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)

Проект системы для приложения «Плечики для одежды»

«По дисциплине «Основы разработки САПР»»

Студент гр. 588-3

\_\_\_\_\_\_\_\_\_\_\_\_ Погорелов С.С.

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

Руководитель:

к.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_\_А.А. Калентьев

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

**Реферат**

Лабораторная работа 27 с., 15 рис., 10 источников.

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

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

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

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2016.

Оглавление

[1. Описание САПР 4](#_Toc93668133)

[1.1 Описание Компас-3D 4](#_Toc93668134)

[1.2 Описание API 5](#_Toc93668135)

[1.3 Обзор аналогов 9](#_Toc93668136)

[2 Описание проекта проектирования 11](#_Toc93668137)

[2.1 Выбор инструментов и средств реализации 12](#_Toc93668138)

[2.2 Назначение плагина 13](#_Toc93668139)

[3 Проект программы 14](#_Toc93668140)

[3.1 Диаграмма классов 14](#_Toc93668141)

[3.2 Макет пользовательского интерфейса 16](#_Toc93668142)

[4. Тестирование плагина 18](#_Toc93668143)

[4.1 Функциональное тестирование 18](#_Toc93668144)

[4.2 Модульное тестирование 19](#_Toc93668145)

[4.3 Нагрузочное тестирование 21](#_Toc93668146)

[Заключение 23](#_Toc93668147)

[Список источников 24](#_Toc93668148)

[Приложение А 25](#_Toc93668149)

# **Описание САПР**

# **1.1 Описание Компас-3D**

«Компас» — семейство систем автоматизированного проектирования, универсальная система автоматизированного проектирования, позволяющая в оперативном режиме выпускать чертежи изделий, схемы, спецификации, таблицы, инструкции, расчетно-пояснительные записки, технические условия, текстовые и прочие документы. Изначально система ориентирована на оформления документации в соответствии с ЕСКД, ЕСТД, СПДС и международными стандартами, но этим возможности системы не ограничиваются.

Система «Компас-3D» предназначена для создания трёхмерных ассоциативных моделей отдельных деталей (в том числе, деталей, формируемых из листового материала путём его гибки) и сборочных единиц, содержащих как оригинальные, так и стандартизованные конструктивные элементы. Параметрическая технология позволяет быстро получать модели типовых изделий на основе проектированного ранее прототипа. Многочисленные сервисные функции облегчают решение вспомогательных задач проектирования и обслуживания производства.

Система «Компас-3D» включает следующие компоненты: система трёхмерного твердотельного моделирования, универсальная система автоматизированного проектирования «Компас-График» и модуль формирования спецификаций. Ключевой особенностью «Компас-3D» является использование собственного математического ядра и параметрических технологий. [1]

## **1.2 Описание API**

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

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

Главным интерфейсом API системы КОМПАС-3D является KompasObject. Получить указатель на этот интерфейс 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 [2].

Ниже в таблицах 1.1-1.5 представлены основные свойства и методы интерфейсов.

Таблица 1.1. – Методы интерфейса KompasObject.

|  |  |  |
| --- | --- | --- |
| Метод | Возвращаемое значение | Описание |
| Document3D() | Указатель на интерфейс документа трёхмерной модели ksDocument3D | Даёт возможность получить указатель на интерфейс трёхмерного документа (детали или сборки) |
| Visible |  | Свойство видимости приложения |
| Quit |  | Метод для завершения программы Kompas-3D |
| ActivateControllerAPI |  | Метод для активации контроллера API |

Продолжение таблицы 1.1

|  |  |  |
| --- | --- | --- |
| Метод | Возвращаемое значение | Описание |
| ksDocument2D |  | Интерфейс событий графического документа, события интерфейса позволяют контролировать состояние документа. |

Таблица 1.2 — Методы интерфейса IPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| GetDefaultEntity  (short objType) | |  |  | | --- | --- | | objType | - тип объекта. | | |  | | --- | | Указатель на интерфейс [ksEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm) или [IEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm). | | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart(int type) | |  |  | | --- | --- | | type | - тип компонента. | | указатель на интерфейс компонента [ksPart](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksPart.htm) или [IPart](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksPart.htm). | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity(short objType) | |  |  | | --- | --- | | ob | jType- [тип объекта](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/Obj3dType_NewEntil_Part.htm). | | указатель на интерфейс [ksEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm) или [IEntity](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntity.htm). | Создать новый интерфейс объекта и получить указатель на него |

Таблица 1.3 — Методы интерфейса ksDocument3D.

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| Create (bool invisible, bool \_typeDoc) | invisible – признак режима редактирования документа  (TRUE – невидимый режим,  FALSE – видимый режим),  typeDoc – тип документа  (TRUE – деталь,  FALSE – сборка). | TRUE – в случае успешного завершения. | Дает возможность создать пустой документ (деталь или сборку) |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. |  | Получить указатель на интерфейс компонента в соответствии с заданным типом |

Таблица 1.4 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| long ksLineSeg (double x1, double y1, double x2, double y2, long style) | x1, y1 - координаты первой точки отрезка, x2, y2 - координаты второй точки отрезка, style - стиль линии. | указатель на отрезок - в случае удачного завершения, 0 - в случае неудачи. | Метод для создания отрезка. |

Таблица 1.5 —Методы интерфейса [ksEntity](mk:@MSITStore:C:\Program%20Files\ASCON\KOMPAS-3D%20v18%20Study\SDK\SDK.chm::/ksEntity_props.htm)

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| ksBossExtrusionDefinition(BOOL forward, short type, double depth, double draftValue, BOOL draftOutward); | Forward- - направление выдавливания: TRUE - прямое направление, FALSE - обратное направление, type - тип выдавливания, depth - глубина выдавливания, draftValue - угол уклона, draftOutward - направление уклона: FALSE - уклон наружу, TRUE - уклон внутрь. | TRUE – в случае успешного завершения. | Метод выдавливает эскиз в одном направлении |
| ksCutExtrusionDefinition (BOOL forward, short type, double depth, double draftValue, BOOL draftOutward); | Forward- - направление выдавливания: TRUE - прямое направление, FALSE - обратное направление, type - тип выдавливания, depth - глубина выдавливания, draftValue - угол уклона, draftOutward - направление уклона: FALSE - уклон наружу, TRUE - уклон внутрь. | TRUE – в случае успешного завершения. | Метод вырезания выдавливанием эскиз в одном направлении |

**1.3 Обзор аналогов**

T-FLEX Мебель – специализированное решение для трехмерного проектирования мебели любого уровня сложности. Система позволяет автоматизировать все этапы работы с изделием - от оформления заказа до проектирования и изготовления мебели, учитывая особенности как серийных, так и позаказных типов производств.

Система состоит из нескольких специализированных редакций: T-FLEX Мебель.Салон, T-FLEX Мебель.Конструктор, T-FLEX Мебель.Эксперт. Каждая из редакций может работать как в составе единого комплекса, так и отдельно от него.

В нашем случае рассмотрим редакцию T-FLEX Мебель.Конструктор.

Редакция «Конструктор» – Редакция предназначена для проектирования мебели по индивидуальным проектам. Редакция позволяет получить полный комплект проектной документации, рассчитать стоимость изделия и сформировать управляющие программы для изготовления деталей на станках с ЧПУ.

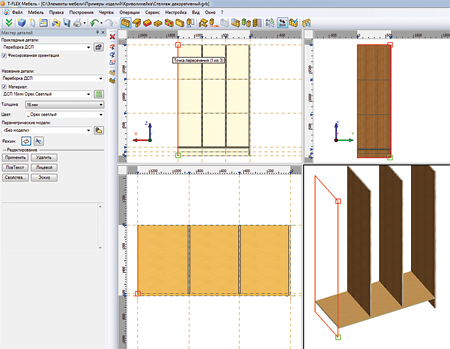


Рисунок 1.1 – Редакция T-FLEX Мебель.Конструктор

Особенности системы

T-FLEX Мебель базируется на мощном трёхмерном ядре, которое хорошо зарекомендовало себя в других областях проектирования, таких как: машиностроение, авто-, авиа- и приборостроение, архитектура и проектирование строительных металлоконструкций. Использование геометрических и параметрических свойств ядра определяет главную отличительную черту системы – отсутствие ограничений на сложность создаваемой геометрии и логику перестроения моделей [3].

**2 Описание проекта проектирования**

Вешалка (также плечики, тремпель) — приспособление для хранения одежды в подвешенном состоянии. Одежда на ней вешается на крючки либо на плечики (которые тоже называются вешалкой). Плечики используются для демонстрации одежды в магазинах и торговых залах, для этого плечики с одеждой развешивают на отдельно стоящие вешала, либо при помощи кронштейнов на эконома панель Одежда, хранящаяся на плечиках, не мнётся[4].

Изображение моделируемого объекта на рисунке 2.1

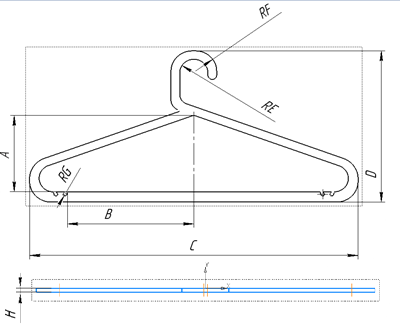


Рисунок 2.1 – Плечики

Плагин должен уметь принимать такие параметры как:

1. Высота плечиков внутреннего отдела = A = (D–10)/2 мм;
2. Длина от центра до выемок = B= C/4 мм;
3. Длина плечиков = 390 мм ≤С≤ 470 мм;
4. Высота плечиков = 200 мм ≤D≤ 230мм;
5. Радиус внутреннего отдела = 15 мм ≤ RE ≤ 20 мм;
6. Радиус наружного отдела = RF = RE + 10 мм;
7. Радиус выемки = 3 ≤ RG ≤ 4 мм;
8. Ширина плечиков = 4 мм ≤ H ≤ 6 мм.

Плагин имеет следующие зависимости:

1. Длина от центра до выемок (B) должна быть равна длина плечиков (C), поделенная на четыре;

2. Высота плечиков внутреннего отдела (A) должна быть меньше высоты плечиков (D) по зависимости (D-10)/2;

3. Радиус наружного отдела (RF) должен быть строго больше на 10 мм.

Также по ходу проектирования плагина добавлена возможность создания крепления для брюк. Вид плечиков с креплениями для брюк представлен на рисунке 2.2.

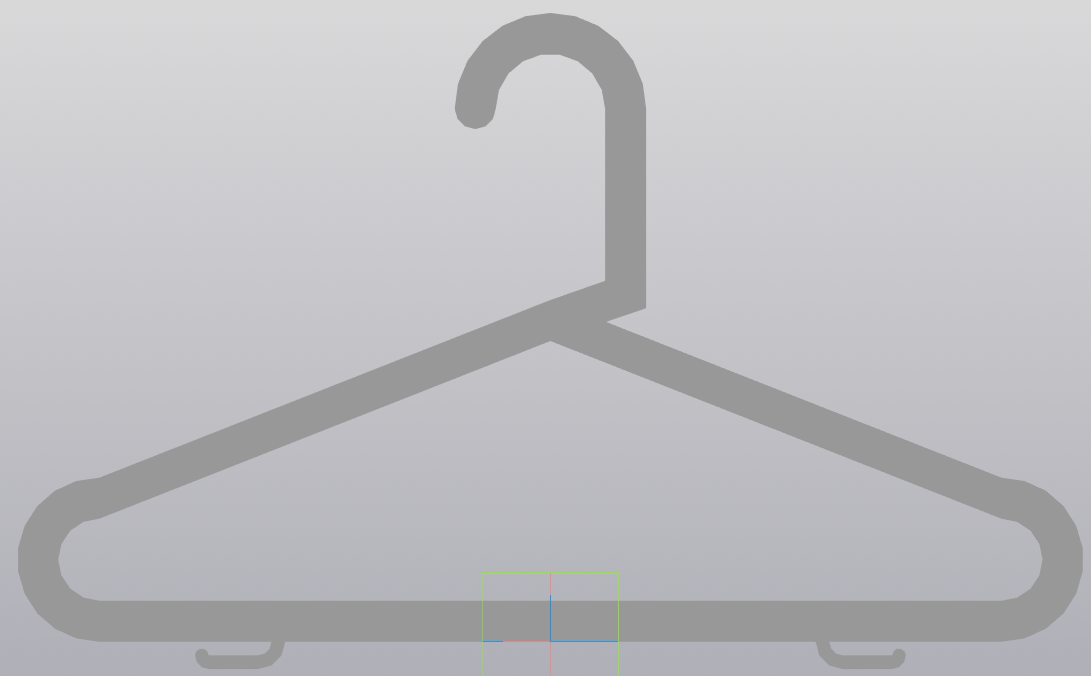


Рисунок 2.2 – Плечики с креплениями для брюк

## **2.1 Выбор инструментов и средств реализации**

На основе требований к техническому заданию программа выполнена на языке программирования C# в среде Microsoft Visual Studio 2019 с использованием .NET Framework 4.7.2 [8], библиотека «Kompas6API5» [9] 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].

## **2.2 Назначение плагина**

Назначение разрабатываемого плагина обусловлено быстрым моделированием плечиков, с разными параметрами длины, высоты и радиусов. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

**3 Проект программы**

**3.1 Диаграмма классов**

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними [5]. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами  
(рисунок 3.1).

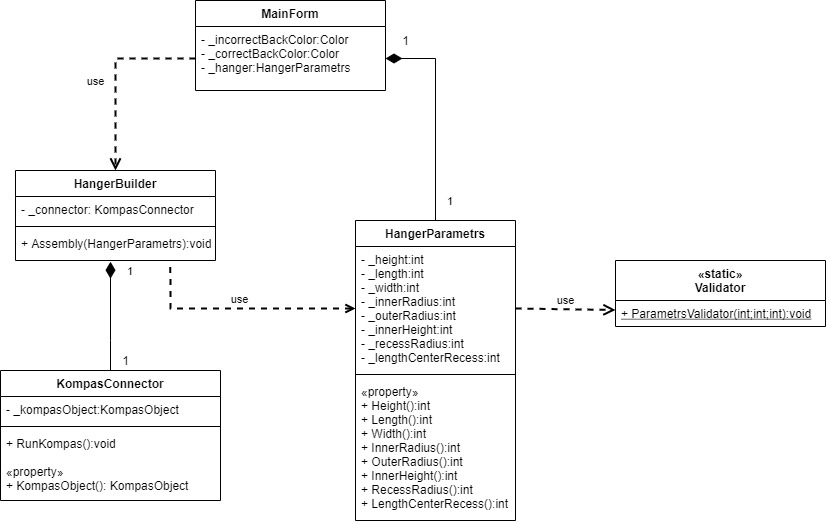


Рисунок 3.1 – Изначальная диаграмма классов

1) MainForm – класс диалогового окна, который обеспечивает взаимодействие между пользователем и программой;

2) HangerParametrs − класс, хранящий в себе все параметры 3D-модели;

3) Validator − класс, проверяющий правильность введенных параметров;

4) KompasConnector – класс для работы с API КОМПАС 3D;

5) HangerBuilder – класс, осуществляющий вызов методов API, необходимых для постройки 3D-модели.

Диаграмма классов, после добавления дополнительной функциональности изображена на рисунке 3.2.

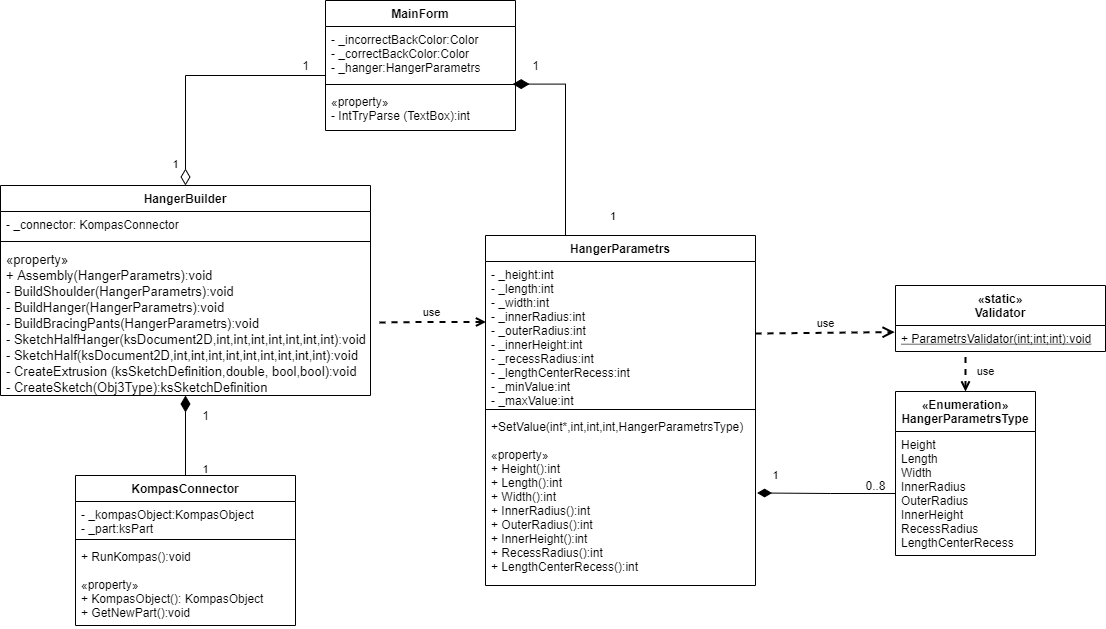


Рисунок 3.2 – Диаграмма классов с дополнительной функциональностью

Опишем некоторые изменения, возникшие после добавления дополнительной функциональности и переработки кода в ходе разработки плагина.

Были добавлены свойства для построения детали, для чего потребовалось создание нескольких эскизов и соответствующих методов для них. Так же был изменен тип связи между классами MainForm и HangerBuilder с использования на агрегирование. В классе MainForm был добавлен метод IntTryParse для перевода типа string в тип int. Добавлено перечисление, которое упрощает использование методов в построение детали.

**3.2 Макет пользовательского интерфейса**

Макет пользовательского интерфейса представляет собой форму для ввода параметров. При запуске программы в полях для ввода параметров отсутствуют значения. Пользователь может менять данные параметры (рисунок 3.2).

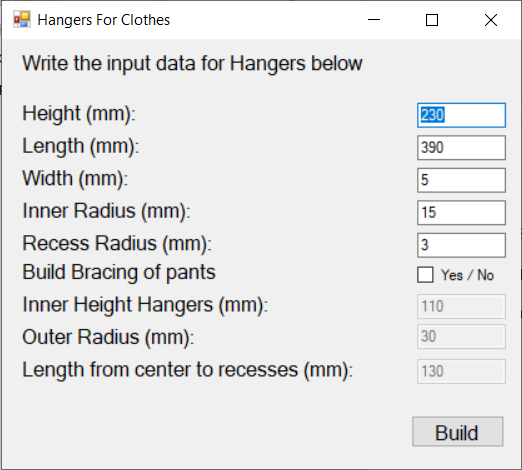


Рисунок 3.2 – Макет пользовательского интерфейса

Поле, где было введено некорректное значение изменит цвет на светло-красный (рисунок 3.3).

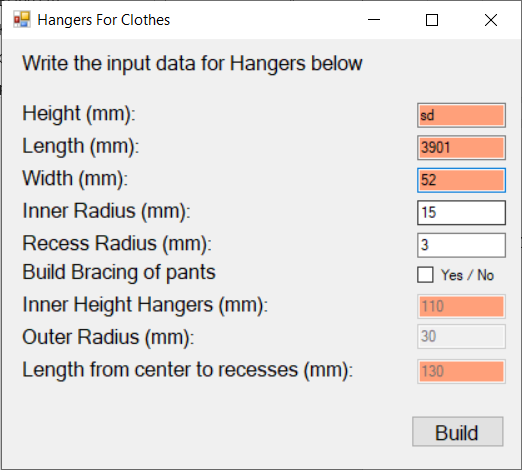


Рисунок 3.3 – Поля с некорректными параметрами

Поля Наружный радиус, Внутренняя высота плечиков и Длина от центра до выемок, рассчитываются по зависимостям, поэтому данные поля нельзя заполнить, но при этом будет выведен их размер (рисунок 3.4).

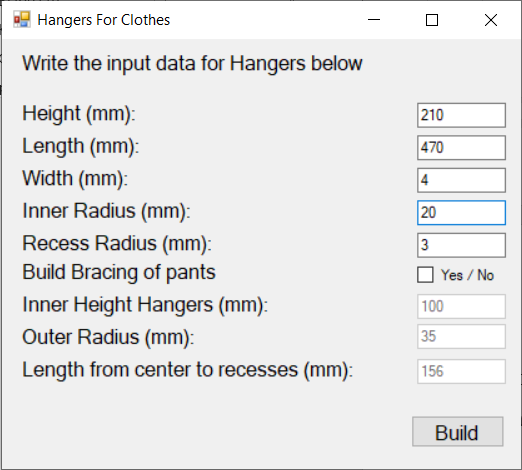


Рисунок 3.4 – Вывод зависимых полей

# **4. Тестирование плагина**

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

## **4.1 Функциональное тестирование**

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

Проведено тестирование максимальных и минимальных параметров модели.

На рисунке 4.1 представлена модель с минимально введенными параметрами.

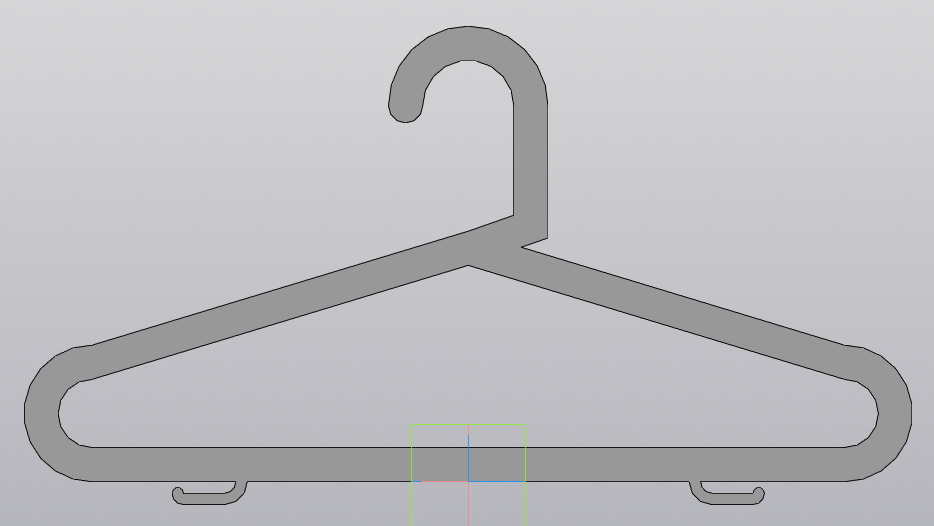


Рисунок 4.1 – Модель плечиков с минимальными введенными параметрами

На рисунке 4.2 представлена модель плечиков с максимальными введенными параметрами.

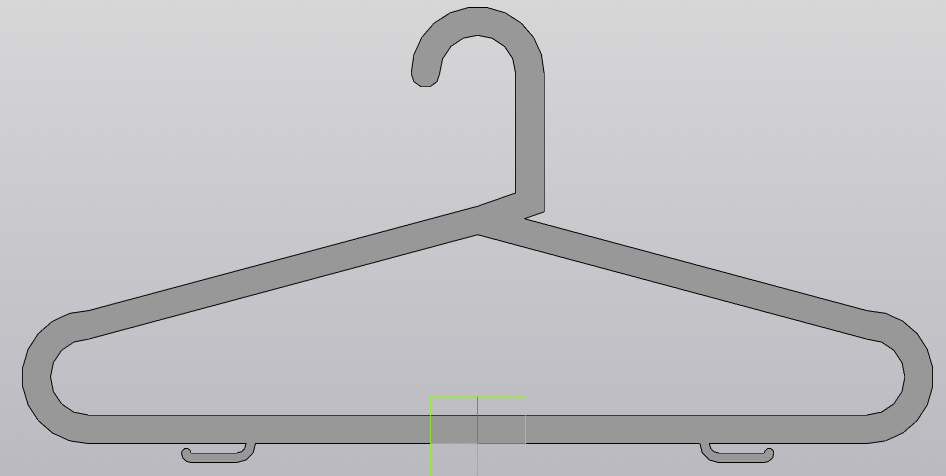


Рисунок 4.2 – Модель плечиков с максимальными введенными параметрами

На рисунке 4.3 представлена модель плечиков без крепления для брюк и средними введенными параметрами

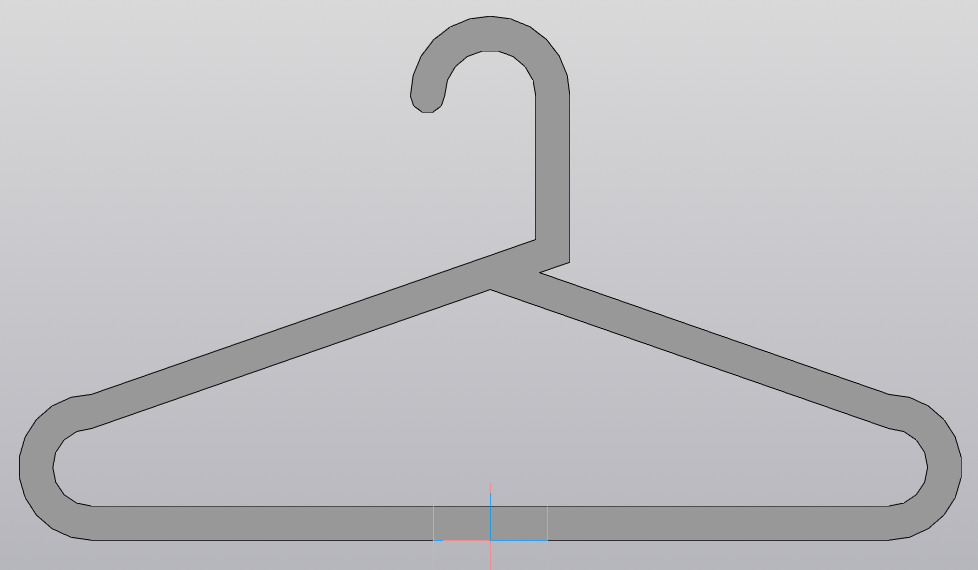


Рисунок 4.3 – Модель плечиков без крепления для брюк

## **4.2 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: HangerParametrs, ValidatorHangerParametrs. На рисунке 4.4 представлена информация о модульном тестировании программы. Покрытие модели тестами составило сто процентов, что показано на рисунке 4.5.

На основе таблицы приведенных в приложении А тестовых сценариев (таблица А.1), проводилось тестирование корректности входных параметров 3D-модели.

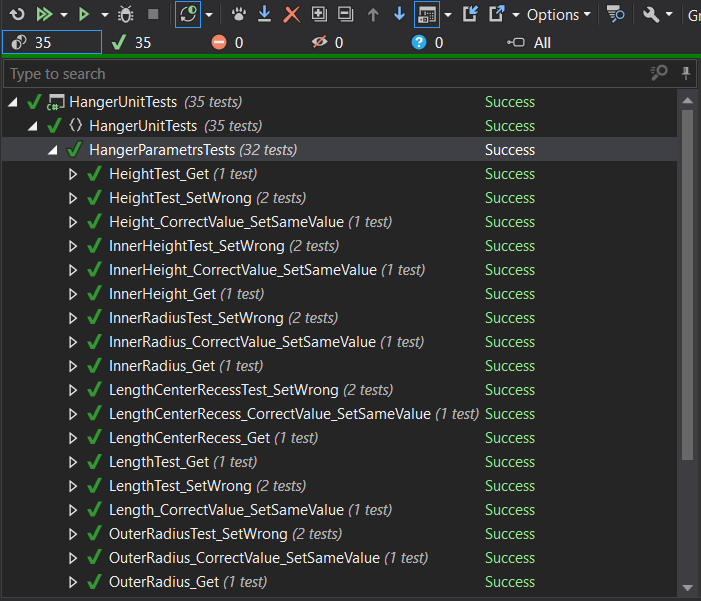


Рисунок 4.4 – Модульное тестирование плагина

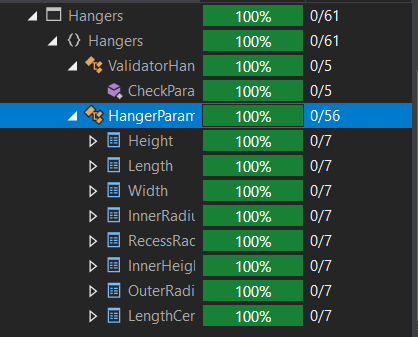


Рисунок 4.5 – Покрытие кода тестами

## **4.3 Нагрузочное тестирование**

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

– ЦП AMD Ryzen 3 3200U with Radeon Vega Mobile Gfx 2.60 GHz;

– 8 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 плечиков с параметрами по умолчанию. На рисунке 4.6 представлен график зависимости времени от количества построенных деталей.

Рисунок 7.6 – График зависимости времени построения

от количества построенных плечиков

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

Рисунок 7.7 – График зависимости загруженности памяти

от количества построенных плечиков для одежды

Тестирование длилось три минут, было построено пятьдесят три моделей плечиков со стандартными параметрами, работа плагина завершилась сообщением об ошибки памяти в Visual Studio. Исходя из графика, представленного на рисунке 4.5 можно увидеть, что построения детали в основном занимает менее трех секунд, но после построения сорока седьмой детали, время построения некоторых деталей стало возрастать, данную тенденцию можно соотнести с самым большим изменением занимаемой памяти показанное на рисунке 4.7. Начиная с сороковой детали потребляемая ОЗУ не меняется, это можно объяснить тем что компьютер мог начать использовать файл 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].

# **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 плечиков по заданным параметрам.

Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирование на платформе Windows 10.

# **Список источников**

1. Компас 3D [Электронный ресурс]. – Режим доступа: https://kompas.ru/kompas-3d/about/ Дата обращения (12.01.2022)

2. Кидрук Максим. КОМПАС-3D V10 на 100% / М. Кидрук. – СПб.: Питер, 2009 – 560 с.

3. Профессиональный софт для работы с мебелью. [Электронный ресурс] — Режим доступа: https://gidmaster.info/soft-T-FLEX-text\_20161209000002 (дата обращения 12.01.2022)

4. Вешалка(плечики) — Википедия. [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Вешалка (дата обращения 12.01.2022)

5. М. Фаулер. UML. Основы, 3-е издание. — Пер. с англ. — СПб: символ-Плюс, 2004 – 192 с.

6. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 12.01.2022).

7. Введение в Windows Forms [Электронный ресурс]. – URL: https://metanit.com/sharp/windowsforms/1.1.php (дата обращения: 12.01.2022).

8. Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 12.01.2022).

9. КОМПАС-3D для разработчиков. [Электронный ресурс]. – URL: https://kompas.ru/solutions/developers/ (дата обращения: 12.01.2022).

10.Виртуальная память — Википедия. [Электронный ресурс] — Режим доступа: https://ru.wikipedia.org/wiki/Виртуальная память (дата обращения 12.01.2022)

**Приложение А**

Таблица А.1 – Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| TestValidator\_InvalidValue(int invalidValue,int minValue, int maxValue, HangerParametersType type) | 100, 1, 50, HangerParametersType.Height | Value over than range |
| 10, 15, 20, HangerParametersType.Height | Value over than range |
| TestValidator\_ValidValue(int validValue,int minValue, int maxValue, HangerParametersType type) | 10, 1, 20, HangerParametersType.Height | Value in the range |
| TestGetErrors\_HaveErrorsValue(int wrongValue,HangerParametersType type) | 1, HangerParametersType.InnerRadius | InnerRadius value less than range |
| 700, HangerParametersType.InnerRadius | InnerRadius value over than range |
| 1, HangerParametersType.Height | Height value less than range |
| 700, HangerParametersType.Height | Height value over than range |
| 1, HangerParametersType.Length, | Length value less than range |
| 700, HangerParametersType.Length, | Length value over than range |
| 1, HangerParametersType.Width, | Width value less than range |
| 700, HangerParametersType.Width, | Width value over than range |
| 1, HangerParametersType.InnerHeight, | Inner Height value less than range |
| 700, HangerParametersType.InnerHeight, | Inner Height value over than range |
| 2, HangerParametersType.OuterRadius, | Outer Radius value less than range |

Продолжение Таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| TestGetErrors\_HaveErrorsValue(int wrongValue,HangerParametersType type) | 700, HangerParametersType.OuterRadius, | Outer Radius value over than range |
| 1, HangerParametersType.RecessRadius, | Recess Radius value less than range |
| 700, HangerParametersType.RecessRadius, | Recess Radius value over than range |
| 1, HangerParametersType.LengthCenterRecess, | Length Center Recess value less than range |
| 700, HangerParametersType.LengthCenterRecess, | Length Center Recess value over than range |
| HeightTest\_Get() | 230 | Positive test getter Height |
| LengthTest\_Get | 390 | Positive test getter Length |
| WidthTest\_Get | 5 | Positive test getter Width |
| InnerHeight\_Get | 110 | Positive test getter InnerHeight |
| InnerRadius\_Get | 15 | Positive test getter InnerRadius |
| LengthCenterRecess\_Get | 390 | Positive test getter LengthCenterRecess |
| OuterRadius\_Get | 15 | Positive test getter OuterRadius |
| RecessRadius\_Get | 3 | Positive test getter RecessRadius |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| TestGetErrors\_NoErrorsValue | HangersParametrs | Отправить значения по умолчанию |
| Height\_CorrectValue\_SetSameValue(int wrongHeight) | 230 | Assigning the correct Height |
| Length\_CorrectValue\_SetSameValue | 390 | Assigning the correct Length |
| OuterRadius\_CorrectValue\_SetSameValue | 15 | Assigning the correct OuterRadius |
| InnerHeight\_CorrectValue\_SetSameValue | 110 | Assigning the correct InnerHeight |
| LengthCenterRecess\_CorrectValue\_SetSameValue | 390 | Assigning the correct LengthCenterRecess |
| InnerRadius\_CorrectValue\_SetSameValue | 15 | Assigning the correct InnerRadius |
| RecessRadius\_CorrectValue\_SetSameValue(int value) | 3 | Assigning the correct RecessRadius |
| Width\_CorrectValue\_SetSameValue(int value) | 5 | Assigning the correct Width |