Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Каменєва Іллі Федоровича*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для розширення бібліотек нових виробів в найбільш поширених CAD - системах*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-17-1* (група) *Каменєва Іллі Федоровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для розширення бібліотек нових виробів в найбільш поширених CAD - системах*

затверджена наказом ректора НТУ «ДП» від *07.06.2021р.* № *317-С*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ (підпис) *доц. Спірінцев В.В* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Каменєв І.Ф.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 79 с., 16 рис., 2 табл., 3дод., 18 джерела.

Об'єкт розробки: програмний модуль для розширення бібліотек нових виробів в найбільш поширених CAD-системах (КОМПАС-3D, Autodesk Inventor, SolidWorks) за рахунок оновлення номенклатури виробів на базі вже розроблених вузлів (модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних) з урахуванням специфіки підприємства.

Мета кваліфікаційної роботи: розробка програмного забезпечення з використанням API інтерфейсу для розширення бібліотек нових виробів в найбільш поширених системах автоматизованого проектування: КОМПАС-3D, Autodesk Inventor, SolidWorks.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку програмного модулю, описана робота додатку, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленого програмного забезпечення, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні програмного забезпечення, що дозволить: розширити бібліотеку нових виробів та вирішити завдання розрахунку і проектування конкретного класу виробів враховуючи специфіку підприємства; оновити номенклатуру виробів на базі вже розроблених вузлів за рахунок модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних. При цьому необхідно об'єднати розрахунковий модуль, який визначає розмірні та інші параметри проектованого об'єкта з уже наявним в САПР тривимірним геометричним ядром. Цей процес здійснюється за допомогою API - інтерфейсу прикладної програми.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, що дозволяють підвищити продуктивність праці проектувальників (за рахунок звільнення проектувальника від проведення багатьох рутинних операцій), скоротити терміни проектування та знизити витрати на розробку технічної документації, збільшити якість проектування за рахунок підвищення точності виготовлення виробів.

Список ключових слів: САПР, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, C#, API, БІЛІОТЕКА КЛАСІВ, КОМПАС-3D, AUTODESK INVENTOR, SOLIDWORKS.

ABSTRACT

Explanatory note: 79 pp., 16 fig., 2 table, 3 extra, 18 sources.

Development object: software module for expansion of libraries of new products in the most common CAD systems (KOMPAS-3D, Autodesk Inventor, SolidWorks) by updating the product range based on already developed nodes (modification of the previously created node geometry in accordance with new design data), taking into account the specifics of the enterprise.

Purpose of qualification work: software development using the API interface to expand libraries of new products in the most common CAD systems KOMPAS-3D, Autodesk Inventor, SolidWorks.

The introduction examines the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic and clarifies the problem statement.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the statement of the problem is formulated, the requirements for software implementation, technologies and software are indicated.

In the second section, the existing solutions are analyzed, the platform for development is selected, the design and development of the software module is carried out, the operation of the application, the algorithm and structure of its functioning, as well as calling and loading the application are described, the input and output data are determined, the composition of the parameters of the technical means is characterized.

In the economic section, the complexity of the developed software is determined, the cost of the work on creating the application is calculated and the time for its creation is calculated.

The practical value lies in the creation of software that will allow: expand the library of new products and carry out the task of calculating and designing a specific class of products, taking into account the specifics of the enterprise; update the product range based on already developed assemblies by modifying the previously created geometry of the assembly in accordance with the new design data. In this case, it is necessary to combine the calculation module that determines the dimensional and other parameters of the designed object with the 3D geometric kernel already available in the CAD. This process is carried out using the API - Application Programming Interface.

Relevance of this software product is determined by the great demand for such developments, which make it possible to increase the productivity of designers (by freeing the designer from carrying out many routine operations), reduce the design time and reduce the cost of developing technical documentation, and increase the design quality by increasing the accuracy of manufacturing products.

Keyword list: CAD, SOFTWARE, C #, API, CLASS LIBRARY, КОМПАС-3D, AUTODESK INVENTOR, SOLIDWORKS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЖЦ - Життєвий цикл

САПР - Система Автоматизованого Проектування

ІС - Інформаційна система

ПЗ - Програмне забезпечення

API - Application Programming Interface

CAD - Computer Aided Design

COM - Component Object Model

ВСТУП

Прискорення темпів науково-технічного прогресу є вирішальною умовою підвищення якості продукції. Об'єктивною перешкодою підвищенню якості проектів і скороченню термінів їх розробки є невідповідність між складністю об'єктів виробництва і застарілими методами і засобами їх проектування.

Впровадження сучасних комп'ютерних технологій на промислових підприємствах України дозволяє досягти значних успіхів на ринку машинобудування в умовах жорсткої конкуренції. Автоматизація підготовки виробництва дає можливість підприємствам швидко реагувати на зміну попиту, у короткий термін випускати нові види продукції, швидко модернізувати випускаєму продукцію, відслідковувати життєвий цикл виробів, ефективно підвищувати якість. Підприємства, що не використовують інформаційні технології, виявляються не конкурентоздатними внаслідок як великих матеріальних і тимчасових витрат на проектування, так і невисокої якості проектів, що може привести до їх банкрутсва.

Найбільш прогресивною та перспективною умовою удосконалення процесу проектування є створення і впровадження в практику систем автоматизованого проектування (САПР), забезпечених сучасними ПЕОМ з розвиненими термінальними системами.

На багатьох підприємствах широко розвернувся процес впровадження різних підсистем САПР (CAD/CAM-систем), що дозволяють автоматизувати весь цикл: від розробки виробів до проектування технології їх виготовлення.

Метою кваліфікаційної роботи є розробка програмного забезпечення з використанням API інтерфейсу для розширення бібліотек нових виробів в найбільш поширених системах автоматизованого проектування: КОМПАС-3D, Autodesk Inventor, SolidWorks. Побудова програмного модуля буде реалізована за допомогою програмного пакету C#.

Об'єкт розробки: програмний модуль для розширення бібліотек нових виробів в найбільш поширених CAD-системах (КОМПАС-3D, Autodesk

Inventor, SolidWorks) за рахунок оновлення номенклатури виробів на базі вже розроблених вузлів (модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних) з урахуванням специфіки підприємства.

Практичне значення отриманих результатів полягає в наданні проектувальнику можливостей підвищити продуктивність праці (за рахунок звільнення проектувальника від проведення багатьох рутинних операцій), скороченні термінів проектування та витрат на розробку технічної документації, збільшенні якості проектування за рахунок підвищення точності виготовлення виробів.

Для досягнення поставленої мети в роботі необхідно виконати наступні задачі:

- здійснити аналіз предметної галузі з метою виявлення сучасного стану обраної проблематики;
- здійснити аналіз існуючих CAD-систем з метою виявлення їх властивостей та потенційних можливостей;
- розглянути існуючий механізм взаємодії систем автоматизованого проектування із програмним модулем;
- здійснити вибір інструментальних засобів для розробки модуля API;
- розробити алгоритм роботи програмного модуля САПР;
- розробити спеціалізовану САПР на базі API найпоширеніших CAD – систем (КОМПАС-3D, Autodesk Inventor, SolidWorks) для розширення бібліотек нових виробів;
- розробити інструкцію користувача розробленого програмного модуля.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Гнучкі виробничі системи та комп'ютеризовані інтегровані виробництва

Починаючи з 1980-х рр., одним з напрямків підвищення ефективності виробництва стало широке застосування інформаційних технологій. Важливим етапом розвитку на цьому шляху були розробка й застосування гнучких виробничих систем. Гнучка виробнича система (ГВС) - це керована засобами обчислювальної техніки сукупність технологічного встаткування, що складається з різних сполучень гнучких виробничих модулів й/або гнучких виробничих осередків, автоматизованої системи технологічної підготовки виробництва і системи забезпечення його функціонування [3,4]. Така система має важливу властивість автоматизованого переналагодження при зміні програми виробництва виробів. Принциповою особливістю ГВС є наявність нового компонента - комп'ютерного блоку управління, що забезпечує можливість ув'язування окремих процесів, функцій і завдань у єдину систему.

Впровадження ГВС призвело до зменшення розмірів підприємств, збільшенню коефіцієнта використання встаткування й зниженню накладних витрат, значному зменшенню обсягу незавершеного виробництва, скороченню витрат на робочу силу, прискоренню змінюваності моделей випускаємої продукції, що, відповідно до вимог ринку й скороченню строків поставок продукції й підвищенню її якості.

Подальший розвиток робіт у даному напрямку наприкінці 1980-х - початку 1990-х рр. привів до створення комп'ютеризованого інтегрованого виробництва. Його концепція складається в новому підході до організації й управління виробництвом, що полягає не тільки в застосуванні комп'ютерних

технологій для автоматизації технологічних процесів й операцій, але й у створенні інтегрованої інформаційної системи підприємства. Інформаційна інтеграція процесів досягалася шляхом використання загальних баз даних (БД), що дозволяють більш ефективно вирішувати питання розробки й проектування виробів, підготовки виробництва, планування й управління виробництвом, рішення завдань матеріально-технічного забезпечення, що охоплює всі сторони діяльності підприємства.

У концепції комп'ютеризованого інтегрованого виробництва роль інтегрованої автоматизованої системи управління стала ще більш вагомою. На неї були покладені не тільки функції автоматизації процесів проектування й виробництва виробів, але й зовсім нові завдання, пов'язані із забезпеченням інформаційної інтеграції процесів. Така інтеграція могла здійснюватися за рахунок спільного використання однієї й тієї ж інформації (в електронному виді) для рішення різних завдань.

Практика показала, що із всіх завдань інтегрованої автоматизованої системи управління найбільш типізованими виявилися завдання автоматизації проектування і підготовки виробництва, а також завдання управління підприємством. На світовому ринку з'явилися самостійні програмно-технічні рішення, придатні для використання на підприємствах з різним рівнем автоматизації. Виникли нові автоматизовані системи: CAD/CAM/CAE і MRP (MRP II).

Перша система - CAD/CAM/CAE - це комплекс програмних засобів комп'ютерного проектування, підготовки виробництва й інженерних розрахунків.

Друга система - MRP (Materials Requirement Planning) - призначена для планування потреб у матеріалах, а вдосконалений варіант цієї системи - MRP II (Manufacturing Resource Planning) - став загальноприйнятим засобом рішення комплексу завдань управління фінансово-господарською діяльністю підприємства: планування виробництва, матеріально-технічного постачання,

управління фінансовими ресурсами й ін. З'явилися перші стандарти й специфікації, що визначають функціональні вимоги до зазначених систем.

На початку 1990-х рр. консалтинговою фірмою Gartner Group (США) була запропонована концепція ERP (Enterprise Resource Planning - управління ресурсами підприємства). У цей час терміни MRP II й ERP поряд з терміном інтегрованої автоматизованої системи управління стали звичним для фахівців позначенням класу інтегрованих інформаційних систем, призначених для управління виробничо-господарською діяльністю підприємства.

Концепція комп'ютеризованого інтегрованого виробництва стала важливим етапом розвитку промислових інформаційних технологій. На цій стадії розвитку виник і був частково апробований ряд фундаментальних ідей, принципів і технологій:

1. Сформувався клас систем автоматизації праці в процесах розробки виробу й підготовки виробництва. На перших етапах це були завдання автоматизації створення традиційної (паперової) конструкторської документації. За допомогою автоматизованих систем проектування (CAD) створювалося електронне креслення - плоска геометрична модель виробу. Згодом почалося використання поверхневих і твердотільних об'ємних моделей компонентів виробу. Необхідність забезпечення сумісності таких геометричних моделей, розроблювальних за допомогою різних програмних систем, обумовила стандартизацію форматів даних.

2. На основі конструкторських геометричних моделей виробу за допомогою автоматизованих систем технологічної підготовки виробництва (CAM) розроблялися програми для верстатів із числовим програмним управлінням. Обмін геометричними даними в електронному виді між системами CAD і CAM став одним з перших реальних прикладів інформаційної інтеграції процесів.

3. Виникнення систем класу MRP II, що володіють певним набором функцій і взаємозв'язків між цими функціями, створило основу для формування функціонального стандарту, що регламентує загальноприйняту управлінську

технологію, реалізовану з використанням комп'ютерних систем. Характерною рисою цієї технології з'явилося спільне використання загальних БД при виконанні різних процесів.

4. У процесі комп'ютеризованого інтегрованого виробництва вперше не тільки вирішувалися завдання автоматизації окремих виробничих процесів, але й почалася часткова реалізація принципів інформаційної інтеграції.

1.1.2. Автоматизовані системи підтримки життєвого циклу виробів

Аналіз розвитку інформаційних технологій у виробничій сфері показує, що одним з напрямків такого розвитку є всебічний охват цими технологіями різних етапів і стадій життєвого циклу (ЖЦ) виробів. Гнучкі виробничі системи вирішували завдання, що стосувалися винятково виробництва виробів.

К середині 1990-х рр. з'явилося усвідомлення необхідності створення інтегрованої інформаційної системи, що підтримує весь ЖЦ виробу. Життєвий цикл виробу - це сукупність процесів (етапів), що виконуються від моменту виявлення потреб суспільства в даному виробі до моменту задоволення цих потреб й утилізації цього виробу (рис. 1.1) [3,4].

Різноманіття процесів у ході ЖЦ виробу й необхідність їхньої інтенсифікації вимагають активної інформаційної взаємодії підприємств, що беруть участь у підтримці такого ЖЦ. З ростом числа етапів і стадій ЖЦ росте обсяг використовуваної й переданої інформації. Традиційний підхід, що склався в первісний період впровадження комп'ютерної техніки у виробничі процеси, полягав у тому, що з її допомогою вирішувалися окремі, локальні завдання, що ставилися до різних стадій ЖЦ виробів. Потім стали розроблятися й впроваджуватися системи автоматизованого проектування (САПР), які дозволяли використати засоби комп'ютерної техніки в процесах конструкторської й технологічної підготовки виробництва. Основні етапи ЖЦ промислових виробів і типи автоматизованих систем, що використовуються у їх ЖЦ, представлені на рис. 1.2, де прийняті наступні позначення:

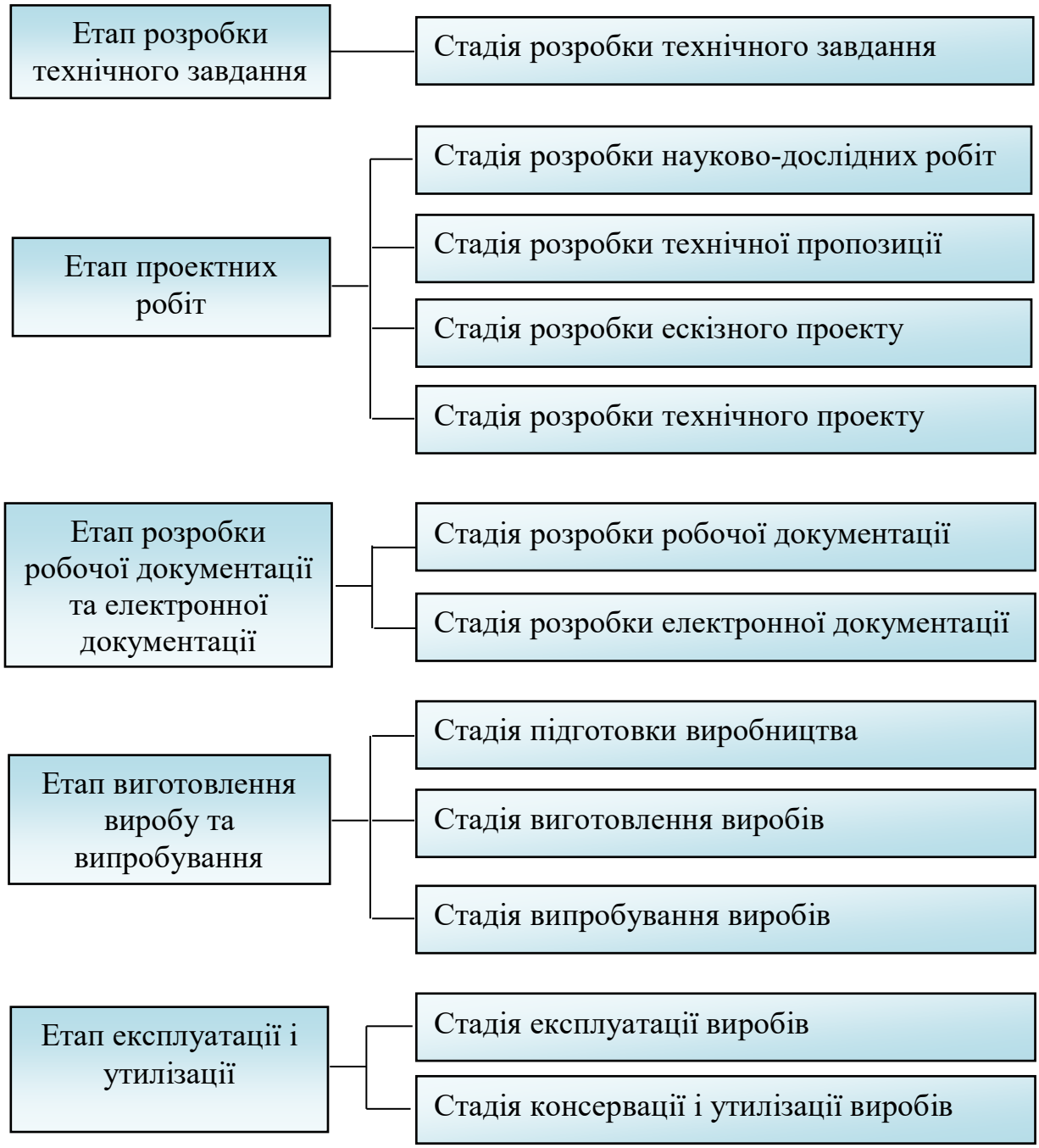

Рис. 1.1. Життєвий цикл виробу

- CAE (Computer Aided Engineering) - автоматизовані розрахунки й аналіз;
- CAD (Computer Aided Design) - автоматизоване проектування;
- CAM (Computer Aided Manufacturing) - автоматизована технологічна підготовка виробництва;
- PDM (Product Data Management) - управління проектними даними;
- ERP (Enterprise Resource Planning) - планування й управління підприємством;

– MRP II (Manufacturing Requirement Planning) - планування виробництва;
– MES (Manufacturing Execution System) - виробнича виконавча система;
– SCM (Supply Chain Management) - управління ланцюжками поставок;
– CRM (Customer Relationship Management) - управління взаєминами із замовниками;
– SCADA (Supervisory Control And Data Acquisition) - диспетчерське управління виробничими процесами;
– CNC (Computer Numerical Control) - комп'ютерне числове управління;
– S&SM (Salesand Service Management) - управління продажами й обслуговуванням;
– PLM (Product Lifecycle Management) - систему управління життєвим циклів продукції.

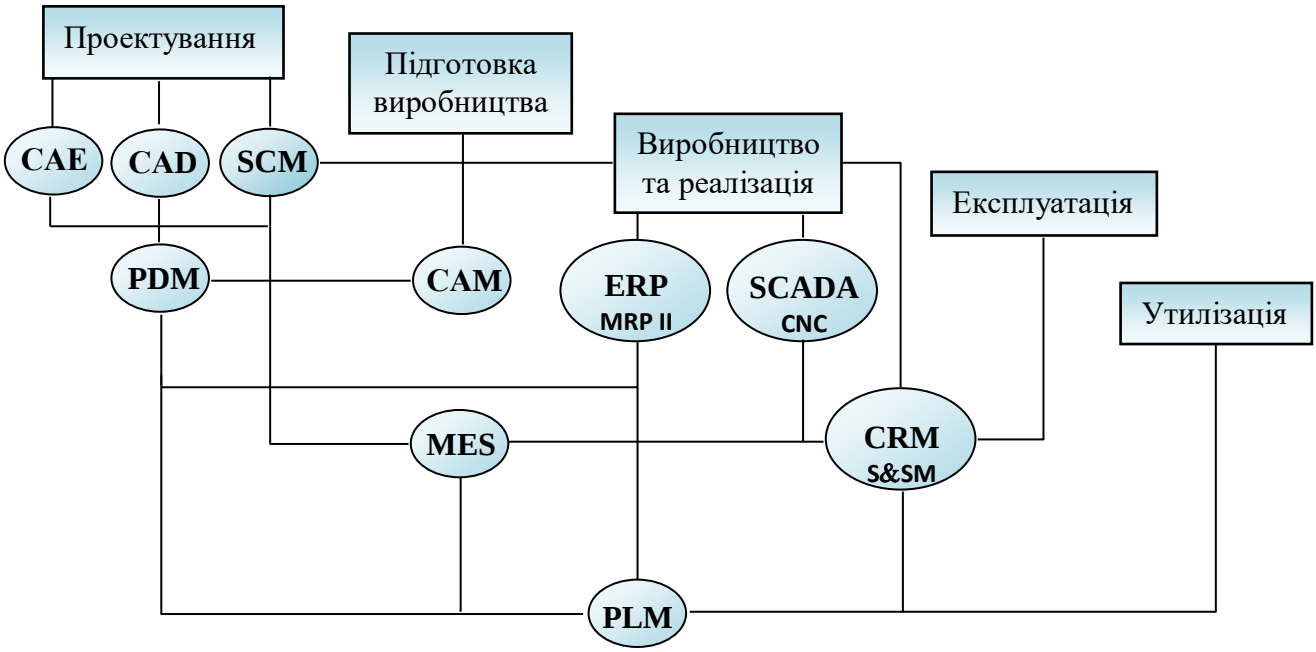

Рис. 1.2. Етапи життєвого циклу виробів в системі їх автоматизації

З рис.1.2 видно, що автоматизовані системи підтримують певні етапи й процедури в ЖЦ виробів.

Сучасні САПР, що забезпечують наскрізне проектування складних виробів мають багатомодульну структуру. Модулі розрізняються своєю орієнтацією на ті або інші проектні завдання стосовно до тих або інших типів пристроїв і конструкцій. При цьому виникають проблеми, пов'язані з побудовою загальних баз даних, з вибором протоколів, форматів даних й інтерфейсів різнорідних підсистем, з організацією спільного використання модулів при груповій роботі. Рішення цих проблем ускладнюється на підприємствах, що здійснюють складні вироби. Для рішення проблем спільного функціонування компонентів САПР різного виробничого призначення були розроблені спеціальні засоби управління проектними даними - системи PDM. Ці системи або входять до складу модулів конкретної САПР, або мають самостійне значення й можуть функціонувати разом з різними САПР.

Уже на етапі проектування потрібні послуги системи SCM, іноді її називають системою управління поставками комплектуючих, яка на етапі виробництва забезпечує поставки необхідних матеріалів і комплектуючих.

Автоматизовані системи технологічної підготовки виробництва, що становлять основу системи CAM, виконують синтез технологічних процесів і програм для встаткування з числовим програмним забезпеченням, вибір технологічного встаткування, інструмента, оснащення, розрахунок норм часу й т.ін. Модулі системи CAM звичайно входять до складу розвинених САПР, тому інтегровані САПР часто називають автоматизованими системами CAE/CAD/CAM/PDM.

Функції управління на промислових підприємствах звичайно виконуються за допомогою автоматизованих систем на декількох ієрархічних рівнях. Автоматизацію управління на верхніх рівнях здійснюють системи, що класифікуються як системи ERP або MRP II. Найбільш розвинені системи ERP виконують різні бізнес-функції, пов'язані з плануванням виробництва, закупівлями, збутом продукції, аналізом перспектив маркетингу, управлінням фінансами, персоналом, складським господарством, обліком основних фондів і

т.ін. Системи MRP II орієнтовані головним чином на бізнес-функції, безпосередньо пов'язані з виробництвом.

Автоматизовані системи управління технологічними процесами контролюють і використають дані, що характеризують стан технологічного встаткування й протікання технологічних процесів. Саме їх найчастіше називають системами промислової автоматизації. Для виконання диспетчерських функцій (збір й обробка даних про стан устаткування й технологічних процесів) і розробки програмного забезпечення для вбудованого встаткування до складу даних систем уводять систему SCADA. Для безпосереднього програмного управління технологічним устаткуванням використають системи CNC на базі контролерів - спеціалізованих комп'ютерів, вбудованих у технологічне встаткування.

На етапі реалізації продукції виникають завдання керування відносинами із замовниками й покупцями. Для рішення цих завдань проводиться аналіз ринкової ситуації, визначаються перспективи попиту на плановані до випуску вироби, які здійснюються за допомогою системи CRM. Маркетингові функції іноді покладають на систему S&SM, що крім того призначена для рішення проблем обслуговування.

На рішення оперативних завдань управлінням проектуванням, виробництвом і маркетингом орієнтовані системи MES, які близькі по деяких виконуваних функціях до систем ERP, PDM, SCM й S&SM, але відрізняються від них оперативністю й можливістю прийняття рішень у реальному часі, причому важливе значення надається оптимізації цих рішень із урахуванням поточної інформації про стан устаткування й процесів.

На етапі експлуатації застосовуються спеціалізовані комп'ютерні системи, пов'язані з рішенням питань ремонту, контролю й діагностики експлуатованих систем. Обслуговуючий персонал використовує інтерактивні навчальні посібники й технічні керівництва, а також засоби для дистанційного консультування при пошуку несправностей, програми для автоматизованого замовлення нових деталей замість тих, що відмовили.

Функції деяких автоматизованих систем часто перекриваються. Наприклад, управління маркетингом може виконуватися як системою ERP, так і системами CRM або S&SM.

Всі перераховані вище автоматизовані системи можуть працювати автономно. У цей час звичайно так і відбувається. Однак ефективність автоматизації буде помітно вище, якщо дані, що генеруються в одній із систем, будуть доступні в інших системах, що досягається створенням єдиного інформаційного простору не тільки на окремих підприємствах, але й у рамках об'єднання підприємств. Єдиний інформаційний простір забезпечується завдяки уніфікації як форми, так і змісту інформації про конкретні вироби на різних етапах їх ЖЦ.

Потреба в існуванні інтегрованої системи підтримки ЖЦ виробу й систематизації інформаційної взаємодії компонентів такої системи в рамках єдиного інформаційного простору обумовила необхідність створення інтегрованого інформаційного середовища. В основі якого лежить використання відкритих архітектур, міжнародних стандартів, спільне використання даних й апробованих програмно-технічних засобів.

1.1.3. CALS - технологія – основа автоматизованого проектування

Інформаційне забезпечення процесів управління в умовах глобалізації економіки та посилення конкуренції є важливим чинником прийняття управлінських рішень у ринковому середовищі. На сьогодні кінцева мета виробництва полягає не тільки в автоматизації процесів ЖЦ виробів, але й у реальному зниженні загальних витрат часу й засобів на весь ланцюжок: розробка засобів розробки - розробка виробу-виготовлення виробу. Досягнення цієї мети можливо по трьох основних напрямках: прискорення й здешевлення проектування виробу; прискорення й здешевлення виготовлення виробу; прискорення й здешевлення просування виробу на ринок і його експлуатації. При цьому кожне із зазначених напрямків є важливим саме по собі, і всі ці

напрямки зв'язані між собою. Взаємозв'язок цих трьох напрямків припускає їхню інформаційну інтеграцію, що являє собою один з найважливіших засобів прискорення й здешевлення виробництва виробу на кожному з етапів його ЖЦ, а також при переході від одного етапу ЖЦ до іншого.

В сучасних умовах на конкурентному ринку CALS-технологія (Continuous Acquisitionand Lifecycle Support) стає безперервною інформаційною підтримкою життєвого циклу продукту[3,4].

Вираз Continuous Acquisition (безперервний супровід) означає безперервність інформаційної взаємодії із замовником у ході формалізації його вимог, формування замовлення, процесу постачання й т.ін. (тобто має на увазі оптимізацію процесів взаємодії замовника й постачальника в ході розробки, проектування й виробництва складної продукції, термін життя якої з урахуванням різних модернізацій становить десятки років). Для забезпечення ефективності, а також скорочення витрат засобів і часу процес взаємодії замовника й постачальника дійсно повинен бути безперервним протягом тривалого строку.

Вираз Life Cycle Support (підтримка ЖЦ виробу) означає системність підходу до інформаційної підтримки всіх процесів ЖЦ виробу, у тому числі процесів експлуатації, обслуговування, ремонту, постачання запасними частинами, модернізації, утилізації, і полягає в оптимізації цих процесів. Оскільки витрати на підтримку складного наукомісткого виробу в працездатному стані часто рівні або навіть помітно перевищують витрати на його придбання, принципове скорочення вартості експлуатації виробу забезпечується інвестиціями в створення системи підтримки його ЖЦ.

Використання вказаних технологій дозволяє на вітчизняних підприємствах підвищити ефективність діяльності учасників створення, виробництва та використання продукту. Таким чином, для забезпечення процесів ефективного управління важливим стає впровадження нових інформаційних технологій. Ці технології входять до складу інформаційних ресурсів підприємств. Їх можливості та мобілізація в зовнішньому

конкурентному середовищу сприяє збільшенню інформаційного потенціалу підприємства. Особливе місце в умовах конкурентного ринку приділяється управлінню інформаційними потоками підприємства та розвитку інформаційного потенціалу, як утвореної видової прояву потенціалу підприємства в сфері дії закону інформаційності – упорядкування динамічних систем.

CALS-технології забезпечують комплексну комп'ютеризацію всіх сфер промислового виробництва, уніфікацію й стандартизацію специфікацій виробів на всіх етапах їх ЖЦ. Основні специфікації представлені проектною, технологічною, виробничою, маркетинговою, експлуатаційною документацією. CALS-технології у складі інформаційних ресурсів підприємства сприяють скороченню обсягів проектних робіт, тому що описи багатьох складових частин обладнання, машин та систем, спроектованих раніше, зберігаються в уніфікованих форматах даних сітьових серверів, доступних будь-якому користувачу CALS-технологій; полегшують вирішення проблем ремонтопридатності, інтеграції продукції у різноманітних системах та середовищах, а також адаптації до мінливих умов експлуатації, спеціалізації проектних організацій для забезпечення успіху на конкурентному ринку складної технічної продукції.

Розвиток CALS-технологій призводить до появи «віртуальних виробництв», в яких процес створення специфікації з інформацією для програмно-керованого технологічного обладнання може бути розподілений у часі та просторі між багатьма організаційно-автономними проектними студіями.

Досягненнями CALS-технологій є можливість оперативного розповсюдження проектних рішень, а також багаторазове відтворення та повторення частин проекту у нових розробках. Комплексна реалізація програмних продуктів у CALS-технологіях передбачає їх поділ на: програмні продукти для створення та перетворення інформації про вироби, виробниче середовище та виробничі процеси, застосування яких не залежить від реалізації

CALS-технологій; програмні продукти, які безпосередньо пов`язані з CALS-технологіями за відповідними стандартами.

До першої групи програмних продуктів в CALS-технологіях можуть бути віднесені програмні засоби та системи підготовки текстової та табличної документації; автоматизації інженерних розрахунків та ескізного проектування; автоматизації конструювання; автоматизації технологічної підготовки планування та управління виробництвом; ідентифікації та аутентифікації інформації.

До другої групи програмних продуктів в CALS-технологіях можуть бути віднесені засоби та системи управління проектами, даними про продукцію та її конфігурацію, інформаційними потоками, а також функціональне моделювання, аналіз та реінжиніринг бізнес-процесів. Важливою особливістю CALS-технологій є те, що вони працюють на основі системи міжнародних стандартів ISO 9000:2000. Таким чином, забезпечується технологічна захищеність формування інформаційних комплексів. Інформаційне забезпечення з використанням CALS-технологій є одним з ресурсів інвестиційного потенціалу підприємства, а науково-технічна та економічна інформація, яка є їх об’єктом, складає зміст нематеріальних ресурсів підприємства. Таким чином, основна увага спеціалістів у сфері застосування інформаційних технологій сконцентрована на використанні комп’ютерної техніки на всіх стадіях життєвого циклу виробу, забезпечуючи при цьому єдиноподібні способи управління процесами і взаємодії всіх учасників цього циклу. На нашу думку, розглянуті загальні знання про інформаційне забезпечення, яке мають визначати CALS-технології, полягають в тому, що вони можуть слугувати засобом оптимізації виробничого процесу, дозволяють підприємцям та менеджерам підвищити продуктивність праці своїх робітників, скоротити витрати та підвищити якість продукції.

1.1.4. Функціональна схема процесу автоматизованого проектування

Зараз на світовому ринку наукомістких промислових виробів чітко спостерігаються три основні тенденції: підвищення складності й ресурсоємкості виробів, підвищення конкуренції на ринку й розвиток кооперації між учасниками життєвого циклу виробу. Найбільш прогресивною та перспективною умовою удосконалення процесу проектування є створення і впровадження в практику систем автоматизованого проектування, забезпечених сучасними ПЕОМ з розвиненими термінальними системами. Автоматизація підготовки виробництва дає можливість підприємствам швидко реагувати на зміну попиту, скорочувати терміни проектування і виробництва нової продукції, швидко модернізувати продукцію, що випускається, відслідковувати життєвий цикл виробів, ефективно підвищувати якість, знижувати собівартість продукції.

В роботі [8] була запропонована функціональна схема автоматизованого проектування (рис.1.3), що дозволяє підвищити продуктивність праці проектувальників, скоротити терміни проектування та витрати на розробку технічної документації, збільшити кількість річних проектів.

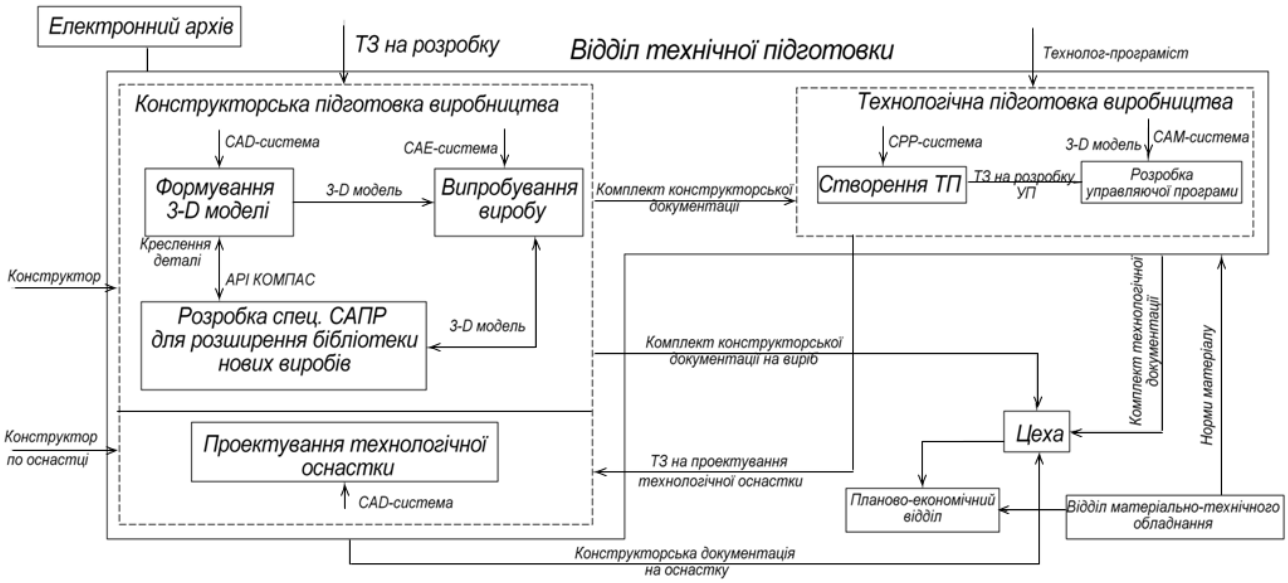

Рис.1.3. Функціональна схема автоматизованого проектування

1.2. Призначення розробки та галузь застосування

Як об'єкт впровадження розроблюваної спеціалізованої САПР розглядається універсальне програмне забезпечення на базі мови програмування C# з використанням API для розширення бібліотек в найбільш поширених системах автоматизованого проектування: КОМПАС, Autodesk Inventor, SolidWorks.

Запропонований програмний модуль дозволяє:

1) створити параметризований об'єкт;
2) здійснити підключення до наявного програмного середовища (Autodesk Inventor, SolidWorks, Компас 3D), в якому необхідно здійснити модифікацію існуючого об'єкту (моделі);
3) завантажити об'єкт в програмний модуль, при цьому у вікно програмного модуля динамічно завантажаться параметри моделі;
4) здійснити модифікацію (редагування) геометрії об'єкту, що було завантажено (згідно до нових розрахункових даних та параметрів креслення);
5) перебудувати об'єкт та зберегти поточний результат.

Запропонований програмний модуль дозволяє підвищити продуктивність праці проектувальників, скоротити терміни проектування та знизити витрати на розробку технічної документації, збільшити якість проектування за рахунок підвищення точності виготовлення виробів.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-С від 07.06.2021 р.;
- завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення для розширення бібліотек нових виробів в найбільш поширених CAD-системах».

1.4. Постановка завдання

Сучасний ринок програмного забезпечення автоматизації підготовки виробництва насичений найрізноманітнішими універсальними САПР, що здатні істотно полегшити роботу проектувальника (звільнити проектувальника від проведення багатьох рутинних операцій, що відчутно підвищує швидкість створення проектів, знижує ймовірність допустити помилки в процесі проектування). Разом з тим, не дивлячись на величезну кількість такого виду інструментальних засобів автоматизації інженерної діяльності, універсальні системи часто недостатньо ефективні для вирішення конкретного завдання користувача, не мають алгоритмів автоматизації окремих вузькоспрямованих дій (рутинних операцій, типових проектних рішень) в певних областях проектування, зачасту інформаційно несумісні. Подальший розвиток виробництва спричиняє удосконалювання технологічних процесів і устаткування, та ставить перед наукою нові задачі, що орієнтовані на пошук нових підходів та рішень в організації сучасного виробництва [7-10].

Більшість застосовуваних у промисловості тривимірних САПР зачасту вирішують задачі усередненого підприємства без врахування його специфіки. Підвищення конкуренції на ринку вимагає від підприємств постійного оновлювання номенклатура виробів. Оскільки номенклатура виробів постійно оновлюється на базі вже розроблених вузлів,тому в більшості випадків робота проектувальника зводиться до модифікації раніше створеної геометрії вузла

відповідно до нових розрахункових даних. Тому для ефективної роботи сучасного підприємства необхідно розробляти спеціалізовані САПР, що здатні розширити бібліотеку нових виробів та вирішувати завдання розрахунку і проектування конкретного класу виробів. Більшість застосовуваних у промисловості тривимірних САПР можуть бути використані як основа для побудови спеціалізованих САПР, які вирішують завдання розрахунку і проектування конкретного класу виробів. При цьому необхідно об'єднати розрахунковий модуль, який визначає розмірні та інші параметри проектованого об'єкта з уже наявним в САПР тривимірним геометричним ядром. Цей процес здійснюється за допомогою API - інтерфейс прикладної програми. Набір таких інтерфейсів забезпечує взаємозв'язок між зовнішніми модулями прикладної програми і низькорівневими функціями ядра, а так само між компонентами ядра – різними бібліотеками, що істотно підвищує потенційні можливості застосування універсальних систем в специфічних предметних областях.

Також, слід відзначити, що більшість виробників в процесі проектування виробів використовують декілька різних пакетів CAD- систем[1]. Причинами для цього є:

- зростаюча складність проектованих виробів (сучасні вироби створюються в декількох варіантах, включають різні рівні технологій, що потребує різних інструментів розробки);
- розвиток глобального конструкторського аутсорсингу (існує необхідність пристосовуватися до умов партнерів за ланцюгом постачань, що використовують різні засоби CAD для забезпечення спільної роботи);
- об'єднання і придбання (в умовах сучасного глобального ринку об'єднання і придбання здійснюються частіше, інтеграція вимагає ефективної асиміляції існуючих CAD-систем для підтримки цілісності даних).

Метою кваліфікаційної роботи є розробка програмного забезпечення з використанням API інтерфейсу для розширення бібліотек нових виробів в найбільш поширених системах автоматизованого проектування: КОМПАС-3D, Autodesk Inventor, SolidWorks. Розроблений програмний модуль дозволить підвищити продуктивність праці проектувальників (за рахунок звільнення проектувальника від проведення багатьох рутинних операцій), скоротити терміни проектування та знизити витрати на розробку технічної документації, збільшити якість проектування за рахунок підвищення точності виготовлення виробів.

Для досягнення поставленої мети в роботі необхідно виконати наступні задачі:

- здійснити аналіз предметної галузі з метою виявлення сучасного стану обраної проблематики;
- здійснити аналіз існуючих CAD-систем з метою виявлення їх властивостей та потенційних можливостей;
- розглянути існуючий механізм взаємодії систем автоматизованого проектування із програмним модулем;
- здійснити вибір інструментальних засобів для розробки модуля API;
- розробити алгоритм роботи програмного модуля САПР;
- розробити спеціалізовану САПР на базі API найпоширеніших CAD – систем (КОМПАС-3D, Autodesk Inventor, SolidWorks) для розширення бібліотек нових виробів;
- розробити інструкцію користувача розробленого програмного модуля.

1.5. Вимоги до програми або програмного виробу

Вимоги до програми або програмного продукту – це вхідні дані, на підставі яких проектуються і створюються програмний продукт.

Основними критеріями ефективності впровадження програмного продукту є максимальна віддача та недопустимість виникнення збоїв та помилок при

роботі. Для досягнення максимуму ефективності при роботі з програмним продуктом сформуємо ряд вимог, які необхідно дотриматись при створенні програмного модулю.

Виходячи із бачення кінцевого продукту, можна виділити ряд вимог:

- програмний продукт має виконувати свою головну функцію (створювати параметризований об'єкт, здійснити підключення до наявного програмного середовища, завантажити об'єкт в програмний модуль, здійснити модифікацію (редагування) геометрії об'єкту, перебудувати об'єкт та зберегти поточний результат) за умови дотримання вимог надійності, продуктивності, швидкості роботи та гнучкості архітектури;
- програмний продукт має бути орієнтований на користувача і задовольняти його потреби, пов'язані зі сферою роботи програми та вимогами зручності (простота використання, інтуїтивність інтерфейсу користувача, повна зрозумілість);
- програмний продукт має бути економічно ефективним.

Всі інші вимоги, які деталізують сам програмний продукт, можуть змінюватись у рамках уніфікованого підходу та ітеративної розробки.

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в програмному модулі, що розробляється, повинні бути реалізовані:

- можливість створення параметричної збірки проектованого механізму для внесення ряду розмірів об'єкту в змінні моделі;
- можливість під’єднання до однієї з CAD-систем (КОМПАС-3D, Autodesk Inventor, SolidWorks) для об'єднання розрахункового модулю, який визначає розмірні та інші параметри проектованого об'єкта з вже наявним в САПР тривимірним геометричним ядром;

- можливість обрання тривимірної моделі, яку необхідно модифікувати (модуль зчитує поточні значення параметрів моделі та динамічно завантажує їх у інтерфейс, де користувач має змогу змінити їх та перебудувати модель).
- можливість контролю із зовнішньої програми за допомогою API-інтерфейсів, які надають доступ до параметрів збірки.
- можливість збереження поточного результату (нової геометрії виробу);
- програмно-апаратна переносимість.

1.5.2. Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програмного модулю необхідно реалізувати:

- контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- платформну незалежність.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програмного модулю, повинні виконуватися певні вимоги до технічних засобів.

- процесор з тактовою частотою не менш 2,5GHz;
- оперативна пам'ять 4Gb;
- жорсткий диск 20Гб;
- монітор – SVGA, з діагоналлю не менше 17';
- операційна система: Microsoft Windows 7 /8/ 10;

– клавіатура;

– маніпулятор типу «миша».

Програмне забезпечення (хоча б одна з CAD-систем), а саме: Autodesk Inventor (2015 SP1 і вище), SolidWorks (2013 SP3 і вище), КОМПАС-3D (2014 V15 і вище)повинно бути встановлене завчасно.

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програмного модулю необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати модуль, відповідало наступним вимогам:

– операційна система Microsoft Windows 7 /8/ 10;

Програмний модуль САПР має бути реалізовано на мові програмування C#. CAD – системи повинні підтримувати API – технології.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Як об'єкт впровадження розроблюваної спеціалізованої САПР розглядається універсальне програмне забезпечення на базі мови програмування C# з використанням API для розширення бібліотек нових виробів в найбільш поширених системах автоматизованого проектування: КОМПАС-3D, Autodesk Inventor, SolidWorks.

Призначення розробленого модулю:

- розширити бібліотеку нових виробів та вирішити завдання розрахунку і проектування конкретного класу виробів враховуючи специфіку підприємства;
- оновити номенклатуру виробів на базі вже розроблених вузлів за рахунок модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних;
- підвищення продуктивності праці проектувальників, скорочення термінів проектування та зниження витрат на розробку технічної документації;
- збільшення якості проектування за рахунок підвищення точності виготовлення виробів.

Розроблений програмний модуль має наступну структуру:

- Клас CADet - головний клас, являє собою Windows Form, містить користувальницький інтерфейс, виконує функцію менеджера функцій (підключитися до програми, створити об'єкт ProggramObject).
- Клас ProggramObject - управляючий батьківський загальний клас для AutodeskObject класу, SolidObject класу і KompasObject класу. Містить віртуальні методи, призначені для наслідування:

– Клас AutodeskObject (успадкований від ProggramObject, призначений для управління Autodesk COM об'єктом);
– Клас SolidObject (успадкований від ProggramObject, призначений для управління SolidWorks COM об'єктом);
– Клас KompasObject (успадкований від ProggramObject, призначений для управління КОМПАС-3D COM об'єктом).

2.2. Опис застосованих математичних методів

Більшість застосовуваних у промисловості тривимірних САПР можуть бути використані як основа для побудови спеціалізованих САПР, які вирішують завдання розрахунку і проектування конкретного класу виробів. В даній роботі розроблено програмний модуль, який визначає розмірні та інші параметри проектованого об'єкта з вже наявним в САПР тривимірним геометричним ядром. Цей процес здійснюється за допомогою API - інтерфейс прикладної програми. Набір таких інтерфейсів забезпечує взаємозв'язок між зовнішніми модулями прикладної програми і низькорівневими функціями ядра, а так само між компонентами ядра – різними бібліотеками, що істотно підвищує потенційні можливості застосування універсальних систем в специфічних предметних областях.

Розроблений програмний модуль дозволяє під’єднатися до однієї з CAD-систем (КОМПАС-3D, Autodesk Inventor, SolidWorks), що використовують різні математичні ядра[2].

КОМПАС-3D використовує ядро C3D Toolkit, що об'єднує в собі найважливіші модулі САПР [10,13]:

– геометричне ядро C3D Modeler: здійснює побудову і редагування геометричних моделей, розрахунок тріангуляції, побудова плоских проекцій, обчислення масово-центровочних характеристик і визначення зіткнень елементів моделі;

- параметричне ядро C3D Solver: встановлює залежності між різними елементами геометричної моделі за допомогою розмірів і логічних обмежень і забезпечує збереження заданих зв'язків при зміні розмірів, параметрів моделі, переміщенні окремих елементів моделі або внесення інших змін до геометрії моделі;
- модуль візуалізації C3D Vision: управляє настройками візуальної сцени для статичного і динамічного режимів відтворення геометрії, а також забезпечує інтерактивну взаємодію користувача з інтерфейсом додатків;
- модуль обміну C3D Converter: здійснює обмін даними про геометричній моделі з іншими системами проектування, підтримує формати: JT, IGES, STEP (AP203, AP214, AP242), Parasolid X_T / X_B, ACIS SAT, STL, VRML.

SolidWorks використовує ядро Parasolid [10,16] - це найшвидше ядро, доступне для ліцензування, розроблено UGS. Parasolid забезпечує технологію для твердотільного моделювання, узагальненого пористого моделювання, інтегровані поверхні вільної форми і листове моделювання. Parasolid дозволяє розробникам швидко створювати конкурентоспроможні продукти використовуючи ці технології. На цьому ядрі розроблено багато CAD / CAM / CAE систем високого і середнього рівня. Parasolid підтримує SMP (багатопроцесорне апаратне забезпечення), що дозволяє збільшити продуктивність. Parasolid включає більш ніж 600 об'єктно-орієнтованих функцій для додатків під управлінням Windows NT, UNIX, і LINUX.

Autodesk Inventor використовує ядро **ACIS** [10,11] - це об'єктно-орієнтована геометрична бібліотека на мові C ++, яка включає каркасні структури, поверхні і твердотільне моделювання, причому підтримується і гібридне моделювання. ACIS здійснює виведення в формат файлів SAT, який будь-яка програма, що підтримує ACIS може читати безпосередньо.

2.3. Опис використаної архітектури та шаблонів проектування

Розробка програмного модуля для CAD-систем за допомогою API загалом має типову структуру. Структура модуля, який включає в себе підтримку таких CAD-систем, як Autodesk Inventor, SolidWorks і КОМПАС-3D, і написаний на мові C #, представлена на рис.2.1.

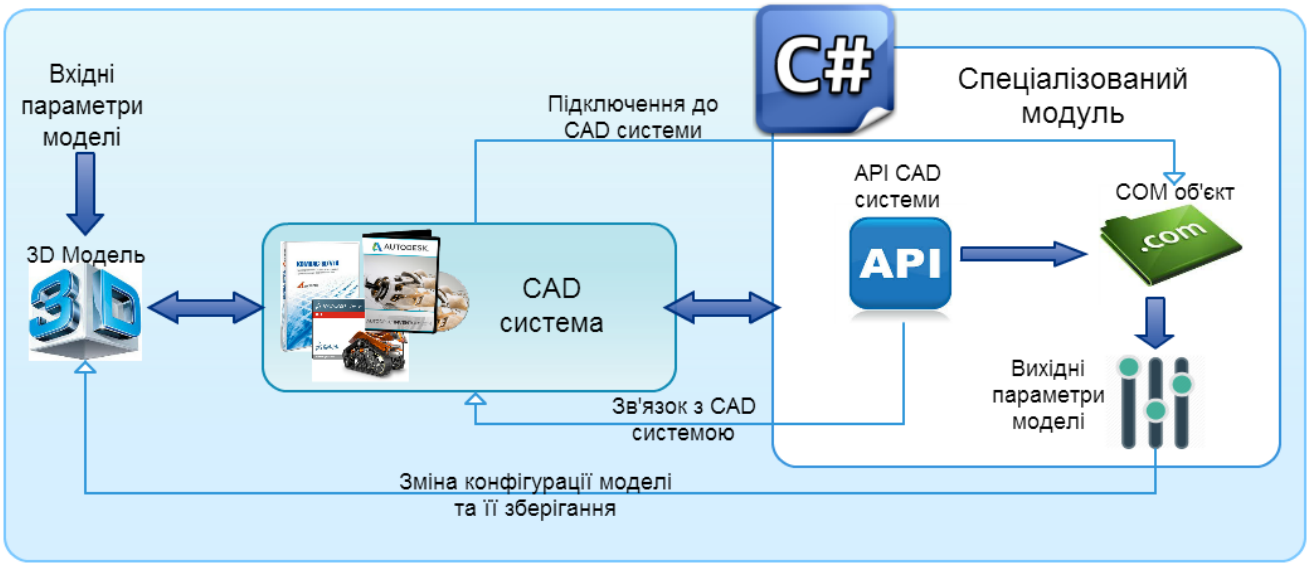

Рис.2.1. Схема зміни конфігурації 3D моделі

Робота з процесами CAD-систем здійснюється за допомогою COM об'єктів.

COM (Component Object Model - об'єктна модель компонентів) - це технологічний стандарт від компанії Microsoft, призначений для створення програмного забезпечення на основі взаємодіючих компонентів, кожен з яких може використовуватися в багатьох програмах одночасно. Стандарт втілює в собі ідеї поліморфізму і інкапсуляції об'єктно-орієнтованого програмування[12].

Основним поняттям, яким оперує стандарт COM, є COM-компонент. Програми, побудовані на стандарті COM, фактично не є автономними програмами, а являють собою набір взаємодіючих між собою COM-компонентів. Кожен компонент має унікальний ідентифікатор (GUID) і може

одночасно використовуватися багатьма програмами. Компонент взаємодіє з іншими програмами через COM-інтерфейси - набори абстрактних функцій і властивостей. Кожен COM-компонент повинен, як мінімум, підтримувати стандартний інтерфейс «IUnknown», який надає базові засоби для роботи з компонентом. Інтерфейс «IUnknown» включає в себе три методи: QueryInterface, AddRef, Release.

Windows API надає базові функції, що дозволяють використовувати COM-компоненти. Бібліотеки MFC і, особливо, ATL / WTL надають більш гнучкі та зручні засоби для роботи з COM. Бібліотека ATL від Microsoft досі залишається найпопулярнішим засобом створення COM-компонентів. Але найчастіше COM-розробка залишається ще досить складною справою, програмістам доводиться вручну виконувати багато рутинні завдання, пов'язані з COM (особливо це помітно у разі розробки на C ++). Згодом (в технологіях COM + і особливо .NET) Microsoft спробувала спростити завдання розробки COM-компонентів [12].

Загальний алгоритм роботи модулю:

1. Використовуючи системний метод CreateInstance () створюється COM об'єкт. В якості параметру виступає тип CAD системи, об'єкт якої створюється.

2. Відкривається файл моделі, яку необхідно модифікувати відповідно до нових даних (креслення деталі).

3. Отримуємо список параметрів за допомогою API функцій відповідної CAD-системи та записуємо їх у користувальницький інтерфейс.

4. Здійснюємо модифікацію тривимірної моделі встановлюючи нові параметри.

5. Перебудовуємо модель натисненням кнопки «Перебудувати» та зберігаємо результат перебудови.

Ієрархія класів реалізованого програмного модуля «CADet» показана на рис.2.2.

Клас CADet - головний клас, являє собою Windows Form, містить користувальницький інтерфейс, виконує функцію менеджера функцій - підключитися до програми, створити об'єкт ProggramObject.

Клас ProggramObject - управляючий батьківський загальний клас для AutodeskObject класу, SolidObject класу і KompasObject класу. Містить віртуальні методи, призначені для наслідування:

- Booleaninit () - ініціалізація COM об'єкта CAD-системи;
- VoidCloseAllInstance () - закрити всі існуючі процеси відповідної CAD-системи;
- VoidReBuild () - перебудувати поточну модель, з урахуванням змінених параметрів;
- VoidSetSettings () - встановити поточні значення параметрів в користувальницький інтерфейс.

Клас AutodeskObject - успадкований від ProggramObject, призначений для управління Autodesk COM об'єктом;

Клас SolidObject - успадкований від ProggramObject, призначений для управління SolidWorks COM об'єктом;

Клас KompasObject - успадкований від ProggramObject, призначений для управління Компас 3D COM об'єктом.

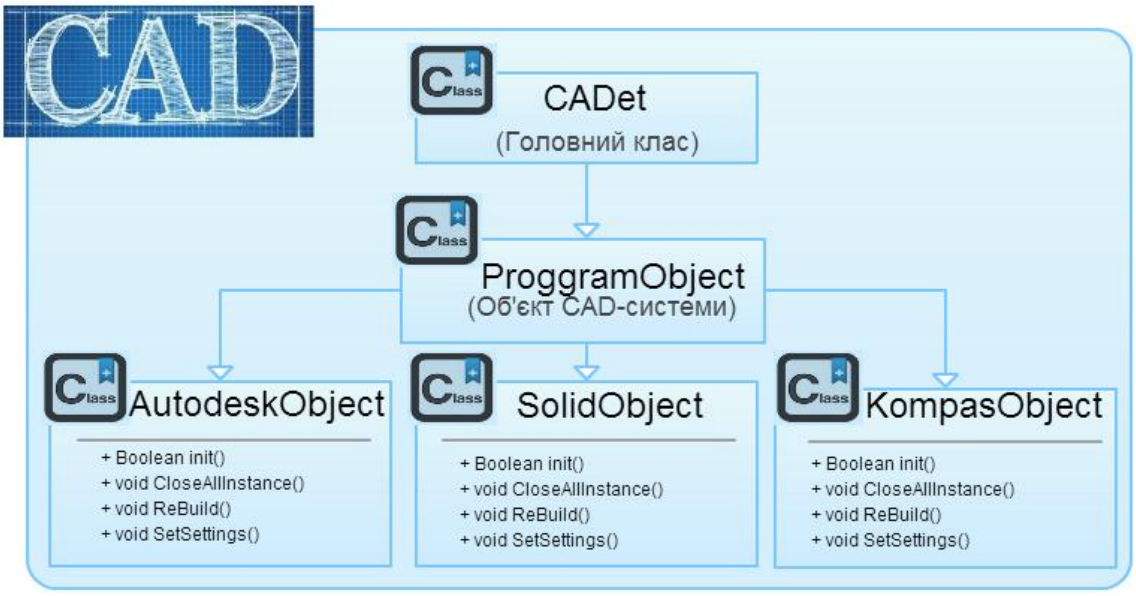

Рис.2.2. Ієрархія класів програмного модуля "CADet"

2.4. Опис використаних технологій та мов програмування

2.4.1. Прикладний програмний інтерфейс API

API (Application Program Interface) – інтерфейс прикладної програми. Набір таких інтерфейс забезпечує взаємозв'язок між зовнішніми модулями прикладної програми і низькорівневими функціями ядра, а так само між компонентами ядра та різними бібліотеками[6].

Орієнтація API насамперед на забезпечення для користувачів багатьох можливостей зручного підключення зовнішніх модулів, написаних на мовах високого рівня, а також на роботу в мережі, істотно підвищує потенційні можливості застосування універсальних систем в специфічних предметних областях, незважаючи на відомі переваги спеціалізованих графічних редакторів: простоту і відсутність функціональної надмірності.

API визначає функціональність, яку надає програма (модуль, бібліотека), при цьому API дозволяє абстрагуватися від того, як саме ця функціональність реалізована.

Програмні компоненти взаємодіють один з одним за допомогою API. При цьому зазвичай компоненти утворюють ієрархію - високорівневі компоненти використовують API низькорівневих компонентів, а ті, в свою чергу, використовують API ще більш низькорівневих компонентів.

На сьогоднішній день немає готових користувацьких програмних модулів, які можна придбати та встановити. Тому виникає необхідність у їх розробці.

Більшість застосовуваних у промисловості CAD – систем можуть бути використані як основа для побудови спеціалізованої САПР, вирішуючи завдання розрахунку і проектування конкретного класу виробів.

Одними з найпоширеніших представників CAD – систем, що підтримують API-технології є КОМПАС, Autodesk Inventor, Solid Works. Кожен програмний пакет має ряд позитивних та негативних критеріїв.

КОМПАС 3D – система тривимірного моделювання, що стала стандартом для тисяч підприємств, завдяки вдалому поєднанню простоти засвоєння і легкості роботи з потужними функціональними можливостями твердотільного і поверхневого моделювання, які вирішують всі основні завдання користувачів[10,13].

Базова функціональність продукту легко розширюється за рахунок різних додатків, які доповнюють функціонал КОМПАС 3D ефективним інструментарієм для вирішення спеціалізованих інженерних завдань.

Система автоматизованого проектування Solid Works створена для використання на персональному комп'ютері в операційному середовищі Microsoft Windows. У SolidWorks використовується принцип тривимірного твердотільного і поверхневого параметричного проектування, що дозволяє конструктору створювати об'ємні деталі і компонувати збірки у вигляді тривимірних електронних моделей, за якими створюються двовимірні креслення і специфікації відповідно до вимог ЕСКД [18].

Autodesk Inventor є світовим лідером серед систем автоматизованого проектування та надає своїм користувачам концептуальний 2D і 3D дизайн разом з гнучкими інструментами для створення документації до проектів [11].

Завдяки потужному механізму моделювання, Autodesk Inventor збільшує загальну продуктивність роботи над будь – якими проектами, і однаково добре підходить таким фахівцям, як архітектори, інженери, дизайнери і т.д.

Autodesk Inventor має потужну функціональність у плані 3D моделювання і дозволяє створювати і налаштовувати будь – які об'єкти.

2.4.2.Вибір інструментальних засобів для розробки модуля API

Для реалізації API – інтерфейсу можуть бути використані різні мови програмування, зокрема, C++, Pascal, Delphi, C# і т.д. Розглянемо деякі з них.

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Мова має строгу статичну типізацію,

підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML [5,14].

TurboPascal (названий на честь Блеза Паскаля) – це чисто процедурна мова програмування, що часто використовується для навчання структурному програмуванню [17].

Особливостями мови є сувора типізація і наявність засобів структурного (процедурного) програмування. Крім того, мова надає ряд вбудованих структур даних: записи, масиви, файли, множини і покажчики.

Проте, мова має безліч недоліків: неможливість передачі функцій масивів змінної довжини, відсутність нормальних засобів роботи з динамічною пам'яттю, обмежена бібліотека введення – виведення.

Delphi – це середовище швидкої розробки, в якій в якості мови програмування використовується ObjectPascal. В основі ідеології Delphi лежить технологія візуального проектування і методологія об'єктно – орієнтованого подієвого програмування [15].

Останні версії мови отримали підтримку деяких концепцій функціонального програмування без застосування потворних інверсій абстракції, які зазвичай супроводжують функціональні мови.

Серед мов програмування найбільш широкий розвиток отримали універсальні мови, за допомогою яких зручно вирішувати як логічні завдання, так і обчислювальні. При цьому універсальна мова програмування повинна бути зручний для обробки текстів, зображень, файлів, а складена програма повинна добре читатися і розумітися однозначно всіма програмістами, щоб її надалі можна було модифікувати.

C# найкращим чином відповідає всім цим потребам. C# – це комбінація кількох найважливіших технологій:

– високопродуктивний компілятор в машинний код;

– об'єктно – орієнтована модель компонентів;

Вибір оптимальної мови програмування, для створення API – інтерфейсу, здійснюємо по методу аналізу ієрархій опираючись на наступні критерії:

- сприйняття коду;
- функціональність;
- простота освоєння;
- вартість придбання та супроводу;
- інтерфейс;
- швидкість розробки рішення;
- швидкість роботи додатків.

Порядок застосування методу аналізу ієрархій:

- побудова якісної моделі проблеми у вигляді ієрархії, що включає мету, альтернативні варіанти досягнення цілі та критерії для оцінки якості альтернатив;
- визначення пріоритетів всіх елементів ієрархії з використанням методу парних порівнянь;
- синтез глобальних пріоритетів альтернатив шляхом лінійної згортки пріоритетів елементів на ієрархії;
- перевірка суджень на узгодженість;
- ухвалення рішення на основі отриманих результатів.

Таблиця 2.1

Визначення вектора головного пріоритету

	К1	К2	К3	К4	К5	К6	К7	ВГП
C#	0,47	0,44	0,44	0,37	0,44	0,32	0,39	0,41
Pascal	0,31	0,38	0,35	0,43	0,34	0,47	0,36	0,38
Delphi	0,22	0,18	0,21	0,20	0,22	0,21	0,25	0,21

Отже, як видно з таблиці 2.1, вектор головного пріоритету найвищий у програмного пакету C#. Тому саме цю мову програмування було обрано в дипломному проекті для розробки спеціалізованої САПР.

2.4.3.Огляд існуючих засобів роботи з КОМПАС API

Головним інтерфейсом API системи КОМПАС є KompasObject. Отримати покажчик на цей інтерфейс (якщо бути точним, на інтерфейс програми API 5) можна за допомогою експортної функції CreateKompasObject () [10,13]. Методи цього інтерфейсу, реалізують найбільш загальні функції роботи з документами системи, системними налаштуваннями, файлами, а також дають можливість отримати покажчики на інші інтерфейси (інтерфейси динамічного масиву, роботи з математичними функціями, бібліотеки моделей або фрагментів і різних структур параметрів певного типу) [10,13]. В таблиці 2.2 представлена частина методів інтерфейсу KompasObject.

Таблиця 2.2

Деякі методи інтерфейсу KompasObject

ActiveDocument2D	Дозволяє отримати покажчик на активний графічний документ
ActiveDocument3D	Дозволяє отримати покажчик на активний графічний тривимірний документ
Document2D	Дозволяє отримати покажчик на інтерфейс графічного документа (креслення або фрагмента)
Document3D	Дозволяє отримати покажчик на інтерфейс тривимірного графічного документа (деталі або збірки)
GetDynamicArray	Повертає покажчик на інтерфейс динамічного масиву

2.4.4.Огляд існуючих засобів роботи з API Autodesk Inventor

AutodeskInventor.NET API дозволяє управляти додатком Autodesk Inventor і файлами креслень на програмному рівні з використанням доступних зборок або бібліотек. Ці об'єкти можуть бути доступні для безлічі різних мов програмування і різних середовищ розробки програмного забезпечення [11].

Ось деякі з переваг, одержуваних від застосування .NET API в Autodesk Inventor:

- Відкритий програмний доступ до креслень Autodesk Inventor з різних середовищ розробки додатків. Перед .NET API, розробники були обмежені ActiveX і мовами, що підтримують COM, AutoLISP і C ++ з ObjectARX.
- Інтеграція з іншими Windows-додатками, такими як MS Excel, і Word стає більш ефективною і легкої при використанні «рідного» .NET API або доступних ActiveX / COM бібліотек.
- Бібліотека .NET Framework розроблена як для 32-бітної, так і для 64-бітної версій операційної системи Windows. Мова VBA працювала тільки на 32-бітової версії.
- Дозволений доступ до налаштувань програмного інтерфейсу для інших мов, що не вважаються традиційними, подібно до того, як C ++.
- Об'єкти є основними будівельними блоками AutodeskInventor .NET API. Кожен з таких об'єктів точно відповідає деякому певному типу примітиву, наявному в Autodesk Inventor. Класи таких об'єктів згруповані в різних бібліотеках і просторах назв.

В Autodesk Inventor .NET API існує безліч різних типів об'єктів. Наприклад:

- графічні об'єкти: відрізки, окружності, текст і розміри;
- налаштування стилів: стилі тексту і стилі розмірів;
- організаційні структури: шари, групи і блоки;
- відображення креслення: види і видові екрани;
- креслення і додаток Autodesk Inventor.

Для роботи з Autodesk Inventor можна використовувати бібліотеки безпосередньо з Autodesk Inventor, або використовувати ObjectARX.

ObjectARX - це великий набір бібліотек, призначений для розробки додатків для AutodeskInventor в середовищі програмування Microsoft Visual C++. Сам Autodesk Inventor розроблений з використанням ObjectARX.

"Чистий" ObjectARX призначений для роботи з Autodesk Inventor за допомогою C ++, однак частина бібліотек представляє собою ні що інше, як обгортки для класів ObjectARX та роботи через .net.

ObjectARX загальнодоступний і його можна завантажити з сайту Autodesk. Крім того, за цією ж адресою можна скачати велику документацію по SDK.

Необхідно використовувати однакові версії ObjectARX і Autodesk Inventor. Сумісність роботи бібліотеки різних версій Autodesk Inventor не гарантується. Так що при зміні версії використовуваного Autodesk Inventor необхідно перекомпілювати програму з новими бібліотеками.

Існує два основних способи взаємодії Autodesk Inventor і C # за допомогою .Net API:

1. Програма реалізується у вигляді окремого виконуваного файлу для роботи з файлами Autodesk Inventor через COM-інтерфейси бібліотеки AutodeskInventor.Interpop.Common. Даний прийом дозволяє отримати звичайний виконуваний exe-файл, який буде працювати з dwg-файлами через COM. Даний спосіб має своє право на існування, проте, вельми обмежений функціонально через малу числа доступних способів "впливу" на креслення і не рекомендується в більшості випадків.

2. У вигляді розширення (plugin) Autodesk Inventor. Результатом роботи буде dll-файл, який завантажується в Autodesk Inventor командою "netload" і визначає нові команди (операції) та / або нову поведінку стандартних операцій.

Набір бібліотек ObjectARX надає розробнику величезний набір інструментів як для роботи з кресленнями, так і з вікнами Autodesk Inventor.

Основні можливості, що надаються ObjectARX:

- створення нового файлу креслення;
- редагування існуючих креслень, яке включає в себе: редагування примітивів, блоків, креслення;

- додавання нових команд;
- зміна інтерфейсу Autodesk Inventor (додавання нових кнопок, панелей, закладок).

2.4.5. Огляд існуючих засобів роботи з API SolidWorks

Використання SolidWorks API - найбільш дешевий і зручний спосіб гнучко налаштувати інформаційну систему на вирішення завдань конкретного підприємства. Завдяки цьому зараз SolidWorks є однією з найпопулярніших систем проектування як в Україні, так і в усьому світі.

SolidWorks API - це інтерфейс прикладного програмування, що дозволяє розробляти власні програми на платформі САПР SolidWorks [10,18]. API містить сотні функцій, які можна викликати із програм Microsoft VisualBasic, VBA (Microsoft Excel, Word, Access і т.д.), Microsoft Visual C, C ++, .NET або з файлів-макросів SolidWorks. API-функції забезпечують прямий програмний доступ до функціональних можливостей пакета SolidWorks, що дозволяє створювати і модифікувати як 2D-, так і 3D-геометрію. На практиці виявляється, що якщо для проектування виробів в середовищі CAD-системи можливостей звичайного користувача інтерфейсу більш ніж достатньо, то для вирішення завдань інтеграції додатків на рівні єдиної інформаційної системи підприємства потрібно пов'язати між собою різнорідні програмні продукти, налагодивши між ними повноцінне інформаційне взаємодія.

Таким чином, найбільш поширеними сферами застосування інтерфейсу прикладного програмування SolidWorks є:

- інтеграція SolidWorks з різними Windows-додатками (CAD / CAM / CAE / PDM / ERP, MS Office, Windows API та ін.), Що припускає створення інтерфейсів передачі даних, виклику сервісних утиліт, перетворення даних і т.ін.;

– розробка прикладних модулів, що додають до базових можливостей САПР SolidWorks додатковий функціонал в будь-якій спеціальній предметній області.

Інтерфейс прикладного програмування поставляється в складі базової конфігурації САПР SolidWorks. Крім базового конструкторського рішення API є у всіх основних модулів, що входять в пакет SolidWorks, до яких насамперед належать Toolbox, FeatureWorks, Utilities, PhotoWorks, eDrawings, Routing, SWR-PDM, SWR-Електрика. Динамічні бібліотеки типів і констант, що відповідають за роботу API, автоматично встановлюються на комп'ютер при установці програми. Таким чином, кожне робоче місце САПР SolidWorks за замовчуванням оснащене інтерфейсом прикладного програмування, що відкриває перед розробниками широкі можливості.

2.5. Опис структури програми та алгоритмів її функціонування

Автоматизація процесів проектування дозволяє звільнити інженера від проведення багатьох рутинних операцій, що відчутно підвищує швидкість створення проектів, знижує ймовірність допустити помилки в процесі проектування. Однак, спеціалізовані програмні продукти та технічні засоби для автоматизації проектних робіт в початковій комплектації не мають алгоритмів автоматизації окремих вузькоспрямованих дій (рутинних операцій, типових проектних рішень) в певних областях проектування.

Більшість застосовуваних у промисловості тривимірних САПР можуть бути використані як основа для побудови спеціалізованої САПР, яка вирішує завдання розрахунку і проектування конкретного класу виробів. При цьому необхідно об'єднати розрахунковий модуль, який визначає розмірні та інші параметри проектованого об'єкта з уже наявним в САПР тривимірним геометричним ядром (див. п.2.2).

Для цього спочатку створюється параметрична збірка проектованого механізму, в якій ряд розмірів винесений в змінні моделі. Розрахунковий

модуль (це зовнішній exe-файл або dll-бібліотека, що підключається до САПР) може розрахувати необхідні значення змінних моделі і автоматично змінити їх, в результаті чого буде отримано новий варіант 3D збірки. Таким чином, відразу ж після розрахунку буде отримана нова геометрія виробу.

Головну складність представляє не стільки виконання розрахунків, скільки організація взаємодії розрахункового модуля і САПР. Як правило, управління із зовнішньої програми здійснюється за допомогою технології API. API-технологія надає розробнику набір процедур і функцій для управління САПР, але не дає прямого доступу до властивостей і методів об'єктів всередині САПР, що робить код програми дещо більш громіздким і менш зрозумілим.

Загальний алгоритм роботи САПР схематично зображено на рисунку 2.3.

Рис.2.3. Схематичний алгоритм роботи програмного модуля САПР

Основні принципи роботи програмного модулю:

1. Користувач має готовий параметризований об'єкт, розроблений в CAD-системі (Autodesk Inventor, SolidWorks, Компас 3D).

2. Деталь відкривається в CAD-системі;

3. Програмний модуль отримує доступ до параметрів моделі, використовуючи засоби API;

4. Модуль САПР змінює параметри моделі (за допомогою алгоритмів модифікації, користувальницького інтерфейсу та ін.);

5. Модуль САПР перебудовує 3D модель і відображає готову деталь в CAD-системі.

Ідея параметричного підходу дуже проста та зрозуміла. Якщо уявити, що на креслення великої будівлі було витрачено кілька місяців, а, надавши креслення замовнику, він попросив розширити віконні отвори на кілька сантиметрів. Без параметризації це означатиме серйозну переробку, яка може займати багато часу. Якщо ж правильно виставити розміри і геометричні обмеження, то завдання вирішується автоматично зміною всього лише одного параметра.

У разі реальних креслень - результуюча система рівнянь може вийти дуже великою і складною. Вона може містити багато тисяч рівнянь і мільйони змінних. Жодна сучасна параметрична CAD-система не в змозі обробляти параметризовані моделі такої складності. Дуже часто їх поведінка нестабільна вже для моделей, що містять невелику кількість геометричних примітивів. Більше того, навіть якщо вони будуть здатні обробити відносно велику модель, для отримання результату може знадобитися багато часу, що робить роботу з такою моделлю некомфортною.

Побудова 3D моделі деталі складається з двох типів її змінних - статичних і змінних. Статичні параметри - незмінні параметри, які є основоположним для даного виробу, змінні - параметри, які можуть бути модифіковані користувачем, для зміни розмірів, відповідно до технічного завдання.

Для того, щоб вказані параметри можна було задавати із зовнішньої програми, їх треба оголосити, як змінні моделі. В CAD системах 3D змінними моделі можуть бути будь-які розміри, проставлені на ескізах, а також розміри, що вводяться при виконанні формоутворюючих операцій (наприклад, висота

видавлювання ескізу). Для доступу до змінних на рівні деталі їх треба оголосити, як зовнішні змінні, призначивши їм псевдоніми. Псевдонім - це ім'я, під яким змінну ескізу або розмір операції видно на рівні деталі (рис.2.4).

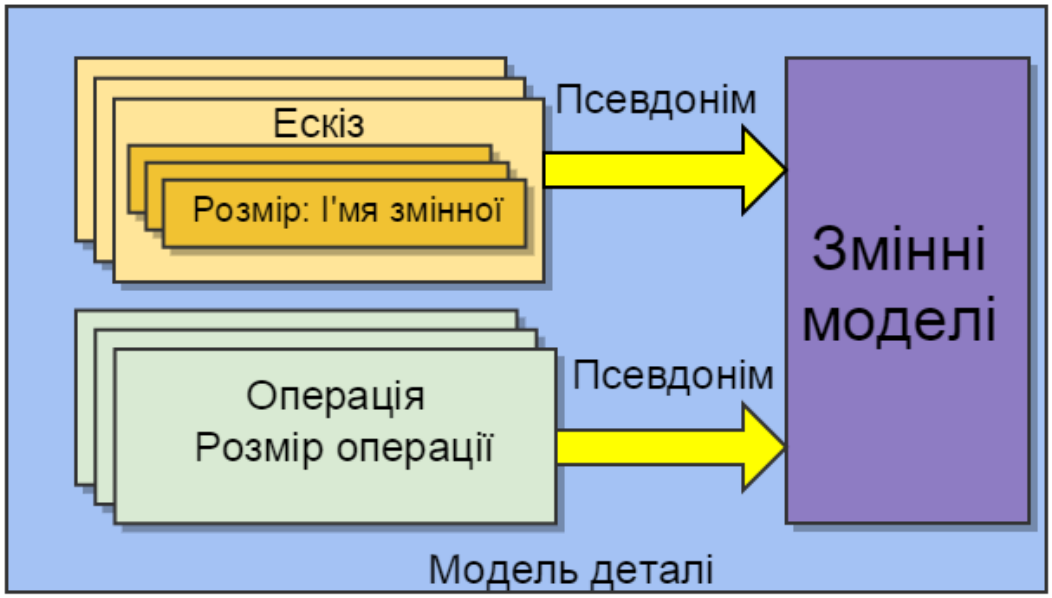

Рис.2.4. Структура змінних моделі

Таким чином, для роботи САПР, необхідно параметризувати модель, що, по-перше, дозволить модифікувати ці параметри із зовнішньої програми, по-друге, дозволить коректно змінювати формоутворюючі розміри деталі без її деформації.

Розрахунковий модуль крім власне розрахунків повинен виконувати наступні функції:

- підключення до CAD-системі і завантаження в неї параметричної збірки;
- отримання поточних значень змінних деталей, що входять в збірку, назв деталей і назви самої збірки;
- зміна значень змінних, перебудова і збереження моделі.

Розробка програмного забезпечення на базі API КОМПАС

Для отримання можливості працювати з API Компас 3D, в проект необхідно додати посилання на бібліотеки - Kompas6Constants.dll і

Kompas6API5.dll (ці бібліотеки лежать в кореневій папці з встановленою програмою Компас 3D, також їх можна завантажити з офіційного сайту).

Перед тим як створити новий процес Компас 3D, необхідно закрити всі поточні процеси Компас 3D (з метою оптимізації роботи):

```
Process[] processes = Process.GetProcessesByName("KOMPAS");
foreach (Processprocesinprocesses)
   {
 proces.CloseMainWindow();
 proces.Kill();
   }
```

Далі створюємо новий процес Компас 3D і приводимо тип COM об'єкта до типу KompasObject:

```
Type t = Type.GetTypeFromProgID("KOMPAS.Application.5");
kompas = (KompasObject)Activator.CreateInstance(t);
```

KompasObject - посилання на об'єкт Компас 3D, точка входу для використання API Компас 3D.

Створюємо новий 3D документ типу ksDocument3D і відкриваємо 3D модель, використовуючи метод Open об'єкта ksDocument3D:

```
doc3D = (ksDocument3D)kompas.Document3D();
if (doc3D != null)
{
doc3D.Open(@"F:\5
курс\Диплом\ДИПЛОМ\Детали\Корпус\Деталь.a3d", false);
}
```

При натисканні на кнопку «Перебудувати», викликається метод ReBuild (), який отримує посилання на функціональні частини 3D документа типу ksPart:

```
ksPartsinglePart = (ksPart)doc3D.GetPart(-1);
```

З об'єкта singlePart отримуємо посилання на список змінних 3D документа типу ksVariableCollection:

```
ksVariableCollectionVarCol =
(ksVariableCollection)singlePart.VariableCollection();
```

Далі, проходячи циклом з цієї колекції, змінюємо значення параметрів. Після зміни значень параметрів, необхідно оновити функціональну частину і перебудувати 3D документ:

```
singlePart.RebuildModel();
doc3D.RebuildDocument();
```

Таким чином, за допомогою API Компас 3D, програмний модуль змінює значення параметрів 3D моделі. При необхідності, можна зберегти поточний результат або ще раз змінити значення параметрів і знову перебудувати модель.

Розробка програмного забезпечення на базі API Autodesk Inventor

В проект додаємо посилання на бібліотеку - Interop.dll для отримання можливості працювати з API Autodesk Inventor. За замовчуванням ця бібліотека перебуває в кореневій папці

«C:\Windows\Microsoft.NET\assembly\GAC_MSIL\Autodesk.Inventor.Interop \v4.0_19.0.0.0__d84147f8b4276564\autodesk.inventor.interop.dll» (для 64-розрядної операційної системи), також її можна завантажити з офіційного сайту).

З метою оптимізації роботи закриваємо всі поточні процеси AutodeskInventor ():

```
Process[] processes = Process.GetProcessesByName("INVENTOR");
foreach (Processprocesinprocesses)
{
proces.CloseMainWindow();
proces.Kill();
}
```

Далі створюємо новий процес Autodesk Inventor і наводимо тип COM об'єкта до типу Inventor.Application - посилання на об'єкт Autodesk Inventor, точка входу для використання API Autodesk Inventor:

```
Type t = Type.GetTypeFromProgID("INVENTOR.Application");
v_invApp = (Inventor.Application)Activator.CreateInstance(t);
```

Створюємо новий 3D документ типу Inventor_Document і відкриваємо 3D модель використовуючи метод Open об'єкта Inventor_Document.Documents:

```
    Inventor._Document v_file;
    v_file =
v_invApp.Documents.Open(@"F:\4курс\Диплом\ДИПЛОМ\Детали\Корпус\Дет
аль.ipt");
```

При натисканні на кнопку «Перебудувати», викликається метод ReBuild (), який отримує посилання на активний документ - 3D документ типу:

```
    dynamicotherdoc = v_invApp.ActiveDocument;
```

З об'єкту otherdoc отримуємо посилання на список змінних 3D документа типу oParameters і посилання на список параметрів поточної моделі:

```
    ParametersoParameters
=otherdoc.ComponentDefinition.Parameters();
    ModelParametersModelParams = oParameters.ModelParameters;
```

Далі, проходячи циклом з цієї колекції, змінюємо значення параметрів. Після зміни значень параметрів, необхідно перебудувати 3D документ:

```
    v_file.Update();
```

Таким чином, за допомогою API Autodesk Inventor, програмний модуль змінює значення параметрів 3D моделі. При необхідності, можна зберегти поточний результат або ще раз змінити значення параметрів і знову перебудувати модель.

Розробка програмного забезпечення на базі API SolidWorks

Для отримання можливості працювати з API SolidWorks, в проект необхідно додати посилання на бібліотеки - SolidWorks.Interop.sldworks.dll, SolidWorks.Interop.swcommands.dll та SolidWorks.Interop.swconst.dll (за замовчуванням для 64-розрядної операційної системи, ці бібліотеки перебувають в кореневій папці«C:\ProgramFiles\SolidWorksCorp\SolidWorks\Toolbox\toolboxconfigure», також їх можна завантажити з офіційного сайту).

Перед тим як створити новий процес SolidWorks, необхідно закрити всі поточні процеси SolidWorks (з метою оптимізації роботи):

```
    Process[] processes = Process.GetProcessesByName("SLDWORKS");
    foreach (Processprocesinprocesses)
    {
```

```
proces.CloseMainWindow();
proces.Kill();
}
```

Далі створюємо новий процес SolidWorks і наводимо тип COM об'єкта до типу SldWorks:

```
objectv_processSW =
System.Activator.CreateInstance(System.Type.GetTypeFromCLSID(v_myG
uid));
v_App = (SldWorks)v_processSW;
```

SldWorks - посилання на об'єкт SolidWorks , точка входу для використання API SolidWorks.

Створюємо новий 3D документ типу IModelDoc2 і відкриваємо 3D модель використовуючи метод OpenDoc об'єкта IModelDoc2:

```
IModelDoc2 v_Model;
v_Model = (IModelDoc2)v_App.OpenDoc(FilePath, 1);
```

При натисканні на кнопку «Перебудувати», викликається метод ReBuild (), отримуємо посилання на менеджер параметрів поточної моделі – EquationMgr:

```
EquationMgrswEquationMgr = default(EquationMgr);
swEquationMgr = (EquationMgr)v_Model.GetEquationMgr();
```

Далі, проходячи циклом з цієї колекції, змінюємо значення параметрів. Після зміни значень параметрів, необхідно перебудувати 3D документ:

```
Part.EditRebuild3();
```

Таким чином, за допомогою API SolidWorks, програмний модуль змінює значення параметрів 3D моделі. При необхідності, можна зберегти поточний результат або ще раз змінити значення параметрів і знову перебудувати модель.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Розроблений програмний модуль дозволяє вирішувати завдання розрахунку і проектування конкретного класу виробів за рахунок його об'єднання з вже наявним в САПР тривимірним геометричним ядром. Для цього спочатку створюється параметрична збірка проектованого механізму, в

якій ряд розмірів винесений в змінні моделі. Розрахунковий модуль (це зовнішній exe-файл або dll-бібліотека, що підключається до САПР) може розрахувати необхідні значення змінних моделі і автоматично змінити їх, в результаті чого буде отримано новий варіант 3D збірки. Таким чином, відразу ж після розрахунку буде отримана нова геометрія виробу. Контроль із зовнішньої програми здійснюється за допомогою API-інтерфейсів, які надають доступ до параметрів збірки.

Розроблений програмний модуль дозволяє під'єднатися до однієї з CAD-систем (КОМПАС-3D, Autodesk Inventor, SolidWorks); обрати необхідну модель, що потребує модифікації; здійснити модифікацію обраної моделі (програмний модуль зчитує поточні значення параметрів моделі та динамічно завантажує їх у інтерфейс, де користувач має змогу змінити їх та перебудувати модель); зберегти поточний результат (модифіковану модель) при необхідності, або ще раз змінити значення параметрів і знову перебудувати модель.

Вхідні дані:

– створений параметризований об'єкт в якому ряд розмірів винесений в змінні моделі (3D моделі, що потребує модифікації);

Вихідні дані:

– модифікована тривимірна модель об'єкту.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для технічних засобів рекомендована конфігурація, що забезпечує ефективну роботу розробленого програмного забезпечення:

– тип центрального процесора (один із зазначених класів процесорів): AMD Athlon 64 з технологією SSE2, AMD Opteron ™ з технологією SSE2, Intel® Xeon® з підтримкою Intel EM64T і технологією SSE2,

IntelPentium 4 з підтримкою Intel EM64T і технологією SSE2 (з тактовою частотою не менш 3 ГГц);

- оперативна пам’ять: не менше 4 ГБ;
- розширення екрану: 1024 x 768 (рекомендується 1600 x 1050 або вище) з підтримкою повнокольорового режиму TrueColor;
- відеоадаптер: для ОС Windows з підтримкою дозволу 1024 x 768 і повнокольорового режиму TrueColor. Відеокарта з підтримкою DirectX® 9 або DirectX 11 (рекомендується, але не обов'язково).
- місце на диску: 6 ГБ вільного місця на диску;
- рідкокристалічний монітор з діагоналлю не менше 17 ";
- доступ до мережі Internet;
- клавіатура;
- маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.7.2. Використані програмні засоби

Проект реалізований на мові програмування C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

IDE розробки - Visual Studio 2019.

Необхідні програмні засоби для клієнтського комп'ютера:

- ОС Microsoft Windows7/8/10;

– хоча б одна з CAD-систем: Autodesk Inventor (2015 SP1 і вище), SolidWorks (2013 SP3 і вище), КОМПАС-3D (2014 V15 і вище);
– мова програмування C#.

2.7.3. Виклик та завантаження програми

Щоб скористатися послугами розроблено програмного модулю, необхідно запустити зовнішній exe-файл CADet.exe (рис.2.5) та потрапити до користувальницького інтерфейсу.

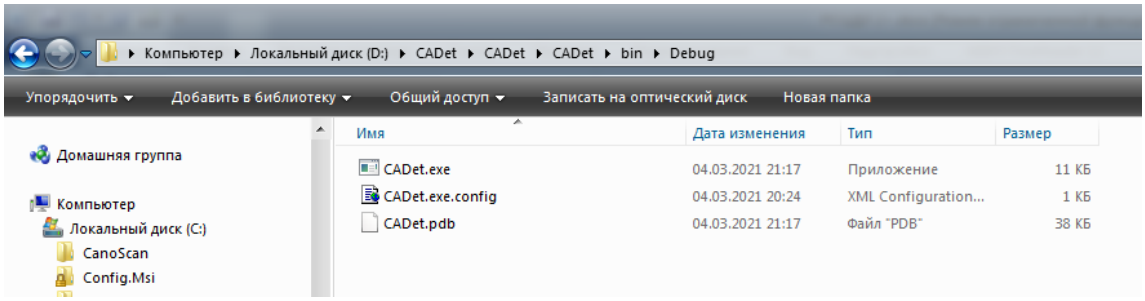

Рис.2.5. Виклик та завантаження програми

2.7.4. Опис інтерфейсу користувача

Для роботи з програмним модулем, користувач, насамперед, повинен встановити хоча б одну з CAD-систем (бажано, що б версії програм збігалися):

– Autodesk Inventor (2015 SP1 і вище);
– SolidWorks (2013 SP3 і вище);
– КОМПАС-3D (2014 V15 і вище).

Користувальницький інтерфейс розробленого програмного модуля представлений на рис.2.6.

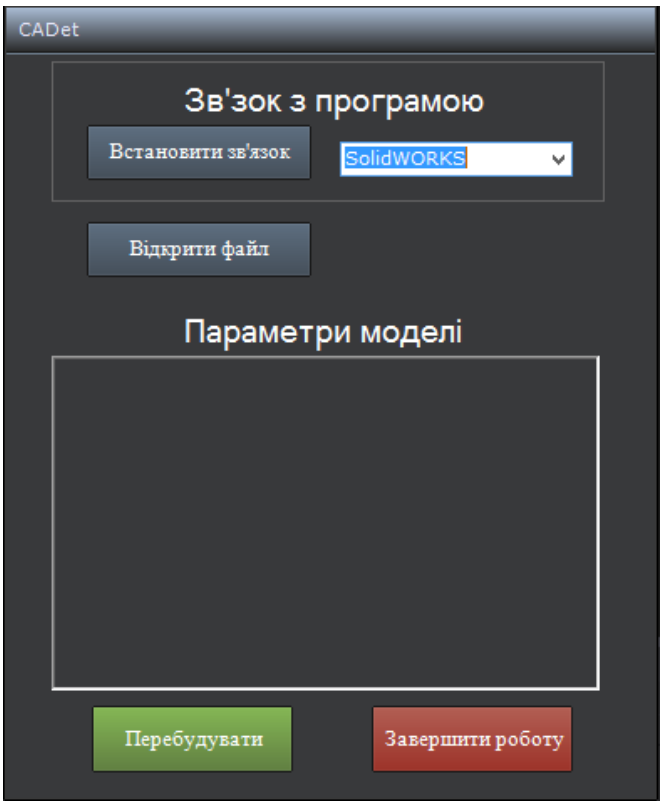

Рис.2.6. "CADet" - користувальницький інтерфейс модуля САПР

Як було сказано раніше, програмний модуль САПР істотно спрощує роботу проектувальника, скорочуючи час роботи над об'єктами і над проектами в цілому. Виходячи з цього, інструкція використання модуля досить проста. Розглянемо її на прикладі Autodesk Inventor:

1. Створення параметризованого об'єкту (побудова тривимірної моделі в одній із CAD-систем).

2. Підключення до програмного середовища (рис.2.7), в якому необхідно модифікувати існуючу геометрію об'єкту(Autodesk Inventor, SolidWorks, КОМПАС-3D). При успішному встановленні з'єднання з обраною зі списку CAD-системою з'явиться наступний значок  .

3. Завантаження об'єкта в програмний модуль. При натисканні на кнопку "Відкрити файл" з'являється можливість обрання необхідної моделі. При цьому, у вікно програмного модуля динамічно завантажаться параметри моделі (рис.2.8).

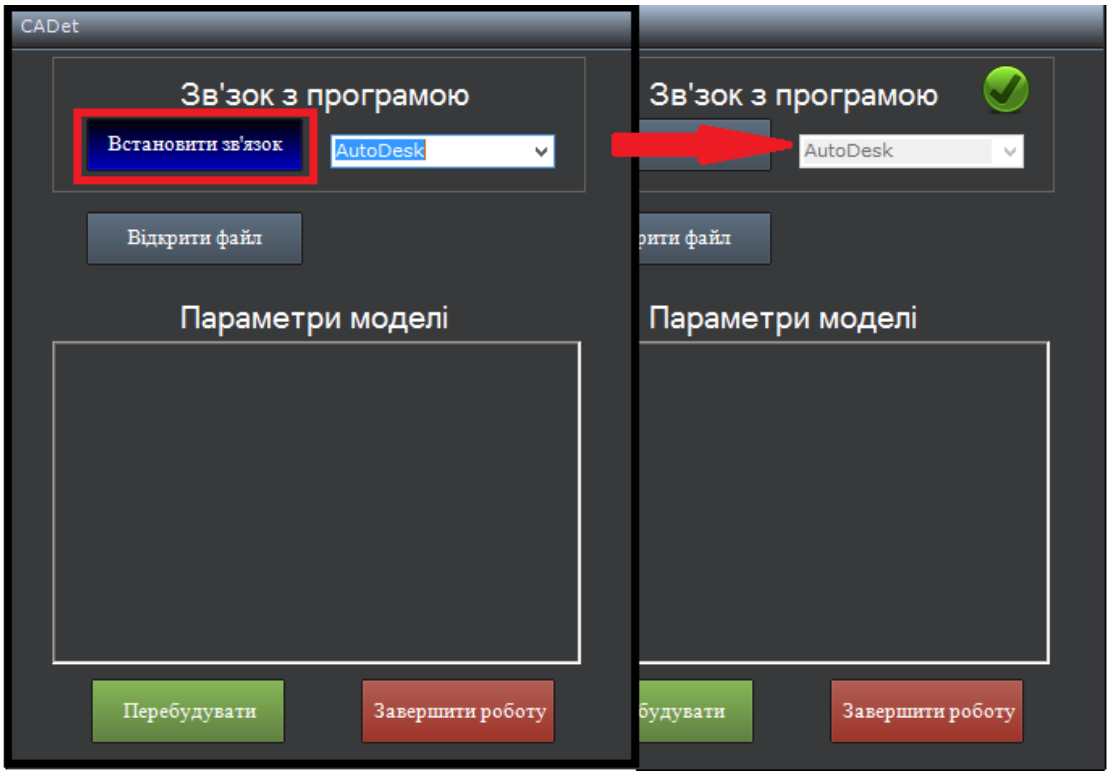

Рис.2.7. Підключення до CAD-системи

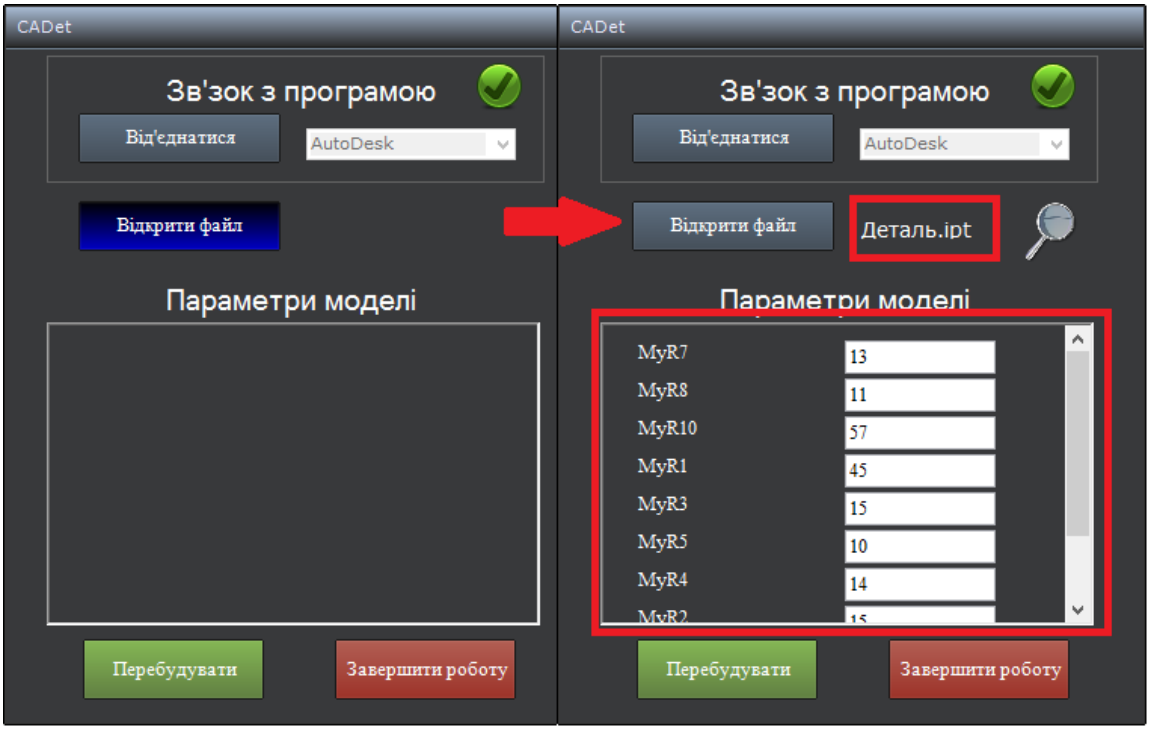

Рис.2.8. Завантаження об'єкта в програмний модуль

4. Тепер користувач має змогу переглянути модель (рис.2.9).

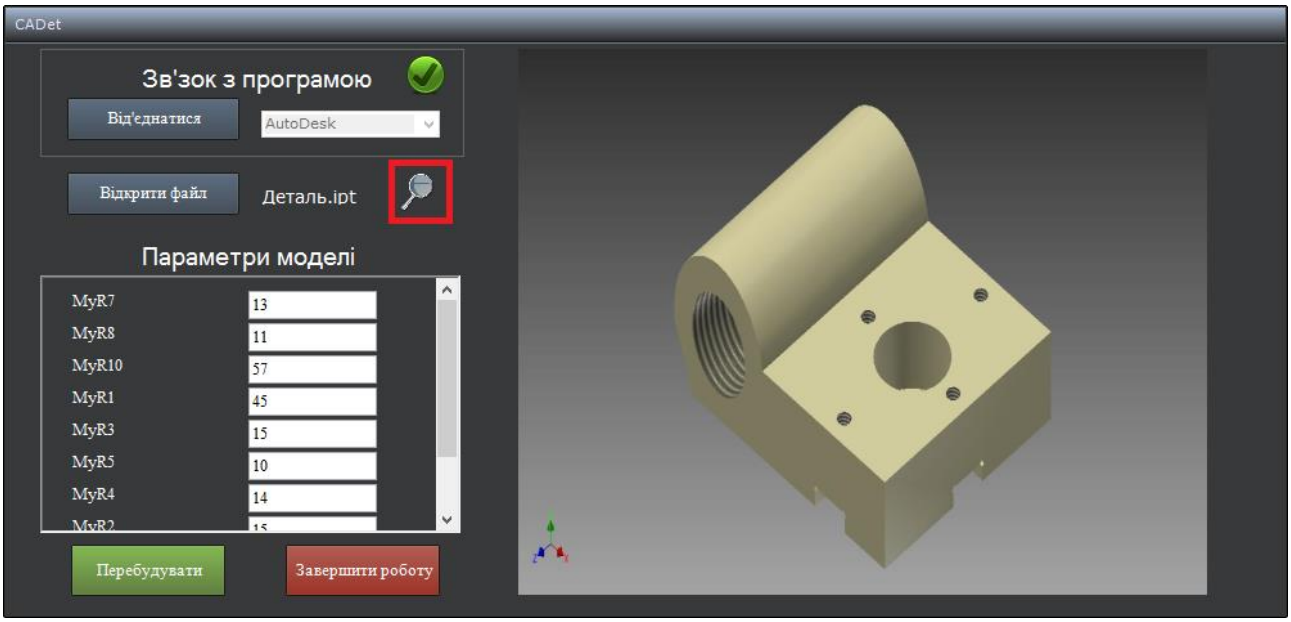

Рис.2.9. Швидкий перегляд моделі

5. Модифікація існуючої геометрії моделі (зміна параметрів моделі у відповідному вікні "Параметри моделі"згідно до заданого креслення).

6. Перебудова, збереження об'єкта.

Для демонстрації роботи програмного модуля було використано 3D-модель деталі "Корпус". Початковий вид даної деталі представлений на рис.2.10.

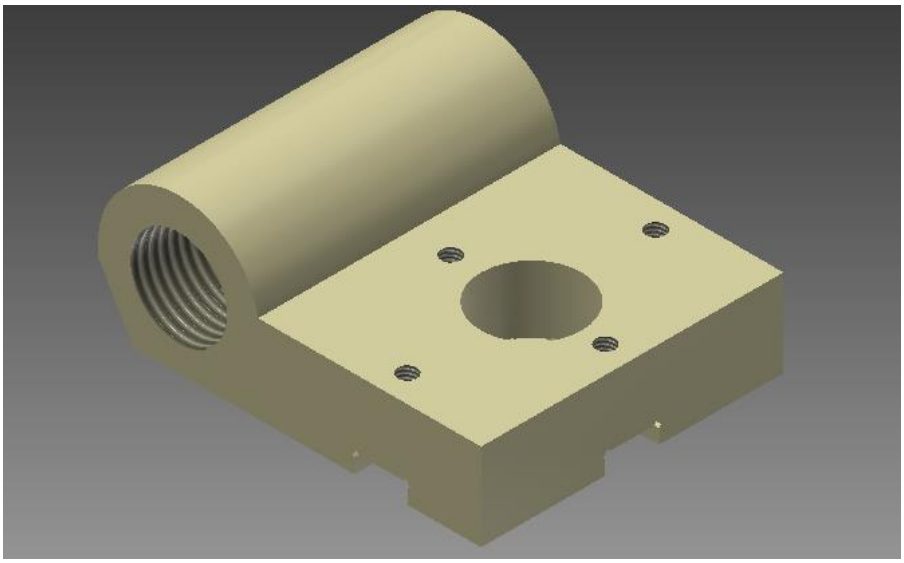

Рис.2.10. 3D-модель деталі "Корпус"

Після натискання кнопки «Перебудувати», отримуємо результат у відповідному програмному CAD середовищі, зі зміненими параметрами.

На рисунках нижче (рис.2.11-2.13), представлені результати роботи в різних CAD-системах з використанням різних параметрів.

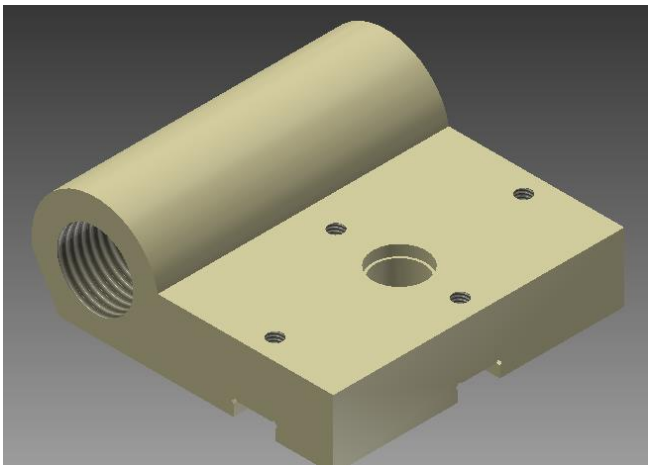

Рис.2.11. 3D-модель деталі "Корпус"- модифікована в Autodesk Inventor

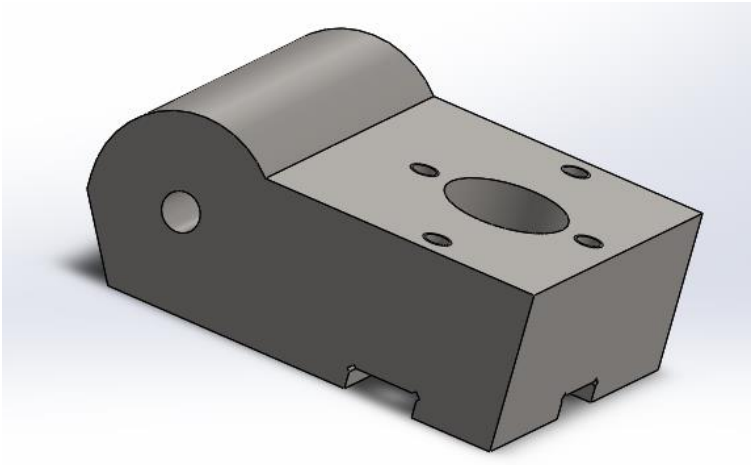

Рис.2.12. 3D-модель деталі "Корпус" - модифікована в SolidWorks

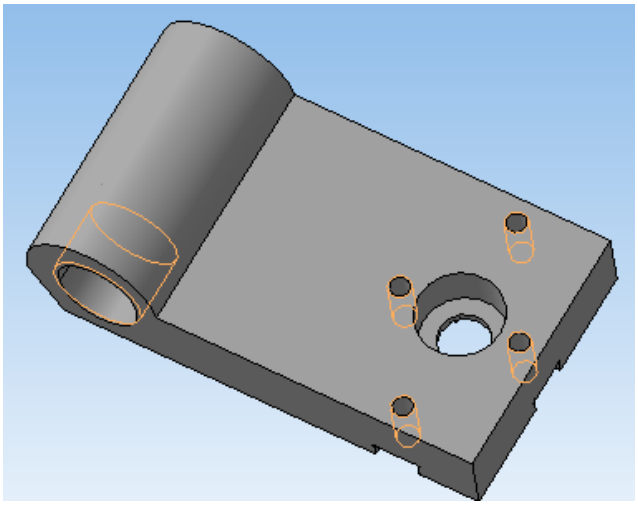

Рис.2.13. 3D-модель деталі "Корпус" - модифікована в Компас-3D

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми - 500;
2. Коефіцієнт складності програми (1,25...2,0) - 1,5;
3. Коефіцієнт корекції програми в ході її розробки (0,05...0,1) - 0,05;
4. Годинна заробітна плата програміста - 84грн/год;
5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (від 3-хдо 5 років) – 1,1;
7. Вартість машино-години ЕОМ - 14грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів програми (q=500);

C - коефіцієнт складності програми (C =1,5);

p - коефіцієнт корекції програми в ході її розробки (p=0,05).

Звідси умовне число операторів в програмі:

$$Q = 500 \cdot 1,5 \cdot (1 + 0,05) = 787,5.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,} \tag{3.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,1.

Приймемозбільшеннявитратпрацівнаслідокнедостатньогоописузавдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,1, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (787,5 \cdot 1,2) / (75 \cdot 1,1) = 11,45\text{людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k}\ \text{, людино-годин,} \tag{3.4}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 787,5 / (20 \cdot 1,1) = 35,79 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,} \qquad (3.5)$$

t_n = 787,5 / (25 · 1,1) = 28,63людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин,} \qquad (3.6)$$

$t_{отл}$ =787,5 / (5 · 1,1) = 143,18 людино-годин.

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5\cdot t_{отл}\text{, людино-годин,} \qquad (3.7)$$

$t^k_{отл}$ = 1,5 · 143,18 = 214,77 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}\ \text{, людино-годин,} \qquad (3.8)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}\ \text{, людино-годин,} \qquad (3.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75\cdot t_{др}\ \text{, людино-годин.} \qquad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 787,5 / (18 · 1,1) = 39,77 людино-годин.

$t_{до}$ = 0,75 · 39,77 = 29,82 людино-годин.

$t_д$=39,77 + 29,82 = 69,59 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 11,45+ 35,79 + 28,63 + 143,18 + 69,59 =338,64 людино-годин.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 60грн / год, отримуємо:

$$З_{ЗП} = 338{,}64 \cdot 60 = 20318{,}4 \text{грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.12) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 143{,}18 \cdot 14 = 2004{,}52 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 20318{,}4 + 2004{,}52 = 22322{,}92 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс,} \quad (3.13)$$

де B_k- число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 338{,}64 / 1 \cdot 176 \approx 2 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для автоматизації процесу проектування типових виробів за рахунок розширення бібліотек в найбільш поширених системах автоматизованого проектування: КОМПАС, Autodesk Inventor, SolidWorks. Вартість даного програмного забезпечення становить 22322,92грн. і не вимагає додаткових витрат при впровадженні та експлуатації програми. Очікуваний час розробки становить 2 місяці. Цей термін пов'язаний з числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Сучасний ринок програмного забезпечення автоматизації підготовки виробництва насичений найрізноманітнішими універсальними САПР, що здатні істотно полегшити роботу проектувальника (звільнити проектувальника від проведення багатьох рутинних операцій, що відчутно підвищує швидкість створення проектів, знижує ймовірність допустити помилки в процесі проектування). Разом з тим, не дивлячись на величезну кількість такого виду інструментальних засобів автоматизації інженерної діяльності, універсальні системи часто недостатньо ефективні для вирішення конкретного завдання користувача, не мають алгоритмів автоматизації окремих вузькоспрямованих дій (рутинних операцій, типових проектних рішень) в певних областях проектування, зачасту інформаційно несумісні.

Більшість застосовуваних у промисловості тривимірних САПР зачасту вирішують задачі усередненого підприємства без врахування його специфіки. Підвищення конкуренції на ринку вимагає від підприємств постійного оновлювання номенклатура виробів. Оскільки номенклатура виробів постійно оновлюється на базі вже розроблених вузлів,тому в більшості випадків робота проектувальника зводиться до модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних. Тому для ефективної роботи сучасного підприємства необхідно розробляти спеціалізовані САПР, що здатні розширити бібліотеку нових виробів та вирішувати завдання розрахунку і проектування конкретного класу виробів. Більшість застосовуваних у промисловості тривимірних САПР можуть бути використані як основа для побудови спеціалізованих САПР, які вирішують завдання розрахунку і проектування конкретного класу виробів. При цьому необхідно об'єднати розрахунковий модуль, який визначає розмірні та інші параметри проектованого об'єкта з уже наявним в САПР тривимірним геометричним ядром. Цей процес здійснюється за допомогою API - інтерфейс прикладної програми. Набір таких інтерфейсів забезпечує взаємозв'язок між зовнішніми

модулями прикладної програми і низькорівневими функціями ядра, а так само між компонентами ядра – різними бібліотеками, що істотно підвищує потенційні можливості застосування універсальних систем в специфічних предметних областях.

Також, слід відзначити, що більшість виробників в процесі проектування виробів використовують декілька різних пакетів CAD - систем. Причинами для цього є: зростаюча складність проектованих виробів; розвиток глобального конструкторського аутсорсингу; об'єднання і придбання. Тому розробка програмного забезпечення,що буде здатне розширити бібліотеку нових виробів та вирішувати завдання розрахунку і проектування конкретного класу виробів для найбільш поширених CAD - систем є актуальною і потребує подальшого вивчення.

Метою кваліфікаційної роботи є розробка програмного забезпечення з використанням API інтерфейсу для розширення бібліотек нових виробів в найбільш поширених системах автоматизованого проектування: КОМПАС-3D, Autodesk Inventor, SolidWorks.

Програмний модуль являє собою систему, в якій реалізовані:

- можливість створення параметричної збірки проектованого механізму для внесення ряду розмірів об'єкту в змінні моделі;
- можливість під’єднання до однієї з CAD-систем (КОМПАС-3D, Autodesk Inventor, SolidWorks) для об'єднання розрахункового модулю, який визначає розмірні та інші параметри проектованого об'єкта з вже наявним в САПР тривимірним геометричним ядром;
- можливість обрання тривимірної моделі, яку необхідно модифікувати (модуль зчитує поточні значення параметрів моделі та динамічно завантажує їх у інтерфейс, де користувач має змогу змінити їх та перебудувати модель).
- можливість контролю із зовнішньої програми за допомогою API-інтерфейсів, які надають доступ до параметрів збірки;

– можливість збереження поточного результату (нової геометрії виробу);
– програмно-апаратна переносимість.

Розроблене програмне забезпечення призначене для застосування в будь-якому підприємстві з подібним родом діяльності і схожими функціональними вимогами.

Призначення розробленого модулю:

– розширити бібліотеку нових виробів та вирішити завдання розрахунку і проектування конкретного класу виробів враховуючи специфіку підприємства;
– оновити номенклатуру виробів на базі вже розроблених вузлів за рахунок модифікації раніше створеної геометрії вузла відповідно до нових розрахункових даних.

Розроблений програмний модуль дозволить підвищити продуктивність праці проектувальників (за рахунок звільнення проектувальника від проведення багатьох рутинних операцій), скоротити терміни проектування та знизити витрати на розробку технічної документації, збільшити якість проектування за рахунок підвищення точності виготовлення виробів.

Програмний модуль САПР реалізовано на мові програмування C#. Програмне забезпечення (хоча б одна з CAD-систем), а саме: Autodesk Inventor (2015 SP1 і вище), SolidWorks (2013 SP3 і вище), КОМПАС-3D (2014 V15 і вище)повинно бути встановлене завчасно. CAD – системи повинні підтримувати API – технології.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (338,64 людино-годин), підраховані витрати на створення програмного забезпечення (22322,92 грн.) і очікуваний період розробки (становить 2 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Большаков В.П., Бочков А.Л., Лячек Ю.Т. Проблемы обмена графическими данными между CAD-системами. Компьютерные инструменты в образовании, Вып. №2 (2013). [Електронний ресурс] – Режим доступу до ресурсу: http://cte.eltech.ru/ojs/index.php/kio/article/view/1342/1339 (дата звернення: 08.03.2021).

2. Геометрическое ядро САПР [Електронний ресурс]/ URL: https://helpiks.org/9-69882.html (дата звернення: 08.03.2021).

3. Кунву Ли. Основы САПР CAD/CAM/CAE/ Ли Кунву - СПб.: Питер, 2004.

4. Норенков И. П. Основы автоматизированного проектирования/ И. П. Норенков - М.: МГТУ имени Н.Э.Баумана, 2002.

5. Подбельский В.В. Язык C#. Базовый курс. 2-е изд./ В.В.Подбельский, В. Борисок, Ю. Корвель; Питер, 2007. – 89 с.

6. Прикладний програмний інтерфейс [Електронний ресурс]//Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс (дата звернення: 08.03.2021).

7. Потемкин А. Трехмерное твердотельное моделирование. / А.Потемкин. – М.: КомпьютерПресс, 2002. – 296 с.: ил/.

8. Спірінцев В.В. Розробка функціональної схеми процесу автоматизованого проектування/ В.В.Спірінцев, І.В.Пихтєєва, Ю.О.Дмитрієв// Регіональний міжвузівський збірник наукових праць. Дніпропетровськ: Системні технології.-2013. Випуск 1(84). - С.129-135.

9. Спірінцев В.В. Розробка спеціалізованого програмного модуля для проектування типових деталей /В.В.Спірінцев// Сучасні проблеми геометричного моделювання// Збірник праць XIV Міжнародної науково-практичної конференції. - Мелітополь, ТДАТУ 2012. – С.103-107.

10. Журнал **«САПР и Графика»** [Електронний ресурс]/URL: https://sapr.ru(дата звернення: 08.03.2021).

11. Autodesk [Електронний ресурс]/ URL: https://www.autodesk.ru/ (дата звернення: 08.03.2021).

12. Component Object Model [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Component_Object_Model (дата звернення: 08.03.2021).

13. C3D Toolkit [Електронний ресурс]/ URL: https://ascon.ru/products/1259/review/ (дата звернення: 08.03.2021).

14. C Sharp [Електронний ресурс]// Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/C_Sharp (дата звернення: 08.03.2021).

15. Delphi (язык_программирования) [Електронний ресурс]// Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/ Delphi_(язык_программирования) (дата звернення: 08.03.2021).

16. Parasolid - ядро твердотельного геометрического моделирования [Електронний ресурс]/ URL: https://pro-spo.ru/-cad-cam-windows/3484-parasolid-yadro-tverdotelnogo-geometricheskogo-modelirovaniya (дата звернення: 08.03.2021).

17. Pascal [Електронний ресурс]// Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Pascal (дата звернення: 08.03.2021).

18. Solidworks [Електронний ресурс]/ URL: https://www.solidworks.com/ru (дата звернення: 08.03.2021).

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг коду користувацького інтерфейсу

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;
using System.Threading;

namespace SolidTest
{
   public partial class Form1 : Form
   {
      CADet.ProggramObject App = new CADet.ProggramObject();
      Boolean NeedProgramOpen = true;
      Boolean OpenPreview = true;
      String fileFormat = "";
      public Form1()
      {
         InitializeComponent();
         ProgramComboBox.SelectedIndex = 1;
      }

      private void ReBuild_Click(object sender, EventArgs e)
      {
         App.ReBuild();
      }

      private void GetProggram_Click(object sender, EventArgs e)
      {
         if (NeedProgramOpen)
         {
            switch (ProgramComboBox.Text)

{
               case "AutoDesk":
                  App = new CADet.AutodeskObject();
                  fileFormat = "AutoDesk model|*.ipt";
                  break;
               case "SolidWORKS":
                  App = new CADet.SolidObject();
                  fileFormat = "SolidWorks model|*.SLDPRT";
                  break;
               case "Kompas":
                  App = new CADet.MyKompasObject();
                  fileFormat = "Kompas model|*.a3d";
                  break;
               default:
                  MessageBox.Show("NotFound");
                  return;
            }
            App.RootForm = this;
```

```
            if (App.init())
            {
               ProgramComboBox.Enabled = false;
               ConnectDoneIMG.Visible = true;
               MainPanel.Enabled = true;
               GetProggram.Text = "Від'єднатися";
               FilePanel.Enabled = true;
               NeedProgramOpen = false;
            }
         }else
         {
            if (App != null)
            {
               App.CloseAllInstance();
            }
            FileName.Text = "";
            ReBuild.Enabled = false;
            ProgramComboBox.Enabled = true;
            ConnectDoneIMG.Visible = false;
            MainPanel.Enabled = false;
                              GetProggram.Text = "Встановити зв'язок";
            FilePanel.Enabled = false;
            NeedProgramOpen = true;
            FilePreview.Visible = false;
            ParamsPanel.Controls.Clear();
         }

      }

      private void close_Click(object sender, EventArgs e)
      {
         this.Close();
      }

      private void pictureBox1_Click(object sender, EventArgs e)
      {
         if (OpenPreview)
         {
            this.FilePreview.Width = 30;
            this.FilePreview.Height = 30;
            this.Width = 1000;
            OpenPreview = false;
         }
         else
         {
            this.FilePreview.Width = 40;
            this.FilePreview.Height = 40;
            this.Width = 391;
            OpenPreview = true;
         }

      }

      private void btnFileOpen_Click(object sender, EventArgs e)
      {
         using (OpenFileDialog dialog = new OpenFileDialog())
         {
            dialog.Filter = fileFormat;

if (dialog.ShowDialog() == DialogResult.OK)
{
               App.OpenFile(dialog.FileName);               App.SavePicterForPreview();
               try {
```

```
                    PreviewPicture.ImageLocation = @"F:\temp.jpg";
                }
                catch(Exception){}
                FilePreview.Visible = true;
                FileName.Text = dialog.SafeFileName;

                MainPanel.Enabled = true;
                ReBuild.Enabled = true;
                App.SetSettings();
            }
        }
    }
}
```

Лістинг коду головного об’єкта

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Inventor;
using System.ComponentModel;
using System.Data;
using System.Drawing;

namespace CADet
{
    public class ProggramObject
    {

        public Form RootForm;
        public Control MainPanel;

        protected Dictionary<string, string> ParamsList = new Dictionary<string, string>();
        protected Dictionary<string, System.Windows.Forms.TextBox> ParamsListOnPanel = new
Dictionary<string, System.Windows.Forms.TextBox>();
        protected string ParamsSearchSuffix = "My";

        protected string FilePath;

        //Settings
        public ProggramObject() {

        }

        //Public methods

        public virtual void CloseAllInstance() { }
        public virtual void ReBuild() { }
        public virtual void GetParamsDictionary() { }
        public virtual void SavePicterForPreview() { }

        public Boolean init()
        {
            preInit();
            return initApp();
        }
        public virtual Boolean initApp()
        {
            return false;
        }
        public virtual void SetSettings()
```

```
        {
            ParamsList.Clear();
            ParamsListOnPanel.Clear();
            GetParamsDictionary();
            AppendParamsIntoForm();
        }
        public virtual void OpenFile(string filePath)
        {
            FilePath = filePath;
            OpenFileByPath();
        }

        protected virtual void OpenFileByPath() { }

        //private methods
        private void preInit()
        {
            MainPanel = RootForm.Controls["MainPanel"].Controls["ParamsPanel"];
        }
        private void AppendParamsIntoForm() {
            if (ParamsList.Count == 0) return;

            var v_i = 0;

foreach (KeyValuePair<string, string> v_param in ParamsList)
            {
                AddParamBlock(v_param, v_i);
                v_i++;
            }
        }
        private void AddParamBlock(KeyValuePair<string, string> v_param, int v_index)
        {
            int v_shift = 10;
            Panel v_panel = new Panel();
            v_panel.Location = new System.Drawing.Point(10, v_index * 25 + v_shift);
            v_panel.Width = 280;
            v_panel.Height = 25;
            MainPanel.Controls.Add(v_panel);


            AddLabel(v_param.Key, v_panel);
            ParamsListOnPanel.Add(v_param.Key, AddTextBox(v_param.Value, v_panel));
        }
        private void AddLabel(string v_value, Panel v_panel)
        {
            Label v_label = new Label();
            v_label.Text = v_value;

            v_label.Location = new System.Drawing.Point(10, 0);
            v_panel.Controls.Add(v_label);
        }
        private System.Windows.Forms.TextBox AddTextBox(string v_value, Panel v_panel)
        {
            System.Windows.Forms.TextBox v_textBox = new System.Windows.Forms.TextBox();
            v_textBox.Text = v_value;
                v_textBox.Width = 100;
            v_textBox.Location = new System.Drawing.Point(150, 0);
            v_panel.Controls.Add(v_textBox);
            return v_textBox;
        }

    }
}
```

Лістинг коду об'єкта KompasObject

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using Kompas6Constants;
using Kompas6API5;
using System.Diagnostics;

namespace CADet
{
   class MyKompasObject : ProggramObject
   {
      private KompasObject kompas;
      private ksDocument3D doc3D;

      public override Boolean initApp()
      {
         CloseAllInstance();

         Type t = Type.GetTypeFromProgID("KOMPAS.Application.5");
                  kompas = (KompasObject)Activator.CreateInstance(t);
         return true;
      }
      public override void CloseAllInstance()
      {
         //убиваем процесс
         Process[] processes = Process.GetProcessesByName("KOMPAS");
         foreach (Process proces in processes)
         {
            proces.CloseMainWindow();
            proces.Kill();
         }
      }

public override void ReBuild() {

         ksPart singlePart = (ksPart)doc3D.GetPart(-1);
         ksVariableCollection VarCol = (ksVariableCollection)singlePart.VariableCollection();

         ksVariable Var;

         for (var v_i = 1; v_i < VarCol.GetCount(); v_i++)
         {
            Var = (ksVariable)VarCol.GetByIndex(v_i);
            if (Var.name.StartsWith(ParamsSearchSuffix))
            {
               Var.value = Convert.ToDouble(ParamsListOnPanel[Var.name.Substring(2)].Text);
            };
         }
         singlePart.RebuildModel();
         doc3D.RebuildDocument();
      }
      public override void GetParamsDictionary()
      {
         ksPart singlePart = (ksPart)doc3D.GetPart(-1);
         ksVariableCollection VarCol = (ksVariableCollection)singlePart.VariableCollection();

         ksVariable Var;
         for (var v_i = 0; v_i < VarCol.GetCount(); v_i++ )
         {
```

```
            Var = (ksVariable)VarCol.GetByIndex(v_i);
            if (Var.name.StartsWith("My"))
            {
               ParamsList.Add(Var.name.Substring(2), (Convert.ToDouble(Var.value)).ToString());
            };
         }
}
      public override void SavePicterForPreview()
      {
         kompas.Visible = true;
         ksPart singlePart = (ksPart)doc3D.GetPart(-1);
         doc3D.SaveAsToAdditionFormat(@"F:\temp1.jpg", 2);
         return;
      }
      protected override void OpenFileByPath()
      {
         kompas.Visible = true;

         doc3D = (ksDocument3D)kompas.Document3D();
         doc3D.Open(FilePath, false);
      }
   }
}
```

Лістинг коду об’єкта AutodeskObject

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Inventor;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace CADet
{
   class AutodeskObject : ProggramObject
   {
      Inventor.Application v_invApp;
      Inventor._Document v_file;

      public override Boolean initApp()
      {
         CloseAllInstance();
         try
         {
            Type v_invAppType = Type.GetTypeFromProgID("Inventor.Application");
            v_invApp = (Inventor.Application)Activator.CreateInstance(v_invAppType);
         }
         catch (System.Exception ex2)
         {
            return false;
         }

         return true;
      }
      public override void CloseAllInstance()
      {
         //убиваем процесс

Process[] processes = Process.GetProcessesByName("INVENTOR");
         foreach (Process proces in processes)
         {
            proces.CloseMainWindow();
```

```
                proces.Kill();
            }
        }
        public override void ReBuild()
        {
            Inventor._Document doc = v_invApp.ActiveDocument;
            dynamic otherdoc = v_invApp.ActiveDocument;
            Parameters oParameters = otherdoc.ComponentDefinition.Parameters();
            ModelParameters ModelParams = oParameters.ModelParameters;

            for (var v_i = 1; v_i < ModelParams.Count; v_i++)
            {
                if (ModelParams[v_i].Name.StartsWith(ParamsSearchSuffix))
                {
                    ModelParams[v_i].Value = Convert.ToDouble(ParamsListOnPanel[ParamsSearchSuffix +
ModelParams[v_i].Name.Substring(2)].Text) / 10;
                };
            }

            v_file.Update();
        }
        public override void GetParamsDictionary()
        {
            dynamic otherdoc = v_invApp.ActiveDocument;
            Parameters oParameters = otherdoc.ComponentDefinition.Parameters();
            ModelParameters ModelParams = oParameters.ModelParameters;
            for (var v_i = 1; v_i < ModelParams.Count; v_i++) {

    if (ModelParams[v_i].Name.StartsWith(ParamsSearchSuffix))
                {
                    ParamsList.Add(ModelParams[v_i].Name, (Convert.ToDouble(ModelParams[v_i].Value) *
10).ToString());
                };
            }
        }
        public override void SavePicterForPreview()
        {
            v_invApp.Visible = true;
            ((Inventor.Application)(v_invApp)).ActiveView.GoHome();
            v_invApp.ActiveDocument.SaveAs(@"F:\temp.jpg", true);
        }
        protected override void OpenFileByPath()
        {
            v_invApp.Visible = true;
            v_file = v_invApp.Documents.Open(FilePath);
        }
    }
}
```

Лістинг коду об’єкта SolidObject

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SolidWorks.Interop.sldworks;
using SolidWorks.Interop.swcommands;
using SolidWorks.Interop.swconst;
using System.Diagnostics;
using System.Threading;

namespace CADet
{
```

```
        class SolidObject : ProggramObject
        {
            private SldWorks v_App;      //приложение
            private IModelDoc2 v_Model;    //чертеж
            public override Boolean initApp()
            {
                CloseAllInstance();
                try
                {
                    //создаем приложение
                    Guid v_myGuid = new Guid("0D825E02-9000-4D82-B4AB-D6BDC2872797");

                    object v_processSW =
System.Activator.CreateInstance(System.Type.GetTypeFromCLSID(v_myGuid));
                    v_App = (SldWorks)v_processSW;
                    Thread.Sleep(5000);
                }
                catch (InvalidCastException v_e)
                {
                    return false;
                }

                return true;

            }
            public override void CloseAllInstance()
            {
                //убиваем процесс
                Process[] processes = Process.GetProcessesByName("SLDWORKS");
                foreach (Process proces in processes)
                {
                    proces.CloseMainWindow();
                    proces.Kill();
                }
            }
            public override void ReBuild() {

                IModelDoc2 Part = v_App.IActiveDoc2;
                EquationMgr swEquationMgr = default(EquationMgr);
                swEquationMgr = (EquationMgr)v_Model.GetEquationMgr();

                int v_count = swEquationMgr.GetCount();
                for (var v_i = 0; v_i < v_count; v_i++)
                {
                    string v_param = swEquationMgr.get_Equation(v_i);
                    string v_paramName = v_param.Split(new Char[] { '=' })[0];
                    if (v_paramName.Contains(ParamsSearchSuffix))
                    {
                        swEquationMgr.Add2(v_i, v_paramName + " =" +
Convert.ToDouble(ParamsListOnPanel[v_paramName].Text).ToString() + "mm", false);
                        swEquationMgr.EvaluateAll();
                        swEquationMgr.Delete(v_i + 1);
                    };
                }

                Part.EditRebuild3();
                return;
            }

        public override void GetParamsDictionary()
            {

                IModelDoc2 Part = v_App.IActiveDoc2;
```

```
            EquationMgr swEquationMgr = default(EquationMgr);
            swEquationMgr = (EquationMgr)v_Model.GetEquationMgr();

            for (var v_i = 0; v_i < swEquationMgr.GetCount(); v_i++)
            {
                string v_param = swEquationMgr.get_Equation(v_i);
                string v_paramName = v_param.Split(new Char[] {'='})[0];
                if (v_paramName.Contains(ParamsSearchSuffix))
                {
                    ParamsList.Add(v_paramName, swEquationMgr.Value[v_i].ToString());
                };
            }
        }
        public override void SavePicterForPreview()
        {

        }
        protected override void OpenFileByPath()
        {
            v_App.Visible = true;
            v_Model = (IModelDoc2)v_App.OpenDoc(FilePath, 1);
            v_Model = v_App.IActiveDoc2;
        }
    }
}
```

ДОДАТОК Б

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім'я файла	**Опис**
Пояснювальні документи	
Диплом_Каменєв.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_Каменєв.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Каменєв.pptx	Презентація дипломного проекту