Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ СОЗДАНИЯ ТЕХНОЛОГИЧЕСКИХ КАРТ ПРЕДВАРИТЕЛЬНОЙ МЕХАНИЧЕСКОЙ ОБРАБОТКИ ЗАГОТОВОК ДЕТАЛЕЙ

Выпускная квалификационная работа
по направлению подготовки 09.03.02 Информационные системы
и технологии
профилю подготовки «Информационные технологии в медиаиндустрии»

Идентификационный номер ВКР: 675

Екатеринбург 2018

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующая кафедрой ИС
_____ Н. С. Толстова
«____» _____ 2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ СОЗДАНИЯ ТЕХНОЛОГИЧЕСКИХ КАРТ ПРЕДВАРИТЕЛЬНОЙ МЕХАНИЧЕСКОЙ ОБРАБОТКИ ЗАГОТОВОК ДЕТАЛЕЙ

Исполнитель:
обучающийся группы ДЗИТм-511 С. А. Шимов
Руководитель:
М. Ю. Черноскутов
Нормоконтролер: Н. В. Хохлова

Екатеринбург 2018

АННОТАЦИЯ

Выпускная квалификационная работа состоит из программного продукта для автоматического создания технологических карт предварительной механической обработки заготовок деталей и пояснительной записки на 90 страницах, содержащей 48 рисунков, 8 таблиц, 32 источника литературы, а также 2 приложения на 34 страницы.

Ключевые слова: ТЕХНОЛОГИЧЕСКАЯ КАРТА, РАЗРАБОТКА КОМПЬЮТЕРНОГО ПРИЛОЖЕНИЯ, C#.

Шимов С. А. Разработка приложения для автоматизации создания технологических карт предварительной механической обработки заготовок деталей: выпускная квалификационная работа / С. А. Шимов ; Рос. гос. проф.-пед. ун-т., Ин-т инж.-пед. образования, Каф. информ. систем и технологий. — Екатеринбург, 2018. — 90 с.

В работе рассмотрен процесс разработки приложения для автоматического создания технологических карт предварительной механической обработки заготовок деталей.

Целью работы является создание программного продукта с учётом требований и специфики конкретного предприятия. В ходе взаимодействия со специалистом, были изучены требования и алгоритм разработки технологических карт. На основе данных требований и алгоритма было создано приложение. Изучение новых технологий и прикладных программных интерфейсов, позволило создать более удобный и функциональный продукт.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во всё более возрастающей экономической конкуренции между странами, особо важным элементом является производительность труда. Россия отстает от ведущих стран по этому показателю примерно в 3,5 раза. Высокая производительность труда важна не только для экономики страны в целом, но и для каждого человека в частности, ведь выполнение большего объема работы за меньшую единицу времени даёт большую продукцию при том же затраченном времени, а значит несомненно должно привести к увеличению заработной платы, либо к увеличению свободного времени у работника (при ограниченном, запланированном объеме выпускаемой продукции) [16].

Одним из способов увеличения производительности труда является техническое переоснащение производства, использование нового эффективного оборудования, новых технологий. В наше время, когда стремительно компьютеризируются все сферы человеческой деятельности, очень важно не отставать от передовых стран и использовать потенциал, который дают компьютеры и информационные технологии.

Информационные технологии позволяют автоматизировать какие-либо рутинные действия, которые приходится делать человеку, повышая не только скорость, но и точность выполняемой работы. Таким образом, заменяя человека компьютером, можно сэкономить на оплате труда. В случае если полная замена не возможна, один работник, использующий автоматизирующие возможности компьютера, может заменить нескольких работников, которые выполняют работу без использования информационных технологий.

В данной работе будет предпринята попытка изучить степень, в которой может быть применена автоматизация при использовании информационных технологий в машиностроении. А конкретнее, при разработке технологических карт механической обработки деталей на одном из предприятий г. Екатеринбурга.

Таким образом, целью данной работы является создание компьютерной программы, которая помогла бы разработчику технологических карт (технологу-нормировщику) максимально упростить и ускорить его действия. Тем самым повысив эффективность труда.

Для осуществлений поставленной цели необходимо выполнить следующие задачи:

1. Изучить технологический процесс. Узнать по какому алгоритму разрабатываются технологические карты. Какие используются исходные данные и что должно получиться в результате. Изучить требования технологов.

2. Оценить текущий уровень систем разработки программного обеспечения, чтобы подобрать наиболее эффективный, удобный и функциональный инструмент.

3. Изучить требования к интерфейсу современного программного обеспечения.

4. Спроектировать интерфейс приложения в соответствии с рекомендуемыми требованиями таким образом, чтобы он был максимально удобным и интуитивно понятным, учитывая пожелания пользователей, для которых разрабатывается приложение.

5. Изучить прикладные программные интерфейсы, которые понадобятся при разработке приложения.

6. Преобразовать алгоритм, полученный в п.1 в алгоритм понятный компьютеру, то есть написать код программы, который будет производить вычисления, построение чертежа, формирование текстовой информации.

7. Запрограммировать вывод полученной информации в удобный для дальнейшего использования формат, в соответствии с требованиями пользователей (технологов).

8. Протестировать программный продукт на наличие ошибок. Реализовать средства предупреждения ввода пользователем заведомо неверных данных.

9. Создать руководство пользователя.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ и общая характеристика предметной области

Разработка программного обеспечения — это процесс, включающий в себя компьютерное программирование, документирование, тестирование и исправление ошибок, в результате которого создаётся программный продукт. Программный продукт (компьютерная программа) — набор инструкций, который выполняет определенную задачу при запуске на компьютере [6].

Компьютерные программы могут создаваться для самых разнообразных целей, но три наиболее часто встречающиеся это удовлетворение конкретных потребностей конкретного клиента/организации; удовлетворение предполагаемых потребностей некоторых пользователей; либо индивидуальное использование [3].

Процесс разработки программного обеспечения разделяется на отдельные фазы, что позволяет улучшить проектирование и управление. Обычно процесс разработки программного обеспечения включает в себя следующие стадии [10]:

- анализ задачи;
- исследование рынка;
- сбор требований;
- проектирование плана разработки;
- разработка (написание кода);
- тестирование;
- развертывание;
- поддержка и исправление ошибок.

Эти стадии часто называют жизненным циклом программного обеспечения.

При разработке программного обеспечения (ПО) необходимо учесть, что оно должно быть практичным. Потери, возникающие из-за плохо разработанного, тяжелого в использовании ПО несут не только конечные пользователи, но и сами производители. Непрактичность предлагаемых свойств и возможностей заставляет людей обращаться к создателям программ с просьбами о доработках. Доработки и переделки программ чаще всего связаны с их изначальной непрактичностью. В результате вместо того, чтобы работать над новыми проектами, разработчики оказываются привязанными к старым; они пересматривают их и пытаются устранить дефекты.

1.2 Анализ существующих разработок

1. КОМПАС-График v17 — универсальная система автоматизированного проектирования, позволяющая в оперативном режиме выпускать чертежи изделий, схемы, спецификации, таблицы, инструкции, расчётно-пояснительные записки, технические условия, текстовые и прочие документы [7].

Эта система является универсальной, а значит, имеет массу избыточных возможностей, которые усложняют интерфейс и взаимодействие пользователя с программой.

Данная система является платной. Что повлечет дополнительные значительные расходы.

Универсальная система далеко не всегда может удовлетворить требования конкретного предприятия, так как ориентирована на отрасль в целом.

2. Система ADEM. Эта система предназначена для автоматизации решения проектных, конструкторских и технологических задач в области машиностроения [18]. Она применяется для следующих видов деятельности:

- проектирование изделий;
- объемное и плоское моделирование;
- оформление чертежей и другой конструкторской документации;

- проектирование техпроцессов;
- оформление технологической и сопроводительной документации;
- программирование станков с числовым программным управлением;
- управление архивами и проектами;
- обновление накопленных знаний;
- укрупнённое трудовое нормирование;
- управление справочными данными.

Таким образом, функционал данной системы является избыточным. Этот факт, несомненно, скажется на сложности при обучении пользователей. Кроме того, универсальность данной системы придется приспосабливать под потребности конкретного предприятия. Данная система автоматизированного проектирования является платной.

1.3 Анализ средств разработки

Для создания приложения будет использоваться язык программирования C#, который позволит использовать программную платформу .NET Framework. Эта платформа была разработана корпорацией Microsoft в 2002 году и используется главным образом в операционной системе Windows, которая является самой распространенной операционной системой, используемой на стационарных компьютерах и ноутбуках.

Программная платформа .NET Framework имеет ряд важных преимуществ: управление памятью, обработку исключений, а также включает богатую библиотеку встроенных прикладных программных интерфейсов. Данные обстоятельства позволят значительно упростить и ускорить разработку, осуществить взаимодействие с текстовым процессором Microsoft Word, который является наиболее популярным и распространенным [17].

Приложение будет разрабатываться в интегрированной среде разработки Microsoft Visual Studio 2017. Данная среда разработки использует платформы Windows API, Windows Forms, Windows Presentation Foundation [11].

Внешний вид окна интегрированной среды разработки Microsoft Visual Studio 2017 Community приведен на рисунке 1. В центральной части интерфейса находится редактор кода, который выделяет фрагменты кода определенным цветом, в зависимости от их смысла. Также редактор кода поддерживает IntelliSense — компонент автодополнения кода, которые помогает избежать ошибок при написании. Справа находится обозреватель решения, с помощью которого осуществляется навигация в проекте; сверху находится панель инструментов.

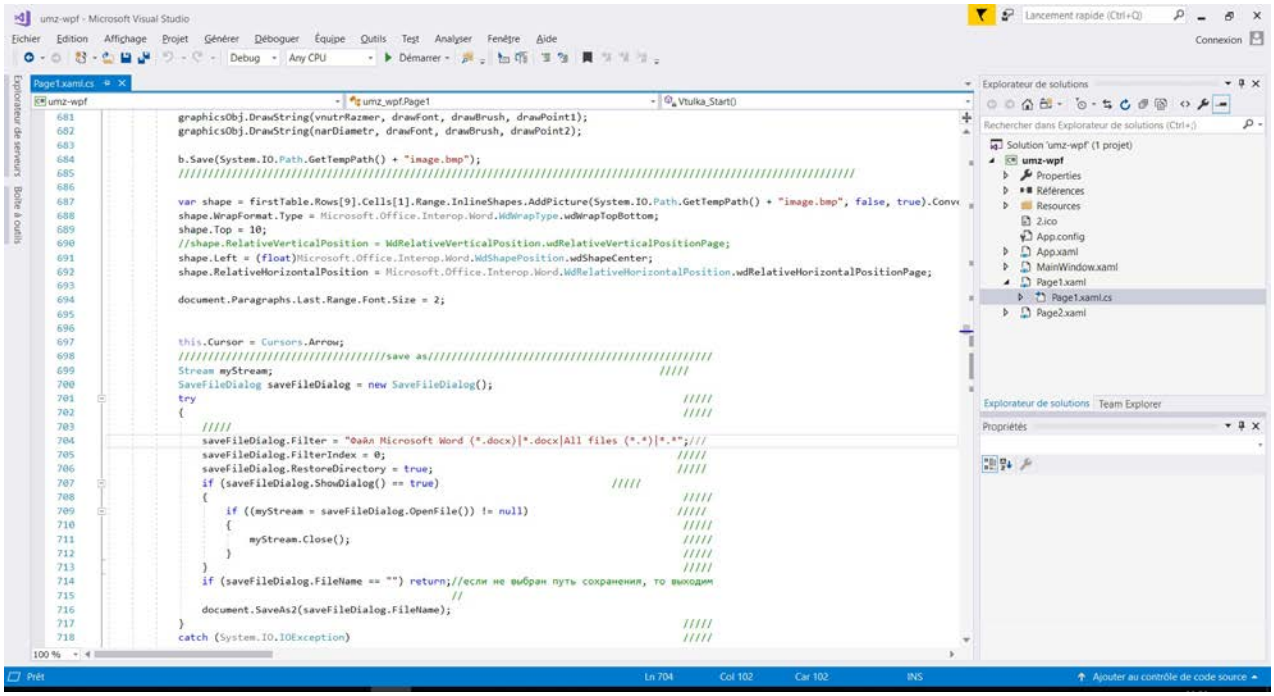

Рисунок 1 — Интерфейс интегрированной среды разработки Microsoft Visual Studio 2017 Community

Также Visual Studio содержит встроенный отладчик, компоненты для создания графического пользовательского интерфейса, инструменты, помогающие при тестировании [9].

Таким образом, данная интегрированная среда разработки упростит и сделает более эффективной разработку приложения. А наличие базовой бес-

платной версии Visual Studio — Microsoft Visual Studio Community станет несомненным плюсом.

1.4 Общий алгоритм реализации проекта

1. Создание эскиза интерфейса в каком-либо графическом редакторе. На данном этапе будет необходимо сделать эскиз интерфейса приложения, который позволит получить представление о внешнем виде окна приложения, расположении элементов управления, оценить какое расположение элементов сделает взаимодействие с программой наиболее интуитивно понятным и удобным. Для создания эскиза будет использоваться продукт Adobe Photoshop.
2. Создание графического интерфейса пользователя в Microsoft Visual Studio 2017. После создания эскиза интерфейса и его анализа, следует реализовать этот интерфейс непосредственно в Visual Studio.
3. Написание кода приложения в Microsoft Visual Studio 2017. На данном этапе реализуется функционал приложения. То, каким образом будут формироваться технологические карты.
4. Тестирование приложения. Приложение должно работать правильно и стабильно, чтобы не допускать логических ошибок и не доставлять неприятный опыт пользователю. Для этого программный продукт должен быть способен проверять вводимые пользователем данные, чтобы они не были заведомо ошибочными. Кроме того, содержание формируемых приложением технологических карт должно быть проверено на соответствие значениям, полученным без использования приложения.

1.5 Формулировка требований к разрабатываемому продукту

1. Для работы приложения должна использоваться операционная система Windows, так как эта система используется на предприятии. Кроме то-

го, у пользователей в целом, чаще всего, есть опыт работы именно в этой операционной системе.

2. Необходимо, чтобы приложение имело простой и интуитивно понятный интерфейс. Количество элементов управления следует свести к минимуму, также, элементы управления должны быть сгруппированы по своему назначению. При вводе заведомо неправильных данных пользователем, приложение должно остановить свою работу и уведомить пользователя о необходимости привести введенные данные в соответствие.

3. Приложение должно насколько возможно сократить участие технолога в разработке технологических карт. Для этого оно должно самостоятельно производить вычисления, создавать чертежи, выводить информацию в отформатированном виде.

4. Нужно чтобы приложение для вывода информации использовало текстовый процессор Microsoft Word, так как он используется на компьютерах предприятия, является одним из самых популярных текстовых процессоров, позволяет создавать таблицы, форматировать текст, работать с изображениями. Приложение должно самостоятельно создать шаблон технологической карты, произвести его форматирование и заполнить соответствующей информацией.

5. В случае возникновения ошибок, необходимо чтобы приложение уведомляло пользователя и останавливало свою работу.

6. Следует, чтобы приложение было легко расширяемо, для обеспечения возможности увеличения числа деталей по мере потребности. Для подтверждения способности приложения к расширению, оно должно быть в состоянии формировать технологические карты для, как минимум, двух типов деталей.

1.6 Проектирование эскиза интерфейса

Проектирование эскиза интерфейса приложения позволят заранее продумать, то, какие элементы управления он должен будет содержать, выбрать

такое их расположение, которое будет наиболее удобным при работе пользователя. Тщательно проработанный эскиз интерфейса не только обеспечит быструю его реализацию, но и поможет избежать каких-либо крупных корректировок на поздних этапах проектирования приложения.

При проектировании интерфейса нужно учитывать, кто будет являться пользователем программного продукта, поэтому необходимо узнать [14]:

- какую цель пользователи ставят при работе с программным обеспечением, для которого создаётся интерфейс;
- определенные задачи, которые они выполняют, чтобы достичь этих целей;
- язык и слова, при помощи которых они описывают свои действия;
- их навыки в использовании программного обеспечения, аналогичного тому, которое для них разрабатывается.

Эскиз интерфейса создадим в программе Adobe Photoshop. При создании эскиза необходимо учесть следующее:

1. Окно программы должно содержать заголовок с кнопками закрытия и сворачивания. В заголовке будет отображаться название программы, а кнопки закрытия и сворачивания позволят соответственно закрыть программу, либо свернуть её окно, после чего, при нажатии на вкладку программы на панели задач, окно программы может быть восстановлено.

2. Так как типов деталей несколько и им присущи различные параметры и операции, необходимо, чтобы имелся элемент управления, позволяющий выбирать конкретную деталь и, в соответствии с выбором, менять интерфейс. В качестве такого элемента управления, выберем Combo Box (Комбинированный список) — сочетание выпадающего списка (раскрывающегося при щелчке мыши) и однострочного текстового поля. Он позволит, раскрываясь, выбрать пользователю тип детали из списка, а затем закроется и будет отображать название типа выбранной детали. Этот элемент управления позволит уместить потенциально довольно длинный список, в одну строку и не будет загромождать интерфейс. Кроме того, стоит учесть, что выбор типа де-

тали — это первоочередная задача, поэтому следует, чтобы элемент управления Combo Box был самым первым и достаточно заметным.

3. От пользователя необходимо получить сведения о параметрах деталей (длина, высота, ширина и т. п.). Для этого интерфейс должен содержать элементы для ввода данных и подписи к ним. При этом список параметров должен иметь свой заголовок, чтобы пользователю было проще ориентироваться в интерфейсе.

4. Список операций, производимых над деталями, также должен иметь свой заголовок, но выбор операций не требует от пользователя ввода каких-либо данных, достаточно просто указать необходимость конкретных операций. Поэтому, в этом случае, больше подойдет элемент управления Check Box (Флажок), позволяющий пользователю управлять параметром с двумя состояниями — ☑ включено (будет обозначить наличие операции) и ☐ выключено (будет обозначать отсутствие операции).

5. Следует сделать так, чтобы список требований был оформлен подобным списку операций образом. Наличие или отсутствие какого-либо требования также удобно представить с помощью элемента управления Check Box (Флажок).

6. Наконец, необходимо чтобы интерфейс программы содержал кнопку, которая будет запускать вычисления и формирование технологической карты. Поместим кнопку в стороне от других элементов управления, чтобы пользователь легко мог понять её особую роль. Положение в правом нижнем углу подразумевает, что пользователь будет обращаться к этому элементу управления в последнюю очередь, после заполнения исходных данных.

7. Отключим возможность работы в полноэкранном режиме, так как компоненты приложения скомпонованы довольно компактно, а большое расстояние между ними скорее понизит читабельность, так как пользователь будет вынужден переключать внимание на различные части экрана.

В результате, получился эскиз, представленный на рисунке 2.

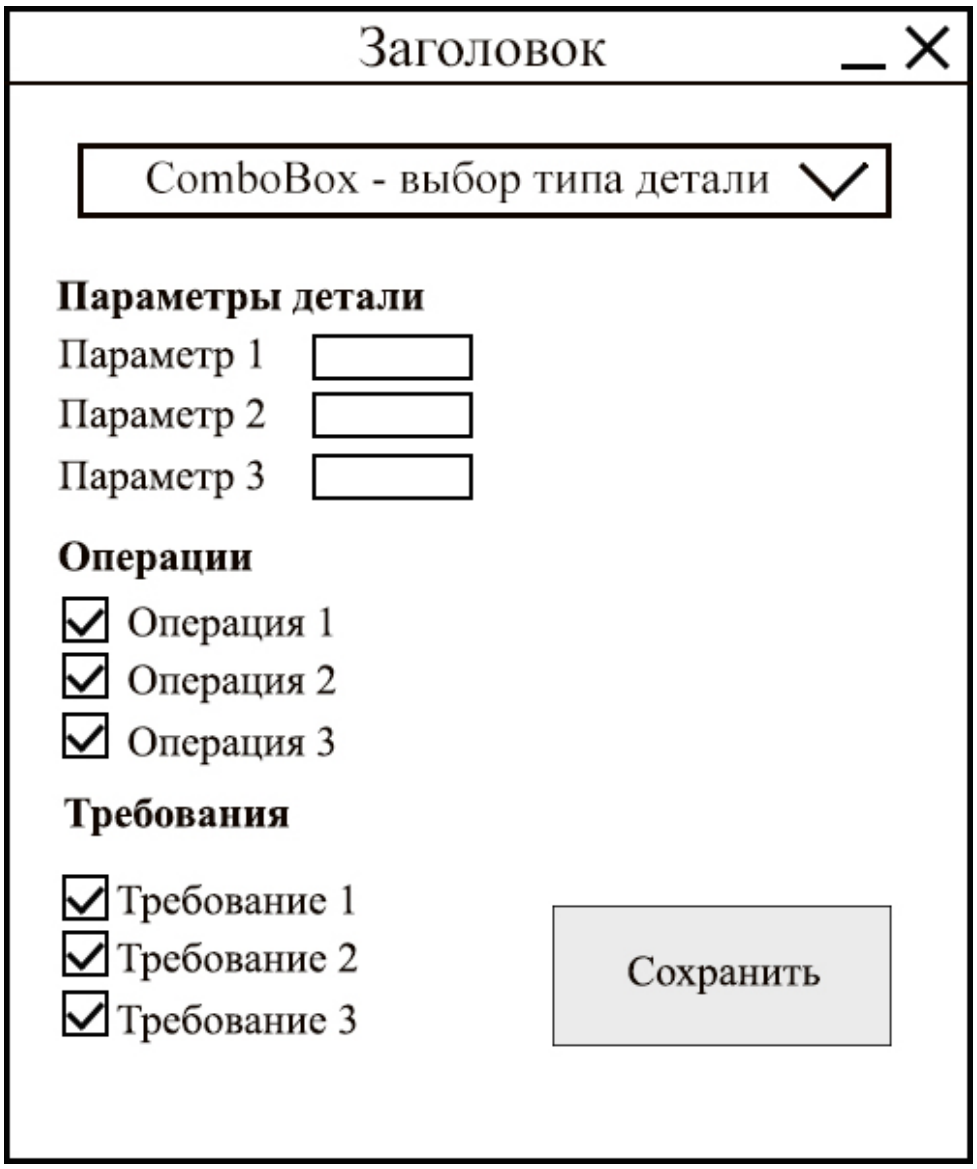

Рисунок 2 — Эскиз приложения для автоматизации разработки технологических карт

2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.1 Реализация интерфейса

Теперь, с учетом эскиза, разработаем интерфейс приложения для первой детали — втулки с бахромой с помощью Windows Forms [12].

В параметрах должны быть следующие значения: наружный диаметр, внутренний диаметр, длина и твёрдость. В операциях должны быть представлены: отрезная операция, токарная, отрезка пробы, вырезка заготовок образцов. В требованиях — термическа обработка (ТО), ультразвуковая дефектоскопия (УЗД), УЗД по техническому решению.

Кроме того, понадобился новый список — вид стали. Так как сталь может быть только одного вида, то лучше всего для реализации этого списка подойдет группа радиокнопок — этот элемент управления позволит пользователю выбрать только одну опцию из предопределенного набора.

На рисунке 3 показан реализованный в Windows Forms интерфейс для втулки с бахромой.

Так как Windows Forms является устаревающей технологией (в 2014 году корпорация Microsoft объявила об остановке дальнейшего развития этой технологии) и имеет ряд недостатков, реализуем интерфейс программы с помощью Windows Presentation Foundation.

Windows Presentation Foundation (WPF) — это современная графическая система, разработанная корпорацией Microsoft для создания пользовательского интерфейса в операционной системе Windows. Она имеет ряд инновационных особенностей, таких как встроенное аппаратное ускорение и независимость от используемого разрешения экрана. Для создания интерфейса используется язык разметки XAML [28].

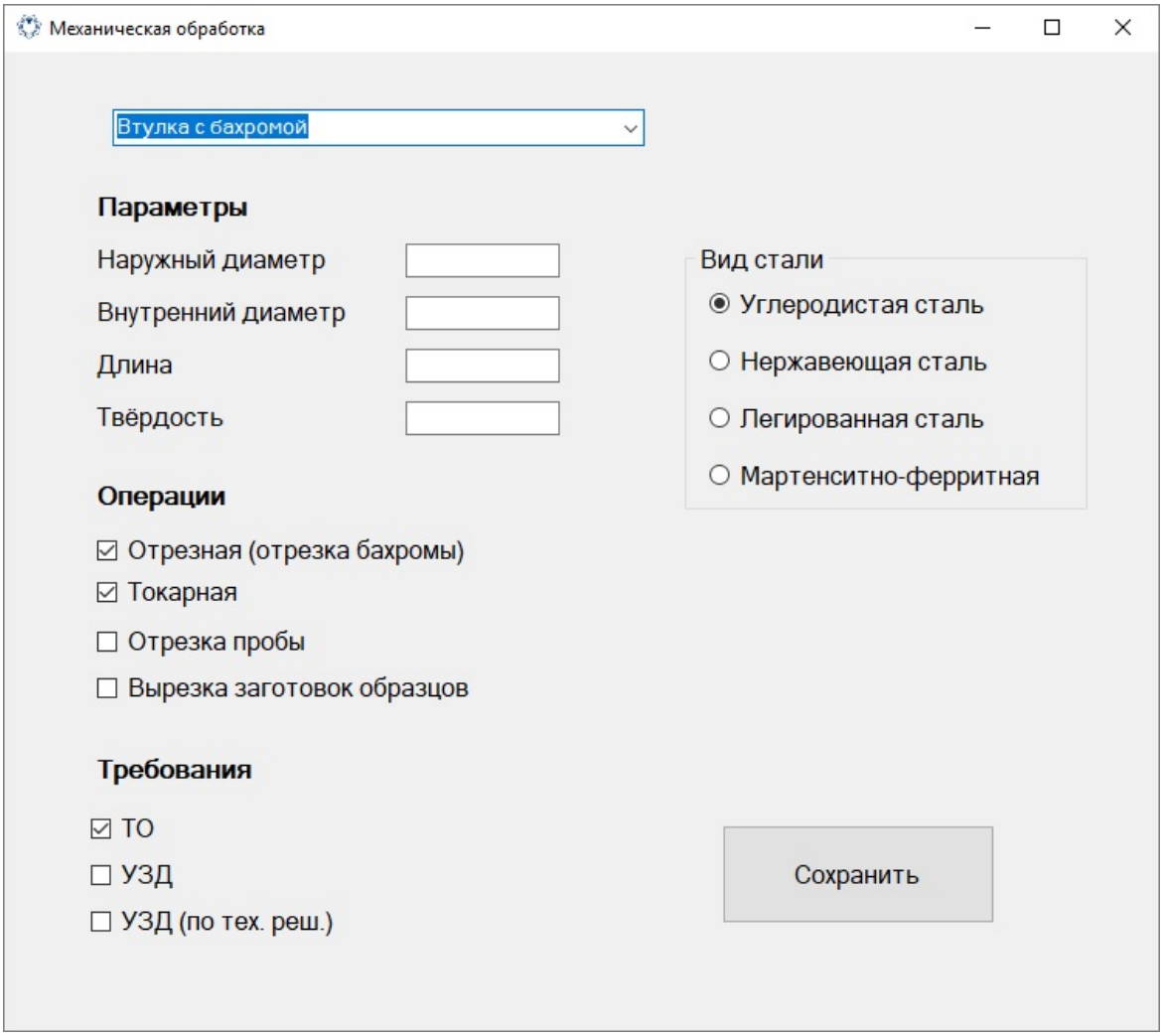

Рисунок 3 — Интерфейс в Windows Forms для втулки с бахромой

Пользовательский интерфейс в традиционных приложениях Windows не является масштабируемым, поэтому, при использовании мониторов с высоким разрешением, окно приложения становится маленьким и трудным для чтения, особенно это касается экранов ноутбуков, в которых плотность точек равна 120 dpi или 144, вместо традиционных 96 dpi. К примеру, на моём ноутбуке с разрешением 1920*1080, плотность точек будет равна $\frac{\sqrt{(1920^2+1080^2)}}{15} = 146{,}8$ dpi. При такой плотности, элементы управления и текст будут неудобно мелкими. WPF справляется с этой проблемой следующим образом: при более высоком разрешении, элементы управления визуализируются с бóльшим количеством пикселей [8].

Интерфейс приложения, реализованный с помощью Windows Presentation Foundation, представлен на рисунке 4.

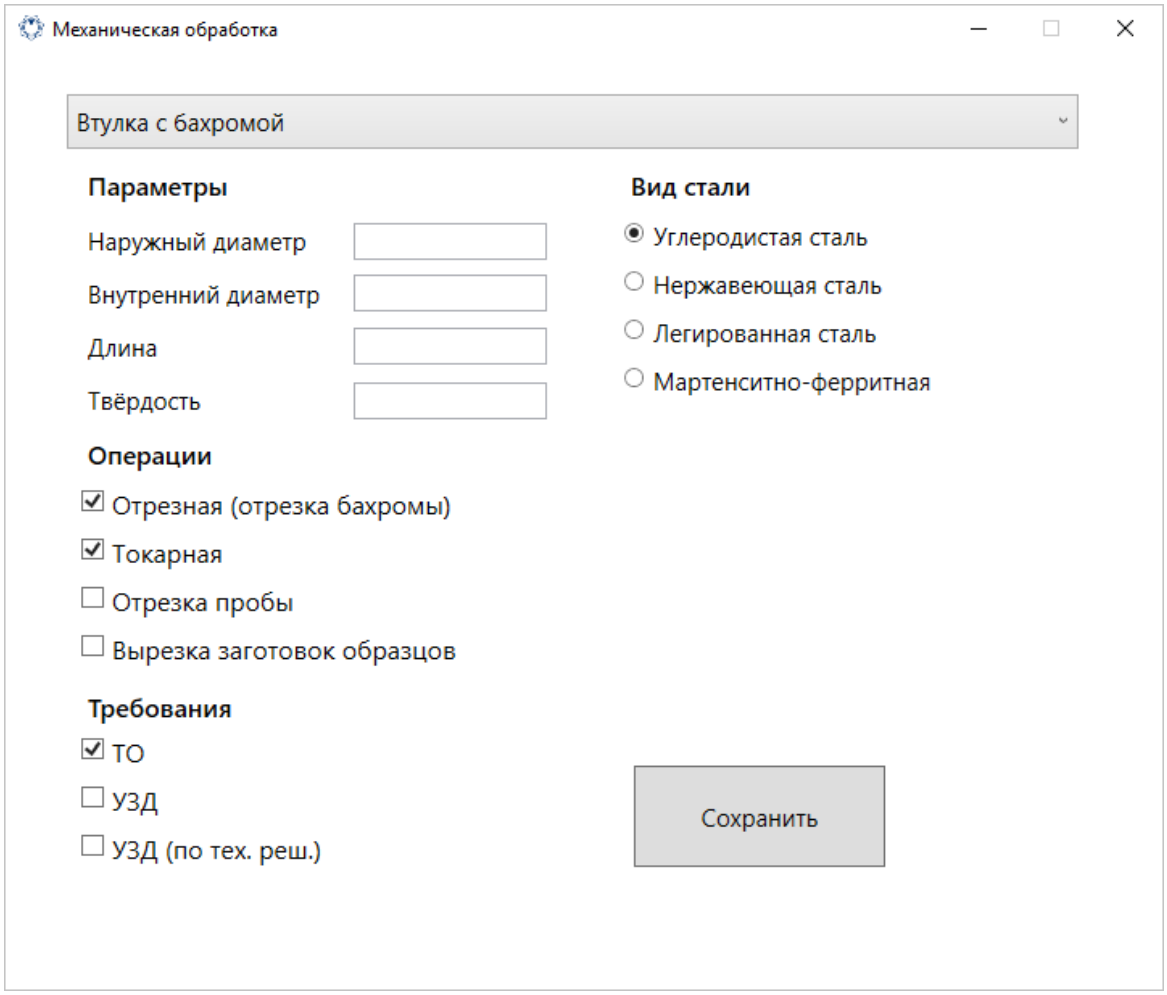

Рисунок 4 — Интерфейс в Windows Presentation Foundation для втулки с бахромой

Внешне интерфейс, выполненный в Windows Presentation Foundation не слишком отличается от традиционного в Windows Forms. При сравнении двух интерфейсов на ноутбуке при увеличенной системной плотности до 120 dpi, можно заметить, что шрифт, выполненный в Windows Forms менее чёткий. На рисунке 5 слева показан список параметров в Windows Forms, справа — в Windows Presentation Foundation.

В интерфейсе для втулки нужно учесть ещё один момент. При наличии операции «Отрезка пробы», возникает необходимость получить данные о твёрдости после термической обработки и длине пробы. Чтобы не загромождать интерфейс, эти параметры сделаем отображающимися только тогда, когда эта операция выбрана. Код для этого приведен в приложении Б.

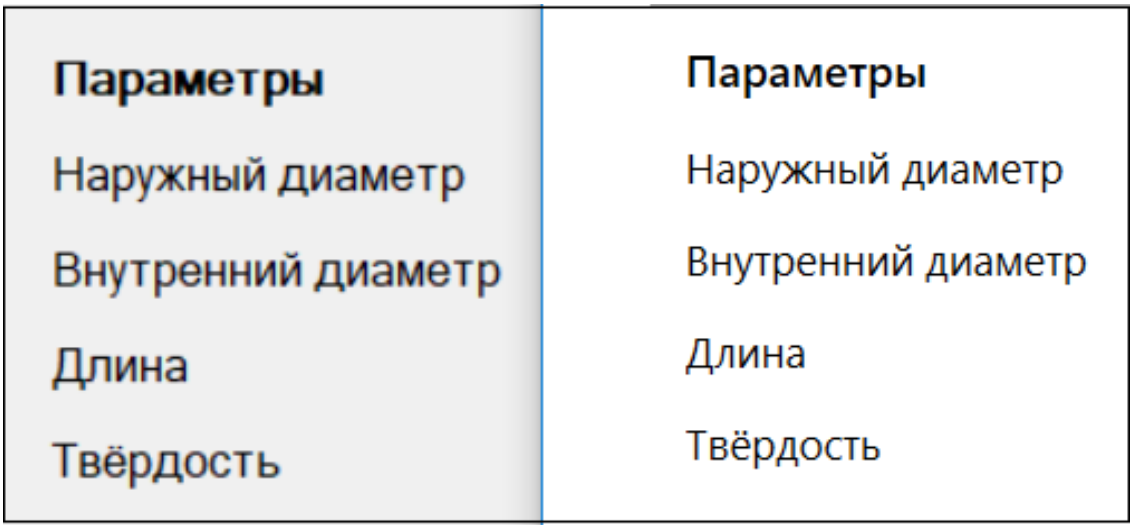

Рисунок 5 — Сравнение чёткости шрифтов при увеличенной системной плотности количества точек на дюйм в Windows Forms и Windows Presentation Foundation

При установке/снятии флажка перед операцией «Отрезка пробы» возникает событие CheckBoxOtrezProbyChanged, которое мы привязываем в атрибутах XAML кода для этого флажка <CheckBox x:Name="checkOtrezProby" Checked="CheckBoxOtrezProbyChanged" Unchecked="CheckBoxOtrezProby Changed" … />. При установке флажка, дополнительные текстовые поля и подписи к ним становятся видимыми, при снятии, скрываются. Окно интерфейса для втулки с выбранной операцией «Отрезка пробы» показано на рисунке 6.

Ввиду того, что деталей должно быть несколько и у них могут быть разные параметры, требования, необходимо предусмотреть возможность изменения интерфейса в зависимости от выбранной в Combo Box детали, для этого используем элемент управления Frame [19]. Frame представляет собой область окна приложения, в которой будет производиться переключение между страницами, на которых будет реализован интерфейс, соответствующий конкретной детали. На рисунке 7 представлен фрейм в среде разработке Visual Studio.

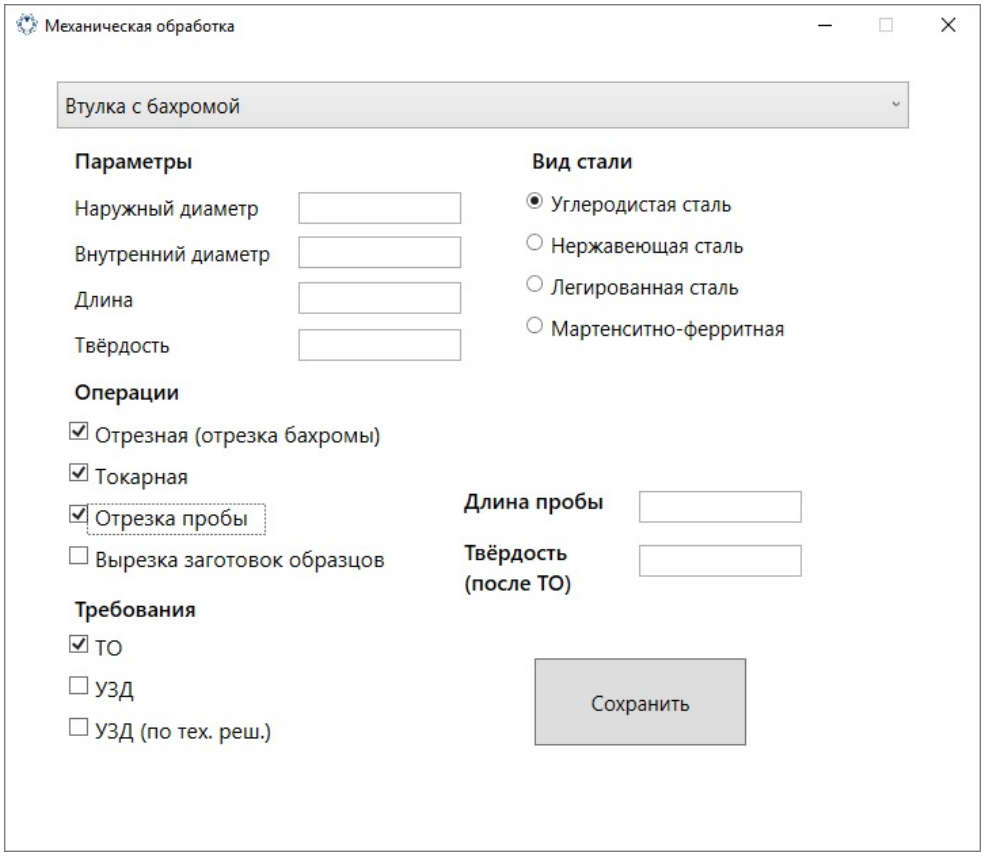

Рисунок 6 — Отображение скрытых дополнительных элементов управления при выборе операции «Отрезка пробы»

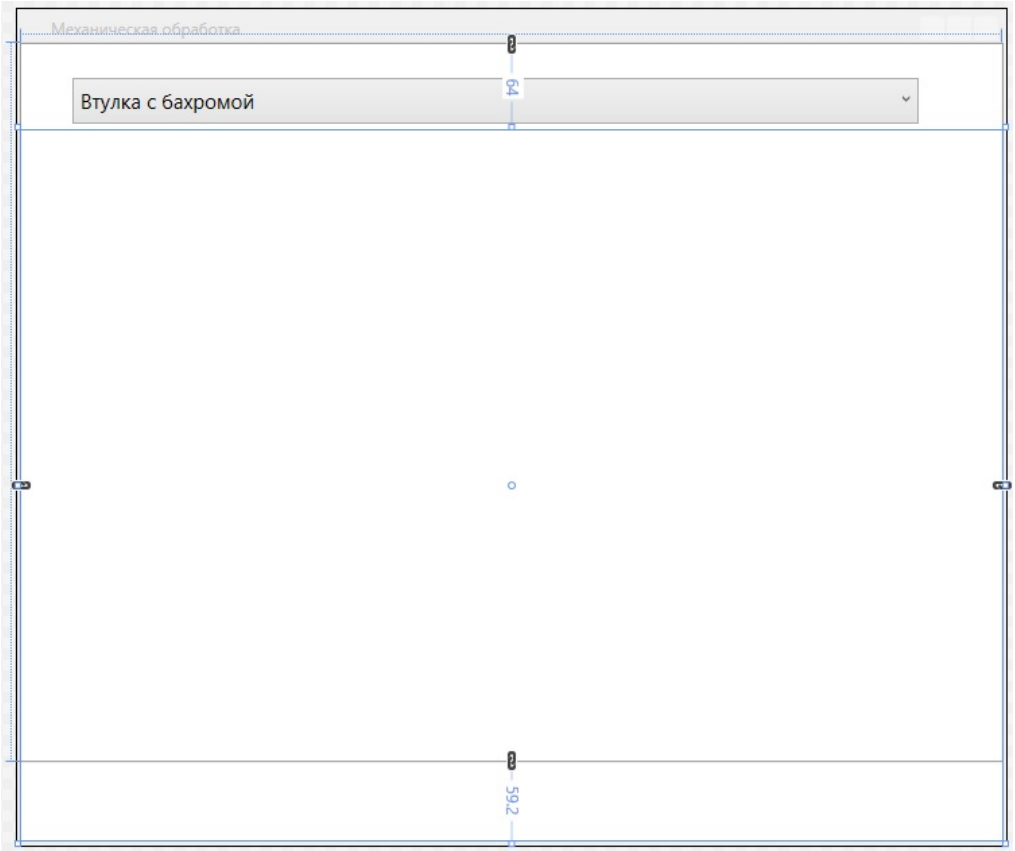

Рисунок 7 — Фрейм в среде разработки Visual Studio

На языке XAML код фрейма будет выглядеть так, как показано на рисунке 8.

```
<Frame Name="frame1" NavigationUIVisibility="Hidden" Margin="0,64,0,-59.2" />
```

Рисунок 8 — Код фрейма на языке XAML

Затем достаточно просто добавлять в приложение новые страницы [23] для новых деталей и производить переключение между страницами с помощью кода, представленного на рисунке 9.

В этом листинге, в зависимости от индекса выбранной в Combo Box детали, создаётся объект Page (страница), соответствующий странице с интерфейсом для данной детали и производится отображение этой страницы во фрейме через обращение к созданному объекту.

```
private void ComboBoxTitle_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
if (frame1 != null)
{
if (ComboBoxTitle.SelectedIndex == 0)
{
Page1 p1 = new Page1();
frame1.Navigate(p1);
}
}
}
```

Рисунок 9 — Код для перехода между станицами с разными элементами управления

Для детали типа «Пластина» интерфейс представлен на рисунке 10.

Из-за отличий в способе вычисления нормативов времени в интерфейсе появилось 2 набора параметров — это ковочные размеры (изначальные) и обдирочные (требуемые), а также изменились некоторые операции и требования.

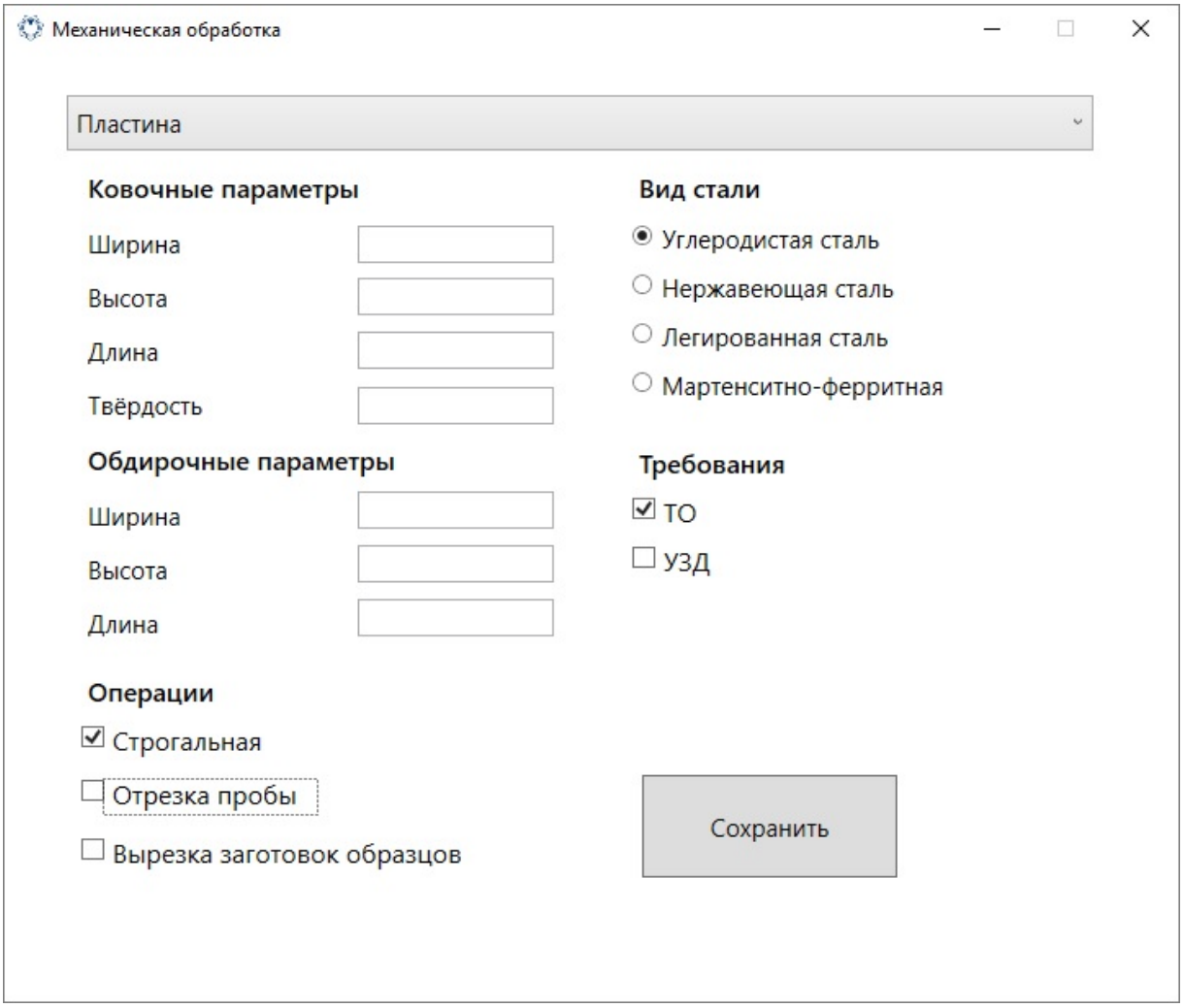

Рисунок 10 — Интерфейс для детали типа «Пластина»

2.2 Создание шаблона технологической карты

После создания интерфейса, нужно написать код, который будет выполнять основное предназначение программы: генерация технологической карты на основе полученных от пользователя данных.

Первым делом нужно обеспечить программное создание шаблона технологической карты в Microsoft Word, так как требуется, чтобы она была оформлена определённым образом.

Чтобы обеспечить работу с Microsoft Word, необходимо подключить библиотеку Microsoft.Office.Interop.Word в Visual Studio [25]. Для этого в Solution Explorer нажимаемой правой кнопкой мыши на References и выбираем

Add Reference, затем в Reference Manager нужно найти необходимую библиотеку и поставить флажок, как показано на рисунке 11.

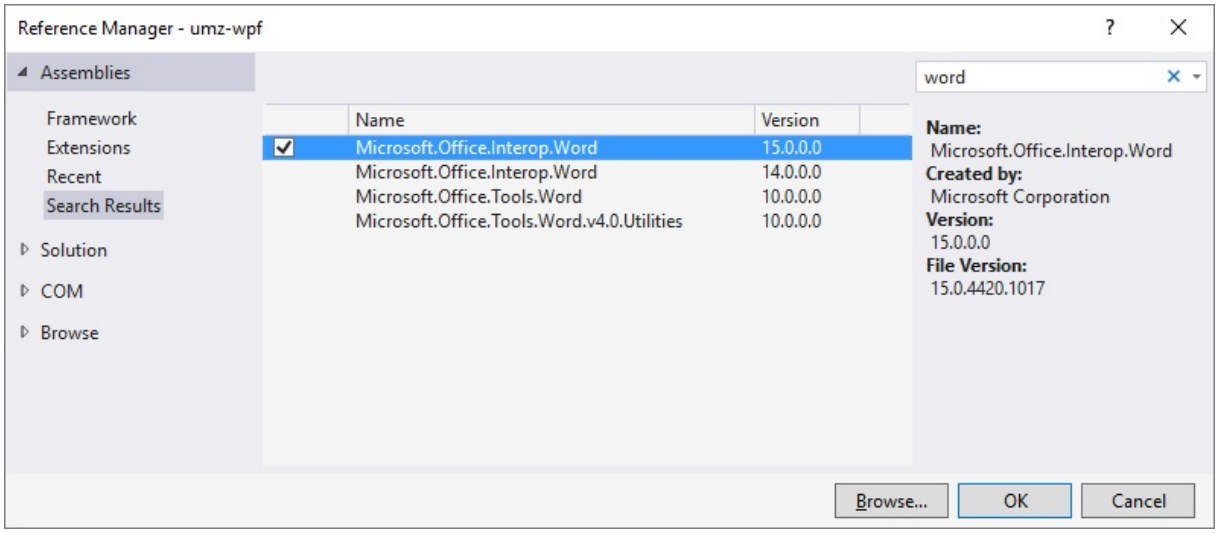

Рисунок 11 — Подключение библиотеки Microsoft.Office.Interop.Word

После объявления пространства имен using Microsoft.Office.Interop.Word; можно программно создать новый документ Microsoft Word с помощью кода, представленного на рисунке 12.

```
Microsoft.Office.Interop.Word.Application winword = new
Microsoft.Office.Interop.Word.Application();
Microsoft.Office.Interop.Word.Document document = new Document();
document = winword.Documents.Add();
```

Рисунок 12 — Код для создания нового документа Microsoft Word

Затем можно задать отступы на страницах созданного. Код для этого приведен на рисунке 13.

```
document.PageSetup.LeftMargin = 40;
document.PageSetup.RightMargin = 40;
document.PageSetup.BottomMargin = 40;
document.PageSetup.TopMargin = 40;
```

Рисунок 13 — Код для установки отступов в документе Microsoft Word

Для того чтобы можно было добавлять текст в созданный документ, нужно создать новый абзац (параграф), для этого используется код, представленный на рисунке 14.

```
Microsoft.Office.Interop.Word.Paragraph para1 =
document.Content.Paragraphs.Add();
```

Рисунок 14 — Код для создания нового абзаца в документе Microsoft Word

После создания нового абзаца, в него можно добавлять текст, изменять размер шрифта и совершать другие операции с текстом. Это осуществляется с помощью свойства Range, которое возвращает объект Range, представляющий часть документа, которая находится в пределах этого абзаца. В первый абзац шаблона технологической карты необходимо добавить текст о необходимости следования инструкциям безопасности, для этого используется код, приведенный на рисунке 15.

```
para1.Range.Font.Size = 10;
para1.Range.Text = "При выполнении работ соблюдать безопасные приемы
работы в соответствии с инструкциями по\n ТБ № ИОТ 01-2010, ИОТ 02-
2010, ИОТ 08-2016, ИОТ 14-2011, ИОТ 36-2014, ИОТ 71-2014";
```

Рисунок 15 — Код для добавления информации о необходимости следования инструкциям безопасности

Теперь создадим таблицу [26], которую изменим в соответствии с требованиями. Код, представленный на рисунке 16, создаст таблицу, в которой будет 13 строк и 20 ячеек в каждой строке.

```
Microsoft.Office.Interop.Word.Table firstTable =
document.Tables.Add(para1.Range, 13, 20);
```

Рисунок 16 — Код для создания таблицы

В созданной таблице количество ячеек выбрано заранее избыточным, чтобы обеспечить симметричность, поэтому часть ячеек нужно удалить. Для этого воспользуемся методом Merge, который позволяет объединить соседние ячейки.

Затем таблица будет отредактирована таким образом, чтобы она соответствовала требуемому внешнему виду для шаблона технологической кар-

ты. После этого таблица заполняется необходимыми данными. Полный код создания шаблона представлен в приложении Б. Получившийся в результате шаблон представлен на рисунке 17.

Чтобы пользователь мог в любое время получить доступ к сформированной технологической карте, необходимо реализовать механизм сохрания документа Microsoft Word [15]. Для открытия диалогового окна «Сохранить как», автоматического добавления расширения файла и последующего его сохранения используется код, который приведен на рисунке 18.

При выполнении работ соблюдать безопасные приемы работы в соответствии с инструкциями по ТБ № ИОТ 01-2010, ИОТ 02-2010, ИОТ 08-2016, ИОТ 14-2011, ИОТ 36-2014, ИОТ 71-2014

Т Б М О	Технологическая карта								
Наименование			Обозначение						
№ заказа			№ чертежа						
Шифр заказчика			Марка						
Опросный лист			Отходы						
Кол.дет. в пок./масса			Стандарт						
Размеры поковки			Группа						
№	УЧ	ОПЕР	КОД, НАИМЕНОВАНИЕ ОПЕРАЦИИ, ОБОРУДОВАНИЯ			ПРОФ	Р	ЕН	Тпз	Тшт
Технолог										
Нач. ТБМО										
Нормировщик										
Утвердил										

Рисунок 17 — Шаблон технологической карты в Microsoft Word

```
SaveFileDialog saveFileDialog = new SaveFileDialog();
saveFileDialog.Filter = "Файл Microsoft Word (*.docx)|*.docx|All files
(*.*)|*.*";
saveFileDialog.FilterIndex = 0;
saveFileDialog.RestoreDirectory = true;
saveFileDialog.ShowDialog;
if (saveFileDialog.FileName == "") return;
document.SaveAs2(saveFileDialog.FileName);
```

Рисунок 18 — Код для сохранения документа Microsoft Word

2.3 Создание чертежей

Теперь, когда шаблон технологической карты готов, перейдем к его заполнению. Сначала создадим чертежи деталей.

Для того чтобы создать чертёж с помощью языка программирования C#, используем класс Bitmap из пространства имен System.Drawing и создадим объект этого класса [13], код для создания объекта класса Bitmap приведен на риунке 19.

```
Bitmap b = new Bitmap(560, 480);
```

Рисунок 19 — Код для создания объекта класса Bitmap

Этот объект будет использоваться в качестве «холста», на котором будет производиться непосредственное черчение с помощью объекта из класса System.Drawing.Graphics. Создание объекта Graphics, который будет привязан к созданному выше объекту Bitmap, приведено на рисунке 20.

```
System.Drawing.Graphics graphicsObj =
System.Drawing.Graphics.FromImage((System.Drawing.Image)b);
```

Рисунок 20 — Код для создания объекта класса Graphics

Полный код создания чертежа втулки приведен в приложении Б.

Для указания размеров используем чертёжный шрифт GOST 2.304–81 type A. Так как этот шрифт не входит в стандартный пакет шрифтов Windows, следует подключить его к приложению, для этого сначала нужно сделать новый шрифт частью решения.

В Solution Explorer (обозреватель решения) необходимо нажать правой кнопкой мыши на папке Resources и выбрать Add Existing Item… (добавить существующий объект) затем указать путь к шрифту. Чтобы шрифт стал частью сборки, в свойствах ресурса нужно свойству Build Action присвоить параметр Embedded Resource.

Теперь необходимо реализовать возможность использования нового шрифта [31], для этого подключим динамически подключаемую библиотеку gdi32.dll и объявим методы для добавления шрифта, как ресурса. Код для этих действий приведен на рисунке 21.

```
[DllImport("gdi32.dll")]
private static extern IntPtr AddFontMemResourceEx(IntPtr pbFont, uint
cbFont, IntPtr pdv, [In] ref uint pcFonts); public static
PrivateFontCollection private_fonts = new PrivateFontCollection();
```

Рисунок 21 — Подключение библиотеки gdi32.dll

Добавим метод для подгрузки шрифта. В этом методе подключается поток ресурсов, затем создаётся небезопасный блок памяти для данных, после этого создаётся буферный массив, в который будет произведено чтение потока, затем данные копируются из этого массива в небезопасный блок памяти, шрифт регистрируется в системе, добавляется в коллекцию шрифтов, наконец, поток закрывается и очищается небезопасный блок памяти. Метод для подгрузки шрифта представлен на рисунке 22.

```
public static void LoadFont(string FontResourceName)
{
Stream fontStream =
System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceS
tream(FontResourceName);
System.IntPtr data = Marshal.AllocCoTaskMem((int)fontStream.Length);
Byte[] fontData = new Byte[fontStream.Length];
fontStream.Read(fontData, 0, (int)fontStream.Length);
Marshal.Copy(fontData, 0, data, (int)fontStream.Length);
uint cFonts = 0;
AddFontMemResourceEx(data, (uint)fontData.Length, IntPtr.Zero, ref
cFonts);
private_fonts.AddMemoryFont(data, (int)fontStream.Length);
fontStream.Close();
Marshal.FreeCoTaskMem(data);
}
```

Рисунок 22 — Метод для подгрузки шрифта

Вся инфраструктура для работы с подключаемым шрифтом GOST 2.304–81 type A реализована, теперь можно приступить к добавлению размеров втулки на чертёж [4]. Код для добавления размеров приведен в приложении Б.

На рисунках 23 и 24 представлены параметры втулки и соответствующий им чертёж.

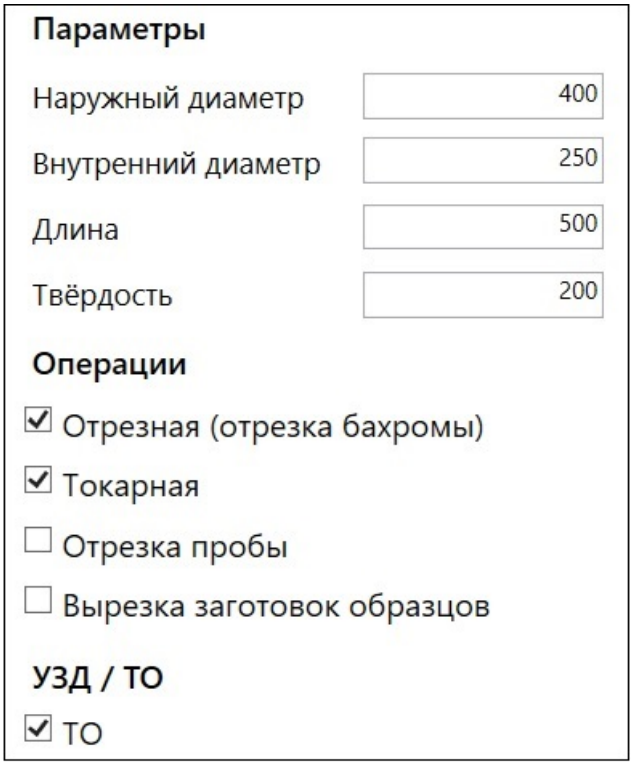

Рисунок 23 — Набор параметров для чертежа втулки

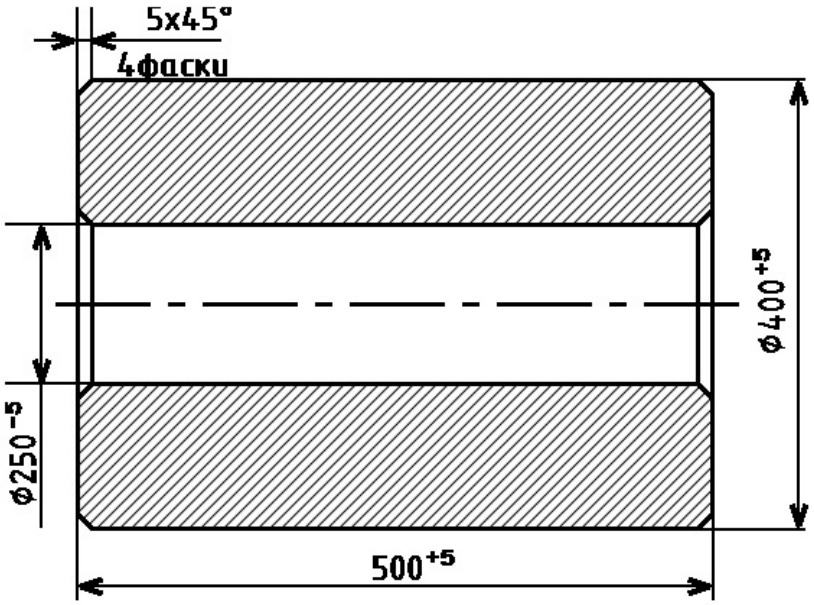

Рисунок 24 — Получившийся чертёж втулки

Для детали типа «Пластина» чертёж строится подобным образом, что и для втулки: чертятся 3 многоугольника, соответствующих виду спереди и сбоку, затем добавляются размерные линии и подписываются размеры. Код для построения чертежа пластины приведен в приложении Б.

Пример получившегося чертежа приведен на рисунке 25.

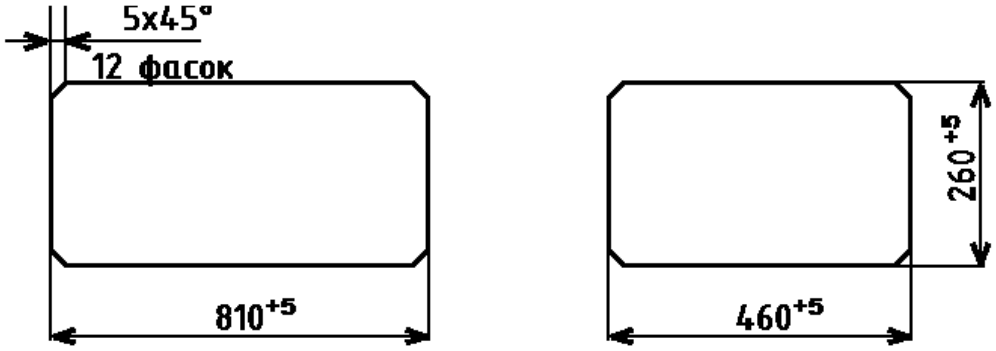

Рисунок 25 — Пример чертежа пластины

2.4 Класс «Втулка с бахромой»

Класс Vtulka_s_bachromoj (Втулка с бахромой) реализует возможность создавать объекты этого класса с определенным набором свойств (размеры, твёрдость) и позволит получать информацию об оборудовании, нормативах времени для различных операций, производимых с втулками для конкретного набора свойств.

Класс Vtulka_s_bachromoj будет иметь такие поля, как наружный диаметр, внутренний диаметр, длина втулки, твёрдость, длина пробы, твёрдость после термической обработки. Код для объявления данных полей представлен на рисунке 26.

```
private int naruzhnij_diametr;
private int vnutrennij_diametr;
private int dlina;
private int hardness;
private int dlina_proby = 0;
private int tverdPosleTO;
```

Рисунок 26 — Поля класса Vtulka_s_bachromoj

В конструкторе класса будут определяться первые 4 поля, которые являются обязательными параметрами. Конструктор класса Vtulka_s_bachromoj приведен на рисунке 27.

```
public Vtulka_s_bachromoj(int naruzhnij_diametr, int
vnutrennij_diametr, int dlina, int hardness)
{
this.naruzhnij_diametr = naruzhnij_diametr;
this.vnutrennij_diametr = vnutrennij_diametr;
this.dlina = dlina;
this.hardness = hardness;
}
```

Рисунок 27 — Конструктор класса Vtulka_s_bachromoj

Методы для получения и изменения значений членов (полей) класса представлены в приложении Б.

Определим метод для получения кода пилы. На предприятии используется 3 пилы. Пила выбирается зависимости от величины наружного диаметра. Если наружный диаметр равен 300 мм и меньше, будет использоваться пила с кодом 4812, если наружный диаметр равен 450 мм и меньше, используется пила с кодом 4814, при наружном диаметре 1100 мм и меньше — пила с кодом 4816. Метод для определения кода пилы представлен на рисунке 28.

```
public int get_code_of_saw()
{
int code = 0;
if (naruzhnij_diametr <= 300) code = 4812;
else if (naruzhnij_diametr <= 450) code = 4814;
else if (naruzhnij_diametr <= 1100) code = 4816;
return code;
}
```

Рисунок 28 — Метод для определения кода пилы

Теперь создадим метод, с помощью которого будет определяться норматив времени для отрезной операции. Норматив времени определяется сле-

дующей формулой $\frac{D}{K*60}$, где D — наружный диаметр втулки, а K — коэффициент, который берётся из таблицы и зависит от величины наружного диаметра и твёрдости.

Таблица 1 — Коэффициенты для формулы определения норматива времени отрезной операции

D \ H	≤200	≤247	≤269	≥270
≤450	5	3	2,5	1,6
≤1100	6	3,5	3,0	2,0

Для определения норматива времени токарной операции, используется таблица 2. В данной таблице D — наружный диаметр втулки, d — внутренний диаметр, а L — длина. Общий норматив определяется как сумма норматива для обработки наружного диаметра и норматива для обработки внутреннего диаметра. Код метода для определения норматива времени токарной операции приведен в приложении Б.

Таблица 2 — Нормативы времени для токарной операции

D	d \ L	80	120	160	200	240	280	320	360	400	450	500	550	600
225	140	1,3	1,6	1,8	2,2	2,5	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3
250	160	1,6	1,8	2,2	2,5	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6
275	180	1,9	2,2	2,5	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0
300	200	2,2	2,5	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3
325	220	2,5	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9
350	240	2,8	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1
400	260	3,1	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5
450	280	3,5	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8
500	300	3,8	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1
550	320	4,2	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4
600	340	4,5	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4	8,7
650	360	5,0	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4	8,7	9,0
700	380	5,3	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4	8,7	9,0	9,4
750	400	5,6	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4	8,7	9,0	9,4	9,8
800	420	6,0	6,3	6,9	7,1	7,5	7,8	8,1	8,4	8,7	9,0	9,4	9,8	10,2

Код метода для определения норматива времени отрезной операции представлен в приложении Б.

2.5 Класс «Пластина»

Класс Plastina (Пластина) будет позволять создавать объекты этого класса с присущими реальной детали этого типа параметрами. Этот класс, подобно классу Vtulka_s_bachromoj позволит получать информацию об оборудовании, а самое главное о нормативах времени для различных операций.

Класс Plastina будет иметь поля, представленные на рисунке 29. Они соответствуют параметрам, представленным в интерфейсе для детали типа «Пластина».

В конструкторе класса будут определяться семь обязательных полей. Конструктор класса Plastina приведен на рисунке 30.

```
private int height;
private int width;
private int length;
private int hardness;
private int heightObd;
private int widthObd;
private int lengthObd;
private int dlinaProby = 0;
private int hardnessProba = 0;
```

Рисунок 29 — Поля класса Plastina

```
public Plastina(int height,int width, int length, int hardness, int
heightObd, int widthObd, int lengthObd)
{
this.height = height;
this.width = width;
this.length = length;
this.hardness = hardness;
this.heightObd = heightObd;
this.widthObd = widthObd;
this.lengthObd = lengthObd;
}
```

Рисунок 30 — Конструктор класса Plastina

Для всех полей также определим соответствующие им свойства, которые, в случае необходимости, позволят получить, либо установить значения полей. Именно с помощью свойств можно будет изменять поля dlinaProby (длина пробы) и hardnessProba (твёрдость при отрезке пробы), в которых появляется необходимость лишь при наличии операции «отрезка пробы», для этих двух полей свойства будут выглядеть так, как показано на рисунке 31.

Метод для определения кода пилы в классе Plastina будет выглядеть подобным образом, что и в классе Vtulka_s_bachromoj, разница лишь в том, что код пилы будет зависеть от максимального значения из высоты и ширины, а не от наружного диаметра. Метод для определения кода пилы представлен в приложении Б.

```
public int HardnessProba
{
get { return hardnessProba; }
set { hardnessProba = value; }
}
public int DlinaProby
{
get { return dlinaProby; }
set { dlinaProby = value; }
}
```

Рисунок 31 — Свойства класса Plastina

Метод, с помощью которого будет определяться норматив времени для отрезной операции, также будет выглядеть подобно тому, что использовался в классе Vtulka_s_bachromoj. Вместо наружного диаметра в формуле $\frac{D}{K*60}$, будет использоваться среднее квадратическое высоты и ширины пластины $\frac{\sqrt{\frac{a^2+b^2}{2}}}{K*60}$, где a — высота, b — ширина, K — коэффициент из таблицы 3. Код метода приведен в приложении Б.

Таблица 3 — Коэффициенты для формулы определения норматива времени отрезной операции

H / Max(a,b)	≤200	≤247	≤269	≥270
≤450	5	3	2,5	1,6
≤1100	6	3,5	3,0	2,0

В классе Vtluka_s_bachromoj отсутствовал метод для вычисления норматива времени для установки детали на станок, так как этот норматив был постоянным, для пластины это не так, поэтому нужно определить метод, вычисляющий этот норматив времени.

Норматив времени будет зависеть от веса и определяться по таблице 4.

Так как обрабатывать нужно 6 сторон, то установка детали осуществляется 6 раз, то есть каждое значение Тпз из таблицы 4 умножается на 6. Вес будем вычислять по следующей формуле: $Q = a * b * c * 7850$, где a, b, c — размеры пластины в метрах, а 7850 кг/м^3 — удельный вес стали.

Таблица 4 — Нормативы времени для установки детали в зависимости от веса

Q, кг	1	3	5	6	12	20	30	60
Тпз	0,029	0,04	0,047	0,053	0,06	0,04	0,08	0,195
Q, кг	100	200	500	1000	3000	8000	15000	
Тпз	0,23	0,275	0,333	0,409	0,475	0,505	0,77	

Метод для определения норматива времени для установки детали на строгальный станок представлен в приложении Б.

В классе Plastina появляется ещё один особый метод, который позволит вычислять норматив времени для снятия фасок. Этот норматив времени будет зависеть от длины строгания, то есть от длины каждой стороны детали и определяться по таблице 5.

В таблице 5 представлены нормативы для одной фаски, а всего их будет 12, 4 для каждого размера (длины, ширины, высоты). Метод для определения норматива времени для снятия фасок представлен в приложении Б.

Таблица 5 — Нормативы времени для строгания фасок

Длина строгания, мм до	500	1000	1500	2000
Норматив времени	0,023	0,025	0,026	0,027

Нормативы времени на строгание пластины определяются следующим образом: берётся ковочный размер с допуском и из него вычитается обдирочный размер, получившаяся разница делится на глубину резания (8 для ширины и высоты, 5 для длины), в результате получается число проходов. Чтобы число проходов было целым, округление до верхнего целого числа будет производиться с помощью метода Ceiling(), класса Math. Затем по таблицам 6 и 7 определяется норматив времени для соответствующей ширины и длины обработки, этот норматив умножается на число проходов. Этот процесс производится для всех трех размеров.

Таблица 6 — Нормативы времени на строжку плоскостей для длины обработки 900 и менее

W / L	20	30	50	75	100	150
100	0,023	0,028	0,033	0,038	0,048	0,057
200	0,028	0,033	0,043	0,057	0,066	0,058
300	0,036	0,038	0,052	0,067	0,085	0,11
400	0,038	0,043	0,067	0,085	0,104	0,142
500	0,043	0,048	0,071	0,095	0,12	0,16
600	0,047	0,057	0,076	0,1	0,14	0,19
700	0,052	0,061	0,085	0,12	0,15	0,21
800	0,057	0,066	0,095	0,133	0,17	0,23
900	0,066	0,071	0,1	0,14	0,19	0,26
W / L	200	250	300	350	400	450
100	0,067	0,08	0,09	0,1	0,11	0,13
200	0,104	0,133	0,152	0,17	0,2	0,22
300	0,142	0,17	0,2	0,24	0,27	0,3
400	0,18	0,22	0,27	0,3	0,35	0,4
500	0,21	0,26	0,31	0,36	0,41	0,46
600	0,25	0,3	0,36	0,43	0,49	0,54
700	0,28	0,35	0,41	0,49	0,55	0,62
800	0,31	0,38	0,46	0,54	0,62	0,69
900	0,35	0,43	0,52	0,6	0,69	0,78

Таблица 7 — Нормативы времени на строжку плоскостей для длины обработки 2000 и менее

W / L	20	30	50	75	100	125	150
1000	0,076	0,095	0,127	0,19	0,21	0,26	0,29
1200	0,08	0,11	0,142	0,2	0,25	0,29	0,34
1600	0,114	0,12	0,17	0,23	0,28	0,34	0,41
2000	0,12	0,14	0,2	0,26	0,34	0,42	0,48
W / L	175	225	300	400	550	750	1000
1000	0,36	0,4	0,53	0,7	1	1,3	1,74
1200	0,38	0,49	0,64	0,81	1,11	1,49	1,96
1600	0,47	0,60	0,76	1	1,35	1,81	2,41
2000	0,55	0,69	0,89	1,13	1,60	2,12	2,89

Метод определения норматива времени для строгальной операции представлен в приложении Б.

В этом методе присутствует приватный метод get_table_time_norm(), который и производит выбор табличных нормативов на основе длины и ширины.

При наличии строгальной операции под ультразвуковую дефектоскопию, добавляется ещё один проход, для того, чтобы была выдержана шероховатость поверхности определенного уровня, нормативы времени для этого прохода вычисляются следующим образом: для каждой грани берутся её размеры и в таблице 8 выбираются соответствующие размерам грани длина и ширина, обрабатываемой плоскости. После этого все полученные нормативы складываются. Метод, позволяющий вычислить нормативы времени для строгальной операции под УЗД представлен в приложении Б.

Таблица 8 — Нормативы времени для строгания под ультразвуковую дефектоскопию

W / L	50	75	100	150	200	250	300	350
500	0,11	0,12	0,13	0,16	0,19	0,22	0,25	0,28
1000	0,14	0,16	0,19	0,23	0,3	0,37	0,43	0,48
1500	0,16	0,18	0,21	0,28	0,35	0,43	0,5	0,58
2000	0,18	0,20	0,26	0,33	0,43	0,53	0,63	0,65

Окончание таблицы 8

W / L	400	450	500	600	700	800	900	1000
500	0,32	0,35	0,38	0,43	0,5	0,57	0,62	0,68
1000	0,55	0,63	0,68	0,78	0,92	1,05	1,17	1,3
1500	0,65	0,73	0,8	0,92	1,08	1,25	1,38	1,48
2000	0,83	0,92	1,03	1,18	1,38	1,57	1,78	1,97

2.6 Формирование информации об операциях и нормативах времени в технологической карте для втулки

Пришло время перейти к заключительному этапу формирования технологической карты для механической обработки втулки. На этом этапе будет осуществляться работа с текстовой информацией. В технологическую карту будут занесены описания необходимых операций, а также их нормативы времени.

Первым делом, создадим объект класса Vtulka_s_bachromoj и объявим переменную, в которой будет находиться вся текстовая информация. Код для выполения этих действий представлен на рисунке 32.

```
Vtulka_s_bachromoj vtulka = new Vtulka_s_bachromoj(dlina,
naruzhnij_diametr,vnutrennij_diametr,hardness);
string result = "";
```

Рисунок 32 — Создание объекта класса «Втулка с бахромой» и переменной result

Если сталь, из которой изготовлена втулка, является углеродистой, то в зависимости от операции применяются понижающие коэффициенты — 0.8 для токарной операции под термическую обработку, 0.7 для токарной операции под ультразвуковую диагностику.

Для нержавеющей стали устанавливается повышающий коэффициент 2.

Для легированной и мартенситно-ферритной стали устанавливаются повышающие коэффициенты в зависимости от твёрдости, при твёрдости менее 174 коэффициент равен 1, при твёрдости менее 203 — коэффициент ра-

вен 1.2, при твёрдости менее 260 — 1.3, при твёрдости более 260 коэффициент равен 1.5 (1.7 для мартенситно-ферритной).

Код, устанавливающий коэффициенты на нормативы времени для токарной операции в зависимости от вида стали, приведен в приложении Б.

При наличии отрезной операции, в итоговую переменную result помещаем информацию о коде пилы, название операции, нормативы времени, последовательность действий и требуемые размеры. Код формирования информации об отрезной операции, представлен в приложении Б.

Для токарной операции текстовая информация, добавляемая в итоговую переменную, будет выглядеть подобно отрезной, но со своим содержанием. Кроме того, информация меняется в зависимости от требований (ТО/УЗД). Код для формирования описания токарной операции приведен в приложении Б.

После завершения любой из предыдущих операций, втулка должна быть подвергнута контролю для проверки размеров, а затем, в зависимости от требований проверена на наличие дефектов ультразвуковой диагностикой, либо отправлена для прохождения термической обработки. Код для указания этих действий представлен в приложении Б.

Если присутствуют операции «Отрезка пробы» и «Вырезка заготовок образцов» то в итоговую переменную result также нужно добавить соответствующую информацию. Код для этого приведен в приложении Б.

Накопившуюся в переменной result информацию помещаем в шаблон технологической карты с помощью кода [5], представленного на рисунке 33.

```
firstTable.Rows[9].Cells[1].Range.Text = result;
```

Рисунок 33 — Код для помещения информации из переменной result в шаблон

Таким образом, в зависимости от заданных пользователем исходных данных содержимое переменной result будет меняться. Для исходных данных, приведенных на рисунке 34, полученная текстовая информация, выведенная из переменной result в шаблон технологической карты, будет выглядеть так, как показано на рисунке 35.

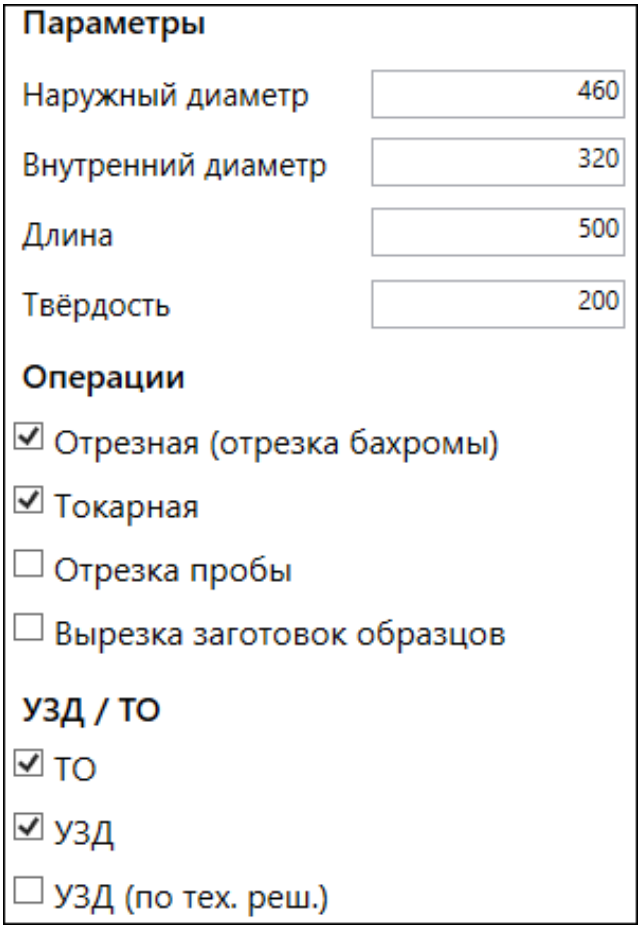

Рисунок 34 — Некоторый набор исходных данных

```
4816            Отрезная                                460  3  1  0.50  2.56

  Установить, выверить и закрепить.
  Отрезать с переустановкой бахрому в размер 520 (2 реза)

4150            Токарная                                474  3  1  0.42  12.24

  Установить, выверить и закрепить деталь в кулаках.
  С переустановкой детали точить Ф465, расточить Ф315 и
  подрезать торцы в размер 505 под УЗК.
  Выдержать шероховатость Ra 3.2. Снять фаски 5х45° по
  СТП 203-2010 п.2.11

                  Контрольная

  Подвергнуть деталь контролю ультразвуком согласно чертежу.
  Отправить на ТО.
```

Рисунок 35 — Текстовая информация для набора исходных данных из рисунка 34

2.7 Формирование информации об операциях и нормативах времени в технологической карте для пластины

Сначала необходимо инициализировать переменные для хранения текстовой информации и итогового норматива времени для строгальной операции и создать объект класса Plastina. Код для этих действий приведен на рисунке 36.

```
string result = "";
double resultStrogNorm = 0;
Plastina plastina = new
Plastina(height,width,length,hardness,heightObd,widthObd,lengthObd);
```

Рисунок 36 — Инициализация переменных и создание объекта класса Plastina

Затем, если имеется строгальная операция, аналогично, как и для втулки, устанавливаются коэффициенты для разных видов сталей и твёрдостей.

Далее, в зависимости от выбранных операций, переменная result, будет наполняться информацией об этих операциях и нормативами времени. Полный код для этих действий приведен в приложении Б.

3 КОНТРОЛЬ ВВОДИМЫХ ДАННЫХ И ТЕСТИРОВАНИЕ

3.1 Контроль вводимых пользователем исходных данных для детали типа «Втулка»

При работе с программой пользователь может ошибиться, задав неверные исходные данные по своей невнимательности, либо из-за неумения пользоваться программой, непонимания того, что от него требуется. Таким образом, при разработке программного обеспечения должны быть предусмотрены механизмы для контроля вводимых пользователем исходных данных.

В интерфейсе для втулки имеется четыре текстовых поля, в которые обязательно должны быть введены некоторые целочисленные значения (размеры в миллиметрах и твёрдость). Чтобы удостовериться, что пользователь действительно ввёл целочисленные значения, используем метод int.TryParse(), который пытается считать введенную в текстовое поле информацию в переменную типа int и при удачном исходе возвращает true, а при неудачном false [21]. Если пользователь ввёл что-то отличное от целого числа, либо не ввёл ничего, он будет уведомлен о некорректности введенных данных всплывающим сообщением, после чего выполнение программы будет остановлено. Для поля «Наружный диаметр» этот код представлен на рисунке 37.

Далее должна быть осуществлена проверка на соответствие введенных данных диапазону значений, с которым может работать программа. Для размеров втулки диапазон значений устанавливает таблица 2, для твёрдости возьмем нижней границей твёрдость алюминия (15 HB), за верхнюю границу инструментальную сталь (700 HB). Код для проверки вхождения введенных данных в диапазон допустимых значений приведен в приложении Б.

```
if (!int.TryParse(textNarD.Text, out naruzhnij_diametr))
{
MessageBox.Show("Наружный диаметр втулки указан некорректно");
return;
}
```

Рисунок 37 — Код для уведомления пользователя о некорректности введенного размера наружного диаметра

Кроме того, нужно проконтролировать логичность введенных данных. Внутренний диаметр не может быть большим, либо равным наружному, код для такой проверки приведен на рисунке 38.

```
if (vnutrennij_diametr >= naruzhnij_diametr)
{
MessageBox.Show("Внутренний диаметр не может быть больше наружного");
return;
}
```

Рисунок 38 — Код для уведомления пользователя о том, что внутренний диаметр не может быть больше, чем наружный

Следующий момент — это список операций. Операции могут присутствовать как все вместе, так и в любой комбинации. Если не выбрано ни одной операции, в этом будет мало смысла, но возможно пользователю будет нужен шаблон с чертежом, поэтому не стоит считать отсутствие выбранных операций за ошибку.

При выборе операции «Отрезка пробы» появляются два текстовых поля — длина пробы и твёрдость после ТО. Для длины пробы укажем диапазон возможных значений от 5 до 500, а для твёрдости от 15 до 700. Код для выполнения этих действий приведен на рисунке 39.

И последние элементы интерфейса, в которых можно предупредить заведомо ошибочные действия пользователя, это требования. Все они относятся к токарной операции, поэтому если токарная операция присутствует, то должно присутствовать одно из требований, если же токарная операция от-

сутствует, то они должны стать недоступными для выбора. Код для этой проверки приведн в приложении Б.

```
if (!int.TryParse(textDlinaProby.Text, out dlina_proby) ||
(dlina_proby<5 || dlina_proby > 500) )
{
this.Cursor = Cursors.Arrow;
MessageBox.Show("Длина пробы указана некорректно");
return;
}
vtulka.set_dlina_proby(dlina_proby);
if (!int.TryParse(textTverdPosleTO.Text, out tverd_posle_TO) ||
(tverd_posle_TO<15 || tverd_posle_TO >700))
{
this.Cursor = Cursors.Arrow;
MessageBox.Show("Твёрдость после ТО указана некорректно");
return;
}
```

Рисунок 39 — Код для установки диапазона возможных значений для длины пробы и твёрдости

На рисунке 40 показано уведомление, возникающее при выборе токарной операции в отсутствие требований.

Кроме того, не могут быть одновременно выбраны требования «УЗД» и «УЗД по техническому решению». Код, представленный на рисунке 41, будет автоматически снимать выбор с одного при выборе другого.

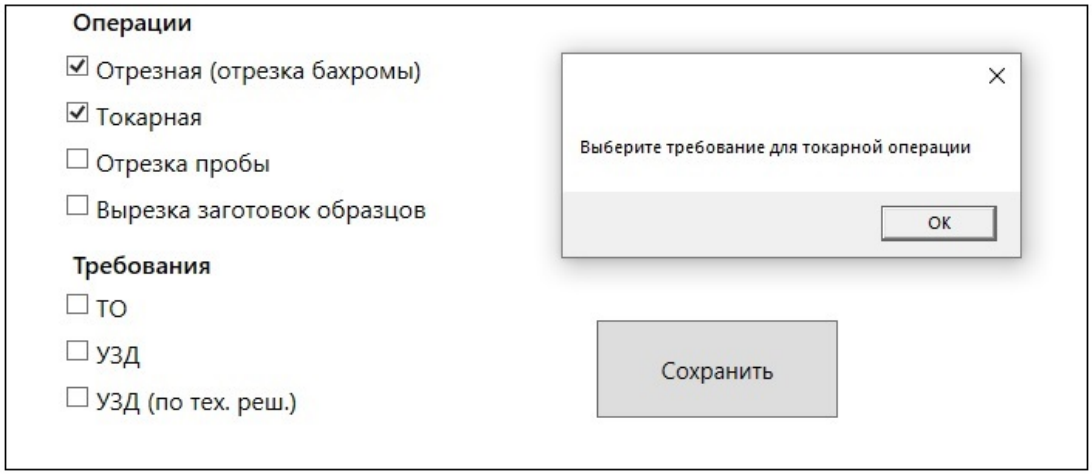

Рисунок 40 — Уведомление, возникающее при отсутствии выбранных требований

```
private void checkBoxUZD_Checked(object sender, RoutedEventArgs e)
{
if (checkBoxUZD.IsChecked == true) checkBoxUZDTech.IsChecked = false;
}
private void checkBoxUZDTech_Checked(object sender, RoutedEventArgs e)
{
if (checkBoxUZDTech.IsChecked.Value == true) checkBoxUZD.IsChecked =
false;
}
```

Рисунок 41 — Код для предотвращения выбора двух несовместимых требований

3.2 Контроль вводимых пользователем исходных данных для детали типа «Пластина»

Для детали типа «Пластина», вводимые данные, как и в случае с деталью типа «Втулка», должны быть целочисленными. Диапазон допустимых значений определяется таблицами 6, 7, 8. Для высоты и ширины максимальное значение установим 1000 мм, а для длины — 2000 мм, минимальные значения для всех размеров — 10 мм. Для твёрдости возьмем тот же диапазон значений, что и в случае с втулкой (15-700). Код для установления этих ограничений представлен в приложении Б.

Теперь установим логический контроль. Будем считать, что длина должна быть самым длинным размером. Обдирочные размеры не могут превосходить поковочные. Код для этих действий представлен в приложении Б.

3.3 Тестирование выходных данных для втулки

Любое программное обеспечение должно работать правильно и отлаженно, таким образом, чтобы пользователь не сомневался в точности полученных результатов.

Как видно из рисунка 42, наиболее продуктивными оказываются проекты, в которых до выпуска программного продукта было устранено примерно 95 % ошибок. Фирма IBM ещё 35 лет назад обнаружила, что в проектах, концентрирующих свои усилия на минимизации сроков, часто превышаются и сроки, и расходы, а там, где усилия фокусируются на минимизации количества дефектов, достигаются кратчайшие сроки и наивысшая производительность работ. Таким образом, качественное тестирование на стадии разработки позволит сократить общие затраты времени и усилий при реализации проекта.

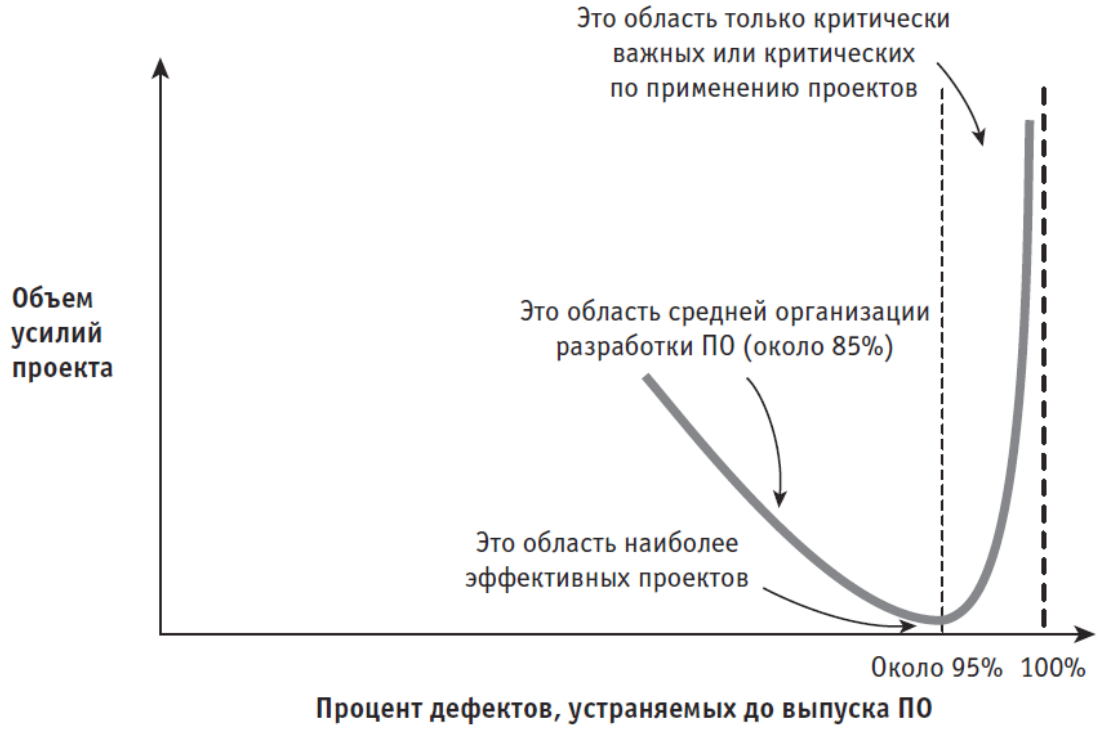

Рисунок 42 — Зависимость объема усилий от процента устраняемых дефектов

При создании технологической карты, выходные данные должны соответствовать исходным данным. То есть в технологической карте должны присутствовать выбранные операции, должны учитываться выбранные требования, а при вычислениях — получаться значения, соответствующие введенным параметрам и виду стали.

Так как число вариантов набора исходных данных очень велико, проверим правильность работы программы лишь на некоторой выборке.

Для набора исходных данных, представленных на рисунке 43 в технологической карте должно получиться следующее:

1. На чертеже должны быть подписаны размеры. Внутренний диаметр — 170-5, наружный диаметр — 350+5, длина — 410+5, а также 4 фаски 3x45°.

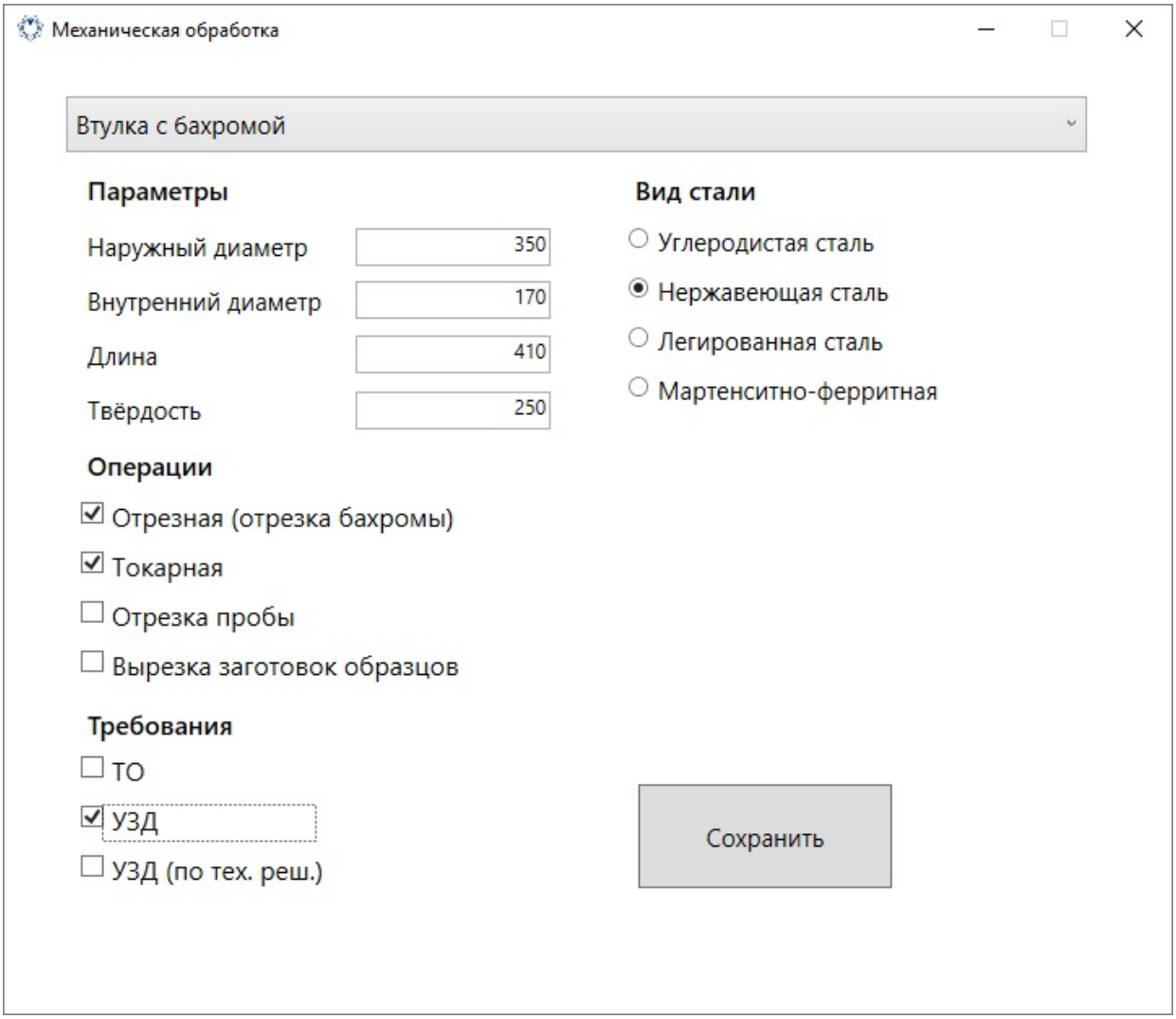

Рисунок 43 — Исходные данные для проверки выходных данных

2. Должны присутствовать 2 операции (отрезная и токарная под УЗД), с соответствующей им информацией. Кроме того, должна присутствовать контрольная операция, которая обязательна при наличии любых других.

3. Код пилы 4814 (так как наружный диаметр меньше 450, но больше 300). Норматив времени для отрезной операции, высчитываемая по формуле $\frac{D}{K*60}$, где D =350 и K = 2,5 (как видно из таблицы 1 при H=250), будет равна

$\frac{350}{2,5*60} = 2,(3)$ или при округлении до 3 значащих цифр 2,33. Так как бахрома отрезается с двух сторон, умножаем 2,33 на 2, получается 4,66.

4. Норматив времени для токарной операции будет равен сумме 6,0 (соответствует длине 450 и меньше и наружному диаметру 350 и меньше (см. таблицу 1)) + 5,0 (соответствует длине 450 и меньше и внутреннему диаметру 180 и меньше), умноженной на 2 (коэффициент для нержавеющей стали) или же 22.

5. Наконец должно иметься требование о необходимости подвергнуть деталь контролю ультразвуком.

На рисунке 44 представлен фрагмент технологической карты, созданной программой.

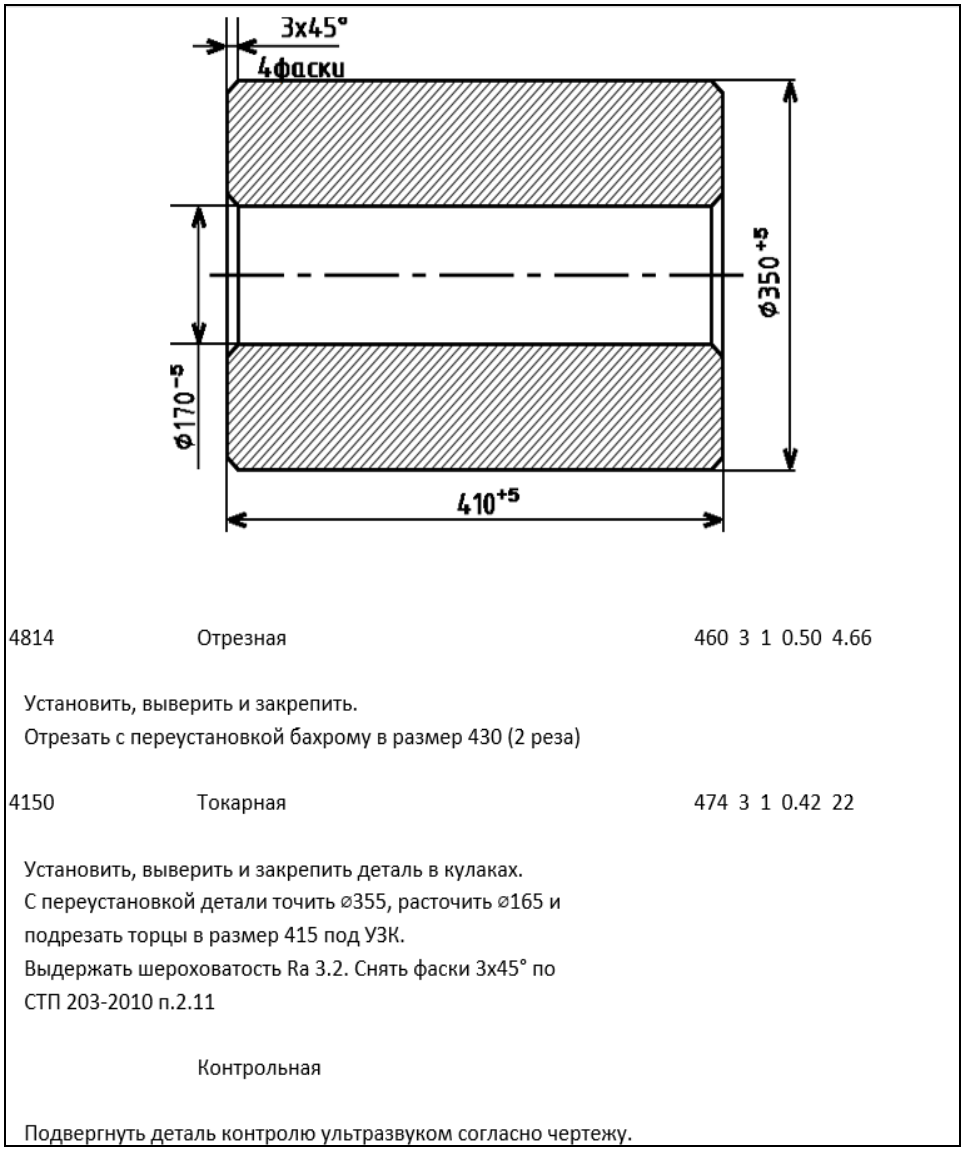

Рисунок 44 — Фрагмент технологической карты для исходных данных из рисунка 16

Сравним требуемые значения с представленными в программе:

1. Размеры соответствуют требованиям.
2. Операции соответствуют требованиям.
3. Код пилы и норматив отрезной операции соответствуют требованиям.
4. Норматив времени для токарной операции соответствует требованиям.
5. Требование о необходимости подвергнуть деталь контролю ультразвуком присутствует.

Таким образом, содержание фрагмента технологической карты, созданной программой, полностью соответствует приведенным выше требованиям.

3.4 Тестирование выходных данных для пластины

Для детали типа «Пластина» также посчитаем переменные величины, которые зависят от заданных пользователем исходных данных (операции, нормативы времени, код оборудования) без использования программы, а затем сравним со значениями, полученными с помощью программы.

Для набора исходных данных, представленных на рисунке 45 в технологической карте должно получиться следующее:

1. На чертеже должны быть подписаны размеры. Ширина — 460+5, высота — 260+5, длина — 810+5, а также 12 фасок 5x45°.
2. Должны присутствовать 2 операции (строгальная и отрезная), с соответствующей им информацией.
3. Вычислим норматив времени для установки. Масса будет равна 0,5*0,3*0,9*7800 = 1053 кг. По таблице 4 этой массе соответствует значение 0,475 или для 6 сторон 2,85.

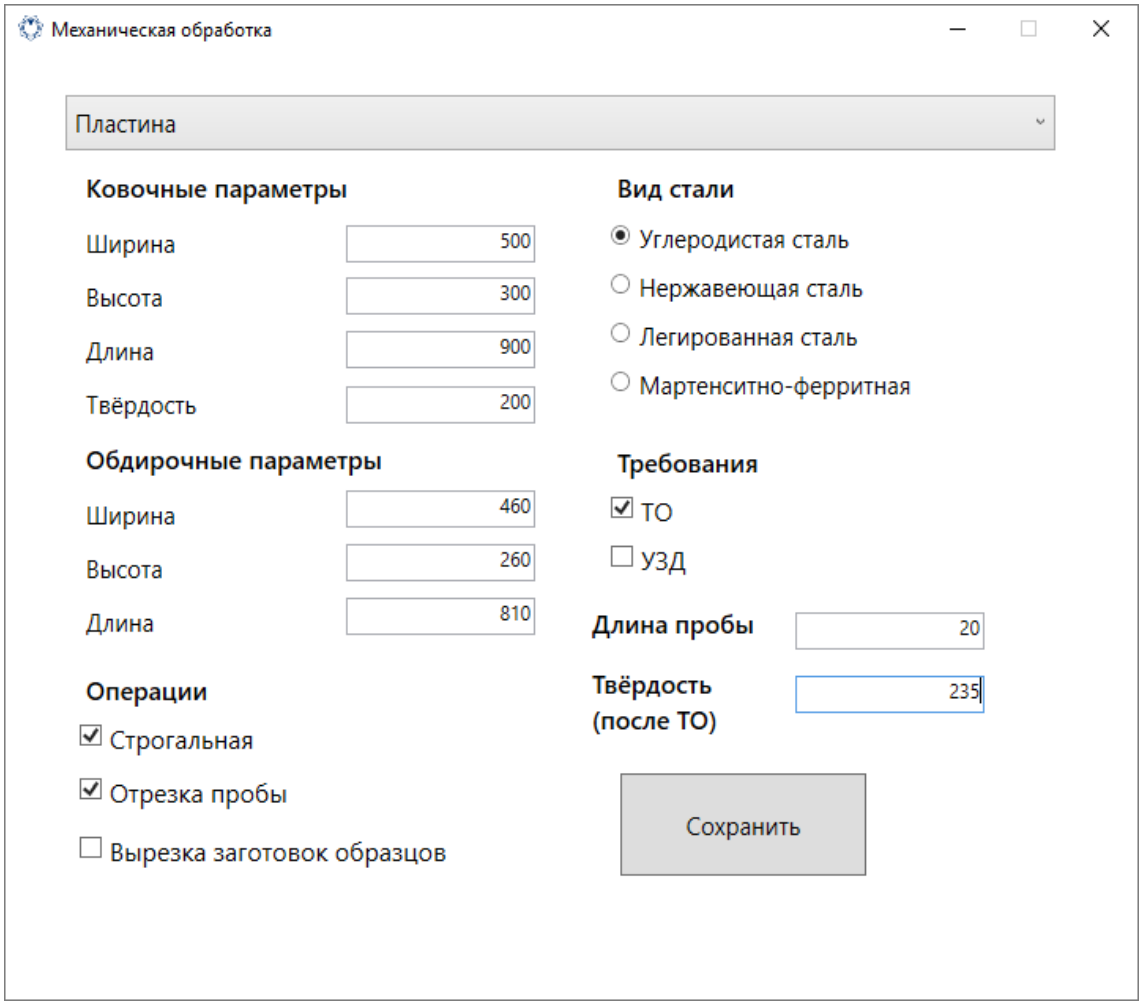

Рисунок 45 — Набор исходных значений для проверки выходных данных

4. Норматив времени для строгальной операции состоит из двух компонентов, норматива для снятия фасок и норматива для строгания 6 граней (плоскостей). В соответствии с таблицей 5, норматив для снятия фасок будет равен 0,023*4 + 0,023*4 + 0,025*4=0,284. Теперь вычислим норматив для строгания граней.

Число проходов для строгания по высоте = ⌈(300+7-260)/8⌉ = 6.

Норматив времени для строгания по высоте = число проходов * норматив времени на один проход = 6 * 1(норматив из таблицы 7 для размера 900x500) = 6.

Число проходов для строгания по ширине = ⌈(500+7-460)/8⌉ = 6.

Норматив времени для строгания по ширине = 6 * 0,52 (норматив из таблицы 6 для размера 900x260) = 3,12.

Число проходов для строгания по длине = ⌈(900+7-810)/5⌉ = 20.

Норматив времени для строгания по длине = 20 * 0,31 (норматив из таблицы 6 для размера 460x260) = 6,2.

Теперь сложим все нормативы времени и получим итоговый норматив для строгальной операции: 0,284+6+3,12+6,2 = 15,604.

5. Код пилы для ширины более 450 мм будет иметь значение 4816.

6. Норматив времени для отрезной операции определим по формуле $\frac{\sqrt{\frac{a^2+b^2}{2}}}{K*60}$, где a и b — обдирочные ширина и высота, а коэффициент K определяется по таблице 3: $\frac{\sqrt{\frac{460^2+260^2}{2}}}{3,5*60} = 1,779$.

На рисунке 46 представлен фрагмент технологической карты, сформированной программой. Содержимое фрагмента соответствует представленным выше требованиям. Значения нормативов равны значениям, полученным без использования программы.

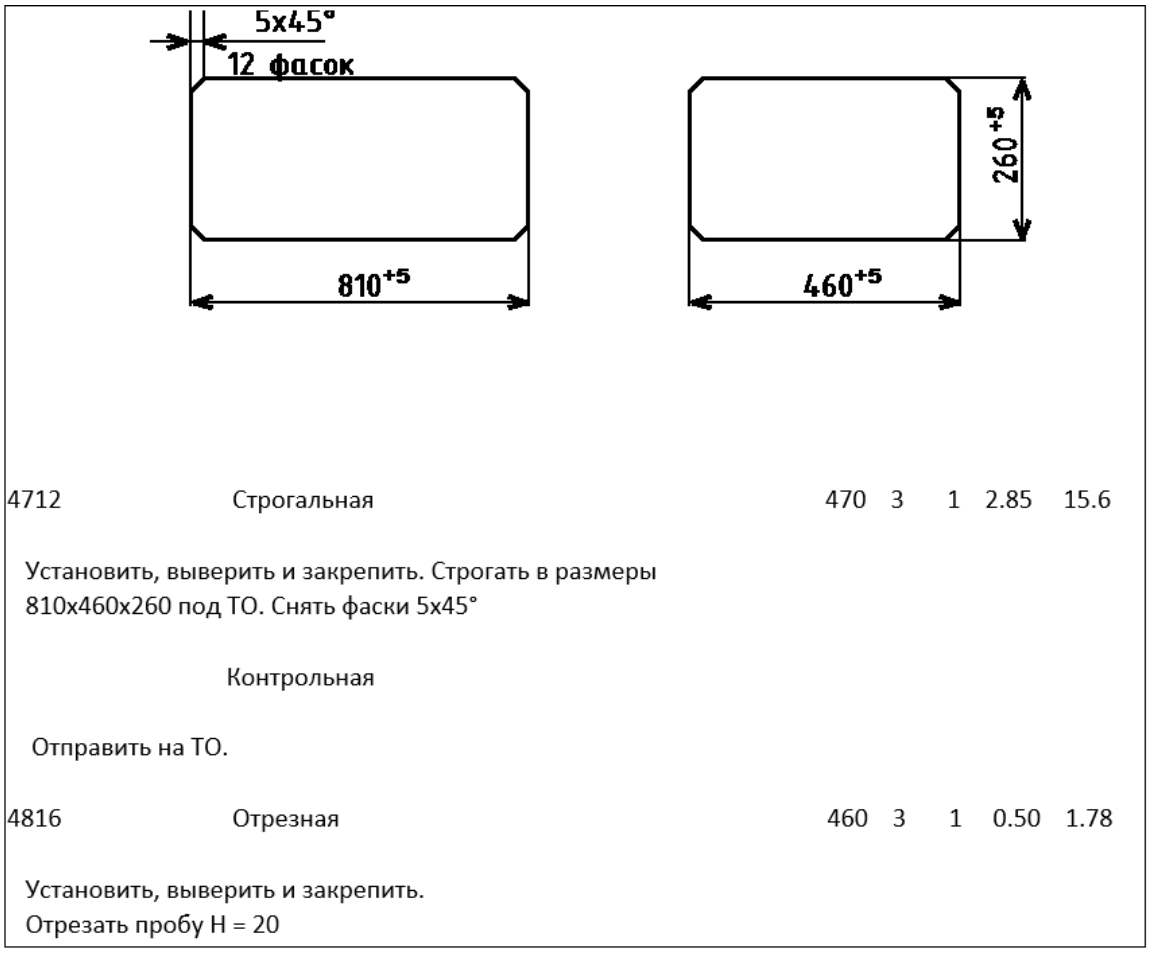

Рисунок 46 — Фрагмент технологической карты, соответствующий набору исходных данных из рисунка 45

3.5 Руководство пользователя

Приложение предназначено для автоматического формирования технологических карт для предварительной механической обработки (обдирки) поковок. Окно приложения после его запуска представлено на рисунке 47, а на рисунке 48 оно разделено на области, в которых элементы управления объединены общим смыслом.

Первым делом следует выбрать тип детали. При нажатии на кнопку, находящуюся в области 1, раскроется список с доступными типами деталей. Нажав затем на название определенного типа детали, он будет выбран и отобразится на этой кнопке.

В области 2 находятся текстовые поля для ввода информации с соответствующими им подписями. Эти текстовые поля предназначены для ввода размеров и твёрдости детали. Размеры вводятся в миллиметрах, значение твёрдости — по Бринеллю.

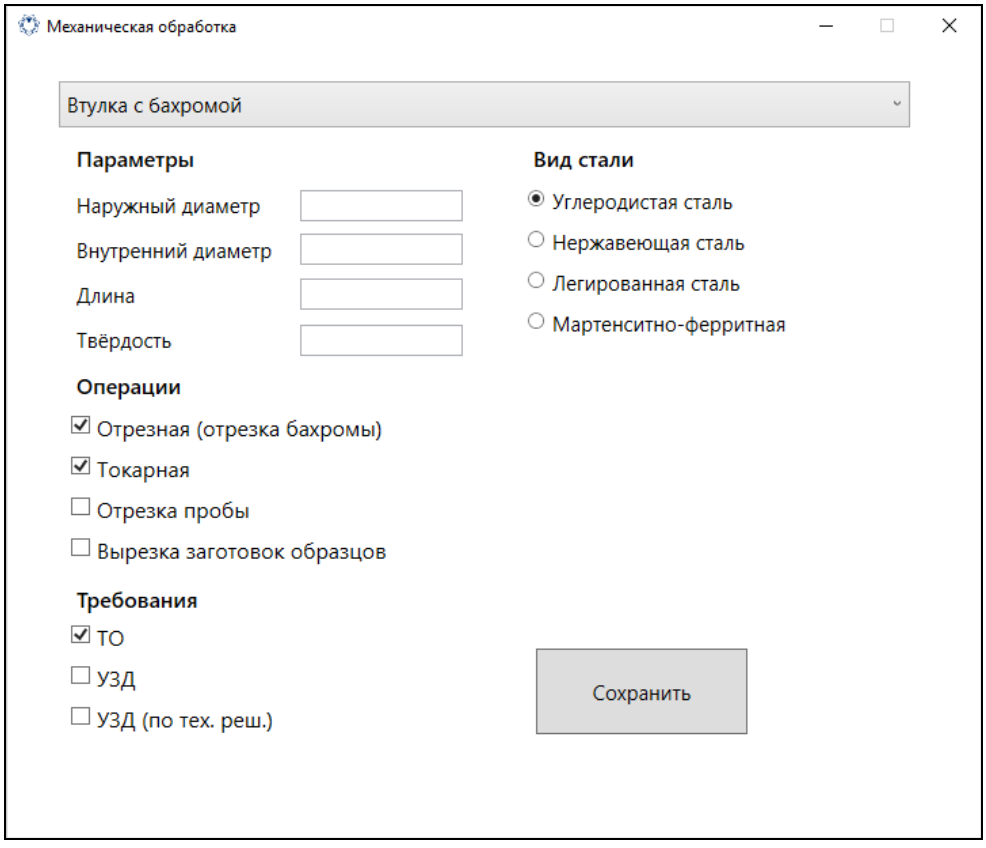

Рисунок 47 — Окно приложения после запуска

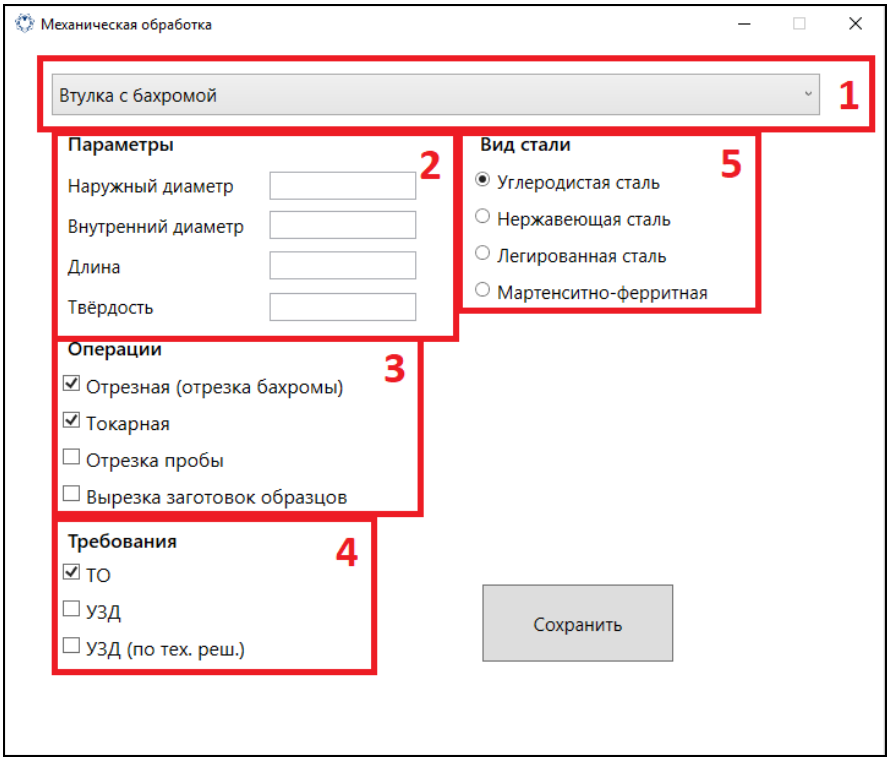

Рисунок 48 — Разделение окна приложения на области

В области 3 выбираются необходимые операции. При нажатии на название операции в окошечке ставится/снимается флажок. Наличие флажка перед названием операции соответствует присутствию данной операции.

Подобным же образом выбираются требования в области 4. Так как деталь откована из одного вида стали, то в области 5 можно выбрать только одно значение.

После того как все исходные данные заданы, нужно нажать на кнопку «Сохранить», после чего курсор мыши перейдет в режим ожидания, это означает, что программа производит необходимые вычисления. После завершения вычислений откроется диалоговое окно «Сохранить как», в котором нужно выбрать местоположение и название сформированной технологической карты.

3.6 Калькуляция проекта

В ходе выполнения проекта были изучены следующие технологии:

1. Разработка интерфейсов в Windows Presentation Foundation.

2. Работа с формами.

3. Работа с Microsoft Word с помощью C# (создание таблиц, добавление текстовой информации и её форматирование).

4. Подключение к проекту сторонних шрифтов.

5. Создание чертежей с помощью C#.

В итоге было создано расширяемое приложение, содержащее интерфейсы для двух деталей, между которыми легко можно переключаться, было написано около 3,5 тысяч строк кода, в которых реализовано создание шаблона технологической карты в Microsoft Word, создаются чертежи для двух деталей, подключается внешний шрифт GOST A, с помощью которого подписываются размеры на чертежах, определено 2 класса, с помощью которых производится вычисление различных нормативов времени, что, в том числе, потребовало перевода громоздких таблиц нормативов времени в понятную для языка программирования форму.

ЗАКЛЮЧЕНИЕ

Перед началом выполнения проекта ставилась задача создать приложение, автоматизирующее создание технологических карт таким образом, чтобы оно было максимально удобным для пользователя и при этом брало на себя настолько много работы, насколько возможно.

Для выполнения поставленной задачи понадобилось изучить несколько технологий.

Изучение технологии Windows Foundation Presentation позволило создать для приложения интерфейс более современный, чем это было бы возможно с Windows Forms, а также реализовать возможность переключения между интерфейсами для разных типов деталей, что сделало приложение легко расширяемым.

Изучение прикладного интерфейса Microsoft.Office.Interop.Word позволило реализовать возможность создания отформатированного шаблона технологической карты и его последующее заполнение, благодаря чему приложение выдаёт данные не в каком-то «сыром» виде, нуждающемся в дооформлении пользователем, а в совершенно готовом к печати.

Использование класса Graphics, а также подключение внешнего шрифта GOST A, позволило создать чертежи деталей и вставить их непосредственно в готовый шаблон, одновременно изменяя подписанные размеры на размеры, заданные пользователем. Это значит, что пользователь будет освобожден не только от создания чертежа, но даже от необходимости подписывать размеры.

Выбор табличных констант, а также проведение вычислений полностью возьмет на себя приложение и сделает это точно и быстро, что также станет несомненным плюсом для пользователя.

Наконец, в зависимости от выбранных операций и требований, программа выведет о них информацию.

Таким образом, созданное приложение способно значительно упростить, ускорить и сделать более точной разработку технологических карт.

При этом у приложения имеются и недостатки. Прежде всего, это довольно долговременный и трудозатратный процесс разработки, требующий от разработчика тесного взаимодействия со специалистами-технологами. Необходимо не только разобраться в последовательности действий, которые совершают технологи при создании технологических карт, но и учесть массу нюансов, что дополнительно усложняет разработку, так как многие из них обнаруживаются уже на стадии тестирования.

Другой недостаток приложения — это ограниченность сферы его применения. Оно способно работать лишь с деталями некоторой стандартной формы, стандартных размеров и требований, но будущее отрасли машиностроения и экономики в целом не в стандартизации, а в приспособлении к потребностям заказчиков, которые могут меняться непредсказуемым образом.

Итак, созданное приложение, при дальнейшем его расширении и подстраивании под новые требования, может стать незаменимым инструментом при разработке технологических карт.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Албахари Д. C# 6.0. Справочник. Полное описание языка профессионалов [Текст] / Д. Албахари, Б. Албахари. — 6-е издание. — Москва: Вильямс, 2017. — 1040 с.

2. ГОСТ2.307–68 Нанесение размеров и предельных отклонений [Электронный ресурс]. — Введ. 01.01.1971. — Режим доступа: https://graph.power.nstu.ru/wolchin/umm/eskd/eskd/GOST/2_307.htm (дата обращения: 23.05.18).

3. Гриффитс Й. Программирование на C# 5.0 [Текст] / Й. Гриффитс. — Москва: Эксмо, 2014. — 1136 с.

4. Гуриков С. Введение в программирование на языке Visual C# [Текст] / С. Гуриков. — Москва: Дрофа, 2013. — 448 с.

5. Зиборов В. Visual C# 2012 на примерах [Текст] / В. Зиборов. — Санкт-Петербург: БХВ-Петербург, 2013. — 480 с.

6. Ишкова Э. Самоучитель C#. Начала программирования [Текст] / Э. Ишкова. — Санкт-Петербург: Наука и техника, 2013. — 496 с.

7. Макдональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов [Текст] / М. Макдональд. — 4-е издание. — Москва: Вильямс, 2013. — 1024 с.

8. Нагел К. C# 5.0 и платформа .NET 4.5 для профессионалов [Текст] / К. Нагел, Б. Ивьен, Д. Глинн. — Москва: Вильямс, 2014. — 1440 с.

9. Официальный сайт САПР КОМПАС [Электронный ресурс]. — Режим доступа: https://kompas.ru/kompas-grafik/about/ (дата обращения: 23.05.18).

10. Петцольд Ч. Программирование для Microsoft Windows 8. [Текст] / — 6-е издание. — Санкт-Петербург: Питер, 2014. — 1008 с.

11. Рихтер Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# [Текст] / Д. Рихтер. — 4-е издание. — Санкт-Петербург: Питер, 2017. — 896 с.

12. Скит Д. C# для профессионалов. Тонкости программирования [Текст]: научно-популярное издание / Д. Скит. — 3-е издание. — Москва: Вильямс, 2017. — 608 с.

13. Стиллмен Э. Изучаем C# [Текст] / Э. Стиллмен, Д. Грин. — 3-е издание. — Санкт-Петербург: Питер, 2014. — 816 с.

14. Тидвелл Д. Разработка пользовательских интерфейсов [Текст] / Д. Тидвел. — 2-е издание. — Санкт-Петербург: Питер, 2011. — 480 с.

15. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5 [Текст] / Э. Троелсен. — 6-издание. — Москва: Вильямс, 2015. — 1312 с.

16. Управление производством. Пути повышения производительности труда [Электронный ресурс]. — Режим доступа: http://www.up-pro.ru/encyclopedia/povyshenie-proizvoditelnosti.html (дата обращения: 23.05.18).

17. Шарп Д. Microsoft Visual C#. Подробное руководство [Текст] / Д. Шарп. — 6-е издание. — Санкт-Петербург: Питер, 2017. — 848 с.

18. ADEM — Автоматизация проектно-конструкторской и технологической подготовки производства [Электронный ресурс]. — Режим доступа: http://adem.ru/products/ (дата обращения: 23.05.18).

19. Cansever A. WPF Frame Kullanımı [Электронный ресурс]. — Режим доступа: https://www.ahmetcansever.com/c-2/wpf-frame-kullanimi/ (дата обращения: 23.05.18).

20. Chaudhary H. Head First C# Programming [Text] / H. Chaudhary. — North Charleston: Createspace, 2014. — 250 p.

21. De Smet B. C# 5.0 Unleashed [Text] / B. De Smet. — London: Pearson Education, 2013. — 1700 p.

22. Doberenz W. Visual C# 2015 [Text] / W. Doberenz, T. Gewinnus. — München: Hanser, 2015. — 1202 p.

23. Mahesh C. Using XAML Frame in WPF [Электронный ресурс]. — Режим доступа: https://www.c-sharpcorner.com/UploadFile/mahesh/using-xaml-frame-in-wpf857/ (дата обращения: 23.05.18).

24. McGrath M. C# Programming in easy steps [Text] / M. McGrath. — In Easy Steps, 2016. — 192 p.

25. Microsoft Docs. Access Office Interop Objects by Using Visual C# Features [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/programmingguide/interop/how-to-access-office-onterop-objects (дата обращения: 23.05.18).

26. Microsoft Docs. Programmatically Create Word Tables [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/en-us/visualstudio /vsto/how-to-programmatically-create-word-tables#creating-tables-in-document-level-customizations objects (дата обращения: 23.05.18).

27. Nakov S. Fundamentals of Computer Programming with C# [Text] / S. Nakov, D. Dimitrov, H. Germanov. — introprogramming.info, 2013. — 1122 p.

28. Nathan A. WPF 4.5 Unleashed [Text] / A. Nathan. — London: Pearson Education, 2014. — 864 p.

29. Sempf B. C# 5.0 all-in-one for dummies [Text] / B. Sempf, C. Sphar, S. R. Davis. — Hoboken: John Wiley & Sons, 2013. — 843 p.

30. Software blog. Using custom fonts in C# application revisited [Электронный ресурс]. — Режим доступа: http://www.swsoftware.net/using-custom-fonts-c-application-revisited/ (дата обращения: 25.05.2018).

31. Stephens R. C# 5.0. Programmer’s Reference [Text] / R.Stephens. — Hoboken: John Wiley & Sons, 2014. — 962 p.

32. Strauss D. C# 7 and .NET Core Cookbook [Text]: textbook / D. Strauss. — Birmingham: Packt Publishing, 2017. — 628 p.

ПРИЛОЖЕНИЕ А

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования

Кафедра информационных систем и технологий

Направление подготовки 09.03.02 Информационные системы и технологии

Профиль подготовки «Информационные технологии в медиаиндустрии»

УТВЕРЖДАЮ
Заведующий кафедрой

_____ Н.С. Толстова
подпись и.о. фамилия

« ___ » _____ 2018 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студента (ки) *5* курса группы *ДЗИТм-511*
Шимова Сергея Александровича
фамилия, имя, отчество полностью

1. Тема *Приложение для автоматизации создания технологических карт предварительной механической обработки заготовок деталей*
утверждена распоряжением по институту от «_____» _____ 20 г. № ___

2. Руководитель *Черноскутов Михаил Юрьевич*
фамилия, имя, отчество полностью

.	*Старший преподаватель кафедры ИС*	*РГППУ*
ученая степень / ученое звание	должность	место работы

3. Место преддипломной практики *ФГАОУ ВО РГППУ, ИПО, кафедра информационных систем и технологий*
4. Исходные данные к ВКР *алгоритм разработки технологических карт, таблицы нормативов времени, требования к оформлению*

5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов)
интерфейс приложения, формирование шаблона технологической карты, создание чертежей, реализация алгоритма вычисления нормативов времени и других параметров,

формирование и вывод информации

6. Перечень демонстрационных материалов *презентация, выполненная в MS Power Point, обзорный видеоролик по приложению, приложение для автоматизации создания технологических карт*

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной квалификационной работе	23.04.2018	10%	
2	Выполнение работ по разрабатываемым вопросам и их изложение в пояснительной записке:	07.05.2018	60%	
2.1	Разработка интерфейса приложения	26.04.2018	10%	
2.2	Формирование шаблона технологической карты в Microsoft Word	29.04.2018	10%	
2.3	Разработка механизма автоматического создания чертежей	01.05.2018	10%	
2.4	Реализация алгоритма вычисления нормативов времени	04.05.2018	20%	
2.5	Создание механизма оформления и вывода информации	07.05.2018	10%	
3	Оформление текстовой части ВКР	15.05.2018	10%	
4	Выполнение демонстрационных материалов к ВКР	01.06.2018	10%	
5	Нормоконтроль	06.06.2018	5%	
6	Подготовка доклада к защите в ГЭК	13.06.2018	5%	

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата

Руководитель _____ _____ *Задание получил* _____

_____ _____ подпись дата

подпись студента дата

9. Дипломная работа и все материалы проанализированы.

Считаю возможным допустить *Шимова С.А.* к защите выпускной квалификационной работы в государственной экзаменационной комиссии.

Руководитель _____ _____
подпись дата

10. Допустить *Шимова С.А.* к защите выпускной квалификационной работы
фамилия и. о. студента

в государственной экзаменационной комиссии (протокол заседания кафедры от «_____» _____ 20_____г., № _____)

Заведующий кафедрой _____ _____
подпись дата

ПРИЛОЖЕНИЕ Б

Код, позволяющий скрыть параметры для операции «отрезка пробы»:

```
private void CheckBoxOtrezProbyChanged(object sender, RoutedEventArgs e)
{
if (checkOtrezProby.IsChecked.Value)
{
label7.Visibility = Visibility.Visible;
label9.Visibility = Visibility.Visible;
textDlinaProby.Visibility = Visibility.Visible;
textTverdPosleTO.Visibility = Visibility.Visible;
}
else
{
label7.Visibility = Visibility.Hidden;
label9.Visibility = Visibility.Hidden;
textDlinaProby.Visibility = Visibility.Hidden;
textTverdPosleTO.Visibility = Visibility.Hidden;
}
}
```

Код для создания шаблона технологической карты:

```
Microsoft.Office.Interop.Word.Table firstTable =
document.Tables.Add(para1.Range, 13, 20);
firstTable.Borders.Enable = 1;
for (int i = 1; i <= 10; i++)
{
for (int c = 1; c <= 13; c++)
{
if (c == 8) continue;
firstTable.Rows[c].Cells[i].Merge(firstTable.Rows[c].Cells[i + 1]);
}
}
firstTable.Rows[1].Cells[1].Merge(firstTable.Rows[1].Cells[2]);
firstTable.Rows[1].Cells[2].Merge(firstTable.Rows[1].Cells[3]);
firstTable.Rows[1].Cells[2].Merge(firstTable.Rows[1].Cells[4]);
firstTable.Rows[1].Cells[2].Merge(firstTable.Rows[1].Cells[5]);
firstTable.Rows[1].Cells[2].Merge(firstTable.Rows[1].Cells[3]);
for (int i = 2; i <= 7; i++)
```

```
{
firstTable.Rows[i].Cells[1].Merge(firstTable.Rows[i].Cells[2]);
firstTable.Rows[i].Cells[2].Merge(firstTable.Rows[i].Cells[3]);
firstTable.Rows[i].Cells[2].Merge(firstTable.Rows[i].Cells[3]);
firstTable.Rows[i].Cells[3].Merge(firstTable.Rows[i].Cells[4]);
firstTable.Rows[i].Cells[4].Merge(firstTable.Rows[i].Cells[5]);
firstTable.Rows[i].Cells[4].Merge(firstTable.Rows[i].Cells[5]);
}
for (int i = 1; i <= 10; i++)
{
firstTable.Rows[8].Cells[4].Merge(firstTable.Rows[8].Cells[5]);
}
firstTable.Rows[8].Cells[9].Merge(firstTable.Rows[8].Cells[10]);
for (int i = 1; i <= 9; i++)
{
firstTable.Rows[9].Cells[1].Merge(firstTable.Rows[9].Cells[2]);
}
for (int i = 10; i <= 13; i++)
{
firstTable.Rows[i].Cells[1].Merge(firstTable.Rows[i].Cells[2]);
firstTable.Rows[i].Cells[2].Merge(firstTable.Rows[i].Cells[3]);
firstTable.Rows[i].Cells[3].Merge(firstTable.Rows[i].Cells[4]);
firstTable.Rows[i].Cells[3].Merge(firstTable.Rows[i].Cells[4]);
firstTable.Rows[i].Cells[4].Merge(firstTable.Rows[i].Cells[5]);
firstTable.Rows[i].Cells[4].Merge(firstTable.Rows[i].Cells[5]);
}
firstTable.Rows[1].Cells[1].Height = 20;
firstTable.Rows[1].Cells[1].Range.Font.Size = 12;
firstTable.Rows[1].Cells[1].VerticalAlignment =
WdCellVerticalAlignment.wdCellAlignVerticalCenter;
firstTable.Rows[1].Cells[1].Range.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
firstTable.Rows[1].Cells[1].Range.Text = "Т  Б  М  О";
firstTable.Rows[1].Cells[2].Range.Font.Size = 12;
firstTable.Rows[1].Cells[2].VerticalAlignment =
WdCellVerticalAlignment.wdCellAlignVerticalCenter;
firstTable.Rows[1].Cells[2].Range.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
firstTable.Rows[1].Cells[2].Range.Text = "Технологическая карта";
firstTable.Rows[2].Cells[1].Range.Text = "Наименование";
firstTable.Rows[3].Cells[1].Range.Text = "№ заказа";
firstTable.Rows[4].Cells[1].Range.Text = "Шифр заказчика";
firstTable.Rows[5].Cells[1].Range.Text = "Опросный лист";
firstTable.Rows[6].Cells[1].Range.Font.Size = 10;
```

```
firstTable.Rows[6].Cells[1].Range.Text = "Кол.дет. в пок./масса";
firstTable.Rows[7].Cells[1].Range.Text = "Размеры поковки";
firstTable.Rows[8].Cells[1].Range.Text = "№";
firstTable.Rows[8].Cells[2].Range.Text = "УЧ";
firstTable.Rows[8].Cells[3].FitText = true;
firstTable.Rows[8].Cells[3].Range.Font.Bold = 1;
firstTable.Rows[8].Cells[3].Range.Text = "ОПЕР";
firstTable.Rows[8].Cells[4].Range.Text = "КОД, НАИМЕНОВАНИЕ ОПЕРАЦИИ,
ОБОРУДОВАНИЯ";
firstTable.Rows[8].Cells[5].FitText = true;
firstTable.Rows[8].Cells[5].Range.Font.Bold = 1;
firstTable.Rows[8].Cells[5].Range.Text = "ПРОФ";
firstTable.Rows[8].Cells[6].Range.Text = "Р";
firstTable.Rows[8].Cells[7].Range.Text = "ЕН";
firstTable.Rows[8].Cells[8].FitText = true;
firstTable.Rows[8].Cells[8].Range.Font.Bold = 1;
firstTable.Rows[8].Cells[8].Range.Text = "Тпз";
firstTable.Rows[8].Cells[9].Range.Text = "Тшт";
firstTable.Rows[8].Cells[9].VerticalAlignment =
WdCellVerticalAlignment.wdCellAlignVerticalCenter;
firstTable.Rows[8].Cells[9].Range.ParagraphFormat.Alignment =
WdParagraphAlignment.wdAlignParagraphCenter;
firstTable.Rows[2].Cells[3].Range.Text = "Обозначение";
firstTable.Rows[3].Cells[3].Range.Text = "№ чертежа";
firstTable.Rows[4].Cells[3].Range.Text = "Марка";
firstTable.Rows[5].Cells[3].Range.Text = "Отходы";
firstTable.Rows[6].Cells[3].Range.Text = "Стандарт";
firstTable.Rows[7].Cells[3].Range.Text = "Группа";
firstTable.Rows[10].Cells[1].Range.Text = "Технолог";
firstTable.Rows[11].Cells[1].Range.Text = "Нач. ТБМО";
firstTable.Rows[12].Cells[1].Range.Text = "Нормировщик";
firstTable.Rows[13].Cells[1].Range.Text = "Утвердил";
firstTable.Rows[9].Cells[1].Range.Font.Size = 12;
```

Код для создания чертежа втулки:

```
Bitmap b = new Bitmap(560, 480);
System.Drawing.Graphics graphicsObj =
System.Drawing.Graphics.FromImage((System.Drawing.Image)b);
if (checkBoxTokar1.IsChecked == true)
{
System.Drawing.Pen myPen = new
System.Drawing.Pen(System.Drawing.Color.Black, 3);
System.Drawing.Point pnt1 = new System.Drawing.Point(50, 60);
```

```
System.Drawing.Point pnt12 = new System.Drawing.Point(60, 50);
System.Drawing.Point pnt2 = new System.Drawing.Point(480, 50);
System.Drawing.Point pnt23 = new System.Drawing.Point(490, 60);
System.Drawing.Point pnt3 = new System.Drawing.Point(490, 140);
System.Drawing.Point pnt34 = new System.Drawing.Point(480, 150);
System.Drawing.Point pnt4 = new System.Drawing.Point(60, 150);
System.Drawing.Point pnt41 = new System.Drawing.Point(50, 140);
System.Drawing.Point[] polygonPoints = { pnt1, pnt12, pnt2, pnt23, pnt3,
pnt34, pnt4, pnt41 };
graphicsObj.DrawPolygon(myPen, polygonPoints);
System.Drawing.Drawing2D.HatchBrush htchBrush = new
System.Drawing.Drawing2D.HatchBrush(System.Drawing.Drawing2D.HatchStyle.B
ackwardDiagonal, System.Drawing.Color.Black, System.Drawing.Color.White);
graphicsObj.FillPolygon(htchBrush, polygonPoints);
System.Drawing.Point pnt5 = new System.Drawing.Point(50, 270);
System.Drawing.Point pnt56 = new System.Drawing.Point(60, 260);
System.Drawing.Point pnt6 = new System.Drawing.Point(480, 260);
System.Drawing.Point pnt67 = new System.Drawing.Point(490, 270);
System.Drawing.Point pnt7 = new System.Drawing.Point(490, 350);
System.Drawing.Point pnt8 = new System.Drawing.Point(60, 360);
System.Drawing.Point pnt85 = new System.Drawing.Point(50, 350);
System.Drawing.Point[] polygonPoints2 = { pnt5, pnt56, pnt6, pnt67, pnt7,
pnt78, pnt8, pnt85 };
graphicsObj.DrawPolygon(myPen, polygonPoints2);
graphicsObj.FillPolygon(htchBrush, polygonPoints2);
graphicsObj.DrawLine(myPen, pnt85, pnt1);
graphicsObj.DrawLine(myPen, pnt7, pnt23);
graphicsObj.DrawLine(myPen, pnt4, pnt56);
graphicsObj.DrawLine(myPen, pnt34, pnt6);
graphicsObj.DrawLine(myPen, pnt1, pnt12);
graphicsObj.DrawLine(myPen, pnt12, pnt2);
graphicsObj.DrawLine(myPen, pnt2, pnt23);
graphicsObj.DrawLine(myPen, pnt3, pnt34);
graphicsObj.DrawLine(myPen, pnt34, pnt4);
graphicsObj.DrawLine(myPen, pnt4, pnt41);
graphicsObj.DrawLine(myPen, pnt5, pnt56);
graphicsObj.DrawLine(myPen, pnt56, pnt6);
graphicsObj.DrawLine(myPen, pnt6, pnt67);
graphicsObj.DrawLine(myPen, pnt7, pnt78);
graphicsObj.DrawLine(myPen, pnt78, pnt8);
graphicsObj.DrawLine(myPen, pnt8, pnt85);
float[] dashValues = { 22, 4, 4, 4 };
System.Drawing.Pen dashPen = new
System.Drawing.Pen(System.Drawing.Color.Black, 3);
```

```
dashPen.DashPattern = dashValues;
graphicsObj.DrawLine(dashPen, new System.Drawing.Point(35, 205), new
System.Drawing.Point(510, 205));
System.Drawing.Pen sizePen = new
System.Drawing.Pen(System.Drawing.Color.Black, 2);
System.Drawing.Point point17 = new System.Drawing.Point(50, 0);
System.Drawing.Point point18 = new System.Drawing.Point(50, 410);
graphicsObj.DrawLine(sizePen, point17, point18);
System.Drawing.Point point19 = new System.Drawing.Point(491, 350);
System.Drawing.Point point20 = new System.Drawing.Point(491, 410);
graphicsObj.DrawLine(sizePen, point19, point20);
System.Drawing.Point point21 = new System.Drawing.Point(50, 400);
System.Drawing.Point point22 = new System.Drawing.Point(490, 400);
graphicsObj.DrawLine(sizePen, point21, point22);
System.Drawing.Pen arrowPen = new
System.Drawing.Pen(System.Drawing.Color.Black, 4);
System.Drawing.Point point23 = new System.Drawing.Point(52, 400);
System.Drawing.Point point24 = new System.Drawing.Point(67, 395);
graphicsObj.DrawLine(arrowPen, point23, point24);
System.Drawing.Point point25 = new System.Drawing.Point(52, 400);
System.Drawing.Point point26 = new System.Drawing.Point(67, 405);
graphicsObj.DrawLine(arrowPen, point25, point26);
System.Drawing.Point point27 = new System.Drawing.Point(488, 400);
System.Drawing.Point point28 = new System.Drawing.Point(473, 395);
graphicsObj.DrawLine(arrowPen, point27, point28);
System.Drawing.Point point29 = new System.Drawing.Point(488, 400);
System.Drawing.Point point30 = new System.Drawing.Point(473, 405);
graphicsObj.DrawLine(arrowPen, point29, point30);
System.Drawing.Point point31 = new System.Drawing.Point(60, 150);
System.Drawing.Point point32 = new System.Drawing.Point(0, 150);
graphicsObj.DrawLine(sizePen, point31, point32);
System.Drawing.Point point33 = new System.Drawing.Point(60, 260);
System.Drawing.Point point34 = new System.Drawing.Point(0, 260);
graphicsObj.DrawLine(sizePen, point33, point34);
System.Drawing.Point point35 = new System.Drawing.Point(25, 150);
System.Drawing.Point point36 = new System.Drawing.Point(25, 360);
graphicsObj.DrawLine(sizePen, point35, point36);
System.Drawing.Point point37 = new System.Drawing.Point(25, 152);
System.Drawing.Point point38 = new System.Drawing.Point(30, 167);
graphicsObj.DrawLine(arrowPen, point37, point38);
System.Drawing.Point point39 = new System.Drawing.Point(25, 152);
System.Drawing.Point point40 = new System.Drawing.Point(20, 167);
graphicsObj.DrawLine(arrowPen, point39, point40);
System.Drawing.Point point41 = new System.Drawing.Point(25, 258);
```

```
System.Drawing.Point point42 = new System.Drawing.Point(30, 243);
graphicsObj.DrawLine(arrowPen, point41, point42);
System.Drawing.Point point43 = new System.Drawing.Point(25, 258);
System.Drawing.Point point44 = new System.Drawing.Point(20, 243);
graphicsObj.DrawLine(arrowPen, point43, point44);
System.Drawing.Point point45 = new System.Drawing.Point(480, 50);
System.Drawing.Point point46 = new System.Drawing.Point(555, 50);
graphicsObj.DrawLine(sizePen, point45, point46);
System.Drawing.Point point47 = new System.Drawing.Point(480, 361);
System.Drawing.Point point48 = new System.Drawing.Point(555, 361);
graphicsObj.DrawLine(sizePen, point47, point48);
System.Drawing.Point point49 = new System.Drawing.Point(550, 50);
System.Drawing.Point point50 = new System.Drawing.Point(550, 360);
graphicsObj.DrawLine(sizePen, point49, point50);
System.Drawing.Point point51 = new System.Drawing.Point(550, 52);
System.Drawing.Point point52 = new System.Drawing.Point(555, 67);
graphicsObj.DrawLine(arrowPen, point51, point52);
System.Drawing.Point point53 = new System.Drawing.Point(550, 52);
System.Drawing.Point point54 = new System.Drawing.Point(545, 67);
graphicsObj.DrawLine(arrowPen, point53, point54);
System.Drawing.Point point55 = new System.Drawing.Point(550, 358);
System.Drawing.Point point56 = new System.Drawing.Point(555, 343);
graphicsObj.DrawLine(arrowPen, point55, point56);
System.Drawing.Point point57 = new System.Drawing.Point(550, 358);
System.Drawing.Point point58 = new System.Drawing.Point(545, 343);
graphicsObj.DrawLine(arrowPen, point57, point58);
System.Drawing.Point point59 = new System.Drawing.Point(60, 0);
System.Drawing.Point point60 = new System.Drawing.Point(60, 50);
graphicsObj.DrawLine(sizePen, point59, point60);
System.Drawing.Point point61 = new System.Drawing.Point(20, 23);
System.Drawing.Point point62 = new System.Drawing.Point(150, 23);
graphicsObj.DrawLine(sizePen, point61, point62);
System.Drawing.Point point63 = new System.Drawing.Point(47, 23);
System.Drawing.Point point64 = new System.Drawing.Point(32, 18);
graphicsObj.DrawLine(arrowPen, point63, point64);
System.Drawing.Point point65 = new System.Drawing.Point(47, 23);
System.Drawing.Point point66 = new System.Drawing.Point(32, 28);
graphicsObj.DrawLine(arrowPen, point65, point66);
System.Drawing.Point point67 = new System.Drawing.Point(61, 23);
System.Drawing.Point point68 = new System.Drawing.Point(76, 28);
graphicsObj.DrawLine(arrowPen, point67, point68);
System.Drawing.Point point69 = new System.Drawing.Point(61, 23);
System.Drawing.Point point70 = new System.Drawing.Point(76, 18);
```

```
graphicsObj.DrawLine(arrowPen, point69, point70);
}
```

Код для добавления размеров на чертеже втулки:

```
String drawString = "";
if(checkBoxTO.IsChecked==true) drawString="  5x45°\n4фаски";
else drawString = "  3x45°\n4фаски";
String lengthString = "";
if (checkBoxTokar1.IsChecked == true)
lengthString=(vtulka.get_dlina()).ToString() + "\x207A\x2075";
else lengthString = (vtulka.get_dlina()+20).ToString() + "\x207A\x2075";
LoadFont(umz_wpf.Properties.Resources.MyFont);
System.Drawing.Font drawFont = new
System.Drawing.Font(private_fonts.Families[0], 16);
SolidBrush drawBrush = new SolidBrush(System.Drawing.Color.Black);
PointF drawPoint = new PointF(73, -10);
PointF drawPointLength = new PointF(250, 370);

if (checkBoxTokar1.IsChecked == true) graphicsObj.DrawString(drawString,
drawFont, drawBrush, drawPoint);
graphicsObj.DrawString(lengthString, drawFont, drawBrush,
drawPointLength);
String vnutrRazmer = "ø" + (vtulka.get_vnutrennij_diametr()).ToString() +
"\x207B\x2075";
String narDiametr = "ø" + (vtulka.get_naruzhnij_diametr()).ToString() +
"\x207A\x2075";
PointF drawPoint1 = new PointF(130, -565);
PointF drawPoint2 = new PointF(235, -46);
graphicsObj.TranslateTransform(b.Width, b.Height);
graphicsObj.RotateTransform(-90);
graphicsObj.DrawString(vnutrRazmer, drawFont, drawBrush, drawPoint1);
graphicsObj.DrawString(narDiametr, drawFont, drawBrush, drawPoint2);
```

Код для создания чертежа пластины:

```
Bitmap b = new Bitmap(674, 350);
System.Drawing.Graphics graphicsObj =
Sytem.Drawing.Graphics.FromImage((System.Drawing.Image)b);
System.Drawing.Pen myPen = new
Sytem.Drawing.Pen(System.Drawing.Color.Black, 3);
System.Drawing.Point pnt1 = new System.Drawing.Point(50, 60);
System.Drawing.Point pnt12 = new System.Drawing.Point(60, 50);
System.Drawing.Point pnt2 = new System.Drawing.Point(290, 50);
```

```
System.Drawing.Point pnt23 = new System.Drawing.Point(300, 60);
System.Drawing.Point pnt3 = new System.Drawing.Point(300, 160);
System.Drawing.Point pnt34 = new System.Drawing.Point(290, 170);
System.Drawing.Point pnt4 = new System.Drawing.Point(60, 170);
System.Drawing.Point pnt41 = new System.Drawing.Point(50, 160);
System.Drawing.Point[] polygonPoints = { pnt1, pnt12, pnt2, pnt23,
pnt3, pnt34, pnt4, pnt41 };
graphicsObj.DrawPolygon(myPen, polygonPoints);
System.Drawing.Point pnt5 = new System.Drawing.Point(420, 60);
System.Drawing.Point pnt56 = new System.Drawing.Point(430, 50);
System.Drawing.Point pnt6 = new System.Drawing.Point(610, 50);
System.Drawing.Point pnt67 = new System.Drawing.Point(620, 60);
System.Drawing.Point pnt7 = new System.Drawing.Point(620, 160);
System.Drawing.Point pnt78 = new System.Drawing.Point(610, 170);
System.Drawing.Point pnt8 = new System.Drawing.Point(430, 170);
System.Drawing.Point pnt85 = new System.Drawing.Point(420, 160);
System.Drawing.Point[] polygonPoints2 = { pnt5, pnt56, pnt6, pnt67, pnt7,
pnt78, pnt8, pnt85 };
graphicsObj.DrawPolygon(myPen, polygonPoints2);
System.Drawing.Pen sizePen = new
Sytem.Drawing.Pen(System.Drawing.Color.Black, 2);
System.Drawing.Point point1 = new System.Drawing.Point(50, 162);
System.Drawing.Point point2 = new System.Drawing.Point(50, 220);
graphicsObj.DrawLine(sizePen, point1, point2);
System.Drawing.Point point3 = new System.Drawing.Point(301, 162);
System.Drawing.Point point4 = new System.Drawing.Point(301, 220);
graphicsObj.DrawLine(sizePen, point3, point4);
System.Drawing.Point point5 = new System.Drawing.Point(420, 162);
System.Drawing.Point point6 = new System.Drawing.Point(420, 220);
graphicsObj.DrawLine(sizePen, point5, point6);
System.Drawing.Point point7 = new System.Drawing.Point(621, 162);
System.Drawing.Point point8 = new System.Drawing.Point(621, 220);
graphicsObj.DrawLine(sizePen, point7, point8);
System.Drawing.Point point9 = new System.Drawing.Point(612, 50);
System.Drawing.Point point10 = new System.Drawing.Point(670, 50);
graphicsObj.DrawLine(sizePen, point9, point10);
System.Drawing.Point point11 = new System.Drawing.Point(612, 171);
System.Drawing.Point point12 = new System.Drawing.Point(670, 171);
graphicsObj.DrawLine(sizePen, point11, point12);
System.Drawing.Point point13 = new System.Drawing.Point(50, 217);
System.Drawing.Point point14 = new System.Drawing.Point(301, 217);
graphicsObj.DrawLine(sizePen, point13, point14);
System.Drawing.Point point15 = new System.Drawing.Point(420, 217);
System.Drawing.Point point16 = new System.Drawing.Point(621, 217);
```

```
graphicsObj.DrawLine(sizePen, point15, point16);
System.Drawing.Point point17 = new System.Drawing.Point(667, 50);
System.Drawing.Point point18 = new System.Drawing.Point(667, 171);
graphicsObj.DrawLine(sizePen, point17, point18);
System.Drawing.Pen arrowPen = new
Sytem.Drawing.Pen(System.Drawing.Color.Black, 4);
System.Drawing.Point point23 = new System.Drawing.Point(52, 217);
System.Drawing.Point point24 = new System.Drawing.Point(66, 221);
graphicsObj.DrawLine(arrowPen, point23, point24);
System.Drawing.Point point25 = new System.Drawing.Point(52, 217);
System.Drawing.Point point26 = new System.Drawing.Point(66, 212);
graphicsObj.DrawLine(arrowPen, point25, point26);
System.Drawing.Point point27 = new System.Drawing.Point(299, 217);
System.Drawing.Point point28 = new System.Drawing.Point(285, 221);
graphicsObj.DrawLine(arrowPen, point27, point28);
System.Drawing.Point point29 = new System.Drawing.Point(299, 217);
System.Drawing.Point point30 = new System.Drawing.Point(285, 212);
graphicsObj.DrawLine(arrowPen, point29, point30);
System.Drawing.Point point31 = new System.Drawing.Point(422, 217);
System.Drawing.Point point32 = new System.Drawing.Point(436, 221);
graphicsObj.DrawLine(arrowPen, point31, point32);
System.Drawing.Point point33 = new System.Drawing.Point(422, 217);
System.Drawing.Point point34 = new System.Drawing.Point(436, 212);
graphicsObj.DrawLine(arrowPen, point33, point34);
System.Drawing.Point point35 = new System.Drawing.Point(619, 217);
System.Drawing.Point point36 = new System.Drawing.Point(605, 221);
graphicsObj.DrawLine(arrowPen, point35, point36);
System.Drawing.Point point37 = new System.Drawing.Point(619, 217);
System.Drawing.Point point38 = new System.Drawing.Point(605, 212);
graphicsObj.DrawLine(arrowPen, point37, point38);
System.Drawing.Point point51 = new System.Drawing.Point(667, 52);
System.Drawing.Point point52 = new System.Drawing.Point(661, 66);
graphicsObj.DrawLine(arrowPen, point51, point52);
System.Drawing.Point point53 = new System.Drawing.Point(667, 52);
System.Drawing.Point point54 = new System.Drawing.Point(672, 66);
graphicsObj.DrawLine(arrowPen, point53, point54);
System.Drawing.Point point55 = new System.Drawing.Point(667, 169);
System.Drawing.Point point56 = new System.Drawing.Point(661, 155);
graphicsObj.DrawLine(arrowPen, point55, point56);
System.Drawing.Point point57 = new System.Drawing.Point(667, 169);
System.Drawing.Point point58 = new System.Drawing.Point(672, 155);
graphicsObj.DrawLine(arrowPen, point57, point58);
System.Drawing.Point point59 = new System.Drawing.Point(60, 0);
System.Drawing.Point point60 = new System.Drawing.Point(60, 50);
```

```
graphicsObj.DrawLine(sizePen, point59, point60);
System.Drawing.Point point591 = new System.Drawing.Point(50, 0);
System.Drawing.Point point601 = new System.Drawing.Point(50, 60);
graphicsObj.DrawLine(sizePen, point591, point601);
System.Drawing.Point point61 = new System.Drawing.Point(20, 23);
System.Drawing.Point point62 = new System.Drawing.Point(150, 23);
graphicsObj.DrawLine(sizePen, point61, point62);
System.Drawing.Point point63 = new System.Drawing.Point(47, 23);
System.Drawing.Point point64 = new System.Drawing.Point(32, 18);
graphicsObj.DrawLine(arrowPen, point63, point64);
System.Drawing.Point point65 = new System.Drawing.Point(47, 23);
System.Drawing.Point point66 = new System.Drawing.Point(32, 28);
graphicsObj.DrawLine(arrowPen, point65, point66);
System.Drawing.Point point67 = new System.Drawing.Point(61, 23);
System.Drawing.Point point68 = new System.Drawing.Point(76, 28);
graphicsObj.DrawLine(arrowPen, point67, point68);
System.Drawing.Point point69 = new System.Drawing.Point(61, 23);
System.Drawing.Point point70 = new System.Drawing.Point(76, 18);
graphicsObj.DrawLine(arrowPen, point69, point70);
LoadFont(umz_wpf.Properties.Resources.MyFont);
System.Drawing.Font drawFont = new
Sytem.Drawing.Font(private_fonts.Families[0], 16);
System.Drawing.SolidBrush drawBrush = new
System.Drawing.SolidBrush(System.Drawing.Color.Black);
String drawString = "";
if (checkBoxTO.IsChecked.Value) drawString = "  5x45°\n12 фасок";
else if (checkBoxUZD.IsChecked.Value) drawString = "  3x45°\n12 фасок";
PointF drawPoint = new PointF(73, -10);
graphicsObj.DrawString(drawString, drawFont, drawBrush, drawPoint);
String lengthString = plastina.LengthObd.ToString()+"\x207A\x2075";
System.Drawing.PointF drawPoint2 = new System.Drawing.PointF(155, 187);
graphicsObj.DrawString(lengthString, drawFont, drawBrush, drawPoint2);
String widthString = plastina.WidthObd.ToString()+"\x207A\x2075";
System.Drawing.PointF drawPoint3 = new System.Drawing.PointF(500, 187);
graphicsObj.DrawString(widthString, drawFont, drawBrush, drawPoint3);
String heightString = plastina.HeightObd.ToString()+"\x207A\x2075";
System.Drawing.PointF drawPoint1 = new System.Drawing.PointF(217, -37);
graphicsObj.TranslateTransform(b.Width, b.Height);
graphicsObj.RotateTransform(-90);
graphicsObj.DrawString(heightString, drawFont, drawBrush, drawPoint1);
b.Save(System.IO.Path.GetTempPath() + "image.bmp");
```

Методы для получения и изменения членов класса Vtulka_s_bachromoj:

```
public void set_naruzhnij_diametr(int naruzhnij_diametr)
{
this.naruzhnij_diametr = naruzhnij_diametr;
}
public int get_naruzhnij_diametr()
{
return this.naruzhnij_diametr;
}
public void set_vnutrennij_diametr(int vnutrennij_diametr)
{
this.vnutrennij_diametr = vnutrennij_diametr;
}
public int get_vnutrennij_diametr()
{
return this.vnutrennij_diametr;
}
public void set_dlina(int dlina)
{
this.dlina = dlina;
}
public int get_dlina()
{
return this.dlina;
}
public void set_hardness(int hardness)
{
this.hardness = hardness;
}
public int get_hardness()
{
return this.hardness;}
public void set_dlina_proby(int dlina_proby)
{
this.dlina_proby = dlina_proby;
}
public int get_dlina_proby()
{
return this.dlina_proby;
}
public void set_tverdPosleTO(int tverdPosleTO)
{
this.tverdPosleTO = tverdPosleTO;
}
public int get_tverdPosleTO()
```

```
{
return this.tverdPosleTO;
}
```

Метод для определения норматива времени отрезной операции из класса Vtulka_s_bachromoj:

```
public double get_time_norm_otrez()
{
double norm = 0;
if (naruzhnij_diametr <= 450)
{
if (hardness <= 200)
{
norm = naruzhnij_diametr / (5.0 * 60);
}
else if (hardness <= 247)
{
norm = naruzhnij_diametr / (3.0 * 60);
}
else if (hardness <= 269)
{
norm = naruzhnij_diametr / (2.5 * 60);
}
else if (hardness >= 270)
{
norm = naruzhnij_diametr / (1.6 * 60);
}
}
else if (naruzhnij_diametr <= 1100)
{
if (hardness <= 200)
{
norm = naruzhnij_diametr / (6.0 * 60);
}
else if (hardness <= 247)
{
norm = naruzhnij_diametr / (3.5 * 60);
}
else if (hardness <= 269)
{
norm = naruzhnij_diametr / (3.0 * 60);
}
```

```
else if (hardness >= 270)
{
norm = naruzhnij_diametr / (2.0 * 60);
}
}
return (Math.Round(norm, 2));
}
```

Метод для определения норматива времени токарной операции из класса Vtulka_s_bachromoj:

```
public double get_time_norm_tokar()
{
double DL = 0, dL = 0;
while (true)
{
int len = 0;
len = 80;
if (naruzhnij_diametr <= 225 && dlina <= len)
{
DL = 1.3;
break;
}
if (naruzhnij_diametr <= 250 && dlina <= len)
{
DL = 1.6;
break;
 }
if (naruzhnij_diametr <= 275 && dlina <= len)
{
 DL = 1.9;
 break;
 }
if (naruzhnij_diametr <= 300 && dlina <= len)
{
DL = 2.2;
break;
 }
if (naruzhnij_diametr <= 325 && dlina <= len)
{
DL = 2.5;
break; }
if (naruzhnij_diametr <= 350 && dlina <= len)
```

```
{
 DL = 2.8;
break;
 }
if (naruzhnij_diametr <= 400 && dlina <= len)
{
DL = 3.1;
break;
 }
if (naruzhnij_diametr <= 450 && dlina <= len)
{
 DL = 3.5;
break;
 }
if (naruzhnij_diametr <= 500 && dlina <= len)
{
DL = 3.8;
 break;
 }
if (naruzhnij_diametr <= 550 && dlina <= len)
{
DL = 4.2;
 break;
 }
if (naruzhnij_diametr <= 600 && dlina <= len)
{
DL = 4.5;
break;
 }
if (naruzhnij_diametr <= 650 && dlina <= len)
{
DL = 5.0;
break;
 }
if (naruzhnij_diametr <= 700 && dlina <= len)
{
DL = 5.3;
break;
 }
if (naruzhnij_diametr <= 750 && dlina <= len)
{
DL = 5.6;
break;
}
```

```
if (naruzhnij_diametr <= 800 && dlina <= len)
{
 DL = 6.0;
 break;
 }
//часть кода пропущена
}
return dL + DL;
}
```

Метод для определения кода пилы из класса Plastina:

```
public int get_code_of_saw()
{
int code = 0;
int mediumNum=width;
if (length >= height && height >= width) mediumNum = height;
if (width >= height && height >= length) mediumNum = height;
if (height >= length && length >= width) mediumNum = length;
if (width >= length && length >= height) mediumNum = length;
if (mediumNum <= 300)
{
code = 4812;
}
else if (mediumNum <= 450)
{
code = 4814;
}
else if (mediumNum <= 1100)
{
code = 4816;
}
return code;
}
```

Метод для определения норматива времени для отрезной операции из класса Plastina:

```
public double get_time_norm_otrez()
{
double norm = 0;
int mediumNum = width;
if (mediumNum <= 450)
```

```
{
if (hardnessProba <= 200)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(5.0 * 60);
}
else if (hardnessProba <= 247)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(3.0 * 60);
}
else if (hardnessProba <= 269)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(2.5 * 60);
}
else if (hardnessProba >= 270)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(1.6 * 60);
}
}
else if (mediumNum <= 1100)
{
if (hardnessProba <= 200)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(6.0 * 60);
}
else if (hardnessProba <= 247)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(3.5 * 60);
}
else if (hardnessProba <= 269)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(3.0 * 60);
}
else if (hardnessProba >= 270)
{
norm = Math.Sqrt((widthObd * widthObd + heightObd * heightObd) / 2) /
(2.0 * 60);
}
```

```
}
return (Math.Round(norm, 2));
}
```

Метод для определения норматива времени установки детали из класса Plastina:

```
public double get_time_norm_installing()
{
double weight = length / 1000.0 * height / 1000.0 * width / 1000.0 *
7850;
if (weight > 8000 && weight<15000) return (0.77 * 6.0);
if (weight > 3000) return (0.565 * 6.0);
if (weight > 1000) return (0.475 * 6.0);
if (weight > 500) return (0.409 * 6.0);
if (weight > 200) return (0.333 * 6.0);
if (weight > 100) return (0.275 * 6.0);
if (weight > 60) return (0.23 * 6.0);
if (weight > 30) return (0.195 * 6.0);
if (weight > 20) return (0.08 * 6.0);
if (weight > 12) return (0.07 * 6.0);
if (weight > 6) return (0.06 * 6.0);
if (weight > 5) return (0.053 * 6.0);
if (weight > 3) return (0.047 * 6.0);
if (weight > 1) return (0.04 * 6.0);
if (weight > 0) return (0.029 * 6.0);
else return 0;
}
```

Метод для определения норматива времени для снятия фасок из класса Plastina:

```
public double get_time_norm_faski()
{
double time_norm_faski = 0;
time_norm_faski += calculate_time_norm_faski(lengthObd) * 4;
time_norm_faski += calculate_time_norm_faski(heightObd) * 4;
time_norm_faski += calculate_time_norm_faski(widthObd) * 4;
return time_norm_faski;
}
private double calculate_time_norm_faski(int length_of_side)
{
if (length_of_side <= 500) return 0.023;
```

```
if (length_of_side <= 1000) return 0.025;
if (length_of_side <= 1500) return 0.026;
if (length_of_side <= 2000) return 0.027;
else return 0;
}
```

Метод определения норматива времени строгальной операции из класса Plastina:

```
public double get_time_norm_strog()
{
double time_norm_height = 0;
double time_norm_width = 0;
double time_norm_length = 0;

time_norm_height = Math.Ceiling((height + 7 - heightObd) / 8.0);
if(length>width)
{
time_norm_height = time_norm_height * get_table_time_norm(length, width);
}
else
{
time_norm_height = time_norm_height * get_table_time_norm(width, length);
}
time_norm_width = Math.Ceiling((width + 7 - widthObd) / 8.0);
if (length > heightObd)
{
time_norm_width = time_norm_width * get_table_time_norm(length,
heightObd);
}
else
{
time_norm_width = time_norm_width * get_table_time_norm(heightObd,
length);
}
time_norm_length = Math.Ceiling((length + 10 - lengthObd) / 5.0);
if (widthObd > heightObd)
{
time_norm_length = time_norm_length * get_table_time_norm(widthObd,
heightObd);
}
else
{
```

```
time_norm_length = time_norm_length * get_table_time_norm(heightObd,
widthObd);
}
return time_norm_length+time_norm_height+time_norm_width;
}
private double get_table_time_norm(int big_side, int small_side)
{
if (big_side <= 100 && small_side <= 20) return 0.023;
if (big_side <= 100 && small_side <= 30) return 0.028;
if (big_side <= 100 && small_side <= 50) return 0.033;
if (big_side <= 100 && small_side <= 75) return 0.038;
if (big_side <= 100 && small_side <= 100) return 0.048;}
//часть кода пропущена
if (big_side <= 2000 && small_side <= 20)
{
return 0.12;
}
if (big_side <= 2000 && small_side <= 30)
{
return 0.14;
}
if (big_side <= 2000 && small_side <= 50)
{
return 0.2;
}
if (big_side <= 2000 && small_side <= 75)
{
return 0.26;
}
if (big_side <= 2000 && small_side <= 100)
{
return 0.34;
}
if (big_side <= 2000 && small_side <= 125)
{
return 0.42;
}
if (big_side <= 2000 && small_side <= 150)
{
return 0.48;
}
if (big_side <= 2000 && small_side <= 175)
{
return 0.55;
```

```
}
if (big_side <= 2000 && small_side <= 225)
{
return 0.69;
}
if (big_side <= 2000 && small_side <= 300)
{
return 0.89;
}
if (big_side <= 2000 && small_side <= 400)
{
return 1.13;
}
if (big_side <= 2000 && small_side <= 550)
{
return 1.6;
}
if (big_side <= 2000 && small_side <= 750)
{
return 2.12;
}
if (big_side <= 2000 && small_side <= 1000)
{
return 2.89;
}
else return 0;
}
```

Метод для определения норматива времени строгальной операции под УЗД:

```
public double get_time_norm_UZD()
{
double time_norm_height = 1;
double time_norm_width = 1;
double time_norm_length = 1;

if (lengthObd > widthObd)
{
time_norm_height = 2 * time_norm_height *
get_table_time_norm_UZD(lengthObd, widthObd);
}
else
```

```
{
time_norm_height = 2 * time_norm_height *
get_table_time_norm_UZD(widthObd, lengthObd);
}
if (lengthObd > heightObd)
{
time_norm_width = 2 * time_norm_width *
get_table_time_norm_UZD(lengthObd, heightObd);
}
else
{
time_norm_width = 2 * time_norm_width *
get_table_time_norm_UZD(heightObd, lengthObd);
}
if (widthObd > heightObd)
{
time_norm_length = 2 * time_norm_length *
get_table_time_norm_UZD(widthObd, heightObd);
}
else
{
time_norm_length = 2 * time_norm_length *
get_table_time_norm_UZD(heightObd, widthObd);
}

return time_norm_length + time_norm_height + time_norm_width;
}

private double get_table_time_norm_UZD(int big_side, int small_side)
{
if (big_side <= 500 && small_side <= 50)
{
return 0.11;
}
if (big_side <= 500 && small_side <= 75)
{
return 0.12;
}
if (big_side <= 500 && small_side <= 100)
{
return 0.13;
}
if (big_side <= 500 && small_side <= 150)
{
```

```
return 0.16;
}
if (big_side <= 500 && small_side <= 200)
{
return 0.19;
}
if (big_side <= 500 && small_side <= 250)
{
return 0.22;
}
if (big_side <= 500 && small_side <= 300)
{
return 0.25;
}
if (big_side <= 500 && small_side <= 350)
{
return 0.28;
}
if (big_side <= 500 && small_side <= 400)
{
return 0.32;
}
if (big_side <= 500 && small_side <= 450)
{
return 0.35;
}
if (big_side <= 500 && small_side <= 500)
{
return 0.38;
}
//часть кода пропущена
else return 0;
}
```

Код, устанавливающий коэффициенты на нормативы времени для токарной операции в зависимости от вида стали:

```
double tokarNorm = 0;
if (radioUgler.IsChecked == true && checkBoxTO.IsChecked == true)
{
tokarNorm = vtulka.get_time_norm_tokar() * 0.8;
}
else if (radioUgler.IsChecked == true && (checkBoxUZD.IsChecked == true
```

```
|| checkBoxUZDTech.IsChecked == true))
{
tokarNorm = vtulka.get_time_norm_tokar() * 0.7;
}
if (radioNerzh.IsChecked == true)
{
tokarNorm = vtulka.get_time_norm_tokar() * 2.0;
}
if (radioLegir.IsChecked == true)
{
if (hardness < 174)
{
tokarNorm = vtulka.get_time_norm_tokar();
}
else if (hardness < 203)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.2;
}
else if (hardness < 260)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.3;
}
else if (hardness >= 260)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.5;
}
}
if (radioMartenFer.IsChecked == true)
{
if (hardness < 174)
{
tokarNorm = vtulka.get_time_norm_tokar();
}
else if (hardness < 203)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.2;
}
else if (hardness < 260)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.3;
}
else if (hardness >= 260)
{
tokarNorm = vtulka.get_time_norm_tokar() * 1.7;
```

```
}
}
```

Код для формирования информации об отрезной операции (втулка):

```
if (checkOtrez1.IsChecked.Value)
{
result += vtulka.get_code_of_saw().ToString() + "
Отрезная
460  3  1  0.50  " + (vtulka.get_time_norm_otrez() * 2).ToString() +
"\r\n\r\n" + "   Установить, выверить и закрепить." + "\r\n   Отрезать с
переустановкой бахрому в размер " + (vtulka.get_dlina() + 20 +
vtulka.get_dlina_proby()).ToString() + " (2 реза)" + "\r\n\r\n";
}
```

Код для формирования информации о токарной операции:

```
if (checkBoxTokar1.IsChecked.Value && checkBoxUZDTech.IsChecked.Value)
{
result += "4150 Токарная 474    3       1    0.42     " +
tokarNorm.ToString(engCulture) + "\r\n\r\n   Установить, выверить и
закрепить деталь в кулаках." + "\r\n   С переустановкой детали точить ø"
+ (vtulka.get_naruzhnij_diametr() + 10).ToString() + ", расточить ø" +
(vtulka.get_vnutrennij_diametr()).ToString() + " и\r\n   подрезать торцы
в размер " + (vtulka.get_dlina() + 5 +
vtulka.get_dlina_proby()).ToString() + " под УЗК по тех.решению.\r\n
Выдержать шероховатость Ra 3.2. Снять фаски 3х45° по\r\n   СТП 203-2010
п.2.11";
}
else if (checkBoxTokar1.IsChecked.Value && checkBoxUZD.IsChecked.Value)
{
result += "4150 Токарная 474    3       1    0.42     " +
tokarNorm.ToString(engCulture) + "\r\n\r\n   Установить, выверить и
закрепить деталь в кулаках." + "\r\n   С переустановкой детали точить ø"
+ (vtulka.get_naruzhnij_diametr() + 5).ToString() + ", расточить ø" +
(vtulka.get_vnutrennij_diametr() - 5).ToString() + " и\r\n   подрезать
торцы в размер " + (vtulka.get_dlina() + 5 +
vtulka.get_dlina_proby()).ToString() + " под УЗК.\r\n   Выдержать
шероховатость Ra 3.2. Снять фаски 3х45° по\r\n   СТП 203-2010 п.2.11";
}
else if (checkBoxTokar1.IsChecked.Value && checkBoxTO.IsChecked.Value)
{
result += "4150 Токарная 474    3       1    0.42     " +
tokarNorm.ToString(engCulture) + "\r\n\r\n   Установить, выверить и
```

```
закрепить деталь в кулаках." + "\r\n   С переустановкой детали точить ø"
+ (vtulka.get_naruzhnij_diametr() + 10).ToString() + ", расточить ø" +
(vtulka.get_vnutrennij_diametr()).ToString() + " и\r\n   подрезать торцы
в размер " + (vtulka.get_dlina() + 5 +
vtulka.get_dlina_proby()).ToString() + " под ТО.\r\n   Снять фаски 5х45°
по СТП 203-2010 п.2.11 по эскизу";
}
```

Код для указания дальнейших действий после завершения токарной операции:

```
result += "\r\n\r\n  Контрольная" + "\r\n\r\n";
if (checkBoxUZD.IsChecked.Value || checkBoxUZDTech.IsChecked.Value)
{
result += "   Подвергнуть деталь контролю ультразвуком согласно
чертежу.\r\n";
}
if (checkBoxTO.IsChecked.Value && CheckVyrezObraz.IsChecked.Value)
{
result += "   Отправить на ТО и получить обратно.\r\n\r\n";
}
else if (checkBoxTO.IsChecked.Value)
{
result += "   Отправить на ТО.\r\n ";
}
```

Код для формирования информации об операциях «Отрезка пробы» и «Вырезка заготовок образцов»:

```
if (checkOtrezProby.IsChecked.Value == true)
{
result += vtulka.get_code_of_saw().ToString() + " Отрезная 460  3  1
0.50  " + vtulka.get_time_norm_otrez().ToString() + "\r\n\r\n" + "
Установить, выверить и закрепить." + "\r\n   Отрезать пробу H = " +
vtulka.get_dlina_proby().ToString() + "\r\n\r\n";
}
if (CheckVyrezObraz.IsChecked.Value == true)
{
result += "\r\n1211 Вырезка заготовок образцов 008  3  1  2.70\r\n\r\n
Вырезать заготовки образцов для механических \r\n   испытаний по типовому
техпроцессу СТП 0112-348-72:\r\n          - 1 разрывной, - 2 ударных";
result += "\r\n\r\n Слесарная\r\n\r\n   Зачистить заусенцы на
образцах.\r\n\r\n Маркирование\r\n\r\n   Маркировать номер плавки, номер
```

```
заказа и обозначение\r\n   детали на образцах и остатке кольца.\r\n
Маркировку нанести на боковой стороне заготовок\r\n   образцов по РДИ
0112-824-86";
result += "\r\n\r\n Контрольная\r\n\r\n   Заготовки образцов передать в
ЦЗЛ.\r\n   Сдать ОТК остаток кольца с маркировкой\r\n   на случай
повторных испытаний.\r\n   К работе приступать только после получения\r\n
удовлетворительных результатов мех.испытаний.";
}
```

Код для формирования информации об операциях и нормативах времени для пластины:

```
if (checkBoxTO.IsChecked.Value && checkBoxStrogaln.IsChecked.Value &&
checkBoxUZD.IsChecked.Value)
{
result += "4712 Строгальная 470    3       1    " +
Math.Round(plastina.get_time_norm_installing(), 2).ToString(engCulture) +
"     " + Math.Round(resultStrogNorm, 2).ToString(engCulture) + "\r\n\r\n
Установить, выверить и закрепить. Строгать в размеры" + "\r\n   " +
plastina.LengthObd.ToString() + "x" + plastina.WidthObd.ToString() + "x"
+ plastina.HeightObd.ToString() + " под УЗД. Снять фаски 5х45°";
result += "\r\n\r\n Контрольная";
result += "\r\n\r\n    Подвергнуть деталь контролю ультразвуком согласно
чертежу.";
result += "\r\n\r\n    Отправить на ТО.";
}
else if (checkBoxTO.IsChecked.Value && checkBoxStrogaln.IsChecked.Value)
{
result += "4712 Строгальная 470    3       1    " +
Math.Round(plastina.get_time_norm_installing(), 2).ToString(engCulture) +
"     " + Math.Round(resultStrogNorm, 2).ToString(engCulture) + "\r\n\r\n
Установить, выверить и закрепить. Строгать в размеры" + "\r\n   " +
plastina.LengthObd.ToString() + "x" + plastina.WidthObd.ToString() + "x"
+ plastina.HeightObd.ToString() + " под ТО. Снять фаски 5х45°";
result += "\r\n\r\n Контрольная";
result += "\r\n\r\n    Отправить на ТО.";
}
else if (checkBoxUZD.IsChecked.Value && checkBoxStrogaln.IsChecked.Value)
{
result += "4712 Строгальная 470    3       1    " +
Math.Round(plastina.get_time_norm_installing(), 2).ToString(engCulture) +
"     " + Math.Round(resultStrogNorm, 2).ToString(engCulture) + "\r\n\r\n
Установить, выверить и закрепить. Строгать в размеры" + "\r\n   " +
```

```
plastina.LengthObd.ToString() + "x" + plastina.WidthObd.ToString() + "x"
+ plastina.HeightObd.ToString() + " под УЗД. Снять фаски 3х45°";
result += "\r\n\r\n Контрольная";
result += "\r\n\r\n    Подвергнуть деталь контролю ультразвуком согласно
чертежу.";
}
if (checkOtrezProby.IsChecked.Value == true)
{
result += "\r\n\r\n" + plastina.get_code_of_saw().ToString() + " Отрезная
460    3       1     0.50    " +
plastina.get_time_norm_otrez().ToString(engCulture) + "\r\n\r\n" + "
Установить, выверить и закрепить." + "\r\n   Отрезать пробу H = " +
plastina.DlinaProby.ToString() + "\r\n\r\n";
}
if (CheckVyrezObraz.IsChecked.Value == true)
{
result += "\r\n1211 Вырезка заготовок образцов 008  3  1  2.70\r\n\r\n
Вырезать заготовки образцов для механических \r\n   испытаний по типовому
техпроцессу СТП 0112-348-72:\r\n          - 1 разрывной, - 2 ударных";
result += "\r\n\r\n Слесарная\r\n\r\n   Зачистить заусенцы на
образцах.\r\n\r\n Маркирование\r\n\r\n   Маркировать номер плавки, номер
заказа и обозначение\r\n   детали на образцах и остатке кольца.\r\n
Маркировку нанести на боковой стороне заготовок\r\n   образцов по РДИ
0112-824-86";
result += "\r\n\r\n Контрольная\r\n\r\n   Заготовки образцов передать в
ЦЗЛ.\r\n   Сдать ОТК остаток кольца с маркировкой\r\n   на случай
повторных испытаний.\r\n   К работе приступать только после получения\r\n
удовлетворительных результатов мех.испытаний.";
}
```

Код для проверки вхождения введенных пользователем данных в диапазон допустимых значений (втулка):

```
if (dlina > 600 || dlina <= 40)
{
MessageBox.Show("Длина втулки выходит за рамки возможностей этой
программы");
return;
}
if (naruzhnij_diametr > 800 || naruzhnij_diametr <= 200)
{
MessageBox.Show("Наружный диаметр выходит за рамки возможностей этой
программы");
```

```
return;
}
if (vnutrennij_diametr > 420 || vnutrennij_diametr <= 100)
{
MessageBox.Show("Внутренний диаметр выходит за рамки возможностей этой
программы на данный момент");
return;
}
if (hardness <= 15 || hardness > 700)
{
MessageBox.Show("Твёрдость указана некорректно");
return;
}
```

Код для уведомления о необходимости выбрать требования при наличии токарной операции и отключения возможности выбирать требования при её отсутствии:

```
if (checkBoxTokar1.IsChecked==true && checkBoxUZD.IsChecked==false &&
checkBoxUZDTech.IsChecked==false&& checkBoxTO.IsChecked==false)
{
MessageBox.Show("Выберите требование для токарной операции");
return;
}
private void checkBoxTokar1_Checked(object sender, RoutedEventArgs e)
{
if (checkBoxTokar1.IsChecked == false)
{
checkBoxTO.IsEnabled = false;
checkBoxUZD.IsEnabled = false;
checkBoxUZDTech.IsEnabled = false;
checkBoxTO.IsChecked = false;
checkBoxUZD.IsChecked = false;
checkBoxUZDTech.IsChecked = false;
}
else if (checkBoxTO!=null)
{
checkBoxTO.IsEnabled = true;
checkBoxUZD.IsEnabled = true;
checkBoxUZDTech.IsEnabled = true;
}
}
```

Код для проверки вхождения введенных пользователем данных в диапазон допустимых значений (пластина):

```
if (!int.TryParse(textWidth.Text, out width) || width > 1000 || width <
10)
{
MessageBox.Show("Поковочная ширина указана некорректно");
return;
}
if (!int.TryParse(textHeight.Text, out height) || height > 1000 || height
< 10)
{
MessageBox.Show("Поковочная высота указана некорректно");
return;
}
if (!int.TryParse(textLength.Text, out length) || length > 2000 || length
< 10)
{
MessageBox.Show("Поковочная длина указана некорректно");
return;
}
if (!int.TryParse(textHardness.Text, out hardness) || hardness > 700 ||
hardness < 15)
{
MessageBox.Show("Твёрдость указана некорректно");
return;
}
if (!int.TryParse(textWidthObd.Text, out widthObd) || widthObd > 1000 ||
widthObd < 10)
{
MessageBox.Show("Обдирочная ширина указана некорректно");
return;
}
if (!int.TryParse(textHeightObd.Text, out heightObd) || heightObd > 1000
|| heightObd < 10)
{
MessageBox.Show("Обдирочная высота указана некорректно");
return;
}
if (!int.TryParse(textLengthObd.Text, out lengthObd) || lengthObd > 2000
|| lengthObd < 10)
{
MessageBox.Show("Обдирочная длина указана некорректно");
```

```
return;
}
```

Код для проверки того, что длина имеет самую большую протяженность, а обдирочные размеры не превосходят поковочные:

```
if(length<width||length<height)
{
MessageBox.Show("Поковочная длина должна быть самым большим размером");
return;
}
if(lengthObd<widthObd||lengthObd<heightObd)
{
MessageBox.Show("Обдирочная длина должна быть самым большим размером");
return;
}
if(lengthObd>=length)
{
MessageBox.Show("Обдирочная длина должна быть меньше ковочной");
return;
}
if (widthObd >= width)
{
MessageBox.Show("Обдирочная ширина должна быть меньше ковочной");
return;
}
if (heightObd >= height)
{
MessageBox.Show("Обдирочная высота должна быть меньше ковочной");
return;
}
```