Министерство науки и высшего образования Российской Федерации   
Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Пояснительная записка по лабораторному проекту

«Разработка плагина «Резьбовая шпилька» для САПР   
КОМПАС - 3D»  
по дисциплине «Основы разработки САПР»

Выполнил:

Студент гр. 588-2

\_\_\_\_\_\_\_\_\_ Л.Л.Мурыхин

« » 2022

Проверил:

к.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_ А.А.Калентьев  
« » 2022

Томск 2022

**Реферат**

Пояснительная записка 37 с., 18 рис., 15 таблиц, 7 источников.

Ключевые слова: КОМПАС-3D, VISUAL STUDIO, C#, КОМПАС API, ПЛАГИН, РЕЗЬБОВАЯ ШПИЛЬКА, САПР.

Целью данной работы является разработка плагина для создания трехмерной модели резьбовой шпильки, согласно заданным параметрам, для системы автоматизированного проектирования «КОМПАС-3D v.20»

В результате разработан плагин, строящий трехмерную модель шпильки в КОМПАС-3D.

Отчет по пояснительной записке выполнен в текстовом редакторе Microsoft Word 2016.

Оглавление

[1. Описание САПР 4](#_Toc99005076)

[1.1 Описание программы КОМПАС-3D 4](#_Toc99005077)

[1.2 Описание API САПР КОМПАС-3D 5](#_Toc99005078)

[1.3 Обзор аналогов 10](#_Toc99005079)

[2 Описание предмета проектирования 12](#_Toc99005080)

[3 Стек технологий и системные требования 13](#_Toc99005081)

[4. Описание плагина 14](#_Toc99005082)

[4.1 Диаграмма классов 14](#_Toc99005083)

[4.2 Макет пользовательского интерфейса 19](#_Toc99005084)

[5 Тестирование программы 22](#_Toc99005085)

[5.1 Функциональное тестирование 22](#_Toc99005086)

[5.2 Модульное тестирование 29](#_Toc99005087)

[5.3 Нагрузочное тестирование 32](#_Toc99005088)

[Заключение 34](#_Toc99005089)

[Список используемых источников 35](#_Toc99005090)

[Приложение А 36](#_Toc99005091)

# 1. Описание САПР

## 1.1 Описание программы КОМПАС-3D

ПРОЕКТИРОВАНИЕ— это комплекс работ с целью получения описаний нового или модернизируемого технического объекта, достаточных для реализации или изготовления объекта в заданных условиях. В процессе проектирования возникает необходимость создания описания, необходимого для построения еще не существующего объекта. Получаемые при проектировании описания бывают окончательными или промежуточными. Окончательные описания представляют собой комплект конструкторско-технологической документации в виде чертежей, спецификаций, программ для ЭВМ и автоматизированных комплексов и т.д.[1]

САПР – Система автоматизированного проектирования автоматизированная система, реализующая информационную технологию выполнения функций проектирования, представляет собой организационно техническую систему, предназначенную для автоматизации процесса проектирования, состоящую из комплекса технических, программных и других средств автоматизации его деятельности [2].

КОМПАС-3D – система трехмерного проектирования, ставшая стандартом для тысяч предприятий, благодаря сочетанию простоты освоения и легкости работы с мощными функциональными возможностями твердотельного и поверхностного моделирования. Ключевой особенностью продукта является использование собственного математического ядра С3D и параметрических технологий, разработанных специалистами АСКОН. КОМПАС-3D обеспечивает поддержку наиболее распространенных форматов 3D-моделей (STEP, ACIS, IGES, DWG, DXF), что позволяет организовывать эффективный обмен данными со смежными организациями и заказчиками, использующими любые CAD / CAM / CAE-системы в работе [3].

## 1.2 Описание API САПР КОМПАС-3D

API (англ. Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой.

В КОМПАС-3D существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и дополняют друг друга. Обе версии программных интерфейсов в равной мере поддерживаются и развиваются с учетом самих изменений в системе. В основном, для создания полноценных подключаемых модулей достаточно методов и свойств интерфейсов API 5.

Главным интерфейсом API системы КОМПАС-3D является KompasObject. Получить указатель на этот интерфейс (на интерфейс приложения API 5) можно при работе под управлением внешнего приложения (контроллера) – после вызова стандартной системной функции. Методы этого интерфейса реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы.

Ниже в таблице 1.1 представлены основные свойства и методы интерфейса KompasObject.

Таблица 1.1 – Методы и свойства интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные данные | Тип возвращаемых данных | Описание |
| Document3D() |  | ksDocument | Метод для получения указателя на интерфейс трехмерного графического документа (детали или сборки) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| GetParamStruct  (short structType) | structType – тип интерфейса параметров | StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible |  | bool | Свойство видимости приложения |
| Quit() |  |  | Метод для закрытия активного окна приложения КОМПАС |

В таблице 1.2 представлены методы интерфейса ksEntity, необходимые для разработки плагина.

Таблица 1.2 – Методы интерфейса ksEntity

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| Create() | bool | Создать объект модели |
| GetDefinition() | IUnkown | Получить указатель на интерфейс параметров объектов и элементов |
| Update() | bool | Изменить свойства объекта (используя ранее установленные свойства) |

В таблице 1.3 представлены свойства и методы интерфейса ksDocument2D, необходимые для разработки плагина

Таблица 1.3 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| ksRectangle  (ksRectangleParam param, int style) | param– параметры прямоугольника.  style–стиль линии. | int | Получить указатель на прямоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksCircle  (double xc, double yc, double rad, int style) | xc, yc – координаты центра окружности.  rad – радиус окружности.  style – стиль линии. | int | Получить указатель на окружность на двумерной плоскости либо 0 в случае ошибки |

В таблице 1.4 представлены свойства и методы интерфейса ksDocument3D, необходимые для разработки плагина.

Таблица 1.4 – Методы интерфейса ksDocument3D

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| Create (bool invisible, bool \_typeDoc) | invisible – признак режима редактирования документа (true – невидимый режим,  false – видимый режим),  typeDoc – тип документа (true – деталь, false – сборка). | bool | Создать документ-модель (деталь или сборку) |

Продолжение таблицы 1.4

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 1.5 представлены методы интерфейса ksPart, необходимые для разработки плагина.

Таблица 1.5 – Свойства и методы интерфейса ksPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Тип возвращаемых данных | Описание |
| EntityCollection  (short objType) | objType – тип объектов, содержащихся в массиве. | ksEnintyCollection | Формирует массив объектов и возвращает указатель на его интерфейс |
| GetDefaultEntity  (short objType) | objType – тип объекта | ksEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart (int type) | Type – тип компонента | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity (short objType) | objType – тип объекта | ksEntity | Создать новый интерфейс объекта и получить указатель на него |

В таблице 1.6 представлены типы объектов документа-модели, необходимые для разработки плагина.

Таблица 1.6 – Некоторые типы объектов документа-модели

|  |  |  |
| --- | --- | --- |
| Идентификатор объекта | Название объекта | Интерфейс параметров |
| o3d\_unknown | Неизвестный (включает все объекты) |  |
| o3d\_planeXOZ | Плоскость XOZ | ksPlaneParam |
| o3d\_planeYOZ | Плоскость YOZ | ksPlaneParam |
| o3d\_planeXOY | Плоскость XOY | ksPlaneParam |
| o3d\_sketch | Эскиз | ksSketchDefinition |
| o3d\_face | Грань | ksFaceDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_fillet | Операция «скругление» | ksFilletDefinition |

## 1.3 Обзор аналогов

**Плагин «Custom Screw Creator» для САПР Fusion 360**

Плагин «Custom Screw Creator» позволяет моделировать болты и шпильки с различными изменяемыми параметрами. Интерфейс плагина представлен на рисунке 1.1. Пример модели, построенной с помощью плагина, представлен на рисунке 1.2.[4]

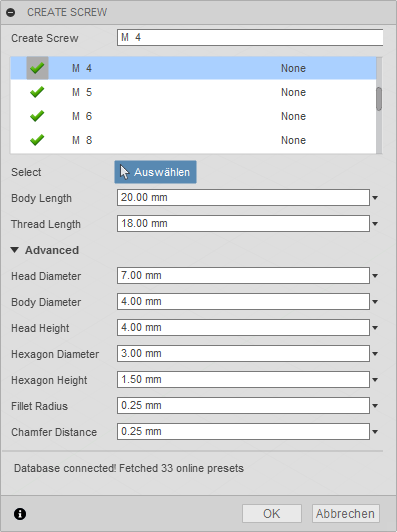


Рисунок 1.1 – Интерфейс плагина «Custom Screw Creator»

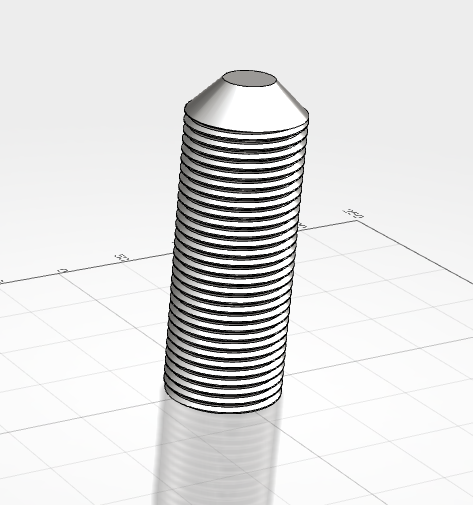


Рисунок 1.2 – Модель, построенная с помощью плагина «Custom Screw Creator»

# 2 Описание предмета проектирования

Предметом проектирования является модель резьбовой шпильки (чертёж резьбовой шпильки представлен на рисунке 2.1).

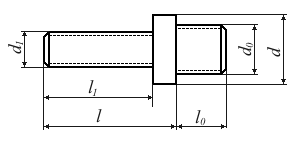


Рисунок 2.1 – Чертёж резьбовой шпильки

Модель шпилька обладает следующими изменяемыми параметрами:

1. Диаметр шпильки (*d* на рисунке 2.1);
2. Диаметр ввинчиваемой резьбы (d0 на рисунке 2.1);
3. Диаметр гаечной резьбы (d1 на рисунке 2.1);
4. Длина шпильки (l на рисунке 2.1);
5. Длина ввинчиваемой резьбы (l0 на рисунке 2.1);
6. Длина гаечной резьбы (l1 на рисунке 2.1).

Ограничение параметров модели:

1. Диаметр шпильки (4 – 16 мм);
2. Диаметр ввинчиваемой резьбы (4 – 16 мм);
3. Диаметр гаечной резьбы (4 – 16 мм);
4. Длина шпильки (12 – 100 мм);
5. Длина ввинчиваемой резьбы (12 – 48 мм);
6. Длина гаечной резьбы (12 – 100 мм);
7. Значение длины гаечной резьбы (l1) не должно превышать значение длины шпильки (l);
8. Значение длины ввинчиваемой резьбы (l0) не должно превышать значение длины шпильки (l).

# 3 Стек технологий и системные требования

Язык программирования: C#;

Среда разработки: Visual Studio 2022 версия 17.1.0 с применением Microsoft.NET Framework 4.7.2;

Тестирование с помощью библиотек: NUnit версия 3.13.3.

САПР КОМПАС 3D v20.

Для реализации пользовательского интерфейса использовался API WinForm.

Взаимодействие плагина с системой КОМПАС (с функциями моделирования, математическими функциями ядра системы и пр.) осуществляется посредством программных интерфейсов, называемых API. В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и взаимно дополняют друг друга [5].

Для создания плагина, будет достаточно методов и свойств интерфейсов API 5.

# 4. Описание плагина

## 4.1 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.[6] Целью создания диаграммы классов является графическое представление статической структуры элементов системы.

Изначальная диаграмма классов плагина представлена на рисунке 3.1.

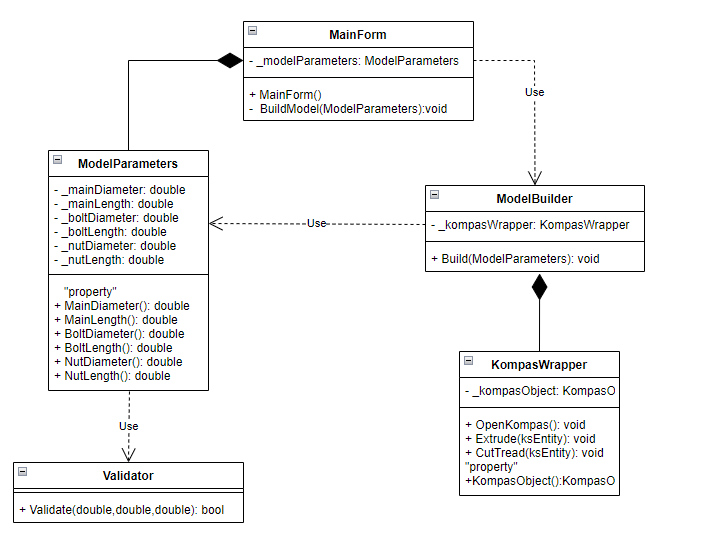


Рисунок 4.1 – Диаграмма классов плагина

Класс «MainForm» содержит в себе экземпляр классa «ModelParameters». «ModelParameters» хранит в себе параметры модели и использует «Validator» для проверки их корректности. «ModelBuilder» содержит в себе метод для построения модели. Класс «KompasWrapper» используется для работы с САПР КОМАПС 3D.

Диаграмма классов после добавления дополнительной функциональности представлена на рисунке 4.2.

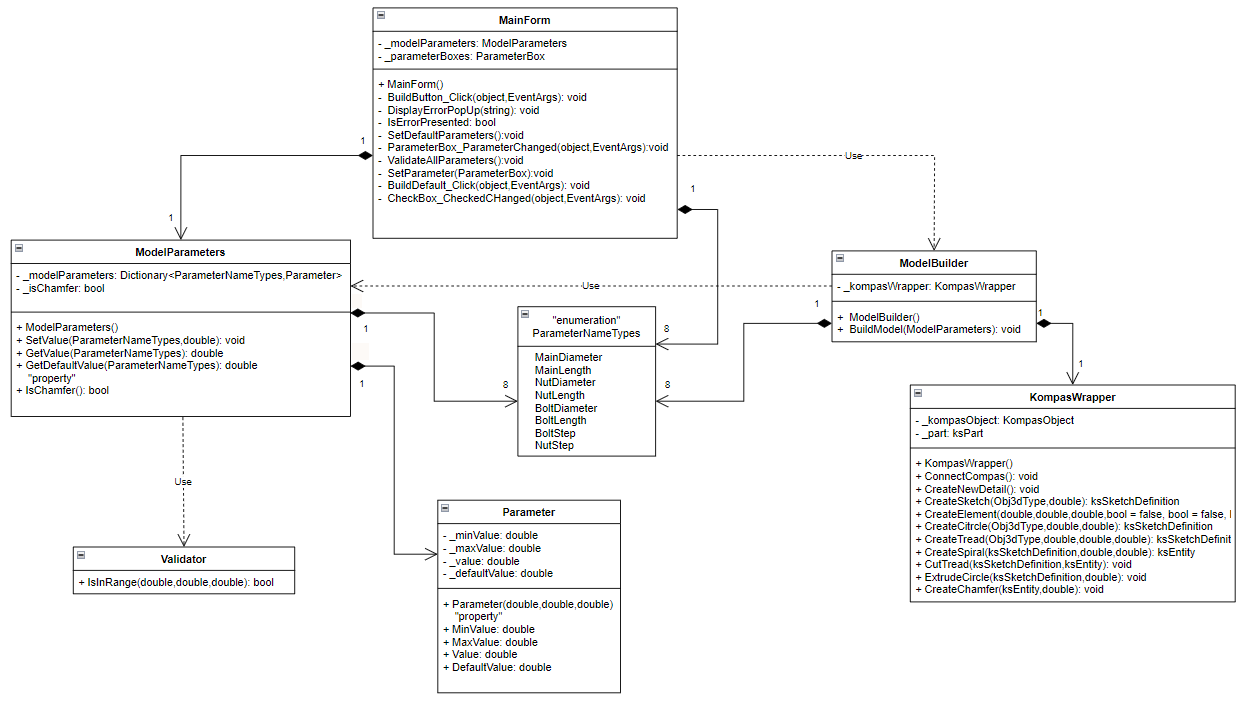


Рисунок 4.2 – Диаграмма классов плагина с дополнительной функциональностью

В процессе работы изначальная UML-диаграмма претерпела следующие изменения:

1. Было добавлено перечисление имён параметров ParameterNameTypes;
2. Был добавлен класс параметра Parameter;
3. Были переработаны существующие классы.

В таблицах 4.1 – 4.6 представлены описания полей, свойств и методов разработанных классов.

Таблица 4.1 – Класс Parameter

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых  данных | Описание |
| \_minValue | double | Закрытое поле. Минимальное значение параметра. |
| \_maxValue | double | Закрытое поле. Максимальное значение параметра. |
| \_value | double | Закрытое поле. Значение параметра. |
| \_defaultValue | double | Закрытое поле. Значение параметра по умолчанию. |
| DefaultValue() | double | Возвращает и устанавливает значение параметра по умолчанию. |
| Value() | double | Возвращает и устанавливает значение параметра. |
| MinValue() | double | Возвращает и устанавливает минимальное значение. |
| MaxValue() | double | Возвращает и устанавливает максимальное значение. |
| IsInRange(  double,  double,double) | bool | True, если значение принадлежит диапазону, иначе False |
| Parameter(  double,double,  double,double) |  | Конструктор параметра |

Таблица 4.2 – Класс ModelParameters

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_modelParameters | double | Закрытое поле. Словарь числовых параметров модели |
| \_isChamfer | bool | Закрытое поле. Параметр модели, отвечающий за нанесение фаски. |
| SetValue(  ParameterNameTypes,  doublу) |  | Устанавливает значение параметра, проверяя значения зависимых параметров |
| GetValue (ParameterNameTypes) | double | Возвращает значение параметра. |
| GetDefaultValue (ParameterNameTypes) | double | Возвращает значение параметра по умолчанию. |
| ModelParameters() |  | Конструктор параметров модели |

Таблица 4.3 – Класс ModelBuilder

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_kompasWrapper | KompasWrapper | Закрытое поле. Объект KompasWrapper |
| BuildModel(ModelParameters) | double | Создаёт модель резьбовой шпильки |
| ModelBuilder() | double | Конструктор ModelBuilder |

Таблица 4.4 – Класс KompasWrapper

|  |  |  |
| --- | --- | --- |
| Название | Тип  возвращаемых данных | Описание |
| \_kompasObject | KompasObject | Закрытое поле. Объект API КОМПАС-3D |
| \_part | ksPart | Закрытое поле. Объект детали |
| ConnectKompass() |  | Подключается к открытому компас-3D. |
| CreateNewDetail() |  | Создаёт новую деталь в компас-3D |
| CreateSketch(  Obj3dType, double) | ksSketchDefinition | Создаёт новый эскиз в плоскости |
| CreateElement(  double,double,  double, bool,  bool, bool,  double) |  | Создаёт конструктивный элемент шпильки |
| CreateCircle(Obj3dType,  double, double) | ksSketchDefinition | Создаёт эскиз окружности |
| CreateTread(Obj3dType,  double, double, double) | ksSketchDefinition | Создаёт эскиз сечения резьбы |
| ExtrudeCircle(  ksSketchDefinition, double) |  | Выдавливает цилиндр |
| CreateChamfer(  ksEntity, double) |  | Добавляет фаску на грань |
| CreateSpiral(  ksSketchDefinition, double, double) | ksEntity | Создаёт объект спирали |
| CutTread(  ksSketchDefinition, ksEntity) | ksEntity | Вырезает резьбу по спирали |
| KompasWrapper() |  | Конструктор KompasWrapper |

## 4.2 Макет пользовательского интерфейса

Макет пользовательского интерфейса создан с помощью Windows Form. На рисунке 4.3 представлен макет пользовательского интерфейса для ввода параметров модели. Так же на рисунке 4.3 представлено, как будет отображаться ввод некорректных параметров (над полем ввода появляется сообщение об ошибке), если значения введены правильно, то сообщение не отображается. Перед пользователем представлены 6 полей, предназначенный для ввода параметров (в мм) детали. Помимо этого, присутствует кнопка для построения модели, в дальнейшем при нажатии на которую производиться построение модели в открытой САПР КОМАПС 3D.

Модель пользовательского интерфейса представлена на рисунке 4.3.

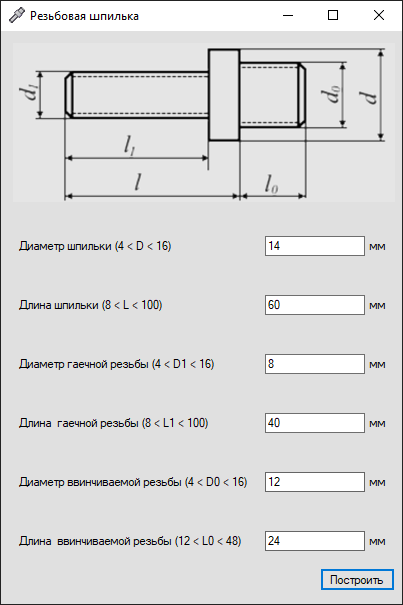


Рисунок 4.3 – Макет пользовательского интерфейса со значениями по умолчанию

После нажатия кнопки «Построить» появляется сообщение об ошибке, если введены не все данные, или какие-либо данные введены неверно. Если пользователь ввел неверное значение в поле ввода, появляется соответствующие сообщение. Соответствующие сообщения об ошибках представлены на рисунках 4.4-4.5.

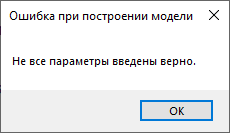


Рисунок 4.4 – Сообщение об ошибке при нажатии кнопки «Построить»

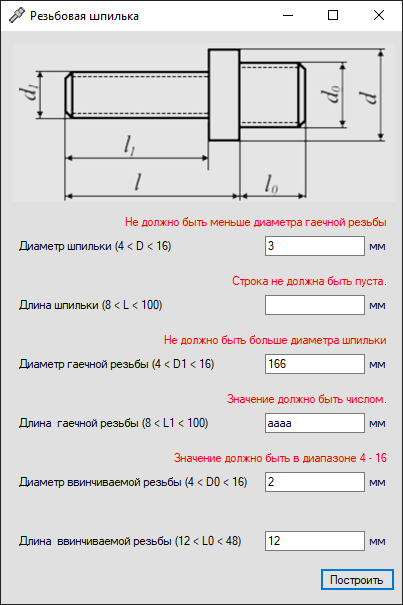
****

Рисунок 4.5 – Сообщение об ошибке при вводе неверных данных

Макет пользовательского интерфейса после добавления дополнительной функциональности представлен на рисунке 4.6. Макет пользовательского интерфейса дополнился тремя полями ввода параметров, и кнопкой, нажатие на которую восстанавливает значения параметров по умолчанию.

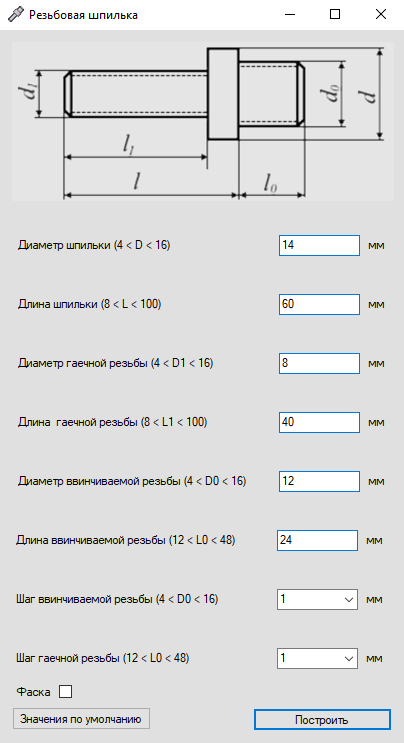


Рисунок 4.6 – Макет пользовательского интерфейса после добавления дополнительной функциональности.

# 5 Тестирование программы

Тестирование — это проверка соответствия объекта желаемым и требуемым критериям [7]. Несоответствие критериям называется ошибкой.

В проекте будет применятся три вида тестирования: функциональное тестирование, модульное тестирование, нагрузочное тестирование.

## 5.1 Функциональное тестирование

При функциональном тестировании плагина проверялось соответствие полученной трёхмерной модели с введенными параметрами.

Проведено тестирование с максимальными, минимальными параметрами и параметрами по умолчанию.

На рисунках 5.1-5.2 представлены пользовательский интерфейс с введёнными минимальными параметрами и построенная модель соответственно.

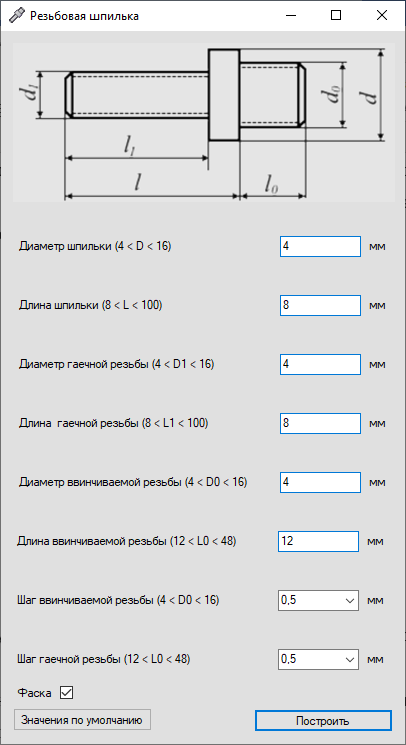


Рисунок 5.1 – Пользовательский интерфейс с минимальными параметрами.

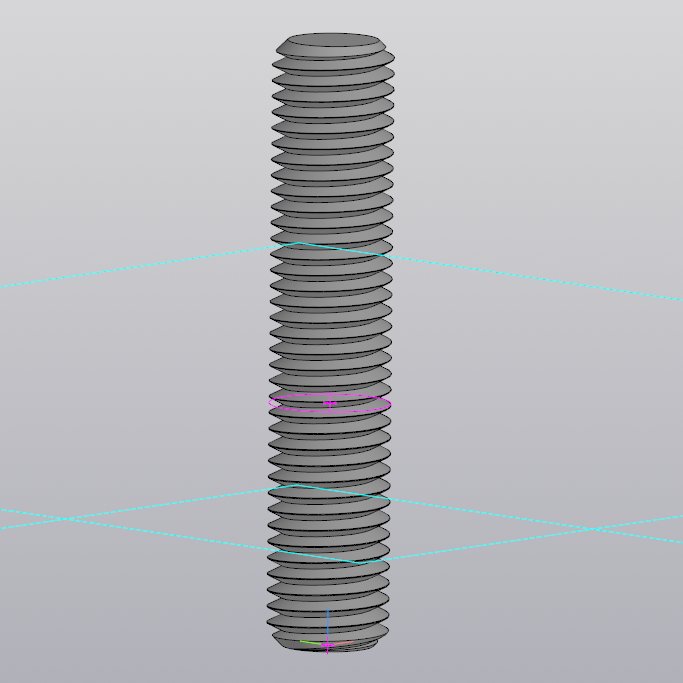


Рисунок 5.2 – Модель, построенная с минимальными параметрами.

На рисунках 5.3-5.4 представлены пользовательский интерфейс с введёнными максимальными параметрами и построенная модель соответственно.

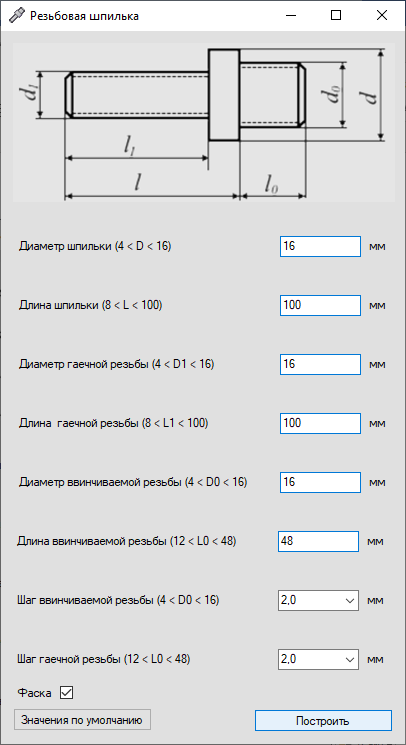


Рисунок 5.3 – Пользовательский интерфейс с максимальными параметрами.

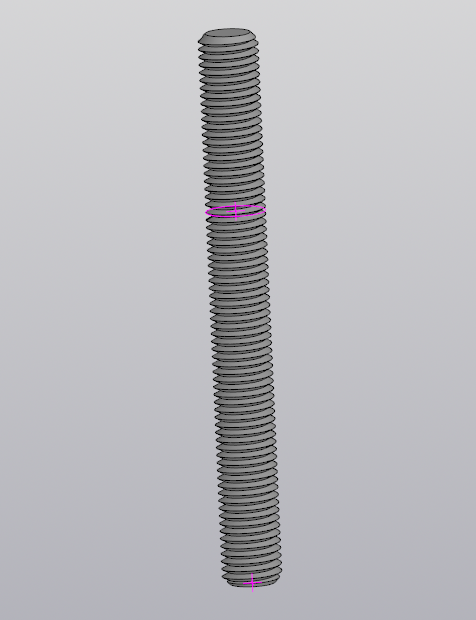


Рисунок 5.4 – Модель, построенная с максимальными параметрами.

На рисунках 5.5-5.6 представлены пользовательский интерфейс с введёнными параметрами по умолчанию и построенная модель соответственно.

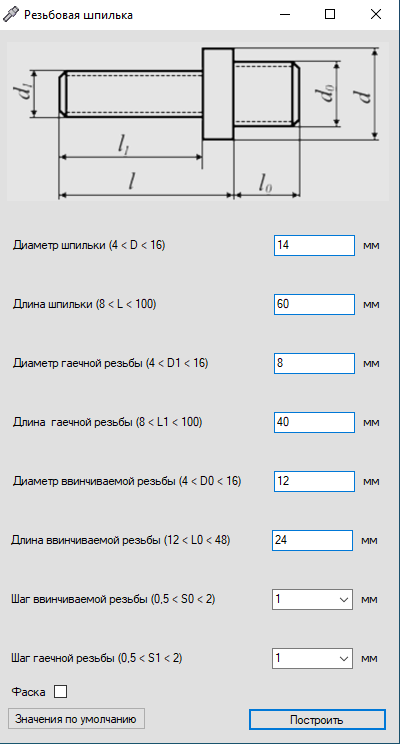


Рисунок 5.5 – Пользовательский интерфейс с параметрами по умолчанию.

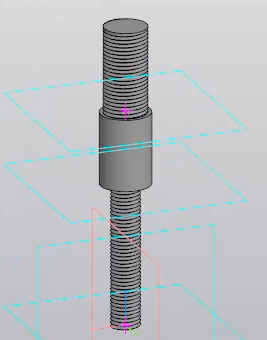


Рисунок 5.6 – Модель, построенная с параметрами по умолчанию.

## 5.2 Модульное тестирование

Юнит-тестирование (модульное тестирование) — тестирование отдельного элемента изолированно от остальной системы [7]. На основе тестовых сценариев проводилось тестирование работы методов и свойств классов при помощи обозревателя тестов Visual Studio. Проверялись открытые поля и методы, для этого были созданы тестовые классы:

В качестве проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версия 3.13.3 выполнено модульное тестирование, проверялись открытые поля, свойства и методы.

На рисунках 5.7-5.8 представлены результаты модульного тестирования и анализ покрытия тестирования соответственно. Описания тестов и их параметров представлены в приложении А.

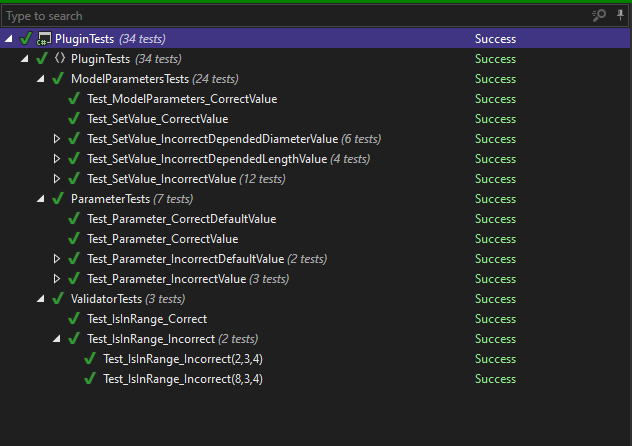


Рисунок 5.7 – Результаты модульного тестирования.

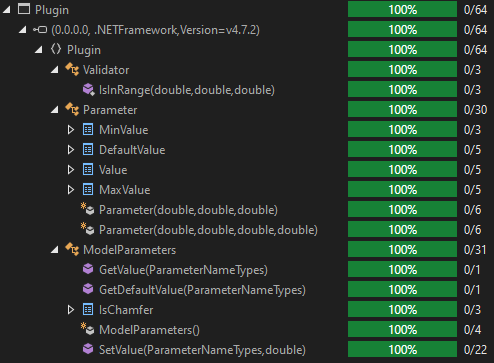


Рисунок 5.8 – Анализ покрытия модульного тестирования.

## 5.3 Нагрузочное тестирование

Нагрузочное тестирование — тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству) [7].

1) Тестирование проводилось на Windows 10 x64.

2) Процессор Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz 3.20 GHz

3) 8,00 ГБ ОЗУ

Для нагрузочного тестирования был задан цикл повторяющегося построения детали. Для измерения времени был использован класс Stopwatch. Тестирование заключалось в построении шпильки с параметрами по умолчанию. На рисунке 5.9 представлен график зависимости использования оперативной памяти от количества построенных деталей.

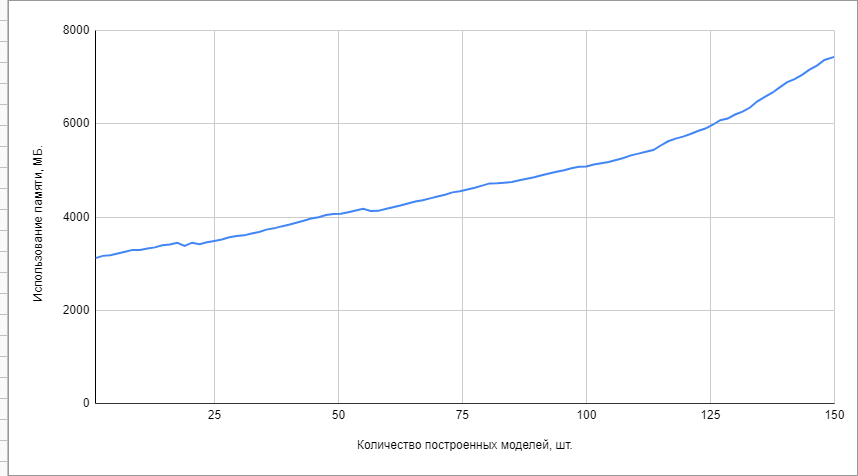


Рисунок 5.9 – График зависимости используемой оперативной памяти от количества построенных моделей.

На рисунке 5.10 представлен график зависимости времени построения одной детали от количества построенных деталей.

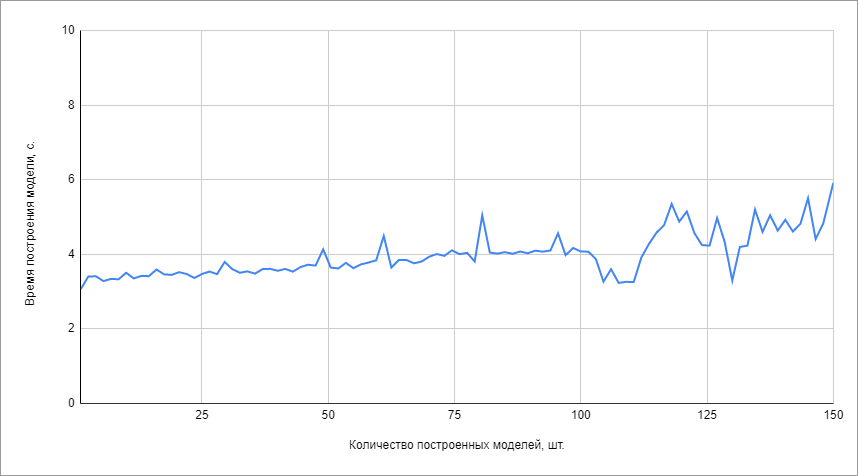


Рисунок 5.10 – График зависимости времени построения одной детали от количества построенных деталей.

При тестировании было построено 152 модели за 6 минут 41 секунду.

Из графика зависимости времени построения видно, что модели были построены примерно за один промежуток времени. Среднее время построения одной модели составило 3,97 секунды.

Из графика зависимости используемой оперативной памяти видно, что потребление оперативной памяти системой Компас 3D находится в линейной зависимости от количества построенных деталей.

# Заключение

В ходе выполнения лабораторных работ были изучены основные этапы проектирования программного продукта, изучена предметная область предмета проектирования, также было изучено API системы автоматизированного проектирования КОМПАС-3D. Было составлено техническое задание, разработан проект системы, составлены UML диаграммы классов, разработан макет пользовательского интерфейса.

В результате работы был разработан и реализован плагин для САПР КОМПАС-3D, выполняющий построение 3D-модели резьбовой шпильки по заданным параметрам. Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирование на платформе Windows 10.

# Список используемых источников

1. Общие сведения о САПР [Электронный ресурс]. – Режим доступа: http://www.hi-edu.ru/e-books/xbook116/01/part-002.htm, свободный (дата обращения: 17.12.2021).
2. САПР – Википедия. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Система\_автоматизированного\_проектирования (дата обращения 17.12.2021).
3. КОМПАС-3D: О программе. [Электронный ресурс]. – Режим доступа: http://kompas.ru/kompas-3d/about/ (дата обращения 17.12.2021)
4. Custom Screw Driver: Help Document. [Электронный ресурс]. – Режим доступа: <https://apps.autodesk.com/FUSION/> (дата обращения 17.12.2021)
5. Расширение возможностей системы КОМПАС [электронный ресурс]. – режим доступа: [http://www.k2x2.info/kompas\_3d\_v10\_na\_100/p9.php](http://www.k2x2.info/kompas_3d_v10_na_100/p9.php%20%20)  (дата обращения 17.12.2021).
6. UML. [Электронный ресурс]. – Режим доступа: http://www.uml.org/ (дата обращения 17.12.2021)
7. Виды тестирования [электронный ресурс]. – режим доступа: <https://qa-academy.by/qaacademy/news/klassifikaciya-vidov-testirovaniya/> (дата обращения: 17.12.2021).

# Приложение А

Таблица А.1 – Описание тестов ModelParametersTests.

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| Test\_  ModelParameters\_  CorrectValue() |  | Позитивный тест конструктора со значениями по умолчанию |
| Test\_SetValue\_  CorrectValue() |  | Позитивный тест сеттеров и геттеров |
| Test\_SetValue\_  IncorrectDepended DiameterValue  (ParameterNameTypes  , double,  double) | MainDiameter,  16, 15 | Негативный тест сеттера для зависимости длина-диаметр |
| NutDiameter,  14, 12 |
| BoltDiameter,  16, 15 |
| MainLength,  15,16 |
| NutLength,  15, 16 |
| BoltLength,  15, 16 |
| Test\_SetValue\_  IncorrectDepended LengthValue  (ParameterNameTypes,  double, double) | MainLength,  40, 50 | Негативный тест сеттера для зависимости длина-длина/диметр-диаметр |
| NutLength,  50, 40 |
| MainDiameter,  12, 13 |
| NutDiameter,  13, 12 |

Продолжение Таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| Test\_SetValue\_IncorrectValue(  ParameterNameTypes,  double) | MainDiameter, 0 | Негативный тест сеттера |
| NutDiameter, 0 |
| BoltDiameter, 0 |
| MainLength, 0 |
| NutLength, 0 |
| BoltLength, 0 |
| MainDiameter, 120 |
| NutDiameter, 60 |
| BoltDiameter, 60 |
| MainLength, 120 |
| NutLength, 120 |
| BoltLength, 120 |

Таблица А.2 – Описание тестов ParameterTests.

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| Test\_Parameter\_CorrectValue() |  | Позитивный тест конструктора параметра |
| Test\_Parameter\_  CorrectDefaultValue() |  | Позитивный тест конструктора параметра со значением по умолчанию |
| Test\_Parameter\_  IncorrectValue(  double, double, double) | 2, 3, 4 | Негативный тест конструктора параметра |
| 7, 6, 4 |
| Test\_Parameter\_  IncorrectDefaultValue(  double, double, double) | 2, 3, 4 | Негативный тест конструктора параметра со значениями по умолчанию |
| 8, 3, 4 |