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 "Корпус для компьютера" для САПР компас-3D v20  
по дисциплине «Основы разработки САПР»

Выполнил:

студент гр. 588-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Третьяков В.В.

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

Руководитель:

к.т.н., доцент каф. КСУП

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

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

**Реферат**

Пояснительная записка 37 с., 19 рис., 13 таблиц, 8 источников.

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

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

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

Так же, после получения задания на добавление дополнительной функциональности, плагин получил возможность строить трехмерную модель в Inventor.

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

Оглавление

[1. Описание САПР 4](#_Toc97803446)

[1.1 Описание программы «Компас - 3D» 4](#_Toc97803447)

[1.2 Описание API САПР КОМПАС-3D 5](#_Toc97803448)

[1.3 Обзор аналогов 10](#_Toc97803449)

[2 Описание предмета проектирования 11](#_Toc97803450)

[3 Стек технологий и системные требования 13](#_Toc97803451)

[4. Описание плагина 14](#_Toc97803452)

[4.1 Диаграмма классов 14](#_Toc97803453)

[4.2 Макет пользовательского интерфейса 23](#_Toc97803454)

[5 Тестирование программы 26](#_Toc97803455)

[5.1 Функциональное тестирование 26](#_Toc97803456)

[5.2 Модульное тестирование 30](#_Toc97803457)

[5.3 Нагрузочное тестирование 33](#_Toc97803458)

[Заключение 36](#_Toc97803459)

[Список используемых источников 37](#_Toc97803460)

# 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].

## 1.2Описание API САПР КОМПАС-3D

API (англ. ApplicationProgrammingInterface) – 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  (shortstructType) | 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 интерфейсаksDocument2D, 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(inttype) | 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  (shortobjType) | objType – тип объектов, содержащихся в массиве. | ksEnintyCollection | Формирует массив объектов и возвращает указатель на его интерфейс |
| GetDefaultEntity  (shortobjType) | objType – тип объекта | ksEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart (inttype) | Type – тип компонента | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity (shortobjType) | 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

**Edelws онлайн конфигуратор**

Edelws конфигуратор — инструмент для выбора и создания системного блока из представленных на сайте комплектующих [5].

Данный конфигуратор содержит множество всевозможных комплектующих, разделенных на группы. Пользователи могут создавать собственные сборки и сохранять под своей учетной записью. При добавлении комплектующих, справа отображается общая цена всех комплектующих.

Интерфейс конфигуратора представлен на рисунке 1.1

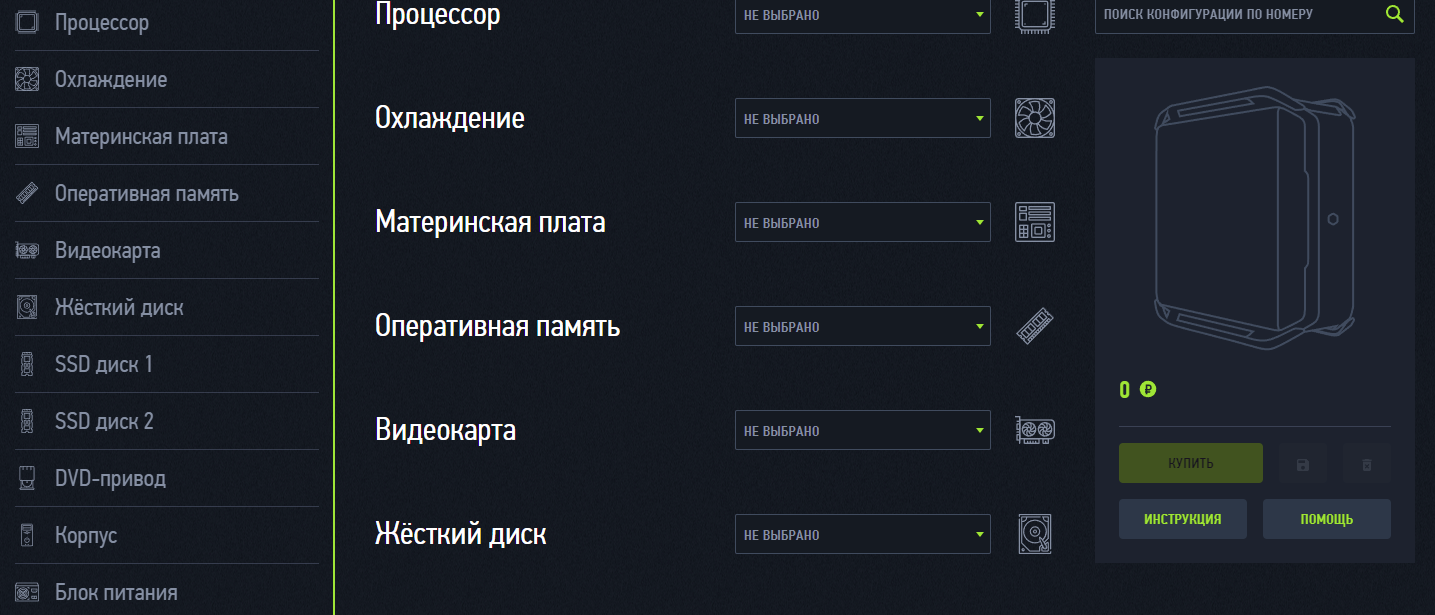


Рисунок 1.1 – Интерфейс конфигуратора Edelws.

Библиотека помогает в решении следующих задач:

– автоматизация сложных повторяющихся процессов;

– графический редактор профессионального уровня для планов высочайшей точности;

– запуск трехмерного и двухмерного виртуального просмотра для более наглядной работы;

– создание реалистичных изображений с солнечными бликами и тенями, направлением света и пр. [4].

# 2 Описание предмета проектирования

Компьютерный корпус (англ. Computer case) — физически представляет собой базовую несущую конструкцию (шасси), которая предназначена для последующего наполнения аппаратным обеспечением с целью создания компьютера.

Модель компьютерного корпуса представлена на рисунке 2.1.

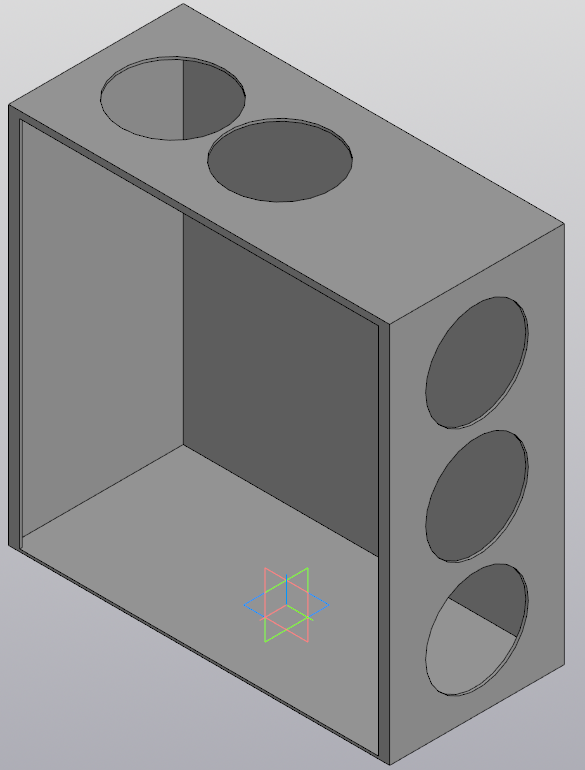


Рисунок 2.1 – Компьютерный корпусПараметры компьютерного корпуса (рисунок 2.2):

* высота корпуса A (330мм-500мм);
* длинна корпуса B (244мм-500мм);
* ширина корпуса C (140мм-250мм);
* диаметр передних отверстий для вентиляторов E (40мм-140мм);
* диаметр верхних отверстий для вентиляторов D (40мм-140мм);
* тип материнской платы (ATX / micro-ATX).

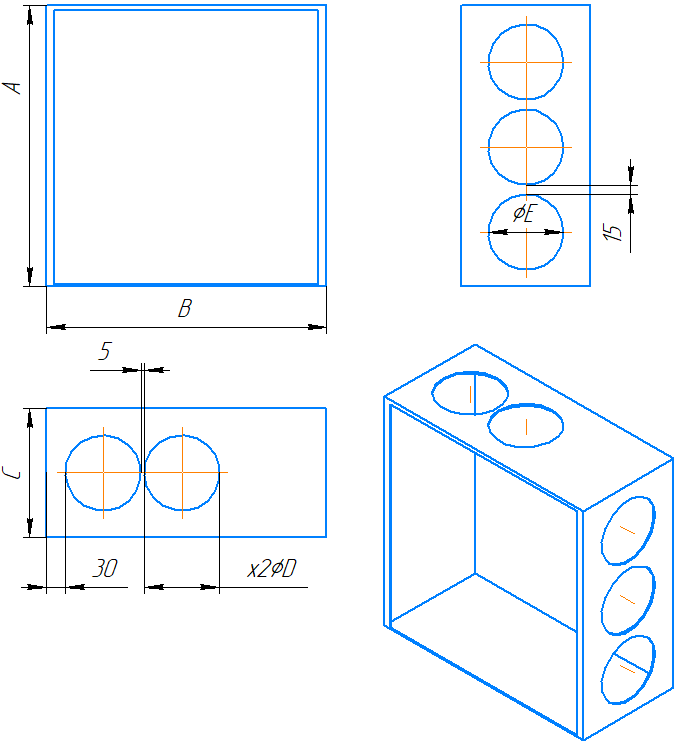


Рисунок 2.2 – Параметры компьютерного корпуса

Ограничения:

* диаметр отверстий под вентиляторы не должен быть больше ширины корпуса;
* диаметр верхних вентиляторов не должен быть больше длинны корпуса;
* диаметр передних вентиляторов не должен быть больше высоты корпуса.

# 3 Стек технологий и системные требования

Язык программирования: C#;

Среда разработки: VisualStudio 2019 версия 16.11.10 с применением Microsoft.NET Framework 4.7.2;

Тестирование с помощью библиотек: NUnit версия 3.12.0, NUnit3 TestAdapter версия 3.16.1.;

Система КОМПАС 3D v20.

Система Inventor 2022.

Для реализации пользовательского интерфейса использовался 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 [6].

Для взаимодействия плагина с системой Inventor 2022, использована библиотека «Inventor».

# 4. Описание плагина

## 4.1 Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы и их взаимосвязи между ними, их коопераций, атрибутов (полей), методов [7]

Изначальная диаграмма классов плагина представлена на рисунке 4.1.

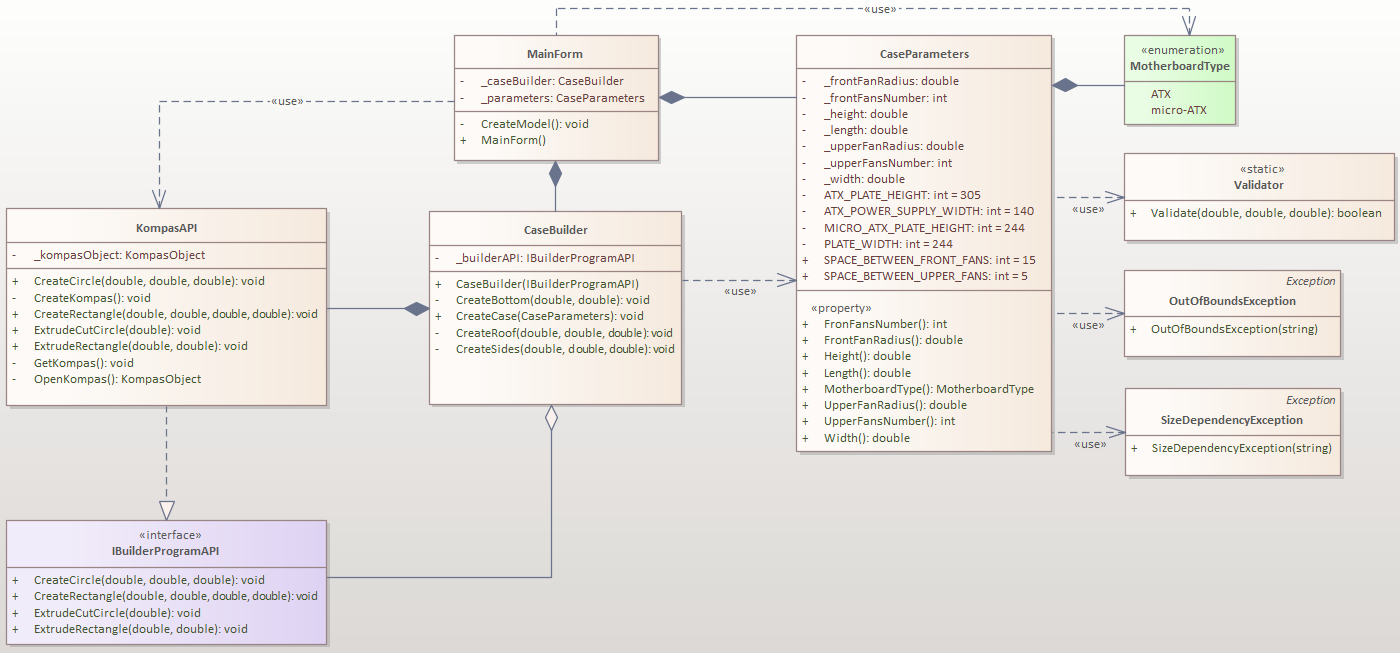


Рисунок 4.1 – Диаграмма классов плагина «Компьютерный корпус»

Класс MainForm содержит в себе экземпляры классов CaseParameters и CaseBuilder.CaseParameters содержит в себе данные для построения корпуса, перед присвоением параметров, данные проходят проверку используя статический класс Validator.

При попытке занесения данных в поля CaseParameters, вызывается статический метод Validate, принимающий два ограничения и само число, которое необходимо проверить. При неудачной проверке, в зависимости от проверяемого поля, генерируется исключение. OutOfBoundException– если число не входит в указанный диапазон. SizeDependencyException – если ошибка возникла в соотношениях зависимых полей.

Класс KompassApi, реализует интерфейс IBuilderProgramAPI для работы с САПР Компас-3D и построения в нем необходимых элементов. В перечислении ParameterName хранятся именования параметров.

Таблица 3.1 – КлассCaseBuilder

|  |  |  |
| --- | --- | --- |
| Название | Тип входных параметров | Описание |
| CaseBuilder() | IBuilderProgramAPI – экземпляр объекта класса, реализующего данный интерфейс | Конструктор класса CaseBuilder |
| CreateBotom() | double – ширина корпуса  double – длина корпуса | Вызывает методы необходимые для создания дна корпуса, используя объект builderAPI |
| CreateRoof() | double – ширина корпуса;  double – длина корпуса;  int – количество вентиляторов | Вызывает методы необходимые для создания крыши корпуса с отверстиями под вентиляторы, используя объект builderAPI |
| CreateSides() | double – ширина корпуса  double – высота корпуса  double – длинна корпуса  int – количество вентиляторов | Вызывает методы необходимые для создания стенок корпуса с отверстиями под вентиляторы на передней стороне, используя объект builderAPI |
| \_builderAPI | IBuilderProgramAPI– объекткласса, реализующегоинтерфейсIBbuilderProgramAPI, содержащийметодыдляпостроениеэлементов | Используется для вызова методов построения элементов корпуса |

Таблица 3.2 – КлассKompasAPI

|  |  |  |
| --- | --- | --- |
| Название | Тип входных параметров | Описание |
| CreateCircle() | double – диаметр отверстия  double – центр по Х  double – центр по Y | Создает эскиз для отверстия под вентилятор |
| CreateRectangle() | double – верхний край по X  double – верхний край по Y  double – нижний край по X  double – нижний край по Y | Создает эскиз прямоугольника |

Прододжение таблицы 3.2

|  |  |  |
| --- | --- | --- |
| Название | Тип входных параметров | Описание |
| ExtrudeCutCircle() | double – глубина, на которую необходимо вырезать | Вырезает круг на указанную глубину |
| ExtrudeRectangle() | double – глубина, на которую необходимо вырезать | Выдавливает эскиз прямогульника |
| OpenCompas() |  | Открывает программу компас и возвращает объект KompasObject |
| \_kompasObject | KompasObject - главный интерфейс API системы КОМПАС, реализующий основной функционал взаимодействия | Закрытое поле, содержащее объект, API системы КОМПАС, реализующий основной функционал взаимодействия |

Таблица 3.3 – КлассValidator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| Validate() | double– верхняя числовая граница  double–нижняя числовая граница  double–число, которое необходимо проверить | true – если число входит в указанный диапазон  false – если число не входит в указанный диапазон | Проверяет, входит ли число в указанный диапазон значений |

Таблица 3.4 – Класс CaseParameters

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| FrontFansNumber() | int – количествоотверстий вентиляторовна передней части корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |
| FrontFansRadius() | double – радиус отверстий под вентиляторы на передней части корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |
| Height() | double – высота корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |
| Length() | double – длина корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |
| MotherboardType() | enum– описывающий тип материнской платы | При установке значения проверяется правильность введенных значений, используя класс Validator |
| UpperFansRadius() | double – радиус отверстий под вентиляторы на верхней части корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |

Продолжение таблицы 3.4

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| UpperFansNamber() | int–количество отверстий вентиляторов на передней части корпуса | При установке значения проверяется правильность введенных значений, используя класс Validator |
| \_frontFansNumber | int – количество отверстий вентиляторов на передней части корпуса | Закрытое поле, содержащее количество вентиляторов на передней части корпуса |
| \_frontFansRadius | double – радиус отверстий под вентиляторы на передней части корпуса | Закрытое поле, содержащее радиус отверстий под вентиляторы на передней части корпуса |
| \_height | double – высота корпуса | Закрытое поле, содержащее высоту корпуса |
| \_length | double – длина корпуса | Закрытое поле, содержащее длину корпуса |
| \_upperFansRadius | double – радиус отверстий под вентиляторы на верхней части корпуса | Закрытое поле, содержащее радиус отверстий под вентиляторы на верхней части корпуса |
| \_upperFansNamber | int–количество отверстий вентиляторов на передней части корпуса | Закрытое поле, содержащее количество отверстий вентиляторов на передней части корпуса |

Продолжение таблицы 3.4

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| ATX\_PATE\_HEIGHT | int – высота ATXматеринской платы | Константа, описывающая высоту ATXматеринской платы, равный 305 |
| MICRO\_ATX\_PLATE\_HEIGHT | int – высота micro-ATXматеринской платы | Константа, описывающая высоту micro-ATXматеринской платы, равный 244 |
| PLATE\_WIDTH | int – ширина материнских плат ATXи micro-ATX | Константа, описывающая ширину материнских плат ATXи micro-ATX |
| SPACE\_BEETWEEN\_FRONT | int – расстояние между передними вентиляторами | Константа, описывающая расстояние между передними вентиляторами |
| SPACE\_BEETWEEN\_UPPER | int – расстояние между верхними вентиляторами | Константа, описывающая расстояние между верхними вентиляторами |

Таблица 3.5 – Класс MainForm

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| CreateModel() |  | Вызывает метод CreateCase, используя объекткласса CaseBuilder |
| \_caseBuilder | CaseBuilder – объект класcа, осуществляющего построение корпуса | Закрытое поле объекта класcа, осуществляющего построение корпуса |
| \_parameters | CaseParameters – объект класса, хранящего в себе данные корпуса | Закрытое поле, объекта класса, хранящего в себе данные корпуса |

Диаграмма классов после добавления дополнительной функциональности представлена на рисунке 4.2.

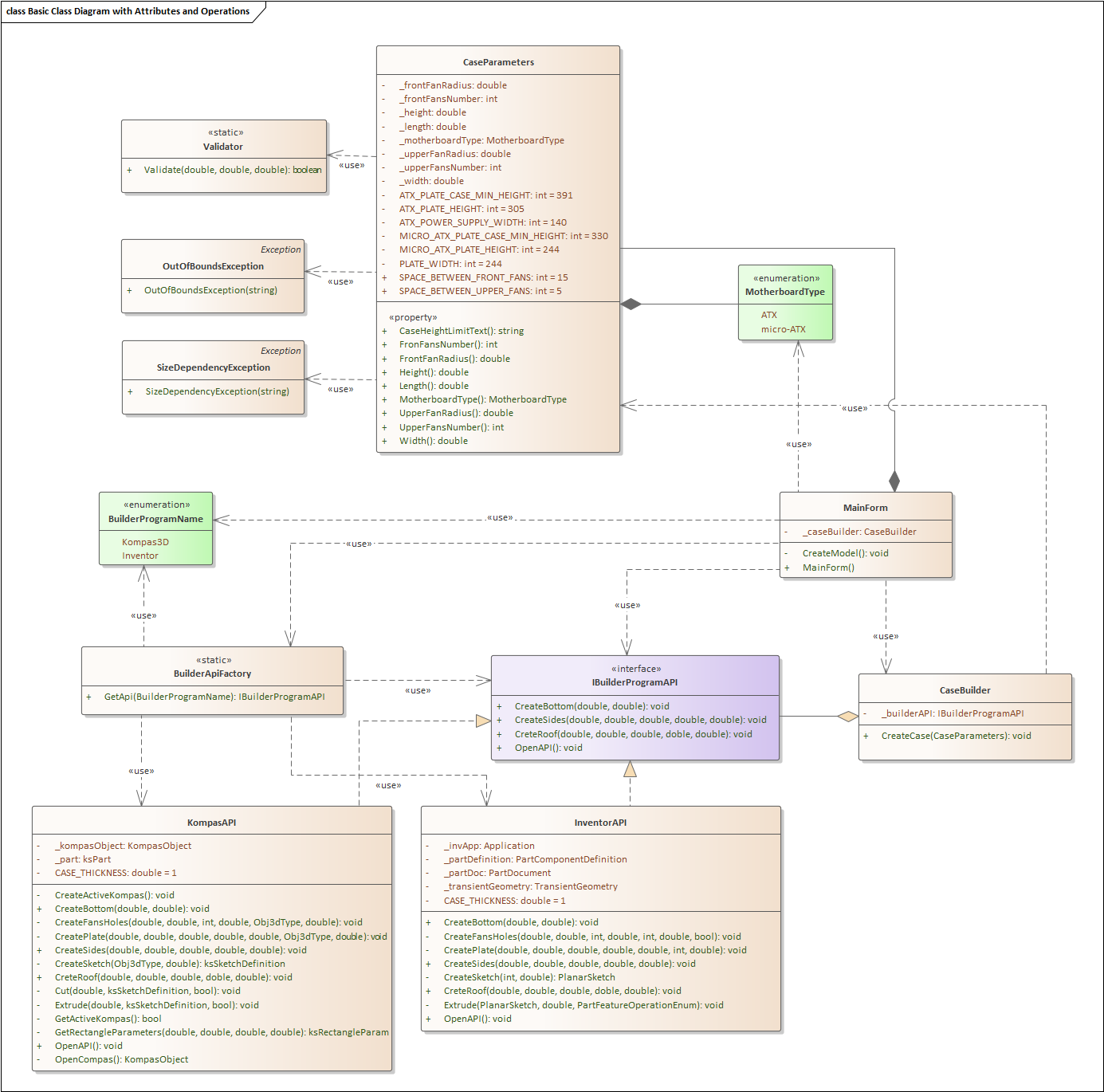


Рисунок 4.2 – Диаграмма классов с дополнительной функциональностью

В процессе работы изначальная UML-диаграмма была изменена, список изменений:

– Добавлено перечисление BuilderProgramName, использующееся для выбора программы для построения корпуса.

– Добавлен класс InventorAPI, реализующий интерфейс IBuilderProgram и осуществляющий работу с программой Inventor.

– Добавлен сервисный класс BuilderApiFactory, возвращающий по полученному значению перечисления BuilderProgramName класс, осуществляющий построение корпуса в выбранном API.

– Изменены методы в интерфейсе IBuilderProgram, так как изначальный подход с прошлыми методами оказался неэффективен.

## 4.2 Макет пользовательского интерфейса

Пользовательский интерфейс представляет собой форму для ввода параметров. При нажатии на кнопку «Построить» строится 3D-модель корпуса. В правой части интерфейса расположены подсказки в виде картинки. На рисунке 4.3 представлен макет пользовательского интерфейса.

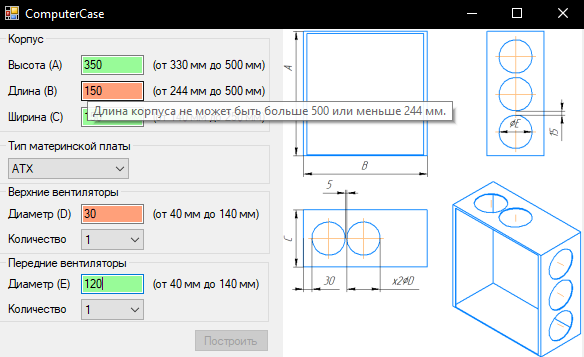


Рисунок 4.3 – Макет пользовательского интерфейса

Кнопка «Построить» остается неактивной до тех пор, пока в поля со значениями не будут введены корректные значения. При вводе некорректных значений, соответствующее поле будет подсвечено красным. Поле с некорректным значением, при наведении курсора мыши, выводит конкретную ошибку, допущенную пользователем.

При возникновении ошибок соотношений зависимостей, например, диаметр верхних вентиляторов и длина корпуса, красным цветом подсвечиваются оба поля.

На рисунке 4.4 изображено диалоговое окно документа САПР «КОМПАС – 3D» после нажатия кнопки «Построить модель» и построения модели.

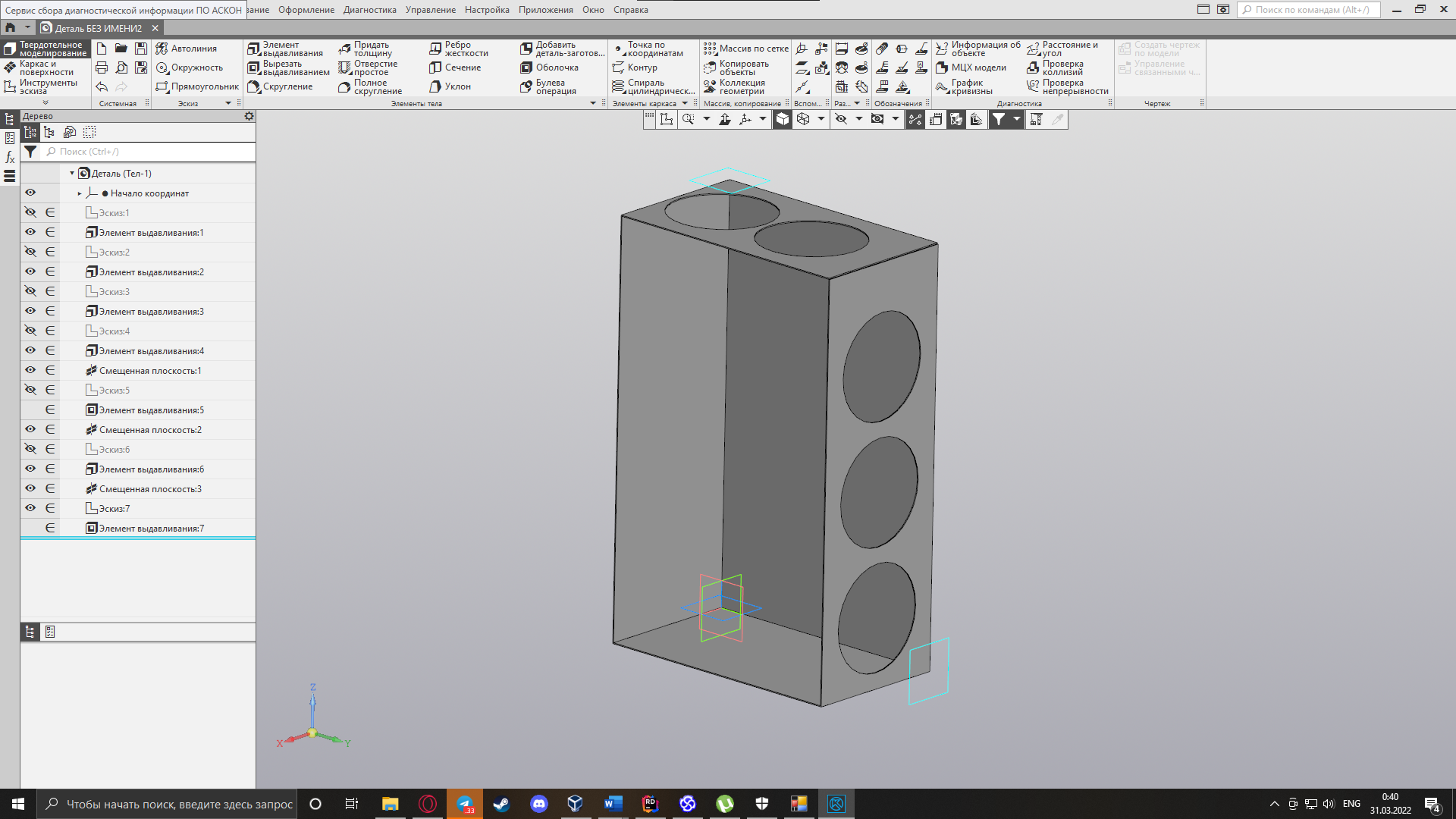


Рисунок 4.4 – Диалоговое окно документа САПР «КОМПАС – 3D»

Макет пользовательского интерфейса после добавления дополнительной функциональности представлен на рисунке 4.5.

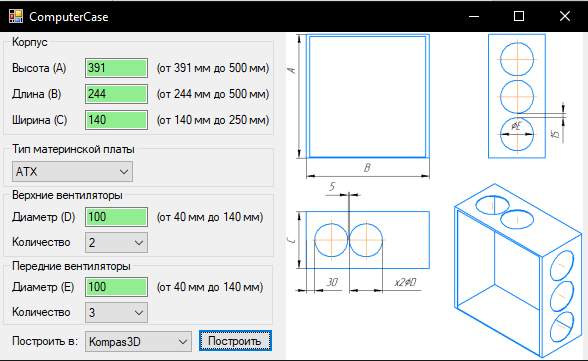


Рисунок 4.5 – Макет пользовательского интерфейса после добавления дополнительной функциональности.

На рисунке 4.6 изображено диалоговое окно документа САПР «Inventor» после нажатия кнопки «Построить модель» и построения модели с дополнительной функциональностью. Сама деталь изменениям не подвергалась.

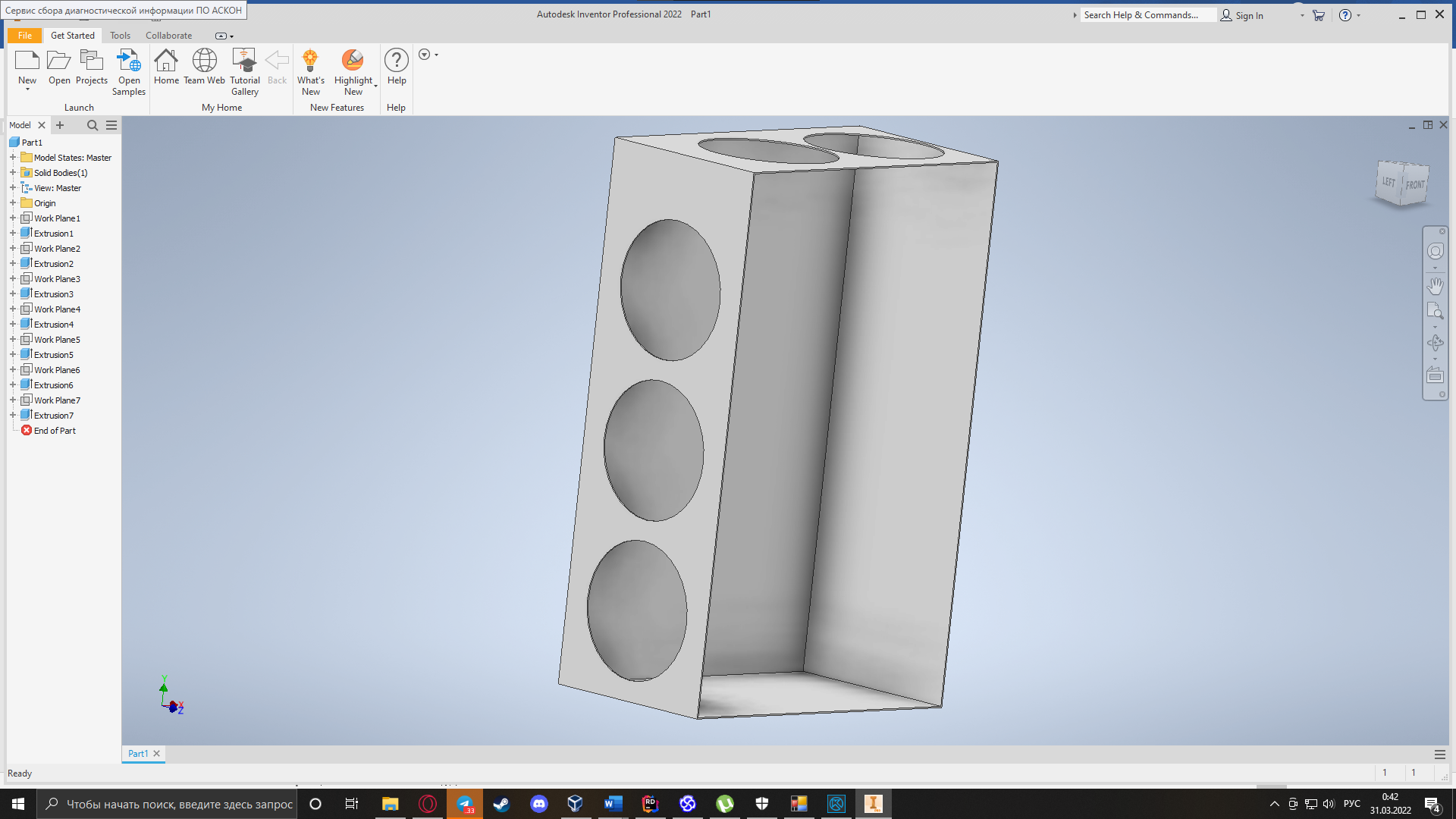


Рисунок 4.6 – Диалоговое окно документа САПР «Inventor»

# 5 Тестирование программы

Тестирование — это проверка соответствия объекта желаемым и требуемым критериям [8]. Несоответствие критериям называется ошибкой.

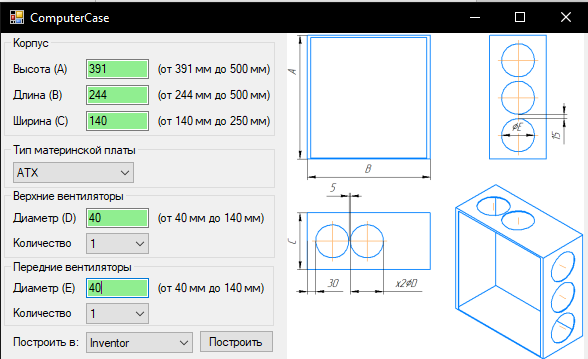
В проекте будет применятся три вида тестирования: функциональное тестирование, модульное тестирование, нагрузочное тестирование.

## 5.1 Функциональное тестирование

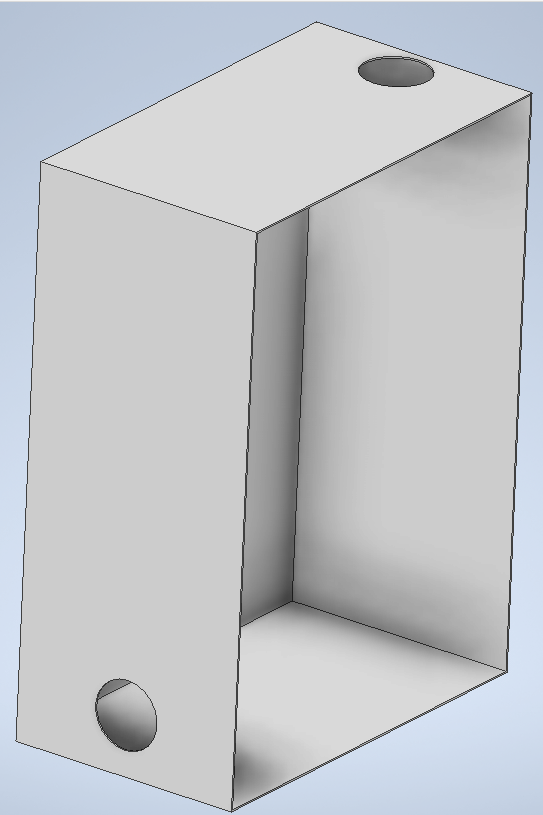
При функциональном тестировании корпуса проверялось соответствие полученной трёхмерной модели с введенными параметрами.

Проведено тестирование с максимальными, минимальными данными.

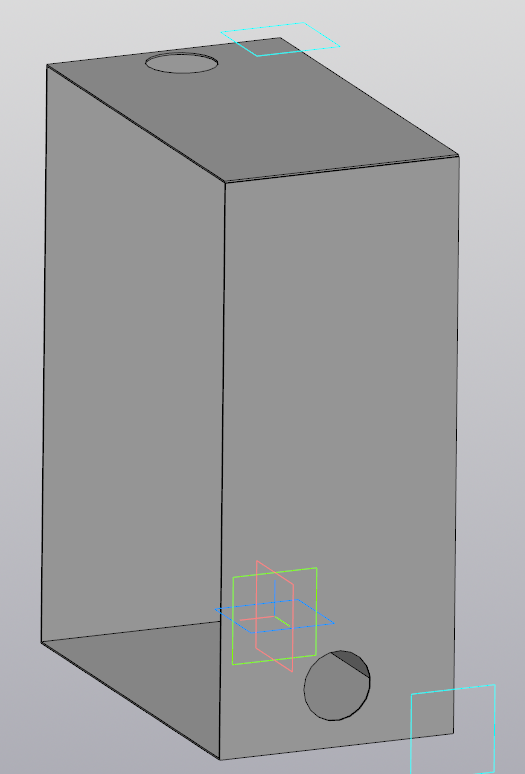
На рисунках 5.1 (а,б,в) представлены пользовательский интерфейс, модель с минимальными данными в САПР «Компас-3D» и в САПР «Inventor», соответственно.



а)



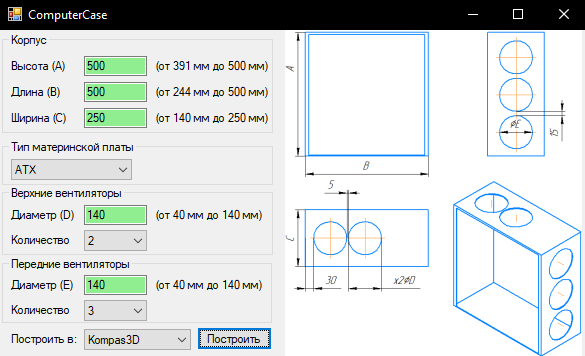
б)



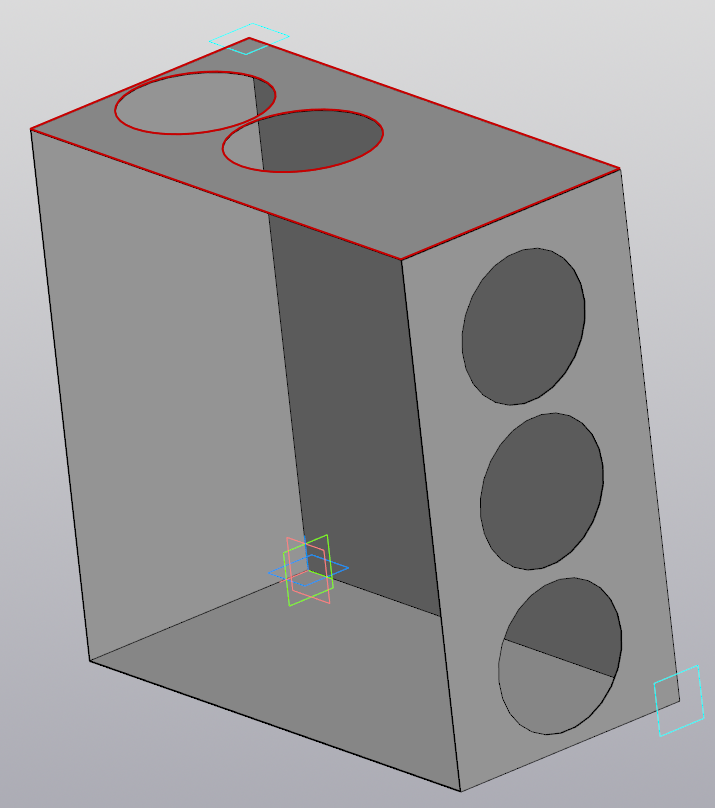
в)

Рисунок 5.1(а,б,в) – Пользовательский интерфейс и модель корпуса с минимальными данными.

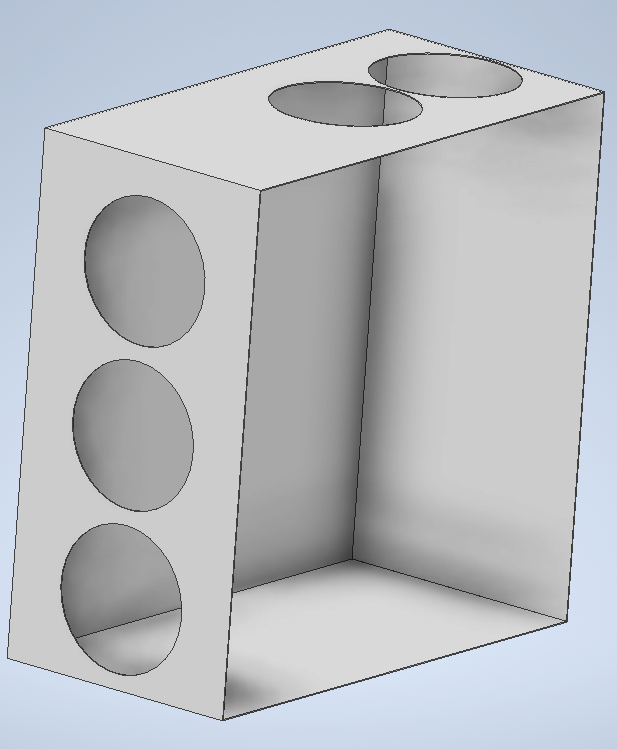
На рисунках5.2(а,б,в) представлены пользовательский интерфейс, модель с минимальными данными в САПР «Компас-3D» и в САПР «Inventor», соответственно.



а)



б)



в)

Рисунок 5.2 (а,б,в) – Пользовательский интерфейс и модель корпуса с максимальными значениями.

## 5.2 Модульное тестирование

Юнит-тестирование (модульное тестирование) — тестирование отдельного элемента изолированно от остальной системы [8]. На основе тестовых сценариев проводилось тестирование работы методов и свойств классов при помощи обозревателя тестов VisualStudio. Проверялись открытые поля и методы, для этого были созданы тестовые классы:

В качестве проверки корректности работы методов и свойств классов при помощи тестового фреймворкаNUnit версия 3.13.2выполнено модульное тестирование, проверялись открытые поля, свойства и методы.

На рисунке 5.5 представлен список тестов.

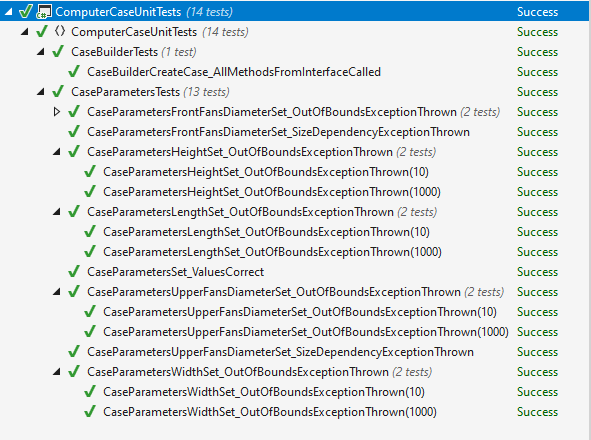


Рисунок 5.5 – список тестов.

Процент покрытия тестами представлен на рисунке 5.6.

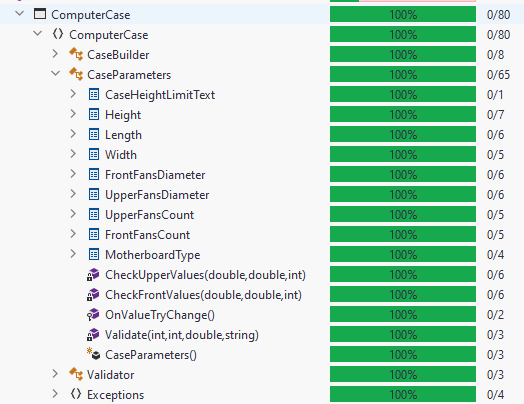


Рисунок 5.6 – Процент покрытия тестами.

Описание тестов класса CaseParameters представлено в таблице 5.1.

Таблица 5.1 – Описание тестов класса CaseParameters.

|  |  |
| --- | --- |
| Название теста | Описание |
| CaseParametersSet\_ValuesCorrect() | Позитивный тест сеттеров и геттеров параметра |
| CaseParametersHeightSet  \_OutOfBoundsExceptionThrown(double) | Негативный тест на сеттер высоты  (Значения больше и меньше допустимых) |
| CaseParametersLengthSet\_ OutOfBoundsExceptionThrown(double) | Негативный тест на сеттер длины  (Значения больше и меньше допустимых) |
| CaseParametersWidthSet\_  OutOfBoundsExceptionThrown(double) | Негативный тест на сеттер ширины (Значения больше и меньше допустимых) |
| CaseParametersFrontFansDiameterSet  \_OutOfBoundsExceptionThrown(double) | Негативный тест на сеттер диаметра отверстий под вентиляторы (Значения больше и меньше допустимых) |

Продолжение таблицы 5.1

|  |  |
| --- | --- |
| Название теста | Описание |
| CaseParametersUpperFansDiameterSet  \_OutOfBoundsExceptionThrown(double) | Негативный тест на сеттер диаметра отверстий под верхние вентиляторы  (Значения больше и меньше допустимых) |
| CaseParametersUpperFansDiameterSet  \_SizeDependencyExceptionThrown() | Негативный тест на ошибку зависимых размеров: длины, ширины корпуса и диаметра отверстий |
| CaseParametersFrontFansDiameterSet  \_SizeDependencyExceptionThrown() | Негативный тест на ошибку зависимых размеров: высоты корпуса и диаметра отверстий |

Описание тестов класса CaseBuilder представлено в таблице 5.2

Таблица 5.2 – Описание тестов класса CaseBuilder.

|  |  |
| --- | --- |
| Название теста | Описание |
| CaseBuilderCreateCase  \_AllMethodsFromInterfaceCalled | Тест, проверяющий вызов методов, необходимых для построения корпуса |

## 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 10x64;

2) Процессор AMD Ryzen 5 3600 6-Core Processor 3.6 GHz;

3) Оперативная память: 16,00 ГБ;

4) Видеоускоритель: 6,00 ГБ.

Для нагрузочного тестирования был задан бесконечный цикл построения детали. Для измерения времени был использован класс Stopwatch. Тестирование заключалось в построении корпуса со стандартными параметрами.

Стандартные параметры модели:

* Высота – 391 мм;
* Длина – 400 мм;
* Ширина – 160 мм;
* Тип материнской платы: ATX;
* Диаметр верхних вентиляторов – 120 мм;
* Количество верхних вентиляторов – 2;
* Диаметр передних вентиляторов – 120 мм;
* Количество передних вентиляторов – 3;

Тестирование будет проходить в обоих САПР.

На графике, изображенном на рисунке 5.6 ось «Х» - количество построенных деталей, ось «Y» - количество потребляемой оперативной памяти. На графике, изображенном на рисунке 6.8 в текущей главе, ось «X» – время в секундах, ось «Y» – количество построенных деталей. На протяжении всех тестов (продолжительностью до сбоя Kompas 3D и Inventor) общая загруженность процессора была в пределах 22 процентов, потребление ОЗУ плагином прямолинейное от 14МБ до 20МБ для тестирования на САПР Inventor. Для САПР Kompas 3D общая загруженность процессора была в пределах 20 процентов, потребление ОЗУ плагином прямолинейное от 14мб до 20мб.

Рисунок 5.6 – График зависимости загруженности памяти от количества деталей.

Рисунок 5.7 – График зависимости времени от количества построенных деталей с параметрами по умолчанию.

Исходя из вышеуказанных графиков на рисунках 5.6 и 5.7, скорость построения в САПР Kompas 3D выше, чем на САПР Inventor. На рисунке 5.6 видно, что до запуска плагина, при запуске Inventor 2022 и Kompas 3D было занято около 4000 МБ оперативной памяти системой и сторонними процессами, которые к самому плагину отношения не имеют. САПР Kompas 3D потребляет больше оперативной памяти, чем Inventor.

# Заключение

В ходе выполнения лабораторных работ были изучены предметная область проектирования, предмет проектирования, аналоги предмета проектирования, API, функциональное и нагрузочное тестирование и на основании полученных данных были спроектированы UML диаграммы классов, разработан плагин для создания 3D моделей «Корпус компьютера» в САПР Inventor 2022 и Kompas 3D, и проведено функциональное и нагрузочное тестирование плагина.

# Список используемых источников

1. Общие сведения о САПР [Электронный ресурс]. – Режим доступа: http://www.hi-edu.ru/e-books/xbook116/01/part-002.html, свободный (дата обращения: 20.03.2022).
2. САПР – Википедия. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Система\_автоматизированного\_проектирования (дата обращения 20.03.2022).
3. Официальный сайт Kompas. [Электронный ресурс] – Режим доступа: https://kompas.ru/kompas-3d/about/ (дата обращения 20.03.2022).
4. Autodesk Inventor— Википедия. [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Autodesk\_Inventor (дата обращения 20.03.2022).
5. Онлайн конфигуратор ПК [Электронный ресурс] – Режим доступа: https://edelws.ru/constructor/ (дата обращения 20.03.2022).
6. 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 20.03.2022).
7. Мартин Ф. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. Изд: Символ-Плюс,2011, с.192 (3-е издание);
8. Виды тестирования [электронный ресурс]. – режим доступа: <https://qa-academy.by/qaacademy/news/klassifikaciya-vidov-testirovaniya/> (дата обращения: 20.03.2022).