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)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:

студент гр. 586–1

\_\_\_\_\_\_\_А. А. Мазуренко

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Руководитель:

к.т.н, доцент каф. КСУП

\_\_\_\_\_\_\_ А.А. Калентьев

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Томск 2020

**РЕФЕРАТ**

Пояснительная записка 33 стр., 18 рис., 2 таблицы, 10 источников.

AUTOCAD, ПЛАГИН, МОДЕЛЬ ГРАФИНА, САПР, АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ.

Целью данной работы является разработка плагина автоматизирующего построения модели графина для системы автоматизированного проектирования (САПР) AutoCAD, с помощью интегрированной среды разработки Visual Studio Community 2019.

В разработке использовалась система контроля версий Git.

В результате работы был разработан плагин, автоматизирующий построение графина.

Отчет по пояснительной записке выполнен в текстовом процессоре Microsoft Word 2016.

Оглавление

[1 Введение 4](#_Toc40675758)

[2 Постановка и анализ задачи 5](#_Toc40675759)

[2.1 Описание предмета проектирования 5](#_Toc40675760)

[2.2 Выбор инструментов и средств реализации 6](#_Toc40675761)

[2.3 Назначение плагина 7](#_Toc40675762)

[2.4 Описание аналогов разрабатываемого продукта 7](#_Toc40675763)

[2.4.1 Плагин SHVAC–RD–3D для AutoCAD 7](#_Toc40675764)

[2.4.2 Плагин Bolts 3D imperial для AutoCAD 8](#_Toc40675765)

[3 Описание реализации 10](#_Toc40675766)

[3.1 Диаграмма прецедентов 10](#_Toc40675767)

[3.2 Диаграмма классов 12](#_Toc40675768)

[4 Описание программы для пользователя 15](#_Toc40675769)

[5 Тестирование программы 18](#_Toc40675770)

[5.1 Функциональное тестирование 18](#_Toc40675771)

[5.2 Модульное тестирование 23](#_Toc40675772)

[5.3 Нагрузочное тестирование 25](#_Toc40675773)

[Заключение 29](#_Toc40675774)

[Список использованных источников 30](#_Toc40675775)

[Приложение А 31](#_Toc40675776)

# Введение

Автоматизация – эффективный способ повышения производительности труда инженерно-технических отрасли. При использовании в проектировании автоматизации, позволяет быстро реагировать на появление новых решений и технологий.

Огромный перечень областей применения систем автоматизированного проектирования (САПР) – это причина, по которой любая из современных CAD–систем должна быть максимально открытой и содержать в себе инструменты для создания сторонних библиотек.

В каждой крупной САПР есть свои средства для разработки, которые предоставляются с целью дать возможность разработчикам расширять систему под свои нужды. Таким средством является API – интерфейс программирования приложений. API позволяет определить возможности, которые предоставляет приложение разработчикам, при этом абстрагируясь от того, как она реализована.

Расширение возможностей, в основном, подразумевает разработку библиотеки на основе предоставленного API. Библиотекой является независимо компилируемый программный модель, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей [1].

Целью данной работы является разработка плагина, автоматизирующего построение модели графина, для системы автоматизированного проектирования «AutoCAD» [2], с помощью интегрированной среды разработки Visual Studio Сommunity 2019 [3].

# Постановка и анализ задачи

В рамках учебной дисциплины «Основы разработки САПР» требовалось разработать плагин в соответствии с техническим заданием. На основе заданных параметров библиотека, взаимодействуя с САПР «AutoCAD», должна строить 3D–модель «Графина».

Кроме этого, требовалось, чтобы плагин позволял изменять входные и параметры, а именно: высота графина, диаметр основания, диаметр горла, наличие или отсутствие пробки, высота пробки, наличие или отсутствие ручки, угол наклона ручки и длина ручки.

## 2.1 Описание предмета проектирования

Предметом проектирования является модель графина.

Графин – сосуд, предназначенный для кратковременного хранения и подачи к столу жидкостей. Изготавливается только из стекла или хрусталя.

Конструкцию графина можно условно разделить на две основные части. Нижняя — это вместилище для жидкости; верхняя — удлиненное горлышко для налива.

Дополнительно изделие оснащается крышкой в виде пробки.

Параметры графина должны соответствовать следующим значениям:

* Высота графина H: от 50 до 300 мм;
* Диаметр основания D: от 50 до 100 мм;
* Диаметр горла графина d: от 25 мм до диаметра основания;
* Наличие / отсутствие пробки;
* Высота пробки h: от 10 мм до 25 мм;
* Наличие / отсутствие ручки;
* Угол наклона ручки : от 0 до 90 градусов;
* Длина ручки l: от 25 мм до 2/3 высоты графина.

На рисунке 2.1 представлен графин с геометрическими параметрами.



Рисунок 2.1 – Геометрические параметры графина

Обозначенные параметры:

H – Высота графина;

h – Высота пробки графина;

D – Диаметр основания графина;

d – Диаметр горла графина;

l – Длина ручки;

– Угол наклона ручки.

## 2.2 Выбор инструментов и средств реализации

Исходя из требований в техническому задания программа выполнена на языке C# в среде Microsoft Visual Studio 2019 [3], с использованием .NET Framework 4.8.0, для САПР «AutoCAD».

Инструментом тестирования и создания модульных тестов был выбран стандартный обозреватель тестов среды Microsoft Visual Studio 2019 [3] с тестовым фреймворком NUnit версии 3.12.0.

Для реализации пользовательского интерфейса использовался WinForm.

Взаимодействие плагина с системой AutoCAD 2020 [1] осуществляется посредством интерфейсов, называемого API. Явных преимуществ между версиями нет, поскольку обе версии реализуют различные функции системы и взаимно дополняют друг друга.

Для выполнения лабораторных использовался AutoCAD .NET API.

## 2.3 Назначение плагина

Назначение, разрабатываемого плагина обусловлено быстрым моделированием графина. Благодаря данному расширению, можно наглядно посмотреть результат работы и при необходимости изменить некоторые параметры, без предварительного изготовления графина.

## 2.4 Описание аналогов разрабатываемого продукта

### 2.4.1 Плагин SHVAC–RD–3D для AutoCAD

Данный плагин, от компании Soft Draft, предназначен для моделирования угольника, выступа и трубу. [4]

На рисунке 2.2 представлен пользовательский интерфейс плагина, для построения выступа.



Рисунок 2.2 – Пользовательский интерфейс плагина SHVAC–RD–3D

### 2.4.2 Плагин Bolts 3D imperial для AutoCAD

Плагин создает болты, гайки и шайбы в виде трехмерных тел для удобного размещения в трехмерных моделях конструкционной стали.[4]

Встроенный поворот позволяет размещать болты в направлении вниз, сбоку или вверх.

Размерные данные для болтов хранятся во внешних файлах, в которые возможно добавить дополнительную информацию о размерах болтов или гаек, которые являются уникальными для конкретного применения.

На рисунке 2.3 представлен пользовательский интерфейс плагина.



Рисунок 2.3 – Пользовательский интерфейс плагина Bolts 3D Imperial

# 3 Описание реализации

На этапе разработки проекта программы для формального описания архитектурной особенности, пользовательского сценария системы был выбран унифицированный язык моделирования (UML) [5].

На основе UML построены: диаграммы вариантов использования диаграммы классов, диаграммы пакетов.

В процессе реализации диаграммы классов, вариантов использования и пакетов были дополнены.

Дополнения были обусловлены нарастанием функционала в реализации мелких детальных особенностей и изменений требований заказчика.

## 3.1 Диаграмма прецедентов

Диаграмма прецедентов — диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. [5]

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. [6]

На рисунке 3.1 представлена диаграмма прецедентов плагина.

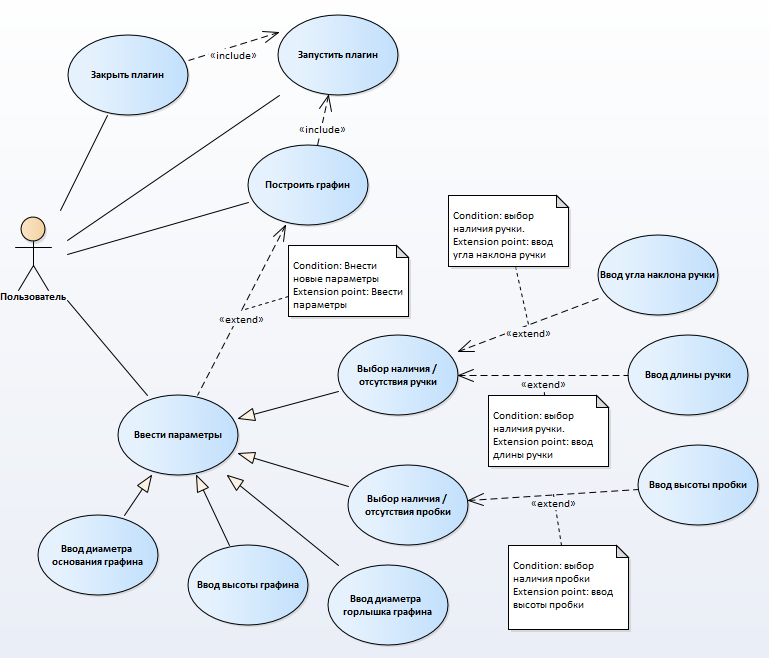


Рисунок 3.1 – Диаграмма прецедентов плагина

Действующим лицом выступает пользователь.

Для пользователя доступы следующие возможности:

* Ввести параметры;
* Построить графина;
* Запустить плагин;
* Закрыть плагин.

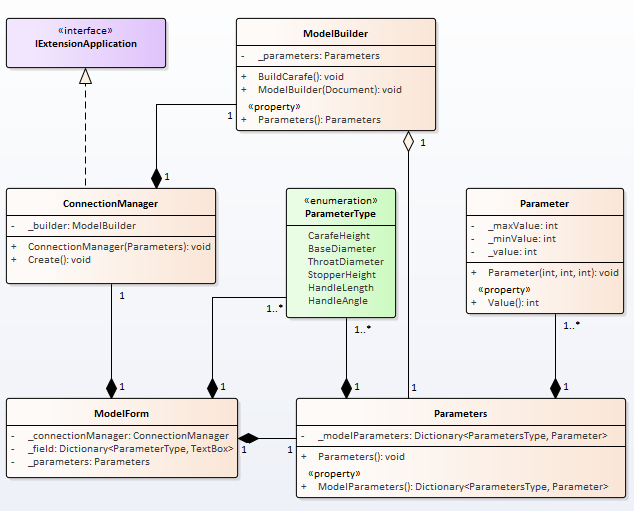
В результате изменений, внесенных заказчиком, диаграмма прецедентов не подвергалась изменениям.

Добавлена возможность отрисовки модели графина во время ввода параметров.

## 3.2 Диаграмма классов

UML диаграммы классов представляют собой графическую интерпретацию классов системы, их атрибутов, методов и взаимосвязей между ними. [7]

На рисунке 3.2 представлена изначальная диаграмма классов разрабатываемого плагина.

Рисунок 3.2 – Изначальная диаграмма классов плагина

В процессе работы, диаграмма классов была изменена, в результате изменений, внесенных заказчиком.

На рисунке 3.3 представлена конечная диаграмма классов плагина.

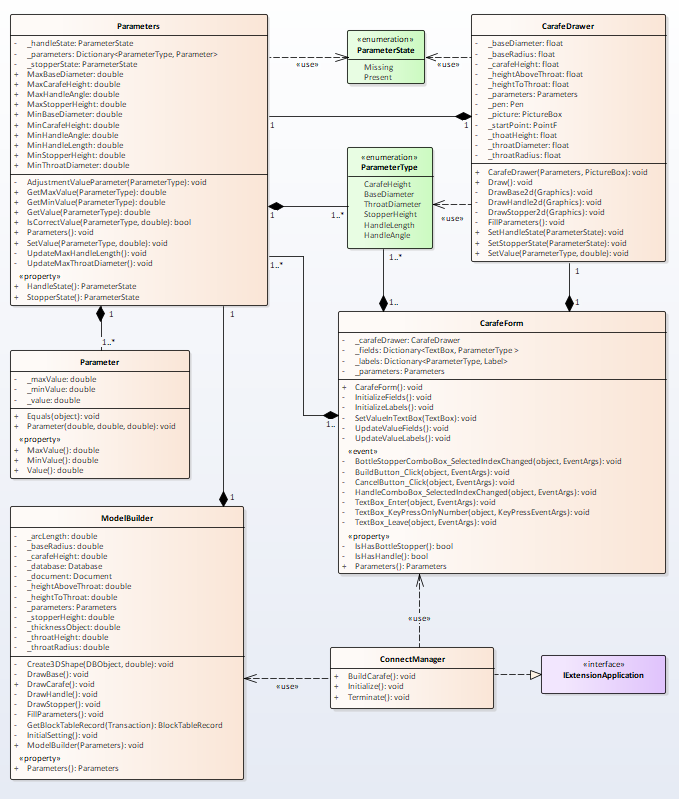


Рисунок 3.3 – Конечная диаграмма классов плагина.

Класс CarafeForm является формой пользовательского интерфейса. Реализует методы, используемые для взаимодействия с пользователем.

ModelBuilder – класс, который хранит параметры модели, введенные пользователем. В данном классе реализуются методы, используемые для построения 3D–модели.

ConnectManager – класс, реализующий методы для связи программы с САПР.

Класс Parameter хранит информацию об одном параметре проектируемой модели.

Класс Parameters хранит словарь со всеми параметрами модели, также содержит метод, который задает новое значение параметра.

CarafeDrawer – класс, методы которого реализуют отрисовку чертежа графина на форме CarafeForm.

ParameterState – перечисление, содержащее состояния параметра при построении – наличие, отсутствие.

ParameterType – перечисление, содержащее название параметров графина.

# 4 Описание программы для пользователя

Плагин представляет собой диалоговое окно с полями для ввода и выбора соответствующих параметров.

На диалоговом окне содержится 4 группы:

Группа 1 – область ввода основных параметров (диаметр основания, высота и диаметр горлышка;

Группа 2 – область параметров пробки графина;

Группа 3 – область параметров ручки графина;

Группа 4 – область управления построением.

Начальный вид диалогового окна плагина изображен на рисунке 4.1.

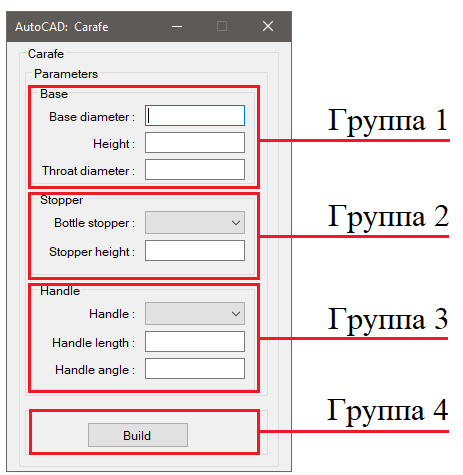


Рисунок 4.1 – Начальный вид диалогового окна плагина

В результате изменений, внесенных заказчиком, интерфейс программы подвергся изменениям.

Были добавлены:

* Блок отрисовки чертежа графина;
* Кнопка «Cancel».

На рисунке 4.2 представлена конечный вид диалогового окна плагина.

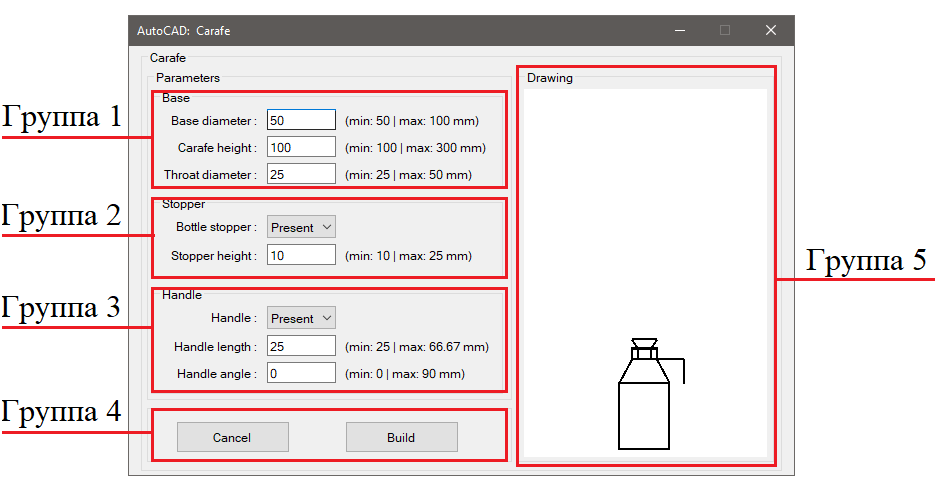


Рисунок 4.2 – Конечный вид диалогового окна плагина

Конечный вид диалогового окна содержит 5 групп:

Группа 1 – область ввода основных параметров (диаметр основания, высота и диаметр горлышка;

Группа 2 – область параметров пробки графина;

Группа 3 – область параметров ручки графина;

Группа 4 – область управления построением;

Группа 5 – область отрисовки чертежа графина.

При запуске плагина все параметры будут заполнены минимальными значениями.

При попытки ввести в поля параметров графина что–либо кроме цифр и точки/запятой, ввод игнорируется и символ не отображается.

При изменении значений параметров графина и переключения зависимых элементов («Bottle stopper» и «Handle») происходит отрисовка графина на главном окне плагина.

Нажатие кнопки «Build» производит закрытие окна плагина и построение детали, с учетом заданных параметров, в текущем документе программы AutoCAD. Нажатие кнопки доступно с момента запуска плагина.

Нажатие кнопки «Cancel» закрывает окно плагина. Нажатие кнопки доступно с момента запуска плагина.

# 5 Тестирование программы

Тестирование позволяет выявлять ошибки в программе в процессе разработки и при выпуске промежуточных и финальных версий приложений.

## 5.1 Функциональное тестирование

При функциональном тестировании [8] проверялась корректность работы плагина «Графин», а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами.

Проведем тестирование минимальных и максимальных параметром модели.

Минимальные параметры модели:

Высота графина: 50 мм;

Диаметр основания графина: 100 мм;

Диаметр горла графина: 25 мм;

Наличие пробки: присутствует;

Высота пробки: 10 мм;

Наличие ручки: присутствует;

Длина ручки: 25 мм;

Угол наклона ручки: 0 градусов.

На рисунке 5.1 представлена модель графина с минимальными параметрами.

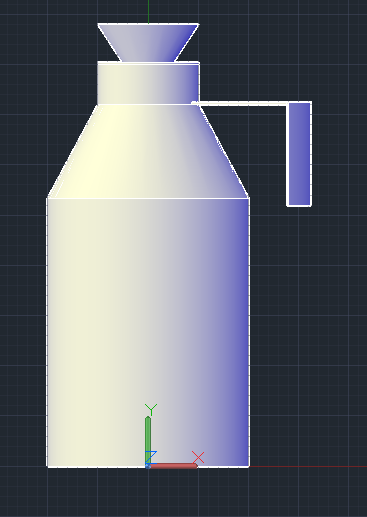


Рисунок 5.1 – Модель графина с минимальными параметрами

Максимальные параметры модели:

Высота графина: 100 мм;

Диаметр основания графина: 300 мм;

Диаметр горла графина: 100 мм;

Наличие пробки: присутствует;

Высота пробки: 25 мм;

Наличие ручки: присутствует;

Длина ручки: 200 мм;

Угол наклона ручки: 90 градусов.

На рисунке 5.2 представлена модель графина с максимальными параметрами.

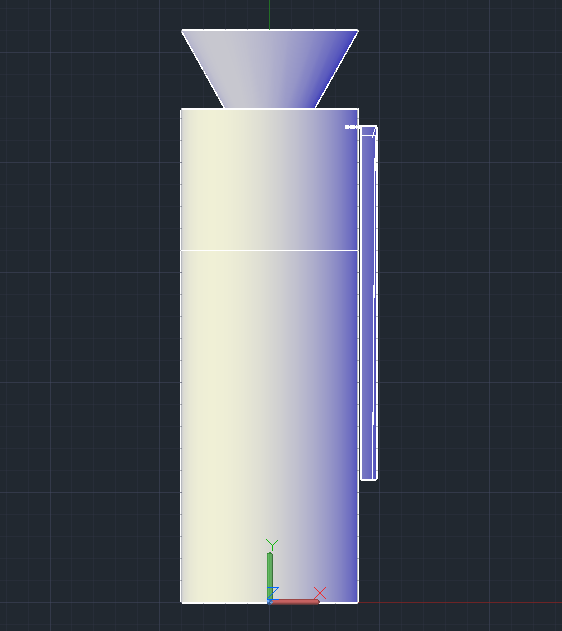


Рисунок 5.2 – Модель графина с максимальными параметрами

Проведем тестирование построения графина с минимальными параметрами, но без построения ручки.

На рисунке 5.3 представлена модель графина без ручки, с минимальными параметрами.

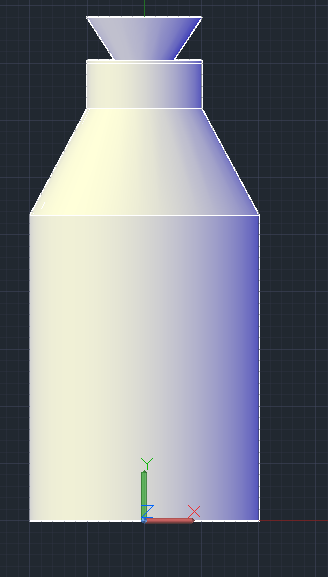


Рисунок 5.3 – Модель графина без ручки, с минимальными параметрами.

Проведем тестирование построения графина с минимальными параметрами, но без построения пробки.

На рисунке 5.4 представлена модель графина без пробки, с минимальными параметрами.

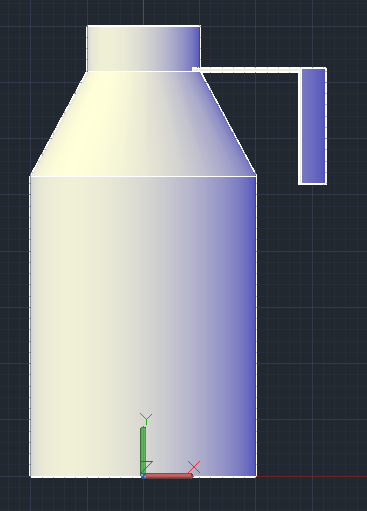


Рисунок 5.4 – Модель графина без пробки, с минимальными параметрами.

Проведем тестирование построения графина с минимальными параметрами, без построения пробки и ручки.

На рисунке 5.5 представлена модель графина без пробки и ручки, с минимальными параметрами.

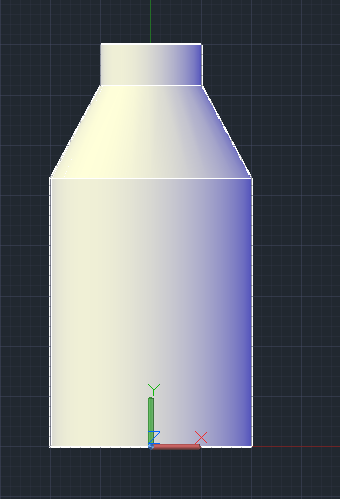


Рисунок 5.5 – Модель графина без пробки и ручки, с минимальными параметрами.

## 5.2 Модульное тестирование

В целях проверки корректности работы методов и свойств классов, при помощи обозревателя тестов Visual Studio, было проведено модульное тестирование [9].

Проверялись открытые методы и свойства, для этого были созданы тестовые классы.

* ParameterTest – класс, тестирующий свойства и методы класса Parameter. Описание класса ParameterTest в Приложении А (таблица А.1);
* ParametersTest – класс, тестирующий свойства и методы класса Parameters. Описание класса ParametersTest в Приложении А (таблица А.2).

На рисунке 5.6 представлено диалоговое окно состояний запущенных тестов для классов ParametersTest и ParameterTest.

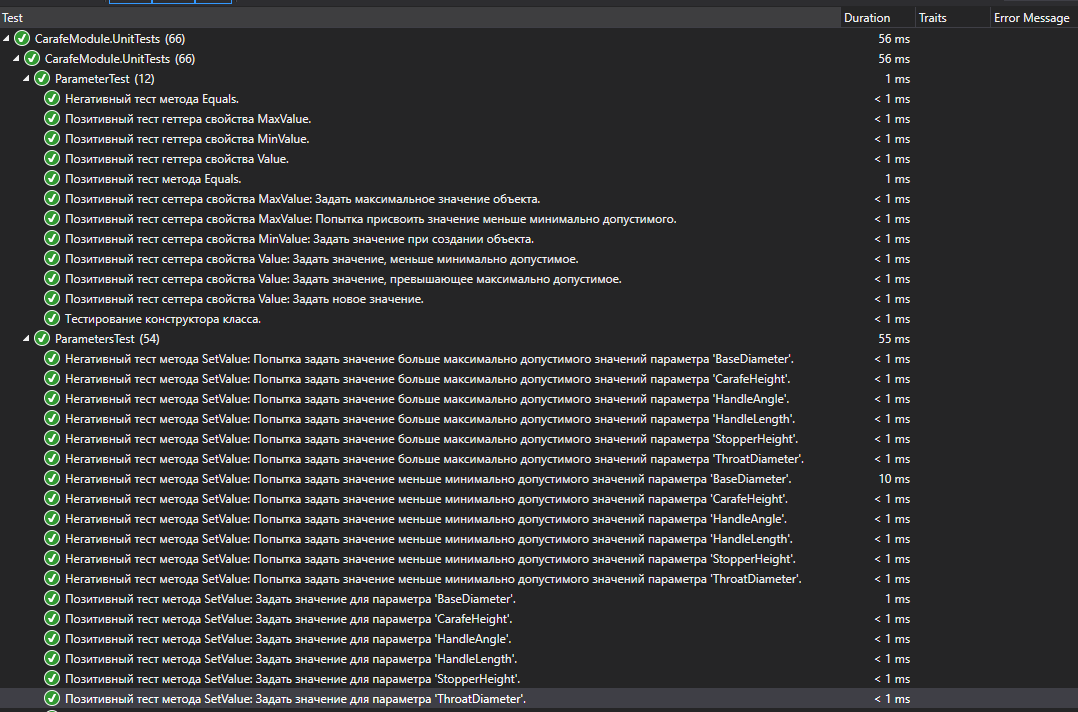


Рисунок 5.6 – Диалоговое окно состояний запущенных тестов для классов ParametersTest и ParameterTest

На рисунке 5.7 представлено диалоговое окно состояний запущенных тестов для класса ParametersTest.

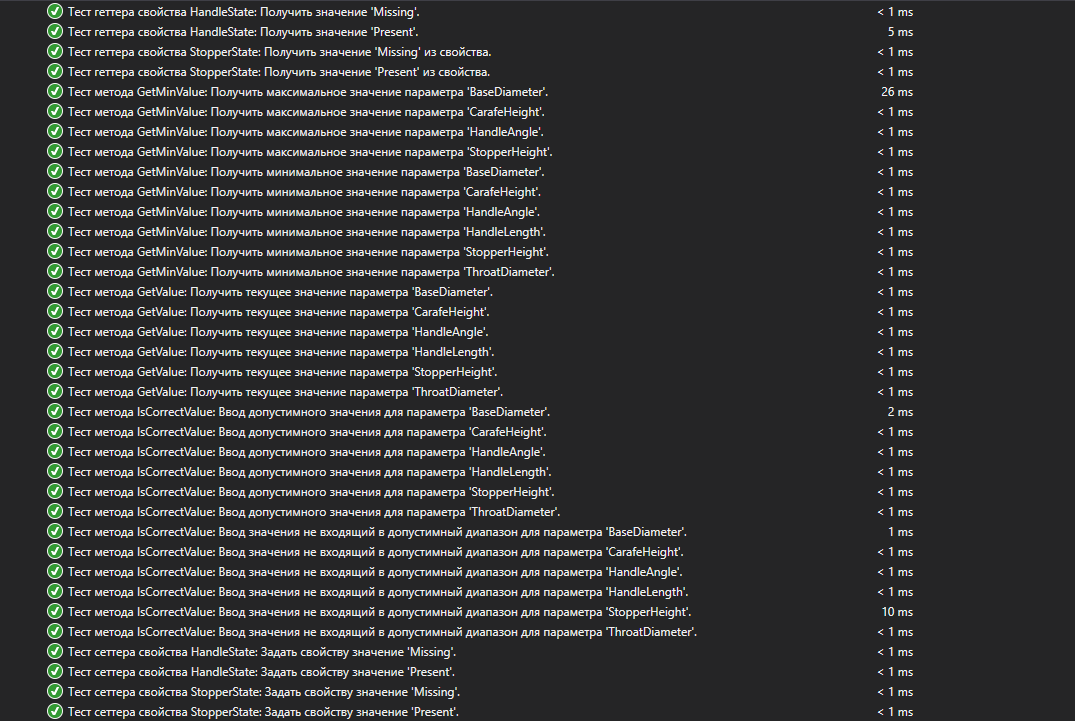


Рисунок 5.7 – Диалоговое окно состояний запущенных тестов для класса ParamtersTest

## 5.3 Нагрузочное тестирование

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование [10].

Конфигурация ПК, на котором проводилось тестирование плагина:

* Процессор: Intel i5–4210h с тактовой частотой 2.9 ГГц;
* ОЗУ: 16 ГБ DDR3L 800 МГц;
* HDD: 750 ГБ;
* Графическое устройство: NVIDIA GeForce GTX 850M.

Для проведения нагрузочного тестирования был добавлен секундомер («Stopwatch»), который засекал время от начала построения, с каждым успешным построением модели производилась запись результатов в текстовый файл «log.txt».

На рисунке 5.8 представлен код зацикливания построения модели.

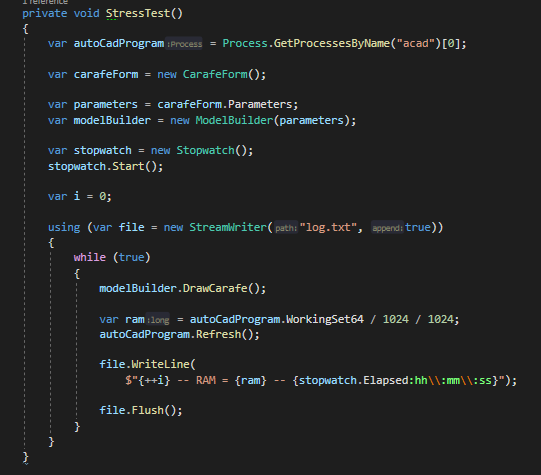


Рисунок 5.8 – Зацикливание построения модели

Тестирование выполнялось с минимальными значениями параметров графина:

* Высота графина: 50 мм;
* Диаметр основания графина: 100 мм;
* Диаметр горла графина: 25 мм;
* Наличие пробки: присутствует;
* Высота пробки: 10 мм;
* Наличие ручки: присутствует;
* Длина ручки: 25 мм;
* Угол наклона ручки: 0 градусов.

Выполнение теста продлилось 125 минут, после чего произошел сбой программы «AutoCAD».

На протяжении всего выполнения теста, общая загруженность процессора была в пределах от 15 до 23 процентов.

Потребление ОЗУ плагином, варьировалось в пределах 2–4 Мб за каждое построение.

На рисунках 5.9, 5.10 представлены графики тестирование зацикленного построения модели с минимальными параметрами.

Рисунок 5.9 – График зависимости количества моделей от времени

Рисунок 5.10 – График зависимости количества моделей от количества физической памяти

По графику, изображенного на рисунке 5.9, можно заметить, что скорость построения за все время тестирования, почти всегда была одинаковой.

Исходя из графика, изображенного на рисунке 5.10, потребляемая физическая память росла и достигла пика при построения, примерно, 2300–ой модели, после чего потребление оперативной памяти идет на спад до значений 1400 Мб.

# Заключение

В ходе выполнения лабораторных работ были изучены предметная область проектирования, предмет проектирования, аналоги предмета проектирования, API и на основании полученных данных были спроектированы архитектура и макет системы, создан плагин «Графин», проведены модульные, функциональные и нагрузочные тесты.

# Список использованных источников

1. Официальный сайт AutoDesk. Продукт AutoCAD [Электронный ресурс] – Режим доступа: [https://www.autodesk.ru/products/autocad/](https://www.autodesk.ru/products/autocad/overview) (дата посещения: 28.02.2020);
2. Официальный сайт AutoDesk. [Электронный ресурс] – Режим доступа: [https://www.autodesk.ru/autodesk-developer-network/software-platform-russian /develop-autocad](https://www.autodesk.ru/autodesk-developer-network/software-platform-russian%20/develop-autocad) (дата посещения: 28.02.2020);
3. Microsoft Visual Studio 2019 [Электронный ресурс]. – URL: <https://visualstudio.microsoft.com/ru/vs/> (дата обращения: 15.05.2020);
4. Официальный сайт SoftDraft. [Электронный ресурс] – Режим доступа: <https://www.softdraft.com/> (дата посещения: 13.03.2020);
5. ITVDN Forum [Электронный ресурс] – Режим доступа: <https://forum.itvdn.com/t/urok-2-raznovidnosti-uml-diagramm/3315> (дата посещения: 20.03.2020);
6. Блог программиста [Электронный ресурс] – Режим доступа: <https://pro-prof.com/archives/2594> (дата посещения: 20.03.2020);
7. Мартин Фаулер. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. / М. Фаулер – Изд: Символ–Плюс, 2011, с.192 (3-е издание);
8. Функциональное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 13.04.2020);
9. Модульное тестирование [Электронный ресурс]. – URL: <http://espressocode.top/unit-testing-software-testing/> (дата обращения: 13.04.2020);
10. Нагрузочное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/nagruzochnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 13.04.2020).

# Приложение А

(Справочное)

В таблицах приложения для обозначения модификаторов доступа полей приняты следующие условные знаки:

* « – » обозначение private (закрытого) поля;
* « + » обозначение public (открытого) поля.

Описание полей и методов используемых для проверки текстовых случаев класса Parameter представлено в таблице A.1

Таблица А.1 – Описание полей и методов класса ParameterTest.

|  |  |
| --- | --- |
| Наименование | Описание |
| – \_minValue: double | Поле хранит минимальное значение параметра. |
| – \_maxValue: double | Поле хранит максимальное значение параметра. |
| – \_value: double | Поле хранит значение параметра. |
| – \_parameter: Parameter | Поле хранит данные параметра модели. |
| + CreateTestParameter() | Метод заполняет поле \_parameter тестовыми данными: \_minValue, \_maxValue, \_value. |
| + Test\_MinValue\_Set\_CorrectValue  (double, string) | Метод для проверки корректности записи минимального значения. |
| + Test\_MaxValue\_Set\_CorrectValue  (double, string) | Метод для проверки корректности записи максимального значения. |
| + Test\_MinValue\_Get\_CorrectValue  (double, string) | Метод для проверки корректности возврата минимального значения. |
| + Test\_MaxValue\_Get\_CorrectValue  (double, string) | Метод для проверки корректности возврата максимального значения. |
| + Test\_Value\_Set\_CorrectValue  (double, string) | Метод для проверки корректности записи текущего значения. |
| + Test\_Value\_Get\_CorrectValue  (double, string) | Метод для проверки корректности возврата текущего значения. |
| + Test\_EqualsParameter\_CorrectValue  (string) | Метод для проверки корректности сравнения одинаковых элементов. |

Продолжение таблицы А.1

|  |  |
| --- | --- |
| Наименование | Описание |
| + Test\_EqualsParameter\_WrongValue  (string) | Метод для проверки корректности сравнения различных элементов. |
| + Test\_Parameter(string) | Метод для проверки корректности работы конструктора. |

Описание полей и методов используемых для проверки тестовых случаев класса Parameters представлено в таблице А.2.

Таблица А.2 – Описание полей и методов класса ParametersTest.

|  |  |
| --- | --- |
| Наименование | Описание |
| – \_parameters: Parameters | Поле хранит данные параметров модели. |
| + CreateParameters() | Метод инициализирует поле \_parameters. |
| +Test\_StopperState\_Set\_CorrectValue  (ParameterState, string) | Метод для проверки корректности записи состояния крышки графина. |
| + Test\_StopperState\_Get\_CorrectValue  (ParameterState,string) | Метод для проверки корректности возврата состояния крышки графина |
| +Test\_HandleState\_Set\_CorrectValue  (ParameterState, string) | Метод для проверки корректности записи состояния ручки графина. |
| + Test\_HandleState\_Get\_CorrectValue  (ParameterState,string) | Метод для проверки корректности возврата состояния ручки графина. |
| + Test\_GetMinValue\_CorrectValue  (ParameterType, double, string) | Метод для проверки корректности возврата минимального значения указанного параметра. |
| + Test\_GetMaxValue\_CorrectValue  (ParameterType, double, string) | Метод для проверки корректности возврата максимального значения указанного параметра. |

Продолжение таблицы А.2

|  |  |
| --- | --- |
| Наименование | Описание |
| + Test\_ GetValue\_CorrectValue  (ParameterType, double, string) | Метод для проверки корректности возврата значения указанного параметра. |
| + Test\_ SetValue\_CorrectValue  (ParameterType, double, string) | Метод для проверки корректности записи значения указанного параметра. |
| + Test\_ SetValue\_WrongValue  (ParameterType, double, string) | Метод для проверки записи неверного значения указанного параметра. |
| + Test\_IsCorrectValue\_CorrectValue  (ParameterType, double, string) | Метод для проверки корректности метода проверки значения, входящим в допустимый диапазон. |
| + Test\_IsCorrectValue\_WrongValue  (ParameterType, double, string) | Метод для проверки корректности метода проверки значения, не входящим в допустимый диапазон. |