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)

«Построение лабораторных стаканов с крышкой системе КОМПАС-3D v.20»

Пояснительная записка по дисциплине

«Основы разработки САПР» (ОрСАПР)

Выполнил:

Студент гр.588-3

\_\_\_\_\_\_\_Скоробогатов Д.Д.

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

Руководитель:

к.т.н., доцент

каф. КСУП

\_\_\_\_\_\_\_Калентьев А.А.

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

**РЕФЕРАТ**

Лабораторная работа 25 с., 16 рис., 11 источников.

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

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

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

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2016.

# Введение

Автоматизация моделирования имеет огромное значение для развития науки, техники и производства в современном обществе. В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Использование автоматизации в проектировании позволяет создавать все более сложные технические объекты и гибко реагировать на появление новых решений и технологий в той или иной области техники. Она позволяет значительно повысить точность расчетов, выбрать наилучшие варианты для реализации на основе строгого математического анализа всех или большинства вариантов проекта с оценкой технических, технологических и экономических характеристик производства и эксплуатации проектируемого объекта, значительно повысить качество конструкторской документации, существенно сократить сроки проектирования и передачи конструкторской документации в производство, эффективнее использовать технологическое оборудование с программным управлением [1].

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели «Волновод» для системы автоматизированного проектирования КОМПАС-3D с помощью интегрированной среды разработки Visual Studio 2019 Сommunity. [2]

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

# Описание API

Аббревиатура ***API*** расшифровывается как «Application Programming Interface» (интерфейс программирования приложений, программный интерфейс приложения). [4]

Сегодня встречаются задачи, решение которых не реализованы в CAD-системах. Чаще всего это очень узкоспециализированные задачи, которые встречаются на каком-то конкретном предприятии или подотрасли. Для решения подобных задач и их автоматизации используется API.

***API КОМПАС-3D*** — это ориентированные на прикладного программиста инструментальные средства разработки приложений (библиотек конструктивов, прикладных САПР) на базе системы КОМПАС. API КОМПАС-3D включает в свой состав API 5 и API 7.

Главным интерфейсом API системы КОМПАС является ***KompasObject.***[5] Методы этого интерфейса, реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы (интерфейсы динамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

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

Таблица 1.1 – Методы и свойства интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | 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(short structType) | 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 |

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 в модели |
| GetDefinition() | IUnkown | Получить указатель на интерфейс параметров объектов и элементов |
| Update() | bool | Изменить свойства объекта (используя ранее установленные свойства) |
| NewEntity (short objType) | указатель на интерфейс ksEntity или IEntity. | Реальный объект создается в модели после вызова метода [Create](mk:@MSITStore:D:\KOMPAS\SDK\SDK.chm::/ksEntity_Create.htm). |

В таблице 1.3 представлены свойства и методы интерфейса ksDocument2D, 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) | |
|  | |  |
| GetPart(int type) | 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(short objType) | objType – тип объектов, содержащихся в массиве. | ksEnintyCollection | Формирует массив объектов и возвращает указатель на его интерфейс |
| GetDefaultEntity(short objType) | objType – тип объекта | ksEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart(int type) | Type – тип компонента | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity(short objType) | 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\_bossExtrusion | Приклеивание выдавливанием | ksBossExtrusionDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_fillet | Операция "скругление" | ksFilletDefinition |
| o3d\_edge | Ребро | ksEdgeDefinition |
| o3d\_planeOffset | Смещённая плоскость | ksPlaneOffsetDefinition |

# Обзор аналогов

Model ChemLab — интерактивное лабораторное моделирование для Windows.

Model ChemLab - это продукт, включающий в себя интерактивное моделирование и рабочее пространство лабораторной записной книжки с отдельными областями для теории, процедур и наблюдений студентов. Обычно используемое лабораторное оборудование используется для моделирования. [3]

Интерфейс приложения представлен на рисунке 1.1



Рисунок 3.1 – Интерфейс приложения Model ChemLab

# Описание предмета проектирования

Предметом проектирования является стакан лабораторный с крышкой.

Стаканчики предназначены для взвешивания и хранения веществ при лабораторных работах.

Под желание заказчика в стакане могут изменять параметры, приведенные ниже:

1. Ширина колбы D (200≤D≤400) мм;
2. Ширина крышки B (200≤d≤400) мм;
3. Высота колбы H (300≤H≤650) мм;
4. Радиус основания ручки С (10≤С≤B\*1/4) мм;
5. Длинна ручки L (10≤B≤30) мм;
6. Радиус ручки R (С≤R≤40) мм;
7. Толщина стенки колбы A(7≤A≤20) мм.

На рисунке 2.1 представлен общий вид стакана с крышкой.



Рисунок 4.1 – Общий вид стакана с крышкой

Чертеж модели с обозначениями показан на рисунке 2.2



Рисунок 4.2 – Чертеж модели с обозначениями

## 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 [2], библиотека «Kompas6API5» [5] 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].

## Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием лабораторных колб разного вида с крышкой, с высотой, шириной и разными размерами ручки. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# Описание реализации

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценарии действия) использован стандарт UML.

UML язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML – моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML возможна генерация кода и наоборот. [8]

При использовании UML были простроена диаграмма классов.

## 5.1 Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

На рисунке 5.1 представлена изначальная диаграмма классов.

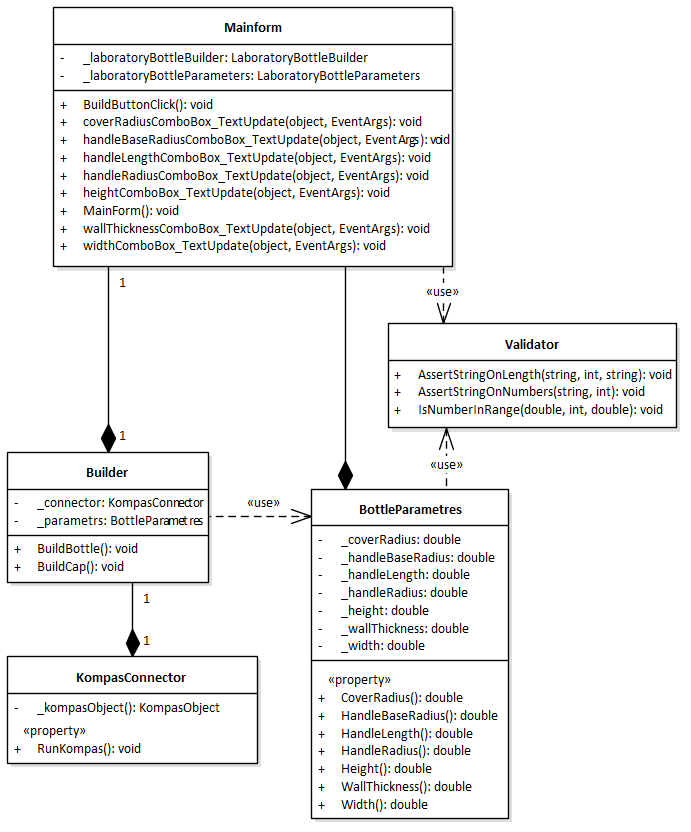


Рисунок 5.1 – Изначальная диаграмма классов

1. MainForm — класс диалогового окна, который обеспечивает взаимодействие между пользователем и программой;

2. BottleParameters — класс, хранящий введенные параметры лабораторных стаканов;

3. Builder — класс, осуществляющий вызов методов API, необходимых для постройки 3D-модели;

4. KompasConnector — класс для работы с API КОМПАС 3D.

5. Validator – имеет методы для проверки входных значений на корректность.

В итоговом проекте созданы следующие классы и методы, которые отображены на итоговой диаграмме классов (рисунок 5.2).

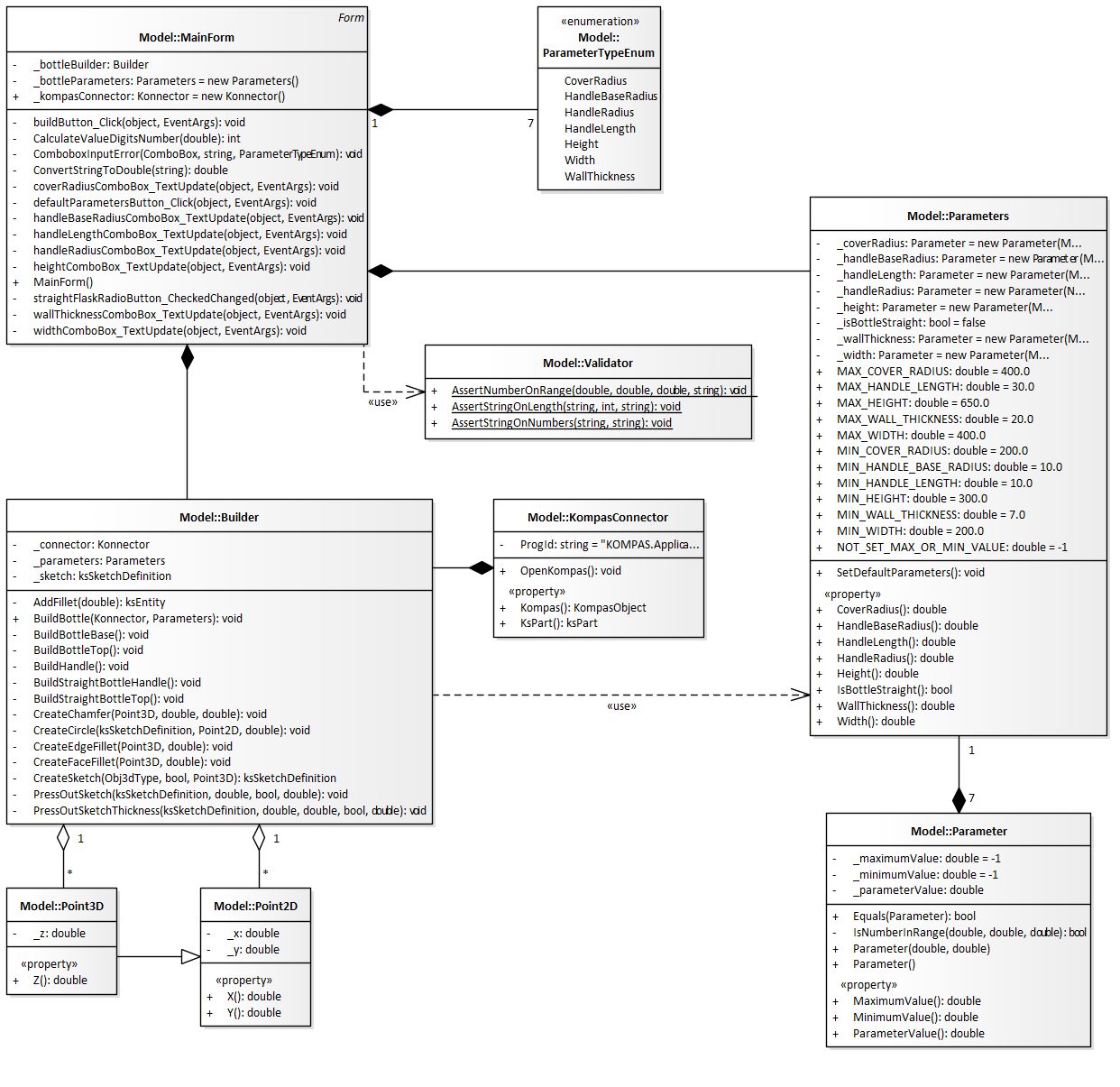


Рисунок 5.2 – Итоговая диаграмма классов

В процессе разработки была переосмыслена изначальная диаграмма классов, в результате чего:

Класс Validator теперь использует только MainForm’a, а валидация параметров происходит внутри класс Parameters.

Константы минимальных и максимальных значений параметров были занесены в поля класса Parameters.

Было добавлено перечисление ParameterTypeEnum для проверки вносимых значений и отображения неправильно введенных значений в форму.

Были добавлены два вспомогательных класса Point3D и Point2D для создания 3D и 2D точек соответственно.

## Unit Тестирование

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 5.3, представлен список тестов.

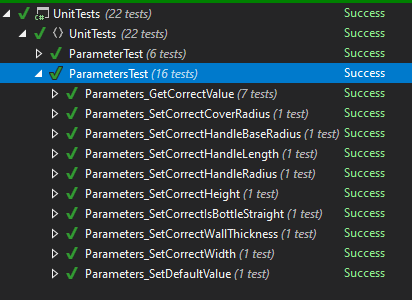


Рисунок 5.3 – Список тестов

Результат покрытия классов «Parameter» и «Parameters» тестами приведен на рисунке 5.4.

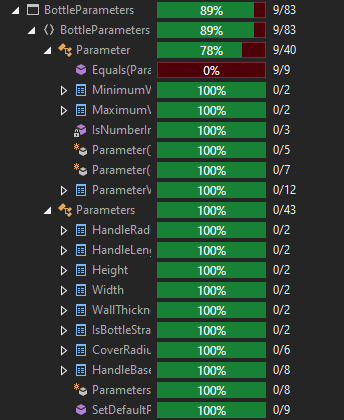


Рисунок 5.4 – Степень покрытия тестами бизнес-логики плагина

# Описание программы для пользователя

Пользовательский интерфейс представляет собой форму для ввода параметров. Так же на форме представлен чертеж с указанием всех параметров, для наглядности. Пользователь вводит значения самостоятельно, либо может выбрать значения из предложенного списка. Некоторые поля заблокированы для ввода, до тех пор, пока не будут введены параметры, от которых они зависят. Ввод проверяется на соответствие значения диапазону, отображенному рядом с заполняемым полем, а также на корректность ввода, то есть ввод других символов, кроме цифр. При неправильном вводе поле поле подкрашивается красным цветом. До тех пор, пока все поля не будут заполнены корректно, кнопка построить будет выключена.

Макет пользовательского интерфейса программы, а также примеры отображения при вводе представлен на рисунках 6.1 – 6.4

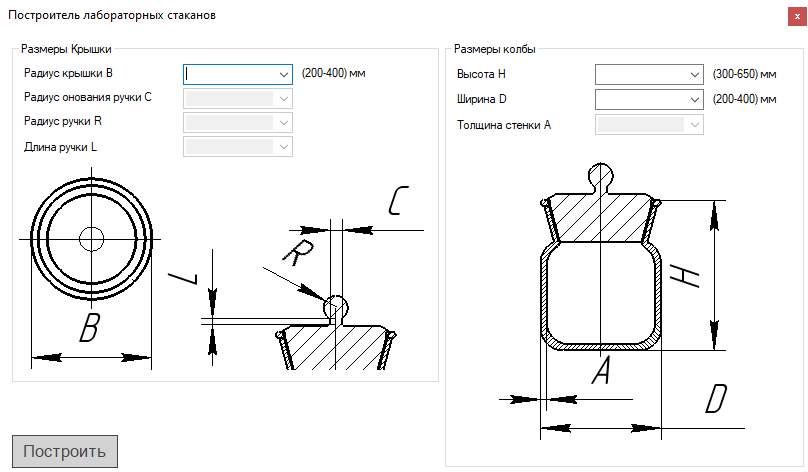


Рисунок 6.1 – Макет пользовательского интерфейса программы\

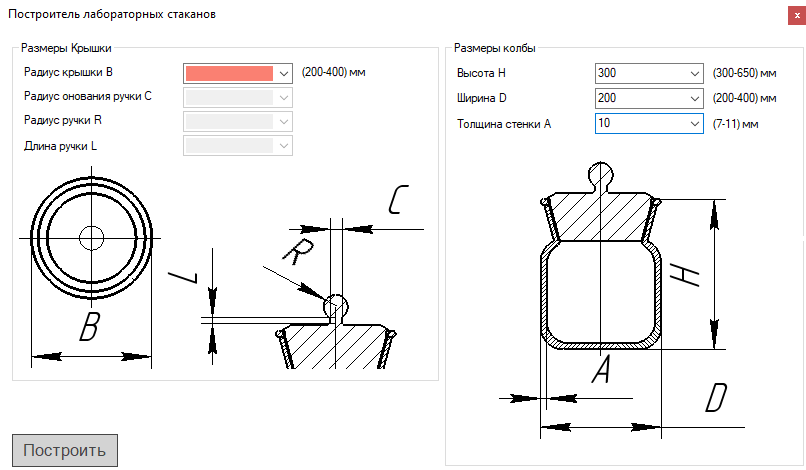


Рисунок 6.2 – Пример обработки ошибок при вводе параметров

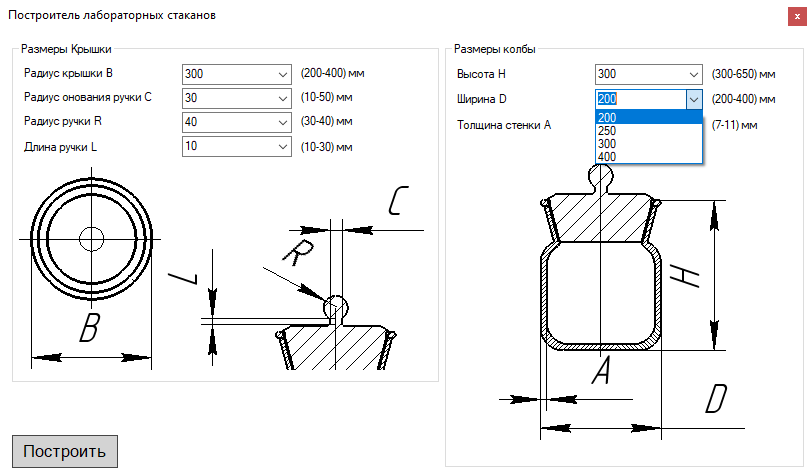


Рисунок 6.3 – Пример полностью введенных параметров

# Нагрузочное тестирование

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование [8]. Тестирование производилось на ПК со следующей конфигурацией:

* ЦП Intel Core i3-7100U;
* 6 ГБ ОЗУ;
* Графический процессор объемом памяти 2 ГБ.

Для нагрузочного тестирования создан метод с бесконечным циклом построения детали. Для измерения времени использовался класс Stopwatch.

Первое тестирование заключалось в построении детали с базовыми параметрами. Тестирование проводилось 1 час 2 минуты, построено 76 моделей. На протяжении тестирования загруженность процессора составляла около 84 процентов. Результаты тестирования представлены на рисунках 7.1 и 7.2. Параметры модели, используемой для тестирования минимальные.

Рисунок 7.1 – График зависимости количества потребляемой оперативной памяти от количества деталей для модели с минимальными параметрами

Рисунок 7.2 – График зависимости времени построения одной детали

от количества деталей для модели с минимальными параметрами

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 лабораторной бутылки с крышкой.

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, и измерено влияние количества одновременно открытых построенных моделей на расход оперативной памяти компьютера.

# Список использованных источников

1. Система трёхмерного моделирования КОМПАС-3D [электронный ресурс]. – URL: https://ascon.ru/products/7/review/ (дата обращения 26.10.2021).
2. Базовые интерфейсы API системы КОМПАС [электронный ресурс]. – URL: https://it.wikireading.ru/23741 (дата обращения 26.10.2021).
3. MechaniCS [электронный ресурс]. – URL: https://axoft.ru/vendors/CSoft-Development/CSoft-MechaniCS/ (дата обращения 27.10.2021).
4. Фаулер М. UML. Основы. 3-е издание / М.Фаулер. – 3-е изд., пер. с англ. – СПб.: Символ-Плюс, 2019. – 192 с. (дата обращения 09.11.2021).
5. Функциональное тестирование [Электронный ресурс]. – URL: https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/ (дата обращения: 10.12.2021).
6. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 10.12.2021).