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 "Кронштейн" ДЛЯ САПР КОМПАС-3D V20

Проект системы по лабораторному проекту

по дисциплине «ОСНОВЫ РАЗРАБОТКИ САПР»

Выполнил:

студент гр. 588-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лапардин А.С

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

Руководитель:

к.т.н., доцент каф. КСУП

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

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

## Содержание

[Описание САПР 2](#_Toc91691620)

[1.1 Описание программы 2](#_Toc91691621)

[1.2 Описание API 3](#_Toc91691622)

[2 Описание предмета проектирования 6](#_Toc91691623)

[3 Проект программы 8](#_Toc91691624)

[3.1 Описание технических и функциональных аспектов проекта 8](#_Toc91691625)

[3.2 Диаграмма классов 9](#_Toc91691626)

[3.3 Макет пользовательского интерфейса 11](#_Toc91691627)

[4 Тестирование программы 14](#_Toc91691628)

[4.1 Функциональное тестирование 14](#_Toc91691629)

[4.2 Модульное тестирование 16](#_Toc91691630)

[4.3 Нагрузочное тестирование 18](#_Toc91691631)

[Заключение 22](#_Toc91691632)

[Список использованной литературы 23](#_Toc91691633)

# Описание САПР

# Описание программы

САПР (Система автоматизированного проектирования) — автоматизированная система, реализующая информационную технологию выполнения функций проектирования, представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования, состоящую из персонала и комплекса технических, программных и других средств автоматизации его деятельности [1].

Компас — семейство систем автоматизированного проектирования, универсальная система автоматизированного проектирования, позволяющая в оперативном режиме выпускать чертежи изделий, схемы, спецификации, таблицы, инструкции, расчётно-пояснительные записки, технические условия, текстовые и прочие документы [2].

# Описание API

API (англ. Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой [5].

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 [3].

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 1.1 приведены методы интерфейса KompasObject [3].

Таблица 1.1 – 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() | structType – тип интерфейса параметров | Указатель на интерфейс указанного типа из StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 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 1.2.

Таблица 1.2 – 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 – в случае успешного завершения | Дает возможность создать пустой документ (деталь или сборку) |
| GetPart() | Type – тип компонента из перечисления | Указатель на интерфейс компонента ksPart или IPart | Позволяет получить указатель на интерфейс компонента в сборке |
| UpdateDocumentParam() |  | TRUE – в случае успешного завершения. | Позволяет обновить настройки документа |

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.

Таблица 1.3 – Методы интерфейса ksPart.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип входных параметров | Тип возвращаемых данных | Описание |
| EntityCollection() | objType – тип объектов | В случае успеха указатель на интерфейс ksEntityCollection или IEntityCollection, в случае неудачи – NULL. | Формирует динамический массив трехмерных объектов и возвращает указатель на его интерфейс |
| GetPart() | type – тип компонента из перечисления | Указатель на интерфейс компонента ksPart или IPart | Получить указатель на интерфейс компонента |
| NewEntity() | objType – тип объекта. | Указатель на интерфейс ksEntity или IEntity. | Создает интерфейс нового трехмерного объекта и возвращает указатель на него |
| GetDefaultEntity() | objType – тип объекта. | Указатель на интерфейс ksEntity или IEntity. | Возвращает указатель на интерфейс объекта, создаваемого системой в трехмерном документе по умолчанию |

# Описание предмета проектирования

Кронштейн (нем. Kragstein — выступающий камень) — опорная деталь или конструкция, служащая для крепления на вертикальной плоскости (стене или колонне) выступающих или выдвинутых в горизонтальном направлении частей.

Модель кронштейна представлена на рисунке 2.1.



Рисунок 2.1– Кронштейн

Параметры кронштейна:

1. ширина пластины кронштейна A (70мм-100мм) (рисунок 2.2);
2. длина пластины кронштейна B (100мм-130мм) (рисунок 2.2);
3. внешний диаметр трубки C (50мм-70мм) (рисунок 2.2);
4. радиус крепежного отверстия E (5мм-12мм) (рисунок 2.3);
5. высота крепежного отверстия D (7мм-15мм) (рисунок 2.3);
6. высота боковой стенки F (20мм-30мм) (рисунок 2.3);
7. внешний диаметр трубки должен быть меньше, чем расстояние между боковыми стенками;
8. крепежное отверстие не должно выходить за пределы боковой стенки.



Рисунок 2.2 – Чертеж с обозначением, ширины пластины кронштейна, длины пластины кронштейна, внешнего диаметра трубки



Рисунок 3 – Чертеж с обозначением радиуса малого отверстия, высоты крепежного отверстия, высоты боковой стенки

# Проект программы

# Описание технических и функциональных аспектов проекта

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 [4].

При использовании UML была простроена диаграмма классов.

# Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы и их взаимосвязи между ними, их коопераций, атрибутов (полей), методов [4].

На рисунке 3.2 представлена диаграмма классов.

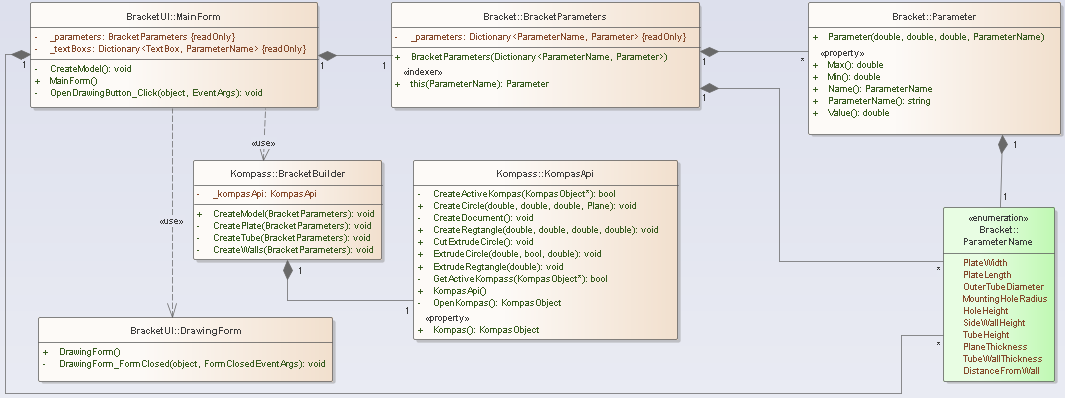


Рисунок 3.2 – Диаграмма классов

Класс MainForm содержит в себе экземпляры классов BracketParameters и BracketBuild. BracketParameters в себе список экземпляров класса Parameter, где хранятся значения параметра. BracketBuild обладает методами для построения моделей. Класс KompassApi содержит в себе методы для работы с САПР Компас 3D. В перечислении ParameterName хранятся именования параметров. Класс DrawingForm показывает общий чертеж.

На рисунке 3.3 показана итоговая диаграмма классов.

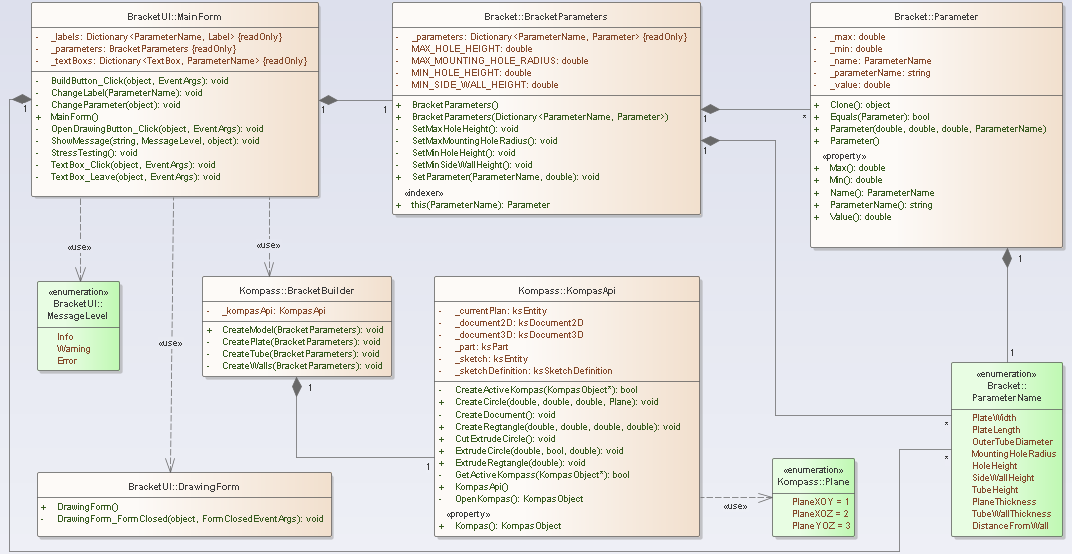


Рисунок 3.3 – Итоговая диаграмма классов

Добавлены два перечисления MessageLevel, здесь находится информация об уровне сообщения, и Plane, где хранятся именования плоскостей.

В классе BracketParameters добавлены методы SetMinHoleHeight, SetMaxHoleHeight, SetMinSideWallHeight, SetMaxMountingHoleRadius которые отвечают за установку граничных значений у параметров. Также в классе MainForm были созданы обработчики событий для получения данных от пользователя. В классе KompasApi были добавлены поля.

# Макет пользовательского интерфейса

Пользовательский интерфейс представляет собой форму для ввода параметров. При нажатии на кнопку «Построить» строится 3D-модель формы для льда. В правой части интерфейса расположены подсказки в виде картинок. На рисунке 3.4 представлен макет пользовательского интерфейса.

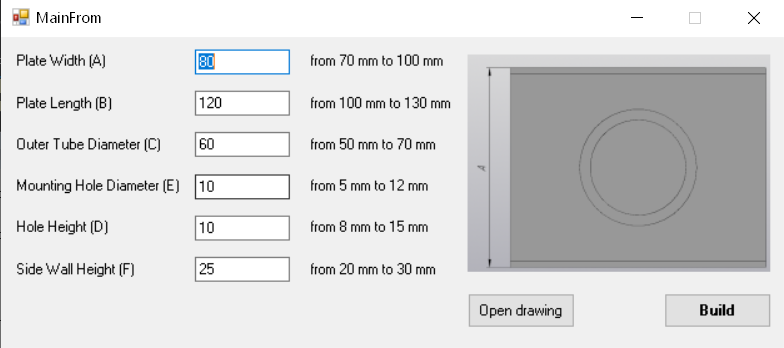


Рисунок 3.4 – Макет пользовательского интерфейса

После того как пользователь переходит к редактированию другого параметра или нажимает на кнопки «Build» или «Open drawing», в случае некорректного ввода всплывет окно с ошибкой, где будет написано причина, по которой возникла ошибка. При нажатии на кнопку «Build» будет запущена программа Компас и в ней построится модель по заданным параметрам. При нажатии на кнопку «Open drawing» в новом окне откроется чертеж модели, сама кнопка заблокируется, но после закрытия окна с чертежом вновь разблокируется. Окно с ошибками показано на рисунке 3.6, окно с чертежом показано на рисунке 3.7.

На рисунке 3.5 показан итоговый макет пользовательского интерфейса.

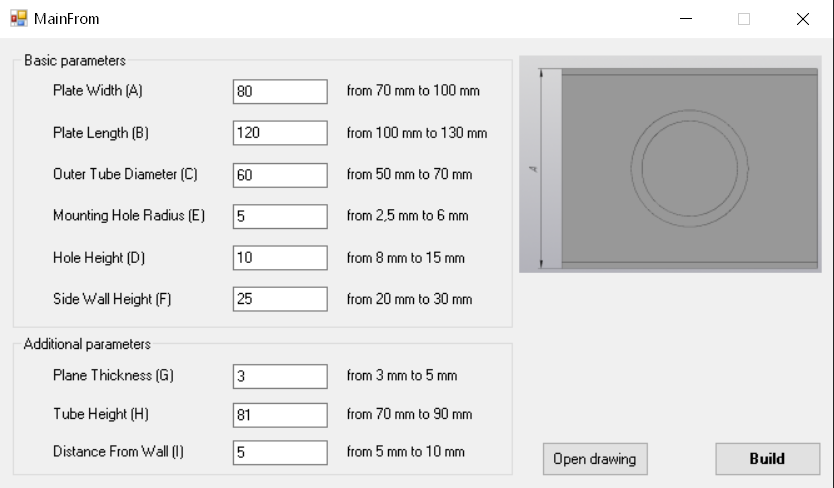


Рисунок 3.5 – Итоговый макет пользовательского интерфейса



Рисунок 3.6 – Окно с ошибкой



Рисунок 3.7 – Окно с чертежом

# Тестирование программы

# Функциональное тестирование

При функциональном тестировании проверялось корректность работы плагина «Кронштейн», а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами [6]. Проведено тестирование максимальных и минимальных параметров модели.

На рисунке 4.1 показана модель с параметрами, заданными по умолчанию (ширина пластины 80мм, длина пластины 120мм, внешний диаметр трубки 60мм, радиус крепежного отверстия 5мм, высота крепежного отверстия 10мм, высота боковой стенки 25мм, толщина пластины 3мм, высота трубки 81мм, расстояние крепежного отверстия до начала боковой стенки 5мм).

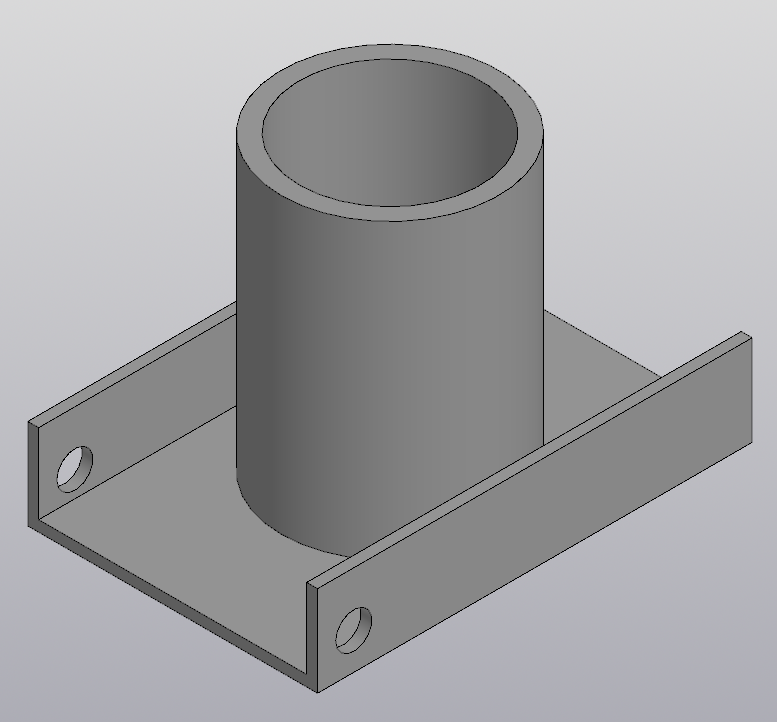


Рисунок 4.1 – Модель с параметрами по умолчанию

На рисунке 4.2 показана модель с минимальными параметрами (ширина пластины 70мм, длина пластины 100мм, внешний диаметр трубки 50мм, радиус крепежного отверстия 2,5мм, высота крепежного отверстия 7мм, высота боковой стенки 20мм, толщина пластины 3мм, высота трубки 70мм, расстояние крепежного отверстия до начала боковой стенки 5мм).

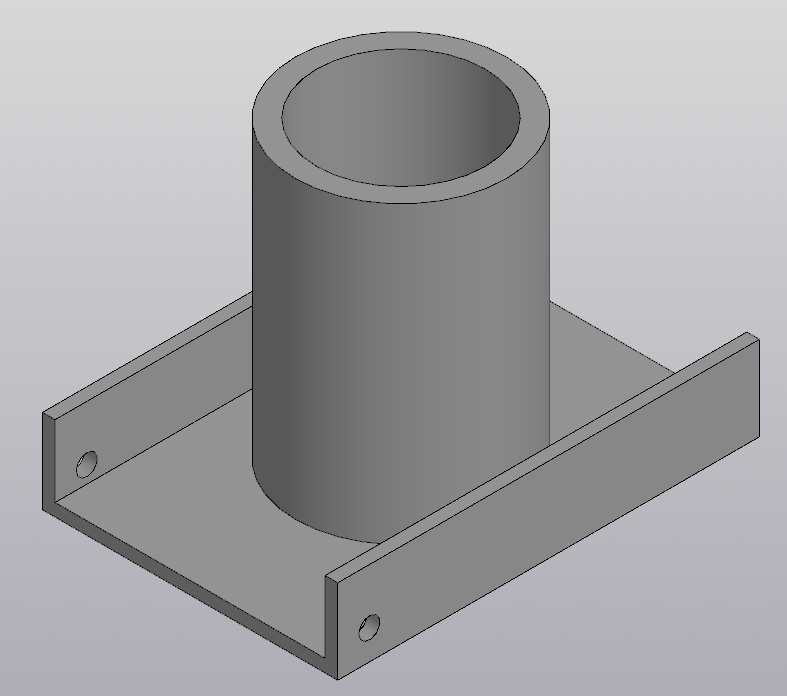


Рисунок 4.2 – Модель с минимальными параметрами

На рисунке 4.3 показана модель с максимальными параметрами (ширина пластины 100мм, длина пластины 130мм, внешний диаметр трубки 70мм, радиус крепежного отверстия 6мм, высота крепежного отверстия 14мм, высота боковой стенки 30мм, толщина пластины 5мм, высота трубки 90мм, расстояние крепежного отверстия до начала боковой стенки 10мм).

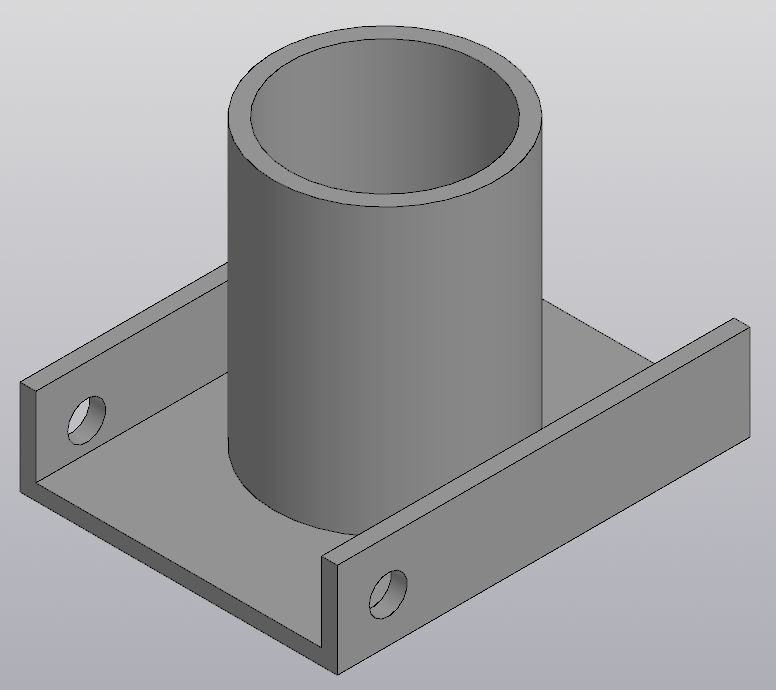


Рисунок 4.3 – Модель с максимальными параметрами

# Модульное тестирование

В целях проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версии 3.13 проведено модульное тестирование [7], проверялись открытые поля и методы. На рисунке 4.4 и 4.5 показаны результаты тестов классов Parameter и BracketParameters.

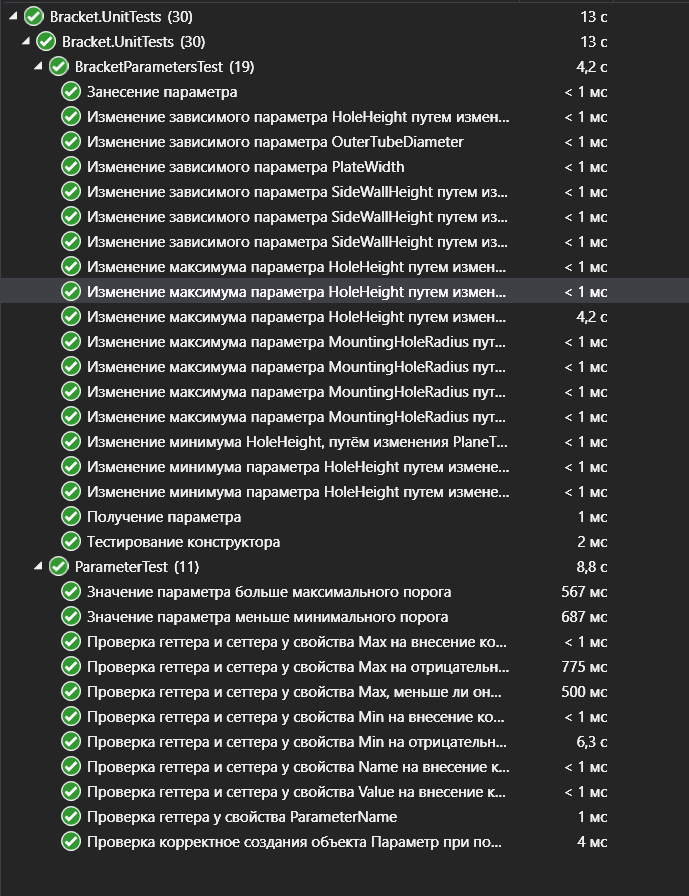


Рисунок 4.4 – Результат тестов

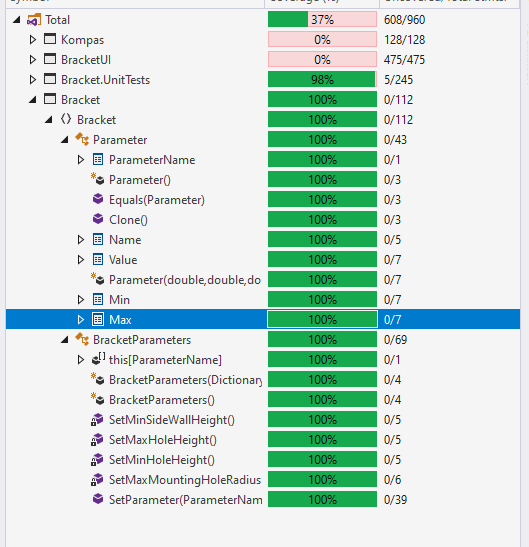


Рисунок 4.5 – Покрытие юнит-тестами

# Нагрузочное тестирование

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

1. ЦП Intel Core i5-9300HF 2.40 ГГц;
2. ОЗУ объемом 16 ГБ;
3. Графический процессор объемом 6 ГБ.

Для нагрузочного тестирования создан метод с бесконечным циклом построения детали, представленный на рисунке 4.6. Для измерения времени использовался класс Stopwatch.



Рисунок 4.6 – Метод для стресс-теста

Во время тестирования было построено 427 моделей кронштейна за 50 минут. Модели были построены с параметрами по умолчанию. На протяжении тестирования загруженность процессора была в районе 30%, с временными скачками до 50%.

На рисунках 4.7 и 4.8 показаны результаты тестирования программы.

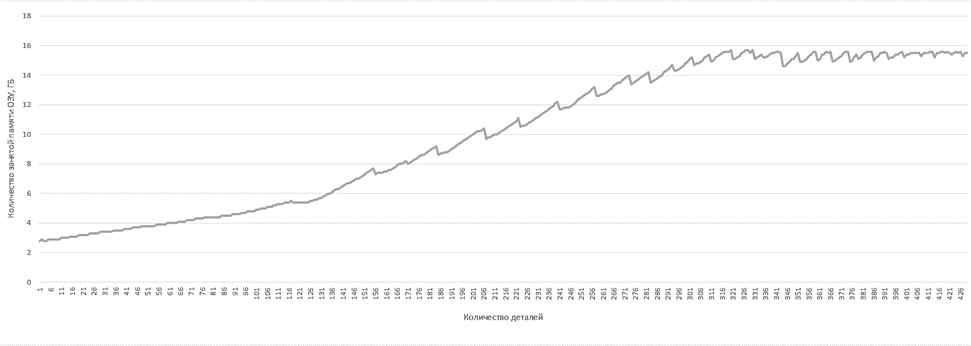


Рисунок 4.7 – График зависимости количества потребляемой оперативной памяти от количества деталей для модели с параметрами по умолчанию

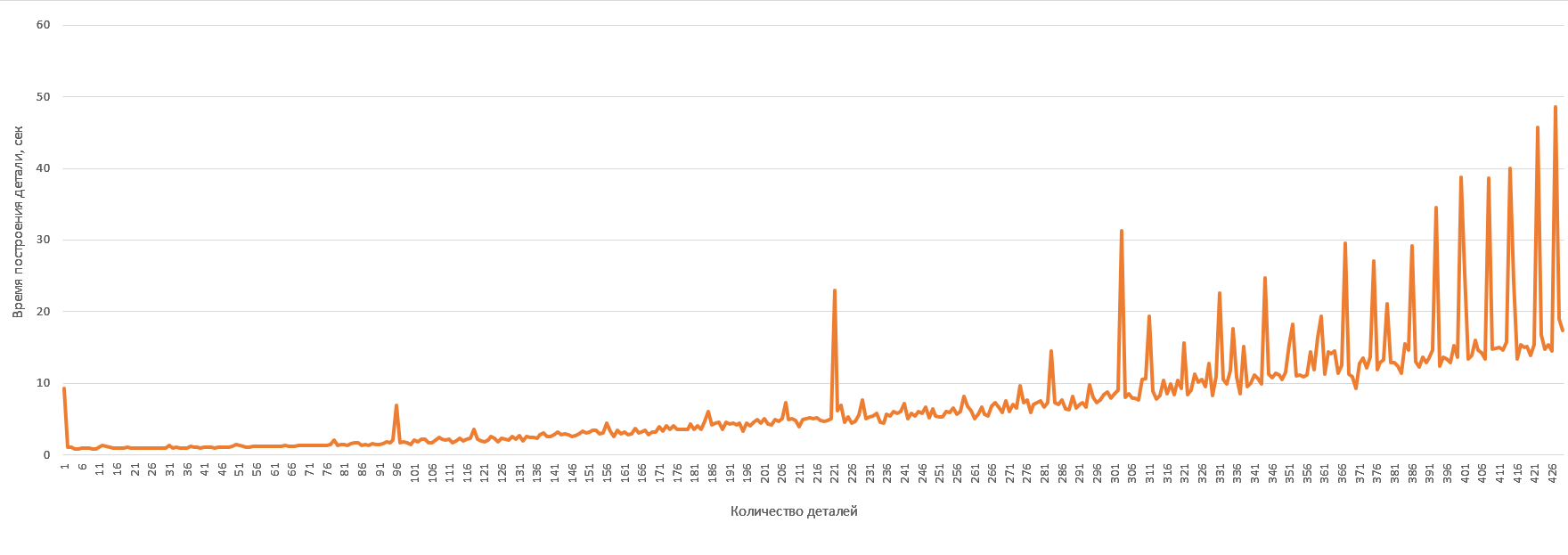


Рисунок 4.8 – График зависимости времени построения одной детали

от количества деталей для модели с параметрами по умолчанию

Исходя из приведенных графиков на рисунках 4.7 и 4.8, можно увидеть, что чем больше построено моделей, то тем больше занимается оперативной памяти и тем медленнее стоится модель. Оперативная память заполняется пока есть место, как только место начинает заканчиваться система пытается очистить её. От этого и получается задержки в построении детали. Из этого следует, что построение множества моделей не имеет смысла.

# Заключение

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

Плагин выполнен как отдельное приложение, которое подключается к САПР Компас 3D и с помощью его API строит модель. Заявленная функциональность реализована полностью, включая дополнительную функциональность. Тестирование программы проведено в три этапа: функциональное, модульное и нагрузочное.

# Список использованной литературы

1. САПР — Википедия. [Электронный ресурс]. — Режим доступа: [https://ru.wikipedia.org/wiki/Система\_автоматизированного\_проектирования](https://ru.wikipedia.org/wiki/Система_автоматизированного_проектирования%20) (дата обращения 22.10.2021).
2. Официальный сайт Kompas. [Электронный ресурс] – Режим доступа: <https://kompas.ru/kompas-3d/about/> (дата обращения 26.10.2021);
3. API – Библиотека обучающей и информационной литературы. [Электронный ресурс] – Режим доступа: <http://www.k2x2.info/kompyutery_i_internet/kompas_3d_v10_na_100/p9.php> (дата обращения 26.10.2021);
4. Мартин Ф. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. Изд: Символ-Плюс,2011, с.192 (3-е издание).
5. API — Википедия. [Электронный ресурс]. — Режим доступа: <https://ru.wikipedia.org/wiki/API> (дата обращения 22.10.2021).
6. Функциональное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 3.11.2021).
7. Юнит-тестирование для чайников [Электронный ресурс]. – URL: <https://habr.com/ru/post/169381/> (дата обращения: 13.12.2021).
8. Нагрузочное тестирование: с чего начать и куда смотреть [Электронный ресурс]. – URL: <https://habr.com/ru/company/jugru/blog/329174/> (дата обращения: 28.12.2021).

# Приложение А

Таблица А.1 – Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| TestConstructor() | – | Тестирование конструктора |
| GetParameter\_ResultCorrect() | – | Получение параметра |
| SetParameter\_ResultCorrect() | – | Занесение параметра |
| ChangePlateWidthParameter\_  SetValueOuterTubeDiameter\_  ResultCorrect() | – | Изменение минимума параметра PlateWidth изменением OuterTubeDiameter |
| ChangeOuterTubeDiameterParameter\_  SetValuePlatreWidth\_  ResultCorrect() | – | Изменение максимума параметра OuterTubeDiameter изменением PlateWidth |

|  |  |  |
| --- | --- | --- |
| Тестовый метод | Входные параметры | Описание тестового случая |
| ChangeDependentParameter  Minimum\_  SetValue\_  ResultCorrect(double expected, ParameterName parameter, double value, ParameterName changeableParametr) | 9, ParameterName.  MountingHoleRadius, 6, ParameterName.  HoleHeight | Изменение минимума параметра HoleHeight путем изменения MountingHoleRadius на максимальное значение |
| 7, ParameterName.  MountingHoleRadius, 2.5, ParameterName.  HoleHeight | Изменение минимума параметра HoleHeight путем изменения MountingHoleRadius на минимальное значение |
| 25, ParameterName.  HoleHeight, 15, ParameterName.  SideWallHeight | Изменение минимума параметра SideWallHeight путем изменения HoleHeight |
| 21, ParameterName.  MountingHoleRadius, 6, ParameterName.  SideWallHeight | Изменение минимума параметра SideWallHeight путем изменения MountingHoleRadius |
| 8, ParameterName.  PlaneThickness, 3, ParameterName.  HoleHeight | Изменение минимума HoleHeight, путём изменения PlaneThickness |
| 10, ParameterName.  PlaneThickness, 5, ParameterName.  HoleHeight | Изменение минимума HoleHeight, путём изменения PlaneThickness |