# Реферат

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

**??**

**?? ?? ??**

Объект ВКР — учебники издательства УрФУ.

Цель работы — разработать программное обеспечение для дешифро­ вания 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 — тех­ нологическая разработка управляющих программ для производства изде­ лий на станках с числовым программным управлением.

Значимость работы заключается в практической реализации ком­ мерческого заказа ООО «Униматик», создании специализированного про­ граммного обеспечения — модуля экспорта DXF-файлов в САПР.

# Содержание

[Введение](#_bookmark0) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1. [Определение области применения и главных требований к разра­](#_bookmark1) [батываемым конвертерам](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . 8
   1. [Область применения ПО «Primiview»](#_bookmark2) . . . . . . . . . . . . 8
   2. [Главные требования ко входному файлу](#_bookmark3) . . . . . . . . . . 9
   3. [Описание формата DXF-файлов](#_bookmark4) 10
   4. [Параметр «bulge»](#_bookmark9) 13
2. [Разработка программного обеспечения «DXF Primiview»](#_bookmark21) 19
   1. [Выбор языка программирования](#_bookmark22) 19
   2. [Разработка алгоритмов](#_bookmark23) 20
      1. [Структура программы](#_bookmark24) 20
      2. [Алгоритмы модулей программы](#_bookmark26) 22
   3. [Разработка программного обеспечения](#_bookmark39) 33
3. [Экономическая часть](#_bookmark40) 34
   1. [Экономическое обоснование](#_bookmark41) 34
      1. [Разработка проекта](#_bookmark42) 34
      2. [Дерево задач проекта](#_bookmark43) 36
      3. [Построение диаграмм проекта](#_bookmark45) 40
   2. [Сравнительная экономическая эффективность](#_bookmark46) 43
      1. [Исходные данные](#_bookmark47) 43
      2. [Расчёты и анализ](#_bookmark50) 46
      3. [Выводы по результатам расчётов.](#_bookmark55) 50

[Заключение](#_bookmark56) 52

[Список использованных источников](#_bookmark57) 53

# Глоссарий

В настоящей работе применяют следующие термины с соответству­ ющими определениями:

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

**Система автоматизированного проектирования** — организационно-техническая система, входящая в структуру проект­ ной 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 Государ­ ственного комитета СССР по стандартам от 26.06.87 № 2668).

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

ния:

# Обозначения и сокращения

В настоящей ВКР применяют следующие сокращения и обозначе­

**САПР** — Cистема 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 гра­

фической информацией между приложениями САПР, созданный фирмой Autodesk для системы AutoCAD в 1982 г.

**TXT** — Текстовый формат файлов, представляющий собой после­ довательность 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 дан­ ными, основанный на языке программирования JavaScript.

# Введение

**Актуальность темы исследования.** В настоящее время в рам­ ках проектов Уральского Федерального Университета (УрФУ) и Уральско­ го межрегионального научно-образовательного центра (УМНОЦ) происхо­ дит 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 дру­ гие форматы. Большая их часть лишь визуализирует графическую инфор­ мацию, представленную в том или ином DXF-файле, и конвертирует в наи­ более популярные форматы изображений (JPEG, PNG). Для использова­ ния конвертера на предприятии-заказчике необходим особый формат дан­ ных, в которые конвертируется DXF. Такие форматы не могут быть обес­ печены 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 соответству­ ет требованиям, предъявляемым к TXT-, SVG- и JSON-файлов, компании

«Unimatic». Ещё одним недостатком зарубежного ПО является ежемесяч­ ная плата разработчикам. Одной из задач современной Российской Феде­ рации является импортозамещение на производстве, поэтому разработка

собственного ПП увеличит независимость и самостоятельность предприя­ тий и компаний, пользующихся этим ПП.

**Цель работы** заключается в разработке программного обеспече­ ния для дешифрования информации из файлов типа DXF с геометрической информацией объектов специального типа, извлечении из них информации о геометрических примитивах, вывода полученной графической информа­ ции 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 век­ торной графики в формате SVG с выводом данных в исходном виде DXF, формирования текстового файла (JSON) с выводом данных в виде коорди­ нат 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 разраба­ тываемым ПО;
2. выявить структурно-содержательные особенности файлов формата DXF для последующей работы с разрабатываемым ПО;
3. проанализировать возможности применения разных ЯП для разра­ ботки ПО;
4. разработать алгоритмы;
5. разработать ПО;
6. провести анализ экономической целесообразности разрабатываемо­ го проекта (сравнительная экономическая эффективность).

**Объект исследования** — формат файлов обмена графической ин­ формацией 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, используе­ мый в данной работе, может быть реализован для создания ПП по конвер­ тированию в другие файловые форматы;

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

в) САПР, использующие разработанный модуль по конвертации фай­ лов с графической информацией, получают независимость от использова­ ния аналогичного ПО зарубежного производства;

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

д) коммерческая выгода заказчика.

# Определение области применения и главных требований к разрабатываемым конвертерам

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

## Область применения ПО «Primiview»

Конвертер в TXT по типу DXF применяется для контроля содержа­ ния 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 приме­ няется для автоматизированного технологического проектирования, для формирования УП. Получаемая в результате работы ПО информация о примитивах изображения контура детали используется для непосредствен­ ного 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 объ­ ёму 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 техноло­ гического проектирования. Это ПО может быть интегрировано в разные

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

## Главные требования ко входному файлу

Так как данная работа нацелена на создание ПО для обработки гео­ метрической информации 2D-объектов специального типа, то конвертиро­ ваться из DXF-файлов должна не вся информация, содержащаяся в них. В первую очередь, заказчиком работы было определено, что на входе бу­ дет 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 геометриче­ ская информация должна содержать определённый набор геометрических примитивов, с которым может работать система исполнительных органов станков с ЧПУ. Этот набор ограничивается тем, что исполнительные ор­ ганы 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 разрабатывае­ мые 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.

## 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 дан­ ные 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 указы­ вает смысл элемента данных для данного типа объекта. Практически вся указанная пользователем информация в файле чертежа может быть пред­ ставлена в формате DXF [**?**]. В DXF файлах, в зависимости от их содержа­ ния, 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 за­ казчика 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 ре­ ального отображения (см. табл. [1.1](#_bookmark5)).

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*, необходи­ мые для её реального отображения (см. табл. [1.2](#_bookmark6)).

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 вхож­  дения; по одному вхождению для каждой вершины DXF: значение X |
| 20 | DXF: значение Y координат вершин (в с.к. объекта), мно­  жественные вхождения; по одному вхождению для каждой вершины |
| 42 | *Bulge*. Выпуклость (множественные вхождения - для каж­  дой вершины), (необязательно; по умолч. =0) |

**ARC.** Рассмотрим тэги сущности *Дуга*, необходимые для её реаль­ ного отображения (см. табл. [1.3](#_bookmark7)).

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](#_bookmark8)).

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.

## Параметр «bulge»

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

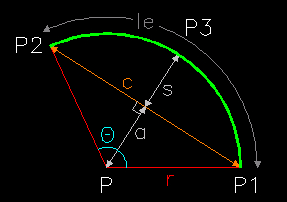


Рисунок 1.1 — Геометрия дуги окружности

Так как дуга окружности описывает часть этой окружности, то она и обладает всеми атрибутами данной окружности (см. рис. [1.1](#_bookmark10)). 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, следую­ щие специфичные атрибуты:

* Начальная и конечная точка (𝑃 1, 𝑃 2) — 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, проходя­ щая через её вершины;
* Середина дуги (𝑃 3) — 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 пер­ пендикулярно к хорде.

Кроме самой себя, дуга может, также, и описывать другие геомет­ рические 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 чет­ верти угла дуги между выбранной вершиной и следующей вершиной в спис­ ках вершин полилиний. Отрицательность параметра *bulge* указывает на то, что дуга отрисовывается по часовой стрелке от выбранной вершины к сле­ дующей. Выпуклость, равная нулю — прямой сегмент, выпуклость, равная единице — половина окружности.

Проблема «расшифровки» атрибутов дуги для дальнейших манипу­ ляций 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 рассматривае­ мая 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](#_bookmark11)).

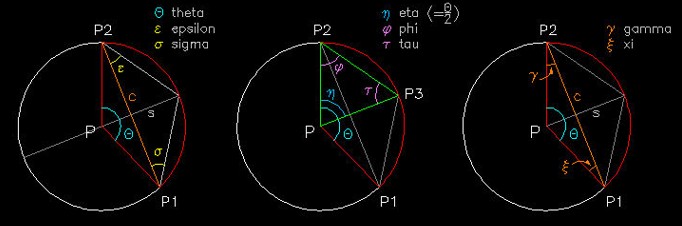


Рисунок 1.2 — Дуга окружности с проведённой хордой и углами при ней

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](#_bookmark12)):

Θ

𝜙 = 𝜏 =

(180*∘ −* Θ) 2

*⇒* 𝜙 = 90*∘ −* 4

(1.1)

Теперь посмотрим на хорду 𝑐, проведённую от 𝑃 1 до 𝑃 2. Вместе с красными катетами угла Θ она тоже образует равнобедренный треуголь­ ник, а значит, 𝛾 = 𝜉. Угол при вершине треугольника 𝑃 *−* 𝑃 1 *−* 𝑃 2 —

2

это центральный угол Θ, поэтому 𝛾 и 𝜉 вычисляются следующим образом

([1.2](#_bookmark13)):

𝛾 = 𝜉 = 180*∘ −* Θ

2

2

*⇒* 𝛾 = 90*∘ −* Θ

(1.2)

Таким образом, желтый угол 𝜀 должен быть равняться разнице меж­ ду фиолетовым углом 𝜙 и оранжевым углом 𝛾. Другими словами, 𝜀 — это четверть центрального угла Θ ([1.4](#_bookmark14)):

𝜀 = (90*∘ −* Θ) *−* (90*∘ −* Θ) *⇒* 𝜀 = Θ *−* Θ = Θ

4

2

2

4

4

(1.3)

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

от 𝑃 3 до 𝑃 4). Высота образует катет прямоугольного треугольника с углом, равным четверти центрального угла (см. желтый треугольник 𝑃 *−* 𝑃 2 *−* 𝑃 3 на рис. [1.3](#_bookmark15)), и поскольку тангенс описывает отношение между катетами в

прямоугольном треугольнике, легко описать геометрию с помощью этого одного угла ([1.4](#_bookmark14)):

sin(𝜀) cos(𝜀)

= tan(𝜀) (1.4)

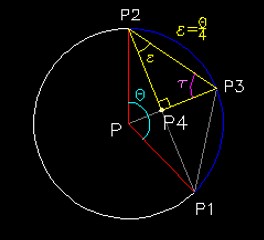


Рисунок 1.3 — Связь угла 𝜀 с центральным углом

Мы, также, могли бы найти тангенс угла 𝜀, просто разделив противо­ лежащий 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](#_bookmark16)):

𝑐

Примем

𝑠 =

2 *·* tan(𝜀) (1.5)

tan(𝜀) = 𝑏𝑢𝑙𝑔𝑒 (1.6)

Тогда

𝑠 =

𝑐

2 *·* 𝑏𝑢𝑙𝑔𝑒 (1.7)

Таким образом, радиус дуги может быть найден следующим образом

([1.8](#_bookmark17)):

𝑟 =

( 𝑐 )2 + 𝑠2

2𝑠

2

(1.8)

Знак той или иной выпуклости важен для определения дуги отно­ сительно 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 абсо­ лютного значения выпуклости, а не фактического значения, иначе можно получить отрицательный радиус.

Итак, поняв, что 𝑏𝑢𝑙𝑔𝑒 = 𝑡𝑎𝑛(Θ), в согласовании с документацией

4

AutoCAD [**?**] примем, что *bulge* положителен, когда при передвижении от начальной точки дуги к конечной движение происходит против часовой стрелки.

Ясно, что когда Θ = 0, то и 𝑏𝑢𝑙𝑔𝑒(Θ) = 0. Для углов в 180*∘* услов­ но принимается, что 𝑏𝑢𝑙𝑔𝑒(Θ) = *±*1. В случае, когда Θ = 90*∘*, получим

следующее ([1.9](#_bookmark18)):

𝑏𝑢𝑙𝑔𝑒(90*∘*) = tan(90*∘* ) = tan(𝜋) (1.9)

4 8

Используя зависимость для тангенса половинного аргумента ([1.10](#_bookmark19)):

𝛼

tan(

2

sin(𝛼)

) = 2

*±*

cos(𝛼)

2 sin2(𝛼)

= 2

*±*

2 sin(𝛼) cos(𝛼)

= 1 *−* cos(𝑥)

sin(𝑥)

*±*

(1.10)

Для 𝛼 = 𝜋

8

2 2 2

получим ([1.11](#_bookmark20)):

𝜋 1 *−* cos(𝜋 )

1 *− √*2

1 *− √*1 *√*

𝑏𝑢𝑙𝑔𝑒(90*∘*) = tan(

4

8 ) = *±*

sin(𝜋 ) = *±*

= 2

*±*

2 *√*1

2

*√*

= *±*(

2 *−* 1)

4 2 2

(1.11)

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

а) 𝑏𝑢𝑙𝑔𝑒 = 0 для отрезка прямой,

б) 𝑏𝑢𝑙𝑔𝑒 = 1 для дуги в 180*∘* (половина окружности),

*±*

в) 𝑏𝑢𝑙𝑔𝑒 = *±*(*√*2 *−* 1) *≈* 0.41421... для четвертей окружностей, когда

угол раствора дуги равен 90*∘*.

# Разработка программного обеспечения «DXF Primiview»

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

## Выбор языка программирования

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

**Библиотеки.** Для создания описанного ПП необходимо привлече­ ние различных библиотек. Кроме стандартной библиотеки, с Python мож­ но использовать множество прикладных библиотек, несколько из которых будут описаны в следующих частях работы. Специфичные библиотеки, поз­ воляющие «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 копи­ рованием кода. Кроме того, в процессе разработки ПО, для реализации пользовательского интерфейса используется набор расширений Qt, кото­ рый тоже работает на таких платформах, как Linux и другие UNIX-подоб­ ные 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 про­ граммного 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 длин­ ные этапы компиляции и связывания, необходимые в некоторых других ЯП, что, также, увеличивает производительность труда программиста [**?**].

## Разработка алгоритмов

Для создания программного обеспечения необходимо сначала разра­ ботать концепцию функционирования программы, опираясь на её назначе­ ние, на основные её функции. Когда определены модули, блоки и функ­ циональные части ПО, можно приступать к разработке его на выбранном ЯП.

## 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 верифика­ ции обработанных данных и, соответственно, подтверждения соответствия их исходным, пользователю должна предоставляться возможность конвер­ тировать эти данные в какой-либо из предлагаемых форматов. За это от­ вечает модуль экспорта, который, в свою очередь, подразделяется на четы­ ре 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 — началь­ ная 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 фор­ мату TXT (x, y, r), содержаться точки, олицетворяющие начало и конец то­ го или иного примитива. В данном случае информация в файле тэгирован­ ная, 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](#_bookmark25).

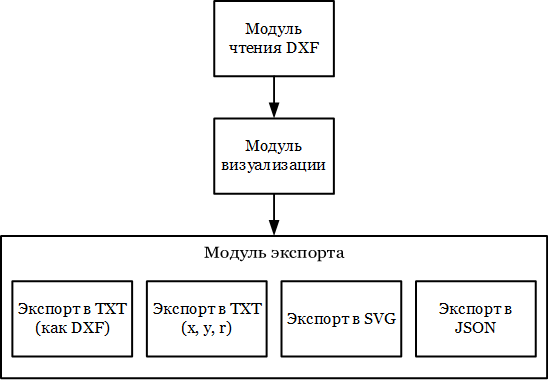


Рисунок 2.1 — Принципиальная структура ПО «Primiview»

## Алгоритмы модулей программы

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

**Алгоритм чтения DXF-файлов.** Алгоритм [1](#_bookmark27) 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 биб­ лиотекой *ezdxf*, которая используется для чтения DXF-файлов.

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

изменяемый упорядоченный тип данных, представляющих собой последо­ вательность элементов, разделённых между собой запятой и заключённых в квадратные скобки. Данный тип данных используется в ЯП Python, по­ этому для соблюдения общности, термин будет применяться и в части [2.3](#_bookmark39) этого раздела.

Примем ряд условных обозначений:

𝑚𝑠𝑝 — пространство объектов модели (modelspace);

--� — 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

1. **конец**
2. **цикл** *∀ сущн. ∈* 𝑚𝑠𝑝 **выполнять**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**15**

**16**

**17**

**если** *сущ явл. линией* **тогда**

сущн. --� множ. линий

**конец**

**иначе если** *сущн. явл. дугой* **тогда**

сущн. --� множ. дуг

**конец**

**иначе если** *сущн. явл. окружностью* **тогда**

сущн. --� множ. окружностей

**конец**

**иначе если** *сущн. явл. полилинией* **тогда**

сущн. --� множ. полилиний

**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](#_bookmark29)) 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](#_bookmark28)):

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

LINE(#01) 0 . 1 0 . 1

0 . 1 0 . 1

1

2

3

**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

## конец

1. **цикл** *∀ ARC ∈ множ. дуг* **выполнять**
2. записать в файл: атрибут сущности, 𝑥0, 𝑦0, 𝑥1, 𝑦1, 𝑟;

перевести курсор на новую строку

## конец

1. **цикл** *∀ CIRCLE ∈ множ. окруж.* **выполнять**
2. записать в файл: атрибут сущности, 𝑥𝑐, 𝑦𝑐, 𝑟; перевести

курсор на новую строку

## конец

1. **цикл** *∀ LWPOLYLINE ∈ множ. полилин.* **выполнять**
2. записать в файл атрибут сущности;
3. **цикл** *∀ точки LWPOLYLINE* **выполнять**
4. **цикл** *∀ координаты x, y* **выполнять**
5. записать в файл значение координаты

## конец

1. **конец**
2. перевести курсор на новую строку

## конец

**Алгоритм 2 —** Запись примитивов в TXT (как DXF)

В алгоритме [2](#_bookmark29) 𝑥0, 𝑦0 — координаты начала примитива; 𝑥1, 𝑦1 — ко­ ординаты конца примитива; 𝑥𝑐, 𝑦𝑐 — координаты центра окружности; 𝑟 — радиус дуги или окружности.

**Алгоритм записи примитивов в TXT (x,y,r).** Алгоритм [4](#_bookmark34) при­ зван, так же как и в прошлом случае, в открытый только что созданный

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

Записи в текстовом файле должны выглядеть следующим образом (см. листинг [2.2](#_bookmark30)):

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 объ­ ектах LWPOINT сущности LWPOLYLINE степень искривления показывает параметр *bulge*, суть которого подробно описана в разделе [1.3](#_bookmark4). Так как же­ лаемый формат вывода информации о примитивах содержит именно ради­ ус примитива, а не параметр искривления, то необходимо удобно получить радиус из *bulge*.

Для этого воспользуемся уже выведенной зависимостью [[1](#_bookmark58)] и приме­ ним её в принятых обозначениях ([2.1](#_bookmark31)):

𝑅 = *|*𝑏𝑢𝑙𝑔𝑒 + 1 *| · |*𝐴 *−* 𝑍*|* , (2.1)

𝑏𝑢𝑙𝑔𝑒 4

где A — начальная точка; Z — конечная точка.

Примем, что 𝐴(𝑥0,𝑦0), 𝐵(𝑥1,𝑦1). Тогда 𝐴𝑍(𝑥1 *−* 𝑥0; 𝑦1 *−* 𝑦0). Таким

образом, длина вектора через декартовы координаты ([2.2](#_bookmark32)):

*|*𝐴 *−* 𝑍*|* = √︀(𝑥1 *−* 𝑥0)2 + (𝑦1 *−* 𝑦0)2 (2.2)

**Алгоритм 3 —** Вычисление радиуса сегмента полилинии Примем условные обозначения:

**Исходные данные:** текущая точка, следующая точка

**Результат:** радиус сегмента полилинии

1. инициализация;
2. **если** 𝑏𝑢𝑙𝑔𝑒 *(текущей точки)* = 0 **тогда**

**3** 𝑟 = 0

1. **конец**
2. **иначе**

**6**

𝑟 = *|*𝑏𝑢𝑙𝑔𝑒 + 𝑏𝑢𝑙𝑔𝑒 *| ·*

1

*√*(𝑥𝑛𝑒𝑥𝑃 *−*𝑥𝑝𝑟𝑒𝑣𝑃 )2+(𝑦𝑛𝑒𝑥𝑃 *−*𝑦𝑝𝑟𝑒𝑣𝑃 )2

4

**7 конец**

⇝ — запись в файл;

𝑛𝑒𝑤𝑙𝑖𝑛𝑒 — перевод на новую строку.

**Исходные данные:** путь к имя.txt

**Результат:** имя.txt (x,y,r)

**1** инициализация; **цикл** *∀ LINE ∈ множ. линий* **выполнять**

**2** 𝑥0 𝑦0 0 ⇝ имя.txt; 𝑥1 𝑦1 0 ⇝ имя.txt

## конец

1. **цикл** *∀ ARC ∈ множ. дуг* **выполнять**

**5** 𝑥0 𝑦0 0 ⇝ имя.txt; 𝑥1 𝑦1 𝑟 ⇝ имя.txt

## конец

1. **цикл** *∀ CIRCLE ∈ множ. окруж.* **выполнять**
2. 𝑥𝑐 + 𝑟 𝑦𝑐 0 ⇝ имя.txt; /\* первая половина окруж. \*/

**9** 𝑥𝑐 *−* 𝑟 𝑦𝑐 𝑟 ⇝ имя.txt

**10** 𝑥𝑐 *−* 𝑟 𝑦𝑐 0 ⇝ имя.txt; /\* вторая половина окруж. \*/

**11** 𝑥𝑐 + 𝑟 𝑦𝑐 𝑟 ⇝ имя.txt

## конец

1. **цикл** *∀ POLYLINE ∈ множ. полилиний*
2. *prevPoint=None ;* /\* предыдущая точка \*/

## выполнять

1. **цикл** *LWPOINT ∈ множ. точек. полилинии* **выполнять**
2. **если** *prevPoint̸*=*None* **тогда**
3. r = алгоритм [3](#_bookmark33) (prevPoint, LWPOINT)

𝑥(𝑝𝑟𝑒𝑣𝑃 ) 𝑦(𝑝𝑟𝑒𝑣𝑃 ) 0 ⇝ имя.txt

**19** 𝑥(𝑙𝑤𝑝𝑜𝑖𝑛𝑡) 𝑦(𝑙𝑤𝑝𝑜𝑖𝑛𝑡) 𝑟 ⇝ имя.txt

**20 конец**

**21** 𝑝𝑟𝑒𝑣𝑃 𝑜𝑖𝑛𝑡 = 𝑙𝑤𝑝𝑜𝑖𝑛𝑡

## конец

1. **если** *контур замкнут* **тогда**
2. r = алгоритм [3](#_bookmark33) (prevPoint, LWPOINT)

**25** 𝑥(𝑙𝑤𝑝𝑜𝑖𝑛𝑡посл) 𝑦(𝑙𝑤𝑝𝑜𝑖𝑛𝑡посл) 0 ⇝ имя.txt

**26** 𝑥(𝑙𝑤𝑝𝑜𝑖𝑛𝑡перв) 𝑦(𝑙𝑤𝑝𝑜𝑖𝑛𝑡перв) 𝑟 ⇝ имя.txt

## конец

1. **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 представ­ ляет 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

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

1

Следующие две строки должны содержать определение типа доку­ мента (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

<svg version =" 1.1 " width =" 100% " height =" 100% " view Box =" 102 . 1188828597992 - 211 . 7921423734452

50 . 000000000000014 26.0 "

base Profile =" full "

xmlns =" http :// www . w3 . org /2000/ svg " xmlns : xlink =" http :// www. w3 . org / 1999/ xlink " xmlns : ev=" http :// www . w3 . org /2001/ xml - events " >

1

2

3

4

5

6

В листинге [2.4](#_bookmark35) присутствует необязательный элемент **viewBox**, ко­ торый представляет собой параметр с четырьмя значениями, отделяемыми пробелами, определяющими квадратную рамку, в которой будет распола­ гаться 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 — соот­ ветственно, 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: алго­ ритм [5](#_bookmark36), алгоритм [6](#_bookmark37) и алгоритм [7](#_bookmark38).

**Исходные данные:** списки примитивов с параметрами

**Результат:** список координат 𝑥, список координат 𝑦

**1** инициализация; пустой список координат 𝑥, пустой список координат 𝑦 **цикл** *∀ LINE ∈ множ. линий* **выполнять**

**2** 𝑥0,𝑥1 --� список 𝑥

**3** 𝑦0,𝑦1 --� список 𝑦

## конец

1. **цикл** *∀ ARC ∈ множ. дуг* **выполнять**

**6** (𝑥𝑐 + 𝑟),(𝑥𝑐 *−* 𝑟) --� список 𝑥

**7** (𝑦𝑐 + 𝑟),(𝑦𝑐 + 𝑟) --� список 𝑦

## конец

1. **цикл** *∀ CIRCLE ∈ множ. окруж.* **выполнять**

**10** (𝑥𝑐 + 𝑟),(𝑥𝑐 *−* 𝑟) --� список 𝑥

**11** (𝑦𝑐 + 𝑟),(𝑦𝑐 + 𝑟) --� список 𝑦

## конец

1. **цикл** *∀ LWPOLYLINE ∈ множ. полилин.* **выполнять**
2. **цикл** *∀точки в множ. LWPOINTS* **выполнять**

**15** 𝑥 --� список 𝑥

**16** 𝑦 --� список 𝑦

## конец

1. **конец**
2. вернуть список координат 𝑥 и список координат 𝑦

**Алгоритм 5 —** Вычленение координат изображения из DXF в отдельные списки

**Алгоритм 6 —** Поиск наименьших координат изображения из DXF

**Исходные данные:** список координат 𝑥, список координат 𝑦

**Результат:** 𝑥𝑀𝐼𝑁 , 𝑦𝑀𝐼𝑁

1. инициализация;
2. использовать алгоритм [5](#_bookmark36)
3. вернуть 𝑥𝑀𝐼𝑁 , 𝑦𝑀𝐼𝑁 из списков стандартными функциями сортировки ЯП

**Алгоритм 7 —** Поиск длины и высоты изображения из DXF

**Исходные данные:** список координат 𝑥, список координат 𝑦

**Результат:** ширина и высота рамки изображения

**1** инициализация; использовать алгоритм [5](#_bookmark36)

**2** определить 𝑥𝑀𝐼𝑁 , 𝑥𝑀𝐴𝑋, 𝑦𝑀𝐼𝑁 , 𝑦𝑀𝐴𝑋 из списков стандартными функциями сортировки ЯП

**3** ширина = 𝑥𝑀𝐴𝑋 *−* 𝑥𝑀𝐼𝑁

**4** высота = 𝑦𝑀𝐴𝑋 *−* 𝑦𝑀𝐼𝑁

**5** вернуть значения ширины и высоты

## Разработка программного обеспечения

# Экономическая часть

В данном разделе описаны экономические аспекты проекта по созда­ нию ПО «Primiview» для обработки геометрической информации 2D-объ­ ектов специального типа.

## Экономическое обоснование

Целью экономического обоснования проекта является представле­ ние разработанного ПП в качестве проекта для реализации на предприя­ тии, что позволит провести планирование и корректировку последователь­ ности работ (при необходимости).

## Разработка проекта

**Цель проекта** — создание модуля конвертации форматов, как от­ дельного 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 понима­ ется минимально-необходимая квалификация проектировщика для выпол­ нения задач технологического проектирования.

Трудоёмкость проектирования, в данном случае, выражается в сред­ нем 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 проекти­ рование, 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, корректиров­ ки по которым оператор станка с ЧПУ вносит после того, как автоматизи­ рованное проектирование выполнено.

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

По численным критериям:

* сложность проектирования: инженер-технолог II категории по ква­ лификационному справочнику [[2](#_bookmark59)],
* сроки (время) проектирования: 3 часа,
* себестоимость проектирования: 1420000 руб. (см. раздел [3.2.2](#_bookmark50)),
* качество результатов проектирования: 10 ошибок.

**Целевое состояние системы клиента**: усовершенствованная, бо­ лее гибкая и универсальная версия этого ПО.

По численным критериям:

* сложность проектирования: инженер-технолог (без категории) по квалификационному справочнику [[2](#_bookmark59)],
* 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, прово­ дит планирование проекта, оценку его экономической эффективности, кон­ троль 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 вы­ полняют проверку работоспособности ПП, ищут и сообщают отделу разра­ ботчиков о найденных и необходимых к устранению ошибок и недочётов программы.

## 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 ис­ полнителями, 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 ша­ гом данного уровня будет *разработка алгоритмов* программного обеспече­ ния. Данный этап необходим, так как именно на нём ещё можно решить несостыковки в логической части программы, исправление которых на сле­ дующих этапах редко бывает возможным.

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

Завершающим этапом второго уровня является *сдача проекта* за­ казчику. Эта стадия может быть выполнена только при полном выполне­ нии 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 про­ дукта в открытом доступе рынка. Если есть, то чем заказчика не устраи­

вает их использование;

в) Требования к продукту. Определяются численные критерии, кото­ рым 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 документиру­ ется 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 по­ ставленных задач на уровне логики и математики, строится набор взаи­ мосвязей алгоритмов. По возможности, рассматривается применение уже существующих универсальных (реже, специальный) алгоритмов.

Этап разработки ПО содержит такие стадии:

а) Выбор ЯП. Данная стадия подразумевает проведение сравнительно­ го 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 за­ висит, 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 программ­ ный 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, содер­ жит 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 ре­ зультаты своей деятельности для предоставления заказчику;

в) Обучение Сотрудников. При разработке нового ПП, команде разра­ ботчиков 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 со­ стоит 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](#_bookmark44)). Уровни иерархической структуры оформлены таким образом, что, чем конкретнее описаны элементы структуры, тем насыщеннее цвет заливки.

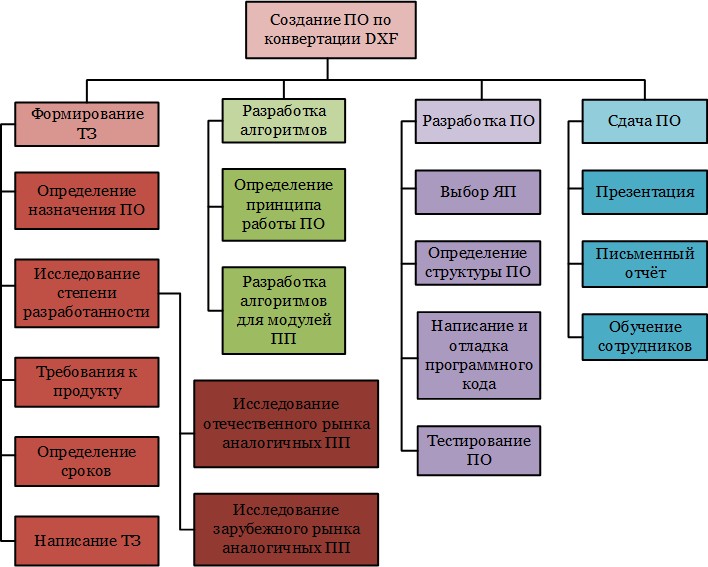


Рисунок 3.1 — Дерево задач проекта

## Построение диаграмм проекта

На данном этапе строится диаграмма Ганта (англ. Gantt Chart, так­ же, ленточная диаграмма, график Ганта) — это тип столбчатых диаграмм, использующийся для иллюстрации плана, графика работ проекта. Также, является методом планирования проекта. Эта диаграмма представляет со­ бой отрезки (графические плашки), размещающиеся на горизонтальной шкале времени. Каждый отрезок соответствует отдельно задаче (подза­ даче). 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 вы­ полнения задачи; показывается вертикальная линия, отвечающая моменту

«сегодня».

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

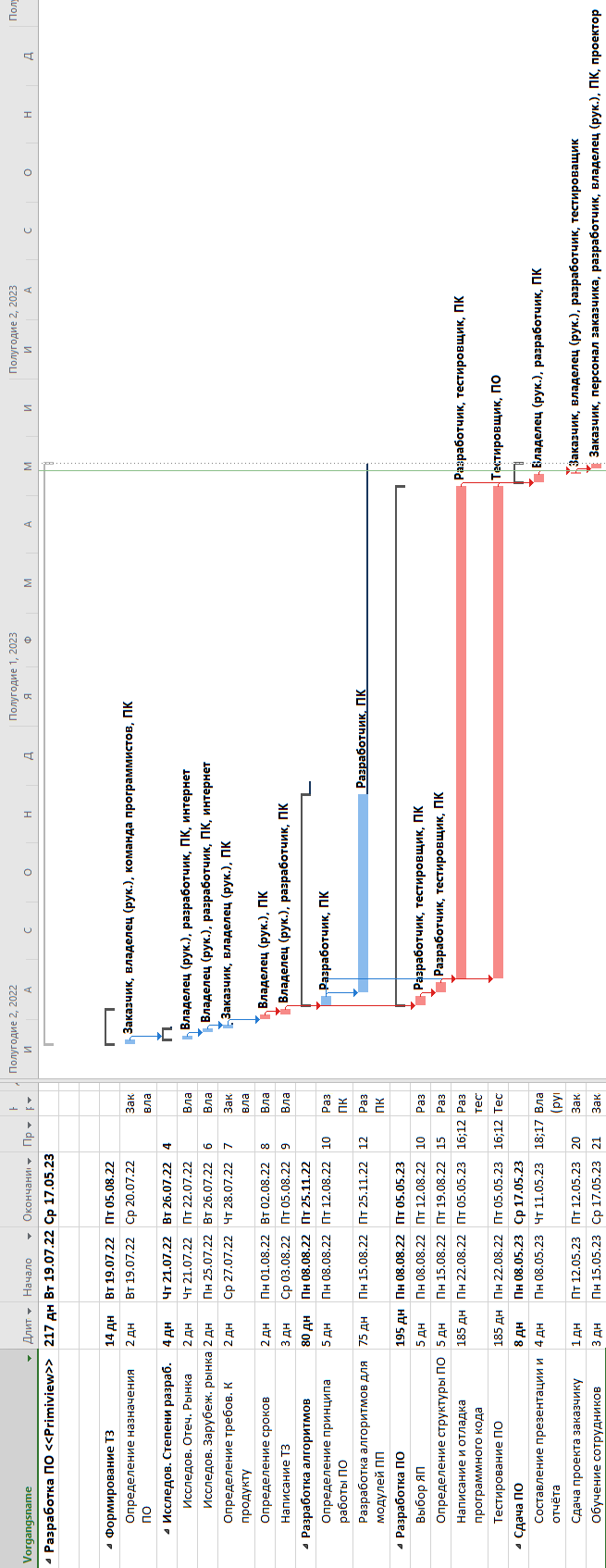


Рисунок 3.2 — Диаграмма Ганта для проекта

## Сравнительная экономическая эффективность

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

## 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, ко­ торый, обычно, берёт чертёж детали и либо вручную пишет УП, либо ис­ пользует иностранные CAM-системы, предварительно создавая 3d-модель по выданному чертежу. В рамках данной (третьей) части ВКР будет рас­ сматриваться 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 ситу­ ацию с точки зрения покупателя оборудования рассматриваемого станко­ строительного предприятия. Соберём основные данные в таблицу [3.1](#_bookmark48).

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 про­ изводства инженеров-технологов-про­ граммистов для написания УП | Привлекаются технологи из имеюще­ гося штата сотрудников для выпол­ нения дополнительных обязанностей по контролю создания УП с помощью  купленного ПО |
| Время написания УП в три раза доль­  ше, чем во втором варианте | Время написания УП в течение одно­  го часа (в среднем) |

Варианты рассматриваются с точки зрения потребителя оборудова­ ния. 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 рассматривае­ мом предприятии-клиенте ситуация с производством каждый месяц прак­ тически не меняется. Также, большинство справочных величин ссылаются именно на годовой период, что тоже является подтверждением равномер­ ной распределённости экономических характеристик внутри отдельно взя­ тых 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 до­ полнительных капитальных вложений, в данном случае – в токарный ста­ нок с ЧПУ.

Срок полезного использования оборудования – 10 лет.

Срок контракта на выпуск продукции с использованием данного обо­ рудования – в рассматриваемой ситуации нет ограничений, токарная обра­ ботка постоянно проводится на предприятии.

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

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

**Определение состава затрат по вариантам (результат – пе­ речень затрат).** Корректировка затрат в соответствии с возможностями Методики сравнительной эффективности (включаем в расчет только раз­ личающиеся по альтернативам затраты). Деление затрат на переменные и постоянные. Формирование списка исходных данных для выполнения рас­ четов (см. таблицу [3.2](#_bookmark49)).

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](#_bookmark49)

|  |  |  |
| --- | --- | --- |
| Постоянные затраты (на единицу оборудования) | | |
| Цена годовой лицензи­ и/контракта обслужи­ вания CAM-системы,  руб | 50000 | 55000 |

## Расчёты и анализ

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

## Исходные данные.

Среднее годовое количество УП на предприятии-покупателе станков

𝑁 = 200 шт;

Срок полезного использования оборудования:

𝑇𝑚𝑎𝑐ℎ𝑖𝑛𝑒𝑟𝑦 = 10 лет; Требования инвестора по окупаемости ИП:

𝑇𝑖𝑛𝑣 = 5 лет; Принятая норма окупаемости:

𝑇𝑛 = 𝑇 𝑖𝑛𝑣 = 5 лет; Наладка полной группы станков:

𝐶𝐴𝑀𝑇 𝑒𝑟𝑚 = 20000 руб;

Цена встроенной CAM-системы на единицу оборудования:

𝐶𝐴𝑀 2 = 70000 руб;

Цена станка без встроенной CAM-системы:

𝑀 = 2750000 руб;

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

𝐶𝐴𝑀 1𝑃𝑒𝑟𝑚 = 50000 руб;

𝐶𝐴𝑀 2𝑃𝑒𝑟𝑚 = 55000 руб; Среднее время написания одной УП:

𝑡1 = 3 часа;

𝑡2 = 1 час; Почасовая оплата технолога:

𝑆𝑎𝑙 = 1500 руб; Страховые сбора от заработной платы:

𝑓 𝑒𝑒𝑠 = 30%

Количество покупаемых станков:

𝑁𝑀 = 5 шт;

## Расчёт.

Себестоимость использования оборудования и ПО:

𝐶1 = 𝑆𝑎𝑙 *·* 𝑡1 *·* 𝑁 *·* (100% + 𝑓 𝑒𝑒𝑠) + 𝑁𝑀 *·* 𝐶𝐴𝑀1𝑃𝑒𝑟𝑚 =

= 1500 *·* 3 *·* 200 *·* (100% + 30%) + 5 *·* 50000 = 1420000 руб;

𝐶2 = 𝑆𝑎𝑙 *·* 𝑡2 *·* 𝑁 *·* (100% + 𝑓 𝑒𝑒𝑠) + 𝑁𝑀 *·* 𝐶𝐴𝑀2𝑃𝑒𝑟𝑚 =

= 1500 *·* 1 *·* 200 *·* (100% + 30%) + 5 *·* 55000 = 665000 руб;

(3.1)

(3.2)

Условно-годовая экономия (на себестоимости):

𝐸 = *|*𝐶1 *−* 𝐶2*|* = *|*1420000 *−* 665000*|* = 755000 руб; (3.3)

Капитальные вложения предприятия-покупателя станков:

𝐾1 = 𝑀 *·* 𝑁𝑀 + 𝐶𝐴𝑀𝑇 𝑒𝑟𝑚 = 2750000 *·* 5 + 20000 = 13770000 руб; (3.4)

𝐾2 = 𝑁𝑀 *·* (𝑀 + 𝐶𝐴𝑀 2) + 𝐶𝐴𝑀𝑇 𝑒𝑟𝑚 =

= 5 *·* (2750000 + 70000) + 20000 = 14120000 руб;

Дополнительные капитальные вложения:

(3.5)

𝐾𝑒𝑥𝑡𝑟 = *|*𝐾1 *−* 𝐾2*|* = *|*13770000 *−* 14120000*|* = 350000 руб; (3.6)

𝐾𝑒𝑥𝑡𝑟 = 𝑁𝑀 *·* 𝐶𝐴𝑀 2 = 5 *·* 70000 = 350000 руб. (проверка); (3.7) Срок окупаемости дополнительных капитальных вложений:

𝑇𝑝𝑎𝑦𝑏𝑎𝑐𝑘

= 𝐾𝑒𝑥𝑡𝑟

𝐸

350000

=

755000

= 0,464 лет; (3.8)

Приведённые затраты по вариантам:

1

𝑍1 = 𝐶1 +

𝑇

𝑛

1

𝑍2 = 𝐶2 +

𝑇

𝑛

*·* 𝐾1 = 1420000 + 5 *·* 13770000 = 4174000 руб; (3.9)

*·* 𝐾2 = 665000 + 5 *·* 14120000 = 3489000 руб; (3.10)

1

1

Годовой экономический эффект:

𝐸𝑎𝑛𝑛𝑢𝑎𝑙 = *|*𝑍1 *−* 𝑍2*|* = *|*4174000 *−* 34890000*|* = 685000 руб; (3.11) Минимальный годовой объём деятельности, при котором обеспечи­

вается приведённый годовой экономический эффект:

𝑁𝑀 *·* 𝐶𝐴𝑀1𝑃 𝑒𝑟𝑚 *−* 𝑁𝑀 *·* 𝐶𝐴𝑀2𝑃 𝑒𝑟𝑚 *−* 𝐾𝑒𝑥𝑡𝑟

𝑁𝑐𝑟 =

𝑇𝑛 =

𝑆 *·* 𝑡2 *·* (100% + 𝑓 𝑒𝑒𝑠) *−* 𝑆𝑎𝑙 *·* 𝑡1 *·* (100% + 𝑓 𝑒𝑒𝑠)

(3.12)

5 *·* 50000 *−* 5 *·* 55000 *−* 350000

5

=

1500 *·* 1 *·* 100% + 30% *−* 1500 *·* 3 *·* 100% + 30%

= 24,359 шт;

По вычисленным в формулах [3.9](#_bookmark51), [3.10](#_bookmark52) затратам изобразим на гра­ фике (см. рисунок [3.3](#_bookmark53)) границы целесообразности рассматриваемых вари­ антов.

*·*106

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

4.2



𝑍, руб. (приведённые затраты)

4

3.8

3.6

3.4

3.2

3

0 20 40 60 80 100 120 140 160 180 200

𝑁, шт. (объём деятельности)

Рисунок 3.3 — Границы целесообразности рассматриваемых вариантов

Получив необходимые значения по критериям сравнения, сведём ре­ зультаты в таблицу [3.3](#_bookmark54)).

Таблица 3.3 — Сравнительная характеристика рассматриваемых вариантов по показателям эффективности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование показателя | Ед. изм. | По вариантам: | | Отклонения показателей |
| Вариант  без ПО | Вариант  c 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 дополни­  тельных кап. вложений |  | 0,464 |  |  |
| Приведённые затраты по ва­  риантам | руб. | 4174000 | 3489000 | 658000 |
| Годовой экономический эф­  фект |  |  | 685000 |  |

## Выводы по результатам расчётов.

Так как на первых этапах расчёта по методу сравнительной эффек­ тивности ИП нельзя было сделать конкретный вывод по поводу целесооб­ разности одного из предлагаемых вариантов по причине того, что по пер­ вому 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 ка­ питальных 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 ва­ риант, 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 оказа­ лись больше, чем по второму.

Действительно, экономия времени на создании УП нивелирует боль­ шие 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.