Реферат

Отчёт содержит 53 страниц, 0 рисунков, 0 таблиц, 12 формул, 0 приложений.

??

?? ?? ??

Объект ВКР — учебники издательства УрФУ.

Цель работы — разработать программное обеспечение для дешифрования информации из файлов типа DXF, извлечении из них информации о геометрических примитивах, сформировать текстовый файл, содержащий всю необходимую информацию в читабельном виде.

Методы исследования: теоретический анализ и последующий синтез информации, моделирование, тестирование.

Результатом работы стало программное обеспечение «DXF Primiview».

Область применения полученного программного обеспечения — технологическая разработка управляющих программ для производства изделий на станках с числовым программным управлением.

Значимость работы заключается в практической реализации коммерческого заказа ООО «Униматик», создании специализированного программного обеспечения — модуля экспорта DXF-файлов в САПР.

Содержание

Глоссарий

В настоящей работе применяют следующие термины с соответствующими определениями:

Программный продукт — объект, состоящий из программ, процедур, правил, а также, если предусмотрено, сопутствующих им документации и данных, относящихся к функционированию системы обработки информации. ГОСТ 28806-90 «Качество программных средств. Термины и определения» (утверждён и введён в действие Постановлением Госстандарта СССР от 25 декабря 1990 г. № 3278).

Система автоматизированного проектирования — организационно-техническая система, входящая в структуру проектной организации и осуществляющая проектирования при помощи комплекса средств автоматизированного проектирования (КСАП). ГОСТ 23501.101-87 «Системы автоматизированного проектирования. Основные положения» (утверждён и введён в действие Постановлением Государственного комитета СССР по стандартам от 26.06.87 № 2668).

Язык разметки — набор символов или последовательностей символов, вставляемых в текст для передачи информации о его отображении или строении.

Обозначения и сокращения

В настоящей ВКР применяют следующие сокращения и обозначения:

САПР — Система автоматизированного проектирования.

УП — Управляющая программа.

ЧПУ — Числовое программное управление.

ПО — Программное обеспечение.

ПП — Программный продукт.

ЯП — Язык программирования.

DXF — Drawing eXchange Format. Формат файлов для обмена графической информацией между приложениями САПР, созданный фирмой Autodesk для системы AutoCAD в 1982 г.

TXT — Текстовый формат файлов, представляющий собой последовательность строк электронного текста.

SVG — Scalable Vector Graphics. Язык разметки масштабируемой векторной 2D-графики на основе XML, поддерживающий интерактивность и анимацию. Разрабатывается консорциумом World Wide Web с 1999 года по сегодняшний день.

XML — eXtensible Markup Language. Расширяемый язык разметки подобный HTML.

JSON — JavaScript Object Notation. Текстовый формат обмена данными, основанный на языке программирования JavaScript.

Введение

Актуальность темы исследования. В настоящее время в рамках проектов Уральского Федерального Университета (УрФУ) и Уральского межрегионального научно-образовательного центра (УМНОЦ) происходит разработка специализированных САПР. Создаваемые ПП работают с файлами, содержащими геометрическую 2D-информацию. Используются следующие форматы файлов:

а) DXF,
б) TXT,
в) SVG,
г) JSON.

В связи с этим требуется разработать ПО по конвертации данных форматов файлов.

Степень разработанности темы исследования. В открытом доступе сети Интернет существуют онлайн-конвертеры файлов DXF в другие форматы. Большая их часть лишь визуализирует графическую информацию, представленную в том или ином DXF-файле, и конвертирует в наиболее популярные форматы изображений (JPEG, PNG). Для использования конвертера на предприятии-заказчике необходим особый формат данных, в которые конвертируется DXF. Такие форматы не могут быть обеспечены существующим в открытом рынке ПО — таким, как, например, «DXF Reader GT» от компании «Gray Technical». Несмотря на высокую степень проработки программы, формат выходных данных не соответствует требованиям, предъявляемым к TXT-, SVG- и JSON-файлов, компании «Unimatic». Ещё одним недостатком зарубежного ПО является ежемесячная плата разработчикам. Одной из задач современной Российской Федерации является импортозамещение на производстве, поэтому разработка

собственного ПП увеличит независимость и самостоятельность предприятий и компаний, пользующихся этим ПП.

Цель работы заключается в разработке программного обеспечения для дешифрования информации из файлов типа DXF с геометрической информацией объектов специального типа, извлечении из них информации о геометрических примитивах, вывода полученной графической информации на экран для верификации, формирования текстового файла (TXT) с выводом данных в исходном виде DXF, формирования текстового файла (TXT) с выводом данных в виде координат точек и радиуса примитивов (линий, дуг) между ними, формирования файла-описание двумерной векторной графики в формате SVG с выводом данных в исходном виде DXF, формирования текстового файла (JSON) с выводом данных в виде координат точек и степенями кривизны примитивов между ними.

Данные конвертеры необходимы, в частности, при разработке таких САПР, как Сириус, ТокКТЭ и других.

Для достижения этой цели поставлены следующие **задачи**:

1) проанализировать входные данные DXF, в соответствии с разрабатываемым ПО;

2) выявить структурно-содержательные особенности файлов формата DXF для последующей работы с разрабатываемым ПО;

3) проанализировать возможности применения разных ЯП для разработки ПО;

4) разработать алгоритмы;

5) разработать ПО;

6) провести анализ экономической целесообразности разрабатываемого проекта (сравнительная экономическая эффективность).

Объект исследования — формат файлов обмена графической информацией DXF.

Предмет исследования — проектирование ПО для конвертации файлов и формата DXF в форматы TXT, SVG, JSON.

Теоретическая и практическая значимость работы

а) исследование универсально, то есть подход к разработке, используемый в данной работе, может быть реализован для создания ПП по конвертированию в другие файловые форматы;

б) разработанное ПО можно внедрять в существующие САПР, работающие с форматом файлов DXF;

в) САПР, использующие разработанный модуль по конвертации файлов с графической информацией, получают независимость от использования аналогичного ПО зарубежного производства;

г) положенные в основу алгоритмы и написанное ПО имеет открытый доступ в сети Интернет и имеет перспективу развития в полноценный модуль импорта/экспорта файлов;

д) коммерческая выгода заказчика.

1 Определение области применения и главных требований к разрабатываемым конвертерам

В данном разделе поясняется область применения разрабатываемого ПО, а также, указываются требования к разрабатываемым конвертерам.

1.1 Область применения ПО «Primiview»

Конвертер в TXT по типу DXF применяется для контроля содержания необходимых (поддерживаемых) примитивов (объектов) в DXF-файле. Также, с помощью данного формата может производится расчёт длины траектории контура детали (обычно, в поперечном её сечении). Это может быть полезно при применении ПО в области лазерной резки с помощью станков с ЧПУ, а, в частности, в ПО «Сириус».

Конвертер из DXF в TXT в формате координат и радиуса применяется для автоматизированного технологического проектирования, для формирования УП. Получаемая в результате работы ПО информация о примитивах изображения контура детали используется для непосредственного составления УП, так как каждая последующая точка имеет не только плоские координаты, но и способ достижения этой точки (тип примитива: отрезок, если радиус равен нулю; дуга, если радиус ненулевой).

Конвертер в SVG-формат полезен для последующего формирования других типов файлов (например, DBS), а также, для компактного по объёму файла векторного представления контура детали.

Конвертер в JSON удобен для хранения информации об объектах контура детали. В этом формате работает и другое ПО разрабатываемой САПР для формирования УП для станков с ЧПУ.

В целом, разрабатываемый набор конвертеров (модуль экспорта) представляет собой цельный ПП, сочетающий в себе набор необходимых разработчику УП начальных функций для автоматизированного технологического проектирования. Это ПО может быть интегрировано в разные

ПП, так как по сути универсально в своём применении (используется в области 2D-резки, токарной обработке).

1.2 Главные требования ко входному файлу

Так как данная работа нацелена на создание ПО для обработки геометрической информации 2D-объектов специального типа, то конвертироваться из DXF-файлов должна не вся информация, содержащаяся в них. В первую очередь, заказчиком работы было определено, что на входе будет подаваться 2D-контур деталей типа «Втулка», то есть тел вращения. Так как сконвертированная геометрия данных объектов в последующем предполагает разработку УП для токарных станков с ЧПУ, то геометрическая информация должна содержать определённый набор геометрических примитивов, с которым может работать система исполнительных органов станков с ЧПУ. Этот набор ограничивается тем, что исполнительные органы станков с ЧПУ способны перемещаться либо с помощью **линейной**, либо с помощью **круговой интерполяции**. Из этого следует, что для корректной работы САПР, для которых предназначаются разрабатываемые конвертеры, геометрия в DXF-файле на входе конвертеров должна состоять из, как минимум одного из представленных далее примитивов:

а) линия (отрезок),
б) полилиния,
в) дуга,
г) окружность.

Остальные типы геометрии, реализуемой в формате DXF, такие как *эллипс*, *сплайн*, будут игнорироваться ПО.

1.3 Описание формата DXF-файлов

На вход разрабатываемому конвертеру подаётся файл формата DXF. Формат DXF представляет собой совокупность данных с тегами всей информации, содержащейся в файле чертежа AutoCAD. Тегированные данные означают, что каждому элементу данных в файле предшествует целое число, называемое групповым кодом. Значение группового кода указывает, какой тип данных имеет следующий элемент. Это значение также указывает смысл элемента данных для данного типа объекта. Практически вся указанная пользователем информация в файле чертежа может быть представлена в формате DXF [**?**]. В DXF файлах, в зависимости от их содержания, существуют сущности, представляющие для нас интерес. Среди них следующие:

а) LINE (Линия),
б) LWPOLYLINE (Полилиния),
в) ARC (Дуга),
г) CIRCLE (Окружность),
д) INSERT (Вставка).

Как уже и было отмечено, существуют и другие примитивы (ELLIPSE, SPLINE и др.), однако, основываясь на конкретных целях заказчика по возможности применения выходных файлов для генерации УП, ПП проектируется только с указанными примитивами и сущностями DXF.

Рассмотрим каждую из сущностей подробнее.

LINE. Рассмотрим тэги сущности *Линия*, необходимые для её реального отображения (см. табл. 1.1).

Таблица 1.1 — Рассматриваемые групповые коды сущности LINE

Групповой код	Описание
39	Толщина (необязательный; по умолч. = 0)
10	Начальная точка (в с.к. объекта) DXF: значение X
20, 30	DXF: Y и Z значения начальной точки (в с.к. объекта)
11	Конечная точка (в с.к. объекта) DXF: значение X
21, 31	DXF: Y и Z значения конечной точки (в с.к. объекта)

LWPOLYLINE. Рассмотрим тэги сущности *Полилиния*, необходимые для её реального отображения (см. табл. 1.2).

Таблица 1.2 — Рассматриваемые групповые коды сущности POLYLINE

Групповой код	Описание
70	«Флаг» полилинии (бит-закодировано); по умолч. = 0; 1 – закрыта
39	Толщина (необязательный; по умолч. = 0)
10	Координаты вершин (в с.к. объекта), множественные вхождения; по одному вхождению для каждой вершины DXF: значение X
20	DXF: значение Y координат вершин (в с.к. объекта), множественные вхождения; по одному вхождению для каждой вершины
42	*Bulge*. Выпуклость (множественные вхождения - для каждой вершины), (необязательно; по умолч. =0)

ARC. Рассмотрим тэги сущности *Дуга*, необходимые для её реального отображения (см. табл. 1.3).

Таблица 1.3 — Рассматриваемые групповые коды сущности ARC

Групповой код	Описание
39	Толщина (необязательный; по умолч. = 0)
10	Центр дуги (в с.к. объекта) DXF: значение X
20, 30	DXF: Y и Z значения центра дуги (в с.к. объекта)
40	Радиус
50	Начальный угол
51	Конечный угол

CIRCLE. Рассмотрим тэги сущности *Окружность*, необходимые для её реального отображения (см. табл. 1.4).

Таблица 1.4 — Рассматриваемые групповые коды сущности CIRCLE

Групповой код	Описание
39	Толщина (необязательный; по умолч. = 0)
10	Центр дуги (в с.к. объекта) DXF: значение X
20, 30	DXF: Y и Z значения центра дуги (в с.к. объекта)
40	Радиус
50	Начальный угол
51	Конечный угол

INSERT. Данная сущность представляет собой вставку блоков с геометрией. Её необходимо рассматривать, так как геометрия может быть вложенной и, таким образом, не видна обзорщиком сущностей, так как вложена. У этой сущности поиск информации по тэгам в программе не потребуются.

1.4 Параметр «bulge»

Особый интерес представляет параметр *bulge* (выпуклость) для каждой из вершин полилинии. Чтобы понять сущность данного параметра, который представляет собой некоторую степень кривизны дуги окружности между двумя точками, необходимо сначала разобраться с геометрией дуг.

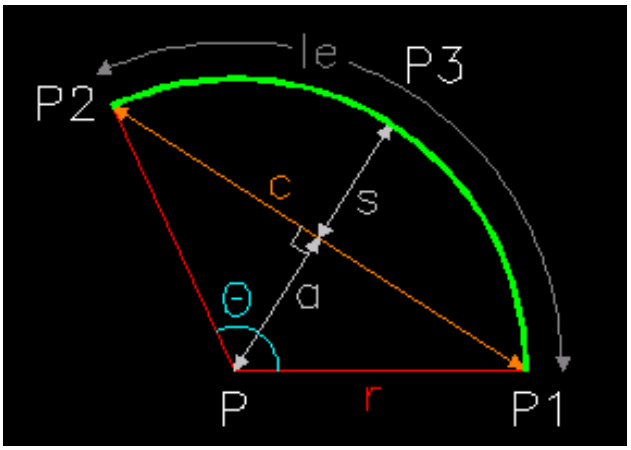

Рисунок 1.1 — Геометрия дуги окружности

Так как дуга окружности описывает часть этой окружности, то она и обладает всеми атрибутами данной окружности (см. рис. 1.1). Среди них:

— Радиус (r) — радиус дуги такой же, как и у окружности;
— Центр (P) — тот же, что и у окружности;
— Центральный угол (Θ) — в окружности равен 360°;
— Длина дуги (le) — является частью периметра (длины) окружности.

Для дальнейшей работы с геометрией дуг примем, также, следующие специфичные атрибуты:

— Начальная и конечная точка ($P1, P2$) — это «вершины» дуги. Хотя иногда и целесообразно говорить о конкретных точках, не лежащих на концах дуги;

— Длина хорды (c) — у дуг и окружностей можно провести бесконечное количество хорд, но для нас интерес представляет только хорда, проходящая через её вершины;

— Середина дуги ($P3$) — точка, делящая дуги с данными вершинами на две, равные по длине, дуги;

— Апофема (a) — это отрезок, вершинами которого являются середина дуги и её центр. Апофема перпендикулярна хорде;

— Высота дуги (s) — это отрезок, проведённый из середины дуги перпендикулярно к хорде.

Кроме самой себя, дуга может, также, и описывать другие геометрические формы: круговой сегмент и сектор. Обе геометрические формы включают в себя все вышеперечисленные атрибуты, однако для выведения формулы параметра *bulge* (выпуклости), потребуется рассмотрение только кругового сектора.

В документации AutoCAD [**?**] выпуклостью называется тангенс четверти угла дуги между выбранной вершиной и следующей вершиной в списках вершин полилиний. Отрицательность параметра *bulge* указывает на то, что дуга отрисовывается по часовой стрелке от выбранной вершины к следующей. Выпуклость, равная нулю — прямой сегмент, выпуклость, равная единице — половина окружности.

Проблема «расшифровки» атрибутов дуги для дальнейших манипуляций с ней заключается в том, что входными данными являются только координаты вершин и рассматриваемый параметр — *bulge*.

В самом деле, взяв арктангенс от параметра *bulge* и умножив его на 4, легко получить центральный угол, на который опирается рассматриваемая дуга. Результат получен в радианах. Для перевода значения в градусы, необходимо умножить это значение на π и разделить на 180°.

Для вывода данной зависимости, рассмотрим дугу окружности (см. рис. 1.2).

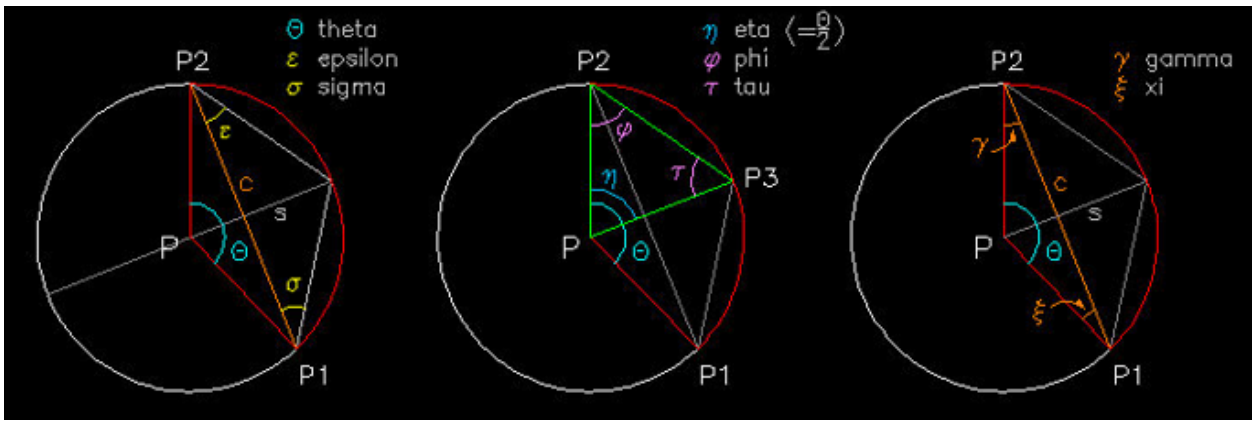

Рисунок 1.2 — Дуга окружности с проведённой хордой и углами при ней

Если провести к углу Θ биссектрису, то получится синий угол η. В итоге, мы получим равнобедренный треугольник (зеленый), в котором углы φ и τ равны. Поскольку сумма углов в треугольнике всегда равна $180°$ градусам, мы теперь знаем, что углы φ и τ равны следующему (1.1):

$$\varphi = \tau = \frac{(180° - \frac{\Theta}{2})}{2} \Rightarrow \varphi = 90° - \frac{\Theta}{4} \tag{1.1}$$

Теперь посмотрим на хорду c, проведённую от $P1$ до $P2$. Вместе с красными катетами угла Θ она тоже образует равнобедренный треугольник, а значит, $\gamma = \xi$. Угол при вершине треугольника $P - P1 - P2$ — это центральный угол Θ, поэтому γ и ξ вычисляются следующим образом (1.2):

$$\gamma = \xi = \frac{180° - \Theta}{2} \Rightarrow \gamma = 90° - \frac{\Theta}{2} \tag{1.2}$$

Таким образом, желтый угол ε должен быть равняться разнице между фиолетовым углом φ и оранжевым углом γ. Другими словами, ε — это четверть центрального угла Θ (1.4):

$$\varepsilon = (90° - \frac{\Theta}{4}) - (90° - \frac{\Theta}{2}) \Rightarrow \varepsilon = \frac{\Theta}{2} - \frac{\Theta}{4} = \frac{\Theta}{4} \tag{1.3}$$

Параметр *bulge* (выпуклость) описывает, насколько дуга «выпирает» из вершин, то есть насколько велика высота дуги (s) (или расстояние

от $P3$ до $P4$). Высота образует катет прямоугольного треугольника с углом, равным четверти центрального угла (см. желтый треугольник $P-P2-P3$ на рис. 1.3), и поскольку тангенс описывает отношение между катетами в прямоугольном треугольнике, легко описать геометрию с помощью этого одного угла (1.4):

$$\frac{\sin(\varepsilon)}{\cos(\varepsilon)} = \tan(\varepsilon) \tag{1.4}$$

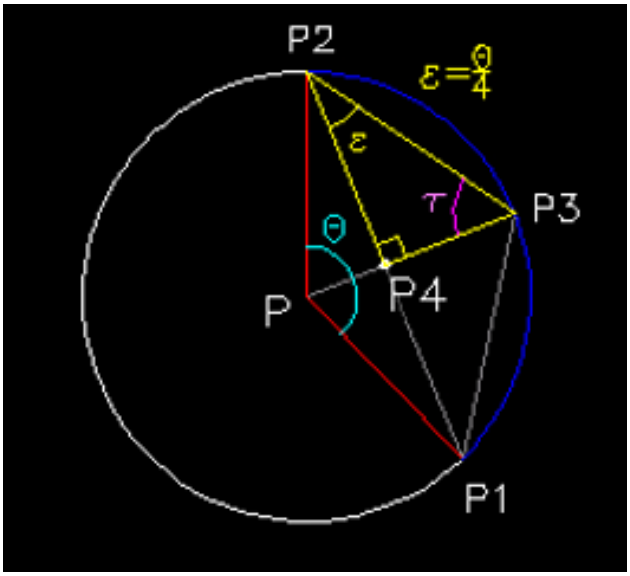

Рисунок 1.3 — Связь угла ε с центральным углом

Мы, также, могли бы найти тангенс угла ε, просто разделив противолежащий катет на смежный катет — что означает высоту дуги s, делённую на половину длины хорды c, — но не зная s и уже имея тангенс ε, полезнее найти s (1.5):

$$s = \frac{c}{2} \cdot \tan(\varepsilon) \tag{1.5}$$

Примем

$$\tan(\varepsilon) = bulge \tag{1.6}$$

Тогда

$$s = \frac{c}{2} \cdot bulge \tag{1.7}$$

Таким образом, радиус дуги может быть найден следующим образом (1.8):

$$r = \frac{(\frac{c}{2})^2 + s^2}{2s} \tag{1.8}$$

Знак той или иной выпуклости важен для определения дуги относительно вершин. Если выпуклость положительна, это означает, что дуга идёт против часовой стрелки от начальной вершины до конечной вершины. Если выпуклость отрицательна, это означает, что дуга идет, наоборот — по часовой стрелке.

Поэтому все приведенные выше формулы должны касаться абсолютного значения выпуклости, а не фактического значения, иначе можно получить отрицательный радиус.

Итак, поняв, что $bulge = tan(\frac{\Theta}{4})$, в согласовании с документацией AutoCAD [**?**] примем, что *bulge* положителен, когда при передвижении от начальной точки дуги к конечной движение происходит против часовой стрелки.

Ясно, что когда $\Theta = 0$, то и $bulge(\Theta) = 0$. Для углов в 180° условно принимается, что $bulge(\Theta) = \pm 1$. В случае, когда $\Theta = 90°$, получим следующее (1.9):

$$bulge(90°) = \tan(\frac{90°}{4}) = \tan(\frac{\pi}{8}) \tag{1.9}$$

Используя зависимость для тангенса половинного аргумента (1.10):

$$\tan(\frac{\alpha}{2}) = \pm\frac{\sin(\frac{\alpha}{2})}{\cos(\frac{\alpha}{2})} = \pm\frac{2\sin^2(\frac{\alpha}{2})}{2\sin(\frac{\alpha}{2})\cos(\frac{\alpha}{2})} = \pm\frac{1-\cos(x)}{\sin(x)} \tag{1.10}$$

Для $\alpha = \frac{\pi}{8}$ получим (1.11):

$$bulge(90^\circ) = \tan(\frac{\pi}{8}) = \pm\frac{1-\cos(\frac{\pi}{4})}{\sin(\frac{\pi}{4})} = \pm\frac{1-\frac{\sqrt{2}}{2}}{\frac{\sqrt{2}}{2}} = \pm\frac{1-\frac{1}{\sqrt{2}}}{\frac{1}{\sqrt{2}}} = \pm(\sqrt{2}-1) \tag{1.11}$$

В результате, математические данные совпадают с документацией AutoCAD [?] и гласят, что

а) $bulge = 0$ для отрезка прямой,

б) $bulge = \pm 1$ для дуги в 180° (половина окружности),

в) $bulge = \pm(\sqrt{2}-1) \approx 0.41421...$ для четвертей окружностей, когда угол раствора дуги равен 90°.

2 Разработка программного обеспечения «DXF Primiview»

В данном разделе описывается процесс создания ПП для обработки геометрической информации 2D-объектов специального типа. Обосновывается выбор языка программирования, разрабатываются алгоритмы для разработки ПО, описывается процесс разработки конвертеров.

2.1 Выбор языка программирования

Для написания программы выбран язык программирования Python версии 3.11. Выбор языка программирования обоснован несколькими факторами.

Библиотеки. Для создания описанного ПП необходимо привлечение различных библиотек. Кроме стандартной библиотеки, с Python можно использовать множество прикладных библиотек, несколько из которых будут описаны в следующих частях работы. Специфичные библиотеки, позволяющие «читать» DXF-файл и обрабатывать содержимое внутри него, написаны не для каждого ЯП. Библиотека «eazy dxf» для Python позволяет быстро и удобно выполнять данные операции.

Кроссплатформенность. Большинство программ, написанных на Python, выполняются корректно на всех основных платформах. Перенос программы между операционными системами реализуется простым копированием кода. Кроме того, в процессе разработки ПО, для реализации пользовательского интерфейса используется набор расширений Qt, который тоже работает на таких платформах, как Linux и другие UNIX-подобные ОС, macOS и Windows.

Скорость и удобство разработки. Удобочитаемсть, ясность и высокое качество этого языка позволяют повысить производительность разработчика во много раз, сравнивая, например, с компилирующими или

строго типизированными языками, такими как C, C++ и Java. Объём программного кода на языке Python обычно составляет треть или даже пятую часть эквивалентного программного кода на языке C++ или Java. Кроме того, при запуске программы, написанной на ЯП Python минуются длинные этапы компиляции и связывания, необходимые в некоторых других ЯП, что, также, увеличивает производительность труда программиста [**?**].

2.2 Разработка алгоритмов

Для создания программного обеспечения необходимо сначала разработать концепцию функционирования программы, опираясь на её назначение, на основные её функции. Когда определены модули, блоки и функциональные части ПО, можно приступать к разработке его на выбранном ЯП.

2.2.1 Структура программы

Основываясь на цели, поставленной во введении, разрабатываемая утилита должна принимать на входе DXF-файл, то есть открывать его и обрабатывать его содержимое. Для проверки правильности обработанных данные, то есть, для верификации содержимого DXF-файла, программа должна визуализировать для пользователя обработанное. После верификации обработанных данных и, соответственно, подтверждения соответствия их исходным, пользователю должна предоставляться возможность конвертировать эти данные в какой-либо из предлагаемых форматов. За это отвечает модуль экспорта, который, в свою очередь, подразделяется на четыре модуля, отвечающие за преобразование данных в различные форматы. Среди них следующие:

а) Модуль экспорта в TXT-файл, где данные будут представлены в такой же форме, как и в оригинальном DXF-файле, за исключением того,

что содержаться в нём будут только поддерживаемые сущности (LINE, POLYLINE, ARC, CIRCLE).

б) Модуль экспорта в TXT-файл, в котором поддерживаемые сущности будут представлены сочетанием двух строк, первая из которых — начальная точка примитива, вторая — конечная. Вторая строка содержит в себе радиус скругления примитива, переходящего из первой точки во вторую.

в) Модуль экспорта в формат SVG, который, при открытии, векторно отображает информацию в нём.

г) Модуль экспорта JSON-формат. В данном формате, по подобию формату TXT (x, y, r), содержаться точки, олицетворяющие начало и конец того или иного примитива. В данном случае информация в файле тэгированная, что означает, что в дальнейшем несложно будет получить желаемые куски данных из потенциально объёмного JSON-файла путём обращения по желаемому тэгу.

Схему, отображающую основное содержание разрабатываемого ПО, можно наблюдать на рисунке 2.1.

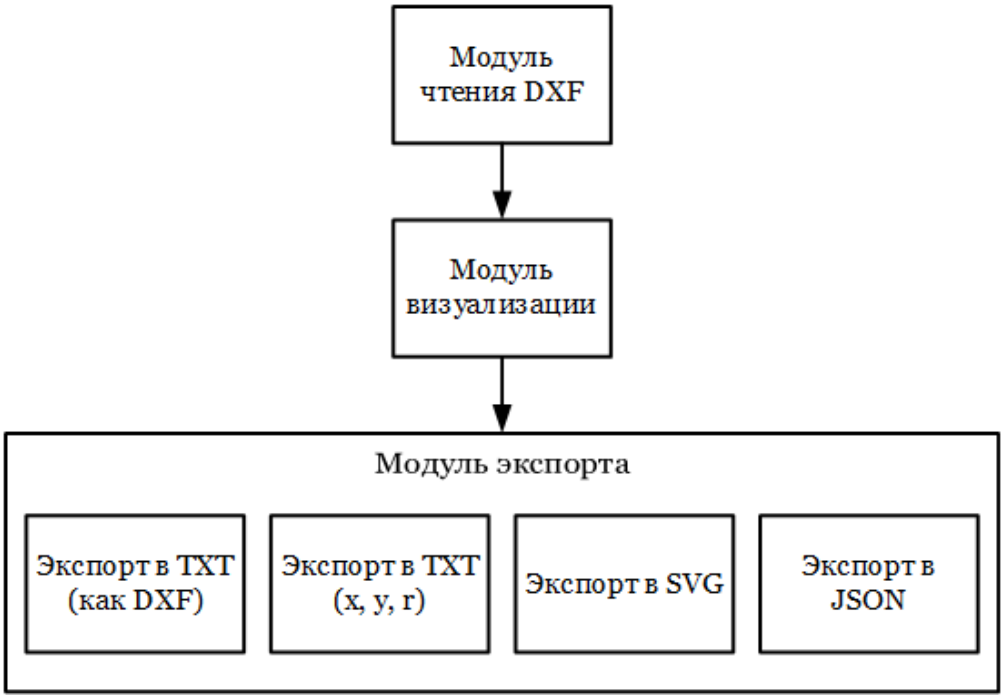

Рисунок 2.1 — Принципиальная структура ПО «Primiview»

2.2.2 Алгоритмы модулей программы

В данном пункте приводится описание алгоритмов ПО «Primiview» в виде псевдокода.

Алгоритм чтения DXF-файлов. Алгоритм 1 показывает схему работы процесса извлечения поддерживаемых ПО Primiview примитивов из выбранного DXF-файла.

На первой итерации осуществляется разбиение блоков, в которых могут быть «спрятаны» остальные сущности. В случае, если пропустить данный этап, то объекты, находящиеся внутри блоков не будут видны библиотекой *ezdxf*, которая используется для чтения DXF-файлов.

После «разрушения» всех блоков примитивы становятся «видимыми». Далее запускается цикл, итерирующий объекты пространства объектов модели (*modelspace*). В случае совпадения объекта с одной из поддерживаемых сущностей, она сохраняется в список соответствующих объектов в оперативной памяти программы. Списком далее будет называться

изменяемый упорядоченный тип данных, представляющих собой последовательность элементов, разделённых между собой запятой и заключённых в квадратные скобки. Данный тип данных используется в ЯП Python, поэтому для соблюдения общности, термин будет применяться и в части 2.3 этого раздела.

Примем ряд условных обозначений:

msp — пространство объектов модели (modelspace);

$\dashrightarrow$ — запись объекта в множество.

Исходные данные: путь к DXF-файлу

Результат: массивы примитивов и информации о них в оперативной памяти программы

```
1  инициализация;
2  цикл ∀ сущн. типа Вставка (INSERT) ∈ msp выполнять
3  |  разбиение сущности
4  конец
5  цикл ∀ сущн. ∈ msp выполнять
6  |  если сущ явл. линией тогда
7  |  |  сущн. --→ множ. линий
8  |  конец
9  |  иначе если сущн. явл. дугой тогда
10 |  |  сущн. --→ множ. дуг
11 |  конец
12 |  иначе если сущн. явл. окружностью тогда
13 |  |  сущн. --→ множ. окружностей
14 |  конец
15 |  иначе если сущн. явл. полилинией тогда
16 |  |  сущн. --→ множ. полилиний
17 |  конец
18 конец
```

Алгоритм 1 — Сохранение поддерживаемых примитивов из DXF в оперативную память программы

Алгоритм записи примитивов в TXT (как DXF). Принцип данного алгоритма (см. алгоритм 2) основан на открытии созданного TXT-файла, а после — перебора прочитанных из DXF примитивов и записи из каждого из них необходимой информации в открытый для редактирования TXT-файл.

Записи в текстовом файле должны выглядеть следующим образом (см. листинг 2.1):

Листинг 2.1 — Пример содержания TXT-файла (как DXF)

```
1    LINE(#01)
2    0.1  0.1
3    0.1  0.1
```

Исходные данные: путь к имя.txt

Результат: имя.txt (как DXF)

```
1  инициализация;
2  цикл ∀ LINE ∈ множ. линий выполнять
3  |  записать в файл: атрибут сущности, x0, y0, x1, y1; перевести
   |   курсор на новую строку
4  конец
5  цикл ∀ ARC ∈ множ. дуг выполнять
6  |  записать в файл: атрибут сущности, x0, y0, x1, y1, r;
   |   перевести курсор на новую строку
7  конец
8  цикл ∀ CIRCLE ∈ множ. окруж. выполнять
9  |  записать в файл: атрибут сущности, xc, yc, r; перевести
   |   курсор на новую строку
10 конец
11 цикл ∀ LWPOLYLINE ∈ множ. полилин. выполнять
12 |  записать в файл атрибут сущности;
13 |  цикл ∀ точки LWPOLYLINE выполнять
14 |  |  цикл ∀ координаты x, y выполнять
15 |  |  |  записать в файл значение координаты
16 |  |  конец
17 |  конец
18 |  перевести курсор на новую строку
19 конец
```

Алгоритм 2 — Запись примитивов в TXT (как DXF)

В алгоритме 2 x_0, y_0 — координаты начала примитива; x_1, y_1 — координаты конца примитива; x_c, y_c — координаты центра окружности; r — радиус дуги или окружности.

Алгоритм записи примитивов в TXT (x,y,r). Алгоритм 4 призван, так же как и в прошлом случае, в открытый только что созданный

текстовый файл записать информацию о примитивах, которые были прочитаны из выбранного DXF-файла.

Записи в текстовом файле должны выглядеть следующим образом (см. листинг 2.2):

Листинг 2.2 — Пример содержания TXT-файла (x, y, r)

```
1    1.52  1.86  0
2    1.12  2.08  0
3    1.16  2.04  4.0
4    ...
```

Полилиния может содержать в себе, как отрезки, так и дуги. В объектах LWPOINT сущности LWPOLYLINE степень искривления показывает параметр *bulge*, суть которого подробно описана в разделе 1.3. Так как желаемый формат вывода информации о примитивах содержит именно радиус примитива, а не параметр искривления, то необходимо удобно получить радиус из *bulge*.

Для этого воспользуемся уже выведенной зависимостью [1] и применим её в принятых обозначениях (2.1):

$$R = |bulge + \frac{1}{bulge}| \cdot \frac{|A - Z|}{4}, \tag{2.1}$$

где A — начальная точка;

Z — конечная точка.

Примем, что $A(x_0,y_0), B(x_1,y_1)$. Тогда $AZ(x_1 - x_0; y_1 - y_0)$. Таким образом, длина вектора через декартовы координаты (2.2):

$$|A - Z| = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{2.2}$$

Исходные данные: текущая точка, следующая точка

Результат: радиус сегмента полилинии

1 инициализация;

2 **если** $bulge$ *(текущей точки)* $= 0$ **тогда**

3 $\quad r = 0$

4 **конец**

5 **иначе**

6 $\quad r = |bulge + \frac{1}{bulge}| \cdot \frac{\sqrt{(x_{nexP} - x_{prevP})^2 + (y_{nexP} - y_{prevP})^2}}{4}$

7 **конец**

Алгоритм 3 — Вычисление радиуса сегмента полилинии

Примем условные обозначения:

$\rightsquigarrow$ — запись в файл;

$newline$ — перевод на новую строку.

Исходные данные: путь к имя.txt

Результат: имя.txt (x,y,r)

1 инициализация; **цикл** $\forall$ *LINE* $\in$ *множ. линий* **выполнять**

2 | $x_0 \quad y_0 \quad 0 \rightsquigarrow$ имя.txt; $\quad x_1 \quad y_1 \quad 0 \rightsquigarrow$ имя.txt

3 **конец**

4 **цикл** $\forall$ *ARC* $\in$ *множ. дуг* **выполнять**

5 | $x_0 \quad y_0 \quad 0 \rightsquigarrow$ имя.txt; $\quad x_1 \quad y_1 \quad r \rightsquigarrow$ имя.txt

6 **конец**

7 **цикл** $\forall$ *CIRCLE* $\in$ *множ. окруж.* **выполнять**

8 | $x_c + r \quad y_c \quad 0 \rightsquigarrow$ имя.txt; `/* первая половина окруж. */`

9 | $x_c - r \quad y_c \quad r \rightsquigarrow$ имя.txt

10 | $x_c - r \quad y_c \quad 0 \rightsquigarrow$ имя.txt; `/* вторая половина окруж. */`

11 | $x_c + r \quad y_c \quad r \rightsquigarrow$ имя.txt

12 **конец**

13 **цикл** $\forall$ *POLYLINE* $\in$ *множ. полилиний*

14 *prevPoint=None ;* `/* предыдущая точка */`

15 **выполнять**

16 | **цикл** *LWPOINT* $\in$ *множ. точек. полилинии* **выполнять**

17 | | **если** *prevPoint*$\neq$*None* **тогда**

18 | | | r = алгоритм 3 (prevPoint, LWPOINT)

| | | $x(prevP) \quad y(prevP) \quad 0 \rightsquigarrow$ имя.txt

19 | | | $x(lwpoint) \quad y(lwpoint) \quad r \rightsquigarrow$ имя.txt

20 | | **конец**

21 | | $prevPoint = lwpoint$

22 | **конец**

23 | **если** *контур замкнут* **тогда**

24 | | r = алгоритм 3 (prevPoint, LWPOINT)

25 | | $x(lwpoint_{\text{посл}}) \quad y(lwpoint_{\text{посл}}) \quad 0 \rightsquigarrow$ имя.txt

26 | | $x(lwpoint_{\text{перв}}) \quad y(lwpoint_{\text{перв}}) \quad r \rightsquigarrow$ имя.txt

27 | **конец**

28 **конец**

Алгоритм 4 — Запись примитивов в TXT (x, y, r)

Алгоритм записи примитивов в SVG. Формирования файла типа SVG отличается от предыдущих двух, так как этот формат представляет собой язык разметки, а значит, имеет правила синтаксиса, грамматики и т.д. Это расширение языка разметки XML, поэтому в начале, в преамбуле, указывается версия XML, кодировка символов и указание синтаксическому анализатору об игнорировании любых объявлений разметки в определении типа документа.

Листинг 2.3 — Первая строка SVG-файлов

```
1   <?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
```

Следующие две строки должны содержать определение типа документа (заголовок DOCTYPE), однако, данное объявление может оказаться источником ошибок при применении в браузере Mozilla Firefox. Поэтому вместо этого используется атрибут **baseProfile** со значением «full» внутри элемента <svg>.

Начиная с четвёртой строки объявляется корневой элемент <svg>:

Листинг 2.4 — Первая строка SVG-файлов

```
1   <svg version="1.1" width="100%" height="100%"
2   viewBox="102.1188828597992 -211.7921423734452
      50.000000000000014 26.0"
3   baseProfile="full"
4   xmlns="http://www.w3.org/2000/svg"
5   xmlns:xlink="http://www.w3.org/1999/xlink"
6   xmlns:ev="http://www.w3.org/2001/xml-events">
```

В листинге 2.4 присутствует необязательный элемент **viewBox**, который представляет собой параметр с четырьмя значениями, отделяемыми пробелами, определяющими квадратную рамку, в которой будет располагаться графика. Данный атрибут позволяет автоматически масштабиро-

вать изображение до размеров указанного контейнера, причём, без потери качества, так как графическая информация храниться и воспроизводится в векторном формате.

Первые два значение — минимальные координаты x и y рамки, в которой располагается изображение. Третье и четвёртое значения — соответственно, ширина и высота рамки, в которой находится изображение. Значения указываются в пикселях.

Таким образом, чтобы перенести данные из DXF в SVG, сначала определяются эти четыре значения. Алгоритмы для их определения: алгоритм 5, алгоритм 6 и алгоритм 7.

Исходные данные: списки примитивов с параметрами

Результат: список координат x, список координат y

1 инициализация; пустой список координат x, пустой список координат y **цикл** $\forall$ *LINE* $\in$ *множ. линий* **выполнять**

2 $\quad$ x_0,x_1 --→ список x

3 $\quad$ y_0,y_1 --→ список y

4 **конец**

5 **цикл** $\forall$ *ARC* $\in$ *множ. дуг* **выполнять**

6 $\quad$ $(x_c + r)$,$(x_c - r)$ --→ список x

7 $\quad$ $(y_c + r)$,$(y_c + r)$ --→ список y

8 **конец**

9 **цикл** $\forall$ *CIRCLE* $\in$ *множ. окруж.* **выполнять**

10 $\quad$ $(x_c + r)$,$(x_c - r)$ --→ список x

11 $\quad$ $(y_c + r)$,$(y_c + r)$ --→ список y

12 **конец**

13 **цикл** $\forall$ *LWPOLYLINE* $\in$ *множ. полилин.* **выполнять**

14 $\quad$ **цикл** $\forall$*точки в множ. LWPOINTS* **выполнять**

15 $\quad\quad$ x --→ список x

16 $\quad\quad$ y --→ список y

17 $\quad$ **конец**

18 **конец**

19 вернуть список координат x и список координат y

Алгоритм 5 — Вычленение координат изображения из DXF в отдельные списки

Исходные данные: список координат x, список координат y

Результат: x_{MIN}, y_{MIN}

1 инициализация;

2 использовать алгоритм 5

3 вернуть x_{MIN}, y_{MIN} из списков стандартными функциями сортировки ЯП

Алгоритм 6 — Поиск наименьших координат изображения из DXF

Исходные данные: список координат x, список координат y

Результат: ширина и высота рамки изображения

1 инициализация; использовать алгоритм 5

2 определить $x_{MIN}, x_{MAX}, y_{MIN}, y_{MAX}$ из списков стандартными функциями сортировки ЯП

3 ширина $= x_{MAX} - x_{MIN}$

4 высота $= y_{MAX} - y_{MIN}$

5 вернуть значения ширины и высоты

Алгоритм 7 — Поиск длины и высоты изображения из DXF

2.3 Разработка программного обеспечения

3 Экономическая часть

В данном разделе описаны экономические аспекты проекта по созданию ПО «Primiview» для обработки геометрической информации 2D-объектов специального типа.

3.1 Экономическое обоснование

Целью экономического обоснования проекта является представление разработанного ПП в качестве проекта для реализации на предприятии, что позволит провести планирование и корректировку последовательности работ (при необходимости).

3.1.1 Разработка проекта

Цель проекта — создание модуля конвертации форматов, как отдельного ПП, для внедрения в ПО для автоматизации технологического проектирования обработки деталей типа «Втулка» на станках с ЧПУ к 10.05.2023.

Численные критерии сравнения состояний системы клиента:

— сложность проектирования,
— сроки (время) проектирования,
— себестоимость проектирования,
— качество результатов проектирования.

Пояснения к критериям. Под сложностью проектирования понимается минимально-необходимая квалификация проектировщика для выполнения задач технологического проектирования.

Трудоёмкость проектирования, в данном случае, выражается в среднем времени создания (написания) одной УП.

Себестоимость проектирования оценивается не в расчёте на одну УП, а в рамках одного рабочего года. В себестоимость проектирования

входят такие элементы, как зарплата сотрудника, производящего проектирование, а также, цена годовой лицензии/контракта обслуживания CAM-системы для данного количества оборудования на предприятии.

В качестве численного критерия для оценки качества результатов проектирования принято среднее количество типов ошибок, корректировки по которым оператор станка с ЧПУ вносит после того, как автоматизированное проектирование выполнено.

Текущее состояние системы клиента: основное ПО по автоматизации технологического проектирования введено в эксплуатацию, базовые потребности системы по конвертации DXF-файлов работают.

По численным критериям:

— сложность проектирования: инженер-технолог II категории по квалификационному справочнику [2],
— сроки (время) проектирования: 3 часа,
— себестоимость проектирования: 1420000 руб. (см. раздел 3.2.2),
— качество результатов проектирования: 10 ошибок.

Целевое состояние системы клиента: усовершенствованная, более гибкая и универсальная версия этого ПО.

По численным критериям:

— сложность проектирования: инженер-технолог (без категории) по квалификационному справочнику [2],
— сроки (время) проектирования: максимум 1,5 часа,
— себестоимость проектирования: максимум 1 млн.руб.,
— качество результатов проектирования: максимум 5 типов ошибок.

Результатом проекта является созданные и готовый к работе ПП, на вход которому подаётся DXF, а на выходе создаются файлы форматов TXT, SVG и JSON.

Команда проекта сформирована из 10 человек, среди которых:

а) 1 владелец,

б) 2 программиста, среди которых:

б.1) 1 разработчик,

б.2) 1 тестировщик.

Владелец проекта организует работу остальной команды, проводит планирование проекта, оценку его экономической эффективности, контроль за выполнением подчинёнными задач проекта.

Программисты занимаются непосредственно созданием продукта проекта, то есть написанием ПО. Разработчики отвечают за написание программного кода по техническому заданию проекта. Тестировщики выполняют проверку работоспособности ПП, ищут и сообщают отделу разработчиков о найденных и необходимых к устранению ошибок и недочётов программы.

3.1.2 Дерево задач проекта

Целью данного этапа является построение иерархического дерева, включающего в себя последовательное разбиение общей цели проекта на подцели и задачи.

Первый уровень иерархии.

Главной целью проекта, как уже было сказано, является разработка программного обеспечения «Primiview» по конвертации файлов в формате DXF в форматы TXT, SVG, JSON. Данная цель в проекте единственная и находится на высшем уровне иерархии.

Второй уровень иерархии.

В целях определения и формализации цели, структуры и методов проекта, чтобы исключить неоднозначное их понимание и толкование исполнителями, первый этап, стоящий в иерархии на втором уровне, — это *формирование технического задания (ТЗ).*

При параллельном методе разработке, когда этапы проекта могут начинаться тогда, пока предыдущие ещё не закончились, следующим шагом данного уровня будет *разработка алгоритмов* программного обеспечения. Данный этап необходим, так как именно на нём ещё можно решить несостыковки в логической части программы, исправление которых на следующих этапах редко бывает возможным.

Параллельно с разработкой алгоритмов начинается этап непосредственно *разработки ПО*. Эти этапе происходят одновременно, так как они тесно взаимосвязаны, и, например, не зная на каком ЯП будет разрабатываться ПО, трудно будет рационально подобрать алгоритмы, отвечающие возможностям ЯП.

Завершающим этапом второго уровня является *сдача проекта* заказчику. Эта стадия может быть выполнена только при полном выполнении предыдущих стадий, если иное не было предварительно оговорено с заказчиком.

Третий уровень иерархии.

Этап формирования ТЗ подразделяется на следующие задачи:

а) Определение назначения ПО. Здесь формализуются цели и функции ПП. Впоследующем, они вносятся в ТЗ. Это смысл выполнения проекта, то, к чему стремится вся его команда, что хочет получить в итоге заказчик;

б) Исследование степени разработанности. На стадии предпроектного исследования выполняется проверка, существуют ли аналоги данного продукта в открытом доступе рынка. Если есть, то чем заказчика не устраивает их использование;

в) Требования к продукту. Определяются численные критерии, которым должен соответствовать результат проекта на этапе его сдачи;

г) Определение сроков. Всем участникам проекта необходимо знать, к какому сроку они должны выполнить определённый ранее объём работ. Сотрудникам, при согласовании ТЗ, необходимо оценить эти сроки, и в слу-

чае невозможности из соблюдения, просить корректировки и согласования с заказчиком более позднего выполнения задач;

д) Написание ТЗ. Завершающая стадия этапа формирования ТЗ — здесь собирается вся информация с предыдущих стадий и документируется согласно стандартам организации. ТЗ должно быть согласовано со всему участниками проекта.

Этап разработки алгоритмов подразделяется на следующие стадии:

а) Определение принципа работы ПО. Составляется принципиальная схема работы ПО, связь модулей, определяется предназначение каждого из модулей;

б) Разработка алгоритмов для модулей ПП. Происходит решение поставленных задач на уровне логики и математики, строится набор взаимосвязей алгоритмов. По возможности, рассматривается применение уже существующих универсальных (реже, специальный) алгоритмов.

Этап разработки ПО содержит такие стадии:

а) Выбор ЯП. Данная стадия подразумевает проведение сравнительного анализа существующих инструментов различных ЯП применительно к разрабатываемому ПП;

б) Определение структуры ПО. Эта стадия необходима для понимания разработчика функционала каждого из модулей программы. От этого зависит, на какие части (из каких файлов) будет состоять ПО;

в) Написание и отладка программного кода. Главная часть создания продукта. На этой стадии разработчики непосредственно пишут программный код и отлаживают его работу.

г) Тестирование ПО. Команда тестировщиков, также, пишет тестовый код, который проверяет разрабатываемое ПО на корректность работы его функционала.

Завершающий этап на втором уровне иерархии — сдача ПО, содержит следующие стадии:

а) Презентация. Команда проекта презентует результаты своей работы заказчику. Отчитывается по выполнению всех этапов, указанных в ТЗ;

б) Письменный отчёт. Команда проекта оформляет документально результаты своей деятельности для предоставления заказчику;

в) Обучение Сотрудников. При разработке нового ПП, команде разработчиков необходимо обучить персонал заказчика работе в новом ПО.

Четвёртый уровень иерархии.

Описание данного уровня иерархии приведено кратко для примера. В самом деле, каждая из стадий третьего уровня подразделяется на задачи четвёртого уровня.

Стадия исследования степени разработанности проблемы в этапе формирования ТЗ подразделяется на следующие задачи:

а) Исследование отечественного рынка аналогичных ПП. Задача состоит в поиске решений по аналогичным проектам в открытом доступе в рамках отчественного рынка;

б) Исследование зарубежного рынка аналогичных ПП. Данная задача отличается от предыдущей сложностью поиска аналогов (на зарубежном рынке(пространстве)), так как для проведения данного анализа необходим высокий уровень владения иностранным (английским) языком, а также, требуется знание достоверных ресурсов (источников) информации.

Описанные элементы иерархии сведены в иерархическое дерево (см. рисунок 3.1). Уровни иерархической структуры оформлены таким образом, что, чем конкретнее описаны элементы структуры, тем насыщеннее цвет заливки.

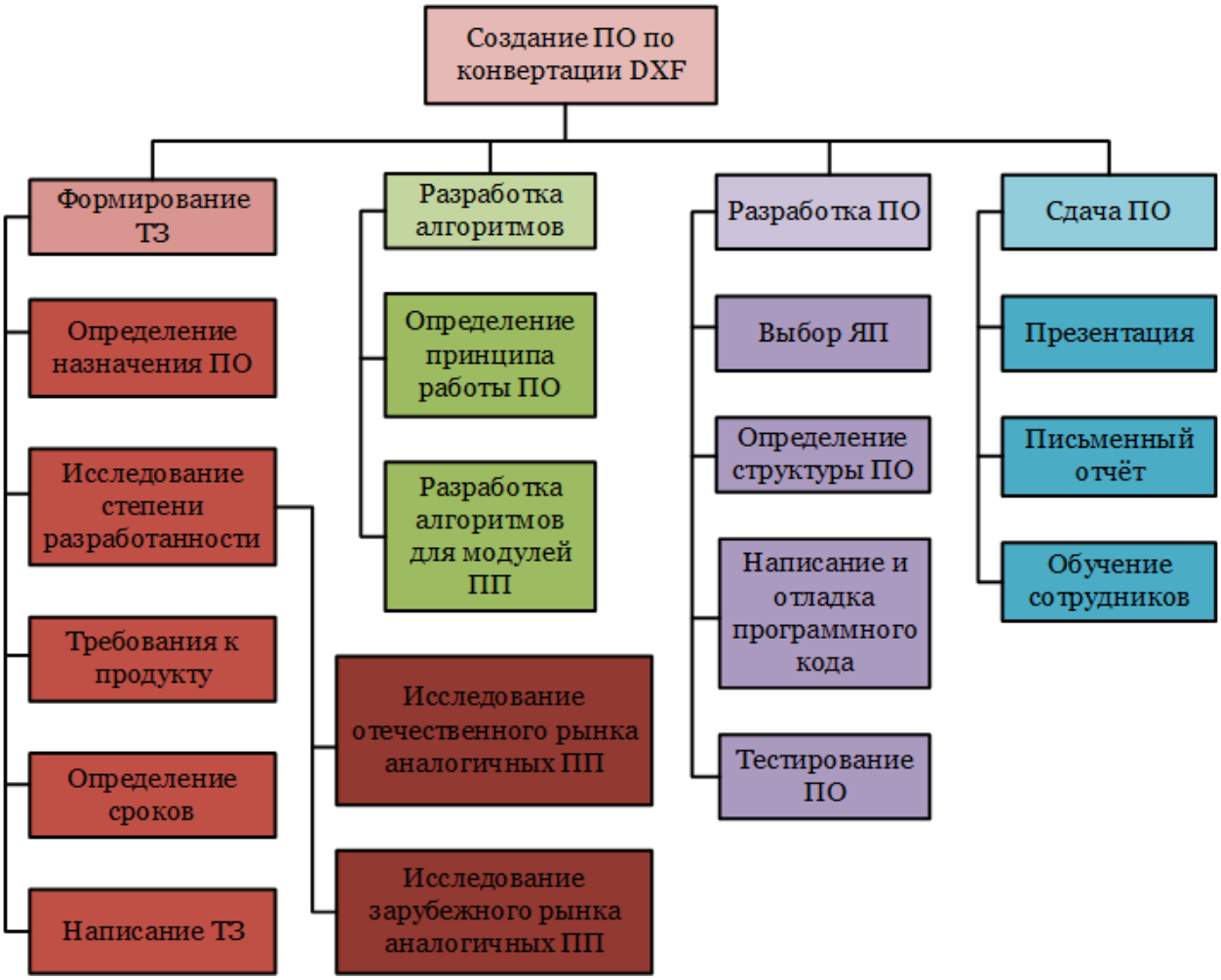

Рисунок 3.1 — Дерево задач проекта

3.1.3 Построение диаграмм проекта

На данном этапе строится диаграмма Ганта (англ. Gantt Chart, также, ленточная диаграмма, график Ганта) — это тип столбчатых диаграмм, использующийся для иллюстрации плана, графика работ проекта. Также, является методом планирования проекта. Эта диаграмма представляет собой отрезки (графические плашки), размещающиеся на горизонтальной шкале времени. Каждый отрезок соответствует отдельно задаче (подзадаче). Начало, конец и длина отрезка на шкале времени соответствуют началу, концу и длительности задачи. Диаграмма может использоваться для представления текущего состояния выполнения работ: часть прямо-

угольника, отвечающего задаче, заштриховывается, отмечая процент выполнения задачи; показывается вертикальная линия, отвечающая моменту «сегодня».

Рядом с самой диаграммой располагается таблица со списком работ, строки которой соответствуют отдельным задачам, отображённым на диаграмме, в то время как столбцы содержат дополнительную информацию о задаче.

Рисунок 3.2 — Диаграмма Ганта для проекта

3.2 Сравнительная экономическая эффективность

Расчеты сравнительной экономической эффективности капитальных вложений (инвестиций) применяются при сопоставлении нескольких возможных для осуществления вариантов инженерных решений: при решении задач по выбору взаимозаменяемых материалов, внедрению новых видов техники, модернизации оборудования, способов организации производственных процессов и т. п. То есть для оценки решений, которые являются альтернативными для обеспечения одинаковых конечных результатов деятельности. При этом конечные результаты (производство конкретной продукции с определенными характеристиками в заданном объеме) уже известны, есть необходимость определить, какой способ ее изготовления на том или ином этапе деятельности предприятия является более выгодным.

3.2.1 Исходные данные

Станкостроительное предприятие рассматривает заказ на создание программного обеспечения для своего оборудования (токарных станков с ЧПУ). Это ПО автоматизирует процесс создания управляющих программ для станков с ЧПУ, взамен работе инженера-технолога-программиста, который, обычно, берёт чертёж детали и либо вручную пишет УП, либо использует иностранные CAM-системы, предварительно создавая 3d-модель по выданному чертежу. В рамках данной (третьей) части ВКР будет рассматриваться инвестиционный проект (ИП) с точки зрения покупателя оборудования у предприятия, которое привлекло силы университета для создания описанного ПО. Сравниваются два варианта – покупка станков без ПО и, соответственно, с ним.

Сравнительная характеристика вариантов. Рассмотрим ситуацию с точки зрения покупателя оборудования рассматриваемого станкостроительного предприятия. Соберём основные данные в таблицу 3.1.

Таблица 3.1 — Сравнительная характеристика вариантов ИП

Вариант 1	Вариант 2
Покупка станка без ПО	Покупка станка вместе с ПО (CAM-системой) за большую цену
Наём во время этапа подготовки производства инженеров-технологов-программистов для написания УП	Привлекаются технологи из имеющегося штата сотрудников для выполнения дополнительных обязанностей по контролю создания УП с помощью купленного ПО
Время написания УП в три раза дольше, чем во втором варианте	Время написания УП в течение одного часа (в среднем)

Варианты рассматриваются с точки зрения потребителя оборудования. За сопоставляемые характеристики принимаются следующие:

— Объём производства (серийное),
— Частота создания УП в год (200 новых УП в среднем).

Выбор единичного периода времени. В качестве единичного периода времени для расчётов примем один год, так как на рассматриваемом предприятии-клиенте ситуация с производством каждый месяц практически не меняется. Также, большинство справочных величин ссылаются именно на годовой период, что тоже является подтверждением равномерной распределённости экономических характеристик внутри отдельно взятых месяцев.

Состав и описание капитальных вложений по вариантам. В капитальные вложения входят следующие величины:

— Цена станка с без ПО – 2 750 000 руб.,
— Цена встроенной CAM-системы на единицу оборудования– 70 000 руб.,

— Наладка полной группы станков – 20 000 руб.

Принятие решения по нормативному сроку окупаемости и его обоснование. Соответствует требованиям к сроку окупаемости дополнительных капитальных вложений, в данном случае – в токарный станок с ЧПУ.

Срок полезного использования оборудования – 10 лет.

Срок контракта на выпуск продукции с использованием данного оборудования – в рассматриваемой ситуации нет ограничений, токарная обработка постоянно проводится на предприятии.

Требования собственника, инвестора – предприятие установило желаемый срок окупаемости – 5 лет.

Следовательно, задаём Тн (нормативный срок окупаемости) равным 5 лет, так как временные рамки требований инвестора меньше срока полезного использования оборудования.

Определение состава затрат по вариантам (результат – перечень затрат). Корректировка затрат в соответствии с возможностями Методики сравнительной эффективности (включаем в расчет только различающиеся по альтернативам затраты). Деление затрат на переменные и постоянные. Формирование списка исходных данных для выполнения расчетов (см. таблицу 3.2).

Таблица 3.2 — Исходные данные для расчётов текущих затрат

	Вариант 1	Вариант 2
Переменные затраты (на единицу объема деятельности (одну УП))		
Зарплата технолога, руб	1500	1500
Время на написание одной УП, час	3	1

Продолжение на след. стр.

Продолжение таблицы 3.2

Постоянные затраты (на единицу оборудования)		
Цена годовой лицензии/контракта обслуживания CAM-системы, руб	50000	55000

3.2.2 Расчёты и анализ

Так как выбран нормативный срок окупаемости, равный одному году, то к нему будут приведены расчёты по приведённым затратам.

Исходные данные.

Среднее годовое количество УП на предприятии-покупателе станков

$$N = 200 \text{ шт};$$

Срок полезного использования оборудования:

$$T_{machinery} = 10 \text{ лет};$$

Требования инвестора по окупаемости ИП:

$$T_{inv} = 5 \text{ лет};$$

Принятая норма окупаемости:

$$T_n = T_{inv} = 5 \text{ лет};$$

Наладка полной группы станков:

$$CAM_{Term} = 20000 \text{ руб};$$

Цена встроенной CAM-системы на единицу оборудования:

$$CAM_2 = 70000 \text{ руб};$$

Цена станка без встроенной CAM-системы:

$$M = 2750000 \text{ руб};$$

Цена годовой лицензии/контракта обслуживания CAM-системы на единицу оборудования для вариантов 1 и 2, соответственно:

$$CAM_{1Perm} = 50000 \text{ руб};$$

$$CAM_{2Perm} = 55000 \text{ руб};$$

Среднее время написания одной УП:

$$t_1 = 3 \text{ часа};$$

$$t_2 = 1 \text{ час};$$

Почасовая оплата технолога:

$$Sal = 1500 \text{ руб};$$

Страховые сбора от заработной платы:

$$fees = 30\%$$

Количество покупаемых станков:

$$N_M = 5 \text{ шт};$$

Расчёт.

Себестоимость использования оборудования и ПО:

$$C_1 = Sal \cdot t_1 \cdot N \cdot (100\% + fees) + N_M \cdot CAM_{1Perm} = \\ = 1500 \cdot 3 \cdot 200 \cdot (100\% + 30\%) + 5 \cdot 50000 = 1420000 \text{ руб}; \tag{3.1}$$

$$C_2 = Sal \cdot t_2 \cdot N \cdot (100\% + fees) + N_M \cdot CAM_{2Perm} = \\ = 1500 \cdot 1 \cdot 200 \cdot (100\% + 30\%) + 5 \cdot 55000 = 665000 \text{ руб}; \tag{3.2}$$

Условно-годовая экономия (на себестоимости):

$$E = |C_1 - C_2| = |1420000 - 665000| = 755000 \text{ руб;} \tag{3.3}$$

Капитальные вложения предприятия-покупателя станков:

$$K_1 = M \cdot N_M + CAM_{Term} = 2750000 \cdot 5 + 20000 = 13770000 \text{ руб;} \tag{3.4}$$

$$\begin{gathered} K_2 = N_M \cdot (M + CAM_2) + CAM_{Term} = \\ = 5 \cdot (2750000 + 70000) + 20000 = 14120000 \text{ руб;} \end{gathered} \tag{3.5}$$

Дополнительные капитальные вложения:

$$K_{extr} = |K_1 - K_2| = |13770000 - 14120000| = 350000 \text{ руб;} \tag{3.6}$$

$$K_{extr} = N_M \cdot CAM_2 = 5 \cdot 70000 = 350000 \text{ руб. (проверка);} \tag{3.7}$$

Срок окупаемости дополнительных капитальных вложений:

$$T_{payback} = \frac{K_{extr}}{E} = \frac{350000}{755000} = 0{,}464 \text{ лет;} \tag{3.8}$$

Приведённые затраты по вариантам:

$$Z_1 = C_1 + \frac{1}{T_n} \cdot K_1 = 1420000 + \frac{1}{5} \cdot 13770000 = 4174000 \text{ руб;} \tag{3.9}$$

$$Z_2 = C_2 + \frac{1}{T_n} \cdot K_2 = 665000 + \frac{1}{5} \cdot 14120000 = 3489000 \text{ руб;} \tag{3.10}$$

Годовой экономический эффект:

$$E_{annual} = |Z_1 - Z_2| = |4174000 - 34890000| = 685000 \text{ руб;} \tag{3.11}$$

Минимальный годовой объём деятельности, при котором обеспечивается приведённый годовой экономический эффект:

$$\begin{gathered} N_{cr} = \frac{N_M \cdot CAM_{1Perm} - N_M \cdot CAM_{2Perm} - \frac{K_{extr}}{T_n}}{S \cdot t_2 \cdot (100\% + fees) - Sal \cdot t_1 \cdot (100\% + fees)} = \\ = \frac{5 \cdot 50000 - 5 \cdot 55000 - \frac{350000}{5}}{1500 \cdot 1 \cdot 100\% + 30\% - 1500 \cdot 3 \cdot 100\% + 30\%} = 24{,}359 \text{ шт;} \end{gathered} \tag{3.12}$$

По вычисленным в формулах 3.9, 3.10 затратам изобразим на графике (см. рисунок 3.3) границы целесообразности рассматриваемых вариантов.

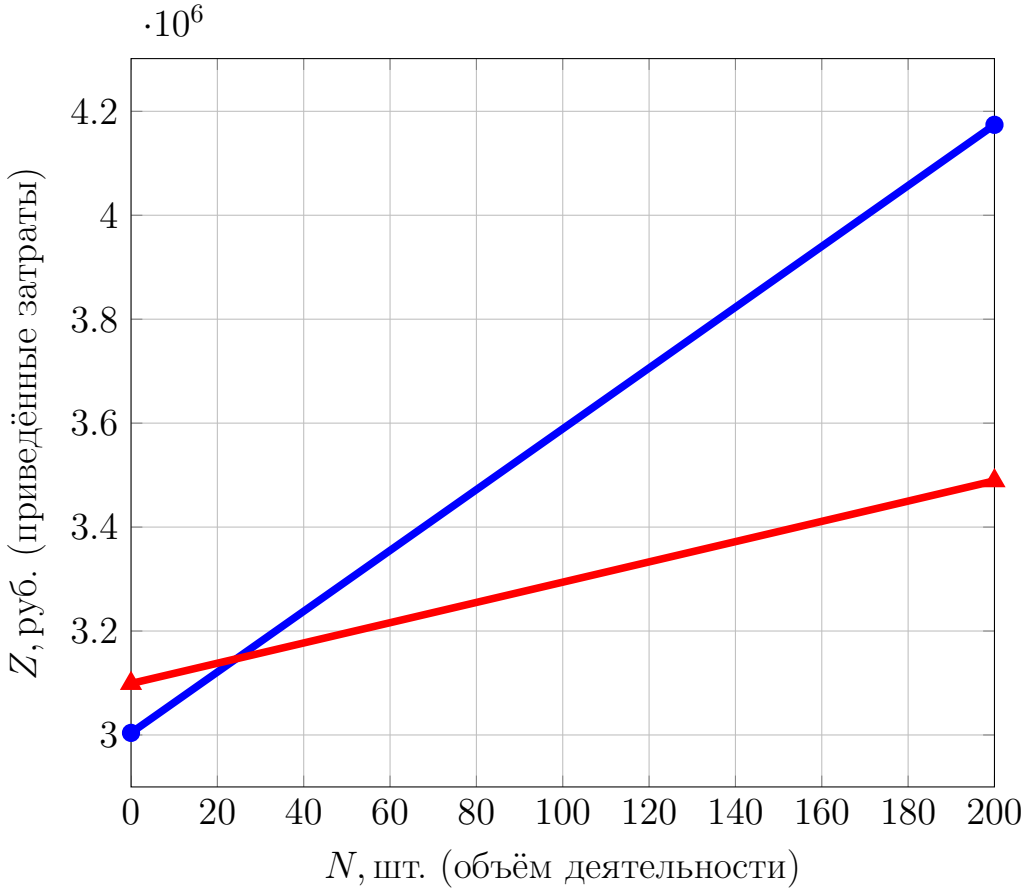

Рисунок 3.3 — Границы целесообразности рассматриваемых вариантов

Получив необходимые значения по критериям сравнения, сведём результаты в таблицу 3.3).

Таблица 3.3 — Сравнительная характеристика рассматриваемых вариантов по показателям эффективности

Наименование показателя	Ед. изм.	По вариантам:		Отклонения показателей
		Вариант без ПО	Вариант с ПО	
Годовой объем деятельности	шт.	200	200	-
Капитальные вложения, всего	руб.	13770000	14120000	350000
в том числе:				
Наладка станков	руб.	20000	20000	-
Цена станка (Вар. 2 + ПО)	руб.	13750000	665000	755000
Срок окупаемости дополнительных кап. вложений		0,464		
Приведённые затраты по вариантам	руб.	4174000	3489000	658000
Годовой экономический эффект			685000	

3.2.3 Выводы по результатам расчётов.

Так как на первых этапах расчёта по методу сравнительной эффективности ИП нельзя было сделать конкретный вывод по поводу целесообразности одного из предлагаемых вариантов по причине того, что по первому варианту себестоимость ИП была больше в сравнении со вторым, а капитальные вложения, соответственно, меньше, то расчёт был продолжен до момента вычисления расчётного срока окупаемости дополнительных капитальных вложений, а также расчёта приведённых затрат по каждому из вариантов.

Исходя из расчётов и построенного по ним графика, сделаем вывод, что, производя уже 25 УП за год, выгоднее становится вариант с ПО, так

как приведённые затраты для соответствующего количество производимых УП для этого варианта оказываются меньше.

Анализируя итоговые данные, выбираем для реализации второй вариант, то есть покупка оборудования вместе со встроенным ПО (CAM-системой), объясняя выбор тем, что расчётный срок окупаемости оказался намного меньше рассматриваемого нормативного срока окупаемости (0,4 и 5 лет, соответственно), а приведённые затраты по первому варианту оказались больше, чем по второму.

Действительно, экономия времени на создании УП нивелирует большие капитальные вложения на этапе инвестиционного периода УП.

Заключение

В результате проделанной работы стало ясно, что ничего не ясно...

Список использованных источников

1. *С.С., Уколов.* Разработка алгоритмов оптимальной маршрутизации инструмента для САПР управляющих программ машин листовой резки с ЧПУ / Уколов С.С. — Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, Екатеринбург: 2021. — P. 135.

2. *России, Минтруда.* Квалификационный справочник должностей руководителей, специалистов и других служащих / Минтруда России. — Минтруд РФ, М., 2002., 20.06.2002. — P. 293.