Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждения

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:

студент гр. 586-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Суберляк Е. М.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Проверил:

к.т.н., доцент каф. КСУП

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

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Томск 2020 г.

**РЕФЕРАТ**

Пояснительная записка 26 с., 13 рис., 1 таблица, 8 источников.

КОМПАС-3D, ПЛАГИН, МОДЕЛЬ ЭЛЕКТРИЧЕСКОГО ЧАЙНИКА, АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ, САПР.

Целью данной работы является разработка плагина, автоматизирующего построение модели электрического чайника, для системы автоматизированного проектирования КОМПАС-3D v18.1, с помощью интегрированной среды разработки Visual Studio Сommunity 2019.

В процессе работы изучены основные функции системы автоматизированного проектирования КОМПАС-3D v 18.1

В результате работы был создан плагин, автоматизирующий построение электрического чайника.

Отчёт по пояснительной записке выполнен в текстовом редакторе Microsoft Word 2016.

Оглавление

[**РЕФЕРАТ** 2](#_Toc40478832)

[1 Введение 4](#_Toc40478833)

[2 Постановка и анализ задачи 5](#_Toc40478834)

[2.1 Описание предмета проектирования 5](#_Toc40478835)

[2.2 Выбор инструментов и средств реализации 6](#_Toc40478836)

[2.3 Назначение плагина 7](#_Toc40478837)

[2.4 Обзор аналогов 7](#_Toc40478838)

[2.4.1 Плагин PDF для САПР КОМПАС-3D 7](#_Toc40478839)

[3 Проект программы 9](#_Toc40478840)

[3.1 Описание технических и функциональных аспектов проекта 9](#_Toc40478841)

[3.2 Диаграмма вариантов использования (Use Cases) 9](#_Toc40478842)

[3.3 Диаграмма классов 11](#_Toc40478843)

[3.3 Макет пользовательского интерфейса 13](#_Toc40478844)

[4 Тестирование 16](#_Toc40478845)

[4.1 Функциональное тестирование 16](#_Toc40478846)

[4.2 Модульное тестирование 18](#_Toc40478847)

[4.3 Нагрузочное тестирование 21](#_Toc40478848)

[Заключение 25](#_Toc40478849)

[Список литературы 26](#_Toc40478850)

# 1 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.

Практическая реализация методов и идей автоматизированного моделирования происходит в рамках систем автоматизированного проектирования (САПР). Однако мало создать высокопроизводительные современные САПР. Надо уметь их эффективно использовать. Для этого нужны квалифицированные инженеры-пользователи САПР. В рамках современного «компьютеризированного» общества инженер любой специальности, занимающийся разработкой технических устройств, должен уметь использовать средства автоматизированного проектирования. Это позволяет повысить эффективность моделирования, улучшить его качество, снизить материальные затраты и уменьшить число разработчиков.

Целью данной работы является разработка плагина, автоматизирующего построение модели «Электрический чайник» для системы автоматизированного проектирования «Компас-3D» v 18.1 [1], с помощью интегрированной среды разработки Visual Studio Сommunity 2019 [3].

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.

# 2 Постановка и анализ задачи

В рамках лабораторных работ в соответствии с технически заданием требовалось разработать плагин, который на основе входных параметров, интегрируя с системой КОМПАС-3D [1], строит модель «Электрический чайник». Необходимо чтобы плагин позволял создавать новый проект, а также менять входные параметры формы такие как диаметр корпуса, высота корпуса, длина носика, ширина носика, высота носика, размер ручки, цвет корпуса, цвет ручки и носика.

# 2.1 Описание предмета проектирования

Электрический чайник — прибор для нагревания питьевой воды, работающий на [электричестве](https://dic.academic.ru/dic.nsf/ruwiki/1308) [2].

Параметры чайника:

* радиус чайника L (от 100 до 140 мм);
* высота чайника H (от 150 до 200 мм);
* длина носика F (от 20 до 25 мм);
* цвет чайника;
* цвет ручки чайника;
* высота носика: D = (1/5)\*L;
* ширина носика: S = (1/5)\*L;
* размер ручки R (от 95 до 125 мм);

Пример модели приведен ниже, на рисунке 2.1.

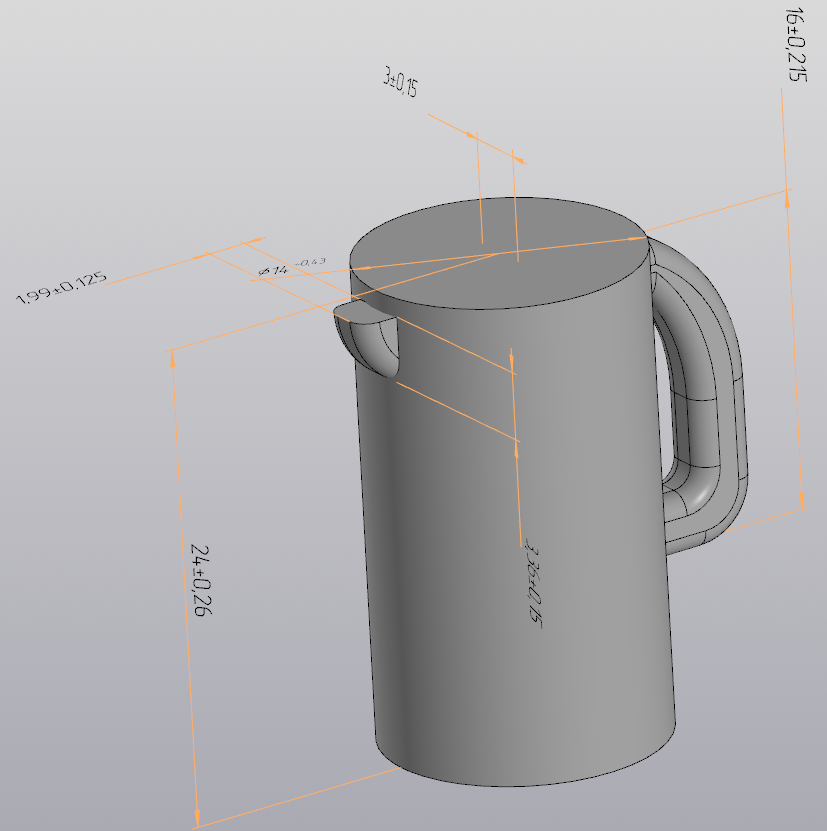


Рисунок 2.1 – Модель электрического чайника

# 2.2 Выбор инструментов и средств реализации

В связи с требованием технического задания программа выполнена на языке C# в среде Microsoft Visual Studio 2019 [3], с использованием .NET Framework 4.6.1, для системы КОМПАС-3D V18.1. Инструментом тестирования и создания модульных тестов был выбран стандартный обозреватель тестов среды Microsoft Visual Studio 2019 с тестовым фреймворком NUnit версии 3.12.0.

Для реализации пользовательского интерфейса использовался WinForms.

Взаимодействие плагина с системой КОМПАС-3D осуществляется посредством интерфейсов, называемых API. В КОМПАС-3D на данный момент существует API двух версий API 5 [4] и API 7. Явных преимуществ между версиями нет, поскольку обе версии реализуют различные функции системы и взаимно дополняют друг друга. Для разработки была выбрана версия API 5, так как для полноценной реализации плагина «Электрический чайник» достаточно методов и свойств интерфейсов API 5.

# 2.3 Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием форм для выпечки разных типов. Благодаря данному расширению, можно наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# Обзор аналогов

# 2.4.1 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.

# 3 Проект программы

# 3.1 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 обрасти 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 [6].

При использовании UML были простроены: диаграмма использования и диаграмма классов.

# 3.2 Диаграмма вариантов использования (Use Cases)

Диаграмма вариантов использования (англ. *use case diagram)* в UML – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, пользователю, а также разработчику обсуждать проектируемую или существующую систему. Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются (для них составляется другой документ) [6].

На рисунке 3.1 представлена изначальная диаграмма вариантов использования.

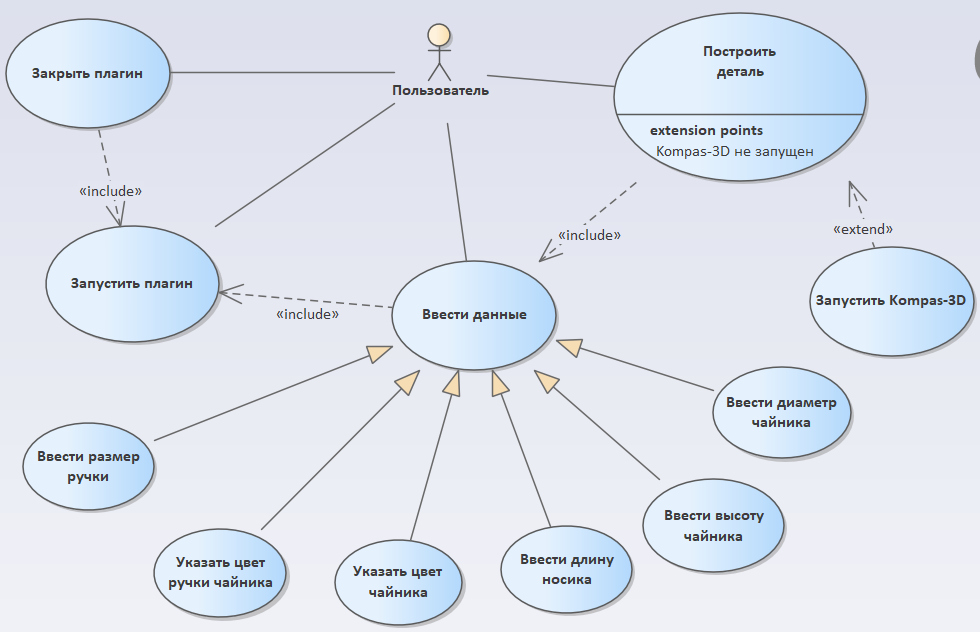


Рисунок 3.1– Изначальная диаграмма вариантов использования.

В результате изменений, связанных с включением в плагин дополнительной функциональности, был добавлен параметр «Выбрать опцию открытой или закрытой ручки» к сущности «Ввести данные».

На рисунке 3.2 представлена измененная диаграмма вариантов использования.

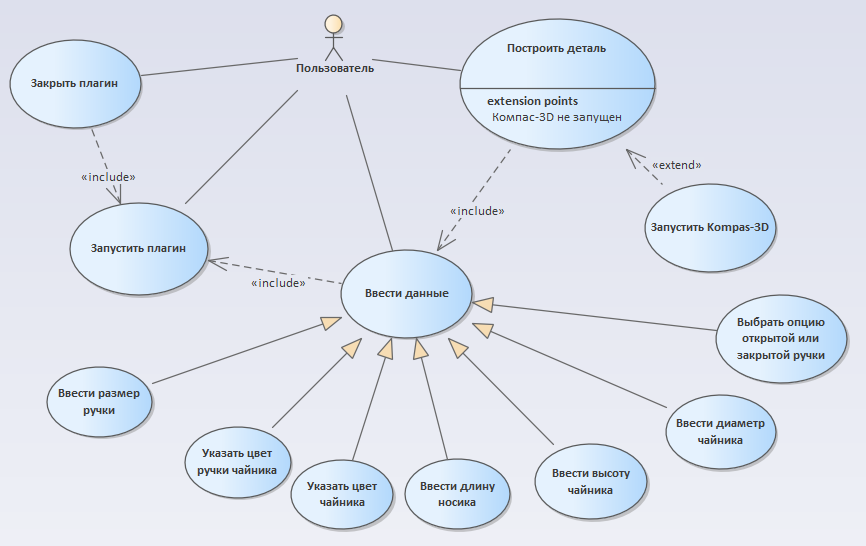


Рисунок 3.2– Измененная диаграмма вариантов использования.

# 3.3 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 [6].

На рисунке 3.3 представлена изначальная диаграмма классов.

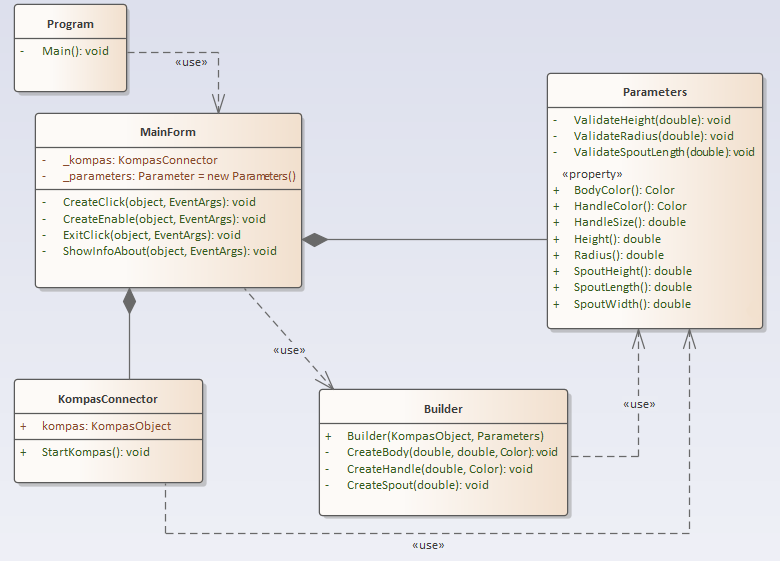


Рисунок 3.3 – Изначальная диаграмма классов

Класс «Program», использует «MainForm» для обработки действий в графическом интерфейсе. «DetailCreator» содержит в себе методы создания 3D модели в «Компас 3D», класс «Validator» проверяет входные данные, введенные в графическом интерфейсе.

Во время реализации плагина диаграмма классов подверглась изменениям.

В класс «Main» было добавлено несколько новых методов:

«ColorValidate\_Values» – Необходим для проверки на корректность параметров;

«AutoFill» – Необходим для автозаполнения полей параметров;

«textbox\_KeyPress» – Обработчик события нажатия на клавишу;

«textbox\_TextChanged» – Обработчик события изменения текста;

При реализации дополнительной функциональности, в класс «Builder» был добавлен ещё один метод:

«BuildClosed» – Постройка закрытой ручки.

На рисунке 3.4 представлена измененная диаграмма классов.

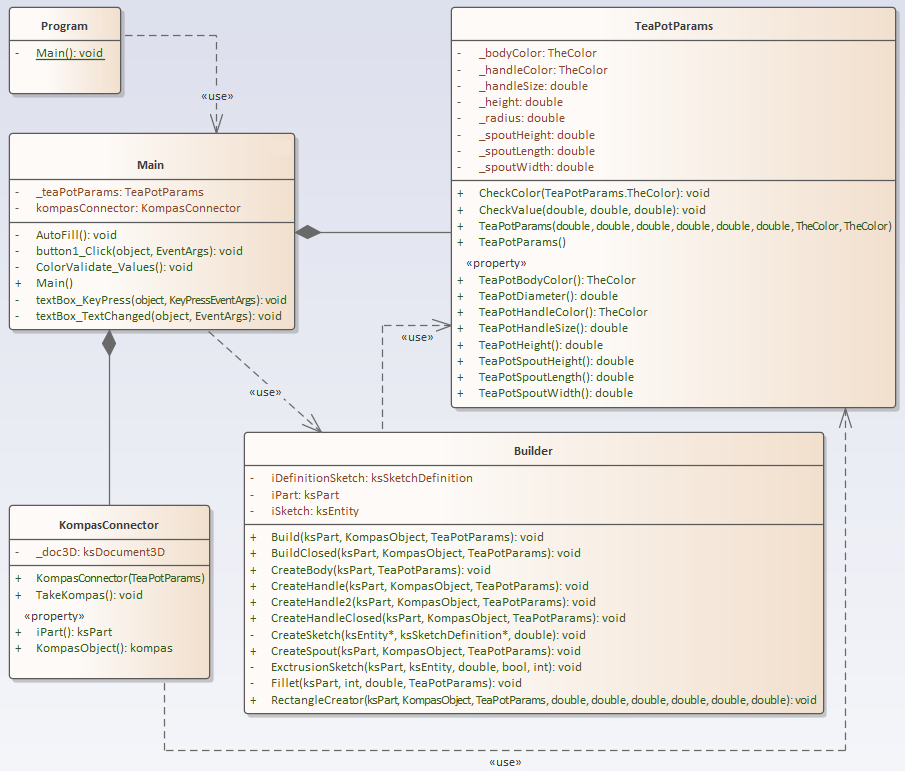
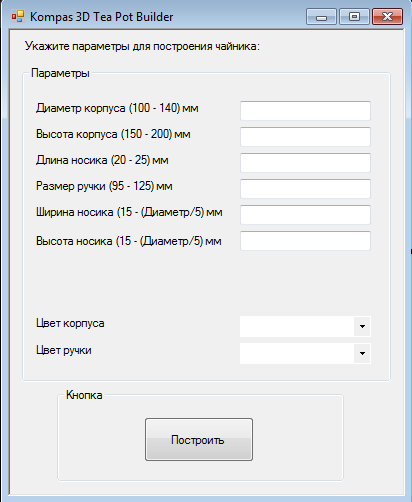


Рисунок 3.4 – Измененная диаграмма классов

# 3.3 Макет пользовательского интерфейса

Макет пользовательского интерфейса представляет собой форму для ввода параметров электрического чайника. Построение модели осуществляется путем нажатия на кнопку «Построить». При попытке ввода недопустимых символов, они не будут вводиться в строку (если необходимо ввести цифры, то невозможно будет ввести другие символы). Если в поля ввода, будут введены некорректные значения или зависимые параметры будут противоречить друг другу, то поля этих параметров будут подсвечиваться красный цветом, а также кнопка «Построить» будет недоступна до тех пор, пока пользователь не введет корректные значения.

На рисунке 3.5 представлен макет изначального пользовательского интерфейса.

  
Рисунок 3.5 – Макет изначального пользовательского интерфейса

Для построения модели «Электрический чайник» необходимо:

* Ввести параметры чайника;
* Запустить построение модели (нажать кнопку «Построить»).

В результате изменений, внесенных заказчиком, была добавлена опция открытой/закрытой ручки.

На рисунке 3.6 представлен макет измененного пользовательского интерфейса.

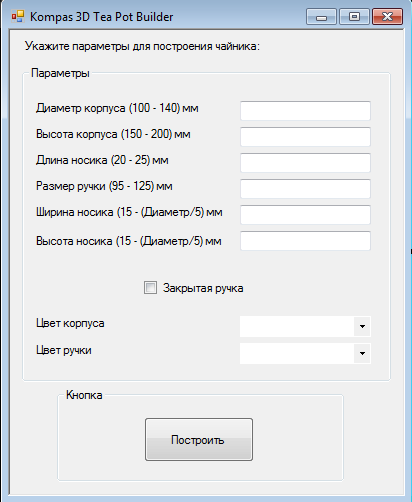


Рисунок 3.6 – Макет измененного пользовательского интерфейса

# 4 Тестирование

# 4.1 Функциональное тестирование

Функциональное тестирование — тестирование функциональности объекта, т.е. правильно ли объект выполняет свои функции. Фактически, выполняется проверка правильности выходных данных при соответствующих входных [7]. Проведем тестирование при минимальных параметрах модели:

* Диаметр корпуса 100 мм;
* Высота корпуса 150 мм;
* Длина носика 20 мм;
* Размер ручки 95 мм;
* Ширина носика 15 мм;
* Высота носика 15 мм;

Модель с минимальными входными параметрами представлена на рисунке 4.1.

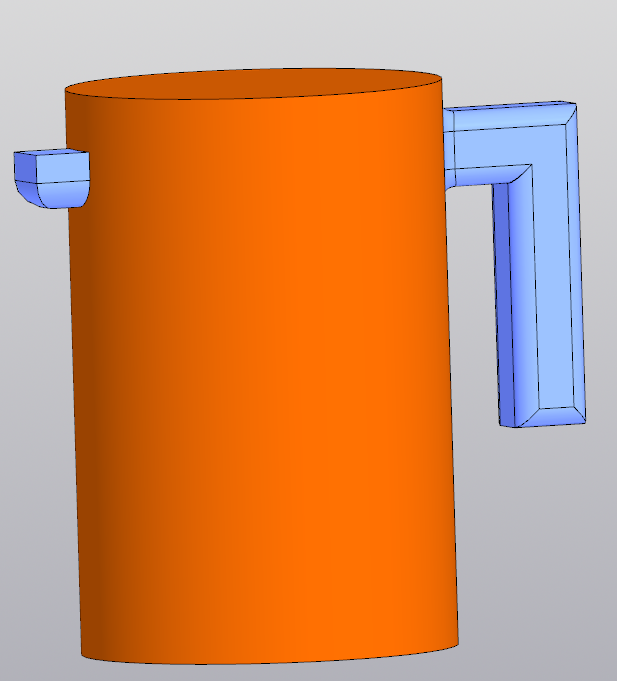


Рисунок 4.1 — Модель чайника с минимальными входными параметрами

Далее проведем тестирование при максимальных параметрах модели:

* Диаметр корпуса 140 мм;
* Высота корпуса 200 мм;
* Длина носика 25 мм;
* Размер ручки 125 мм;
* Ширина носика 28 мм;
* Высота носика 28 мм.

Модель с максимальными входными параметрами представлена на рисунке 4.2.

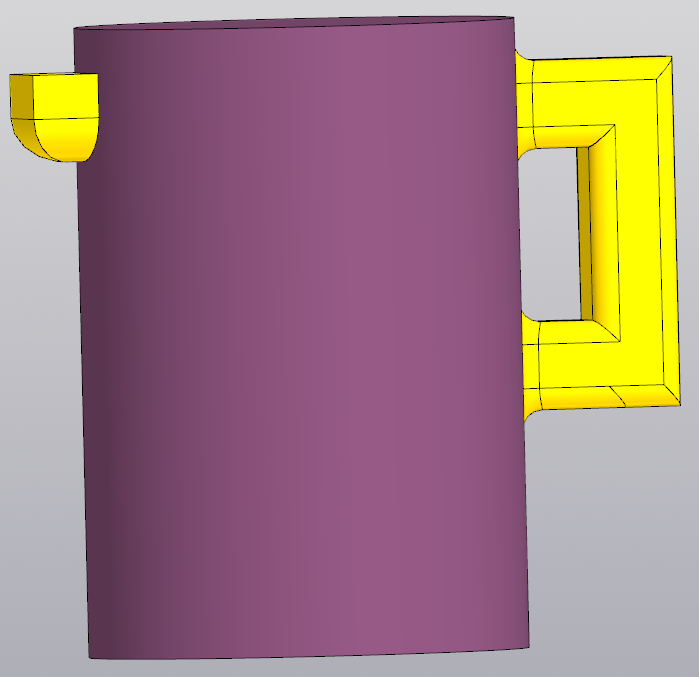


Рисунок 4.2 — Модель чайника с максимальными входными параметрами и закрытой ручкой

При выходе из зоны допустимых значений параметров поле подсвечивается красным цветом, появится сообщение о том, что введены не корректные данные, а также кнопка «Построить» становится недоступной.

Пример приведен на рисунке 4.3. Рамками выделены сообщение об ошибке и поле, в котором значение задано не корректно.

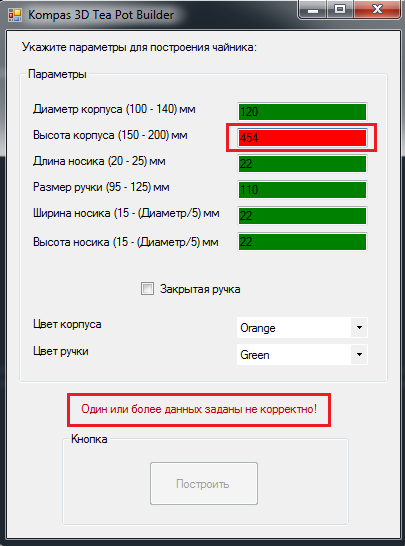


Рисунок 4.3 – Ошибка при вводе некорректных параметров.

# 4.2 Модульное тестирование

Юнит-тестирование (блочное тестирование, «unit-testing») —тестирование отдельного элемента изолированно от остальной системы [7].

Список тестовых сценариев для модульного тестирования граничных значений входных параметров представлен в таблице 4.1.

Таблица 4.1 – Список тестовых сценариев.

|  |  |
| --- | --- |
| Название тестового метода | Описание |
| TestGet\_CorrectDiameter(double Diameter) | Тестирование при вводе корректных граничных значений диаметра корпуса чайника |
| TestGet\_CorrectHeight(double Height) | Тестирование при вводе корректных граничных значений высоты корпуса чайника |

Продолжение таблицы 4.1

|  |  |
| --- | --- |
| TestGet\_CorrectSpoutLength(double SpoutLength) | Тестирование при вводе корректных граничных значений длины носика чайника |
| TestGet\_CorrectSpoutWidth  (double Diameter, double SpoutWidth) | Тестирование при вводе корректных граничных значений ширины носика чайника (зависимого параметра) |
| TestGet\_CorrectSpoutHeight  (double Diameter, double SpoutHeight) | Тестирование при вводе корректных граничных значений высоты носика чайника (зависимого параметра) |
| TestGet\_CorrectHandleSize(double  HandleSize) | Тестирование при вводе корректных граничных значений размера ручки чайника |
| TestGet\_CorrectBodyColor  (TeaPotParams.TheColor theColor) | Тестирование при вводе корректного значения цвета корпуса чайника |
| TestGet\_CorrectHandleColor  (TeaPotParams.TheColor theColor1) | Тестирование при вводе корректного значения цвета ручки и носика чайника |
| TestGet\_BadDiameter(double Diameter) | Тестирование при вводе некорректных значений диаметра корпуса чайника |
| TestGet\_BadHeight(double Height) | Тестирование при вводе некорректных значений высоты корпуса чайника |
| TestGet\_BadSpoutLength(double SpoutLength) | Тестирование при вводе некорректных значений длины носика |
| TestGet\_BadSpoutWidth(double Diameter, double SpoutWidth) | Тестирование при вводе некорректных значений ширины носика (зависимый параметр) |
| TestGet\_BadSpoutHeight(double Diameter,  double SpoutHeight) | Тестирование при вводе некорректных значений высоты носика (зависимый параметр) |

Окончание таблицы 4.1

|  |  |
| --- | --- |
| TestGet\_BadHandleSize(double HandleSize) | Тестирование при вводе некорректных значений размера ручки чайника |
| TestGet\_BadBodyColor  (TeaPotParams.TheColor theColor) | Тестирование при вводе некорректного значения цвета корпуса чайника |
| TestGet\_BadValues  (TeaPotParams.TheColor theColor1) | Тестирование при вводе некорректного значения цвета ручки и носика чайника |
| TestCorrectConstructor(double Diameter, double Height, double SpoutLength, double SpoutWidth, double SpoutHeight, double HandleSize, TeaPotParams.TheColor theColor,  TeaPotParams.TheColor theColor1) | Тестирование при корректных граничных значениях конструктора TeaPotParams |
| TestBadConstructor(double Diameter, double Height, double SpoutLength, double SpoutWidth, double SpoutHeight, double HandleSize, TeaPotParams.TheColor theColor, TeaPotParams.TheColor theColor1) | Тестирование при некорректных значениях конструктора TeaPotParams |
| Test\_Getters() | Тестирование геттеров |

Результаты успешного прохождения всех модульных тестов приведены на рисунке 4.4.

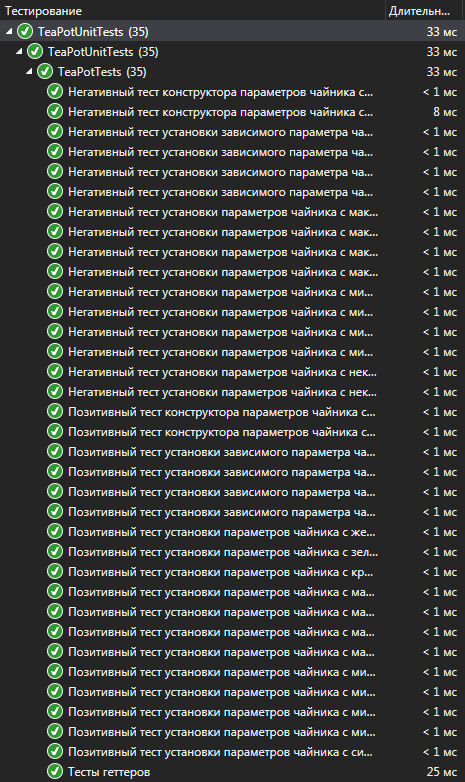


Рисунок 4.4 - Результаты модульных тестов

# 4.3 Нагрузочное тестирование

Нагрузочное тестирование — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству) [8].

Тестирование производилось на ПК со следующей аппаратной конфигурацией:

Процессор: Intel Core i7-2600

Графическая система: GeForce GTX 1050 Ti

ОЗУ: 12 GB

После построения 207 деталей с минимальными параметрами программа завершилась со сбоем из-за нехватки памяти. На момент сбоя количество потребляемой памяти было около 900 МБ.

После построения 158 деталей с максимальными параметрами программа также завершилась со сбоем. На момент сбоя количество потребляемой памяти было около 1 ГБ.

На рисунках 4.5 и 4.6 изображены графики зацикленного построения фигуры с минимальными (оранжевая линия) и максимальными (голубая линия) параметрами.

Минимальные параметры:

* Диаметр корпуса 100 мм;
* Высота корпуса 150 мм;
* Длина носика 20 мм;
* Размер ручки 95 мм;
* Ширина носика 15 мм;
* Высота носика 15 мм;

Максимальные параметры:

* Диаметр корпуса 140 мм;
* Высота корпуса 200 мм;
* Длина носика 25 мм;
* Размер ручки 125 мм;
* Ширина носика 28 мм;
* Высота носика 28 мм;

На рисунке 4.5 изображен график зависимости времени в минутах (ось OX) от количества непрерывно-строящихся фигур (ось OY).

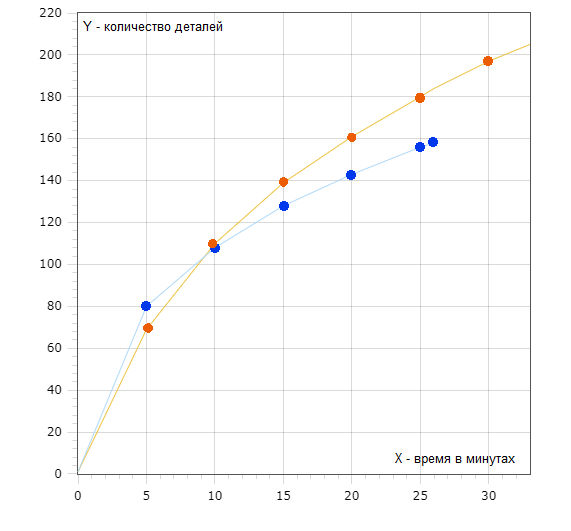


Рисунок 4.5 – График зависимости количества построенных фигур с минимальными параметрами от времени

По данному графику можно определить, что при увеличении количества деталей, постепенно увеличивается время, затраченное на создание одной фигуры.

На рисунке 4.6 изображен график зависимости количества непрерывно-строящихся деталей (ось OX) от объема памяти в MB (ось OY), используемой «КОМПАС-3D».

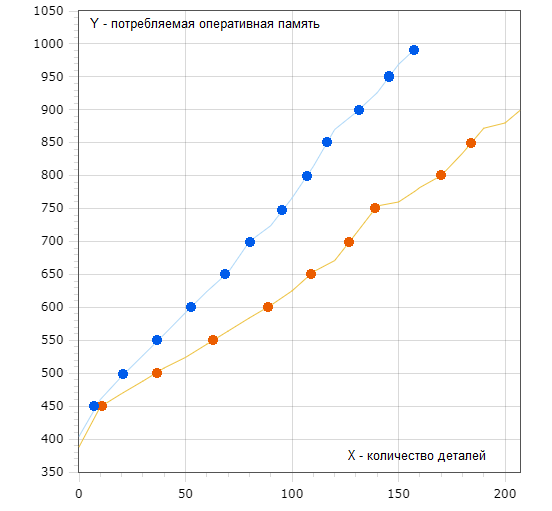


Рисунок 4.6 – График зависимости загруженности памяти от количества деталей.

По данному графику можно определить, что при увеличении количества деталей, оперативная память увеличивается линейно.

# Заключение

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

# Список литературы

КОМПАС – 3D [Электронный ресурс]. – URL: https://kompas.ru/ (дата обращения: 21.02.2020);

Электрический чайник. [Электронный ресурс]. — Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/703986 (дата обращения 22.02.2020);

Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 20.04.2020);

Что такое API? Хабр [Электронный ресурс]. – URL: https://habr.com/ru/post/464261/ (дата обращения: 20.04.2020);

Плагин PDF [Электронный ресурс]. — Режим доступа: http://gkmsoft.ru/ru/#/pdf3d\_kompas (дата обращения 02.03.2020);

UML. [Электронный ресурс]. – Режим доступа: http://www.uml.org/ (дата обращения 02.03.2020);

Новые технологии в программировании: учебное пособие / А.А. Калентьев, Д. В. Гарайс, А. Е. Горяинов. – Томск, 2014. − ­ 176 стр. (дата обращения: 01.05.2020);

Виды тестирования [электронный ресурс]. – режим доступа: https://qa-academy.by/qaacademy/news/klassifikaciya-vidov-testirovaniya/ (дата обращения: 02.05.2020).