# Реферат

Отчёт содержит 38 страниц, 0 рисунков, 0 таблиц, 12 формул, 0 приложений.

**????**

Объект ВКР — учебники издательства УрФУ.

Цель работы — разработать программное обеспечение для дешиф­ рования информации из файлов типа DXF, извлечении из них информа­ ции о геометрических примитивах, сформировать текстовый файл, со­ держащий всю необходимую информацию в читабельном виде.

Методы исследования: теоретический анализ и последующий син­ тез 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 ком­ мерческого заказа ООО «Униматик», создании специализированного программного обеспечения — модуля экспорта DXF-файлов в САПР.

# Содержание

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

1. [Определение области применения и главных требований к раз­](#_bookmark1) [рабатываемым конвертерам](#_bookmark1) . . . . . . . . . . . . . . . . . . . . 8
   1. [Главные требования ко входному файлу](#_bookmark2) . . . . . . . . . 8
   2. [Описание формата DXF-файлов](#_bookmark3) . . . . . . . . . . . . . . 9 [1.3 Параметр «bulge»](#_bookmark8) 11
2. [Разработка программного обеспечения «DXF Primiview»](#_bookmark20) 17
   1. [Выбор языка программирования](#_bookmark21) 17
   2. [Разработка алгоритмов](#_bookmark22) 18
      1. [Структура программы](#_bookmark23) 18
      2. [Алгоритмы модулей программы](#_bookmark25) 19
   3. [Разработка программного обеспечения](#_bookmark38) 28
3. [Экономическое обоснование проекта](#_bookmark39) 29
   1. [Сравнительная экономическая эффективность](#_bookmark40) 29
      1. [Исходные данные](#_bookmark41) 29
      2. [Расчёты и анализ](#_bookmark44) 32
      3. [Выводы по результатам расчётов.](#_bookmark49) 36

[Заключение](#_bookmark50) 37

[Список использованных источников](#_bookmark51) 38

# Глоссарий

В настоящей работе применяют следующие термины с соответ­ ствующими определениями:

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

**Система автоматизированного проектирования** — организационно-техническая система, входящая в структуру про­ ектной 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 отоб­ ражении 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 обме­

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

**TXT** — Текстовый формат файлов, представляющий собой по­ следовательность строк электронного текста.

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

**XML** — eXtensible Markup Language. Расширяемый язык размет­ ки 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 (УМ­ НОЦ) происходит разработка специализированных САПР. Создаваемые ПП работают с файлами, содержащими геометрическую 2D-информа­ цию. 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 конверти­ рует в наиболее популярные форматы изображений (JPEG, PNG). Для использования конвертера на предприятии-заказчике необходим особый формат данных, в которые конвертируется DXF. Такие форматы не мо­ гут 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-фай­ лов, компании «Unimatic». Ещё одним недостатком зарубежного ПО яв­ ляется ежемесячная плата разработчикам. Одной из задач современной Российской Федерации является импортозамещение на производстве, по­ этому разработка собственного ПП увеличит независимость и самостоя­ тельность предприятий и компаний, пользующихся этим ПП.

**Цель работы** заключается в разработке программного обеспе­ чения для дешифрования информации из файлов типа DXF с геомет­

рической информацией объектов специального типа, извлечении из них информации о геометрических примитивах, вывода полученной графиче­ ской 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 ра­ диуса примитивов (линий, дуг) между ними, формирования файла-опи­ сание двумерной векторной графики в формате SVG с выводом данных в исходном виде DXF, формирования текстового файла (JSON) с выво­ дом данных в виде координат точек и степенями кривизны примитивов между ними.

Данные конвертеры необходимы, в частности, при разработке та­

ких САПР, как Сириус, ТокКТЭ и других.

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

1. проанализировать входные данные DXF, в соответствии с разра­ батываемым ПО;
2. выявить структурно-содержательные особенности файлов форма­ та DXF для последующей работы с разрабатываемым ПО;
3. проанализировать возможности применения разных ЯП для раз­ работки ПО;
4. разработать алгоритмы;
5. разработать ПО;
6. провести анализ экономической целесообразности разрабатывае­ мого 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, исполь­ зуемый в данной работе, может быть реализован для создания ПП по конвертированию в другие файловые форматы;

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

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

г) положенные в основу алгоритмы и написанное ПО имеет откры­ тый доступ в сети Интернет и имеет перспективу развития в полноцен­ ный модуль импорта/экспорта файлов;

д) коммерческая выгода заказчика.

# Определение области применения и главных требований к разрабатываемым конвертерам

В данном разделе поясняется область применения разрабатывае­ мого ПО, а также, указываются требования к разрабатываемым конвер­ терам.

## Главные требования ко входному файлу

Так как данная работа нацелена на создание ПО для обработки геометрической информации 2D-объектов специального типа, то конвер­ тироваться 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 по­ следующем предполагает разработку УП для токарных станков с ЧПУ, то геометрическая информация должна содержать определённый набор геометрических примитивов, с которым может работать система испол­ нительных 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 предназна­ чаются разрабатываемые конвертеры, геометрия в DXF-файле на входе конвертеров должна состоять из, как минимум одного из представлен­ ных 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. Тегирован­ ные данные означают, что каждому элементу данных в файле предше­ ствует целое число, называемое групповым кодом. Значение группового кода указывает, какой тип данных имеет следующий элемент. Это зна­ чение также указывает смысл элемента данных для данного типа объ­ екта. 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](#_bookmark4)).

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) |

Продолжение на след. стр.

Продолжение таблицы [1.1](#_bookmark4)

|  |  |
| --- | --- |
| 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*, необхо­ димые для её реального отображения (см. табл. [1.2](#_bookmark5)).

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 ре­ ального отображения (см. табл. [1.3](#_bookmark6)).

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 |

Продолжение на след. стр.

Продолжение таблицы [1.3](#_bookmark6)

|  |  |
| --- | --- |
| 50 | Начальный угол |
| 51 | Конечный угол |

**CIRCLE.** Рассмотрим тэги сущности *Окружность*, необходи­ мые для её реального отображения (см. табл. [1.4](#_bookmark7)).

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 каждой из вершин полилинии. Чтобы понять сущность данного пара­ метра, который представляет собой некоторую степень кривизны дуги окружности между двумя точками, необходимо сначала разобраться с геометрией дуг.

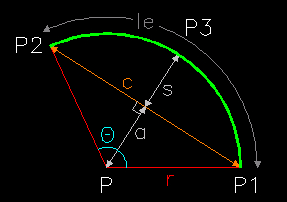


Рисунок 1.1 — Геометрия дуги окружности

Так как дуга окружности описывает часть этой окружности, то она и обладает всеми атрибутами данной окружности (см. рис. [1.1](#_bookmark9)). Сре­ ди 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) окружно­

сти.

Для дальнейшей работы с геометрией дуг примем, также, следу­

ющие специфичные атрибуты:

* Начальная и конечная точка (𝑃 1, 𝑃 2) — это «вершины» дуги. Хо­ тя иногда и целесообразно говорить о конкретных точках, не лежащих на концах дуги;
* Длина хорды (𝑐) — у дуг и окружностей можно провести беско­ нечное количество хорд, но для нас интерес представляет только хорда, проходящая 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 сере­ дина 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 геомет­ рические формы: круговой сегмент и сектор. Обе геометрические формы включают в себя все вышеперечисленные атрибуты, однако для выведе­ ния 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* указы­ вает на то, что дуга отрисовывается по часовой стрелке от выбранной вершины к следующей. Выпуклость, равная нулю — прямой сегмент, вы­ пуклость, равная единице — половина окружности.

Проблема «расшифровки» атрибутов дуги для дальнейших ма­ нипуляций 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 рассмат­ риваемая 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

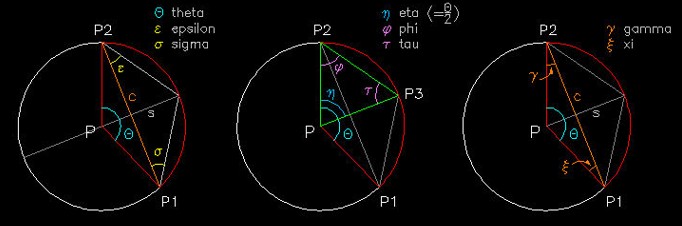
(см. рис. [1.2](#_bookmark10)).

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](#_bookmark11)):

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 — это

2

Θ

центральный угол Θ, поэтому 𝛾 и 𝜉 вычисляются следующим образом

([1.2](#_bookmark12)):

𝛾 = 𝜉 = 180*∘ −* Θ

2

*⇒* 𝛾 = 90*∘ −* Θ

(1.2)

Таким образом, желтый угол 𝜀 должен быть равняться разнице между фиолетовым углом 𝜙 и оранжевым углом 𝛾. Другими словами, 𝜀

2

— это четверть центрального угла Θ ([1.4](#_bookmark13)):

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

4

2

2

4

4

(1.3)

Параметр *bulge* (выпуклость) описывает, насколько дуга «выпи­ рает» из вершин, то есть насколько велика высота дуги (𝑠) (или расстоя­ ние 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](#_bookmark14)), и поскольку тангенс описывает отношение меж­

ду катетами в прямоугольном треугольнике, легко описать геометрию с

помощью этого одного угла ([1.4](#_bookmark13)):

sin(𝜀) cos(𝜀)

= tan(𝜀) (1.4)

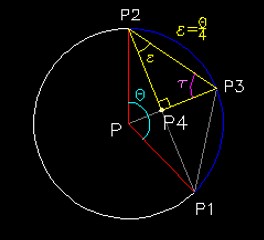


Рисунок 1.3 — Связь угла 𝜀 с центральным углом

Мы, также, могли бы найти тангенс угла 𝜀, просто разделив про­ тиволежащий катет на смежный катет — что означает высоту дуги 𝑠, делённую на половину длины хорды 𝑐, — но не зная 𝑠 и уже имея тан­ генс 𝜀, полезнее найти 𝑠 ([1.5](#_bookmark15)):

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 об­ разом ([1.8](#_bookmark16)):

𝑟 =

( 𝑐 )2 + 𝑠2

2𝑠

2

(1.8)

Знак той или иной выпуклости важен для определения дуги от­ носительно 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 мож­ но 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*∘*

условно принимается, что 𝑏𝑢𝑙𝑔𝑒(Θ) = *±*1. В случае, когда Θ = 90*∘*,

получим следующее ([1.9](#_bookmark17)):

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

4 8

Используя зависимость для тангенса половинного аргумента

([1.10](#_bookmark18)):

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](#_bookmark19)):

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 обработ­ ки геометрической информации 2D-объектов специального типа. Обосно­ вывается 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 несколь­ кими факторами.

**Библиотеки.** Для создания описанного ПП необходимо привле­ чение различных библиотек. Кроме стандартной библиотеки, с Python можно использовать множество прикладных библиотек, несколько из которых будут описаны в следующих частях работы. Специфичные биб­ лиотеки, 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, написан­ ных на Python, выполняются корректно на всех основных платформах. Перенос программы между операционными системами реализуется про­ стым копированием кода. Кроме того, в процессе разработки ПО, для реализации пользовательского интерфейса используется набор расшире­ ний 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. Объ­ ём программного кода на языке Python обычно составляет треть или да­ же пятую часть эквивалентного программного кода на языке C++ или

Java. Кроме того, при запуске программы, написанной на ЯП Python ми­ нуются длинные этапы компиляции и связывания, необходимые в неко­ торых других ЯП, что, также, увеличивает производительность труда программиста [**?**].

## Разработка алгоритмов

Для создания программного обеспечения необходимо сначала раз­ работать концепцию функционирования программы, опираясь на её на­ значение, на основные её функции. Когда определены модули, блоки и функциональные части ПО, можно приступать к разработке его на вы­ бранном ЯП.

## Структура программы

Основываясь на цели, поставленной во введении, разрабатывае­ мая утилита должна принимать на входе DXF-файл, то есть открывать его и обрабатывать его содержимое. Для проверки правильности обра­ ботанных данные, то есть, для верификации содержимого DXF-файла, программа должна визуализировать для пользователя обработанное. По­ сле верификации обработанных данных и, соответственно, подтвержде­ ния 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 дан­ ных 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 сущ­ ности будут представлены сочетанием двух строк, первая из которых

— начальная точка примитива, вторая — конечная. Вторая строка содер­

жит в себе радиус скругления примитива, переходящего из первой точки во вторую.

в) Модуль экспорта в формат SVG, который, при открытии, вектор­ но 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](#_bookmark24).

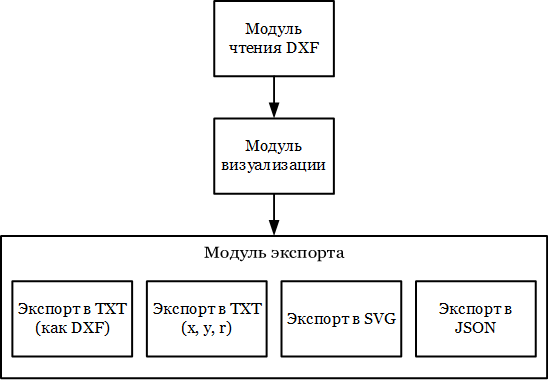


Рисунок 2.1 — Принципиальная структура ПО «Primiview»

## Алгоритмы модулей программы

В данном пункте приводится описание алгоритмов ПО

«Primiview» в виде псевдокода.

**Алгоритм чтения DXF-файлов.** Алгоритм [1](#_bookmark26) показывает схе­ му работы процесса извлечения поддерживаемых ПО Primiview прими­ тивов 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 «види­ мыми». 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 назы­ ваться изменяемый упорядоченный тип данных, представляющих собой последовательность элементов, разделённых между собой запятой и за­ ключённых в квадратные скобки. Данный тип данных используется в ЯП Python, поэтому для соблюдения общности, термин будет применяться и в части [2.3](#_bookmark38) 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](#_bookmark28)) 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 обра­ зом (см. листинг [2.1](#_bookmark27)):

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

LINE(#01) 0 . 1 0 . 1

1

2

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

## 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](#_bookmark28) 𝑥0, 𝑦0 — координаты начала примитива; 𝑥1, 𝑦1 — координаты конца примитива; 𝑥𝑐, 𝑦𝑐 — координаты центра окружности;

𝑟 — радиус дуги или окружности.

**Алгоритм записи примитивов в TXT (x,y,r).** Алгоритм [4](#_bookmark33) призван, так же как и в прошлом случае, в открытый только что создан­ ный текстовый файл записать информацию о примитивах, которые были прочитаны из выбранного DXF-файла.

Записи в текстовом файле должны выглядеть следующим обра­ зом (см. листинг [2.2](#_bookmark29)):

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 пока­ зывает параметр *bulge*, суть которого подробно описана в разделе [1.2](#_bookmark3). 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](#_bookmark52)] и при­ меним её в принятых обозначениях ([2.1](#_bookmark30)):

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](#_bookmark31)):

**Алгоритм 3 —** Вычисление радиуса сегмента полилинии

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

(2.2)

**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**

Примем условные обозначения:

⇝ — 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](#_bookmark32) (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](#_bookmark32) (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 пред­ ставляет собой язык разметки, а значит, имеет правила синтаксиса, грам­ матики 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 раз­ метки 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 до­ кумента (заголовок DOCTYPE), однако, данное объявление может ока­ заться 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](#_bookmark34) присутствует необязательный элемент **viewBox**, который представляет собой параметр с четырьмя значениями, отделя­ емыми пробелами, определяющими квадратную рамку, в которой будет располагаться графика. Данный атрибут позволяет автоматически мас­ штабировать изображение до размеров указанного контейнера, причём,

без потери качества, так как графическая информация храниться и вос­ производится 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 изображе­ ние. Значения указываются в пикселях.

Таким образом, чтобы перенести данные из DXF в SVG, снача­ ла определяются эти четыре значения. Алгоритмы для их определения: алгоритм [5](#_bookmark35), алгоритм [6](#_bookmark36) и алгоритм [7](#_bookmark37).

**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](#_bookmark35)
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](#_bookmark35)

**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

# Экономическое обоснование проекта

В данном разделе описано экономическое обоснование создания ПО «Primiview» для обработки геометрической информации 2D-объек­ тов специального типа.

## 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 но­ вых видов техники, модернизации оборудования, способов организации производственных процессов и т. п. То есть для оценки решений, кото­ рые 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.

## Исходные данные

Станкостроительное предприятие рассматривает заказ на созда­ ние программного обеспечения для своего оборудования (токарных стан­ ков с ЧПУ). Это ПО автоматизирует процесс создания управляющих программ для станков с ЧПУ, взамен работе инженера-технолога-про­ граммиста, который, обычно, берёт чертёж детали и либо вручную пи­ шет 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 точ­ ки зрения покупателя оборудования у предприятия, которое привлекло силы университета для создания описанного ПО. Сравниваются два ва­ рианта – покупка станков без ПО и, соответственно, с ним.

**Сравнительная характеристика вариантов.** Рассмотрим си­ туацию с точки зрения покупателя оборудования рассматриваемого стан­ костроительного предприятия. Соберём основные данные в таблицу [3.1](#_bookmark42).

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 подготов­ ки производства инженеров-техно­ логов-программистов для написа­ ния УП | Привлекаются технологи из имею­ щегося штата сотрудников для вы­ полнения дополнительных обязанно­ стей по контролю создания УП с по­  мощью купленного ПО |
| Время написания УП в три раза  дольше, чем во втором варианте | Время написания УП в течение од­  ного часа (в среднем) |

Варианты рассматриваются с точки зрения потребителя оборудо­ вания. 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 рассматри­ ваемом предприятии-клиенте ситуация с производством каждый месяц практически не меняется. Также, большинство справочных величин ссы­ лаются именно на годовой период, что тоже является подтверждением равномерной распределённости экономических характеристик внутри от­ дельно 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) рав­ ным 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 возмож­ ностями 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](#_bookmark43)).

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 |
| Постоянные затраты (на единицу оборудования) | | |

Продолжение на след. стр.

Продолжение таблицы [3.2](#_bookmark43)

|  |  |  |
| --- | --- | --- |
| Цена годовой лицен­ зии/контракта обслу­ живания CAM-систе­  мы, руб | 50000 | 55000 |

## 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 стан­

ков

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;

Условно-годовая экономия (на себестоимости):

(3.1)

(3.2)

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 обеспе­

чивается 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](#_bookmark45), [3.10](#_bookmark46) затратам изобразим на гра­ фике (см. рисунок [3.1](#_bookmark47)) границы целесообразности рассматриваемых ва­ риантов.

*·*106

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

4.2

𝑍, руб. (приведённые затраты)

4

3.8

3.6

3.4

3

0 20 40 60 80 100 120 140 160 180 200

𝑁, шт. (объём деятельности)

Рисунок 3.1 — Границы целесообразности рассматриваемых вариантов

Получив необходимые значения по критериям сравнения, сведём результаты в таблицу [3.3](#_bookmark48)).

Таблица 3.3 — Сравнительная характеристика рассматриваемых вариан­ тов 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, все­  го | руб. | 13770000 | 14120000 | 350000 |
| в том числе: | | | | |
| Наладка станков | руб. | 20000 | 20000 | - |
| Цена станка (Вар. 2 + ПО) | руб. | 13750000 | 665000 | 755000 |

Продолжение на след. стр.

Продолжение таблицы [3.3](#_bookmark48)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Срок окупаемости дополни­  тельных кап. вложений |  | 0,464 |  |  |
| Приведённые затраты по ва­  риантам | руб. | 4174000 | 3489000 | 658000 |
| Годовой экономический эф­  фект |  |  | 685000 |  |

## 3.1.3 Выводы по результатам расчётов.

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

Исходя из расчётов и построенного по ним графика, сделаем вы­ вод, что, производя уже 25 УП за год, выгоднее становится вариант с ПО, так как приведённые затраты для соответствующего количество произ­ водимых 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 оказал­ ся намного меньше рассматриваемого нормативного срока окупаемости (0,4 и 5 лет, соответственно), а приведённые затраты по первому вариан­ ту 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.