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 "НАПРАВЛЯЮЩАЯ" ДЛЯ САПР КОМПАС 3D

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
на разработку плагина моделирования мебельной направляющей для системы «Компас 3D V20»

Выполнил:

студент гр. 588-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Матинин А.С.

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

Руководитель:

к.т.н., доцент каф. КСУП

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

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

Томск 2022

+Содержание

[1 Описание САПР 3](#_Toc93907145)

[1.1 Описание программы 3](#_Toc93907146)

[1.2 Описание API 3](#_Toc93907147)

[1.3 Обзор аналогов 9](#_Toc93907148)

[2 Описание предмета проектирования 11](#_Toc93907149)

[3 Проект программы 12](#_Toc93907150)

[3.1 Описание технических и функциональных аспектов проекта 12](#_Toc93907151)

[3.2 Диаграмма классов 12](#_Toc93907152)

[3.3 Макет пользовательского интерфейса 15](#_Toc93907153)

[4 Тестирование программы 17](#_Toc93907154)

[4.1 Функциональное тестирование 17](#_Toc93907155)

[4.2 Модульное тестирование 18](#_Toc93907156)

[4.3 Нагрузочное тестирование 20](#_Toc93907157)

[Заключение 25](#_Toc93907158)

[Список литературы 26](#_Toc93907159)

# 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 комплекса технических, программных и других средств автоматизации его деятельности [1].

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 [2].

## Описание API

API (англ. Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой[3].

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 (на интерфейс приложения API 5) 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.

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) |
| 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 КОМПАС |

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) |

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 – | bool | Создать документ-модель (деталь или сборку) | |
|  | видимый режим),  typeDoc – тип документа (true – деталь, false – сборка). |  | |  |
| 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\_baseExtrusion | Базовая операция выдавливания | ksBaseExtrusionDefinition |

## Обзор аналогов

БАЗИС – комплексная система автоматизации проектирования, технологической подготовки производства и реализации корпусной мебели.

БАЗИС-Мебельщик - основной модуль системы БАЗИС. Он предназначен для создания изделий корпусной мебели любой сложности, с возможностью автоматического получения полного комплекта чертежей и спецификации. Применение модуля БАЗИС-Мебельщик позволяет сократить время проектирования и технологической подготовки производства изделий в 10-15 раз по сравнению с ручной работой при значительном сокращении количества субъективных ошибок. [4]

На рисунках 1.1 – 1.3 представлено окно программы «Базис-мебельщик».



Рисунок 1.1 – Главное окно программы



Рисунок 1.2 – Окно выбора фурнитуры



Рисунок 1.3 – Окно для редактирования некоторых параметров направляющей

# Описание предмета проектирования

Предметом проектирования является модель направляющей.

Параметры направляющей:

А. Длина направляющей (50мм – 150мм)

Б. Ширина направляющей (10мм – 30мм)

В. Толщина направляющей (5мм – 20мм)

Г. Длина хода крепления (15мм – 90мм; больше чем 5 \* ширина хода крепления)

Д. Ширина хода крепления (3мм-20мм; больше чем 0.3 \* Ширина направляющей и меньше чем 0.7 \* Ширина направляющей)

Е. Диаметр отверстия для крепления к плоскости (2мм -20мм)

Ж. Угол наклона направляющей (65° - 270°)

На рисунке 2.1 представлен 3D модель с выноской размеров.



Рисунок 2.1 – 3D модель с выноской размеров

# 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. [5]

При использовании UML были простроена диаграмма классов.

## Диаграмма классов

Диаграмма классов показывает набор классов, интерфейсов и коопераций, а также их связи. Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. [5]

На рисунке 3.1 представлена первая версия диаграммы классов.

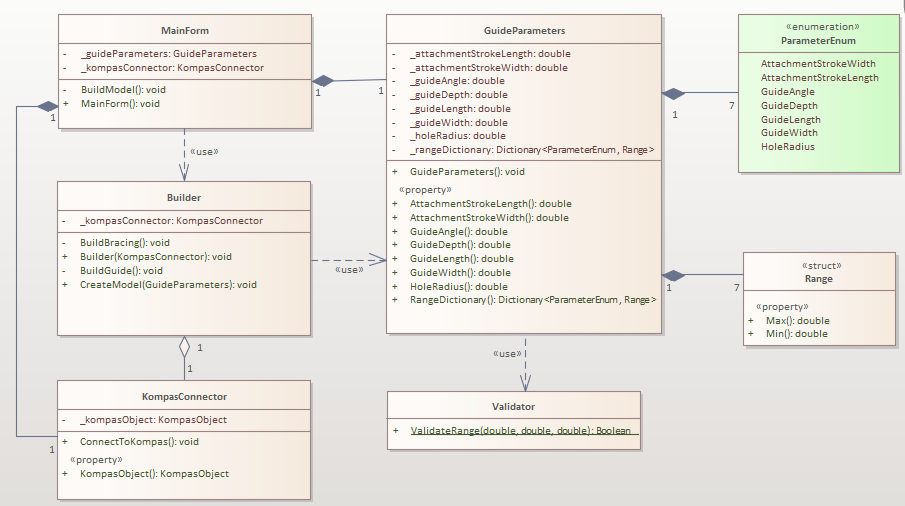


Рисунок 3.1 – Первая версия UML диаграмма классов

Класс MainForm хранит в себе экземпляры классов KompasConnector и GuideParameters. Класс KompasConnector обеспечивает подключение к «Компас 3D». Класс Builder включает методы для постройки направляющей. Класс Guide Parameters хранит в себе информацию о параметрах направляющей, а также изменяет их при необходимости или выдает исключения об ошибках. Класс Validator имеет метод, проверяющий вхождение параметра в нужный диапазон значений.

Финальная версия диаграммы классов представлена на рисунке 3.2.

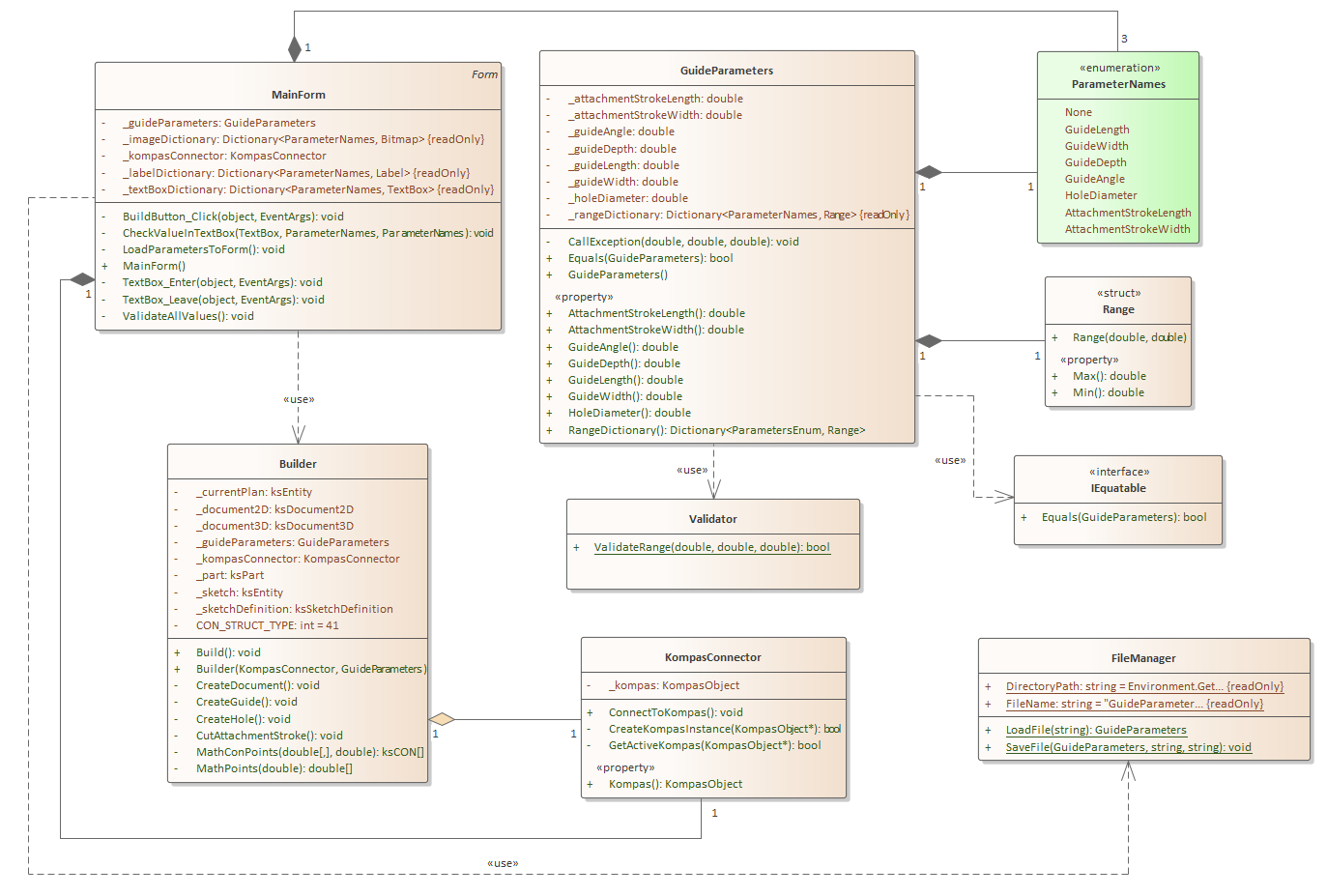


Рисунок 3.2 – Финальная версия UML диаграмма классов

В класс MainForm были добавлены словари с элементами формы, для удобного поиска нужного элемента по названию параметра. Добавлены методы для событий потери и наведения фокуса. Добавлены методы проверки всех введенных параметров и каждого отдельно. Так же добавлен метод для выведения параметров на форму.

Был добавлен класс FileManager отвечающий за загрузку и сохранение данных в json формате.

В класс GuideParameters был добавлен метод, вызывающий исключение с текстом ошибки.

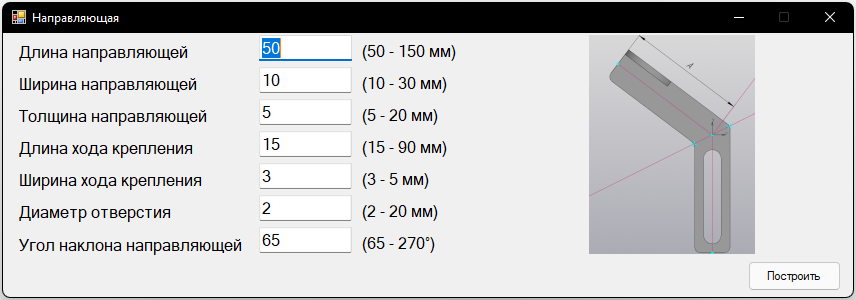
В классе KompasConnector изменены методы подключения к api компаса.

Для класса Builder были добавлены вспомогательные методы для постройки модели и переменные для хранения объектов api компаса.

## Макет пользовательского интерфейса

Макет пользовательского интерфейса представляет собой форму для ввода параметров направляющей (рисунок 3.3). Построение модели осуществляется путем нажатия на кнопку «Построить». Если в поле были введены некорректные данные, при потере фокуса, поле с некорректными данными окрасится в красный и на экране высветиться сообщение с информацией об ошибке (рисунок 3.4-3.5). Пока все параметры не будут введены корректно, кнопка «Построить» будет неактивна. При нажатии на поле, в правой части окна будет выводится изображение, показывающее изменяемый параметр.

Так же после корректного ввода, параметры сохраняются в json файл. При последующем запуске, параметры загружаются в плагин. Если json файл испорчен или не существует, параметры задаются по умолчанию минимальными.

Рисунок 3.3 – Макет пользовательского интерфейса

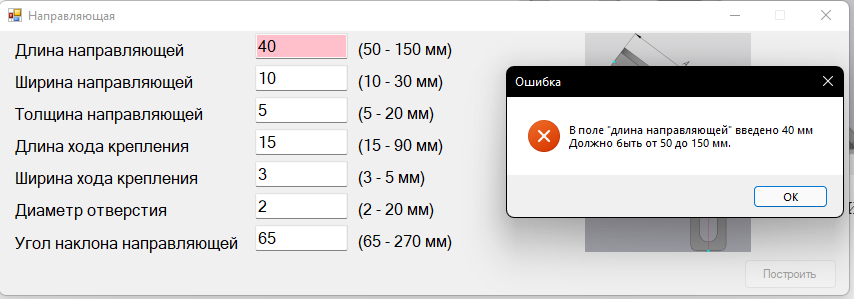


Рисунок 3.4 – Макет пользовательского интерфейса с введенными данными, выходящими за диапазон допустимых значений

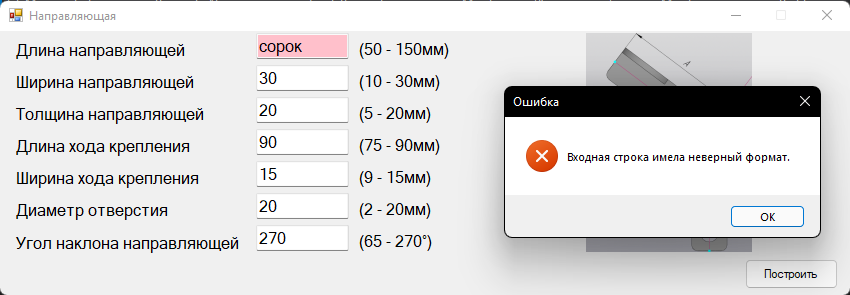


Рисунок 3.5 – Макет пользовательского интерфейса с введенными недопустимыми символами

# Тестирование программы

## Функциональное тестирование

Во время функционального тестирования [6] выполнялась проверка работы плагина, а именно, соответствие построенной модели с веденными параметрами.

На рисунке 4.1 представлена модель направляюще с минимальными параметрами (длина направляющей – 50 мм, ширина направляюще – 10 мм, толщина направляющей – 5 мм, длина хода крепления – 15 мм, ширина хода крепления – 3 мм, диаметр отверстия – 2 мм, угол наклона направляющей – 65°)

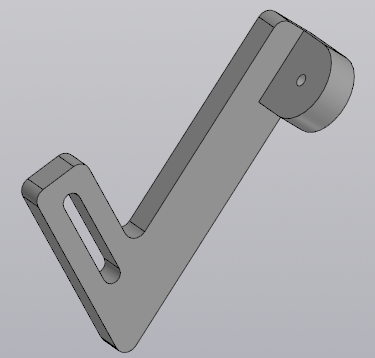


Рисунок 4.1 – Модель с минимальными параметрами

На рисунке 4.2 представлена модель направляюще с максимальными параметрами (длина направляющей – 150 мм, ширина направляюще – 310 мм, толщина направляющей – 20 мм, длина хода крепления – 90 мм, ширина хода крепления – 15 мм, диаметр отверстия – 20 мм, угол наклона направляющей – 270°)

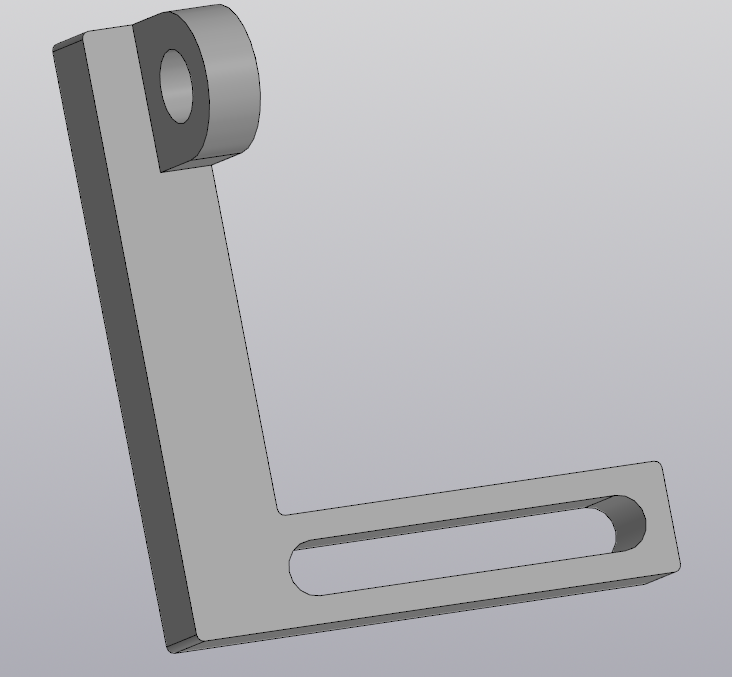


Рисунок 4.2 – Модель с максимальными параметрами

## Модульное тестирование

В целях проверки бизнес логики плагина было проведено модульное тестирование. Тестирование проводилось при помощи тестового фреймворка NUnit [7], проверялись открытые поля и методы. На рисунке 4.3 и 4.4 показаны результаты тестов классов GuideParameters, Range, Validator и FileManager.

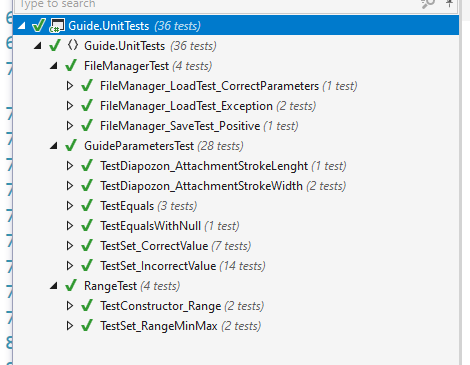


Рисунок 4.3 – Результат тестов

Описание тестов приведено в приложении А.

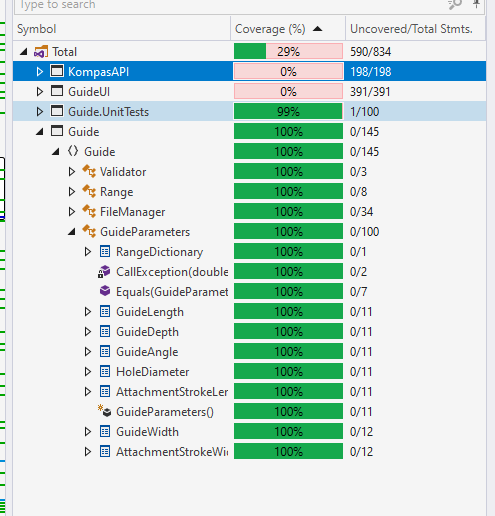


Рисунок 4.4 – Покрытие юнит-тестами

## Нагрузочное тестирование

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

* ЦП AMD Ryzen 5 4600H with Radeon Graphics 3.00 ГГц;
* 16 ГБ ОЗУ;

Для нагрузочного тестирования создан метод с бесконечным циклом построения детали, представленный на рисунке 4.5. Для измерения времени использовался класс Stopwatch.

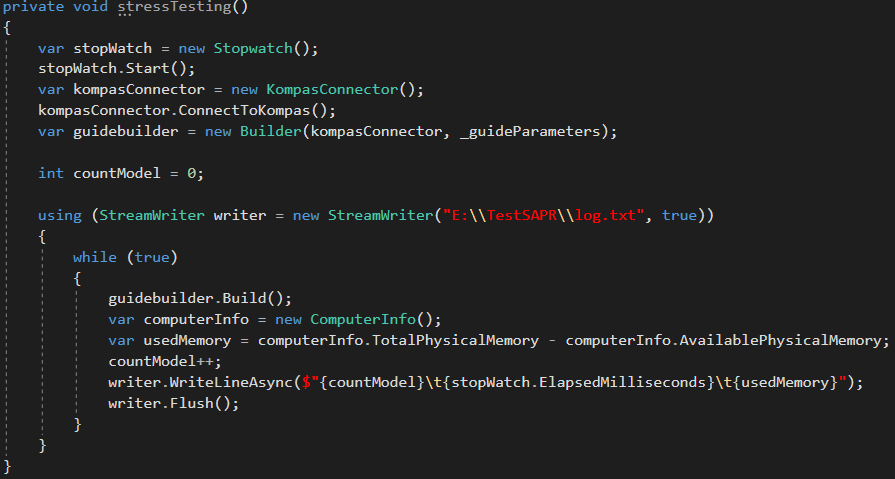


Рисунок 4.5 – Реализация зацикленного перестроения модели

Во время тестирования было построено 228 моделей кронштейна за 21 минуту. Модели были построены с минимальными параметрами. На протяжении тестирования процессор работал на частоте 3.6-3.8 ГГц, а его загруженность была в районе 20-25%.

На рисунках 4.6 и 4.7 показаны результаты тестирования программы.

Рисунок 4.6 – График зависимости количества потребляемой оперативной памяти от количества построенных моделей

Рисунок 4.7 – 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.

# Заключение

В ходе выполнения лабораторных работ были изучены предметная область проектирования, предмет проектирования, аналоги предмета проектирования, API, функциональное и нагрузочное тестирование и на основании полученных данных были спроектированы UML диаграммы классов, разработан плагин для создания 3D моделей «Направляющая» в САПР Компас 3D v20 и проведено функциональное и нагрузочное тестирование плагина.

# Список литературы

1. САПР — Википедия. [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Система\_автоматизированного\_проектирования (дата обращения 22.10.2021).

2. КОМПАС-3D. Официальный сайт САПР КОМПАС. [Электронный ресурс]. Режим доступа: https://kompas.ru (дата обращения 22.10.2021).

3. API – Википедия. [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/API (дата обращения 22.10.2021).

4. Базис-центр – Комплекс программ для проектирования и продажи мебели. — Режим доступа: https://www.bazissoft.ru (дата обращения 03.11.2021).

5. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: ДМК Пресс, 2006. – 496 с.

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 A.l – Тестовые сценарии

|  |  |  |
| --- | --- | --- |
| Название теста | Входные параметры | Описание теста |
| FileManager\_LoadTest\_CorrectParameters() |  | Тест загрузки параметров с корректными значениями |
| public void FileManager\_LoadTest\_Exception(string path) | @"TestData\CorruptedGuideParameters.json" | Тест загрузки некорректных файлов |
| @"TestData\NonExistentParameters.json" |
| public void FileManager\_SaveTest\_Positive() |  | Позитивный тест созранения параметров |
| public void TestSet\_CorrectValueParameterNames ParameterNames, double value) | ParameterNames.GuideLength,60 | Тест на введение корректных параметров |
| ParameterNames.GuideWidth, 11 |
| ParameterNames.GuideDepth, 6 |
| ParameterNames.AttachmentStrokeLength, 16 |
| ParameterNames.AttachmentStrokeWidth, 4 |
| ParameterNames.HoleDiameter, 3 |
| ParameterNames.GuideAngle, 100 |

Продолжение таблица А.1

|  |  |  |
| --- | --- | --- |
| Название теста | Входные параметры | Описание теста |
| TestSet\_IncorrectValue(ParameterNames ParameterNames, double value) | ParameterNames.GuideLength, 30 | Тест на введение некорректных параметров |
| ParameterNames.GuideLength, 160 |
| ParameterNames.GuideWidth, 5 |
| ParameterNames.GuideWidth, 60 |
| ParameterNames.GuideDepth, 1 |
| ParameterNames.GuideDepth, 60 |
| ParameterNames.AttachmentStrokeLength, 1 |
| ParameterNames.AttachmentStrokeLength, 100 |
| ParameterNames.AttachmentStrokeWidth, 1 |
| ParameterNames.AttachmentStrokeWidth, 60 |
| ParameterNames.HoleDiameter, 1 |
| ParameterNames.HoleDiameter, 60 |
| ParameterNames.GuideAngle, 60 |
| ParameterNames.GuideAngle, 300 |

Продолжение таблицы А.1

|  |  |  |
| --- | --- | --- |
| Название теста | Входные параметры | Описание теста |
| public void TestDiapozon\_AttachmentStrokeWidth(bool isMaxBorder) | true | Тест на изменение диапазона значений для AttachmentStrokeWidth |
| false |
| TestDiapozon\_AttachmentStrokeLenght() |  | Тест на изменение диапазона значений для AttachmentStrokeLenght |
| public void TestEquals(bool sameParameters, bool equal) | true,true | Тест для метода Equals |
| false, true |
| true, false |
| TestEqualsWithNull() |  | Тест для метода Equals с значение null |
| TestConstructor\_Range(bool isMaxBorder) | true | Тест для конструктора структуры Range |
| false |
| TestSet\_RangeMinMax(bool isMaxBorder) | true | Тест для максимальных и минимальных значений структуры Range |
| false |