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)

БИБЛИОТЕКА ЗВЁЗДНОГО ИСТРЕБИТЕЛЯ T-65 «X-WING» ДЛЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

КОМПАС-3D

Пояснительная записка по дисциплине

«Основы разработки САПР»

Студент гр. 589-1

\_\_\_\_\_\_\_\_\_ Ю. Н. Кобзарь

\_\_\_\_\_\_\_\_\_

Руководитель

преподаватель каф., к.т.н.

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ А.А. Калентьев

(оценка) \_\_\_\_\_\_\_\_\_

Томск 2022

**Реферат**

Учебная работа 47 страниц, 12 таблиц, 34 рисунка, 16 источников.

Ключевые слова: КОМПАС-3D, ЗВЁЗДНЫЙ ИСТРЕБЛИТЕЛЬ, X-WING, СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, БИБЛИОТЕКА.

Целью данной работы является разработка библиотеки звёздного истребителя T-65 «X-Wing» для системы автоматизированного проектирования Компас-3D.

В процессе работы должны были пройдены все стадии разработки программного обеспечения: выбор темы, составление технического задания, составление проекта системы, создание первой реализации и её тестирование, создание конечной библиотеки.

Отчет по учебной работе выполнен в текстовом редакторе Microsoft Word 2016.

**Оглавление**

[Введение 4](#_Toc119668672)

[1 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ 5](#_Toc119668673)

[1.1 Описание предмета проектирования 5](#_Toc119668674)

[1.2 Описание инструментов и средств реализации 7](#_Toc119668675)

[1.3 Назначение библиотеки 8](#_Toc119668676)

[2 ОБЗОР АНАЛОГОВ 9](#_Toc119668677)

[2.1 Плагин «Airplane Design Workbench» для FreeCAD 9](#_Toc119668678)

[2.2 Плагин «Rocket Workbench» для FreeCAD 10](#_Toc119668679)

[3 ОПИСАНИЕ РЕАЛИЗАЦИИ 11](#_Toc119668680)

[4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ 30](#_Toc119668681)

[5 ТЕСТИРОВАНИЕ ПРОГРАММЫ 33](#_Toc119668682)

[5.1 Функциональное тестирование 33](#_Toc119668683)

[5.2 Модульное тестирование 38](#_Toc119668684)

[5.3 Нагрузочное тестирование 41](#_Toc119668685)

[Заключение 45](#_Toc119668686)

[Список использованных источников 46](#_Toc119668687)

# Введение

Автоматизация проектирования 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 проектированием 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.

Практическая реализация методов и идей автоматизированного проектирования происходит в рамках систем автоматизированного проектирования (САПР). В рамках современного «компьютеризированного» общества инженер любой специальности, занимающийся разработкой технических устройств, должен уметь использовать средства автоматизированного проектирования. Это позволяет повысить эффективность проектирования, улучшить его качество, снизить материальные затраты и уменьшить число разработчиков.

Таким образом, целью учебной работы является разработка библиотеки звёздного истребителя T-65 «X-Wing» для системы автоматизированного проектирования Компас-3D. «Компас» – это семейство систем автоматизированного проектирования, универсальная система автоматизированного проектирования, позволяющая в оперативном режиме выпускать чертежи изделий, схемы, спецификации, таблицы, инструкции, расчётно-пояснительные записки, технические условия, текстовые и прочие документы [1].

# 1 ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ

В рамках учебной дисциплины «Основы разработки САПР» требовалось разработать библиотеку в соответствии с техническим заданием. На основе заданных параметров библиотека, взаимодействуя с САПР «Компас-3D», должна строить трёхмерную модель звёздного истребителя T-65 «X-Wing» [2]. Также библиотека должна позволять изменять входные параметры звёздного истребителя.

Изменяемые параметры:

* длина корпуса звездолёта;
* ширина крыльев звездолёта;
* длина носовой части корпуса звездолёта;
* длина острия оружейного бластера звездолёта;
* длина турбины ускорителя звездолёта;
* длина сопла ускорителя звездолёта;
* высота установок крыши корпуса.

# 1.1 Описание предмета проектирования

Выбранным предметом для проектирования является звёздный истребитель T-65 «X-Wing» из вселенной Star Wars. Оригинал разрабатываемого объекта представлен на рисунке 1.1.



Рисунок 1.1 – Звёздный истребитель T-65 «X-Wing»

На рисунках 1.2-1.3 представлены 2D-чертежи трёхмерной модели звёздного истребителя T-65 «X-Wing» в видах снизу и спереди.

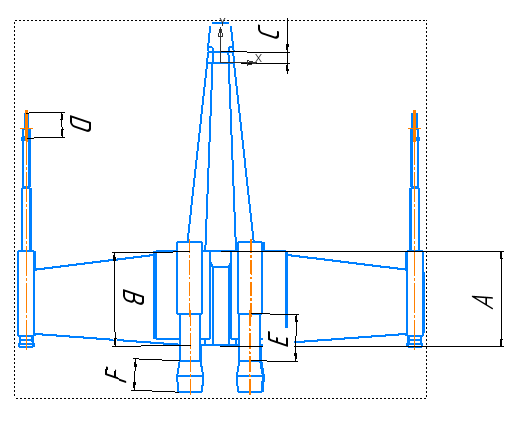


Рисунок 1.2 – Чертёж звёздного истребителя T-65 «X-Wing» в виде снизу

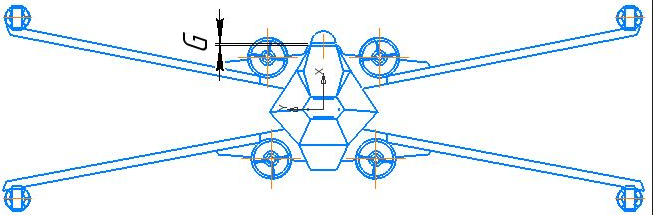


Рисунок 1.3 – Чертёж звёздного истребителя T-65 «X-Wing» в виде спереди

Параметры звёздного истребителя:

* длина корпуса звездолёта: от 300 до 400 мм; параметр представлен на чертеже как A;
* ширина крыльев звездолёта: от 300 до 400 мм; параметр представлен на чертеже как B;
* длина носовой части корпуса звездолёта: от 50 до 100 мм; параметр представлен на чертеже как C;
* длина острия оружейного бластера звездолёта: от 80 до 130 мм; параметр представлен на чертеже как D;
* длина турбины ускорителя звездолёта: от 150 до 250 мм; параметр представлен на чертеже как E;
* длина сопла ускорителя звездолёта: от 50 до 100 мм; параметр представлен на чертеже как F;
* высота установок крыши корпуса звездолёта: от 10 до 20 мм; параметр представлен на чертеже как G.

**Взаимосвязь параметров:**

* ширина крыльев B должна быть не меньше длины корпуса A более, чем на 20 мм, и не больше, чем сама длина корпуса A;
* длина турбины ускорителя E не должна быть более, чем в 4 раза больше, чем длина сопла F;
* длина острия D не должна быть более, чем в 2 раза больше, чем длина носовой части C;
* длина носовой части C должна быть не больше, чем длина острия D.

# 1.2 Описание инструментов и средств реализации

Для создания библиотеки используется среда разработки Visual Studio 2019 [3]. Библиотека написана с пользовательским интерфейсом на WindowsForms с использованием .NET Framework 4.7.2 [4]. Инструментом для тестирования является библиотеки NUnit 3.13.3 [5] и NUnit3TestAdapter 4.2.1 [5]. В качестве системы автоматизированного проектирования выбран «Компас-3D» v.20 [1].

# 1.3 Назначение библиотеки

Программа предназначена для автоматизации моделирования детали «Звёздный истребитель T-65 “X-Wing”».

Плагин позволяет пользователю ввести вышеперечисленные значения через графический интерфейс. В программе предусмотрена проверка корректности введенных данных и сообщение пользователю о неправильно заполненных полях с помощью цветового выделения и всплывающих подсказок.

При запуске моделирования с некорректными значениями программа выводит сообщение об ошибке и отменяет построение модели.

При правильно введенных значениях результатом работы программы будет созданная по ним модель звёздного истребителя T-65 «X-Wing». Размеры всех параметров, необходимых при построении, рассчитываются автоматически.

# 2 ОБЗОР АНАЛОГОВ

# 2.1 Плагин «Airplane Design Workbench» для FreeCAD

Данный экспериментальный плагин предназначен для проектирования крыльев и объектов самолетов [6][7]. На рисунке 2.1 представлен ввод параметров проектируемого крыла, на рисунке 2.2 представлено крыло, построенное с помощью данного плагина.

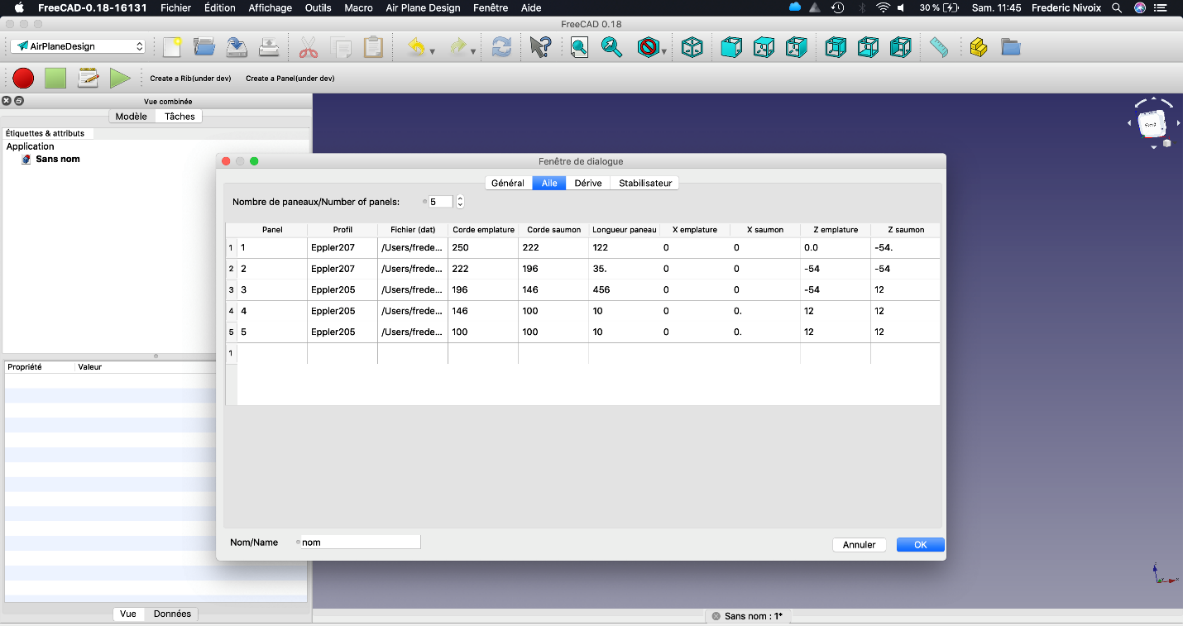


Рисунок 2.1 – Ввод параметров крыла через пользовательский интерфейс

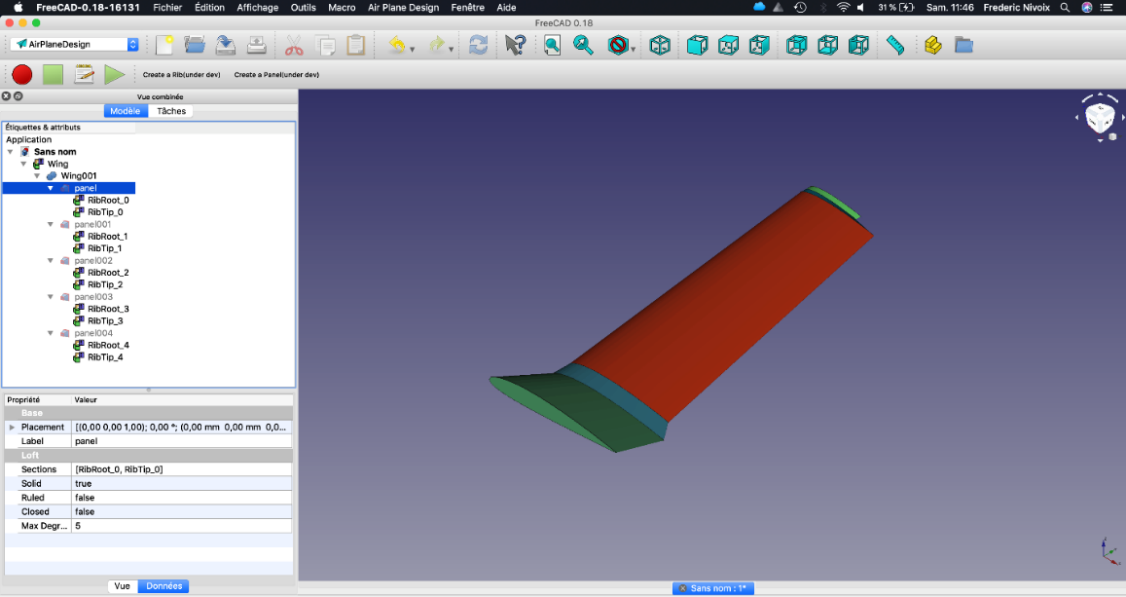


Рисунок 2.2 – Построенное крыло с помощью плагина во FreeCAD

# 2.2 Плагин «Rocket Workbench» для FreeCAD

Rocket Workbench – это набор инструментов для проектирования ракет и их компонентов. Целевая аудитория – производители моделей и любительских ракет всех типов и размеров. В настоящее время инструмент используется в основном для создания отдельных компонентов, а не полных ракет. Детали создаются путем указания параметров в диалоге задачи [6][8].

Преимущество использования Rocket Workbench заключается в том, что детали можно создавать с гораздо меньшим количеством операций [8]:

* носовой обтекатель – создание носового обтекателя;
* переход – создание перехода;
* трубка корпуса – создание полую трубку корпуса;
* центрирующее кольцо – создание центрирующего кольца;
* переборка – создание перегородки;
* плавник – создание плавника;
* руководство по запуску – создание руководства по запуску.

На рисунке 2.3 показаны примеры объектов, которые можно создавать с помощью данного плагина.

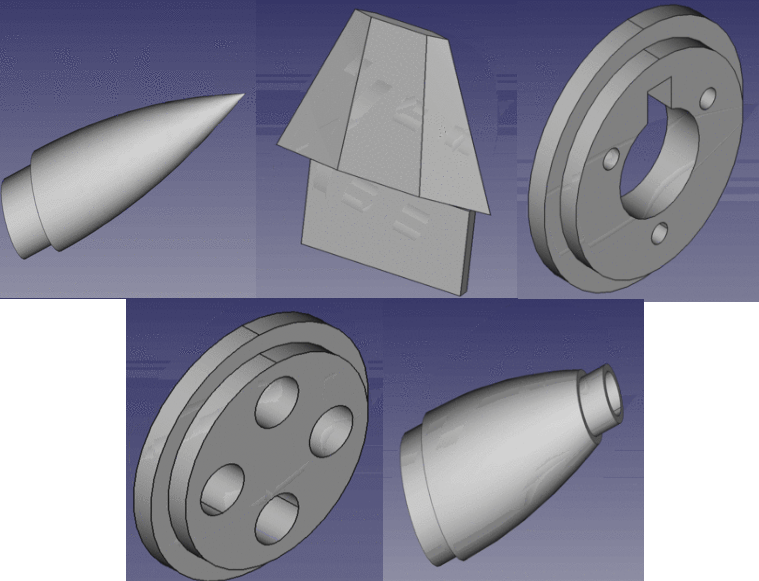


Рисунок 2.3 – Примеры созданных объектов с помощью плагина

# 3 ОПИСАНИЕ РЕАЛИЗАЦИИ

**UML (Unified Modeling Language)** – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Словарь UML включает три вида строительных блоков: диаграммы, сущности, связи [9].

**Sparx Systems Enterprise Architect** – это инструмент визуального моделирования и проектирования, основанный на OMG UML. Платформа поддерживает: проектирование и построение программных комплексов; моделирование бизнес-процессов; и моделирование отраслевых доменов. Он используется предприятиями и организациями не только для моделирования архитектуры своих систем, но и для обработки реализации этих моделей на протяжении всего жизненного цикла разработки приложений [10].

В проекте системы с помощью инструмента Enterprise Architect были созданы диаграммы классов в нотации UML. Из-за громоздкости диаграмма всего проекта была поделена на несколько частей. Класс KompasWrapper присутствует на двух частях диаграммы, рисунки 3.1-3.2; класс XWingBuilder присутствует на всех частях диаграммы; на четырёх частях, рисунки 3.2-3.5, присутствуют четыре вспомогательных класса фигур: Point2D, Point3D, Circle, Arc. Эти четыре класса и класс XWingBuilder связаны с классами констант, которые необходимы для построения детали, связи всех этих классов можно наблюдать на рисунках 3.2-3.5. Все диаграммы представлены на рисунках 3.1-3.5. На этих рисунках представлены диаграммы из проекта системы, то есть то, что было до всей реализации.

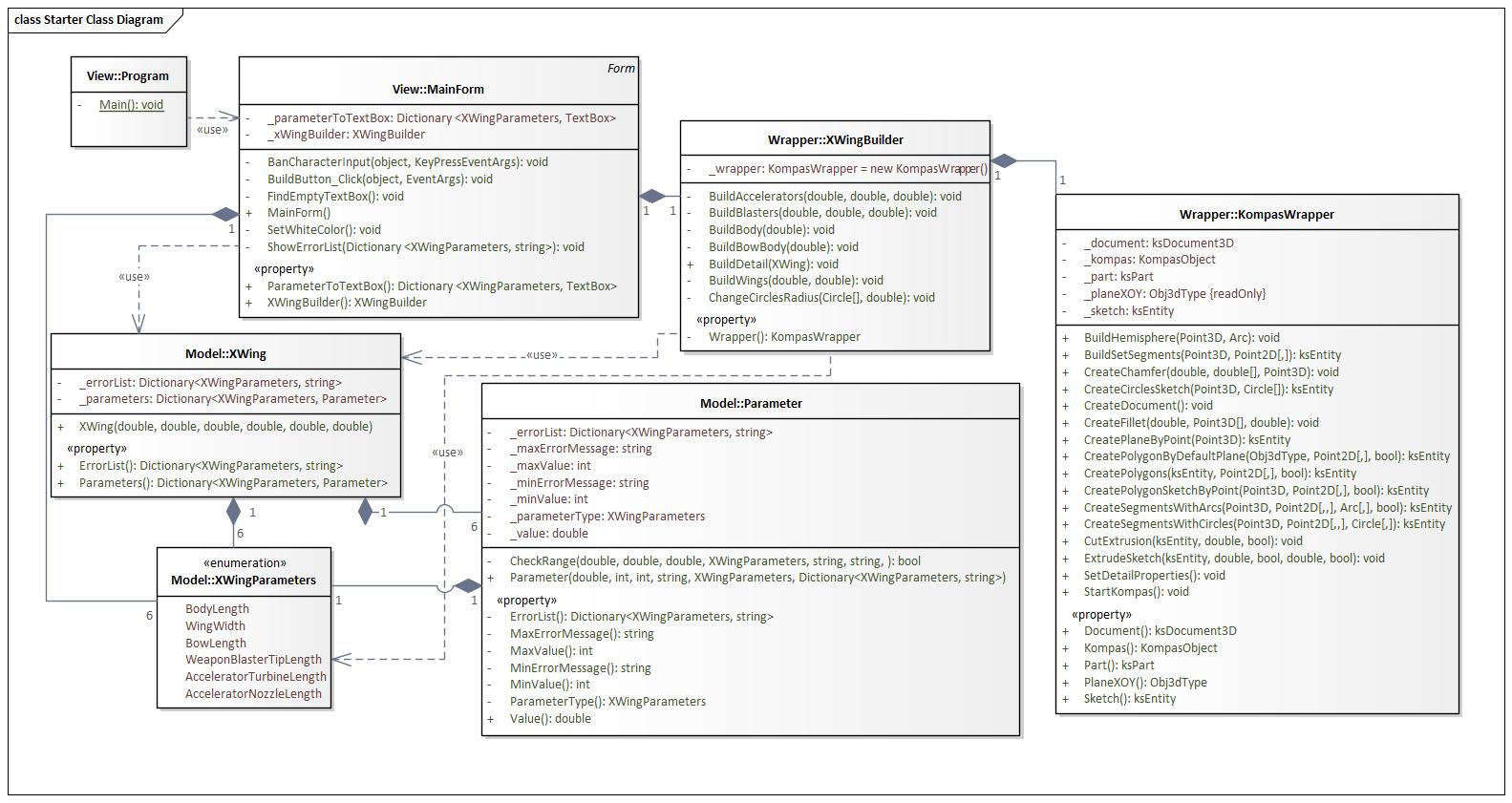


Рисунок 3.1 – Связь основной формы с классами параметров, построителя, обёртки

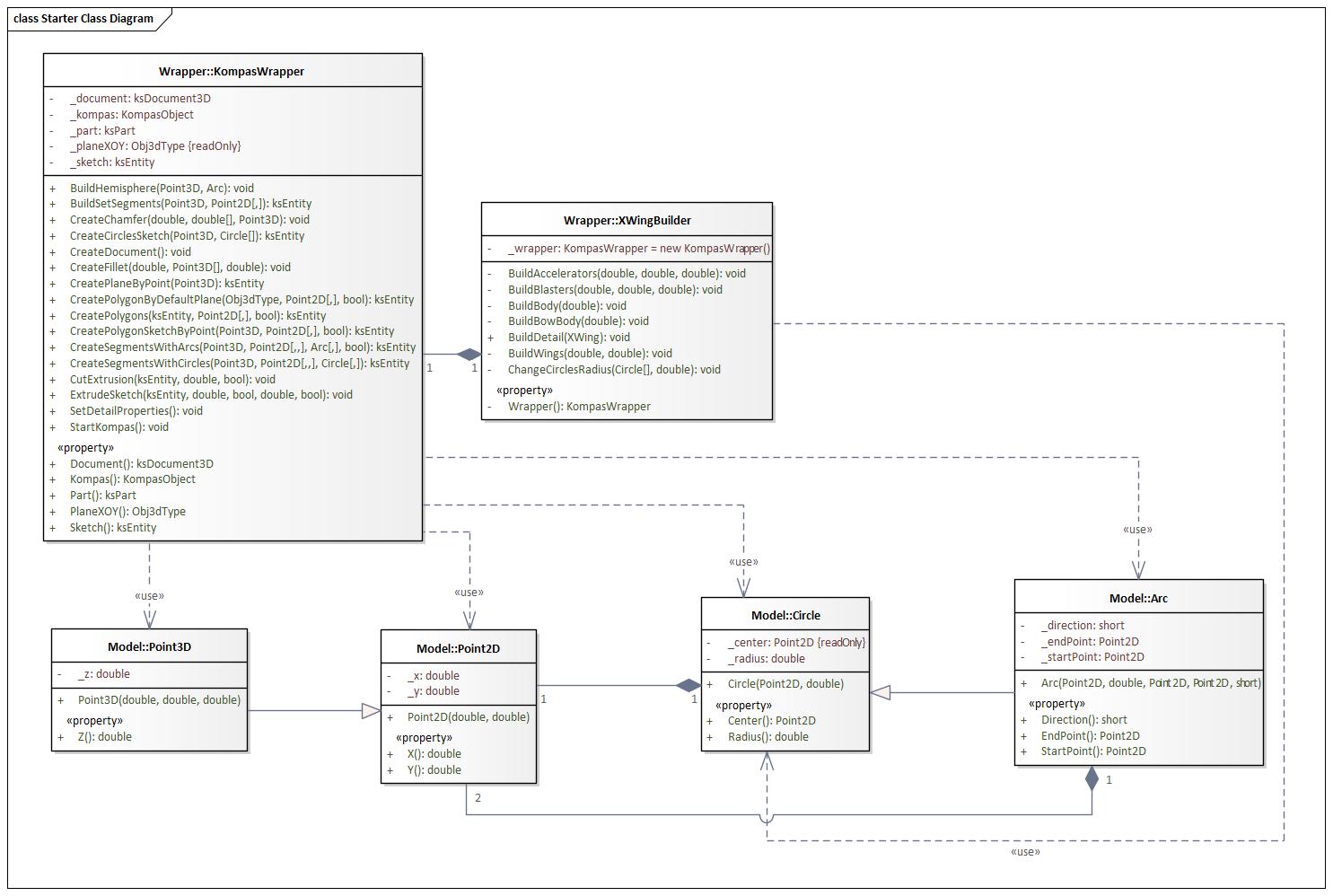
****

Рисунок 3.2 – Связь обёртки с построителем и классами геометрических фигур

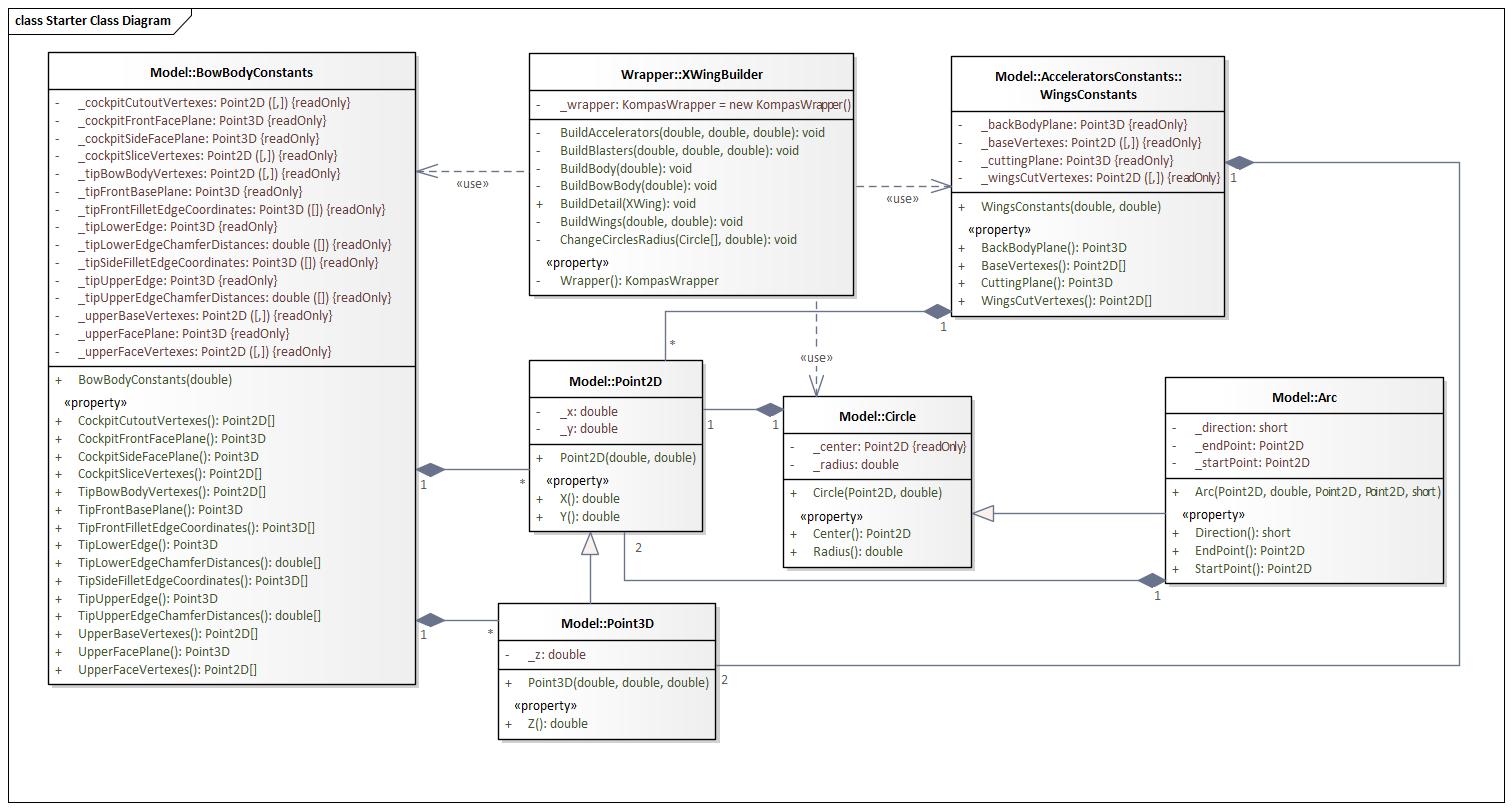
****

Рисунок 3.3 – Связь построителя с константами построения носовой части корпуса и крыльев

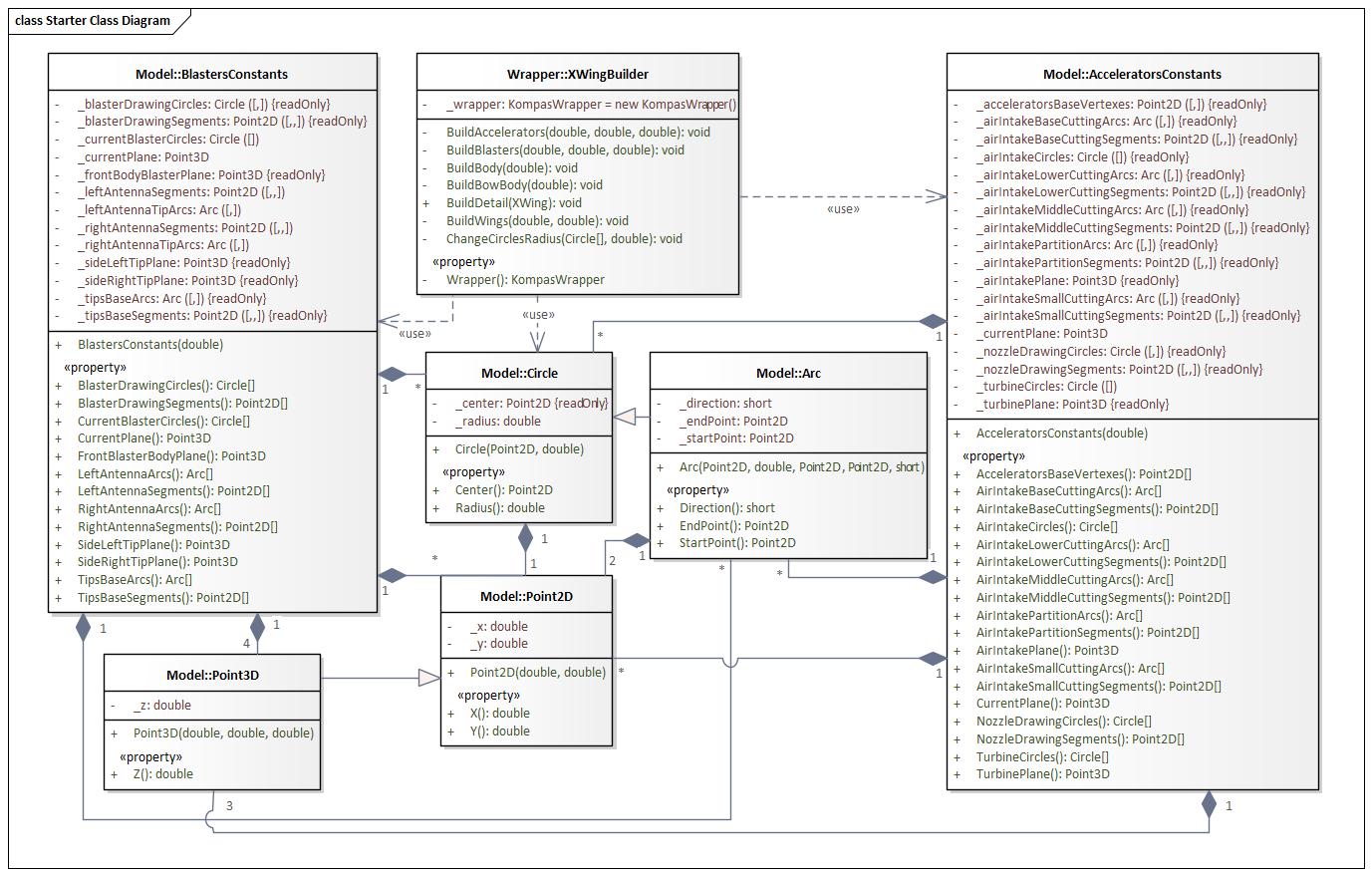


Рисунок 3.4 – Связи построителя с константами построения бластеров и ускорителей

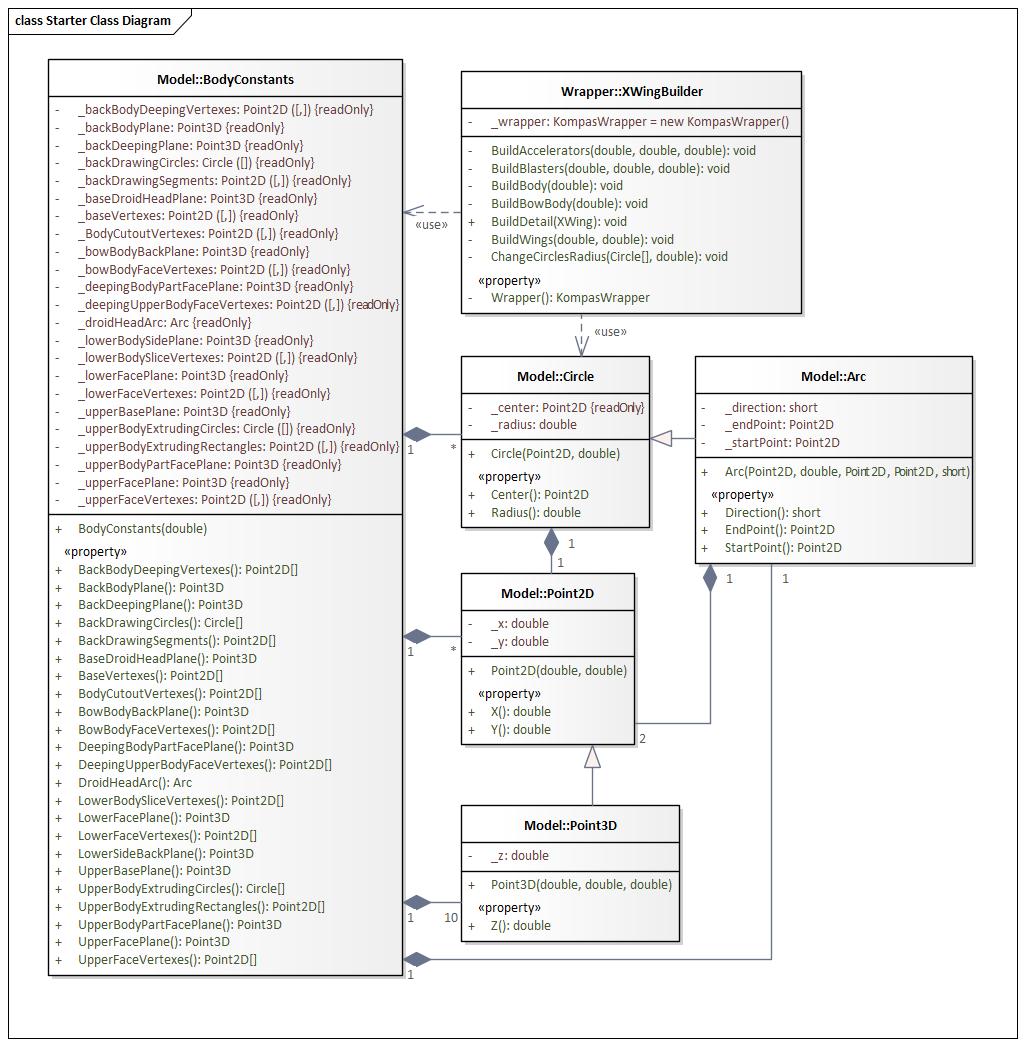
****

Рисунок 3.5 – Связь построителя с константами построения корпуса

Класс Parameter был предназначен для описания любого параметра истребителя и его валидацию, класс XWing был предназначен для объединения всех параметров воедино, также класс отвечает частично за валидацию. Класс KompasWrapper был предназначен для взаимодействия с САПР «Компас-3D», в нём содержатся методы для связи с САПР и рисования в ней, класс XWingBuilder был предназначен для построения звёздного истребителя, класс MainForm был предназначен для описания пользовательского интерфейса. Перечисление XWingParameters было необходимо для валидации параметров и корректной связи их с MainForm.

Классы Point2D, Point3D, Circle и Arc представляли собой соответственные фигуры точек двухмерного и трёхмерного пространства, круга и дуги. Классы BowBodyConstants, BodyConstants, WingsConstants, AcceleratorsConstants и BlastersConstants являлись классами заранее вычисленных констант, необходимых для построения детали, поля этих классов через свойства передавались в соответствующие методы класса XWingBuilder. Связь между классами фигур и классами констант – композиция, кратность можно увидеть на диаграммах классов на рисунках 3.3-3.5. Класс XWingBuilder использует все эти классы.

В таблицах 3.1-3.6 представлены перечисление XWingParameters и основные классы Parameter, XWing, KompasWrapper, XWingBuilder и MainForm, их поля, методы и свойства на момент составления проекта системы.

Таблица 3.1 – Класс Parameter

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_value | double | Значение параметра. |
| \_minValue | int | Минимальное допустимое значение параметра. |
| \_maxValue | int | Максимальное допустимое значение параметра. |
| \_minErrorMessage | string | Сообщение о несоблюдении границы минимума. |
| \_maxErrorMessage | string | Сообщение о несоблюдении границы максимума. |
| \_parameterType | XWingParameters | Тип параметра. |
| \_errorList | Dictionary<  XWingParameters, string> | Список ошибок введенного параметра. |
| **Свойства** | Value | double | Устанавливает и возвращает значение параметра. |
| MinValue | int | Устанавливает и возвращает минимальное допустимое значение параметра. |
| MaxValue | int | Устанавливает и возвращает максимальное допустимое значение параметра. |

Окончание таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Свойства** | MinErrorMessage | string | Устанавливает и возвращает сообщение о несоблюдении границы минимума. |
| MaxErrorMessage | string | Устанавливает и возвращает сообщение о несоблюдении границы максимума. |
| ParameterType | XWingParameters | Устанавливает и возвращает тип параметра. |
| ErrorList | Dictionary<  XWingParameters, string> | Устанавливает и возвращает список ошибок параметра. |
| **Методы** | Parameter | Конструктор | Создает объект класса параметра. |
| CheckRange | bool | Проверка принадлежности диапазону введённого параметра. |

Таблица 3.2 – Класс XWing

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_parameters | Dictionary <  XWingParameters, Parameter> | Словарь, где ключ: тип параметра звездолёта из перечисления, значение: соответствующий параметр. |
| \_errorList | Dictionary <  XWingParameters, string> | Список ошибок введенного параметра. |
| **Свойства** | Parameters | Dictionary <  XWingParameters, Parameter> | Устанавливает и возвращает словарь типов параметров - параметров. |
| ErrorList | Dictionary<XWingParameters, string> | Устанавливает и возвращает список ошибок параметра. |
| **Методы** | XWing | Конструктор | Создаёт объект класса звездолёта для построения. |

Таблица 3.3 – Перечисление XWingParameters

|  |  |
| --- | --- |
| **Название элемента** | **Описание** |
| BodyLength | Длина корпуса звездолёта. |
| WingWidth | Ширина крыльев звездолёта. |
| BowLength | Длина носовой части корпуса звездолёта. |
| WeaponBlasterTipLength | Длина острия оружейного бластера звездолёта. |
| AcceleratorTurbineLength | Длина турбины ускорителя звездолёта. |
| AcceleratorNozzleLength | Длина сопла ускорителя звездолёта. |

Таблица 3.4 – Класс KompasWrapper

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_kompas | KompasObject | Объект Компас API. |
| \_part | ksPart | Деталь. |
| \_document | ksDocument3D | Документ-модель. |
| \_sketch | ksEntity | Эскиз. |
| \_planeXOY | Obj3dType | Базовая плоскость XOY. |
| **Свойства** | KompasObject | KompasObject | Устанавливает и возвращает объект Компас API. |
| Part | ksPart | Устанавливает и возвращает деталь X-Wing. |
| Document | ksDocument3D | Устанавливает и возвращает документ-модель. |
| Sketch | ksEntity | Устанавливает и возвращает эскиз. |
| PlaneXOY | Obj3dType | Возвращает базовую плоскость XOY. |
| **Методы** | StartKompas | void | Запуск Компас-3D. |
| CreateDocument | void | Создание документа в Компас-3D. |
| SetDetailProperties | void | Установка свойств детали: цвета и имени. |
| BuildHemisphere | void | Построение полусферы. |
| BuildSetSegments | ksEntity | Построение набора отрезков. |
| ExtrudeSketch | void | Выдавливание эскиза на определенное расстояние. |
| CutExtrusion | void | Вырезание выдавливанием по эскизу. |
| CreatePlaneByPoint | ksEntity | Создание плоскости эскиза по точке. |

Окончание таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Методы** | CreatePolygonByDefaultPlane | ksEntity | Создание эскиза многоугольника по базовой плоскости. |
| CreatePolygonSketchByPoint | ksEntity | Создание эскиза по выбранной плоскости. |
| CreatePolygons | ksEntity | Создание многоугольников по координатам вершин. |
| CreateCirclesSketch | ksEntity | Создание эскиза окружностей. |
| CreateChamfer | void | Создания фаски. |
| CreateFillet | void | Создания скругления. |
| CreateSegmentsWithArcs | ksEntity | Создание эскиза из отрезков и дуг. |
| CreateSegmentsWithCircles | ksEntity | Создание эскиза из отрезков и кругов. |

Таблица 3.5 – Класс XWingBuilder

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_wrapper | KompasWrapper | Связь с Компас-3D. |
| **Свойства** | Wrapper | Wrapper | Устанавливает и возвращает связь с Компас-3D. |
| **Методы** | BuildDetail | void | Построение детали по заданным параметрам. |
| BuildBowBody | void | Построение носовой части корпуса. |
| BuildBody | void | Построение корпуса. |
| BuildWings | void | Построение крыльев. |
| BuildBlasters | void | Построение бластерного оружия. |
| BuildAccelerators | void | Построение ускорителей. |
| ChangeCirclesRadius | void | Изменение радиуса у массива кругов. |

Таблица 3.6 – Класс MainForm

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_xWingBuilder | XWingBuilder | Объект класса построения детали. |
| \_parameterToTextBox | Dictionary <  XWingParameters, string> | Словарь, где ключ: параметр звездолёта, значение: соответствующий тексбокс. |

Окончание таблицы 3.6

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Свойства** | XWingBuilder | XWingBuilder | Устанавливает и возвращает объект класса построения детали. |
| ParameterToTextBox | Dictionary <XWingParameters, TextBox> | Устанавливает и возвращает словарь параметров-текстбоксов. |
| **Методы** | MainForm | Конструктор | Конструктор основной формы. |
| BuildButton\_Click | void | Построение по введённым параметрам звездолёт. |
| BanCharacterInput | void | Запрет ввода символов и больше одной точки в число. |
| SetWhiteColor | void | Установка белого цвета тексбоксов. |
| ShowErrorList | void | Демонстрация неправильно введённых параметров. |
| FindEmptyTextBox | void | Поиск пустых тексбоксов. |

В процессе реализации библиотеки и её тестирования были изменены поля классов MainForm, XWingBuilder, KompasWrapper, Parameter – теперь все приватные поля не имеют свойств. В классах констант поля со стандартными геттерами и сеттерами имеют авто свойства. Также у авто свойств констант были изменены некоторые типы данных, чтобы исправить дублирование кода, из-за чего поменялась и кратность связей. Классам Arc, Circle, Parameter, Point2D и Point3D был добавлен интерфейс IEquatable для возможности сравнивать объекты этих классов.

В перечисление был добавлен новый элемент как дополнительная функциональность. В главной форме изменились методы для проверки ошибок введённых значений в режиме реального времени. Поэтому некоторые методы по перекраске полей ушли, вместо них появились другие методы, направленные на поиск ошибок.

В таблицах 3.7-3.12 представлены перечисление XWingParameterType и основные классы Parameter, XWing, KompasWrapper, XWingBuilder и MainForm, их поля, методы и свойства в итоговой реализации.

Таблица 3.7 – Класс Parameter

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_value | double | Значение параметра. |
| \_minValue | int | Минимальное допустимое значение параметра. |
| \_maxValue | int | Максимальное допустимое значение параметра. |
| \_minErrorMessage | string | Сообщение о несоблюдении границы минимума. |
| \_maxErrorMessage | string | Сообщение о несоблюдении границы максимума. |
| \_parameterType | XWingParameterType | Тип параметра. |
| \_errors | Dictionary<  XWingParameters, string> | Список ошибок введенного параметра. |
| **Свойства** | Value | double | Устанавливает и возвращает значение параметра. |
| **Методы** | Parameter | Конструктор | Создает объект класса параметра. |
| CheckRange | bool | Проверка принадлежности диапазону введённого параметра. |
| Equals | bool | Проверка на равенство объектов класса. |

Таблица 3.8 – Класс XWing

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Свойства** | Parameters | Dictionary <  XWingParameterType, Parameter> | Устанавливает и возвращает словарь типов параметров - параметров. |
| Errors | Dictionary< XWingParameterType, string> | Устанавливает и возвращает список ошибок параметра. |
| **Методы** | SetParameters | Конструктор | Создаёт объект класса звездолёта для построения. |
| CheckParametersRelationship | void | Проверка взаимосвязи параметров между собой. |

Таблица 3.9 – Перечисление XWingParameterType

|  |  |
| --- | --- |
| **Название элемента** | **Описание** |
| BodyLength | Длина корпуса звездолёта. |
| WingWidth | Ширина крыльев звездолёта. |
| BowLength | Длина носовой части корпуса звездолёта. |
| WeaponBlasterTipLength | Длина острия оружейного бластера звездолёта. |
| AcceleratorTurbineLength | Длина турбины ускорителя звездолёта. |
| AcceleratorNozzleLength | Длина сопла ускорителя звездолёта. |
| CaseBodySetHeight | Высота установок крыши корпуса. |

Таблица 3.10 – Класс KompasWrapper

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_kompas | KompasObject | Объект Компас API. |
| \_part | ksPart | Деталь. |
| \_document | ksDocument3D | Документ-модель. |
| **Свойства** | DefaultPlaneXoY | Obj3dType | Возвращает базовую плоскость XOY. |
| **Методы** | StartKompas | void | Запуск Компас-3D. |
| CreateDocument | void | Создание документа в Компас-3D. |
| SetDetailProperties | void | Установка свойств детали: цвета и имени. |
| CreatePlaneByPoint | ksEntity | Создание плоскости эскиза по точке. |
| CreateSketchDefinition | ksSketchDefinition | Формирование параметров эскиза. |
| CreateSegments | void | Создание на эскизе отрезков. |
| CreateCircles | void | Создание на эскизе окружностей. |
| CreateArs | void | Создание на эскизе дуг. |
| ExtrudeSketch | void | Выдавливание эскиза на определенное расстояние. |
| CutExtrusion | void | Вырезание выдавливанием по эскизу. |
| CreateChamfer | void | Создание фаски. |
| CreateFillet | void | Создание скругления. |
| BuildPolygonByDefaultPlane | ksEntity | Создание эскиза многоугольника по базовой плоскости. |
| BuildPolygonSketchByPoint | ksEntity | Создание эскиза многоугольника по точке. |

Окончание таблицы 3.10

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Методы** | BuildPolygons | ksEntity | Создание многоугольников по координатам вершин. |
| BuildCirclesSketch | ksEntity | Создание эскиза окружностей. |
| BuildSegmentsWithArcs | ksEntity | Создание эскиза из отрезков и дуг. |
| BuildSegmentsWithCircles | ksEntity | Создание эскиза из отрезков и кругов. |
| BuildSetSegments | ksEntity | Построение набора отрезков. |
| BuildHemisphere | void | Построение полусферы. |

Таблица 3.11 – Класс XWingBuilder

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_wrapper | KompasWrapper | Связь с Компас-3D. |
| **Методы** | BuildDetail | void | Построение детали по заданным параметрам. |
| BuildBowBody | void | Построение носовой части корпуса. |
| BuildBody | void | Построение корпуса. |
| BuildWings | void | Построение крыльев. |
| BuildBlasters | void | Построение бластерного оружия. |
| BuildAccelerators | void | Построение ускорителей. |
| ChangeCirclesRadius | void | Изменение радиуса у массива кругов. |
| BuildBlasterCylindersWithShift | void | Построение цилиндров со смещением плоскости их основания. |
| BuildAntenna | void | Построение антенны. |

Таблица 3.12 – Класс MainForm

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Поля** | \_xWingBuilder | XWingBuilder | Объект класса построения детали. |
| \_parameterToTextBox | Dictionary <  XWingParameters, string> | Словарь Тип параметра-TextBox. |
| \_xWing | XWing | Объект параметров X-Wing. |
| **Методы** | MainForm | Конструктор | Конструктор основной формы. |
| BuildButton\_Click | void | Построение по введённым параметрам звездолёт. |

Окончание таблицы 3.12

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Тип** | **Описание** |
| **Методы** | BanCharacterInput | void | Запрет ввода символов и больше одной точки в число. |
| FindError | void | Проверка введенных значений в режиме реального времени. |
| CheckEmptyTextBox | bool | Поиск пустых TextBox. |

Из-за большого числа классов диаграмму вновь необходимо делить на несколько частей. Метод представления архитектуры программы аналогичен тому, что представлен выше на рисунках 3.1-3.5. Итоговые диаграммы представлены на рисунках 3.6-3.9.

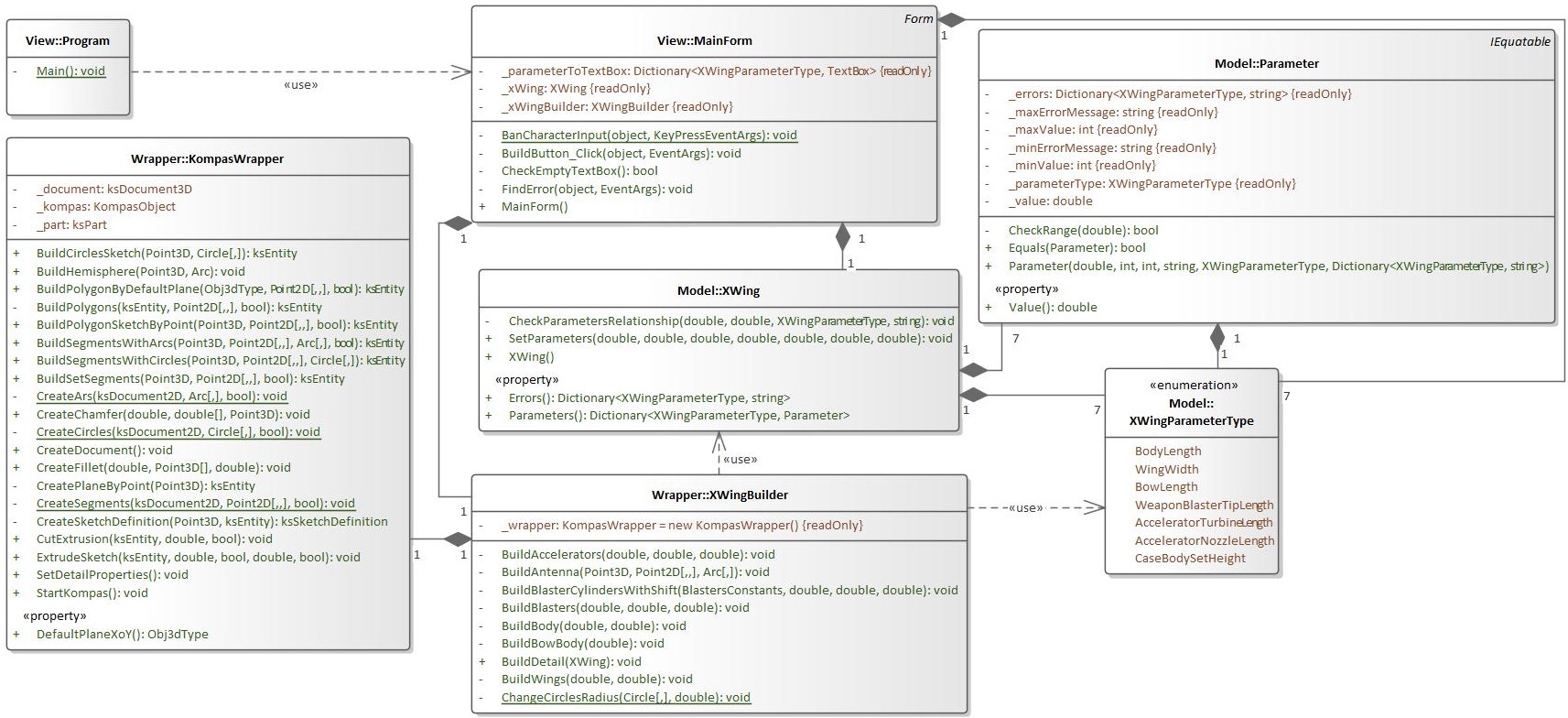


Рисунок 3.6 – Связь основной формы с классами параметров, построителя, обёртки

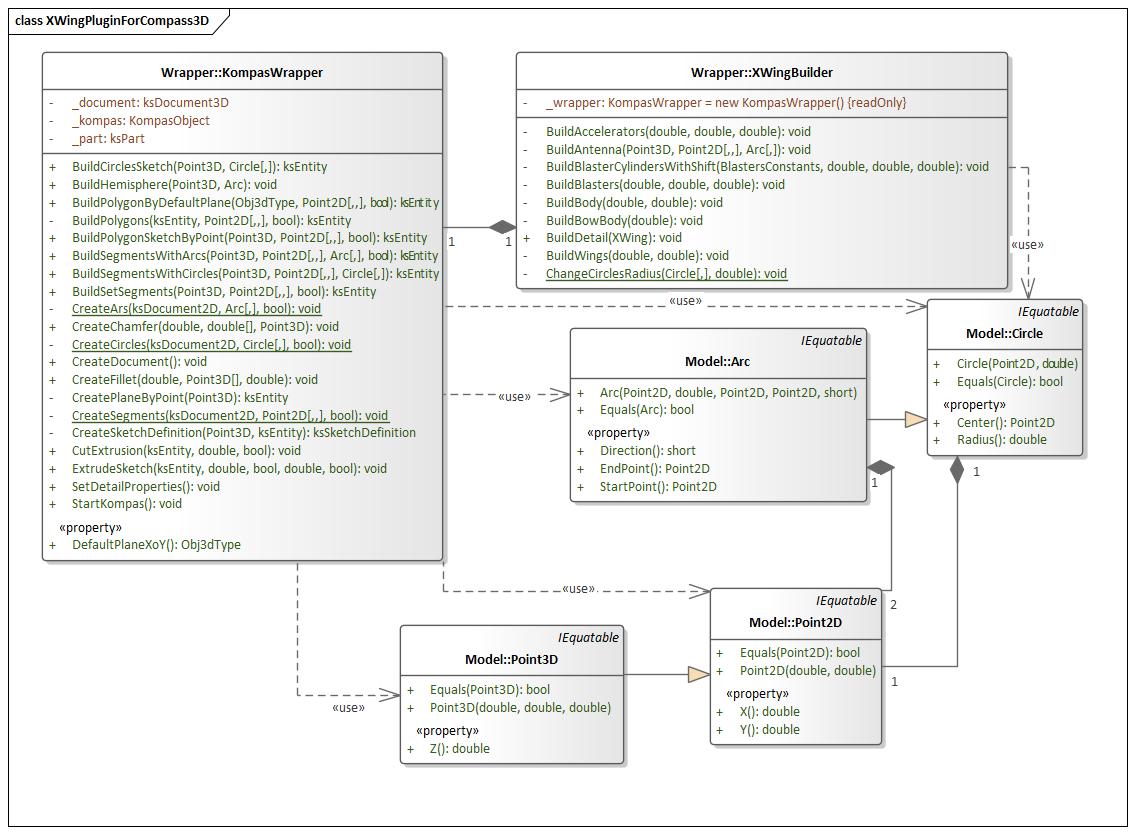


Рисунок 3.7 – Связь обёртки с построителем и классами геометрических фигур

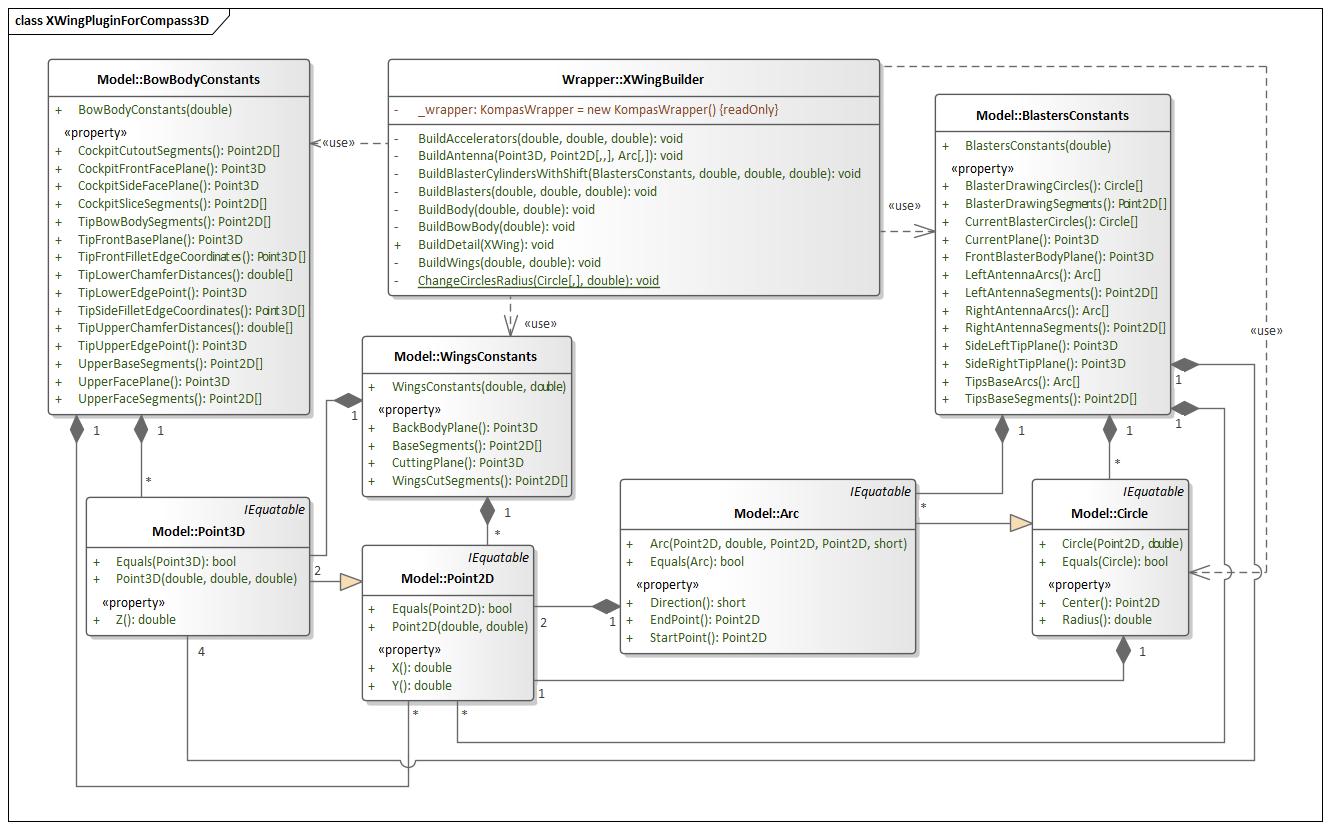


Рисунок 3.8 – Связь построителя с константами построения носовой части корпуса, крыльев и бластеров

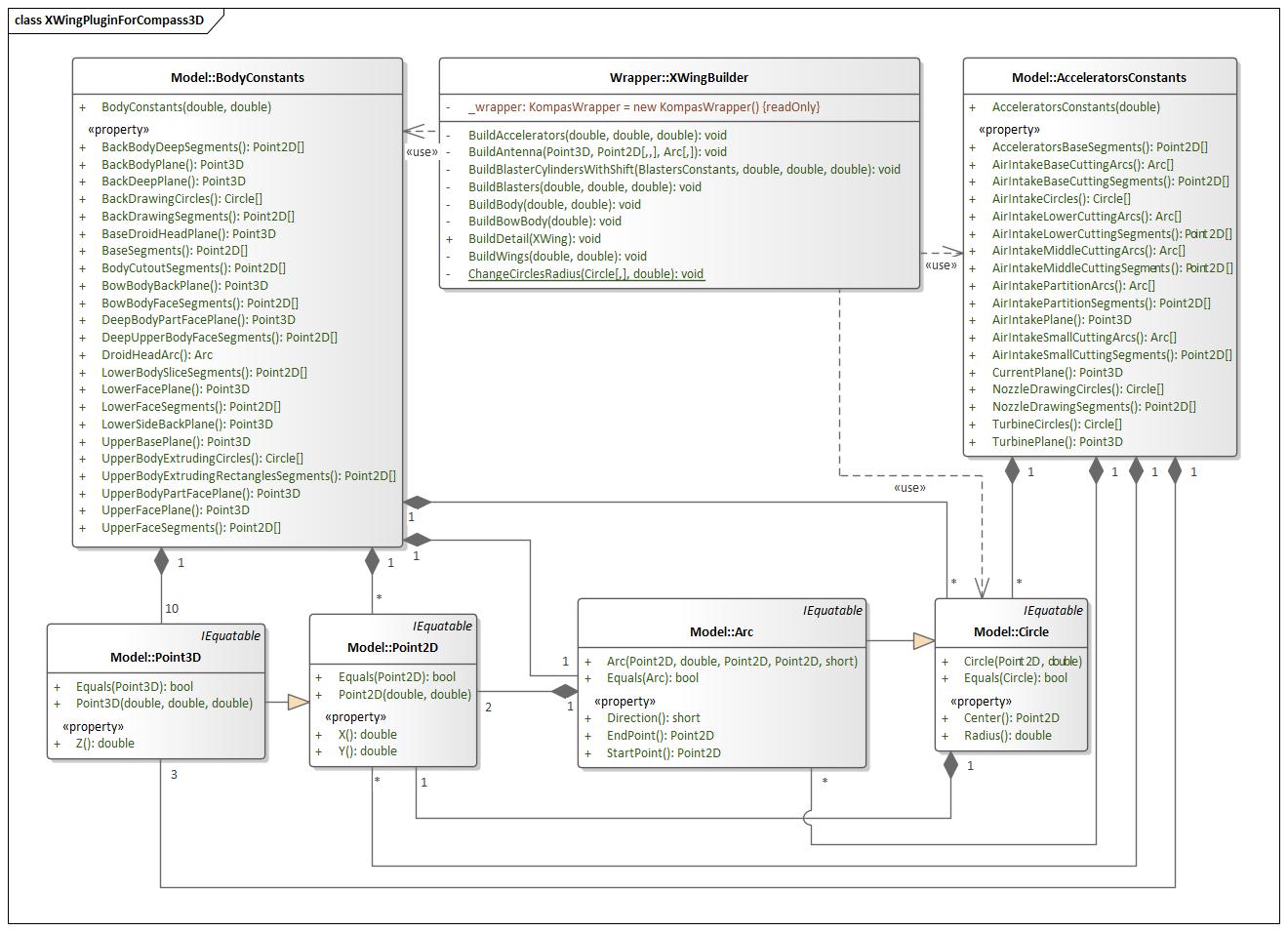


Рисунок 3.9 – Связи построителя с константами построения корпуса и ускорителей

# 4 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПОЛЬЗОВАТЕЛЯ

Пользовательский интерфейс (UI) – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы [11].

Плагин представляет собой пользовательскую форму с полями для ввода соответствующих параметров, форму можно увидеть на рисунке 4.1. Посредством кнопки «Построить» осуществляется запуск САПР «Компас-3D», на рабочей области которой построится трехмерная модель детали по заданным параметрам. Если построение детали выполняется несколько раз, то плагин не запускает несколько копий программы «Компас-3D», а создает в ней новый документ. В случае ввода значений параметров, не в ходящих в допустимый диапазон, поле для ввода окрашивается в розовый цвет и выводится окно в режиме реального времени. При нажатии на кнопку построить появляется окно, информирующее пользователя о некорректности введенного значения. Пример неправильного ввода представлен на рисунке 4.2, пример вывода списка ошибок представлен на рисунке 4.3.

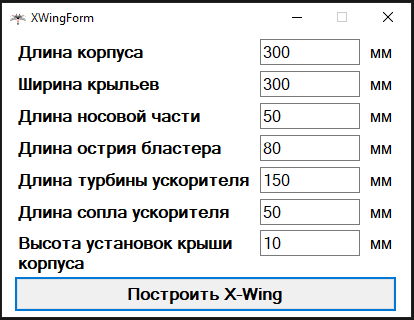


Рисунок 4.1 – Макет пользовательского интерфейса

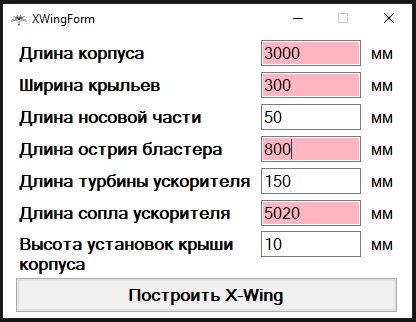


Рисунок 4.2 – Реакция приложения на ввод некорректных значений

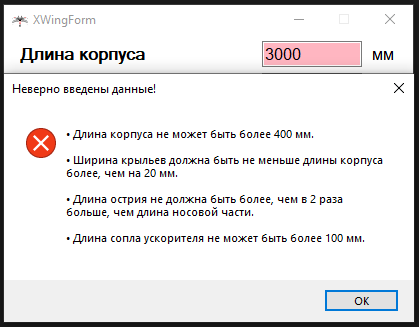


Рисунок 4.3 – Список ошибок на ввод некорректных значений

Если пользователь ввёл правильные значения, при нажатии кнопки «Построить» происходит построение детали. Трёхмерная модель звёздного истребителя T-65 «X-Wing», построенная в системе автоматизированного проектирования Компас-3D с помощью разработанной библиотеки представлена на рисунках 4.5 и 4.6.

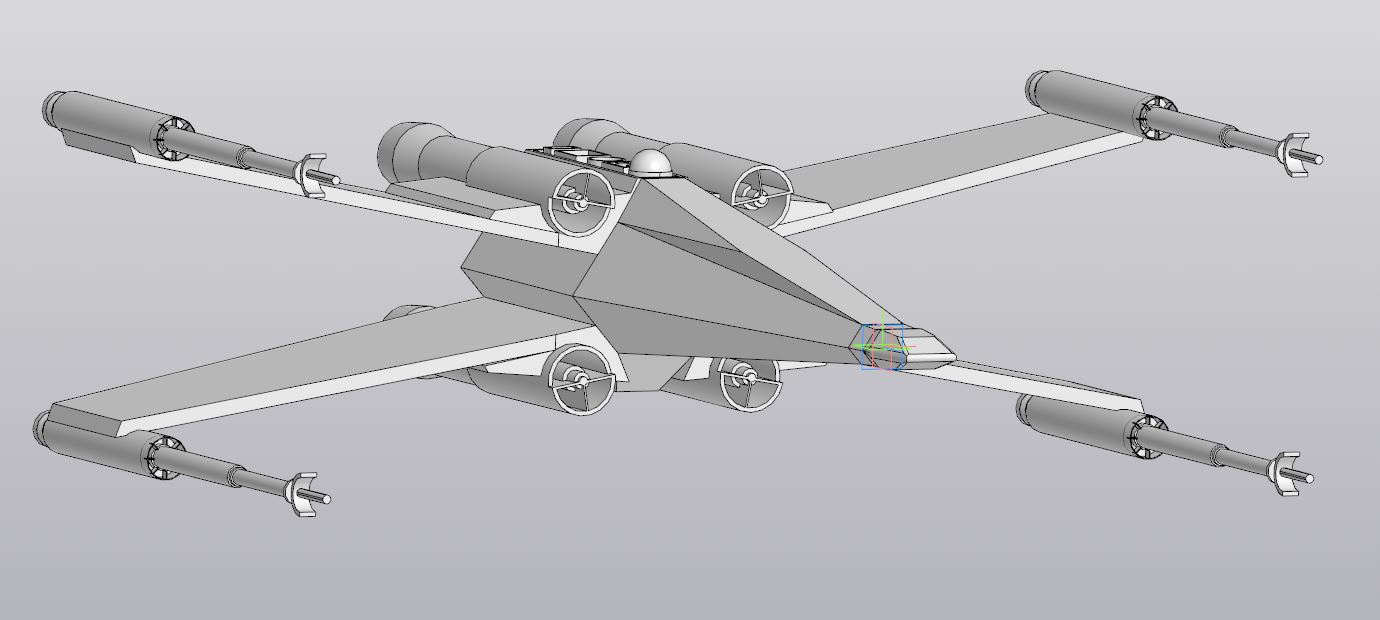


Рисунок 4.5 – Звёздный истребитель T-65 «X-Wing» в виде спереди

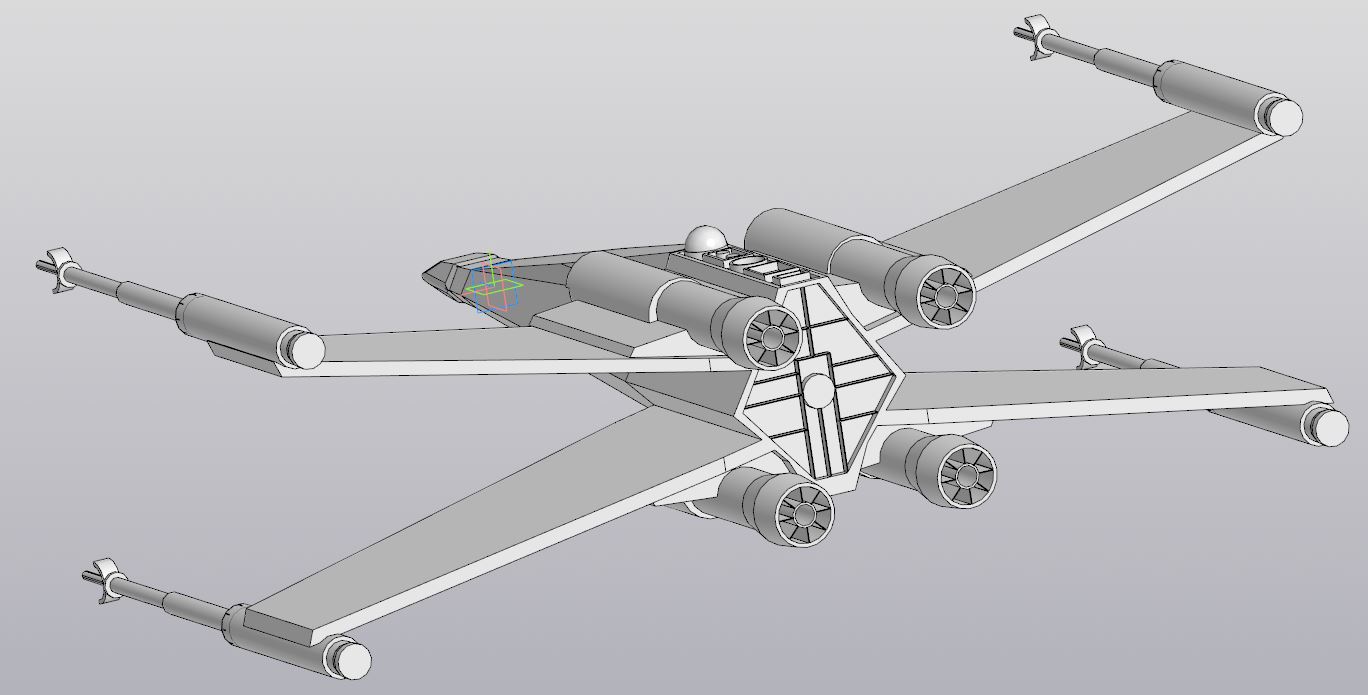


Рисунок 4.6 – Звёздный истребитель T-65 «X-Wing» в виде сзади

# 5 ТЕСТИРОВАНИЕ ПРОГРАММЫ

# 5.1 Функциональное тестирование

Функциональное тестирование – это тип тестирования программного обеспечения, при котором тестовые сценарии выполняются тестировщиком вручную без использования автоматизированных инструментов. Целью ручного тестирования является выявление ошибок, проблем и дефектов в программном приложении. Функциональное тестирование программного обеспечения – это самый примитивный метод из всех видов тестирования. Концепции ручного тестирования не требуют знания какого-либо инструмента тестирования [12].

В рамках функционального тестирования будет проверка библиотеки при введённых минимальных, средних и максимальных параметрах.

Тестирование при минимальных параметрах:

* длина корпуса равна 300 мм;
* ширина крыльев равна 300 мм;
* длина носовой части равна 50 мм;
* длина острия оружейного бластера равна 80 мм;
* длина турбины ускорителя равна 150 мм;
* длина сопла ускорителя равна 50 мм;
* высота установок крыши корпуса равна 10 мм.

Результат построения при заданных параметрах представлен на рисунках 5.1-5.2.

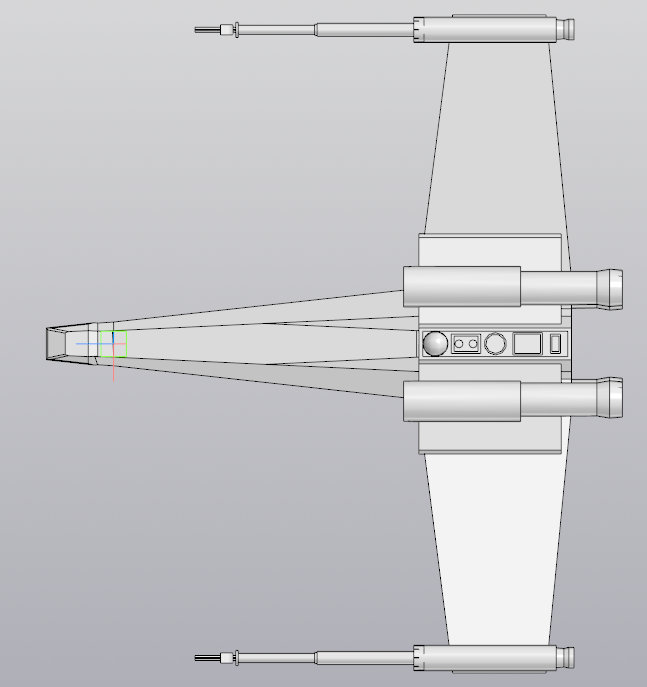


Рисунок 5.1 – Вид на модель в плоскости XZ

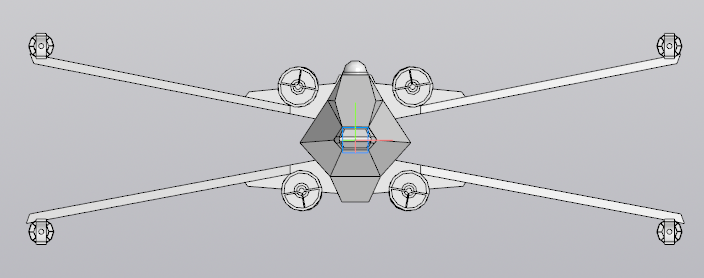


Рисунок 5.2 – Вид на модель в плоскости XY

Тестирование при средних параметрах:

* длина корпуса равна 350 мм;
* ширина крыльев равна 350 мм;
* длина носовой части равна 75 мм;
* длина острия оружейного бластера равна 105 мм;
* длина турбины ускорителя равна 200 мм;
* длина сопла ускорителя равна 75 мм;
* высота установок крыши корпуса равна 15 мм.

Результат построения при заданных параметрах представлен на рисунках 5.3-5.4.

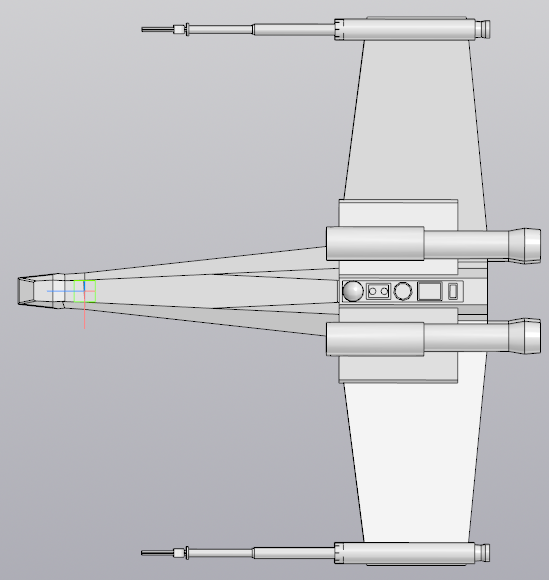


Рисунок 5.3 – Вид на модель в плоскости XZ

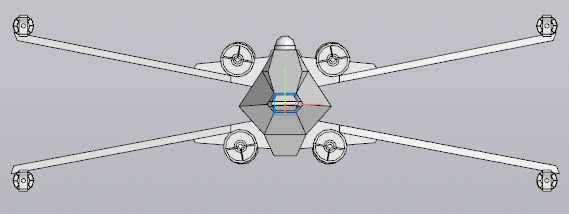


Рисунок 5.4 – Вид на модель в плоскости XY

Тестирование при максимальных параметрах:

* длина корпуса равна 400 мм;
* ширина крыльев равна 400 мм;
* длина носовой части равна 100 мм;
* длина острия оружейного бластера равна 130 мм;
* длина турбины ускорителя равна 250 мм;
* длина сопла ускорителя равна 100 мм;
* высота установок крыши корпуса равна 20 мм.

Результат построения при заданных параметрах представлен на рисунках 5.5-5.6.

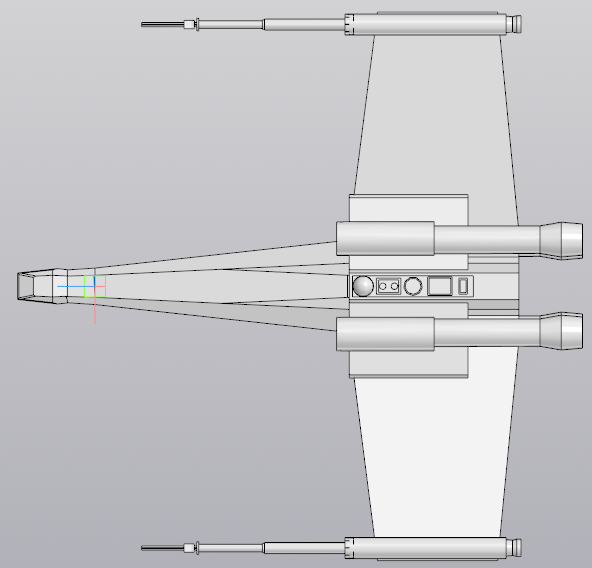


Рисунок 5.5 – Вид на модель в плоскости XZ

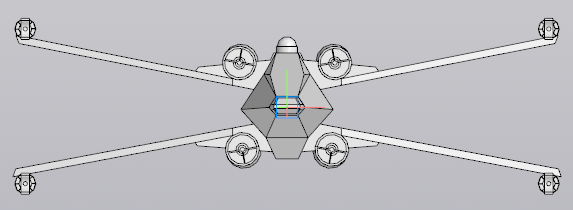


Рисунок 5.6 – Вид на модель в плоскости XY

# 5.2 Модульное тестирование

**Модульное тестирование (Unit Testing)**– это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект [13].

Для тестирования используется библиотеки NUnit 3.13.3 [5] и NUnit3TestAdapter 4.2.1 [5]. Тестами покрыта вся логика программы. Всего было написано 94 теста. На рисунках 5.7-5.12 представлено тестирование логики.

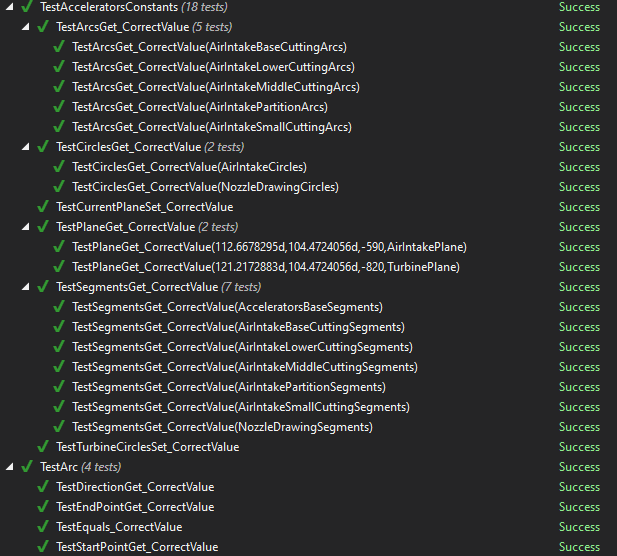


Рисунок 5.7 – Тестирование логики классов AcceleratorsConstants и Arc

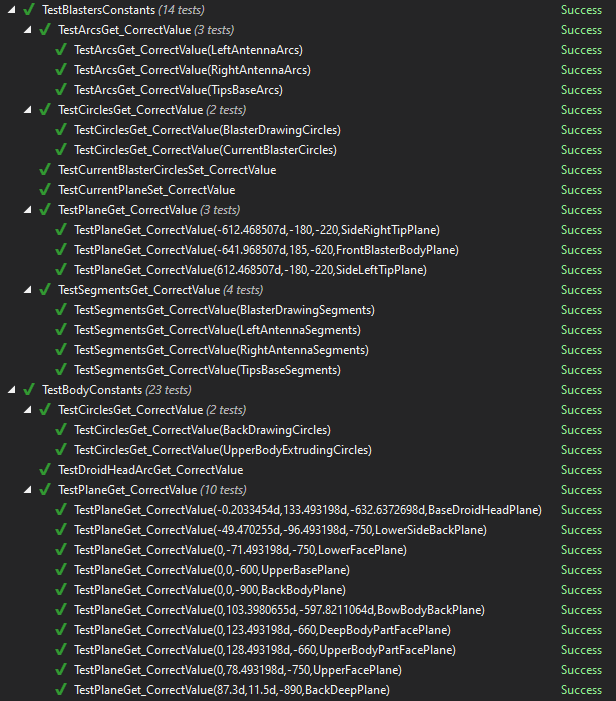


Рисунок 5.8 – Тестирование логики класса BlastersConstants

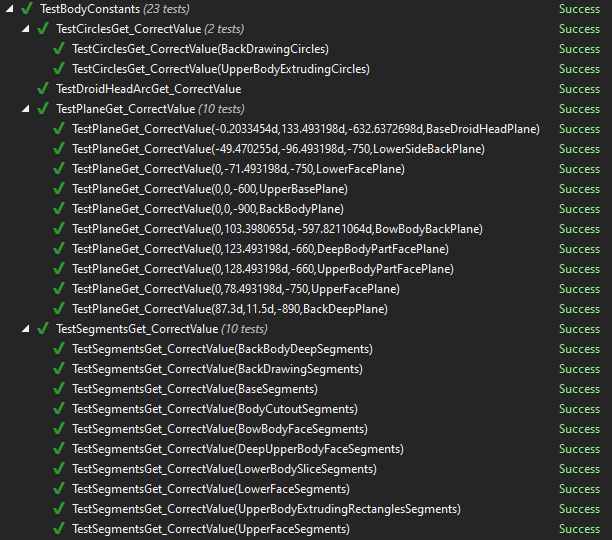


Рисунок 5.9 – Тестирование логики класса BodyConstants

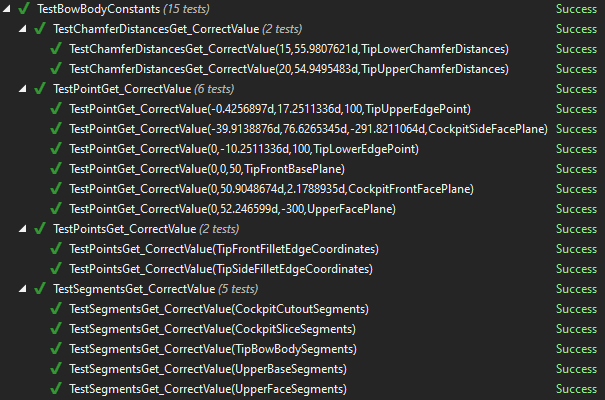


Рисунок 5.10 – Тестирование логики класса BowBodyConstants

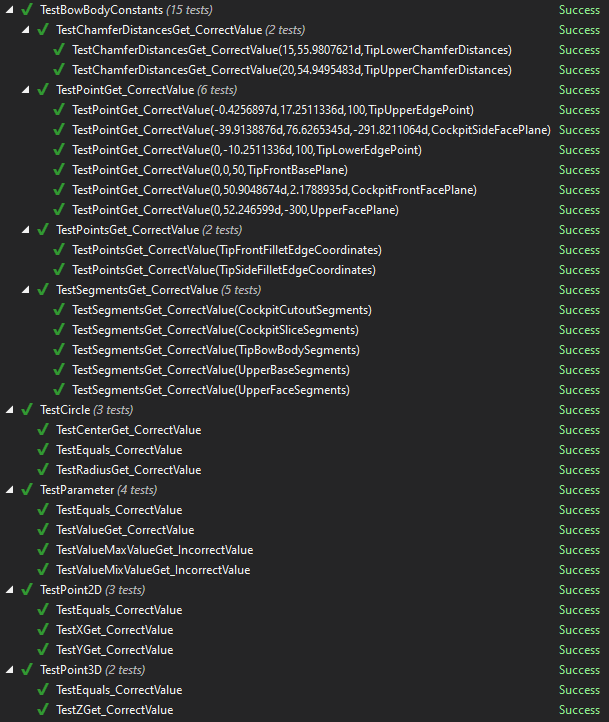


Рисунок 5.11 – Тестирование логики классов Circle, Parameter, Point2D и Point3D

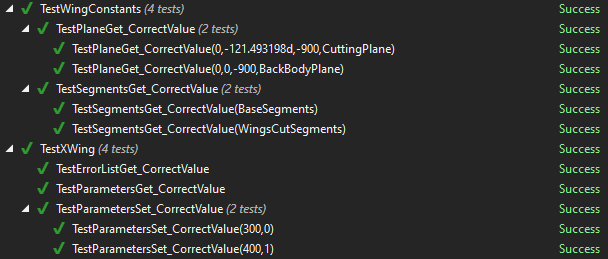


Рисунок 5.12 – Тестирование логики классов WingsConstants и XWing

# 5.3 Нагрузочное тестирование

Нагрузочное тестирование – это подвид 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) [14].

Нагрузочное тестирование проводилось на персональном компьютере со следующей конфигурацией:

* процессор Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz 4.20 GHz;
* оперативная память 16,0 ГБ;
* графический процессор объёмом памяти 12 ГБ.

Для тестирования будет зациклено построение модели со следующими параметрами:

* длина корпуса равна 300 мм;
* ширина крыльев равна 300 мм;
* длина носовой части равна 50 мм;
* длина острия оружейного бластера равна 80 мм;
* длина турбины ускорителя равна 150 мм;
* длина сопла ускорителя равна 50 мм;
* высота установок крыши корпуса равна 10 мм.

Было произведено 7 нагрузочных тестов. В результате любого теста было построено 99 деталей. По полученным измерениям был рассчитан средний результат, по которому были построены графики. На рисунке 5.13 показана зависимость количества используемой оперативной памяти в гигабайтах от количества построенных деталей.



Рисунок 5.13 – Зависимость оперативной памяти от количества деталей

Из графика видно, что зависимость является линейной. На графики имеются скачки вниз, то есть происходит уменьшение нагрузки на оперативную память. Это обусловлено устройством оперативной памяти, что для её регенерации периодически приостанавливается обращение, это снижает среднюю скорость обмена и понижает нагрузку [15]. Также в конце графика видно, что нагрузка на память не уменьшается. Это связано с тем, что операционная система имеет файл подкачки, в который выгружаются неактивные и неиспользуемые данные, снимая нагрузку с оперативной памяти [16].

На рисунке 5.14 показана зависимость потраченного времени от количества построенных деталей.



Рисунок 5.14 – Зависимость времени от количества построенных деталей

Из графика видно, что зависимость количества необходимого времени от числа деталей является линейной.

# Заключение

В результате выполнения данной учебной работы были пройдены все стадии разработки программного обеспечения: выбор темы, составление технического задания, составление проекта системы, создание первой реализации и её тестирование, создание конечной библиотеки. Был изучен API приложения Компас-3D. Также были найдены аналоги разрабатываемого плагина. Были спроектированы UML диаграммы классов и было проведено функциональное, модульное и нагрузочное тестирования созданной библиотеки.

# Список использованных источников

1. Официальный сайт системы автоматизированного проектирования «Компас-3D» [Электронный ресурс] – Режим доступа: https://kompas.ru/kompas-3d/about/ (дата обращения: 30.09.2022)
2. Компас (САПР) [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Компас\_(САПР) (дата обращения: 30.09.2022)
3. Заметки о выпуске Visual Studio 2019 версии 16.10 2019 [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/releases/2019/release-notes-v16.10 (дата обращения: 19.11.2022)
4. Платформа .NET Framework 4.7.2 для Windows [Электронный ресурс] – Режим доступа: https: //support.microsoft.com/ru-ru/topic/автономный-установщик-microsoft-платформа-net-framework-4-7-2-для-windows-05a72734-2127-a15d-50cf-daf56d5faec2 (дата обращения: 19.11.2022)
5. Фреймворк NUnit [Электронный ресурс] – Режим доступа: https://nunit.org/ (дата обращения: 19.11.2022)
6. External workbenches [Электронный ресурс] – Режим доступа: https://wiki.freecadweb.org/External\_workbenches (дата обращения: 06.10.2022)
7. Airplane Design Workbench [Электронный ресурс] – Режим доступа: https://github.com/FredsFactory/FreeCAD\_AirPlaneDesign (дата обращения: 06.10.2022)
8. Rocket Workbench [Электронный ресурс] – Режим доступа: https://wiki.freecadweb.org/Rocket\_Workbench (дата обращения: 06.10.2022)
9. UML-диаграммы классов [Электронный ресурс] – Режим доступа: https://prog-cpp.ru/uml-classes (дата обращения: 06.10.2022)
10. Enterprise Architect (software) [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Enterprise\_Architect\_(software) (дата обращения: 06.10.2022)
11. Интерфейс пользователя [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Интерфейс\_пользователя (дата обращения: 06.10.2022)
12. Функциональное тестирование [Электронный ресурс] – Режим доступа: https://logrocon.ru/functional\_testing (дата обращения: 30.10.2022)
13. Модульное тестирование [Электронный ресурс] – Режим доступа: https://logrocon.ru/news/unit\_testing (дата обращения: 30.10.2022)
14. Нагрузочное тестирование [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Нагрузочное\_тестирование (дата обращения: 19.11.2022)
15. Оперативная память [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Оперативная\_память (дата обращения: 30.10.2022)
16. Файл подкачки [Электронный ресурс] – Режим доступа: https://www.zeluslugi.ru/info-czentr/it-glossary/term-file-podkachki (дата обращения: 19.11.2022)