Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет»
институт Математики и информационных технологий
кафедра Информационных систем и компьютерного моделирования

УТВЕРЖДАЮ
Зав. каф. ИСКМ
_____ А.В. Хоперсков
«____» _____ 20____ г.

ОТЧЕТ
о прохождении производственной практики, преддипломной практики

Студент	Пестряков Павел Андреевич	
Направление подготовки	09.03.04 Программная инженерия	
Группа	ПРИ-171	
Руководитель практики	С.О. Зубович	к.ф.-м.н., ведущий специалист отдела аналитики и тестирования Волгоградского филиала ООО «Миго-групп»
Ответственный за организацию практики	С.С. Храпов	к.ф.-м.н., доцент каф. ИСКМ
Место прохождения практики	ФГАОУ ВО «ВолГУ», каф. ИСКМ	
Сроки прохождения практики	«____» _____ 20___ г.	«____» _____ 20___ г.

1. Отметка о прохождении практики

Прибыл на практику

«10» февраля 2020 г.

Студент

_____ Фамилия И.О.

(подпись)

Руководитель практики от университета

_____ Фамилия И.О.

(подпись)

Выбыл с практики

«23» мая 2020 г.

Студент

_____ Фамилия И.О.

(подпись)

Руководитель практики от университета

_____ Фамилия И.О.

(подпись)

2. Цель и индивидуальные задания на период практики

В качестве цели практики были выделены следующие направления:

1. получение навыков и опыта по направлению подготовки;

2. сбор материала для написания выпускной квалификационной работы (бакалаврской работы);

3. …

В результате прохождения производственной практики, преддипломной должны быть сформированы следующие компетенции:

УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач;

УК-4 Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах);

ПК-5 Способен осуществлять концептуальное, функциональное и логическое проектирование программно-информационных систем;

ПК-1 Способен проводить научно-исследовательские и опытно-конструкторские разработки;

ПК-3 Способен разрабатывать тестовые случаи, проводить тестирование и исследовать результаты.

За период практики студент должен:

Получить знания:

-

-

приобрести умения:

-

-

Сформировать навыки:

-

-

-

Трудовые функции, формируемые на производственной практике, преддипломной

Трудовые функции из профессиональных стандартов (указать выходные данные этих стандартов)

Профессиональный стандарт "Программист".

06.001 Разработка программного обеспечения

Обобщенные трудовые функции			Трудовые функции		
Код	Наименование	уровень квалификации	наименование	код	уровень (подуровень) квалификации
A	Разработка и отладка программного кода	3	Формализация и алгоритмизация поставленных задач	A/01.3	3
			Написание программного кода с использованием языков программирования, определения и манипулирования данными	A/02.3	3

Индивидуальные задания студента от руководителя практики от университета:

1. …

2. …

3. …

4. …

3. Описание выбранной темы выпускной квалификационной работы (бакалаврской работы)

Название темы

Научный руководитель

Описание темы

Актуальность работы

(не менее 1 страницы)

4. Характеристика полученных результатов

1 Анализ предметной области для создания программного комплекса для управления процессом 3D-печати

1.1 Обзор метода аддитивного производства. Моделирование методом послойного наплавления

1.1.1 FDM-технология создания трехмерных моделей

Основой 3D-печати является технология послойного нанесения расплавленной полимерной нити – FDM-технология (Fused Deposition Modeling), которая была разработана в 1988 году С. Крампом. Технология является достаточно простой и понятной в применении, при этом она справляется с достаточно сложной геометрией моделей [10]. Также применение принтеров, которые функционируют на основе FDM-технологии, является наиболее экономически выгодным. Именно поэтому таким методом 3D-печати пользуются чаще всего.

FDM-технология печати заключается в следующем: выдавливающая головка с контролируемой температурой, которая называется экструдер, разогревает до полужидкого состояния нити из ABC-пластика, воска или поликарбоната, и с высокой точностью подает полученный термопластичный моделирующий материал тонкими слоями на рабочую поверхность 3D-принтера, называемую платформой или координатным столом. Слои наносятся друг на друга, соединяются между собой и отвердевают, постепенно формируя готовое изделие. На рисунке 1 представлен процесс FDM-печати с изображением основных составляющих принтера и материала, составляющего структуру печатаемого изделия.

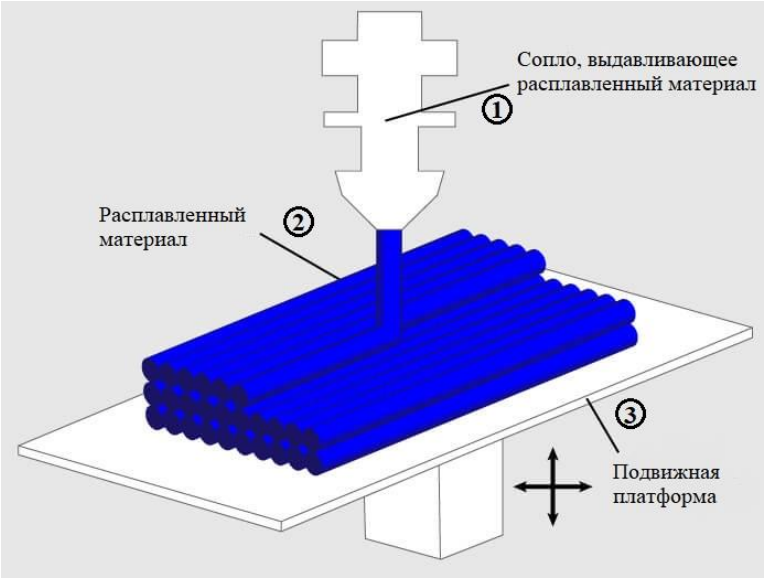

Рисунок 1 – Процесс FDM-печати, где 1 – сопло экструдера, 2 – расплавленный материал, 3 – платформа

1.1.2 Преимущества FDM технологии создания трехмерных моделей

FDM-технология обладает рядом значительных и немаловажных характеристик, которые обозначают её преимущество по сравнению с другими технологиями 3D-печати [15], а именно:

1) наиболее простой принцип печати, который легко реализуется на основе широкого спектра выбора электро-компонентов;

2) возможность использования наиболее распространенных термопластичных материалов с разнообразными характеристиками, в том числе невредных для здоровья (как во время процесса, так и использования готового изделия) и не нуждающихся в специальных условиях хранения и использования;

3) реализация в виде компактных персональных печатающих устройств, которые не требуют специализированных знании по установке и эксплуатации;

4) прототипирование объектов со сложной геометрией и полостями, которые не могут быть реализованы при помощи других технологий;

5) отсутствие шумовых загрязнений и отходов производства, требующих утилизации или специальных мест для установки;

6) большая разрешающая способность (до 20 микрон), возможность печати сразу несколькими материалами, которые могут быть различных цветов;

7) довольно низкая себестоимость, как самих устройств, так и применяемых материалов, возможность самостоятельной сборки печатающего устройства из готового конструктора или набора компонентов;

8) распечатанные изделия имеют высокие эксплуатационные характеристики и могут применяться как серийные изделия;

9) открытость технологии позволяет работать над совершенствованием и внедрением 3D-печатающих устройств в различные сферы.

1.2 Обработка 3D-модели в процессе трехмерной печати

1.2.1 Роль слайсинга в процессе трехмерной печати

В процессе 3D-печати одним из основных этапов является подготовка 3D-модели к печати. На данном этапе производится обработка модели в специальном программном обеспечении, которое называют слайсером [23]. Он выполняет генерацию управляющего G-кода для 3D-принтера по данным, которые хранятся в файле STL-формата.

Главной задачей слайсера является деление модели на плоские слои, по которым будет производиться пошаговое создание модели [10]. Также слайсер создаёт набор команд, задающих направление движения экструдера и платформы, для корректной печати изделия. Здесь же по возможности может указываться температура нагревания платформы и экструдера, темп работы устройства охлаждения [30].

Вся необходимая информация по созданию модели преобразуется в G-код, который передаётся 3D-принтеру и является для него инструкцией, по которой он осуществляет свою работу.

На сегодняшний день предоставлен большой выбор среди слайсеров, которые имеют свои достоинства и недостатки. Одни имеют широкий диапазон возможностей, другие выполняют узкий круг выполняемых задач.

1.2.2 Особенности переменной толщины слоя в процессе обработки 3D-модели

Возможность установки различных значений толщины для разных слоёв решает проблему выбора между качеством и временем. Переменная толщина ускоряет процесс печати при малозаметном изменении её качества. Более малый интервал между слоями задаётся для таких частей модели, где необходима высокая детализация. Если в какой-то части модели нет в этом необходимости, то на этом участке устанавливается максимально разрешённая высота печати. При этом, не для каждой модели имеет смысл использовать алгоритм определения переменной толщины слоя. Если изделие не имеет изгибов, то её стоит печатать с постоянной толщиной, при которой будет получено удовлетворительное качество изделия за приемлемые затраты времени на его изготовление.

Переменная толщина слоёв модели уменьшает объём занимаемой памяти, необходимой для хранения обработанной модели. Такая модель будет хранить меньше информации о печатаемых слоях, чем модель с минимальным интервалом между слоями.

Также большое количество печатаемых слоёв увеличивает нагрузку на 3D-принтер. При долгом беспрерывном использовании оборудования увеличивается риск искажений, засоряется сопло экструдера. Это может сказаться на качестве печати или вообще привести к браку в изделии.

1.2.3 Особенности формата файлов STL для хранения данных о 3D-модели

Так сложилось, что большинство современных слайсеров работают в основном с файлами формата STL. Этот формат разрабатывался для технологии StereoLithography, но сейчас применяется практически во всех технологиях 3D-печати [36].

Файл формата STL применяется для хранения данных о 3D-модели. Любая трёхмерная модель состоит из треугольников. Каждый из них имеет по три вершины, которые описываются наборами из трёх координат, и единичному вектору нормали, который показывает направление ориентации лицевой стороны треугольника. В совокупности информация об одном треугольнике образует фасет. Файл формата STL хранит в себе информацию о группе таких фасетов, которые и формируют 3D-модель. На рисунке 2 показан пример бинарного STL-файла.

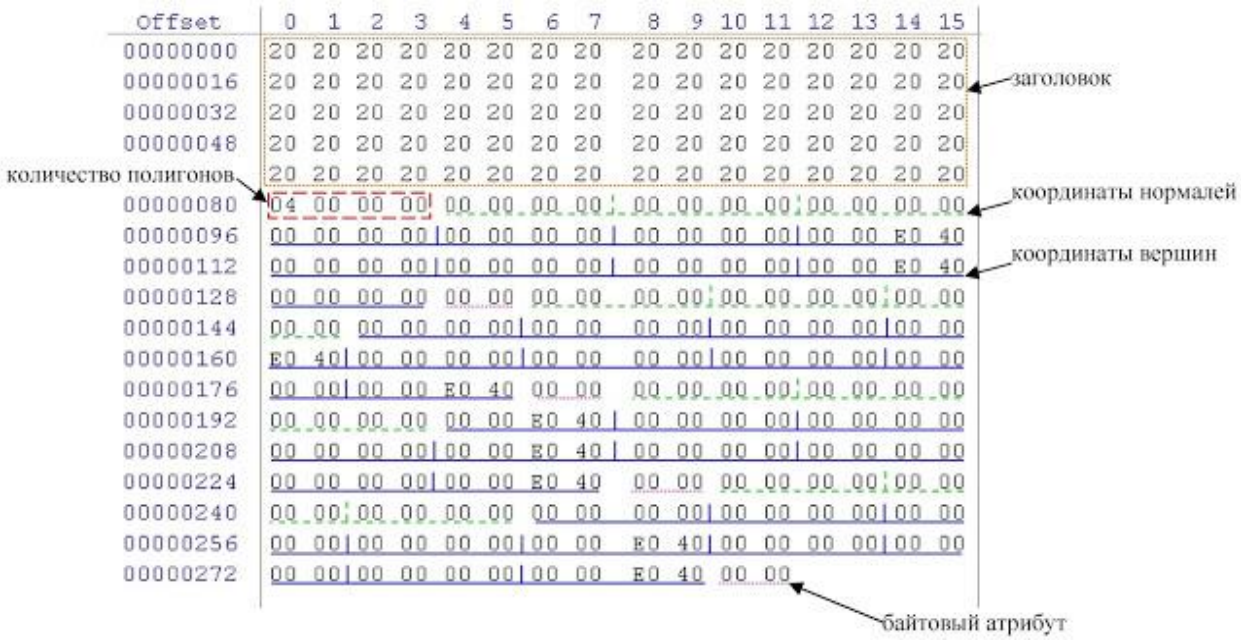

Рисунок 2 – Содержание бинарного STL файла

Перечисление вершин в файле должно быть в определённом порядке, от этого зависит ориентация вектора нормали. Направление вектора нормали

определяется правилом правой руки: большой палец руки показывает направление вектора нормали, а остальные согнутые пальцы показывают направление нумерации и порядок описания вершин. На рисунке 3 показано определение вектора нормали для фасета.

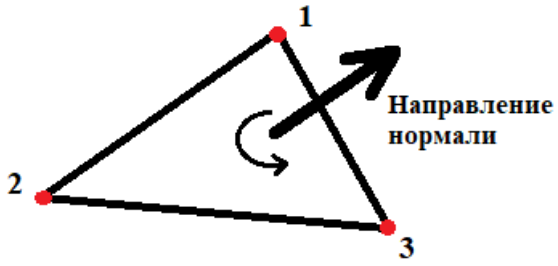

Рисунок 3 – Вектор нормали для фасета

Файлы формата STL представляются в двух видах: текстовом и бинарном. Применение бинарных файлов позволяет снизить объём хранимых данных, следовательно, при его обработке слайсером потребуется меньше времени.

1.3 Обзор языков программирования, необходимых для реализации программного комплекса

1.3.1 Язык программирования для формирования управляющего кода

1.3.2 Язык программирования для обработки модели и реализации графической оболочки программы

1.3.3 Визуализация 3D-модели посредством языка программирования Python

2 Проектирование программного комплекса для управления процессом 3D-печати

2.1 Проектирование информационной модели программного комплекса для управления процессом 3D-печати

2.1.1 Назначения и возможности программного комплекса

Разрабатываемый программный комплекс включает в себя модуль для формирования плоских срезов переменной толщины из трёхмерной модели и создания на их основании управляющего кода, модуль визуализации и модуль управления FDM 3D-принтером.

Исходными данными для программы являются бинарный файл формата STL, который содержит модель, минимальное и максимальное значение высоты слоя, а также необходимая информация об используемом принтере и параметрах, необходимых для настройки процесса печати.

Для корректной работы программы исходные данные должны соответствовать следующим требованиям:

1) 3D-модель должна соответствовать требованиям, предъявляемым к модели, предназначенной для печати на 3D-принтере;

2) файл STL должен быть бинарным;

3) шаг по оси Oz должен представлять из себя число, минимальное и максимальное значение которого находится в границах указанных значений, при этом максимальное должно быть меньше длины модели по рассматриваемой оси. Однако для корректной последующей печати по созданному коду при выборе максимальной толщины слоя необходимо учитывать возможности используемого принтера;

4) целая и дробные части указываемых значений разделяются точкой;

5) физические параметры используемого принтера, указываемые в программе, должны быть достоверны, в противном случае возможен некачественный результат печати по сформированному коду.

Результатами работы программы являются сформирований G-код с адаптивной толщиной слоёв модели, визуализация модели на 3D-сцене, распечатанная 3D-модель с переменной толщиной слоёв.

2.1.2 Планирование процесса проектирования и реализации программного комплекса

Важной частью любого проекта и одним из первых шагов по его разработке является процесс проектирования. Процесс детального проектирования программных средств заключается в декомпозиции его структуры до элементарных программных компонентов, которые могут быть верифицированы относительно установленных требований к архитектуре программного продукта [7].

Описание структуры проекта позволяет определить состав входящих в него задач и взаимосвязей между ними. Данная процедура может быть выполнена различными способами, одним из которых является составление диаграммы Ганта [5]. Диаграмма Ганта является структурой проекта и используется для визуального представления плана, графика работ, описания задач и их взаимосвязей. Это один из удобных методов для планирования проектов [8]. Для разрабатываемого программного комплекса был выбран именно этот способ описания структуры проекта.

Изначально был составлен список наиболее важных этапов разработки проекта, таких как «Подготовка к разработке программного комплекса», «Разработка информационной модели программного комплекса», «Разработка математической модели программного комплекса» и непосредственно сам этап «Разработки программного комплекса». Также были определены типы связей между задачами верхнего уровня, сроки их выполнения, произведена их декомпозиция и определены вехи [1]. Данные задачи отображены на рисунке 6.

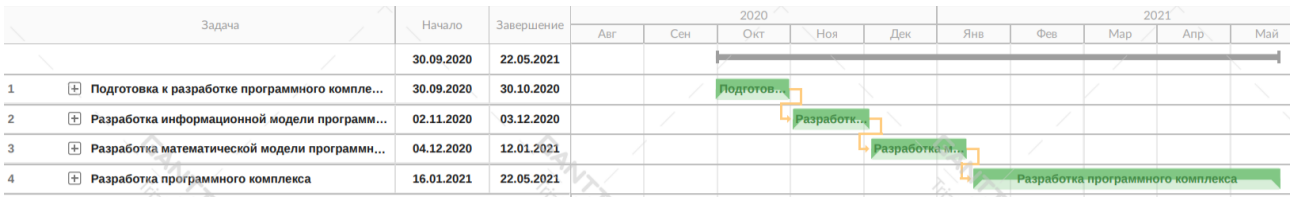

Рисунок 6 – Диаграмма Ганта

Рассмотрим задачу «Подготовка к разработке программного комплекса», которая изображена на рисунке 7.

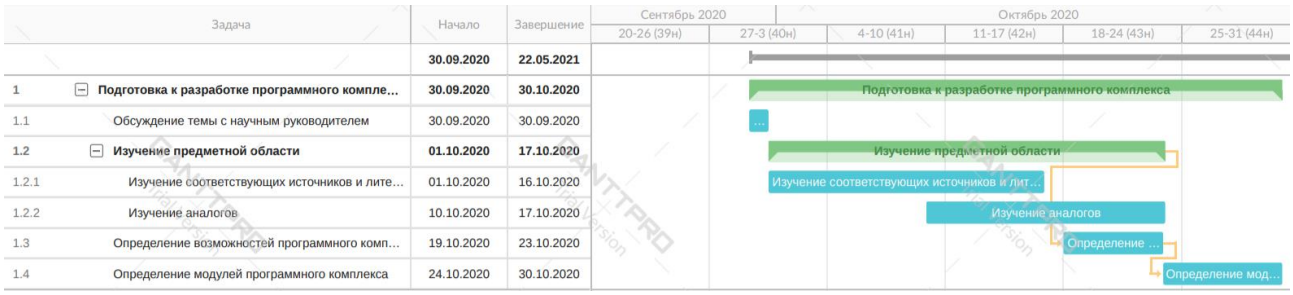

Рисунок 7 – Диаграмма Ганта задачи «Подготовка к разработке программного комплекса»

На данном этапе были определены несколько подзадач. Первой является «Обсуждение темы с научным руководителем». Второй подзадачей является «Изучение предметной области», которая включает в себя «Изучение соответствующих источников и литературы» и «Изучение аналогов». Третьей и четвертой подзадачами являются «Определение возможностей программного комплекса» и «Определение модулей программного комплекса» соответственно.

Задача «Разработка информационной модели программного комплекса» изображена на рисунке 8.

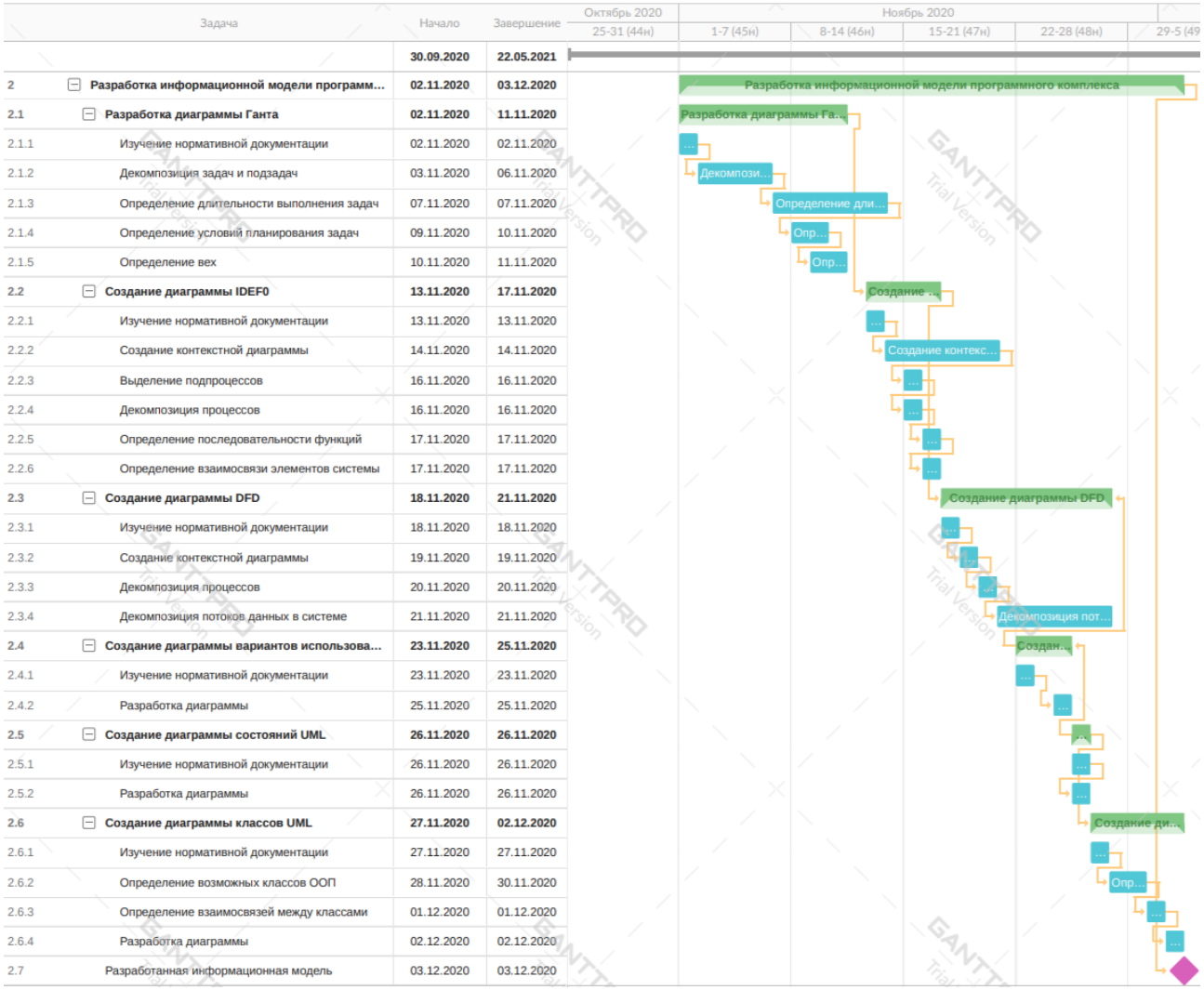

Рисунок 8 – Диаграмма Ганта задачи «Разработка информационной модели программного комплекса»

Рассмотрим задачу «Разработка информационной модели программного комплекса». Целью данной задачи является составление диаграмм, которые описывают структуру, состояния, способы использования, функционал, характеризующие программный комплекс, а также этапы и сроки его разработки. Подзадача «Разработка диаграммы Ганта» включает в себя «Изучение нормативной документации», «Декомпозиция задач и подзадач», «Определение длительности выполнения задач», «Определение условий планирования задач», «Определение вех». Подзадача «Создание диаграммы IDEF0» подразделяется на «Изучение нормативной документации», «Создание контекстной диаграммы», «Выделение подпроцессов», «Декомпозиция процессов», «Определение

последовательности функций», «Определение взаимосвязи элементов системы». Подзадача «Создание диаграммы DFD» включает в себя «Изучение нормативной документации», «Создание контекстной диаграммы», «Декомпозиция процессов», «Декомпозиция потоков данных в системе». Подзадачи «Создание диаграммы вариантов использования UML» и «Создание диаграммы состояний UML» имеют идентичную декомпозицию и подразделяются на «Изучение нормативной документации» и «Разработка диаграммы». Подзадача «Создание диаграммы классов UML» состоит из таких подзадач, как «Изучение нормативной документации», «Определение возможных классов ООП», «Определение взаимосвязей между классами» и «Разработка диаграммы». Также в данной задаче определена веха «Разработанная информационная модель».

Задача «Разработка математической модели программного комплекса» показана на рисунке 9. Данная задача включает в себя алгоритмы, применяемые для обработки модели и формирования управляющего кода. Каждая подзадача подразумевает описание определенного порядка действий и математических расчетов для выполнения данного этапа обработки модели. Эта задача состоит из таких подзадач, как «Составление алгоритма деления модели на слои», «Составление алгоритма нахождения переменной толщины слоя», «Составление алгоритма построения контуров», «Составление алгоритма построения дублирующих контуров», «Составление алгоритма заполнения контуров модели», «Составление алгоритма расчета количества пластика и построения управляющего кода», а также веху «Разработанная математическая модель».

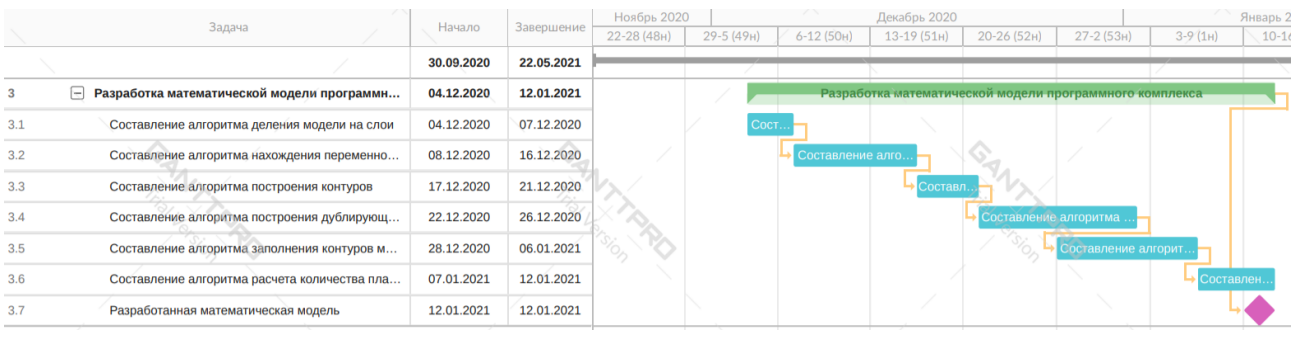

Рисунок 9 – Диаграмма Ганта задачи «Разработка математической модели программного комплекса»

Следующая задача «Разработка программного комплекса» изображена на рисунке 10.

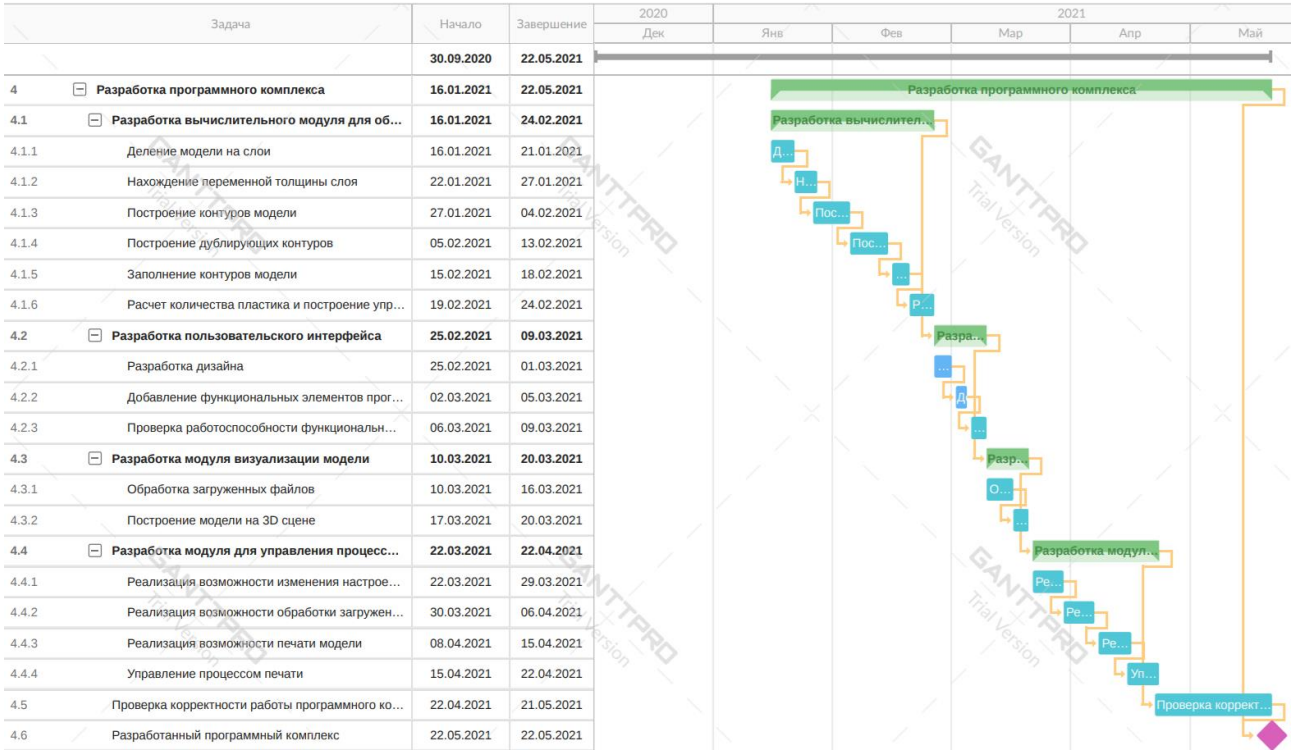

Рисунок 10 – Диаграмма Ганта задачи «Разработка программного комплекса»

При выполнении задачи «Разработка программного комплекса» должна производиться разработка модулей, составляющих программный комплекс. Заключающей подзадачей является «Разработанный программный комплекс». Данная задача имеет самую большую длительность выполнения. Подзадача «Разработка пользовательского интерфейса» состоит из подзадач «Разработка дизайна», «Добавление функциональных элементов программы», «Проверка работоспособности функциональных элементов». Подзадача «Модификация разработанного модуля для обработки модели» включает в себя такие этапы, как «Разработка алгоритма заполнения слоев модели» и «Доработка алгоритма формирования управляющего кода». Подзадача «Разработка модуля визуализа-

ции модели» включает в себя подзадачи «Обработка загруженных файлов» и «Построение модели на 3D сцене». Подзадача «Разработка модуля для управления процессом печати на FDM 3D принтере» декомпозирована на «Реализация возможности изменения настроек принтера», «Реализация возможности обработки загруженного G-кода», «Реализация возможности печати модели», «Управление процессом печати». Подзадача «Проверка корректности работы программного комплекса и исправление найденных ошибок» подразумевает под собой поиск неисправностей программного комплекса, с дальнейшим их устранением.

2.1.3 Функциональное моделирование программного комплекса

Методология функционального моделирования IDEF0 – это система принципов, положений и методов описания системы в целом как множества взаимозависимых действий, или функций [2]. IDEF0 имеет функциональную направленность – функции системы исследуются независимо от объектов, которые обеспечивают их выполнение. Функциональная точка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации [20]. С помощью методологии IDEF0 было осуществлено функциональное моделирование разрабатываемого программного комплекса.

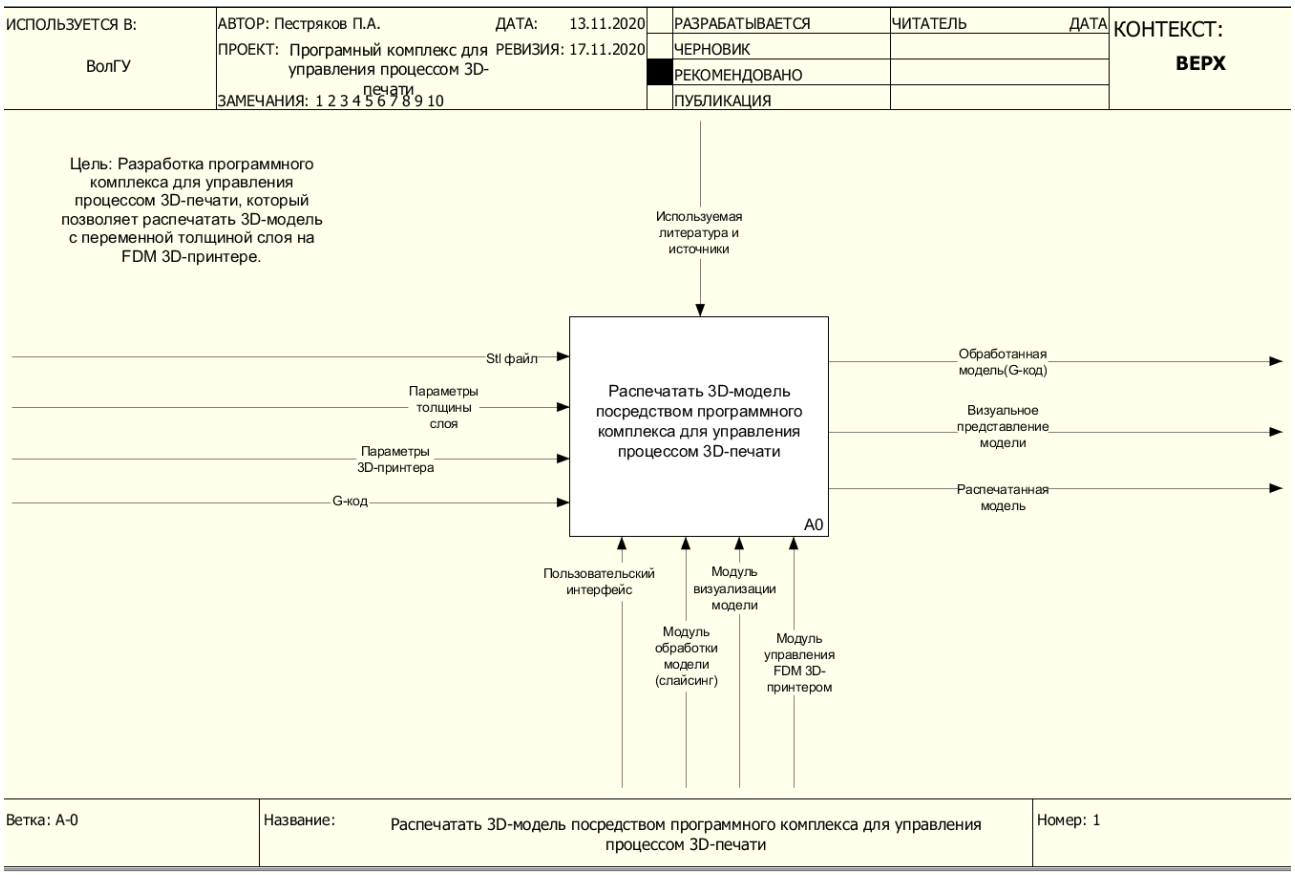

Рисунок 10 – Контекстная диаграмма

На рисунке 10 показан главный процесс диаграммы. В качестве процесса на контекстной диаграмме было выбрано «Распечатать 3D-модель посредством программного комплекса для управления процессом 3D-печати».

Целью является: «Разработка программного комплекса для управления процессом 3D-печати, который позволяет распечатать 3D-модель с переменной толщиной слоя на FDM 3D-принтере».

Компоненты контекстной диаграммы:

1) стрелка механизма использования: Пользовательский интерфейс, Модуль обработки модели (слайсинг), Модуль визуализации модели, Модуль управления FDM 3D-принтером;

2) стрелка управления: Используемая литература и источники;

3) входная стрелка: Stl файл, Параметры толщины слоя, Параметры 3D-принтера, G-код;

4) выходная стрелка: Обработанная модель (G-код), Визуальное представление модели, Распечатанная модель.

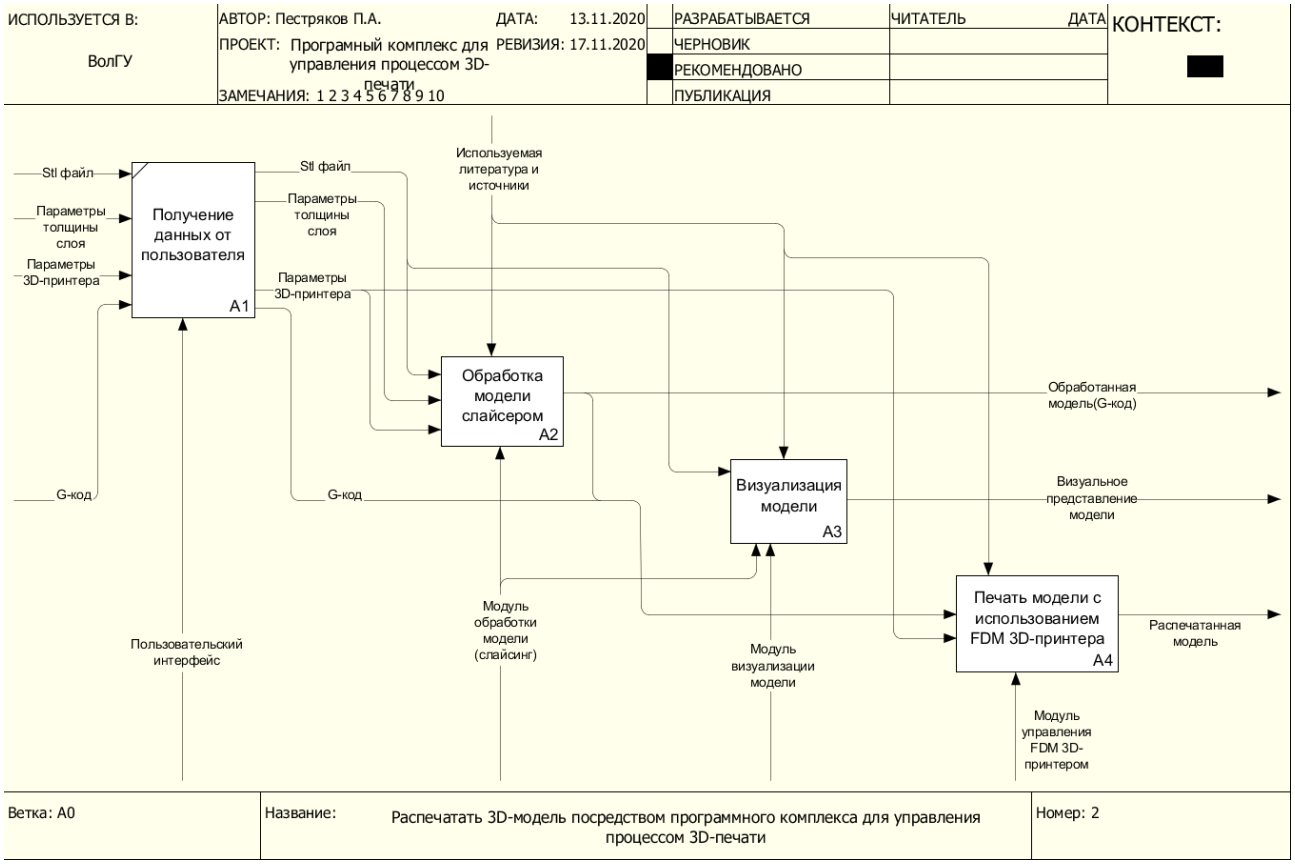

Рисунок 11 – Диаграмма декомпозиции процесса A0

На рисунке 11 представлена диаграмма декомпозиции главного процесса A0, который подразделяется на:

1) получение данных от пользователя (процесс A1);

2) обработка модели слайсером (процесс A2);

3) визуализация модели (процесс A3);

4) печать модели с использованием FDM 3D принтера (процесс A4).

Результатом выполнения процесса «Обработка модели слайсером» является «Обработанная модель (G-код)». Процесс визуализации модели имеет выходную стрелку «Визуальное представление модели», а у процесса печати модели с использованием FDM 3D-принтера выходная стрелка это «Распечатанная модель».

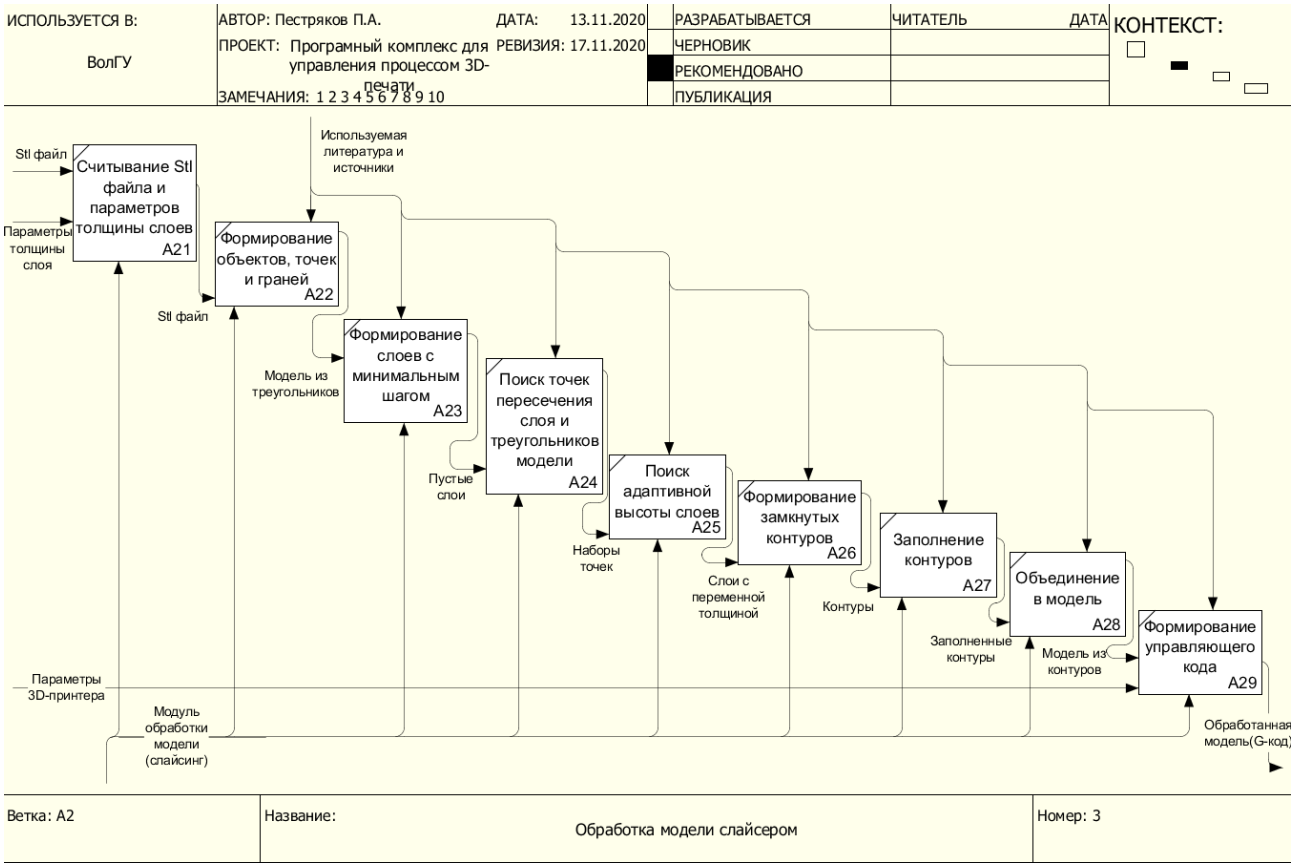

Рисунок 12 – Диаграмма декомпозиции процесса «Обработка модели слайсером»

На рисунке 12 представлена диаграмма декомпозиции процесса A2, который подразделяется на:

1) считывание Stl файла и параметров толщины слоев (процесс A21);

2) формирование объектов, точек и граней (процесс A22);

3) формирование слоев с минимальным шагом (процесс A23);

4) поиск точек пересечения слоя и треугольников модели (процесс A24);

5) поиск адаптивной высоты слоев (процесс A25);

6) формирование замкнутых контуров (процесс A26);

7) заполнение контуров (процесс A27);

8) объединение в модель (процесс A28);

9) формирование управляющего кода (процесс A29).

Процесс A21 отвечает за считывание данных с загруженного пользователем файла формата бинарного Stl и параметров толщины слоя, минимальное и максимальное значения.

В процессе A22 производится формирование таких объектов как точек, составляющих вершины треугольников и координаты вектора нормали, и граней треугольников. В результате получается модель из треугольников и направляется на последующий процесс.

В ходе выполнения процесса A23 производится формирование слоев будущей обработанной модели, высота которых равна минимальному значению толщины слоя, установленное пользователем. Сформированные пустые слои направляются на процесс A24, на котором происходит поиск точек пересечения треугольников с данными слоями. Эти точки образуют наборы точек, принадлежащих слою.

Процесс A25 предназначен для поиска адаптивной высоты слоя, которая позволяет сократить количество печатаемых принтером слоев, с минимальными потерями качества готовой модели.

Из слоев с переменной толщиной, полученных на предыдущем этапе образуются замкнутые контуры на процессе A26, заполнение которых осуществляется на процессе A27.

Процесс A28 отвечает за объединение заполненных контуров в модель, которая далее обрабатывается процессом A29. Результатом выполнения всех процессов является «Обработанная модель (G-код)».

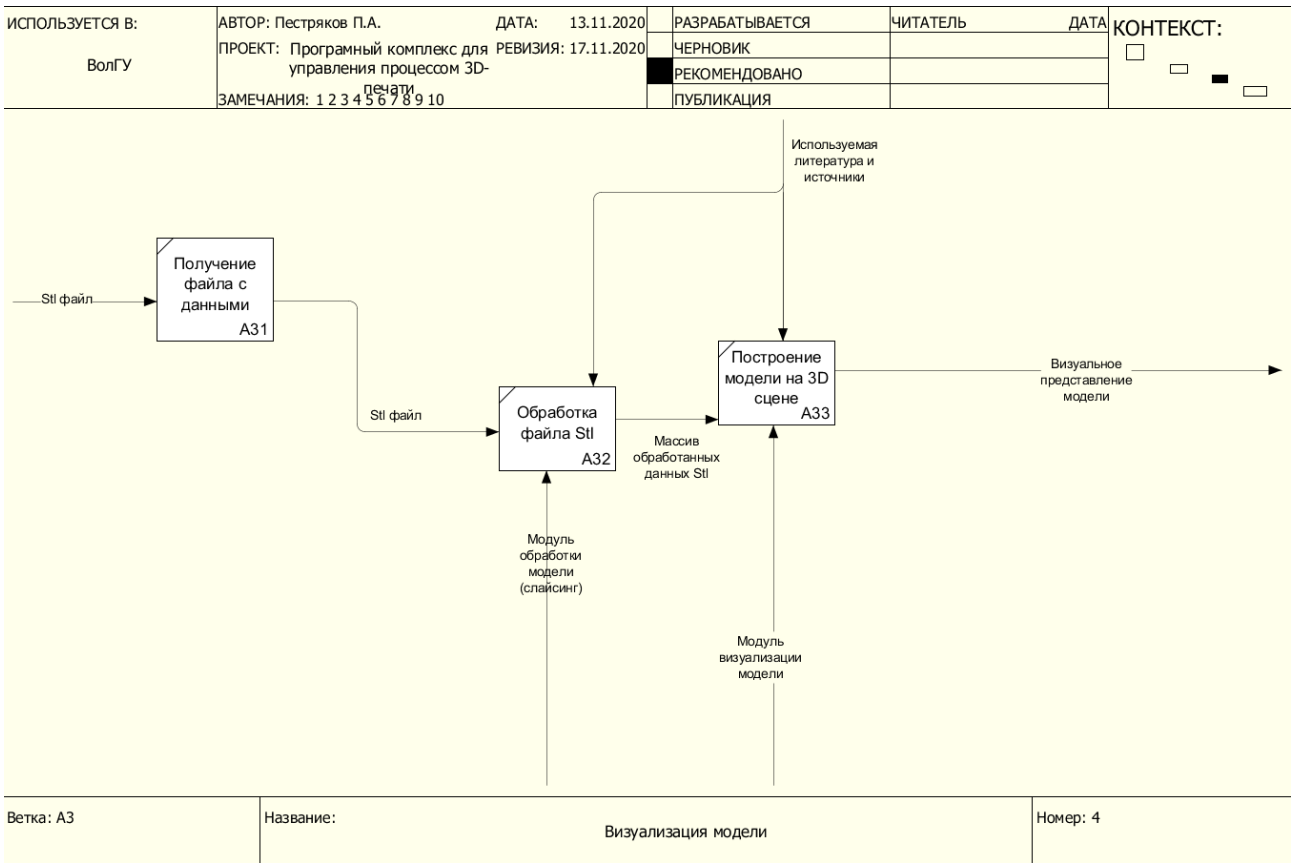

Рисунок 13 – Диаграмма декомпозиции процесса «Визуализация модели»

На рисунке 13 изображена декомпозиция процесса A3, который делится на:

1) получение файла с данными (процесс A31);

2) обработка файла Stl (процесс A32);

3) построение модели на 3D сцене (процесс A34).

Из входного Stl файла считываются и обрабатываются данные о 3D-модели. Результатом выполнения процессов на данном этапе является визуальное представление 3D-модели, состоящей из треугольников.

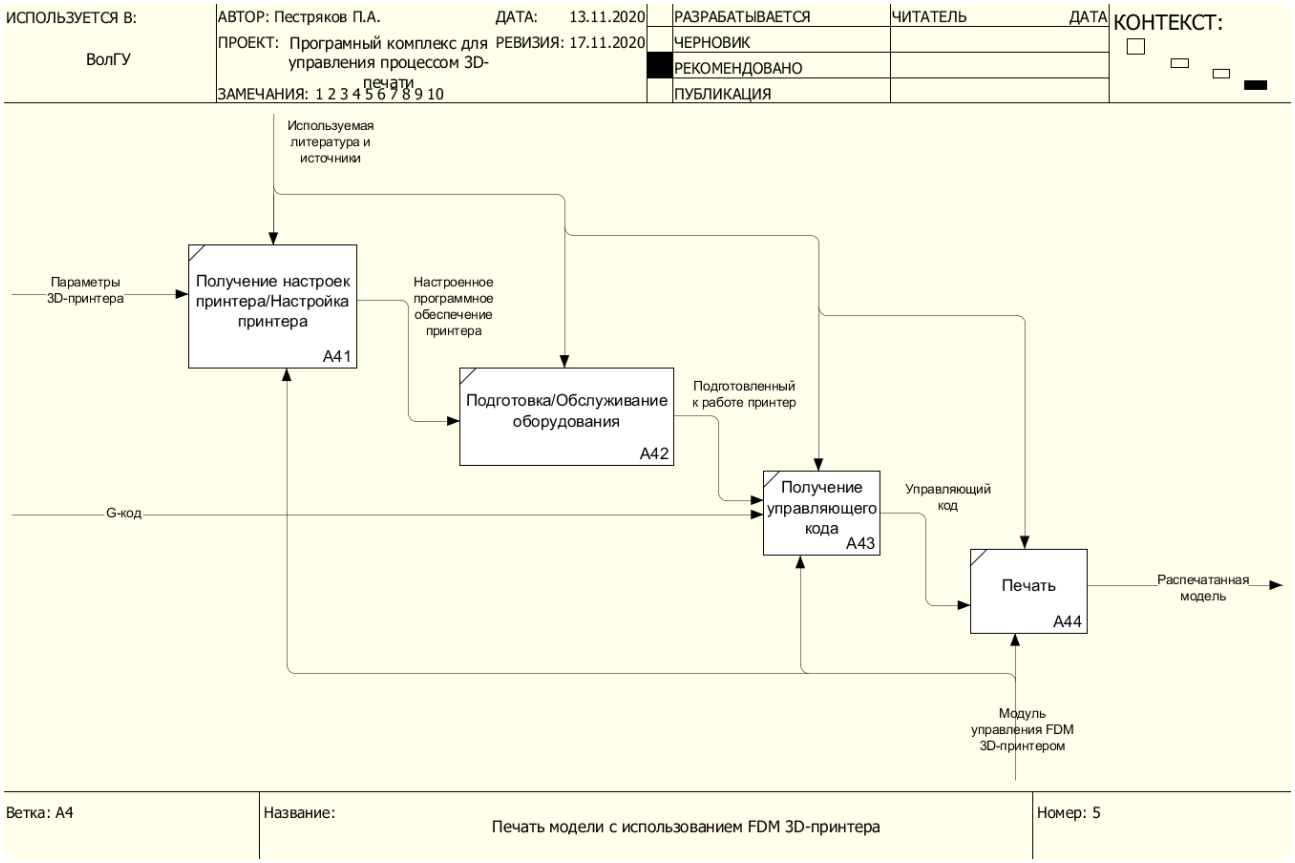

Рисунок 14 – Диаграмма декомпозиции процесса «Печать модели с использованием FDM 3D-принтера»

На рисунке 14 изображена декомпозиция процесса A4, который делится на:

1) получение настроек принтера/Настройка принтера (процесс A41);

2) подготовка/Обслуживание оборудования (процесс A42);

3) получение управляющего кода (процесс A43);

4) печать (процесс A44).

Результатом выполнения процессов на данном этапе является распечатанная модель.

2.1.4 Моделирование потоков данных программного комплекса

Диаграммы потоков данных (Data Flow Diagrams - DFD) используются для описания движения документов и обработки информации как дополнение к IDEF0. В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, стрелки в DFD показывают лишь то, как объекты (включая данные) движутся от одной работы к другой. Это инструмент моделирования, который визуально представляет систему в виде сети функциональных процессов [12].

Целью DFD является визуальное представление каждого процесса, взаимосвязей между ними. Также DFD показывает как каждый процесс преобразует входные данные в выходные [20].

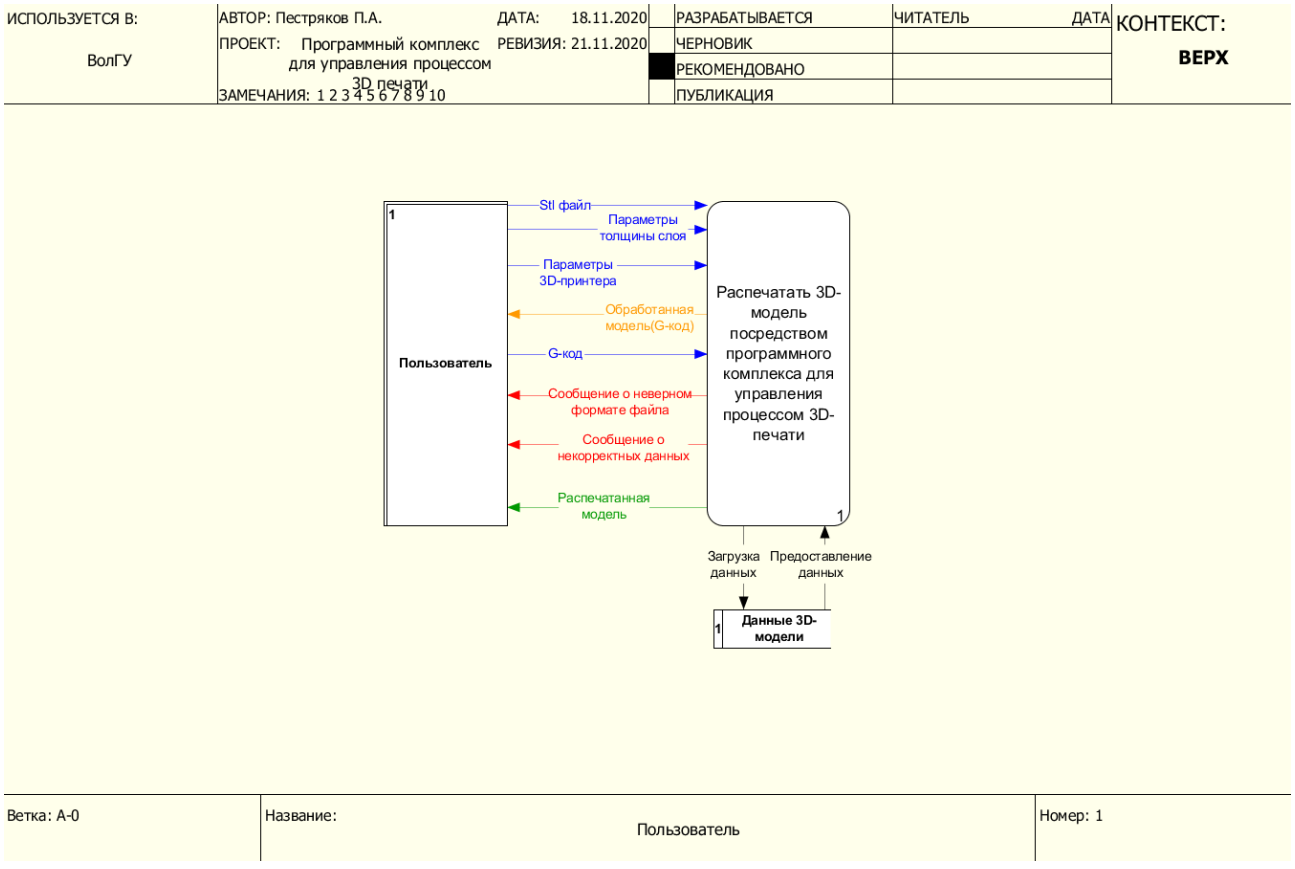

Рисунок 15 – Контекстная диаграмма

На рисунке 15 представлена контекстная диаграмма потоков данных, которая включает в себя:

1) процесс: «Распечатать 3D-модель посредством программного комплекса для управления процессом 3D-печати»;

2) внешняя сущность: «Пользователь»;

3) хранилище данных: «Данные 3D-модели».

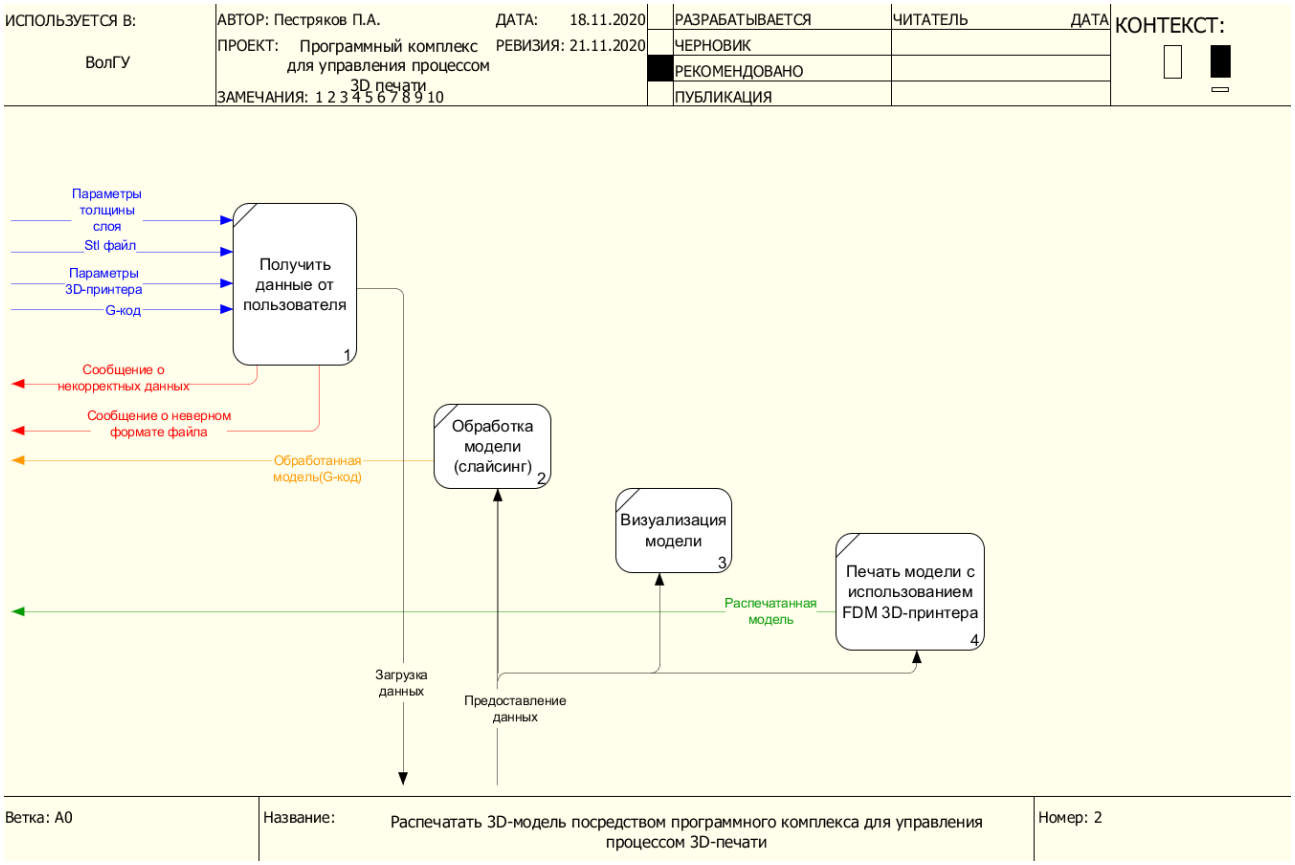

Рисунок 16 – Декомпозиция контекстной диаграммы

На рисунке 16 изображена декомпозиция контекстной диаграммы. Необходимо производить декомпозицию основного процесса, чтобы лучше понимать предметную область, основной процесс и его этапы. Главный процесс состоит из более конкретных небольших процессов, на которых производятся конкретные действия. Декомпозиция данного процесса включает:

1) получить данные от пользователя;

2) обработка модели (слайсинг);

3) визуализация модели;

4) печать модели с использованием FDM 3D-принтера.

Процесс 1 включает в себя получение данных от пользователя, таких как параметры минимальное и максимальное значения толщины слоя, Stl файл, со-

держащий изначальную модель, параметры для настройки 3D-принтера и управления его работой, G-код, содержащий данные обработанной 3D-модели. Все данные с этого процесса направляются в хранилище данных для дальнейшего их распределения между другими процессами. Также на данном этапе производится проверка корректности введенных пользователем данных и загруженных файлов. В случае возникновения внештатной ситуации формируется сообщение с соответствующей ошибкой или предупреждением.

Процесс 2 отвечает за обработку полученных данных с Stl файла. Производятся необходимые манипуляции и расчеты, которые формируют данные для составления управляющего G-кода. В результате получается обработанная модель, состоящая из слоёв с адаптивной высотой.

Процесс 3 необходим для визуализации представленных данных. Визуальное представление модели позволяет пользователю наглядно оценить исходную 3D-модель, с которой будут производиться дальнейшие действия.

Процесс 4 включает в себя настройку FDM 3D-принтера для печати, а также сам процесс печати необходимой модели. В результате данного процесса получается напечатанная модель с переменной толщиной слоя.

2.1.5 Диаграммы, отражающие структуру, состояния и способы использования программного комплекса

На рисунке 17 представлена диаграмма классов, отражающая классы, используемые в ходе выполнения программы и взаимосвязи между ними.

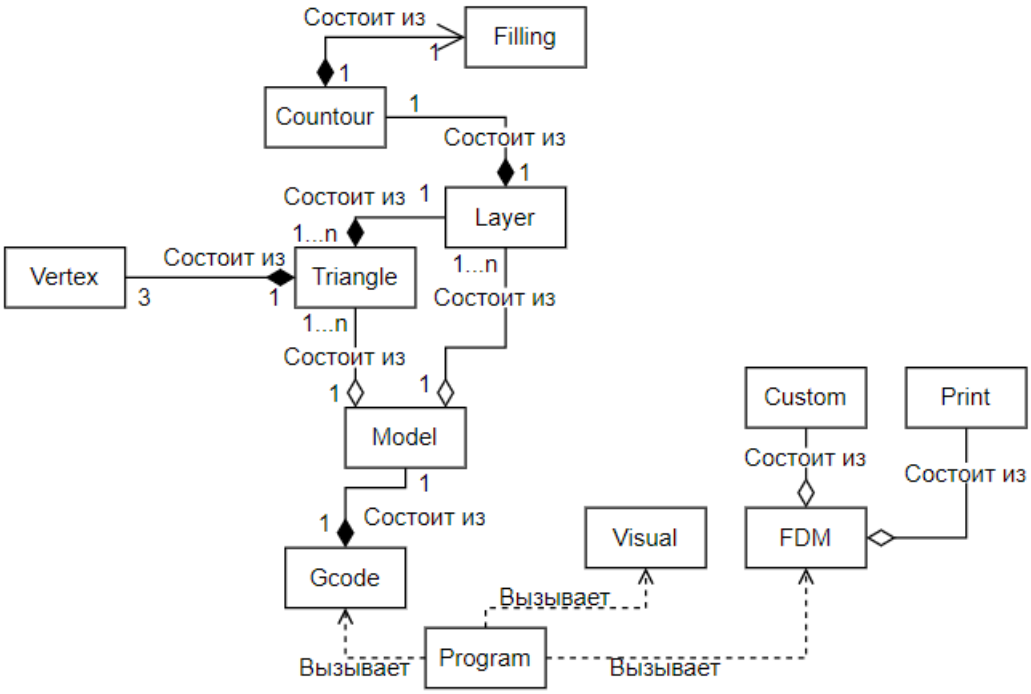

Рисунок 17 – Диаграмма классов

В класс `Program` принимаются указания и данные от пользователя, производится считывание данных из исходного файла, которые в свою очередь распределяются между модулями программного комплекса, в зависимости от поставленной пользователем задачей. Класс `Gcode` занимается формированием управляющего кода из обработанных данных классом `Model`. `Model` содержит списки всех объектов, которые составляют как исходную модель из файла формата Stl, так и полученных после ее обработки.

Класс `Vertex` содержит поля, определяющие вершины треугольников, составляющих исходную модель, по осям Ox, Oy и Oz.

Класс `Triangle` включает в себя объекты класса `Vertex`, хранящие в себе значения вершин и нормали треугольника. Этот класс содержит методы, необходимые для определения среди вершин треугольников, принадлежащих слою, максимального и минимального значения по оси Oz.

В классе `Layer` производятся действия по распределению треугольников по слоям, определению переменной толщины слоя и уменьшения количества слоев.

Формирование контуров модели производится в классе `Layer`, за заполнение которых отвечает класс `Filling`.

Класс `Visual` принимает данные обработанной модели или данные управляющего кода и отвечает за визуализацию 3D-модели на трехмерной сцене.

Класс `FDM` принимает данные управляющего кода в формате G-кода и данные от пользователя для настройки работы FDM 3D-принтера и управления координатным столом и экструдером. Основной задачей данного класса является управление процессом 3D-печати.

На диаграмме состояний, показанной на рисунке 18, отображены основные состояния модели в процессе её обработки от 3D-модели до записи управляющего G-кода.

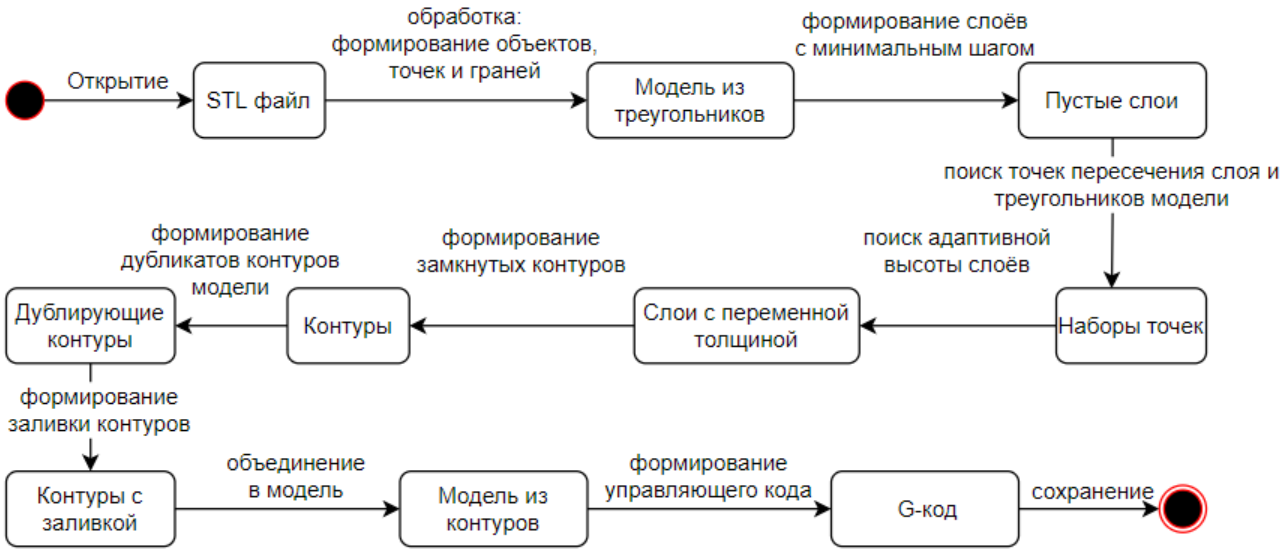

Рисунок 18 – Диаграмма состояний

На данной диаграмме прямоугольниками обозначены полученные результате на каждом этапе. Прямоугольник означает завершение выполнения какой-либо задачи с полученным после ее выполнения результатом. Стрелками обозначены события, которые инициируют переход программы в новое обработанное состояние модели. Таким образом, на диаграмме состояний отражены все ключевые состояния программы в процессе обработки 3D-модели.

Диаграмма вариантов использования показана на рисунке 19. Она отображает возможные действия пользователя при работе с программой.

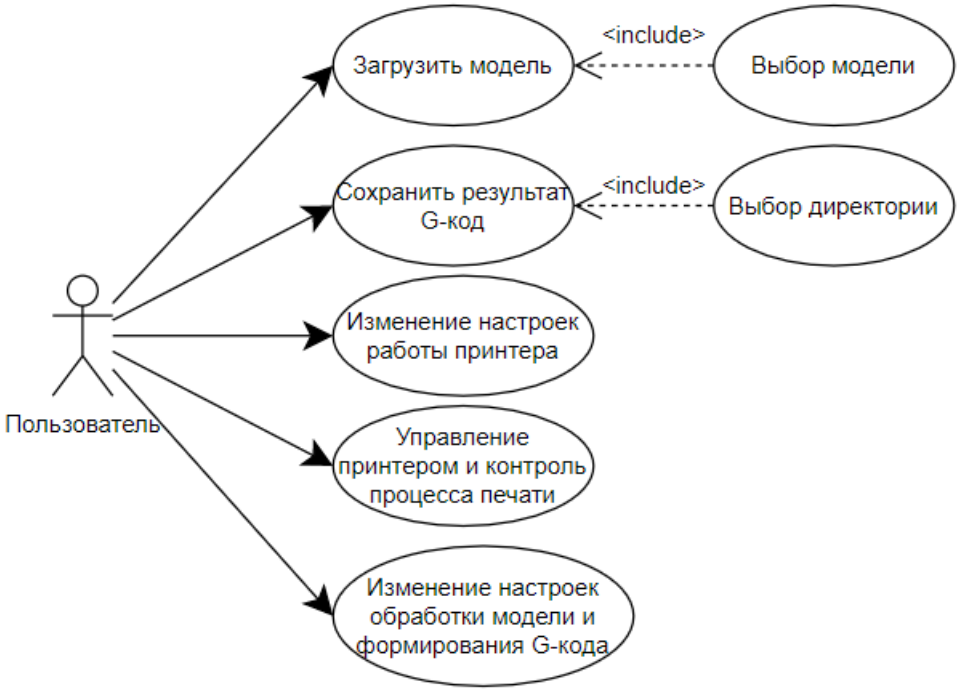

Рисунок 19 – Диаграмма вариантов использования

2.2 Проектирование математической модели программы для обработки 3D-модели и формирования управляющего кода

2.2.1 Алгоритм деления модели на слои. Определение треугольников модели, принадлежащих каждому слою

Прежде всего, перед тем как слайсер будет выполнять свои функции, необходимо получить набор данных о 3D модели из файла STL. Требуется извлечь информацию о каждом фасете, который формирует модель. Для этого производится считывание файла, при котором координаты каждой вершины фасета в определённом порядке заносятся в массив значений. Также извлекаются данные о единичных векторах нормали каждого фасета [20].

После получения необходимых сведений, производится деление модели на слои. При постоянной толщине слоя, пользователь сам определяет толщину, которая ему необходима. Значение толщины слоя варьируется в зависимости от характеристик имеющегося оборудования.

Для определения количества печатаемых слоёв, необходимо найти среди всех вершин фасетов такую вершину, которая будет иметь максимальное значение координаты Z. Это значение определяет высоту модели. Количество слоёв N, на которые разрезается модель, определяется по формуле:

$$N = \frac{Z_{max}}{h}, \quad (1)$$

где Z_{max} – максимальное значение координаты Z, h – постоянная высота слоя, заданная пользователем. Это означает, что наша модель будет состоять из N напечатанных слоёв, каждый из которых будет иметь одинаковую толщину, равную h.

После нахождения N, необходимо определить какие треугольники принадлежат каждому слою. Под принадлежностью следует понимать пересечение треугольником плоскости слоя. Если треугольник имеет с плоскостью общие точки, то он относится к данному слою и необходимо учитывать точки его пе-

ресечения с плоскостью слоя. Для определения принадлежности треугольника к слою необходимо выполнение хотя бы одного из условий системы:

$$\begin{cases}(z_1 - z_0)(z_2 - z_0) \leq 0, \\ (z_1 - z_0)(z_3 - z_0) \leq 0, \\ (z_3 - z_0)(z_2 - z_0) \leq 0, \end{cases} \quad (2)$$

где z_1, z_2, z_3 – значения координат z для вершин треугольника, а z_0 – значение z плоскости слоя. Если координаты треугольника удовлетворяют одному из условий системы (2), то значения его вершин сохраняются.

Существует несколько вариантов пересечения треугольником плоскости слоя [26]. На рисунке 6 показаны варианты таких пересечений.

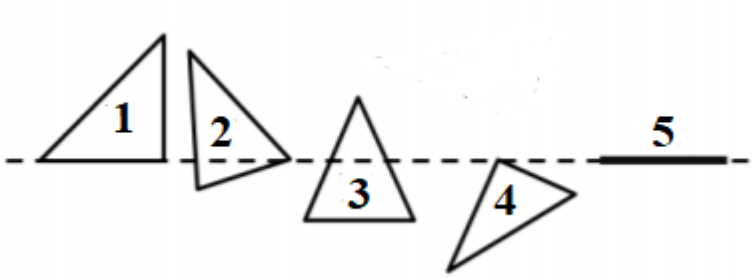

Рисунок 6 – Варианты пересечения треугольником плоскости слоя

В случаях 1, 4, 5 треугольник пересекается с плоскостью одной или несколькими из своих вершин. Это означает, что значение координаты z у вершины и слоя одинаково. В ситуации номер 2, когда пересечение происходит в двух точках, одна из которых – вершина треугольника, а вторая принадлежит грани треугольника и плоскости слоя одновременно. В третьем случае две точки пересечения принадлежат граням треугольника и плоскости одновременно.

Координаты всех точек, которые пересекают слой, необходимо запомнить. В дальнейшем из них формируется контур печатаемого слоя.

2.2.2 Алгоритм нахождения переменной толщины слоя

Для определения адаптивной высоты слоя, будем опираться на значение косинуса угла для каждого слоя [22]. После того, как определились треугольники, которые пересекают слой, необходимо проанализировать значения их векторов нормали. Среди всех треугольников слоя находится минимальное значение косинуса угла между вектором нормали треугольника и осью Oz, которая совпадает с вектором нормали плоскости слоя. Косинус угла рассчитывается по формуле:

$$cos(angle) = \sqrt{1 - \left(\frac{z}{x^2 + y^2 + z^2}\right)^2}, \tag{3}$$

где $cos(angle)$ – косинус угла, x, y, z – координаты вектора нормали треугольника.

Есть 2 случая, когда высота слоя либо минимальна, либо максимальна [26]:

$$\begin{cases} cos(angle) = 1 \rightarrow h = h_{min}, \\ cos(angle) = 0 \rightarrow h = h_{max}, \end{cases} \tag{4}$$

где h – высота слоя, h_{min} – минимальное допустимое значение высоты, h_{max} – максимальное допустимое значение высоты. На рисунке 7 показан случай, когда значение косинуса угла между вектором нормали и осью находится между 0 и 1.

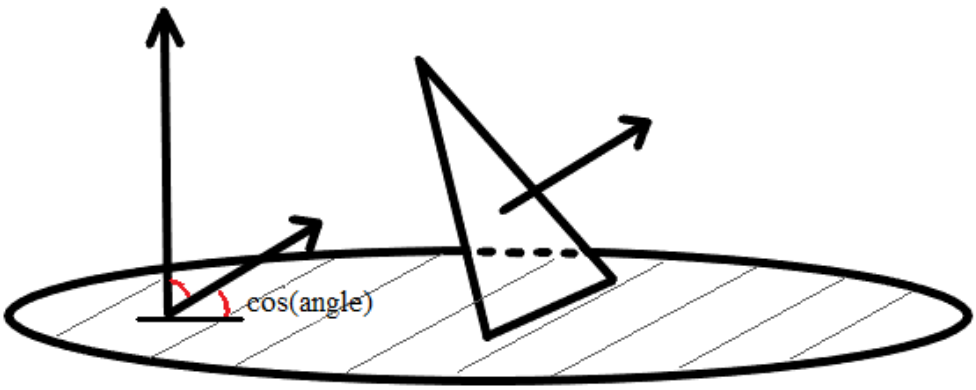

Рисунок 7 – Положение треугольника в зависимости от вектора нормали

Значение толщины слоя h находится в диапазоне:

$$h_{min} \leq h \leq h_{max}. \quad (5)$$

При этом шаг будет равен минимальному допустимому значению:

$$\Delta h_0 = h_{min}, \quad (6)$$

где Δh_0 – шаг.

Пусть переменная x принимает значение минимального вектора нормали:

$$x = cos(angle). \quad (7)$$

Тогда будем находить первичное значение высоты на данном слое по формуле:

$$h(x) = (1 - x)(\, h_{max} - h_{min}) + h_{min}, \quad (8)$$

где $h(x)$ – первичное значение высоты, в зависимости от переменной x. Проверим:

$$h(0) = (1 - 0)(\, h_{max} - h_{min}) + h_{min} \rightarrow h(0) = h_{max}, \quad (9)$$
$$h(1) = (1 - 1)(\, h_{max} - h_{min}) + h_{min} \rightarrow h(1) = h_{min.}$$

Из уравнений (9) видно, что первичное значение высоты соответствует условию (4).

Для нахождения окончательного значения толщины слоя, будем использовать формулу:

$$h^* = int\left(\frac{h_{max}}{h(x)} + 0.5\right) h_{min}, \quad (10)$$

где h^* – окончательное значение толщины, которое принимает значение, кратное шагу (6) и удовлетворяет условию (5). Толщина слоя h будет принимать значение h^*.

По формулам (8) и (10) производится расчёт адаптивной высоты для слоёв, из которых будет состоять модель. Шаг от слоя к слою рассчитывается по следующей формуле:

$$z_{k+1} = z_k + h_k, \quad (11)$$

где z_{k+1} – значение по координате z для следующего слоя, z_k для начала текущего, а h_k – значение высоты, рассчитанное по формуле (10) для текущего слоя.

2.2.3 Построение контуров каждого сформированного слоя модели

Для построения контуров модели в первую очередь необходимо найти уравнение плоскости, на которой установлена модель. Уравнение плоскости:

$$Ax + By + Cz + D = 0, \quad (12)$$

где A, B, C и D – коэффициенты уравнения плоскости.

В этой плоскости лежит точка с координатами $(0,0,0)$, и искомая плоскость перпендикулярна оси Oz, поэтому коэффициенты уравнения плоскости составляют:

$$A = 0, B = 0, C = 1, D = 0. \quad (13)$$

Находим уравнение плоскости для текущего слоя. Она параллельна плоскости основания, но имеет другие координаты по оси Oz, найденные на предыдущем шаге, поэтому коэффициенты равны:

$$Al = 0, Bl = 0, Cl = z, Dl = 0, \quad (14)$$

где Al, Bl, Cl и Dl – коэффициенты уравнения текущей плоскости.

Точки, которые составляют контур, являются точками пересечения сторон каждого треугольника в слое с плоскостью слоя. Для того чтобы найти пересечение сторон треугольника с плоскостью слоя, необходимо найти вершину, которая удовлетворяет одному из условий:

$$\begin{cases} z_1 > z \\ z_2 \leq z \\ z_3 \leq z \end{cases} \text{ или } \begin{cases} z_1 < z \\ z_2 \geq z, \\ z_3 \geq z \end{cases} \quad (15)$$

где z_1, z_2, z_3 – координаты вершин треугольника, z – координата плоскости слоя.

Из вершины с координатой z_1 проводим прямые к двум другим вершинам треугольника. Значения координат вершин известны заранее, а координаты точек граней необходимо найти. На рисунке 8 изображена ситуация, в которой необходимо определить координаты двух точек, лежащих на гранях треугольника.

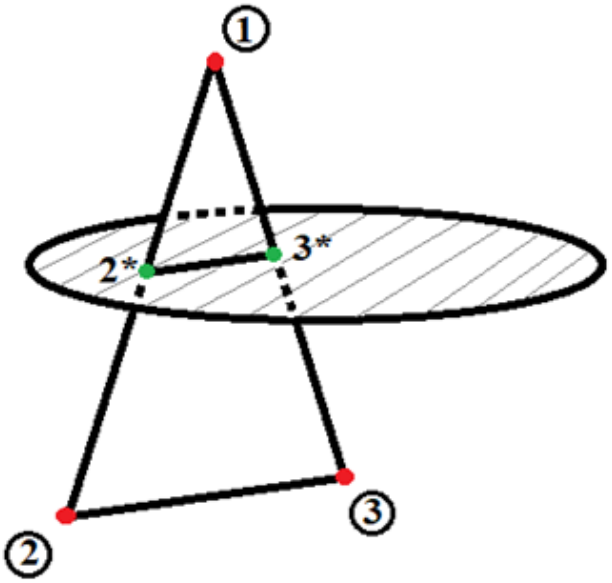

Рисунок 8 – Пример пересечения треугольником плоскости

Для того чтобы найти значения координат точки 2^*, рассматривается грань, на которой она лежит [1]. Необходимо составить уравнение прямой, находящейся между точками 1 и 2. Пусть точка 1 это $M_1(x_1; y_1; z_1)$, а точка 2 это $M_2(x_2; y_2; z_2)$.

Уравнение прямой в пространстве определяется по формуле [16]:

$$\frac{x_2 - x_1}{m} = \frac{y_2 - y_1}{n} = \frac{z_2 - z_1}{p} = \lambda. \quad (16)$$

Для точки M_1 уравнение (16) предстоит в виде:

$$\frac{x - x_1}{m} = \frac{y - y_1}{n} = \frac{z - z_1}{p} = \lambda. \quad (17)$$

Решая совместно уравнения эти уравнения, получим:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1} = \lambda. \quad (18)$$

Уравнение (18) – это уравнение прямой, проходящей через две точки в пространстве [7]. Оно также подходит для уравнения прямой, проходящей через точки M_1 и M_2.

Найдём координаты точки 2^*, принадлежащей плоскости. Определим параметр слоя из уравнения (18):

$$\lambda_0 = \frac{z_0 - z_1}{z_2 - z_1}, \tag{19}$$

где λ_0 – параметр слоя, z_0 – значение z плоскости слоя.

Из уравнения (18) получим параметрическое уравнение прямой в пространстве [20]:

$$\begin{cases} x = x_1 + \lambda(x_2 - x_1), \\ y = y_1 + \lambda(y_2 - y_1), \\ z = z_1 + \lambda(z_2 - z_1). \end{cases} \tag{20}$$

Для точки $2^*(x_i; y_i; z_i)$, подставляя в (20), получим:

$$\begin{cases} x_i = x_1 + \lambda_0(x_2 - x_1), \\ y_i = y_1 + \lambda_0(y_2 - y_1), \\ z_i = z_0. \end{cases} \tag{21}$$

Это уравнения для нахождения координат x_i и y_i в точке 2^*. Значение z_i находить нет необходимости, оно равно значению z_0 слоя. Уравнение вида (21) позволяет определить точки пересечения прямой и плоскости.

Теперь, если вершины с такими координатами не существует в массиве вершин контура, заносим его в массив, если есть, то запоминаем ее индекс в массиве.

Две вершины, которые образованы пересечением двух соответствующих прямых с плоскостью, необходимо соединить в линию, поскольку они принадлежат одному и тому же треугольнику и однозначно должны быть соединены друг с другом. Поскольку каждая вершина в контуре соединена только с двумя другими, необходимо произвести поиск последовательности вершин, которые составляют контур. Такая последовательность и будет являться контуром слоя.

2.2.4 Построение дублирующих контуров на слоях модели

Построение контура внутри имеющегося и повторяющие его очертания, происходит по схеме, изображенной на рисунке.

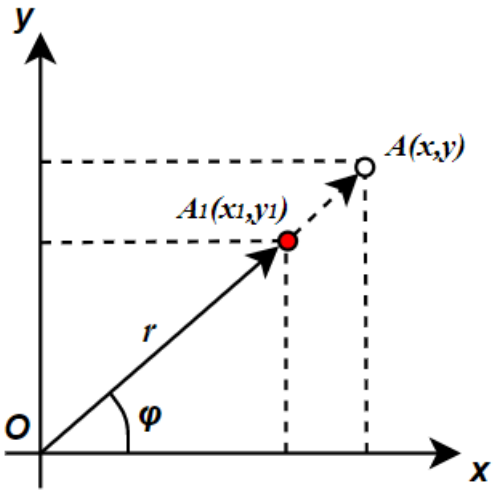

Рисунок 8 – Поиск точки дублирующего контура

Для поиска точек дублирующего контура используется алгоритм перевода координат точек сформированного контура модели из декартовой системы координат в полярную. Необходимо учитывать, что в полярной системе начало координат лежит в точке (0;0). Поэтому изначально необходимо определить точку центра текущего контура на плоскости и изменять значения координат точек контура, сдвигая её к началу полярной системы координат. Поиск центра контура производится путем поиска центра масс системы точек, формирующих данный контур. Центр масс ищется для оси абсцисс и ординат и рассчитывается по формуле (номер). Найденные значения $(x; y)$ будут являться координатами центра текущего контура.

$$center = \frac{\sum_0^N coord}{N}, \quad (19)$$

где $center$ – координата x или y, N – количество точек контура, $coord$ – значение координаты контура.

Значения координат текущей точки сдвигаются к началу системы координат, и для новой точки рассчитывается полярный радиус, который является значением расстояния от начала координат до данной точки и рассчитывается по формуле (номер).

$$r = \sqrt{x^2 + y^2}, \quad (19)$$

где r – полярный радиус, x и y – координаты точки.

Затем производится поиск коэффициента отношения новых координат и старых по формуле (номер).

$$k = \frac{r - H}{r}, \quad (19)$$

где r – полярный радиус, а H – значение, на которое сужается исходный контур.

Последним действием производится умножение координат точки на коэффициент отношения и обратное смещение координат на значения координат центра контура в декартовой системе. Найденные координаты являются значениями точки нового дублированного контура.

2.2.5 Алгоритм заполнения контуров модели

Для заполнения контура модели, первоначально необходимо определить значения экстремумов координат среди точек текущего контура. Эти значения будут являться границами текущего контура по осям абсцисс и ординат. Они необходимы для составления уравнений прямых, пересекающих текущий контур модели и дальнейшего поиска точек пересечения таких прямых с прямыми, формирующими замкнутый контур. Уравнение прямой на плоскости имеет вид:

$$y = kx + b, \quad (22)$$

где x и y – значения координат точек, k – угловой коэффициент, b – значение пересечения прямой с осью ординат.

Коэффициенты k и b рассчитываются по формулам (номер) и (номер) соответственно и необходимы для определения координат точки пересечения прямых на плоскости.

$$k = \frac{y_2 - y_1}{x_2 - x_1}, \tag{22}$$

где x_1 и y_1 – значения координат первой точки, а x_2 и y_2 – координаты второй точки, через которые проходит прямая.

$$b = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}, \tag{22}$$

где x_1 и y_1 – значения координат первой точки, а x_2 и y_2 – координаты второй точки, через которые проходит прямая.

Точка пересечения является координатой, необходимой для дальнейшего заполнения контура слоя. Значения её координат рассчитываются по формулам:

$$\begin{cases} x = \dfrac{b_2 - b_1}{k_1 - k_2}, \\ y = \dfrac{k_1 b_2 - k_2 b_1}{k_1 - k_2}, \end{cases} \tag{22}$$

где x и y – координаты точки пересечения двух прямых, k_1 и b_1 – значения коэффициентов уравнения первой прямой, а k_2 и b_2 второй.

Совокупность найденных точек пересечения формирует облако точек на плоскости, по значениям координат которых будет передвигаться печатающая головка принтера, формируя заполнение контура прямыми линиями. Также необходимо установить порядок точек таким образом, чтобы контур заполнялся последовательно, и не возникло ситуаций пересечения печатаемых прямых. Для этого производится сортировка точек по условию равенства значений координат вдоль оси ординат. Порядок перехода от одной пары точек к другой должен быть таким, чтобы были наименьшие затраты времени перехода печатающей головки от точки к точке. Заполнение контура производится от минимального до максимального значений координат точек вдоль оси ординат, с определенным шагом. Вариант такого передвижения изображен на рисунке

(номер), где пунктирными линиями обозначено перемещение печатающей головки.

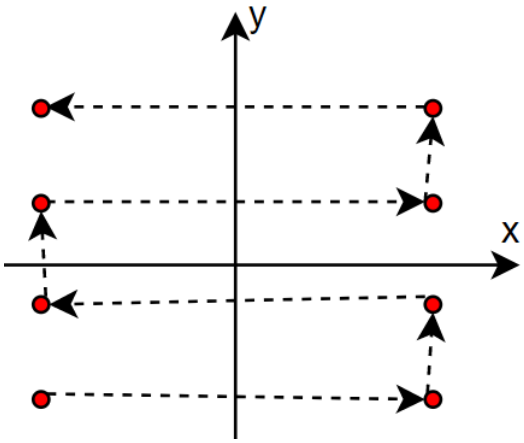

Рисунок 8 – Перемещение печатающей головки в процессе заполнения контура

2.2.6 Расчет количества пластика и построение управляющего кода

G-код – условное название языка программирования, предназначенного для устройств с числовым программным управлением.

Программа, которая написана на таком языке программирования, имеет достаточно жесткое строение. Все элементы управления объединяются в кадры, завершающиеся символом перевода строки. Кадры в свою очередь состоят из одной или более команд. Команды бывают двух типов: непосредственно управляющие движением экструдера и управляющие служебными действиями (запуск и остановка FDM 3D-принтера, включение охлаждения и т.д.) [25].

Традиционно предполагается, что первыми в кадре указываются подготовительные команды, затем команды перемещения, затем выбора режимов обработки и технологические команды [16].

Для построения модели используются команды перемещения, в которых определяются точки, непосредственно к которым производится перемещение экструдера, и количество пластика, подаваемого на экструдер. Выдавливаемый пластик, при движении экструдера от одной точки к другой, формирует линии,

составляющие слои модели. Количество подаваемого пластика производится по формулам:

$$E_i = E_{i-1} + dE, \tag{22}$$

где E_i – длина подаваемого пластика на текущем шаге, а E_{i-1} на предыдущем, а dE – количество подаваемого пластика на текущем шаге и рассчитывается по формуле

$$dE = \frac{hDeL}{\left(\frac{Dp}{2}\right)^2 \pi}, \tag{23}$$

где De – диаметр сопла, Dp – диаметр пластика, h – высота слоя, L – длина печатаемого отрезка, которая рассчитывается по формуле

$$L = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \tag{24}$$

где x_i , y_i – координаты точки, x_{i+1}, y_{i+1} – координаты следующей точки.

3 Реализация программного комплекса для управления процессом 3D-печати. Результаты работы программного комплекса

3.1 Модуль обработки 3D-модели и формирования управляющего кода

3.1.1 Обработка данных о 3D-модели из файла формата STL

Считанные данные из STL-файла передаются в объект класса `Triangle`. Метод `read_file()` (листинг 1) осуществляет побайтовый разбор содержимого исходного файла.

Листинг 1 – Метод `read_file()`

```
FileStream fs = new FileStream(filename, FileMode.Open);
BinaryReader fread = new BinaryReader(fs);
byte[] header = new byte[80];
UInt16 attribute;
Console.WriteLine("Считывание файла");
header = fread.ReadBytes(80);
Console.WriteLine("Заголовок:");
Console.WriteLine(header);
num_tri = fread.ReadUInt32();
Console.WriteLine("\nУказанное количество треугольников:" + num_tri);
  triangles = new Triangle[num_tri];
  float x, y, z;
  Vertex v1, v2, v3, n;
       for (int i = 0; i < num_tri; i++)
       {
           x = fread.ReadSingle();
           y = fread.ReadSingle();
           z = fread.ReadSingle();
           n = new Vertex(x, y, z);
           x = fread.ReadSingle();
           x = (float)Math.Round(x, 2);
           y = fread.ReadSingle();
           y = (float)Math.Round(y, 2);
           z = fread.ReadSingle();
           z = (float)Math.Round(z, 2);
           v1 = new Vertex(x, y, z);
           x = fread.ReadSingle();
           x = (float)Math.Round(x, 2);
           y = fread.ReadSingle();
           y = (float)Math.Round(y, 2);
           z = fread.ReadSingle();
           z = (float)Math.Round(z, 2);
```

```
                v2 = new Vertex(x, y, z);
                x = fread.ReadSingle();
                x = (float)Math.Round(x, 2);
                y = fread.ReadSingle();
                y = (float)Math.Round(y, 2);
                z = fread.ReadSingle();
                z = (float)Math.Round(z, 2);
                v3 = new Vertex(x, y, z);
                triangles[i] = new Triangle(v1, v2, v3, n);
                attribute = fread.ReadUInt16();
            }
        fread.Close();
        fs.Close();
```

3.1.2 Поиск минимального и максимального значений координат по оси аппликат

Минимальное значение по координате z определяется с помощью метода `find_zmin()` (листинг 2) и необходимо для сдвига модели к нулевой плоскости, если она находится выше или ниже её. Максимальное значение определяется при помощи метода `find_zmax()` (листинг 3) и применяется в дальнейшей обработке модели.

Листинг 2 – Метод `find_zmin()`

```
    float zmin = triangles[0].v1.Z;
            for (int i = 0; i < num_tri; i++)
            {
                zmin = (zmin < triangles[i].v1.Z) ? zmin : trian-
gles[i].v1.Z;
                zmin = (zmin < triangles[i].v2.Z) ? zmin : trian-
gles[i].v2.Z;
                zmin = (zmin < triangles[i].v3.Z) ? zmin : trian-
gles[i].v3.Z;
            }
            if (zmin != 0)
            {
                for (int i = 0; i < num_tri; i++)
                {
                    triangles[i].normal.Z -= zmin;
                    triangles[i].v1.Z -= zmin;
                    triangles[i].v2.Z -= zmin;
                    triangles[i].v3.Z -= zmin;
                }
            }
```

Листинг 3 – Метод find_zmax()

```
float zmax = triangles[0].v1.Z;
        for (int i = 0; i < num_tri; i++)
        {
            zmax = (zmax > triangles[i].v1.Z) ? zmax : trian-
gles[i].v1.Z;
            zmax = (zmax > triangles[i].v2.Z) ? zmax : trian-
gles[i].v2.Z;
            zmax = (zmax > triangles[i].v3.Z) ? zmax : trian-
gles[i].v3.Z;
        }
        Console.WriteLine("\nzmax = " + zmax);
        return zmax;
```

Возвращаемым значением является максимальное найденное значение координаты z.

3.1.3 Деление модели на слои и определение треугольников модели, принадлежащих каждому слою

Определение слоёв модели с постоянной толщиной слоя осуществляется при помощи метода `find_triangles()` (листинг 4). Здесь производится расчёт такого количества слоёв, и дальнейшее определение принадлежности треугольников к каждому слою.

Листинг 4 – Метод `find_triangles()`

```
        N = (int)(zmax / h_min) + 1;
    Console.WriteLine("\nПоиск количества слоев N с толщиной h = " +
h_min.ToString() + "\nN = " + N.ToString());
    h_triangles = new Triangle[num_tri];
    layers = new Layer[N];
    float z0 = 0;
        for (int i = 0; i < N; i++)
        {
            h_triangles = Triangle.find_layer_triangles(triangles,
num_tri, z0);
            layers[i] = new Layer(h_triangles, 0, 0, 0);
            layers[i].Z = z0;
            if (z0 < zmax) { z0 += h_min; }}
```

3.1.4 Поиск синуса угла для каждого слоя

Определение синуса угла между треугольником и осью производится в методе `angle_btwnTri()` (листинг 5) и используется для дальнейшего нахождения косинуса.

Листинг 5 – Метод `angle_btwnTri()`

```
float angle;
float angle_normal;
       for (int i = 0; i < N; i++)
       {
           angle_normal = 1.0f;
           for (int j = 0; j < num_tri; j++)
           {
               if (layers[i].h2_tri[j] != null)
               {
                   angle    =    (layers[i].h2_tri[j].Normal.Z)    /
(float)(Math.Sqrt(Math.Pow(layers[i].h2_tri[j].Normal.X,       2)       +
Math.Pow(layers[i].h2_tri[j].Normal.Y,                 2)                +
Math.Pow(layers[i].h2_tri[j].Normal.Z, 2)));
                   angle_normal = (angle_normal >= angle) ? angle :
angle_normal;
               }
           }
           Console.WriteLine("\nСинус угла между треугольником и
осью: " + angle_normal.ToString());
           layers[i].angle = angle_normal;
       }
```

3.1.5 Вычисление значения высоты для каждого слоя

В листинге 6 представлен метод `h_layers()` для определения адаптивной высоты слоёв. Также здесь производится подсчёт окончательного количества слоёв модели.

Листинг 6 – Метод `h_layers()`

```
float first_h;
int zk;
       for (int i = 0; i < N; i += zk)
```

```
            {
                layers[i].angle = (float)Math.Sqrt(1 - layers[i].angle *
layers[i].angle);
                first_h = (1 - layers[i].angle) * (h_max - h_min) +
h_min;
                Console.WriteLine("\nПервичное значение высоты  слоя = "
+ first_h.ToString());
                layers[i].h = (float)Math.Floor((h_max / first_h) + 0.5)
* h_min;
                layers[i].h = (layers[i].h > h_max) ? h_max : lay-
ers[i].h;
                Console.WriteLine("\nОкончательное значение высоты  слоя
= " + layers[i].h.ToString());
                zk = (int)(layers[i].h / h_min);
                Console.WriteLine("\nШаг до следующего слоя = " +
zk.ToString());
            }
for (int i = 0; i < N; i++)
            {
                if (layers[i].h == 0)
                {
                    for (int j = i; j < N - 1; j++)
                    {
                        layers[j] = layers[j + 1];
                    }
                    layers[N - 1] = null;
                    N--;
                    i--;
                }
            }
            Console.WriteLine("\nКонечное количество слоев = " +
N.ToString());
            return N;
```

Возвращаемым значением является N – конечное количество слоёв модели.

3.1.6 Построение контуров для каждого сформированного слоя модели

В листинге 7 представлена часть метода `contour_layer()`, который отвечает за формирование контуров модели. В данном методе производится поиск точек, в которых треугольник пересекается с плоскостью слоя, и расчёт их координат по осям Ox и Oy, удаление замкнутых линий и определение индексов вершин, принадлежащих слою. Полный код приведён в листинге А.3.

Листинг 7 – Метод `contour_layer()`

```
    float lambda = (layers[i].z - v1.Z) / (v2.Z - v1.Z);
                        float z1 = layers[i].z;
                        float x1 = v1.X + lambda * (v2.X - v1.X);
                        float y1 = v1.Y + lambda * (v2.Y - v1.Y);
                        vc[0]  =  new  Vertex((float)Math.Round(x1,  3),
(float)Math.Round(y1, 3), (float)Math.Round(z1, 3));
                        //vc[0] = new vertex(x1, y1, z1);
                        int k = 0;
                        if (count == 20)
                            k = 0;
                        for (int v = 0; v < count; v++)
                        {
                            if (V[v] != null)
                            {
                                if  (V[v].X  ==  vc[0].X  &&  V[v].Y  ==
vc[0].Y && V[v].Z == vc[0].Z)
                                    k = 1;
                            }
                        }
                        if (k == 0)
                        {
                            V[count]  =  new  Vertex(vc[0].X,  vc[0].Y,
vc[0].Z);
                            count++;
                        }
                        lambda = (layers[i].z - v1.Z) / (v3.Z - v1.Z);
                        float z2 = z1;
                        float x2 = v1.X + lambda * (v3.X - v1.X);
                        float y2 = v1.Y + lambda * (v3.Y - v1.Y);
                        vc[1]  =  new  Vertex((float)Math.Round(x2,  3),
(float)Math.Round(y2, 3), (float)Math.Round(z2, 3));
                        // vc[1] = new vertex(x2, y2, z2);
                        k = 0;
                        for (int v = 0; v < count; v++)
                            if (V[v].X == vc[1].X && V[v].Y == vc[1].Y &&
V[v].Z == vc[1].Z)
                                k = 1;
                        if (k == 0)
                        {
                            V[count]  =  new  Vertex(vc[1].X,  vc[1].Y,
vc[1].Z);
                            count++;
                        }
                        int line0 = 0, line1 = 0;
                        for (int v = 0; v < count; v++)
                        {
                            if (V[v].X == vc[0].X && V[v].Y == vc[0].Y &&
V[v].Z == vc[0].Z)
                            { line0 = v; }
                            if (V[v].X == vc[1].X && V[v].Y == vc[1].Y &&
V[v].Z == vc[1].Z)
                            { line1 = v; }
                        }
```

```
                    Lines[L_count, 0] = line0;
                    Lines[L_count, 1] = line1;
                    L_count++;
```

3.1.7 Построение дублирующих контуров на слоях модели

3.1.8 Заполнение сформированных контуров модели

3.1.9 Расчет количества пластика и построение управляющего кода

Метод `gcode()`, часть которого представлена в листинге 8, предназначен для записи в файл результатов обработанной модели на языке G-кода. Здесь производится расчёт количества пластика, для каждого участка печати и последовательная запись слоёв модели в файл. Полный код приведен в листинге А.4.

Листинг 8 – Метод `gcode()`

```
 for (int i = 0; i < N; i++)
{
    Console.WriteLine("{0:0.00}", 1.0);
```

```
        sw.Write("G1 F1600\n");
                for (int j = 0; j < layers[i].Num_v_layer - 1; j++)
                {
                    L = (float)Math.Sqrt(Math.Pow(layers[i].V[j + 1].X -
layers[i].V[j].X, 2) + Math.Pow(layers[i].V[j + 1].Y - layers[i].V[j].Y,
2));
                    E = E + (float)((layers[i].H * De * L) / (Math.Pow(Dp
/ 2, 2) * Math.PI));
                    sw.Write("G1   X"   +   (layers[i].V[j].X   /   2   +
100).ToString("f3").Replace(",", ".") + " Y" + (layers[i].V[j].Y / 2 +
100).ToString("f3").Replace(",",       ".")      +      "      E"      +
E.ToString("f5").Replace(",", ".") + "\n");
                }
                L  =  (float)Math.Sqrt(Math.Pow(layers[i].V[0].X  -  lay-
ers[i].V[layers[i].Num_v_layer - 1].X, 2) + Math.Pow(layers[i].V[0].Y -
layers[i].V[layers[i].Num_v_layer - 1].Y, 2));
                E = E + (float)((layers[i].H * De * L) / (Math.Pow(Dp /
2, 2) * Math.PI));
                sw.Write("G1  X" + (layers[i].V[layers[i].Num_v_layer  -
1].X  /  2  +  100).ToString("f3").Replace(",",  ".")  +  "  Y"  +  (lay-
ers[i].V[layers[i].Num_v_layer       -       1].Y       /       2       +
100).ToString("f3").Replace(",",       ".")      +      "      E"      +
E.ToString("f5").Replace(",", ".") + "\n");
                if (layers[i + 1] != null)
                {
                    h = h + layers[i + 1].H;
                }
                sw.Write("G0 Z" + (h).ToString("f3").Replace(",", ".") +
" F7800.000\n");  }
```

3.2 Модуль графического интерфейса программного комплекса

3.2.1 Визуализация 3D-модели, обрабатываемой программным комплексом

3.2.2 Взаимодействие компонент программного комплекса

3.3 Тестирование и результаты работы программного комплекса

3.3.1 Тестирование работы модуля обработки 3D-модели и формирования управляющего кода. Результаты обработки

3.3.2 Тестирование работы модуля графического интерфейса программного комплекса. Результаты взаимодействия компонент программного комплекса

Литература

1. Белоусова И.Д., Информационный менеджмент как новая методология построения системы управления информацией / И.Д. Белоусова. // Современные научные исследования и инновации. – 2014. – № 9–1 (41) – с. 12–15.
2. Бистерфельд О.А., Методология функционального моделирования IDEF0: учебно-методическое пособие / О.А. Бистерфельд – Ряз. гос. ун-т им. С.А. Есенина. — Рязань, 2008. — 48 с.
3. Бортаковский, А.С. Аналитическая геометрия в примерах и задачах: Учеб. пособие / А.С. Бортаковский, А.В. Пантелеев. – М.: Высш. шк., 2005. – 496 с.
4. Большаков, В.П. Основы 3D-моделирования [Текст] / В.П. Большаков, А.Л. Бочков, А.А. Сергеев. – СПб.: Питер, 2010. – 100 с.
5. Васильева А. О., ЧТО ТАКОЕ ДИАГРАММА ГАНТА? / А.О. Васильева, Е.О. Васильева // РЕДКОЛЛЕГИЯ. – 2018. –168 с.
6. Гельфанд, И.М. Лекции по линейной алгебре [Текст] / И.М. Гельфанд. – М.: Добросвет, МЦНМО, 1998. – 319 с.
7. Добрынин А.С., Формирование расписаний в задачах временного планирования / А.С. Добрынин, С.М. Кулаков, Р.С. Койнов, А.В. Грачёв // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. – 2014. – №. 4 – с. 103-109.
8. Ехлаков Ю.П., Управление программными проектами: учебник / Ю.П. Ехлаков. – Томск: Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2015. – 216 с.
9. Ильин, В.А. Аналитическая геометрия [Текст] / В.А. Ильин, Э.Г. Позняк. – М.: ФИЗМАТЛИТ, 2002. – 240 с.

10. Кадыров, Ф.М. Программное обеспечение FDM 3D-принтеров / Ф.М. Кадыров. // Проблемы и возможности современной науки. Сборник материалов IV Международной научно-практической конференции. – М.: Издательство Перо, 2015. – №2(73) – 205 с.
11. Канатников, А.Н. Аналитическая геометрия [Текст] / А.Н. Канатников, А.П. Крищенко. – М.: Издательство МГТУ им. Н.Э. Баумана, 2002.– 388 с.
12. Копп А.М., Разработка подхода к анализу и оптимизации диаграмм потоков данных / А.М. Копп, Д.Л. Орловский. // ScienceRise - № 7, 2017. – 33-42 с.
13. Липаев В.В., Программная инженерия сложных заказных программных продуктов: Учебное пособие [Текст] / В.В. Липаев. – М.: МАКС Пресс, 2014. – 312 с.
14. Мальцев, А.И. Основы линейной алгебры / А.И. Мальцев. – М.: Наука, 1970. – 400 с.
15. Овчаров, А.Э. Возможности и преимущества 3D печати / А.Э. Овчаров, П.В. Косовских, В.И. Гончаров. – Томск ТПУ, 2013. – 439-442 с.
16. Попова И.В., Разработка приложений: учеб. пособие. [Текст] / И.В. Попова – Магнитогорск: МаГУ, 2005. – 186 с.
17. Слюсар, В.И. Фаббер-технологии. Новое средство трехмерного моделирования [Текст]/ В.И. Слюсар // Электроника: наука, технология, бизнес. – Рекламно-издательский центр "ТЕХНОСФЕРА", 2003. – № 5 – 54-60 с.
18. Слюсар, В.И. Фабрика в каждый дом / В.И. Слюсар. // Вокруг света. – Январь, 2008. – № 1 (2808) [Текст].– 96-102 с.
19. Холодилов, А.А. Проблемы возникающие при трехмерной печати с использованием технологии FDM / А.А. Холодилов. // Материалы Международной (заочной) научно-практической конференции «Наука, образование, инновации: апробация результатов исследований». – Научноиздательский центр "Мир науки", 2017. – 199-204 с.

20. Черемных С.В., Моделирование и анализ систем. IDEF-технологии: Практикум / С.В. Черемных, И.О. Семенов, С.В. Ручкин. — М. : Финансы и статистика, 2006 – 196 с.
21. Шафаревич, И.Р. Линейная алгебра и геометрия [Текст] / И.Р. Шафаревич, А.О. Ремизов. – М.: Физматлит, 2009. – 511 с.
22. Barnatt, C. 3d printing the next industrial revolution / C. Barnatt. – CreateSpace, 2013. – 276 p.
23. Beltur B. R. Adaptive Slicing in Additive Manufacturing using Strip Tree Data Structures : дис. – University of Cincinnati, 2016. – 79 p.
24. Brown A. C., Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3-D printer / A. C. Brown, De D. Beer // 2013 Africon. – IEEE, 2013. – С. 1-5.
25. Choi S. H., A memory efficient slicing algorithm for large STL files / S. H. Choi, K. T. Kwok // 1999 International Solid Freeform Fabrication Symposium. – The University of Hong Kong, 1999. – 155-162 p.
26. Chua, C.K. Rapid Prototyping: Principles and Applications. Singapore: World Scientific [Текст] / C.K. Chua, K.F. Leong, C.S. Lim. – World Scientific Publishing Company, 2003. – 124 p.
27. Eragubi M. Slicing 3D CAD model in STL format and laser path generation / M. Eragubi // International journal of innovation, management and technology [Текст]. – 2013. – Т. 4. – №. 4. – 410 p.
28. Gibson I., Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing, Second Edition / I. Gibson, D.W. Rosen, B. Stucker [Текст]. – New York: Springer-Verlag, 2015. – 498 p.
29. Hamzah, H.H. 3D printable conductive materials for the fabrication of electrochemical sensors: A mini review / H. H. Hamzah, A. S. Saiful, A. Aya, B.A. Patel // Electrochemistry Communications. – Elsevier, 2018. – № 96 – 27-31 p.
30. Horvath J., Mastering 3D printing. / J. Horvath, R. Cameron – Berkeley, CA : Apress, 2014. – С. 85.

31. Hu J. Study on STL-based slicing process for 3D printing / J. Hu // Solid Freeform. – 2017. – 885-895 p.
32. Minetto R. et al. An optimal algorithm for 3D triangle mesh slicing / R. Minetto //Computer-Aided Design [Текст]. – Universiti Malaysia Pahang, 2017. – T. 92. – 23 p.
33. Pandey P. M., Slicing procedures in layered manufacturing: a review / P. M. Pandey, N. V. Reddy, S. G. Dhande // Rapid prototyping journal [Текст]. – Indian Institute of Technology Kanpur, 2003. – 1-15 p.
34. Shi K. et al. Slicing and support structure generation for 3D printing directly on B-rep models / K. Shi //Visual Computing for Industry, Biomedicine, and Art. – 2019. – T. 2. – №. 1. – 10 p.
35. Siraskar N., Adaptive slicing in additive manufacturing process using a modified boundary octree data structure / N. Siraskar, R. Paul, S. Anand // Journal of Manufacturing Science and Engineering. – the University of Cincinnati, 2015. – T. 137. – №. 1. – 89 p.
36. Topçu O., A method for slicing CAD models in binary STL format / O. Topçu, Y. Taşcıoğlu, H. Ö. Ünver // 6th International Advanced Technologies Symposium (IATS'11). – TOBB University of Economics and Technology, Ankara, Turkey, 2011. – T. 163. – 141-145 p.

5. Характеристика консультаций у специалистов по теме выпускной квалификационной работы (бакалаврской работы)

С кем консультировались, в чем помогли

Что сказали изменить, дополнить

Здесь вы можете указывать преподавателей кафедры либо специалистов из сторонних организаций (с указанием должности и ФИО), НО ТОЛЬКО НЕ СВОЕГО НАУЧНОГО РУКОВОДИТЕЛЯ!

ЗАКЛЮЧЕНИЕ
о прохождении производственной практики, преддипломной

По итогам прохождения производственной практики, преддипломной, заслушан отчет студента Фамилия И.О. группы ПРИ-171.

Принято решение поставить оценку _____ студенту Фамилия И.О. группы ПРИ-171 по производственной практике, преддипломной.

Утверждено на заседании кафедры ИСКМ,
протокол № ___ от «____»_____20____г.
зав. каф. ИСКМ _____ А.В. Хоперсков
(подпись)

Отзыв руководителя практики от университета

Ф.И.О. (студента) во время весеннего семестра в период с _____ по _____ года проходила производственную практику, преддипломную на кафедре Информационных систем и компьютерного моделирования Волгоградского государственного университета.

Во время производственной практики, преддипломной И.О. ознакомилась ...

По результатам производственной практики, преддипломной рекомендуется оценить работу Ф.И.О. как _____.

Руководитель практики от университета:

к.ф.-м.н., доцент каф. ИСКМ, Ф.И.О. _____

(подпись)