НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____Наталія АУШЕВА

«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Розширення функціоналу CAD-системи для взаємодії з PDM-системою»

Виконав:
Студент IV курсу, групи ТР-81
Кузяк Назарій Іванович _____

Керівник:
доцент, кандидат технічних наук
Кублій Лариса Іванівна

_____

Рецензент:
професор Дрогобицького державного педагогічного університету імені Івана Франка,
доктор фізико-математичних наук, Столярчук І. Д _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2022 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність: 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ
на дипломну роботу студенту
_____Кузяк Назарій Іванович_____
(прізвище, ім’я, по батькові)

1. Тема роботи: Розширення функціоналу CAD-системи для взаємодії з PDM-системою_____

керівник роботи: Кублій Лариса Іванівна, кандидат технічних наук, доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “8” червня 2022 р. № 965-с.

2. Строк подання студентом роботи: 10 червня 2022 р.
3. Вихідні дані до роботи: мова програмування C#, середовище розробки Microsoft Visual Studio
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): створити розширення за допомогою якого CAD-система зможе взаємодіяти з PDM-системою; розглянути варіанти інтеграції з різними CAD-системами; описати спосіб роботи користувача з розробленою системою.
5. Перелік ілюстративного матеріалу: «Системи автоматизованого проектування», «Система керування даними виробу», «Засоби розширення функціоналу CAD-

систем», «Засоби розробки», «Методи роботи користувача з розширенням CAD-системи для взаємодії з PDM-системою», «Висновки».

6. Дата видачі завдання «10» вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25 травня 2022 р.	
2.	Вивчення та аналіз задачі	02-04 травня 2022 р.	
3.	Розробка архітектури та загальної структури системи	05-06 травня 2022 р.	
4.	Розробка структур окремих підсистем	07-09 травня 2022 р.	
5.	Програмна реалізація системи	10-27 травня 2022 р.	
6.	Оформлення пояснювальної записки	28 травня 2022 р.	
7.	Захист програмного продукту	29 травня 2022 р.	
8.	Передзахист	06 червня2022 р.	
9.	Захист	20 червня 2022 р.	

Студент _____ _____Кузяк Н. І._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Кублій Л. І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи — створення програмного забезпечення, яке розширить функціонал CAD-системи шляхом додавання нових команд в панелі інструментів системи автоматизованого проектування. Створене розширення повинно забезпечити взаємодію з PDM-системою.

Для написання програмного забезпечення обрано високорівневу мову програмування C#; в якості середовища розробки — Microsoft Visual Studio; для створення інсталятора — розширення середовища розробки Microsoft Visual Studio Installer Projects.

Дипломна записка складається із вступу, 5 розділів і висновків. Записка містить 84 сторінки, 37 рисунків, 6 таблиць і 24 посилання.

Ключові слова: CAD-система, PDM-система, API, інтеграція.

ABSTRACT

The purpose of the work is to create software that will expand the functionality of the CAD system by adding new commands in the toolbar of the computer-aided design system. The created extension should provide interaction with the PDM system.

A high-level C # programming language was chosen for writing the software; as a development environment - Microsoft Visual Studio; to create an installer — expand the development environment of Microsoft Visual Studio Installer Projects.

The diploma note consists of an introduction, 5 chapters and conclusions. The note contains 84 pages, 37 figures, 6 tables and 24 references.

Keywords: CAD system, PDM system, API, integration.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CAD — система автоматизованого проектування (англ. Computer Aided Design)

PDM — керування даними про продукт (англ. Product Data Management)

САП — система автоматизованого проектування

САПР — система автоматизованого проектування і розрахунку

АСП — автоматизована система проектування

MCAD — механічне автоматизоване проектування (англ. Mechanical Computer Aided Design)

EDA — автоматизація електронного проектування (англ. Electronic Design Automation)

AEC CAD — колекція архітектури, інженерії та будівництва (англ. Architecture, Engineering and Construction Collection)

CALS — безперервна інформаційна підтримка життєвого циклу продукції (англ. Continuous Acquisition and Life Cycle Support)

MCAE — механічна обчислювальна техніка (англ. Mechanical Computer Aided Engineering)

MBD — визначення на основі моделі (англ. Model Based Definitions)

BIM — інформаційне моделювання будівель (англ. Building Information Model)

АС — автоматизована система

EDM — керування інженерними даними (англ. Engineering Data Management)

PIM — керування інформацією про виріб (англ. Product Information Management)

TDM — керування технічними даними (англ. Technical Data Management)

TIM — керування технічною інформацією (англ. Technical Information Management)

API — прикладний програмний інтерфейс (англ. Application Programming Interface)

COM — компонентна об’єктна модель (англ. Component Object Model)

ПЗ — програмне забезпечення

CLR — загальномовне виконуюче середовище (англ. Common Language Runtime)

CTS — система загальних типів (англ. Common Types System)

CLS — специфікація загальної мови (англ. Common Language Specification)

REST — передача репрезентативного стану (англ. Representational State Transfer)

WCF — платформа для створення додатків, орієнтованих на службу (англ. Windows Communication Foundation)

WPF — платформа користувацького інтерфейсу для створення клієнтських додатків для настільних систем (англ. Windows Presentation Foundation)

GUI — графічний інтерфейс користувача (англ. Graphical User Interface)

WinForms — платформа для створення додатків робочого столу в операційній системі Windows

ВСТУП

У сучасному світі темпи виготовлення нових продуктів постійно зростають. Постійно збільшується попит на різні типи товарів, через що потрібно збільшувати пропозицію. Реалії сьогодення вимагають постійно переглядати ринок і негайно змінювати продукцію під потреби користувача.

Керівництво компаній зацікавлене в максимально ефективному використанні програмного забезпечення для проектування нової продукції. Для збільшення швидкості виготовлення окремих компонентів продукту відбувається автоматизація процесу виробництва. Здійснюється заміна старого обладнання на нове і використання новітніх технології для досягнення поставленої мети.

Розробка нової продукції включає в себе багато етапів, один з яких проектування нового продукту. Проектування на виробництві здійснюється колективно. Різні підрозділи здійснюють проектування різних частин виробу за допомогою систем автоматизованого проектування CAD.

Кожен етап життєвого циклу продукції супроводжується великою кількістю конструкторської і технічної документації. Під час проектування кожної частини виробу створюється велика кількість документів. PDM-системи дають можливість керувати даними про виріб на підприємстві.

У дипломній роботі створено програмне забезпечення, здатне розширити функціонал CAD-системи для взаємодії з PDM-системою. Для прикладу наведено результати розширення CAD-системи CATIA, щоб взаємодіяти з PDM-системою IT-Enterprise.

Отриманий функціонал дає можливість контролювати документацію виготовлення виробу на підприємстві.

1 ЗАДАЧА РОЗШИРЕННЯ ФУНКЦІОНАЛУ CAD-СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З PDM-СИСТЕМОЮ

Проектування нових виробів для впровадження їх у виробництво пов’язане з взаємодією між CAD-системою та PDM-системою. Кожного разу, коли користувач змінює деталь в CAD-системі, йому потрібно змінити дані про деталь в PDM-системі. Через людський фактор не можна бути впевненим, що всі дані будуть заповнені правильно і вчасно. Для полегшення роботи користувача доцільно створити розширення, яке зчитує дані з сесії CAD-системи, передає їх в PDM-систему, де правильно заповнює інформацію про виріб.

1.1 Постановка задачі

Метою даної роботи є розробка засобів інтеграції між CAD-системою та PDM-системою.

Для системи автоматизованого проектування треба створити новий функціонал, який реєструється в CAD-системі як команди, і відповідні піктограми розміщуються на панелі інструментів. Для взаємодії з CAD-системою треба розробити прикладний програмний інтерфейс.

Щоб розширити PDM-систему, необхідно написати новий програмний інтерфейс, який за допомогою команд дасть можливість взаємодіяти з системою керування даними про виріб.

Основними завданнями, які необхідно виконати для реалізації системи, є:

— дослідити способи розширення функціоналу різних CAD-систем;

— створити програмне забезпечення, яке дасть можливість взаємодіяти з PDM-системою;

— створити програмне забезпечення, яке дасть можливість взаємодіяти з CAD-системою;

— описати процес встановлення нового програмного забезпечення.

Реалізована система повинна забезпечувати такі функції:

— відкриття документа з PDM-системи в CAD-системі;

— проектування нового виробу за шаблоном, вибраним з PDM-системи;

— взяття документа на зміну користувачем;

— збереження проекту виробу з CAD-системи;

— повернення документа в PDM-систему.

1.2 Проблеми реалізації

Першою задачею, яку необхідно виконати, це знайти спосіб інтеграції з CAD-системою. Найпоширеніший спосіб досягти бажаного є використання API, прикладного програмного інтерфейсу. Як правило бібліотеки коду, для взаємодії з програмним забезпеченням, поставляються разом з самим програмним забезпеченням, а документація розміщується в директорії разом з бібліотеками або перебуває в вільному доступі в мережі.

Мова програмування C# і середовище розробки Microsoft Visual Studio мають весь функціонал, необхідний для підключення сторонніх бібліотек коду.

CAD-система CATIA реєструє бібліотеки коду прикладного програмного інтерфейсу під час встановлення програмного забезпечення CATIA. Документацію можна отримати в вікні довідки системи автоматизованого проектування.

Наступною задачею є створення програмного інтерфейсу PDM-системи, який забезпечить виконання необхідних функцій. Прикладне програмне забезпечення (API) буде виконувати функції обробника команд, які приходять з CAD-системи. Система

керування даними про виріб повинна бути “гнучкою” і давати можливість створювати розширення для неї.

Система керування даними про виріб IT-Enterprise має “гнучкий” функціонал і дає можливість створювати користувацькі бібліотеки коду, за допомогою яких можна реалізувати API.

Основною мовою програмування в системі є C#. Для керування базами даних використовується декларативна мова програмування SQL. Дана система не створювалася для взаємодії з однією PDM-системою, тому вона підходить для виконання поставленої задачі.

Висновки до розділу 1

У розділі сформульовано мету, визначено кроки виконання і необхідний функціонал для розв’язання поставленої задачі. Вказано проблеми даної задачі і способи їхнього розв’язання. В якості CAD-системи було обрано систему автоматизованого проектування CATIA. В якості PDM-системи було обрано систему керування даними про виріб IT-Enterprise. Визначено способи розширення систем.

2 СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ І КЕРУВАННЯ ДАНИМИ ПРО ВИРІБ

Система CAD, САП, або САПР, або АСП — автоматизована система, призначена для автоматизації технологічного процесу проектування виробу, результатом якого є комплект проектно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об’єкта проектування [1].

Система PDM — це система керування даними виробу. Під «даними» мається на увазі вся інформація про використання виробу — проектні дані, технологічні маршрути, результати технічних випробувань, дані про партії та окремі примірники і багато інших документів. Під «виробом» мається на увазі, як правило, якась високотехнологічна продукція (автомобілі, кораблі, літаки), при проектуванні, виробництві, експлуатації та утилізації яких необхідно обробляти і контролювати великі обсяги інженерно-технічних даних [2].

2.1 Етапи розробки виробів на підприємстві

Життєвий цикл продукту — це послідовність етапів у житті продукту. До основних етапів можна віднести: аналіз прогнозованих потреб у виробах, пошук ідеї нового товару, попередня оцінка ідеї і вибір найбільш прийнятного шляху її здійснення, дослідження споживчих властивостей нового продукту і попередній аналіз ринку, вибір критеріїв проектованого виробу відповідно до вимог ринку, визначення необхідних властивостей виробу на основі вибору альтернатив проектних характеристик, вивчення особливостей процесу виробництва і можливості адаптації нового продукту до існуючих умов підприємства, проектування нового продукту, проектування процесу виробництва, організація

дослідного виробництва і пробного збуту, перехід до серійного виробництва і здійснення комплексної програми маркетингу [3].

Основні етапи життєвого циклу процесу створення нового продукту відображено на рисунку 2.1.

Рисунок 2.1 — Життєвий цикл процесу створення нового продукту

На виникнення потреб, як правило, впливають наукові дослідження, рішення центральних органів, думка споживачів, продукція конкуруючих фірм тощо. На етапі аналізу потреб важливо враховувати перспективні потреби користувачів.

Основним джерелом для пошуку нових ідей є опитування існуючих і потенційних клієнтів, дослідження ринків, думки фахівців винахідників, працівників-патентознавців.

На етапі попередньої оцінки ідеї найперше відсіюються найбільш непридатні до впровадження проекти. З усіх ідей, які залишилися відбираються найбільш вдалі та багатообіцяючі.

Етап дослідження споживчих властивостей нового продукту виконується для проведення дослідження технічного та економічного характеру, також проводиться дослідження ринку.

Під час етапу вибору критеріїв проектованого виробу проектувальник повинен розглянути значимість: вартості, якості, дизайн, термін служби та інші.

Далі проектувальник враховує розміри, форми виробу, матеріали, модульні компоненти, надлишкові компоненти для підвищення надійності виробу, елементи безпеки.

На етапі вивчення особливостей процесу виробництва і можливості адаптації нового продукту до існуючих умов підприємства аналізується стан технічної бази підприємства, її готовність виробляти новий продукт. Чим менше змін в процесі вдосконалення підприємства, тим менші витрати на освоєння виробництва нового продукту.

На наступному етапі відбувається підготовка завдань на проектування, виготовлення і випробування функціональних зразків і прототипів виробів.

Далі відбувається проектування повного циклу виробництва виробу на підприємстві, враховується час, необхідний на вироблення кожного компонента виробу.

На етапі організації дослідного виробництва і пробного збуту, маючи достатню кількість товару, фахівці з організації збуту проводять дослідження серед найближчих клієнтів. Даний етап дає можливість досить точно визначити фактичний рівень витрат виробництва, встановити ціну і намітити шляхи та способи просування товару.

На останньому етапі здійснюється передача проекту в серійне виробництво та здійснюється комплексна програма його збуту, закінчується комплексна підготовка до випуску новинки на ринок у повному обсязі (створення упаковки, тари, розробка реклами, підготовка документації, технічних інструкцій тощо).

2.2 Призначення CAD-систем

Системи CAD призначені для автоматизації проектування, конструювання та виготовлення продукції. Такі системи дають можливість проектувати 2D та 3D об’єкти. Для використання даного функціоналу не потрібно мати навички у сфері програмування. Сучасні CAD-системи дають можливість створювати складні об’єкти за допомогою елементарних графічних примітивів, таких як коло, лінії, дуги, текст. Також формування складних об’єктів здійснюється за рахунок функцій видавлювання, витягування та переміщення.

Більшість CAD-систем дають можливість відкривати кілька деталей в одній сесії програми. Це здійснюється за рахунок того, що в одній сесії можна створити кілька вікон і прив’язати деталь до певного вікна.

Функціонал CAD-системи надає можливість об’єднувати прості деталі в складні збірки. В сучасних системах автоматизованого проектування під час зміни деталі, яка входить в збірку, одразу ж відбувається її зміна у вікні зі збіркою.

Можливість прив’язки розмірів видавлювання, витягування чи переміщення до властивостей деталі дає можливість створювати різні виконання деталей.

Така система сильно скорочує час, необхідний конструкторам на проектування виробу на підприємстві.

2.3 Класифікація CAD-систем

За галузевим призначенням CAD-системи можна класифікувати на: MCAD, EDA, AEC CAD, CALS.

Програмне забезпечення механічного комп'ютерного проектування MCAD використовується для створення і модифікації двовимірної і тривимірної геометрії з метою проектування, оцінки та документування механічних компонентів і компонентів, вузлів і виробів із листового металу. Ці компоненти включають форми, матриці та інші інструменти.

У процесі розробки програмне забезпечення MCAD можна використовувати для:

— дослідження різних ітерацій дизайну. Користувачі можуть створювати різні альтернативи та варіанти, порівнюючи їх один з одним;

— оцінки форми та відповідності конструкцій. Це включає вимірювання ваги конструкції або перевірку перешкод;

— формування інженерної документації. Ця документація надходить до виробництва як частина специфікацій, що використовуються для отримання, виготовлення та виробництва продукції.

Моделювання 3D дає можливість користувачам створювати і змінювати 3D-геометрію. Цей набір можливостей включає параметричне моделювання і пряме моделювання.

У параметричному моделюванні геометрія створюється і змінюється за допомогою стандартних визначень геометрії, які називають ознаками. Ці характеристики визначаються чисельно за допомогою параметрів. Функції

послідовно створюються і взаємозалежні, так що модифікація протікає через всю модель.

У прямому підході до моделювання геометрія створюється з попередньо визначеними формами, які не зберігають своє початкове визначення. Геометрія змінюється за допомогою маніпуляцій штовхання (push), витягування (pull) і перетягування (drag).

Ескізи 2D надають способи створення і зміни 2D-геометрії. Цей набір можливостей включає параметричний ескіз і пряме створення ескізів.

За допомогою параметричного підходу розділи 2D-об’єктів повністю визначаються і контролюються обмеженнями і параметричними розмірами.

У прямому підході двовимірні об’єкти можна штовхати, витягувати і перетягувати безпосередньо, коли програмне забезпечення робить розумні припущення щодо відповідної геометрії.

Моделювання складання надає спосіб зібрати багато окремих 3D-моделей з обмеженнями. Такі моделі можна використовувати для кінематичного й динамічного аналізу, як це передбачено програмним забезпеченням MCAE.

Інженерна документація надає різні методи для створення стандартної документації, яка потім випускається у виробництво. Цей набір можливостей включає: створення креслень, MBD.

Створювати креслення можна безпосередньо з моделі, параметри, використані для створення моделі, також можна відобразити в автоматизованому вигляді. Також користувачі можуть вручну маніпулювати двовимірними об’єктами в ескізах або кресленнях за допомогою таких команд, як розширити та обрізати.

Можливості визначення на основі моделі надають засоби для додавання інформації про виробництво продукту, такої як розміри та примітки до 3D-моделі. Отриману анотовану 3D-модель потім можна використовувати для доповнення або заміни 2D-креслень.

Оцінки на основі геометрії надають інструменти для оцінки й вимірювання характеристик і ознак 3D та 2D-геометрії.

Оцінки на основі об’єму засновані на об’ємі або тривимірній формі конструкції і включають вимірювання відстані, масу та об’ємні властивості, а також перевірку перешкод і зазору.

Двовимірні оцінки засновані на 2D-об’єктах, які розміщуються в одній площині. Це включає властивості площі та вимірювання відстані [4].

Електронна автоматизація проектування — це сегмент ринку, що складається з програмного забезпечення, апаратного забезпечення і послуг із спільною метою надання допомоги у визначенні, плануванні, проектуванні, реалізації, перевірці і подальшому виробництві напівпровідникових приладів або мікросхем. Щодо виробництва цих пристроїв, то основними постачальниками цієї послуги є ливарні напівпровідникові заводи. Ці надзвичайно складні та дорогі об’єкти належать і керуються великими вертикально інтегрованими напівпровідниковими компаніями або функціонують як незалежні постачальники виробничих послуг «чистого виду». Ця остання категорія стала домінуючою бізнес-моделлю.

Хоча рішення EDA не беруть безпосередньої участі у виробництві мікросхем, вони відіграють важливу роль у трьох аспектах.

По-перше, інструменти EDA використовують для проектування та перевірки процесу виробництва напівпровідників, щоб забезпечити необхідну продуктивність і щільність. Цей сегмент EDA називають технологією автоматизованого проектування.

По-друге, інструменти EDA використовують для перевірки того, що дизайн відповідає всім вимогам виробничого процесу. Недоліки в цій сфері можуть призвести до того, що отриманий чіп або не функціонує, або функціонує зі зниженою ємністю. Є також ризики надійності. Ця область уваги відома як дизайн для технологічності.

Після виготовлення мікросхеми зростає потреба в моніторингу продуктивності пристрою від тестування після виготовлення до розгортання в польових умовах. Метою цього моніторингу є гарантування того, що пристрій продовжує працювати належним чином протягом усього терміну його служби, а також гарантування, що пристрій не змінено. Ця третя програма називається керування життєвим циклом мікросхем [5].

Колекція AEC надає дизайнерам, інженерам і підрядникам набір інструментів BIM і CAD, які підтримуються загальним хмарним середовищем даних, що полегшує виконання проекту від початкового етапу проектування до будівництва [6].

Інструменти концептуального і детального проектування створюють високоякісні, високопродуктивні проекти будівель та інфраструктури.

Інструменти інтегрованого аналізу, генеративного проектування, а також інструменти візуалізації та моделювання допомагають в оптимізації проектів.

Інструменти, які максимізують конструктивність і координацію проекту покращують передбачуваність на місцях.

За допомогою інтегрованих робочих процесів для керування документами, концептуального проектування, моделювання, координації та документування можна прискорити і покращити якість процесів проектування.

Розробка концепції CALS зумовлена розвитком таких нових напрямків науки і техніки, як автоматизоване проектування, керування виробництвом, використання комп’ютерів для зберігання та обробки інформації, нові засоби зв’язку тощо. Кожен з цих напрямів окремо вніс революційні зміни в усі види людської діяльності, проте їхні значні можливості використовувалися недостатньо. Причиною стало те, що розробники сучасних засобів автоматизації формували свої власні моделі, які нерідко виявлялися несумісними у партнерів по виробництву. Частково ця проблема розв’язувалася ув’язкою різних систем автоматизованого проектування в інтегровані системи шляхом фізичного об’єднання баз даних, проте логічна ув’язка при цьому була відсутня, що

призводило до фрагментації інформації, багаторазового дублювання даних, неможливості інтеграції різних інтегрованих автоматизованих систем керування.

Суть концепції CALS полягає у створенні єдиної інтегрованої моделі виробу, що відображає всі аспекти, пов'язані з його властивостями і виробництвом. Ця модель повинна супроводжувати виріб на всьому протязі його життєвого циклу.

Базовими принципами CALS є безпаперовий обмін даними з використанням електронного цифрового підпису, аналіз і реінжиніринг бізнес-процесів, паралельний інжиніринг, системна організація поствиробничих процесів життєвого циклу виробу — інтегрована логістична підтримка [7].

2.4 Стадії проектування автоматизованої системи

Процес створення автоматизованої системи являє собою сукупність впорядкованих у часі, взаємопов'язаних, об'єднаних у стадії й етапи робіт, виконання яких необхідне і достатнє для створення АС, відповідної заданим вимогам.

Стадії й етапи створення АС виділяються як частини процесу створення з міркувань раціонального планування і організації робіт, закінчуються заданим результатом.

Роботи з розвитку АС здійснюють по стадіях і етапах, які застосовуються для створення АС.

Склад і правила виконання робіт на встановлених цим стандартом стадіях і етапах визначають у відповідній документації організацій, що беруть участь у створенні конкретних видів АС [8].

Стадії проектування АС:

— формування вимог до АС;

— розробка концепції АС;

— технічне завдання;

— ескізний проект;

— технічний проект;

— робоча документація;

— введення в дію;

— супровід АС.

На рисунку 2.2 відображено стадії проектування автоматизованої системи.

Рисунок 2.2 — Стадії проектування АС

Формування вимог до автоматизованої системи вимагає обстеження об’єкта та обґрунтування необхідності створення АС, формування вимог користувача до АС та оформлення звіту про виконану роботу і заявки на розробку АС (тактико-технічного завдання).

Під час розробки концепції проводиться вивчення об’єкта, проводяться необхідні науково-дослідницькі роботи, розробка варіантів концепції АС, що задовольняє вимогам користувача та оформлення звіту про виконану роботу.

Під час стадії технічного завдання виконується розробка і затвердження технічного завдання на створення АС.

Ескізний проект вимагає розробки попередніх проектних рішень щодо системи і її частин та розробки документації на АС і її частини.

Під час стадії технічного проекту відбувається розробка проектних рішень щодо системи і її частин, документації на АС і її частини, розробка та оформлення документації на поставку виробів для комплектування АС і (або) технічних вимог (технічних завдань) на їхню розробку та розробка завдань на проектування в суміжних частинах проекту об’єкта автоматизації.

На стадії робочої документації відбувається розробка робочої документації на систему і її частини та розробка або адаптація програм.

Введення в дію виконує підготовку об’єкта автоматизації до введення АС в дію, підготовку персоналу, комплектація АС, що поставляються виробами (програмними і технічними засобами, програмно-технічними комплексами, інформаційними виробами). Відбуваються будівельно-монтажні роботи, пусконалагоджувальні роботи, проведення попередніх випробувань, дослідної експлуатації, приймальних випробувань.

На фінальній стадії супроводу відбувається виконання робіт відповідно до гарантійних зобов’язань та післягарантійне обслуговування.

2.5 Призначення PDM-систем

Система PDM [9] призначена для зберігання даних про виріб. Виробами можуть бути різні складні технічні об’єкти.

Система керування даними дає можливість будувати зв’язки між документам і переглядати їх у зручній формі (рисунок 2.3).

За допомогою PDM-системи можна контролювати доступ до кожного типу документів. Завдяки цьому можна впевнитися, що документ може редагувати обмежене коло користувачів.

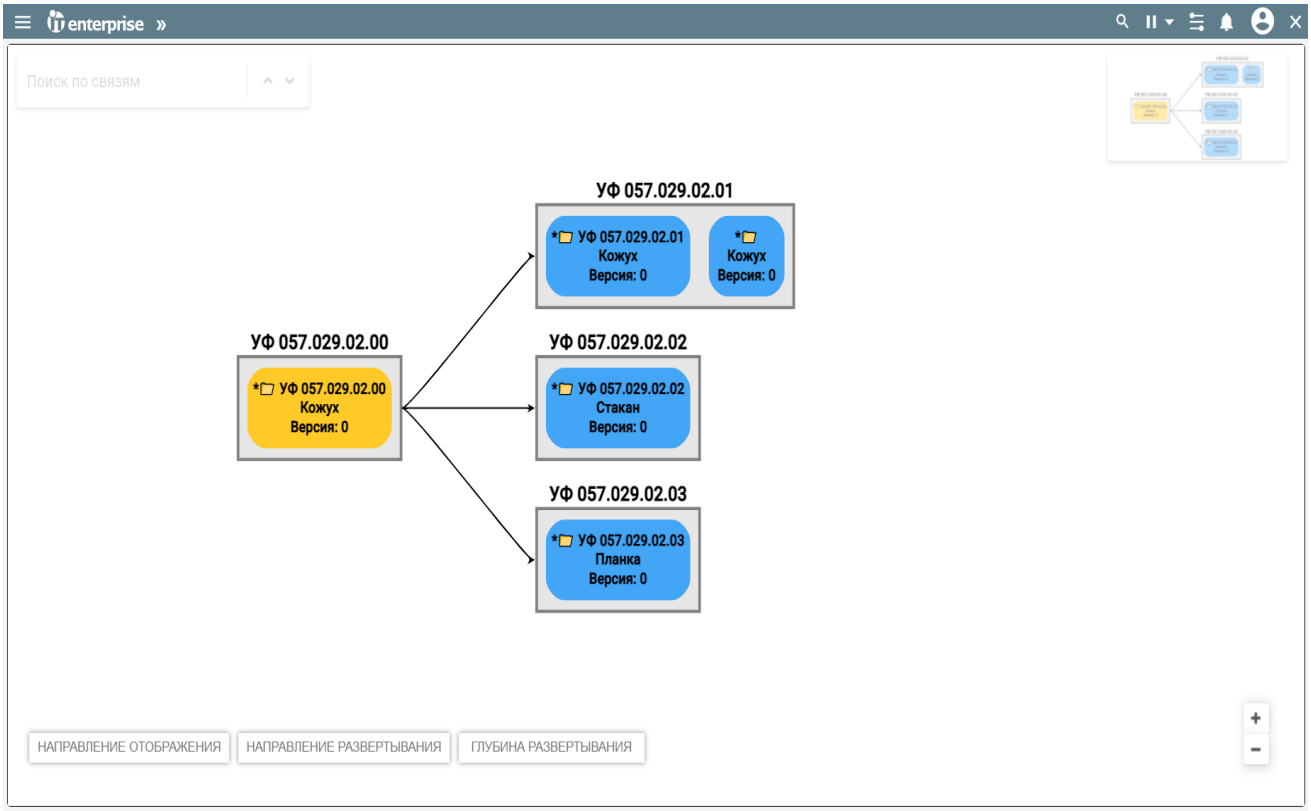

Рисунок 2.3 — Візуальне зображення зв’язків між документами в системі IT-Enterprise

Завдяки системі керування даними про виріб можна дізнатися, хто створив файл і хто вносив в нього зміни останнім. Ця інформація може бути корисною конструкторам.

Кожен документ повинен мати свою категорію. Усіма категоріями можна керувати за допомогою PDM-системи.

2.6 Технології PDM-систем

Системи PDM використовують технології: EDM, PIM, TDM, TIM [9].

Керування інженерними даними (EDM) — це практика свідомого створення систематичної основи для контролю великої кількості даних, які використовуються для створення практично будь-чого.

За допомогою цієї технології можна відобразити кожен етап, який пройде продукт. Перед впорядкуванням своїх інженерних даних, потрібно спочатку їх зафіксувати. Необхідно визначити кожен крок, який продукт проходить на своєму шляху від ідеї до реальності, все, від початкової концепції до моменту, коли він потрапляє на завод.

Ця технологія дає можливість визначити відповідального за кожен крок, можна призначити контактну точку для кожного етапу життєвого циклу. Незалежно від того, чи це керівник відділу, чи керівник групи, важливо, щоб була одна людина, яка несе відповідальність за збір даних у своїй сфері.

Дана технологія дає можливість зібрати усі дані, які можуть знадобитися у майбутньому. На кожному кроці фіксуються документи, креслення та інформацію про компоненти, назву бренду, номер деталі, опис. Навіть метадані мають вирішальне значення: хто їх створив, коли, що це таке, чому вони існують, коли їх востаннє змінювали.

Також вона дає можливість визначити номенклатуру. Встановлення загального стандарту дає можливість легко визначити, що це за файл, для чого він призначений і наскільки він актуальний.

Усі дані зберігаються в одному, безпечному місці. Зберігання всього в одному місці допомагає уникнути дублювання роботи і створює зручне єдине джерело істини.

Дуже просто ділитися документацією. До кожного документа можна надати доступ всім користувачам чи вибірково.

Процеси керування даними можна використати, щоб краще працювати разом. Зберігаючи всі дані в одному місці зі стандартизованою структурою, користувачам легше працювати разом у різних командах. Завдяки централізованій та організованій інформації будь-які запитання можна поставити людині, яка найкраще на них відповість, а не тому, хто є найближчим. Необхідно просто переконатися, що інструкції включають лише одну особу, яка працює над документом у даний час, і система надсилає сповіщення щоразу, коли файл оновлюється.

Обсяг даних, отриманих у процесі розробки лише одного продукту, є дуже великим. Організації щодня розробляють велику кількість продуктів, що робить обсяг даних гігантським. Проте на ринку є багато інструментів для автоматизації керування інженерними даними [10]. Ці інструменти для роботи з даними мають привабливий, інтуїтивно зрозумілий інтерфейс користувача. З їхньою допомогою можна відстежувати й керувати змінами даних, керувати дозволами, використовувати передові методи візуального керування.

Керування інженерними даними може полегшити процес щомісячної звітності. Добре організовані, централізовані файли можуть дати відчутний приріст продуктивності, ефективності й співпраці.

Рішення для керування інформацією про продукт (PIM) — це бізнес-додаток, який надає єдине місце для збору, керування і додавання інформації про продукт, створення каталогу продуктів і розповсюдження його в каналах продажів та електронної комерції.

Рішення PIM дає можливість швидше й простіше створювати привабливий продукт.

Керування інформацією про продукт — це процес керування всією інформацією, необхідною для маркетингу і продажу продуктів через канали збуту.

Інструменти PIM роблять процес керування інформацією про продукт швидким, ефективним і продуктивним. Завдяки PIM усі залучені можуть

співпрацювати, щоб вчасно та в рамках бюджету підготувати продукти до продажу.

За допомогою PIM можна керувати інформацією про: технічні дані (технічні характеристики, розміри, кольори, інгредієнти та інші факти), дані про використання (опис, де і як слід використовувати продукт), емоційні дані (дають можливість керувати переконливими описами та історіями продуктів, перераховувати цінності свого бренду і пов'язувати відповідні зображення, відео й документи з кожним продуктом) [11].

Процес керування технічними даними (TDM) використовується для планування, отримання, доступу, керування, захисту і використання даних технічного характеру для підтримки загального життєвого циклу системи. Керування даними включає розробку, розгортання, експлуатацію і підтримку за межами системного вилучення.

Процес керування даними зображено на рисунку 2.4.

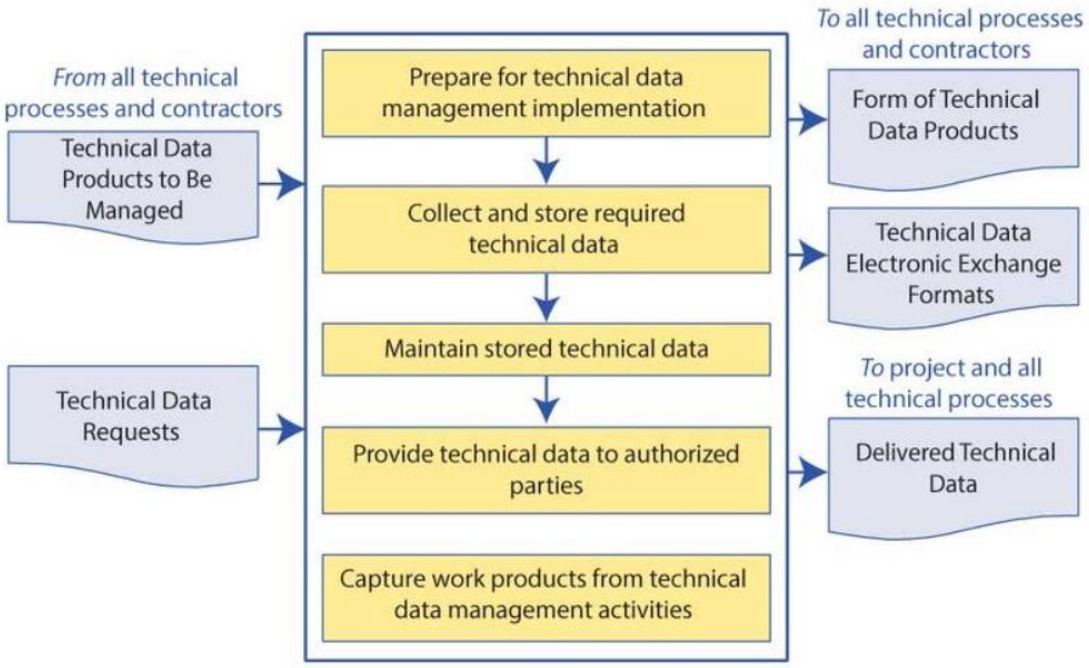

Рисунок 2.4 — Процес керування технічними даними [12]

Ключові аспекти керування даними для системної інженерії включають:

— застосування політики і процедур для ідентифікації й контролю даних;

— своєчасне й економне отримання технічних даних;

— забезпечення адекватності даних та їхнього захисту;

— полегшення доступу і поширення даних до місця використання;

— аналіз використання даних;

— оцінку даних для майбутньої цінності для інших програм/проектів;

— обробку доступу до інформації, записаної у застарілому програмному забезпеченні.

Процеси керування технічними даними і керування конфігурацією працюють одночасно, щоб гарантувати, що вся інформація про проект є безпечною, відомою і доступною. Зміни інформації, що перебувають під контролем конфігурації, вимагають запиту на зміну і, як правило, затверджуються Радою контролю конфігурації. Зміни в інформації в розділі «Керування технічними даними» не потребують запиту, але все одно ним потрібно керувати шляхом визначення, хто може вносити зміни до кожного типу технічних даних.

2.7 Система PDM, IT-Enterprise

PDM-система IT-Enterprise [9] дає можливість ефективно розв’язувати питання: керування конструкторською підготовкою виробництва, конфігураціями і створенням виробів, технічною підготовкою виробництва, нормування виробничих ресурсів та інтеграцією з CAD-системами.

Завдяки автоматизації процесу підготовки виробництва збільшується керованість бізнес-процесами на всіх етапах проектування виробів, якість конструкторсько-технологічної документації, надійність і конфіденційність зберігання інформації, контроль за термінами узгодження документів, рівень виконавчої дисципліни.

Дана система використовує відкриту модель нормування виробничих ресурсів, враховує особливості організації роботи технологічної служби на великих підприємствах. Для розрахунку конструкторських і технологічних показників використовується універсальний конструктор моделей.

Відкриття системи здійснюється переходом в головному меню по шляху «Керування виробництвом → Керування технічним документообігом → Архів документів → Архів документів», після відкриття буде представлено інтерфейс, зображений на рисунку 2.5.

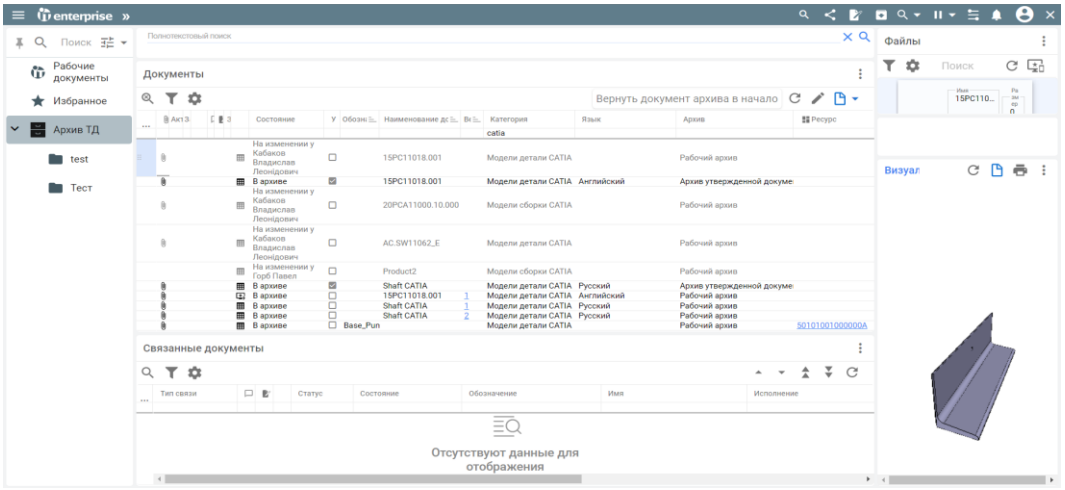

Рисунок 2.5 — PDM-система IT-Enterprise

Інтерфейс системи IT-Enterprise є зручним «user friendly»-інтерфейсом, який містить чотири закладки: документи, пов’язані документи, файли і візуальне представлення файлів. Кожна закладка в інтерфейсах IT-Enterprise дає можливість виконувати стандартний набір функцій. До цього набору входять такі функції: додавання, додавання за зразком, видалення, коригування, оновлення, швидкий перехід у конструктор інтерфейсів з позиціонуванням на даному інтерфейсі та на даній закладці та інші.

Закладка Документи містить основну інформацію про документ, а саме: стан документа, його найменування, позначення, категорія, до якого архіву

відноситься документ, ресурс, дата створення, інформація про користувача, який створив документ, і про користувача, який останнім корегував документ. Ця закладка містить стандартний набір функцій. Дані, які необхідно заповнити під час створення документа, подано на рисунку 2.6.

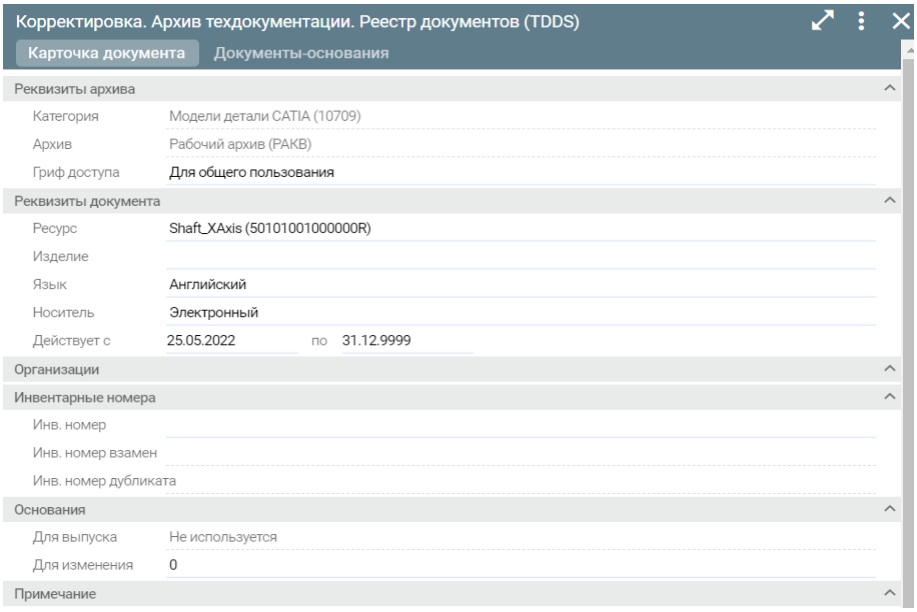

Рисунок 2.6 — Екран коригування картки документа в системі IT-Enterprise

Закладка зв’язаних документів містить інформацію про зв’язані документи, а саме: тип зв’язку, статус, стан, найменування, позначення, виконання документа, кількість, категорія та інші. Зв’язані документи можуть мати типи зв’язку: в комплекті, входить, зв’язаний файл, специфікація, виконання, креслення. Дані, які необхідно заповнити під час створення зв’язку між документами, представлені на рисунку 2.7. Всю інформацію про категорії документів і зв’язки між ними можна змінити за допомогою інтерфейсу «Категорії документів», який можна знайти по шляху «Керування виробництвом → Керування технічним документообігом → Налаштування → Категорії документів в архівах». Дана закладка містить стандартний набір функцій.

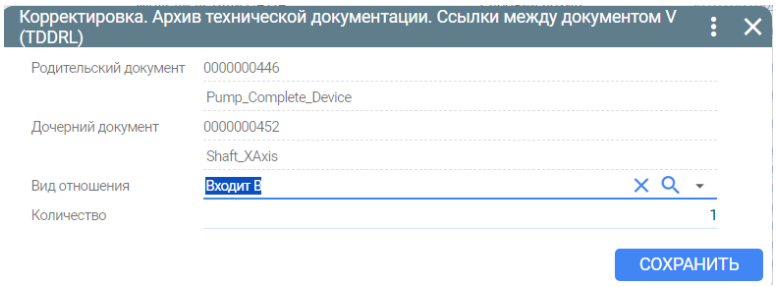

Рисунок 2.7 — Екран коригування зв’язків між документами

Закладки документів і зв’язаних документів мають можливість швидкого фільтру. На рисунку 2.8 зображено панель швидкого фільтру з пошуком за категорією «catia». Даний функціонал дає можливість швидко знайти потрібний документ серед інших.

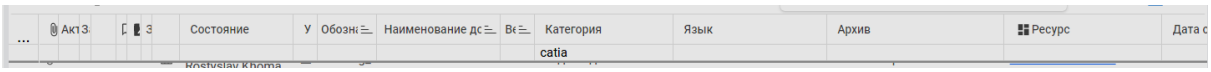

Рисунок 2.8 — Швидкий фільтр документів у системі IT-Enterprise

За допомогою закладки файл, зображеної на рисунку 2.9, можна виконувати стандартний функціонал закладок IT-Enterprise з деяким доповненням. Крім стандартних функцій, можна виконувати функції оновлення файлу в робочій директорії та оновлення файлу з робочої директорії.

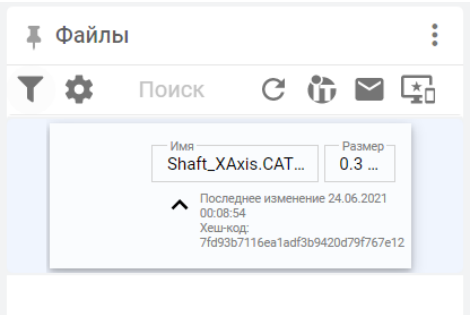

Рисунок 2.9 — Закладка файлів

Закладка візуального відображення файлів, зображена на рисунку 2.10. Вона забезпечує можливість переглядати зображення файлу в різних форматах.

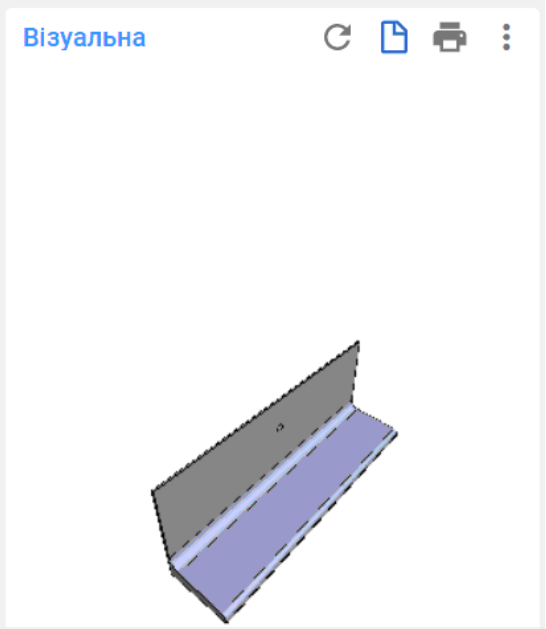

Рисунок 2.10 — Закладка візуального відображення файлів

Для форматів .vrml і .stl використовується спеціальний елемент керування, який дає можливість не тільки переглядати, а й контролювати розміщення деталі, масштаб деталі, дає можливість обертати деталь.

Висновки до розділу 2

У розділі розглянуто етапи розробки виробів на підприємстві. Описано призначення PDM та CAD-систем. Проаналізовано функціональні можливості PDM-системи IT-Enterprise.

3 ОСОБЛИВОСТІ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Розроблена система складається з двох частин: розширення CAD-системи і розширення PDM-системи. Для розширення можливостей CAD-системи було написано бібліотеку коду, яка за допомогою API виконує всі необхідні функції. Розширення для PDM-системи виконує функції обробника команд, які приходять з CAD-системи. Було створено API для можливості реалізації з боку PDM-системи.

3.1 Використання API для взаємодії з різними CAD-системами

Розробники CAD-систем передбачили можливу потребу розширення їхнього функціоналу користувачами. Разом з системою вони написали прикладний програмний інтерфейс (API) до неї. Даний функціонал — це чітко визначені методи, для взаємодії з компонентами системи. Він дає можливість створювати зовнішні програми, які можуть додавати нові команди та їхню реалізацію в систему. Як правило, такі програмні інтерфейси реєструються в системі як COM-компоненти.

До того, як Microsoft випустила мову програмування C# і платформу .NET, розробники програмного забезпечення, які створювали програми для сімейства операційних систем Windows, часто використовували модель програмування COM. Модель COM давала можливість користувачам створювати бібліотеки коду, які можна використовувати для різних мов програмування. Наприклад, програміст C++ міг би створити бібліотеку COM, яка може використовуватися розробником Visual Basic [13].

3.1.1 Програмний інтерфейс Creo VB API

Програмний інтерфейс Creo VB API [14] — це прикладний програмний інтерфейс, який дає можливість взаємодіяти з програмним забезпеченням Creo/Pro Engineering. Функціонал даного API змінюється разом з виходом наступної версії CAD-системи.

Основною мовою програмування є Microsoft Visual Basic.

Інтерфейс реєструється в системі користувача як COM-об’єкт, що дає можливість використовувати його разом з іншими мовами програмування, такими як C# чи JAVA.

Для встановлення даного програмного інтерфейсу необхідно вибрати його серед компонентів Creo під час встановлення програмного забезпечення (рисунок 3.1).

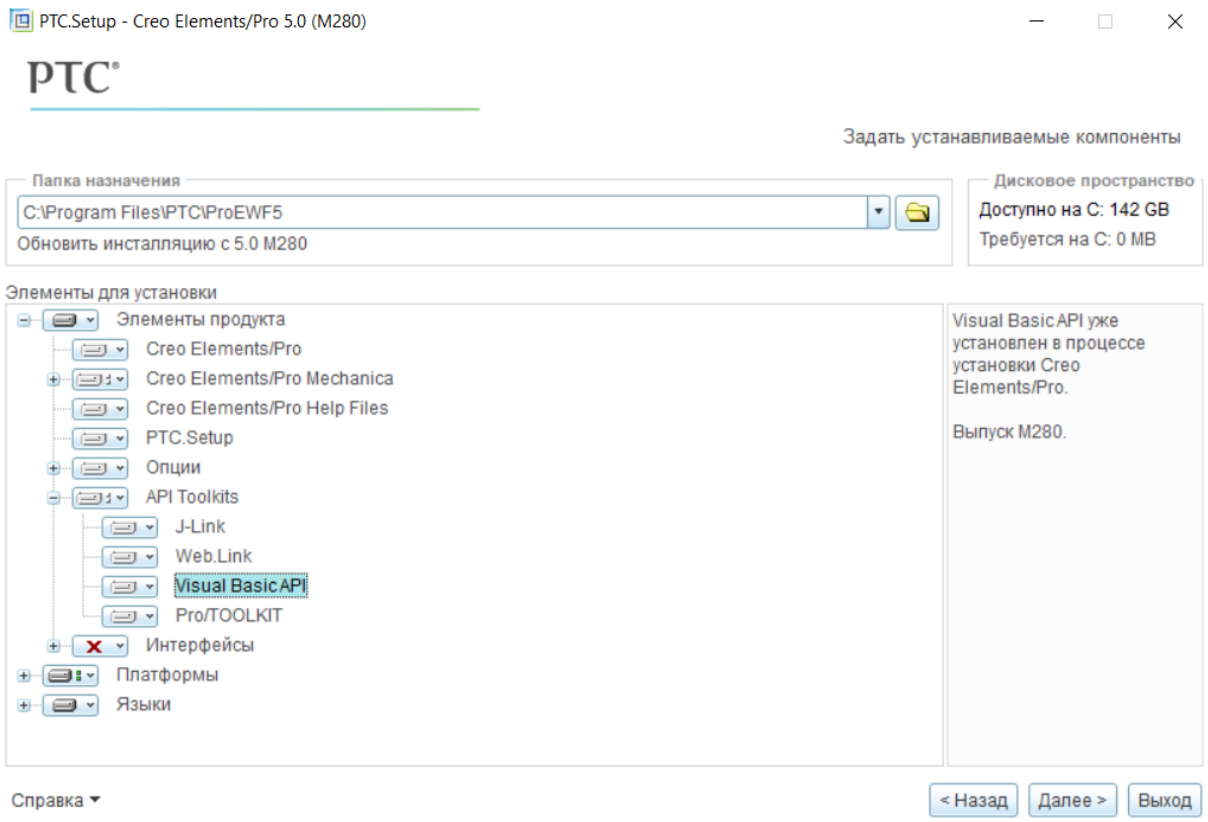

Рисунок 3.1 — Компонента VB API під час встановлення ПЗ Creo Elements

Після розпаковки програми в інсталяційній директорії з’явиться папка «vbapi», в якій будуть міститися деякі приклади використання функціоналу Creo VB API і документація до нього.

Основним елементом API є клас «IpfcAsyncConnection», від якого потім можна доступитися до об’єктів сесії, вікон і моделей. Існує два способи підключення до ПЗ Creo:

— підключитися до вже відкритої сесії, для чого треба використати метод CCpfcAsyncConnection().Connect();

— програмно запустити нову сесію, для чого використовується метод CCpfcAsyncConnection().Start().

3.1.2 Програмний інтерфейс CATIA API

Програмний інтерфейс CATIA API [15] — це прикладний програмний інтерфейс, який дає можливість взаємодіяти з програмним забезпеченням CATIA. Він автоматично встановлюється і реєструється на комп’ютері користувача разом з інсталяцією CAD-системи.

Основним класом даного програмного інтерфейсу є INFITF.Application (сам об’єкт додатку CATIA). З нього можна отримати об’єкти документів за допомогою методу FindDocument чи відкрити новий документ за допомогою методу OpenFile.

З його допомогою не можливо додати команди в панель інструментів CATIA, але це можна зробити за допомогою макрокоманд, користувацького алгоритму дій.

У цій CAD-системі мовою програмування макросів є VBA (Visual Basic for Applications), спрощена реалізація мови програмування Visual Basic. Макрокоманди даної системи мають розширення CATScript, приклад подано на рисунку 3.2.

```
Sub CATMain()

Dim hReq As Object
Set hReq = CreateObject("ITEnterprisePDMBrowser.CATIA.CADConnector.Application")

hReq.Open CATIA.Application

End Sub
```

Рисунок 3.2 — Приклад макроса додатку CATIA

Для того, щоб додати макрос у додаток CATIA, потрібно директорію, де міститься макрос, додати до макро-бібліотек додатку (рисунок 3.3).

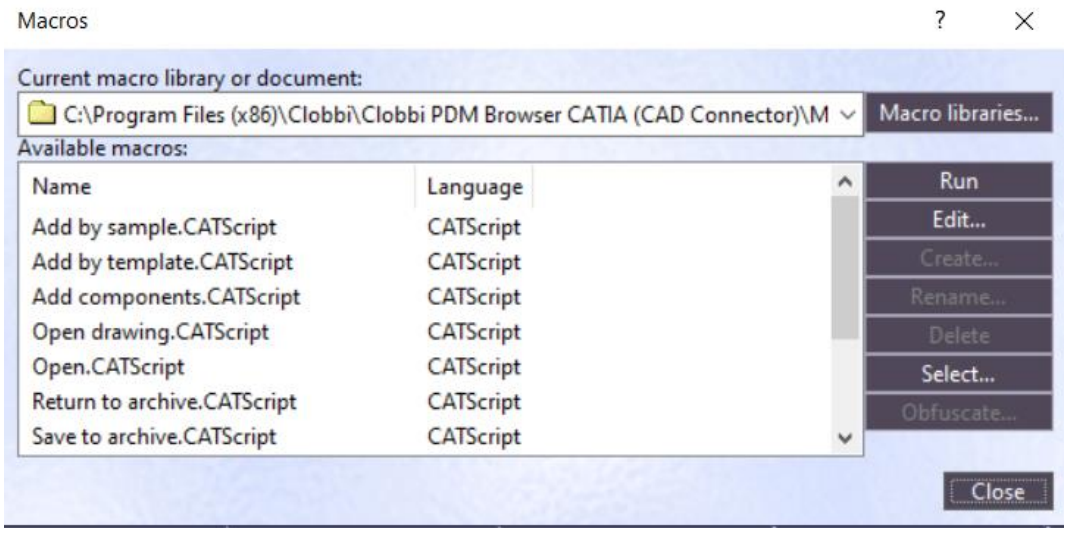

Рисунок 3.3 — Вікно макросів у системі CATIA

Щоб відкрити вікно з вибором бібліотек макрокоманд, потрібно перейти по шляху «Tools → Macro → Macros».

3.1.3 Програмний інтерфейс SolidWorks API

Програмний інтерфейс SolidWorks API [16] — це прикладний програмний інтерфейс, який дає можливість взаємодіяти з програмним забезпеченням

SolidWorks. З кожним оновленням системи автоматизованого проектування виходить нова версія прикладного програмного інтерфейсу.

Бібліотеки, необхідні для його використання, встановлюються разом з інсталяцією основного програмного забезпечення в папку SOLIDWORKS\api\redist.

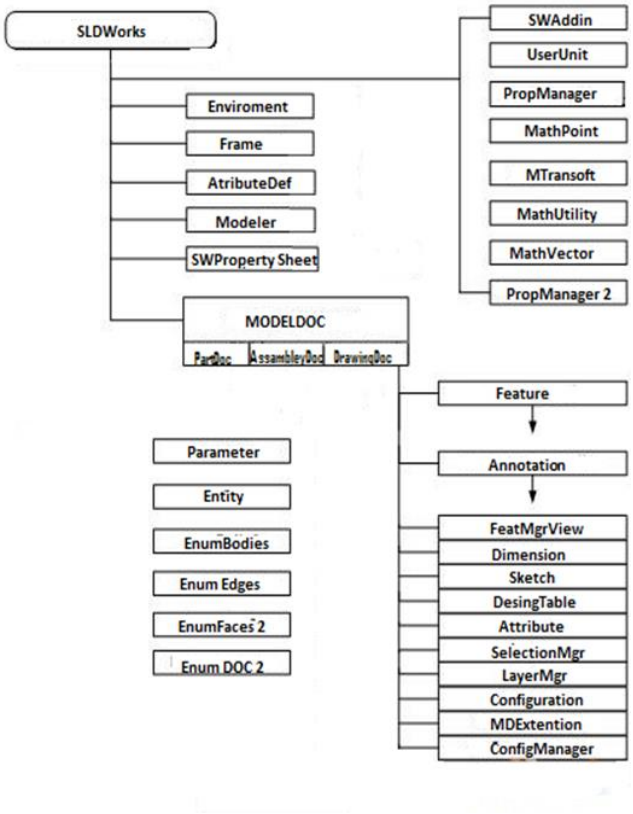

Рисунок 3.4 — Схема API Solidworks [16]

Для роботи з ним необхідно реалізувати інтерфейс ISwAddin, основними методами якого є ConnectToSW і DisconnectFromSW. За допомогою першого методу можна отримати об’єкт додатку SolidWorks (SldWorks), за допомогою якого виконувати подальшу роботу програми, також у цьому методі можна додати нові команди в панель керування програмного забезпечення (схему API Solidworks подано на рисунку 3.4). Він виконується під час старту програми SolidWorks. Друга команда необхідна, щоб видалити свої команди з панелі

інструментів і звільнити пам’ять CAD-системи. Вона виконується під час завершення роботи додатку.

Щоб переглянути, які програми запущено разом з додатком SolidWorks, необхідно перейти в Settings → Add-Ins…, відкриється екран, зображений на рисунку 3.5.

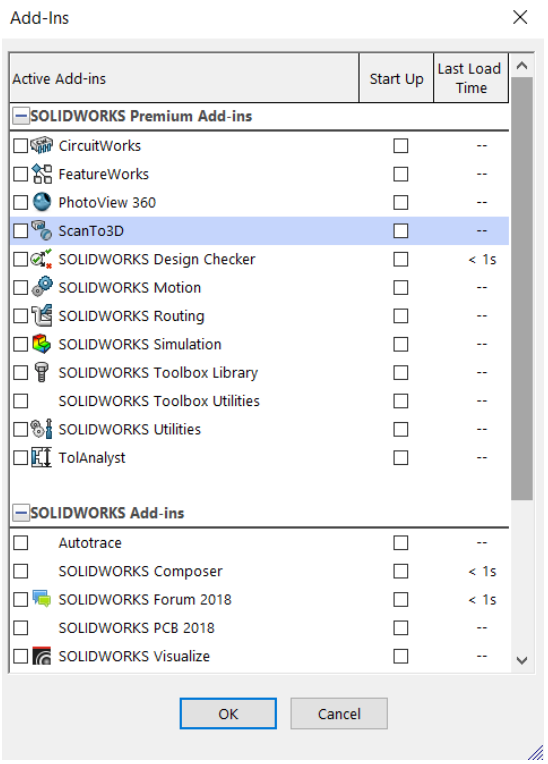

Рисунок 3.5 — Вікно додаткових програм ПЗ SolidWorks

У цьому екрані можна контролювати, які зовнішні програми активні в даний час і які програми потрібно запускати разом з запуском сесії SolidWorks.

3.2 Розширення для PDM-системи IT-Enterprise

Система IT-Enterprise функціонує в трирівневій архітектурі клієнт-сервер. Виділяються шари сервера бази даних, пулу серверів додатків (серверів додатків

може бути скільки завгодно), клієнтського програмного забезпечення. Ця система адаптивна, містить більше 3000 параметрів налаштування системи.

Розширення для PDM-системи виконує функції обробника команд, які приходять з CAD-системи. Було створено API для можливості реалізації з боку PDM-системи.

Основною мовою програмування системи IT-Enterprise є C#. З її допомогою було реалізовано інтерфейс для виклику функцій з PDM-системи. Даний інтерфейс зображено на рисунку 3.6.

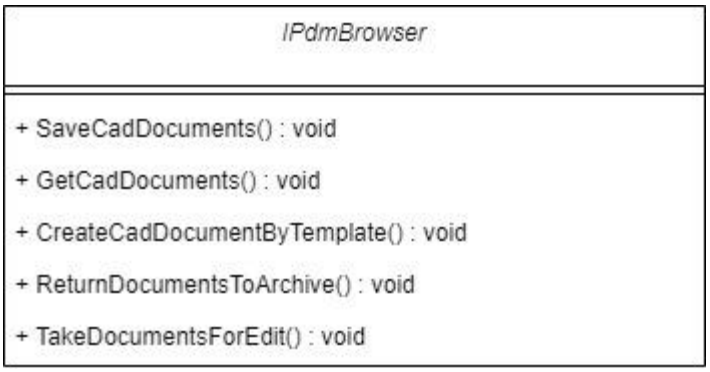

Рисунок 3.6 — API зв’язку з PDM-системою

Основними функціями цього API є:

— GetCadDocuments — метод для відкриття файлу документа з системи;

— TakeDocumentsForEdit — метод для взяття документів на зміну. Документ може перебувати в статусі на зміні в поточного користувача, на зміні в іншого користувача і в архіві (не на зміні в жодного користувача);

— SaveCadDocuments — метод для збереження документів в архів;

— ReturnDocumentsToArchive — метод, який дає можливість повернути документ в архів (змінити статус документа на «в архіві») і зберегти його;

— CreateCadDocumentByTemplate — метод, який створює документ за шаблоном, вибраним у PDM-системі.

Система IT-Enterprise містить API для взаємодії з таблицями бази даних. Щоб виконати SQL-запит, у системі IT-Enterprise потрібно створити команду за допомогою методу SqlClient.Main.CreateCommand(), яка в якості аргументу приймає рядок SQL-запиту, далі залежно від очікуваного результату треба використати метод ExecScalar (для простого результату типу числа чи рядка), ExecScalars (для масиву чисел чи рядків), ExecObject (для отримання об’єкта певного класу), ExecObjects (для отримання масиву об’єктів певного класу) чи ExecNonQuery (для виконання команд, які не повертають результат).

Для взяття документа на зміну (коригування) перевіряється статус документа, якщо статус «в архіві», то його змінює на «на зміні в поточного користувача», якщо статус «на зміні в поточного користувача», то нічого не змінюється, якщо статус «на зміні в іншого користувача», то видається повідомлення про те, що не можна змінити статус документа, оскільки він перебуває на зміні в іншого користувача.

Під час виконання цього методу здійснюється пошук і зміна в таблиці зв’язаних документів в PDM-системі (ITTDPACK). Її структуру подано в таблиці 3.1.

Таблиця 3.1 — Структура таблиці ITTDPACK зв’язаних документів в PDM-системі

Назва поля	Тип поля	Опис
id	varchar(10)	Унікальний ідентифікатор зв’язку
untdds1	varchar(10)	Унікальний ідентифікатор документу, який прив’язаний до поточного
untdds2	varchar(10)	Унікальний ідентифікатор поточного документу
relation	varchar(10)	Тип зв’язку

Під час збереження документа в архіві, якщо такого документа ще немає, відбувається створення нового документа, заповнення його позначення, найменування, ресурсу та іншої інформації, необхідної під час створення документа.

Якщо такий документ вже існує в архіві, відбувається зчитування властивостей деталі, якщо основні властивості типу позначення чи найменування змінилися, відбувається їхня актуалізація в PDM-системі. Останнім кроком стає завантаження деталі, креслення чи збірки в систему. Якщо документ існував у системі, відбувається його заміна.

Повернення документа в архів означає збереження документа і зміна його статусу на «в архіві».

Під час виконання методів збереження і повернення документа в архів змінюються дані таблиць: карточки документів (TDD) — структуру подано в таблиці 3.2; змінних реквізитів документів (TDDS) — структуру подано в таблиці 3.3; файлів документів в PDM-системі (TDFILES) — структуру подано в таблиці 3.4; зв’язаних документів (ITTDPACK) — структуру подано в таблиці 3.5; властивостей (ITTDPROP) — структуру подано в таблиці 3.6.

Таблиця 3.2. Структура таблиці TDD карточки документів

Назва поля	Тип поля	Опис
untdd	varchar(10)	Унікальний ідентифікатор карточки документа
tdk	varchar(10)	Категорія документа
kmat	varchar(15)	Код ресурсу документа
ndoc_full	varchar(255)	Найменування документа
ndoc_sfull	varchar(100)	Позначення документа

Таблиця 3.3. Структура таблиці TDDS змінних реквізитів документів

Назва поля	Тип поля	Опис
untdds	varchar(10)	Унікальний ідентифікатор документа
untdd	varchar(10)	Унікальний ідентифікатор карточки документа
ttdv	varchar(10)	Вид архіву

Таблиця 3.4. Структура таблиці TDFILES файлів документів в PDM-системі

Назва поля	Тип поля	Опис
id	varchar(20)	Унікальний ідентифікатор
filename	varchar(255)	Ім’я файлу
filesize	varchar(15)	Розмір файлу
extension	varchar(15)	Розширення файлу
untdds	varchar(10)	Унікальний ідентифікатор документа

Створення документа на основі шаблону дає можливість вибрати попередньо завантажений в PDM-систему шаблон деталі, збірки чи креслення і з його допомогою створити новий документ.

Шаблони завантажені в системі IT-Enterprise в таблиці TDTEMPL, структуру якої подано в таблиці 3.6.

Таблиця 3.5. Структура таблиці зв’язаних документів ITTDPROP

Назва поля	Тип поля	Опис
id	varchar(10)	Унікальний ідентифікатор
untdds	varchar(10)	Унікальний ідентифікатор документа
name	varchar(255)	Ім’я властивості
value	varchar(255)	Значення властивості

Таблиця 3.6. Структура таблиці властивостей TDTEMPL

Назва поля	Тип поля	Опис
id	varchar(10)	Унікальний ідентифікатор
ttdk	varchar(10)	Код категорії документів
ext	varchar(15)	Розширення шаблону
template	memo(4)	Дані шаблону
name	varchar(255)	Ім’я файлу шаблону
templsize	varchar(10)	Розмір шаблону

Зв’язки між таблицями зв’язаних документів в PDM-системі ITTDPACK, карточки документів TDD, змінних реквізитів документів TDDS, файлів документів в PDM-системі TDFILES, зв’язаних документів ITTDPROP і властивостей TDTEMPL бази даних PDM-системи IT-Enterprise зображено на рисунку 3.7.

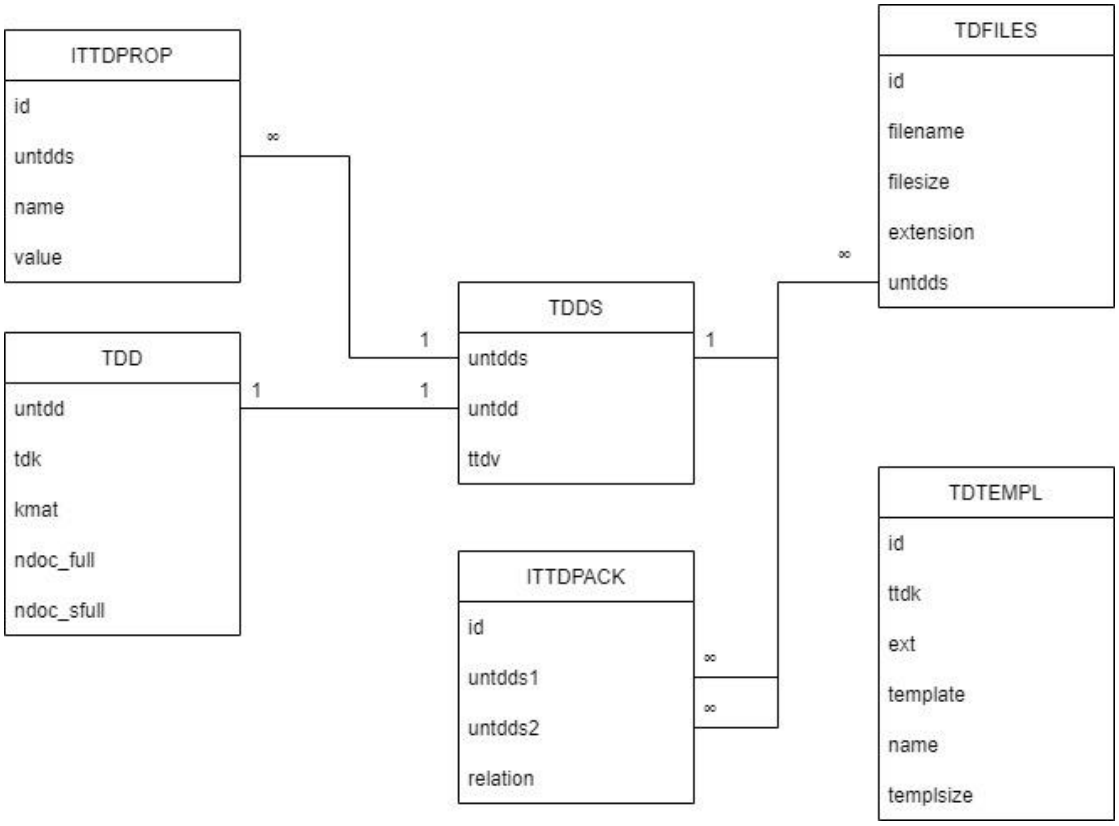

Рисунок 3.7 — Зв’язок між таблицями бази даних PDM-системи IT-Enterprise

Під час завантаження файлу з архіву відкривається інтерфейс з документами архіву, після вибору документа перевіряється наявність у нього прикріплених файлів і наявність зв’язаних файлів. Якщо документ містить прикріплені файли, то відбувається перевірка користувача на доступ до даних файлів, після чого він надсилається на машину клієнта, якщо ні, то виводиться відповідне повідомлення.

Висновки до розділу 3

У розділі описано процес створення розширень для різних CAD-систем та створене розширення для PDM-системи IT-Enterprise.

4 ЗАСОБИ РОЗРОБКИ

Розробка програмного забезпечення — це процес, за допомогою якого окреме або індивідуальне програмне забезпечення створюється з використанням певної мови програмування. Він передбачає написання серії взаємопов’язаних програмних кодів, які забезпечують функціональність розробленого програмного забезпечення [17].

4.1 Середовище розробки Visual Studio

Середовище Visual Studio [18] — інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Це середовище дає можливість створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад, віконні додатки Windows Forms, а також Web-додатки тощо.

Інтегроване середовище Visual Studio дає можливість розробляти додатки, використовуючи різні мови програмування: Visual C#, Visual Basic, Visual F#, Visual C++, Python та інші. Також існує можливість розробляти додатки не тільки під Windows, а й під інші популярні платформи: Android, iOS.

Версія Visual Studio Community є абсолютно безкоштовною для учнів, студентів і розробників програм з відкритим програмним кодом [18].

На даний момент останньою версією продукту є Microsoft Visual Studio 2022. Ця версія, на відміну від попередньої, стабільно працює з великими проектами, швидко здійснює пошук потрібного файлу в них. Система завершення коду побудована на базі штучного інтелекту. За допомогою зручного інтерфейсу легко можна підключати зовнішні компоненти, такі як бази даних чи бібліотеки кодів.

Інтерфейс середовища розробки представлено на рисунку 4.1.

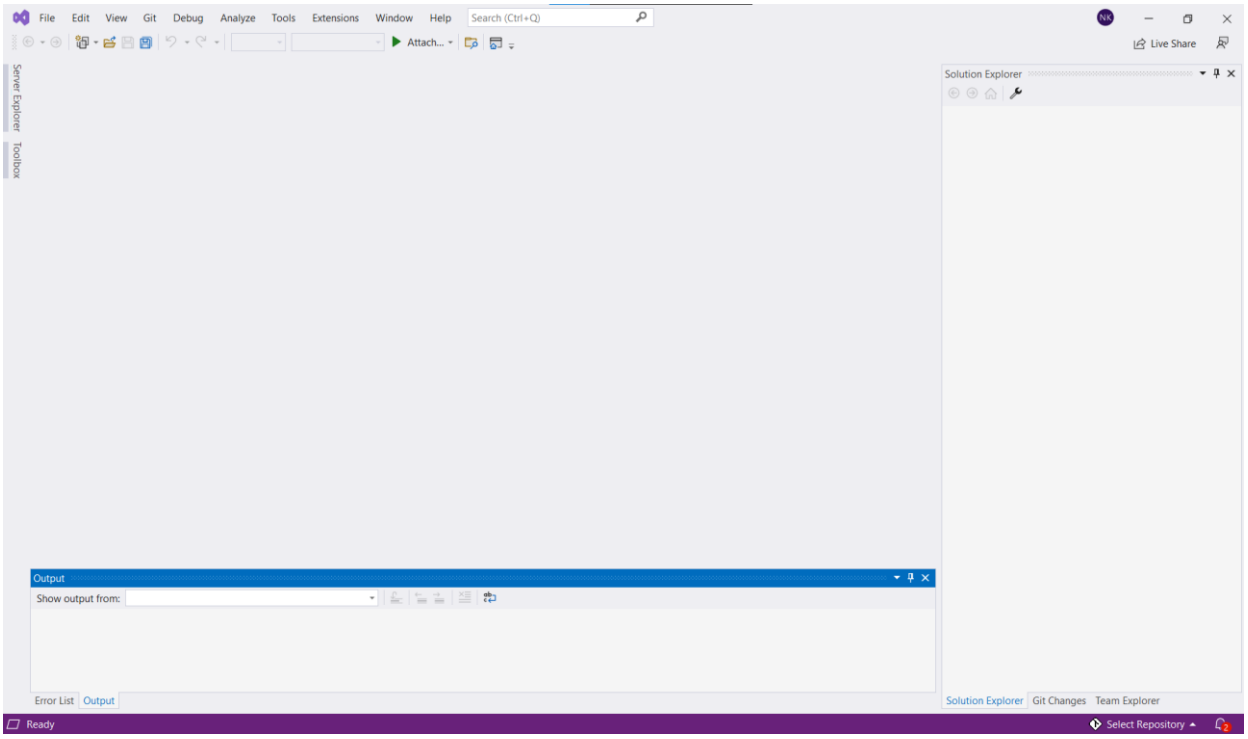

Рисунок 4.1 — Середовище розробки Microsoft Visual Studio 2022

Основним елементом середовища розробки є Solution Explorer. Дана закладка дає можливість переглядати файлову систему проекту і змінювати всі його файли.

4.2 Мова програмування C#

Мова програмування C# [13] — це високорівнева мова програмування. Разом з фреймворком .NET цю мову можна використовувати для написання програм не лише для пристроїв на базі операційної системи Windows, а й для пристроїв на Linux і macOS.

Мова C# і платформа .NET вперше були представлені в 2002 році і мали на меті запропонувати дуже потужну, гнучку і просту модель програмування.

Три ключові переваги, які має .NET, це CLR, CTS і CLS.

Основна роль CLR — знаходити, завантажувати й керувати об'єктами .NET від імені програміста. Загальномовне виконуюче середовище CLR також піклується про ряд низькорівневих деталей, таких як керування пам'яттю, розміщення програм, координація потоків і виконання основних перевірок безпеки (серед інших низькорівневих деталей).

Специфікація CTS повністю описує всі можливі типи даних і всі конструкції програмування, які підтримуються середовищем виконання, вказує, як ці об'єкти можуть взаємодіяти один з одним, а також детально описує, як вони представлені у форматі метаданих .NET.

Специфікація загальної мови, або CLS, — це пов'язана специфікація, яка визначає підмножину загальних типів і конструкцій програмування, з якими можуть погодитися всі мови програмування .NET. Таким чином, якщо програміст створює типи .NET, які надають лише функції, сумісні з CLS, то всі мови, які підтримують .NET, можуть використовувати їх.

На додаток до специфікацій CLR, CTS і CLS, платформа .NET надає бібліотеку базових класів, яка доступна для всіх мов програмування .NET. Ця бібліотека базового класу не тільки інкапсулює різні примітиви, такі як потоки, введення, вивід у файл, графічні системи візуалізації та взаємодія з різними зовнішніми апаратними пристроями, але вона також забезпечує підтримку ряду служб, необхідних для більшості реальних програм.

Бібліотеки базових класів визначають типи, які можна використовувати для створення будь-якого типу програмного забезпечення. Наприклад, можна використовувати ASP.NET для створення веб-сайтів і служб REST, WCF для створення розподілених систем [19], WPF для створення настільних програм [20], GUI тощо.

Відношення між CLR, CTS, CLS і бібліотекою базового класу подано на рисунку 4.2 [13].

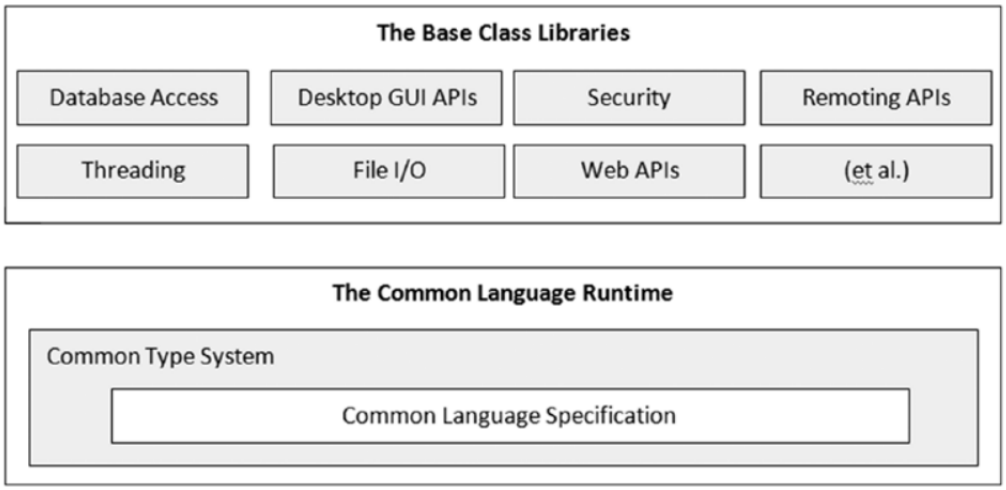

Рисунок 4.2 — Відношення CLR, CTS, CLS та бібліотеки базового класу

Мова програмування C# — сучасна, об’єктно-орієнтована мова програмування, яка дає можливість розробникам створювати багато додатків різноманітних типів.

4.3 Система керування базами даних Microsoft SQL Server

СУБД Microsoft SQL Server [21] — це звичайна програма, яка працює на комп’ютері, сервері, на віртуальній машині чи в хмарі. Вона дає можливість підключитися до неї локально чи через мережу, відправляти команди за спеціальним протоколом TDS та одержати відповідь. Це сучасна платформа, яка дає можливість зберігати дані і взаємодіяти з ними.

Для керування базами даних Microsoft SQL Можна використовувати програму Microsoft SQL Management Studio. З її допомогою можна здійснювати маніпуляції з базами даних.

Інтерфейс системи керування базами даних подано на рисунку 4.3.

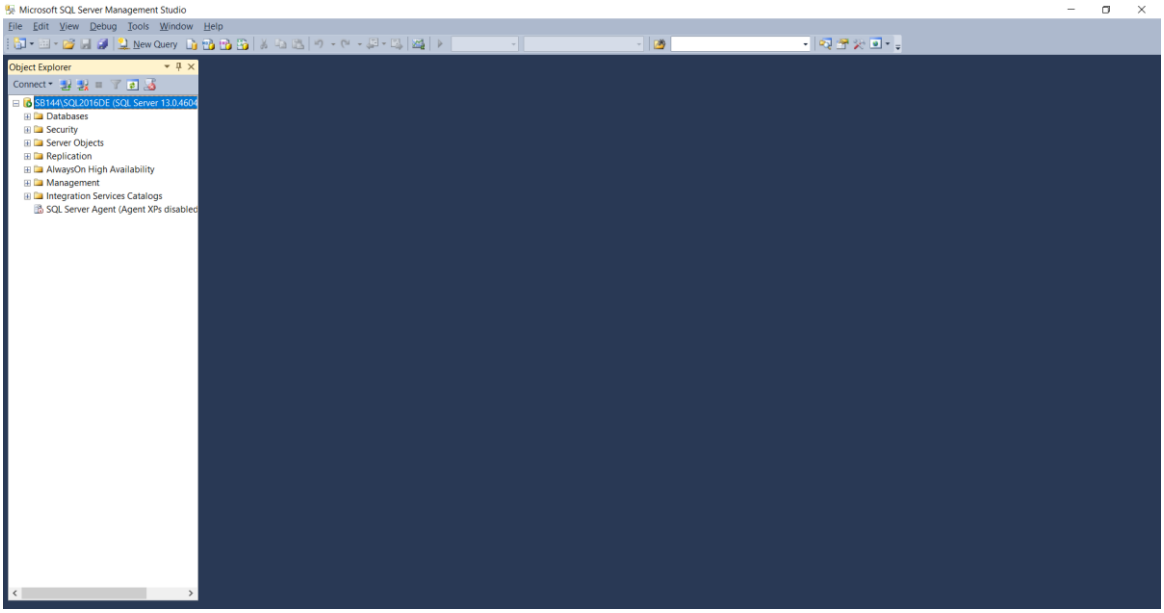

Рисунок 4.3 — Середовище системи керування базами даних Microsoft SQL Management Studio

Головним елементом Microsoft SQL Management Studio є закладка «Object Explorer», яка дає можливість користувачеві керувати об’єктами сервера.

4.4 Фреймворк CefSharp

Обгортка CefSharp [22] — це простий спосіб вбудувати повнофункціональний веб-браузер, що відповідає стандартам, у програму C# або VB.NET. Вона має елементи керування браузером для додатків WinForms і WPF, а також версію для проектів автоматизації. Даний фреймворк заснований на Chromium Embedded Framework, відкритій версії Google Chrome.

Приклад використання даного фреймворку, разом з WinForm додатком подано на рисунку 4.4.

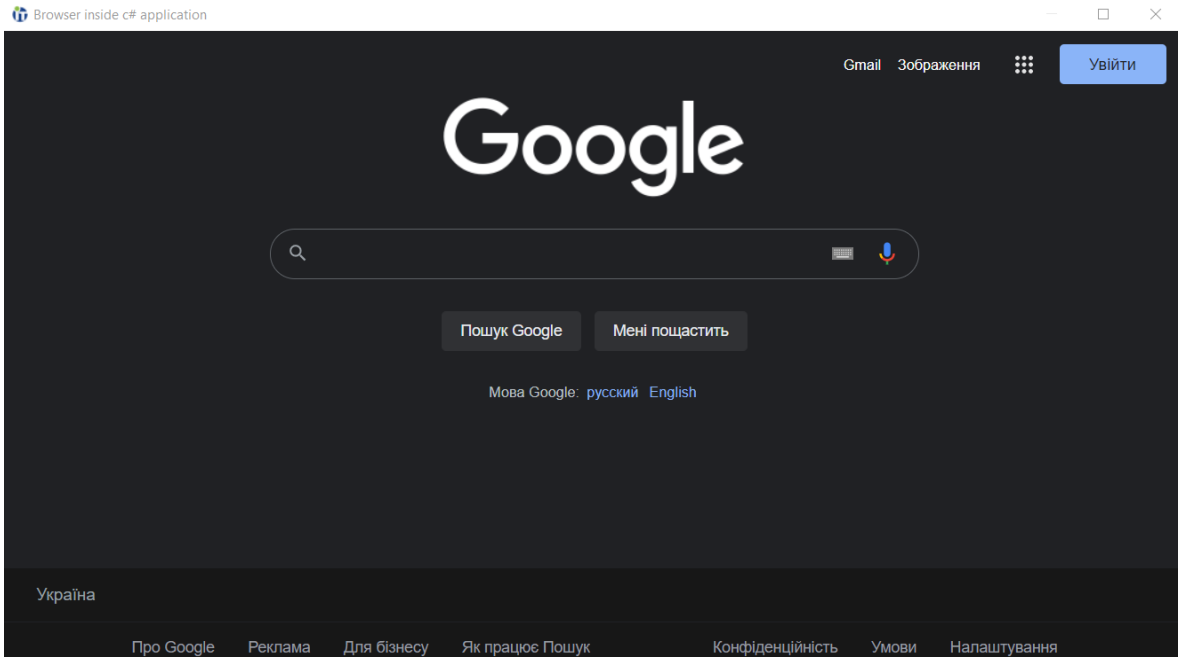

Рисунок 4.4 — Веб-браузер, вбудований в WinForms-додаток

Програмний каркас CefSharp — це обгортка .NET з відкритим вихідним кодом навколо Chromium Embedded Framework. Даний код можна використовувати, щоб покращувати, розширювати, краще налагоджувати розроблювані програми.

Каркас CefSharp можна використовувати для запатентованих і безкоштовних програм з відкритим кодом.

4.5 Розширення середовища розробки Microsoft Visual Studio Installer Projects

Розширення середовища розробки Microsoft Visual Studio Installer Projects дає змогу створювати пакети інсталятора для розповсюдження серед користувачів. Після створення проекту створюються файли установки, які можна

розповсюджувати користувачам. Користувач запускає інсталяційний файл і виконує кроки майстра для встановлення програми.

Служба Microsoft Windows Installer [23] — це служба встановлення і налаштування на основі даних, яка постачається як частина операційної системи Windows.

Підсистема Windows Installer підтримує базу даних про кожну встановлену програму, включаючи файли, ключі реєстру і компоненти. Коли програму видалено, базу даних перевіряють, щоб переконатися, що інші програми не використовують файл, ключ реєстру або компонент, перш ніж видалити їх. Це запобігає вилученню однієї програми від поломки іншої [24].

Під час встановлення програми інсталятором, створеним за допомогою Microsoft Visual Studio Installer Projects потрібно виконати прості кроки, такі як вибрати директорію для установки, представлено на рисунку 4.5, чи підтвердити установку, рисунок 4.6.

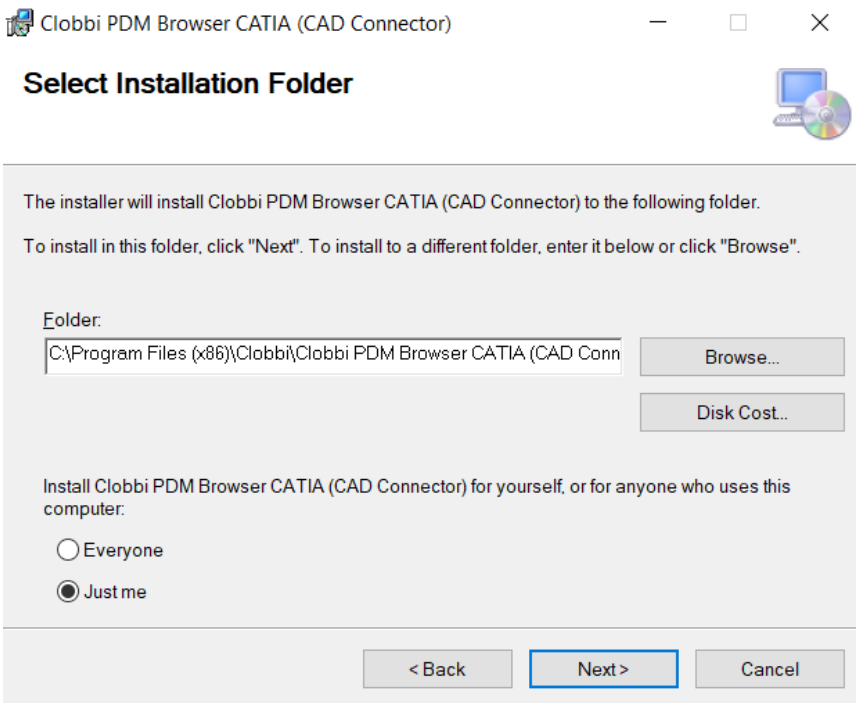

Рисунок 4.5 — Вибір директорії установки програми

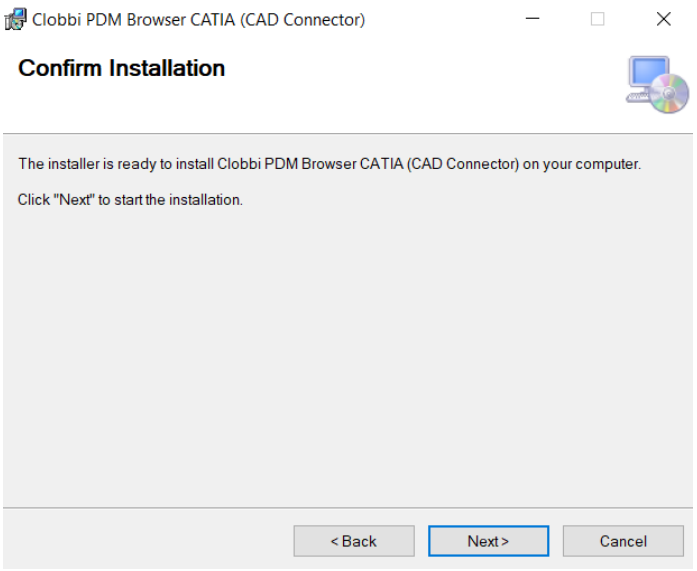

Рисунок 4.6 — Підтвердження установки.

Після успішної установки всі файли, необхідні для роботи програмного забезпечення будуть розпаковані в директорії, яку вказав користувач. Також інсталятор зареєструє програму в операційній системі, це робить неможливим повторну інсталяцію тієї програми, яка вже встановлена на комп’ютер.

Висновки до розділу 4

У розділі розглянуто використані засоби розробки програмного забезпечення для розширення функціоналу CAD-системи для взаємодії з PDM-системою.

5 РОБОТА КОРИСТУВАЧА З РОЗШИРЕННЯМ CAD-СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З PDM-СИСТЕМОЮ

Розділ містить інформацію про встановлення програмного забезпечення і підключення його до CAD-системи CATIA. Також у ньому описано сценарії робити користувача з ПЗ.

5.1 Встановлення програмного забезпечення

Для простого встановлення програмного забезпечення було створено файл установки, який розпаковує всі необхідні файли для роботи розширення в обрану користувачем директорію.

Встановлені макрокоманди треба додати в програмне забезпечення CATIA. Для цього потрібно перейти в «Tools → Macro → Macros» (рисунок 5.1).

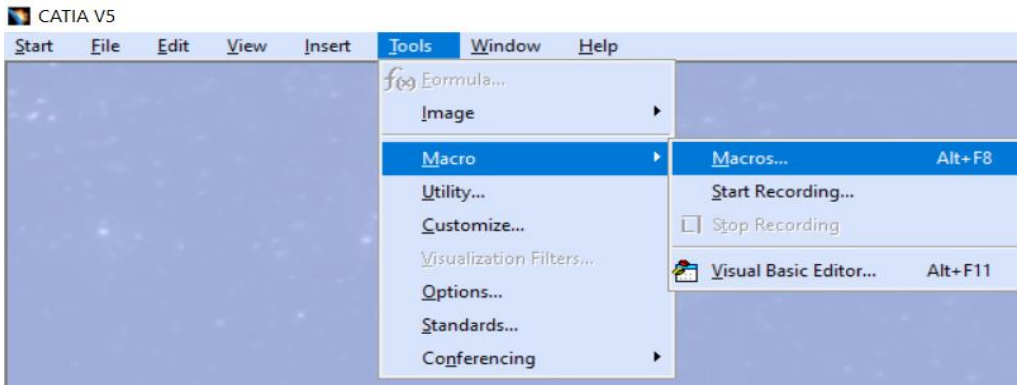

Рисунок 5.1 — Пункт меню макроси в CAD-системі CATIA

Після цього у вікні макросів треба додати існуючу бібліотеку макросів, яку інсталятор завантажив у папку «Macro», яка розміщується у вказаній

користувачем директорії. З’явиться список макрокоманд, поданий на рисунку 5.2.

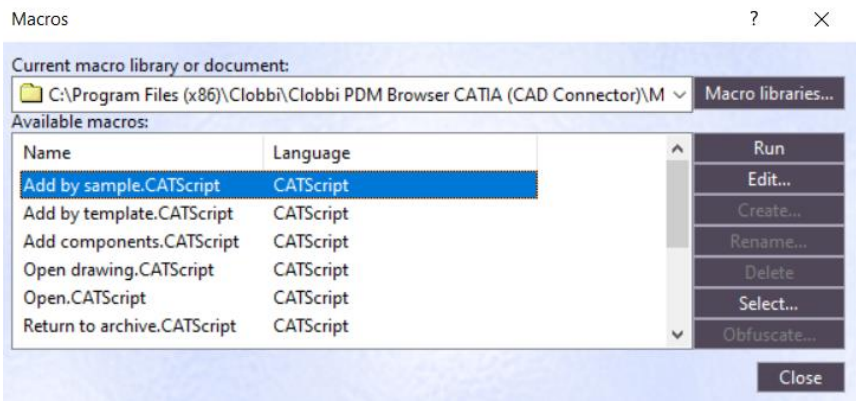

Рисунок 5.2 — Макроси після додавання бібліотеки макрокоманд

Після того, як макроси додадуться в PDM-систему, їх необхідно помістити на панель інструментів. Для цього потрібно перейти в «Tools → Customize» і в відкритому вікні вибрати пункт «Toolbars» (рисунок 5.3).

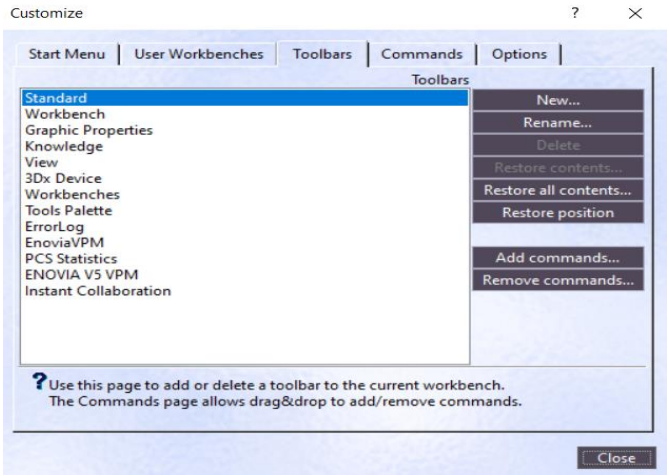

Рисунок 5.3 — Вікно конфігурації панелі інструментів

На даній закладці необхідно створити нову панель інструментів. Далі необхідно додати команди до створеної панелі інструментів. Натиснувши на

кнопку «Add commands», зі списку потрібно вибрати команди з папки «Macro» (рисунок 5.4).

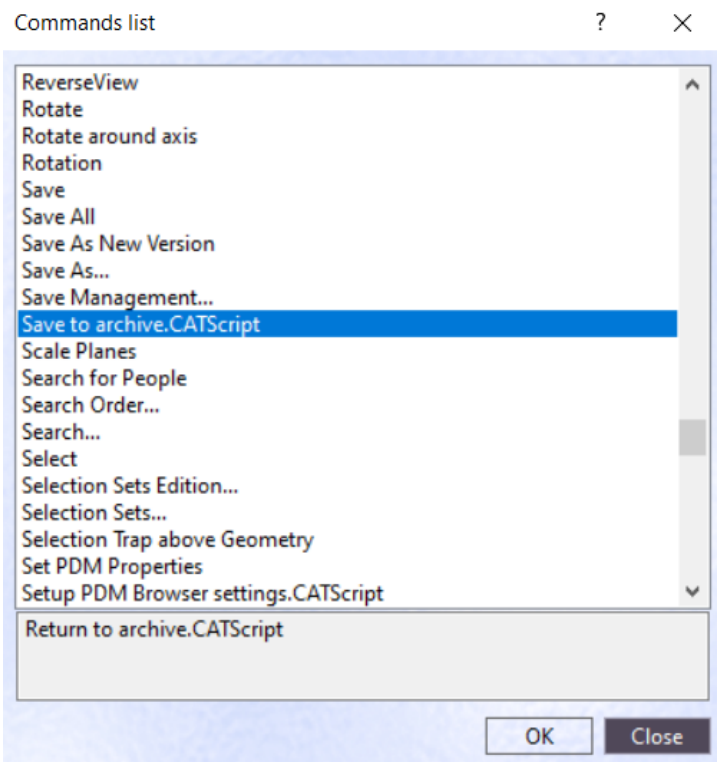

Рисунок 5.4 — Вибір команд, які треба додати в панель інструментів

Після того, як команди були додані в панель інструментів, їм потрібно дати піктограми із вже існуючих в CAD-системі CATIA.

Щоб це зробити, у вікні «Customize» потрібно перейти на закладку «Commands», у списку категорій вибрати «Macros» чи «All Commands», у списку команд знайти потрібну команду і вибрати для неї потрібну піктограму (рисунок 5.5).

Далі можна ввімкнути панель інструментів, для цього потрібно перейти в «View → Toolbars» і в випадаючому списку відмітити щойно створену панель інструментів (рисунок 5.6).

У результаті з’явиться панель інструментів (рисунок 5.7).

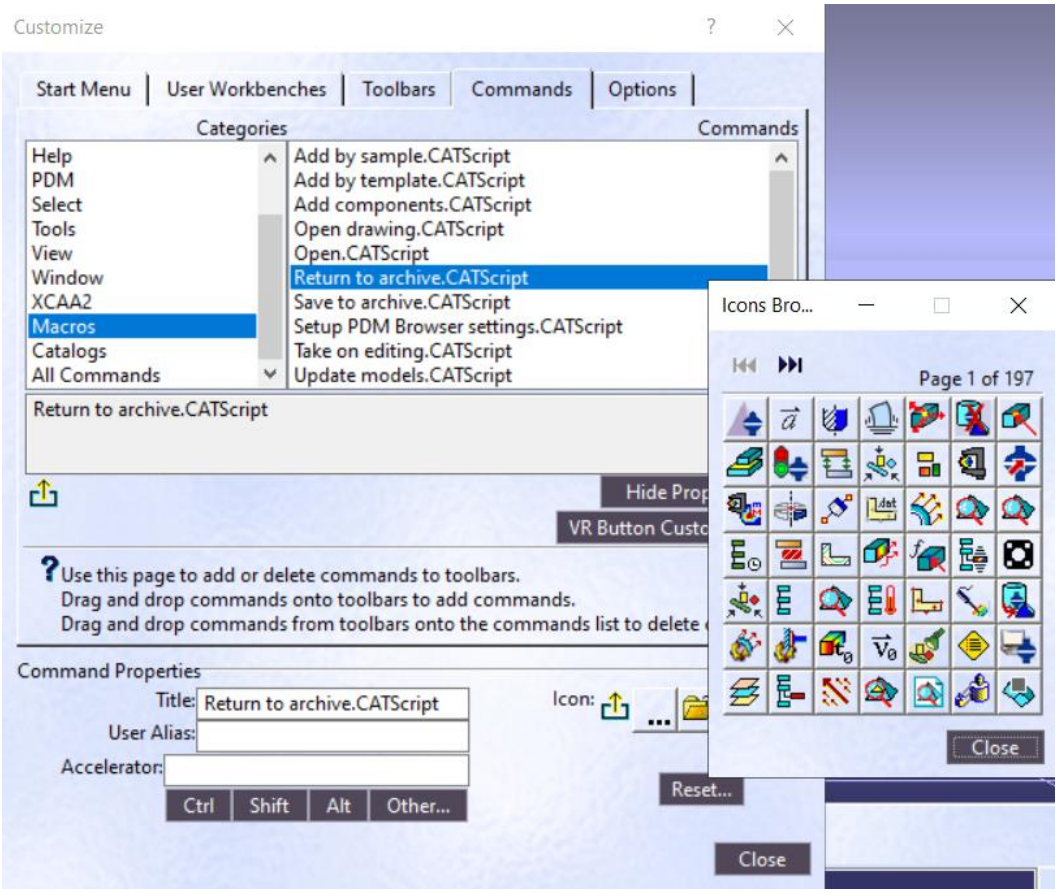

Рисунок 5.5 — Вибір піктограми для команди «Повернення до архіву»

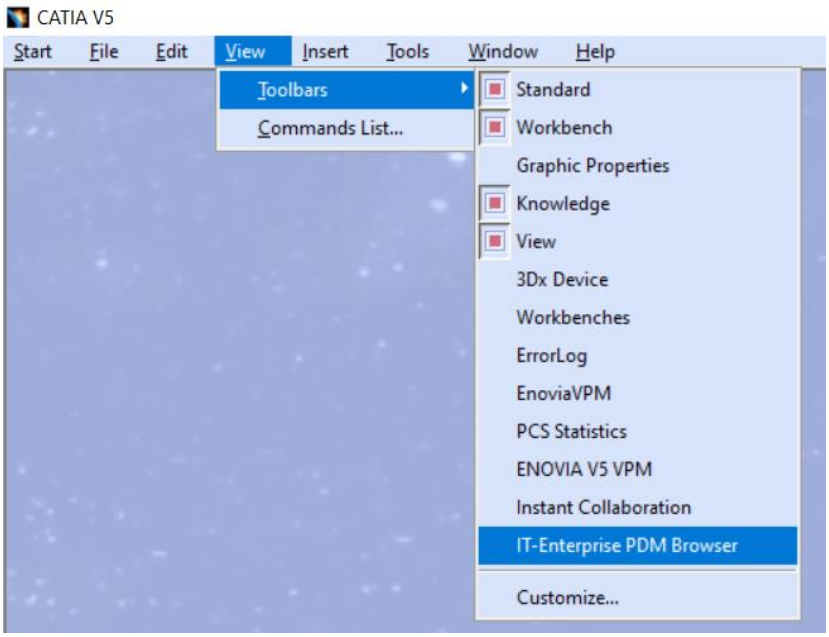

Рисунок 5.6 — Додавання панелі інструментів

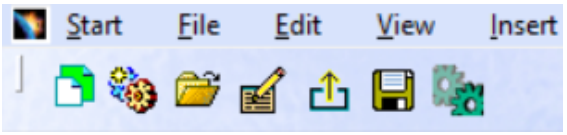

Рисунок 5.7 — Панель інструментів для взаємодії CAD та PDM-систем

Даний спосіб встановлення є дуже зручним, оскільки користувач сам налаштовує панель інструментів з усіма необхідними командами для подальшої взаємодії з ними.

5.2 Робота користувача з програмним забезпеченням

Робота з розробленим програмним забезпеченням передбачає роботу в CAD-системі з панеллю інструментів, на якій розміщуються команди для взаємодії з PDM-системою.

Під час першого входу користувача в систему необхідно пройти авторизацію в системі IT-Enterprise. Перевіряється доступ користувача до PDM-системи і можливість виконання тих чи інших операції. На рисунку 5.8 відображено процес авторизації користувача.

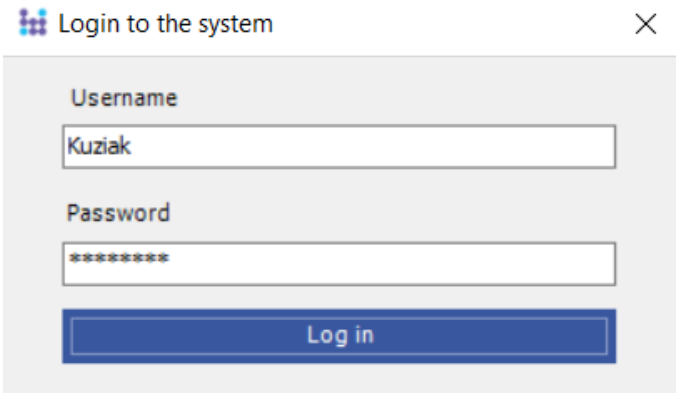

Рисунок 5.8 — Форма авторизації користувача з CAD-системи CATIA

Для того, щоб почати працювати з деталлю, збіркою чи кресленням, які вже є в PDM-системі, необхідно виконати команду «Open». Під час обробки відкриється вікно з веб-браузером, в якому буде відкритий сайт PDM-системи IT-Enterprise в інтерфейсі архіву з фільтром на категорії моделі CATIA (рисунок 5.9).

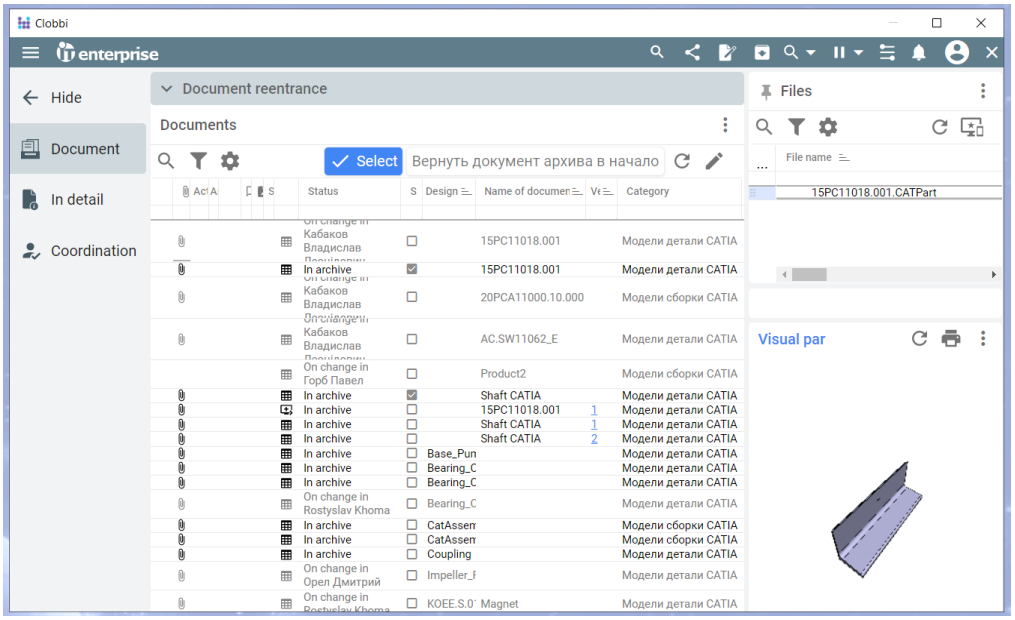

Рисунок 5.9 — Вікно вибору документа під час виконання команди «Відкрити деталь» з PDM-системи

Після вибору дана деталь буде завантажена на комп’ютер користувача і за допомогою API відкрита в поточній сесії додатку CATIA (рисунок 5.10).

Якщо користувач буде створювати абсолютно нову деталь, йому треба вибрати команду «Add by template». Під час обробки програмне забезпечення відкриває вікно вибору шаблону (рисунок 5.11). Після вибору, за допомогою API, буде створено новий файл відповідно до шаблону, вибраного користувачем, і який буде відкрито в сесії CAD-системи CATIA.

Щоб почати роботу з деталлю необхідно взяти її на зміну, щоб інші користувачі бачили, що дана деталь перебуває на зміні в іншого користувача. Для цього використовується команда «Take on editing». Під час виконання ПЗ

перевіряє статус деталі і змінює його на «на зміні в поточного користувача» в разі успіху. У результаті можливі два випадки:

— деталь вже перебуває на зміні в іншого користувача, в такому разі видається повідомлення, що не вдалося взяти деталь на зміну (рисунок 5.12);

— деталь на зміні в поточного користувача чи не на зміні в користувачів, видається повідомлення про успішність операції взяття на зміну (рисунок 5.13).

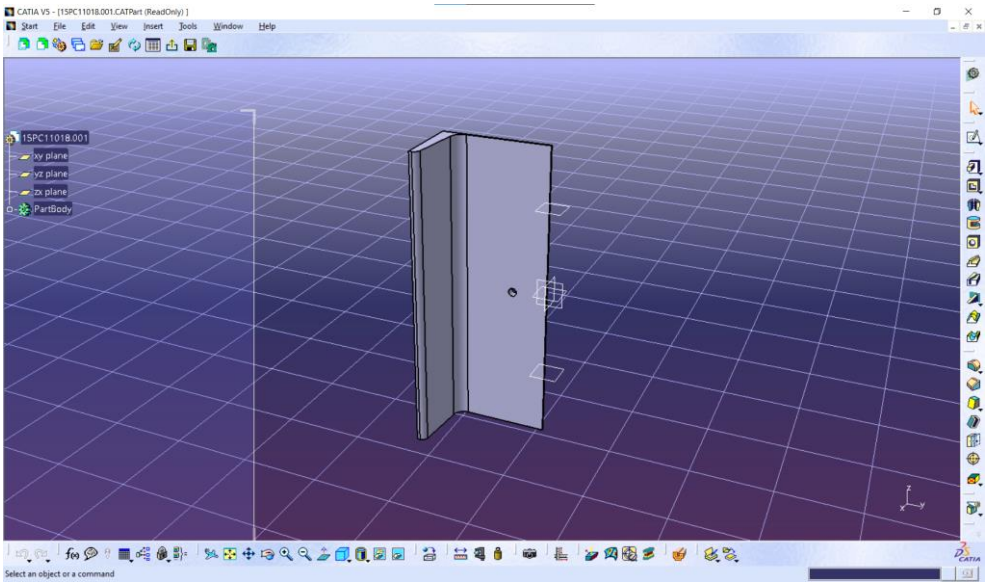

Рисунок 5.10 — Відкрита деталь в CAD-системі

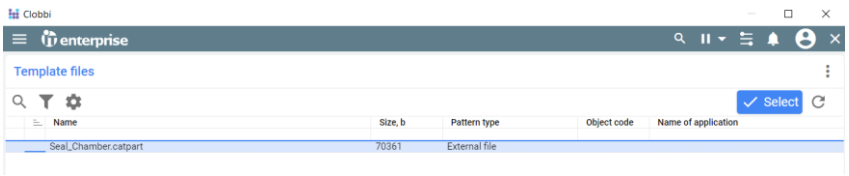

Рисунок 5.11 — Вікно вибору шаблона

Після того, як файл став на зміні в поточного користувача, в ньому можна вносити зміни. Якщо файл — збірка, то одна зі змін, яку можна зробити, це додати компонент. Для того, щоб додати компонент напряму з PDM-системи, потрібно викликати команду «Add components». Під час обробки відбувається перевірка

деталі в активному вікні на те, що це збірка. Після цього відкривається вікно з веб-браузером, в якому є можливість множинного вибору документів (рисунок 5.14).

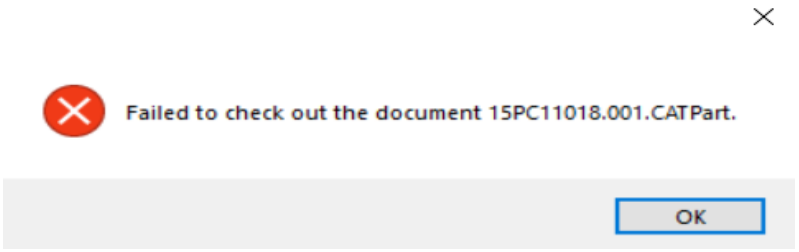

Рисунок 5.12 — Повідомлення про не успішне виконання операції взяття на зміну

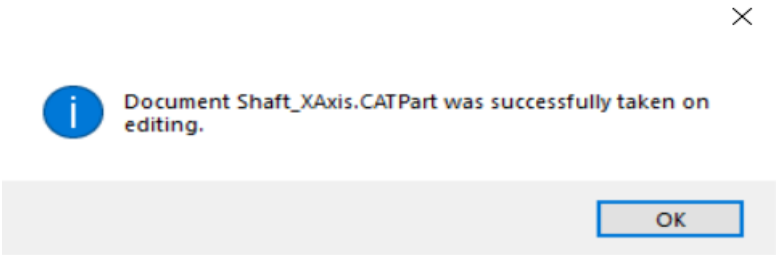

Рисунок 5.13 — Повідомлення про успішне виконання операції взяття на зміну

За допомогою API нові компоненти додаються до відкритої збірки, після чого видається повідомлення про успішне додавання компонентів, які в подальшому можна розмістити на вибір конструктора; додані компоненти зображені на рисунку 5.15.

Після успішних змін документа, його можна зберегти в PDM-систему. Для цього є команда «Save to archive», після виконання якої виводиться повідомлення про успішність чи неуспішність збереження документа (рисунок 5.16). Під час обробки даної команди програмне забезпечення зберігає файл локально, перевіряє, що даний документ перебуває на зміні в поточного користувача, за допомогою API

розгортає дерево зв'язків для збірки чи креслення, створює об'єкти правильної структури, необхідні для передачі та збереження в PDM-системі.

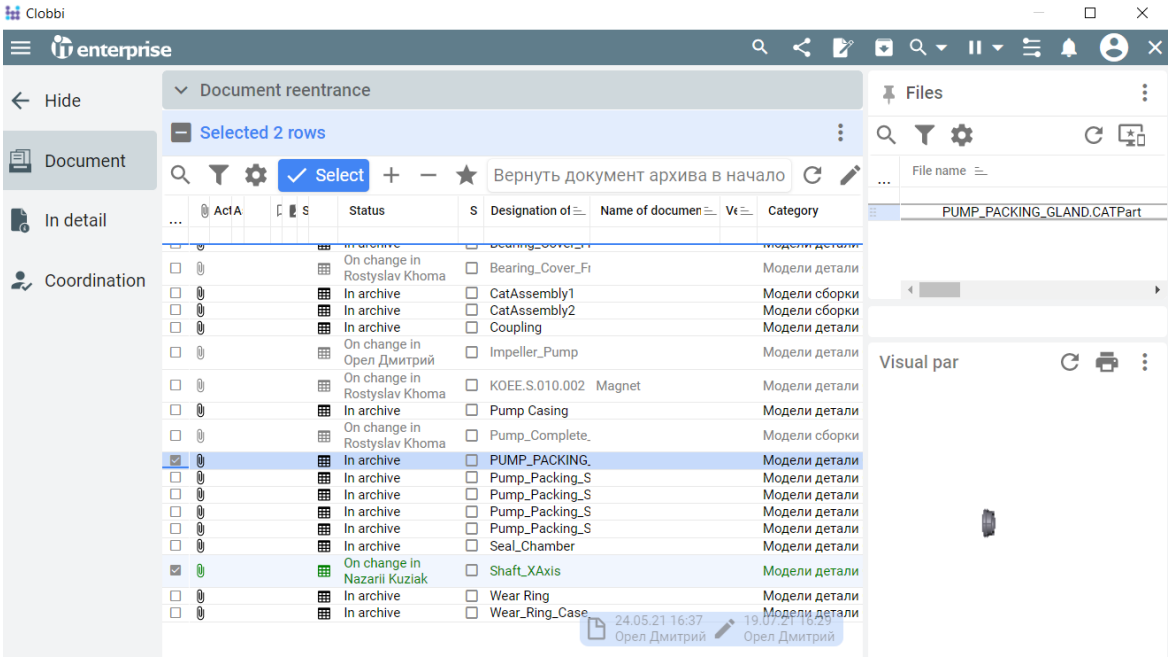

Рисунок 5.14 — Вікно множинного вибору документів

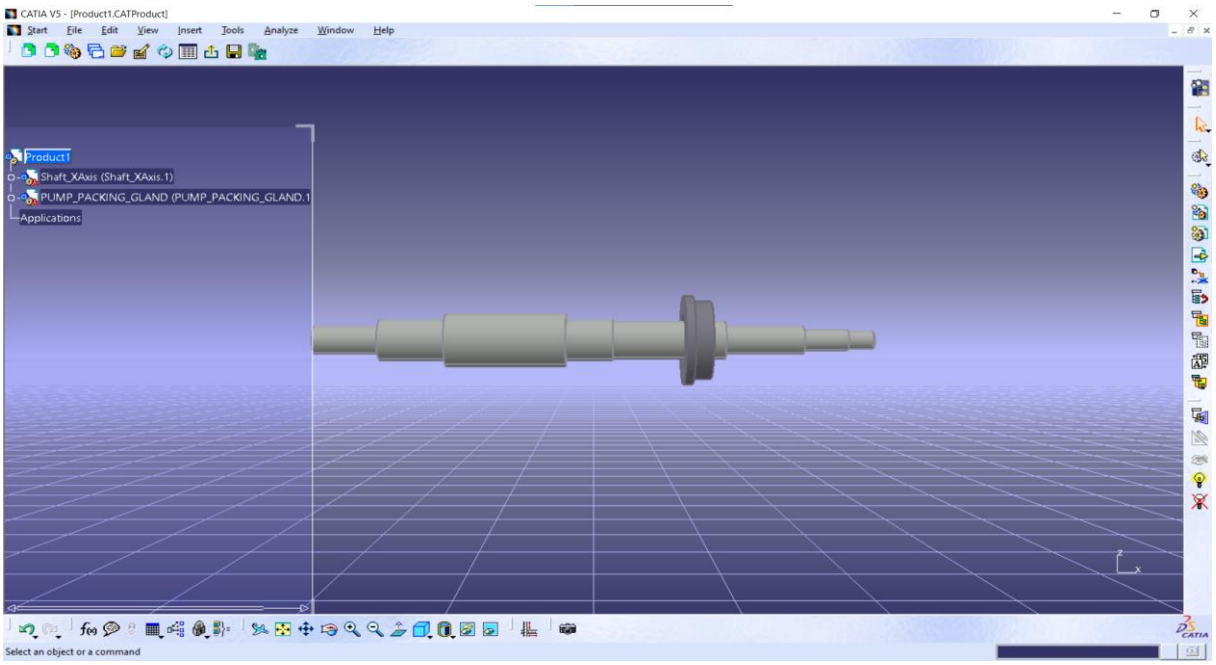

Рисунок 5.15 — CAD-система CATIA після додавання двох нових компонентів

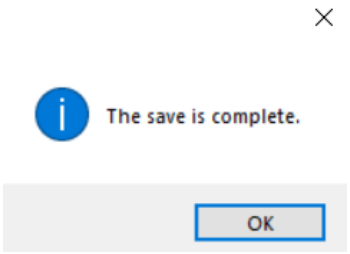

Рисунок 5.16 — Повідомлення про успішне збереження документа в PDM-систему

Якщо конструктор знає, що більше не буде вносити зміни в документ, він може повернути документ в архів. Здійснюється це за допомогою команди «Return to archive», яка спочатку зберігає документ в архіві та змінює його статус на «в архіві».

Висновки до розділу 5

У розділі розглянуто процес встановлення створеного програмного забезпечення. Описано можливі сценарії роботи користувача з ним.

ВИСНОВКИ

У дипломній бакалаврській роботі визначено, що важливим етапом створення нової продукції є етап її проектування. Під час цього етапу створюється велика кількість технічної і конструкторської документації. Для керування даними виробу використовується PDM-система, а для проектування — CAD-система. Взаємодія між цими системами є ключовою ланкою для зменшення часу, необхідного для переходу з етапу проектування до етапу повноцінного виробництва.

Досліджено створення програм для розширення функціоналу різних CAD-систем за допомогою прикладного програмного інтерфейсу (API).

Для взаємодії з PDM-системою IT-Enterprise створено API для виконання поставлених функцій.

Створено розширення CAD-системи CATIA для взаємодії з PDM-системою IT-Enterprise, повністю досліджено процес роботи зі створеним ПЗ. Програмне забезпечення призначене для колективного проектування виробів і швидкого документування їх в систему керування даними про виріб.

Для простої установки ПЗ було створено інсталятор за допомогою розширення середовища розробки Microsoft Visual Studio Installer Projects.

Для створення даного розширення використано такі засоби розробки: середовище розробки Microsoft Visual Studio, мова програмування C#, декларативна мова програмування для взаємодії з базами даних MS SQL.

Було представлено кожен етап зі встановлення ПЗ та розглянуто усі можливості його використання.

Створене розширення CAD-системи CATIA для взаємодії з PDM-системою впроваджено для використання працівниками компанії “IT-Софт”.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 2226-93 Автоматизовані системи. Терміни і визначення. [Чинний від 1994-07-01]. Вид. офіц. Київ: Держспоживстандарт України, 1993. 92 с.
2. Керування даними про продукт, IT-Enterprise: веб-сайт. URL: https://www.it.ua/knowledge-base/technology-innovation/product-data-management-pdm (дата звернення: 07.04.2022).
3. Капінос Г. І., Бабій І. В. Операційний менеджмент: навч. посіб. Київ, 2013. 352 с.
4. Механічне автоматизоване проектування: веб-сайт. URL: https://www.lifecycleinsights.com/tech-guide/mcad/ (дата звернення: 07.04.2022).
5. Автоматизація електронного проектування: веб-сайт. URL: https://www.synopsys.com/glossary/what-is-electronic-design-automation.html (дата звернення: 07.04.2022).
6. Архітектура, інженерія та будівництво: веб-сайт. URL: https://www.autodesk.com/collections/architecture-engineering-construction/overview?term=1-YEAR&tab=subscription (дата звернення: 07.04.2022).
7. Технології безперервного придбання та підтримки життєвого циклу: веб-сайт. URL: https://stud.com.ua/10906/menedzhment/cals_tehnologiyi (дата звернення: 07.04.2022).
8. Гетьман А. Ю. Проектування систем автоматизації: конспект лекцій. Краматорськ : ДДМА, 2017, 79 с.
9. Керування даними про продукт: веб-сайт. URL: https://www.it.ua/products/rd/upravlenie-dannymi-ob-izdelii-pdm (дата звернення: 08.04.2022).

10. Керування інженерними даними: веб-сайт. URL: https://www.technia.com/blog/7-steps-to-successful-engineering-data-management/ (дата звернення: 07.04.2022).

11. Керування інформацією про продукт: веб-сайт. URL: https://www.akeneo.com/what-is-a-pim/ (дата звернення: 08.04.2022).

12. Керування технічними даними: веб-сайт. URL: https://www.nasa.gov/seh/6-6-technical-data-management (дата звернення: 08.04.2022).

13. Andrew Troelsen, Philip Japikse Pro C# 7: Minneapolis, Minnesota, USA: 2018, 1410 p.

14. CREO VB API documentation: Parametric Technology Corporation, Needham, USA: 2011, 506 p.

15. CATIA API documentation: веб-сайт. URL: http://catiadoc.free.fr/online/interfaces/main.htm (дата звернення 08.04.2022).

16. SolidWorks API documentation: веб-сайт. URL: https://help.solidworks.com/2021/English/SolidWorks/sldworks/c_solidworks_api.htm (дата звернення 08.04.2022).

17. Розробка програмного забезпечення: веб-сайт. URL: https://uk.theastrologypage.com/software-development (дата звернення: 10.04.2022).

18. Середовище розробки Microsoft Visual Studio: веб-сайт. URL: https://informatics.in.ua/programming_csharp/part_01.php (дата звернення: 08.04.2022).

19. Windows Communication Foundation: веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/wcf/whats-wcf (дата звернення: 10.04.2022).

20. Windows Presentation Foundation: веб-сайт. URL: https://docs.microsoft.com/ru-ru/visualstudio/designers/getting-started-with-wpf?view=vs-2022 (дата звернення: 10.04.2022).

21. Microsoft SQL Server: веб-сайт. URL: https://kurspc.com.ua/node/333 (дата звернення: 10.04.2022).

22. CEFSharp: веб-сайт. URL: https://cefsharp.github.io/ (дата звернення: 10.04.2022).

23. Windows Installer: веб-сайт. URL: https://docs.microsoft.com/en-us/windows/win32/msi/windows-installer-portal (дата звернення 10.04.2022)

24. Visual Studio Installer: веб-сайт. URL: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/2kt85ked(v%3dvs.100) (дата звернення: 10.04.2022).

ДОДАТОК А

Розширення функціоналу CAD-системи для взаємодії з PDM-системою

Текст програмного модуля

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТР81335_22Б 12-1

Аркушів 14

2022

Макрос в системі CATIA:

```
Sub CATMain()

Dim hReq As Object
Set hReq =
CreateObject("ITEnterprisePDMBrowser.CATIA.CADConnector.Application")

hReq.Open CATIA.Application

End Sub
```

Код виконання команди відкриття документа з PDM-системи:

```
using ITEnterprisePDMBrowser.Common.Dialogs;
using ITEnterprisePDMBrowser.Common.Entities;
using ITEnterprisePDMBrowser.Common.ITEnterprise;
using ITEnterprisePDMBrowser.Common.Requests;
using ITEnterprisePDMBrowser.Common.Tools;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using ITEnterprisePDMBrowser.Common.Properties;
using ITEnterprisePDMBrowser.Common.Services;

namespace ITEnterprisePDMBrowser.Common.Commands
{
	internal class GetDocument : CommandHandlerBase<GetDocumentParams>
	{
		public GetDocument(GetDocumentParams parameters) : base(parameters)
		{
		}

		protected override bool OnCommand()
		{
			//Check if user has access to IT-Enterprise
			var ticket = ITEnterpriseConnector.TryLogIn();
			if (string.IsNullOrEmpty(ticket))
			{
				return false;
			}
```

```
                var activeDocFilePath = Params.Strategy.GetActiveDocFilePath();

                List<string> documentCodes = null;
                if (Params.DocumentCodes == null)
                {
                        //open window with IT-Enterprise client to choose document
to open
                        var guid = Guid.NewGuid().ToString();
                        Log.Info(string.Format("BeforeCallClient({0})", guid));
                        if (!ITEnterpriseConnector.CallCLient(Params.Multi ?
Const.SelectDocumentsKeyMenu : Const.SelectDocumentKeyMenu, guid))
                        {
                                Log.Info("CallClient returned false");
                                return false;
                        }
                        Log.Info(string.Format("AfterCallClient({0})", guid));
                        documentCodes = GetParamsByGuid<List<string>>(guid);
                        Log.Info(string.Format("AfterGetParamsByGuid({0})",
                                documentCodes != null ? string.Join(",",
documentCodes) : string.Empty));
                }
                else
                {
                        documentCodes = Params.DocumentCodes;
                }

                if (documentCodes == null || documentCodes.Count == 0)
                {
                        return false;
                }

                InfoManager.ShowProgress(25, Resources.DocumentLoading);

                //get documents from IT-Enterprise by document codes
                var requestResult = new GetCadDocumentsRequest().GetResult(new
ChangeCadDocsRequestParams(documentCodes), ticket);
                var models = ProcessResult<List<ICadDocument>>(requestResult);
                if (models == null || models.Count == 0)
                {

        InfoManager.ShowMessageBox(Resources.ErrorSelectFileFromArchive,
messageBoxIcon: IconType.Error);
                        return false;
```

```
                }

                var allModelsFilePaths = new List<string>();
                var readOnlyModelsFilePaths = new List<string>();
                var modelsOnChangingThatExistsFileNames = new List<string>();
                foreach (var model in models.Expand())
{
                        allModelsFilePaths.Add(model.FullLocalFilePath);
                        if(model.IsOnChanging &&
existsInWorkDir(model.FullLocalFilePath))
{

        modelsOnChangingThatExistsFileNames.Add(model.FileName);
}
else
{

        readOnlyModelsFilePaths.Add(model.FullLocalFilePath);
                        }
}

                var rewrite = false;
                if (modelsOnChangingThatExistsFileNames.Count > 0)
                {
                        var message =
modelsOnChangingThatExistsFileNames.Count == 1
                                ? string.Format(Resources.FileExistsInWorkDir, "\"" +
modelsOnChangingThatExistsFileNames.First() + "\"")
                                : string.Format(Resources.FilesExistsInWorkDir,
string.Join(",", modelsOnChangingThatExistsFileNames));
                        rewrite = InfoManager.ShowYesNo($"{message}
{Resources.Replace}");
                }

                if(modelsOnChangingThatExistsFileNames.Count ==
allModelsFilePaths.Count && !rewrite)
{
                        return true;
}

                //find documents that should be reopened
                var documentsToCloseFilePaths = new List<string>();
if (rewrite)
```

```
{

        documentsToCloseFilePaths.AddRange(Params.Strategy.GetOpenedDocumentsF
ilePaths(allModelsFilePaths));
}
else
{

        documentsToCloseFilePaths.AddRange(Params.Strategy.GetOpenedDocumentsF
ilePaths(readOnlyModelsFilePaths));
}

                        //close files
                        foreach(var filePath in documentsToCloseFilePaths)
{
                                Params.Strategy.QuitDocument(filePath);
}

                        InfoManager.ShowProgress(50, Resources.Saving);

                        var success = false;

                        var modelCreator = new ModelCreator();
                        modelCreator.CreateModelCADStrategy +=
Params.Strategy.CreateModel;

                        InfoManager.EndProgress();

                        //open files
                        foreach (var model in models)
                        {
                                success |= modelCreator.CreateModelAfterGetDocument(new
CreateModelAfterGetDocumentParams(model, rewrite,
                                        Params.AppendCurrentAssembly ||
!documentCodes.Contains(model.DocumentCode),
                                        Params.Strategy.GetCurrentWorkingDirectory()));
                        }

                        modelCreator.CreateModelCADStrategy -=
Params.Strategy.CreateModel;

                        //open files in CATIA session
                        foreach(var filePath in documentsToCloseFilePaths)
```

```
{
                        Params.Strategy.OpenModel(filePath);
}

                //make the document active in session
if (!string.IsNullOrEmpty(activeDocFilePath) && Params.AppendCurrentAssembly)
{
                        Params.Strategy.ActivateDoc(activeDocFilePath);
}
else
{

        Params.Strategy.ActivateDoc(models.First().FullLocalFilePath);
                }

                if (!success)
                {
                        InfoManager.ShowMessageBox(Resources.OpenFileError,
messageBoxIcon: IconType.Error);
                        return false;
                }

                return true;
        }

        private bool existsInWorkDir(string filePath)
        {
                FileInfo fileInfo;
                try
                {
                        fileInfo = new FileInfo(filePath);
                }
                catch
                {
                        return false;
                }
                return fileInfo.Exists && !fileInfo.IsReadOnly;
        }
    }
}

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using ITEnterprisePDMBrowser.CATIA.CADConnector.Extensions;
using ITEnterprisePDMBrowser.Common.Commands;
using ITEnterprisePDMBrowser.Common.Entities;
using ITEnterprisePDMBrowser.Common.Tools;

namespace ITEnterprisePDMBrowser.CATIA.CADConnector.Strategies
{
	class GetDocumentStrategy : IGetDocumentCADStrategy
	{
		//main CATIA object
		INFITF.Application _catiaApp;
		public GetDocumentStrategy(INFITF.Application catiaApp)
		{
			_catiaApp = catiaApp;
		}

		//function that add components to current assembly
		public bool
AddComponentsToCurrentAssembly(IEnumerable<ICadDocument> docs)
		{
			//check if opened document is assembly
			if (_catiaApp.ActiveDocument is
ProductStructureTypeLib.ProductDocument product)
			{
				var iMethod = "All";
				var documents = docs.ToArray();
				var components = Array.CreateInstance(typeof(object),
documents.Length);
				var allowedExtensions = new HashSet<string> {
"CATPART", "CATPRODUCT" };
				for (var i = 0; i < documents.Length; i++)
				{
					var docExtension =
FileManager.GetFileExtension(documents[i].FullLocalFilePath).ToUpperInvariant();
					if (allowedExtensions.Contains(docExtension))
					{

	components.SetValue(documents[i].FullLocalFilePath, i);
					}
				}
```

```
        product.Product.Products.AddComponentsFromFiles(components, ref iMethod);
                        _catiaApp.ActiveWindow.ActiveViewer.Update();
                        _catiaApp.ActiveWindow.ActiveViewer.Reframe();
                        _catiaApp.ActiveDocument.SaveDocument(false);
                }

                return true;
        }

        //check if active document is assembly
        public bool CheckActiveDocumentIsAssembly()
        {
                return _catiaApp.HasActiveDocument() &&
_catiaApp.ActiveDocument is ProductStructureTypeLib.ProductDocument;
        }

        //get CATIA working directory
        public string GetCurrentWorkingDirectory()
        {
                return _catiaApp.GetCurrentWorkingDirectory();
        }

        //open document in CATIA session
        public bool CreateModel(ICadDocument cadDocument, bool isReadOnly,
bool openSilently)
        {
                Log.Info("Start GetDocumentStrategy.CreateModel method");

                var path = cadDocument.FullLocalFilePath;
                var document = _catiaApp.Documents.Open(ref path);
                if(document == null)
                {
                        return false;
                }

                if (!isReadOnly)
                {
                        document.FillAllModelProperties(cadDocument.Properties);
                }

                //if document should be opened silently it should be closed
                if (openSilently)
```

```
                {
                        document.Close();
                }
                else
                {
                        _catiaApp.ActiveWindow.ActiveViewer.Reframe();
                }

                return true;
        }

        //get file path of opened document
        public string GetActiveDocFilePath()
        {
                return _catiaApp.GetActiveDocumentFilePath();
        }

        //get file paths of documents that should be reopened after downloadind
from PDM system
        public IEnumerable<string>
GetOpenedDocumentsFilePaths(IEnumerable<string> filePathsToSearch)
        {
                var fileNamesToSearch = filePathsToSearch.Select(filePath =>
FileManager.GetFileName(filePath));
                var result = new List<string>();

                foreach(var document in _catiaApp.GetOpenedDocuments())
                {
                        var openedModelFilePath = document.FullName;
                        if(!result.Contains(openedModelFilePath) &&
(fileNamesToSearch.Contains(FileManager.GetFileName(openedModelFilePath,
FileManager.GetFileNameType.WithExtensionWithoutVersion)) ||
                                _catiaApp.HasOpenedReferencedFile(new
List<INFITF.Document>() { document }, fileNamesToSearch.ToList(), new
List<string>())))
 {
                                result.Add(document.FullName);
                        }
                }

                return result;
        }
```

```
            //close document by file path
            public void QuitDocument(string filePath)
            {
                _catiaApp.QuitFile(filePath);
            }

            //open document by file path
            public void OpenModel(string filePath)
            {
                _catiaApp.OpenFile(filePath);
            }

            //make active document by file path
            public void ActivateDoc(string filePath)
            {
                _catiaApp.ActivateDocument(filePath);
            }
        }
}
```

Код в системі IT-Enterprise:

```
using System;
using System.Collections.Generic;
using System.Linq;
using ITnet2.Common.Tools;
using ITnet2.Server.BusinessLogic.Core.Tools;
using ITnet2.Server.BusinessLogic.MP.PLAN;
using ITnet2.Server.BusinessLogic.MP.TDM.Configuration;
using ITnet2.Server.BusinessLogic.MP.TDM.Documents;
using ITnet2.Server.BusinessLogic.MP.TDM.FilesSystem;
using ITnet2.Server.BusinessLogic.MP.TDM.PdmBrowser;
using ITnet2.Server.BusinessLogic.Scenario.Properties;
using ITnet2.Server.Data;

namespace ITnet2.Server.BusinessLogic.Scenario.TDM
{
    internal class PdmBrowserGetModels :
PdmBrowserBase<PdmBrowserGetModelsParams, IChangeCadDocsRequestParams>
    {
        private readonly Lazy<IDocumentsTraverser> _documentsTraversor =
new Lazy<IDocumentsTraverser>(GetService<IDocumentsTraverser>);
```

```
        private Documents _documents;

        protected override bool VerifyCalcInput()
        {
            if (!base.VerifyCalcInput())
            {
                return false;
            }

            var documentCodes = new HashSet<string>();
            if (RequestParams.DocumentCodes != null)
            {
                documentCodes.AddRange(RequestParams.DocumentCodes);
            }
            else
            {
                if (!string.IsNullOrEmpty(RequestParams.FilePath))
                {
                    var docCode =
GetDocumentCodesByFilePath(RequestParams.FilePath).FirstOrDefault();
                    if (!string.IsNullOrEmpty(docCode))
                    {
                        documentCodes.Add(docCode);
                    }
                }

                if (RequestParams.FilePaths != null &&
RequestParams.FilePaths.Count > 0)
                {
                    if (RequestParams.FilePaths.Count == 1)
                    {
                        var docCode =
GetDocumentCodesByFilePath(RequestParams.FilePaths.FirstOrDefault()).FirstOrDefa
ult();
                        if (!string.IsNullOrEmpty(docCode))
                        {
                            documentCodes.Add(docCode);
                        }
                    }
                    else
                    {
                        var docCodes =
GetDocumentCodesByFilePaths(RequestParams.FilePaths);
```

```
                                        foreach (var docCode in docCodes.Where(d =>
!string.IsNullOrEmpty(d)))
                                        {
                                                documentCodes.Add(docCode);
                                        }
                                }
                        }
                }

                if (documentCodes.Count == 0)
                {
                        return false;
                }

                _documents = DocumentsRepository.Value.Retrieve(new
DocumentsCollectionParams(documentCodes) {
                        Columns = new ColumnCollection("TDDS", new List<string>
{ "STATUS", "USERID_COR", "TTDV", "NOM_IZM" }),
                        FilesCollectionParams = new FilesCollectionParams {
LoadContent = false, RetrieveFileFolders = true },
                        DocumentsCardCollectionParams = new
DocumentsCardCollectionParams {
                                Columns = new ColumnCollection("TDD", new
List<string> { "TTDK" })
                        }
                });

                if (_documents.Count == 0)
                {
                        WriteResult(false, null, Resources.LoadDocumentsError);
                        return false;
                }
                return true;
        }

        protected override bool Exec()
        {
                var result = new List<ICadDocument>();
                foreach (var document in _documents)
                {
                        var model = CreateModelByDocument(document);
                        if (model != null)
                        {
```

```
                                result.Add(model);
                                var modelWithReferences = model as
ICadDocumentWithReferences;
                                if (modelWithReferences != null)
                                {

        modelWithReferences.AddReferences(expandDocument(document.DocumentCo
de).OfType<ICadModel>());
                                }


        //result.AddRange(findDrawings(model.DocumentCode));
                        }
                }

                if (result.Count == 0)
                {
                        WriteResult(false, null, Resources.LoadDocumentsError);
                        return false;
                }

                var models = ExpandTree(result);
                FillModelsProperties(models);

                WriteResult(true, result);
                return true;
        }

        private List<ICadDocument> expandDocument(string documentCode)
        {
                var result = new List<ICadDocument>();

                var documents =
_documentsTraversor.Value.GetChildDocuments(new
ChildDocumentsCollectionParams(documentCode, RelationTypesCodes)
                {
                        Columns = new ColumnCollection("TDDS", new List<string>
{ "STATUS", "USERID_COR", "TTDV", "NOM_IZM" }),
                        FilesCollectionParams = new FilesCollectionParams {
LoadContent = false, RetrieveFileFolders = true },
                        DocumentsCardCollectionParams = new
DocumentsCardCollectionParams
                        {
```

```
                                Columns = new ColumnCollection("TDD", new
List<string> { "TTDK" })
                        }
                });
                foreach (var document in documents)
                {
                        var model = CreateModelByDocument(document);
                        if (model != null)
                        {
                                result.Add(model);
                        }
                }
                return result;
        }

        private List<ICadDocument> findDrawings(string documentCode)
        {
                var result = new List<ICadDocument>();

                var drawingsRelations =
DocumentsRelationRepository.Value.Retrieve(new
DocumentsRelationCollectionParams {
                        ChildDocumentCode = documentCode,
                        RelationType = "Ч"
                });

                if (drawingsRelations.Count == 0)
                {
                        return result;
                }

                var documents = DocumentsRepository.Value.Retrieve(new
DocumentsCollectionParams(new HashSet<string>(drawingsRelations.Select(r =>
r.ParentDocumentCode))) {
                        Columns = new ColumnCollection("TDDS", new List<string>
{ "STATUS", "USERID_COR", "TTDV", "NOM_IZM" }),
                        FilesCollectionParams = new FilesCollectionParams {
LoadContent = false, RetrieveFileFolders = true },
                        DocumentsCardCollectionParams = new
DocumentsCardCollectionParams
                        {
                                Columns = new ColumnCollection("TDD", new
List<string> { "TTDK" })
```

```
                }
            });

            foreach (var document in documents)
            {
                var model = CreateModelByDocument(document);
                if (model != null)
                {
                    result.Add(model);
                }
            }
            return result;
        }
    }
}
```

ДОДАТОК Б

Розширення функціоналу CAD-системи для взаємодії з PDM-системою

АПРОБАЦІЯ

АКТ ВПРОВАДЖЕННЯ ТОВ “ІТ-Софт”

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТР81335_22Б

Аркушів 1

2022

Затверджую

Директор ТОВ «IT-Софт»

_____ Михайлов В.В.

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи спеціаліста Кузяк Н. І. на тему “Розширення функціоналу CAD-системи для взаємодії з PDM-системою”, яка виконана в Національному технічному університеті України “Київський політехнічний інститут” (НТУУ “КПІ”).

«___» _____ 2022 р.

Нами, представниками кафедри автоматизації проектування енергетичних процесів і систем НТУУ “КПІ”, та представниками ТОВ "IT - Софт", складено даний акт про те, що для використання в розробках спеціалізованого програмного забезпечення ТОВ "IT- Софт" прийняті результати дипломної роботи бакалавра Кузяка Н. І., а саме програмне забезпечення – розширення CAD-системи для взаємодії з PDM-системою, а також документацію програмного супроводу.

Представник кафедри АПЕПС НТУУ “КПІ”

Керівник дипломної роботи _____ Кублій Л. І.

Представник ТОВ «IT-Софт» _____ Михайлов В.В.

Директор

«IT-Софт» Ідентифікаційний код 31720124