Министерство науки и высшего образования Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

(ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Пояснительная записка

«**Плагин для построения колбы Вюрца в «КОМПАС** **- 3D**»

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель: |  | Выполнил: |  |
| к.т.н., доцент кафедры КСУП | | студент гр. 588-3 | |
| Калентьев А. А. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Зотов А.С. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата: | « » \_\_\_\_\_\_20\_\_ г. | Дата: | « » \_\_\_\_\_20\_\_ г. |

# Реферат

Лабораторная работа, 29 страниц, 22 рисунка, 5 таблиц, 5 источников.

Ключевые слова: САПР, КОМПАС-3D, КОЛБА ВЮРЦА, МОДЕЛЬ, РАЗРАБОТКА, ПЛАГИН, API.

Целью данной работы является разработка плагина для создания трёхмерных моделей колб Вюрца, согласно заданным параметрам для системы автоматизированного проектирования «КОМПАС-3D V.20».

Отчёт по лабораторной работе выполнен в текстовом редакторе Microsoft Word.

**Содержание**

[Реферат 2](#_Toc91334910)

[Введение 4](#_Toc91334911)

[1 Назначение приложения 4](#_Toc91334912)

[2 Стек технологий разработки 4](#_Toc91334913)

[3 Описание задачи 5](#_Toc91334914)

[4 Проект программы-плагина и описание алгоритмов 10](#_Toc91334915)

[5 Пользовательский интерфейс 12](#_Toc91334916)

[6 Тестирование программы 14](#_Toc91334917)

[Заключение 26](#_Toc91334918)

[Список использованных источников 27](#_Toc91334919)

[Приложение А 28](#_Toc91334920)

# Введение

В настоящей пояснительной записке описывается внутренняя структура и логика разработанного плагина к САПР «КОМПАС-3D V.20».

# Назначение приложения

Плагин выполнен в виде пользовательского приложения для операционной системы Windows. Он предназначен для генерирования трёхмерной модели колбы Вюрца, согласно задаваемым пользователем параметрам. Приложение должно предоставить пользователю возможность ввести параметры колбы, а также непосредственно запускать САПР «КОМПАС-3D» и отдавать ему команды для построения модели.

# Стек технологий разработки

1. .NET 5.0. Используется в качестве базового фреймворка для проектов решения.
2. WinForms. Используется для разработки пользовательского интерфейса приложения.
3. API КОМПАС-3D V.20. Используется для связи с САПР «КОМПАС-3D» посредством встроенного API.
4. NUnit 3.13.2. Используется для юнит-тестирования бизнес-логики приложения.

Приложение разработано на языке C# 9.0. Ресурсоёмких задач приложение не содержит, интерфейс должен соответствовать стандартам ОС Windows. Поскольку язык C# является частью .NET, он позволяет реализовать все части приложения в рамках одного базового фреймворка. Поэтому он является оптимальным выбором для данной задачи.

Плагин должен работать на компьютерах следующей конфигурации:

* операционная система Microsoft Windows 10 и новее, 32-битная и 64-битная версия;
* процессор с тактовой частотой от 1 ГГц;
* от 2 гигабайт ОЗУ;
* графическое устройство с поддержкой DirectX 9 или выше.

# Описание задачи

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 [1].

Система «Компас-3D» включает следующие компоненты: система трёхмерного твердотельного моделирования, универсальная система автоматизированного проектирования «Компас-График» и модуль формирования спецификаций.

Ключевой особенностью продукта является обеспечение сквозного процесса проектирования от реализации идеи в 3D до подготовки полного комплекта документации. В основе КОМПАС-3D лежат собственное математическое ядро и параметрические технологии, разработанные специалистами АСКОН. Продукт содержит инструменты для коллективного проектирования изделий и объектов строительного проектирования любой степени сложности и позволяет подготовить полноценную электронную модель изделия, здания и сооружения.

Базовая функциональность продукта легко расширяется за счёт различных приложений, дополняющих функционал КОМПАС-3D эффективным инструментарием для решения прикладных инженерных задач.

Взаимодействие внешнего приложения или подключаемого модуля с системой КОМПАС осуществляется посредством программных интерфейсов, называемых API. В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. [2]

Главным интерфейсом API системы КОМПАС является KompasObject. Получить указатель на этот интерфейс можно с помощью экспортной функции CreateKompasObject(). Методы этого интерфейса реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы. В таблице 3.1 приведены методы интерфейса KompasObject.

Таблица 3.1 – Методы интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Возвращаемое значение | Описание |
| Document3D() | – | Указатель на интерфейс документа трехмерной модели ksDocument3D | Даёт возможность получить указатель на интерфейс трёхмерного документа (детали или сборки) |
| ActiveDocument3D() | – | Указатель на интерфейс документа трехмерной модели ksDocument3D | Дает возможность получить указатель на активный трехмерный документ |
| GetParamStruct(long type) | structType – тип интерфейса параметров | Указатель на интерфейс указанного типа из StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible | – | bool | Свойство видимости приложения |
| GetMathematic2D() | – | Указатель на интерфейс ksMathematic2D | Метод для получения указателя на интерфейс для работы с математическими функциями |

Графические документы имеют собственный интерфейс – ksDocument2D, со своими специфическими свойствами и методами. С помощью функций, присутствующих в ksDocument2D, создаются изображения в эскизах трехмерных операций. Свойства (члены данных) этого интерфейса позволяют динамически управлять настройками любого трехмерного документа системы из модуля. Наиболее используемые из них приведены в таблице 3.2.

Таблица 3.2 – 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) |
| UpdateDocumentParam() | – | TRUE – в случае успешного за­вершения | Активизировать измененные параметры документа |
| GetPart(int type) | type – тип компо­нента из пере­числения [Типы компонентов](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/PartType.htm) | – | Получить указатель на интерфейс компонента в соответствии с заданным типом |

Метод ksDocument3D::GetPart возвращает указатель на интерфейс детали или компонента сборки – ksPart. Свойства и методы этого интерфейса управляют состоянием компонентов сборки, они почти полностью дублируют команды контекстного меню и панели свойств, доступные пользователю при работе с тем или иным компонентом.[2] наиболее используемые из них приведены в таблице 3.3.

Таблица 3.3 – Методы интерфейса IPart

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Возвращаемое значение | Описание |
| EntityCollection  (short objType) | objType – тип объектов, содержащихся в массиве | указатель на интерфейс [ksEntityCollection](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntityCollection.htm) или [IEntityCollection](mk:@MSITStore:D:\INSTAL\KOMPAS-3D%20V17.1\KOMPAS\SDK\SDK.chm::/ksEntityCollection.htm) | Формирует массив объектов и возвращает указатель на его интерфейс |
| 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) | |  |  | | --- | --- | | objType | - [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 |
| BeginEdit() | - | указатель на интерфейс документа [ksDocument3D](mk:@MSITStore:C:\Program%20Files\ASCON\KOMPAS-3D%20v20%20Study\SDK\SDK.chm::/ksDocument3D.htm) | Запустить режим редактирования на месте для данного компонента |
| EndEdit(bool rebuild) | |  |  | | --- | --- | | rebuild | - перестроить компонент | | |  | | --- | | TRUE- в случае успешного завершения. | | Закрыть режим редактирования на месте для данного компонента |

Плагин должен обладать графическим интерфейсом для ввода следующих параметров:

* A — диаметр колбы (от 20 до 100 мм);
* B — длина горла (от 40 до 200 мм);
* C — диаметр отвода (от 5 до 30 мм);
* D — длина отвода (от 30 до 300 мм);
* E — диаметр горла (от 10 до 85 мм).

Плагин имеет зависимые параметры:

* диаметр колбы A ≥ 2\*E диаметра горла;
* диаметр горла ;
* длина отвода D ≤ A+B .

На рисунке 3.1 представлена 3D модель колбы Вюрца с указанными параметрами.



Рисунок 3.1 – 3D-модель колбы Вюрца

Должна обеспечиваться проверка корректности введённых данных. При вводе некорректных данных должно отображаться соответствующее сообщение.

Далее при нажатии на кнопку «Построить», плагин строит 3D модель колбы Вюрца в САПР «Компас-3D» по заданным значениям.

# Проект программы-плагина и описание алгоритмов

Исходя из назначения программы, логично выделить в ней несколько архитектурных единиц: графический интерфейс пользователя, бизнес-логику и внешние библиотеки. В число последних входят фреймворки, используемые при разработке, а также библиотеки API «КОМПАС-3D». Кроме того, необходимо покрыть бизнес-логику юнит-тестами.

Плагин включает в себя следующие классы

1. FlaskWurthzParametrs – класс, хранящий параметры колбы Вюрца и проводящий их валидацию. Данный класс включает в себя набор изменяемых свойств для каждого параметра («FlaskWurthzDiameter», «NeckDiameter», «BendLength» и т.д.), а также приватный метод «SetValue» для присваивания значений в свойства.
2. Перечисление ParameterName – содержит названия всех параметров колбы, а именно «FlaskWurthzDiameter», «NeckDiameter», «BendLength» и других. Используется для минимизации ошибки при передаче имени параметра в какой-либо метод.
3. Validator – статический класс, который содержит единственный статический метод «AssertRangeParameters», который принимает максимальное и минимальное значения диапазона, а также значение которое требуется проверить и текущее название параметра из перечисления «ParameterName»
4. KompasWrapper – класс служит для связи плагина с программой КОМПАС-3D V.20. Содержит публичные поля «\_wrapper» и «\_part», которые в последствии использует FlaskWurthzBuilder для построения модели. Также класс производит запуск программы и проверяет текущую сессию с открытой программой.
5. FlaskWurthzBuilder – класс, содержащий ряд методов для построения 3D модели колбы. Публичный метод «Assembly» принимает объект класса FlaskWurthzParametrs с параметрами для построения модели. Также содержит три метода для построения отдельных частей колбы, а именно метод «BuildFlask», метод «BuildNeck» и метод «BuildBend», которые тоже находятся в данном классе. Также класс содержит ряд вспомогательных методов, которые выполняют конкретную операцию, например, «создать элемент вращения» или «создать отложенную плоскость».
6. MainForm – класс, отвечающий за интерфейс программы. Класс содержит ряд методов для проверки введенных пользователем данных, а также для обновления актуальных допустимых значений зависимых параметров. Содержит поле для вывода ошибок ввода.

Далее рассмотрим диаграммы классов плагина.

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 [3].

UML-диаграммы классов изображены в Приложении А.

Общий алгоритм работы программы заключается в следующем.

1. Пользователь вводит параметры колбы в соответствующие поля для ввода, меняя тем самым значения свойств модели FlaskWurthzParametrs. Если какое-либо из значений оказывается недопустимым, то программа подсветит поле, в котором пользователь совершил ошибку, а также в нижней части формы будет показан текст ошибки. Также, пользователь не сможет нажать на кнопку построения, пока в программе будут присутствовать ошибки ввода.
2. При нажатии кнопки построения модели инициируется процесс построения модели, состоящий из нескольких шагов. За весь процесс построения отвечает класс FlaskWurthzBuilder.
3. Полученные параметры приведутся в значения КОМПАС-3D и будет произведено построение.

Алгоритмы работы программы реализованы согласно описанию. Стоит отметить, что в архитектуре программы имеются допущения в виде достаточно высокой степени связности и обращений бизнес-логики к API конкретному пакету САПР, а не к абстрактному, являющимся прослойкой между бизнес-логикой и API САПР.

После выполнения основной части работы в проект были внесены правки с требованием включить следующие изменения.

Был введен новый параметр «\_numberBends», отвечающий за количество отводов у колбы, который может принимать значения от 1 до 4 (включительно). Для него было написано свойство «NumberBends» для проверки правильности значения.

На форму были добавлены четыре объекта «RadioButton», чтобы пользователь мог выбрать требуемое количество отводов, также был написан метод «CheckRadioButton» для проверки, того какой из RadioButton отмечен пользователем и в зависимости от этого, параметру «\_numberBends» присваивается соответствующее значение.

Также в метод «BuildBend» класса «FlaskWurthzBuilder» был было добавлено несколько условий, чтобы определить какое количество отводов требуется построить.

Данная дополнительная функциональность также была реализована. Диаграмма классов приведена в приложении А, рисунки А.2.

# **Пользовательский интерфейс**

В пакете пользовательского интерфейса реализован класс окна: MainForm.

Представляет собой главное окно программы, в котором пользователь должен будет указать параметры колбы и инициировать построение.

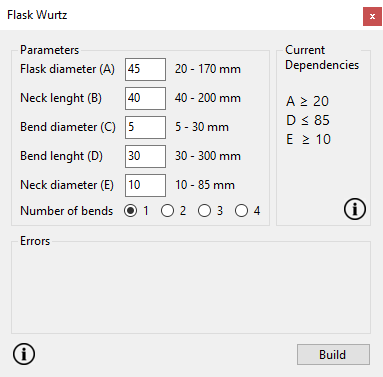


Рисунок 5.1 – Макет пользовательского интерфейса

Перед построением модели коробки пользователю необходимо задать значения ее параметров во всех полях для ввода, на которые наложены ограничения:

1. Пользователь может ввести только положительные значения в миллиметрах.
2. Выбрать количество отводов колбы, нажать на соответствующий RadioButton.
3. Только при корректном заполнении всех полей кнопка «Build» будет активна.

На рисунке 5.2 изображен некорректно заполненный интерфейс.

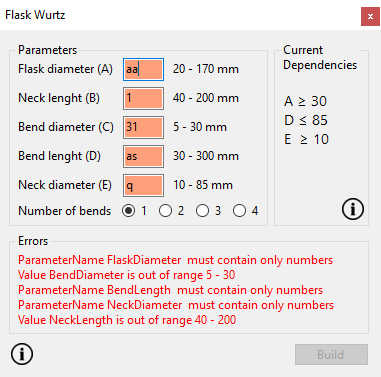


Рисунок 5.2 – Некорректно заполненный интерфейс

Также реализована подсказка в виде кнопки, которая выводит форму с чертежом колбы с проставленными размерами, она изображена на рисунке 6.3.



Рисунок 5.3 – Окно с выводом чертежа

# Тестирование программы

Тестирование программы состоит из нескольких этапов.

1. Функциональное тестирование. Тестируется заявленная функциональность программы.
2. Модульное тестирование. Тестируется бизнес-логика на корректность работы с помощью инструментов юнит-тестирования.
3. Нагрузочное тестирование. Тестируется влияние большого числа построений за короткий промежуток времени на рост использования ресурсов компьютера.

Ввод неправильных значений пресекается программой: при вводе неправильных данных кнопка «Build» заблокируется и в поле для ошибок появится одна или несколько ошибок, которые допустил пользователь.

Начнем с функционального тестирования.

Изначально все поля будут иметь значения по умолчанию.

* + радиус колбы ­45 мм;
  + диаметр горла 10 мм;
  + длина горла 40 мм;
  + диаметр отвода 5 мм;
  + длина отвода 30 мм;
  + количество отводов: одна штука.

При нажатии кнопки «Build» происходит переключение на «КОМПАС-3D».

На рисунке 6.1 изображена 3D модель колбы Вюрца при параметрах по умолчанию.

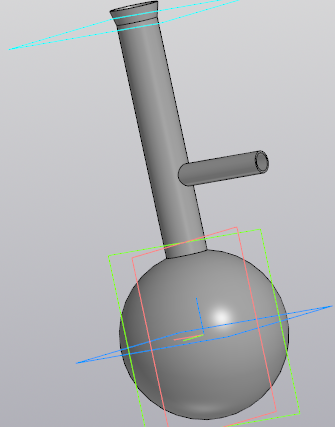


Рисунок 6.1 – 3D модель колбы Вюрца при параметрах по умолчанию

Далее, количество отводов было увеличено до четырех, параметры остались по умолчанию. Результат изображен на рисунке 6.2

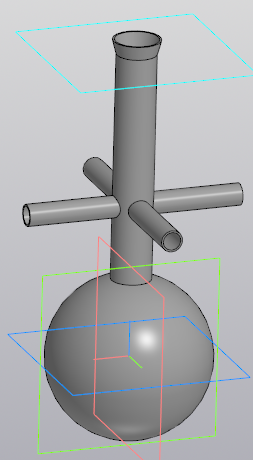


Рисунок 6.2 – 3D модель колбы Вюрца при параметрах по умолчанию с четырьмя отводами

Далее изменим параметры на максимальные, они равны, соответственно:

* радиус колбы ­170 мм;
* диаметр горла 85 мм;
* длина горла 200 мм;
* диаметр отвода 30 мм;
* длина отвода 300 мм.

Количество отводов оставим равным четырем. Результат построения изображен на рисунке 6.3

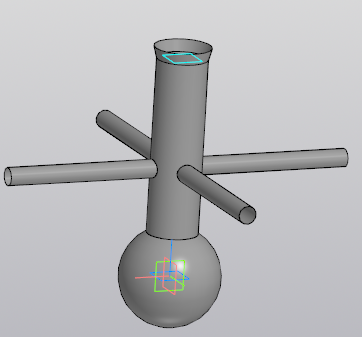


Рисунок 6.3 – 3D модель колбы Вюрца при максимальных параметрах с четырьмя отводами

Оставим максимальные параметры, изменим количество отводов до трех. Результат изображен на рисунке 6.4.

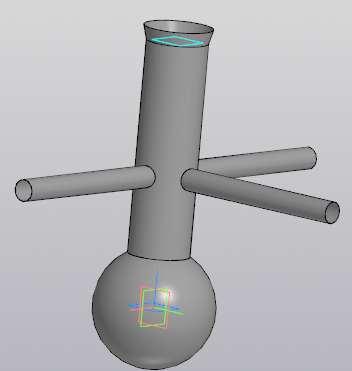


Рисунок 6.4 – 3D модель колбы Вюрца при максимальных параметрах с тремя отводами

Затем произведем построение с минимальными параметрами, они равны:

* радиус колбы ­20 мм;
* диаметр горла 10 мм;
* длина горла 40 мм;
* диаметр отвода 30 мм;
* длина отвода 30 мм.

Количество отводов две штуки. Результат построения изображен на рисунке 6.5.

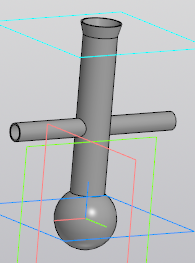


Рисунок 6.5 – 3D модель колбы Вюрца при минимальных параметрах с двумя отводами

Далее, уменьшим количество отводов до одного, результат построения изображен на рисунке 6.6.

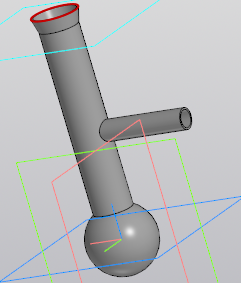


Рисунок 6.6 – 3D модель колбы Вюрца при минимальных параметрах с одним отводом

Далее было произведено юнит-тестирование плагина.

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 [4].

Ниже, на рисунке 6.7, представлен список тестов параметров.

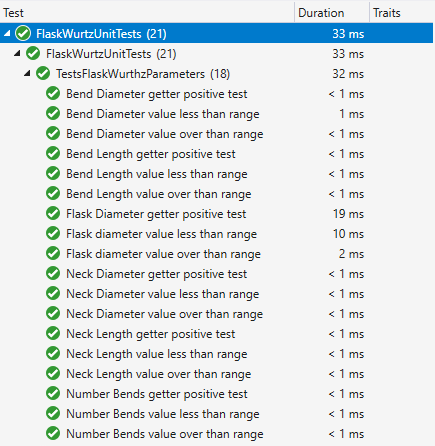


Рисунок 6.7 – Список тестов параметров

На рисунке 6.8 изображен список тестов класса «Validator».

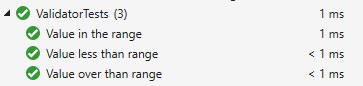


Рисунок 6.8 – Список тестов класса «Validator»

Результат покрытия классов «Validator» и «FlaskWurthzParameters» тестами приведен на рисунке 6.9.

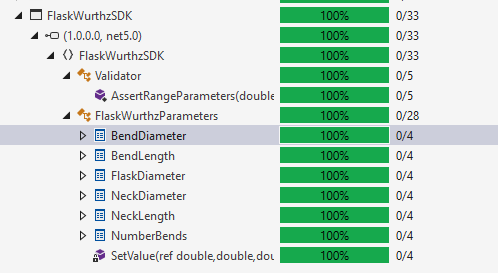


Рисунок 6.9 – Степень покрытия тестами бизнес-логики плагина

Ниже приведена таблица 6.1 со всеми написанными тестами для параметров колбы.

Таблица 6.1 – Методы юнит-тестирования параметров колбы Вюрца

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| TestParametersFlaskWurthz\_CorrectSetValue  (double correctValue, ParameterName parameter) | Реализованы 6 тестовых случаев с корректными значениями для каждого параметра колбы и соответствующим названием. Метод присваивает значение в текущий параметр и сравнивает его с параметром по умолчанию, если они равны, тест считается пройденным успешно. |
| TestParametersFlaskWurthz\_InvalidSetValue(double invalidValue, ParameterName parameter) | Реализованы 12 тестовых случаев с некорректными значениями для каждого параметра колбы (меньше диапазона и выше диапазона) и соответствующим названием. Метод пытается присвоить некорректное значение в текущий параметр и если выбрасывается исключение, тест считается пройденным успешно. |

Ниже представлена таблица 6.2 со всеми написанными тестами для валидатора.

Таблица 6.2 – Методы юнит-тестирования валидатора

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| TestValidator\_InvalidValue(double invalidValue,  double minValue, double maxValue, ParameterName parameter) | Негативный тест валидатора. Принимает два TestCase со значениями диапазона и со значением, которое требуется проверить. В первом TestCase оно меньше диапазона, а во втором, соответственно больше. Если проверка не проходит и выбрасывается исключение, то тест выполнен корректно. |
| TestValidator\_ValidValue(double validValue, double minValue, double maxValue, ParameterName parameter) | Позитивный тест валидатора. Принимает один TestCase со значениями диапазона и со значением, которое требуется проверить. Значение входит в диапазон. Если проверка прошла успешно и не было выброшено исключение, тест выполнен корректно. |

Далее, было выполнено нагрузочное тестирование.

Поскольку потребление памяти приложением статично, а скорость построения зависит от скорости работы «КОМПАС-3D» и от конфигурации компьютера, измерению в стресс-тесте будет подвергаться потребление памяти при многократных построениях. Стоит заранее отметить, что программой не допускается построение нескольких моделей одновременно, а только по очереди.

Конфигурация компьютера, на котором выполнялся тест:

1. X86-совместимый CPU Intel Core i3-7100U @ 2.40GHz.
2. 8 ГБ DDR4 ОЗУ на частоте 2133 МГц в одноканальном режиме.
3. Графический процессор NVidia GeForce MX130 с 2 ГБ видеопамяти.
4. ОС Windows 10 x64.

Нагрузочное тестирование изначально предполагало построение 1000 моделей и удержание их открытыми в «КОМПАС-3D». Тест завершился досрочно на построении восьмидесяти второй модели, так как «КОМПАС-3D» аварийно завершил свою работу без вывода информации об ошибке.

Потребление ОЗУ самой программой составляет в среднем 20 МБ, в то время как потребление памяти пакетом «КОМПАС-3D» сильно увеличивается со временем.

График зависимости потребления ОЗУ «КОМПАС-3D» от количества построенных моделей представлен на рисунке 6.11.

Рисунок 6.11 – Зависимость потребления ОЗУ от количества моделей при стандартных параметрах

Скорость построения напрямую зависит от свободного времени CPU и пропускной способности памяти. Косвенная зависимость от количества памяти заключается в том, что при её нехватке операционная система будет вынуждена предоставлять пакету «КОМПАС-3D» области файла подкачки, что существенно снизит его производительность. Такой ситуации не удалось достичь на вышеуказанной конфигурации по причине избытка памяти и преждевременного завершения работы «КОМПАС-3D».

Рисунок 6.12 – Зависимость затрат времени от количества моделей при стандартных параметрах

На нижеприведённом графике легко видеть, что количество времени, затрачиваемого на построение, растёт с количеством открытых моделей.

Пиковое потребление ОЗУ было при удержании восьмидесяти одной модели, оно составило 6,7 ГБ.

При построении колб с минимальными параметрами программа смогла построить восемьдесят девять моделей, после аварийно завершилась. График потребления ОЗУ и времени построения представлены на рисунках 6.13 и 6.14

Рисунок 6.13 – Зависимость потребления ОЗУ от количества моделей при минимальных параметрах

Рисунок 6.14 – Зависимость затрат времени от количества моделей при минимальных параметрах

При построении колб с максимальными параметрами программа смогла построить семьдесят одну модель, после чего аварийно завершилась. График потребления ОЗУ и времени построения представлены на рисунках 6.15 и 6.16.

Рисунок 6.15 – Зависимость потребления ОЗУ от количества моделей при максимальных параметрах

Рисунок 6.16 – Зависимость затрат времени от количества при максимальных параметрах

Исходя из результатов тестирования, можно сделать вывод, что программа выполняет заявленную функциональность в полном объёме. Использовать её для построения сразу нескольких моделей и удержания их в памяти не представляется возможным, так как среда «КОМПАС-3D» V.20 не выдерживает нагрузки подобного рода. Рекомендуется строить модель колбы для разового применения.

# Заключение

В результате выполнения работы в рамках курса «Основы разработки систем автоматизированного проектирования» была разработана программа-плагин для системы «КОМПАС-3D», выполняющая построение трёхмерной модели колбы Вюрца.

Плагин выполнен в виде отдельного приложения, подключающегося к системе «КОМПАС-3D» и отдающего ему команды на построение модели. Заявленная функциональность реализована полностью, включая дополнительную функциональность. Тестирование программы проведено в три этапа: функциональное, модульное и нагрузочное, по итогам которого были проверены основные функции программы и защита от ввода некорректных данных, протестирована математическая модель колбы Вюрца и измерено влияние количества одновременно открытых построенных моделей на расход оперативной памяти компьютера.

# Список использованных источников

1. КОМПАС-3D – АСКОН. [Электронный ресурс]. – Режим доступа: <https://ascon.ru/products/7/review/> (дата обращения: 20.12.2021)
2. Кидрук Максим. КОМПАС-3D V10 на 100% / М. Кидрук. – СПб.: Питер, 2009 – 560 с.
3. М. Фаулер. UML. Основы, 3-е издание. — Пер. с англ. — СПб: символ-Плюс, 2004– 192 с.
4. Новые технологии в программировании: учебное пособие / А.А.Калентьев, Д.В.Гарайс, А.Е.Горяинов – Томск: Эль Контент, 2018.—176 с.

# Приложение А

Диаграмма классов

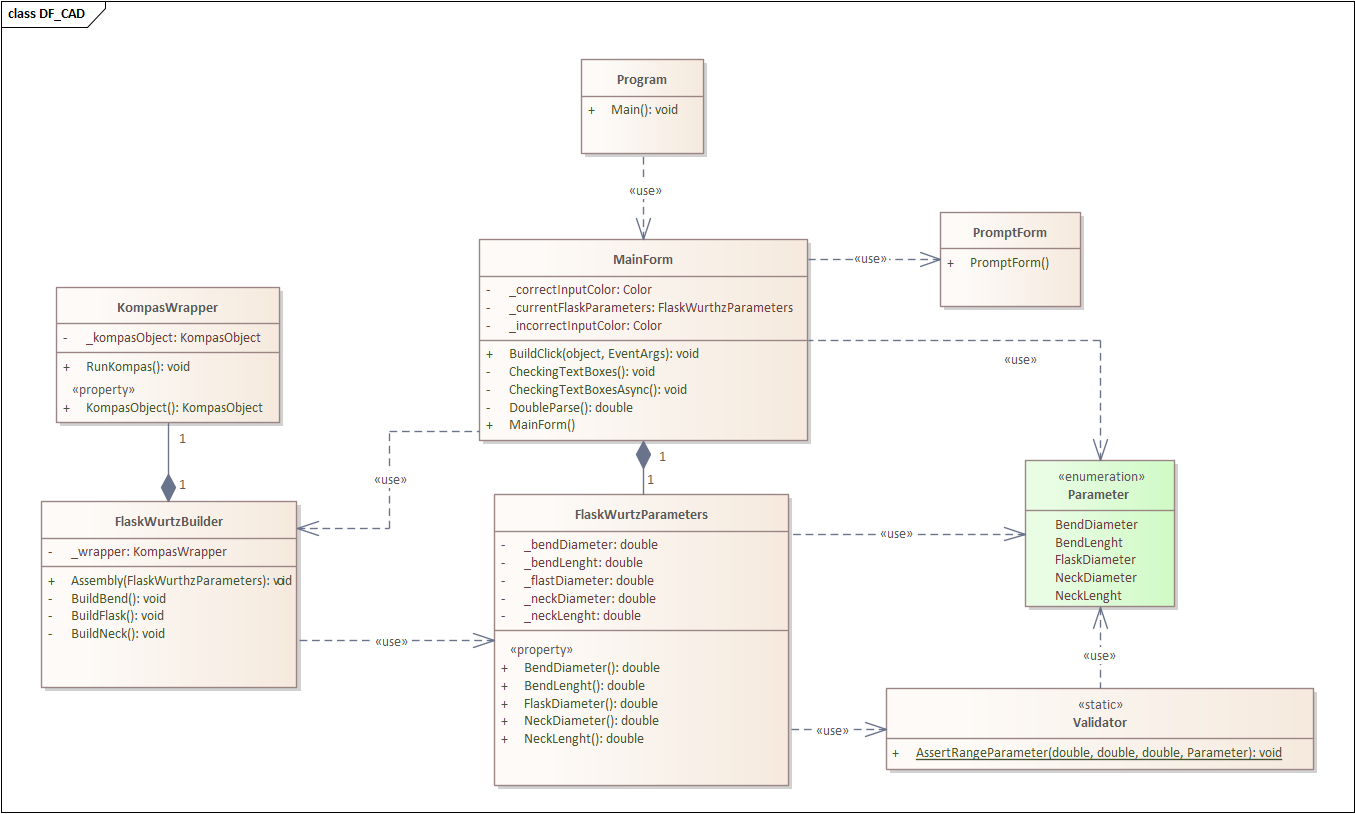


Рисунок А.1 – Изначальная UML диаграмма классов

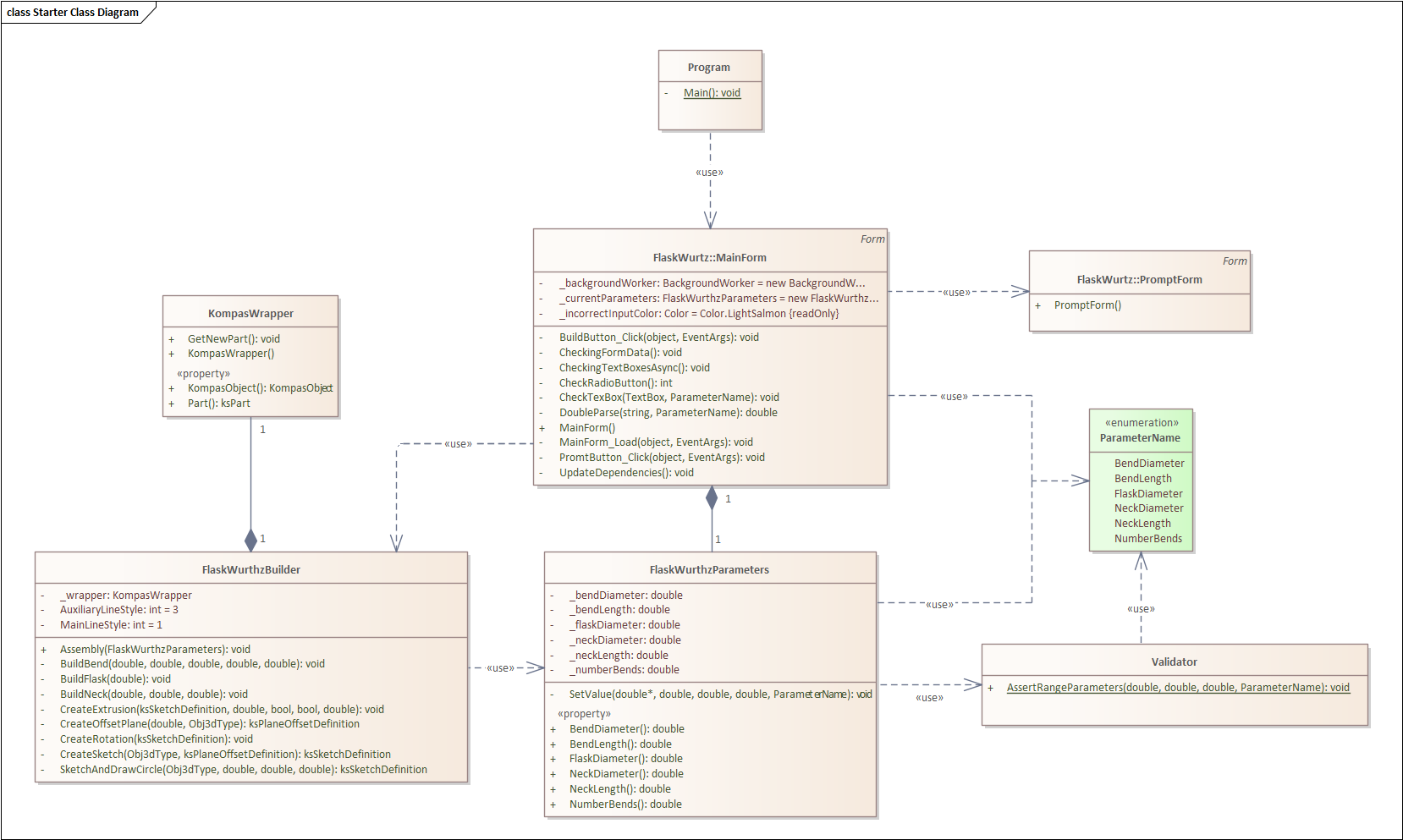


Рисунок А.2 – Итоговая UML диаграмма классов с новой функциональностью