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)

Разработка плагина «Винт» для САПР «КОМПАС-3D» v.21

Пояснительная записка по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:

Студент гр. 589-2

\_\_\_\_\_\_\_ Кабаев В.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Руководитель:

к.т.н., доцент

каф. КСУП

\_\_\_\_\_\_\_Калентьев А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**РЕФЕРАТ**

Лабораторная работа 32 с., 19 рис., 10 источников.

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

Целью данной работы является разработка плагина для автоматизации проектирования модели винта.

Результатом работы является плагин, осуществляющий построение винта по заданным пользователям параметрам.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word.

Содержание

[1 Введение 4](#_Toc92149091)

[2 Описание API 5](#_Toc92149092)

[3 Обзор аналогов 11](#_Toc92149093)

[4 Описание предмета проектирования 13](#_Toc92149094)

[4.1 Выбор инструментов и средств реализации 15](#_Toc92149095)

[4.2 Назначение плагина 15](#_Toc92149096)

[5 Описание реализации 16](#_Toc92149097)

[5.1 Диаграмма классов 16](#_Toc92149098)

[6 Описание программы для пользователя 19](#_Toc92149099)

[7 Тестирование плагина 22](#_Toc92149100)

[7.1 Функциональное тестирование 22](#_Toc92149101)

[7.2 Модульное тестирование 24](#_Toc92149102)

[7.3 Нагрузочное тестирование 26](#_Toc92149103)

[Заключение 30](#_Toc92149104)

[Список использованных источников 31](#_Toc92149105)

# Введение

Автоматизация моделирования имеет огромное значение для развития науки, техники и производства в современном обществе. В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Использование автоматизации в проектировании позволяет создавать все более сложные технические объекты и гибко реагировать на появление новых решений и технологий в той или иной области техники. Она позволяет значительно повысить точность расчетов, выбрать наилучшие варианты для реализации на основе строгого математического анализа всех или большинства вариантов проекта с оценкой технических, технологических и экономических характеристик производства и эксплуатации проектируемого объекта, значительно повысить качество конструкторской документации, существенно сократить сроки проектирования и передачи конструкторской документации в производство, эффективнее использовать технологическое оборудование с программным управлением [1].

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели «Винт» для системы автоматизированного проектирования КОМПАС-3D с помощью интегрированной среды разработки Visual Studio 2022 Сommunity. [2]

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

# Описание API

Аббревиатура ***API*** расшифровывается как «Application Programming Interface» (интерфейс программирования приложений, программный интерфейс приложения). [4]

Сегодня встречаются задачи, решение которых не реализованы в CAD-системах. Чаще всего это очень узкоспециализированные задачи, которые встречаются на каком-то конкретном предприятии или подотрасли. Для решения подобных задач и их автоматизации используется API.

***API КОМПАС-3D*** — это ориентированные на прикладного программиста инструментальные средства разработки приложений (библиотек конструктивов, прикладных САПР) на базе системы КОМПАС. API КОМПАС-3D включает в свой состав API 5 и API 7.

Главным интерфейсом API системы КОМПАС является ***KompasObject.***[5] Методы этого интерфейса, реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы (интерфейсы динамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

Ниже в таблице 2.1 представлены основные свойства и методы интерфейса KompasObject.

Таблица 2.1 – Методы и свойства интерфейса KompasObject

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

Продолжение таблицы 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

В таблице 2.2 представлены методы интерфейса ksEntity, необходимые для разработки плагина.

Таблица 2.2 – Методы интерфейса ksEntity

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| Create() | bool | Создать объект в модели |
| GetDefinition() | IUnkown | Получить указатель на интерфейс параметров объектов и элементов |
| Update() | bool | Изменить свойства объекта (используя ранее установленные свойства) |
| NewEntity (short objType) | указатель на интерфейс ksEntity или IEntity. | Реальный объект создается в модели после вызова метода [Create](mk:@MSITStore:D:\KOMPAS\SDK\SDK.chm::/ksEntity_Create.htm). |

В таблице 2.3 представлены свойства и методы интерфейса ksDocument2D, необходимые для разработки плагина.

Таблица 2.3 – 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 2.4 представлены свойства и методы интерфейса ksDocument3D, необходимые для разработки плагина.

Таблица 2.4 – 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) | |
|  | |  |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. | ksPart | | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 2.5 представлены методы интерфейса ksPart, необходимые для разработки плагина.

Таблица 2.5 – 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 2.6 представлены типы объектов документа-модели, необходимые для разработки плагина.

Таблица 2.6 – 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\_bossExtrusion | Приклеивание выдавливанием | ksBossExtrusionDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_fillet | Операция "скругление" | ksFilletDefinition |
| o3d\_edge | Ребро | ksEdgeDefinition |
| o3d\_planeOffset | Смещённая плоскость | ksPlaneOffsetDefinition |

# Обзор аналогов

Autodesk Inventor – система [трёхмерного](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D1%91%D1%85%D0%BC%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) твердотельного и поверхностного [параметрического проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([САПР](https://ru.wikipedia.org/wiki/%D0%A1%D0%90%D0%9F%D0%A0)) компании [Autodesk](https://ru.wikipedia.org/wiki/Autodesk), предназначенная для создания цифровых прототипов промышленных изделий. Инструменты Inventor обеспечивают полный цикл проектирования и создания конструкторской документации.

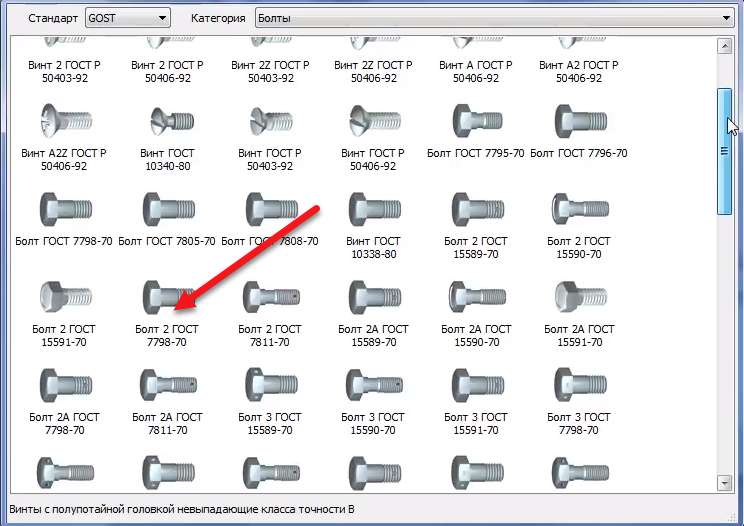


Рисунок 3.1 – Построение винта в Inventor

Inventor имеет готовую библиотеку винтов с соответствующий ГОСТом, которые можно использовать в своих сборках. Помимо данной возможности он имеет возможность создание отверстий с необходимой ГОСТом.

# Описание предмета проектирования

[Винт строительный](https://pkmetiz.ru/vinty/) — вид крепежа, используемый для взаимной фиксации двух деталей, одна из которых снабжена резьбой. Внешне винт выглядит как стержень, на одном конце которого имеется резьба с определенными параметрами (шаг, глубина, профиль), на другом расположена шляпка со шлицем для передачи крутящего момента (завинчивания с помощью отвертки).

Изображение моделируемого объекта на рисунке 4.1



Рисунок 4.1 – Чертеж Винта

Измеряемые параметры для плагина:

* L – Общая длина винта (22-26 мм);
* M –Длина шлица (10 – 13 мм), должна быть < Dk + 2 мм;
* R – Радиус скругления (1 – 2 мм);
* Dk – Диаметр головки (12 – 15 мм), должен быть больше длины шлица M;
* D – Диаметр основания стержня (5-6 мм);
* T – Длина отступа (2 – 3 мм).

На рисунке 4.2 показан общий вид винта:

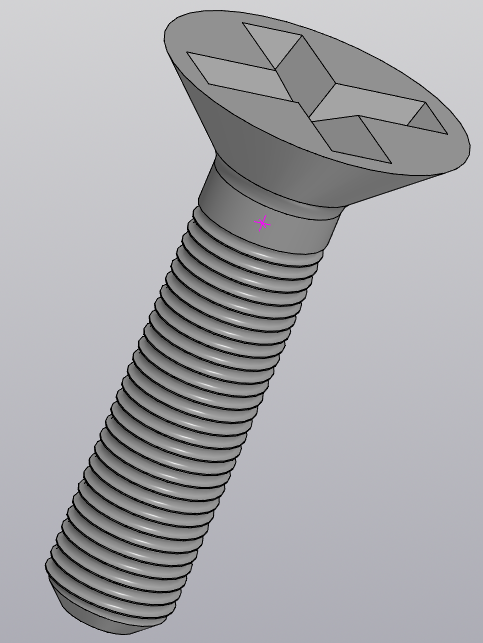


Рисунок 4.2 – Общий вид винта

## Выбор инструментов и средств реализации

На основе требований к техническому заданию программа выполнена на языке программирования C# в среде Microsoft Visual Studio 2022 с использованием .NET Framework 4.7.2 [2], библиотека «Kompas6API5» [5] для основных операций в САПР КОМПАС-3D.

Инструментом тестирования и создания модульных тестов был выбран тестовый фреймворк NUnit [6] версии 3.13.2.

Технология разработки графического интерфейса: Windows Forms [7].

## Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием винта, с разной длиной и размером сечения. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# Описание реализации

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценарии действия) использован стандарт UML.

UML язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML – моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML возможна генерация кода и наоборот. [8]

При использовании UML были простроена диаграмма классов.

## Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

На рисунке 5.1 представлена изначальная диаграмма классов.

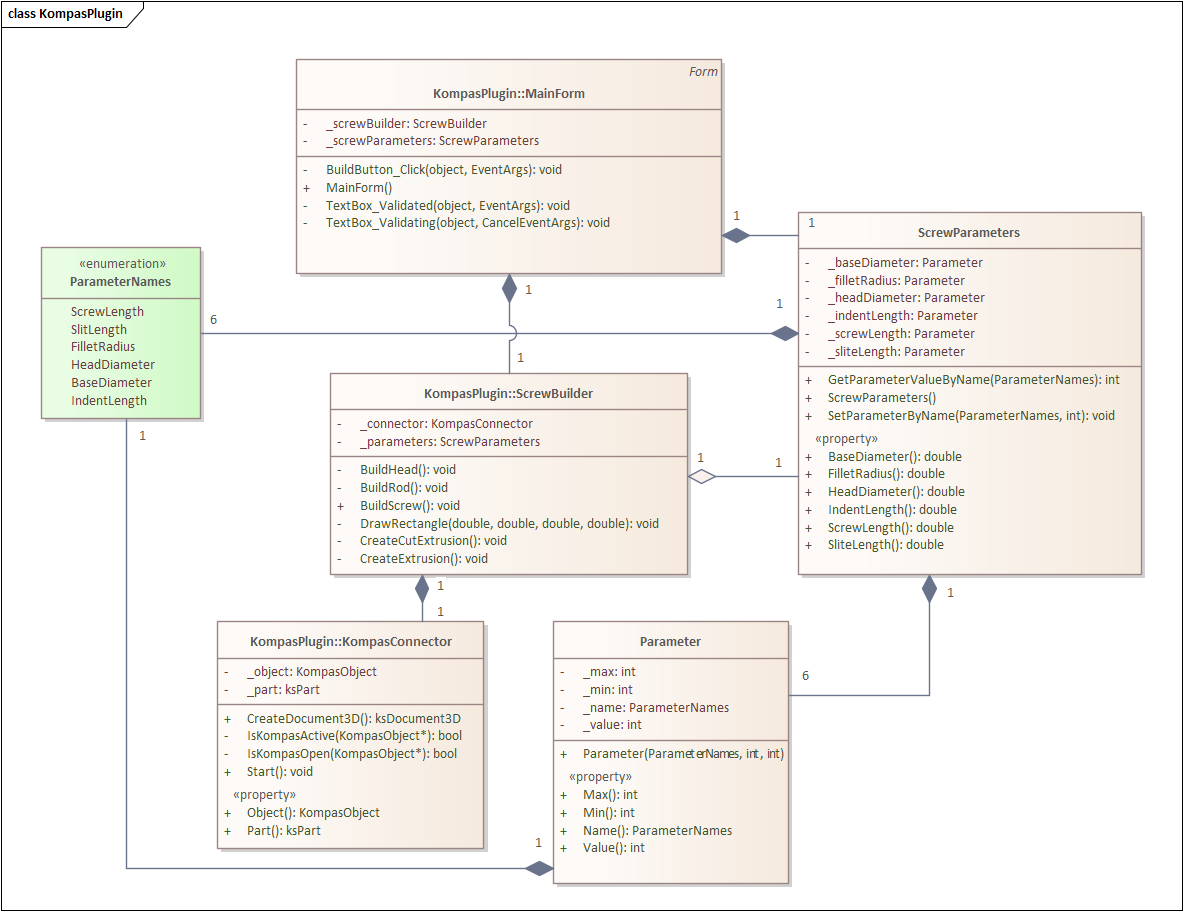


Рисунок 5.1 – Изначальная диаграмма классов

* MainForm является главным элементом управления для обработки действий в графическом интерфейсе;
* WaveguideBuilder – выполняет построение детали;
* WaveguideParameters – содержит в себе параметры волновода, которые проверяются на правильность с помощью класса Validator;
* Validator – имеет метод для сравнивания параметра с максимальным и минимальным доступным значением;
* KompasConnector – класс связи с КОМПАС – 3D.

В итоговом проекте созданы следующие классы и методы, которые отображены на итоговой диаграмме классов (рисунок 5.2).

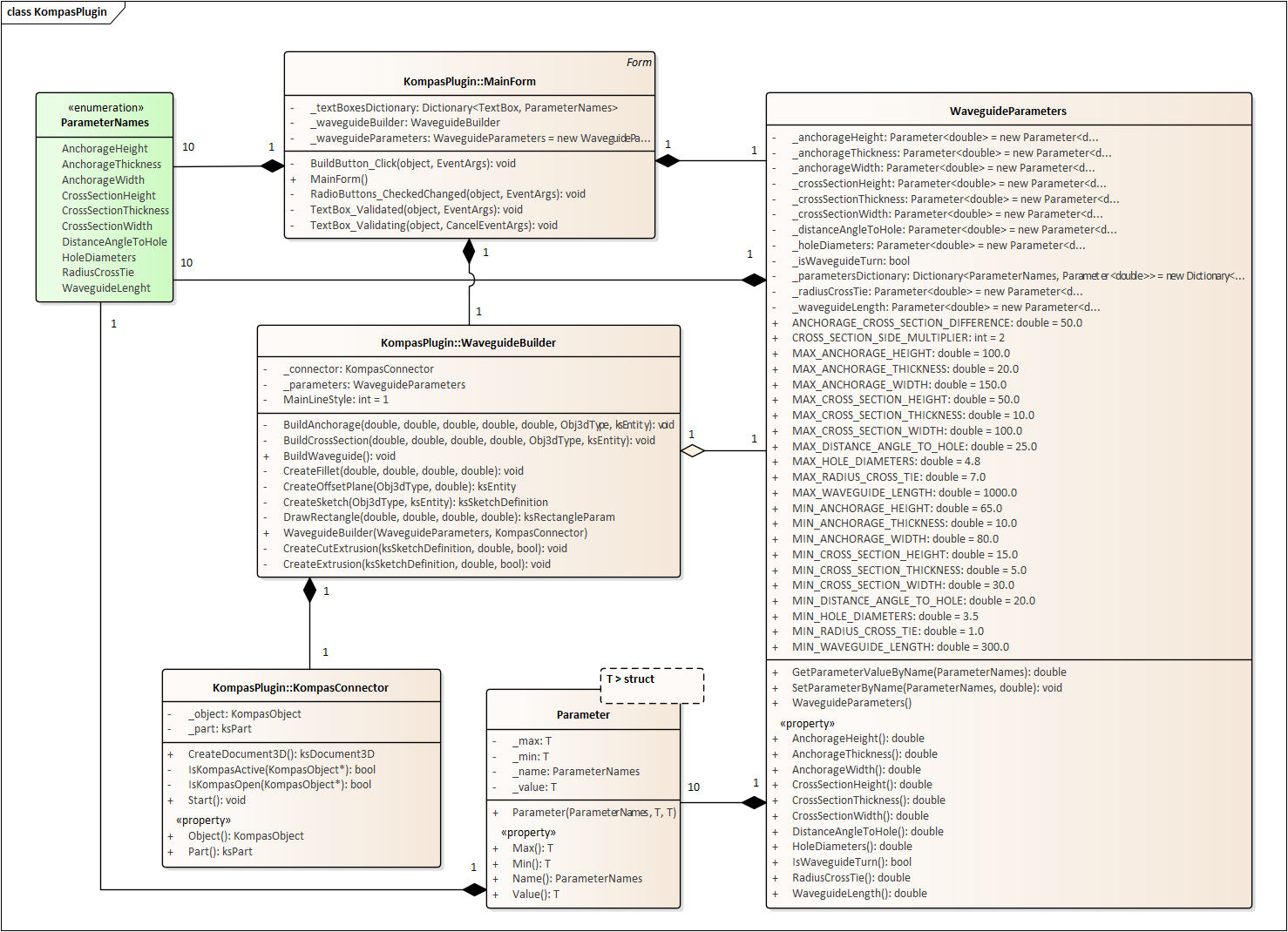


Рисунок 5.2 – Итоговая диаграмма классов

В процессе разработки была переосмыслена изначальная диаграмма классов, в результате чего:

* Решено было не реализовывать класс Validator, так как проверка параметров в заданном диапазоне осуществляется в шаблонном классе Parameter, а проверка зависимых параметров дополнительно проводится в свойстве каждого из них;
* Был разработан шаблонный класс для удобства проверки и задания граничных значений каждого параметра, валидации значения заносимого в параметр, а также формирование сообщения об ошибках ввода;
* Так же было решено отказаться от задания имени в строковом формате, вместо этого было создано перечисление ParameterNames, что предотвратит ошибок при поиске и занесения значений в параметры;
* Константы минимальных и максимальных значений параметров были занесены в поля класса WaveguideParameters для удобства и доступности их значений в пользовательском коде, так же в нескольких классах был использован Ассоциативный массив для уменьшения дублирования кода и упрощения передачи значений параметров.

# Описание программы для пользователя

Пользовательский интерфейс состоит из окна, в котором вводятся данные для построения винта. Если все данные были введены корректно, то при нажатии кнопки «Построить» происходит построение модели. При наличии ошибок в вводе параметров построение не произойдет и пользователю выведется сообщение об ошибке.

Так же под списком параметров расположен переключатель выбора необходимой формы шлица под отвертку.

На рисунке 6.1 представлен макет интерфейса программы.

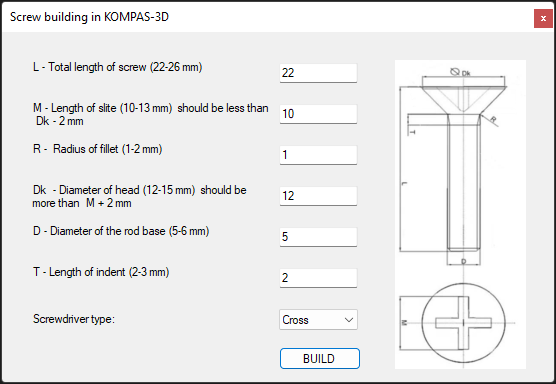


Рисунок 6.1 – Пользовательский интерфейс

Если ввести неверные параметры, то кнопка «Построить» будет заблокирована для нажатия, а поля с неправильными данными будут выделены красным цветом и при наведении на них курсором мыши будет появляться сообщение с описанием ошибки. На рисунке 6.2 представлен макет интерфейса с некорректно введенными данными.

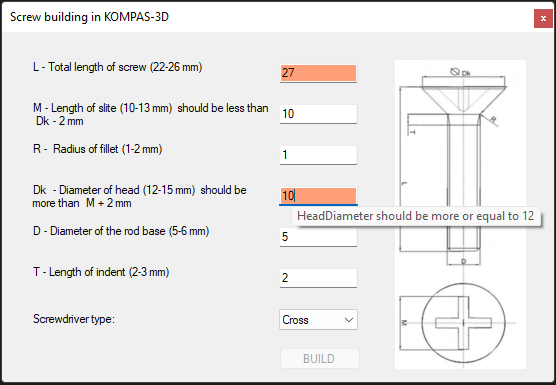


Рисунок 6.2 – Макет интерфейса с неверно введенными данными

После ввода корректных параметров кнопка «Построить» будет снова доступна и при ее нажатии будет построена модель винта.

Винт, построенный по заданным параметрам по умолчанию представлен на рисунке 6.3

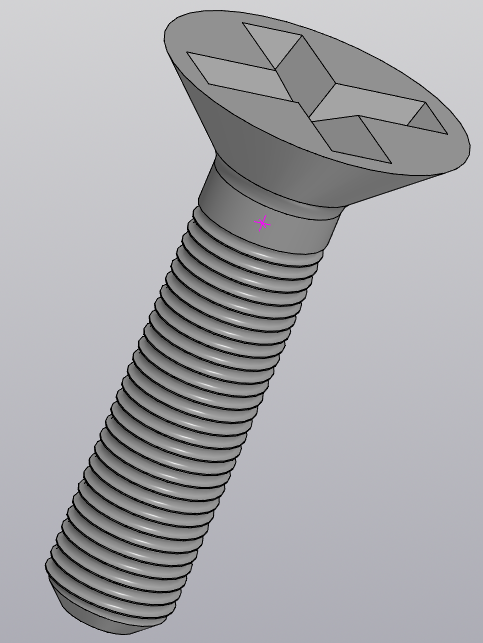


Рисунок 6.3 – Модель винта, построенная

по параметрам по умолчанию

Модель винта с шестигранным шлицом представлена на рисунке 6.4:

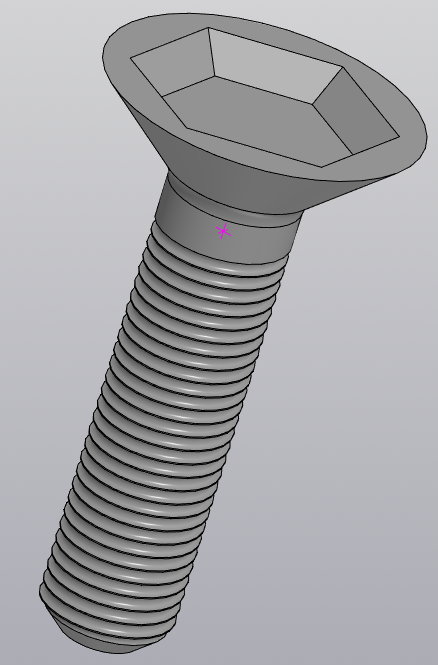


Рисунок 6.4 – Модель винта с шестигранным шлицом

# Тестирование плагина

Тестирование позволяет убедиться в работоспособности программы, выявлять ошибки при изменении какого-либо функционала.

## Функциональное тестирование

При функциональном тестировании проверялось корректность работы плагина, а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами. [9]

Проведено тестирование максимальных и минимальных параметров модели.

Минимальные значения:

1. Общая длина винта = 22 мм;
2. Длина шлица = 10 мм;
3. Радиус скругления = 1 мм;
4. Диаметр головки = 12 мм;
5. Диаметр основания стержня = 5 мм;
6. Длина отступа = 2 мм.

Максимальные значения:

1. Общая длина винта = 26 мм;
2. Длина шлица = 13 мм;
3. Радиус скругления = 2 мм;
4. Диаметр головки = 15 мм;
5. Диаметр основания стержня = 6 мм;
6. Длина отступа = 3 мм.

На рисунке 7.1 представлена модель с минимально введенными параметрами.

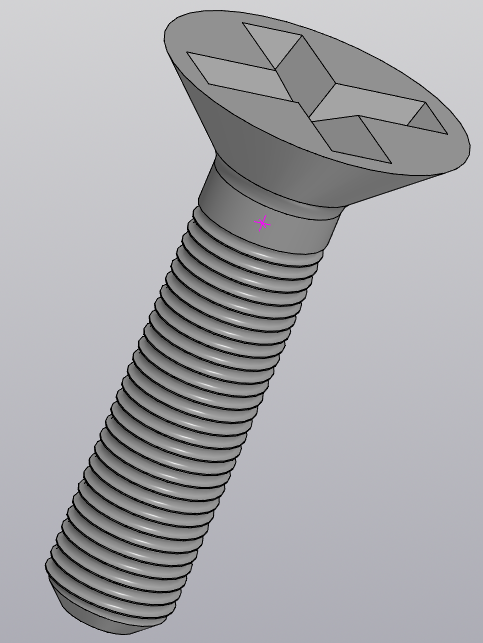


Рисунок 7.1 – Модель винта с минимальными введенными параметрами

На рисунке 7.2 представлена модель винта с максимальными введенными параметрами:

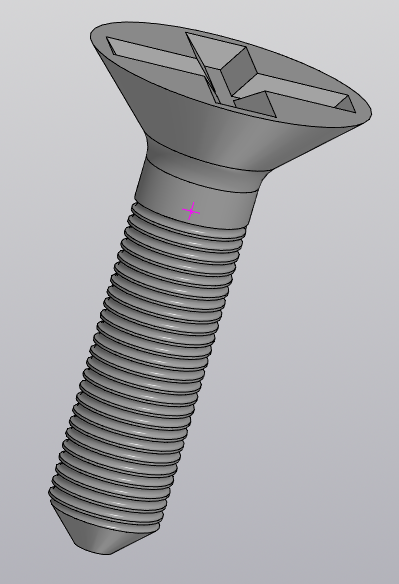


Рисунок 7.2 – Модель винта с максимальными параметрами

На рисунках 7.3–7.4 представлены модели винта с максимальными параметрами и разными шлицами:

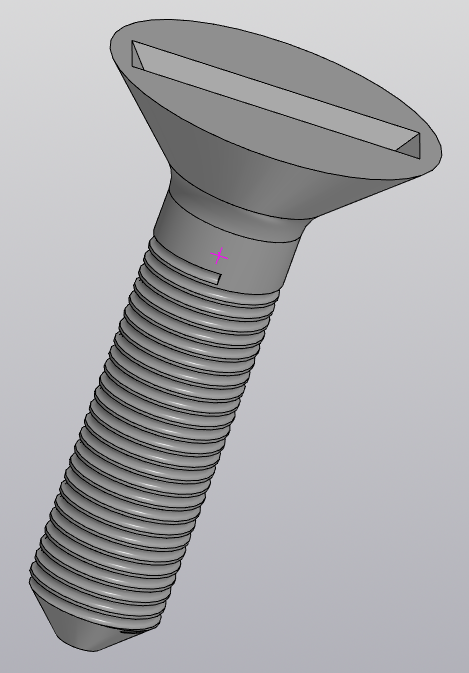


Рисунок 7.3 – Модель винта с максимальными параметрами и плоским вырезом под отвертку

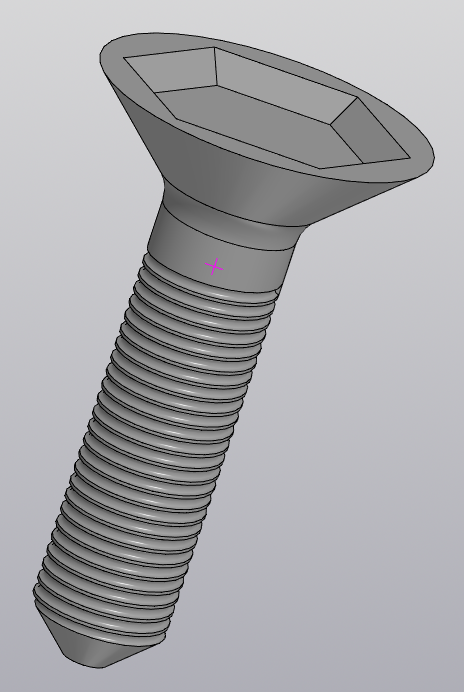


Рисунок 7.4 – Модель винта с максимальными параметрами и шестигранным вырезом под отвертку

## Модульное тестирование

В целях проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версии 3.13 проведено модульное тестирование, проверялись открытые поля, свойства и методы. Были протестированы классы модели: Parameter, WaveguideParameters. На рисунке 7.4 представлена информация о модульном тестировании программы. Покрытие модели тестами составило сто процентов, что показано на рисунке 7.5:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7.4 – Модульное тестирование плагина

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7.5 – Покрытие кода тестами

## Нагрузочное тестирование

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование. Тестирование производилось на ПК со следующей конфигурацией:

– ЦП ntel(R) Core(TM) i5-4200M CPU 2.50GHz;

– 4 ГБ ОЗУ;

– графический процессор объемом памяти 2 ГБ.

Для нагрузочного тестирования был задан бесконечный цикл построения детали. Для измерения времени был использован класс Stopwatch. Тестирование заключалось в построении винта с минимальными параметрами. На рисунке 7.6 представлен график зависимости времени от количества построенных деталей.

Рисунок 7.6 – График зависимости времени построения

от количества построенных винтов

На рисунке 7.7 представлен график зависимости загруженности памяти от количества построенных деталей

Рисунок 7.7 – График зависимости загруженности памяти

от количества построенных винтов

Тестирование длилось одиннадцать минут, было построено шестьдесят шесть моделей винта со стандартными параметрами, работа плагина завершилась сообщением об ошибки памяти в Visual Studio. Исходя из графика, представленного на рисунке 7.6 можно увидеть, что построения детали в основном занимает менее десяти секунд, но после построения пятьдесят седьмой детали, время построения некоторых деталей стало возрастать, данную тенденцию можно соотнести с самым большим изменением занимаемой памяти показанное на рисунке 7.7. Резкие уменьшения потребления памяти скорее всего обусловлены тем, что КОМПАС-3D подключает алгоритмы оптимизации при достижении максимальной занимаемой оперативной памяти компьютера.

Так же немаловажным фактором является использование файла подкачки, а точнее переход в виртуальную память. Виртуальная память — метод [управления памятью](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C%D1%8E) [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), позволяющий выполнять программы, требующие больше [оперативной памяти](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C), чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, [твердотельным](https://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) накопителем)[10].

Также было решено провести второе нагрузочное тестирование в процессе которого будут создаваться винты с шестигранным шлицом, в процессе построения, которых используется больше операций.

На рисунках 7.8 и 7.9 показан результат данного тестирования:

Рисунок 7.8 – График зависимости времени от

количества построенных винтов с шестигранным шлицом



Рисунок 7.9 – График зависимости загруженности памяти

от количества построенных винтов с шестигранным шлицом

Нагрузочное тестирование с построенных винтов с шестигранным шлицом завершилось через десять минут предварительной остановкой работы ПК с сообщением о критическом сбое работы ОС Windows.

Исходя из графиков тестирования 7.8 и 7.9, можно выделить линейное увеличение затрачиваемой памяти на построение деталей до достижения максимального занимаемого объема, после которого скорее всего началась использование файлов подкачки и работа алгоритмов оптимизации. При этом время построения каждой детали оставалось в диапазоне от двух до семи секунд, за исключением деталей с пятьдесят седьмой по шестьдесят первую.

# Заключение

При выполнения лабораторных работ были изучены основные этапы проектирования программного продукта и его реализации, предметная область объекта проектирования, SDK «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 винта по заданным параметрам.

Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирование на платформе Windows 11.

# Список использованных источников

1. Автоматизация вычислительных процедур в прикладных задачах инженерного проектирования [Электронный ресурс]. – URL: https://scienceforum.ru/2014/article/2014000201 (дата обращения: 10.12.2022).
2. Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 10.12.2022).
3. µWave Wizard. [Электронный ресурс]. — Режим доступа: https://www.mician.com/products/μwave-wizard/ (дата обращения 29.10.2022).
4. Болт – Википедия. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/> Винт\_(деталь) (дата обращения 03.09.2022)
5. КОМПАС-3D для разработчиков. [Электронный ресурс]. – URL: https://kompas.ru/solutions/developers/ (дата обращения: 10.12.2022).
6. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 10.12.2022).
7. Введение в Windows Forms [Электронный ресурс]. – URL: https://metanit.com/sharp/windowsforms/1.1.php (дата обращения: 10.12.2022).
8. UML. [Электронный ресурс]. – Режим доступа: http://www.uml.org/ (дата обращения 10.12.2022).
9. Функциональное тестирование [Электронный ресурс]. – URL: https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/ (дата обращения: 10.12.2022).
10. Виртуальная память — Википедия. [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Виртуальная память (дата обращения 17.04.2022)