЇ ПРОГРАМНОЇ СИСТЕМИ ТРИВИМІРНОГО МОДЕЛЮВАННЯ BRL-CAD

Повний діаграмний комплекс, що виконано англійською мовою із дотриманням нотації UML 2.5, який адаптований під безпосередню розробку програмного засобу наведений на рис. Б.1 – Б.3.

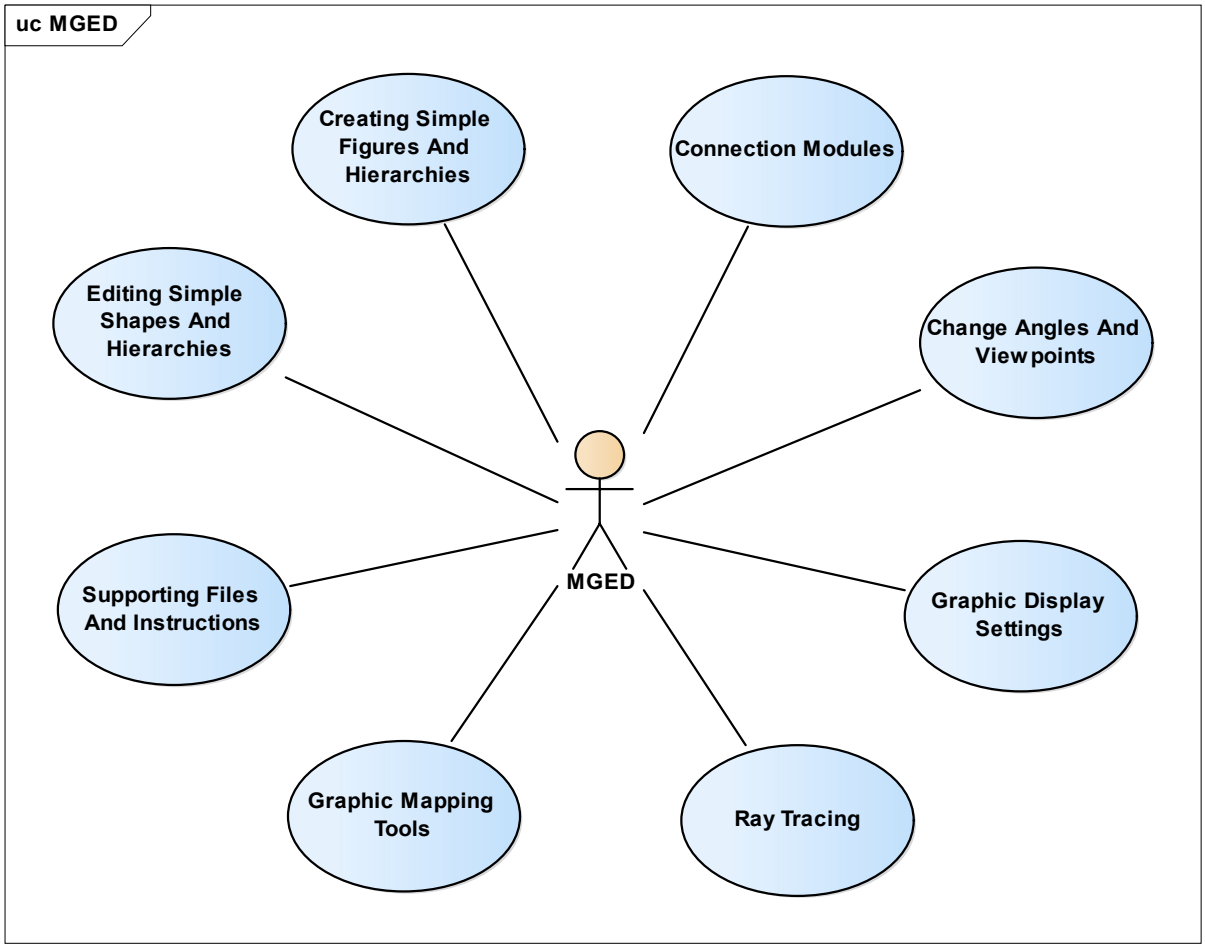


Рисунок Б.1 – Перший рівень прецедентів для "MGED"

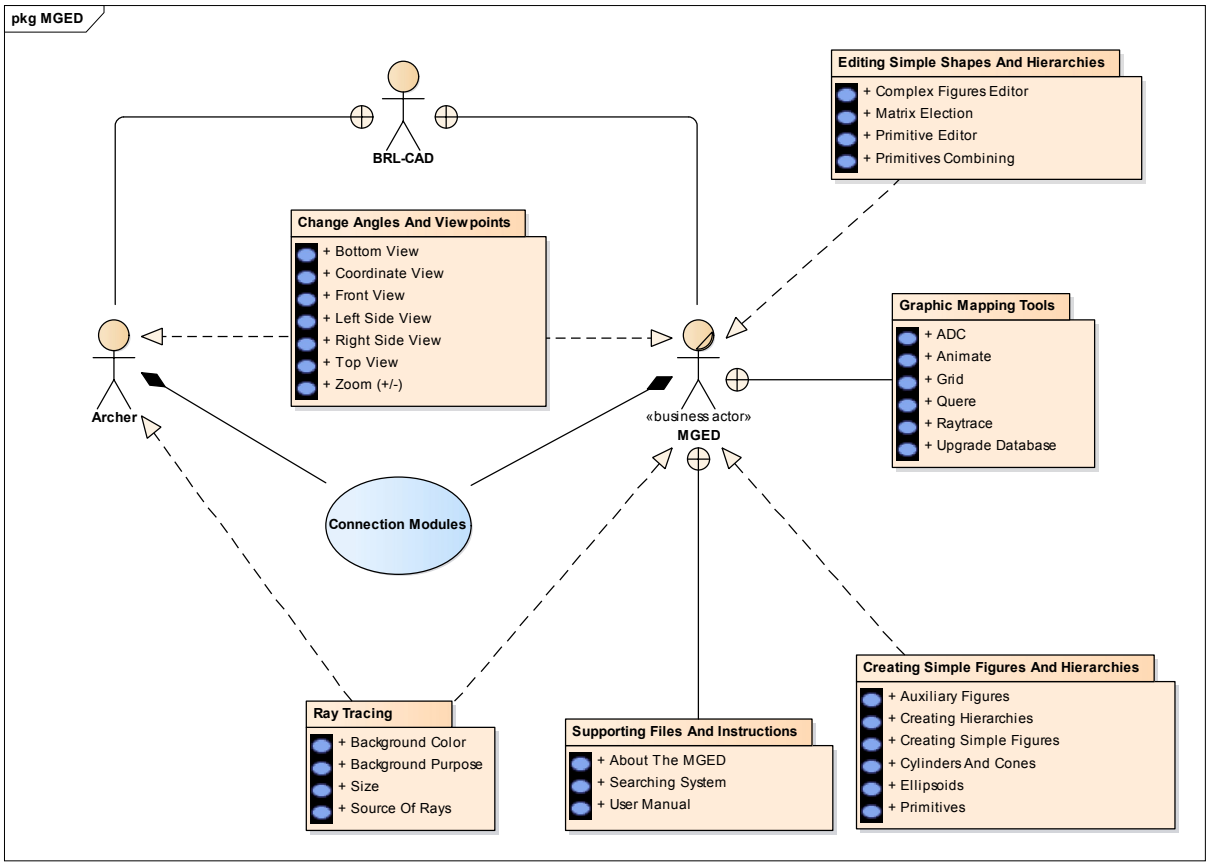


Рисунок Б.2 – Модель управління пакетами для MGED

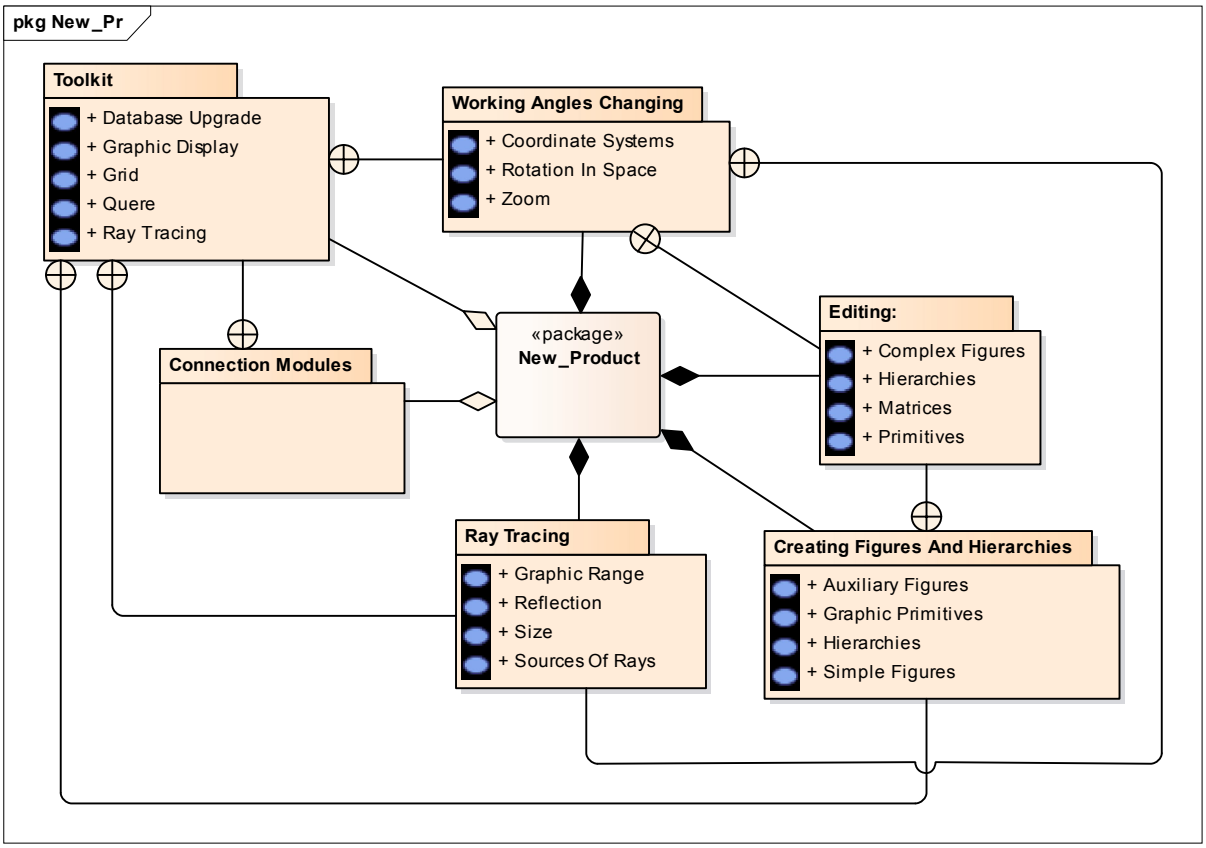


Рисунок Б.3 – Модель управління пакетами з ВВ оновленого продукту

ДОДАТОК В
ДІАГРАМНИЙ МОДЕЛЬНИЙ КОМПЛЕКС ПРОГРАМНОГО ЗАСОБУ ПЛАНУВАННЯ РЕІНЖИНІРИНГУ ПРОГРАМНИХ СИСТЕМ

Повний діаграмний комплекс, що виконано англійською мовою із дотриманням нотації UML 2.5, який адаптований під безпосередню розробку програмного засобу наведений на рис. В.1 – В.7.

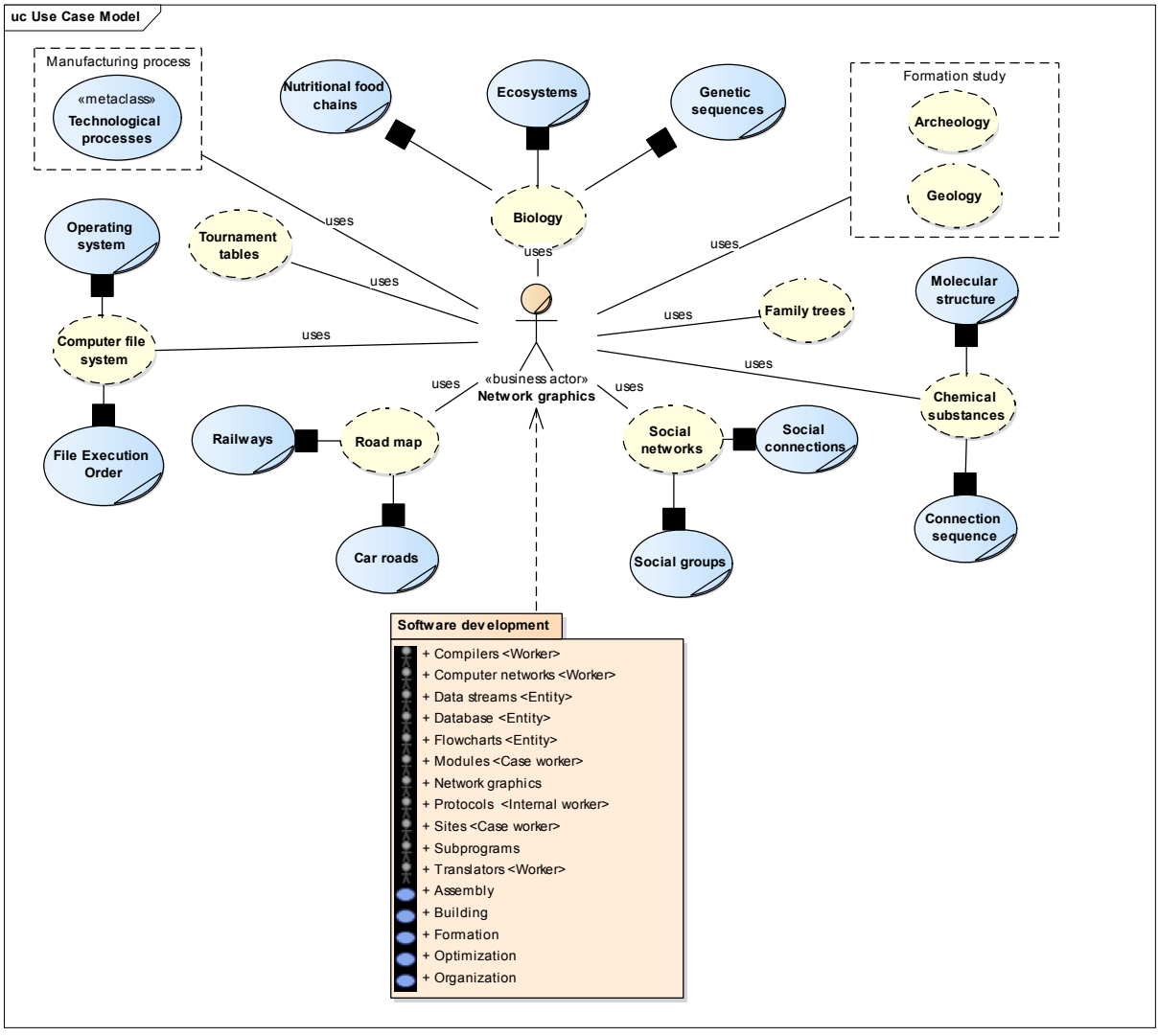


Рисунок В.1 – Модель варіантів використання (Use-Case Model)

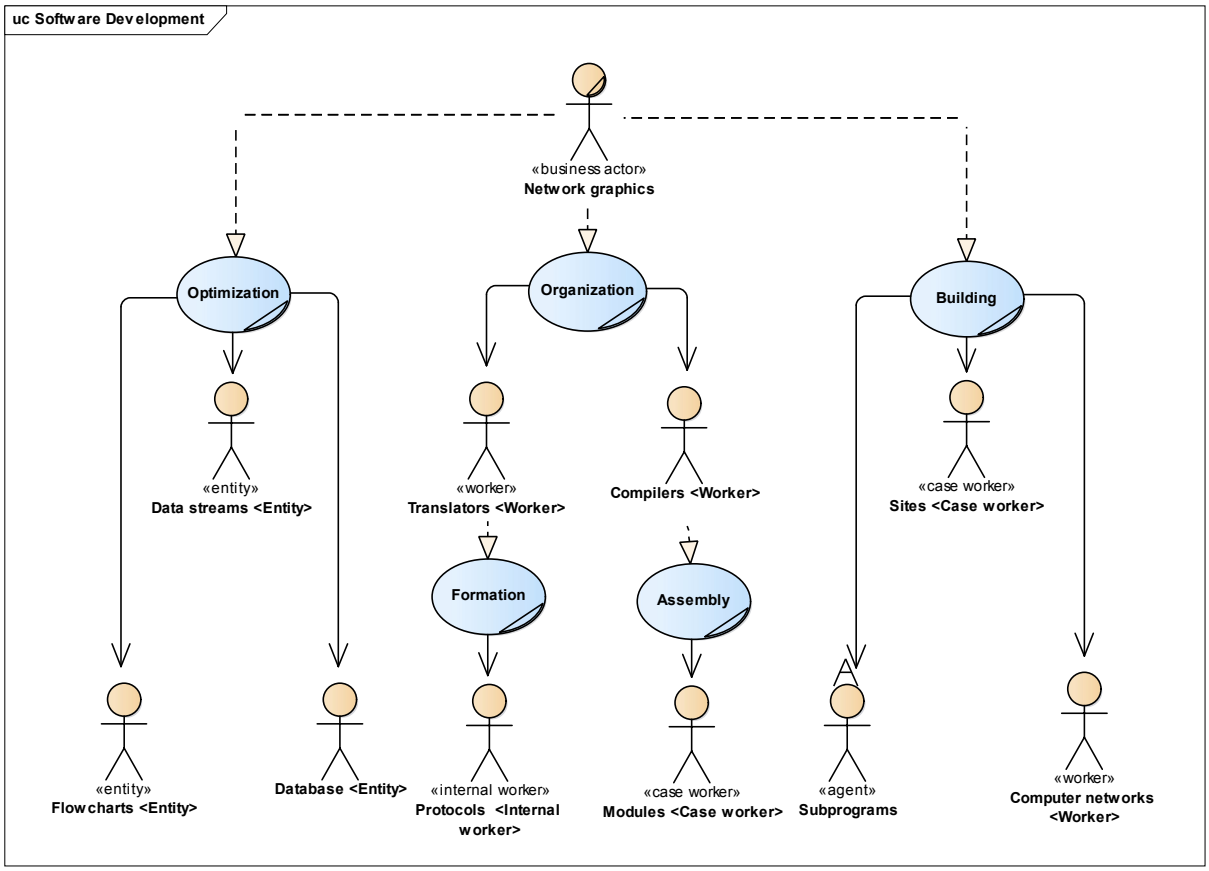


Рисунок В.2 – Модель управління пакетами «Розробка ПЗ» (Diagram-Package «Software Development»)

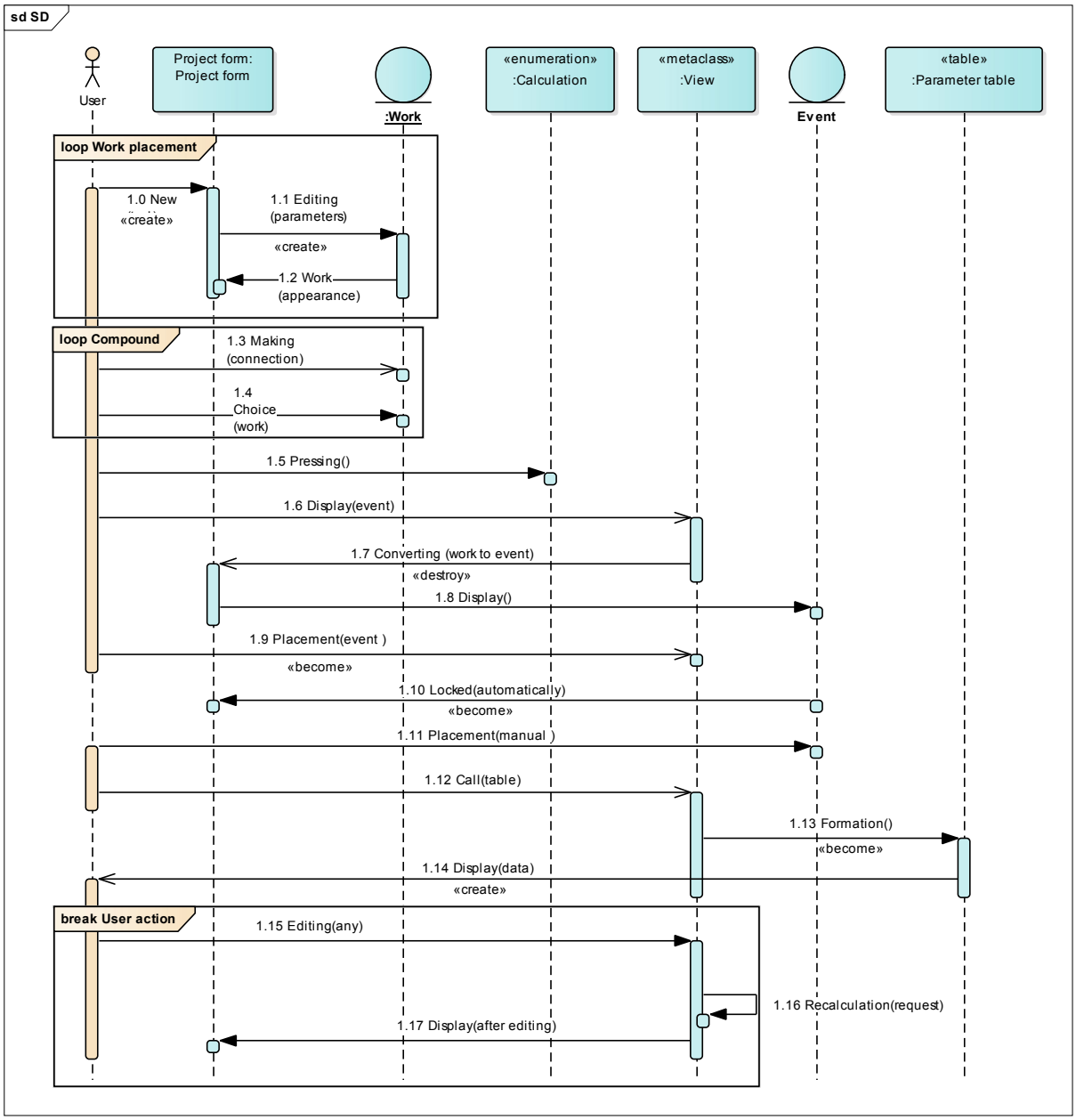


Рисунок В.3 – Модель послідовності розвитку проєкту (Software Sequence Model)

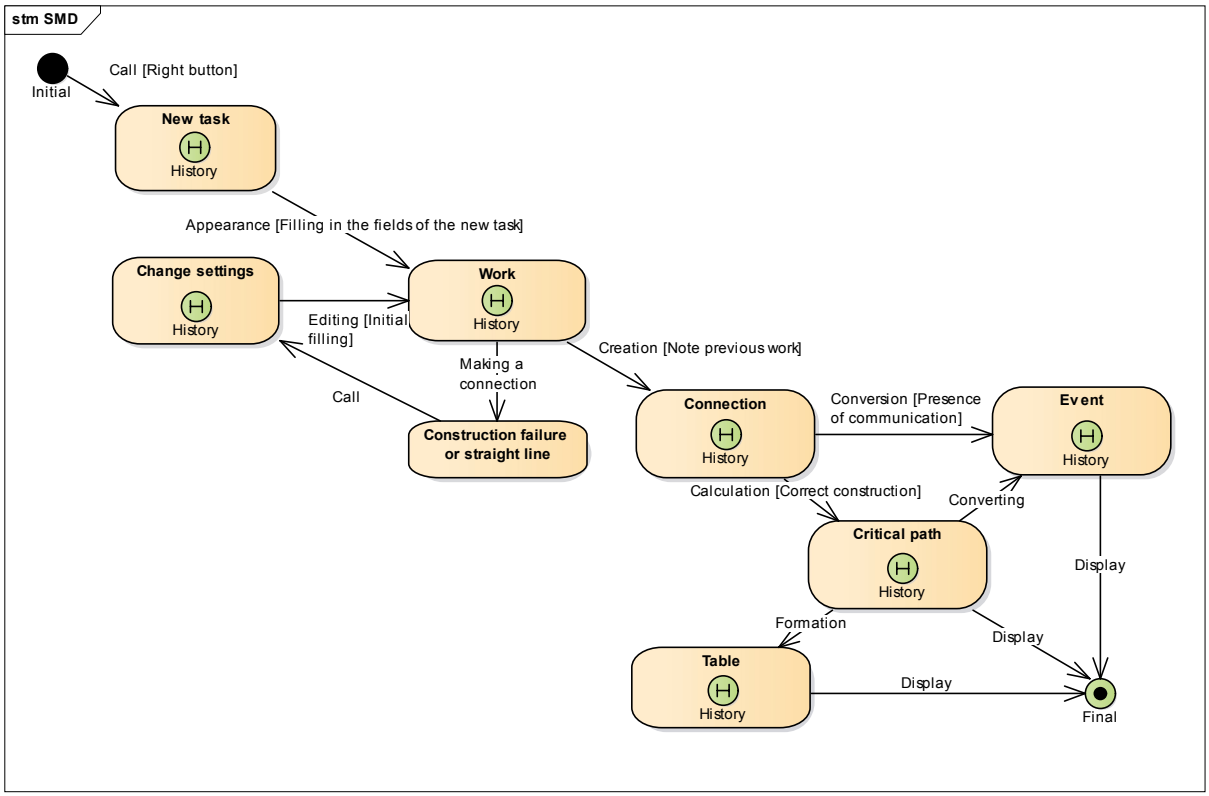


Рисунок В.4 – Модель опису станів для ПМК (State Machine Model for software)

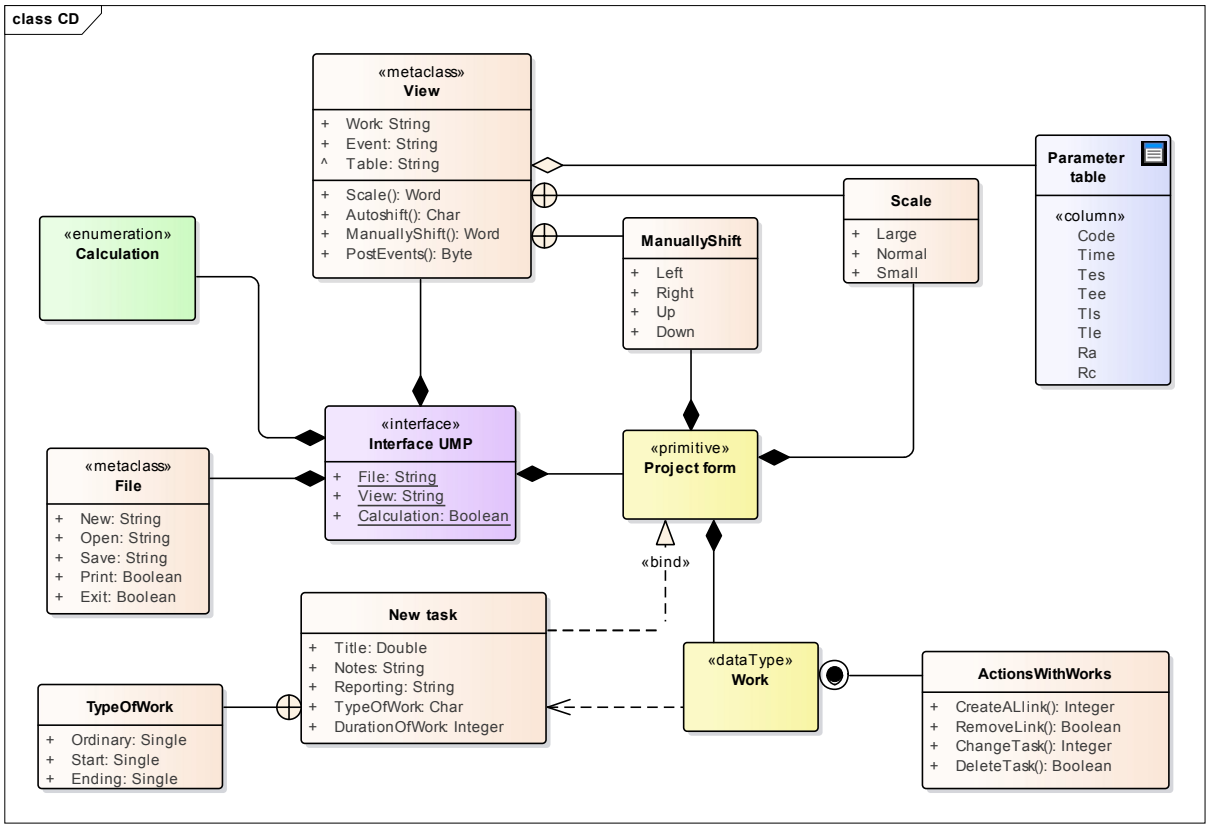


Рисунок В.5 – Модель опису проєктних класів ПМК (Class Model)

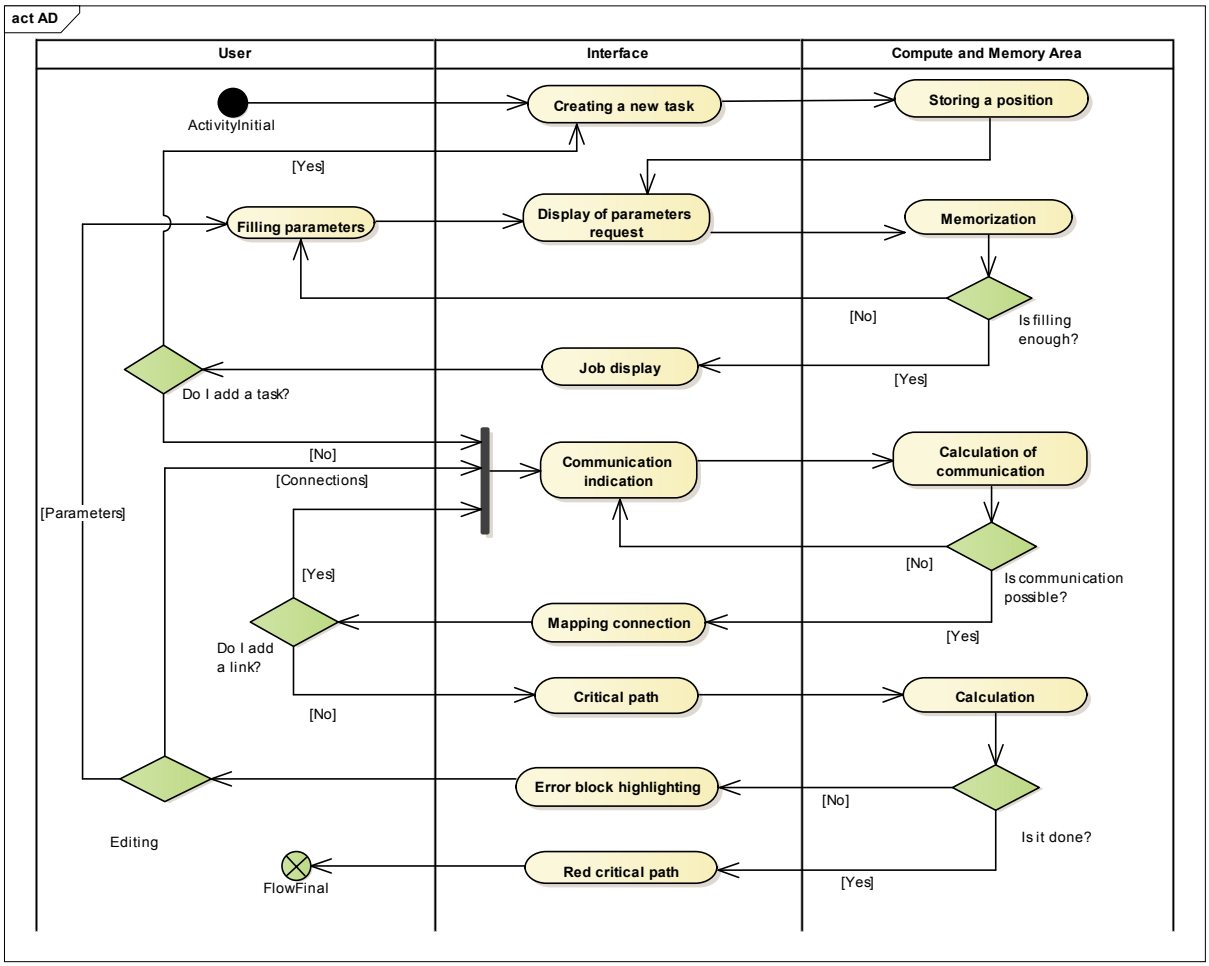


Рисунок В.6 – Динамічна модель діяльності головного інформаційного потоку (Activity Model of the main information flow)

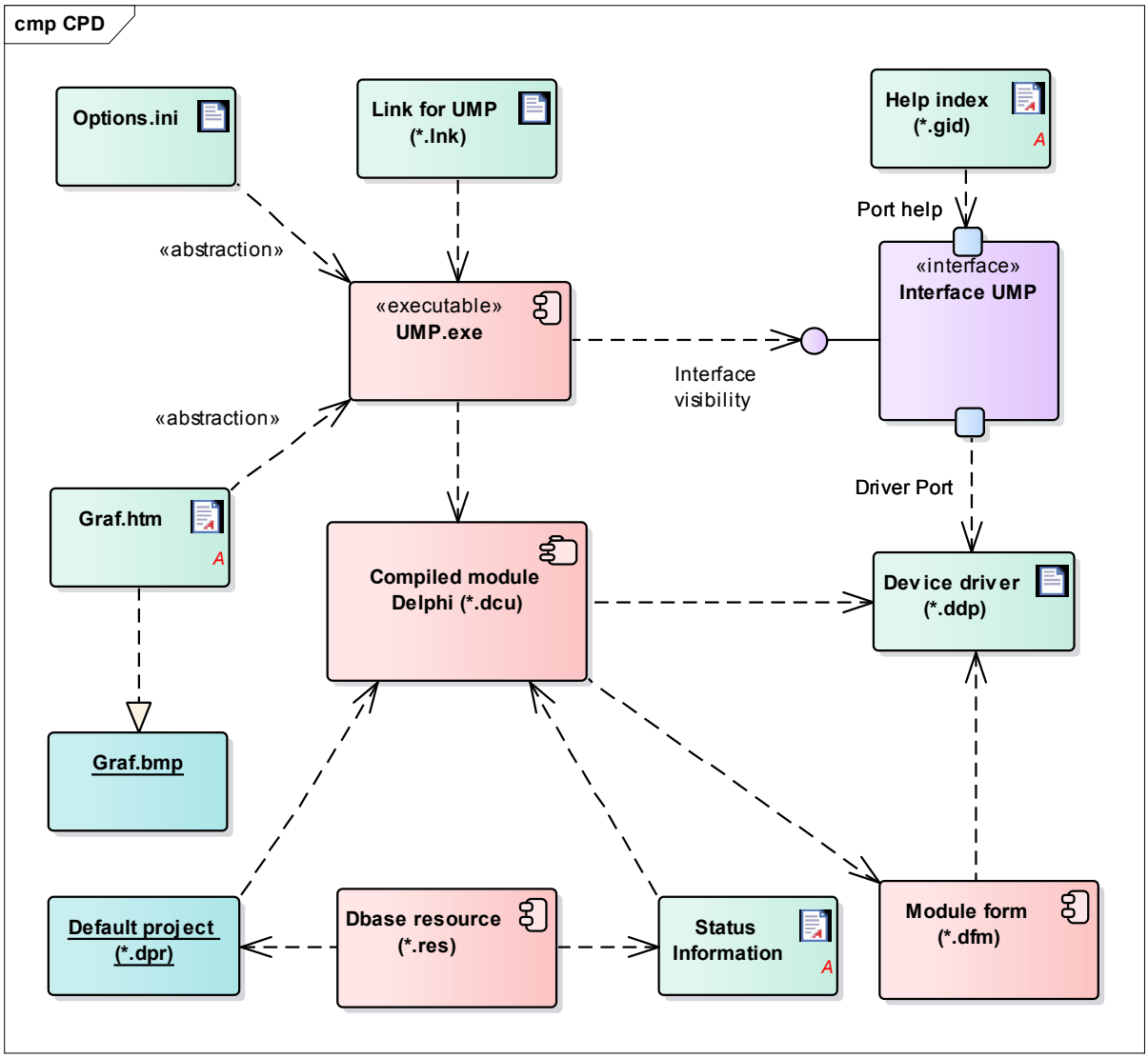


Рисунок В.7 – Модель компонентних рішень (Component Model)

АНОТАЦІЯ

Виконане дослідження розв'язує наукову проблему формування нових моделей та розробки нових методів проактивного управління проектами з розвитку життєвого циклу програмних продуктів, які забезпечують закладання методологічних основ проактивного управління проектами з реінжинірингу програмних систем, що дозволяє працювати з багаторівневими програмними системами, за допомогою комплексного інструментарію проектної оцінки.

В результаті проведення аналізу методологій управління проектами зі створення відкритих, вільних та комерційних програмних продуктів було з’ясовано, що проактивний розвиток програмних систем, на відміну від реактивного являє собою величезну науково-технічну проблему, а його впровадження вимагає значних капіталовкладень, оскільки необхідно управляти проектом з упередженням. За сучасними світовими тенденціями управління проектами, програмні продукти повинні бути такими, що розвиваються. Існує, принаймні, дві вагомі причини, за якими програмні продукти повинні змінюватись за часом. По-перше, розробка такого складного об'єкта як програмна система займає тривалий час та економічно вигідно вводити в експлуатацію частини системи за еволюційним механізмом: введений в експлуатацію базовий варіант – надалі розширюється. По-друге, постійний прогрес об'єктів проектування, технологій управління інформацією, обчислювальної техніки та обчислювальної математики зумовлює появу нових, більш досконалих математичних моделей і методів, які повинні замінювати старі менш вдалі аналоги систем управління проектами.

Дістала подальшого розвитку систематизація провідних методологій управління проектами, яка розширена формуванням концепції проактивного управління проектами з розвитку життєвого циклу програмних систем та продуктів, що дозволяє на етапі оцінки проекту визначити доцільність цільової програми реінжинірингу. Управління проектами реінжинірингу виконується за допомогою комплексу засобів, у тому числі за рахунок застосування компонентів повторного використання та CASE-систем. Згідно з оптимістичними прогнозами застосування компонентів повторного використання здатне у 4 рази знизити вартість програмної системи у порівнянні з новою розробкою.

Формалізовано методи проактивного управління проектами розвитку наскрізного проектування програмних систем, які відрізняються формуванням парадигми системної трансформації, що дозволяє інтегрувати програмні продукти у оновлені проектні комплекси. Згідно із загальною методологією управління проектами, задача проактивного розвитку полягає не тільки у з’ясуванні шляхів, а у визначенні характеру конкретної перебудови процесів проектування програмних систем, яка об'єднує у собі різні види забезпечення: технічне, математичне, програмне, інформаційне, лінгвістичне, методичне, організаційне, ергономічне та правове. Кожний з цих видів забезпечення має параметри та бажані характеристики управління, які обирають проектувальники із максимальним урахуванням особливостей завдань інженерного проектування, конструювання, технологій кодування, виготовлення та інтеграції програмних систем. Рішення цієї проблеми здійснюється шляхом управління проектами збирання, об'єднання або інтеграції різнорідних програмних ресурсів та компонентів повторного використання, що включають модулі, бібліотеки та програмні реалізації проекту деякої складної програмної системи.

Сформовано моделі проактивного управління розвитком лінгвістичного та інформаційного забезпечення проектів за допомогою динамічного оточення породжувальних граматик, що дозволяють позбавитися виведення тупикових ланцюжків при переведенні з однієї мови програмування до іншої. Досягнуті результати, з наукової точки зору, увійдуть до основ методології проактивного управління проектами з розвитку програмних систем та продуктів, а з практичної – стануть у нагоді системним програмістам, які задіяні у перепроектуванні й переробці програмних систем та продуктів, а також системним архітекторам, які працюють із мультимовними надбудовами програмних систем, що вже знаходяться у кількарічній експлуатації і набувають еволюційного розвитку із плином часу та удосконалення у процесі використання.

Удосконалено моделі проактивного управління розвитком поведінкової та структурної частин проектів з реінжинірингу програмних систем, які відрізняються уведенням оновлених структур уніфікованих діаграмних моделей, що дозволяє зберігати позитивні властивості відношень між проектними класами, компонентами та сутностями програмних систем незалежно від мови програмування. На основі управління системною архітектурою проекту складено моделі

проактивного розвитку графічних баз даних для композиційного підключення до оновленого відкритого проекту, а саме моделі: варіантів використання; опису станів для проактивного управління проектами; послідовності розвитку графічного представлення проекту; діяльності реінжинірингу графічної структури проекту; опису оточення об'єктів; операцій із проектними класами; компонентних рішень; управління інформаційним розгортанням проекту.

Удосконалено метод розрахунку показників проекту за проектними точками Карнера, який відрізняється внесенням доповнень та розширень щодо процесів реінжинірингу програмних систем, що дозволяє зменшити похибку у оцінюванні прогнозованого переліку витрат на реалізацію проекту на 17 – 19%, у порівнянні із методом Карнера. Встановлення коефіцієнтів та обрання значення констант засновані на методі Якобсона та перевірені статистикою найбільш схожих проектів. Після аналізу завершеного проекту і вивчення звіту про показники, доступні чинники можуть бути точно відкориговані, щоб дати оцінку часу, необхідного на виконання проекту реінжинірингу. Згодом, можна використовувати ці дані у якості базової траєкторії життєвого циклу проекту. Вимірювання виконуються командою досвідчених системних аналітиків, що спираються на власні уявлення про технічну складність проекту та можливості команди програмістів. Коефіцієнти та константи визначаються, виходячи із статистичних даних вже оцінених аналітиками виконаних проектів із близьким ступенем схожості.

Сформовані ідеалізовані моделі управління процесом оцінки проекту реінжинірингу, які відрізняються уведенням ресурсних та часових обмежень на виконання проекту, що дозволяє підвищити точність оцінки проектних витрат з перепроектування програмних систем. Сформовано представлення оцінки параметрів витрат ресурсів на виконання проекту реінжинірингу програмних систем за допомогою математичного інструментарію опису моделей проектування. Запропоновані ідеалізовані моделі управління проектом реінжинірингу видів забезпечення програмних систем являють собою еволюційні спіралі, які побудовані у циліндричній системі координат. Експериментальні дослідження спіральної ідеалізованої моделі управління проектом реінжинірингу за даними 7 виконаних проектів показали прогнозування витрат із максимальною похибкою 30%. Середнє значення відносної похибки – 19%. Операції із поданими моделями можуть відбуватися у проекціях часу та

витрат, у ізометричній проекції програмних компонентів, у логарифмічній проекції рядків програмного коду.

Розроблено метод представлення оцінки проекту з реінжинірингу програмних систем, який відрізняється застосуванням інструментарію проектних коефіцієнтів, що дозволяє управляти якісними змінами конфігурації графічних моделей перепроектування, у вигляді відповідних годографів проектів реінжинірингу програмних систем. Максимальне значення відносної похибки оцінки проекту реінжинірингу програмних систем за методом проектних коефіцієнтів за даними 12 виконаних проектів не перевищує 16%. Середнє значення відносної похибки – 8%. При використанні методу розрахунку показників проекту за проектними точками та методу представлення оцінки проекту з реінжинірингу програмних систем за допомогою інструментарію проектних коефіцієнтів, виконано оцінку кожного етапу проекту з удосконалення програмних систем. Результати перевірено у Одеському ім. А. Міцкевича відділенні Спілки поляків в Україні із отриманням економічного показника впровадження у вигляді оцінки вартості кожного окремого етапу проекту із удосконалення програмних систем, які не відрізняються від фактичних більш ніж на 7 %.

Узагальнено практично-орієнтовані результати комплексних експериментальних досліджень з проактивного управління проектами з розвитку програмних систем та сформовано рекомендації із розробки системної архітектури програмних продуктів, що планується розвивати. Розроблені рекомендації стануть у нагоді керівникам проектів, проектним менеджерам, системним архітекторам та інженерам-програмістам, які задіяні у перепроектуванні програмних систем та продуктів, що позитивно зарекомендували себе впродовж життєвого циклу їх експлуатації.

Розроблений програмно-методичний комплекс управління плануванням реінжинірингу проектів для використання командою проектного менеджменту у складі офісу управління проектами. Програмно-методичний комплекс містить у собі сукупність програмного, інформаційного, математичного, організаційного та методичного забезпечення для виконання робіт із планування та організації розвитку програмних систем та продуктів. При застосуванні у ТОВ «Люксстройпроект» (м. Харків) програмно-методичного комплексу стосовно до задач управління проектами, досягнуто скорочення строків проектування виробничого процесу за методологією мережевого

планування за рахунок прискорення складання мережевих графіків організації виробництва, внаслідок використання інтерактивних методів управління мережевою моделлю. Зафіксовано скорочення затраченого часу при проектуванні мережевих графіків у межах: 16% ÷ 24%; при трансформації робіт у події та навпаки: 87% ÷ 96%. При плануванні та управлінні життєвим циклом рекламної продукції ТОВ «Рекламне агентство «ТРАСТ МЕДІА», що займається її виготовленням, розміщенням, аналізом та просуванням, знайшли своє застосування розроблені методи та засоби управління мережевим плануванням проекту реінжинірингу програмних продуктів. Застосування програмно-методичного комплексу управління мережевим плануванням проекту реінжинірингу, як складової інформаційної технології із організації створення та просування рекламного продукту, скоротило: на 22% процес складання мережевого графіка у порівнянні із його складанням до застосування програмно-методичного комплексу; на 17% процес підрахунку часу, що закладається у базову траєкторію життєвого циклу проекту; на 8% процес аналізу ефективності рекламного продукту для прийняття рішення щодо його подальшого реінжинірингу та застосування конкретних методів розвитку й просування.

Запропоновані моделі та методи, а також програмно-методичний комплекс проактивного управління проектами з розвитку програмних систем та продуктів, впроваджено у науково-дослідну роботу та навчальний процес у: Одеському державному екологічному університеті, національному університеті «Одеська морська академія» та Одеській національній академії зв’язку ім. О. С. Попова, що підтверджено відповідними актами.

Ключові слова: управління проектами, проактивний розвиток, життєвий цикл, відкритий проект, програмна система, програмний продукт, архітектура проекту, модель управління, метод управління, реінжиніринг, комплексна оцінка проекту, вид забезпечення, системна трансформація, проектний коефіцієнт, програмно-методичний комплекс.

SUMMARY

The research solves the scientific problem of formation of new models and design of new methods of proactive project management in the development of software product lifecycle, which provide the laying of methodological bases of proactive project management for software systems reengineering, which makes it possible to work with multilevel software systems using complex project assessment tools.

As a result of the analysis of project management methodologies for creating open, free and commercial software products, it was found that the proactive development of software systems, unlike the reactive ones, is a huge scientific and technical problem, and its implementation requires considerable investment, since it is necessary to manage the project with prejudice. According to current global project management trends, software products need to be evolving. There are at least two major reasons why software products need to change over time. First, the development of such a complex object as a software system takes a long time and it is economically advantageous to put into operation parts of the system by an evolutionary mechanism: the basic variant put into operation is being expanded further. Second, the constant progress of design objects, information management technologies, computing, and computational mathematics leads to the emergence of new, more sophisticated mathematical models and methods that must replace older, less successful analogues of project management systems.

Systematization of leading project management methodologies has been further developed, it has been expanded by forming the concept of a proactive project management for the development of the life cycle of software systems and products, which makes it possible to determine the feasibility of a targeted re-engineering program at the project assessment stage. Reengineering projects are managed through a set of tools, including through the utilization of reuse components and CASE-systems. According to optimistic forecasts, the utilization of reuse components can reduce the cost of the software system 4 times compared to a new development.

Methods of the proactive project management of the software systems through design development have been formalized, they differ in the system transformation paradigm formation, making it possible to integrate software products into updated design complexes. According to the general project management methodology, the task of the proactive development is not only to

find out ways, but to determine the nature of a specific restructuring of software systems design processes, which combines different types of support: technical, mathematical, software, information, linguistic, methodological, organizational, ergonomic and legal. Each of these types of support has the parameters and desirable control characteristics that designers choose, with the maximum consideration of the peculiarities of the tasks of engineering design, construction, coding technologies, production and integration of software systems. The solution to this problem is realized by the project management of collecting, combining, or integrating heterogeneous software resources and reuse components, including modules, libraries, and program implementations of a complex software system.

Models of the proactive management of projects linguistic and information support development have been formed with the help of dynamic environment of generative grammars, making it possible to get rid of dead-end chains when translating from one programming language to another. The results achieved, from a scientific point of view, will become the basis for the methodology of proactive management of projects for the development of software systems and products, and from the practical point of view they will help system programmers who are involved in the design and processing of software systems and products, as well as system architects who work with multilingual add-ons of software systems that are already in operation for several years and are evolving over time and improving in the process of use.

Models for proactive management of the development of behavioral and structural parts of software systems reengineering projects have been improved, which are characterized by introduction of updated structures of unified diagram models, making it possible to maintain positive properties of relations between design classes, components and entities of software systems, regardless of the programming language. Based on the project system architecture management, the models have been compiled for the proactive development of graphical databases for compositional connection to the updated open project, namely the models of: application versions; state descriptions for the project proactive management; sequence of the project graphical representation development; activities of the project graphic structure re-engineering; the objects surroundings description; operations with project classes; component solutions; management of the project information deployment.

The method of calculating project indicators by Karner project points has been improved, which is characterized by the introduction of additions and

extensions to the processes of software system reengineering, which makes it possible to reduce the estimation error of the projected cost of the project implementation by 17 – 19%, compared with the Karner method. Setting coefficients and selecting constants are based on the Jacobson method and verified by statistics of the most similar projects. After analyzing the completed project and examining the performance report, the available factors can be adjusted accurately to estimate the time required to complete the reengineering project. Subsequently, this data can be used as a baseline trajectory for the project lifecycle. Measurements are performed by a team of experienced systems analysts who rely on their own understanding of the technical complexity of the project and the capabilities of the program team. The coefficients and constants are determined based on the statistics already estimated by the analysts of the completed projects with a similar degree of similarity.

Idealized models for managing the process of re-engineering project evaluation have been developed, which differ in the introduction of resource and time constraints on the project implementation, which makes it possible to increase accuracy estimation of the project cost for the software systems re-design. The representation has been formed for estimating the parameters of the resources consumption for the software systems re-engineering project implementation with the help of mathematical tools of the design models description. The proposed idealized project management models for software systems re-engineering represent evolutionary spirals constructed in the cylindrical coordinate system. Experimental studies of the spiral idealized re-engineering project management model based on 7 completed projects have shown cost forecasts with a maximum error of 30%. The average value of the relative error is 19%. Operations with the presented models can occur in time and cost projections, in isometric projection of software components, in logarithmic projection of lines of program code.

The method for presenting the software systems re-engineering project assessment has been developed, characterized by the use of the design coefficients tool, making it possible to manage the qualitative changes in the of the re-design graphical models configuration in the form of corresponding hodographs of the software systems re-engineering projects. The maximum value of the relative error of software systems re-engineering project assessment does not exceed 16% by the method of design coefficients according to the data of 12 completed projects. The average value of the relative error is 8%. When

using the method of calculating project indicators by project points and the method of presenting the software systems re-engineering project with the tool of project coefficients, the assessment of each stage of the project on software systems improvement was carried out. The results have been checked in A. Mickiewicz Branch of the Union of Poles in Ukraine, Odessa; the economic indicator of implementation in the form of assessment of the cost of each individual stage of the project to improve the software systems did not differ from the actual one more than 7%.

Practically oriented results of complex experimental researches on proactive project management for software systems development have been generalized and recommendations for elaborating software products system architecture, intended for development, have been formed. The developed recommendations will be useful for project managers, system architects and software engineers who are involved in re-designing software systems and products that have positively proven themselves throughout the life cycle of their operation.

The software-methodological complex of projects re-engineering planning management has been developed for use by the project management team as part of the project management office. The program-methodical complex contains a set of software, information, mathematical, organizational and methodological support for the execution of works on planning and organization of the development of software systems and products. When used in Luxstroyproekt LLC (Kharkiv), application of the software and methodological complex to project management tasks, ensured reduction in terms of production process design according to the network planning methodology at the cost of acceleration of network schedules of production organization due to the use of interactive methods of network model management. The reduction in time spent when designing network schedules was recorded within: 16% ÷ 24%; during the transformation of works into events and vice versa: 87% ÷ 96%. When planning and managing the lifecycle of advertising products, LLC TRUST MEDIA Advertising Agency, engaged in advertising products production, placement, analysis and promotion, has found application of the developed methods and means of managing network planning of the software re-engineering project. Application of the program-methodical complex of network planning management of the re-engineering project, as a component of information technology for the organization of creation and promotion of the advertising product, has reduced: the process of drawing up a network schedule by 22%

compared to its preparation before the application of the program-methodical complex; the process for calculating the time that is embedded in the baseline trajectory of the project lifecycle by 17%; the process of analyzing the effectiveness of an advertising product to decide on its further re-engineering and applying specific development and promotion methods by 8%.

The proposed models and methods, as well as the program and methodological complex of proactive project management for the development of software systems and products, were introduced into the research work and educational process at: Odessa State Ecological University, National University "Odessa Maritime Academy" and O. S. Popov Odessa National Academy of Communications, which is confirmed by the relevant acts.

Keywords: project management, proactive development, life cycle, open project, software system, software product, project architecture, management model, management method, re-engineering, complex project assessment, type of support, system transformation, project coefficient, software and methodological complex.

Наукове електронне видання

ВЕЛИКОДНИЙ Станіслав Сергійович

МОДЕЛІ ТА МЕТОДИ ПРОАКТИВНОГО УПРАВЛІННЯ ПРОЄКТАМИ З РОЗВИТКУ ПРОГРАМНИХ СИСТЕМ І ПРОДУКТІВ

Монографія

Видавець і виготовлювач:
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб'єкта видавничої справи
ДК № 5242 від 08.11.2016