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)

Студент гр. 586-1

\_\_\_\_\_\_\_\_\_\_\_\_\_С.А. Леонов

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

Руководитель

К.т.н., доцент каф. КСУП

\_\_\_\_\_\_\_\_\_\_А.А. Калентьев

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

2020

**РЕФЕРАТ**

Пояснительная записка 44 с., 27 рис., 4 таблицы, 12 источника.

КОМПАС-3D, ПЛАГИН, МОДЕЛЬ МУЗЫКАЛЬНОЙ КОЛОНКИ, АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ, САПР.

Целью данной работы является разработка плагина, автоматизирующего построение модели музыкальной колонки, для системы автоматизированного проектирования КОМПАС-3D 16.1, с помощью интегрированной среды разработки Visual Studio Сommunity 2019.

В процессе работы изучено внешнее устройство музыкальной колонки и основные функции системы автоматизированного проектирования «Компас-3D» v 16.

В результате работы был создан плагин, автоматизирующий построение музыкальной колонки.

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

**Содержание**

1 Введение 4

2 Постановка и анализ задачи 8

2.1 Описание предмета проектирования 8

2.2 Выбор инструментов и средств реализации 9

2.3 Назначение плагина 9

2.4 Описание аналогов разрабатываемого продукта 9

2.4.1 Оборудование: Металлоконструкции 11

2.4.2 Оборудование: Кабели и жгуты 13

3 Описание реализации 14

3.1 Диаграмма прецедентов плагина 16

3.2 Диаграмма классов 19

3.3 Диаграмма пакетов 21

4 Описание программы для пользователя 26

5 Тестирование программы 27

5.1 Функциональное тестирование 29

5.2 Модульное тестирование 32

5.3 Нагрузочное тестирование 37

Заключение 36

Список использованных источников 37

Приложение А 39

**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.

Таким образом, целью данной работы является разработка плагина, автоматизирующего построение модели музыкальной колонки, для системы автоматизированного проектирования «Компас-3D» v 16 [1], с помощью интегрированной среды разработки Visual Studio Сommunity 2019 [5].

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 деталь «Музыкальная колонка». Более того, требовалось, чтоб плагин позволял изменять входные параметры в соответствии с требованиями пользователей, а именно: высота корпуса, ширина корпуса, длина корпуса, форма крышки динамика, высота крышки динамика (прямоугольная форма), ширина крышки динамика (прямоугольная форма), толщина крышки динамика, диаметр реле регулировки, количество динамиков на фронтальной стороне корпуса.

**2.1 Описание предмета проектирования**

Акустической системой [2] называется громкоговоритель, предназначенный для использования в качестве функционального звена в бытовой радиоэлектронной аппаратуре. Под «громкоговорителем» понимается «устройство для эффективного излучения звука в окружающее пространство в воздушной среде, содержащее одну или несколько головок громкоговорителей, при наличии акустического оформления, электрических устройств (фильтры, трансформаторы, регуляторы и т.п.).

Так как при проектировании внешнего вида модели, невозможно учесть технические характеристики звукового оборудования, пункты ГОСТ связанные с качеством звучания рассматриваться не будут.

Разрабатываемая модель музыкальной колонки должна соответствовать ГОСТ 23262-88 [3] Системы акустические бытовые. Общие технические условия.

Рассмотрим пункты, связанные в внешним видом модели.

Пункт 2.1.5 По внешнему виду и качеству исполнения АС должна соответствовать образцу – эталону утвержденному в соответствии с ГОСТ 15.009 – 86 (заменен на ГОСТ 15.009-91).

Обратимся к ГОСТ 15.009-91 [4].

ГОСТ 15.009-91 [4] Система разработки и постановки продукции на производство (СРПП). Непродовольственные товары народного потребления.

1.Непродовольственные товары народного потребления (далее - продукция), подлежащие разработке и постановке на производство, должны удовлетворять запросам населения с учетом возрастных, социальных и национальных групп потребителей, направлений развития ассортимента и моды, а также отвечать требованиям, обеспечивающим безопасность для жизни и здоровья населения и охрану окружающей среды.

Исследуем характеристики проектируемой модели.

Акустическое оформление: закрытый ящик.

Описание: Оформление оказывает непосредственное влияние на качество звука. Корпус у акустики закрытого типа представляет собой герметично закрытый ящик с выведенным на фронтальную панель диффузором (излучателем звука) динамика. Недостатки такой акустики – низкая чувствительность, не очень глубокий бас, преимущества – простота конструкции и хорошие переходные характеристики, которые гарантируют низкий уровень искажений и точное воспроизведение звука.

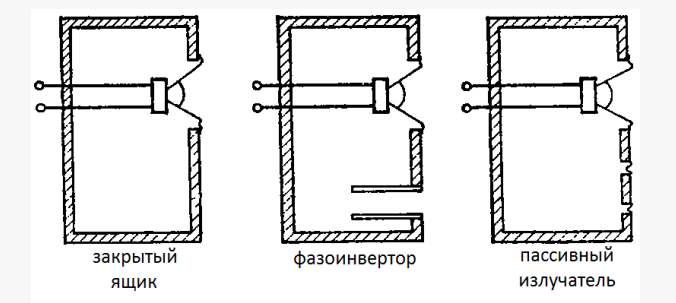


Рисунок 2.1 – Различные виды низкочастотных оформлении

Количество полос фронтальных колонок: 1.

Описание: Количество полос, на которое разбит частотный диапазон фронта колонок.

Ширина фронтальных колонок: от 150 до 200 мм.

Высота фронтальных колонок: от 100 до 500 мм.

Длинна фронтальной колонки: от 200 до 300 мм.

На рисунке 2.2 приведена 3D модель музыкальной колонки с обозначенными параметрами L, H, W, D, LS, HS, WS.

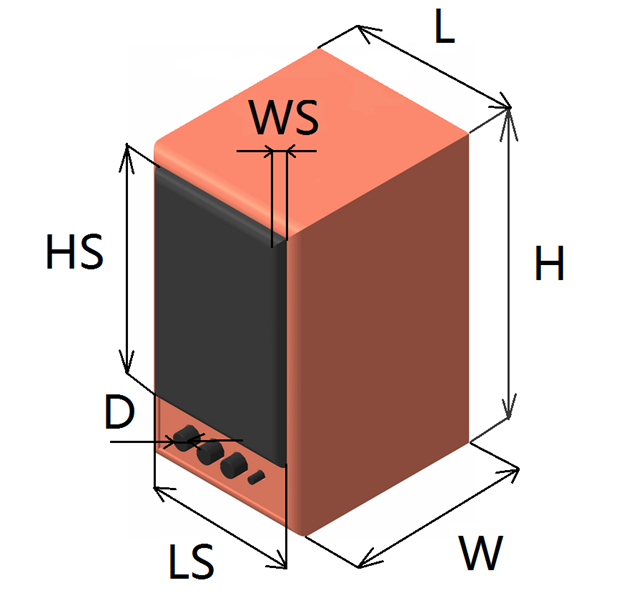


Рисунок 2.2 – 3D модель музыкальной колонки с обозначенными параметрами

Обозначенные параметры:

H – Высота корпуса;

L – Длинна корпуса;

W – Ширина корпуса;

HS – Высота динамик;

WS – Ширина наружного выступа динамика;

LS – Длинна динамика;

D – Диаметр реле регулировки.

Рассмотрим чертеж музыкальной колонки с максимально допустимыми размерами.

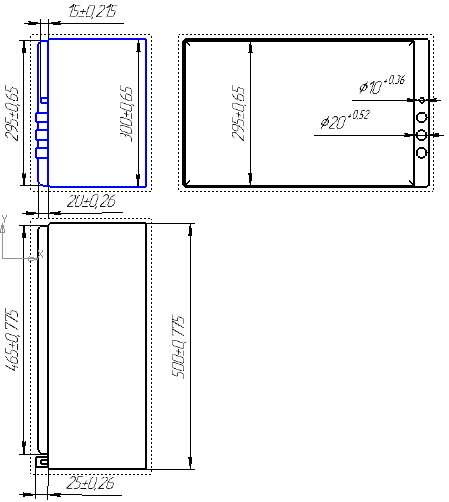


Рисунок 2.3 – Чертеж музыкальной колонки в 3-х проекциях

**2.2 Выбор инструментов и средств реализации**

В связи с требованием технического задания программа выполнена на языке C# в среде Microsoft Visual Studio 2019 [5], с использованием .NET Framework 4.7.0, для системы КОМПАС 3D V16. Инструментом тестирования и создания модульных тестов был выбран стандартный обозреватель тестов среды Microsoft Visual Studio 2019 [5] с тестовым фреймворком NUnit версии 3.12.0.

Для реализации пользовательского интерфейса использовался WinForm.

Взаимодействие плагина с системой КОМПАС 3D [1] осуществляется посредством интерфейсов, называемых API. В КОМПАС 3D на данный момент существует API двух версий API 5 [6] и API 7 [6]. Явных преимуществ между версиями нет, поскольку обе версии реализуют различные функции системы и взаимно дополняют друг друга. Для выполнения лабораторных была выбрана версия API 5, так как для полноценной реализации плагина «Музыкальная колонка» достаточно методов и свойств интерфейсов API 5.

**2.3 Назначение плагина**

Назначение, разрабатываемого плагина обусловлено выбором сферы его применения. В данном проекте выбором сферы применения стало внешнее оформление музыкального электрооборудования, так как данная сфера, одна из наиболее популярных, ввиду повышенного спроса на аудио выводящее оборудование. Для поддержание пользовательского интереса и спроса на продукцию, производителям необходимо часто выпускать товар с обновленным дизайном.

# 2.4 Описание аналогов разрабатываемого продукта

КОМПAС-3D [1] является открытой системой, что позволяет создавать дополнительные программные модули (пользовательские библиотеки) и применять их во время работы над документами. Таким образом, стандартные возможности чертежно-графического редактора и трехмерного моделирования могут быть дополнены исходя из тех специальных задач, которые приходится решать пользователю.

В настоящее время система КОМПAС-3D непрерывно расширяется плагинами и модулями, которые можно разделить на два условных класса: плагины или модули, реализующие (моделирующие) конкретные трехмерный объекты в зависимости от сферы применения.

**2.4.1 Оборудование: Металлоконструкции**

Оборудование: Металлоконструкции [7] — приложение для КОМПАС-3D [1], предназначенное для автоматизации работ по проектированию конструкций из профильного металлопроката. Приложение позволяет быстро проектировать всевозможные рамы и каркасы, автоматически создавать комплект документации.

Создание металлоконструкции в приложении начинается с построения Трехмерного каркаса — геометрических осей, которые являются эскизом конструкции. После чего для каждой из осей назначается профиль. Сортамент профиля может выбираться из нового Каталога профилей, входящего в комплект поставки приложения, либо из Справочника Материалы и Сортаменты для КОМПАС. Для удобства построения и редактирования металлоконструкции в приложении реализован механизм Характерных точек, который позволяет задавать длину и угол поворота профиля непосредственно в окне построения. При изменении Трехмерного каркаса металлоконструкция перестроится автоматически.

После назначения профилей необходимо проработать отдельные узлы металлоконструкции. Для этого в приложении есть специальные инструменты. Можно корректировать длины деталей, задавать угловую или стыковую разделки, строить дополнительные элементы в виде ребер жесткости или фасонок.

Для созданной с помощью приложения конструкции можно автоматически получить спецификацию либо любые другие виды настраиваемых отчетов. Металлоконструкция, спроектированная в приложении, может быть проверена на наличие пересечений. Трехмерная модель позволяет сразу выявить возможные нестыковки. Оборудование: Металлоконструкции позволяет избежать дополнительных затрат на материал и инструмент.

На рисунке 2.4 представлен интерфейс, каталога выбора структуры швеллера, приложения “Оборудование: Металлоконструкции”.

Рисунок 2.4 – Интерфейс, каталога выбора структуры швеллера, приложения “Оборудование: Металлоконструкции”

**2.4.2 Оборудование: Кабели и жгуты**

Оборудование: Кабели и жгуты [8] – специализированное приложение для КОМПАС-3D [1], предназначенное для автоматизации процесса трехмерного моделирования электрических кабелей и жгутов, а также для выпуска конструкторской документации на эти изделия.

Приложение автоматически создает:

* трассы прокладки кабелей и жгутов в пространстве изделия;
* позиционирование кабельных частей соединителей по их приборным (блочным) частям;
* трехмерные модели кабелей и жгутов с учетом количества и диаметра проводников, проходящих по трассам;
* скругления в точках поворота трассы с учетом автоматически рассчитываемого условного диаметра кабеля или ветви жгута;
* сборочный чертеж кабеля или жгута;
* спецификацию к чертежу с подсчетом длин всех проводников и количества различных материалов;
* позиционные обозначения на сборочном чертеже;
* объекты спецификации типа «сборочная единица» для жгутов или кабелей.

В качестве базы данных компонентов используется любой источник трехмерных моделей. Это могут быть отдельные файлы 3D-моделей, библиотеки моделей, а также каталоги Библиотеки Стандартные Изделия.

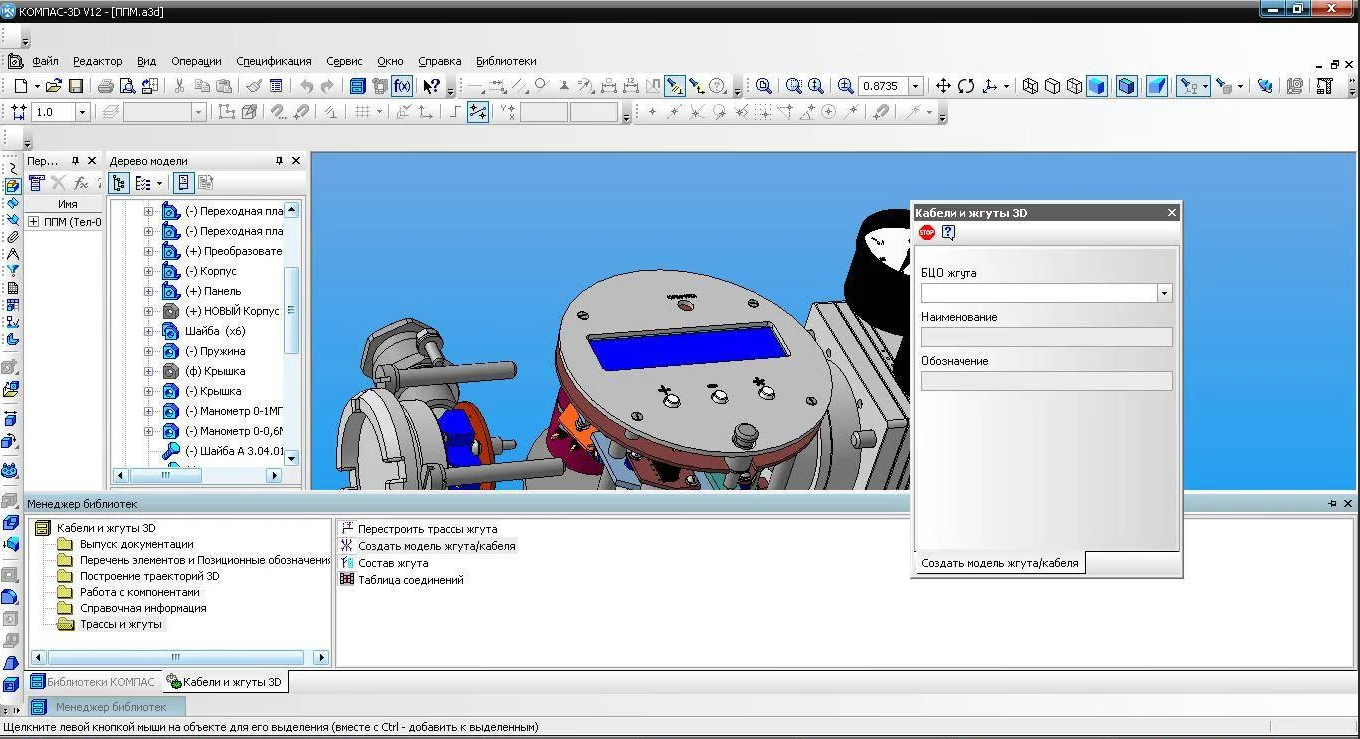
В качестве базы данных проводов и кабелей используются как специальные текстовые файлы в составе приложения, так и Библиотеку Материалы и Сортаменты.

Размеры на получаемых чертежах полностью соответствуют размерам кабелей и жгутов, измеренным по осям их трехмерных моделей. Суммарная длина проводников в спецификации дополнительно учитывает все припуски на монтаж и провисание, заданные пользователем еще на этапе формирования трасс. Пользователи могут добавлять к моделям разъемов таблицы с контактами, соответствующими им сигналами и маркировкой подводящих проводов. При этом данные из блоков и приборов об адресах электрической связи и сигналах автоматически передаются в кабельные части жгутов.

На сборочном чертеже жгута или кабеля конструктор при необходимости может размещать таблицы распайки разъемов.

Приложение применяется в приборостроении для моделирования соединений между блоками и разъемами в целом, а также в электротехнике, где должны моделироваться связи между отдельными контактами.

На рисунке 2.5 представлен интерфейс создания модели кабеля/жгута приложения “Оборудование: Кабели и жгуты”.

Рисунок 2.5 – Интерфейс создания модели кабеля/жгута приложения “Оборудование: Кабели и жгуты”

**3 Описание реализации**

На этапе разработки проекта программы для формального описания архитектурной особенности, пользовательского сценария системы был выбран унифицированный язык моделирования (UML) [9]. 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 плагина**

Прецеденты [9] – это технология определения функциональных требований к системе. Работа прецедентов заключается в описании типичных взаимодействий между пользователем системы и самой системой.

Прецеденты представляют собой ценный инструмент для понимания функциональных требований к системе.

Рассмотрим изначальную версию диаграммы прецедентов на рисунке 3.1.

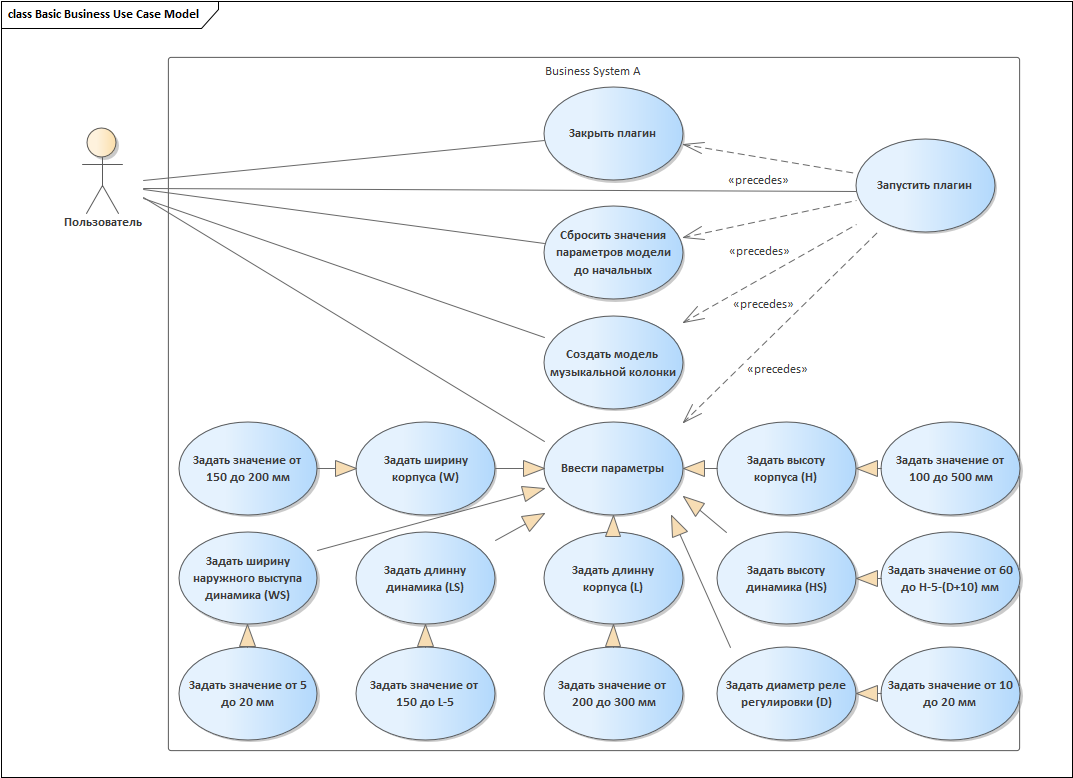


Рисунок 3.1 – Изначальная диаграмма прецедентов плагина

Действующим лицом выступает пользователь.

Для пользователя доступны следующие возможности:

* Сбросить значения параметров модели до начальных;
* Ввести параметры;
* Создать модель музыкальной колонки;
* Запустить плагин;
* Закрыть плагин.

В результате изменений, внесенных заказчиком, диаграмма прецедентов подверглась изменениям.

Добавлены возможности для задания количества и формы динамиков.

Рассмотрим измененную диаграмму прецедентов плагина на рисунке 3.2.

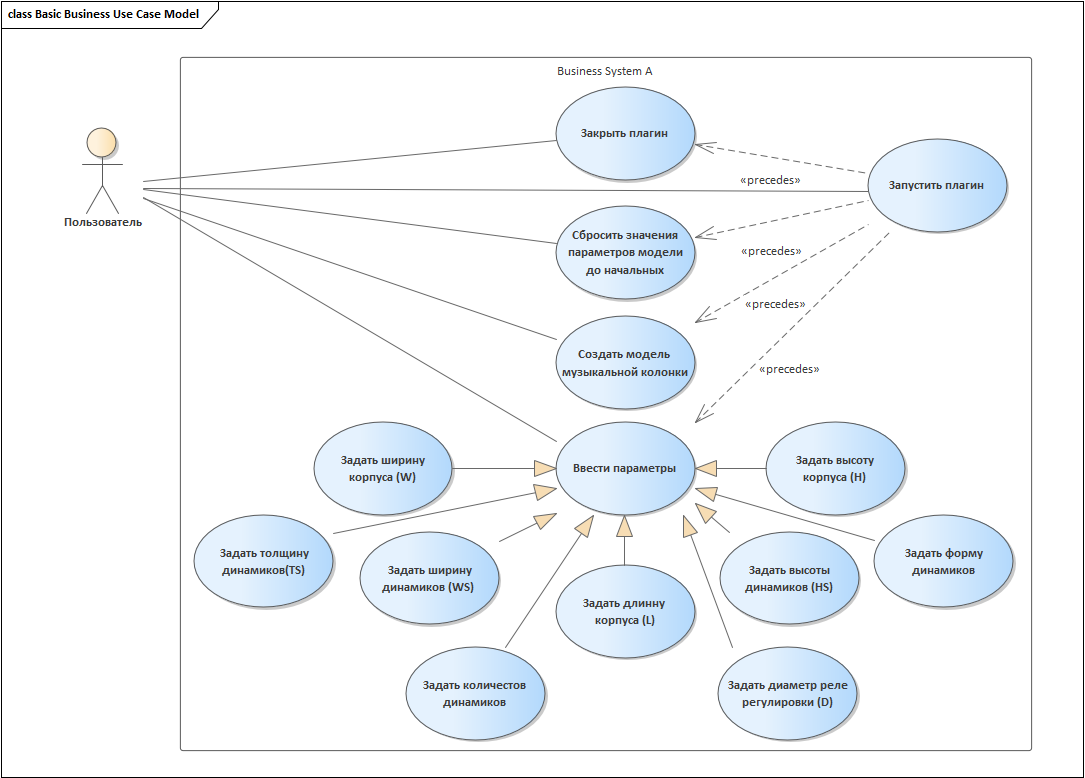


Рисунок 3.2 – Измененная диаграмма прецедентов плагина

**3.2 Диаграмма классов**

Диаграмма классов [9] описывает типы объектов системы и различного рада статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами. В UML термин функциональность применяется в качестве основного термина, описывающего и свойства, и операции класса. Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы.

Рассмотрим изначальную версию диаграммы классов на рисунке 3.3.

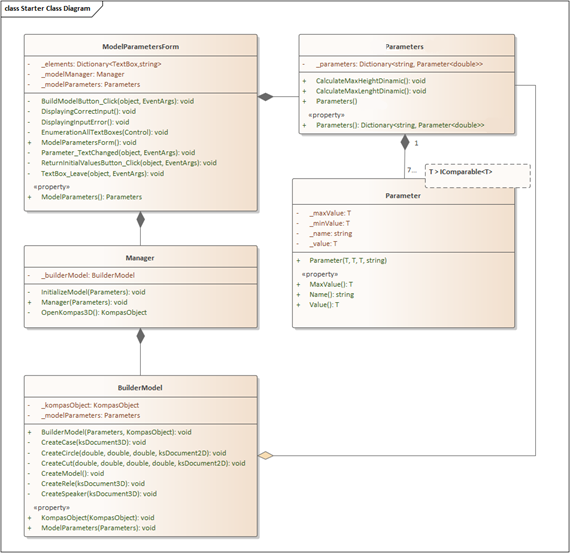


Рисунок 3.3 – Изначальная диаграмма классов плагина

Элемент модели это одна из структурных составляющих модели. В данном случае обязательными элементами являются корпус, динамик 1, реле регулировки, и не обязательными являются динамик 2, динамик 3, динамик 4.

Параметром элемента модели является числовая величина, представляющая определенное геометрическое свойство элемента модели.

Класс Parameter хранит данные параметра модели.

Класс ElementParameters хранит словарь параметров модели.

Класс ModelElement хранить параметры элемента модели и реализует методы для преобразования параметров элемента при изменении формы.

Класс ModelElements хранить словарь элементов модели и реализует методы для преобразования параметров элементов в зависимости от изменения взаимосопряженных параметров.

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

Класс Manager хранит экземпляр класса построителя модели и реализует методы подключения к САПР КОМПАС-3D.

Класс BuilderModel хранит ссылку на экземпляр класса ModelElements и реализует методы построения модели.

В результате изменений, внесенных заказчиком, диаграмма классов подверглась изменениям.

Был добавлен класс ModelElement необходимый для хранения всех элементов модели и осуществления управления параметрами в отдельном классе.

Был добавлен класс ModelElement необходимый для хранения параметров элемента модели отдельном классе.

Были добавлены перечисления ParametersName, ElementFormKey и ElementName, для уменьшения вероятности ошибки при введении названий элементов, параметров и форм.

Рассмотрим измененную диаграмму классов плагина на рисунке 3.4

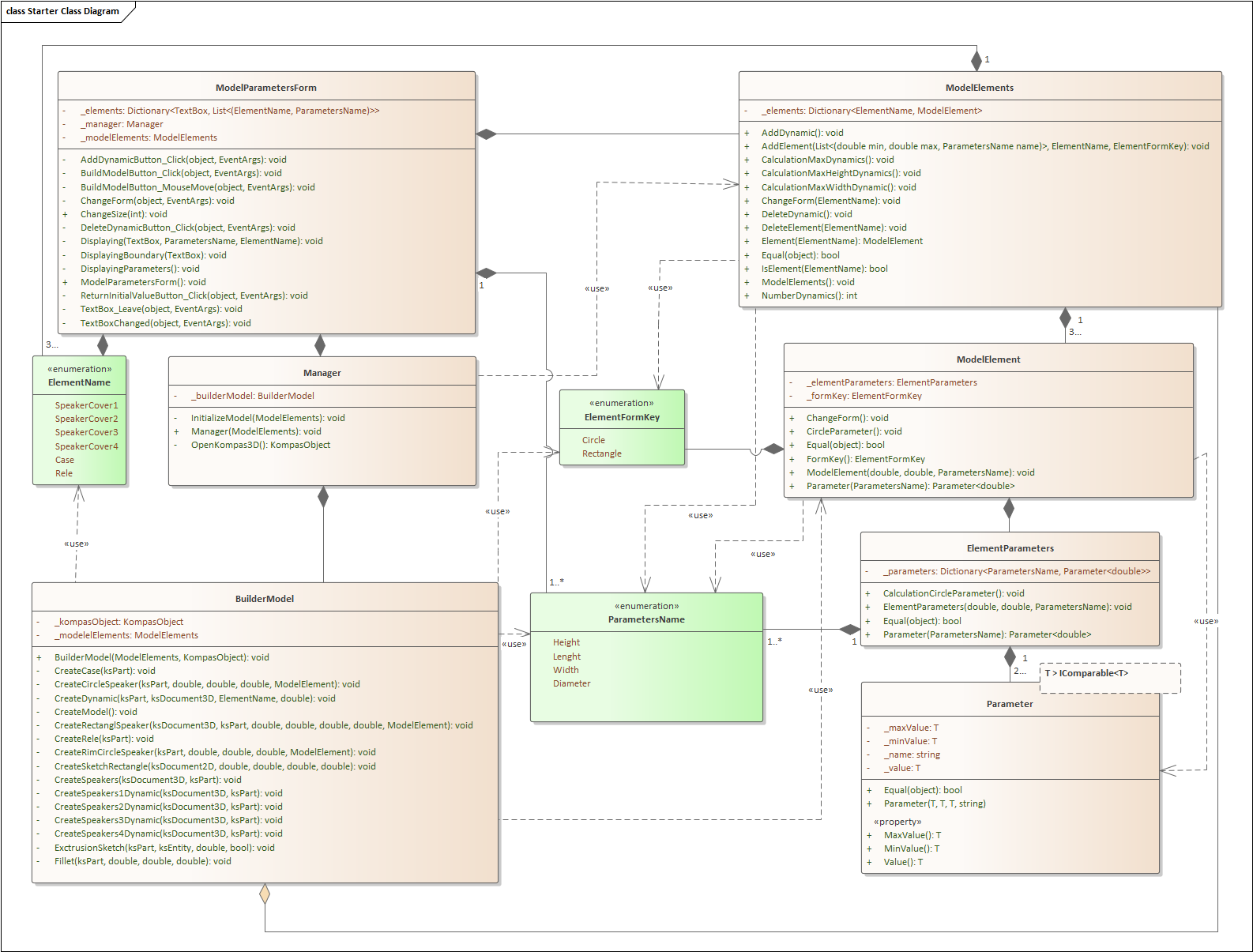


Рисунок 3.4 – Измененная диаграмма классов плагина

**3.3 Диаграмма пакетов**

Пакет (package) [9] – это инструмент группирования, который позволяет взять любую конструкцию UML объединить ее элементы в единицы высокого уровня. В основном пакеты служат для объединения классов в группы, но могут применяться для любой другой конструкции языка UML.

Рассмотрим изначальную версию диаграммы пакетов на рисунке 3.5.

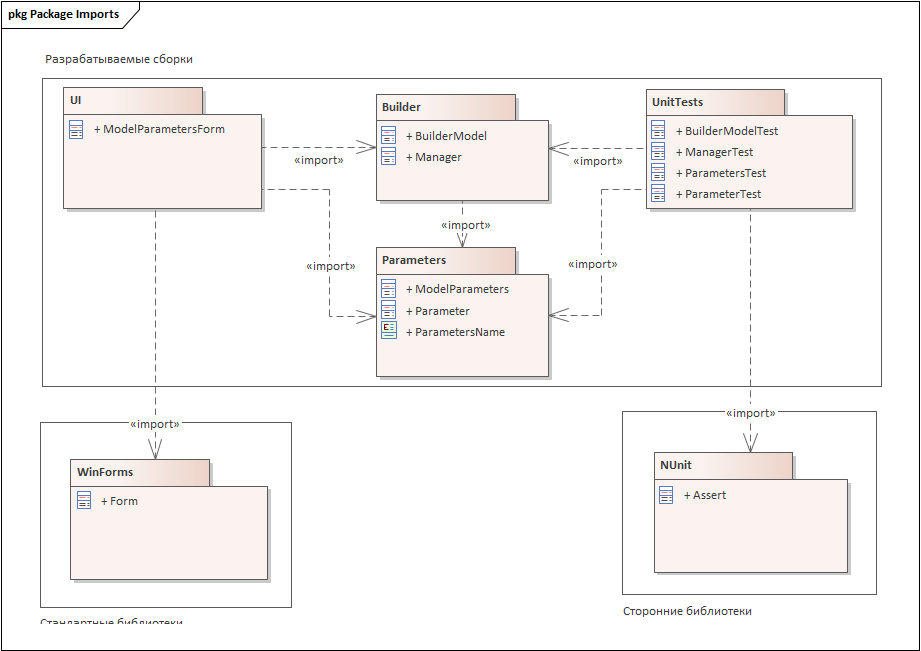


Рисунок 3.5 – Изначальная диаграмма пакетов плагина

Пакеты Builder и Parameters содержат классы бизнес логики плагина.

Пакет Builder хранит классы для построения модели использует Parameters.

Пользовательский интерфейс содержится в пакете UI, при работе данный пакет использует бизнес логику приложения (пакеты Builder и Parameters) и графический интерфейс (пакет WinForms).

Пакет UnitTests хранит классы с юнит-тестами приложения, для работы используется вспомогательный пакет NUnit.

Пакет Parameters содержит классы для хранения параметров модели.

В результате изменений, внесенных заказчиком, диаграмма пакетов подверглась изменениям. В результате были добавлены новые классы и перечисления в пакеты UnitTests и Parameters.

Рассмотрим измененную диаграмму пакетов плагина на рисунке 3.6.

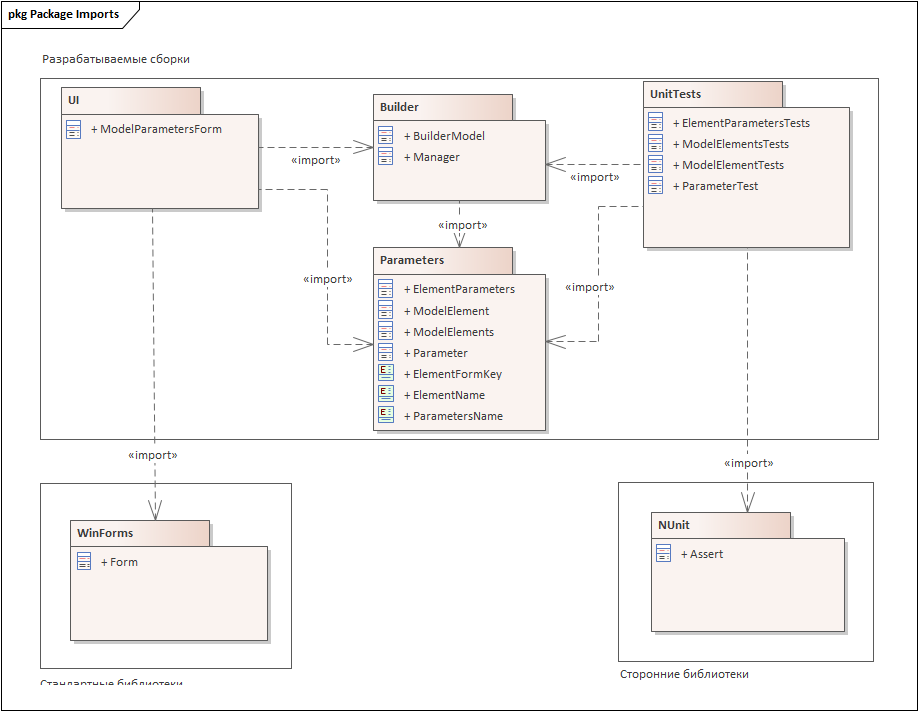


Рисунок 3.6 – Измененная диаграмма пакетов плагина

**4 Описание программы для пользователя**

Плагин состоит из одного диалогового окна с условным разделением на функциональные блоки: “Габариты корпуса”, “Габариты реле регулировки”, “Параметры динамиков”, “Количество динамиков”.

Начальный вид диалогового окна изображен на рисунке 4.1.

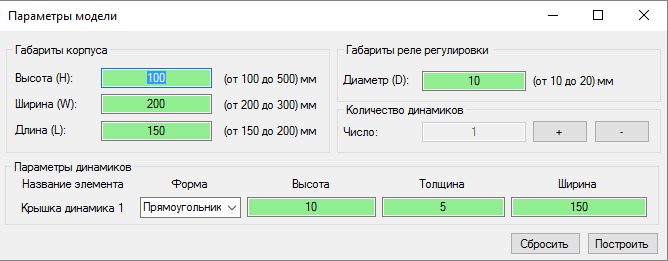


Рисунок 4.1 – Начальный вид диалогового окна

При запуске программы строки “Крышка динамика 2”, “ Крышка динамика 3”, “ Крышка динамика 4” не видны и будут проявляться по мере добавления пользователем числа динамиков.

Диалоговое окно плагина с отображением всех строк отображен на рисунке 4.2.

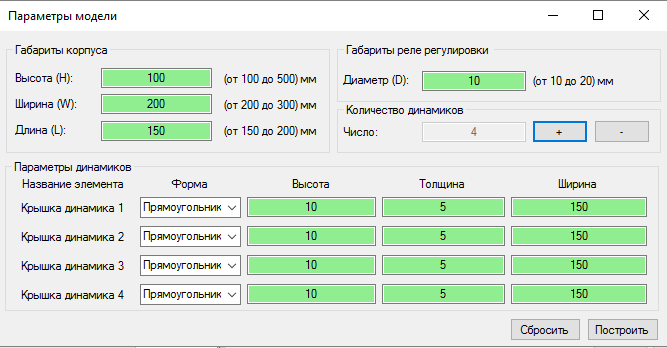


Рисунок 4.2 – Диалоговое окно с отображением всех блоков

При введении значения не входящего в отображенный интервал, поле окрашивается в красный цвет и будет оставаться красным до введения корректного значения или до потери полем фокуса, при потере фокуса, значение поля примет последнее корректное значение и поле окраситься в зеленый цвет. Данная функция работает для всех полей, окрашенных в зеленый цвет при первоначальном отображении.

Интервал возможных значений для полей блока “Параметры динамиков” отображается при наведении на поле для ввода параметра.

Диалоговое окно с введенным некорректным значением и отображенным интервалом параметров изображено на рисунке 4.3.

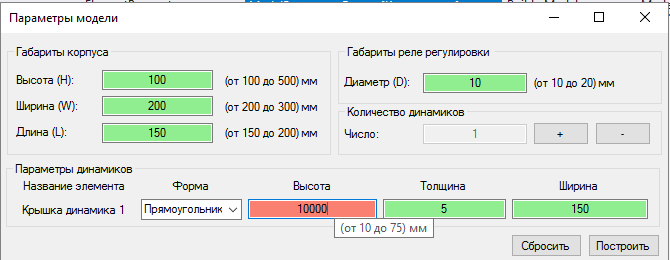


Рисунок 4.3 – Диалоговое окно с некорректно введенным параметром

Блок “Габариты корпуса” состоит из 3 пунктов с возможностью ввода значений, названий пунктов и интервала принимаемых параметров.

При изменении значения поля “Высота (H)” происходит перерасчет интервала высот для всех динамиков.

При изменении значения поля “Ширина (W)” происходит перерасчет интервала широты для всех динамиков.

От значения “Длина (L)” другие параметры не зависят.

Блок “Габариты реле регулировки” состоит из 1 пункта с возможностью ввода значения, названия пункта и интервала принимаемого параметра.

От значения “Диаметр (D)” зависит максимальное значение высоты динамиков.

Блок “Количество динамиков” состоит из 1 пункта, значение которого изменяется с помощью кнопки с надписью “-”, уменьшающей количество динамиков и кнопки с надписью “+”, увеличивающей количество динамиков и названия пункта. Возможное число динамиков от 1 до 4. При изменении количества динамиков, происходит отображение необходимого количества строк в таблице параметров.

Блок “Параметры динамиков” состоит из таблицы параметров динамиков.

При изменении значения поля “Высота (HS)” происходит перерасчет значений максимальных высот других добавленных динамиков.

От значений “Ширина (WS)” и “Толщина (TS)” другие параметры не зависят.

При изменении значения формы с “Прямоугольник” на “Круг”, поле “Ширина (WS)” перестает отображаться.

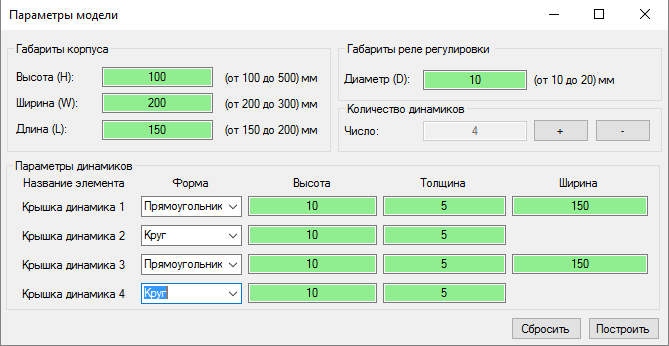


Рисунок 4.4 – Диалоговое окно с установленной круговой формой динамиков

При установке формы “Круг” максимальное значение поля “Высота (H)” зависит от значений полей “Высота (H)” и “Ширина (W)” блока “Габариты корпуса”.

Нажатие кнопки “Сбросить” возвращает параметры модели к начальным значениям. Нажатие кнопки доступно с момента запуска программы.

Нажатие кнопки “Построить” открывает систему автоматизированного проектирования Компас – 3D и создает новый пустой документ типа “Деталь” в котором производиться построение модели, если САПР Компас – 3D уже открыта, то произойдет создание только нового документа типа “Деталь”. Нажатие кнопки возможно в любой момент в течении времени заполнения параметров, начиная с момента запуска программы.

Выделенная фокусом кнопка “Построить” на диалоговом окне изображена на рисунке 4.5.

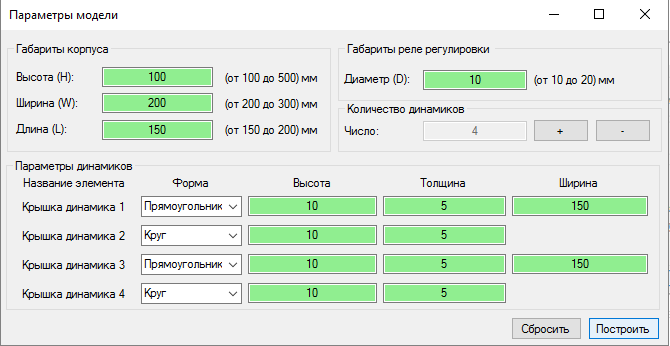


Рисунок 4.5 – Диалоговое окно с выделенной кнопкой “Построить”

Диалоговое окно документа САПР КОМПАС 3D после построения модели изображено на рисунке 4.6.

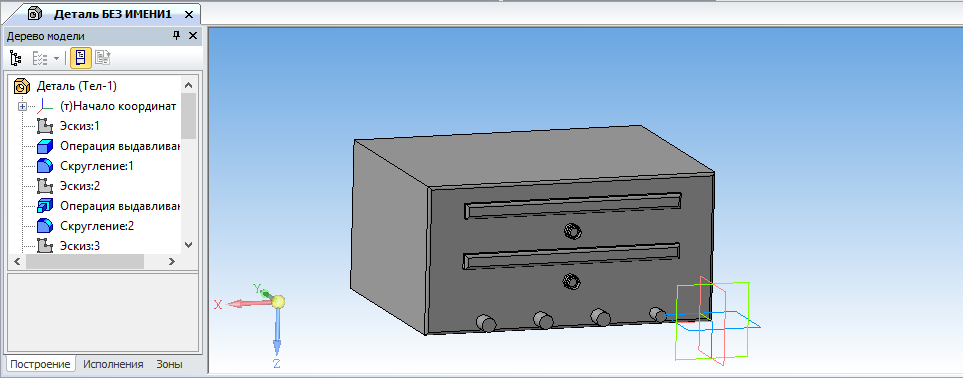


Рисунок 4.6 - Диалоговое окно документа САПР КОМПАС 3D после построения модели

**5 Тестирование программы**

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

**5.1 Функциональное тестирование**

При функциональном тестировании [10] проверялась корректность работы плагина “Музыкальная колонка”, а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами.

Проведем тестирование максимальных и минимальных параметров модели.

Модель с минимальными параметрами представлена на рисунке 5.1.

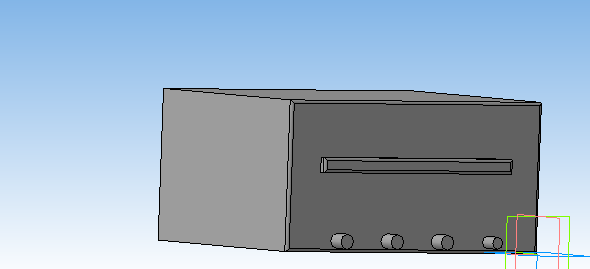


Рисунок 5.1 – Модель музыкальной колонки с минимальными возможными параметрами

Модель с максимальными параметрами представлена на рисунке 5.2.

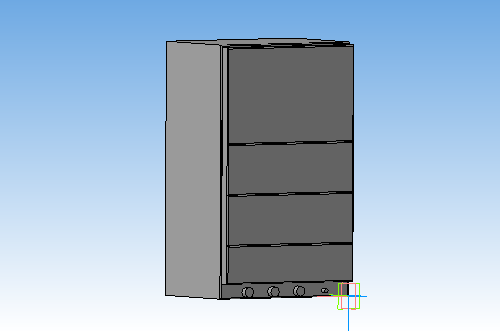


Рисунок 5.2 – Модель музыкальной колонки с максимальными возможными параметрами

Проведем тестирование построения модели с динамиками круглой формы. Результаты тестирования представлены на рисунке 5.3.

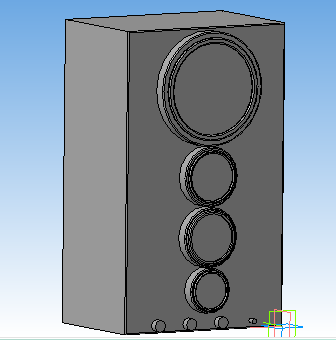


Рисунок 5.3 – Модель музыкальной колонки с динамиками круглой формы

Проведем тестирование построения музыкальной колонки с 2 и 3 динамиками. Построение 1 и 4 динамиков уже тестировались на (Рисунок 5.1) и (Рисунок 5.2).

Результат построения модели с 2 динамиками изображено на рисунке 5.4.

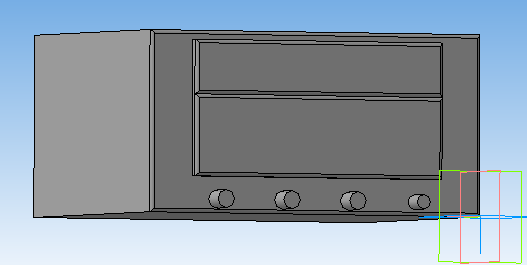


Рисунок 5.4 – Модель музыкальной колонки с 2 динамиками

Результат построения модели с 3 динамиками изображено на рисунке 5.5.

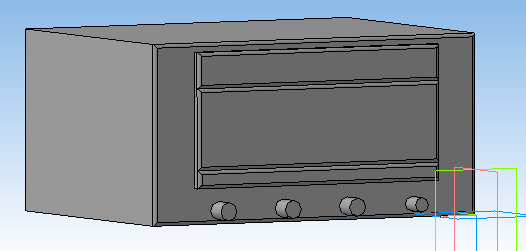


Рисунок 5.5 – Модель музыкальной колонки с 3 динамиками

**5.2 Модульное тестирование**

В целях проверки корректности работы методов и свойств классов при помощи обозревателя тестов Visual Studio было проведено модульное тестирование [11], проверялись открытые поля и методы, для этого были созданы тестовые классы:

* ElementParametersTests – класс тестирующий свойства и методы класса ElementParameters. Описание класса ElementParametersTests в Приложении А (Таблица А.1);
* ModelElementsTests – класс тестирующий свойства и методы класса ModelElements. Описание класса ModelElementsTests в Приложении А (Таблица А.2);
* ModelElementTests – класс тестирующий свойства и методы класса ModelElement. Описание класса ModelElementTests в Приложении А (Таблица А.3);
* ParameterTests – класс тестирующий свойства и методы класса Parameter. Описание класса ParameterTests в Приложении А (Таблица А.4);

Диалоговое окно состояний запущенных тестов для классов ElementParametersTests и ModelElementTests изображены на рисунке 5.6.

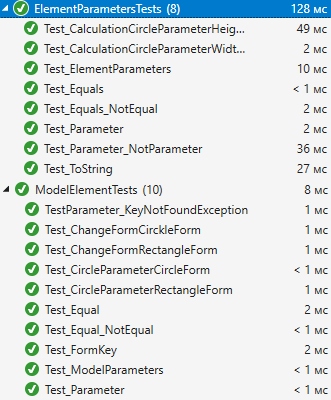
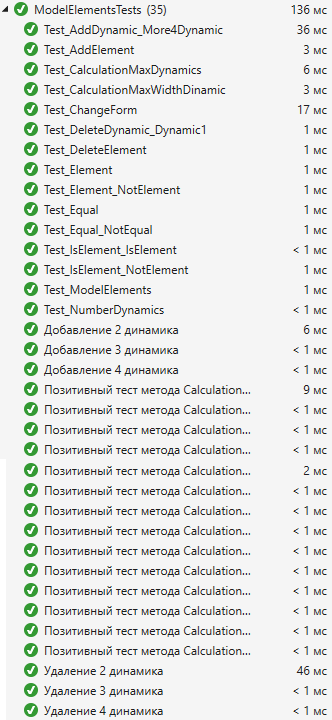


Рисунок 5.6 - Диалоговое окно состояний запущенных тестов для классов ElementParametersTests и ModelElementTests

Диалоговое окно состояний запущенных тестов для класса ModelElementsTests изображены на рисунке 5.7.

  
Рисунок 5.7 - Диалоговое окно состояний запущенных тестов для классов ModelElementsTests

Диалоговое окно состояний запущенных тестов для класса ParameterTests изображены на рисунке 5.8.

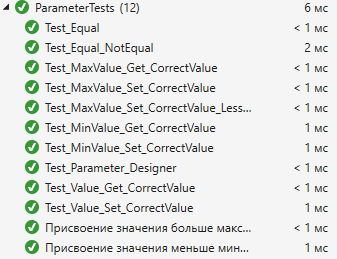


Рисунок 5.8 - Диалоговое окно состояний запущенных тестов для классов ParameterTests

**5.3 Нагрузочное тестирование**

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование [12]. Для проведения нагрузочного тестирования был добавлен таймер, который засекал время от начала до конца построения. Измерялись потребляемые ресурсы процессора и потребляемая оперативная память.

Проведем тестирование для Windows 10 x32, установленной на SSD.

После построения 85 деталей программа КОМПАС-3D выдала сообщение о невозможности создания новой детали.

Начиная с построения 47 детали, общая загрузка центрального процессора достигла максимального значения и затраты оперативной памяти начали уменьшаться, за счет увеличения времени построения детали. Время построения детали при построении первого экземпляра составляло 8.16 секунд, время построения 85 детали 33.69 секунды.

Проведем тестирование для Windows 10 x64, установленной на HDD.

После построения 61 детали программа перестала отвечать и не возобновила работу.

Начиная с построения 20 детали, общая загрузка центрального процессора достигла максимального значения и затраты оперативной памяти начали уменьшаться, за счет увеличения времени построения детали. Начиная с 37 детали общая загрузка диска достигла максимального значения. Время построения детали при построении первого экземпляра составляло 10.4 секунд, время построения 61 детали 183.69 секунды.

Графики потребляемой оперативной памяти относительно числа созданных деталей изображены на рисунке 5.9. Где y – количество используемой оперативной памяти, в мегабайтах, а x – число построенных деталей. Синий график – HDD x64, желтый – SSD x32.

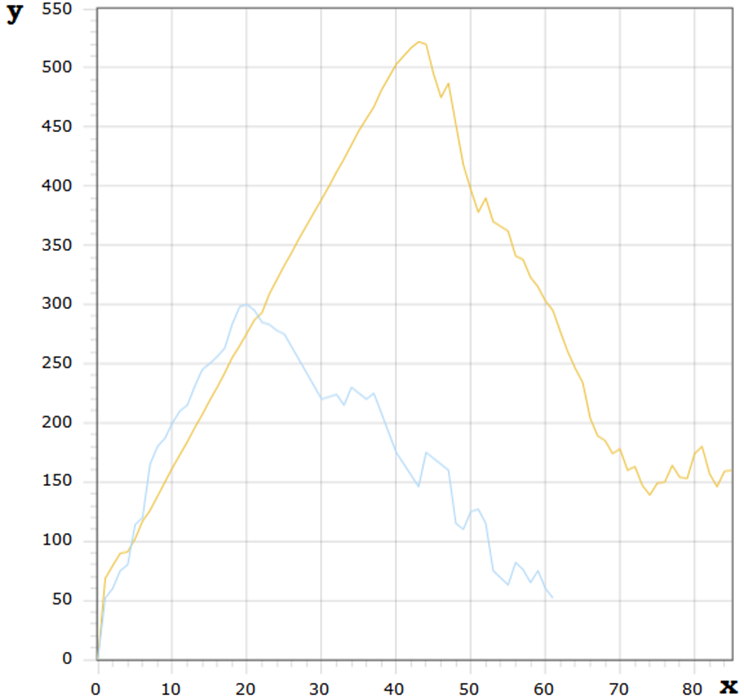


Рисунок 5.9 – Графики использования оперативной памяти относительно числа построенных деталей

Графики зависимости загрузки программой центрального процессора от количества построенных деталей изображены на рисунке 5.10. Где y – загрузка центрального процессора в %, а x – число построенных деталей. Синий график – HDD x64, желтый – SSD x32.

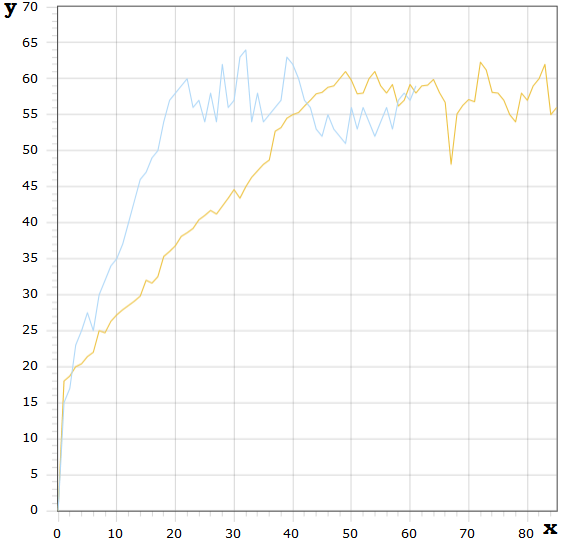


Рисунок 5.10 – Графики использования ресурсов центрального процессора относительно числа построенных деталей

При наличии свободных ресурсов оперативной памяти и процессора, зависимость между количеством построенных деталей и нагрузкой на ресурсы линейна.

При израсходовании свободных ресурсов нагрузка на процессор остается постоянной, но затраты оперативной памяти уменьшаются, так как время построения увеличивается.

**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 «Музыкальная колонка», проведены модульные, функциональные и нагрузочные тесты.

**Список использованных источников**

1. КОМПАС – 3D [Электронный ресурс]. – URL: <https://kompas.ru/> (дата обращения: 13.04.2020);
2. Акустическая система. Основные понятия [Электронный ресурс]. – URL: <http://ldsound.ru/akusticheskaya-sistema-osnovnye-ponyatiya/>(дата обращения: 13.04.2020);

# ГОСТ 23262-88 Системы акустические бытовые. Общие технические условия [Электронный ресурс]. – URL:<https://internet-law.ru/gosts/gost/11778/> (дата обращения: 13.04.2020);

1. ГОСТ 15.009-91 Система разработки и постановки продукции на производство. Непродовольственные товары народного потребления [Электронный ресурс]. – URL: <https://internet-law.ru/gosts/gost/38525/> (дата обращения: 13.04.2020);
2. Microsoft Visual Studio 2019 [Электронный ресурс]. – URL: <https://visualstudio.microsoft.com/ru/vs/> (дата обращения: 13.04.2020);

# Базовые интерфейсы API системы КОМПАС [Электронный ресурс]. – URL: <https://it.wikireading.ru/23741> (дата обращения: 13.04.2020);

1. Приложение “Оборудование: Металлоконструкции” [Электронный ресурс]. – URL: <https://kompas.ru/kompas-3D/application/machinery/steel-constructions-3d/> (дата обращения: 13.04.2020);
2. Приложение “Оборудование: Кабели и жгуты” [Электронный ресурс]. – URL: <https://kompas.ru/kompas-3D/application/instrumentation/equipment-cables-and-harness/> (дата обращения: 13.04.2020);
3. Мартин Фаулер UML Основы. Краткое руководство по стандартному языку объектного моделирования [Электронный ресурс]. – URL: litportal.ru›trial/pdf/24500318.pdf (дата обращения: 13.04.2020);
4. Функциональное тестирование [Электронный ресурс]. – URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения: 13.04.2020);
5. Модульное тестирование [Электронный ресурс]. – URL: <http://espressocode.top/unit-testing-software-testing/> (дата обращения: 13.04.2020);
6. 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 тестовых случаев класса ElementParameters представлено в таблице А.1

|  |  |
| --- | --- |
| Наименование | Описание |
| − \_values: List<(double, double, ParametersName)> | Поле хранит данные параметров модели |
| -\_elementParameters: ElementParameters | Поле хранит данные элемента модели |
| − CreateParameters() | Метод заполняет поля \_values и \_elementParameters тестовыми данными |
| + Test\_Parameter() | Метод для проверки корректности возвращаемых методом Parameter данных |
| + Test\_ToString() | Метод для проверки корректности преобразования элемента перечисления в строку |
| + Test\_CalculationCircleParameterWidthMoreHeight() | Метод для проверки корректности перерасчета параметров при круговой форме при ширине > высоты |
| + Test\_CalculationCircleParameterHeightMoreWidth() | Метод для проверки корректности перерасчета параметров при круговой форме при ширине < высоты |

Продолжение таблицы А.1.

|  |  |
| --- | --- |
| + Test\_Equals() | Метод для проверки корректности сравнения одинаковых элементов |
| + Test\_Equals\_NotEqual() | Метод для проверки корректности сравнения различных элементов |
| + Test\_ElementParameters() | Метод для проверки корректности работы конструктора |

Таблица А.1 – Описание полей и методов класса ElementParametersTests

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

|  |  |
| --- | --- |
| Наименование | Описание |
| − \_values: List<(double, double, ParametersName)> | Поле хранит данные параметров модели |
| \_modelElement: ModelElement | Поле хранит данные элемента модели |
| + CreateParameters() | Метод заполняет поля \_values и \_modelElement тестовыми данными |
| + Test\_Parameter() | Метод для проверки корректности возвращаемых методом Parameter данных |
| + TestParameter\_KeyNotFoundException() | Метод для проверки вызова исключения при отсутствии параметра в словаре |
| + Test\_FormKey() | Метод для проверки корректности возврата ключа формы |
| + Test\_CircleParameterRectangleForm() | Метод для проверки изменения параметров при изменении ключа формы при прямоугольной форме |

Продолжение таблицы А.2.

|  |  |
| --- | --- |
| + Test\_CircleParameterCircleForm() | Метод для проверки изменения параметров при изменении ключа формы при круглой форме |
| + Test\_ChangeFormRectangleForm() | Метод для проверки изменения ключа формы при прямоугольной форме |
| + Test\_ChangeFormCirckleForm() | Метод для проверки изменения ключа формы при круглой форме |
| + Test\_ModelParameters() | Метод для проверки корректности работы конструктора |
| + Test\_Equals() | Метод для проверки корректности сравнения одинаковых элементов |
| + Test\_Equals\_NotEqual() | Метод для проверки корректности сравнения различных элементов |

Таблица А.2 – Описание полей и методов класса ModelElementTests

Описание полей и методов используемых для проверки тестовых случаев класса ModelElements представлено в таблице А.3

|  |  |
| --- | --- |
| Наименование | Описание |
| − \_modelElements: ModelElements | Поле хранит элементы модели |
| + CreateParameters() | Метод заполняет поле \_modelElements тестовыми данными |
| + Test\_Element() | Метод для проверки корректности возвращаемого элемента |
| + Test\_Element\_NotElement() | Метод для проверки вызова исключения при отсутствии элемента |
| + Test\_IsElement\_IsElement() | Метод для проверки наличия элемента при его наличии |
| + Test\_IsElement\_NotElement() | Метод для проверки наличия элемента при его отсутствии |

Продолжение таблицы А.3.

|  |  |
| --- | --- |
| + Test\_AddElement() | Метод для проверки корректности добавления элемента |
| + Test\_Test\_AddDynamic(int, double) | Метод для проверки корректности добавления динамика |
| + Test\_AddDynamic\_More4Dynamic  () | Метод для проверки работы метода при существовании 4 динамиков |
| + Test\_DeleteDynamic (int, double) | Метод для проверки корректности удаления динамика |
| + Test\_DeleteDynamic\_Dynamic1() | Метод для проверки корректности удаления динамика при существовании минимального числа динамиков |
| + Test\_DeleteElement() | Метод для проверки корректности удаления элемента |
| + Test\_CalculationMaxHeightDynamic\_  Dynamic1(ElementName) | Метод для проверки корректности расчета высоты 1 динамика |
| + Test\_CalculationMaxHeightDinamic\_  Dynamic2(ElementName) | Метод для проверки корректности расчета высоты 2 динамиков |
| + Test\_CalculationMaxHeightDynamic\_  Dynamic3(ElementName) | Метод для проверки корректности расчета высоты 3 динамиков |
| + Test\_CalculationMaxHDinamic\_Dinamic4(ElementName) | Метод для проверки корректности расчета высоты 4 динамиков |
| + Test\_CalculationMaxHeightDynamic  (int , double) | Метод для проверки корректности расчета максимальной высоты всех динамиков |
| + Test\_ChangeForm() | Метод для проверки корректности изменения параметров при изменении формы |

Продолжение таблицы А.3.

|  |  |
| --- | --- |
| + Test\_CalculationMaxWidthDinamic() | Метод для проверки корректности расчета максимальной ширины динамиков |
| + Test\_CalculationMaxDynamics() | Метод для проверки корректности расчета максимальной суммы высот динамиков |
| + Test\_NumberDynamics() | Метод для проверки корректности возвращения количества динамиков |
| + Test\_ModelElements() | Метод для проверки корректности работы конструктора |
| + Test\_Equals() | Метод для проверки корректности сравнения одинаковых элементов |
| + Test\_Equals\_NotEqual() | Метод для проверки корректности сравнения различных элементов |

Таблица А.3 – Описание полей и методов класса ModelElementTests

Описание полей и методов используемых для проверки тестовых случаев класса Parameters представлено в таблице А.4.

|  |  |
| --- | --- |
| Наименование | Описание |
| − \_parameter: Parameter<double> | Поле хранит данные параметра |
| + CreateParameters() | Метод заполняет поле \_parameter тестовыми данными |
| + Test\_MaxValue\_Set\_CorrectValue() | Метод для проверки корректности записи максимального значения |
| + Test\_MaxValue\_Set\_  CorrectValue\_LessMinValue() | Метод для проверки корректности записи максимального значения при значении меньше минимального |
| + Test\_MaxValue\_Get\_CorrectValue() | Метод для проверки корректности возврата максимального значения |

Продолжение таблицы А.4.

|  |  |
| --- | --- |
| + Test\_MinValue\_Set\_CorrectValue() | Метод для проверки корректности записи минимального значения |
| + Test\_MinValue\_Get\_CorrectValue() | Метод для проверки корректности возврата минимального значения |
| + Test\_Value\_Set\_CorrectValue() | Метод для проверки корректности записи текущего значения |
| + Test\_Value\_Get\_CorrectValue() | Метод для проверки корректности возврата текущего значения |
| + Test\_Parameter\_Designer() | Метод для проверки корректности работы конструктора |
| + TestLastModTimeSet\_ArgumentException  (string, string) | Метод для проверки вызова исключения при записи некорректных значения |
| + Test\_Equals() | Метод для проверки корректности сравнения одинаковых элементов |
| + Test\_Equals\_NotEqual() | Метод для проверки корректности сравнения различных элементов |

Таблица А.4 – Описание полей и методов класса ParameterTests