Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**РАЗРАБОТКА ПЛАГИНА «КОРПУС ПИШУЩЕЙ РУЧКИ» ДЛЯ «КОМПАС-3D v18.1»**

Пояснительная записка к лабораторному проекту

по дисциплине «Основы разработки САПР»

«Построение корпуса пишущей ручки в системе КОМПАС 3D v18»

Выполнил:

студент гр.587-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Бондарь В. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Руководитель:

к.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Калентьев А. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Оглавление

[Введение 3](#_Toc70374501)

[1 Описание САПР 4](#_Toc70374502)

[1.1 Описание программы 4](#_Toc70374503)

[1.2 Описание API 4](#_Toc70374504)

[1.3 Обзор аналогов 11](#_Toc70374505)

[1.3.1 Плагин PDF для САПР КОМПАС-3D 11](#_Toc70374506)

[2 Описание предмета проектирования 13](#_Toc70374507)

[3 Проект программы 15](#_Toc70374508)

[3.1 Описание технических и функциональных аспектов проекта 15](#_Toc70374509)

[3.3 Диаграмма классов 15](#_Toc70374510)

[3.3 Макет пользовательского интерфейса 19](#_Toc70374511)

[4 Тестирование 22](#_Toc70374512)

[4.1 Функциональное тестирование 22](#_Toc70374513)

[4.2 Модульное тестирование 26](#_Toc70374514)

[4.3 Нагрузочное тестирование 28](#_Toc70374515)

[Заключение 31](#_Toc70374516)

[Список литературы 32](#_Toc70374517)

[Приложение А 33](#_Toc70374518)

# Введение

В настоящее время проектирование в своем понимании представляет собой автоматизированный процесс и, в некотором роде, программно-аппаратный. Проектировщику, который занимается разработкой сложного механизма, или устройства, требующего больших расчетов, математических вычислений при построении модели и высокой точности, подходят системы автоматизации проектных решений — САПР [1].

САПР позволяют уменьшить финансовые затраты на разработку макета (модели) проекта (объекта), а также сократить время, которое тратит проектировщик на создание модели объекта и составление проектной документации.

В каждой крупной САПР есть свой набор средств для разработки, которые предоставляются с целью дать возможность разработчикам расширить функционал данной системы под свои конкретные нужды. Данным средством является API — программный интерфейс приложения [2]. Это набор готовых средств: классов, процедур, функций, структур, констант и т.д. API позволяет определить функциональность, которую предоставляет приложение, при этом абстрагируясь от того, как она реализована.

Расширение функционала, по большей части, подразумевает разработку плагина или библиотеки на основе предоставленного API. Плагин — независимо компилируемый программный модуль, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей [3].

В качестве системы, которая предоставляет API и для которой стоит задача разработать плагин, была выбрана САПР «КОМПАС-3D» версии 18.1.

# 1 Описание САПР

# Описание программы

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.

# 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 [4].

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

Таблица 1.1 – Методы и свойства интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| Document3D() |  | указатель на интерфейс документа трехмерной модели ksDocument3D | Получить указатель на интерфейс документа трехмерной модели |
| ActiveDocument3D() |  | указатель на интерфейс до­кумента трехмерной модели ksDocument3D | Получить указатель на интерфейс текущего документа трехмерной модели |
| GetParamStruct() | structType - тип интерфейса параметров | указатель на интерфейс указанного ти­па из StructType2D | Получить указатель на интерфейс структуры параметров объекта нужного типа |
| Visible |  | bool | Видимость приложения |

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

Таблица 1.2 – Методы интерфейса ksEntity

|  |  |  |
| --- | --- | --- |
| Название | Тип возвращаемых данных | Описание |
| Create() | TRUE - в случае успешного завершения. | Создать объект в моделе |
| GetDefinition() | Указатель на интерфейс IUnknown. | Получить указатель на интерфейс параметров объектов и элементов |

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

Таблица 1.3 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| ksRegularPolygon() | param - указатель на интерфейс параметров многоугольника ksRectangleParam,  centre - признак построения обозначения центра. | Указатель на многоугольник. | Создать многоугольник |
| ksCircle() | xc, yc - координаты центра окружности,  rad - радиус окружности,  style - стиль линии. | Указатель на окружность - в случае удачного завершения,  0 - в случае неудачи | Создать окружность |
| ksLineSeg() | x1, y1 – координаты первой точки отрезка,  x2, y2 – координаты второй точки отрезка,  style - стиль линии. | Указатель на окружность - в случае удачного завершения,  0 - в случае неудачи | Создать отрезок. |

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

Таблица 1.4 – Методы интерфейса ksDocument3D

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| Create() | invisible-признак режима редактирования документа (TRUE-невидимый режим, FALSE-видимый режим),  typeDoc-тип документа (TRUE-деталь, FALSE-сборка). | TRUE - в случае успешного завершения. | Создать документ-модель (деталь или сборку) |
| GetPart() | Type - тип компонента из перечисления | Указатель на интерфейс компонента ksPart или IPart | 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 | Тип входных параметров | Тип возвращаемых данных | Описание |
| GetDefaultEntity() | objType - тип объекта. | Указатель на интерфейс ksEntity или IEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart() | type - тип компонента из перечисления | Указатель на интерфейс компонента ksPart или IPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |

Продолжение таблицы 1.5

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| NewEntity() | objType - тип объекта. | Указатель на интерфейс ksEntity или IEntity. | Создать новый интерфейс объекта и получить указатель на него |

В таблице 1.6 представлены методы интерфейса ksSketchDefinition, необходимые для разработки плагина.

Таблица 1.6 – Свойства и методы интерфейса ksSketchDefinition.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| SetPlane() | Указатель на интерфейс базовой плоскости эскиза ksEntity или IEntity. | TRUE - в случае успешного завершения. | Изменить базовую плоскость эскиза |
| BeginEdit |  | Указатель на интерфейс эскиза ksDocument2D | Включить режим редактирования эскиза (ksDocument2D) |
| EndEdit |  | TRUE - в случае успешного завершения | Выйти из режима редактирования эскиза |

В таблице 1.7 представлены методы интерфейса ksBaseExtrusionDefinition, необходимые для разработки плагина.

Таблица 1.7 – Свойства и методы интерфейса ksBaseExtrusionDefinition.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| SetSideParam | 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,    FALSE - в случае неудачи. | Установить параметры выдавливания в одном направлении |
| SetSketch | sketch - указатель на интерфейс эскиза ksEntity или IEntity. | TRUE - в случае успешного завершения,  FALSE - в случае неудачи. | Задать указатель на интерфейс эскиза элемента |

В таблице 1.8 представлены методы интерфейса ksCutExtrusionDefinition, необходимые для разработки плагина.

Таблица 1.8 – Свойства и методы интерфейса ksCutExtrusionDefinition.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| SetSideParam | 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,  FALSE - в случае неудачи. | Установить параметры выдавливания в одном направлении |
| SetSketch | sketch - указатель на интерфейс эскиза ksEntity или IEntity. | TRUE - в случае успешного завершения,  FALSE - в случае неудачи. | Задать указатель на интерфейс эскиза элемента |

В таблице 1.9 представлены методы интерфейса ksPlaneOffsetDefinition, необходимые для разработки плагина.

Таблица 1.9 – Свойства и методы интерфейса ksPlaneOffsetDefinition.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| SetPlane | plane - указатель на интерфейс плоскости ksEntity или IEntity. | TRUE - в случае успешного завершения. | Изменить указатель на интерфейс базовой плоскости |

В таблице 1.10 представлены типы объектов документа-модели, необходимые для разработки плагина.

Таблица 1.10 – Некоторые типы объектов документа-модели

|  |  |  |
| --- | --- | --- |
| Идентификатор объекта | Название объекта | Интерфейс параметров |
| o3d\_planeXOZ | Плоскость XOZ | ksPlaneParam |
| o3d\_planeXOY | Плоскость XOY | ksPlaneParam |
| o3d\_planeOffset | Смещённая плоскость | ksPlaneOffsetDefinition |

# Обзор аналогов

# 1.3.1 Плагин PDF для САПР КОМПАС-3D

Данный плагин позволяет производить экспорт моделей и сборок из КОМПАС-3D в формат PDF формат [5]. Основной особенностью является возможность интерактивного взаимодействия пользователя с сохраненной 3D сценой внутри PDF файла. Например, пользователь может вращать, масштабировать, передвигать детали и сборки внутри 3D PDF файла. Также доступно создание анимации сборки и разборки изделий. Это полезно для подготовки интерактивных сборочных инструкций, создания маркетинговых материалов, презентаций, а также для налаживания взаимодействия между проектировщиками и заказчиками. В подобных ситуациях традиционным подходом являлся экспорт сборки или детали КОМПАС-3D в промежуточный формат и дальнейшее сохранение в формат 3D PDF. Используемый подход в плагине исключает использование промежуточных файлов для осуществления 3D преобразования, что существенно повышает качество выходной 3D модели в формате PDF. Ключевые возможности:

* сохранение деталей и сборок в формате 3D PDF для интерактивного просмотра при помощи бесплатной программы Adobe Reader;
* создание анимаций, имитирующих естественный порядок сборки и разборки создание имитации анимации гибки листовых тел;
* вставка в существующие PDF документы, содержащие основной текст, фоновые картинки, таблицы спецификаций, эмблемы, логотипы;
* пакетный режим для поочередной конвертации всех файлов.

# 2 Описание предмета проектирования

Предметом проектирования является корпус пишущей ручки [6].

Основной корпус представляет собой центральную часть ручки. Именно он обеспечивает контакт с пальцами для использования данного канцелярского прибора по назначению. Может изготавливаться из разноцветного пластика или металла.

Верх корпуса располагается между резинкой и клипом ручки. В основном, соответствует цвету и материалу основного корпуса;

Низ корпуса – часть, в которой находится отверстие для стержня. Располагается под резинкой;

Параметры корпуса ручки:

* Внешний диаметр D1 (от 10 до 18 мм);
* Диаметр части для резинки D2 (от 7 до 16 мм);
* Внутренний диаметр D3 (от 2 до 7 мм);
* Длина части для резинки L1 (от 15 до 30 мм);
* Длина основной части L2 (от 30 до 70 мм);
* Количество рёбер N (от 4 до 10);

Зависимые параметры:

* Внутренний диаметр D3 < D2 < D1
* Длина части для резинки ;
* D1 – D2 > 2;

Пример модели приведен на рисунках 2.1 – 2.3.

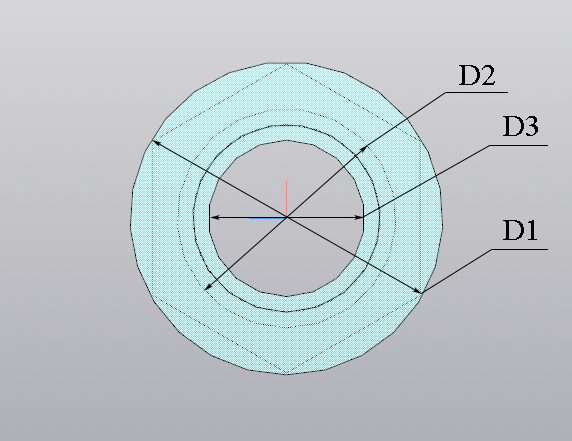


Рисунок 2.1 – Вид модели спереди

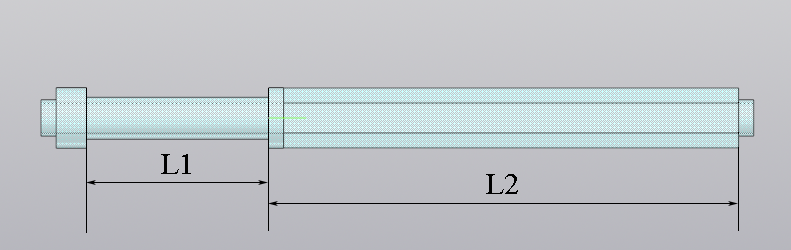


Рисунок 2.2 – Вид модели сбоку

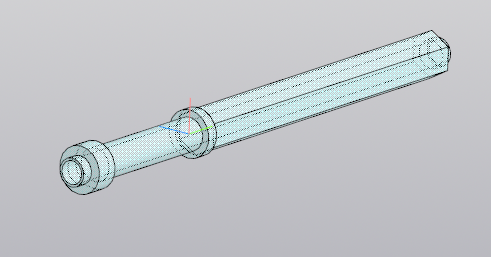


Рисунок 2.3 – 3D вид модели

# 3 Проект программы

# 3.1 Описание технических и функциональных аспектов проекта

Для графического описания абстрактной модели проекта использован стандарт UML (Unified Modelling Language) [7].

UML – унифицированный язык моделирования (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.

При использовании UML была простроена диаграмма классов.

# 3.3 Диаграмма классов

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами [8].

На рисунке 3.1 представлена диаграмма классов.

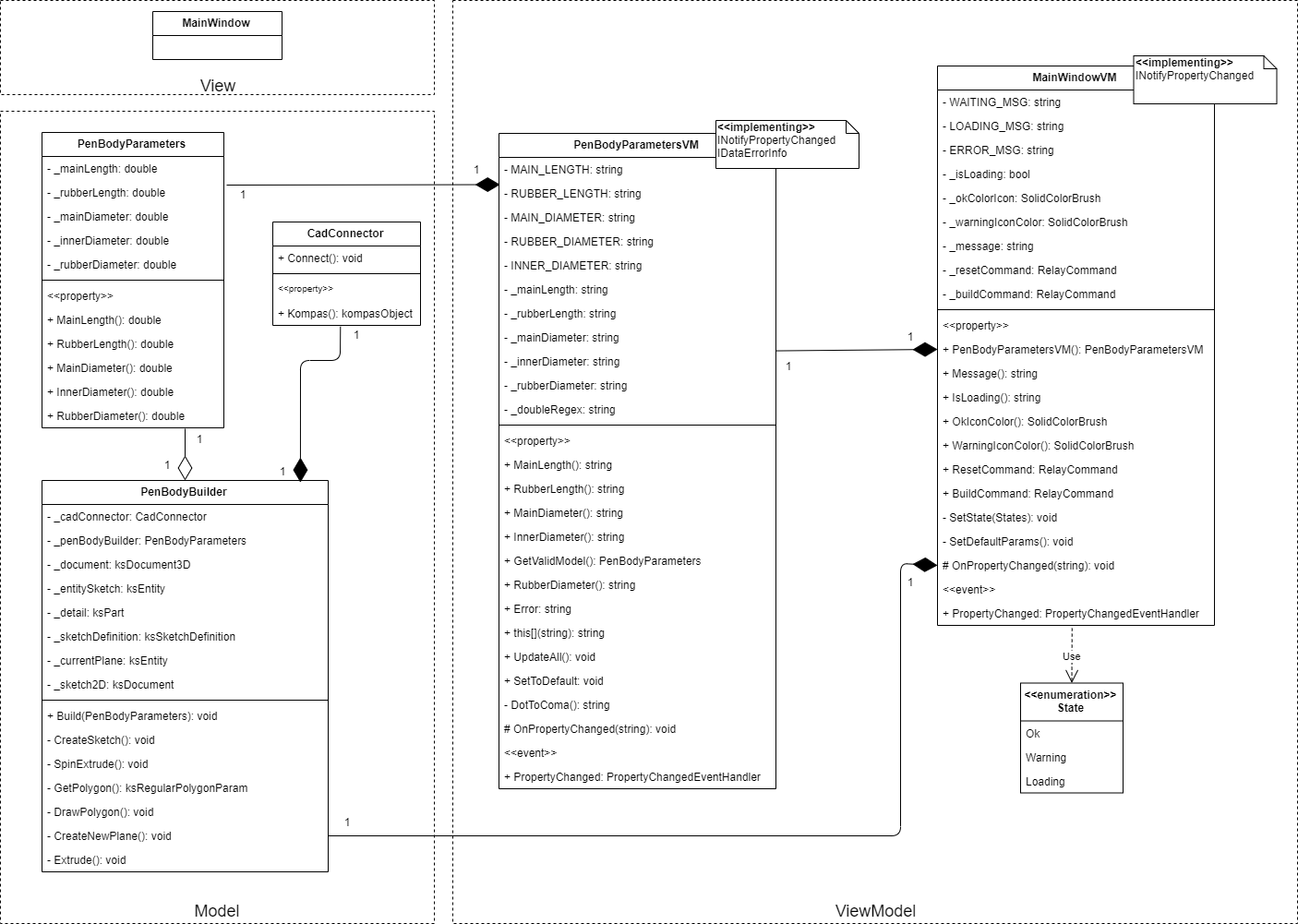


Рисунок 3.1 – Диаграмма классов

Для реализации был выбран шаблон проектирования MVVM [9], реализация которого обеспечена набором следующих классов:

* MainWindowVM – класс управления окном плагина, обеспечивающий взаимодействие приложения с пользователем. Реализует интерфейс INotifyPropertyChanged для уведомления системы об изменении значений каких-либо свойств в реальном времени;
* PenBodyParametersVM – класс, обеспечивающий обработку введенных пользователем данных и вывод сообщения в случае их некорректности, благодаря реализации IDataErrorInfo. Как и MainWindowVM, реализует INotifyPropertyChanged;
* PenBodyParameters – класс, непосредственно описывающий параметры модели для ее построения.
* CadConnector – класс для работы с API КОМПАС 3D;
* PenBodyBuilder – класс, осуществляющий вызов методов API, необходимых для постройки 3D-модели.

В ходе доработок и добавления дополнительной функциональности, архитектура проекта претерпела некоторые изменения.

Так, параметры стали представляться классом PenBodyParametersList, содержащий закрытый список параметров, который представлен набором экземпляров класса PenBodyParameter. Получить нужный параметр можно с помощью индексатора. Добавлена возможность выбора наличия и количества ребер корпуса ручки.

Диаграмма, полученная после этапа реализации, изображена на рисунке 3.2.

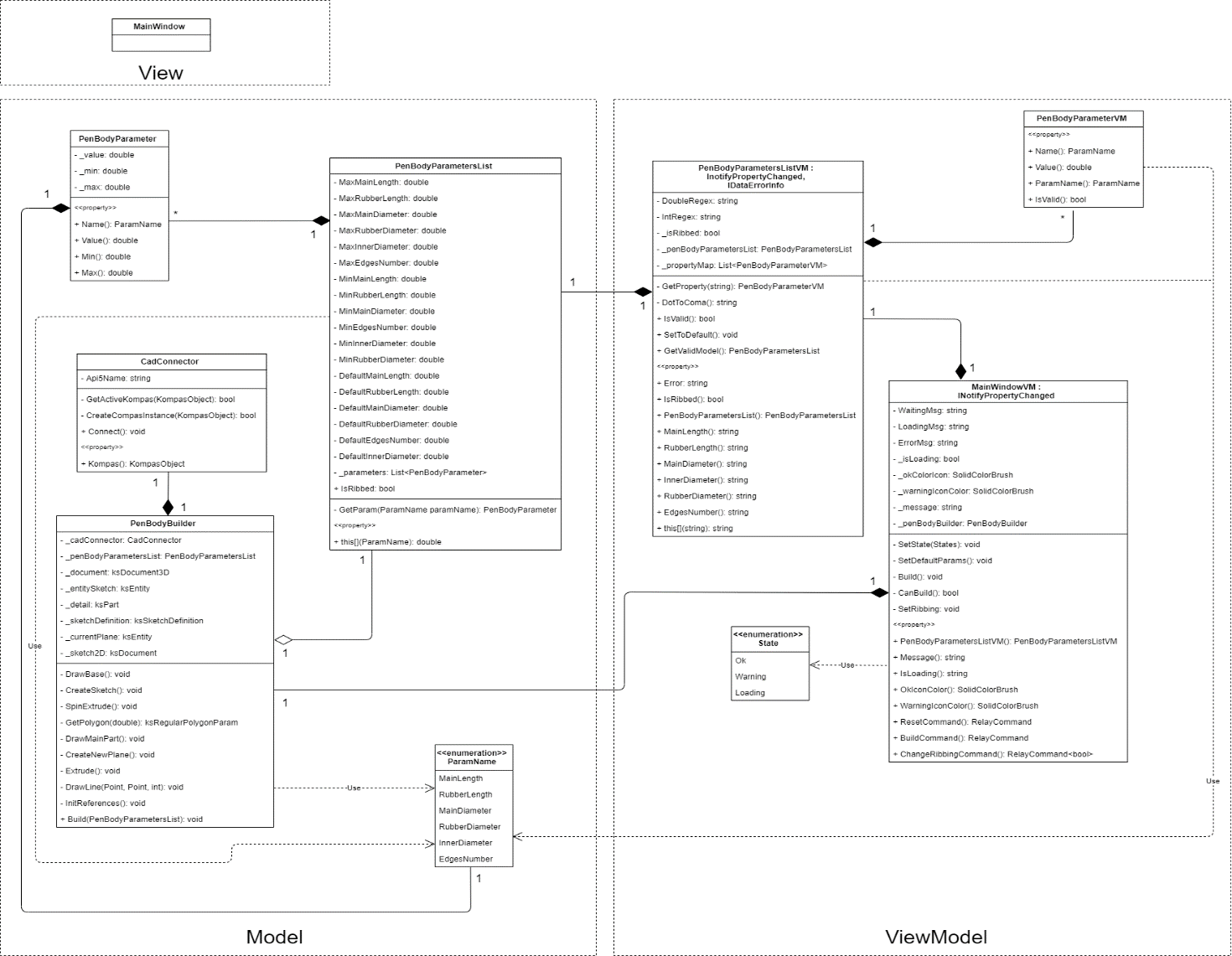


Рисунок 3.2 – Диаграмма классов после реализации и добавления дополнительной функциональности

# 3.3 Макет пользовательского интерфейса

Макет пользовательского интерфейса представляет собой форму для ввода параметров. Числовые значения параметров вводятся пользователем. Построение модели осуществляется нажатием на кнопку «Построить». На рисунке 3.3 представлен макет пользовательского интерфейса.

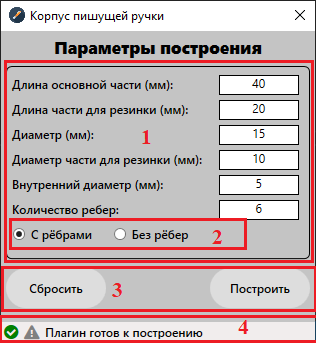


Рисунок 3.3 – Макет пользовательского интерфейса

В области 1 находятся поля для ввода параметров.

Уведомление о некорректности введенных данных обеспечено подсвечиванием поля и выводом сообщения во всплывающем окне, появляющемся при наведении курсора на подсвеченное поле (Рисунок 3.4).

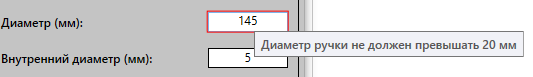


Рисунок 3.4 – Уведомление о некорректности введенных данных

В области 2 находятся радиокнопки переключения наличия рёбер у корпуса ручки. При переключении в режим «Без рёбер», поле «Количество рёбер» исчезает из списка параметров.

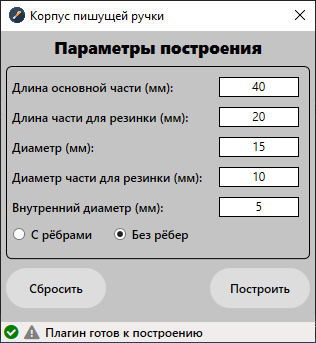


Рисунок 3.5 – Режим «Без рёбер»

В области 3 находится кнопка, активация которой приведет к сбросу введенных пользователем параметров до параметров по умолчанию (приведены на Рисунке 3.3).

В области 4 находится кнопка, нажатие на которую приведет к запуску процесса построения модели в Компас-3D по введенным параметрам, если они корректны. В случае некорректности, кнопка будет заблокирована до ее устранения.

В области 5 находится строка состояния плагина. Состояние отображается текстом, а также иконками и индикатором выполнения. Возможные состояния приведены на рисунке 3.6.

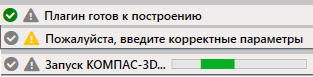


Рисунок 3.6 – Вариации отображения состояния плагина

# 4 Тестирование

# 4.1 Функциональное тестирование

Функциональное тестирование – это тестирование функциональности и поведения программы на соответствие требованиям функциональной спецификации. Функциональная спецификация определяет, что именно делает ПО, какие задачи оно решает. В данном случае будет проверяться правильность построения детали при различных входных параметрах

Построение модели с минимальными входными данными:

* Внешний диаметр D1 – 10 мм;
* Диаметр части для резинки D2 – 7 мм;
* Внутренний диаметр D3 – 2 мм;
* Длина части для резинки L1 – 15 мм;
* Длина основной части L2 – 30 мм;
* Количество рёбер N – 4;

Модели с минимальными входными параметрами показаны на рисунках 4.1 и 4.2.

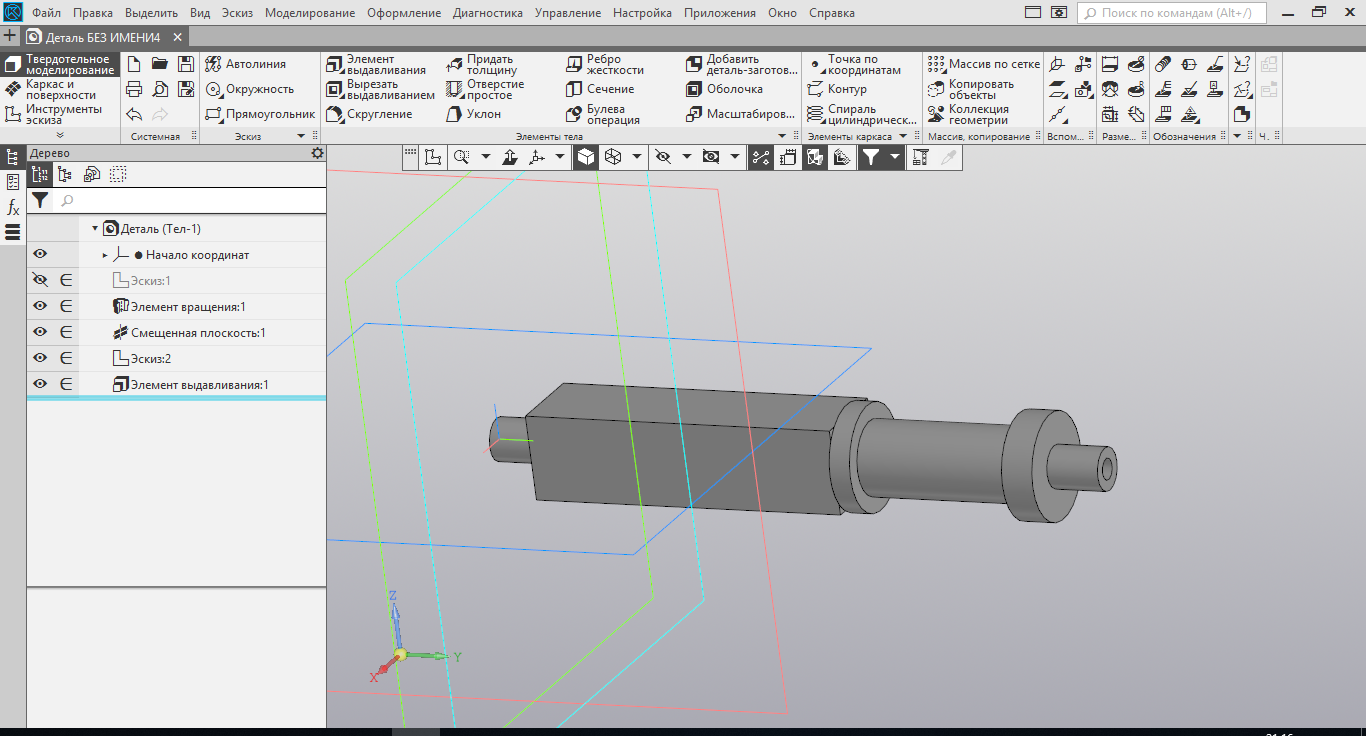


Рисунок 4.1 – Модель с рёбрами, построенная с минимальными входными параметрами

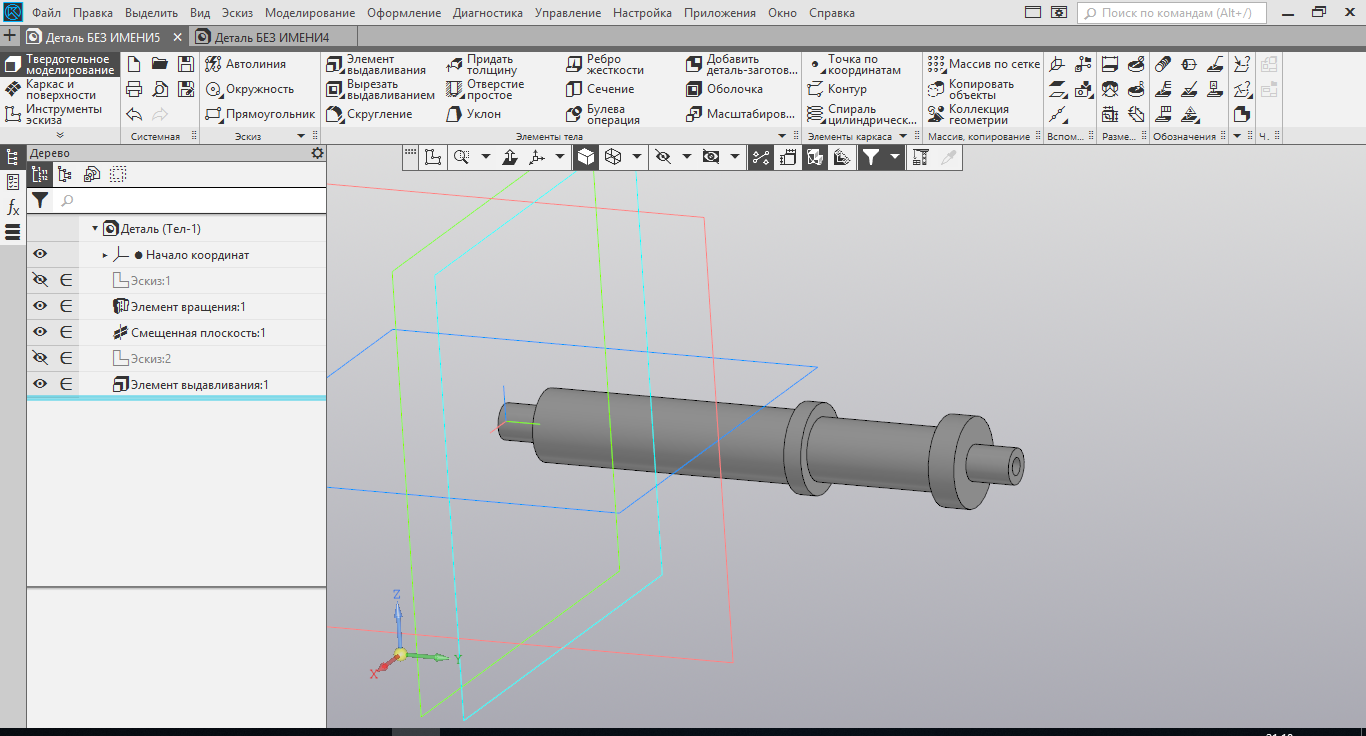


Рисунок 4.2 – Модель без рёбер, построенная с минимальными входными параметрами

Построение модели с минимальными входными данными:

* Внешний диаметр D1 – 18 мм;
* Диаметр части для резинки D2 – 16 мм;
* Внутренний диаметр D3 – 7 мм;
* Длина части для резинки L1 – 30 мм;
* Длина основной части L2 – 70 мм;
* Количество рёбер N – 10;

Модели с максимальными входными параметрами показаны на рисунках 4.3 и 4.2.

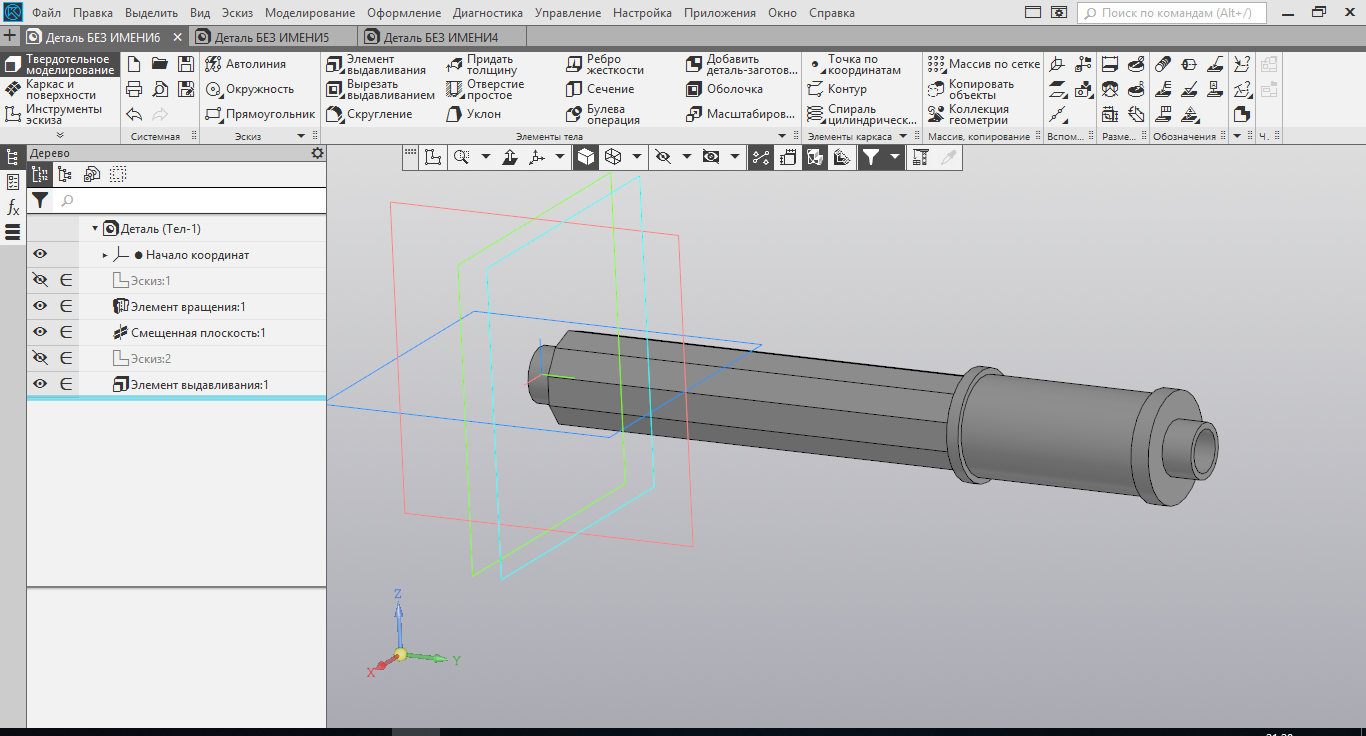


Рисунок 4.3 – Модель с рёбрами, построенная с максимальными входными параметрами

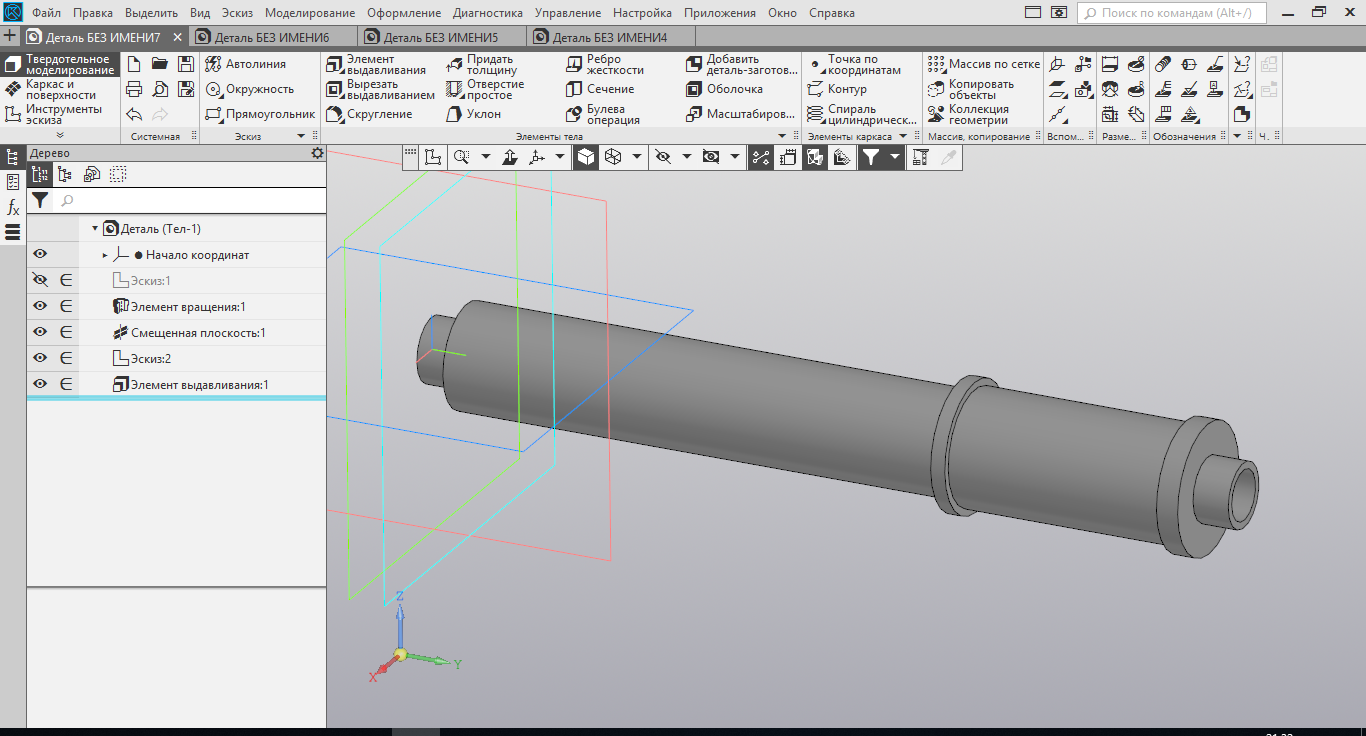


Рисунок 4.4 – Модель без рёбер, построенная с минимальными входными параметрами

При вводе заведомо некорректных данных программа сообщает об ошибках (рис. 4.5 – 4.10).

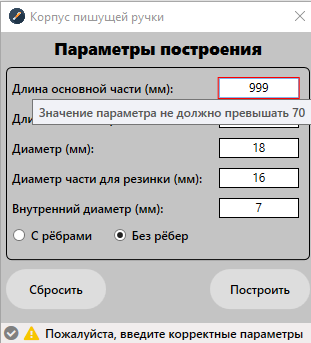


Рисунок 4.5 – Ввод некорректной длины основной части

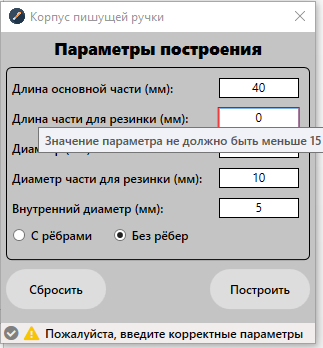


Рисунок 4.6 – Ввод некорректной длины части для резинки

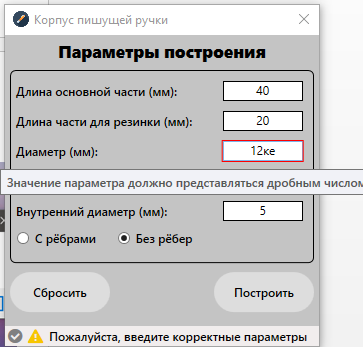


Рисунок 4.7 – Ввод некорректного диаметра ручки

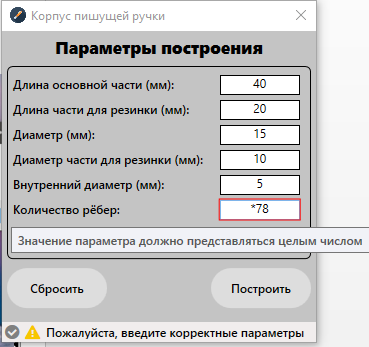


Рисунок 4.8 – Ввод некорректного количества рёбер

# 4.2 Модульное тестирование

Юнит-тестирование (англ. «*unit*-*testing*», или модульное тестирование) — тестирование отдельного элемента изолированно от остальной системы. Относительно парадигмы объектно-ориентированного программирования, системой является вся программа, а отдельным элементом — класс или его метод. Юнит-тестирование предназначено для проверки правильности работы отдельно взятого элемента. Чтобы исключить из результатов тестирования влияние потенциальных ошибок других элементов, тестируемый элемент должен быть максимально изолирован, то есть не использовать объекты и методы других классов [7].

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

Тестирование проводилось с помощью фреймворка модульного тестирования NUnit 3.13.1 для языков платформы .NET.

Результаты прохождения всех модульных тестов приведены на рисунке 4.9.

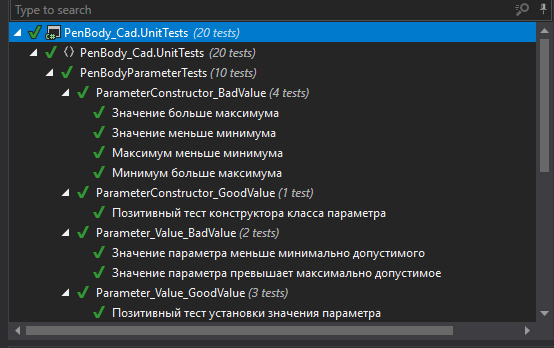


Рисунок 4.9 – Список пройденных юнит-тестов

Результат покрытия моделей PenBodyParameter и PenBodyParametersList тестами приведен на рисунке 4.11. Цикломатическая сложность равна 65.

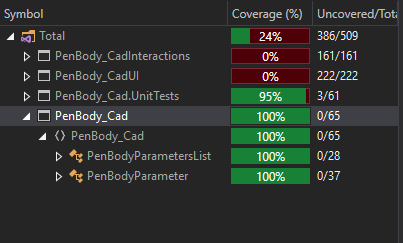


Рисунок 4.10 – Результат покрытия тестами

# 4.3 Нагрузочное тестирование

Нагрузочное тестирование – тип тестирования, который позволяет оценить поведение системы при возрастающей нагрузке, целью нагрузочного тестирования является также определение максимальной нагрузки, которую может выдержать система [10].

Нагрузочное тестирование проводилось на компьютере со следующими характеристиками:

* Процессор: Intel Core i5-6198DU 2.30-2.40 GHz;
* Оперативная память: 8192MB SODIMM 2133 MHz;
* Графический процессор: Intel HD Graphics 510;
* Операционная система: Windows 10 Корпоративная LTSC 64-bit.

При задании максимального количества последовательно строящихся моделей равным 200, САПР «КОМПАС-3D» остановила дальнейшие построения после 160 модели, окно программы перестало реагировать на ввод с сообщением о недостатке оперативной памяти. В диспетчере задач у процесса запущенной САПР отображался статус «Не работает», также произошел сбой и перезагрузка Проводника Windows. Исходя из этого, для выполнения нагрузочного тестирования, количество построенных деталей было принято равным 160.

Зависимость потребления оперативной памяти САПР «КОМПАС-3D» при построенных 160 моделях представлена на графике (рис. 4.12).



Рисунок 4.11 – Зависимость потребление ОЗУ от количества построенных моделей

Потребление памяти резко возросло после построения 25 моделей. Пик потребления был достигнут на 152 построении и составил 1774 МБ.

График на рисунке 4.12 отображает зависимость застрачиваемого времени на каждое новое построение от количества уже построенных моделей. Время на построение резко возрастает на построении 76 модели, составляя 159.5 секунд. Далее наблюдаются резкие колебания во времени построения. Увеличение времени построения происходит по той причине, что скорость чтения и загрузки данных с использованием виртуальной памяти значительно ниже из-за ее меньшей пропускной способности. Пик времени был достигнут на 116 модели и составил 276 секунд.

Рисунок 4.12 – Зависимость затрат времени на построение от количества построенных моделей

# Заключение

В ходе выполнения лабораторных работ в рамках курса «Основы разработки систем автоматизированного проектирования» были изучены основные этапы проектирования программного продукта и его реализации, а также SDK «КОМПАС-3D». Было составлено техническое задание, разработан проект системы, составлена UML диаграмма классов, разработан макет пользовательского интерфейса.

В результате работы был спроектирован и реализован плагин для САПР «КОМПАС-3D», выполняющий построение 3D-модели корпуса пишущей ручки.

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

# Список литературы

Норенков И.П. «Основы автоматизированного проектирования». Издательство: МГТУ; Москва: 2000 – 188 с.

API – Википедия. [Электронный ресурс]. — Режим доступа: <https://ru.wikipedia.org/wiki/API> (дата обращения 20.03.2020)

Плагин – Википедия. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Плагин> (дата обращения 20.03.2020)

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

Плагин PDF [Электронный ресурс]. — Режим доступа: <http://gkmsoft.ru/allcatalog/pdf2dkompas_plugin/> (дата обращения 30.03.2020)

Ручка – Википедия [Электронный ресурс] – Режим доступа: [https://ru.wikipedia.org/wiki/Ручка](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%87%D0%BA%D0%B0_(%D0%BA%D0%B0%D0%BD%D1%86%D0%B5%D0%BB%D1%8F%D1%80%D0%B8%D1%8F)) (дата обращения 20.03.2020)

Новые технологии в программировании: учебное пособие / А.А.Калентьев, Д.В.Гарайс, А.Е.Горяинов— Томск : Эль Контент, 2014.—176 с.

М. Фаулер. «UML. Основы». Издательство: Символ-Плюс; Booch Jacobson Rumbaugh: 2018 – 192 с.

MVVM [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-ViewModel> (дата обращения 20.03.2020)

Нагрузочное тестирование vs Тестирование производительности. [Электронный ресурс]. – <https://performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti> (дата обращения 09.04.2021)

# Приложение А

(справочное)

Таблица А – Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| **Метод** | **Входные данные** | **Описание** |
| Indexer\_Set\_BadValue(ParamName, double) | ParamName.MainLength, 30 | Длина основной части превышает длину части для резинки меньше, чем в 2 раза |
| ParamName.RubberLength, 25 | Длина части для резинки превышает половину длины основной части |
| ParamName.RubberDiameter, 16 | Диаметр части для резинки больше диаметра ручки |
| ParamName.RubberDiameter, 15 | Диаметр части для резинки меньше диаметра ручки меньше, чем на 2 мм |
| ParamName.InnerDiameter, 16 | Внутренний диаметр меньше диаметра части для резинки меньше, чем на 2 мм |
| Indexer\_Set\_GoodValue(ParamName, double) | ParamName.MainLength, 60 | Позитивный тест установки длины основной части |
|  | ParamName.RubberLength, 15 | Позитивный тест установки длины части для резинки |
| ParamName.MainDiameter, 18 | Позитивный тест установки диаметра ручки |
| ParamName.RubberDiameter, 13 | Позитивный тест установки диаметра части для резинки |

Продолжение таблицы A

|  |  |  |
| --- | --- | --- |
|  | ParamName.InnerDiameter, 7 | Позитивный тест установки внутреннего диаметра |
| ParameterConstructor\_BadValue(ParamName, double, double, double) | ParamName.MainLength, 30, 40, 20 | Минимум больше максимума |
| ParamName.MainLength, -20, 10, 20 | Максимум меньше минимума |
| ParamName.MainLength, 30, 10, 5 | Значение меньше минимума |
| ParamName.MainLength, 30, 40, 50 | Значение больше максимума |
| ParameterConstructor\_GoodValue() |  | Позитивный тест конструктора класса параметра |
| Parameter\_Value\_GoodValue(double) | 25 | Позитивный тест установки значения параметра |
| 30 | Позитивный тест установки значения параметра на границе максимального |
| 10 | Позитивный тест установки значения параметра на границе минимального |

Продолжение таблицы A

|  |  |  |
| --- | --- | --- |
| Parameter\_Value\_BadValue | 50 | Значение параметра превышает максимально допустимое |
| 5 | Значение параметра меньше минимально допустимого |