Реферат

Пояснительная записка имеет объём 64 страницы и содержит 6 таблицы, 17 рисунков и список из 22 использованных источников.

Содержание записки характеризуют следующие ключевые слова: автоматизация конструирования, библиотечный элемент вакуумных фильтрова, твердотельное моделирование, геометрические параметры.

Объект исследования или разработки – «Подсистема проектирования вакуумных фильтров на основе NX Open API».

Цель дипломного проекта – Разработка подсистемы моделирования вакуумного фильтра средствами API функций базовой CAD системы (NX).

Полученные результаты и их новизна - результатом выполнения дипломного проекта является реализация подсистемы проектирования вакуумных фильтров средствами NX Open API.

Новизна, разработанной САПР, заключается в частичной автоматизации трудоемкого процесса построения типовых элементов конструкции вакуумных фильтров на основе использования современного программного и информационного обеспечения, а также новых информационных технологий.

Область применения – машиностроительные предприятия.

Прогнозные предположения о развитии объекта исследования – данное ПС является актуальным, так как в современном производстве усиливаются тенденция перехода к автоматизации проектирования и конструкторских работ.

Содержание

Введение

Увеличение производительности труда разработчиков новых изделий, сокращение сроков проектирования, повышение эффективности общественного производства и улучшение качества продукции – важнейшие проблемы, которые определяет уровень ускорения научно-технического прогресса общества.

Развитие систем автоматизированного проектирования (САПР) опирается на прочную научно-техническую базу. Это – современные средства вычислительной техники; новые способы представления и обработки информации, основанные на принципах искусственного интеллекта; создание новых численных методов решения инженерных задач и оптимизации.

Машиностроение – одна из сложнейших и ответственных областей промышленности. Его развитие является одной из предпосылок экономического роста, так как снижение себестоимости и повышение качества продукции позволяет быть конкурентоспособной отечественной продукции на мировом рынке.

САПР в настоящее время находят все более широкое применение в различных отраслях промышленности. Дополнительный толчок развитию и распространению САПР дало создание в промышленности гибких производственных систем (ГПС). Однако область применения САПР - это не только обеспечение функционирования ГПС. Автоматизированное проектирование целесообразно применять и при создании многих технических объектов независимо от того, автоматизировано или не автоматизировано их производство [1].

Внедрение в промышленную область САПР приводит к значительной экономии затрат на создание и эксплуатацию проектируемых изделий, повышению производительности труда проектировщиков, конструкторов и технологов, снижению объема проектной документации.

Программные средства являются непосредственной производительной силой, так как от них в ряде случаев зависят эффективность промышленного производства и качество продукции, создаваемой в технологическом процессе с применением ЭВМ. В то же время они наиболее гибкая и модернизируемая часть систем, обеспечивающая относи-

тельно легкую адаптацию к изменяющимся условиям в процессе развития техники и к особенностям конкретного применения.

Целью выпускной квалификационной работы является создание программной подсистемы проектирования вакуумных фильтров, средствами автоматизации системы NX версии 7.5.

Для ускорения процесса создания системы, она разбита на отдельные функциональные модули, которые объединяет управляющая подсистема. Работа над проектом осуществляется в пакете MS Visual Studio 2013.

В представленной работе будет дано описание программной реализации подсистемы проектирования вакуумных фильтров. Процесс проектирования программы включает такие этапы, как: исследование предметной области, постановка задачи, выбор метода решения, разработка алгоритма, программная реализация задачи.

Созданное программное средство является актуальным и может быть применено на предприятиях, в которых закуплен программный продукт компании Siemens – NX 7.5 и Teamcenter, а также применяется при конструировании изделий типа фильтр. Положительные результаты решения задачи по созданию подсистемы позволят повысить экономическую эффективность автоматизации проектных работ, за счет роста производительности труда конструкторов и повышения качества проектных решений и проектируемых изделий [2].

Развитие новых технологий постоянно ужесточает требования, предъявляемые к инженеру-конструктору. На первое место в современном конструировании выходят скорость и динамичность выполнения проектов, как расчетных составляющих, так и в виде чертежей или моделей, а также возможность быстрого внесения в них изменений, без какого бы то ни было влияния на качество выполняемых работ или проектируемого объекта. Наверное, каждый инженер неоднократно сталкивался с задачей выполнения многократного конструирования однотипных элементов с применением большого количества справочных материалов, когда объекты вроде бы и не очень различаются, но и приходится производить полный пересмотр параметров и характеристик. Для решения этой проблемы целесообразно применять средства программирования и параметриза-

ции, посредством которых можно задать определенные связи между отдельными элементами объекта, позволяющие при последующей разработке типовых конструкций не переделывать все его свойства, а изменить лишь несколько параметров, и провести полный перерасчет и моделирование объекта с применением автоматизированных средств. Это дает возможность многократно использовать единожды построенный алгоритм, значительно сокращающий время на создание новых конструкций.

1 Характеристика существующей системы САПР

1.1 Характеристика библиотек конструирования изделий

Широкое внедрение компьютеризации в условиях научно-технического прогресса обеспечивает рост производительности труда в различных областях общественного производства. Главное внимание при этом обращается на те области, где рост производительности труда до применения ЭВМ проходил крайне медленно. Это, в первую очередь, области, связанные с приложением умственного труда человека, т.е. управление производством, проектирование и исследование объектов и процессов. Если производительность труда в сфере производства с начала века возросла в сотни раз, то в области проектирования только в 1.5 - 2 раза [2, 3]. Это обусловливает большие сроки проектирования новых объектов, что не отвечает потребностям развития экономики.

Прогресс производства в современных условиях связывают с достижениями в области автоматизации производства. Поскольку проектирование и разработка технологии являются ступенью производства (логическим уровнем), то прогресс на этой ступени также должен определяться автоматизацией.

Автоматизированное проектирование позволяет значительно сократить субъективизм при принятии решений, повысить точность расчетов, выбрать наилучшие варианты для реализации на основе строгого математического анализа всех или большинства вариантов проекта с оценкой технических, технологических и экономических характеристик производства и эксплуатации проектируемого объекта, значительно повысить качество конструкторской документации, существенно сократить сроки проектирования и передачи конструкторской документации в производство, эффективнее использовать технологическое оборудование с программным управлением. Автоматизация проектирования способствует более полному использованию унифицированных изделий в качестве стандартных компонентов проектируемого объекта.

Объединение нескольких ПП в единую систему, предназначенную для реализации вполне определенных функций, позволяет говорить о новом, более высоком уровне в иерархии программных комплексов, т.е. САПР. При этом качественные изменения

претерпевают и организация информационного, технического и других видов обеспечения, и, что особенно важно, условия обмена информацией между человеком и ЭВМ. Как правило, эти изменения направлены на повышение гибкости и универсальности системы, улучшение характеристик взаимодействия проектировщика с ЭВМ, повышение качества получаемого результата и снижение времени его получения. Собственно, САПР могут в качестве подсистемы входить в системы более высокого уровня, например, АСУП (автоматизированных систем управления производством).

Формальное определение САПР, определяющее ее главные особенности, — человеко-машинная система, использующая современные математические методы, средства электронно-вычислительной техники и связи, а также новые организационные принципы проектирования для нахождения и практической реализации наиболее эффективного проектного решения существующего объекта.

Прошло время не связанных друг с другом программ и систем, автоматизирующих отдельные звенья технологической цепи производства, как это было на заре компьютерной эры. Теперь пользователь требует от разработчиков законченные решения, обеспечивающие сквозную технологию в рамках единой интегрированной системы автоматизированного проектирования [4]. Такой подход позволяет моделировать изделие на компьютере и выдавать в производство готовые оптимальные решения путем перебора большого числа вариантов на этапе проектирования и сокращать, таким образом, в несколько раз время выпуска готового изделия и создавать информационную основу для качественного принятия решений в управлении производством.

1.2 Перспективы внедрения и адаптации САПР на предприятии

При внедрении и адаптации САПР на предприятии существует общее описание задач, планируемых к решению с помощью САПР. В частности, на предприятии ООО “ЦентрГорода”, обозначены следующие основные задачи, решаемые автоматизацией отдела инноваций.

Цель деятельности отдела инноваций - конструкторская подготовка производства, разработка конструкторской и проектной документации (КПД) на изделия, выпускаемые ООО “ЦентрГорода ”.

Основные задачи, решаемые автоматизацией конструкторских работ:

1. Внедрение программного обеспечения, для поддержки работ отдела инноваций по конструированию и подготовке конструкторской документации изготовления изделий в машиностроительном производстве;

2. Создание отлаженной системы документооборота: сбора (создания), хранения и обработки конструкторской и технологической информации, упрощающей взаимодействие между отделом инноваций и смежными службами производства, снабжение и др.), организующей сквозное прохождение электронной информации через перечисленные службы, средствами Teamcenter;

3. Внедрение системы защиты, создаваемой КПД от несанкционированного доступа.

Приоритетами при автоматизации отдела инноваций считается:

1. Автоматизация процесса передачи конструкторской документации из отдела инноваций в смежные подразделения;

2. Автоматизация процесса конструирования изделий;

3. Автоматизация процесса расчета;

4. Автоматизация электронного документооборота и совместного доступа к базам данных конструкторско-технологических служб.

1.3 Основные требования к внедряемой программе на ООО “ЦЕНТРГОРОДА ”

1. Любые программные системы, разрабатываемые как внутри предприятия, так и сторонними разработчиками, должны поддерживать стандарты отрасли и работать в контексте единой информационной среды предприятия.

2. Функциональное соответствие программы вышеуказанным потребностям компании в поддержке автоматизации конструкторско-технологической деятельности.

3. “Дружественность” программы к пользователям. Возможность быстрого изучения ПО.

4. Поддержка программой положений ГОСТов, ОСТов и ЕСКД.

5. Возможность расширения и функционального наполнения программы.

6. Разумное соотношение “цена / функциональность / качество / надежность”.

7. Наличие сетевого варианта ПО.

8. Возможность выгрузки результатов проектирования в форматы, поддерживаемые на предприятии.

9. Для библиотек стандартных (типовых) объектов желательно наличие БД под специфику предприятия.

10. Наличие решений, позволяющих решить задачу стыковки программы с Teamcenter.

11. Наличие комплексных решений по автоматизации деятельности конструкторских и технологических служб.

12. Наличие решений для автоматизации работ по составлению сетевых графиков для производства, конструкторских и технологических бюро и т.д.

13. Известность компании – поставщика ПО, стаж функционирования компании на российском рынке.

14. Широкие возможности компании по сервисному обслуживанию ПО (включая информационную поддержку, наличие “горячей телефонной линии“ обучения персонала предприятия – клиента, разработку и внедрение обновлений конфигураций ПО на предприятиях не реже 2 раз в год, оказание поддержки в решении текущих проблем при работе с ПО).

15. Использование в качестве платформы распространенной мощной СУБД.

16. Желательно наличие русифицированной версии программы, если ПО импортное.

2 Задачи проектируемой библиотеки

2.1 Структура задач, решаемых при проектировании библиотеки

В ходе проектирования библиотеки элементов станочных приспособлений в первую очередь определяется круг задач, требующих решения:

1. Определение эскиза интерфейса библиотеки;
2. Определение функциональности программы;
3. Определение компонент интерфейса программы;
4. Определение эргономики программы;
5. Определение структуры информационного обеспечения;
6. Определение средств реализации информационного обмена данными;
7. Определение оптимального способа построения объекта проектирования;
8. Определение состава и структуры модулей программы;
9. Определение количества и состава функций NX API;
10. Определение способа загрузки и выгрузки библиотеки;

Данный порядок обусловлен тем, что для расчета каждого следующего этапа необходимы данные из предыдущих этапов.

Например, для определения функциональности программы сначала необходимо определить ее функциональность, а для составления алгоритма ее работы, определение оптимального способа создания объекта автоматизации в NX является задачей более первостепенной.

Рекомендуется придерживаться такого порядка создания, для того чтобы не возникало путаницы при алгоритмизации.

Для улучшения качества процесса проектирования вакуумных фильтров необходимо применение и внедрение информационных технологий, заключающихся, в частности, в разработке программных комплексов. Это позволит расширять, углублять, консервировать знания, сделать их более доступными и наглядными, за счет создания системы с возможностью визуализации исходной информации, хода и результатов работы, графической информации - 2D моделей, позволяющих наглядно воспринимать резуль-

таты проектирования, упрощается, ускоряется и снижается трудоемкость процесса создания и как следствие проектирования в целом.

Комплексной задачей является создание системы автоматизации проектирования всего комплекса стандартных элементов станочных приспособлений, которая включает в себя связь с интегрированной PLM средой Teamcenter.

2.2 Основные требования, предъявляемые к проектируемой библиотеке

В ходе разработки программной подсистемы, разработчиком предъявлялись следующие требования:

- программа должна производить построение требуемой конструкции вакуумного фильтра;
- гибкий интерфейс, который не требует перекомпиляции программы при изменении номенклатуры или параметров, содержащихся в базе данных, следовательно, интерфейс должен изменяться при внесении изменений в базу данных;
- хранить параметры проектируемых вакуумных фильтров в базе данных;
- максимальная простота интерфейса, интерфейс должен быть разработан в расчете на пользователя невысокой квалификации;
- возможность встраиваемости данной программной библиотеки в подсистемы и системы более высокого уровня;
- требования к аппаратным средствам и программному обеспечению, которое требуется для работы программы, должно определятся отраслевыми инструкциями;
- простота редактирования и добавления записей в базе данных;
- наличие оптимизации по минимизации машинного времени;
- высокая точность расчетов (для нецелочисленных значений использование диапазона “double”);
- автоматическое внесение изменений в интерфейс по ходу диалога с пользователем в зависимости от введенных значений;
- наличие справочной системы и руководства для пользователя;

- контроль ошибок ввода с выводом сообщения о типе ошибки и рекомендацией к устранению;

- невозможность ввода или расчета для параметров, которые противоречат друг другу или общим требованиям к программному средству;

- диапазон входных и выходных значений должен обеспечивать возможность проектирования всех вакуумных фильтров из базы данных;

- возможность сохранения полученных результатов средствами NX;

- возможность просмотра сохраненных результатов;

- отсутствие возможности внесения изменения в конструкцию библиотечного элемента;

- оптимизация интерфейса с точки зрения эргономики;

- простота в установке и настройке программного средства.

Таким образом, в ходе разработки программы были обозначены основные результирующие целевые позиции обеспечения функциональности, качества и сервисных возможностей, предоставляемых подсистемой [5].

2.2 Анализ количества и состава входных и выходных информационных массивов разрабатываемой программы

В ходе рассмотрения состава и структуры входной информации подсистемы, следует перечислить ее основные составляющие. Поскольку библиотека считывает параметры вакуумного фильтра из таблицы базы данных, то входными параметрами можно считать значения ее полей. Приведем таблицу параметров вакуумного фильтра и ее эскиз (рис. 1):

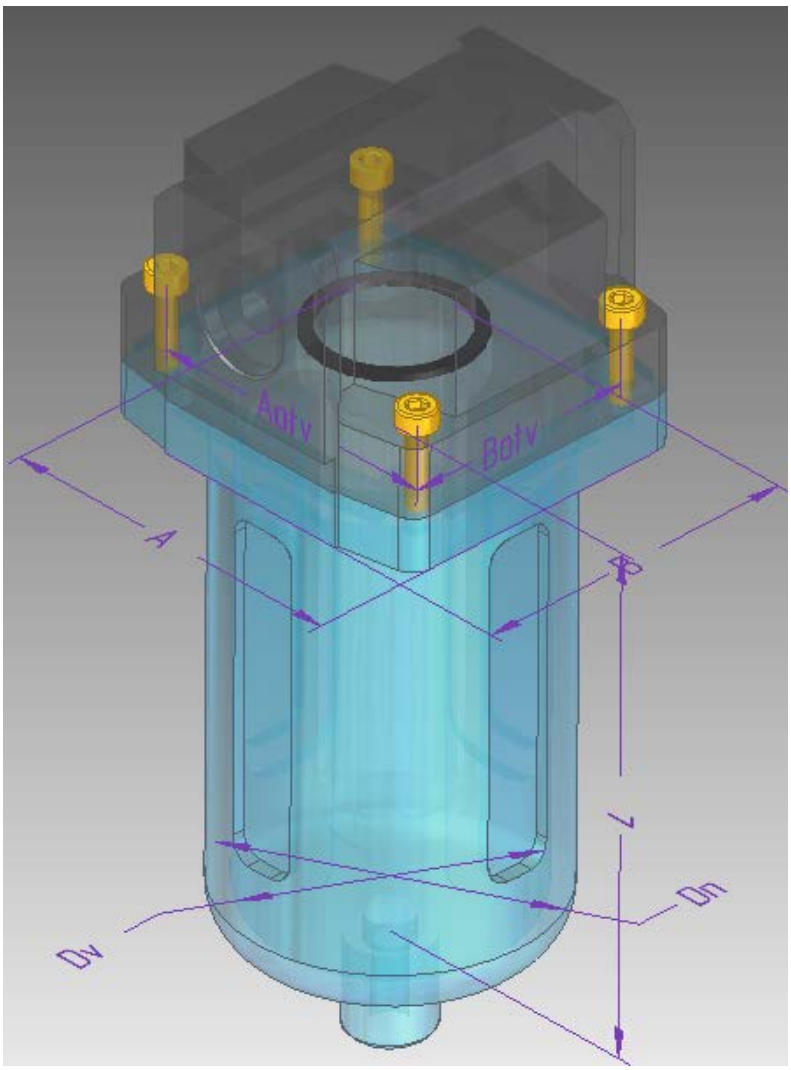

Рисунок 2.1 – Параметрический эскиз вакуумного фильтра

Таблица 2.1 – Таблица параметров вакуумного фильтра

№ п.п.	A	B	L	Aotv	Botv	Dv	Dn	V (л/мин)
1	2	3	4	5	6	7	8	9
2	53	50	100	44	35	38	44	11,8,
3	55	55	120	46	40	44	50	13,3
4	60	55	110	55	45	48	54	15,1
1	2	3	4	5	6	7	8	9
5	60	60	120	55	50	50	56	16,4
6	65	60	120	60	55	55	61	18,2

Данная совокупность входной информации определяет полный массив входной информации, который требуется для создания решающего алгоритма.

В качестве выходной информации будет выступать готовый спроектированный системой NX 7.5 библиотечный элемент, с отключенным при проектировании, режимом истории создания, приведенном на (рис. 2).

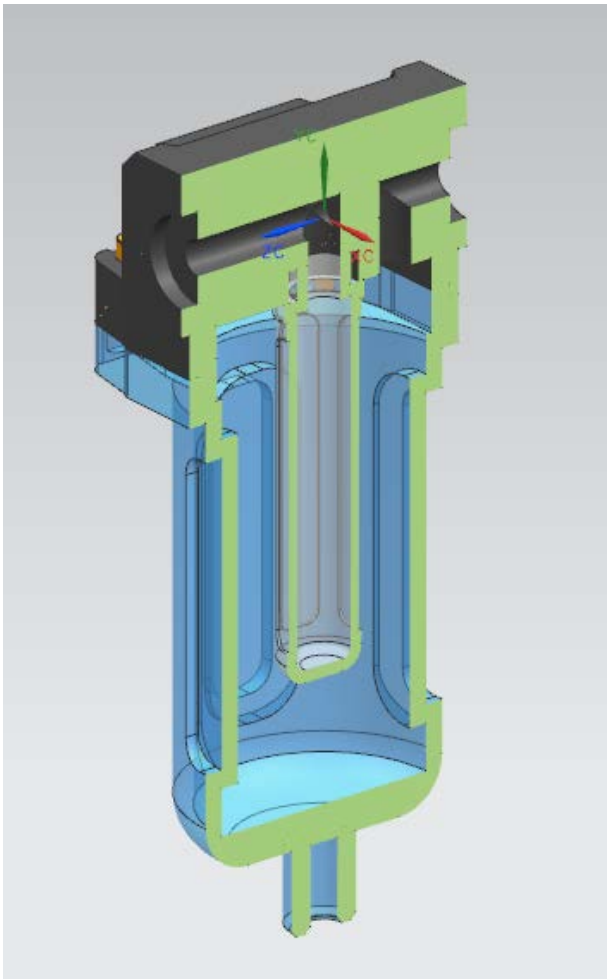

Рисунок 2.2 – Модель вакуумного фильтра

3 Проектирование программы

3.1 Функциональная модель системы

В ходе анализа требований к программе и определения круга задач, подлежащих автоматизации, определяется функциональная модель программы. В ее состав входят функции, связанные как с выполнением сервисных операций, так и операций по построению детали. Решено использовать средства доступа к ядру системы NX [6].

Библиотека построения вакуумных фильтров, имеет следующие функции:

- построение вакуумных фильтров как библиотечного элемента, с невозможностью внесения изменений в созданную конструкцию;
- работа со списком параметров конструкции вакуумного фильтра;
- поиск и выбор типоразмера вакуумного фильтра по любому из параметров;
- передача данных о геометрии приспособления в тело проектирующей процедуры;
- загрузка и выгрузка библиотеки в память системы и обратно;
- визуализация эскиза проектируемого изделия;
- запуск проектирующей процедуры;
- вызов последовательности выполнения NX API.

3.2 Проектирование алгоритма работы программы

3.2.1 Проектирование оптимальной последовательности построения

Прежде чем приступить к разработке программы, прежде всего, следует уяснить оптимальную последовательность выполнения проектных операций и процедур. Для этого следует проделать построение вакуумного фильтра в NX 7.5 в режиме ручного моделирования. Построения будем производить для первой записи таблицы параметров изделия (табл. 2.1).

Во-первых, после запуска NX 7.5 следует выбрать пункт меню – «Создание новой модели». После чего выбрать пункт создать эскиз и в качестве плоскости для постро-

ния, выбрать плоскость OXY. После чего построить эскиз для операции вытягивания со следующими размерами (рис. 3.1):

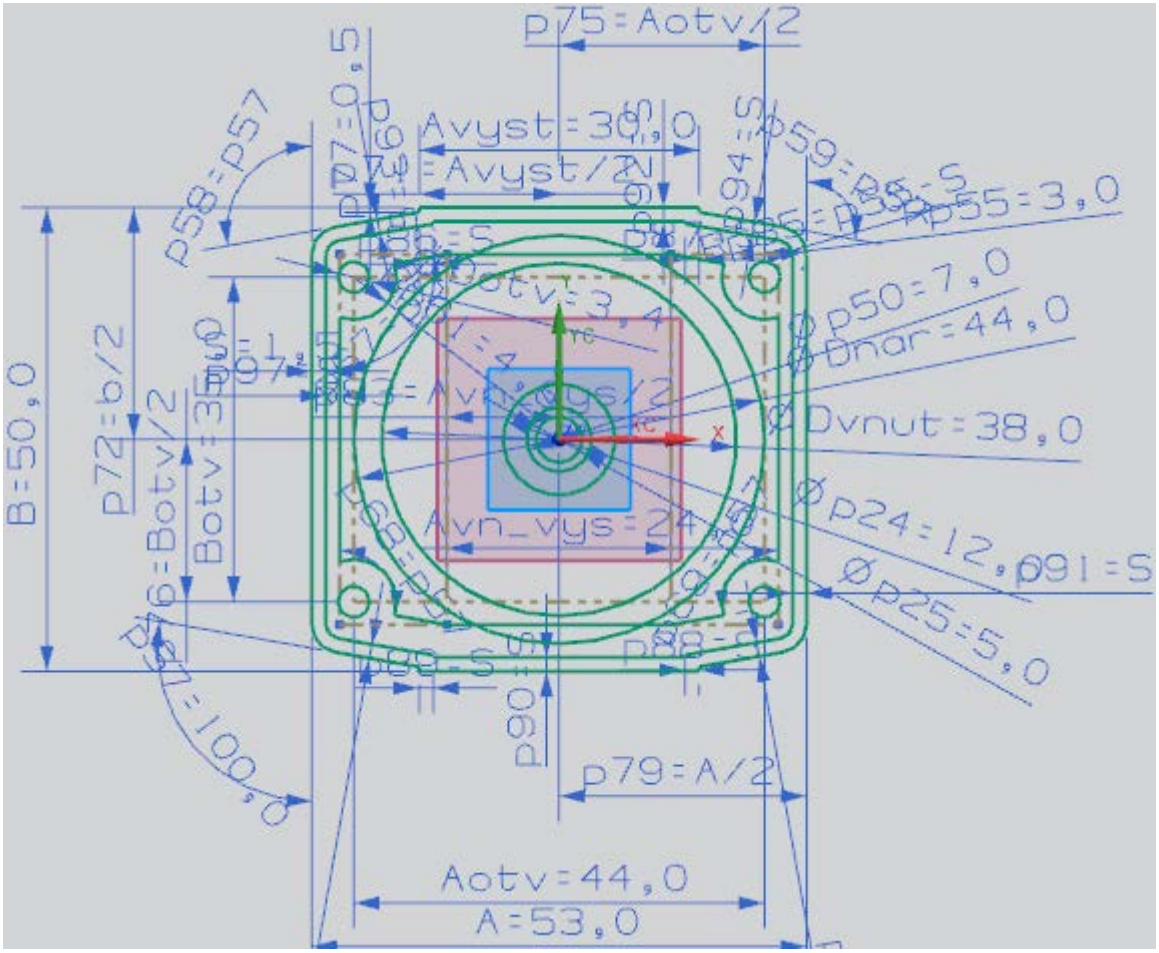

Рисунок 3.1 – Эскиз образующей корпуса вакуумного фильтра

Данные размеры соответствуют первой записи таблицы 1.

Далее в плоскости плоскость OXZ построим второй эскиз с размерими приведенными на рисунке 3.2.

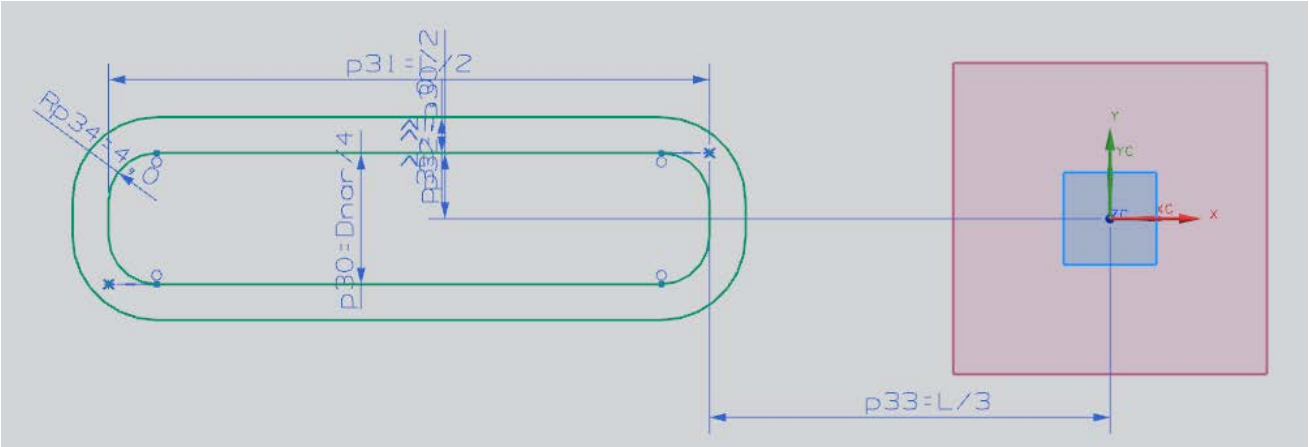

Рисунок 3.2 – Эскиз образующей 2 корпуса вакуумного фильтра

После построения эскизов следует вызвать операцию вытягивания. Работать будем не со всеми контурами эскиза сразу, а лишь с частью. Меню данных операций приведены на рисунках 3.3-3.10.

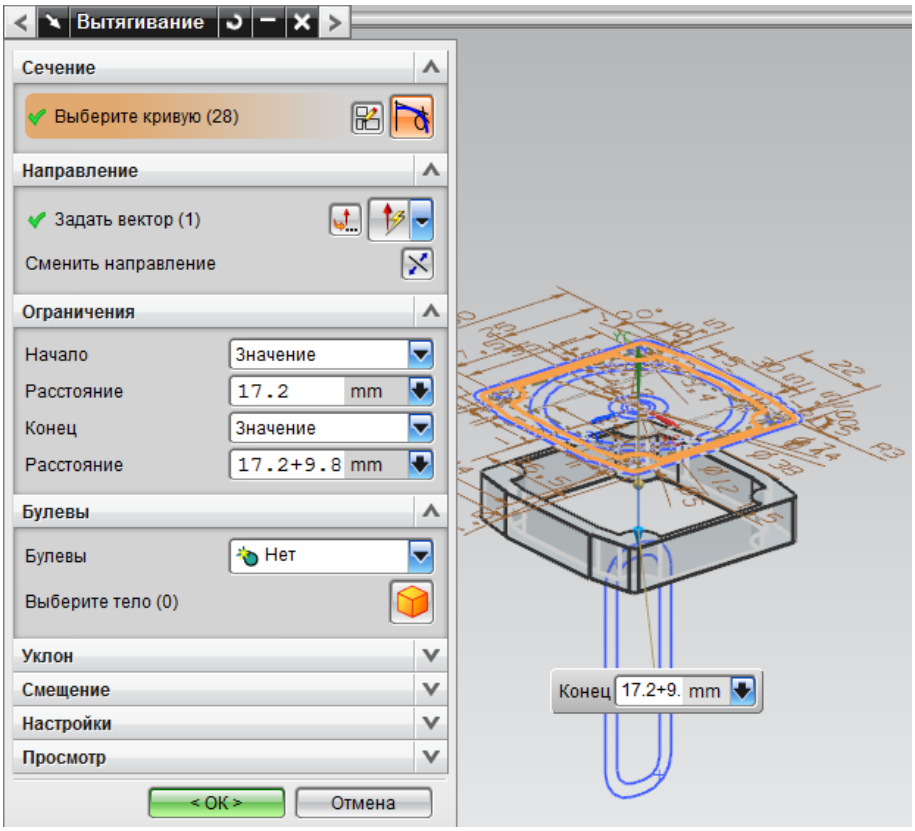

Рисунок 3.3 – Меню операции вытягивания 1

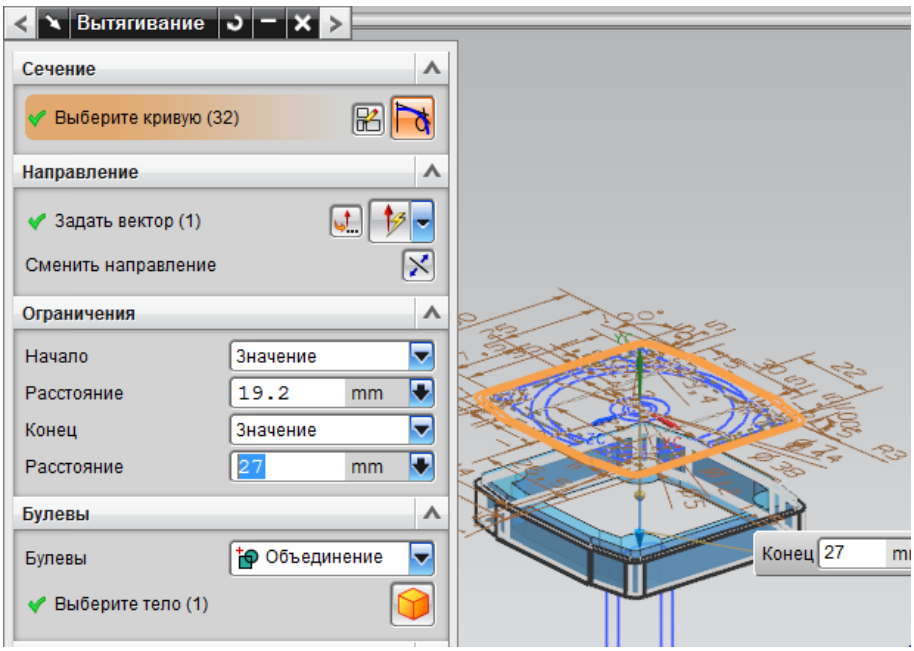

Рисунок 3.4 – Меню операции вытягивания 2

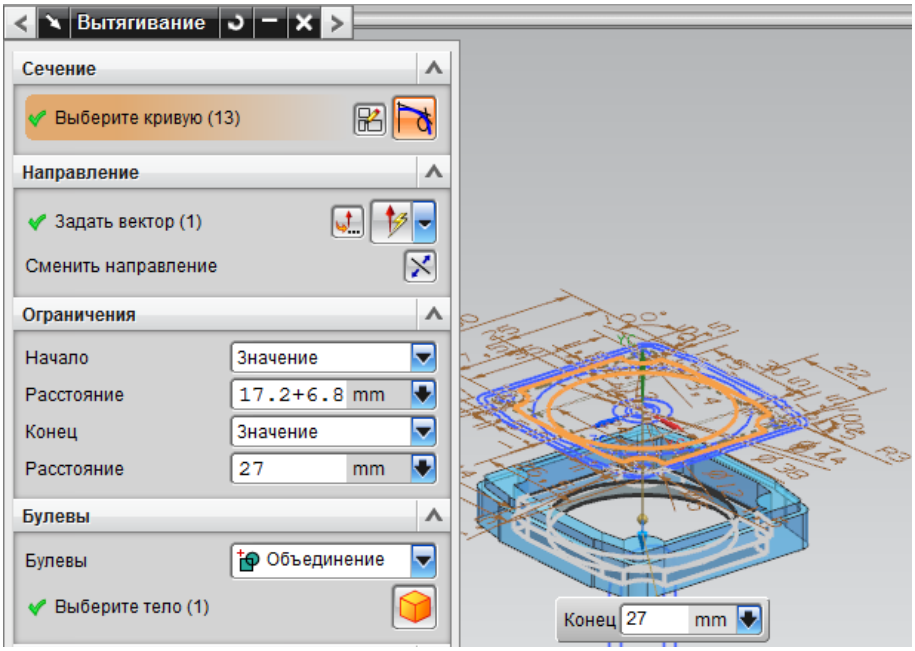

Рисунок 3.5 – Меню операции вытягивания 3

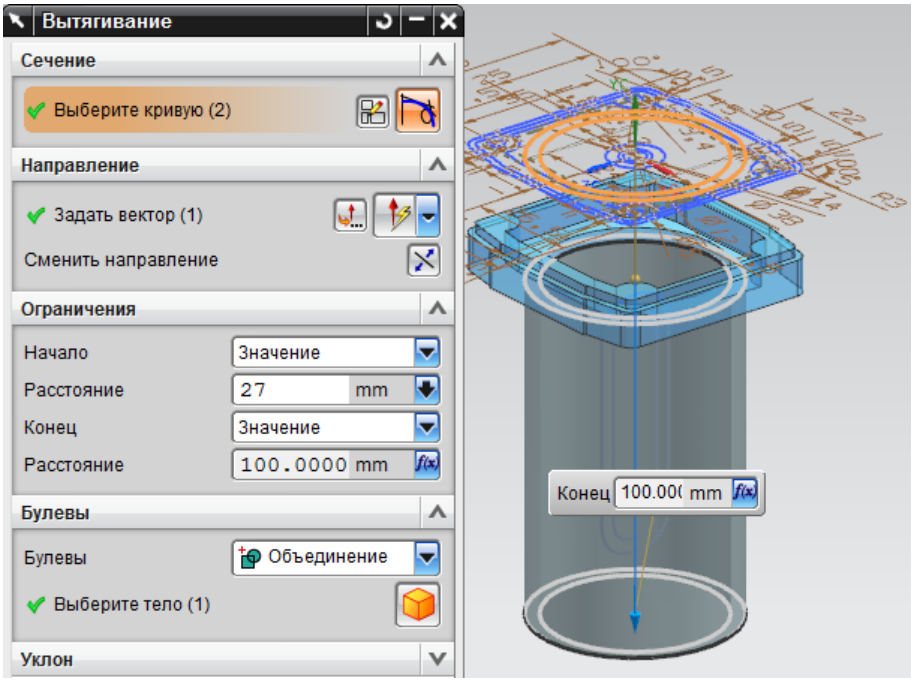

Рисунок 3.6 – Меню операции вытягивания 4

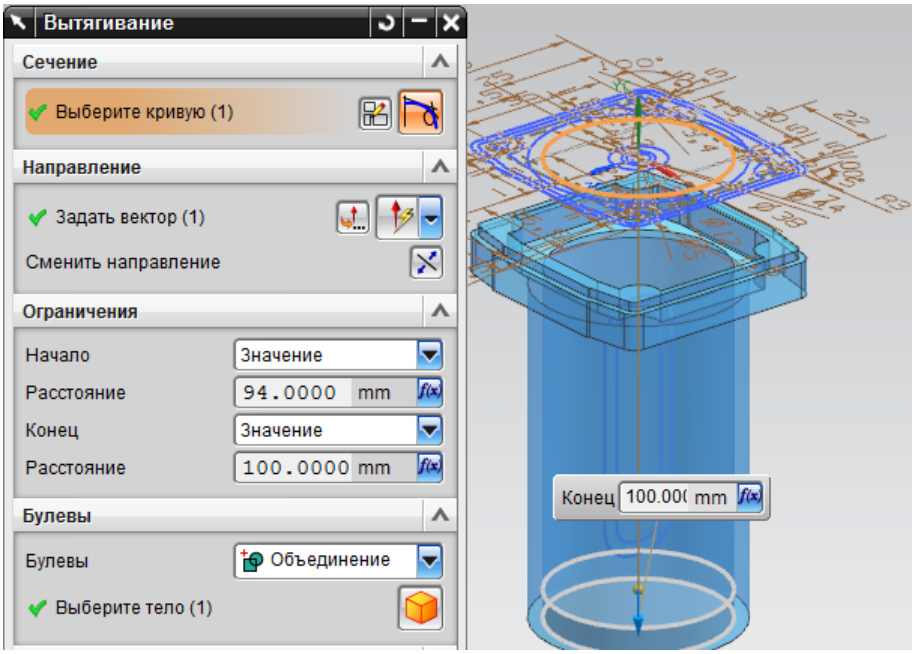

Рисунок 3.7 – Меню операции вытягивания 5

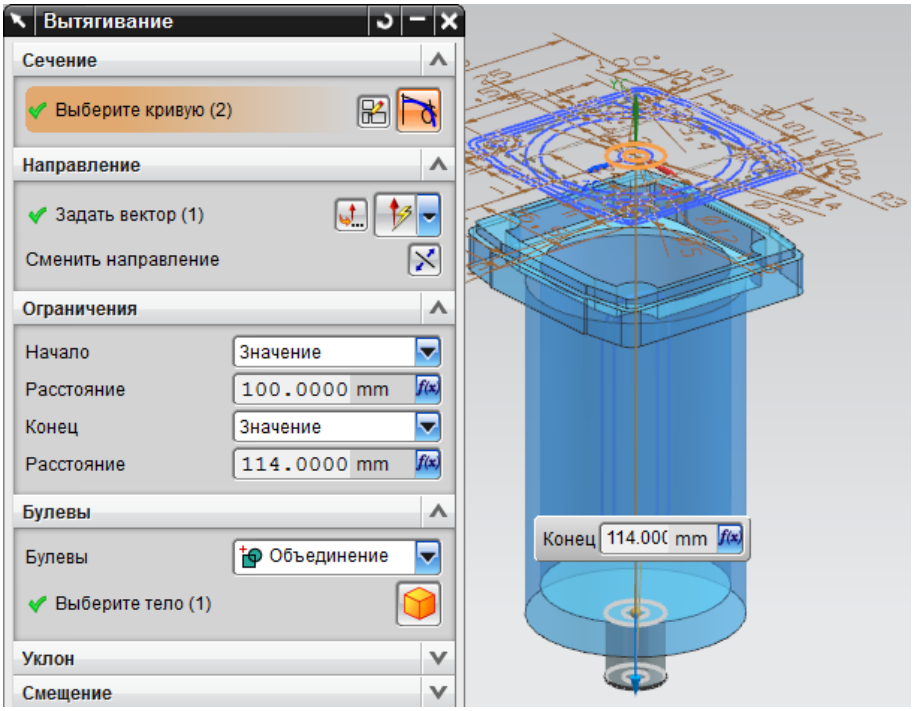

Рисунок 3.8 – Меню операции вытягивания 6

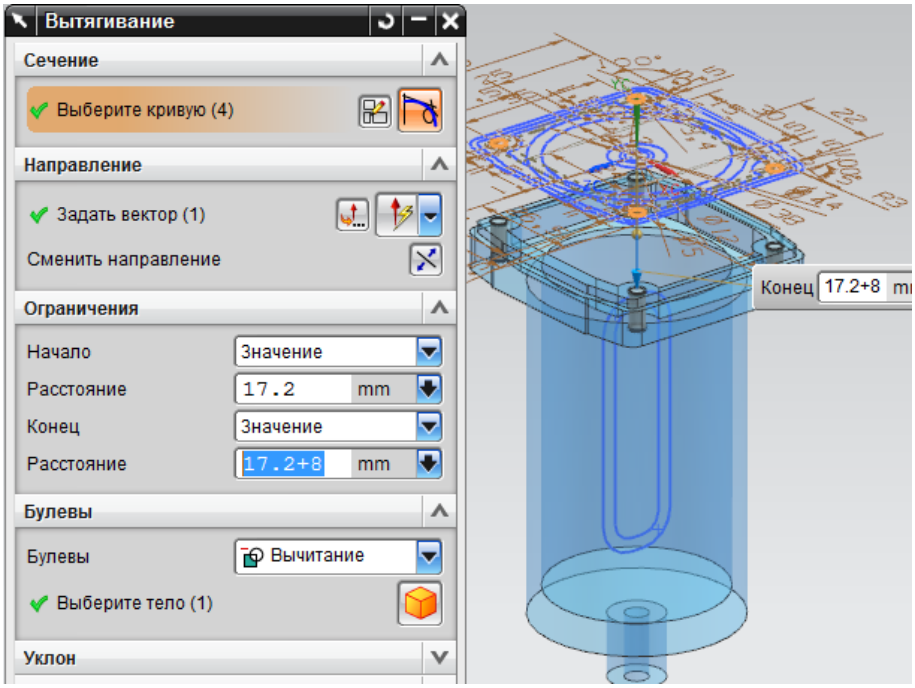

Рисунок 3.9 – Меню операции вытягивания 7

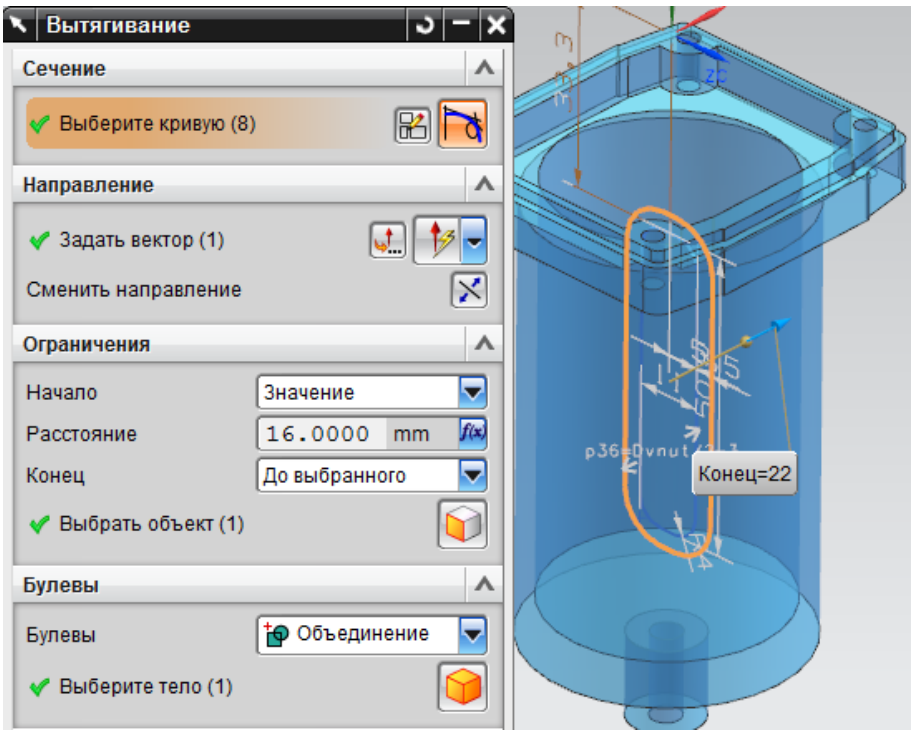

Рисунок 3.10 – Меню операции вытягивания 8

После получения результата для последней операции вытягивания создадим круговой массив полученных конструктивных элементов. Результат создания массива представлен на рисунке 3.11.

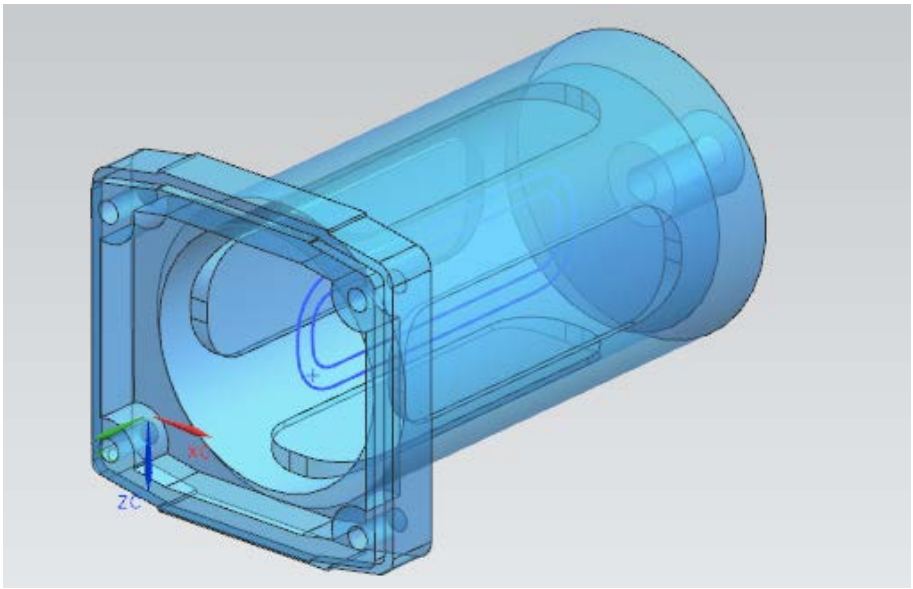

Рисунок 3.11 – Операция круговой массив (результат)

После того, как создался круговой массив элемента вытягивания, выполним вытягивание другого контура этого эскиза, с последующим преобразованием полученной особенности в круговой массив. Результатом должна стать полностью построенная деталь корпуса филтра, представленная на рисунке 3.12.

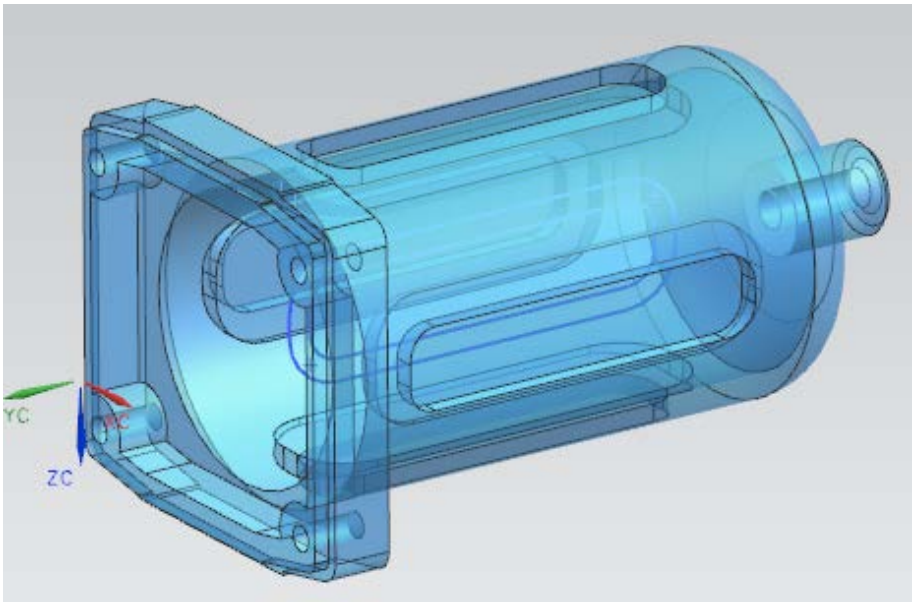

Рисунок 3.12 – Деталь «Корпус фильтра»

Выполним построение второй детали конструкции «Крышки». Для этого, как и при построении корпуса, следует выбрать пункт создать эскиз и в качестве плоскости для построения, выбрать плоскость OXY. После чего построить три эскиза для выполнения операций вытягивания с размерами соответствующими первой записи таблицы 2.1. Данные эскизы приведены на рисунках 3.13-3.15.

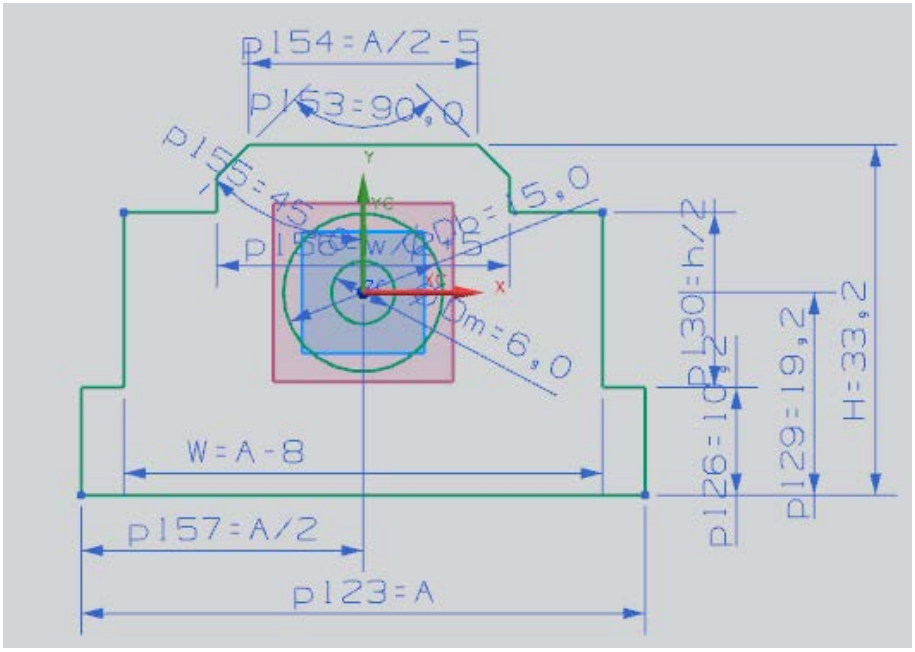

Рисунок 3.13 – Эскиз 1

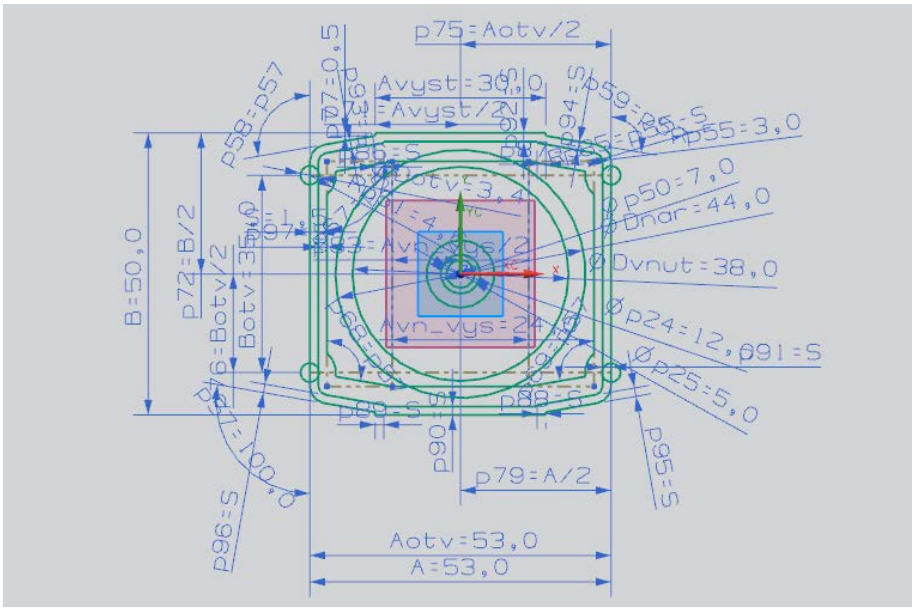

Рисунок 3.14 – Эскиз 2

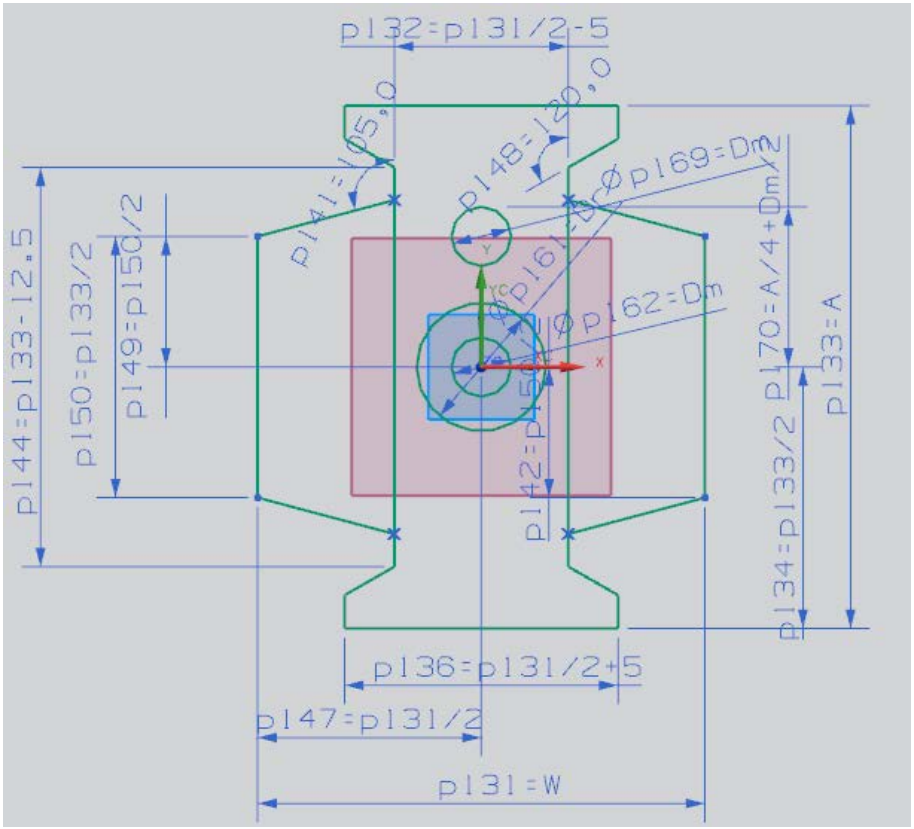

Рисунок 3.15 – Эскиз 3

После построения эскизов следует вызвать операцию вытягивания. Работать будем (как и в предидущей детали) не со всеми контурами эскиза сразу, а лишь с частью. Меню данных операций приведены на рисунках 3.16-3.29.

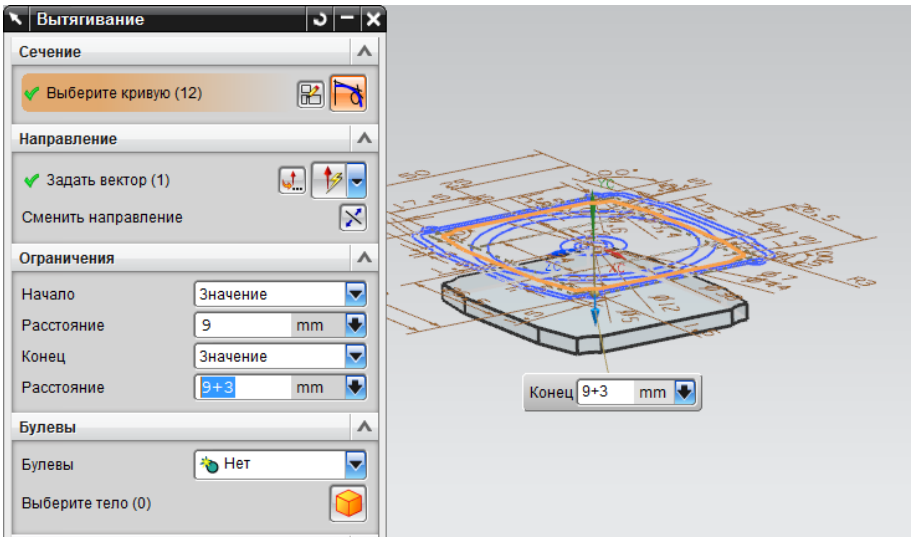

Рисунок 3.16 – Меню операции вытягивания 1

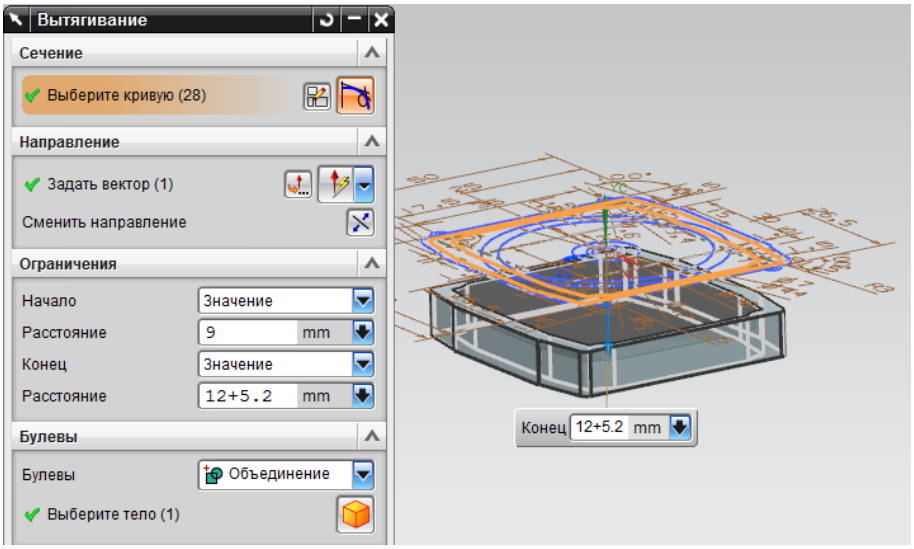

Рисунок 3.17 – Меню операции вытягивания 2

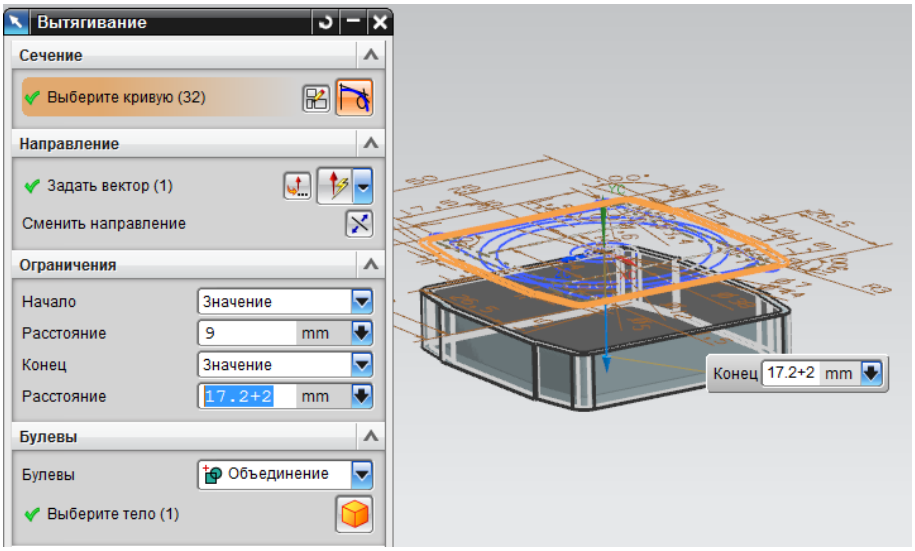

Рисунок 3.18 – Меню операции вытягивания 3

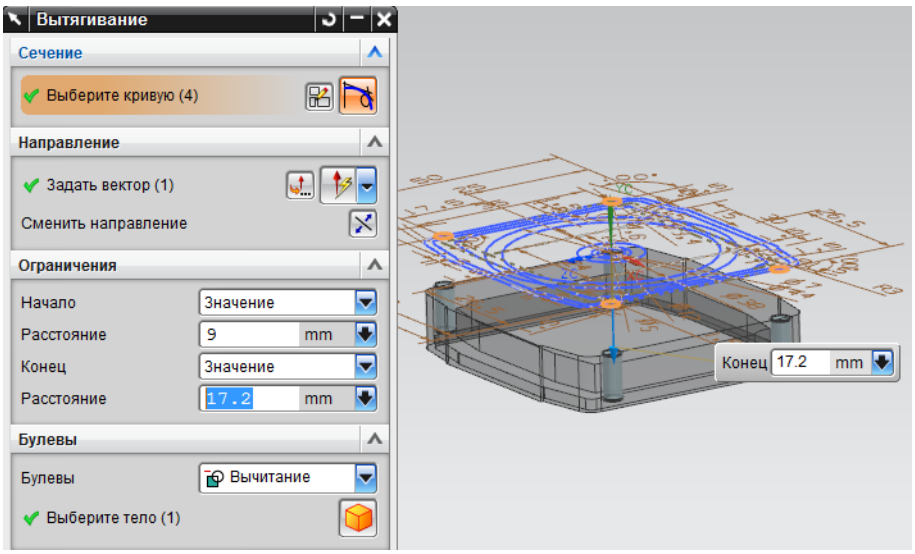

Рисунок 3.19 – Меню операции вытягивания 4

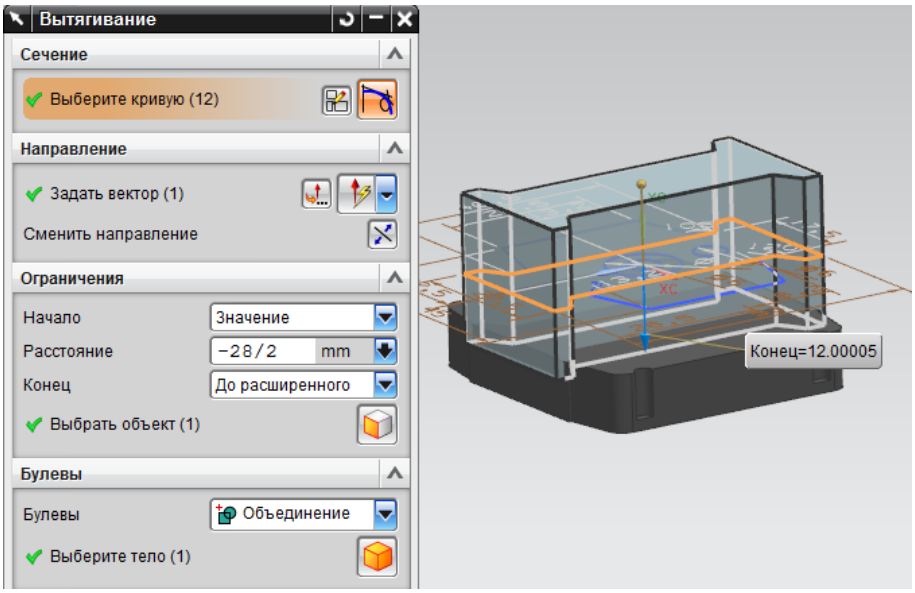

Рисунок 3.20 – Меню операции вытягивания 5

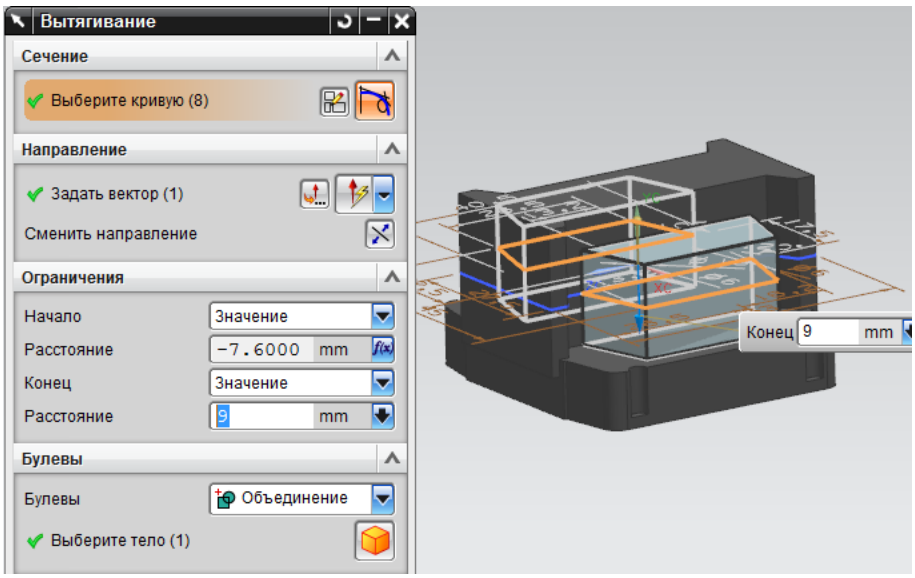

Рисунок 3.21 – Меню операции вытягивания 6

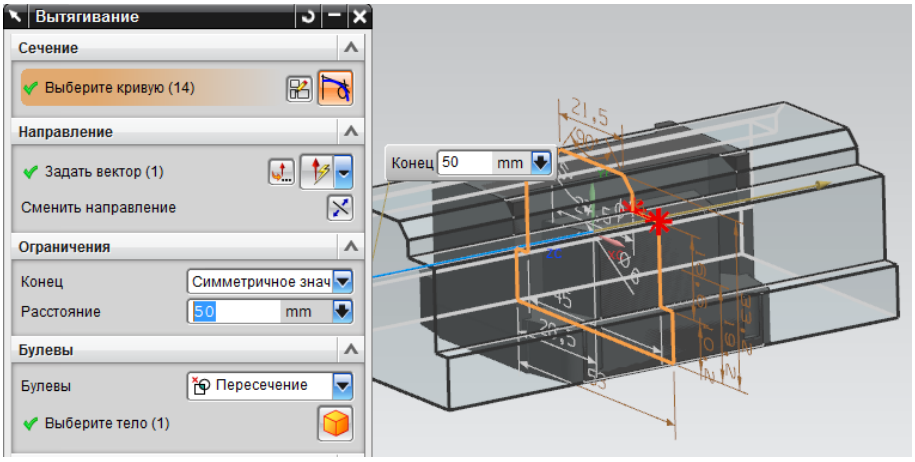

Рисунок 3.22 – Меню операции вытягивания 7

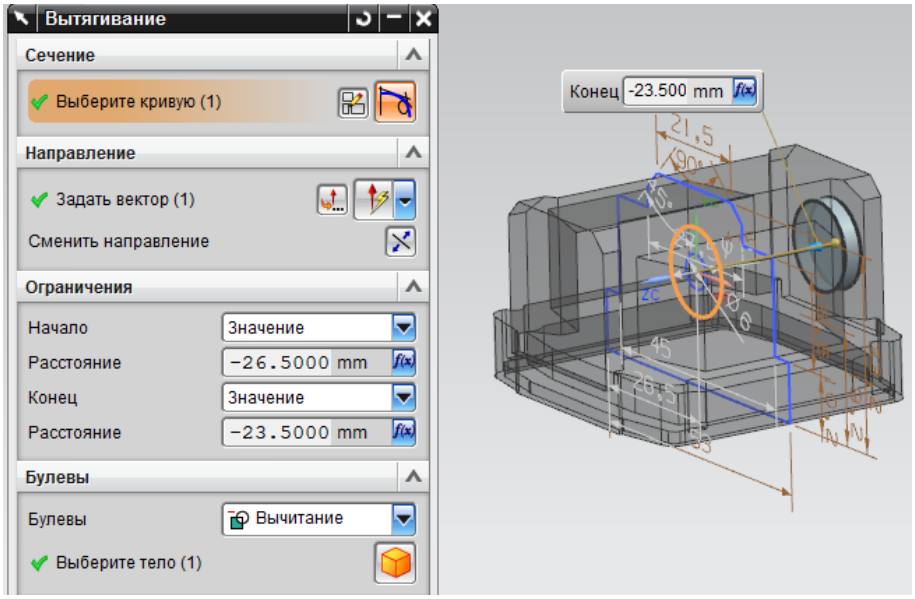

Рисунок 3.23 – Меню операции вытягивания 8

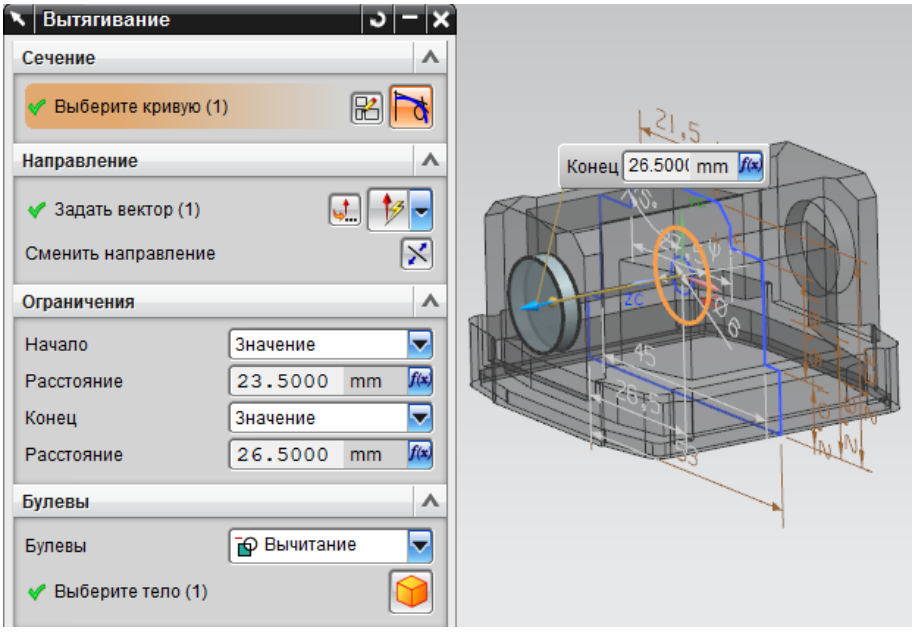

Рисунок 3.24 – Меню операции вытягивания 9

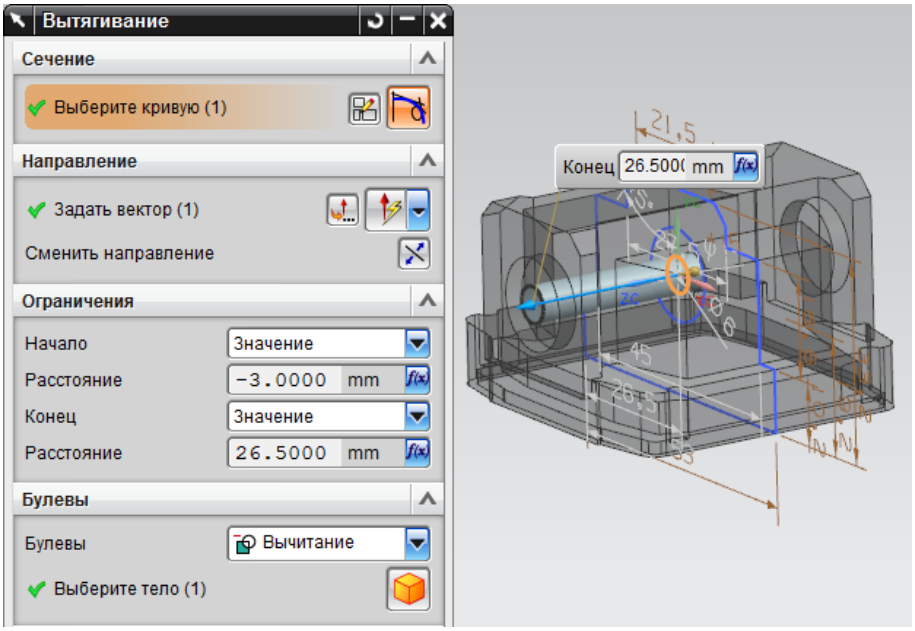

Рисунок 3.25 – Меню операции вытягивания 10

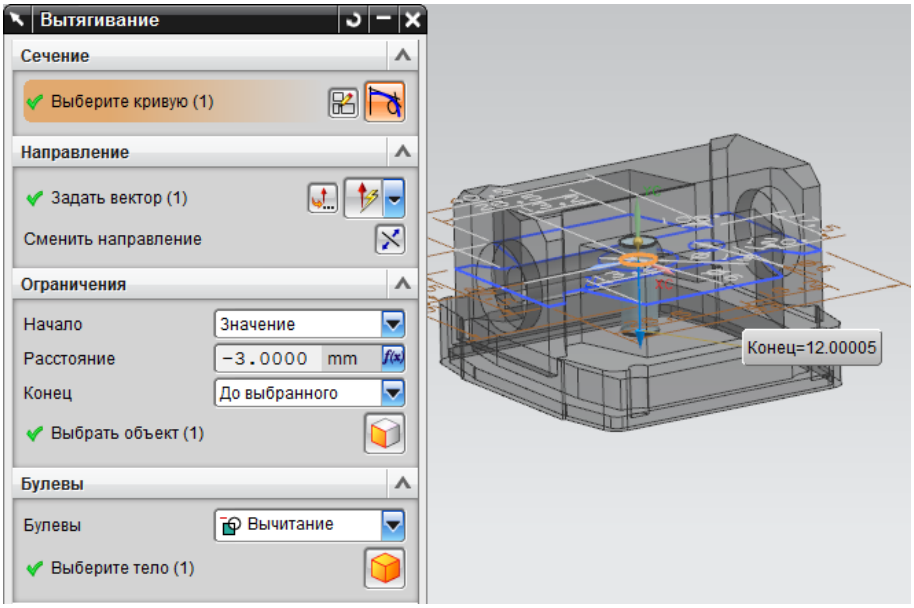

Рисунок 3.26 – Меню операции вытягивания 11

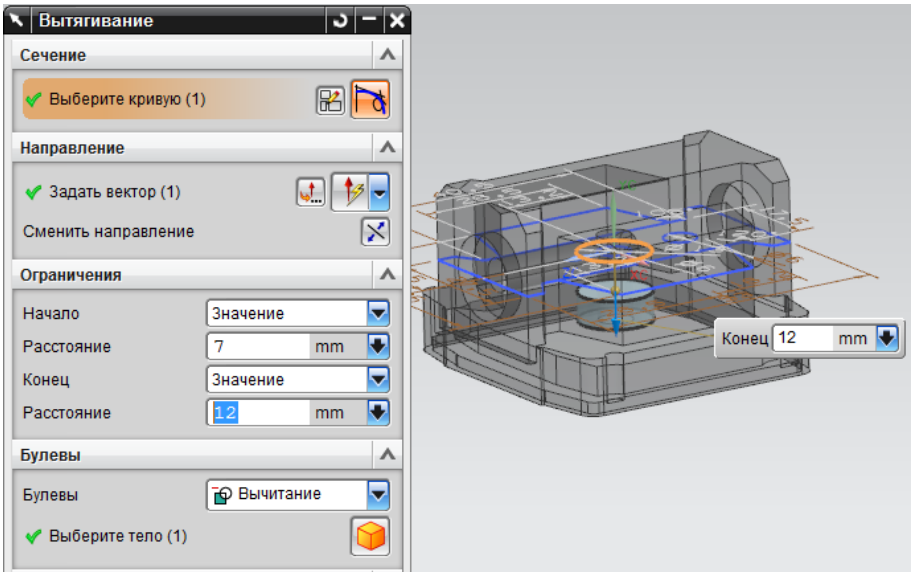

Рисунок 3.27 – Меню операции вытягивания 12

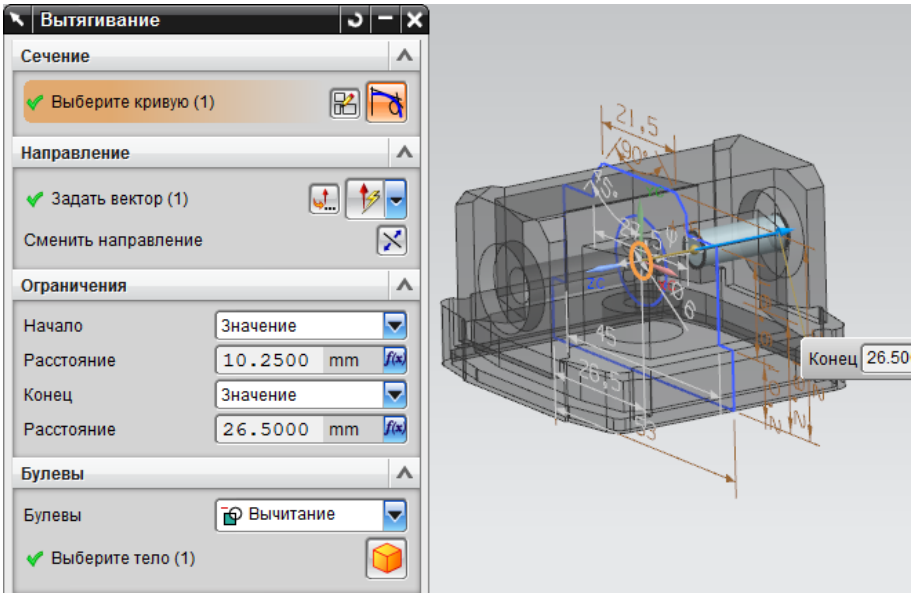

Рисунок 3.28 – Меню операции вытягивания 13

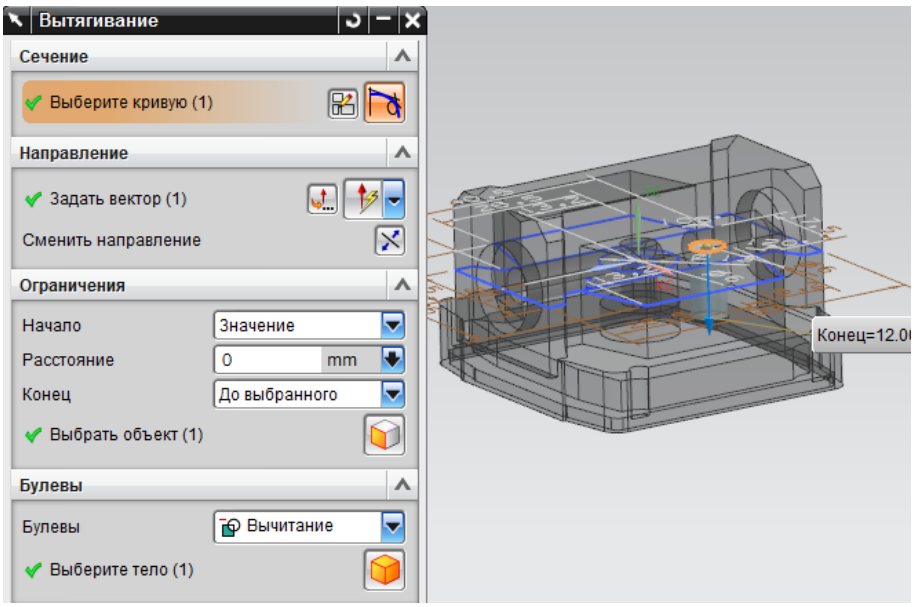

Рисунок 3.29 – Меню операции вытягивания 14

Аналогично приведенным деталям, строим остальные детали конструкции изделия. Описание их построения в данной работе не приводится. Однако, в результате должны плоучиться детали, приведенные на рисунках 3.30-3.31.

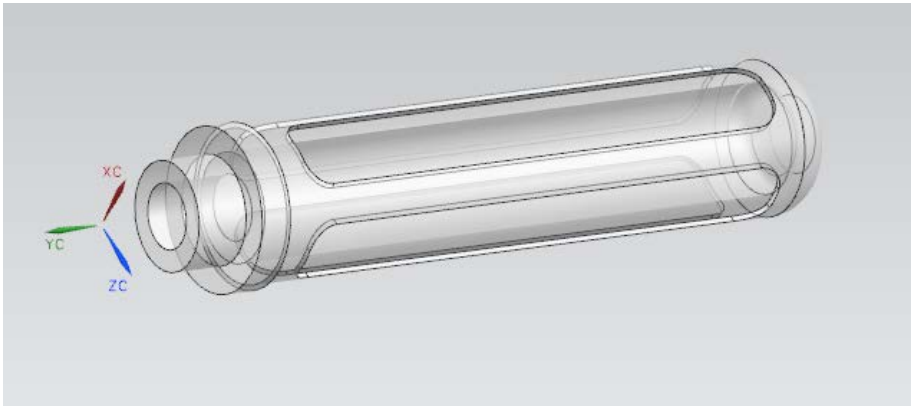

Рисунок 3.30 – Деталь «Элемент фильтрующий»

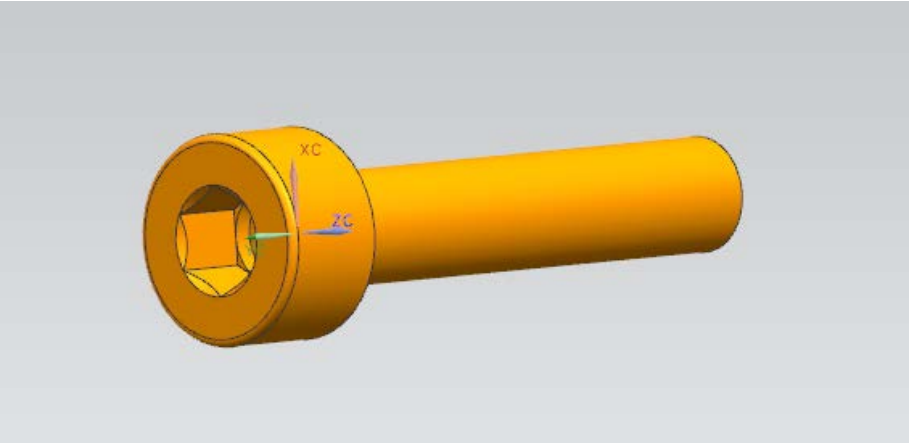

Рисунок 3.31 – Деталь «Винт»

В результате проделанных операций, мною получен оптимальный вариант построения вакуумного фильтра и определены основные операции и их последовательность особенностей, которые требуют параметризации.

3.2.2 Проектирование интерфейса программы

Прежде чем приступить к описанию построения интерфейса обозначим функциональную принадлежность каждого из его элементов. Во-первых, в интерфейсе должен быть представлен эскиз проектируемого изделия, для чего, целесообразнее всего из имеющегося инструментария MS Visual Studio 2013 использовать компонент PictureBox. Далее требуется организовать возможность перебора и поиска требуемой записи таблицы по каждому из исходных параметров таблицы. Как видно из таблицы 1 общее количество таких параметров равно 7 [6-7]. Поэтому, компонент, предоставляющий возможность перебора параметров должен иметь возможность связи с полями таблицы базы данных. Таким компонентом является наиболее подходящий элемент ComboBox. Расположим 7 таких элементов на форме. В качестве группирующего компонента следует использовать компонент GroupBox. В качестве компонента обрабатывающего события действий пользователя используем стандартные кнопки, которых на форме расположим 2. Одна из них отвечает за построение вакуумного фильтра, вторая за вы-

ход из библиотеки. Третья кнопка является системной оконной кнопкой, обозначенной как «?». Этот элемент отвечает за запуск диалогового окна «О программе».

Таким образом, интерфейс главного окна библиотеки в режиме конструктора, будет выглядеть, как показано на рисунке 3.32.

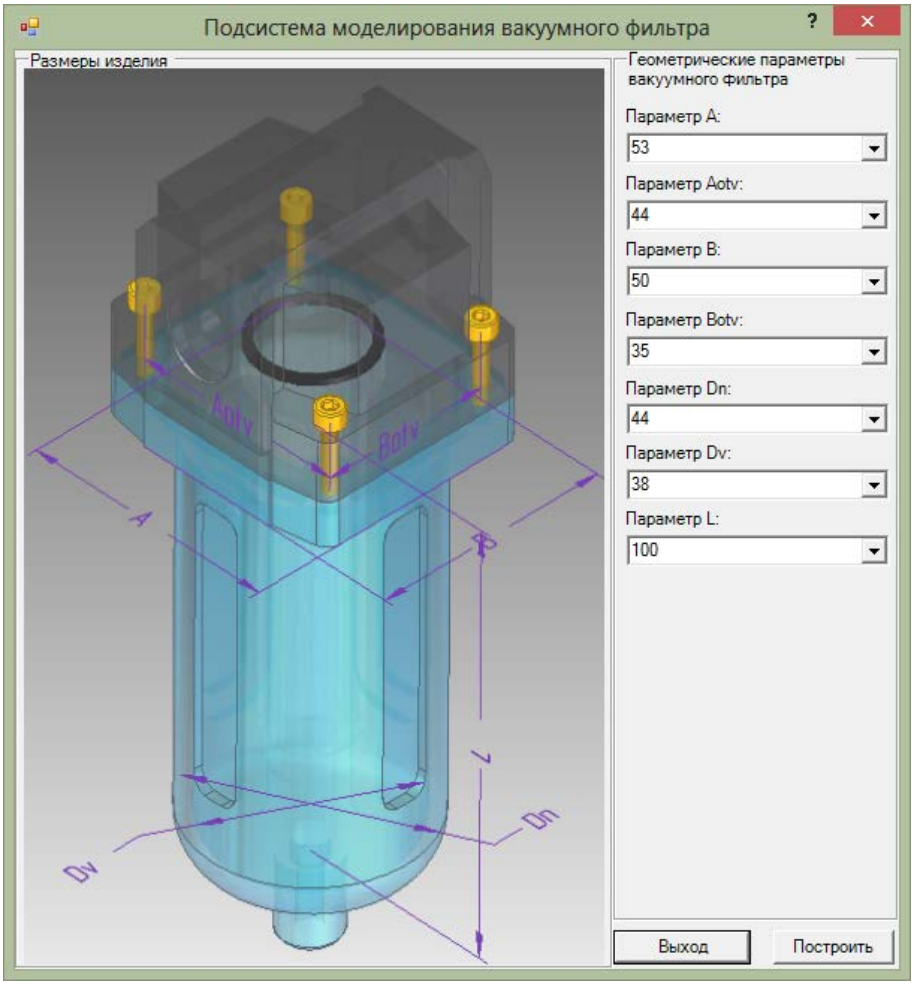

Рисунок 3.32 – Главное окно подсистемы

Кроме того, при построении интерфейса были использованы также не визуальные компоненты, которые осуществляют связь интерфейса с таблицей базы данных. Назначение данных компонент будет описано в следующем разделе.

3.2.3 Проектирование связи интерфейса библиотеки с базой данных

Прежде чем говорить о связи базы данных с интерфейсом, следует остановится на самом способе создания таблицы базы данных, описании полей данной таблицы и вообще о методе, способе и средствах ее создания.

Рассмотрим технологию доступа к базе данных заимствованную из MSDN. Уровни доступа к данным представлены в объектной модели ADO.NET, представленной на рисунке 3.33.

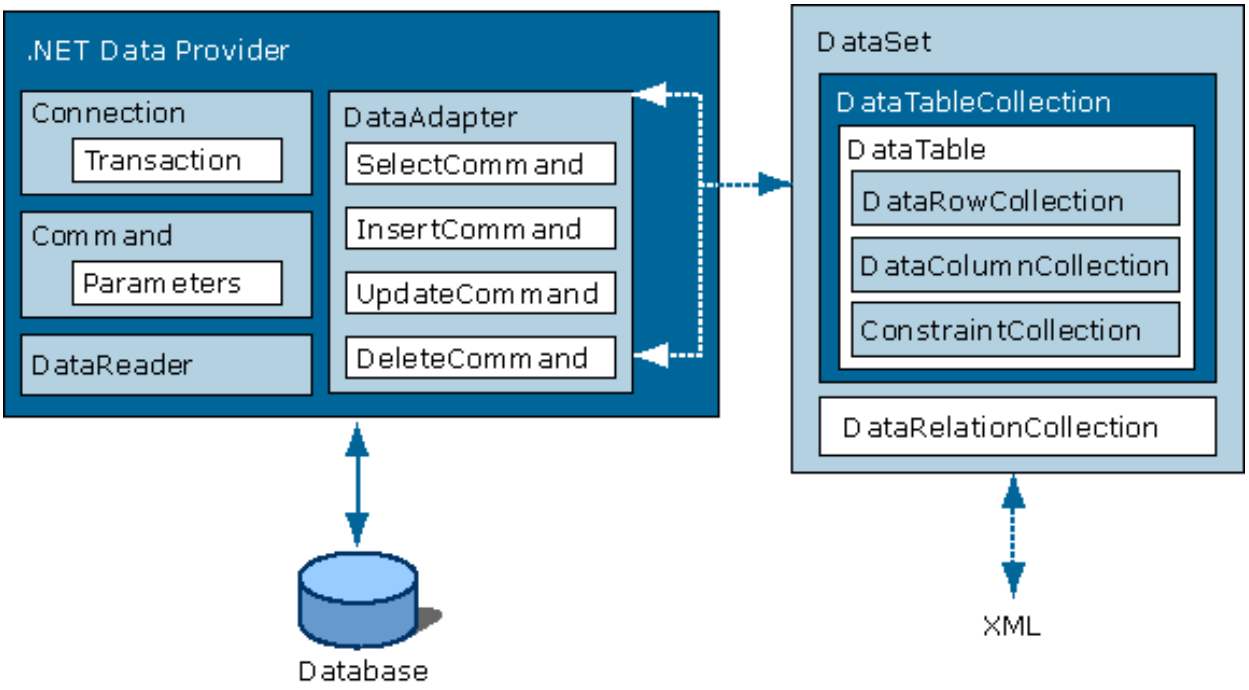

Рисунок 3.33 – Объектная модель ADO.NET

Как видно из данной диаграммы, основу ADO.NET составляют два основных компонента (разумеется, в данном случае речь не идет о компонентах COM): DataSet и Провайдер Данных [8].

Провайдер Данных, как это ясно следует из его названия, отвечает за связь приложения с источником данных и манипуляции данными. Основные объекты, входящие в его состав, - это Connection, Command, DataAdapter и DataReader.

Connection используется для установления соединения с источником данных, а также для управления транзакциями.

Command позволяет манипулировать данными источника, а также выполнять хранимые процедуры. При этом могут использоваться параметры для передачи данных в обоих направлениях.

DataAdapter служит связующим звеном между DataSet и источником данных. Он использует Command для выполнения команд SQL как для заполнения DataSet данными, так и для обратной передачи измененных клиентом данных к источнику. Для выполнения этих функций объект имеет 4 метода: SelectCommand, InsertCommand, UpdateCommand и Deletecommand [9, 10].

DataReader представляет однонаправленный поток данных от источника только на чтение. Если приложение клиента не модифицирует данные и не требуется произвольная выборка данных, а достаточно их однократного просмотра, то использование DataReader вместо DataSet позволит сохранить ресурсы RAM и CPU, а также поднять быстродействие приложения.

DataSet можно представить себе, как небольшую реляционную базу данных, резидентную в оперативной памяти клиента. В DataSet загружаются данные, после чего соединение с их источником (или источниками, которых в принципе может быть несколько, поскольку в одном экземпляре DataSet могут объединяться данные из нескольких источников) может быть разорвано. Далее клиент в автономном режиме производит обработку данных, при необходимости модифицирует их, после чего снова устанавливается соединение с источником и модифицированные данные передаются обратно. Такая схема работы может оказаться предпочтительной для корпоративных приложений, работающих в глобальных сетях, где поддержание постоянного соединения может повлечь дополнительные накладные расходы [11].

DataSet может сохранять свое текущее содержимое в файлах XML, а схему в виде XML Schema definition language (XSD).

Внутреннее устройство DataSet представлено на рисунке 3.34:

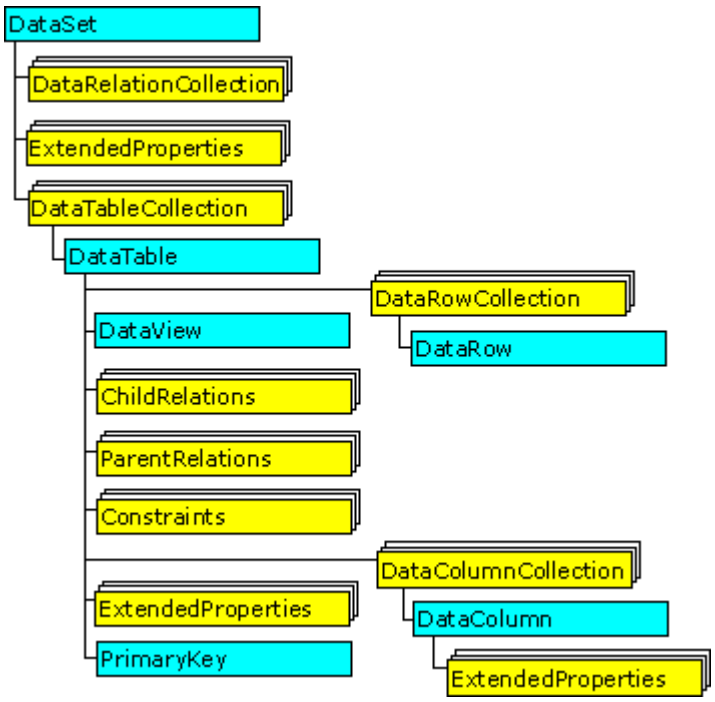

Рисунок 3.34 – Устройство DataSet

Рассмотрим вкратце наиболее важные компоненты DataSet, не углубляясь во второстепенные детали.

DataSet включает две основные коллекции: DataTableCollection и DataRelationCollection.

DataTableCollection это коллекция объектов DataTable, каждый из которых представляет одну таблицу отношения и в свою очередь состоит из коллекций:

DataColumnCollection представляет все столбцы таблицы. Помимо данных, столбец может включать формулу для динамического расчета своего содержимого.

DataRowCollection представляет набор строк таблицы (возможно, пустой). Следует отметить, что хранятся не только текущие значения данных, но и первоначальные, что позволяет выделить измененные данные [13].

ConstraintCollection набор ограничений целостности данных (например, можно задать, допустимо ли для данного столбца значение NULL или должны ли данные быть уникальными).

Другая коллекция из состава DataSet это DataRelationCollection, которая служит для навигации по связанному набору таблиц и содержит ограничения ссылочной целостности DataSet. В ней, например, можно задать ограничения, которое не позволит удалить запись главной таблицы, если на нее есть ссылки в подчиненной таблице посредством внешнего ключа, тем самым, не допуская появления осиротевших (orphaned) записей [14].

Как видно из диаграммы объектной модели ADO.NET, помимо работы с Провайдерами Данных, DataSet может работать и с данными формата XML, причем данные эти могут как храниться в файлах, так и транспортироваться по сетям передачи данных посредством транспорта HTTP. В последнем случае существенно облегчается построение распределенных приложений корпоративного уровня, поскольку HTTP без труда может проходить через брандмауэры, не создавая особой угрозы для безопасности внутренних сетей.

Поскольку информация о параметрах вакуумных фильтров имеет четко определенную структуру, и не выходит за грани избыточности, то целесообразнее представлять ее в виде одной таблицы.

В результате проделанных действий, можно графически представить будущий укрупненный алгоритм работы подсистемы.

4 Реализация подсистемы

4.1 Реализация интерфейса программы

Интерфейс – важная часть любой программы. Современный интерфейс должен отвечать следующим требованиям:

1. Типичность (Дизайн интерфейса должен быть ориентирован на ведущих производителей программных продуктов, например, компанию Microsoft. Именно их интерфейсы, знакомы большинству пользователей по программам Word, Excel и т.д.);

2. Дружественность (интерфейс представляет собой систему подсказок, проводников и. т. д.);

3. Полная функциональность (доступность пользователю всех функций программы простейшими манипуляциями мыши);

4. Оперативность (при переключении режимов работы программы не должно происходить задержек, связанных с медлительностью интерфейса)

В соответствии с выше перечисленными требованиями, разрабатываем интерфейс программы, опираясь на заложенные ранее функции системы и структуру интерфейса, приступаем к его реализации [15]. Визуальная среда MS Visual Studio 2013 обладает всем необходимым набором средств для создания удобного и современного интерфейса. Компоновка и состав используемых компонент уже были описаны в разделе проектирования, поэтому приведем значения свойств всех использованных компонентов.

```
this.btnDraw.Name = "btnDraw";
this.btnDraw.Text = "Построить";
// btnExit //
this.btnExit.Name = "btnExit";
this.btnExit.Text = "Закрыть";
// bp1 //
this.bp1.Image = global::Filtr.Properties.Resources.Filtr;
this.bp1.Name = "bp1";
this.bp1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
```

```
// gbId    //
this.gbId.Controls.Add(this.cbId);
this.gbId.Name = "gbId";
this.gbId.Text = "Идентификатор вакуумного фильтра";
// cbId  //
this.cbId.DataSource = this.dsID1;
this.cbId.DisplayMember = "53942-0XX.Id";
this.cbId.Name = "cbId";
// dsID1  //
this.dsID1.DataSetName = "dsID";
    this.dsID1.SchemaSerializationMode = Sys-
    tem.Data.SchemaSerializationMode.IncludeSchema;
// gbDv  //
this.gbDv.Controls.Add(this.cbD);
this.gbDv.Name = "gbDv";
this.gbDv.Text = "Параметр Dv";
// cbD //
this.cbD.DataSource = this.dsID1;
this.cbD.DisplayMember = "53942-0XX.Dv";
this.cbD.Name = "cbDv";
// gbDn //
this.gbDn.Controls.Add(this.cbDn);
this.gbDn.Name = "gbDn";
this.gbDn.Text = "Параметр Dn";
// cbDn //
this.cbDn.DataSource = this.dsID1;
this.cbDn.DisplayMember = "53942-0XX.Dn";
this.cbDn.Name = "cbDn";
// gbA //
```

```
this.gbA.Controls.Add(this.cbA);
this.gbA.Name = "gbA";
this.gbA.Text = "Параметр A";
// cbA//
this.cbA.DataSource = this.dsID1;
this.cbA.DisplayMember = "53942-0XX.A";
this.cbA.Name = "cbA";
// btnAbout //
this.btnAbout.Name = "btnAbout";
this.btnAbout.Text = "?";
// cbMassa //
this.cbMassa.DataSource = this.dsID1;
this.cbMassa.DisplayMember = "53942-0XX.Massa";
this.cbMassa.Name = "cbMassa";
// gbH  //
this.gbH.Controls.Add(this.cbH);
this.gbH.Name = "gbH";
this.gbH.Text = "Параметр H";
// cbH  //
this.cbH.DataSource = this.dsID1;
this.cbH.DisplayMember = "53942-0XX.H";
this.cbH.Name = "cbH";
// gbL //
this.gbL.Controls.Add(this.cbL);
this.gbL.Name = "gbL";
this.gbL.Text = "Параметр H1";
// cbL   //
this.cbL.DataSource = this.dsID;
this.cbL.DisplayMember = "53942-0XX.L";
```

```
            this.cbL.Name = "cbL";
            // gbB  //
            this.gbB.Controls.Add(this.cbB);
            this.gbB.Name = "gbB";
            this.gbB.Text = "Параметр B";
            //  cbB          //
            this.cbB.DataSource = this.dsID1;
            this.cbB.DisplayMember = "53942-0XX.B";
            this.cbB.Name = "cbB";
            // gbAotv  //
            this.gbAotv.Controls.Add(this.cbAotv);
            this.gbAotv.Location = new System.Drawing.Point(471, 210);
            this.gbAotv.Name = "gbAotv";
            this.gbAotv.Text = "Тип шпинделя по ОСТ 428 ";
            // cbAotv  //
            this.cbAotv.DataSource = this.dsID1;
            this.cbAotv.DisplayMember = "53942-0XX.Aotv_type";
            this.cbAotv.Name = "cbAotv";
            // dbConnect  //
            this.dbConnect.ConnectionString     =     "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=|DataDirectory|\\Filtr.accdb";
            //  oleDbSelectCommand1     //
            this.oleDbSelectCommand1.CommandText    =    "SELECT                    [53942-
0XX].*\r\nFROM          [53942-0XX]";
            this.oleDbSelectCommand1.Connection = this.dbConnect;
            // dbAdapter  //
            this.dbAdapter.SelectCommand = this.oleDbSelectCommand1;
            this.dbAdapter.TableMappings.AddRange(new                              Sys-
tem.Data.Common.DataTableMapping[] {
```

```
            new System.Data.Common.DataTableMapping("Table", "53942-0XX", new Sys-
tem.Data.Common.DataColumnMapping[] {
        new System.Data.Common.DataColumnMapping("Id", "Id"),
        new System.Data.Common.DataColumnMapping("Dv", "Dv"),
        new System.Data.Common.DataColumnMapping("Dn", "Dn"),
        new System.Data.Common.DataColumnMapping("L", "L"),
        new System.Data.Common.DataColumnMapping("A", "A"),
        new System.Data.Common.DataColumnMapping("B", "B"),
        new System.Data.Common.DataColumnMapping("Aotv", "Aotv"),
        new System.Data.Common.DataColumnMapping("Botv ", "Botv "),
         // MaimForm  //
         this.Controls.Add(this.gbAotv);
         this.Controls.Add(this.gbDv);
         this.Controls.Add(this.gbL);
         this.Controls.Add(this.gbDn);
         this.Controls.Add(this.gbA);
         this.Controls.Add(this.gbB);
         this.Controls.Add(this.gbAotv);
         this.Controls.Add(this.gbBotv);
         this.Controls.Add(this.gbId);
         this.Name = "MaimForm";
         this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
         this.Text = "Подсистема проектирования вакуумных фильтров";
         this.Load += new System.EventHandler(this.Form1_Load);
```

4.2 Реализация связи приложения с таблицами баз данных

Для таблицы нам понадобятся поставщик OleDbDataAdapter и набор данных DataSet. Этот элемент управления ComboBox свяжем с наборами данных через их свойство DataSource. Для ComboBox нужно дополнительно еще связать свойства DisplayMember и ValueMember с соответствующими полями (ID и D).

При выборе пользователем элемента списка значение ID будет передаваться как параметр в команду поставщика данных, связанного с таблицей, который загрузит из БД в набор нужные записи для отображения их в других ComboBox. Если пользователь редактировал данные в ComboBox ComboBox, то при смене записи в другом ComboBox эти данные нужно сохранить в БД [16]. Для этого нужно создать соответствующую SQL-команду для поставщика.

Добавьте к решению командой Add/New Project новый проект типа Windows Forms Application с именем MainForm.

Поместите в рабочую область оболочки компонент OleDbDataAdapter, соедините его с файлом Filtr.accdb БД из прилагаемого каталога Source (на предложение оболочки скопировать файл БД в проект ответьте Да) и настройте его только на чтение всех полей таблицы, используя построитель Query Builder.

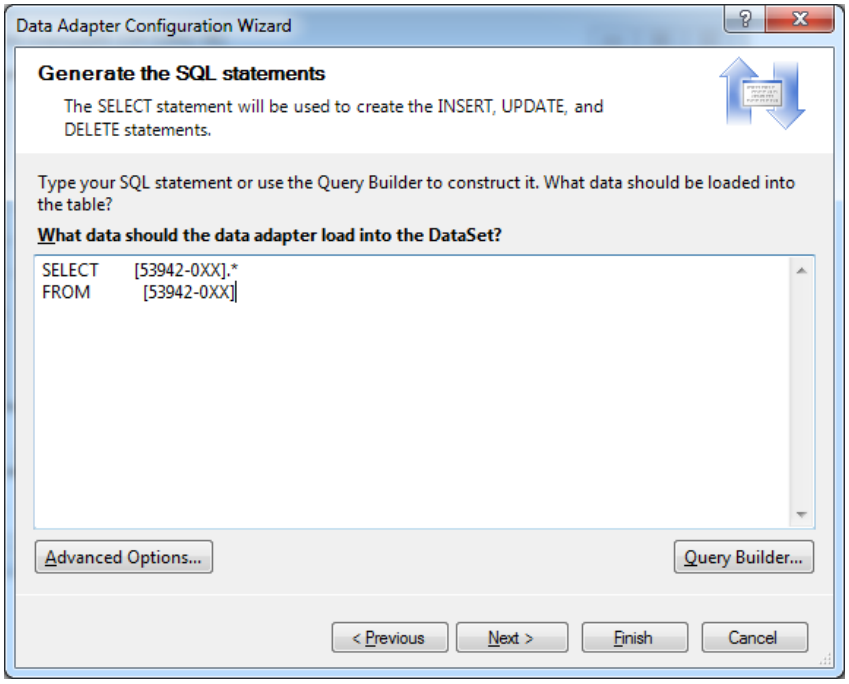

Рисунок 4.1 – Окно конфигуратора DataAdapter

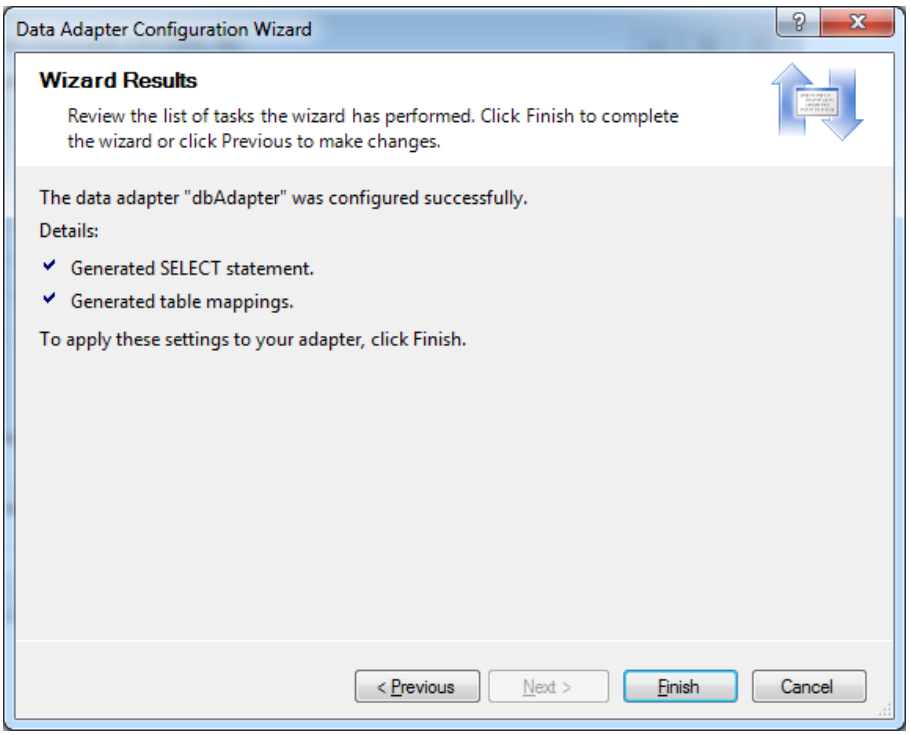

Рисунок 4.2 – Результат конфигурирования компонента DataAdapter

Выделив поставщик oleDbDataAdapter1 и командой меню Data/Generate Dataset создаем для него класс dsID и объект набора данных с автоматически сгенерированным именем dsID1 [17, 18].

Далее установим свойства компонент согласно приведенному выше описанию. Для заполнения компонент содержимым полей таблицы, при загрузке формы следует вызвать метод Fill, компонента DataAdapter.

```
private void MainForm_Load(object sender, EventArgs e)
    {
       oleDbDataAdapter1.Fill(dsID);
    }
```

4.3 Создание формы «AboutBox»

Для создания диалогового окна «О программе» его просто необходимо добавить в проект, путем вызова контекстного меню, новую WindowsForm и в разделе выбора

типа указать окно о программе [19]. Добавив окно просто следует организовать его вызов, для чего в обработчике данного события необходимо прописать следующий код:

```
private void btnAbout_Click(object sender, EventArgs e)
{
    AboutBox About = new AboutBox();
    About.ShowDialog();
}
```

Как видно из рисунка, на форме имеются типовые компоненты, которые определяются автоматически при заполнении соответствующих пунктов меню AssemlyInformation. Для его вызова следует выбрать пункт Project главного меню и войти в раздел Properties. Далее появится следующее окно редактирования свойств проекта:

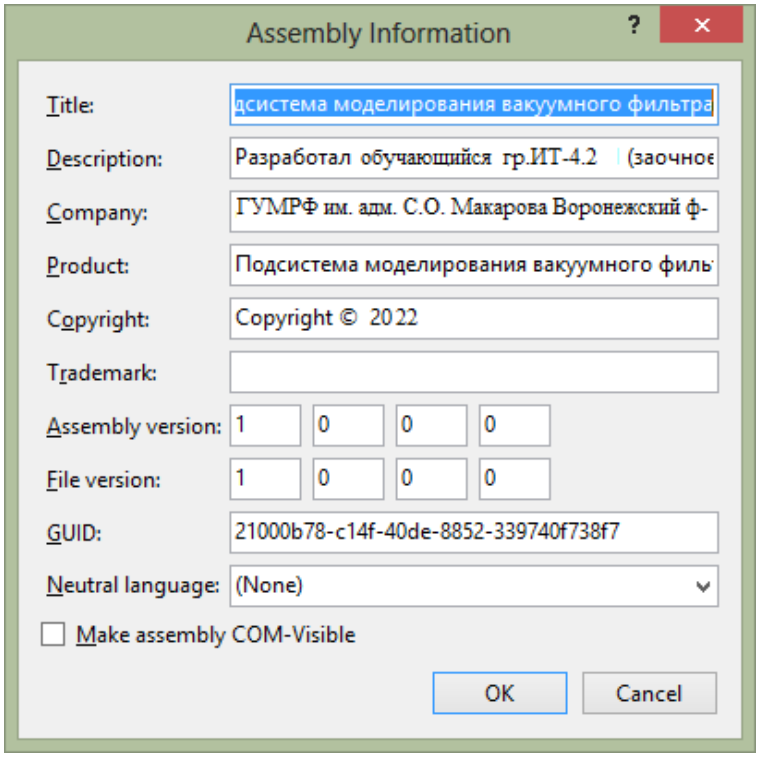

Рисунок 4.4 – Меню информации о проекте

4.4 Создание основной процедуры построения вакуумного фильтра

Реализация на языке программирования C# будет выглядеть согласно листингу, приведенному ниже.

```
class Filtr
{
    //Типоразмеры для построения вакуумного фильтра
    private double dDv;
    private double dDn;
    private double dL;
    private double dA;
    private double dB;
    private double dAotv;
    private double dBotv;
```

Класс DrawingFiltr является основным блоком по реализации и построению конструкции в NX и связан с некоторой переменной окружения класса-temp.

```
DrawingFiltr temp;
```

В процессе инициализации класса запущен конструктор с параметрами: диаметром и длинами, определяющими условия проведения построения.

```
temp.DrawKorpus(A, Aotv, B, Botv, Dnar, Dvnut, L)
```

Закрытые компоненты представлены в виде функций, методов класса, возвращаемых логическое значение: (TRUE) в случае успешного завершения и записи результатов, (FALSE)- в противном случае.

Программная реализация компонентов твердотельной модели сводится к выполнению базовых операций, выполняемых аналогично как при ведении проектов инженером-конструктором. Следуя разработанной схеме реализации, программные компоненты твердотельной модели рекомендуется определить в закрытой части класса. Последовательным вызовом созданных блоков с набором параметров решается задача подготовительного этапа [20].

Для получения компонент твердотельной модели вакуумного фильтра необходимо поочередно выполнить последовательности кода, отвечающие за добавление каждой детали сборку и их построение. В целом данный участок кода выглядит так:

```
DrawingFiltr.DrawCap(A, Aotv, B, Botv, Dnar, Dvnut);
DrawingFiltr.DrawElem(Db, De, L);
DrawingFiltr.DrawScrew();
DrawingFiltr.DrawKorpus(A, Aotv, B, Botv, Dnar, Dvnut, L);
DrawingFiltr.add_Cap();
DrawingFiltr.add_Elem();
DrawingFiltr.add_Screw(Aotv, Botv);
```

Программная реализация модели вакуумного фильтра с учетом передаваемых параметров в функцию выглядит следующим образом:

Построение компонента внутри функции связано с рядом функций, определенных в NX Open API. Для инициализации процесса построения необходимо выполнить следующий программный код. Комментарии приведены в листинге программы. `A, Aotv, B, Botv, Dnar, Dvnut, L`

```
//Внутренние данные для преобразования
    string Dv = text(dDv / 2);
    string Dn = text(dDn / 2);
    string L = text(dL);
    string B = text(dB);
    string A = text(dA / 2);
    string Aotv = text(dAotv);
    string Botv = text(dBotv);
    //Удаление старой копии файла, если она на диске имеется
    delete_files("Filtr.prt");
    Session theSession = Session.GetSession();
    // ----------------------------------------------
    //   Файл->Новый...
    // ----------------------------------------------
   //Создаем новую деталь, объявляя переменную fileNew1
    FileNew fileNew1;
   //Получение сеанса начала работы с объектом
    fileNew1 = theSession.Parts.FileNew();
```

```
//Используем шаблон модели в миллиметрах
  fileNew1.TemplateFileName = "model-plain-1-mm-template.prt";
//Приложение, связанное с процессом моделирования
  fileNew1.Application = FileNewApplication.Modeling;
//Создаваемый объект будет выполнен в указанных
//единицах измерения, в данном случае- миллиметрах
  fileNew1.Units = NXOpen.Part.Units.Millimeters;
//Путь сохранения файла
  fileNew1.NewFileName = "Filtr.prt";
```

Следующий этап программного проектирования указанного компонента заключается в построении контура и вращении его вокруг выбранной оси. Основанием вращения будет выступать одна из координатных осей. Выбор плоскостей, определение исходных точек построения, системы координат приведены в листинге

```
//Создать новый эскиз на плоскости построения
sketchInPlaceBuilder1 = workPart.Sketches.CreateNewSketchInPlaceBuilder(nullSketch);
//Создать переменную окружения по взятию значений единиц //измерения
 Unit unit1 = (Unit)workPart.UnitCollection.FindObject("MilliMeter");
//Построение эскиза по указанному пути (определение исходной //плоскости, системы координат)
SketchAlongPathBuilder sketchAlongPathBuilder1;
sketchAlongPathBuilder1 = workPart.Sketches.CreateSketchAlongPathBuilder(nullSketch); sketchAlongPathBuilder1.PlaneLocation.Expression.RightHandSide = "0";
//Определить координатную плоскость построения
DatumPlane datumPlane1 = (DatumPlane)workPart.Datums.FindObject("DATUM_CSYS(0) XZ plane");
//Создать новый точечный объект
Point3d point1 = new Point3d();
```

```
//Установить точку на координатной плоскости sketchInPlaceBuilder1.PlaneOrFace.SetValue(datumPlane1, workPart.ModelingViews.WorkView, point1);
NXOpen.Features.DatumCsys datumCsys1 = (NXOpen.Features.DatumCsys)workPart.Features.FindObject("DATUM_CSYS(0)");
Point point2 = (Point)datumCsys1.FindObject("HANDLE R-850");
sketchInPlaceBuilder1.SketchOrigin = point2;
sketchInPlaceBuilder1.PlaneOrFace.Value = null;
sketchInPlaceBuilder1.PlaneOrFace.Value = datumPlane1;
//Определяем ось Y, принадлежащей координатной плоскости
DatumAxis datumAxis1 = (DatumAxis)workPart.Datums.FindObject(" DATUM_CSYS(0) Y axis");
sketchInPlaceBuilder1.Axis.Value = datumAxis1;
```

Построение замкнутого контура производится следующим с учетом параметров, заданных пользователем.

```
Point3d startPoint1 = new Point3d(0.0, 0.0, 0.0);
Point3d endPoint1 = new Point3d(0.0, 0, dd1 / 2);
//Определить переменную окружения линии
Line line1;
//Создать линию по двум точкам
line1 = workPart.Curves.CreateLine(startPoint1, endPoint1);
//В активном эскизе добавить линию без привязок
theSession.ActiveSketch.AddGeometry(line1, NXOpen.Sketch.InferConstraintsOption.InferNoConstraints);
//Определить переменную окружения ограничений в эскизе
NXOpen.Sketch.ConstraintGeometry geom1_1;
//Связать линию с геометрией ограничений
geom1_1.Geometry = line1;
//Ограничение по начальной вершине
geom1_1.PointType = NXOpen.Sketch.ConstraintPointType.StartVertex;
```

```
//Сплайн определяющих позиций точек-от 0.
geom1_1.SplineDefiningPointIndex = 0;
//Определить переменную окружения ограничений в эскизе
NXOpen.Sketch.ConstraintGeometry geom2_1;
//Связать точку с созданной геометрией
geom2_1.Geometry = point2;
geom1_1.SplineDefiningPointIndex = 0;
geom2_1.PointType = NXOpen.Sketch.ConstraintPointType.None;
geom2_1.SplineDefiningPointIndex = 0;
//Определить переменную окружения ограничений геометрии эскиза
SketchGeometricConstraint sketchGeometricConstraint1;
//Создать взаимосвязь из двух ограничений в активном эскизе
sketchGeometricConstraint1 = theSes-
sion.ActiveSketch.CreateCoincidentConstraint(geom1_1, geom2_1);
//Определить переменную окружения ограничений в эскизе
NXOpen.Sketch.ConstraintGeometry geom1;
//Связать с линией
geom1.Geometry = line1;
geom1.PointType = NXOpen.Sketch.ConstraintPointType.None;
geom1.SplineDefiningPointIndex = 0;
SketchGeometricConstraint sketchGeometricConstraint2;
sketchGeometricConstraint2 = theSes-
sion.ActiveSketch.CreateVerticalConstraint(geom1);
```

Вращение компонента производится в следующей сессии кода:

```
Section section3;
NXOpen.Features.RevolveBuilder revolveBuilder1;
revolveBuilder1 = workPart.Features.CreateRevolveBuilder(nullFeatures_Feature);
```

```
//Установка начального положения поворота(в градусах) revolveBuild-
er1.Limits.StartExtend.Value.RightHandSide = "0";
//Установка конечного положения поворота(в градусах)
revolveBuilder1.Limits.EndExtend.Value.RightHandSide = "360";
revolveBuilder1.Offset.StartOffset.RightHandSide = "0";
revolveBuilder1.Offset.EndOffset.RightHandSide = "5";
section3.SetAllowedEntityTypes(NXOpen.Section.AllowTypes.OnlyCurves);
NXOpen.Features.Feature[] features1 = new NXOpen.Features.Feature[1];
NXOpen.Features.SketchFeature sketchFeature1 = (NXO-
pen.Features.SketchFeature)feature1;
features1[0] = sketchFeature1;
CurveFeatureRule curveFeatureRule1;
curveFeatureRule1 = workPart.ScRuleFactory.CreateRuleCurveFeature(features1);
section3.AllowSelfIntersection(false);
SelectionIntentRule[] rules4 = new SelectionIntentRule[1];
rules4[0] = curveFeatureRule1;
Point3d helpPoint1 = new Point3d();
section3.AddToSection(rules4, line2, nullNXObject, nullNXObject, helpPoint1, NXO-
pen.Section.Mode.Create, false);
revolveBuilder1.Section = section3;
Direction direction1;
direction1 = workPart.Directions.CreateDirection(line8, Sense.Forward, NXO-
pen.SmartObject.UpdateOption.WithinModeling);
Point nullPoint = null;
Axis axis1;
axis1 = workPart.Axes.CreateAxis(nullPoint, direction1, NXO-
pen.SmartObject.UpdateOption.WithinModeling);
revolveBuilder1.Axis = axis1;
revolveBuilder1.ParentFeatureInternal = false;
```

```
NXOpen.Features.Feature feature2;
feature2 = revolveBuilder1.CommitFeature();
```

Построение скругления происходит согласно ниже приведенному листингу программы:

```
// ----------------------------------------------
//   Меню: Вставить->Detail Feature->Edge Blend...
// ----------------------------------------------
NXOpen.Features.EdgeBlendBuilder edgeBlendBuilder1;
edgeBlendBuilder1 = workPart.Features.CreateEdgeBlendBuilder(nullFeatures_Feature);
NXOpen.GeometricUtilities.BlendLimitsData blendLimitsData1;
blendLimitsData1 = edgeBlendBuilder1.LimitsListData;
Point3d origin2 = new Point3d(0.0, 0.0, 0.0);
Vector3d normal1 = new Vector3d(0.0, 0.0, 1.0);
Plane plane1;
plane1 = workPart.Planes.CreatePlane(origin2, normal1, NXOpen.SmartObject.UpdateOption.WithinModeling);
ScCollector scCollector1;
scCollector1 = workPart.ScCollectors.CreateCollector();
Edge[] seedEdges1 = new Edge[1];
NXOpen.Features.Revolve revolve1 = (NXOpen.Features.Revolve)feature2;
Edge edge1 = (Edge)revolve1.FindObject("EDGE * [CURVE 4 0] * [CURVE 5 0] {(34.6410161513775,33,20)(0,33,-40)(-34.6410161513775,33,20) REVOLVED(2)}");
seedEdges1[0] = edge1;
EdgeMultipleSeedTangentRule edgeMultipleSeedTangentRule1;
edgeMultipleSeedTangentRule1 = workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges1, 0.5, true);
SelectionIntentRule[] rules2 = new SelectionIntentRule[1];
rules2[0] = edgeMultipleSeedTangentRule1;
scCollector1.ReplaceRules(rules2, false);
```

```
        edgeBlendBuilder1.Tolerance = 0.0254;
        edgeBlendBuilder1.AllInstancesOption = false;
        edgeBlendBuilder1.RemoveSelfIntersection = true;
        edgeBlendBuilder1.ConvexConcaveY = false;
        edgeBlendBuilder1.RollOverSmoothEdge = true;
        edgeBlendBuilder1.RollOntoEdge = true;
        edgeBlendBuilder1.MoveSharpEdge = true;
        edgeBlendBuilder1.TrimmingOption = false;
        edgeBlendBuilder1.OverlapOption = NXO-
pen.Features.EdgeBlendBuilder.Overlap.AnyConvexityRollOver;
        edgeBlendBuilder1.BlendOrder = NXO-
pen.Features.EdgeBlendBuilder.OrderOfBlending.ConvexFirst;
        edgeBlendBuilder1.SetbackOption = NXO-
pen.Features.EdgeBlendBuilder.Setback.SeparateFromCorner;
        int csIndex1;
        csIndex1 = edgeBlendBuilder1.AddChainset(scCollector1, R);
        NXOpen.Features.Feature feature3;
        feature3 = edgeBlendBuilder1.CommitFeature();
        edgeBlendBuilder1.Destroy();
```

По завершению построения детали производится ее сохранение средствами NX:

```
PartSaveStatus partSaveStatus2;
partSaveStatus2 = workPart.Save(NXOpen.BasePart.SaveComponents.True,
```

4.5 Тестирование программы

Работоспособность и функциональность программы оценивалась путем многократного проектирования вакуумных фильтров с разными наборами типоразмеров. Полученные результаты подтверждают стабильный и надежный характер функционирования системы.

Проверка производилась по следующим критериям:

1. Требуемая функциональность;

В ходе данной поверки производился многократный процесс проектирования, при различных начальных условиях и входных характеристиках. При этом система показала правильное взаимодействие с таблицей базы данных, корректный расчет параметров, надежное построение моделей [19, 21]. При этом становится очевидным, что вся совокупность определенных на этапе требований функций программы, реализована в полном объеме. Пример результата автоматизированного построения вакуумного фильтра приведен на рисунке 2.2.

2. Защита от некорректного ввода;

При проверке на корректность ввода были введены некорректные исходные параметры, некорректность которых заключалась в вводе нулевых значений числовых параметров, вводе заведомо больших значений величин, вводе данных неверного формата. При этом, в каждом из этих случаев программа не позволяла вводить значения вообще, ввиду особой настройки соответствующих компонент. Это необходимо сделать ввиду того что добавление или изменение соответствующих стандартных конструктивных параметров не должно быть доступно простому пользователю. Тем самым результаты испытаний на корректность ввода система прошла без сбоев.

3. Стресс-тест при максимальной загрузке.

Стресс тест показал, что система корректно работает при большом количестве параллельных процессов, занимающих большую часть аппаратных ресурсов ЭВМ. Что касается перегрузки программы вычислительными процессами, следует отметить невозможность перегрузить программу вычислениями, в связи с особенностями ее реали-

зации. Эти особенности заключаются в реализации линейного подхода к проектированию с запараллеливанием только процессов проверки ошибок. Однако, возможность нестабильной работы программы остается открытой, в связи с невозможностью испытаний программы в условиях многочисленности комбинаций архитектур используемых программных компонент.

4.6 Виды обеспечений САПР

В основе математического обеспечения подсистемы лежит методика пересчета геометрических параметров вакуумного фильтра в параметры геометрических примитивов, из которых состоит базовый эскиз операций вращения, вытягивания, массив.

Применительно к разработанной подсистеме в качестве алгоритмического языка, как компонента лингвистического обеспечения выступает язык C#. Приведем краткие характеристики этого языка:

- C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности - как FCL, так и CLR;
- C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
- C# является мощным объектным языком с возможностями наследования и универсализации;
- C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от C++ к C#;
- сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства C++ как указатели, адресация, разыменование, адресная арифметика;
- благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной ма-

шиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;

- мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

- реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

Кроме того, для программирования NX 7.5 возможно использовать и другие языковые средства, такие как VB и C++.

Программное обеспечение библиотеки проектирования вакуумных фильтров состоит из среды разработки, монитора САПР для управления работой различных систем пользователей в ОС, системы управления базой данных (СУБД) для поддержки баз данных САПР и самих алгоритмов решения различных задач, программных библиотек.

Среда разработки, использованная в проектировании библиотеки – MS Visual Studio 2013. Прикладная библиотека разрабатывалась с учетом архитектуры x86 микропроцессоров Intel, а также их аналогов и прошла успешное тестирование в Siemens PLM NX 7.5.0.32 32bit под управлением операционной системы Windows SP3. Возможен запуск прикладной библиотеки в семействе ОС Windows 8 64 битной архитектурой. Для подключения базы данных и ряда компонентов потребуется установка дополнительных модулей визуальной среды программирования .NET, Framework 3.5 - программной платформы Microsoft и элемент управления базами данных [22].

Монитор подсистемы представляет собой совокупность диалоговых окон, появляющихся на экране в процессе работы программы и позволяющих пользователю проводить контроль.

Система управления базой данных позволяет производить редактирование разработанной базы данных и определяет параметры доступа к таблице базы данных.

Программные средства, обеспечивающие разработку библиотеки: MS Visual Studio 2013, MS Access 2013.

Для работы программы необходимо располагать IBM-совместимой ПЭВМ с установленной операционной системой Windows 7 и выше.

Для удобства подключения библиотеки предлагается вариант с автоматическим размещением требуемых программных компонентов в операционной системе Windows и установленным клиентом NX. Кроме того, при ручной установке библиотеки необходимо: скопировать файл базы данных Shayba.accdb в системную директорию NX (..\UGS\NX 7.5\UGII\). Прикладная библиотека разрабатывалась с учетом архитектуры x86 микропроцессоров Intel, а также их аналогов и прошла успешное тестирование в Siemens PLM NX 7.5.0.32 32bit под управлением операционной системы Windows SP3. Возможен запуск прикладной библиотеки в семействе ОС Windows 8 64 битной архитектурой. Для подключения базы данных и ряда компонентов потребуется установка дополнительных модулей визуальной среды программирования .NET, Framework 4 - программной платформы Microsoft и элемент управления базами данных. Для удобства все выше перечисленные компоненты (за исключением платформонезависимой виртуальной машины Framework 3.5) включены в состав установочного дистрибутива. Установка указанных компонентов для корректной работы обязательна! Для вызова установленной библиотеки в среде NX необходимо воспользоваться комбинацией клавиш CTRL+U или соответствующим разделом в меню системы проектирования.

Выбрав файл библиотеки Filtr.dll загрузится ее главное окно, приведенное на рисунке 3.32.

Выбрав соответствующие параметры вакуумного фильтра пользователю необходимо нажать кнопку Построить. Для выхода из библиотеки, следует нажать кнопку закрыть. Для просмотра информации о программе, нажать кнопку «?». После построения вакуумного фильтра, библиотека автоматически будет закрыта. Для повторного ее запуска следует повторить загрузку.

Для нормального функционирования программы необходимо следующее техническое обеспечение, обусловленное применяемым в ходе работы подсистемы программным обеспечением, а именно системой моделирования NX 7.5:

Минимальная конфигурация ПЭВМ, определяется системными требования NX 7.5:

Рекомендуется компьютер, оснащенный процессором Intel Core 2 Duo/ AMD x64 или совместимым с частотой 2800 МГц или более (двухядерные процессоры)*. Минимальная частота процессора - 2400 МГц.

Рекомендуется 4 Гб ОЗУ. Минимально допустимый объем - 2 ГБ (при наличии 1 ГБ ОЗУ возможно снижение производительности и функциональности).

6 ГБ свободного места на жестком диске.*

Монитор и видеоадаптер Super VGA с разрешением 1280 X 1024 или более высоким.

Дисковод для компакт- или DVD-дисков

Клавиатура и мышь Microsoft или совместимое указывающее устройство.

Информационное обеспечение представляет базу данных входных и справочных параметров, описание которых подробно приведено в п. 2, а также ее систему управления, которой в данном случае является файл-серверная СУБД Access 2013. Структура таблицы описана в п. 2 дипломного проекта.

Что касается анализа существующих СУБД, следует отметить, что на данном этапе развития современных программных средств управления БД, на рынке ПО присутствуют следующие основные СУБД, поддерживающих архитектуру клиент-сервер: Microsoft SQL Server, MySQL, Interbase. В ходе сравнения таких параметров данных систем, как скорость, надежность, цена, простота использования, защищенность, выбор падает на СУБД Access. Данный выбор обусловлен тем, что данная система имеет наименьшую стоимость по сравнению с остальными при требуемой функциональности. При этом данный выбор носит временный характер, поскольку взаимодействие системы NX целесообразнее всего организовывать и интегрированной средой информационного взаимодействия Teamcentre.

Организационное обеспечение разработанного программного средства *рассматривает* структуру службы САПР, а также определяет взаимоотношения подразделений этой службы с подразделениями-пользователями САПР.

При внедрении программы на ООО «ЦентрГорода», г. Воронеж», необходимо отделу информационных технологий, отвечающему за внедрение, эксплуатацию и адаптацию разрабатываемых систем конструкторской и технологической подготовки производства, организовать обеспечение функционирования разработанного программного средства, и выполнять следующие виды работ:

1. Определять состав технических средств, необходимых для внедрения САПР;
2. Получать у разработчиков программное обеспечение для автоматизации проектно-конструкторских работ и документацию к нему;
3. Организовывать централизованное хранение документации по программному обеспечению и программ на носителях, а также поддерживать эти программы в работоспособном состоянии;
4. Получать у разработчиков системы и в родственных организациях, уже внедривших САПР, массивы справочно-нормативной информации, необходимой для машинного выполнения проектных работ, организовывать с привлечением специализированных подразделений предприятия пополнение и корректировку этих массивов с учетом специфики рассматриваемого типа производства;
5. Организовывать обучение сотрудников работе с помощью САПР;
6. Оценивать эффективность использования САПР.

Опираясь на знания, полученные в ходе преддипломной практики, определим те структурные подразделения завода, которые исходя из их функционального назначения, следует оснастить разработанным программным средством. Установлено, что вопросами проектирования и расчета параметров редукторов, занимаются следующие подразделения:

- Специализированные конструкторские бюро;
- Конструкторское бюро инструмента и приспособлений.

В каждом из указанных выше подразделений следует организовать как минимум по одному специализированному автоматизированному рабочему месту, ориентированному на решение именно данного круга задач. Информация должна централизованно приходить с организованных рабочих мест в техническую лабораторию для формирования окончательной конструкторской документации. В данной лаборатории следует располагать и сервер Teamcenter.

Правовое обеспечение системы определяется комплесом должностных инсрукций и руководств, для сотрудников занимающихся поддержкой и эксплуатацией системы, а также ее внедрением. Поскольку на каждом из прдеприятий, уровень автоматизиции разлицается в корне, то необходимо индивидуально подходить к разработке таких документов, учитывая опыт каждого из предприятий в отдельности.

Заключение

Результатом выполнения выпускной квалификационной работы служит разработанная подсистема проектирования конструкции вакуумного фильтра, основными функциями которой являются: построение вакуумных фильтров как библиотечного элемента, с невозможностью внесения изменений в созданную конструкцию, работа со списком параметров конструкции вакуумного фильтра, поиск и выбор типоразмера вакуумного фильтра по любому из параметров, передача данных о геометрии приспособления в тело проектирующей процедуры, загрузка и выгрузка библиотеки в память системы и обратно, визуализация эскиза проектируемого изделия, запуск проектирующей процедуры, вызов последовательности выполнения NX API.

Разработана функциональная модель подсистемы. Для программы составлены схема взаимодействия данных информационного обеспечения и алгоритм работы.

Для программного средства описаны математическое, лингвистическое, информационное, техническое, программное, методическое и организационное обеспечения.

Список литературы

1. Бадд Т. Объектно-ориентированное программирование в действии. / Т. Бадд. - СПб.: Питер, 1997. - 464 с.

2. Биллиг В.А. Основы программирования на C#. / В.А. Биллиг - М.: Изд-во «Интернет-университет информационных технологий — ИНТУИТ.ру», 2017. - 488 с.

3. Гуннерсон Э. Введение в C#. Библиотека программиста. / Э. Гуннерсон - СПб.: Питер, 2021. - 304 с.

4. Павловская Т. А. C#. Программирование на языке высокого уровня / Т.А. Павловская - СПб.: Питер, 2020. - 432 с.

5. Павловская Т. А. C/C++. Программирование на языке высокого уровня / Т.А. Павловская - СПб.: Питер, - 2021. - 464 с.

6. Петцольд Ч. Программирование для MS Windows на C#, ч.1 / Ч. Петцольд. - М.: Издательско-торговый дом «Русская Редакция», 2022. - 576 с.

7. Петцольд Ч. Программирование для MS Windows на C#, ч.2 / Петцольд. - М.: Издательско-торговый дом «Русская Редакция», 2022. - 624 с.

8. Прайс Д. Visual C#.NET. Полное руководство / Д. Прайс, М. Гандэрлой - Киев: «Век», 2014. - 960 с.

9. Секунов Н. Разработка приложений на C++ и C#. Библиотека программиста /Н. Секунов - СПб.: Питер, 2016. - 608 с.

10. Тай Т. Платформа .NET. Основы. /Т. Тай, Х.К. Лэм - СПб.: Символ-Плюс, 2019. - 336 с.

11. Троелсен Э. C# и платформа .NET. Библиотека программиста. / Э. Троелсен - СПб.: Питер, 2022. - 796 с.

12. Фролов А.В. Язык C#. Самоучитель. / А.В. Фролов, Г.В. Фролов М.: Диалог-МИФИ, 2017. -560 с.

13. Шилдт Г. C#: учебный курс. / Г.С. Шилд - СПб.: Питер, 2018. - 512 с.

14. Хомичков Г.И. Идентификация мониторинга по текущим данным на основе правил базы знаний //Автоматизация и современные технологии. – 2012. - №4. – С. 33-35.
15. Википедия Свободная библиотека. Электрон. дан. – Режим доступа: http://ru.wikipedia.org/wiki/Экспертное_Оценивание.
16. А.В. Соколов Методика оценки эффективности поиска по нечетким характеристикам в автоматизированных информационных системах // Автоматизация и современные технологии. – 2019. - №3. – С. 25-27.
17. Маклаков С.В BPWin и ERWin. CASE-средства разработки информационных систем. / С.В. Маклаков– М.:Диалог-МИФИ, 2018. – 223 с.
18. Интернет университет информационных технологий. Жизненный цикл программного обеспечения ИС – Электрон. дан. – Режим доступа: http://www.intuit.ru/department/se/devis/2/.
19. Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. Булдаков М.Б. Дзюбло Д.А. Повышение эффективности управления на основе реинжиниринга бизнес-процессов. – Электрон. дан. – Режим доступа: http://ej.kubagro.ru.
20. Реинжиниринг бизнес-процессов и экономические информационные системы. Актуальные проблемы гуманитарных и естественных наук. №1 2010 Карпухина Н.Н.
21. Официальный сайт компании Microsoft – Электрон. дан. – Режим доступа: http://www.microsoft.com/rus/dynamics/nav/overview.mspx.
22. Официальный сайт компании Epicor – Электрон. дан. – Режим доступа: http://www.epicor.com/russia/Products/Pages/iScala.aspx.