АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: Храпов Сергей Сергеевич khrapov@volsu.ru / ID: 369
Проверяющий: Храпов Сергей Сергеевич (khrapov@volsu.ru / ID: 369)
Организация: Волгоградский государственный университет

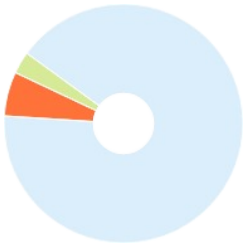
Отчет предоставлен сервисом «Антиплагиат» - http://volsu.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 248
Начало загрузки: 11.06.2021 22:33:22
Длительность загрузки: 00:00:24
Имя исходного файла: ДляПлагиата_Диплом_ПестряковПА_ПРИ-171.docx
Название документа: ДляПлагиата_Диплом_ПестряковПА_ПРИ-171
Размер текста: 1 кБ
Символов в тексте: 78691
Слов в тексте: 10063
Число предложений: 517

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 11.06.2021 22:33:47
Длительность проверки: 00:00:22
Комментарии: не указано
Поиск с учетом редактирования: да
Модули поиска: ИПС Адилет, Библиография, Сводная коллекция ЭБС, Интернет Плюс, Сводная коллекция РГБ, Цитирование, Переводные заимствования (RuEn), Переводные заимствования по eLIBRARY.RU (EnRu), Переводные заимствования по Интернету (EnRu), Переводные заимствования издательства Wiley (RuEn), eLIBRARY.RU, СПС ГАРАНТ, Медицина, Диссертации НББ, Перефразирования по eLIBRARY.RU, Перефразирования по Интернету, Патенты СССР, РФ, СНГ, Шаблонные фразы, Модуль поиска "volsu", Кольцо вузов, Издательство Wiley, Переводные заимствования



ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
5,65%	0%	3,03%	91,32%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Доля в тексте	Источник	Актуален на	Модуль поиска	Блоков в отчете	Блоков в тексте	Комментарии
[01]	0,17%	2,88%	ВПМ http://rsreu.ru	11 Июн 2021	Интернет Плюс	5	14	
[02]	0,11%	2,86%	ОГУ - Аннотации рабочих программ дисциплин http://osu.ru	28 Мая 2020	Интернет Плюс	6	13	
[03]	0,02%	2,81%	https://www.volpi.ru/files/documents/doc_educational/sveden/ann_09.03.04_2019.pdf https://volpi.ru	11 Ноя 2020	Интернет Плюс	1	13	
[04]	2,72%	2,72%	Проект Приказа Министерства образования и науки РФ "Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.04 Программная инженерия (уровень бакалавриата)" (подготовлен Минобрнауки Рос... http://ivo.garant.ru	01 Мар 2018	СПС ГАРАНТ	15	15	
[05]	2,55%	2,59%	ОСОБЕННОСТИ РАЗРАБОТКИ И ПРИМЕНЕНИЯ FDM-ТЕХНОЛОГИИ ПРИ СОЗДАНИИ И ПРОТОТИПИРОВАНИИ 3D-ОБЪЕКТОВ. http://elibrary.ru	27 Мая 2019	Перефразирования по eLIBRARY.RU	3	2	
[06]	0%	2,52%	https://math.spbu.ru/ru/mmeh/PLANS/1/19_5080_090304bPrIng_19_06_04.pdf https://math.spbu.ru	11 Ноя 2020	Интернет Плюс	0	12	
[07]	0%	2,34%	Поступление https://dvfu.ru	20 Янв 2021	Интернет Плюс	0	12	
[08]	0%	2,11%	Проект Приказа Министерства образования и науки РФ "Об утверждении федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.03.03	01 Мар 2018	СПС ГАРАНТ	0	13	

			Прикладная информатика (уровень бакалавриата)" (подготовлен Минобрнауки Ро... http://ivo.garant.ru				
[09]	0%	2,11%	http://bgitu.ru/sveden/education/oop/%D0%9E%D0%9F%D0%9E%D0%9F_09.03.01_%D0%98%D0%92%D0%A2-2019+.pdf http://bgitu.ru	26 Окт 2020	Интернет Плюс	0	11
[10]	0,01%	2,1%	Ссылка http://nirhtu.ru	24 Окт 2018	Интернет Плюс	1	11
[11]	0%	2,03%	http://emirs.miet.ru/oroks-miet/upload/ftp/pub/orioks3/2019/2/Opisanie_09.03.01_ochno-zaochnaya.pdf http://emirs.miet.ru	01 Мар 2019	Интернет Плюс	0	12
[12]	0,11%	1,85%	ОСОБЕННОСТИ РАЗРАБОТКИ И ПРИМЕНЕНИЯ FDM-ТЕХНОЛОГИИ ПРИ СОЗДАНИИ И ПРОТОТИПИРОВАНИИ 3D-ОБЪЕКТОВ. http://elibrary.ru	27 Мая 2019	eLIBRARY.RU	2	20
[13]	0%	1,72%	Образовательный потенциал технологий быстрого прототипирования. http://elibrary.ru	04 Авг 2016	Перефразирования по eLIBRARY.RU	0	1
[14]	0%	1,72%	Заседатель В.С. Образовательный потенциал технологий быстрого прототипирования http://naukovedenie.ru	30 Янв 2017	Перефразирования по Интернету	0	1
[15]	0%	1,63%	Данилова, Любовь Филипповна Методика построения и многопараметрической оптимизации компетентностной модели профессиональной образовательной программы : диссертация ... кандидата технических наук : 05.13.10 Новосибирск 2019 http://dlib.rsl.ru	05 Авг 2019	Сводная коллекция РГБ	0	8
[16]	0%	1,32%	Компьютерное моделирование и виртуальный эксперимент как средство формирования компетенций в процессе преподавания физики. http://elibrary.ru	18 Окт 2019	eLIBRARY.RU	0	7
[17]	0,39%	1,29%	Особенности преподавания математики на компьютерных направлениях. http://elibrary.ru	14 Янв 2020	Перефразирования по eLIBRARY.RU	2	3
[18]	0,03%	1,26%	Куракина, Ирина Игоревна Содержание обучения теории и истории традиционного прикладного искусства в профильном высшем образовании : диссертация ... кандидата педагогических наук : 13.00.08 Санкт-Петербург 2018 http://dlib.rsl.ru	22 Фев 2019	Сводная коллекция РГБ	1	6
[19]	0%	1,21%	ФОРМИРОВАНИЕ УНИВЕРСАЛЬНЫХ, ОБЩЕПРОФЕССИОНАЛЬНЫХ И ПРОФЕССИОНАЛЬНЫХ КОМПЕТЕНЦИЙ СТУДЕНТОВ БАКАЛАВРИАТА ПО НАПРАВЛЕНИЮ ПОДГОТОВКИ СТРОИТЕЛЬСТВО. http://elibrary.ru	16 Июл 2018	eLIBRARY.RU	0	8
[20]	0%	1,21%	Паспорт образовательной программы — Образовательная программа «Бизнес-информатика» — Национальный исследовательский университет «Высшая школа экономики» https://perm.hse.ru	08 Июн 2021	Интернет Плюс	0	6
[21]	0%	1,08%	Моделирование влияния подвижности узлов на связность сетей FANET	26 Июн 2020	Кольцо вузов	0	6
[22]	0%	1,04%	Интеграция курсов "1С" в программу подготовки бакалавров специальности "Информационные системы и технологии". http://elibrary.ru	19 Мар 2020	eLIBRARY.RU	0	7
[23]	0%	1,01%	http://uup.samgtu.ru/sites/uup.samgtu.ru/files/pr-130301-atpipvteitt.doc http://uup.samgtu.ru	13 Апр 2021	Интернет Плюс	0	6
[24]	0%	0,94%	Рюгина А А	10 Июн 2019	Кольцо вузов	0	5
[25]	0%	0,94%	science-education.ru_file_5d565cedbe73f.docx	16 Авг 2019	Кольцо вузов	0	5
[26]	0%	0,94%	science-education.ru_file_5de73bcf01d52.docx	04 Дек 2019	Кольцо вузов	0	5
[27]	0,01%	0,9%	260752 http://biblioclub.ru	19 Апр 2016	Сводная коллекция ЭБС	1	3
			Образовательный потенциал				

[28]	0%	0,9%	технологий быстрого прототипирования. http://elibrary.ru	04 Авг 2016	eLIBRARY.RU	0	10
[29]	0%	0,89%	Особенности преподавания математики на компьютерных направлениях. http://elibrary.ru	14 Янв 2020	eLIBRARY.RU	1	5
[30]	0%	0,88%	Заседатель В.С. Образовательный потенциал технологий быстрого прототипирования http://naukovedenie.ru	раньше 2011	Интернет Плюс	0	9
[31]	0%	0,88%	Есин, Роман Витальевич Формирование математической компетентности бакалавров направления подготовки «Информатика и вычислительная техника» в электронной среде : диссертация ... кандидата педагогических наук : 13.00.02 Красноярск 2019 http://dlib.rsl.ru	15 Окт 2019	Сводная коллекция РГБ	0	4
[32]	0%	0,78%	science-education.ru_file_5d5a8c62a975a.docx	19 Авг 2019	Кольцо вузов	0	4
[33]	0%	0,76%	Diplom_Golshtejn.docx	06 Июн 2019	Кольцо вузов	0	4
[34]	0%	0,76%	diss-Mikhaylov-909.pdf	04 Мар 2019	Кольцо вузов	0	4
[35]	0%	0,76%	expeducation.ru_file_5d81c4f17572c.docx	18 Сен 2019	Кольцо вузов	0	4
[36]	0%	0,76%	Standart_PO_10.10.2019	25 Ноя 2019	Кольцо вузов	0	4
[37]	0%	0,76%	Елена Нестерова на печать.docx	26 Июл 2018	Кольцо вузов	0	4
[38]	0%	0,72%	260407 http://biblioclub.ru	19 Апр 2016	Сводная коллекция ЭБС	0	2
[39]	0%	0,71%	https://educonf.1c.ru/conf2020/%D0%A2%D0%BE%D0%BC1.pdf https://educonf.1c.ru	04 Мая 2020	Интернет Плюс	0	4
[40]	0,65%	0,65%	Моделирование бизнес-процессов-теория http://studfiles.ru	30 Янв 2017	Перефразирования по Интернету	1	1
[41]	0,12%	0,64%	http://fit.nsu.ru/data_/docs/bak/OOP/4_RPD/09.03.01.1/_09.03.01.1_B16_rpd.pdf http://fit.nsu.ru	06 Апр 2021	Интернет Плюс	1	3
[42]	0%	0,63%	не указано http://inion.ru	30 Янв 2017	Перефразирования по Интернету	0	1
[43]	0%	0,63%	PDF http://science-education.ru	08 Янв 2017	Перефразирования по Интернету	0	1
[44]	0%	0,6%	Инструментальные методы и программные средства в экономике: учебное пособие https://e.lanbook.com	22 Янв 2020	Сводная коллекция ЭБС	0	1
[45]	0%	0,6%	Черемных С. В., Семенов И. О., Ручкин В. С. Моделирование и анализ систем. IDEF-технологии: практикум Москва 2006 http://dlib.rsl.ru	12 Июл 2017	Сводная коллекция РГБ	0	1
[46]	0,04%	0,58%	Подготовка и сдача государственного экзамена_Защита выпускной квалификационной работы, включая подготовку к защите и процедуру защиты http://sssu.ru	11 Июн 2021	Интернет Плюс	2	3
[47]	0%	0,58%	СТРУКТУРНО-СОДЕРЖАТЕЛЬНАЯ МОДЕЛЬ МАТЕМАТИЧЕСКОЙ КОМПЕТЕНТНОСТИ БАКАЛАВРОВ ИНФОРМАЦИОННО-ТЕХНОЛОГИЧЕСКИХ НАПРАВЛЕНИЙ ПОДГОТОВКИ. http://elibrary.ru	01 Фев 2021	eLIBRARY.RU	0	3
[48]	0%	0,54%	ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ УПРАВЛЕНИЯ. Учебное пособие для бакалавров.pdf	21 Фев 2017	Сводная коллекция ЭБС	0	1
[49]	0%	0,54%	Моделирование бизнес-процессов-теория http://studfiles.ru	29 Июл 2016	Интернет Плюс	0	1
[50]	0%	0,54%	ИНСТРУМЕНТАЛЬНЫЕ МЕТОДЫ ЭКОНОМИКИ, МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЕ IDEF0, СИНТАКСИС И СЕМАНТИКА МОДЕЛЕЙ IDEF0 - Инструментальные методы и программные средства в экономике https://studref.com	19 Янв 2021	Интернет Плюс	0	1
			ИНСТРУМЕНТАЛЬНЫЕ МЕТОДЫ ЭКОНОМИКИ, МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО				

[51]	0%	0,54%	МОДЕЛИРОВАНИЕ IDEF0, СИНТАКСИС И СЕМАНТИКА МОДЕЛЕЙ IDEF0 - Инструментальные методы и программные средства в экономике https://studref.com	19 Янв 2021	Интернет Плюс	0	1
[52]	0%	0,48%	http://siu.ranepa.ru/Programs/2017/bak/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%20%D0%93%D0%98%D0%90_%D0%AD%D0%BA_%20%D0%9D%D0%B8%D0%9D_%D0%B1%D0%B0%D0%BA.pdf http://siu.ranepa.ru	13 Фев 2019	Интернет Плюс	0	4
[53]	0%	0,46%	Павлуцкая, Нина Максимовна Дифференциация обучения физике бакалавров технических направлений подготовки как условие формирования их общекультурных и общепрофессиональных компетенций : диссертация ... доктора педагогических наук : 13.00.02 Москва 2016 http://dlib.rsl.ru	27 Дек 2019	Сводная коллекция РГБ	0	2
[54]	0%	0,42%	ОЦЕНКА ЭФФЕКТИВНОСТИ УПРАВЛЕНИЯ ЛОГИСТИЧЕСКИМИ ПРОЦЕССАМИ В ОБЕСПЕЧЕНИИ КАЧЕСТВА ЦЕЛЛЮЛОЗНО-БУМАЖНОЙ ПРОДУКЦИИ. http://elibrary.ru	05 Авг 2016	Перефразирования по eLIBRARY.RU	0	1
[55]	0%	0,42%	Обзор технологий 3D печати – Энциклопедия – orgprint.com https://orgprint.com	19 Янв 2021	Интернет Плюс	0	3
[56]	0,4%	0,4%	СКАЧАТЬ ОДНИМ АРХИВОМ (RAR - 81Mb)... http://inueco.ru	20 Ноя 2016	Интернет Плюс	1	1
[57]	0%	0,4%	Лабораторная работа построение диаграммы https://topuch.ru	12 Апр 2021	Интернет Плюс	0	1
[58]	0%	0,4%	Зверева, Елена Александровна Формирование ИКТ-компетентности бакалавров направления «Приборостроение» в процессе производственной практики : диссертация ... кандидата педагогических наук : 13.00.08 Чебоксары 2020 http://dlib.rsl.ru	22 Окт 2020	Сводная коллекция РГБ	0	2
[59]	0%	0,39%	http://edu.sfu-kras.ru/sites/edu.sfu-kras.ru/files/oop/disciplines/09.03.04.30_annotacii_disciplin_0.pdf http://edu.sfu-kras.ru	26 Мар 2020	Интернет Плюс	0	2
[60]	0,31%	0,39%	не указано	раньше 2011	Шаблонные фразы	7	9
[61]	0%	0,39%	Метод функционального моделирования - Архитектура и проектирование программных систем https://studref.com	19 Янв 2021	Интернет Плюс	0	1
[62]	0%	0,39%	Зайцев БИ-132	24 Авг 2018	Модуль поиска "volsu"	0	1
[63]	0%	0,36%	Презентация на тему Сущности и принципы реинжиниринга бизнес-процессов - Скачать презентации по экономике https://prezentacii.org	11 Апр 2019	Интернет Плюс	0	2
[64]	0%	0,36%	Автоматизированная информационная система производства нефтяного оборудования https://knowledge.allbest.ru	14 Фев 2021	Интернет Плюс	0	2
[65]	0%	0,34%	Dfd диаграммы – поток данных https://steptosleep.ru	10 Июн 2019	Интернет Плюс	0	2
[66]	0%	0,32%	Сборник научных работ. Часть 2 (3/3) https://belstu.by	03 Мар 2018	Интернет Плюс	0	2
[67]	0,26%	0,29%	Проблемы и возможности современной науки. http://elibrary.ru	05 Ноя 2015	Перефразирования по eLIBRARY.RU	2	1
[68]	0%	0,29%	https://www.gup.ru/events/news/smi/do20.pdf https://gup.ru	30 Окт 2020	Интернет Плюс	0	1
[69]	0%	0,29%	https://tksu.ru/upload/iblock/5d1/Annotatsii-38.03.04-GiMU-UER-2018-OFO-ZFO.pdf https://tksu.ru	14 Ноя 2020	Интернет Плюс	0	1
[70]	0%	0,29%	http://www2.bigpi.biysk.ru/umu/doc/obrazovanie/praktiki/Praktiki_ZKUz_01.09.16.pdf http://www2.bigpi.biysk.ru	11 Ноя 2020	Интернет Плюс	0	1
			https://3minut.ru/images/PDF/2015/16/p				

[71]	0%	0,29%	rimenenie-tekhnologii-trekhmernoj-pechati.pdf https://3minut.ru	11 Июн 2021	Интернет Плюс	0	2	
[72]	0%	0,29%	АДДИТИВНЫЕ ТЕХНОЛОГИИ В ПРОЦЕССЕ ИЗУЧЕНИЯ ИНЖЕНЕРНОЙ ГРАФИКИ. http://elibrary.ru	20 Авг 2020	eLIBRARY.RU	0	2	
[73]	0%	0,29%	Автоматизация бизнес-процессов с использованием 1С: Предприятие. http://elibrary.ru	26 Мар 2020	Перефразирования по eLIBRARY.RU	0	1	
[74]	0%	0,28%	Альманах: Культура. Искусство. Реставрация. 2015 (1). Часть 1: Реставрация http://ibooks.ru	09 Дек 2016	Сводная коллекция ЭБС	0	1	
[75]	0%	0,28%	https://storage.tusur.ru/files/127272/2019_2.pdf https://storage.tusur.ru	22 Июн 2019	Интернет Плюс	0	1	
[76]	0%	0,28%	Педиатрия. История болезни http://studentlibrary.ru	20 Дек 2016	Медицина	0	1	
[77]	0%	0,28%	Педиатрия. История болезни http://studentlibrary.ru	20 Янв 2020	Медицина	0	1	
[78]	0%	0,28%	НИР Фридман.pdf	20 Окт 2020	Модуль поиска "volsu"	0	1	
[79]	0%	0,28%	Инструментальные средства информационных систем. Учебное пособие http://bibliorossica.com	26 Мая 2016	Сводная коллекция ЭБС	0	2	
[80]	0%	0,28%	История медицины. Книга первая. Руководство к преподаванию http://studentlibrary.ru	26 Янв 2018	Медицина	0	1	
[81]	0%	0,26%	Матрица компетенций http://rea.ru	05 Янв 2018	Переводные заимствования (RuEn)	0	1	
[82]	0%	0,26%	Summary/definition and structure of competence - PDF http://docplayer.net	06 Янв 2018	Переводные заимствования (RuEn)	0	1	
[83]	0%	0,25%	Учебная практика: Методические указания по выполнению программы учебной практики для студентов, обучающихся в магистратуре по направлению 35.04.02 «Технология лесозаготовительных и деревоперерабатывающих производств», профиль «Технология деревообработки» https://e.lanbook.com	22 Янв 2020	Сводная коллекция ЭБС	0	1	
[84]	0%	0,25%	Территория Нефтегаз № 08 2018 http://neftegas.info	19 Янв 2021	Интернет Плюс	0	2	
[85]	0%	0,25%	Территория Нефтегаз № 08 2018 http://neftegas.info	06 Мая 2021	Интернет Плюс	0	2	
[86]	0,24%	0,24%	Бактерии и их метаболиты – основа новых методов оценки антибактериальных свойств биозащищенных материалов http://dep.nlb.by	11 Ноя 2016	Диссертации НББ	1	1	
[87]	0,05%	0,22%	Проектирование информационных систем и баз данных http://studentlibrary.ru	20 Янв 2020	Сводная коллекция ЭБС	1	1	
[88]	0%	0,21%	Автоматизация бизнес-процессов с использованием 1С: Предприятие. http://elibrary.ru	26 Мар 2020	eLIBRARY.RU	0	1	
[89]	0%	0,21%	Информационные системы в экономике http://studentlibrary.ru	20 Дек 2016	Медицина	0	1	
[90]	0,21%	0,21%	Геометрия 1 http://studentlibrary.ru	20 Дек 2016	Медицина	1	1	
[91]	0,2%	0,2%	GCODE: Основы https://3dtoday.ru	11 Июн 2021	Интернет Плюс	2	2	
[92]	0,09%	0,2%	Курс классической математики в примерах и задачах. http://elibrary.ru	04 Сен 2014	eLIBRARY.RU	2	5	
[93]	0%	0,19%	СОВРЕМЕННЫЕ ТРЕБОВАНИЯ К СОДЕРЖАНИЮ ПРАВОВОЙ ПОДГОТОВКИ БАКАЛАВРОВ ПО НАПРАВЛЕНИЮ «ЖУРНАЛИСТИКА» https://e.lanbook.com	22 Янв 2020	Сводная коллекция ЭБС	0	1	Источник исключен. Причина: Маленький процент пересечения.
[94]	0%	0,17%	Практика.pdf	20 Окт 2020	Модуль поиска "volsu"	0	1	Источник исключен. Причина: Маленький процент пересечения.
[95]	0%	0,17%	Химия (для бакалавров) https://book.ru	03 Июл 2017	Сводная коллекция ЭБС	0	1	Источник исключен. Причина: Маленький процент пересечения.
[96]	0%	0,12%	Handbook по дисциплине «Методология и технология проектирования информационных систем» - PDF Free Download	23 Мая 2021	Интернет Плюс	0	2	Источник исключен. Причина: Маленький процент пересечения.

			https://docplayer.ru					
[97]	0%	0,11%	ГОСТ Р ИСО/МЭК 12207-2010 http://docs.cntd.ru	09 Окт 2020	Интернет Плюс	0	1	Источник исключен. Причина: Маленький процент пересечения.
[98]	0%	0,11%	Т.Б. Чистякова «Научно-методические основы формирования модели кадрового обеспечения индустрии переработки и использования вторичных ресурсов» http://technolog.edu.ru	14 Апр 2020	Интернет Плюс	0	1	Источник исключен. Причина: Маленький процент пересечения.
[99]	0%	0,08%	Выбор и обоснование средств и технологий разработки программного обеспечения https://mydocx.ru	05 Июн 2021	Интернет Плюс	0	1	Источник исключен. Причина: Маленький процент пересечения.
[100]	0%	0,07%	Текст http://se.math.spbu.ru	11 Июн 2021	Интернет Плюс	0	1	Источник исключен. Причина: Маленький процент пересечения.
[101]	0%	0,07%	mpl_toolkits.mplot3d.Axes3D Python Example https://programcreek.com	06 Апр 2021	Интернет Плюс	0	1	Источник исключен. Причина: Маленький процент пересечения.

Введение

Быстрые темпы развития промышленности в современном мире требуют использования передовых технологий производства. Такие технологии характеризуются рядом важных критериев, определяющих их достоинства, среди которых выделяют быстрые сроки выполнения поставленных задач и наименьшие затраты на ресурсы. Аддитивные технологии, которые также часто называют 3D-печатью, соответствуют таким темпам развития и на сегодняшний день приобретают большую популярность.

3D-печать – технология производства, при которой производится создание реального объекта по его цифровой трёхмерной модели [22]. Эта технология позволяет изготавливать практически всё, что может потребоваться человеку (от небольших деталей, до медицинских протезов и сложных геометрических объектов). Она часто применяется при создании прототипов изделий из-за довольно высокой точности, приемлемого времени изготовления и возможности «строить то, что видишь» (WYSIWYB – What You See Is What You Build) [31].

Процесс изготовления изделия при помощи технологии 3D-печати напрямую связан с характеристиками используемого оборудования и программного обеспечения. При необходимости получить качественный продукт, который будет изготовлен с высокой точностью и минимальными отклонениями от своей виртуальной модели, возникает потребность в использовании таких инструментов и алгоритмов изготовления, которые позволяют получить такой результат [19]. Оборудование, которое применяется при данной технологии, использует подход послойного создания. Поэтому очень важно учитывать каким именно образом формируются слои изделия, ведь в дальнейшем этот критерий непосредственно влияет на качество изготовляемого предмета. Высокого качества можно добиться путём уменьшения размера слоёв, формирующих саму модель. Чем тоньше слой – тем выше качество. Уменьшение толщины приводит к увеличению количества таких слоёв, соответственно увеличивается и время изготовления изделия [32].

Требуя качества, мы жертвуем временем. Поэтому возникает необходимость в использовании такого алгоритма создания, который бы позволял при повышении качества изделия не терять большое количество временных ресурсов на его изготовление. Одним из решений данной проблемы является формирование изделия из разных по толщине слоёв. Формирование тонких слоёв в местах, где необходимо добиться более высокой детализации, и более толстых слоёв, где существует возможность пренебречь качеством, позволяет получить желаемое качество за более короткие сроки [35].

Целью данной работы является разработка информационной и математической моделей и разработка программного комплекса для управления процессом 3D-печати. Программный комплекс состоит из нескольких модулей, позволяющих производить визуализацию и обработку 3D-модели, формирование управляющего кода для 3D-принтера, а также управление принтером и процессом 3D-печати.

В первой главе производится анализ предметной области, описаны FDM технология печати, роль обработки 3D-моделей в процессе создания распечатанной модели на 3D-принтере. Также в данной главе рассмотрены преимущества STL формата файлов и особенности моделей, используемых для генерации G-кода с переменной толщиной слоя, и произведена краткая характеристика используемых языков программирования, необходимых для реализации программного комплекса.

Во второй главе содержатся описание разработанной информационной и математической моделей программного комплекса. Информационная модель включает в себя диаграмму Ганта, отражающую основные этапы разработки и подзадачи, составляющие их, а также распределение всех этапов по времени, функциональную модель IDEF0, модель потоков данных DFD и диаграммы, отражающие структуру, состояния и способы использования программы. В математической модели содержатся диаграммы, характеризующие программный комплекс, рассмотрены методы и алгоритмы для проектирования программы для создания G-кода.

Третья глава посвящена описанию реализованного программного комплекса.

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, PLA-пластика, воска или поликарбоната [29], 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 [4]. На рисунке 1 представлен процесс FDM-печати с изображением основных составляющих принтера и материала, составляющего структуру печатаемого изделия [28].

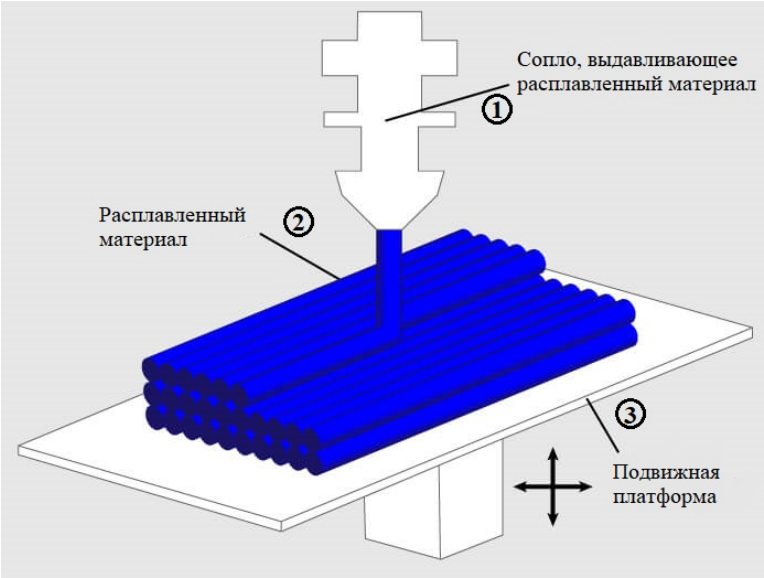


Рисунок 1 – Процесс FDM-печати, где 1 – сопло экструдера, 2 – расплавленный материал, 3 – платформа

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 [34];

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 [33];

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 [18].

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.

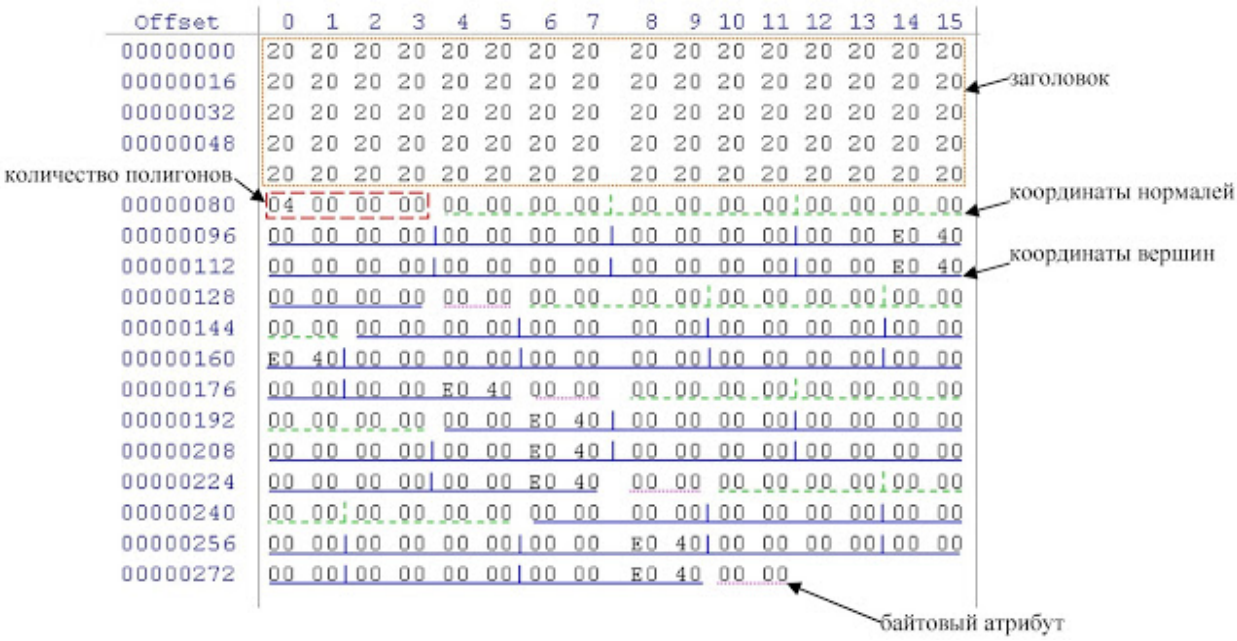


Рисунок 2 – Содержание бинарного STL файла

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.

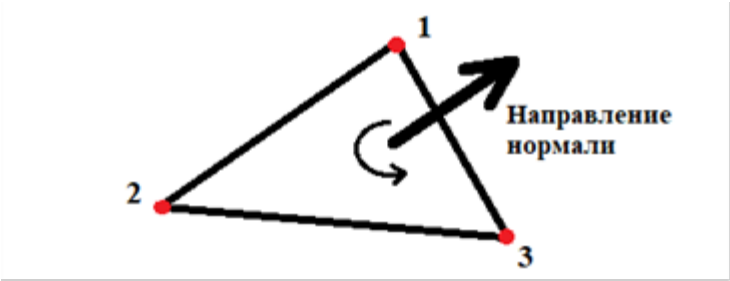


Рисунок 3 – Вектор нормали для фасета

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 [17].

1.3 Обзор языков программирования, необходимых для реализации программного комплекса

1.3.1 Язык программирования для формирования 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].

1.3.2 Язык программирования для обработки модели и графического интерфейса программного комплекса

Язык программирования C# является объектно-ориентированным и компонентно-ориентированным языком. Он представляет большое множество языковых кострукций для непосредственной поддержки такой концепции работы. Этот язык поддерживает новые рабочие нагрузки и соответствует новым рекомендациям разработки программного обеспечения.

Программы выполняются на платформе .NET, в которой присутствует удобный инструмент для создания программ с графическим интерфейсом Windows Forms. В нем присутствует большое разнообразие предустановленных элементов, с различным назначением, что помогает легко создать интерфейсы разных уровней сложности.

1.3.3 Визуализация 3D-модели посредством языка программирования Python

Python является достаточно удобным языком прогограммирования общего назначения. Он обладает динамически строгой типизацией и удобным синтаксисом. Данный язык программирования обладает большим количеством готовых библиотек, которые позволяют выполнить практически любую поставленную задачу.

Библиотеки языка Python позволяют без затруднений обработать 3D-модель формата файла STL и вызуализировать её на трехмерной сцене.

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

Разработанный 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 собой 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 [13]. 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 4.

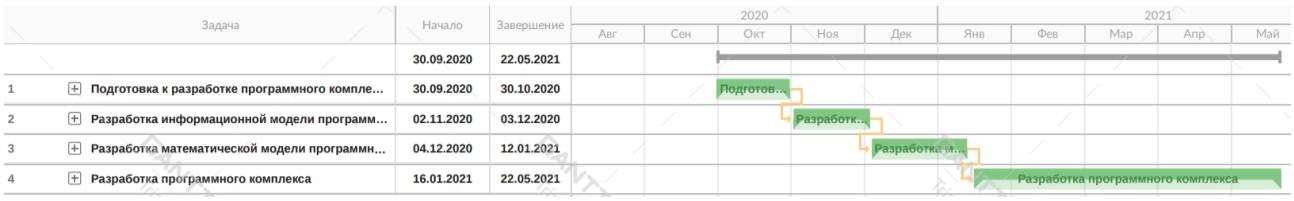


Рисунок 4 – Диаграмма Ганта

Рассмотрим задачу «Подготовка к разработке программного комплекса», которая изображена на рисунке 5.

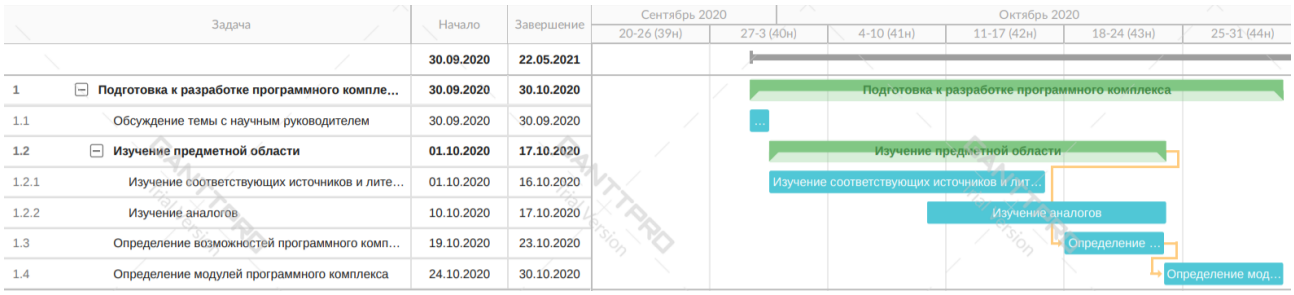


Рисунок 5 – Диаграмма Ганта задачи «Подготовка к разработке программного комплекса»

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 6.

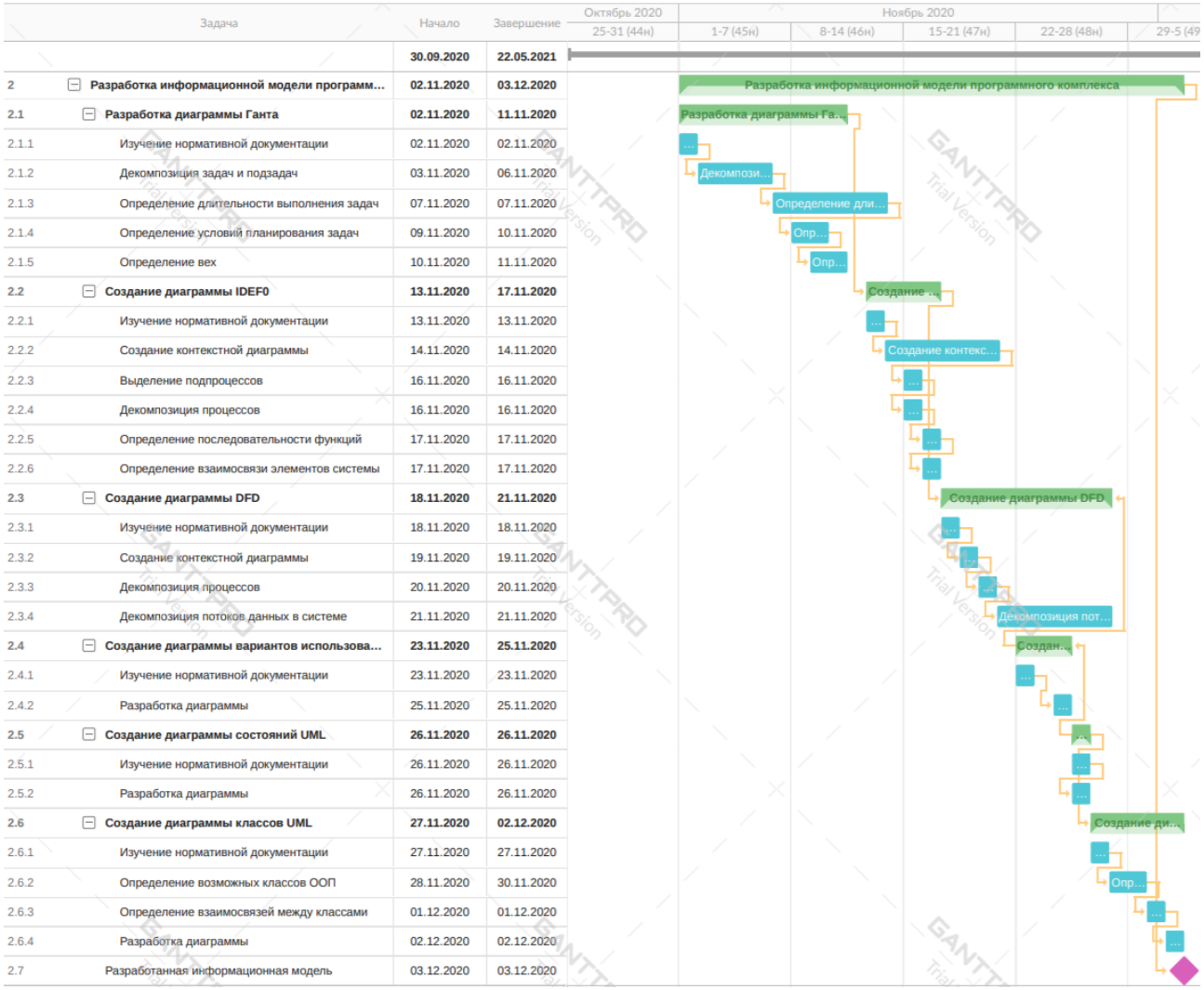
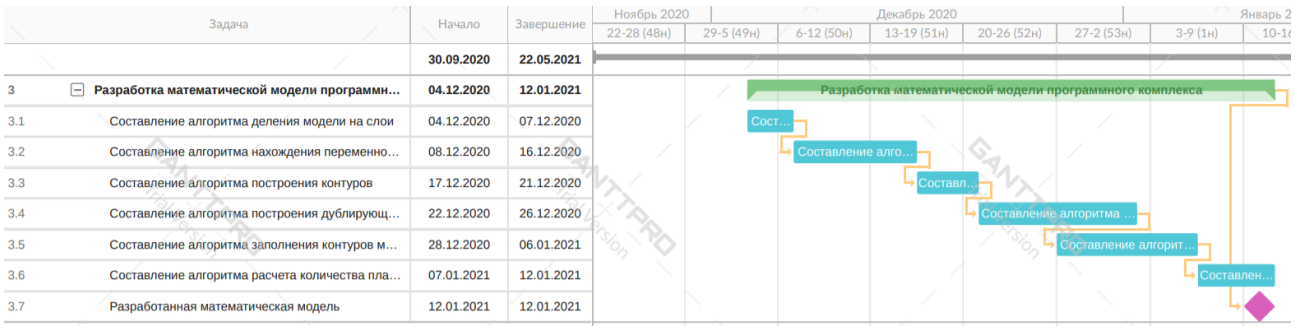


Рисунок 6 – Диаграмма Ганта задачи «Разработка информационной модели программного комплекса»

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».

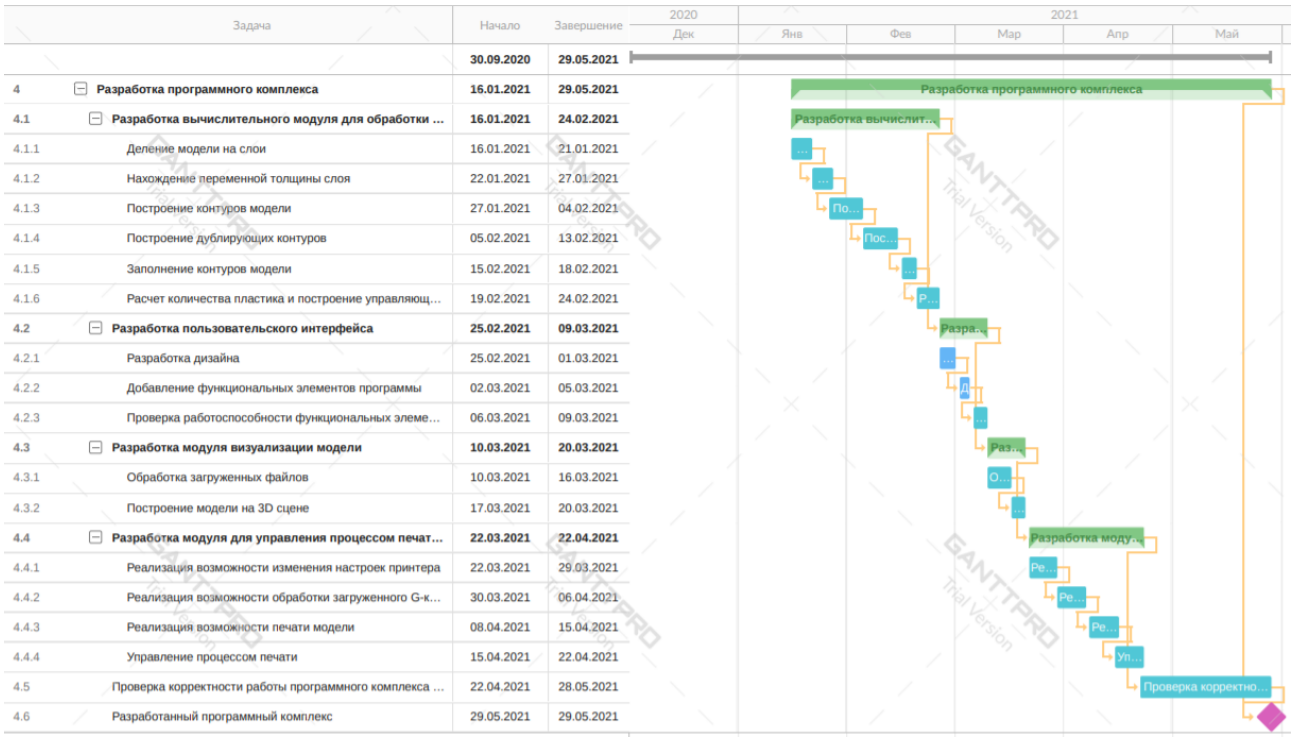
Задача «Разработка математической модели программного комплекса» показана на рисунке 7. Данная задача включает в себя алгоритмы, применяемые для обработки модели и формирования управляющего кода. Каждая подзадача подразумевает описание определенного порядка действий и математических расчетов для выполнения 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».



	Задача	Начало	Завершение
		30.09.2020	22.05.2021
3	**Разработка математической модели программн…**	04.12.2020	12.01.2021
3.1	Составление алгоритма деления модели на слои	04.12.2020	07.12.2020
3.2	Составление алгоритма нахождения переменно…	08.12.2020	16.12.2020
3.3	Составление алгоритма построения контуров	17.12.2020	21.12.2020
3.4	Составление алгоритма построения дублирующ…	22.12.2020	26.12.2020
3.5	Составление алгоритма заполнения контуров м…	28.12.2020	06.01.2021
3.6	Составление алгоритма расчета количества пла…	07.01.2021	12.01.2021
3.7	Разработанная математическая модель	12.01.2021	12.01.2021

Рисунок 7 – Диаграмма Ганта задачи «Разработка математической модели программного комплекса»

Следующая задача «Разработка программного комплекса» изображена на рисунке 8.



	Задача	Начало	Завершение
		30.09.2020	29.05.2021
4	**Разработка программного комплекса**	16.01.2021	29.05.2021
4.1	**Разработка вычислительного модуля для обработки …**	16.01.2021	24.02.2021
4.1.1	Деление модели на слои	16.01.2021	21.01.2021
4.1.2	Нахождение переменной толщины слоя	22.01.2021	27.01.2021
4.1.3	Построение контуров модели	27.01.2021	04.02.2021
4.1.4	Построение дублирующих контуров	05.02.2021	13.02.2021
4.1.5	Заполнение контуров модели	15.02.2021	18.02.2021
4.1.6	Расчет количества пластика и построение управляющ…	19.02.2021	24.02.2021
4.2	**Разработка пользовательского интерфейса**	25.02.2021	09.03.2021
4.2.1	Разработка дизайна	25.02.2021	01.03.2021
4.2.2	Добавление функциональных элементов программы	02.03.2021	05.03.2021
4.2.3	Проверка работоспособности функциональных элеме…	06.03.2021	09.03.2021
4.3	**Разработка модуля визуализации модели**	10.03.2021	20.03.2021
4.3.1	Обработка загруженных файлов	10.03.2021	16.03.2021
4.3.2	Построение модели на 3D сцене	17.03.2021	20.03.2021
4.4	**Разработка модуля для управления процессом печат…**	22.03.2021	22.04.2021
4.4.1	Реализация возможности изменения настроек принтера	22.03.2021	29.03.2021
4.4.2	Реализация возможности обработки загруженного G-к…	30.03.2021	06.04.2021
4.4.3	Реализация возможности печати модели	08.04.2021	15.04.2021
4.4.4	Управление процессом печати	15.04.2021	22.04.2021
4.5	Проверка корректности работы программного комплекса …	22.04.2021	28.05.2021
4.6	Разработанный программный комплекс	29.05.2021	29.05.2021

Рисунок 8 – 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 визуализации 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.

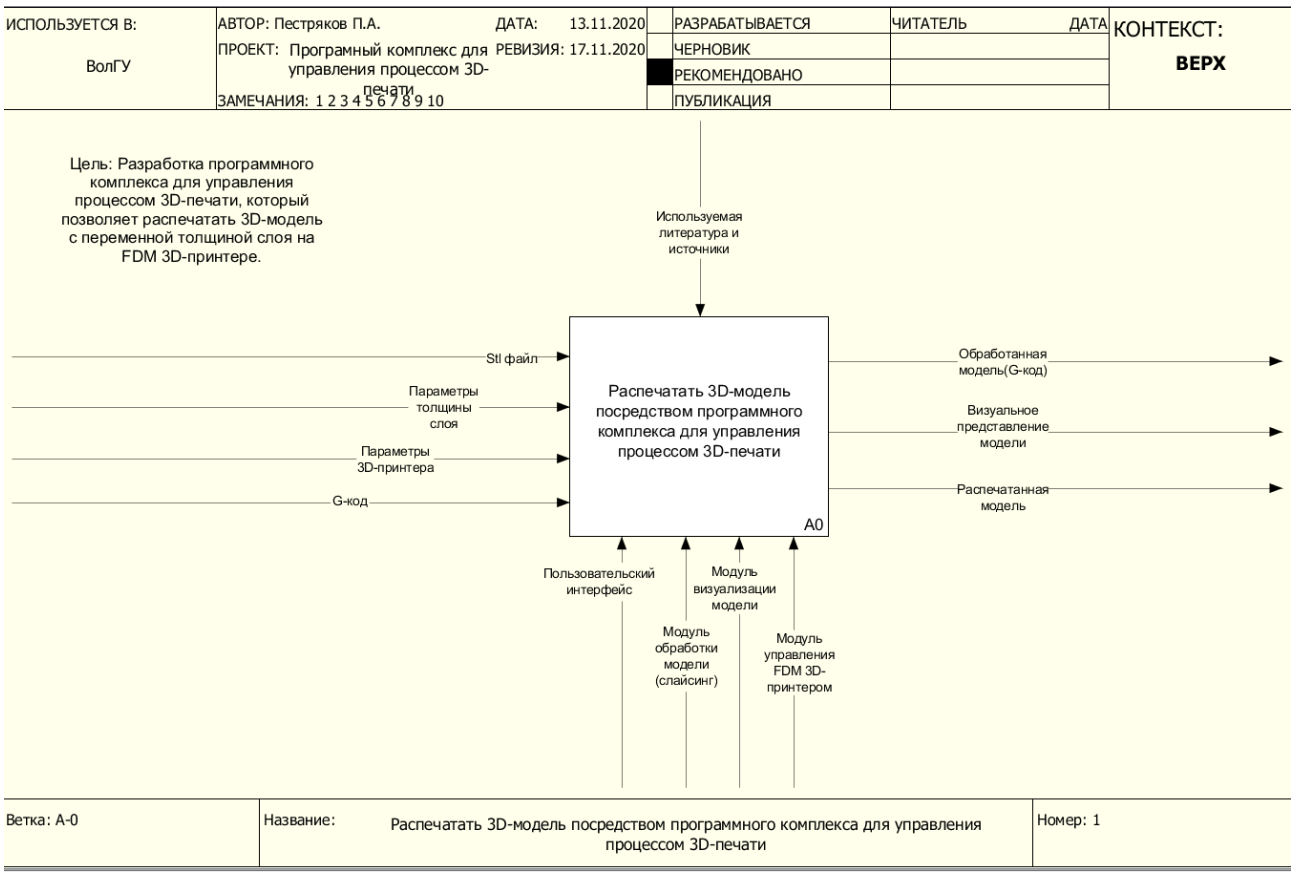


Рисунок 9 – Контекстная диаграмма

На рисунке 9 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.

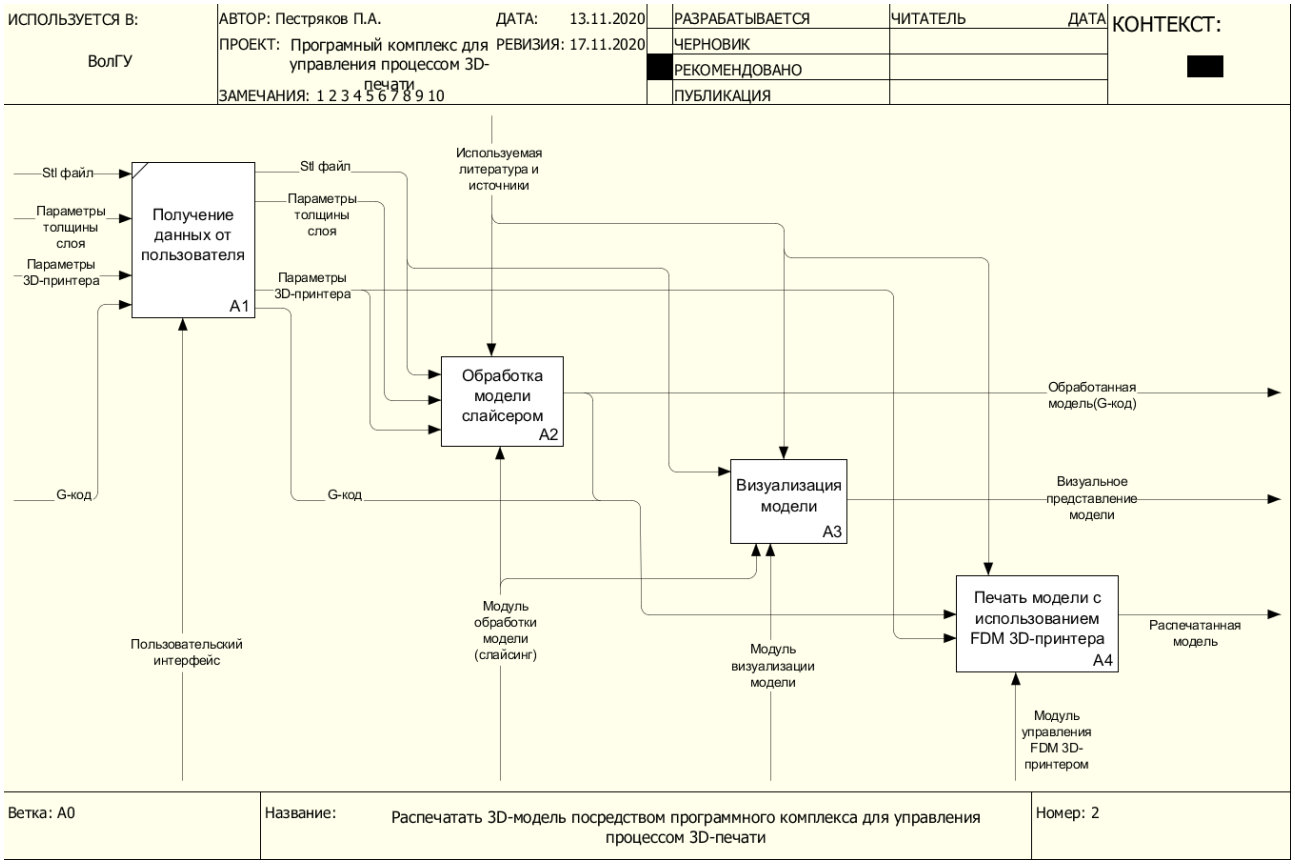


Рисунок 10 – Диаграмма декомпозиции процесса A0

На рисунке 10 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».

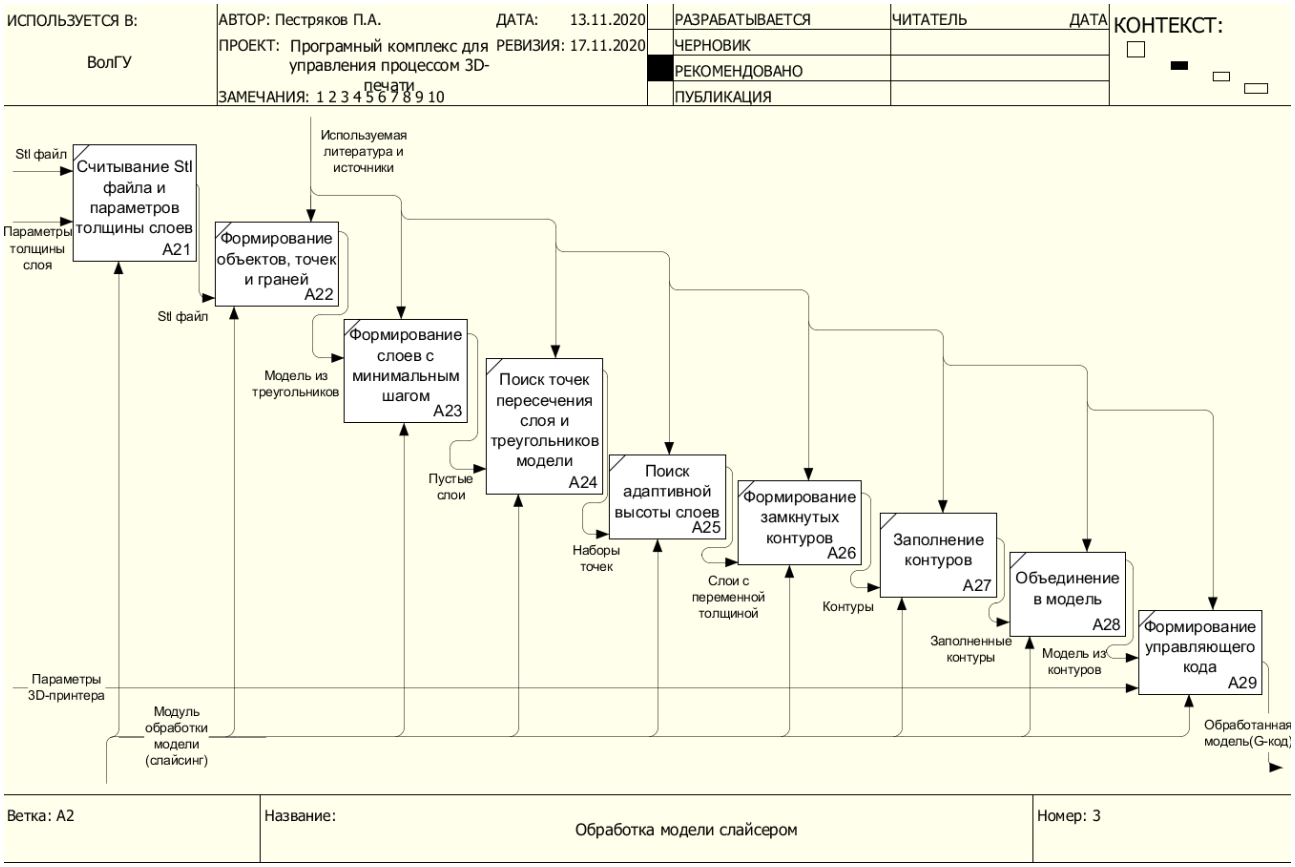


Рисунок 11 – Диаграмма декомпозиции процесса «Обработка модели слайсером»

На рисунке 11 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)».

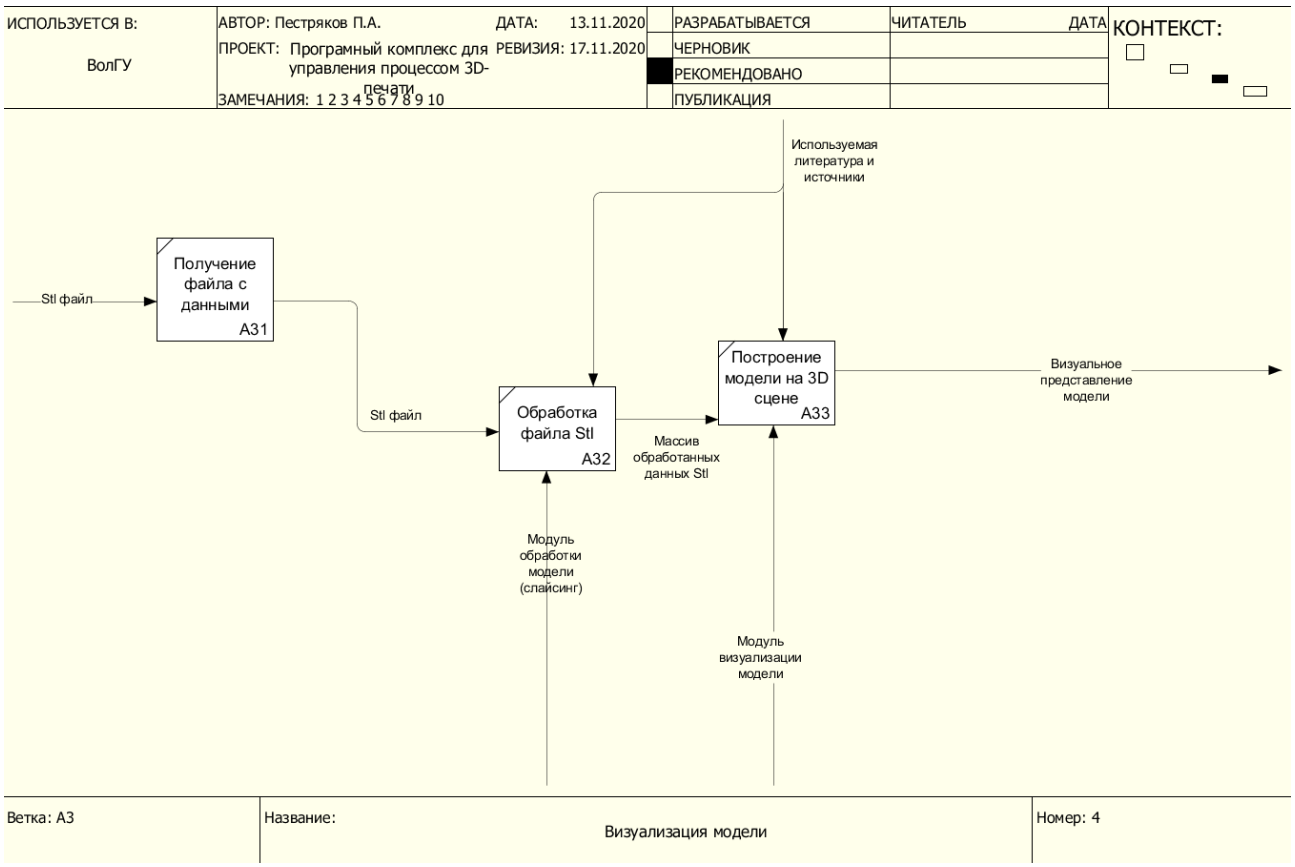


Рисунок 12 – Диаграмма декомпозиции процесса «Визуализация модели»

На рисунке 12 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.

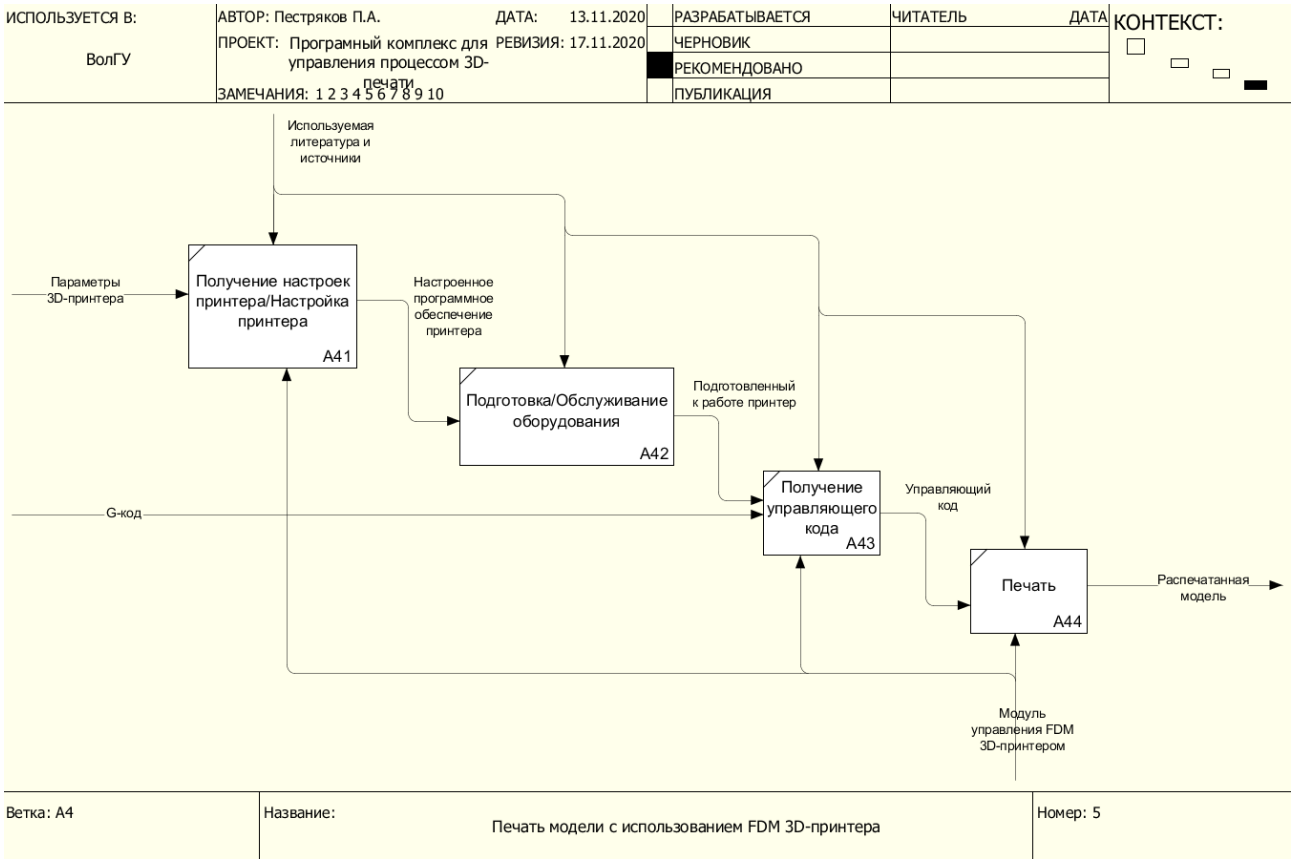


Рисунок 13 – Диаграмма декомпозиции процесса «Печать модели с использованием FDM 3D-принтера»

На рисунке 13 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].

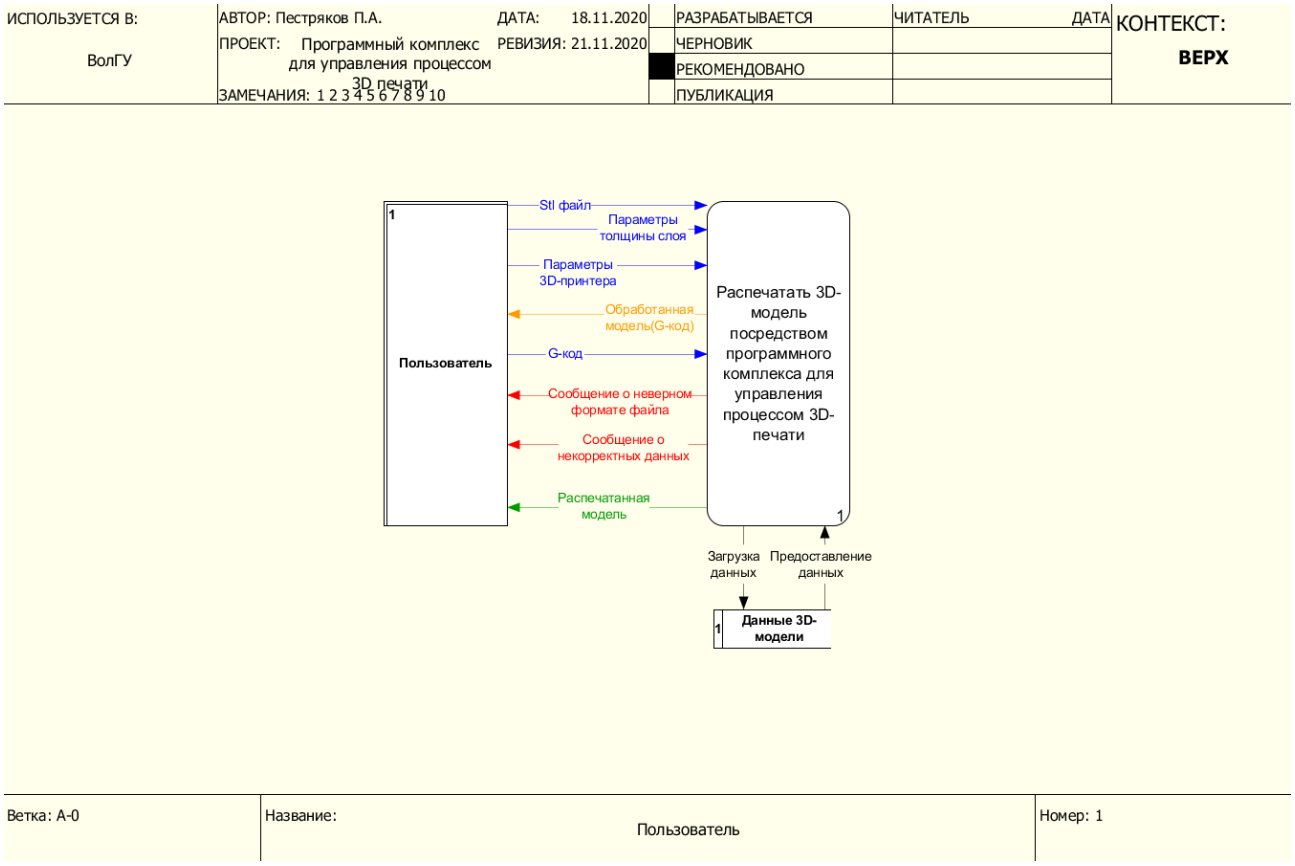


Рисунок 14 – Контекстная диаграмма

На рисунке 14 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».

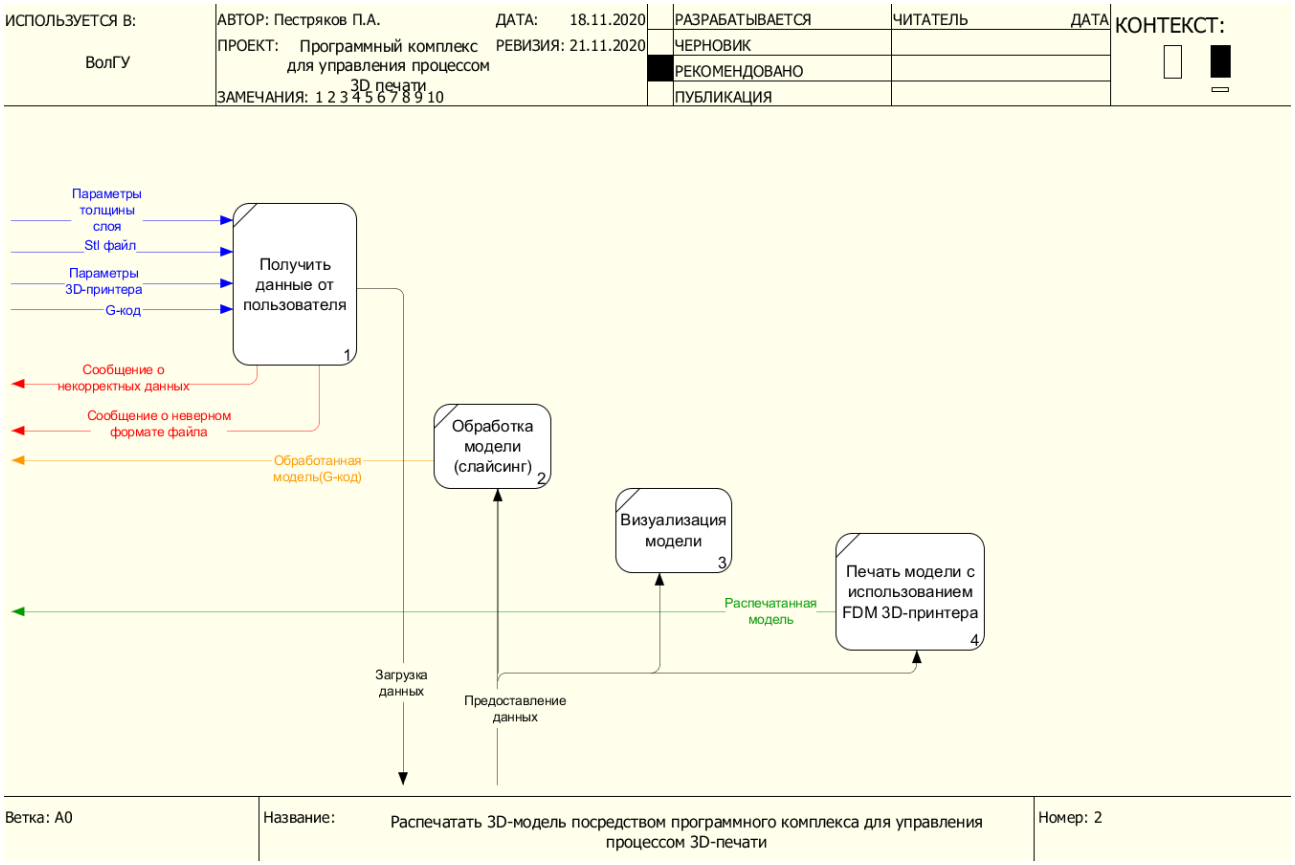


Рисунок 15 – Декомпозиция контекстной диаграммы

На рисунке 15 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, содержащий 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 16 представлена диаграмма классов, отражающая классы, используемые в ходе выполнения программы и взаимосвязи между ними.

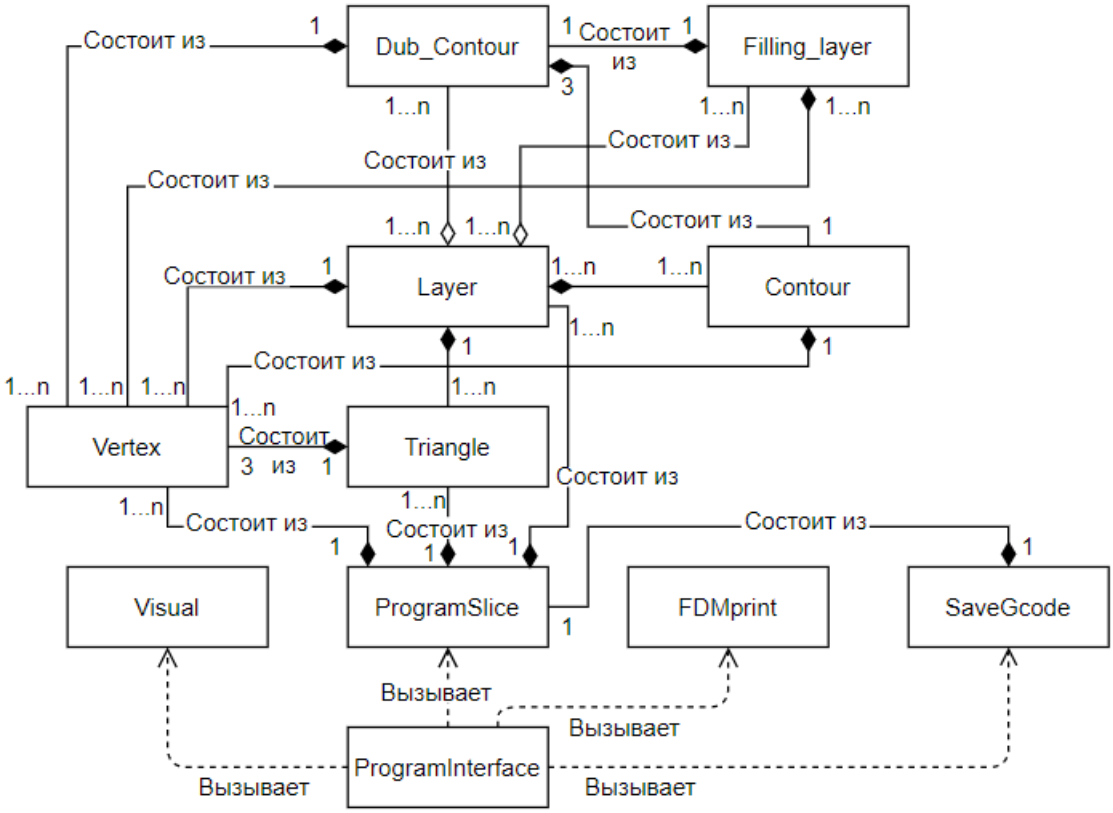


Рисунок 16 – Диаграмма классов

В классе ProgramInterface 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 ProgramSlice отвечает за обработку загруженной модели. Он 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. Для каждого слоя определяются данные для формирования контура, дубликатов контура, заполнения контура и другие.

Формирование контуров модели производится в классе Contour. Контуры, повторяющие очертания и имеющие смещенные к центру фигуры координаты, формируются в классе Dub_Contour. Заполнение контуров производится в классе Filling_layer.

Класс Visual принимает данные модели и отвечает за визуализацию 3D-модели на трехмерной сцене.

Класс SaveGcode принимает данные обработанной модели и отвечает формирование управляющего кода.

Класс FDMprint 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 17, отображены основные состояния модели в процессе её обработки от 3D-модели до записи управляющего G-кода.

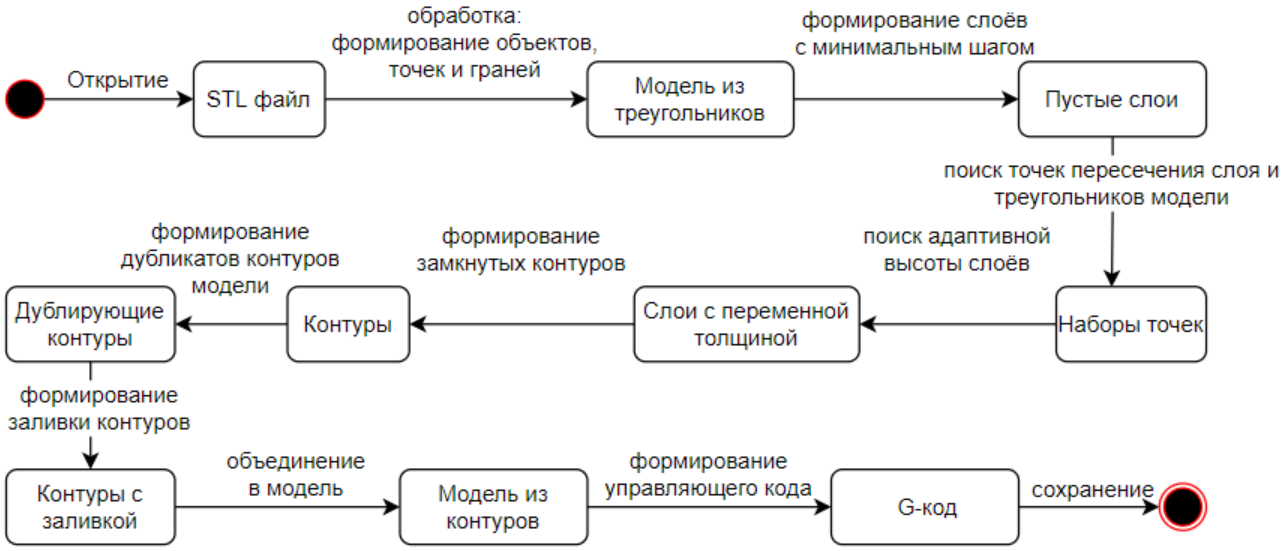


Рисунок 17 – Диаграмма состояний

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 18. Она отображает возможные действия пользователя при работе с программой.

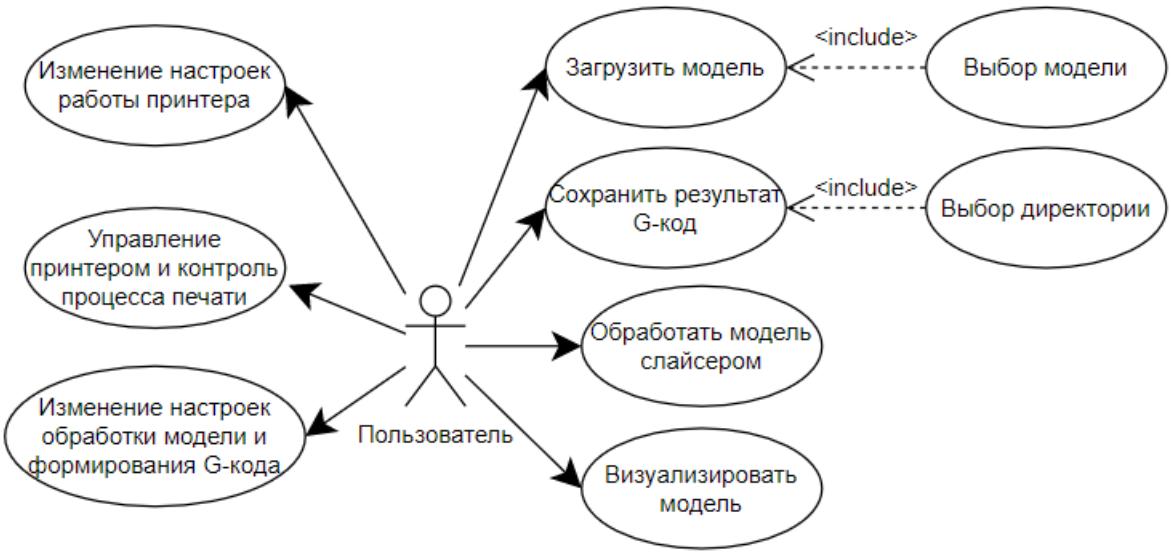


Рисунок 18 – Диаграмма вариантов использования

Диаграмма включает в себя такие варианты действий пользователя, как:

1) выбор и загрузка модели для обработки из указанной директории;
2) обработка модели слайсером;
3) визуализация модели;
4) сохранение результата обработки слайсером в указанной директории;
5) изменение настроек работы принтера;
6) управление принтером и контроль процесса печати;
7) изменение настроек обработки модели и формирования G-кода.

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 [27].

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:

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

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 пересечения с плоскостью слоя. Для определения

принадлежности треугольника к слою необходимо выполнение хотя бы одного из условий системы:

$$z1-z0z2-z0\leq 0,z1-z0z3-z0\leq 0,z3-z0z2-z0\leq 0, \quad (2)$$

где $z1$, $z2$, $z3$ – значения координат z для вершин треугольника, а $z0$ – 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 19 показаны варианты таких пересечений.

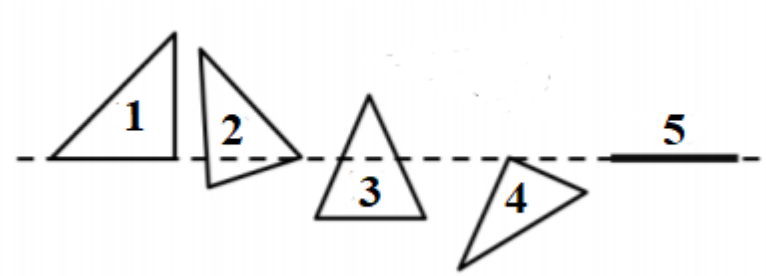


Рисунок 19 – Варианты пересечения треугольником плоскости слоя

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:

$$cosangle = 1\text{-}zx2+y2+z22\ , \quad (3)$$

где cos(angle) – косинус угла, x, y, z – координаты вектора нормали треугольника [3].

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

$$cosangle=1 \rightarrow h=hmin, cosangle=0 \rightarrow h=hmax, \quad (4)$$

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

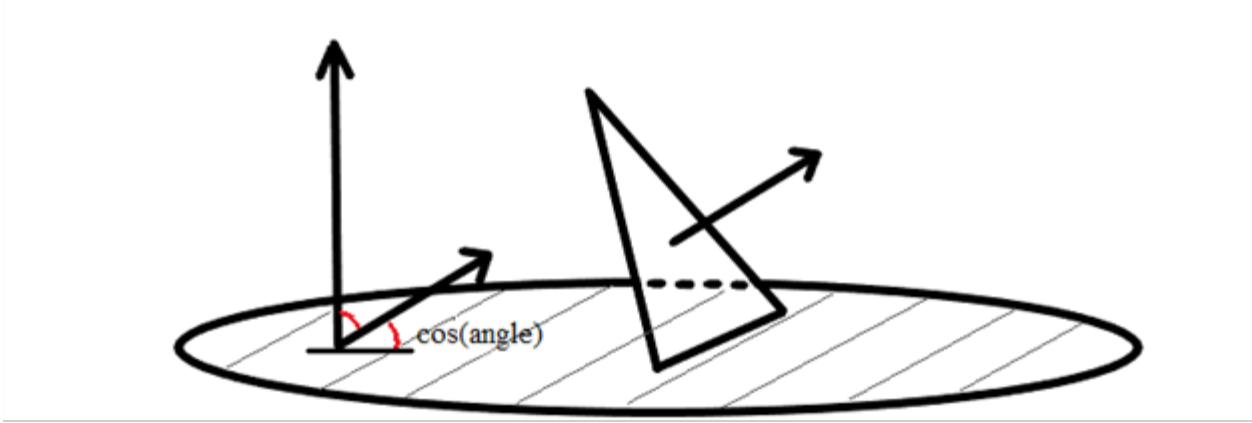


Рисунок 20 – Положение треугольника в зависимости от вектора нормали

Значение толщины слоя h находится в диапазоне:

$$hmin \leq h \leq hmax\ . \quad (5)$$

При этом шаг будет равен минимальному допустимому значению:

$$\Delta h0 = hmin\ , \tag{6}$$

где $\Delta h0$ – шаг.

Пусть переменная x принимает значение минимального вектора нормали:

$$x= cosangle\ . \tag{7}$$

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

$$h\ x = 1\text{-}x\ hmax\text{-}hmin + hmin\ , \tag{8}$$

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

$$h\ 0 = 1\text{-}0\ hmax\text{-}hmin + hmin \rightarrow h\ 0 = hmax\ , \tag{9}$$

$$h\ 1 = 1\text{-}1\ hmax\text{-}hmin + hmin \rightarrow h\ 1 = hmin.$$

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

Для нахождения окончательного значения толщины слоя, будем использовать формулу:

$$h^* = int\ hmaxhx+0.5\ hmin\ , \tag{10}$$

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:

$$zk+1 = zk + hk\ , \tag{11}$$

где zk+1 – значение по координате z для следующего слоя, zk для начала текущего, а hk – 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:

$$z1>zz2\leq zz3\leq z \quad \text{или} \quad z1<zz2\geq zz3\geq z , \quad (15)$$

где $z1$, $z2$, $z3$ – координаты вершин треугольника, z – координата плоскости слоя.

Из вершины с координатой $z1$ 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.

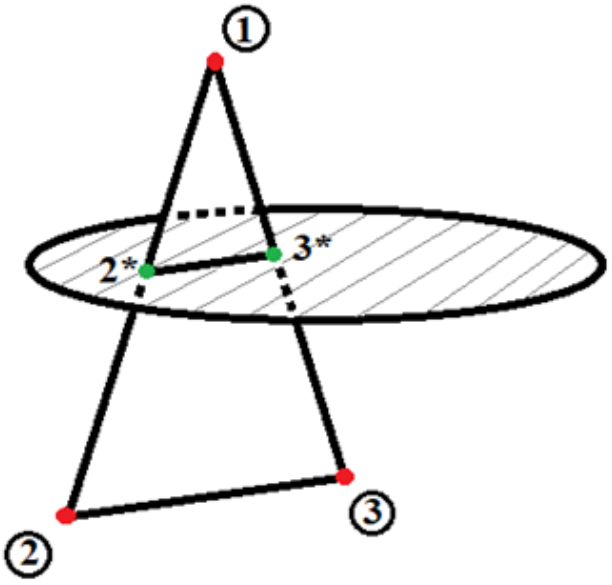


Рисунок 21 – Пример пересечения треугольником плоскости

Для того чтобы найти значения координат точки 2* , рассматривается грань, на которой она лежит [6]. Необходимо составить уравнение прямой, находящейся между точками 1 и 2. Пусть точка 1 это M1 (x1 ; y1 ; z1) , а точка 2 это M2 (x2 ; y2 ; z2) .

Уравнение прямой в пространстве определяется по формуле [16]:

$$x2-x1m = y2-y1n = z2-z1p = \lambda. \quad (16)$$

Для точки M1 уравнение (16) предстоит в виде:

$$x-x1m = y-y1n = z-z1p = \lambda. \quad (17)$$

Решая совместно уравнения эти уравнения, получим:

$$x-x1x2-x1 = y-y1y2-y1 = z-z1z2-z1 = \lambda. \quad (18)$$

Уравнение (18) – это уравнение прямой, проходящей через две точки в пространстве [7]. Оно также подходит для уравнения прямой, проходящей через точки M1 и M2 .

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

$$\lambda 0 = z0-z1z2-z1 , \quad (19)$$

где $\lambda 0$ – параметр слоя, z0 – значение z плоскости слоя.

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

$$x=x1+\lambda x2-x1,y=y1+\lambda y2-y1,z=z1+\lambda z2-z1. \quad (20)$$

Для точки 2* (xi ; yi ; zi) , подставляя в (20), получим:

$$xi=x1+\lambda 0x2-x1,yi=y1+\lambda 0y2-y1,zi=z0. \quad (21)$$

Это уравнения для нахождения координат xi и yi в точке 2* . Значение zi находить нет необходимости, оно равно значению z0 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 22.

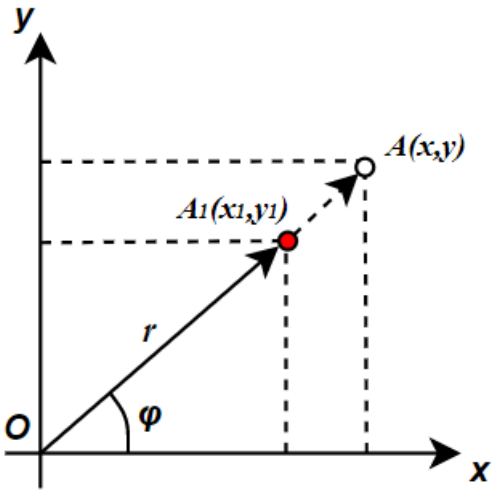


Рисунок 22 – Поиск точки дублирующего контура

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 (22). Найденные значения (x ;y) будут являться координатами центра текущего контура.

$$center = 0NcoordN\ , \tag{22}$$

где center – координата x или y , N – количество точек контура, с oord – значение координаты контура [9].

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 (23).

$$r = x2+y2\ , \tag{23}$$

где r – полярный радиус, x и y – координаты точки [11].

Затем производится поиск коэффициента отношения новых координат и старых по формуле (24).

$$k= r-Hr \ , \quad (24)$$

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:

$$y=kx+b, \quad (25)$$

где x и y – значения координат точек, k – угловой коэффициент, b – значение пересечения прямой с осью ординат [21].

Коэффициенты k и b рассчитываются по формулам (26) и (27) соответственно и необходимы для определения координат точки пересечения прямых на плоскости.

$$k= y2-y1x2-x1 \ , \quad (26)$$

где $x1$ и $y1$ – значения координат первой точки, а $x2$ и $y2$ – координаты второй точки, через которые проходит прямая.

$$b= x2y1-x1y2x2-x1 \ , \quad (27)$$

где x1 и y1 – значения координат первой точки, а x2 и y2 – координаты второй точки, через которые проходит прямая.

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

$$x=b2-b1k1-k2,y=k1b2-k2b1k1-k2, \quad (28)$$

где x и y – координаты точки пересечения двух прямых, k1 и b1 – значения коэффициентов уравнения первой прямой, а k2 и b2 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 (23), где пунктирными линиями обозначено перемещение печатающей головки.

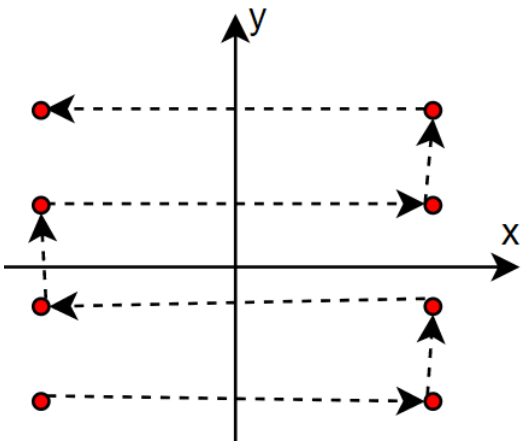


Рисунок 23 – Перемещение печатающей головки в процессе заполнения контура

2.2.6 Расчет количества пластика и построение управляющего кода

Для построения модели используются команды перемещения, в которых определяются точки, непосредственно к которым производится перемещение экструдера. Также они регулируют изменение положения координатного стола. Команды перемещения к следующей точке для экструдера и стола образуют строки, из которых и формируется строение файла с G-кодом модели. Каждая строка файла – это отдельная команда, которую необходимо выполнить принтеру. В одной строке с этими командами указывается 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:

$$Ei = Ei\text{-}1 + dE, \tag{29}$$

где Ei – длина подаваемого пластика на текущем шаге, а Ei-1 на предыдущем, а dE – количество подаваемого пластика на текущем шаге и рассчитывается по формуле:

$$dE = hDeLDp22\pi , \tag{30}$$

где – диаметр сопла, – диаметр пластика, *h* – высота слоя, – длина печатаемого отрезка [24], которая рассчитывается по формуле:

$$L= xi+1\text{-}xi2+yi+1\text{-}yi2 , \tag{31}$$

где xi , yi – координаты точки, xi+1 , yi+1 – координаты следующей точки [14].

Стоит указать, что также важной частью файла с G-кодом модели являются команды подготовки оборудования к печати и команды завершения печати модели. К примеру, для конкретной модели принтера и составляющего его оборудования может быть важным определение точных значений температуры экструдера и стола, до которых необходимо их разогреть, перед процессом печати, или какие действия необходимо предпринять по завершению процесса печати. Также важно учитывать характеристики материала, используемого для печати, которые влияют на значения указываемых

параметров и команд, подаваемых на принтер. Завершение печати обычно включает себя команды перевода координатного стола и экструдера в положение парковки и отключение нагревательных элементов принтера.

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. Полный программный код метода находится в листинге А.7.

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. Полный код методов приведен в листинге А.3.

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 : triangles[i].v1.Z;
    zmin = (zmin < triangles[i].v2.Z) ? zmin : triangles[i].v2.Z;
    zmin = (zmin < triangles[i].v3.Z) ? zmin : triangles[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 : triangles[i].v1.Z;
   zmax = (zmax > triangles[i].v2.Z) ? zmax : triangles[i].v2.Z;
   zmax = (zmax > triangles[i].v3.Z) ? zmax : triangles[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. Полный код приведен в листинге А.3.

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 : layers[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 приведен в листинге А.4.

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

Для нахождения точек контура внутри имеющегося и повторяющие его очертания изначально производится поиск центра контура в методе Find_center(), результатом выполнения которого являются координаты точки со значениями по осям абсцисс и ординат. Координаты дублирующей точки вычисляются в методе New_vertex() (листинг 8). Данный метод выполняется по всем значениям имеющегося контура. В результате таких действий получается массив значений нового контура, преобразованные координаты которого имеют смещение на расстояние, равное диаметру установленного сопла на экструдере. Полный код приведен в листинге А.5.

Листинг 8 – Метод New_vertex()

```
//изменяем точки, смещением в 0,0 - начало координат, для полярной СК
            float x = V.X - center.X;
float y = V.Y - center.Y;
            float   De = 0.4f;

//преобразование  координат  в  полярные
            float   r = (float)Math.Sqrt(x * x + y * y); //полярный  радиус
            // отношение новых и старых координат
            float   k = (r - De) / r;
            //перевод координат в декартовы
x *= k;
y *= k;
x += center.X; //смещаем обратно
            y += center.Y;
            return new Vertex(x, y);
```

3.1.8 Заполнение сформированных контуров модели

Заполнение модели производится послойно, в каждом слое модели выполняется расчет точек для построения линий заполнения. Метод Rtrn_fill() (листинг 9) отвечает за вызов метода поиска экстремумов среди точек слоя для дальнейшей работы метода вычисления координат точек заполнения Find_VertexFill(). Полный код приведен в листинге А.6.

Листинг 9 – Метод Rtrn_fill()

```
                float[] extremum_X = new float[2];
        float[] extremum_Y = new float[2];
        float De = 0.4f;
        extremum_X[0] = V[0].X; extremum_X[1] = V[0].X; //[0] - min; [1] - max
        extremum_Y[0] = V[0].Y; extremum_Y[1] = V[0].Y;
        for (int j = 0; j < Num_v_layer; j++)
        {
           extremum_X = find_extremum(V[j].X, extremum_X);
           extremum_Y = find_extremum(V[j].Y, extremum_Y);
            }
        int Num_fill_L = (int)((extremum_Y[1] - extremum_Y[0]) / De); // кол-во  линий, режущих
слой
        Vertex[] fill_V = find_VertexFill(V, Num_v_layer, extremum_X, extremum_Y, Num_fill_L, H);
                return fill_V;
```

В листинге 10 представлена часть метода Find_VertexFill(). Здесь производится поиск вершин пересечения прямых, режущих слой с прямыми, образующими контур модели.

Листинг 10 – Метод Find_VertexFill()

```
for (int i = 0; i < fill_V.Length; i++)
{
   if (cur_Y <= extremum_Y[1])
   {
      line[0].Y = cur_Y;
      line[1].Y = cur_Y;
                        for (int j = 0; j < Num_v_layer - 1; j++)
                        {
         if (V[j].Y >= cur_Y && V[j + 1].Y < cur_Y)
         {
```

```
                    float k1 = Dub_contour.Calc_k(V[j], V[j + 1]);
                    float b1 = Dub_contour.Calc_b(V[j], V[j + 1]);
                    float k2 = Dub_contour.Calc_k(line[0], line[1]);
                    float b2 = Dub_contour.Calc_b(line[0], line[1]);
                    fill_V[i] = Dub_contour.Find_vertex(k1, b1, k2, b2);
                    i++;
                }
                else if (V[j + 1].Y >= cur_Y && V[j].Y < cur_Y)
                {
                    float k1 = Dub_contour.Calc_k(V[j + 1], V[j]);
                    float b1 = Dub_contour.Calc_b(V[j + 1], V[j]);
                    float k2 = Dub_contour.Calc_k(line[0], line[1]);
                    float b2 = Dub_contour.Calc_b(line[0], line[1]);
                    fill_V[i] = Dub_contour.Find_vertex(k1, b1, k2, b2);
                                        i++;
                }
                else if (V[j].Y == cur_Y && V[j + 1].Y == cur_Y) //обе прямые параллельны Ox и линия
контура лежит на другой
                                        {
                    fill_V[i] = V[j];
                    i++;
                    fill_V[i] = V[j + 1];
                    i++;
                                        }
            }
                                cur_Y += De; //шаг  по  оси  Oy
                                i--;
                        }
        }
```

Затем производится сортировка найденных точек таким образом, чтобы пары точек, образующих линию заполнения, изменяли направление передвижения экструдера, как это описано в пункте 2.2.5. Результатом выполнения метода является массив точек, содержащий координаты передвижения печатающей головки принтера, для формирования заполнения слоя модели.

3.1.9 Расчет количества пластика и построение управляющего кода

Метод Gcode(), часть которого представлена в листинге 11, 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.7.

Листинг 11 – Метод Gcode()

```
            for (int i = 0; i < N; i++)
{
   if (layers[i].Num_v_layer != 0)
   {
      sw.Write("G1 F1600\n");
      //V
      for (int j = 0; j < layers[i].Num_v_layer; j++)
      {
         Vertex curr = layers[i].V[j];
         int j_next = j + 1;
         if (j == layers[i].Num_v_layer - 1) j_next = 0;
         Vertex next = layers[i].V[j_next];
         L = Math.Sqrt(Math.Pow(next.X - curr.X, 2) + Math.Pow(next.Y - curr.Y, 2));
         E += ((layers[i].H * De * L) / (Math.Pow(Dp / 2, 2) * Math.PI));
         sw.Write($"G1 X{(curr.X + 100):F3} Y{(curr.Y + 100):F3} E{E:F5}\n".Replace(",", "."));
      }
      sw.Write($"G1 X{(layers[i].V1[0].X + 100):F3} Y{(layers[i].V1[0].Y + 100):F3}\n".Replace(",", "."));
      //V1
      for (int j = 0; j < layers[i].Num_v_layer; j++)
      {
         Vertex curr = layers[i].V1[j];
         int j_next = j + 1;
         if (j == layers[i].Num_v_layer - 1) j_next = 0;
         Vertex next = layers[i].V1[j_next];
         L = Math.Sqrt(Math.Pow(next.X - curr.X, 2) + Math.Pow(next.Y - curr.Y, 2));
         E += ((layers[i].H * De * L) / (Math.Pow(Dp / 2, 2) * Math.PI));
         sw.Write($"G1 X{(curr.X + 100):F3} Y{(curr.Y + 100):F3} E{E:F5}\n".Replace(",", "."));
      }
      sw.Write($"G1 X{(layers[i].V2[0].X + 100):F3} Y{(layers[i].V2[0].Y + 100):F3}\n".Replace(",", "."));
      //V2
      for (int j = 0; j < layers[i].Num_v_layer; j++)
      {
         Vertex curr = layers[i].V2[j];
         int j_next = j + 1;
         if (j == layers[i].Num_v_layer - 1) j_next = 0;
         Vertex next = layers[i].V2[j_next];
         L = Math.Sqrt(Math.Pow(next.X - curr.X, 2) + Math.Pow(next.Y - curr.Y, 2));
         E += ((layers[i].H * De * L) / (Math.Pow(Dp / 2, 2) * Math.PI));
         sw.Write($"G1 X{(curr.X + 100):F3} Y{(curr.Y + 100):F3} E{E:F5}\n".Replace(",", "."));
      }
      if(layers[i].Num_fill_V != 0)
      {
```

```
                sw.Write($"G1 X{(layers[i].fill_V[0].X + 100):F3} Y{(layers[i].fill_V[0].Y + 100):F3}\n".Replace(",",
".."));
                //fill_V
                for (int j = 0; j < layers[i].Num_fill_V; j++)
                {
                   if (layers[i].fill_V[j] != null)
                   {
                      Vertex curr = layers[i].fill_V[j];
                      int j_next = j + 1;
                      if (j == layers[i].Num_fill_V - 1) j_next = 0;
                      Vertex next = layers[i].fill_V[j_next];
                      L = Math.Sqrt(Math.Pow(next.X - curr.X, 2) + Math.Pow(next.Y - curr.Y, 2));
                      E += ((layers[i].H * De * L) / (Math.Pow(Dp / 2, 2) * Math.PI));
                      sw.Write($"G1 X{(curr.X + 100):F3} Y{(curr.Y + 100):F3} E{E:F5}\n".Replace(",", "."));
                   }
                }
             }
             if (layers[i ] != null)
             {
                h = h + layers[i ].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

Модуль построения 3D-сцены и визуализации модели на ней реализован с использованием библиотек stl, matplotlib и mpl_toolfits на языке программирования Python 3.8.3. В листинге 12 приведен код метода Visual(), в котором производится считывание данных из файла формата STL и дальнейшая их визуализация на 3D-сцене. Данный модуль вызывается событием, при нажатии соответствующей кнопки на графическом интерфейсе программы.

Листинг 12 – Метод Visual()

```
figure=pyplot.figure()
axes=mplot3d.Axes3D(figure)
```

```
model_mesh=mesh.Mesh.from_file(input())
axes.add_collection3d(mplot3d.art3d.Poly3DCollection(model_mesh.vectors))

scale=model_mesh.points.flatten()
axes.auto_scale_xyz(scale,scale,scale)

pyplot.show()
```

3.2.2 Взаимодействие компонент программного комплекса

Графический интерфейс программного комплекса разрабатывался с помощью платформы пользовательского интерфейса Windows Forms. Взаимодействие компонент программного комплекса производится посредством вызова событий и методов, реализованных в соответствующих им визуальных элементах управления.

В листинге 13 показан обработчик события Printer_Click(), срабатывающий при нажатии на визуальный элемент кнопки. Полный код приведен в листинге А.8.

Листинг 13 – Метод Printer_Click()

```
                serialPort = new SerialPort(COMportList.Text, Convert.ToInt32(speedBod.Text));
serialPort.DataReceived += new SerialDataReceivedEventHandler(serialPort_DataReceived);
                if (serialPort.IsOpen)
   MessageBox.Show("Не удалось установить соединение");
else
{
   serialPort.Open();
   MessageBox.Show("Соединение установлено");
                        timer01.Interval = 3000;
   timer01.Tick += new EventHandler(timer1_Tick);
                        timer01.Start();
}
```

Здесь производится подключение программы к 3D-принтеру по последовательному порту. В параметрах подключения передается название

последовательного порта и скорость передачи данных, указанных пользователем. Также пользователю сообщается о результате подключения.

Определение доступных последовательных портов показано в листинге 14. Их названия записываются в визуальный компонент ComboBox с названием COMportList. Также, по умолчанию, производится выбор последовательного порта среди имеющихся, название которого не определено как COM1, так как последовательный порт подключенного принтера будет иметь любое другое название.

Листинг 14 – Метод UpdatePort_Click()

```
COMportList.Items.Clear();
string[] portNames = SerialPort.GetPortNames();
foreach (string p in portNames)
                COMportList.Items.Add(p);
if (COMportList.Items.Count > 0)
{
   for (int i = 0; i < portNames.Length; i++)
      if (portNames[i] != "COM1")
         COMportList.SelectedIndex = i;
              }
```

Получение реакции принтера на команды и её запись в журнал команд выполняется в событии (листинг 15). Также здесь производится запись значений с датчиков температуры в соответствующие им визуальные элементы на графическом окне. Это необходимо для осуществления контроля значений нагревательных элементов принтера и определения пользователем корректных параметров печати изделия и работы принтера.

Листинг 15 – Метод SerialPort_DataReceived()

```
    readSerialPort = serialPort.ReadLine();
this.Invoke(new Action(() =>
{
   Log.AppendText(time + readSerialPort + Environment.NewLine);
}));
if (M105 == true)
{
```

```
    MatchCollection TB = Regex.Matches(readSerialPort, @"\d+(\.\d+)*");
    if (TB.Count > 2)
    {
      this.Invoke(new Action(() =>
      {
        tempExtruderCurrent.Text = Convert.ToString(TB[0]);
        tempTableCurrent.Text = Convert.ToString(TB[2]);
                        }));
      M105 = false;
    }
  }
```

Управление компонентами принтера, такими как двигатели экструдера и координатного стола, охлаждение, нагревательные элементы и остальными, производится аналогичным образом. При нажатии на соответствующую кнопку на последовательный порт должна подаваться определенная команда, с указанными пользователем параметрами. К примеру, в листинге 16 приведен код включения и выключения охлаждения. Полный код приведен в листинге А.8.

Листинг 16 – Метод FunOnOff_Click()

```
  if (!OnOffFan)
{
                    FanOnOff.Text = "Выкл";
    FanOnOff.BackColor = Color.Lime;
    serialPort.WriteLine("M106 S" + speedFunUpDown.Text);
    OnOffFan = !OnOffFan;
}
else
{
    FanOnOff.Text = "Вкл";
    FanOnOff.BackColor = Color.Transparent;
    serialPort.WriteLine("M107");
                    OnOffFan = !OnOffFan;}
```

Порядок отправления команд на принтер для печати модели описывается в методе Print_Click(). Команды построчно считываются с выбранного пользователем файла G-кода модели и отправляются на подключенный последовательный порт, соответствующий 3D-принтеру. Необходимо учитывать, что на выполнение каждой команды, принтеру необходимо

некоторое время. Для этого стоит ожидать реакции принтера на команду и считывать строку, которую подает принтер на последовательный порт.

Также в графическом интерфейсе присутствуют кнопки для вызова модуля обработки модели, выбора файла для обработки и сохранения сформированного управляющего кода в указанную директорию.

3.3 Тестирование и результаты работы программного комплекса

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

Модуль обработки 3D-модели составляет управляющий код для 3D-принтера. Тестирование модуля обработки 3D-модели проводилось методом «Белого ящика». Результатом обработки является G-код модели, в котором построчно указываются команды необходимые для построения слоев модели, формирующих будущее твердотельное изделие. На рисунке 24 показан результат обработки 3D-модели шахматной фигуры. По сформированному G-коду видно, что модель состоит из различных по толщине слоев, имеет дублирующие контуры и плотное заполнение. По результатам обработки видно, что обработанная модель соответствует ожидаемым результатам.

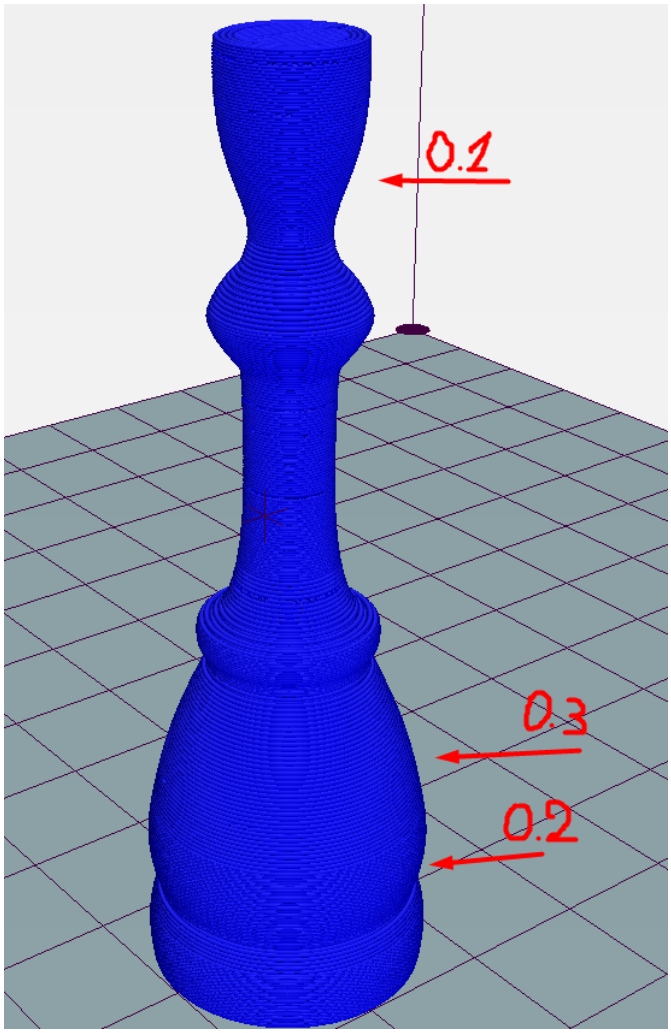


Рисунок 24 – G-код обработанной модели, визуализированный в программе Repetier-Host

Для обработки были указаны параметры толщины модели, где минимальная толщина равна 100 микрон и максимальная 300 микрон, диаметр экструдера 400 микрон.

Дублирование контуров и заполнение модели производятся по алгоритмам, описанным в пунктах 3.1.7 и 3.1.8, соответственно. На рисунке 25 видно, что слой модели заполняется соответственно ожидаемому результату, контуры-дубликаты строятся корректно и повторяют очертания внешнего контура слоя модели. Толщина линий заполнения соответствует толщине текущего слоя.

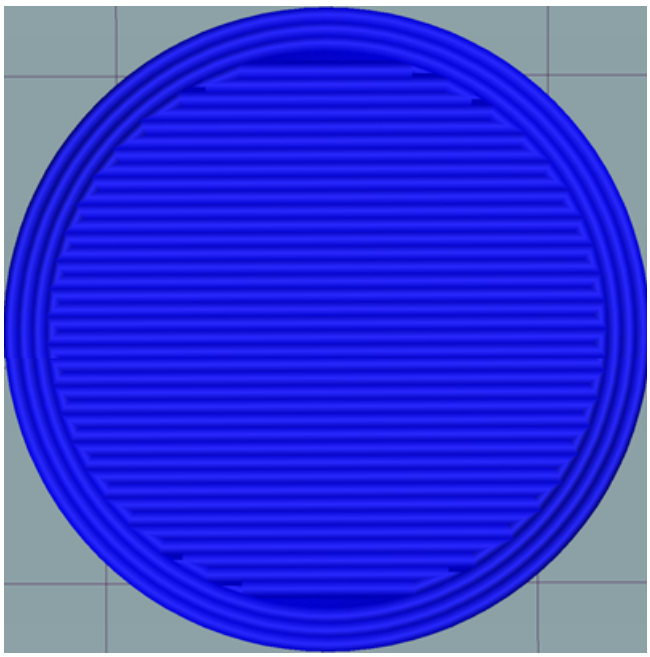


Рисунок 25 – G-код заполнения слоя модели, визуализированный в программе Repetier-Host

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

Графическое окно слайсера показано на рисунке 26. Здесь производится загрузка модели, визуализация, обработка модели и сохранение G-кода.

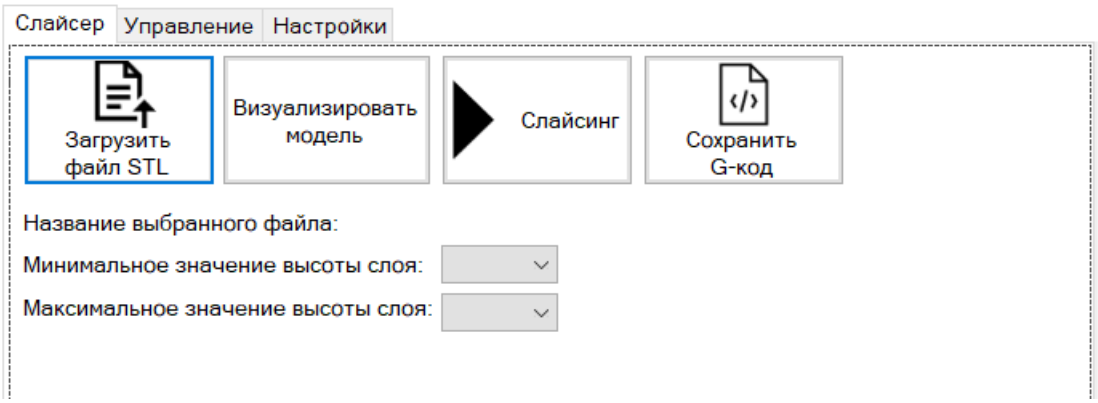


Рисунок 26 – Графическое окно слайсера

Тестирование и проверка работоспособности производилось на 3D-принтере PrintBox3D One с помощью метода «Белого ящика». На рисунке 27 показано окно подключения к последовательному порту и параметры, необходимые для корректной работы 3D-принтера. Здесь можно наблюдать сообщение о корректном подключении к принтеру, название порта и стандартные характеристики работы принтера, что соответствует ожидаемым результатам.

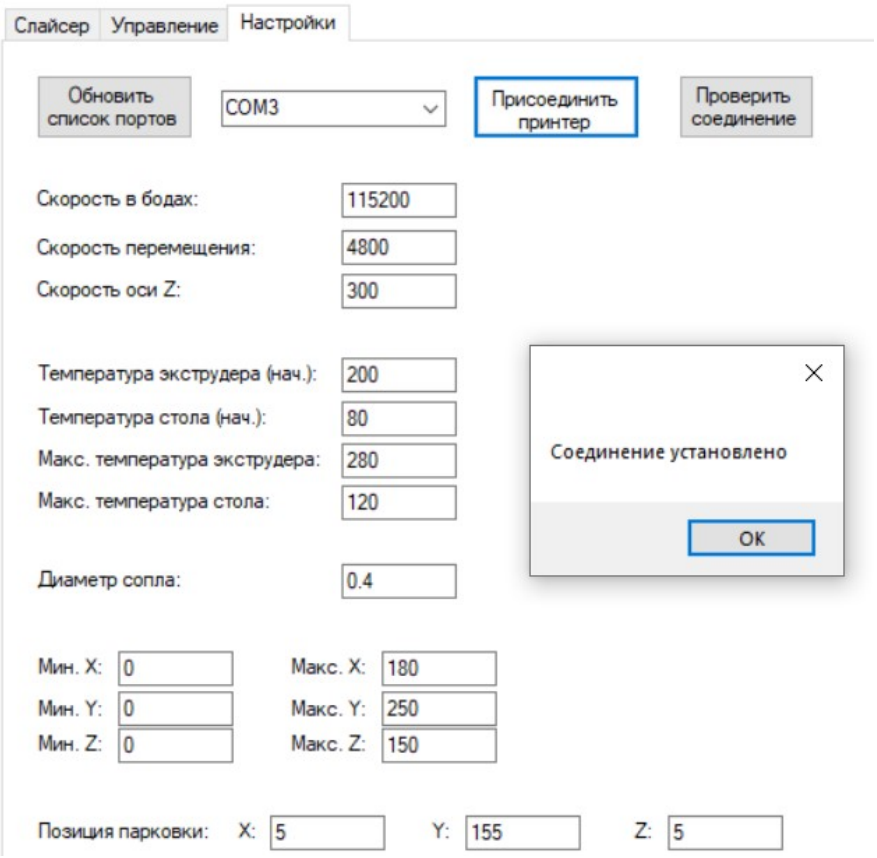


Рисунок 27 – Графическое окно настроек

На рисунке 28 изображено окно управления 3D-принтером, в котором осуществляется контроль датчиков температуры и текущих координат, указываются параметры работы компонент принтера, а также присутствует возможность запуска и остановки печати. Здесь видно, что в журнале команд подается команда на принтер для контроля датчиков температуры. Принтер, в свою очередь, корректно принимает команду и выдает результаты с датчиков.

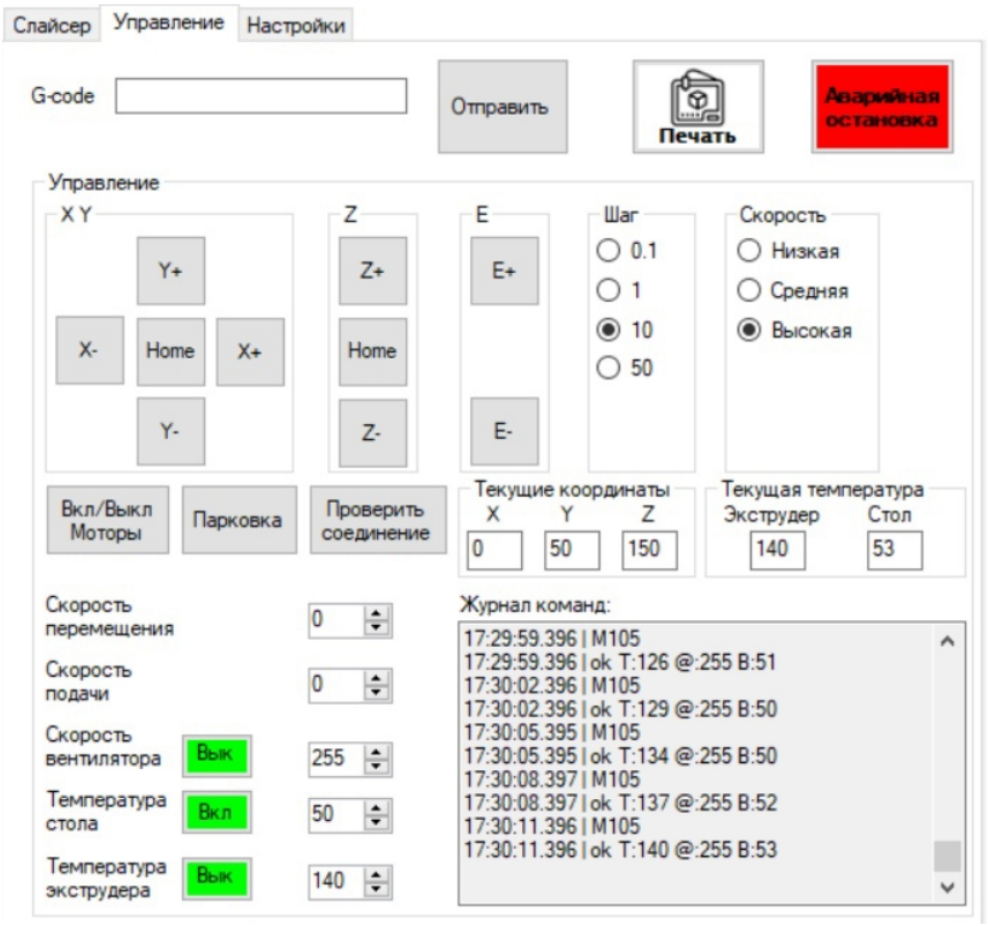


Рисунок 28 – Графическое окно управления принтером

Взаимодействие с компонентами принтера производится успешно. Экструдер и координатный стол корректно меняют своё положение и температуру, соответствуя подаваемым командам на принтер.

На рисунке 29 показано окно визуализации загруженной 3D-модели. На данном рисунке присутствует графическая 3D-модель и сцена с тремя осями декартовой системы координат, на которой располагается модель.

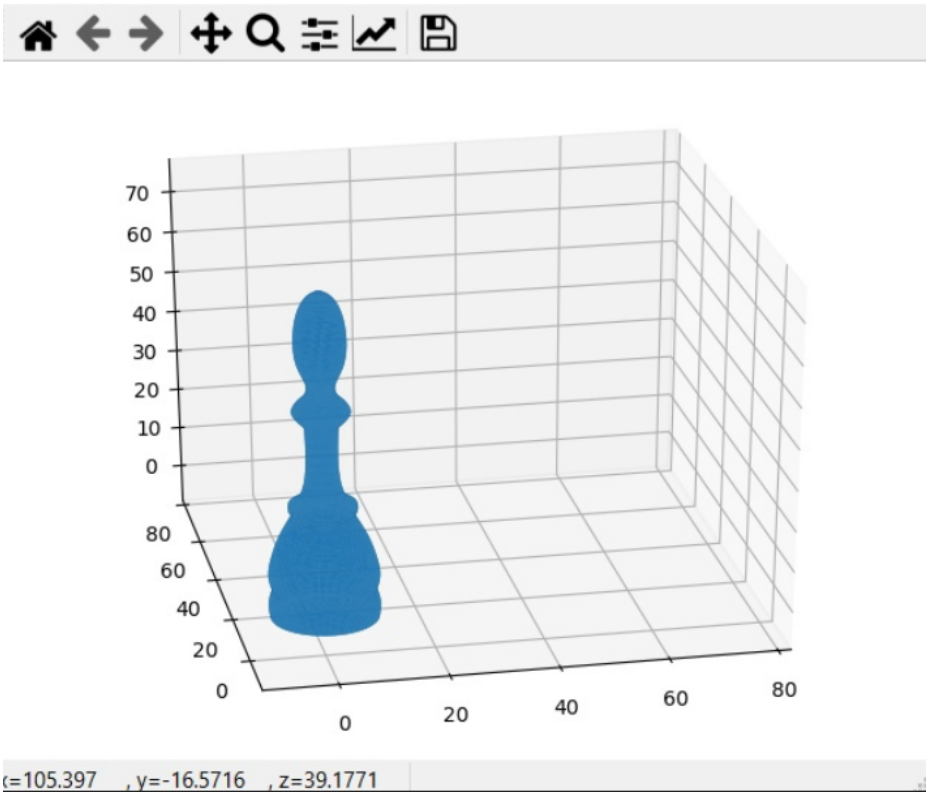


Рисунок 29 – Графическое окно визуализации модели

На рисунке 30 представлена твердотельная модель, распечатанная на 3D-принтере PrintBox3D One. По результату печати видно, что готовое изделие соответствует ожидаемым результатам печати по сформированному G-коду модели. Форма и размеры изделия соответствуют изначальной 3D-модели. Слои, из которых состоит изделие, имеют разную переменную толщину, в рамках указанных параметров при обработке 3D-модели.

Рисунок 30 – Распечатанная модель с переменной толщиной слоя

Заключение

Данная работа посвящена разработке программного комплекса для управления процессом 3D-печати.

В ходе работы были изучены основы технологии FDM-печати, особенности и структура бинарных файлов формата STL, структура управляющего кода, рассмотрены технологии послойного разделения 3D-моделей.

Была разработана информационная модель программного комплекса. Она включает диаграмму Ганта, описывающую этапы и сроки разработки, взаимосвязи между этапами разработки. Были спроектированы и описаны функциональная модель, модель потоков данных, а также произведена декомпозиция каждой разработанной модели. Созданы и описаны диаграммы, описывающие структуру, состояния и способы использования программы.

Также была разработана математическая модель модуля обработки модели. Она включает описание алгоритмов и формул, применяемых в разработке каждого из этапов обработки модели.

Результатом работы является спроектированный и реализованный программный комплекс для управления процессом 3D-печати, который включает в себя модуль обработки модели, модуль визуализации, графический интерфейс с возможностью взаимодействия с 3D-принтером. Программный комплекс был протестирован и были получены положительные результаты каждого входящего в него модуля.

В результате выполнения выпускной квалификационной работы 60 были сформированы следующие компетенции:

При выполнении работы производился поиск информации по следующим темам: «Обработка трехмерных моделей формата STL», «FDM технология 3D-печати», «Адаптивная толщина слоев трехмерной модели», «Алгоритмы заполнения трехмерных моделей», «Создание библиотеки классов и

подключение ее к проекту», «Создание исполняемого файла средствами pyinstaller», «Управление 3D-принтером через последовательное соединение», «Формирование G-кода модели» и ее критический анализ, выявление необходимой информации для реализации конкретной задачи, что соответствует компетенции:

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

Была спроектирована информационная модель программного комплекса, которая содержит диаграммы IDEF0 и DFD, в каждой из которых произведена декомпозиция выделенных процессов. В результате были определены задачи, необходимые для реализации программного комплекса, и разработан календарный график выполнения ВКР на основе диаграммы Ганта для планирования и управления задачами. Также использовалось рекомендуемое программное обеспечение в процессе создания выпускной квалификационной работы. Это соответствует освоению компетенции:

УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений.

В процессе работы над программным комплексом в устной и письменной форме осуществлялась коммуникация с научным руководителем Храповым С.С., по вопросам выполнения выпускной квалификационной работы. Также производились консультации с преподавателями кафедры ИСКМ и сотрудниками ООО «Теленово», специалистами в сфере 3D-моделирования, что соответствует освоению следующей компетенции:

УК-3 Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде.

В ходе выполнения работы в устной и письменной формах осуществлялась коммуникация с научным руководителем, с преподавателями кафедры ИСКМ в лице Сиволобова С.В. по вопросу взаимодействия с 3D-принтером, Радченко В.П. по вопросам обработки трехмерных моделей,

настройке программного комплекса для корректной работы с 3D-принтером, с членами IT-сообщества в рамках русско- и англоязычных тематических форумов, что соответствует компетенции:

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

При выполнении работы была создана диаграмма Ганта, благодаря которой осуществлялось управление своим временем таким образом, чтобы этапы реализации программного комплекса и получение результатов работы производилось в установленные сроки, также были определены траектории саморазвития, что соответствует компетенции:

УК-6 Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни.

При разработке программного комплекса производилось создание его математической модели. Для ее создания применялись знания дисциплин математики и моделирования, естественнонаучные и общеинженерные знания, что соответствует освоению следующей компетенции:

ОПК-1 Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности.

В процессе изучения предметной области были использованы такие электронные библиографические базы, как Лань, Book.ru, elibrary и др. Отчет по выпускной квалификационной работы был сформирован посредством программного обеспечения LibreOffice, разработка программного кода производилась в средах Microsoft Visual Studio и PyCharm, что соответствует компетенции:

ОПК-2 Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности.

Для выполнения выпускной квалификационной работы была изучена предметная область 3D-печати, которая включает 3D-моделирование, слайсинг моделей, управление 3D-принтером. Была разработана информационная модель, включающая диаграммы программного комплекса, созданные посредством программного обеспечения Ramus. Модули программного комплекса разрабатывались при помощи таких средств разработки, как Microsoft Visual Studio и PyCharm, также осуществлялся контроль версий проекта в системе контроля версий Git. Перечисленные задачи соответствуют освоению компетенции:

ОПК-3 Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.

При выполнении работы был составлен отчет, включающий в себя информационную и математическую модели, которые содержат диаграммы, описывающие порядок выполнения задач программным комплексом и пользователем, и алгоритмы разработки частей, составляющих программный комплекс, что соответствует компетенции:

ОПК-4 Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью.

Для разработки программного комплекса была спроектирована его информационная модель, которая включает диаграммы DFD и IDEF0. Для создания диаграмм было установлено и использовано программное обеспечение Ramus, что соответствует компетенции:

ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных стем.

В рамках работы была разработана математическая модель программного комплекса, которая включает в себя алгоритм нахождения переменной толщины слоя, алгоритм дублирования контура модели, алгоритм заполнения слоя модели. По математической модели был разработан программный комплекс, позволяющий управлять процессом 3D-печати и получать твердотельные изделия с переменной толщиной слоя, что соответствует компетенции:

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов.

В процессе реализации программного комплекса применялись принципы программного управления, которые заключались в применении команд, отвечающих за математические вычисления, применяемые для получения результатов обработки 3D-моделей, что соответствует компетенции:

ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой.

При изучении предметной области, осуществлялся поиск и обработка необходимой информации, а также представление её в виде информационной и математической моделей программного комплекса, что соответствует компетенции:

ОПК-8 Способен осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.

При выполнении работы было проведено исследование предметной области и проведена разработка модуля для обработки трехмерных моделей и создания управляющего G-кода с переменной толщиной слоя, что соответствует компетенции:

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

Разработанный программный комплекс состоит из модуля обработки трехмерной модели, модуля визуализации модели, графического интерфейса программного комплекса, что соответствует освоению компетенции:

ПК-2 Способен проводить интеграцию программных модулей и компонент.

При разработке программного комплекса производилось тестирование каждого модуля и сравнение получаемого результата с ожидаемым. Результаты обработки модели сравнивались с результатами обработки аналогов программного комплекса. Также производилось тестирование программного комплекса методом «Белого ящика». Производилось тестирование разработанного программного комплекса с использованием параметров и моделей для обработки, исходя из полученных результатов, принималось решение внесения изменений или исправления ошибок программного комплекса, что соответствует компетенции:

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

В процессе выполнения работы была составлена математическая и информационная модели для разработки программного, на основе которых реализован программный комплекс для управления процессом 3D-печати, что соответствует компетенции:

ПК-4 Способен создавать и анализировать требования на разработку программно-информационных систем и подсистем.

При выполнении данной работы была написана информационная модель программного комплекса для управления процессом 3D-печати, что соответствует компетенции:

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

Список литературы