Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт повышения квалификации и переподготовки

руководящих работников и специалистов

по информационным технологиям и радиоэлектронике

Кафедра «Микропроцессорные системы и сети»

**К защите допустить**

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Полубок

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К ДИПЛОМНОМУ ПРОЕКТУ**

**НА ТЕМУ:**

**«Программное средство для управления производственным процессом на предприятии»**

Дипломник И.Н. Гордейчик

(дата, подпись)

Руководитель Е.Н. Шнейдеров

(дата, подпись)

Консультанты:

по ЕСПД и ЕСКД В.Н. Мухаметов

(дата, подпись)

по специальности Н.Л. Боброва

(дата, подпись)

Рецензент /\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (И.О. Фамилия)

МИНСК 2015

Содержание

[Введение 4](#_Toc414866693)

[1 Описание предметной области и определение требований к системе с точки зрения предметной области 6](#_Toc414866694)

[1.1 Аналитический обзор и анализ существующих аналогов 6](#_Toc414866695)

[1.2 Определение требований к системе 17](#_Toc414866696)

[1.3 Постановка задачи 18](#_Toc414866697)

[2 Методы и модели, положенные в основу проекта 20](#_Toc414866698)

[3 Разработка проекта программного обеспечения 25](#_Toc414866699)

[3.1 Функциональный анализ предметной области и разработка спецификаций требований 25](#_Toc414866700)

[3.2 Разработка структуры объекта в виде схемы ресурсов 35](#_Toc414866701)

[3.3 Разработка схемы работы системы 36](#_Toc414866702)

[3.4 Выбор СУБД и разработка проекта базы данных 36](#_Toc414866703)

[3.5 Разработка структуры классов 48](#_Toc414866704)

[4 Разработка алгоритмов и их программная реализация 49](#_Toc414866705)

[5 Тестирование программного средства 53](#_Toc414866706)

[Заключение 60](#_Toc414866707)

[Список использованных источников 62](#_Toc414866708)

[Приложение А Сценарий создания базы данных 63](#_Toc414866709)

[Приложение Б Диаграмма классов программы 60](#_Toc414866710)

[Приложение В Листинг кода 71](#_Toc414866711)

# Введение

Подготовка производства, планирование и изготовление – это важные этапы производства на любом предприятии.

Одним из первых этапов подготовки производства является разработка межцеховых технологических маршрутов (расцеховок), устанавливающих последовательность прохождения заготовок, деталей и сборочных единиц в производственных подразделениях предприятия. При расчете заготовок определяются нормы расхода материалов, необходимых для изготовления деталей. Заготовка должна быть рассчитана так, чтобы расход материала был наименьшим, но при этом также учитывается трудоёмкость изготовления детали. Поэтому окончательное решение о размере заготовки принимает технолог.

Также на этапе подготовки производства проводится разработка технологических процессов изготовления деталей. В условиях единичного и мелкосерийного типа производства разрабатывается маршрутная технология изготовления изделий на универсальном технологическом оборудовании и оснастке рабочими высокой квалификации. В серийном и массовом производстве разрабатываются подробные технологические процессы.

Этап планирования по сути дела является определяющей функцией управления предприятием, обеспечивающей эффективность производства и его деятельности в целом. На этом этапе определяются поэтапные сроки изготовления деталей и сборочных единиц.

Учет комплектующих, деталей и изделий на предприятии является важным аспектом, обеспечивающим непрерывное производство на предприятии. На этапе изготовления проводятся постоянный контроль количества изготовленных деталей и сборочных единиц, а также контроль выполнения сроков изготовления, определённых на этапе планирования.

*Целью* данного дипломного проекта является автоматизация работ по технологической подготовке производства, планированию и контролю изготовления продукции, а также организация единой базы технологических процессов на предприятии.

Для достижения поставленной цели необходимо решить следующие поставленные *задачи*:

* проведение анализа аналогов и постановка задачи;
* проектирование системы;
* разработка методов и моделей для данной системы;
* тестирование полученного программного продукта.

Для разработки программного продукта будут применяться следующие приложения:

* ERwin – CASE-средство для проектирования и документирования баз данных, а также для построения моделей;
* Rational Rose – средство визуального моделирования объектно-ориентированных информационных систем на языке UML;
* C# – объектно-ориентированный язык программирования.

1. Описание предметной области и определение требований к системе с точки зрения предметной области
   * 1. Аналитический обзор и анализ существующих аналогов

Существует большое количество различных программ управления производственным предприятием. В Беларуси наибольшую популярность на предприятиях получила программа «1С: Предприятие» (далее – «1С»). С помощью этого мощного инструмента можно управлять практически каждым шагом производства изделия. Но не все механизмы даже этой программы являются удобными во время эксплуатации, а некоторые функции порой и вовсе отсутствуют. Давайте рассмотрим достоинства и недостатки данной программы.

На этапе подготовки производства в информационную базу программы вводятся новые детали и сборочные единицы. Для изготовления этих деталей планируется маршрут и норма расхода необходимого материала. В «1С» имеется инструмент для расчета заготовки и нормы расхода материала, но габарит и профиль заготовки определяет технолог. Даже после выбора номенклатуры материала с заданными габаритами «1С» для расчета нормы расхода требует указать профиль материала (рисунок 1.1)

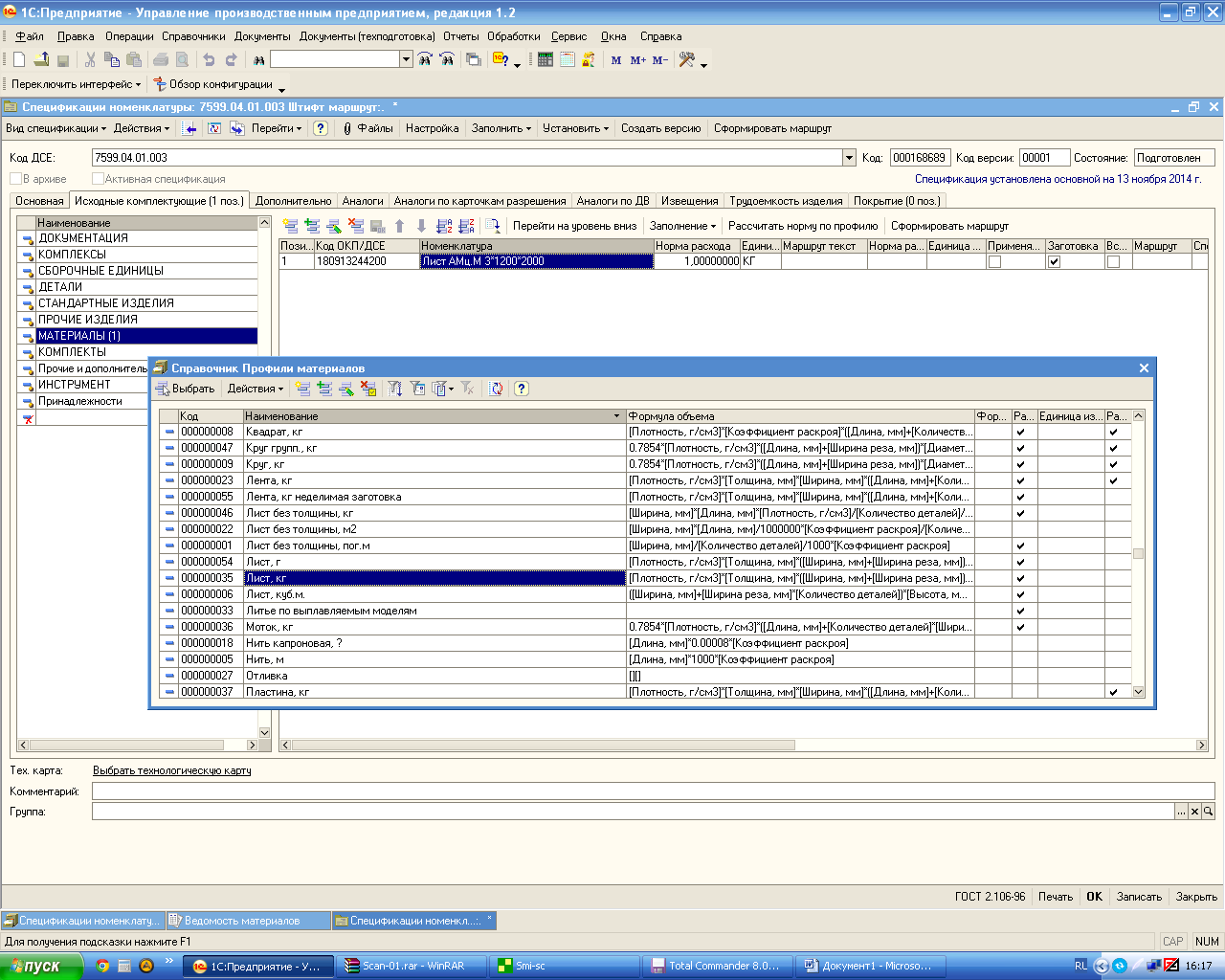


Рисунок 1.1 – Выбор профиля материала для расчёта нормы расхода в программе «1С»

При появлении диалогового окна для расчета нормы расхода поле со значением габарита заготовки остаётся пустым (в нашем случае толщина), хотя по номенклатуре материала можно определить габарит (рисунок 1.2).

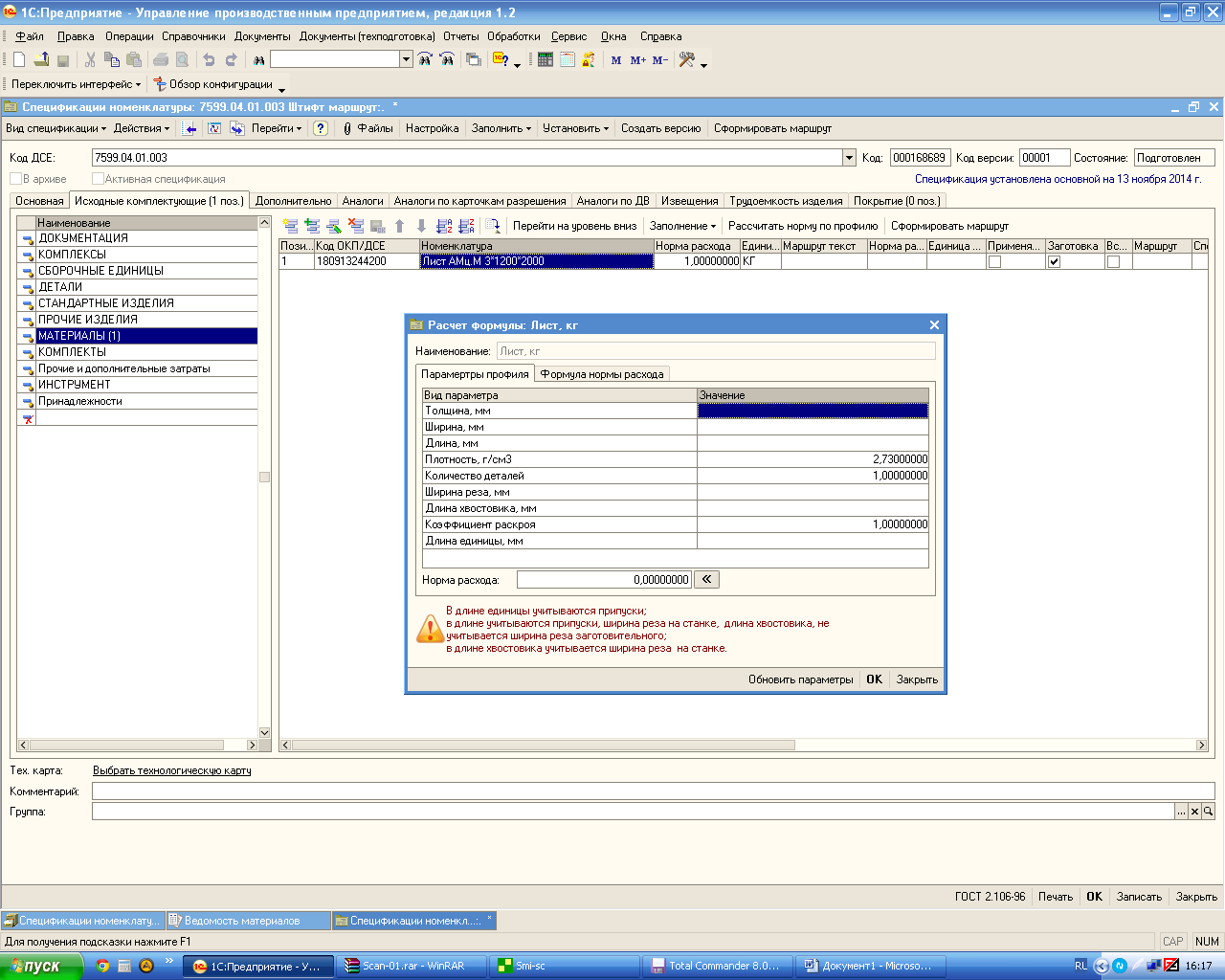


Рисунок 1.2 – Первоначальный вид окна расчёта нормы расхода

в программе «1С»

В последующих полях указываются габариты заготовки, которые должен определить технолог, исходя из необходимых припусков под дальнейшую обработку (рисунок 1.3). Это не очень удобно, так как технологу постоянно необходимо держать в голове все припуски, которые он должен учесть в заготовке: здесь и припуски на термическую обработку, и на механическую обработку, и на шлифовку, и на другие виды работ. Поэтому процесс расчета заготовок и норм расхода в «1С» является трудоёмким, так как требует от пользователя знание огромного количества припусков под различные виды обработки. Все эти припуски на каждом предприятии стандартизуются, и было бы удобно, если бы программа сама выбирала необходимые припуски на обработку, а участие пользователя при расчётах было бы минимальным.

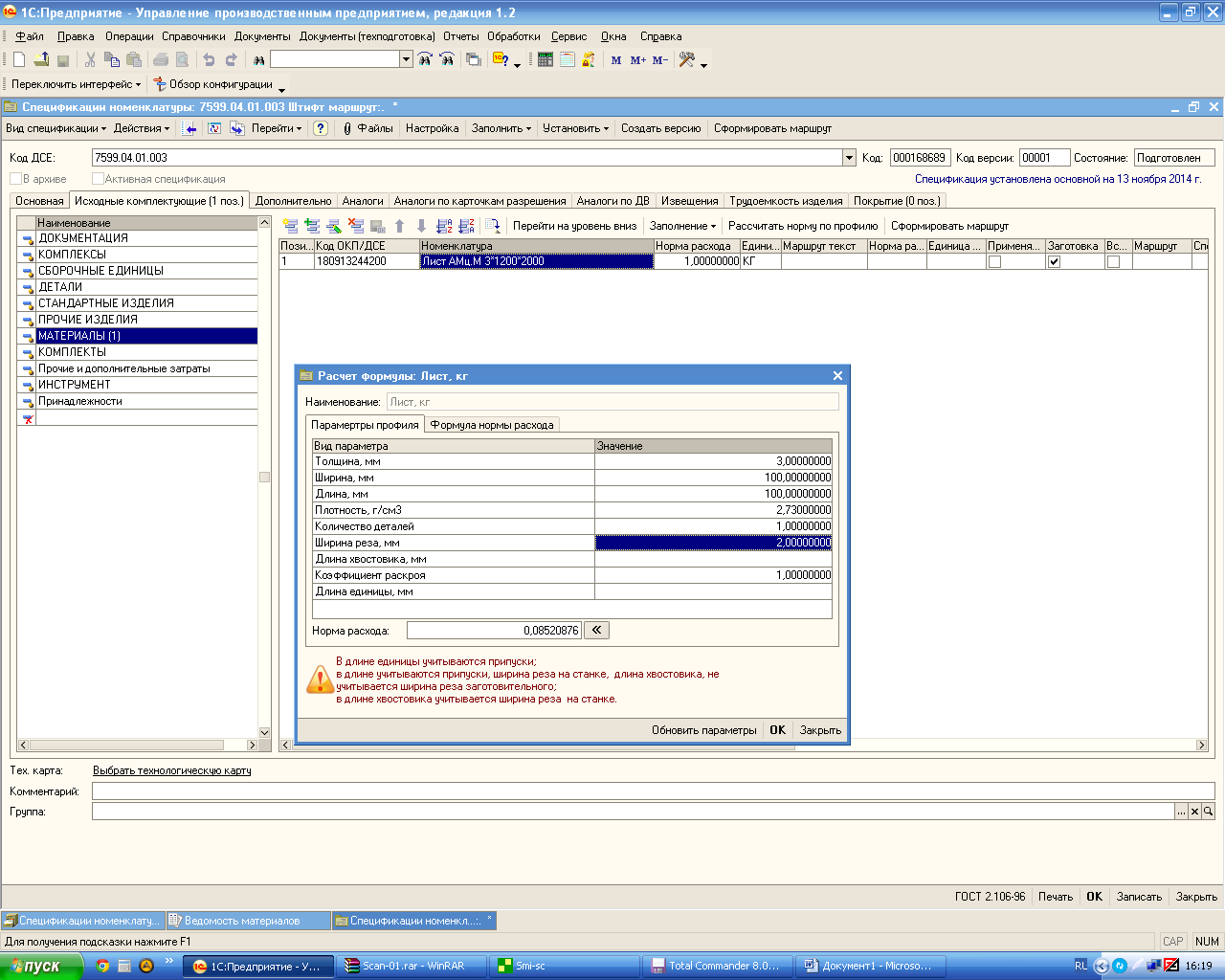


Рисунок 1.3 – Окно расчёта нормы расхода в программе «1С» после заполнения полей технологом ВМ

Расчёт норм расхода вспомогательных материалов на сборочные единицы программа и вовсе не поддерживает, поэтому это всё проводится вручную технологом. Технолог рассчитывает норму расхода, а после вносит её в программу (рисунок 1.4).

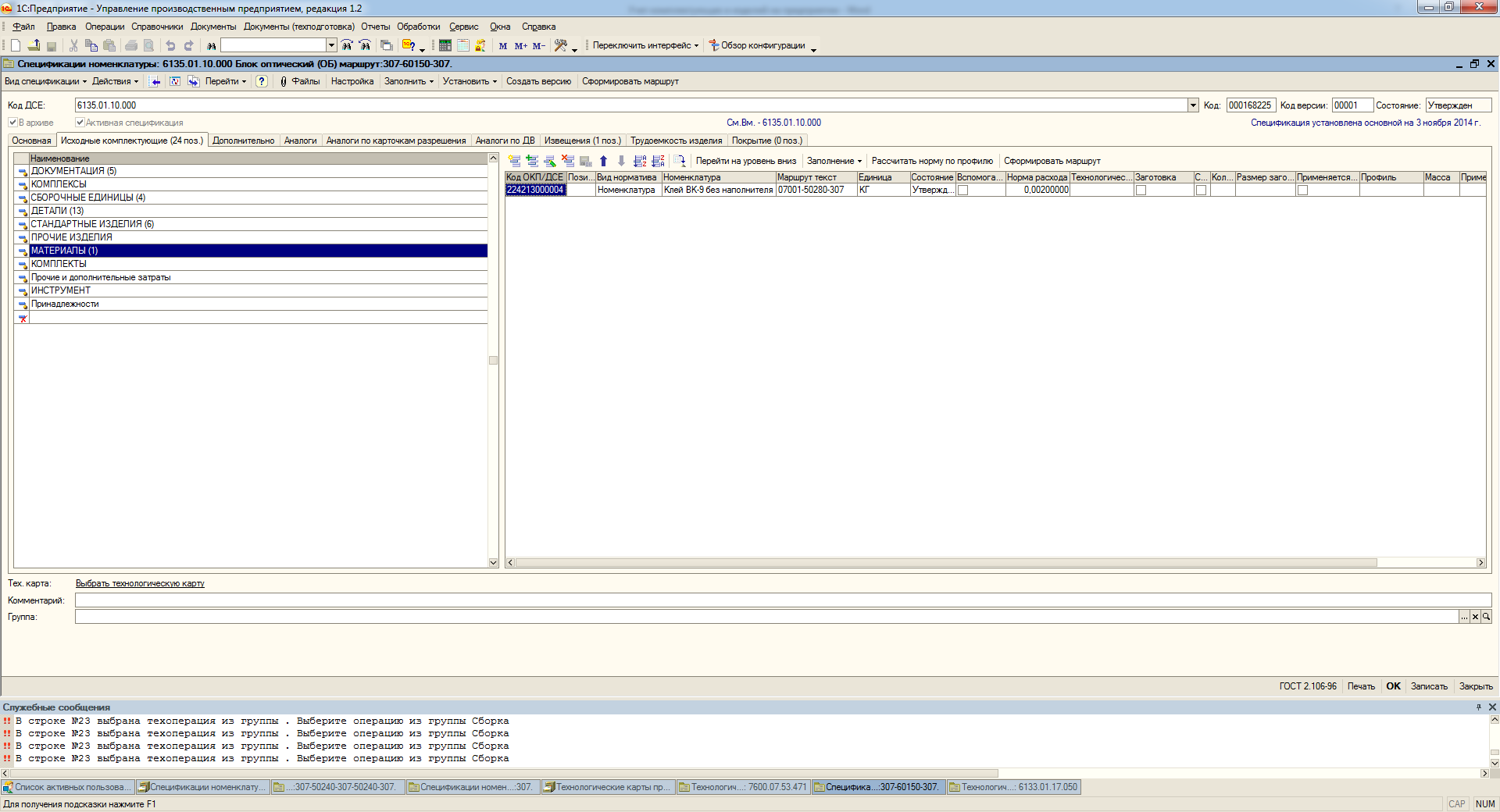


Рисунок 1.4 – Отображение списка материалов на конкретную спецификацию в программе «1С»

На этапе подготовки производства кроме расчёта заготовок и норм расхода материалов разрабатываются технологические процессы (техпроцессы) изготовления деталей и сборочных единиц. Достоинством «1С» является возможность разработки техпроцессов в программе и вывод техпроцессов на печать. Из недостатков можно отметить то, что для вывода на печать используется приложение Microsoft Word 2007. Другие версии просто не поддерживаются программой. Также нет возможности контролировать объём разработанных техпроцессов на изделие, т.е. определять сколько техпроцессов разработано и каких и сколько осталось.

Контроль планирования и изготовления деталей и сборочных единиц в «1С» основан на пооперационном маршруте из технологического паспорта. По операциям техпроцесса (рисунок 1.5) формируется технологический паспорт детали, по которому можно проследить на какой стадии изготовления находится деталь (рисунок 1.6).

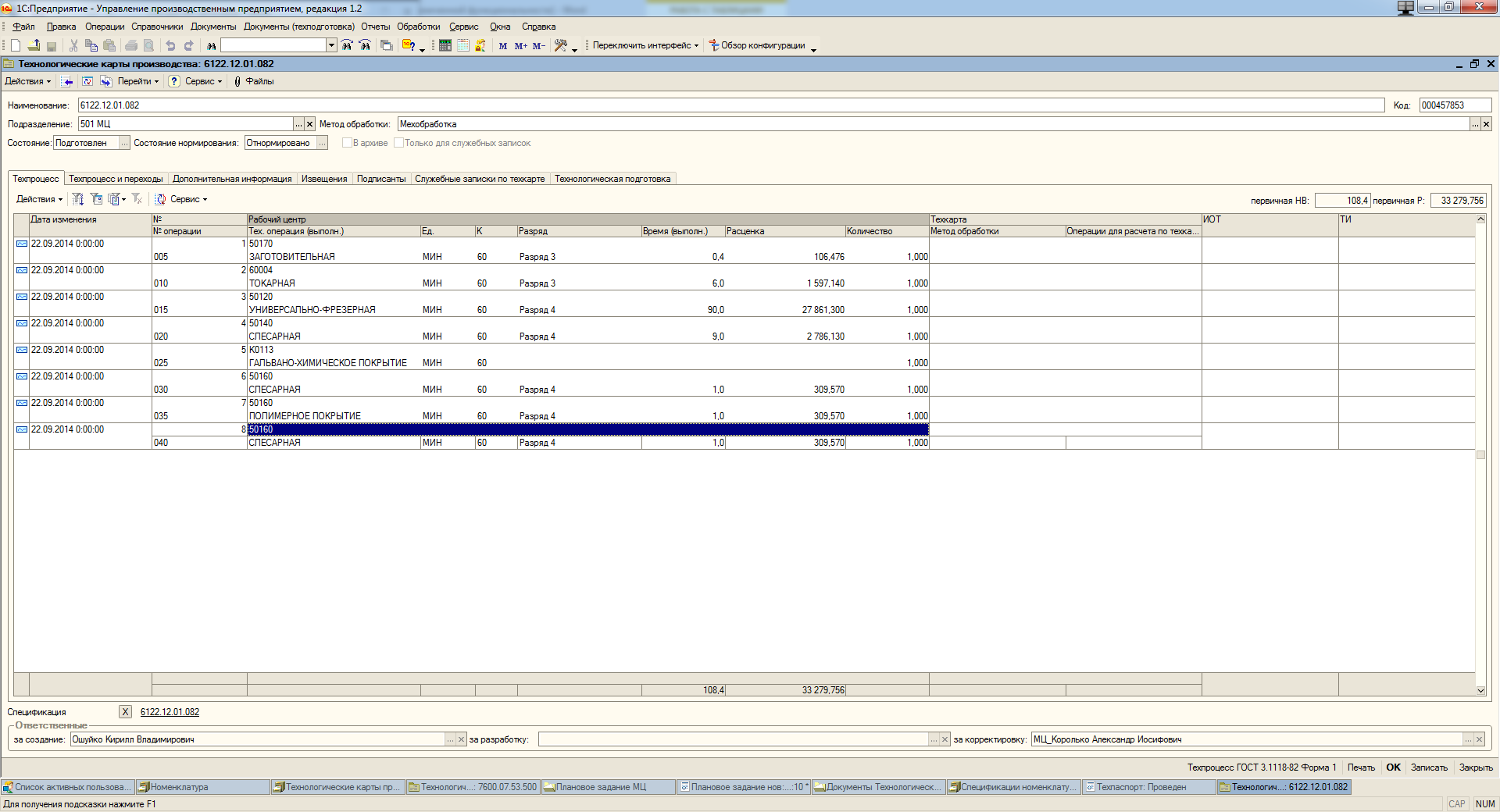


Рисунок 1.5 – Вид пооперационного техпроцесса в программе «1С»

Интерфейс программы для контроля изготовления деталей не удобен для пользователя, т. к. проходит разрозненно по технологическим паспортам. Если какие-то детали запущены по нескольким паспортам, то общей информации по изготовлению получить невозможно. Приходится просматривать по очереди все паспорта с данным номером детали (рисунок 1.7). Стоит также отметить неудобный поиск по конкретному номеру детали (рисунок 1.8).

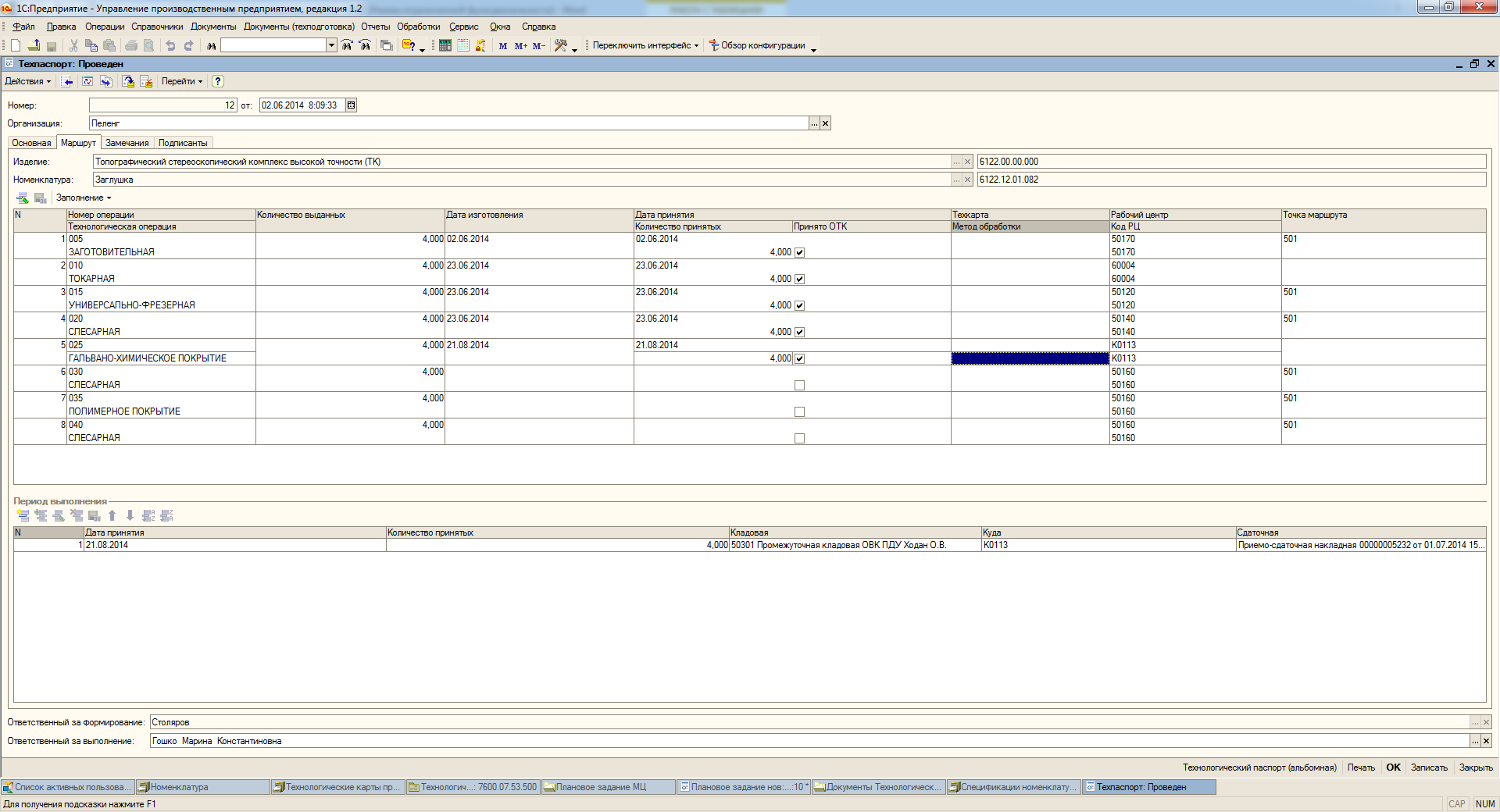


Рисунок 1.6 – Пример заполнения техпаспорта в программе «1С»

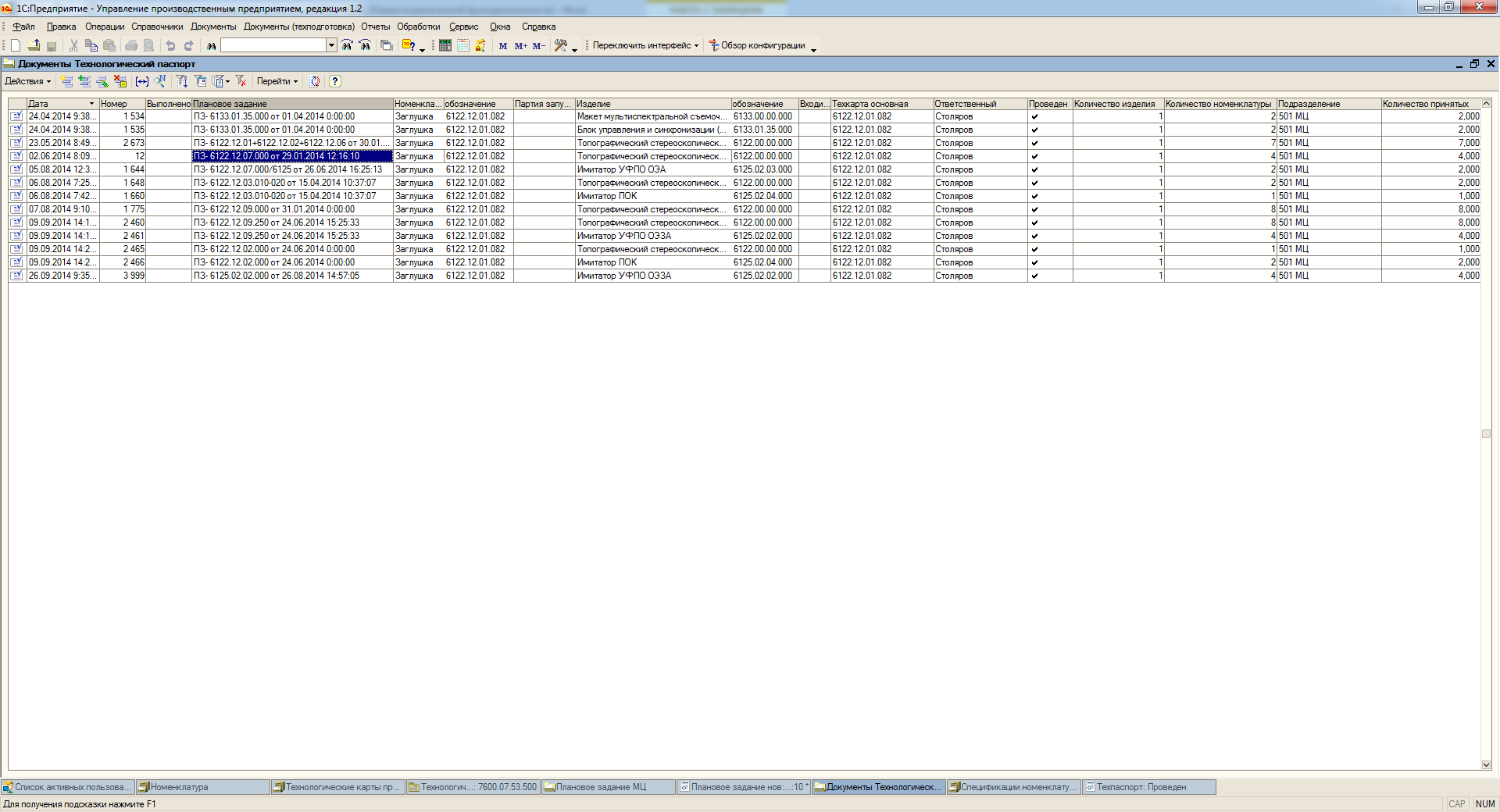


Рисунок 1.7 – Список созданных техпаспортов для одной детали

в программе «1С»

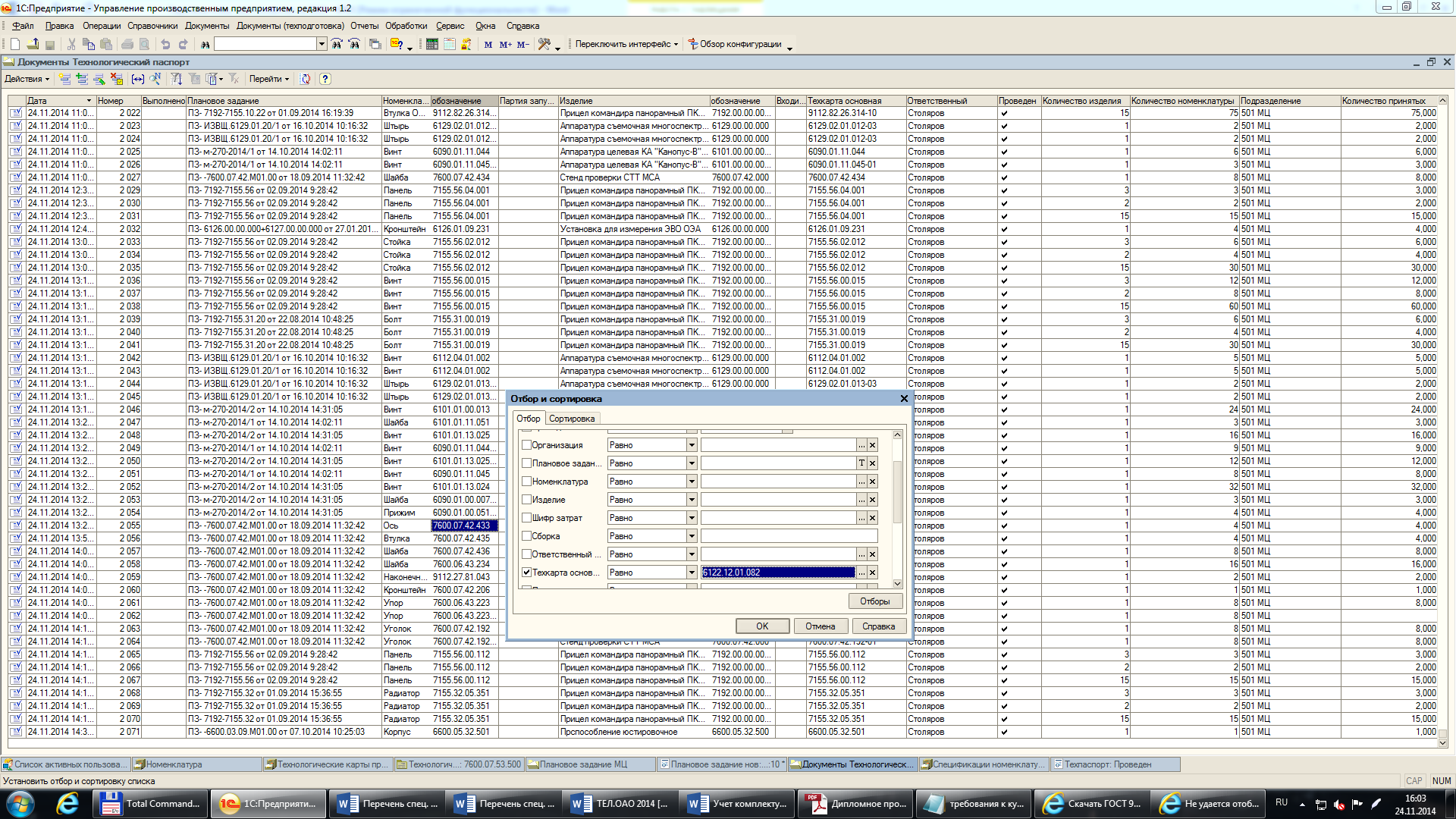


Рисунок 1.8 – Пример поиска техпаспортов по номеру детали в программе «1С»

Программа «1С: Предприятие» является мощным инструментом с огромным функционалом, но всё же существует множество узкоспециализированных программ управления производственным предприятием. Эти программы хоть и обладают меньшим функционалом, зато позволяют решать конкретную задачу с максимальным удобством для пользователя.

Одним из примеров таких программ является программа «ОПП» (оперативно-производственное планирование), предназначенная для учёта деталей и сборочных единиц в производстве, а также для контроля их передвижения между цехами-изготовителями. Поиск по номеру детали или сборочной единицы организован удобно для пользователя, что можно отнести к достоинствам этой программы (рисунок 1.9).

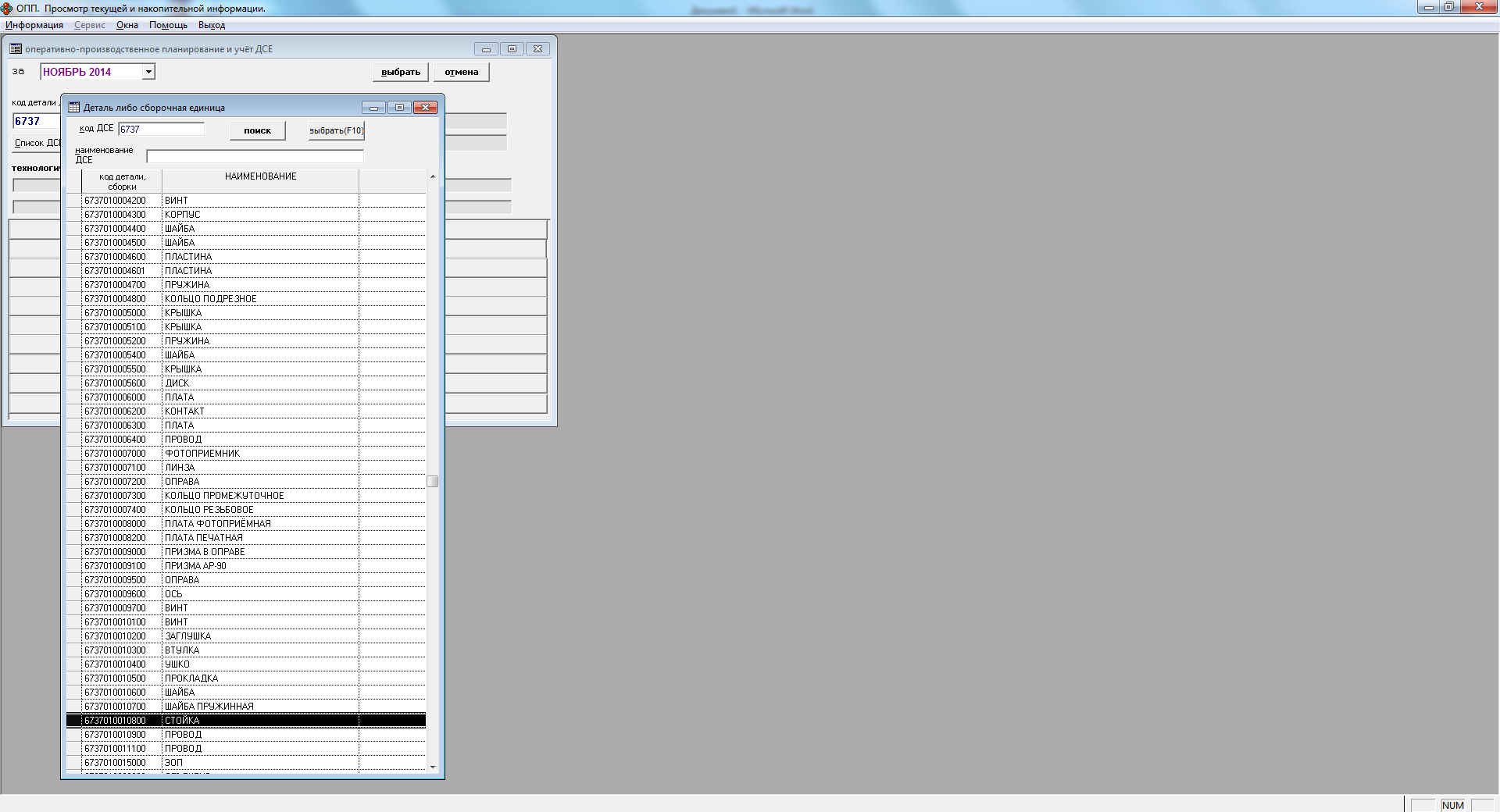


Рисунок 1.9 – Пример поиска деталей, соответствующих заданной маске, в программе «ОПП»

Программа отображает маршрут выбранной детали и позволяет просмотреть информацию о движении данной детали по цехам за конкретный период (рисунок 1.10), а также остатки по цехам (рисунок 1.11).

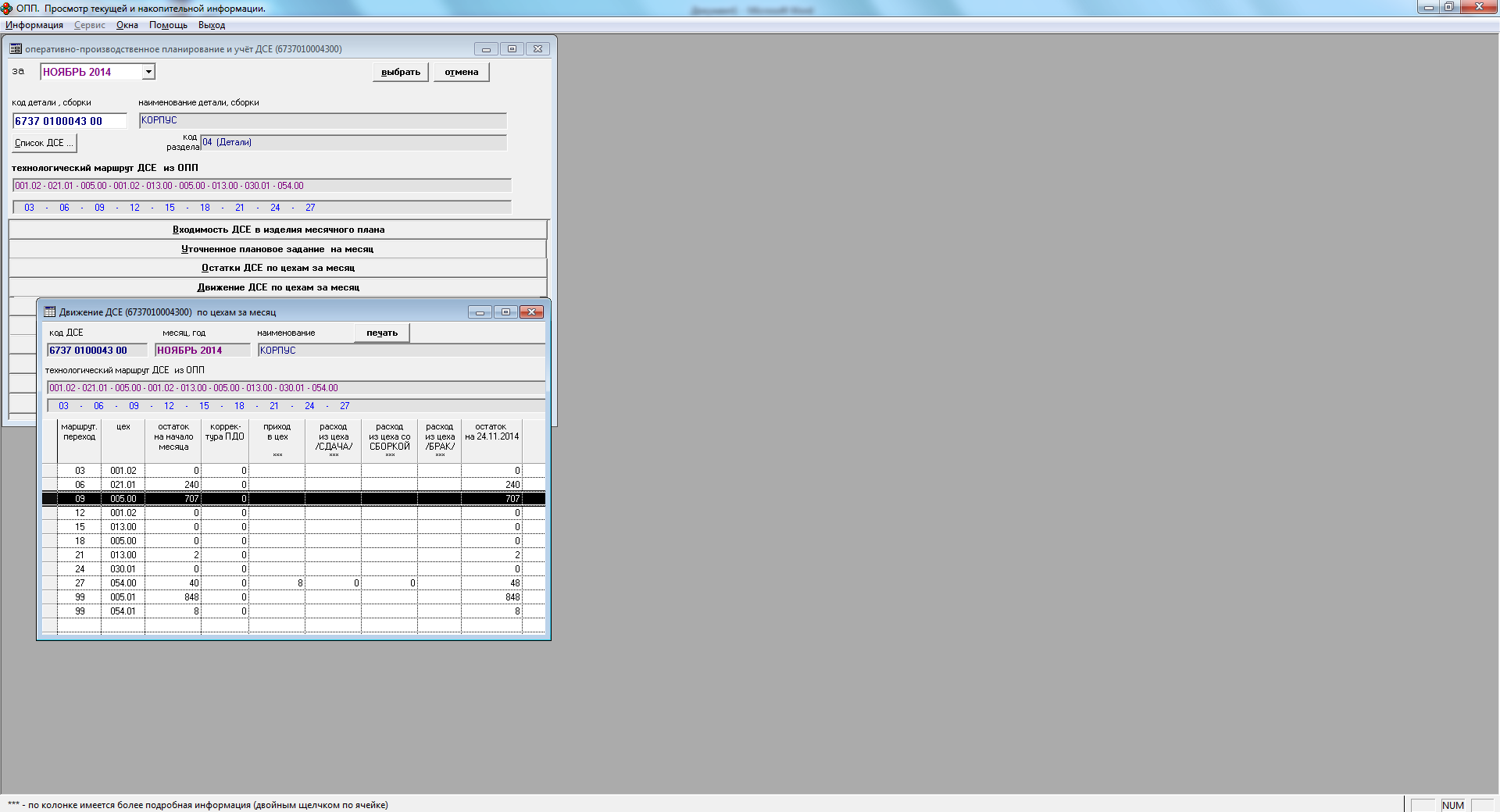


Рисунок 1.10 – Окно программы «ОПП», отображающее движение деталей по цехам за последний месяц

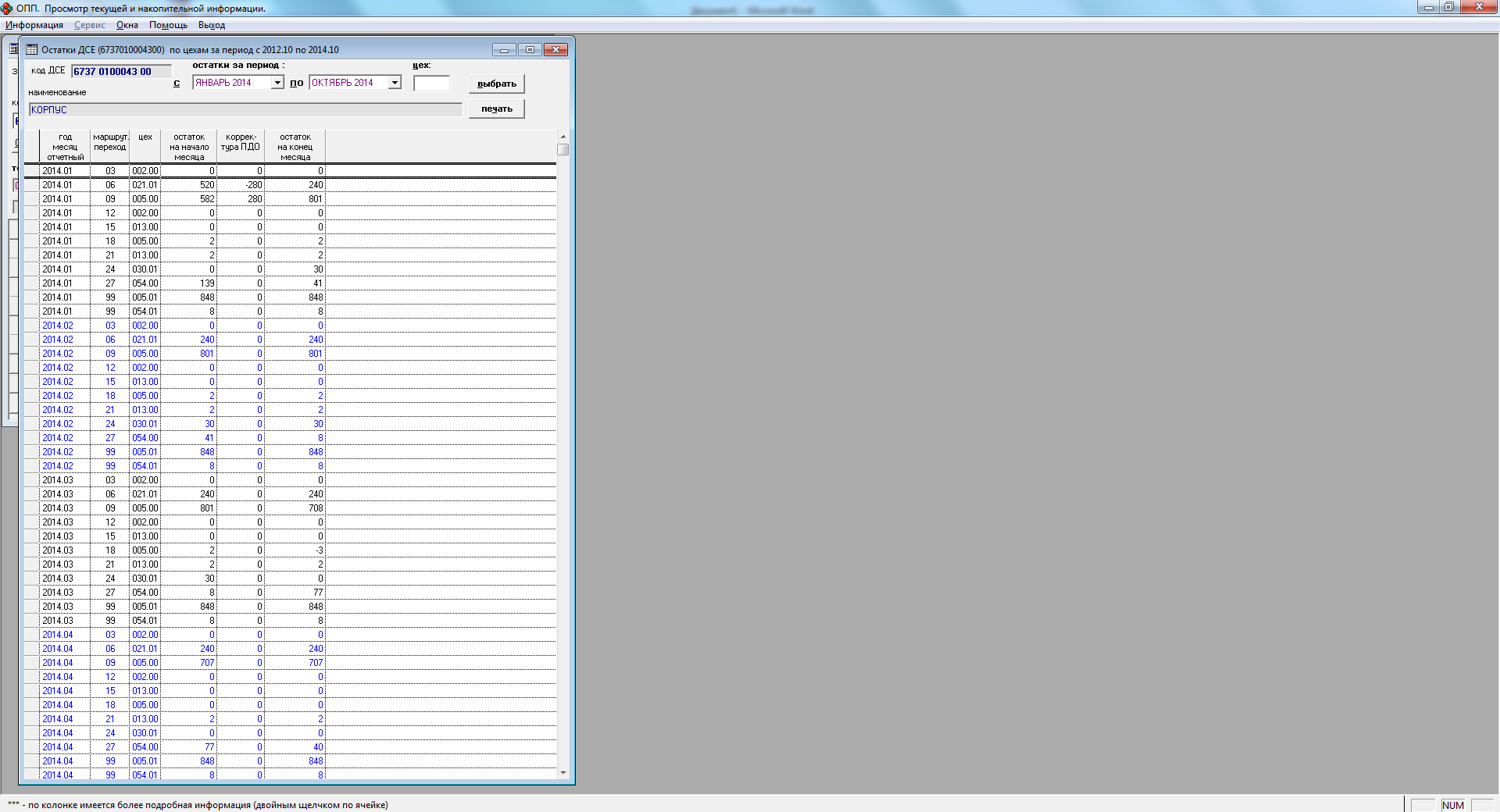


Рисунок 1.11 – Окно программы «ОПП», отображающее информацию об остатках деталей по цехам за выбранный период

Есть также возможность просмотра информации по конкретному цеху. Например, на рисунке 1.12 представлена информация по изделию 6737, детали которого изготавливались в цеху 006 за последний месяц.

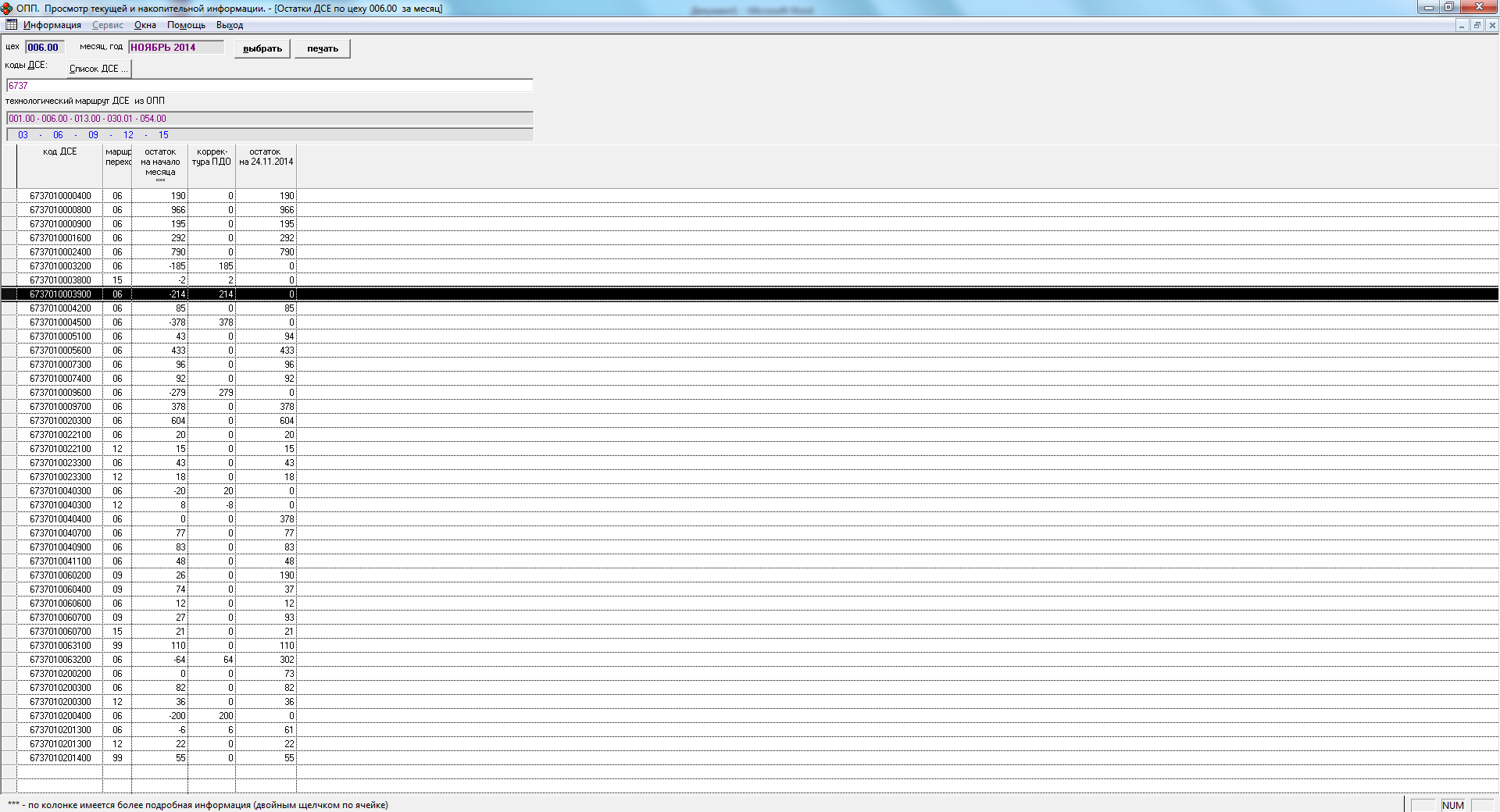


Рисунок 1.12 – Сформированный программой «ОПП» перечень всех деталей изделия 6737, изготавливаемых в цехе 006 в ноябре

При всём удобстве эксплуатации данной программы у неё есть множество недостатков. Во-первых, программа рассчитана в основном на серийное производство, так как для единичного или мелкосерийного удобнее отслеживать изготовление деталей по операциям, а не по маршрутам. Во-вторых, программа не поддерживает вывод отчётов на печать. В-третьих, узконаправленность такой программы делает неудобным её использование на крупных предприятиях, так как удобнее, когда все основные этапы производства управляются единой программой, а не множеством мелких.

Из программ, которые являются мощным инструментом управления предприятием, можно отметить АСУ «Техподготовка производства» (далее – АСУ «ТП»)[1]. Данная программа является неплохим помощником в управлении производством на крупных предприятиях, хотя в функциональности и уступает программе «1С: Предприятие». АСУ «ТП» позволяет работать со спецификациями деталей и сборочных единиц, в которых отражаются данные о заготовках, нормах расхода материалов, маршрутах изготовления и многое другое. Работа со спецификациями в данной программе ведётся в разделе «Основное производство» (рисунок 1.13).

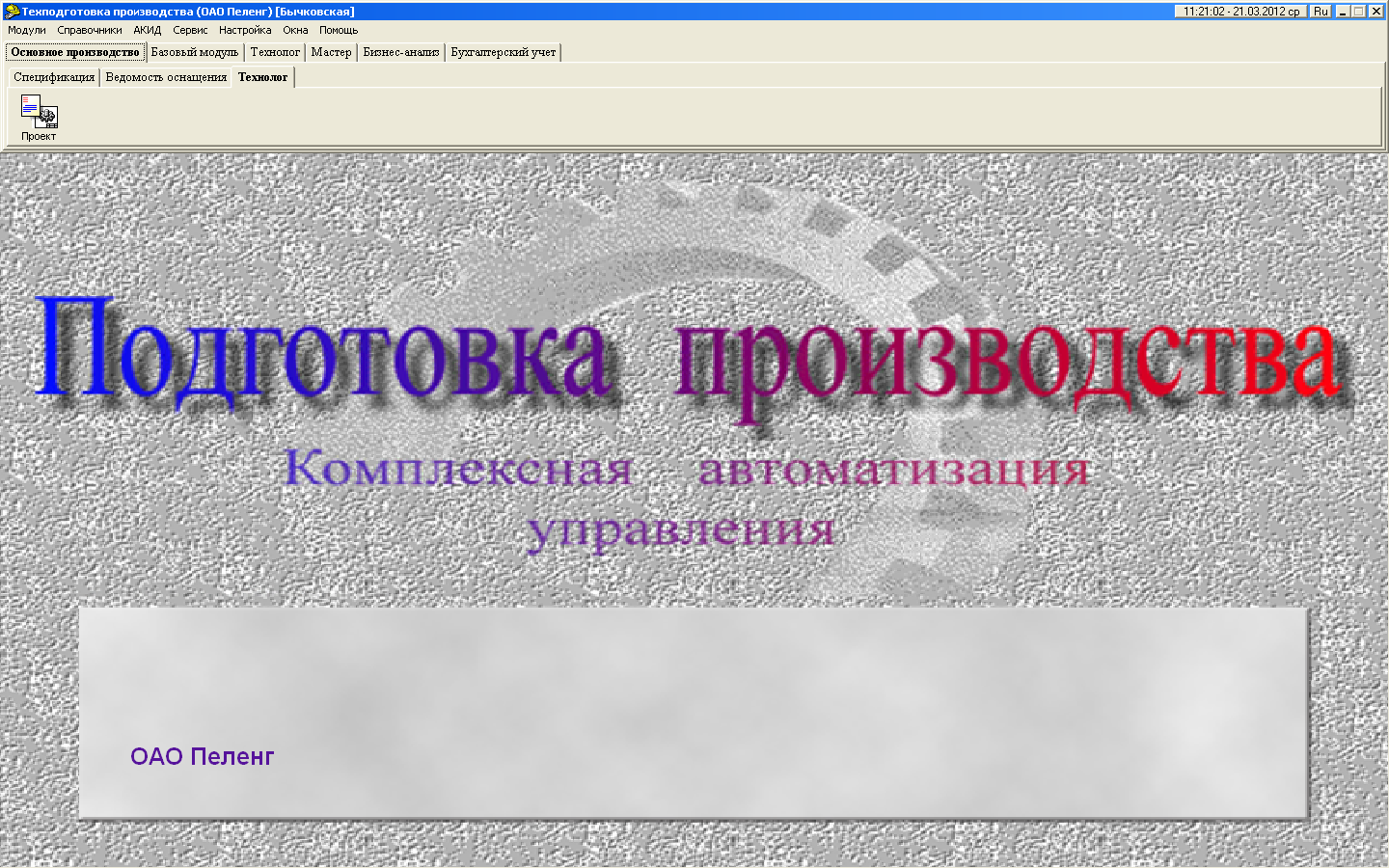


Рисунок 1.13 – Раздел главного меню «Основное производство» программы АСУ «ТП»

При создании спецификации указывается её номер и наименование (рисунок 1.14).



Рисунок 1.14 – Создание новой спецификации в программе АСУ «ТП»

После создания спецификации в появившееся окно вводят данные из конструкторской документации, а также указывают сведения об используемых материалах, размерах заготовок и нормах расхода (рисунок 1.15). В разделе «Спецификация изделия» можно получить всю информацию о выбранной спецификации (рисунок 1.16).

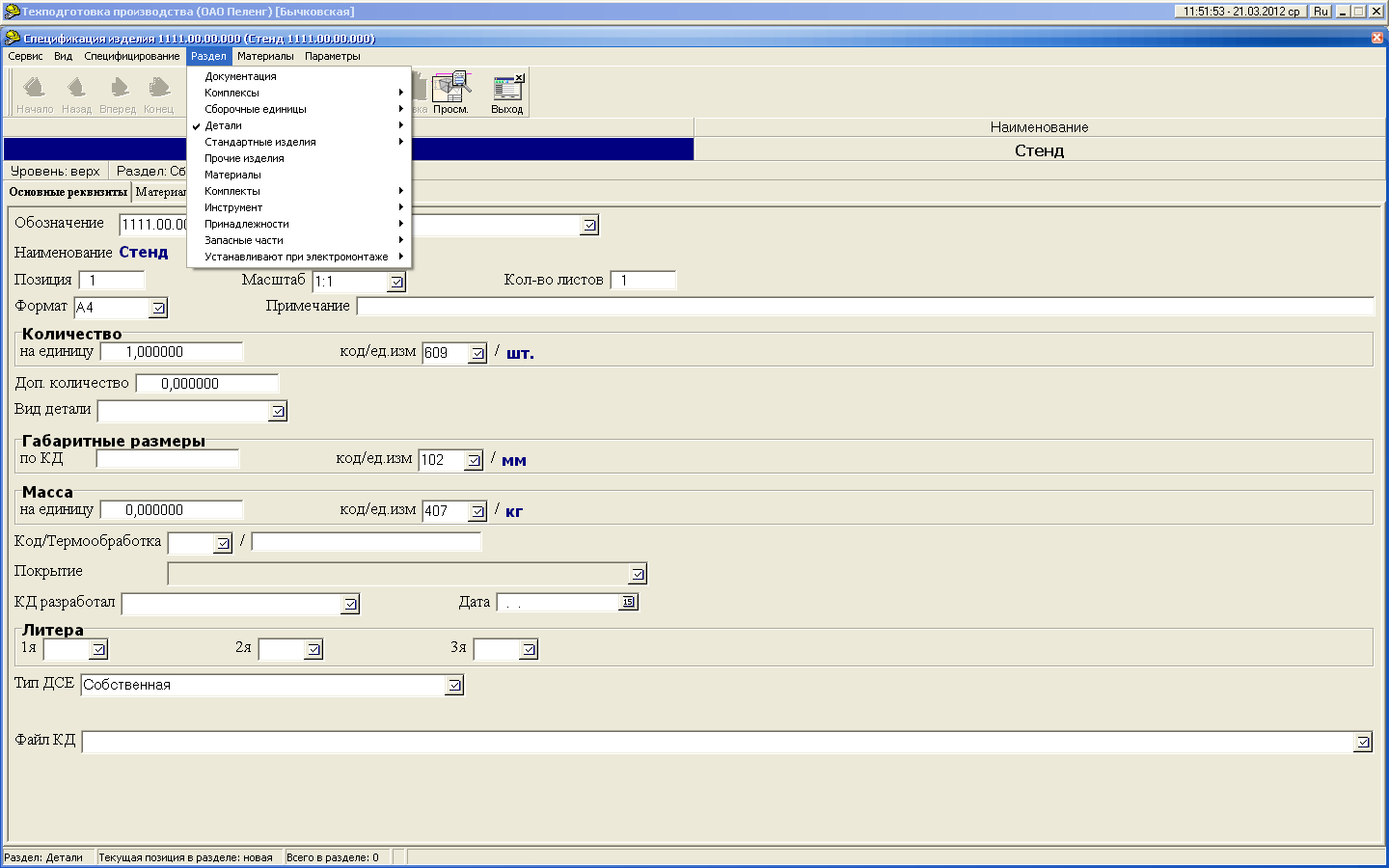


Рисунок 1.15 – Пример заполнения спецификации в АСУ «ТП»



Рисунок 1.16 – Раздел «Спецификация изделия» в АСУ «ТП»

АСУ «ТП» как и «1С» поддерживает формирование отчётов и вывод их на печать (рисунок 1.17). Из недостатков можно отметить то, что для вывода на печать используется приложение Microsoft Word 2003. На данный момент эта версия приложения Microsoft Word используется редко, так как не поддерживает новый формат «docx». Другие же версии приложения Microsoft Word АСУ «ТП» просто не поддерживает.

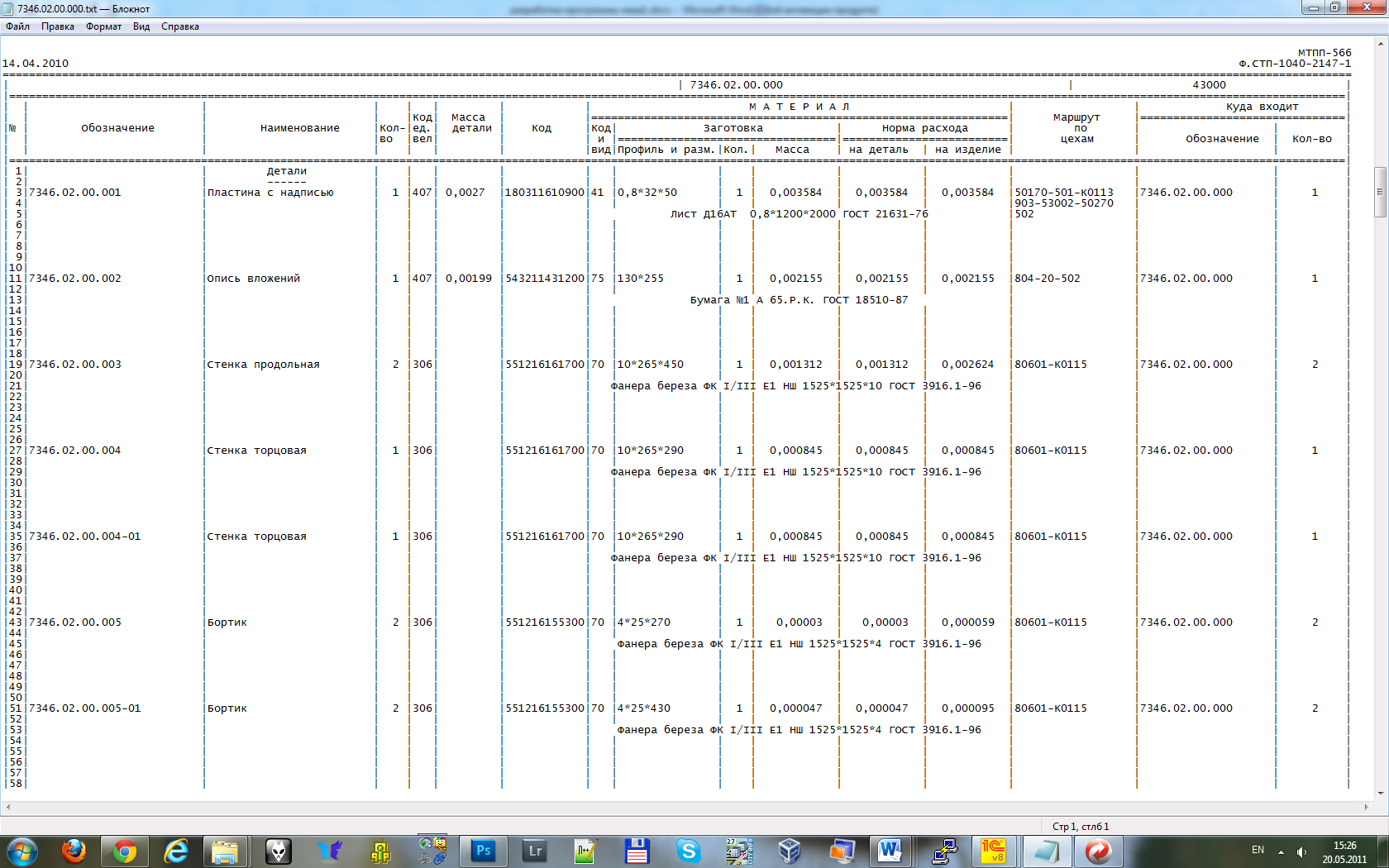


Рисунок 1.17 – Вывод ведомости материалов на печать в АСУ «Техподготовка производства»

Существенным недостатком АСУ «ТП» при работе со спецификациями является невозможность расчёта нормы расхода материалов и размеров заготовок. Технологу приходится подбирать заготовку самому, после чего вручную рассчитывать норму расхода материала, используя различные справочники.

* + 1. Определение требований к системе

В результате анализа аналогов программ по управлению предприятием можно определить необходимые требования к нашей программе:

* наличие единой базы данных, содержащей:
  + полную номенклатуру изделий предприятия со всей информацией по заготовкам, материалам, маршрутам и т.п.;
  + технологические процессы изготовления деталей и сборок;
  + информацию о планировании изготовления и сроках;
  + поэтапную информацию об изготовлении деталей и сборочных единиц.
* разграничение прав доступа к программе и базе данных на уровне пользователей системы;
* возможность вывода сформированных отчётов на печать;
* возможность вывода технологических процессов в документ Microsoft Word (версии 2007, 2010, 2013);
* возможность вывода отчётов, имеющих вид списка, в файл Microsoft Excel (версии 2007, 2010, 2013);
* возможность быстрого вызова калькулятора Windows.
  + 1. Постановка задачи

Основными задачами, которые необходимо решить с помощью приложения, являются:

* ввод новых спецификаций, корректировка и удаление существующих;
* расчет заготовок и норм расхода материалов;
* разработка технологических процессов на детали и сборочные единицы;
* хранение всех данных в единой базе данных;
* отслеживание передвижения деталей и сборочных единиц в производстве по маршруту изготовления и пооперационно;
* работа с плановыми заданиями и техпаспортами;
* формирование отчётов: ведомости материалов, ведомости входимости применяемости, ведомости расцеховок и др.;
* вывод на печать отчётов и технологических процессов;
* вывод технологических процессов в документ Microsoft Word (версии 2007, 2010, 2013);
* вывод отчётов, имеющих вид списка, в файл Microsoft Excel (версии 2007, 2010, 2013);
* разграничение прав доступа к программе и базе данных на уровне пользователей системы;
* предоставление различных интерфейсов взаимодействия с программой разным группам пользователей.

1. Методы и модели, положенные в основу проекта

В современных программах при работе с крупными изделиями, состоящими из большого количества деталей и сборочных единиц, принята древовидная структура записи. Спецификация на каждую деталь или сборочную единицу содержит сведения о входящих в неё деталях и сборочных единицах. Например, в сборку 6122.10.07.500 входят 3 сборки и 1 деталь (рисунки 2.1 и 2.2).

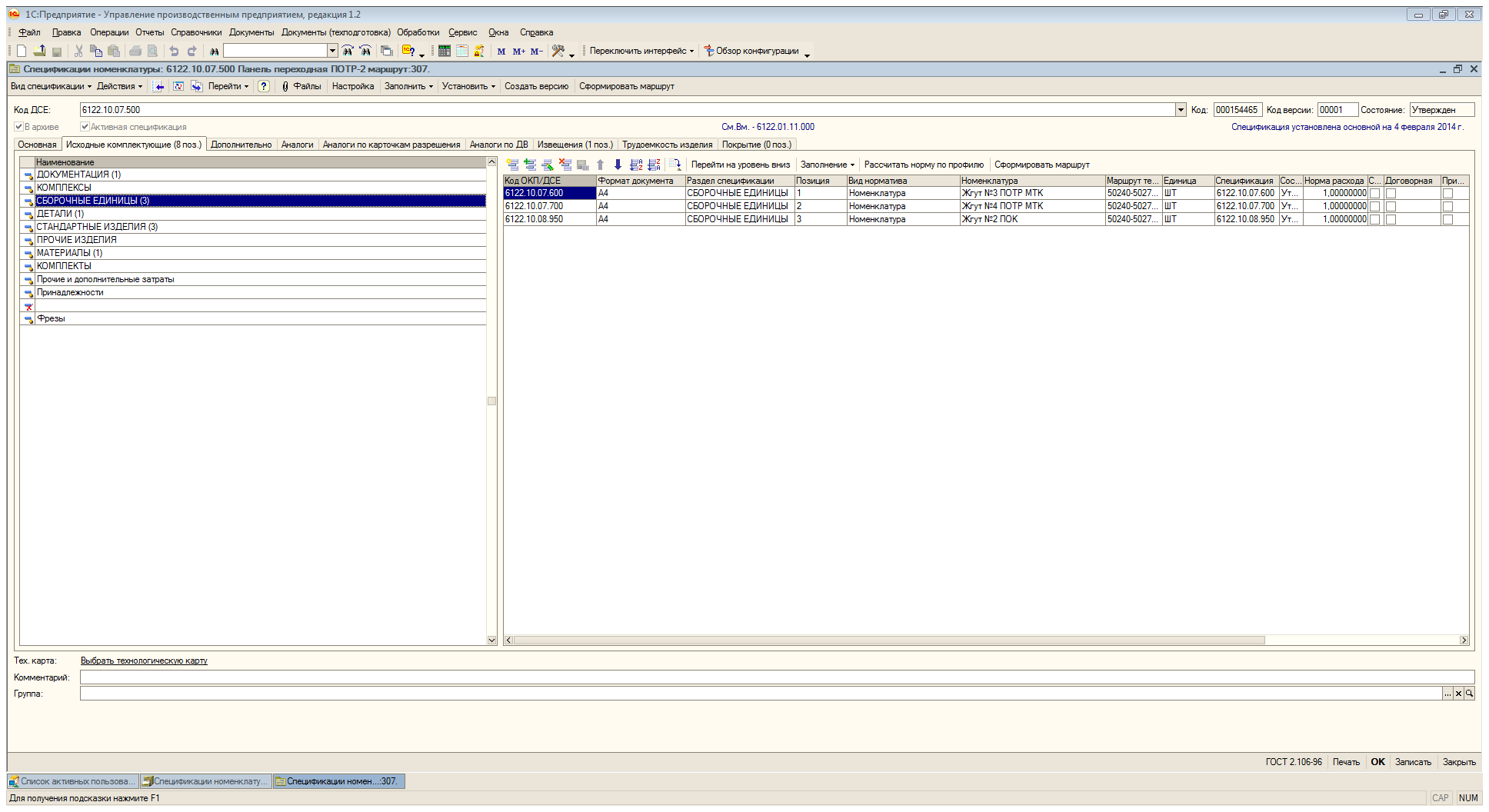


Рисунок 2.1 – Просмотр сборок, входящих в спецификацию 6122.10.07.500, в программе «1С»

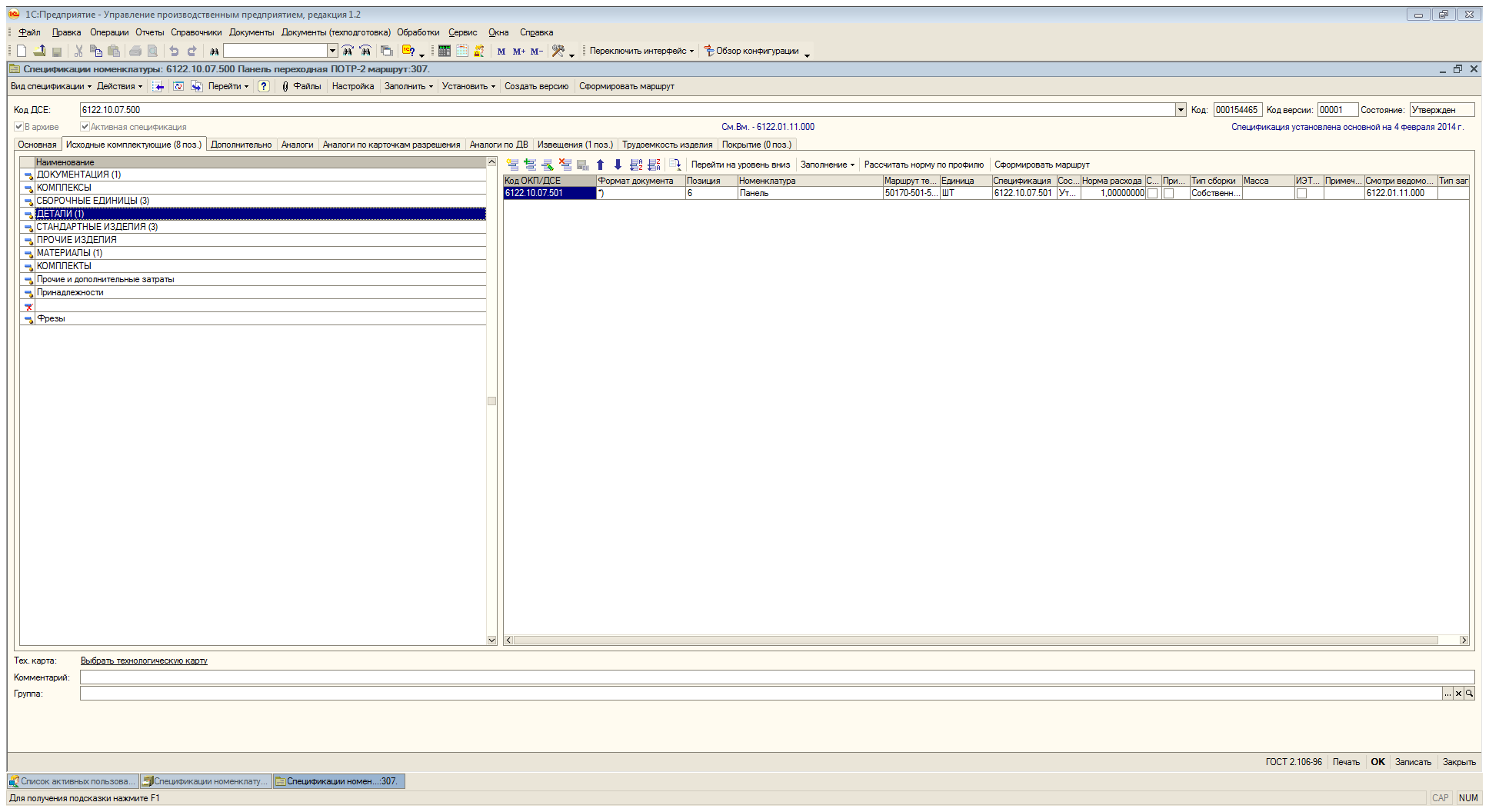


Рисунок 2.2 – Просмотр деталей, входящих в спецификацию 6122.10.07.500, в программе «1С»

Каждая спецификация, являющаяся родителем по отношению к входящим в неё спецификациям, не может в своих дочерних узлах содержать саму себя. Например, не может сборка 6122.10.07.500 входить в сборку 6122.10.07.600, так как сборка 6122.10.07.600 сама входит в сборку 6122.10.07.500 (рисунок 2.3). Такая древовидная структура спецификаций позволяет исключить зацикленность, которую сложно выявить при огромной номенклатуре изделий, когда одни спецификации могут заимствоваться в другие.

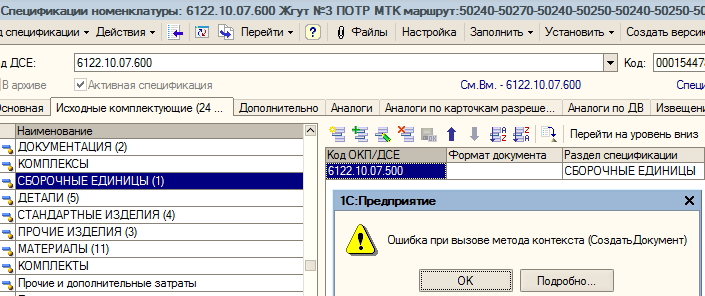


Рисунок 2.3 – Попытка вставить сборку 6122.10.07.500 в сборку, входящую в сборку 6122.10.07.500, в программе «1С»

При работе со спецификациями технологу приходится сталкиваться с выбором оптимальных заготовок для изготовления детали, учитывая при этом норму расхода материала. Чаще всего технолог сам рассчитывает заготовку, исходя из специфики имеющегося оборудования на предприятии. Программа же может посчитать норму расхода материала по указанной заготовке. При этом припуски под обработку и получение заготовки, влияющие на окончательную норму расхода, также указывает технолог.

В основу данного проекта при работе со спецификациями заложен принцип максимальной автоматизации расчёта заготовок и норм расхода. Технолог указывает программе профиль сечения детали, её габаритные размеры и необходимое количество, а программа уже предлагает различные варианты заготовок в зависимости от профиля материала и габаритного размера. Также при расчёте нормы расхода программа учитывает припуски под обработку материала для получения нужной заготовки на имеющемся у предприятия оборудовании.

Так как норма расхода материала зависит от вида заготовки (единичная или групповая), габарита материала, наличия или отсутствия хвостовика (технологического зажима), то и формулы для её расчёта будут разными. При расширенной номенклатуре материала одну и ту же деталь можно выполнить из различных заготовок. Отличаться же эти заготовки будут размерами, нормой расхода материала и трудоёмкостью изготовления. В нашем проекте будут учитываться только размеры заготовки и норма расхода материала. Заготовка с наименьшей нормой расхода материала будет считаться наилучшей из возможных. Так как трудоёмкость изготовления детали из данной заготовки не учитывается, то технолог сам определяет какую из возможных заготовок использовать.

Размер заготовки определяется в зависимости от габаритов детали и припуска под обработку. Если деталь в сечении имеет форму окружности, то заготовка может использоваться как из прутка/круга, так и из листа/плиты. Несомненно, если заготовка вырезана из листа или плиты, то трудоёмкость изготовления такой детали будет большой. Тем более не каждую прямоугольную заготовку можно установить на тот же токарный станок. Заготовка из листа/плиты состоит из трёх размеров txbxh, где t – толщина листа/плиты (в мм), b – ширина заготовки (в мм), а h – длина (в мм). Если деталь имеет размеры Ødxl, то из листа/плиты можно получить следующие заготовки (формулы 1-3):

txbxh = (d+pt)x(d+p)x(l+p), (1)

txbxh = (d+pt)x(l+p)x(d+p), (2)

txbxh = (l+pt)x(d+p)x(d+p), (3)

где d – диаметр детали, мм;

pt – припуск на механическую обработку по толщине, мм;

p – припуск на механическую обработку по ширине и длине, мм;

l – длина детали, мм.

Припуски на механическую обработку зависят от толщины материала и определены в соответствии с принятым стандартом предприятия.

При расчёте заготовок из листа/плиты на деталь с размерами axbxc, где a, b, c – габаритные размеры детали (в мм), используются формулы (4-6):

txbxh = (a+pt)x(b+p)x(c+p), (4)

txbxh = (b+pt)x(a+p)x(c+p), (5)

txbxh = (c+pt)x(a+p)x(b+p). (6)

При этом следует учитывать, что материала с расчётной толщиной может не оказаться на складе, а, следовательно, стоит использовать тот материал, толщина которого является наименьшей, но в то же время больше величины t. Если такого материала не окажется на складе, значит такую заготовку использовать нельзя. Так как каждая плита (лист) имеет ограниченные размеры по ширине и длине, то дополнительно осуществляется проверка на то, чтобы расчётные размеры заготовок не превышали эти значения.

При расчёте нормы расхода материала на заготовку размером txbxh из листа/плиты применяется следующая формула:

N = t(b+ppl)(h+ppl)ρ/106, (7)

где N – норма расхода материала, кг;

ppl – припуск на отрезку заготовки от плиты/листа, мм;

ρ – плотность материала, кг/м3. [2]

Припуск на отрезку заготовки от плиты/листа ppl также зависит от толщины обрабатываемого материала и определён стандартом предприятия.

Заготовка из прутка/круга состоит из двух размеров Ødzxlz, где dz – диаметр прутка/круга (в мм), а lz – длина (в мм). Если деталь имеет размеры Ødxl, то из прутка/круга можно получить единичную (формула 8) или групповую заготовку (формула 9):

Ødzxlz = Ø(d+p)x(l+p), (8)

Ødzxlz = Ø(d+p)x((l+p+pcut)\*n)+ls, (9)

где d – диаметр детали, мм;

p – припуск на механическую обработку торцев детали, мм;

l – длина детали, мм;

pcut – припуск на отрезку детали от групповой заготовки резцом, мм;

n – выход деталей из групповой заготовки, шт.;

ls – длина хвостовика (технологического зажима), мм.

В случае, если материала с расчётным диаметром нет на складе, то выбирается материал, диаметр которого является следующим по возрастанию. При отсутствии такого материала заготовка считается неопределённой, а, следовательно, никаких результатов расчёта не выводится.

Если же деталь в сечении не имеет форму окружности, то для расчёта заготовок из прутка/круга необходимо сначала вычислить описанную окружность, по которой будет определяться габарит заготовки. Так как деталь имеет 3 разные стороны, то и заготовки будут разными. Для начала рассчитаем описанные окружности для детали со сторонами a, b, c (в мм) (формулы 10-12):

dC = , (10)

dB = , (11)

dA = , (12)

где dC – описанная окружность для сторон a и b;

dB – описанная окружность для сторон a и c;

dA – описанная окружность для сторон c и b.

Теперь, подставив полученные значения в формулы (8-9), мы получим следующие формулы для расчёта единичной заготовки (формулы 13-15) и для групповой (формулы 16-18):

Ødzxlz = Ø(dC+p)x(c+p), (13)

Ødzxlz = Ø(dB+p)x(b+p), (14)

Ødzxlz = Ø(dA+p)x(a+p), (15)

Ødzxlz = Ø(dC+p)x((c+p+pcut)\*n)+ls, (16)

Ødzxlz = Ø(dB+p)x((b+p+pcut)\*n)+ls, (17)

Ødzxlz = Ø(dA+p)x((a+p+pcut)\*n)+ls. (18)

Так как норма расхода материала рассчитывается на одну деталь, то для единичной и групповой заготовок из прутка/круга формулы будут различными (формулы 19-20):

N1 = , (19)

Ng = , (20)

где N1 – норма расхода на единичную заготовку, кг;

dz – диаметр заготовки, мм;

lz – длина заготовки, мм;

pd – припуск на отрезку заготовки, мм;

ρ – плотность материала, кг/м3;

Ng – норма расхода на групповую заготовку, кг;

n – выход деталей из групповой заготовки, шт.

В данном проекте заложена идея расчёта различных возможных вариантов заготовок по габаритам детали. Все возможные варианты будут отображаться технологу, а он уже будет определять, какую из заготовок использовать.

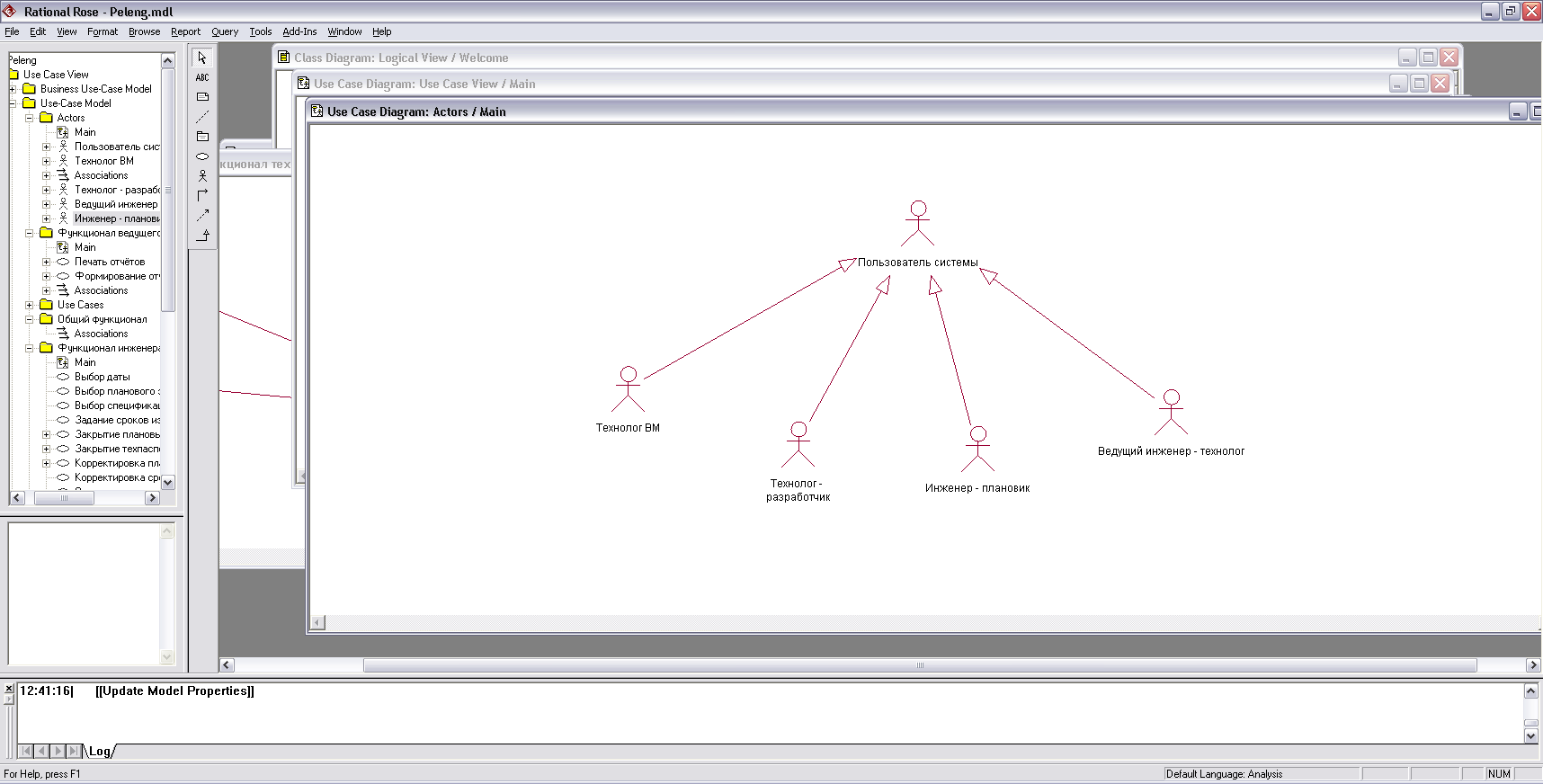
1. Разработка проекта программного обеспечения
2. Функциональный анализ предметной области и разработка спецификаций требований

Для описания моделей представления системы будем использовать построение диаграмм вариантов использования (use case) системы и трассировочные матрицы (таблицы).

Диаграмма вариантов использования (use case diagram) — это диаграмма, отражающая отношения между актёрами (действующими лицами) и вариантами использования системы (use case). Вариант использования служит для описания сервисов, которые система предоставляет актёру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актёром. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актёров с системой.

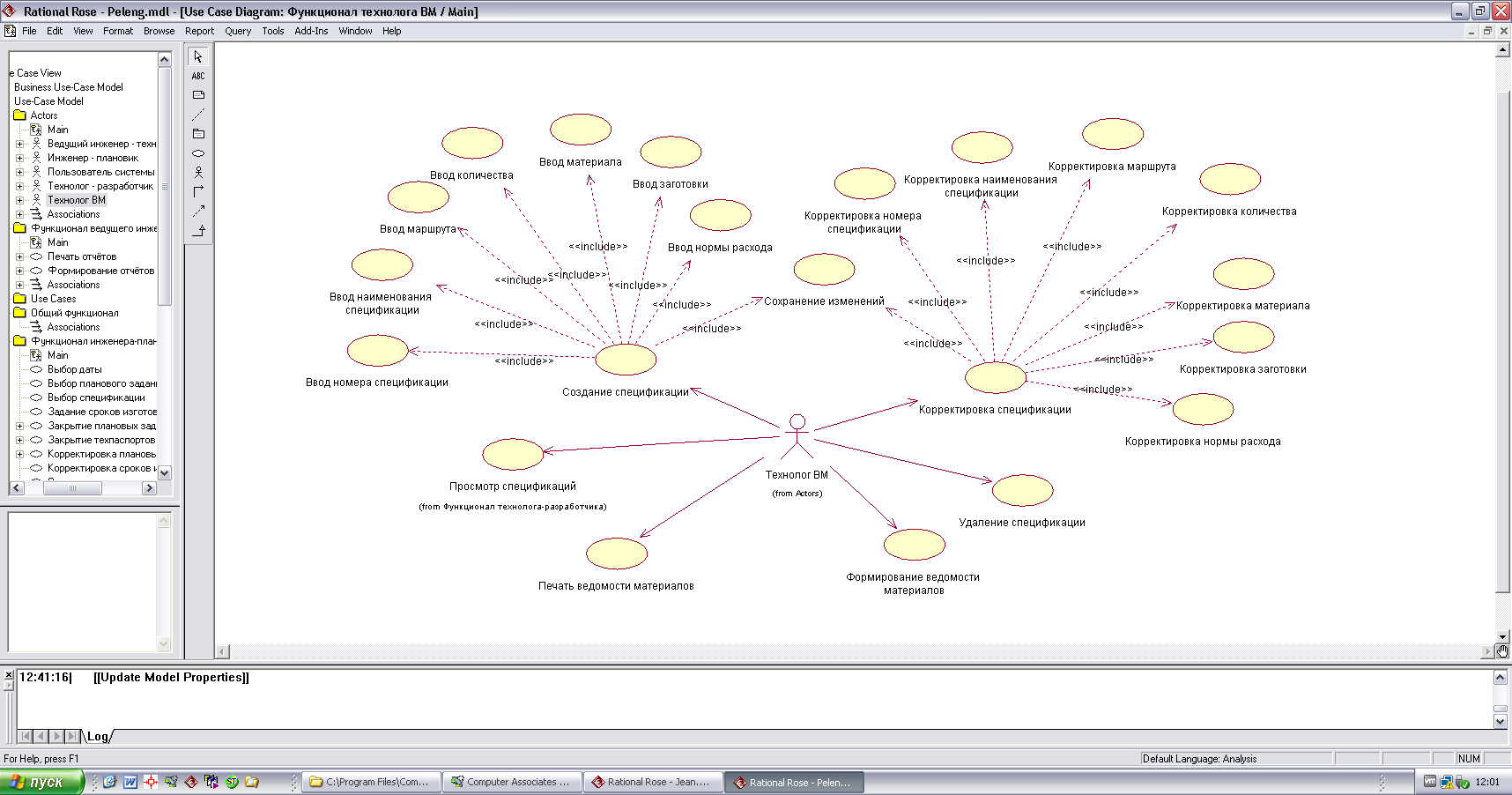
В зависимости от того, к какой группе относится пользователь, система будет предоставлять ему определенный интерфейс и набор функций. Определение группы пользователя происходит на этапе авторизации, когда пользователь вводит свои логин и пароль.

На рисунке 3.1 представлены группы пользователей, которые могут пользоваться данной системой.

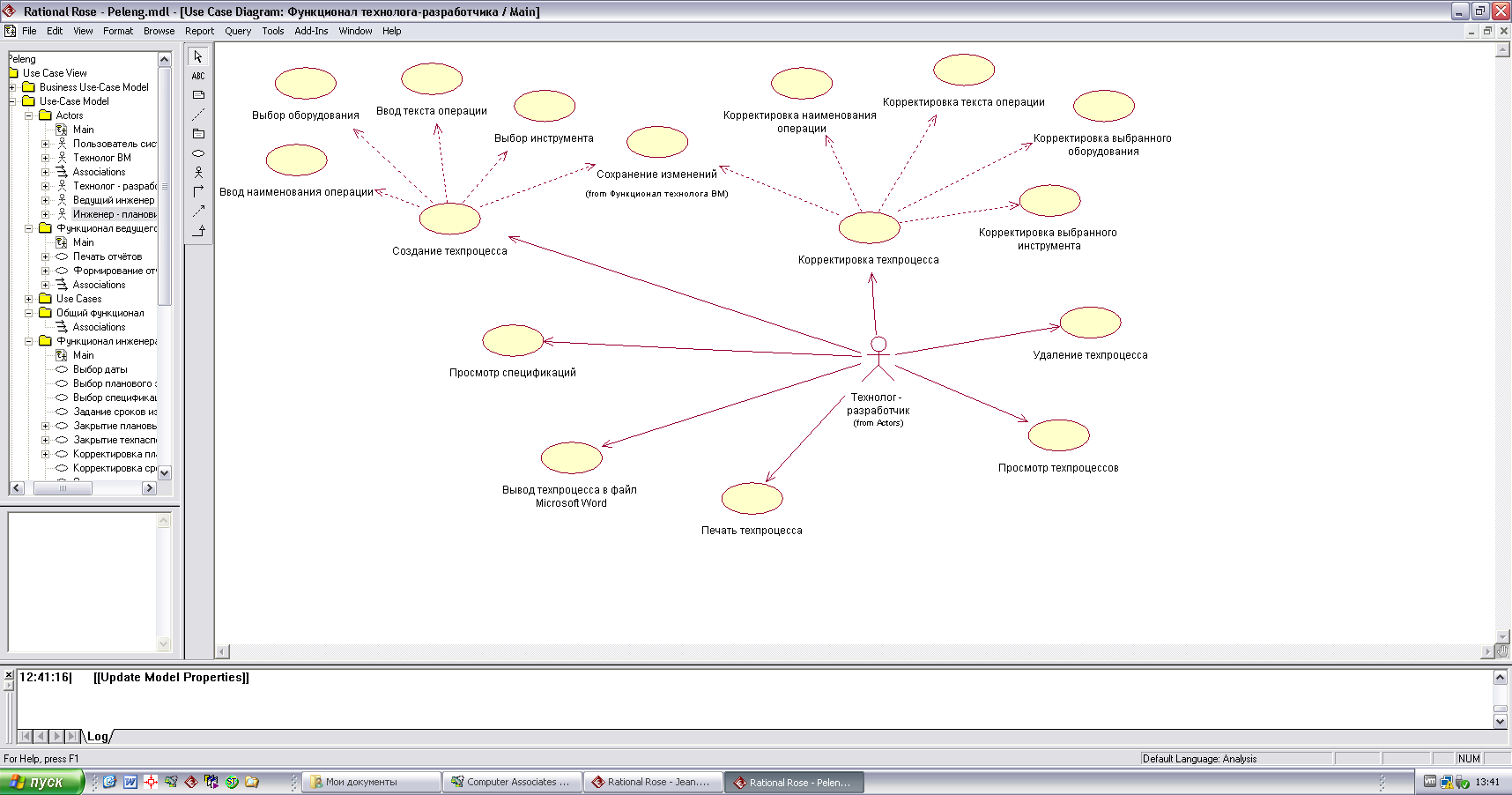
Рисунок 3.1 – Пользователи системы

После запуска программы появляется диалоговое окно авторизации, запрашивающее логин и пароль. Для каждой группы пользователей система предоставляет свой интерфейс с теми функциями, которые должны быть доступны этой группе.

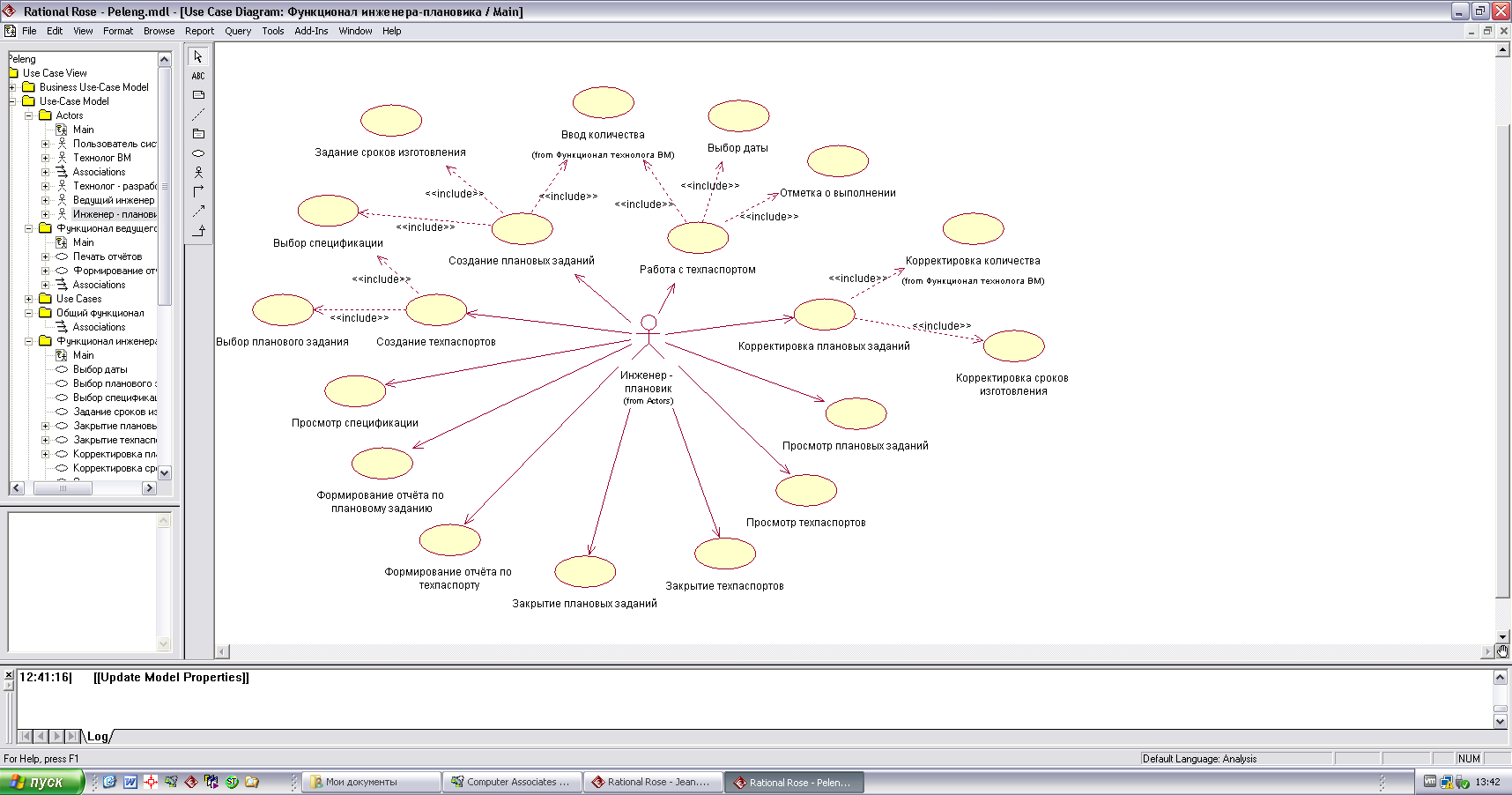
Если пользователь относится к группе технологов ВМ, то ему будут доступны функции по работе со спецификациями (рисунок 3.2). Пользователь может создать новую спецификацию, откорректировать или просмотреть уже имеющуюся или вовсе удалить. При создании спецификации пользователь вводит номер спецификации, наименование, маршрут и количество, а также материал, заготовку и норму расхода материала. Для сохранения введённых значений пользователь нажимает кнопку «Сохранить». При корректировке спецификации пользователь может изменить номер спецификации, наименование, маршрут, количество, материал и норму расхода. Для указания нормы расхода материала пользователь может воспользоваться встроенной программой, нажав на кнопку «Рассчитать», либо ввести значение вручную. При расчёте нормы расходы программой пользователь указывает габариты детали, к какой группе относится материал, наличие шлифованных поверхностей, определяющих габариты, и количество термических обработок, если не был указан маршрут. Рассчитанное значение нормы расхода материала программа записывает в соответствующее поле спецификации. Все изменения сохраняются при нажатии на кнопку «Сохранить». Пользователь также может сформировать ведомость материалов, указав номер нужной спецификации. К тому же ведомость материалов можно вывести на печать.

Рисунок 3.2 – Диаграмма вариантов использования технолога ВМ

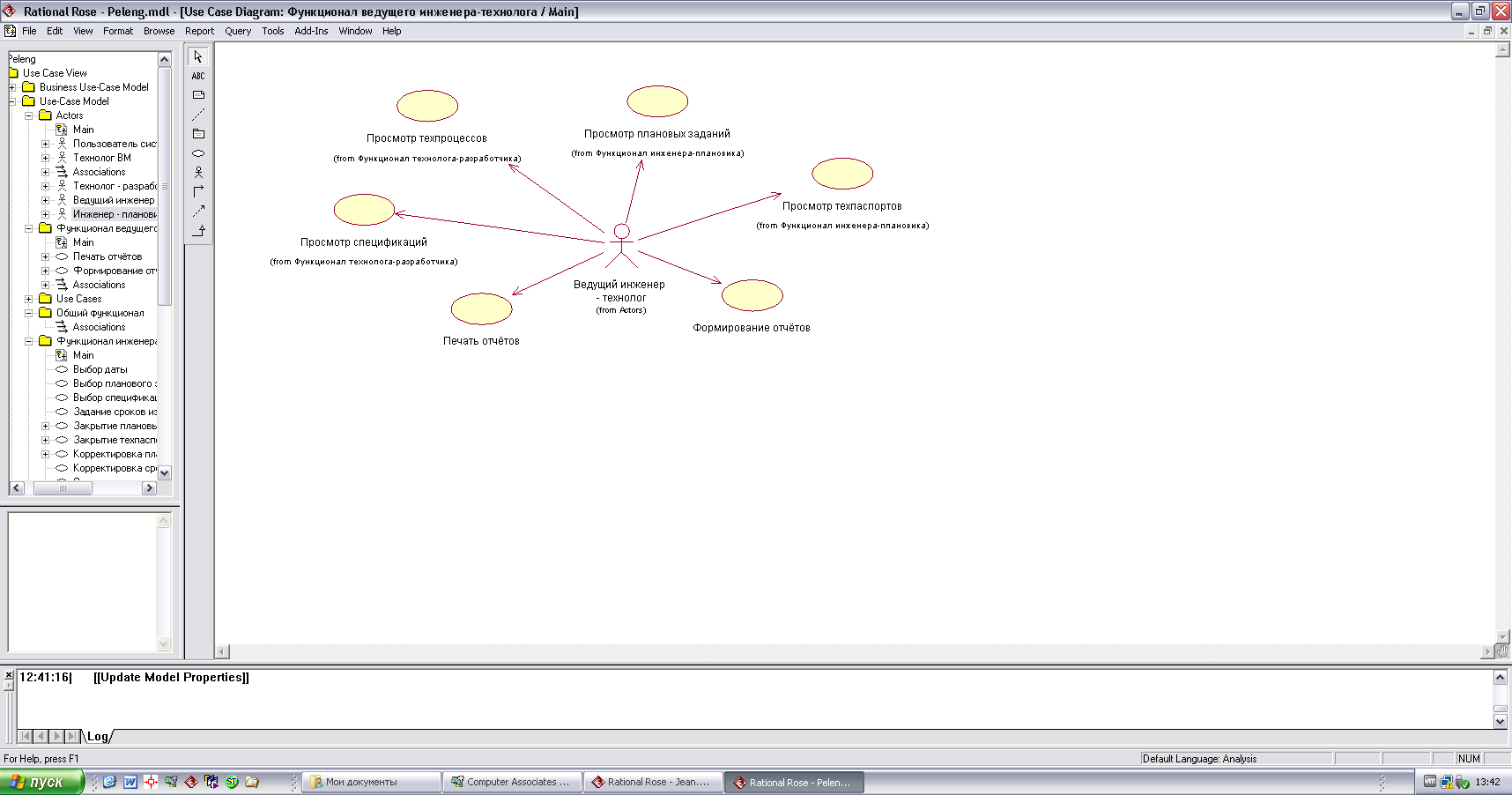
Если пользователь относится к группе технологов-разработчиков, то ему будут доступны функции по работе с техпроцессами (рисунок 3.3). Пользователь может создавать или корректировать техпроцесс, выбрав спецификацию. В техпроцессе пользователь может выбрать наименование операции, используемое оборудование и инструмент, а также вводить текст операции. При корректировке техпроцесса ему доступны функции изменения наименования операции, текста операции, оборудования и инструмента. Сохранение изменений происходит при нажатии на кнопку «Сохранить». Пользователю также доступны функции просмотра спецификаций, удаления техпроцесса, открытие техпроцесса в программе Microsoft Word и печать техпроцесса.

Рисунок 3.3 – Диаграмма вариантов использования технолога-разработчика

Если пользователь относится к группе инженеров-плановиков, то ему доступны функции управления плановыми заданиями (рисунок 3.4). Он может создавать плановые задания, указав номер спецификации, количество и сроки изготовления. При необходимости пользователь может откорректировать созданное плановое задание, изменив количество и сроки изготовления. Каждая позиция (деталь), изготавливаемая по плановому заданию, сопровождается техпаспортом, содержащем сведения об этапах изготовления и количестве. Для создания техпаспорта пользователь выбирает номер планового задания и спецификацию. Если разработан техпроцесс на данную позицию (деталь), то в техпаспорте отражаются названия операций. Плановик при работе с техпаспортом делает отметки о выполнении конкретной операции, указывая дату выполнения и количество. Все изменения при работе с плановыми заданиями и техпаспортами сохраняются после нажатия кнопки «Сохранить». Плановик может просматривать созданные плановые задания, техпаспорта и спецификации. На любом этапе изготовления позиций плановик может сформировать отчёт по плановому заданию, в котором отражаются сведения о стадии готовности каждой позиции, либо отчёт по техпаспорту на конкретную позицию. Когда позиция изготовлена плановик закрывает техпаспорт. Если все техпаспорта, относящиеся к плановому заданию, закрыты, то плановик закрывает и плановое задание.

Рисунок 3.4 – Диаграмма вариантов использования инженера-плановика

Если пользователь относится к группе ведущих инженеров-технологов, то он может просматривать спецификации, техпроцессы, плановые задания и техпаспорта (рисунок 3.5). При этом ему доступны функции формирования отчётов и вывод их на печать.

Рисунок 3.5 – Диаграмма вариантов использования ведущего инженера-технолога

Для отслеживания отношений между требованиями к системе будем использовать трассировочную матрицу. В данной матрице отображаются взаимосвязи между компонентами системы и требованиями к системе. Ниже представлены трассировочные матрицы для разных групп пользователей системы (таблицы 3.1 – 3.4).

Таблица 3.1 – Трассировочная матрица технолога ВМ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Требование заказчика | Функция системы | Use-Case |
| 1 | Пользователь может создать новую спецификацию, откорректировать или просмотреть уже имеющуюся или вовсе удалить | Создание спецификации | Создание спецификации |
| Открытие спецификации для просмотра и корректировки | 1. Корректировка спецификации  2. Просмотр спецификаций |
| Удаление спецификации | Удаление спецификации |
| 2 | При создании спецификации пользователь вводит номер спецификации, наименование, маршрут и количество, а также материал, заготовку и норму расхода материала | Ввести номер спецификации | Ввод номера спецификации |
| Ввести наименование | Ввод наименования спецификации |
| Ввести маршрут | Ввод маршрута |
| Ввести количество | Ввод количества |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ввести материал | Ввод материала |
| Ввести заготовку | Ввод заготовки |
| Ввести норму расхода | Ввод нормы расхода |
| 3 | Для сохранения введённых значений пользователь нажимает кнопку «Сохранить» | Сохранить изменения в базе данных | Сохранение изменений |
| 4 | При корректировке спецификации пользователь может изменить номер спецификации, наименование, маршрут, количество, материал, заготовку и норму расхода | Изменить номер спецификации | Корректировка номера спецификации |
| Изменить наименование спецификации | Корректировка наименования спецификации |
| Изменить маршрут | Корректировка маршрута |
| Изменить количество | Корректировка количества |
| Изменить материал | Корректировка материала |
| Изменить заготовку | Корректировка заготовки |
| Изменить норму расхода | Корректировка нормы расхода |
| 5 | Пользователь может воспользоваться встроенной программой, нажав на кнопку «Рассчитать». При расчёте нормы расходы программой пользователь указывает габариты детали, к какой группе относится материал, наличие шлифованных поверхностей и количество термических обработок. Рассчитанное значение нормы расхода материала программа записывает в соответствующее поле спецификации. | Запуск программы расчёта нормы расхода | Расчёт нормы расхода |
| Ввести габариты | Ввод габаритов |
| Ввести группу материала | Выбор группы материалов |
| Указать наличие шлифованных поверхностей | Указание о наличии шлифовки |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ввести количество термообработок | Ввод количества термообработок |
| Запись результата в поле нормы расхода | Ввод нормы расхода |
| 6 | Пользователь также может сформировать ведомость материалов, указав номер нужной спецификации. | Сформировать ведомость материалов | Формирование ведомости материалов |
| 7 | К тому же ведомость материалов можно вывести на печать. | Послать ведомость материалов на печать | Печать ведомости материалов |

Таблица 3.2 – Трассировочная матрица технолога-разработчика

|  |  |  |  |
| --- | --- | --- | --- |
| № | Требование заказчика | Функция системы | Use-Case |
| 1 | Пользователь может создавать или корректировать техпроцесс. Пользователю также доступны функции просмотра спецификаций, удаления техпроцесса | Создание техпроцесса | Создание техпроцесса |
| Открытие техпроцесса для просмотра и корректировки | 1. Корректировка техпроцесса  2. Просмотр техпроцессов |
| Удаление техпроцесса | Удаление техпроцесса |
| Открытие спецификации для просмотра | Просмотр спецификаций |
| 2 | В техпроцессе пользователь может выбрать наименование операции, используемое оборудование и инструмент, а также вводить текст операции | Ввести наименование операции | Ввод наименования операции |
| Выбрать оборудование | Выбор оборудования |
| Выбрать инструмент | Выбор инструмента |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ввести текст операции | Ввод текста операции |
| 3 | При корректировке техпроцесса ему доступны функции изменения наименования операции, текста операции, оборудования и инструмента | Изменить наименование операции | Корректировка наименования операции |
| Изменить текст операции | Корректировка текста операции |
| Перевыбрать оборудование | Корректировка выбранного оборудования |
| Перевыбрать инструмент | Корректировка выбранного инструмента |
| 4 | Сохранение изменений происходит при нажатии на кнопку «Сохранить» | Сохранить изменения в базе данных | Сохранение изменений |
| 5 | Пользователю также доступны функции … открытие техпроцесса в программе Microsoft Word и печать техпроцесса | 1. Запустить программу Microsoft Word  2. Отобразить техпроцесс в новом окне Microsoft Word | Вывод техпроцесса в файл Microsoft Word |
| Послать техпроцесс на печать | Печать техпроцесса |

Таблица 3.3 – Трассировочная матрица инженера-плановика

|  |  |  |  |
| --- | --- | --- | --- |
| № | Требование заказчика | Функция системы | Use-Case |
| 1 | Пользователь может создавать плановые задания, указав номер спецификации, количество и сроки изготовления | Создание планового задания | Создание планового задания |
| Выбрать спецификацию | Выбор спецификации |
| Указать срок изготовления | Задание сроков изготовления |

Продолжение таблицы 3.3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Ввести количество | Ввод количества |
| 2 | Пользователь может откорректировать созданное плановое задание, изменив количество и сроки изготовления | Изменение планового задания | Корректировка планового задания |
| Изменить срок изготовления | Корректировка сроков изготовления |
| Изменить количество | Корректировка количества |
| 3 | Для создания техпаспорта пользователь выбирает номер планового задания и спецификацию | Создание техпаспорта | Создание техпаспорта |
| Выбрать плановое задание из имеющихся | Выбор планового задания |
| Выбрать спецификацию | Выбор спецификации |
| 4 | Плановик при работе с техпаспортом делает отметки о выполнении конкретной операции, указывая дату выполнения и количество | Выполнение действий с техпаспортом | Работа с техпаспортом |
| Ввести количество изготовленных деталей | Ввод количества |
| Указать дату изготовления | Выбор даты |
| Поставить отметку о выполнении операции | Отметка о выполнении |
| 5 | Все изменения при работе с плановыми заданиями и техпаспортами сохраняются при нажатии кнопки «Сохранить» | Сохранить изменения в базе данных | Сохранение изменений |

Продолжение таблицы 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Плановик может просматривать созданные плановые задания, техпаспорта и спецификации | Открыть плановое задание для просмотра | Просмотр плановых заданий |
| Открыть техпаспорт для просмотра | Просмотр техпаспортов |
| Открыть спецификацию для просмотра | Просмотр спецификации |
| 7 | Плановик может сформировать отчёт по плановому заданию либо отчёт по техпаспорту на конкретную позицию. | Сформировать отчёт по плановому заданию | Формирование отчёта по плановому заданию |
| Сформировать отчёт по техпаспорту | Формирование отчёта по техпаспорту |
| 8 | Когда позиция изготовлена плановик закрывает техпаспорт. Если все техпаспорта, относящиеся к плановому заданию, закрыты, то плановик закрывает и плановое задание. | Изменить состояние техпаспорта на «закрыт» | Закрытие техпаспортов |
| Изменить состояние планового задания на «закрыто» | Закрытие плановых заданий |

Таблица 3.4 – Трассировочная матрица ведущего инженера-технолога

|  |  |  |  |
| --- | --- | --- | --- |
| № | Требование заказчика | Функция системы | Use-Case |
| 1 | Пользователь может просматривать спецификации, техпроцессы, плановые задания и техпаспорта. | Открытие спецификации для просмотра | Просмотр спецификаций |
| Открытие техпроцесса для просмотра | Просмотр техпроцессов |

Продолжение таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Открытие планового задания для просмотра | Просмотр плановых заданий |
| Открытие техпаспорта для просмотра | Просмотр техпаспортов |
| 2 | Пользователю доступны функции формирования отчётов и вывод их на печать. | 1. Выбрать вид отчёта  2. Сформировать отчёт | Формирование отчётов |
| Послать отчёт на печать | Печать отчётов |

1. Разработка структуры объекта в виде схемы ресурсов

Программа представляет собой клиентской приложение, написанное на языке С# и работающее с базой данных, созданной в MS SQL Server 2008. Использование технологии ADO.NET позволяет обращаться к данным на любом языке программирования. ADO.NET предоставляет доступ к таким источникам данных, как SQL Server, а также к источникам данных, предоставляемым посредством ODBC. Пользовательские приложения, использующие общие данные, могут использовать ADO.NET для соединения с этими источниками данных и для получения, обработки и обновления имеющихся в них данных.

Интерфейс взаимодействия с базами данных ODBC позволяет нашему приложению общаться с различными серверами баз данных. Этот интерфейс поддерживает запросы на языке SQL и определяет набор функций, которые можно использовать для доступа к любой реляционной СУБД. Менеджер драйверов ODBC взаимодействует с драйверами SQL Server 2008 и MS Excel, позволяя получать данные из базы данных и выводить их в файл Excel-формата.

Схема ресурсов системы, которая описывает структуру разрабатываемого объекта и функции его составляющих, представлена на РТДП 3032.020.001.

1. Разработка схемы работы системы

Основными операциями системы являются работа со спецификациями и работа с отчётами. При инициализации главного окна пользователь выбирает необходимую ему операцию.

При работе со спецификациями пользователь может создавать новые спецификации, а также просматривать, редактировать или удалять существующие. При создании спецификации задаются номер спецификации, её наименование, и маршрут. Если спецификация является деталью, то дополнительно указываются материал, заготовка и норма расхода материала. Пользователь также может создавать новые спецификации путём копирования существующих. В этом случае создаётся новая спецификация с такими же данными, как у копируемой. При редактировании спецификации пользователь может редактировать любые данные самой спецификации, а также добавлять или удалять из неё входящие сборки и детали. При операции удаления спецификация удаляется из базы данных.

При работе с отчётами пользователь выбирает нужный ему тип отчёта и вводит интересующий его номер спецификации. На экране появится выбранный отчёт, который при необходимости можно сохранить в файле Excel-формата.

Схема работы системы с основными операциями представлена на РТДП 3032.020.002.

1. Выбор СУБД и разработка проекта базы данных

Для проектирования базы данных, а также для построения моделей будем использовать CASE-средство ERwin. ERwin — средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных. Модели данных помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания. [3]

ERwin предоставляет возможности создавать и управлять двумя различными уровнями представления одной модели, равно как и иметь много вариантов отображения на каждом уровне.

Для построения информационной модели данных необходимо пройти следующие этапы:

1. Построение инфологической модели данных, включающее следующие пункты:

* определение сущностей;
* определение зависимостей между сущностями;
* определение атрибутов сущностей.

1. Построение даталогической модели данных, включающее следующие пункты:

* переход к физическому описанию модели: назначение соответствий имя сущности – имя таблицы, атрибут сущности – атрибут таблицы;
* задание первичных и вторичных ключей;
* приведение модели к требуемому уровню нормальной формы.

1. Создание схемы базы данных, включающее следующие пункты:

* задание типов данных колонок таблиц в соответствии с типами целевой СУБД;
* задание ограничений, триггеров и процедур;
* генерация базы данных.

Итак, для построения инфологической модели данных были определены следующие сущности, представленные в таблице 3.5.

Таблица 3.5 – Сущности информационной системы

|  |  |
| --- | --- |
| Наименование сущности | Описание сущности |
| Сборки | Сборочные единицы, изготавливаемые на предприятии |
| Входящие сборки | Сборочные единицы, которые входят в другие сборочные единицы |
| Детали | Детали, изготавливаемые на предприятии |
| Материалы | Материалы, используемые для изготовления деталей |
| Вспомогательные материалы | Вспомогательные материалы, используемые для изготовления сборочных единиц |
| Маршруты | Маршруты изготовления сборочных единиц и деталей на предприятии |
| Подразделения | Имеющиеся на предприятии подразделения, в которых изготавливаются сборочные единицы и детали |

В таблице 3.6 представлены зависимости между сущностями информационной системы.

Таблица 3.6 – Зависимости между сущностями информационной системы

|  |  |
| --- | --- |
| Зависимость | Описание |
| Сборки – Входящие сборки | Одна сборка может содержать много входящих сборок. Каждая входящая сборка входит в какую-нибудь сборку. Если во входящую сборку также входят сборки, то такая сборка по отношению к входящим является уже родительской и зависимость между ними сохраняется на уровне «сборки – входящие сборки» |
| Сборки – Детали | Каждая сборка может содержать детали. Каждая деталь входит в какую-нибудь сборку |
| Детали – Материалы | Каждая деталь изготавливается из какого-нибудь материала. Каждый материал может применяться для изготовления деталей |
| Сборки – Вспомогательные материалы | Для изготовления каждой сборки могут применяться вспомогательные материалы. Каждый материал может быть применим для изготовления любой сборки |
| Сборки – Маршруты | Каждая сборка изготавливается по определённому маршруту. Один маршрут может быть применим для изготовления многих сборок |
| Детали – Маршруты | Каждая деталь изготавливается по определённому маршруту. Один маршрут может быть применим для изготовления многих деталей |
| Маршруты – Подразделения | Каждый маршрут состоит из подразделений предприятия. Одно подразделение может применяться во многих маршрутах |

После определения зависимостей между сущностями нашей информационной системы мы получаем следующую инфологическую модель (рисунок 3.6).

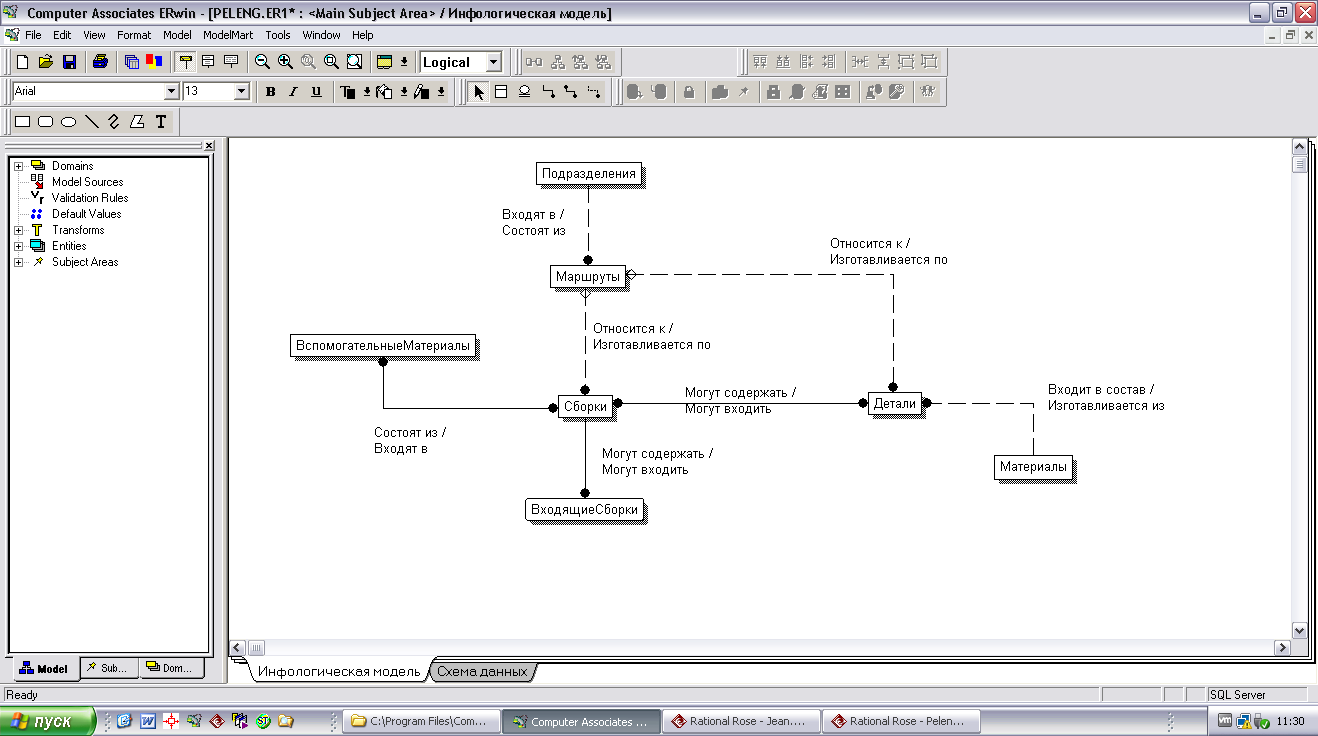


Рисунок 3.6 – Инфологическая модель данных на уровне сущностей с определёнными связями

Далее необходимо определить атрибуты каждой сущности (таблица 3.7)

Таблица 3.7 – Атрибуты сущностей информационной системы

|  |  |  |
| --- | --- | --- |
| Наименование сущности | Наименование атрибута | Описание атрибута |
| Сборки | Номер сборки | Идентификационный номер сборочной единицы |
| Наименование | Наименование сборочной единицы |
| Маршрут | Маршрут изготовления сборочной единицы |
| Входящие сборки | Номер входящей сборки | Идентификационный номер входящей сборочной единицы |
| Количество | Количество входимости данной входящей сборки в родительскую |
| Детали | Номер детали | Идентификационный номер детали |
| Наименование | Наименование детали |
| Норма расхода | Норма расхода материала для изготовления детали |
| Выход деталей | Количество получаемых деталей из одной заготовки |

Продолжение таблицы 3.7

|  |  |  |
| --- | --- | --- |
|  | Длина заготовки | Длина заготовки |
| Ширина заготовки | Ширина заготовки |
| Маршрут | Маршрут изготовления детали |
| Материалы | Наименование материала | Номенклатурное обозначение материала |
| Габарит | Габаритный размер для материала (для кругов – диаметр, для листов – толщина) |
| Ширина | Ширина материала (для листовых профилей) |
| Длина | Длина материала (для листовых профилей) |
| Марка материала | Наименование марки материала |
| Профиль | Профиль материала |
| Вспомогательные материалы | Наименование | Наименование вспомогательного материала |
| Плотность | Плотность вспомогательного материала |
| Норма расхода | Норма расхода вспомогательного материала на одну сборку |
| Маршруты | Порядковый номер | Порядковый номер позиции подразделения в маршруте |
| Подразделения | Номер подразделения | Идентификационный номер подразделения предприятия |
| Название участка | Наименование участка |

Определив атрибуты сущностей, мы получим следующую инфологическую модель данных (рисунок 3.7).

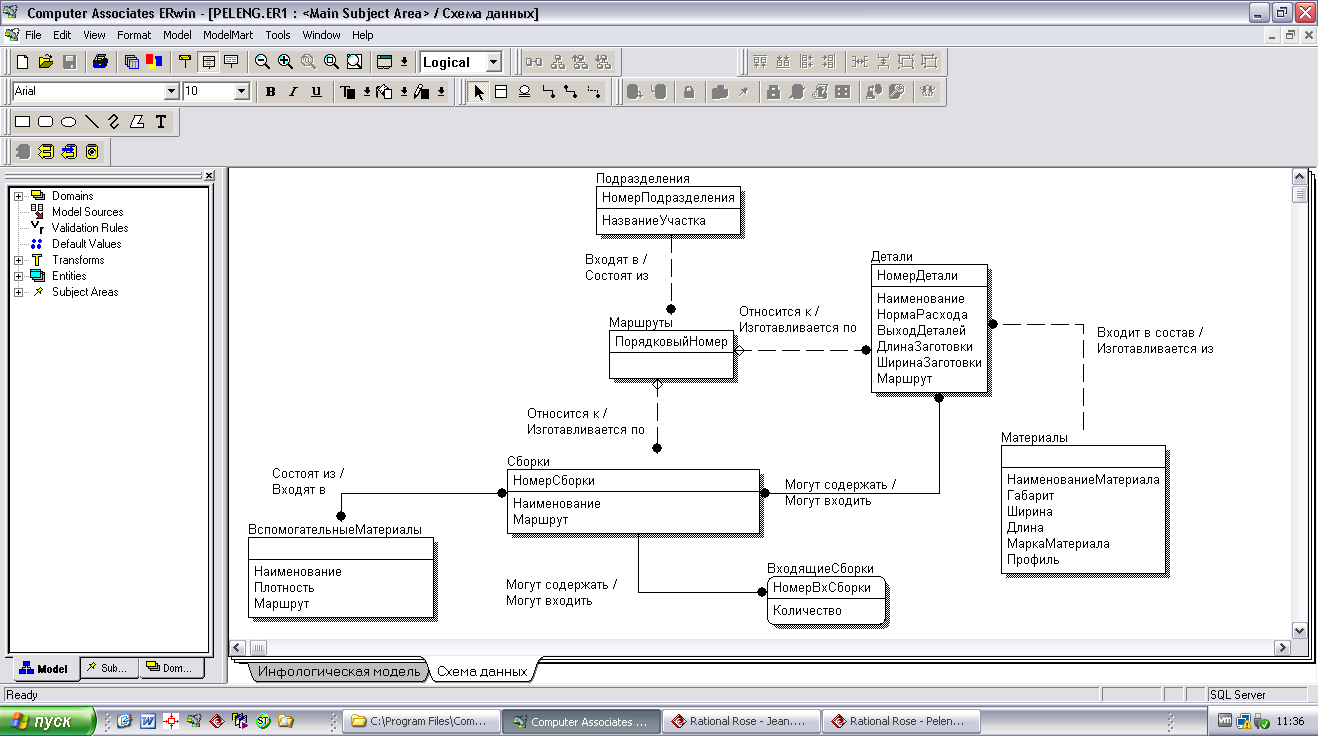


Рисунок 3.7 – Инфологическая модель данных с описанием атрибутов сущностей

Для построения даталогической модели данных необходимо задать первичные и вторичные ключи, а также соответствие атрибутов сущностей атрибутам таблиц. Так как у нас между таблицами есть связи «многие ко многим», то создадим дополнительные таблицы, связывающие предыдущие таблицы и включающие в себя их первичные ключи (таблица 3.8). Так как в таблице Материалы атрибут НаименованиеМатериала зависит от атрибутов МаркаМатериала и Профиль, то вынесем последние атрибуты в отдельные таблицы.

Таблица 3.8 – Атрибуты таблиц информационной системы

|  |  |  |
| --- | --- | --- |
| Наименование таблицы | Наименование атрибута таблицы | Описание атрибута |
| Сборки | НомерСборки (PK) | Идентификационный номер сборочной единицы. Так номер является уникальным, то мы его используем в качестве первичного ключа |
| Наименование | Наименование сборочной единицы |
| МаршрутID (FK) | Идентификатор маршрута изготовления сборочной единицы |

Продолжение таблицы 3.8

|  |  |  |
| --- | --- | --- |
| ВходящиеСборки | НомерВхСборки (PK) | Идентификационный номер входящей сборочной единицы. Является первичным ключом таблицы |
| НомерСборки (FK) | Является первичным ключом таблицы |
| Количество | Количество входимости данной входящей сборки в родительскую |
| Детали | НомерДетали (PK) | Идентификационный номер детали. Является первичным ключом таблицы |
| Наименование | Наименование детали |
| НормаРасхода | Норма расхода материала для изготовления детали |
| ВыходДеталей | Количество получаемых деталей из одной заготовки |
| ДлинаЗаготовки | Длина заготовки |
| ШиринаЗаготовки | Ширина заготовки |
| МаршрутID (FK) | Маршрут изготовления детали |
| Материалы | МатериалID (PK) | Идентификатор материала. Является первичным ключом таблицы |
| НаименованиеМатериала | Номенклатурное обозначение материала |
| Габарит | Габаритный размер для материала (для кругов – диаметр, для листов – толщина) |
| Ширина | Ширина материала (для листовых профилей) |
| Длина | Длина материала (для листовых профилей) |
| МаркаID (FK) | Идентификатор марки материала |
| ПрофильID (FK) | Идентификатор профиля материала |

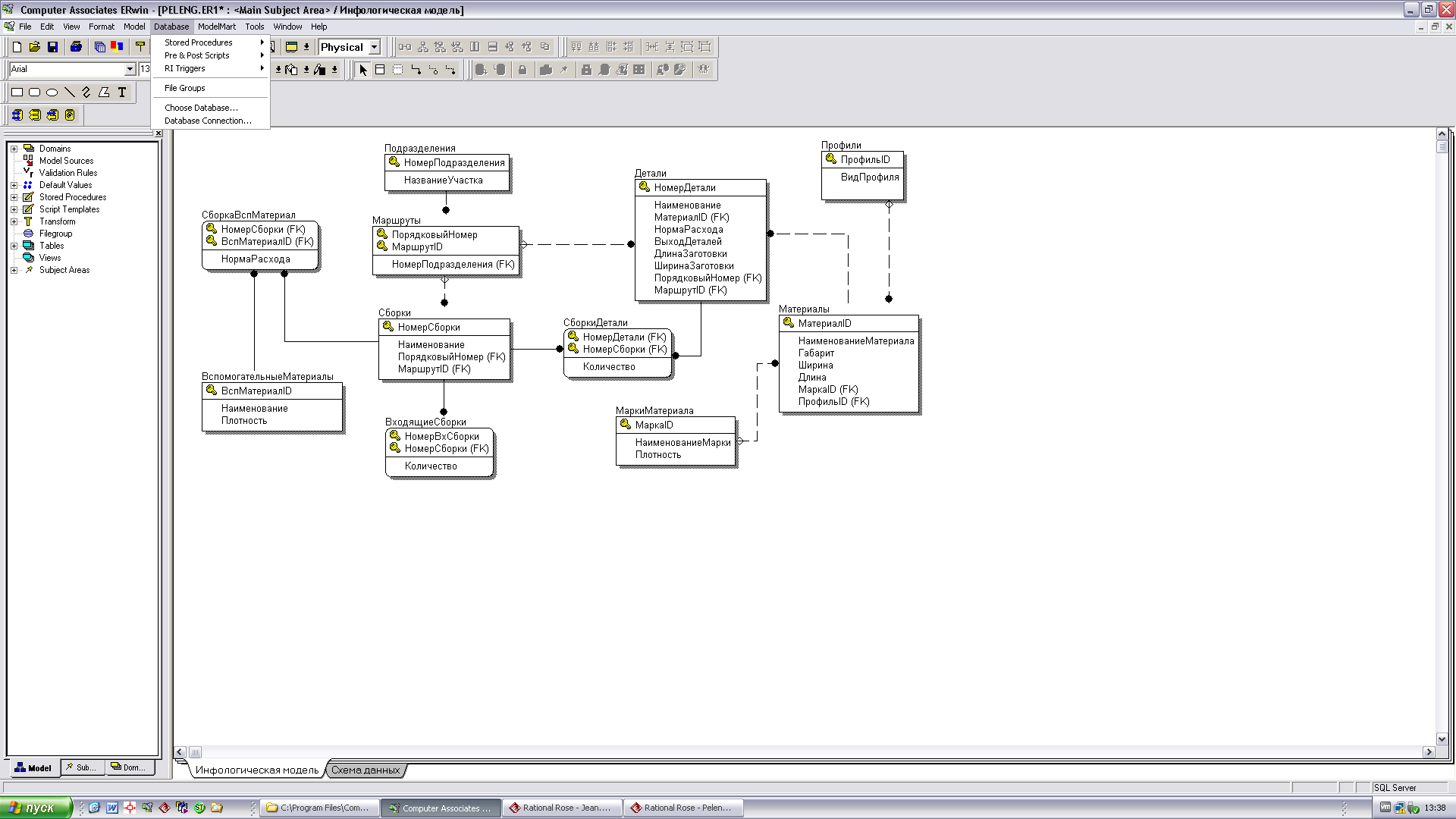
Продолжение таблицы 3.8

|  |  |  |
| --- | --- | --- |
| ВспомогательныеМатериалы | ВспМатериалID (PK) | Идентификатор вспомогательного материала. Является первичным ключом таблицы |
| Наименование | Наименование вспомогательного материала |
| Плотность | Плотность вспомогательного материала |
| Маршруты | МаршрутID (PK) | Идентификатор маршрута. Является первичным ключом таблицы |
| Порядковый номер (PK) | Порядковый номер позиции подразделения в маршруте. Является первичным ключом таблицы |
| НомерПодразделения (FK) | Идентификационный номер подразделения предприятия из таблицы Подразделения |
| Подразделения | НомерПодразделения | Идентификационный номер подразделения предприятия |
| НазваниеУчастка | Наименование участка |
| СборкиДетали | НомерДетали (FK) | Идентификационный номер детали из таблицы Детали. Является первичным ключом таблицы. |
| НомерСборки (FK) | Идентификационный номер сборки из таблицы Сборки. Является первичным ключом таблицы |
| Количество | Количество деталей, входящих в данную сборку |
| СборкаВспМате-риал | НомерСборки (FK) | Идентификационный номер сборки из таблицы Сборки. Является первичным ключом таблицы |

Продолжение таблицы 3.8

|  |  |  |
| --- | --- | --- |
|  | ВспМатериалID (FK) | Идентификационный номер вспомогательного материала из таблицы ВспомогательныеМатериалы. Является первичным ключом таблицы |
| НормаРасхода | Норма расхода вспомогательного материала для изготовления данной сборки |
| МаркиМатериала | МаркаID (PK) | Идентификатор марки материала. Является первичным ключом таблицы |
| НаименованиеМарки | Наименование марки материала |
| Плотность | Плотность материала данной марки |
| Профили | ПрофильID (PK) | Идентификатор профиля материала. Является первичным ключом таблицы |
| ВидПрофиля | Вид профиля |

В результате задания ключей в таблицах и замене связей «многие ко многим» получаем даталогическую модель, представленную на рисунке 3.8.

Рисунок 3.8 – Даталогическая модель данных на уровне сущностей с определёнными связями

Для построения окончательной схемы базы данных необходимо задать каждому атрибуту таблицы типы данных (таблица 3.9)

Таблица 3.9 – Типы данных для атрибутов таблиц

|  |  |  |
| --- | --- | --- |
| Наименование таблицы | Наименование атрибута таблицы | Описание атрибута |
| Сборки | НомерСборки (PK) | varchar (20) NOT NULL |
| Наименование | varchar (max) NOT NULL |
| МаршрутID (FK) | int NOT NULL |
| ВходящиеСборки | НомерВхСборки (PK) | varchar (20) NOT NULL |
| НомерСборки (FK) | varchar (20) NOT NULL |
| Количество | int NOT NULL |
| Детали | НомерДетали (PK) | varchar (20) NOT NULL |
| Наименование | varchar (max) NOT NULL |
| НормаРасхода | float NOT NULL |
| ВыходДеталей | int NOT NULL |
| ДлинаЗаготовки | float NOT NULL |
| ШиринаЗаготовки | float NULL (для круглых профилей не используется) |
| МаршрутID (FK) | int NOT NULL |
| Материалы | МатериалID (PK) | int IDENTITY (1,1) |
| НаименованиеМатериала | varchar (max) NOT NULL |
| Габарит | float NOT NULL |
| Ширина | int NULL (для круглых профилей не используется) |
| Длина | int NULL (для круглых профилей не используется) |
| МаркаID (FK) | int NOT NULL |
| ПрофильID (FK) | int NOT NULL |

Продолжение таблицы 3.9

|  |  |  |
| --- | --- | --- |
| ВспомогательныеМатериалы | ВспМатериалID (PK) | int IDENTITY (1,1) |
| Наименование | varchar (max) NOT NULL |
| Плотность | float NOT NULL |
| Маршруты | МаршрутID (PK) | int IDENTITY (1,1) |
| Порядковый номер (PK) | int NOT NULL |
| НомерПодразделения (FK) | varchar (5) NOT NULL |
| Подразделения | НомерПодразделения | varchar (5) NOT NULL |
| НазваниеУчастка | varchar (max) NOT NULL |
| СборкиДетали | НомерДетали (FK) | varchar (20) NOT NULL |
| НомерСборки (FK) | varchar (20) NOT NULL |
| Количество | int NOT NULL |
| СборкаВспМате-риал | НомерСборки (FK) | varchar (20) NOT NULL |
| ВспМатериалID (FK) | int NOT NULL |
| НормаРасхода | float NOT NULL |
| МаркиМатериала | МаркаID (PK) | int IDENTITY (1,1) |
| НаименованиеМарки | varchar (20) NOT NULL |
| Плотность | float NOT NULL |
| Профили | ПрофильID (PK) | int IDENTITY (1,1) |
| ВидПрофиля | varchar (20) NOT NULL |

Окончательная схема базы данных представлена на рисунке 3.9.

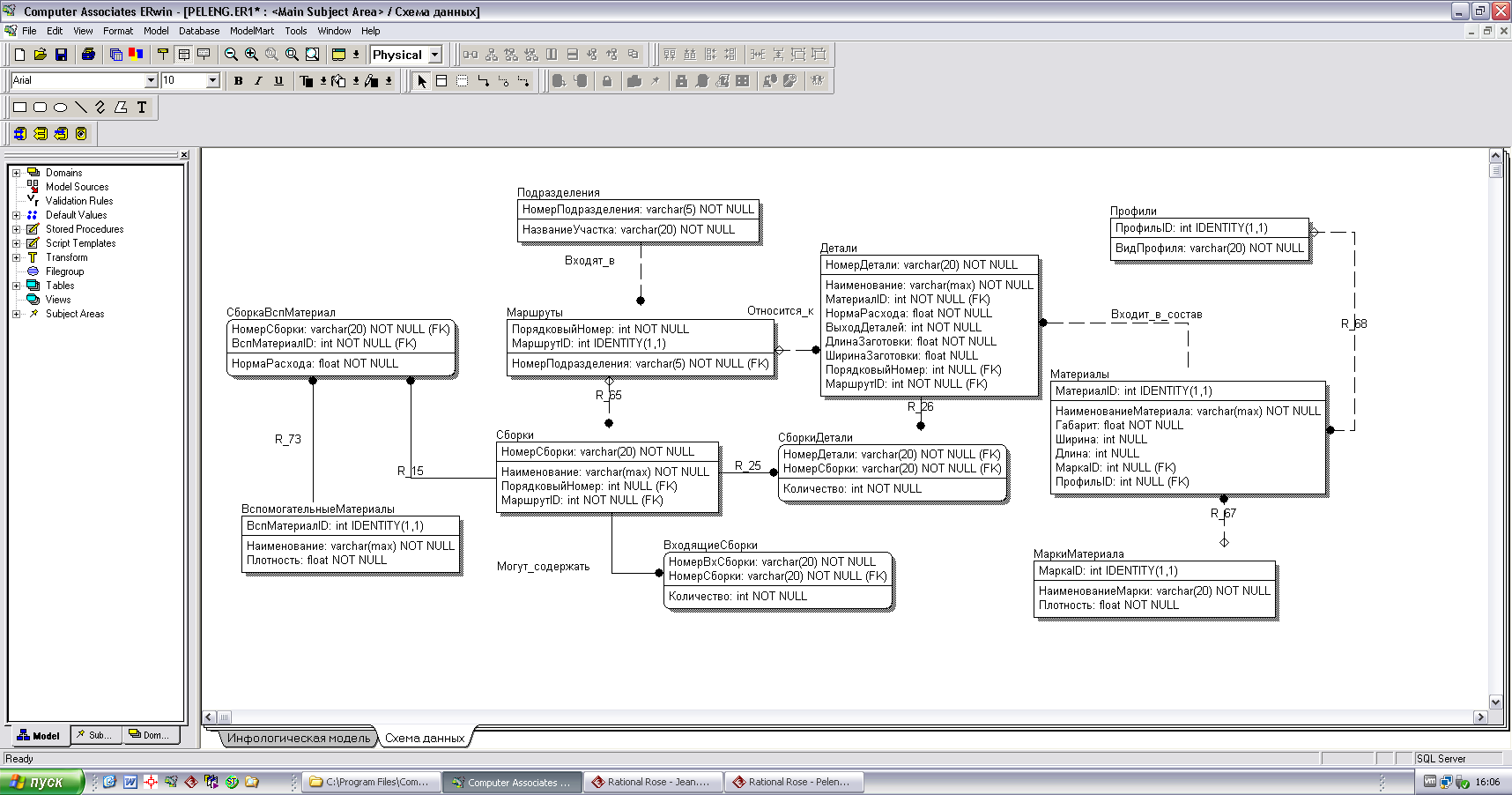


Рисунок 3.9 – Окончательная схема базы данных, полученная в Erwin

Сделаем генерацию создания базы данных в Erwin, после чего по полученному сценарию создадим саму базу данных. Диаграмма базы данных, созданной в программе Microsoft SQL Server Management Studio, представлена на рисунке 3.10. Сценарии создания базы данных и заполнения информацией находятся в приложении А.

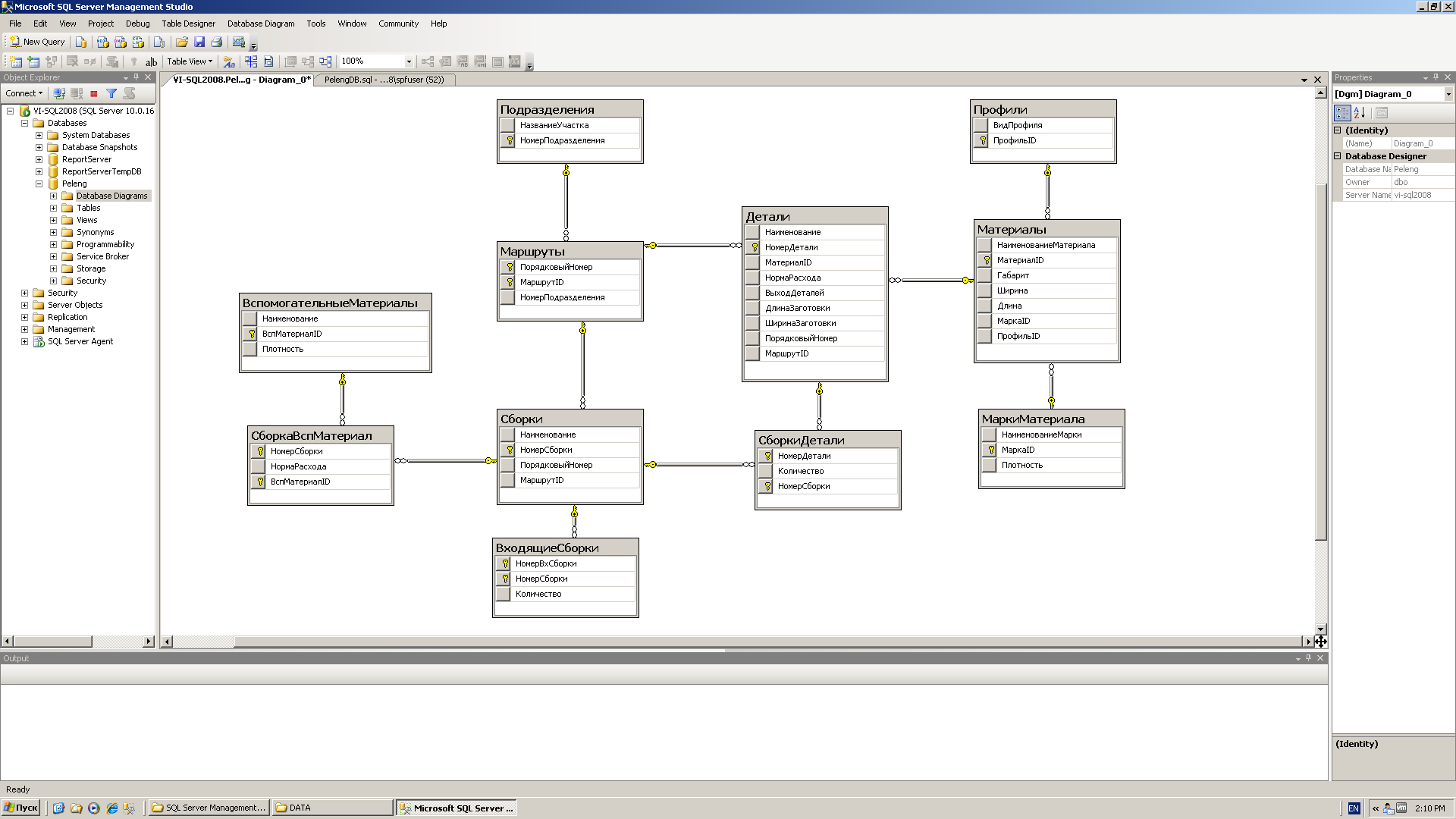


Рисунок 3.10 – Диаграмма базы данных, полученная в Microsoft SQL Server Management Studio

1. Разработка структуры классов

Основными классами программы являются классы, созданные по таблицам базы данных, а также классы форм [4-6]. Описание этих классов и их полей представлено ниже:

Сборки. Сборочные единицы, изготавливаемые на предприятии. Имеет следующие поля: МаршрутID, Наименование и НомерСборки.

Детали. Детали, изготавливаемые на предприятии. Поля: ВыходДеталей, ДлинаЗаготовки, МаршрутID, МатериалID, Наименование, НомерДетали, НормаРасхода, ШиринаЗаготовки.

СборкиДетали. Детали, входящие в сборку. Поля: Количество, НомерДетали, НомерСборки.

ВходящиеСборки. Сборки, входящие в другие сборки. Поля: Количество, НомерВхСборки, НомерСборки.

Материалы. Материалы, имеющиеся на складе предприятия. Поля: Габарит, Длина, МаркаID, МатериалID, Наименование, ПрофильID, Ширина.

МаркиМатериала. Марки материалов, имеющихся на предприятии. Поля: МаркаID, Наименование, Плотность.

Профили. Профили материалов, имеющихся на предприятии. Поля: ВидПрофиля, ПрофильID.

ВспомогательныеМатериалы. Вспомогательные материалы, имеющиеся на предприятии. Поля: ВспМатериалID, Наименование, Плотность.

СборкаВспМатериалы. Вспомогательные материалы, применяемые для изготовления сборочной единицы. Поля: ВспМатериалID, НомерСборки, НормаРасхода.

Подразделения. Подразделения предприятия, задействованные в изготовлении деталей и сборок. Поля: НазваниеУчастка, НомерПодразделения.

Маршруты. Маршрут следования детали или сборки в процессе изготовления. Поля: МаршрутID, НомерПодразделения, ПорядковыйНомер.

ComputeRate. Расчёт заготовок и норм расхода материала для них. Методы: Det – получение детали с выбранной заготовкой и нормой расхода.

ListMaterial. Список материалов, имеющихся на предприятии. Методы: Material – получение выбранного материала со всеми атрибутами.

AddAssembly. Добавление входящей сборки в другую сборку. Методы: AddAssembly – Получение полной информации о входящей сборке и количестве вхождения.

Диаграмма классов программы представлена на рисунке Б.1 приложения Б.

1. Разработка алгоритмов и их программная реализация

Основными функциями при работе со спецификациями являются создание новых спецификаций, добавление информации в созданные спецификации, расчёт нормы расхода материала и заготовки на деталь. Разработаем алгоритмы этих функций.



Рисунок 4.1 – Алгоритм создания новой спецификации

На рисунке 4.1 представлен алгоритм создания новой спецификации в нашей проектируемой системе. Пользователь вводит номер спецификации, которую он хочет создать. Если такой номер уже существует, то пользователь получает уведомление об этом и система ничего не создаёт. Если же номера такого нету, то он проверяется на принадлежность к группе деталей. Все номера, оканчивающиеся на цифру «0», относятся к группе сборочных единиц. Поэтому, если номер относится к детали, то пользователь обязан внести все необходимые данные для детали: наименование, маршрут, материал, норму расхода, выход деталей, длину заготовки. Для сборки достаточно ввести только наименование и маршрут. Для каждой группы данных идёт проверка на правильность ввода. В случае неполных или некорректных данных система вернёт пользователя снова на этап ввода информации. Если же данные верны, то система сохраняет их в базе данных.

На РТДП 3032.020.003 представлен алгоритм расчёта заготовки и нормы расхода материала. Для начала пользователь вводит марку материала, габариты детали и необходимое количество деталей. Если все данные введены верно, то программа продолжит работу. Если же пользователь что-то забыл ввести либо ввёл неправильно, то система уведомит его о необходимости повторного ввода. Программа ведёт расчёт по круглым профилям и листовым. Сначала она по каждому из двух габаритов детали высчитывает описываемую окружность, после чего ведёт поиск наименьшего по габариту материала, но в то же время у которого габарит больше расчётной окружности. При нахождении такого материала система рассчитывает необходимую длину заготовки и норму расхода, после чего заносит эту информацию в список. Потом программа ведёт расчёт по каждому габариту и ищет материал, габарит которого больше габарита детали, но в то же время является наименьшим среди имеющихся. При нахождении такого материала система также рассчитывает необходимые длину заготовки, ширину заготовки и норму расхода, после чего заносит эту информацию в список. После всех расчётов система выдаёт пользователю информацию о подходящих заготовках. Пользователь сам выбирает заготовку, исходя из различных соображений: наименьшая норма расхода, наименьшая трудоёмкость и т. п. Если пользователь не выбрал заготовку, то система выдаст ему напоминание. После выбора заготовки система сохраняет все сведения о её размерах, материале и норме расхода в базе данных.

На рисунке 4.2 представлен алгоритм добавления спецификации во входящие сборки. Если номер родительской спецификации либо входящей является деталью, то такая операция невозможна. Если номер является существующим, то идёт проверка равен ли он родительскому или, возможно, выше в дереве, чем родительский. Если это так, то входящий номер вводится заново, иначе вводится требуемое количество и сохраняется в БД.

На РТДП 3032.020.004 представлен алгоритм вывода ведомости материалов. Ведомость материалов представляет собой отчёт с полными данными о выбранной спецификации. Эти данные включают также сведения обо всех составных спецификациях, которые входят в эту спецификацию либо в её дочерние спецификации. В отчёте отображаются номера всех сборок и деталей, которые являются частью родительской сборки, их количество, маршрут, а также номер сборки, в которую они входят. Для деталей также указывается заготовка, материал и норма расхода. В случае, если выбранная спецификация не содержит входящих элементов, то выводятся сведения только о ней. Если же такая спецификация вообще не существует, то выводится сообщение об ошибке. При желании отчёт можно сохранить в файл Excel-формата.

На РТДП 3032.020.005 представлен алгоритм вывода ведомости расцеховок. Для вывода отчёта необходимо указать номер спецификации и номер подразделения. Из данной спецификации и её составных частей будут отобраны спецификации, маршрут изготовления которых содержит указанный номер подразделения. Если ни одна из спецификаций не удовлетворяет требованиям, то система уведомляет пользователя, что спецификации, изготавливаемые в данном подразделении, не найдены. При наличии хотя бы одной удовлетворяющей записи отчёт можно сохранить в файл Excel-формата.



Рисунок 4.2 – Алгоритм добавления спецификации во входящие сборки

1. Тестирование программного средства

Тестирование является важным этапом разработки программного продукта, позволяя уже на ранних стадиях выявить существенные ошибки. Многозадачные приложения имеют множество функций и подфункций, которые могут быть проверены на функциональсть, не дожидаясь разработки всего приложения. Это позволяет быстрее находить ошибки и устранять их. Ведь гораздо проще протестировать отдельную функцию, сузив рамки появления ошибки, чем тестировать сразу огромное приложение, где ошибки могут быть в любом участке программного кода.

Наше приложение «Управление производственным предприятием» имеет следующий реализованный функционал: работа со спецификациями (создание, корректировка, удаление), расчёт заготовок и норм расхода материала. При расчёте заготовки пользователь может ввести в поля только валидные значения. При создании новой спецификации пользователь может создать спецификацию с номером, которой ещё не существует. Если пользователь пытается создать спецификацию с номером, который уже есть в базе данных, то приложение просто откроет окно с имеющейся информацией по спецификации с данным номером. Листинг кода основных классов разработанного приложения представлен в приложении В. [7-10]

Для тестирования функционала приложения были разработаны тест-кейсы, охватывающие отдельные подмодули приложения (таблица 5.1).

Таблица 5.1 – Тест-кейсы для тестирования функционала приложения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Иден-тифи-катор | Модуль | Подмо-дуль /экран | Описание теста | Ожидаемый результат | Статус |
| ST\_01 | Приложение не запущено |  | Запустить приложение:  1. Запустить файл Peleng.exe | 1. Отобразилось главное окно приложения с активными пунктами меню | Выполнено успешно |
| ST\_02 | Приложение | Меню | Просмотр спецификаций: 1. Нажать на меню «Справочники» 2. Выбрать пункт меню «Спецификации номенклатуры» | 1. Появляется подменю с пунктами «Спецификации номенклатуры», «Технологические карты производства»  2. Появляется окно со списком всех спецификаций | Выполнено успешно |
| ST\_03 | Приложение | Рабочая область | Создание спецификации: 1. Выбрать пункт меню «Спецификации номенклатуры» 2. Нажать кнопку «Создать»  3. Ввести номер спецификации, отсутствующий в базе, правильно заполнив текстовое поле ввода  4. Нажать кнопку «ОК» | 1. Появляется окно со списком всех спецификаций  2. Появляется диалоговое окно с полем для ввода номера спецификации и недоступной кнопкой «ОК»  3. В текстовом поле отображается введённый номер, кнопка «ОК» стала доступной для нажатия  4. Появилось окно спецификации с введённым номером | Выполнено успешно |
| ST\_04 | Приложение | Рабочая область | Копирование спецификации: 1. Выбрать пункт меню «Спецификации номенклатуры»  2. Выделить любую спецификацию для копирования 3. Нажать кнопку «Копировать»  4. Ввести номер спецификации, отсутствующий в базе, правильно заполнив текстовое поле ввода  5. Нажать кнопку «ОК» | 1. Появляется окно со списком всех спецификаций  2. В списке спецификаций выделена строка или ячейка с выбранным номером  3. Появляется диалоговое окно с полем для ввода номера спецификации и недоступной кнопкой «ОК»  4. В текстовом поле отображается введённый номер, кнопка «ОК» стала доступной для нажатия  5. Появилось окно спецификации с введённым номером и полными данными из скопированной спецификации | Выполнено успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ST\_05 | Приложение | Рабочая область | Удаление спецификации: 1. Выбрать пункт меню «Спецификации номенклатуры»  2. Выделить любую спецификацию для удаления 3. Нажать кнопку «Удалить»  4. В появившемся окне нажать кнопку «ОК» | 1. Появляется окно со списком всех спецификаций  2. В списке спецификаций выделена строка или ячейка с выбранным номером  3. Появляется диалоговое окно для подтверждения операции  4. В списке спецификаций удалённая спецификация пропала | Выполнено успешно |
| ST\_06 | Приложение | Рабочая область | Просмотр спецификации: 1. Выбрать пункт меню «Спецификации номенклатуры»  2. Сделать двойной щелчок левой кнопкой мыши по любой спецификации | 1. Появляется окно со списком всех спецификаций  2. Появилось окно спецификации с выбранным номером и полными данными | Выполнено успешно |
| ST\_06 | Приложение | Рабочая область | Корректировка названия спецификации: 1. Открыть любую из спецификаций для просмотра  2. В поле «Название» изменить текст  3. Нажать кнопку «Сохранить» | 1. Появилось окно спецификации с полными данными  2. Текст в поле «Название» изменился  3. Изменения сохранены в базе данных | Выполнено успешно |
| ST\_07 | Приложение | Рабочая область | Корректировка маршрута спецификации: 1. Открыть любую из спецификаций для просмотра  2. В поле «Маршрут» изменить маршрут на любой валидный  3. Нажать кнопку «Сохранить» | 1. Появилось окно спецификации с полными данными  2. Текст в поле «Маршрут» изменился  3. Изменения сохранены в базе данных | Выполнено успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ST\_08 | Приложение | Рабочая область | Добавление существующей входящей сборки: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Сборочные единицы»  3. Нажать кнопку «Добавить»  4. В появившемся окне указать валидный номер существующей сборки, невходящей в данную  5. Нажать кнопку «ОК» | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Сборочные единицы»  3. Появилось диалоговое окно с полем для ввода номера спецификации  4. В текстовом поле отображается номер сборки. Кнопка «ОК» стала доступной  5. В таблице входящих сборок появилась запись с нашей сборкой. | Выполнено успешно |
| ST\_09 | Приложение | Рабочая область | Добавление несуществующей входящей сборки: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Сборочные единицы»  3. Нажать кнопку «Добавить»  4. В появившемся окне указать валидный номер несуществующей сборки  5. Нажать кнопку «ОК»  6. В появившемся диалоговом окне нажать кнопку «ОК»  7. Заполнить новую спецификацию данными и сохранить  8. Закрыть окно с созданной спецификацией | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Сборочные единицы»  3. Появилось диалоговое окно с полем для ввода номера спецификации  4. В текстовом поле отображается номер сборки. Кнопка «ОК» стала доступной  5. Появилось окно с предложением создать новую спецификацию  6. Появилось окно новой спецификации  7. Данные сохранены в базе данных  8. В таблице входящих сборок появилась запись с нашей сборкой. | Выполнено успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ST\_10 | Приложение | Рабочая область | Редактирование количества входящей сборки: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Сборочные единицы»  3. Нажать кнопку «Редактировать» или дважды щёлкнуть по нужной ячейке  4. В появившемся окне в поле «Количество» изменить значение  5. Нажать кнопку «ОК» | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Сборочные единицы»  3. Появилось диалоговое окно с заполненными полями номера и количества выбранной сборки  4. В поле «Количество» изменилось значение  5. В таблице входящих сборок изменилась входимость выбранной сборки | Выполнено успешно |
| ST\_11 | Приложение | Рабочая область | Удаление входящей сборки: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Сборочные единицы»  3. Выделить любую строку или ячейку с нужным номером  4. Нажать кнопку «Удалить» | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Сборочные единицы»  3. Строка или ячейка с выбранным номером выделены цветом  4. В таблице входящих сборок изменилась выбранная сборка исчезла | Выполнено успешно |
| ST\_12 | Приложение | Рабочая область | Выбор материала для заготовки вручную: 1. Открыть любую из спецификаций детали  2. Выбрать вкладку «Материалы»  3. Нажать кнопку «…»  4. Выбрать любой из материалов  5. Нажать кнопку «ОК» | 1. Появилось окно спецификации на деталь с полными данными  2. Отобразилось окно вкладки «Материалы»  3. Появилось окно со списком материалов  4. Выбранный материал выделен цветом  5. В поле «Номенклатура материала» отображается выбранный материал | Выполнено успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ST\_13 | Приложение | Рабочая область | Расчёт заготовки: 1. Открыть любую из спецификаций детали  2. Выбрать вкладку «Материалы»  3. Нажать кнопку «Рассчитать…»  4. Заполнить все поля валидными значениями  5. Выбрать из списка любую заготовку и нажать кнопку «Выбрать и выйти» | 1. Появилось окно спецификации на деталь с полными данными  2. Отобразилось окно вкладки «Материалы»  3. Появилось окно для расчёта заготовки  4. Все поля заполнены, кнопка «Выбрать и выйти» доступна  5. Выбранная заготовка отображается в поле «Заготовка» | Выполнено успешно |
| ST\_14 | Приложение | Рабочая область | Задание нормы расхода вручную: 1. Открыть любую из спецификаций детали  2. Выбрать вкладку «Материалы»  3. В поле «Норма расхода» изменить значение на любое валидное  4. Нажать кнопку «Записать» | 1. Появилось окно спецификации на деталь с полными данными  2. Отобразилось окно вкладки «Материалы»  3. В поле «Норма расхода» изменилось значение  4. Новое значение сохранено в базе данных | Выполнено успешно |
| ST\_15 | Приложение | Рабочая область | Добавление детали в сборку: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Детали»  3. Нажать кнопку «Добавить»  4. Ввести в поле существующий номер детали, невходящей в данную сборку  5. Нажать кнопку «ОК» | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Детали»  3. Появилось диалоговое окно с полем ввода номера  4. В поле номера введён существующий номер детали, кнопка «ОК» доступна  5. В таблице входящих деталей отображается введённая деталь | Выполнено успешно |
| ST\_16 | Приложение запущено |  | Закрытие приложение:  1. Закрыть главное окно приложения | 1. Главное окно приложения закрылось. Процесс Peleng.exe завершён | Выполнено успешно |

Продолжение таблицы 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ST\_17 | Приложение | Меню | Просмотр ведомости материалов: 1. Нажать на меню «Отчёты» 2. Выбрать пункт меню «Ведомость материалов»  3. Указать номер существующей спецификации  4. Нажать кнопку «ОК» | 1. Появляется подменю с пунктами  2. Появилось диалоговое окно с полем для ввода номера спецификации  3. В поле номера введён номер существующей спецификации  4. В файле Excel отображается выбранный отчёт | Выполнено успешно |
| ST\_18 | Приложение | Меню | Просмотр ведомости расцеховок: 1. Нажать на меню «Отчёты» 2. Выбрать пункт меню «Ведомость расцеховок»  3. Указать номер существующей спецификации  4. Нажать кнопку «ОК» | 1. Появляется подменю с пунктами  2. Появилось диалоговое окно с полем для ввода номера спецификации  3. В поле номера введён номер существующей спецификации  4. В файле Excel отображается выбранный отчёт | Выполнено успешно |
| ST\_19 | Приложение | Меню | Просмотр ведомости входимости: 1. Нажать на меню «Отчёты» 2. Выбрать пункт меню «Ведомость входимости»  3. Указать номер существующей спецификации  4. Нажать кнопку «ОК» | 1. Появляется подменю с пунктами  2. Появилось диалоговое окно с полем для ввода номера спецификации  3. В поле номера введён номер существующей спецификации  4. В файле Excel отображается выбранный отчёт | Выполнено успешно |
| ST\_20 | Приложение | Рабочая область | Удаление детали из сборки: 1. Открыть любую из спецификаций сборки  2. Выбрать вкладку «Детали»  3. Выделить в таблице любую из ячеек или строку с номером детали  4. Нажать кнопку «Удалить» | 1. Появилось окно спецификации на сборку с полными данными  2. Отобразилось окно вкладки «Детали»  3. Выбранная деталь выделена цветом  4. В таблице входящих деталей выбранная деталь исчезла | Выполнено успешно |

Заключение

В данном дипломном проекте были проведены следующие работы:

* аналитический обзор и анализ существующих аналогов;
* определение требований к системе и постановка задачи;
* разработка методов и моделей, положенных в основу проекта;
* разработка проекта программного обеспечения, включающая функциональный анализ предметной области и разработку базы данных;
* разработка алгоритмов и схем работы системы;
* тестирование разработанного программного средства.

В разделе аналитического обзора был проведен анализ существующих аналогов по теме управления производственным предприятием. Сравнив различные аналоги по функционалу, интерфейсу и удобству работы, был определён перечень требований к разрабатываемой системе. Постановка задачи основывалась на основных проблемах при управлении производством на предприятии, таких как ведение спецификаций, расчёт оптимальных заготовок, планирование и контроль изготовления продукции, а также ведение единой базы технологических процессов.

В разделе разработки методов и моделей, положенных в основу проекта, были рассмотрены основные способы расчёта заготовок и норм расхода материала, а также выбрана древовидная структура записи спецификаций.

В разделе разработки проекта программного обеспечения был проведён функциональный анализ предметной области с разработкой спецификаций требований. Исходя из поставленных задач, выделили возможных пользователей системы, таких как технолог ВМ, технолог-разработчик, технолог-плановик, ведущий инженер-технолог, а также определили для каждого из них варианты использования системы. С помощью трассировочных матриц были определены спецификации требований к программному продукту. На основании определения сущностей системы и их взаимосвязей была спроектирована база данных системы.

В разделе разработки алгоритмов были приведены основные схемы программы, реализующие такие функции, как создание новой спецификации, расчёт заготовки и нормы расхода материала, добавление входящей сборки.

В разделе тестирования функционала приложения были разработаны тест-кейсы, охватывающие отдельные подмодули приложения, такие как запуск приложения, нажатие меню, создание, редактирование и удаление спецификаций, вывод и сохранение отчётов, а также выход из приложения.

В разработанной программе реализованы работа со спецификациями, расчёт заготовок и норм расхода материала, а также вывод и сохранение типовых отчётов. Для управления всем производственным процессом на предприятии в данной программе необходимо реализовать возможность планирования и контроля изготовления продукции.

Список использованных источников

1. АСУ ТПП. Техописание. Настройка. Руководство по эксплуатации программного средства. Гродно: Алгоритм, 2002. -85с.
2. Плотность различных материалов. Справочная таблица [Электрон. ресурс] - 19.03.2015. - Режим доступа: http://promtk.com/calc/density
3. Проектирование и реализация баз данных и клиентских приложений в среде MS Visual Studio.Net: Учебное пособие / Макурин Ю.Д., Сивохин А. В. Пенза: Частная типография Тугушева, 2010. -230с.
4. LINQ. Карманный справочник / Албахари Дж., Албахари Б. СПб.: БХВ-Петербург, 2009. -240с.
5. LINQ Unleashed for C# / Kimmel Paul. Pearson Education, Inc., 2009. -528с.
6. Programming Microsoft® LINQ in Microsoft.NET Framework 4 / Paolo Pialorsi, Marco Russo. OTSI, Inc., 2010. -680с.
7. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов / Макки Алекс. Москва, 2010. -412 с.
8. Каталог API (Microsoft) и справочных материалов [Электрон. ресурс] - 19.03.2015. - Режим доступа: https://msdn.microsoft.com/library
9. Формы со вкладками. Оформление приложений с помощью многостраничных панелей [Электрон. ресурс] - 19.03.2015. - Режим доступа: http://delphiexample.at.ua/publ/formy\_so\_vkladkami/1-1-0-11
10. Professional Visual Studio 2010 (Wrox Programmer to Programmer) / Nick Randolph and others. Wrox, 2010. -1177с.

Приложение А

(Обязательное)

Сценарий создания базы данных

CREATE TABLE ВспомогательныеМатериалы (

Наименование varchar(max) NOT NULL,

ВспМатериалID int IDENTITY(1,1),

Плотность float NOT NULL)

go

ALTER TABLE ВспомогательныеМатериалы

ADD PRIMARY KEY NONCLUSTERED (ВспМатериалID)

go

CREATE TABLE ВходящиеСборки (

НомерВхСборки varchar(20) NOT NULL,

НомерСборки varchar(20) NOT NULL,

Количество int NOT NULL)

go

ALTER TABLE ВходящиеСборки

ADD PRIMARY KEY NONCLUSTERED (НомерВхСборки, НомерСборки)

go

CREATE TABLE Детали (

Наименование varchar(max) NOT NULL,

НомерДетали varchar(20) NOT NULL,

МатериалID int NOT NULL,

НормаРасхода float NOT NULL,

ВыходДеталей int NOT NULL,

ДлинаЗаготовки float NOT NULL,

ШиринаЗаготовки float NULL,

ПорядковыйНомер int NULL,

МаршрутID int NOT NULL)

go

ALTER TABLE Детали

ADD PRIMARY KEY NONCLUSTERED (НомерДетали)

go

CREATE TABLE МаркиМатериала (

НаименованиеМарки varchar(20) NOT NULL,

МаркаID int IDENTITY(1,1),

Плотность float NOT NULL)

go

ALTER TABLE МаркиМатериала

ADD PRIMARY KEY NONCLUSTERED (МаркаID)

go

CREATE TABLE Маршруты (

ПорядковыйНомер int NOT NULL,

МаршрутID int NOT NULL,

НомерПодразделения varchar(5) NOT NULL)

go

ALTER TABLE Маршруты

ADD PRIMARY KEY NONCLUSTERED (ПорядковыйНомер, МаршрутID)

go

CREATE TABLE Материалы (

НаименованиеМатериала varchar(max) NOT NULL,

МатериалID int IDENTITY(1,1),

Габарит float NOT NULL,

Ширина int NULL,

Длина int NULL,

МаркаID int NULL,

ПрофильID int NULL)

go

ALTER TABLE Материалы

ADD PRIMARY KEY NONCLUSTERED (МатериалID)

go

CREATE TABLE Подразделения (

НазваниеУчастка varchar(max) NOT NULL,

НомерПодразделения varchar(5) NOT NULL)

go

ALTER TABLE Подразделения

ADD PRIMARY KEY NONCLUSTERED (НомерПодразделения)

go

CREATE TABLE Профили (

ВидПрофиля varchar(20) NOT NULL,

ПрофильID int IDENTITY(1,1))

go

ALTER TABLE Профили

ADD PRIMARY KEY NONCLUSTERED (ПрофильID)

go

CREATE TABLE СборкаВспМатериал (

НомерСборки varchar(20) NOT NULL,

НормаРасхода float NOT NULL,

ВспМатериалID int NOT NULL)

go

ALTER TABLE СборкаВспМатериал

ADD PRIMARY KEY NONCLUSTERED (НомерСборки, ВспМатериалID)

go

CREATE TABLE Сборки (

Наименование varchar(max) NOT NULL,

НомерСборки varchar(20) NOT NULL,

ПорядковыйНомер int NULL,

МаршрутID int NOT NULL)

go

ALTER TABLE Сборки

ADD PRIMARY KEY NONCLUSTERED (НомерСборки)

go

CREATE TABLE СборкиДетали (

НомерДетали varchar(20) NOT NULL,

Количество int NOT NULL,

НомерСборки varchar(20) NOT NULL)

go

ALTER TABLE СборкиДетали

ADD PRIMARY KEY NONCLUSTERED (НомерДетали, НомерСборки)

go

ALTER TABLE ВходящиеСборки

ADD FOREIGN KEY (НомерСборки)

REFERENCES Сборки

go

ALTER TABLE Детали

ADD FOREIGN KEY (ПорядковыйНомер, МаршрутID)

REFERENCES Маршруты

go

ALTER TABLE Детали

ADD FOREIGN KEY (МатериалID)

REFERENCES Материалы

go

ALTER TABLE Маршруты

ADD FOREIGN KEY (НомерПодразделения)

REFERENCES Подразделения

go

ALTER TABLE Материалы

ADD FOREIGN KEY (ПрофильID)

REFERENCES Профили

go

ALTER TABLE Материалы

ADD FOREIGN KEY (МаркаID)

REFERENCES МаркиМатериала

go

ALTER TABLE СборкаВспМатериал

ADD FOREIGN KEY (ВспМатериалID)

REFERENCES ВспомогательныеМатериалы

go

ALTER TABLE СборкаВспМатериал

ADD FOREIGN KEY (НомерСборки)

REFERENCES Сборки

go

ALTER TABLE Сборки

ADD FOREIGN KEY (ПорядковыйНомер, МаршрутID)

REFERENCES Маршруты

go

ALTER TABLE СборкиДетали

ADD FOREIGN KEY (НомерДетали)

REFERENCES Детали

go

ALTER TABLE СборкиДетали

ADD FOREIGN KEY (НомерСборки)

REFERENCES Сборки

go

Сценарий заполнения базы данных информацией

use Peleng

go

insert into Подразделения (НомерПодразделения,НазваниеУчастка)

values ('50170','Заготовительный участок'),

('50100','Механический цех'),

('50180','Участок термической обработки'),

('50190','Участок обработки стеклянными шариками'),

('50160','Участок полимерных покрытий'),

('50032','Склад готовых деталей'),

('К0121','Участок термической обработки'),

('К0113','Участок гальванических покрытий'),

('30700','Сборочный участок'),

('90340','Бюро входного контроля'),

('50240','Электромонтажный участок')

go

insert into Профили (ВидПрофиля)

values ('Пруток'),

('Круг'),

('Лист'),

('Плита'),

('Квадрат'),

('Лента'),

('Проволока')

go

insert into МаркиМатериала (НаименованиеМарки,Плотность)

values ('Сталь 20', 7.85),

('Сталь 45', 7.85),

('Сталь А12', 7.85),

('Сталь 20Х', 7.85),

('Сталь 40Х', 7.85),

('Сталь 20Х13', 7.7),

('Сталь 40Х13', 7.72),

('Сталь 95Х18', 7.75),

('Сталь 12Х18Н9Т', 7.9),

('Сталь 12Х18Н10Т', 7.9),

('Сплав Д16', 2.8),

('Сплав В95', 2.8),

('Сплав АМц', 2.73),

('Титан ВТ1-0', 4.52),

('Латунь ЛС59-1', 8.5),

('Латунь Л63', 8.5),

('Медь М1', 8.9),

('Медь М3', 8.9),

('Бронза БрАЖМц', 7.5),

('Бронза БрАЖ9-4', 7.5),

('Бронза Бр Б2', 8.23)

go

insert into Материалы (НаименованиеМатериала,МаркаID,ПрофильID,Габарит,Ширина,Длина)

values ('Пруток Д16Т КР30П ГОСТ 21488-97',11,1,30,NULL,NULL),

('Пруток Д16Т КР40П ГОСТ 21488-97',11,1,40,NULL,NULL),

('Пруток Д16Т КР50П ГОСТ 21488-97',11,1,50,NULL,NULL),

('Пруток Д16Т КР20П ГОСТ 21488-97',11,1,20,NULL,NULL),

('Пруток Д16Т КР22П ГОСТ 21488-97',11,1,22,NULL,NULL),

('Пруток Д16Т КР75П ГОСТ 21488-97',11,1,75,NULL,NULL),

('Пруток Д16Т КР110 ГОСТ 21488-97',11,1,110,NULL,NULL),

('Пруток Д16Т КР24П ГОСТ 21488-97',11,1,24,NULL,NULL),

('Пруток Д16Т КР25П ГОСТ 21488-97',11,1,25,NULL,NULL),

('Пруток Д16Т КР28П ГОСТ 21488-97',11,1,28,NULL,NULL),

('Круг 8-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,8,NULL,NULL),

('Круг 18-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,18,NULL,NULL),

('Круг 20-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,20,NULL,NULL),

('Круг 25-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,25,NULL,NULL),

('Круг 40-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,40,NULL,NULL),

('Круг 80-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,80,NULL,NULL),

('Круг 10-В1 ГОСТ 7317-75 / 95Х18-б ГОСТ 1053-77',8,2,10,NULL,NULL),

('Круг 10-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,10,NULL,NULL),

('Круг 20-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,20,NULL,NULL),

('Круг 30-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,30,NULL,NULL),

('Круг 50-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,50,NULL,NULL),

('Круг 56-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,56,NULL,NULL),

('Круг 90-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,90,NULL,NULL),

('Круг 35-В1 ГОСТ 7317-75 / 45 ГОСТ 1053-77',2,2,35,NULL,NULL)

go

insert into ВспомогательныеМатериалы (Наименование,Плотность)

values ('Клей ВК-9', 3.85),

('Клей К300', 4.08),

('Клей ЭКАН-15', 3.4),

('Смазка ОКБ122-7', 2.79),

('Смазка Braycot', 2.15),

('Грунтовка АК-070 красная(красно-оранжевая)', 6.5),

('Грунтовка АК-070 жёлтая', 6.6),

('Припой ПОС-61', 7.75),

('Припой ПОСК50-18', 7.9)

go

insert into Маршруты (МаршрутID,ПорядковыйНомер,НомерПодразделения)

values (1,1, '50170'),

(1,2, '50100'),

(1,3, '50180'),

(1,4, '50100'),

(1,5, '50032'),

(1,6, '30700'),

(2,1, '50170'),

(2,2, '50100'),

(2,3, '50032'),

(2,4, '30700'),

(3,1, '50170'),

(3,2, '50100'),

(3,3, '50180'),

(3,4, '50100'),

(3,5, 'К0113'),

(3,6, '90340'),

(3,7, '50032'),

(3,8, '30700'),

(4,1, '30700'),

(4,2, '50240'),

(4,3, '30700'),

(5,1, '30700')

go

insert into Детали (НомерДетали,Наименование,МаршрутID,МатериалID,НормаРасхода,ВыходДеталей,ДлинаЗаготовки,ШиринаЗаготовки)

values ('6135.01.11.215','Оправа',3,6,2.85,1,42,NULL),

('6135.01.11.211','Корпус',3,16,4.85,1,115,NULL),

('6135.01.10.011','Винт',1,11,0.055,10,140,NULL),

('6135.01.10.012','Гайка',2,12,0.088,10,150,NULL),

('6135.01.10.013','Шайба',2,11,0.088,30,280,NULL),

('6135.01.01.001','Корпус',3,7,6.465,1,80,NULL)

go

insert into Сборки (НомерСборки,Наименование,МаршрутID)

values ('6135.01.01.000','Корпус объектива', 4),

('6135.01.10.000','Объектив ОЭЗА', 4),

('6135.01.11.000','Теплосток с платой', 4),

('6135.01.10.100','Держатель', 5),

('6135.01.10.200','Зашлушка', 5),

('6135.01.11.050','Плата', 4),

('6135.01.10.210','Табличка', 5)

go

insert into СборкиДетали (НомерСборки,НомерДетали,Количество)

values ('6135.01.01.000','6135.01.01.001', 1),

('6135.01.10.000','6135.01.10.011', 1),

('6135.01.10.000','6135.01.10.012', 2),

('6135.01.10.000','6135.01.10.013', 4),

('6135.01.10.100','6135.01.10.013', 2),

('6135.01.10.200','6135.01.10.013', 4),

('6135.01.11.050','6135.01.11.215', 2),

('6135.01.10.210','6135.01.11.211', 1)

go

insert into ВходящиеСборки (НомерСборки,НомерВхСборки,Количество)

values ('6135.01.01.000','6135.01.10.000', 1),

('6135.01.01.000','6135.01.11.000', 1),

('6135.01.10.000','6135.01.10.100', 1),

('6135.01.10.000','6135.01.10.200', 2),

('6135.01.10.200','6135.01.10.210', 1),

('6135.01.11.000','6135.01.10.210', 2)

go

insert into СборкаВспМатериал (НомерСборки,ВспМатериалID,НормаРасхода)

values ('6135.01.01.000',1, 0.003),

('6135.01.01.000',6, 0.0015),

('6135.01.10.000',6, 0.002),

('6135.01.10.000',2, 0.01),

('6135.01.11.000',8, 0.008)

go

Приложение Б

(Обязательное)

Диаграмма классов программы

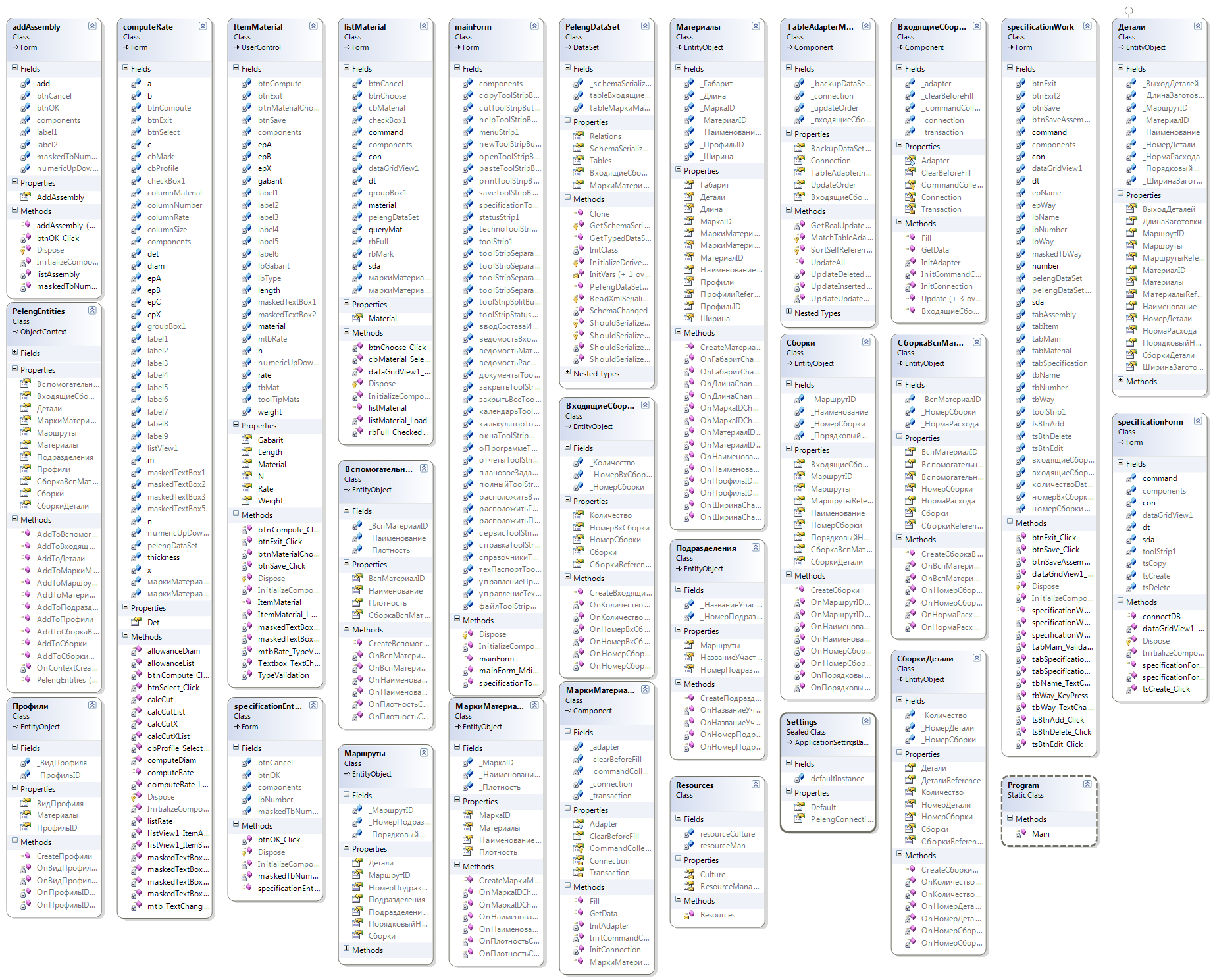


Рисунок Б.1 – Диаграмма классов программы

Приложение В

(Обязательное)

Листинг кода

Класс addAssembly.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Peleng

{

public partial class addAssembly : Form

{

private ВходящиеСборки add = new ВходящиеСборки();

public addAssembly(string number)

{

InitializeComponent();

if (maskedTbNumber.CanFocus)

maskedTbNumber.Focus();

maskedTbNumber.Select(0, 0);

maskedTbNumber.Mask = "0000.00.00.000";

maskedTbNumber.PromptChar = '\_';

add.НомерСборки = number;

}

public addAssembly(string number, string current, int k)

: this(number)

{

this.Text = "Редактирование записи";

maskedTbNumber.Text = current;

numericUpDown1.Value = k;

}

public ВходящиеСборки AddAssembly

{

get

{

return add;

}

}

private void btnOK\_Click(object sender, EventArgs e)

{

string addNumber = maskedTbNumber.Text.Replace(',', '.');

add.Количество = (int)numericUpDown1.Value;

PelengEntities pe = new PelengEntities();

var assemblies = (from m in pe.Сборки

where m.НомерСборки == addNumber

select m);

List<string> inAssemblies = new List<string>();

listAssembly(add.НомерСборки, inAssemblies);

if (assemblies.Count() != 0)

if (addNumber == add.НомерСборки)

{

string message = "Сборка " + addNumber + " не может входить сама в себя!";

MessageBox.Show(message, "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.DialogResult = DialogResult.None;

}

else if (!inAssemblies.Contains(addNumber))

{

add.НомерВхСборки = addNumber;

}

else

{

string message = "Сборка " + addNumber + " не может входить в сборку " + add.НомерСборки + " !";

MessageBox.Show(message, "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.DialogResult = DialogResult.None;

}

else

{

if (MessageBox.Show("Спецификация с таким номером не существует.\nСоздать данную спецификацию?", "Внимание!",

MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.Yes)

{

specificationWork specWork = new specificationWork(addNumber);

if (specWork.ShowDialog(this.ParentForm) == DialogResult.OK)

{

add.НомерВхСборки = addNumber;

this.DialogResult = DialogResult.OK;

}

else

this.DialogResult = DialogResult.Cancel;

}

else

this.DialogResult = DialogResult.None;

}

}

private void maskedTbNumber\_TextChanged(object sender, EventArgs e)

{

if (maskedTbNumber.MaskFull)

{

if (maskedTbNumber.Text[13] != '0')

{

maskedTbNumber.Text = maskedTbNumber.Text.Remove(13);

maskedTbNumber.Text += "0";

}

btnOK.Enabled = true;

}

else

btnOK.Enabled = false;

}

private void listAssembly(string number, List<string> assemblies)

{

PelengEntities pe = new PelengEntities();

List<string> newAssemblies = new List<string>();

var listAssemblies = (from m in pe.ВходящиеСборки

where m.НомерВхСборки == number

select m);

foreach (var l in listAssemblies)

{

if (!assemblies.Contains(l.НомерСборки))

{

newAssemblies.Add(l.НомерСборки);

}

}

assemblies.AddRange(newAssemblies);

if (newAssemblies.Count != 0)

{

foreach (string s in newAssemblies)

{

listAssembly(s, assemblies);

}

}

}

}

}

Класс computeRate.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Peleng

{

public partial class computeRate : Form

{

float a, b, c, x, diam;

int n = 0;

private Детали det = new Детали();

ErrorProvider epA = new ErrorProvider();

ErrorProvider epB = new ErrorProvider();

ErrorProvider epC = new ErrorProvider();

ErrorProvider epX = new ErrorProvider();

public Детали Det

{

get

{

return det;

}

}

public computeRate()

{

InitializeComponent();

maskedTextBox1.ValidatingType = maskedTextBox2.ValidatingType = maskedTextBox3.ValidatingType = typeof(float);

maskedTextBox5.ValidatingType = typeof(uint);

cbProfile.SelectedIndex = 0;

label1.Text = "Диаметр, мм";

}

private void computeRate\_Load(object sender, EventArgs e)

{

try

{

this.маркиМатериалаTableAdapter.Fill(this.pelengDataSet.МаркиМатериала);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

}

}

private void btnCompute\_Click(object sender, EventArgs e)

{

a = float.Parse(maskedTextBox1.Text);

if (cbProfile.SelectedIndex == 1)

b = float.Parse(maskedTextBox2.Text);

c = float.Parse(maskedTextBox3.Text);

if (maskedTextBox5.TextLength != 0)

x = uint.Parse(maskedTextBox5.Text);

n = (int)numericUpDown1.Value;

listRate(a, b, c, n, x);

}

private void mtb\_TextChanged(object sender, EventArgs e)

{

Validate(true);

if (maskedTextBox2.Visible == true)

{

if ((epA.GetError(maskedTextBox1) == "") & (epB.GetError(maskedTextBox2) == "") &

(epC.GetError(maskedTextBox3) == "") & (epX.GetError(maskedTextBox5) == "") &

(maskedTextBox1.TextLength != 0) & (maskedTextBox2.TextLength != 0) &

(maskedTextBox3.TextLength != 0))

btnCompute.Enabled = true;

}

else

{

if ((epA.GetError(maskedTextBox1) == "") & (epC.GetError(maskedTextBox3) == "") & (epX.GetError(maskedTextBox5) == "") &

(maskedTextBox1.TextLength != 0) & (maskedTextBox3.TextLength != 0))

btnCompute.Enabled = true;

}

}

private void maskedTextBox1\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

if (maskedTextBox1.TextLength > 4)

{

epA.SetError(maskedTextBox1, "Слишком длинное значение!");

btnCompute.Enabled = false;

}

else if (!e.IsValidInput)

{

epA.SetError(maskedTextBox1, "Invalid Data Value");

btnCompute.Enabled = false;

}

else if ((maskedTextBox1.Text.Contains('-')) | (maskedTextBox1.Text.Contains('+')) | (maskedTextBox1.Text.Equals("0")))

{

btnCompute.Enabled = false;

epA.SetError(maskedTextBox1, "Invalid Data Value");

}

else

epA.Clear();

}

private void maskedTextBox2\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

if (maskedTextBox2.Visible == true)

{

if (maskedTextBox2.TextLength > 4)

{

epB.SetError(maskedTextBox2, "Слишком длинное значение!");

btnCompute.Enabled = false;

}

else if (!e.IsValidInput)

{

epB.SetError(maskedTextBox2, "Invalid Data Value");

btnCompute.Enabled = false;

}

else if ((maskedTextBox2.Text.Contains('-')) | (maskedTextBox2.Text.Contains('+')) | (maskedTextBox2.Text.Equals("0")))

{

btnCompute.Enabled = false;

epB.SetError(maskedTextBox2, "Invalid Data Value");

}

else

epB.Clear();

}

}

private void maskedTextBox3\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

if (maskedTextBox3.TextLength > 4)

{

epC.SetError(maskedTextBox3, "Слишком длинное значение!");

btnCompute.Enabled = false;

}

else if (!e.IsValidInput)

{

epC.SetError(maskedTextBox3, "Invalid Data Value");

btnCompute.Enabled = false;

}

else if ((maskedTextBox3.Text.Contains('-')) | (maskedTextBox3.Text.Contains('+')) | (maskedTextBox3.Text.Equals("0")))

{

btnCompute.Enabled = false;

epC.SetError(maskedTextBox3, "Invalid Data Value");

}

else

epC.Clear();

}

private void maskedTextBox5\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

if (maskedTextBox5.TextLength > 4)

{

epX.SetError(maskedTextBox5, "Слишком длинное значение!");

btnCompute.Enabled = false;

}

else if ((!e.IsValidInput) & (maskedTextBox5.TextLength != 0))

{

epX.SetError(maskedTextBox5, "Invalid Data Value");

btnCompute.Enabled = false;

}

else if ((maskedTextBox5.Text.Contains('-')) | (maskedTextBox5.Text.Contains('+')))

{

btnCompute.Enabled = false;

epX.SetError(maskedTextBox5, "Invalid Data Value");

}

else

epX.Clear();

}

private float computeDiam(float x, float y)

{

return (float)Math.Pow(x \* x + y \* y, (double)1 / 2);

}

private void listRate(float a, float b, float c, int n, float x)

{

listView1.Items.Clear();

PelengEntities pe = new PelengEntities();

var matsDiam = (from m in pe.Материалы

where ((m.Профили.ВидПрофиля.Contains("Круг") || m.Профили.ВидПрофиля.Contains("Пруток")) &&

m.МаркиМатериала.НаименованиеМарки == cbMark.SelectedValue)

orderby m.Габарит

select m).ToList<Материалы>();

var matsList = (from m in pe.Материалы

where ((!m.Профили.ВидПрофиля.Contains("Круг") & !m.Профили.ВидПрофиля.Contains("Пруток")) &&

m.МаркиМатериала.НаименованиеМарки == cbMark.SelectedValue)

orderby m.Габарит

select m).ToList<Материалы>();

if (cbProfile.SelectedIndex == 1)

diam = computeDiam(a, b);

else

diam = a;

List<Детали> listDet = new List<Детали>();

foreach (var m in matsDiam)

{

if (m.Габарит >= allowanceDiam(diam))

{

listDet.Add(new Детали()

{

ДлинаЗаготовки = (int)((c + 2 + calcCut(m.Габарит)) \* n + x + 0.6F),

ВыходДеталей = n,

МатериалID = m.МатериалID,

НормаРасхода = Math.Round(m.МаркиМатериала.Плотность / 1E6 \* Math.PI \* m.Габарит \* m.Габарит / 4 \* ((int)((c + 2 + calcCut(m.Габарит)) \* n + x + 0.6F) + calcCutX(m.Габарит)) / n, 4)

});

break;

}

}

if (cbProfile.SelectedIndex == 1)

{

diam = computeDiam(a, c);

foreach (var m in matsDiam)

{

if (m.Габарит >= allowanceDiam(diam))

{

listDet.Add(new Детали()

{

ДлинаЗаготовки = (int)((b + 2 + calcCut(m.Габарит)) \* n + x + 0.6F),

ВыходДеталей = n,

МатериалID = m.МатериалID,

НормаРасхода = Math.Round(m.МаркиМатериала.Плотность / 1E6 \* Math.PI \* m.Габарит \* m.Габарит / 4 \* ((int)((b + 2 + calcCut(m.Габарит)) \* n + x + 0.6F) + calcCutX(m.Габарит)) / n, 4)

});

break;

}

}

diam = computeDiam(c, b);

foreach (var m in matsDiam)

{

if (m.Габарит >= allowanceDiam(diam))

{

listDet.Add(new Детали()

{

ДлинаЗаготовки = (int)((a + 2 + calcCut(m.Габарит)) \* n + x + 0.6F),

ВыходДеталей = n,

МатериалID = m.МатериалID,

НормаРасхода = Math.Round(m.МаркиМатериала.Плотность / 1E6 \* Math.PI \* m.Габарит \* m.Габарит / 4 \* ((int)((a + 2 + calcCut(m.Габарит)) \* n + x + 0.6F) + calcCutX(m.Габарит)) / n, 4)

});

break;

}

}

}

foreach (var m in matsList) //расчёт единичных заготовок из листа

{

if (m.Габарит >= allowanceList(a))

{

if (cbProfile.SelectedIndex == 0)

b = a;

int weight, length;

if ((m.Длина > m.Ширина)&(c>b))

{

weight = (int)((b + calcCutList(m.Габарит)) + 0.6F);

length = (int)((c + calcCutList(m.Габарит)) + 0.6F);

if ((m.Ширина >= b) & (m.Ширина < weight))

weight = m.Ширина.Value;

if ((m.Длина >= c) & (m.Длина < length))

length = m.Длина.Value;

}

else

{

weight = (int)((c + calcCutList(m.Габарит)) + 0.6F);

length = (int)((b + calcCutList(m.Габарит)) + 0.6F);

if ((m.Ширина >= c) & (m.Ширина < weight))

weight = m.Ширина.Value;

if ((m.Длина >= b) & (m.Длина < length))

length = m.Длина.Value;

}

if ((m.Ширина >= weight) & (m.Длина >= length))

listDet.Add(new Детали()

{

ДлинаЗаготовки = length,

ШиринаЗаготовки = weight,

ВыходДеталей = 1,

МатериалID = m.МатериалID,

НормаРасхода = Math.Round((m.МаркиМатериала.Плотность / 1E6 \* m.Габарит \* ((int)((b + calcCutList(m.Габарит)) + 0.6F) + calcCutXList(m.Габарит)) \* ((int)((c + calcCutList(m.Габарит)) + 0.6F) + calcCutXList(m.Габарит))),4)

});

break;

}

}

listDet.Sort();

listView1.BeginUpdate();

foreach (Детали detal in listDet)

{

var mat = (from m in pe.Материалы

where m.МатериалID==detal.МатериалID

select m).ToList<Материалы>();

ListViewItem lvi;

listView1.Items.Add(lvi=new ListViewItem(new string[] {mat.First().Ширина.HasValue?mat.First().Габарит + " x " + detal.ШиринаЗаготовки + " x " + detal.ДлинаЗаготовки:"\u00d8 "+ mat.First().Габарит + " x " + detal.ДлинаЗаготовки,

detal.НормаРасхода.ToString(), detal.ВыходДеталей.ToString(),mat.First().НаименованиеМатериала}));

lvi.Tag = detal;

}

listView1.EndUpdate();

if (listView1.Items.Count != 0)

{

btnSelect.Enabled = true;

listView1.Focus();

listView1.Items[0].Selected = true;

}

else

btnSelect.Enabled = false;

}

private float calcCut(double d) //ширина реза детали от заготовки

{

if (d <= 10)

return 2;

if (d <= 40)

return 3;

if (d <= 60)

return 3.5F;

if (d <= 100)

return 4;

else

return 5;

}

private float calcCutX(double d) //ширина реза заготовки ленточной пилой

{

if (d <= 100)

return 1.5F;

if (d <= 200)

return 2;

return 3;

}

private int calcCutXList(double t) //ширина реза заготовки из листа газорезкой

{

if (t <= 6)

return 0;

if (t <= 15)

return 10;

return 12;

}

private int calcCutList(double t) //припуск под механическую обработку заготовки из листа

{

if (t <= 6)

return 5;

if (t <= 10)

return 10;

return 16;

}

private double allowanceDiam(double t) //припуск под механическую обработку из круга

{

if ((checkBox1.Checked) & (cbProfile.SelectedIndex == 0))

return t;

else

return t + 1.8;

}

private double allowanceList(double t) //припуск под механическую обработку из листа

{

if (checkBox1.Checked)

return t;

else

return t + 3.5;

}

private void btnSelect\_Click(object sender, EventArgs e)

{

foreach (ListViewItem l in listView1.SelectedItems)

{

det = (Детали)l.Tag;

}

}

private void cbProfile\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbProfile.SelectedIndex == 0)

{

label1.Text = "Диаметр, мм";

label2.Visible = false;

maskedTextBox2.Text = "";

maskedTextBox2.Visible = false;

btnCompute.Enabled = false;

}

if (cbProfile.SelectedIndex == 1)

{

label1.Text = "Толщина, мм";

label2.Visible = true;

maskedTextBox2.Visible = true;

maskedTextBox2.Clear();

btnCompute.Enabled = false;

}

maskedTextBox1.Clear();

maskedTextBox3.Clear();

}

private void listView1\_ItemSelectionChanged(object sender, ListViewItemSelectionChangedEventArgs e)

{

if (listView1.SelectedItems.Count == 0)

btnSelect.Enabled = false;

else

btnSelect.Enabled = true;

}

private void listView1\_ItemActivate(object sender, EventArgs e)

{

ListView lv = (ListView)sender;

foreach (ListViewItem l in lv.SelectedItems)

{

det = (Детали)l.Tag;

}

this.DialogResult = DialogResult.OK;

}

}

public partial class Детали : IComparable

{

public int CompareTo(object obj)

{

Детали a = (Детали)obj;

if (this.НормаРасхода == a.НормаРасхода)

return 0;

else if (this.НормаРасхода > a.НормаРасхода)

return 1;

else return -1;

}

}

}

Класс ItemMaterial.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Peleng

{

public partial class ItemMaterial : UserControl

{

private Материалы material;

private double gabarit, length, rate, weight;

private int n;

ErrorProvider epA = new ErrorProvider();

ErrorProvider epB = new ErrorProvider();

ErrorProvider epX = new ErrorProvider();

public ItemMaterial()

{

InitializeComponent();

maskedTextBox1.ValidatingType = maskedTextBox2.ValidatingType = typeof(uint);

mtbRate.ValidatingType = typeof(float);

}

public int N

{

get

{

return n;

}

set

{

n = value;

}

}

public Материалы Material

{

get

{

return material;

}

set

{

material = value;

}

}

public double Gabarit

{

get

{

return gabarit;

}

set

{

gabarit = value;

}

}

public double Length

{

get

{

return length;

}

set

{

length = value;

}

}

public double Weight

{

get

{

return weight;

}

set

{

weight = value;

}

}

public double Rate

{

get

{

return rate;

}

set

{

rate = value;

}

}

private void btnCompute\_Click(object sender, EventArgs e)

{

computeRate rateForm = new computeRate();

Детали det = new Детали();

PelengEntities pe = new PelengEntities();

var mat = (from m in pe.Материалы

where (m.МатериалID == det.МатериалID)

select m);

if (rateForm.ShowDialog(this) == DialogResult.OK)

{

det = rateForm.Det;

mtbRate.Text = det.НормаРасхода.ToString();

epA.Clear();

epB.Clear();

epX.Clear();

if (det.ШиринаЗаготовки.HasValue)

{

maskedTextBox1.Text = det.ШиринаЗаготовки.ToString();

maskedTextBox2.Text = det.ДлинаЗаготовки.ToString();

maskedTextBox2.Visible = true;

label6.Visible = true;

lbType.Visible = false;

}

else

{

maskedTextBox1.Text = det.ДлинаЗаготовки.ToString();

maskedTextBox2.Visible = false;

label6.Visible = false;

lbType.Visible = true;

lbType.Text = "\u00d8";

}

numericUpDown1.Value = det.ВыходДеталей;

lbGabarit.Text = mat.First().Габарит.ToString();

tbMat.Text = mat.First().НаименованиеМатериала;

}

}

private void btnSave\_Click(object sender, EventArgs e)

{

PelengEntities pe = new PelengEntities();

var det = (from m in pe.Детали

where (m.НомерДетали == this.ParentForm.Text)

select m);

var mat = (from m in pe.Материалы

where (m.НаименованиеМатериала == tbMat.Text)

select m);

if (det.Count() != 0)

{

det.First().ВыходДеталей = (int)numericUpDown1.Value;

det.First().ДлинаЗаготовки = double.Parse(maskedTextBox1.Text);

det.First().НормаРасхода = double.Parse(mtbRate.Text);

det.First().МатериалID = mat.First().МатериалID;

if (maskedTextBox2.Visible)

{

det.First().ДлинаЗаготовки = double.Parse(maskedTextBox2.Text);

det.First().ШиринаЗаготовки = double.Parse(maskedTextBox1.Text);

}

else

{

det.First().ДлинаЗаготовки = double.Parse(maskedTextBox1.Text);

det.First().ШиринаЗаготовки = null;

}

pe.SaveChanges();

}

}

private void ItemMaterial\_Load(object sender, EventArgs e)

{

toolTipMats.SetToolTip(btnMaterialChoose, "Выбор материала из номенклатуры");

if ((length != 0) & (rate != 0) & (gabarit != 0))

{

if (weight != 0)

{

maskedTextBox1.Text = weight.ToString();

maskedTextBox2.Text = length.ToString();

maskedTextBox2.Visible = true;

label6.Visible = true;

lbType.Visible = false;

}

else

{

maskedTextBox1.Text = length.ToString();

maskedTextBox2.Visible = false;

label6.Visible = false;

lbType.Visible = true;

}

lbType.Text = "\u00d8";

mtbRate.Text = rate.ToString();

lbGabarit.Text = gabarit.ToString();

tbMat.Text = material.НаименованиеМатериала;

tbMat.Tag = material;

numericUpDown1.Value = n;

}

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.ParentForm.Close();

}

private void btnMaterialChoose\_Click(object sender, EventArgs e)

{

listMaterial listMats = new listMaterial();

if (listMats.ShowDialog(this) == DialogResult.OK)

{

Материалы m = new Материалы();

m = listMats.Material;

tbMat.Text = m.НаименованиеМатериала;

lbGabarit.Text = m.Габарит.ToString();

if (m.Ширина.HasValue)

{

lbType.Visible = false;

maskedTextBox2.Visible = true;

label6.Visible = true;

}

else

{

lbType.Visible = true;

maskedTextBox2.Visible = false;

label6.Visible = false;

maskedTextBox2.Clear();

}

tbMat.Tag = m;

}

}

private void Textbox\_TextChanged(object sender, EventArgs e)

{

Validate(true);

if ((epA.GetError(maskedTextBox1) == "") & (maskedTextBox2.Visible == true ? epB.GetError(maskedTextBox2) == "" : true) &

(epX.GetError(mtbRate) == "") & (tbMat.TextLength != 0) & (maskedTextBox1.TextLength != 0) &

(maskedTextBox2.Visible == true ? maskedTextBox2.TextLength != 0 : true) & (mtbRate.TextLength != 0))

btnSave.Enabled = true;

else

btnSave.Enabled = false;

}

private void maskedTextBox1\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

TypeValidation(sender, e, epA);

Материалы m = (Материалы)tbMat.Tag;

int weight1, length1, a, b;

if ((epA.GetError(maskedTextBox1) == "") & (m.Ширина.HasValue) & (m.Длина.HasValue))

{

if (m.Длина > m.Ширина)

{

length1 = m.Длина.Value;

weight1 = m.Ширина.Value;

}

else

{

weight1 = m.Длина.Value;

length1 = m.Ширина.Value;

}

a = Int32.Parse(maskedTextBox1.Text);

if (((epB.GetError(maskedTextBox2) == "") | (epB.GetError(maskedTextBox2) == "Превышен габаритный размер!")) & (maskedTextBox2.TextLength != 0))

{

if (a > (b = Int32.Parse(maskedTextBox2.Text)))

{

if (a > length1)

epA.SetError(maskedTextBox1, "Превышен габаритный размер!");

else

epA.Clear();

if (b > weight1)

epB.SetError(maskedTextBox2, "Превышен габаритный размер!");

else

epB.Clear();

}

else if ((epB.GetError(maskedTextBox2) == "") & (a < b))

{

if (b > length1)

epB.SetError(maskedTextBox2, "Превышен габаритный размер!");

else

epB.Clear();

if (a > weight1)

epA.SetError(maskedTextBox1, "Превышен габаритный размер!");

else

epA.Clear();

}

}

else if (a > length1)

epA.SetError(maskedTextBox1, "Превышен габаритный размер!");

}

}

private void TypeValidation(object obj, TypeValidationEventArgs e, ErrorProvider ep)

{

MaskedTextBox mtb = (MaskedTextBox)obj;

if (!e.IsValidInput)

{

ep.SetError(mtb, "Invalid Data Value");

btnSave.Enabled = false;

}

else if ((mtb.Text.Contains('-')) | (mtb.Text.Contains('+')) | (mtb.Text.Equals("0")))

{

btnSave.Enabled = false;

ep.SetError(mtb, "Invalid Data Value");

}

else if (mtb.TextLength > 7)

{

btnSave.Enabled = false;

ep.SetError(mtb, "Слишком длинное значение!");

}

else

ep.Clear();

}

private void maskedTextBox2\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

TypeValidation(sender, e, epB);

Материалы m = (Материалы)tbMat.Tag;

int weight1, length1, a, b;

if ((epB.GetError(maskedTextBox2) == "") & (m.Ширина.HasValue) & (m.Длина.HasValue))

{

if (m.Длина > m.Ширина)

{

length1 = m.Длина.Value;

weight1 = m.Ширина.Value;

}

else

{

weight1 = m.Длина.Value;

length1 = m.Ширина.Value;

}

b = Int32.Parse(maskedTextBox2.Text);

if (((epA.GetError(maskedTextBox1) == "") | (epA.GetError(maskedTextBox1) == "Превышен габаритный размер!")) & (maskedTextBox1.TextLength != 0))

{

if ((a = Int32.Parse(maskedTextBox1.Text)) > b)

{

if (a > length1)

epA.SetError(maskedTextBox1, "Превышен габаритный размер!");

else

epA.Clear();

if (b > weight1)

epB.SetError(maskedTextBox2, "Превышен габаритный размер!");

else

epB.Clear();

}

else if ((epA.GetError(maskedTextBox1) == "") & (a < b))

{

if (b > length1)

epB.SetError(maskedTextBox2, "Превышен габаритный размер!");

else

epB.Clear();

if (a > weight1)

epA.SetError(maskedTextBox1, "Превышен габаритный размер!");

else

epA.Clear();

}

}

else if (b > length1)

epB.SetError(maskedTextBox2, "Превышен габаритный размер!");

}

}

private void mtbRate\_TypeValidationCompleted(object sender, TypeValidationEventArgs e)

{

TypeValidation(sender, e, epX);

}

}

}

Класс listMaterial.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Peleng

{

public partial class listMaterial : Form

{

private SqlConnection con;

private SqlCommand command;

private SqlDataAdapter sda;

private DataTable dt;

private string queryMat;

private Материалы material = new Материалы();

public listMaterial()

{

InitializeComponent();

}

public Материалы Material

{

get

{

return material;

}

}

private void listMaterial\_Load(object sender, EventArgs e)

{

this.маркиМатериалаTableAdapter.Fill(this.pelengDataSet.МаркиМатериала);

rbFull.Checked = true;

}

private void rbFull\_CheckedChanged(object sender, EventArgs e)

{

try

{

if (rbFull.Checked == true)

{

con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Peleng.mdf;Integrated Security=True;User Instance=True");

con.Open();

command = new SqlCommand(@"SELECT НаименованиеМатериала as [Наименование материала], Габарит

FROM Материалы

ORDER by Габарит");

command.Connection = con;

sda = new SqlDataAdapter(command);

dt = new DataTable();

sda.Fill(dt);

BindingSource bs = new BindingSource();

bs.DataSource = dt;

dataGridView1.DataSource = bs;

}

else

{

queryMat = cbMaterial.Text;

con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Peleng.mdf;Integrated Security=True;User Instance=True");

con.Open();

command = new SqlCommand(@"SELECT НаименованиеМатериала as [Наименование материала], Габарит

FROM Материалы, МаркиМатериала

WHERE Материалы.МаркаID=МаркиМатериала.МаркаID and МаркиМатериала.НаименованиеМарки=@queryMat

ORDER by Габарит");

command.Parameters.Add("@queryMat", SqlDbType.VarChar, 20);

command.Parameters["@queryMat"].Value = queryMat;

command.Connection = con;

sda = new SqlDataAdapter(command);

dt = new DataTable();

sda.Fill(dt);

BindingSource bs = new BindingSource();

bs.DataSource = dt;

dataGridView1.DataSource = bs;

rbMark.Checked = true;

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

con.Close();

}

}

private void cbMaterial\_SelectedValueChanged(object sender, EventArgs e)

{

try

{

rbMark.Checked = true;

queryMat = cbMaterial.Text;

con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Peleng.mdf;Integrated Security=True;User Instance=True");

con.Open();

command = new SqlCommand(@"SELECT НаименованиеМатериала as [Наименование материала], Габарит

FROM Материалы, МаркиМатериала

WHERE Материалы.МаркаID=МаркиМатериала.МаркаID and МаркиМатериала.НаименованиеМарки=@queryMat

ORDER by Габарит");

command.Parameters.Add("@queryMat", SqlDbType.VarChar, 20);

command.Parameters["@queryMat"].Value = queryMat;

command.Connection = con;

sda = new SqlDataAdapter(command);

dt = new DataTable();

sda.Fill(dt);

BindingSource bs = new BindingSource();

bs.DataSource = dt;

dataGridView1.DataSource = bs;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

con.Close();

}

}

private void btnChoose\_Click(object sender, EventArgs e)

{

material.НаименованиеМатериала = dataGridView1.CurrentRow.Cells[0].Value.ToString();

PelengEntities pe = new PelengEntities();

var mats = (from m in pe.Материалы

where m.НаименованиеМатериала == material.НаименованиеМатериала

select m);

material = mats.Single();

}

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.RowIndex == -1)

{

}

else

{

btnChoose\_Click(sender, e);

this.DialogResult = DialogResult.OK;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

}

}

}

}

Класс mainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace Peleng

{

public partial class mainForm : Form

{

public mainForm()

{

InitializeComponent();

toolStripStatusLabel1.Text = this.Text;

}

private void specificationToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (Form o in MdiChildren)

{

if (o.Text=="Спецификации номенклатуры")

{

o.Activate();

return;

}

}

specificationForm specForm = new specificationForm();

specForm.MdiParent = this;

specForm.WindowState = FormWindowState.Maximized;

specForm.Show();

}

private void mainForm\_MdiChildActivate(object sender, EventArgs e)

{

if (ActiveMdiChild != null)

toolStripStatusLabel1.Text = ActiveMdiChild.Text;

else

toolStripStatusLabel1.Text = this.Text;

}

private void расположитьПодрядToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.Cascade);

}

private void расположитьГоризонтальноToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileHorizontal);

}

private void расположитьВертикальноToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.LayoutMdi(MdiLayout.TileVertical);

}

private void ведомостьВходимостиToolStripMenuItem\_Click(object sender, EventArgs e)

{

reportItemEnter report = new reportItemEnter();

if (report.ShowDialog(this) == DialogResult.OK)

{

PelengEntities pe = new PelengEntities();

string name;

if (report.Number.EndsWith("0"))

{

var specs = (from m in pe.Сборки

where m.НомерСборки == report.Number

select m).First();

name = specs.Наименование;

}

else

{

var specs = (from m in pe.Детали

where m.НомерДетали == report.Number

select m).First();

name = specs.Наименование;

}

List<string> fullList = new List<string>();

Dictionary<string, int> dictInAssemblies = new Dictionary<string, int>();

Excel.Application application = new Excel.Application();

Object missing = Type.Missing;

//добавили книгу

application.Workbooks.Add(missing);

Excel.Worksheet sheet = (Excel.Worksheet)application.ActiveSheet;

((Excel.Range)sheet.Columns["A"]).ColumnWidth = 3.57;

((Excel.Range)sheet.Columns["B"]).ColumnWidth = 45.71;

((Excel.Range)sheet.Columns["C"]).ColumnWidth = 9.86;

((Excel.Range)sheet.Columns["D"]).ColumnWidth = 3.86;

((Excel.Range)sheet.Columns["E"]).ColumnWidth = 21.86;

((Excel.Range)sheet.Columns["F"]).ColumnWidth = 11;

addExcelCell(sheet, 1, 1, System.DateTime.Now.ToString(), "Times New Roman", 12, true);

addExcelCell(sheet, 3, 1, "Ведомость входимости применяемости", "Times New Roman", 16, true);

addExcelCell(sheet, 4, 1, report.Number +" "+ name, "Times New Roman", 16, true);

(sheet.Cells[3, 1] as Excel.Range).Borders[Excel.XlBordersIndex.xlEdgeBottom].LineStyle = Excel.XlLineStyle.xlDouble;

application.Visible = true;

}

}

private void addExcelCell(Excel.Worksheet sheet, int Column, int Cell, string Text, string Font, int FontSize, bool FontBold)

{

//вписываем текст

sheet.Cells[Column, Cell] = Text;

//жирность

(sheet.Cells[Column, Cell] as Excel.Range).Font.Bold = FontBold;

(sheet.Cells[Column, Cell] as Excel.Range).Font.Name = Font;

(sheet.Cells[Column, Cell] as Excel.Range).Font.Size = FontSize;

//(sheet.Cells[Column, Cell] as Excel.Range).NumberFormat = "@";

}

}

}

Класс reportItemEnter.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Peleng

{

public partial class reportItemEnter : Form

{

private string number;

public reportItemEnter()

{

InitializeComponent();

comboBox1.SelectedIndex = 0;

if (maskedTbNumber.CanFocus)

maskedTbNumber.Focus();

maskedTbNumber.Select(0, 0);

maskedTbNumber.Mask = "0000.00.00.000";

maskedTbNumber.PromptChar = '\_';

}

public string Number

{

get

{

return number;

}

}

public int TypeReport

{

get

{

return comboBox1.SelectedIndex;

}

}

private void maskedTbNumber\_TextChanged(object sender, EventArgs e)

{

if (maskedTbNumber.MaskFull)

{

btnOK.Enabled = true;

}

else

btnOK.Enabled = false;

}

private void btnOK\_Click(object sender, EventArgs e)

{

number = maskedTbNumber.Text.Replace(',', '.');

PelengEntities pe = new PelengEntities();

if (number.EndsWith("0"))

{

var specs = (from m in pe.Сборки

where m.НомерСборки == number

select m);

if (specs.Count() == 0)

{

MessageBox.Show("Спецификация с таким номером не существует!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.DialogResult = DialogResult.None;

}

}

else

{

var specs = (from m in pe.Детали

where m.НомерДетали == number

select m);

if (specs.Count() == 0)

{

MessageBox.Show("Спецификация с таким номером не существует!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.DialogResult = DialogResult.None;

}

}

}

}

}

Класс specificationEnterNumber.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Peleng

{

public partial class specificationEnterNumber : Form

{

public specificationEnterNumber()

{

InitializeComponent();

if (maskedTbNumber.CanFocus)

maskedTbNumber.Focus();

maskedTbNumber.Select(0, 0);

maskedTbNumber.Mask = "0000.00.00.000";

maskedTbNumber.PromptChar = '\_';

}

private void btnOK\_Click(object sender, EventArgs e)

{

specificationWork specWork = new specificationWork(maskedTbNumber.Text.Replace(',','.'));

specWork.MdiParent = this.Owner;

specWork.Show();

}

private void maskedTbNumber\_TextChanged(object sender, EventArgs e)

{

if (maskedTbNumber.MaskFull)

{

btnOK.Enabled = true;

}

else

btnOK.Enabled = false;

}

}

}

Класс specificationForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Peleng

{

public partial class specificationForm : Form

{

private SqlConnection con;

private SqlCommand command;

private SqlDataAdapter sda;

private DataTable dt;

public specificationForm()

{

InitializeComponent();

connectDB();

}

public void connectDB()

{

try

{

con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Peleng.mdf;Integrated Security=True;User Instance=True");

con.Open();

command = new SqlCommand(@"SELECT Сборки.НомерСборки AS Спецификации, Наименование

FROM Сборки

union all

SELECT Детали.НомерДетали AS Спецификации, Наименование

FROM Детали

order by Спецификации");

command.Connection = con;

sda = new SqlDataAdapter(command);

dt = new DataTable();

sda.Fill(dt);

dataGridView1.DataSource = dt;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

con.Close();

}

}

private void specificationForm\_FormClosed(object sender, FormClosedEventArgs e)

{

}

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.RowIndex == -1)

{

}

else

{

foreach (Form o in this.MdiParent.MdiChildren)

{

if (o.Text == dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString())

{

o.Activate();

return;

}

}

specificationWork specWork = new specificationWork(dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString());

specWork.MdiParent = this.ParentForm;

specWork.WindowState = FormWindowState.Maximized;

specWork.Show();

}

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

}

}

private void tsCreate\_Click(object sender, EventArgs e)

{

specificationEnterNumber specNumber = new specificationEnterNumber();

specNumber.ShowDialog(this.ParentForm);

}

private void specificationForm\_Activated(object sender, EventArgs e)

{

connectDB();

}

}

}

Класс specificationWork.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Peleng

{

public partial class specificationWork : Form

{

private string number;

private string currentWay;

private string currentName;

public specificationWork()

{

InitializeComponent();

}

public specificationWork(string number)

: this()

{

this.number = this.Text = tbNumber.Text = number;

PelengEntities pe = new PelengEntities();

var det = (from m in pe.Детали

where (m.НомерДетали == number)

select m);

var sb = (from m in pe.Сборки

where (m.НомерСборки == number)

select m);

int way = 0;

if (det.Count() != 0)

{

tbName.Text = currentName = det.First().Наименование;

way = det.First().МаршрутID;

btnSave.Enabled = true;

this.DialogResult = DialogResult.OK;

}

else if (sb.Count() != 0)

{

tbName.Text = currentName = sb.First().Наименование;

way = sb.First().МаршрутID;

btnSave.Enabled = true;

this.DialogResult = DialogResult.OK;

}

else

{

btnSave.Enabled = false;

this.DialogResult = DialogResult.OK;

}

if (way != 0)

{

var ways = (from m in pe.Маршруты

where (m.МаршрутID == way)

select m).ToList();

foreach (var w in ways)

{

tbWay.Text += w.НомерПодразделения + "-";

}

tbWay.Text = currentWay = tbWay.Text.Remove(tbWay.Text.Length - 1);

}

var grouped = from p in pe.Маршруты

orderby p.МаршрутID

group p by p.МаршрутID into grp

select new { ID = grp.Key, objWay = grp };

Dictionary<int, string> waysDiction = new Dictionary<int, string>();

foreach (var w in grouped.ToList())

{

string s = "";

foreach (var r in w.objWay)

{

s += r.НомерПодразделения + "-";

}

s = s.Remove(s.Length - 1);

waysDiction.Add(w.ID, s);

}

tbWay.AutoCompleteMode = AutoCompleteMode.SuggestAppend;

tbWay.AutoCompleteSource = AutoCompleteSource.CustomSource;

tbWay.AutoCompleteCustomSource.AddRange(waysDiction.Values.ToArray());

}

private void tabSpecification\_DrawItem(object sender, DrawItemEventArgs e)

{

Graphics g = e.Graphics;

Brush \_textBrush;

// Get the item from the collection.

TabPage \_tabPage = tabSpecification.TabPages[e.Index];

// Get the real bounds for the tab rectangle.

Rectangle \_tabBounds = tabSpecification.GetTabRect(e.Index);

if (e.State == DrawItemState.Selected)

{

// Draw a different background color, and don't paint a focus rectangle.

\_textBrush = new SolidBrush(Color.Coral);

g.FillRectangle(Brushes.Gray, e.Bounds);

}

else

{

\_textBrush = new System.Drawing.SolidBrush(e.ForeColor);

e.DrawBackground();

}

// Use our own font.

Font \_tabFont = new Font("Arial", (float)14.0, FontStyle.Bold, GraphicsUnit.Pixel);

// Draw string. Center the text.

StringFormat \_stringFlags = new StringFormat();

\_stringFlags.Alignment = StringAlignment.Center;

\_stringFlags.LineAlignment = StringAlignment.Center;

g.DrawString(\_tabPage.Text, \_tabFont, \_textBrush, \_tabBounds, new StringFormat(\_stringFlags));

}

private void tabSpecification\_SelectedIndexChanged(object sender, EventArgs e)

{

if (tabSpecification.SelectedTab == tabAssembly)

{

if (number.EndsWith("0"))

входящиеСборкиTableAdapter.Fill(pelengDataSet.ВходящиеСборки, number);

else

{

tsBtnAdd.Enabled = false;

tsBtnDelete.Enabled = false;

tsBtnEdit.Enabled = false;

btnSaveAssembly.Enabled = false;

}

}

if (tabSpecification.SelectedTab == tabMain)

{

tbNumber.Text = number;

}

if (tabSpecification.SelectedTab == tabMaterial)

{

if (!number.EndsWith("0"))

{

ItemMaterial im = new ItemMaterial();

im.Dock = System.Windows.Forms.DockStyle.Fill;

PelengEntities pe = new PelengEntities();

var det = (from m in pe.Детали

where (m.НомерДетали == number)

select m);

foreach (Детали d in det)

{

im.Length = d.ДлинаЗаготовки;

im.Rate = d.НормаРасхода;

im.Material = d.Материалы;

im.Gabarit = d.Материалы.Габарит;

im.N = d.ВыходДеталей;

if (d.ШиринаЗаготовки.HasValue)

im.Weight = d.ШиринаЗаготовки.Value;

}

tabMaterial.Controls.Add(im);

}

}

}

private void specificationWork\_Load(object sender, EventArgs e)

{

tabSpecification\_SelectedIndexChanged(sender, e);

}

private void btnExit\_Click(object sender, EventArgs e)

{

Close();

}

private void btnSave\_Click(object sender, EventArgs e)

{

if (tbWay.Text.StartsWith("-"))

tbWay.Text = tbWay.Text.Remove(0,1);

if (tbWay.Text.EndsWith("-"))

tbWay.Text = tbWay.Text.Remove(tbWay.TextLength - 1);

PelengEntities pe = new PelengEntities();

var det = (from m in pe.Детали

where (m.НомерДетали == number)

select m);

var sb = (from m in pe.Сборки

where (m.НомерСборки == number)

select m);

List<string> listWays = (from m in pe.Подразделения

select m.НомерПодразделения).ToList();

Dictionary<int,string> waysDiction = new Dictionary<int,string>();

var grouped = from p in pe.Маршруты

orderby p.МаршрутID

group p by p.МаршрутID into grp

select new { ID = grp.Key, objWay = grp };

foreach (var w in grouped.ToList())

{

string s="";

foreach (var r in w.objWay)

{

s += r.НомерПодразделения + "-";

}

s = s.Remove(s.Length - 1);

waysDiction.Add(w.ID, s);

}

int numberWay=0;

foreach (int n in waysDiction.Keys)

{

if (tbWay.Text == waysDiction[n])

{

numberWay = n;

break;

}

}

string[] subWays = tbWay.Text.Split('-');

if (numberWay == 0)

{

int z=1;

foreach (string s in subWays)

pe.Маршруты.AddObject(new Маршруты { МаршрутID = waysDiction.Count+1, ПорядковыйНомер = z++, НомерПодразделения = s });

pe.SaveChanges();

waysDiction.Add(numberWay=waysDiction.Count + 1, tbWay.Text);

}

tbWay.AutoCompleteCustomSource.AddRange(waysDiction.Values.ToArray());

if (det.Count() != 0)

{

det.First().Наименование = tbName.Text;

det.First().МаршрутID = numberWay;

pe.SaveChanges();

}

else if (sb.Count() != 0)

{

sb.First().Наименование = tbName.Text;

sb.First().МаршрутID = numberWay;

pe.SaveChanges();

}

else if (number.EndsWith("0"))

{

Сборки assembly = new Сборки();

assembly.Наименование = tbName.Text;

assembly.МаршрутID = numberWay;

assembly.НомерСборки = number;

pe.Сборки.AddObject(assembly);

pe.SaveChanges();

}

currentName = tbName.Text;

currentWay = tbWay.Text;

this.DialogResult = DialogResult.OK;

}

private void tsBtnAdd\_Click(object sender, EventArgs e)

{

PelengEntities pe = new PelengEntities();

var currentAssembly = (from m in pe.Сборки

where m.НомерСборки == number

select m);

if (currentAssembly.Count() == 0)

{

if (MessageBox.Show("Текущая спецификация не внесена в базу.\nСохранить данную спецификацию?", "Внимание!",

MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.No)

return;

else

{

btnSave\_Click(sender, e);

}

}

addAssembly add = new addAssembly(number);

if (add.ShowDialog(this) == DialogResult.OK)

{

bool exist = false;

foreach (DataGridViewRow dr in dataGridView1.Rows)

{

if (dr.Cells[0].Value.ToString() == add.AddAssembly.НомерВхСборки)

{

exist = true;

break;

}

}

if (!exist)

pelengDataSet.ВходящиеСборки.Rows.Add(add.AddAssembly.НомерВхСборки, add.AddAssembly.Количество, number);

}

}

private void dataGridView1\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.RowIndex == -1)

{

}

else

{

tsBtnEdit\_Click(sender, e);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

}

}

private void btnSaveAssembly\_Click(object sender, EventArgs e)

{

входящиеСборкиTableAdapter.Update(pelengDataSet);

}

private void tsBtnDelete\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedCells.Count != 0)

dataGridView1.Rows.RemoveAt(dataGridView1.SelectedCells[0].RowIndex);

}

private void tsBtnEdit\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedCells.Count != 0)

{

int index = dataGridView1.SelectedCells[0].RowIndex;

string current = dataGridView1.Rows[index].Cells[0].Value.ToString();

int k = (int)dataGridView1.Rows[index].Cells[1].Value;

addAssembly add = new addAssembly(number, current, k);

PelengEntities pe = new PelengEntities();

if (add.ShowDialog(this) == DialogResult.OK)

{

if (add.AddAssembly.НомерВхСборки == current && add.AddAssembly.Количество == k)

return;

if (add.AddAssembly.НомерВхСборки == current)

{

dataGridView1.Rows[index].Cells[1].Value = add.AddAssembly.Количество;

}

else

{

bool exist = false;

int newIndex = 0;

foreach (DataGridViewRow dr in dataGridView1.Rows)

{

if (dr.Cells[0].Value.ToString() == add.AddAssembly.НомерВхСборки)

{

exist = true;

newIndex = dr.Index;

}

}

if (!exist)

{

dataGridView1.Rows.RemoveAt(index);

pelengDataSet.ВходящиеСборки.Rows.Add(add.AddAssembly.НомерВхСборки, add.AddAssembly.Количество, number);

}

else

{

dataGridView1.Rows[newIndex].Cells[1].Value = add.AddAssembly.Количество;

dataGridView1.Rows.RemoveAt(index);

}

}

}

}

}

private void specificationWork\_FormClosing(object sender, FormClosingEventArgs e)

{

}

private void tbName\_TextChanged(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(tbName.Text))

{

epName.SetError(tbName, "Не задано наименование!");

}

else if (tbName.TextLength > 80)

{

epName.SetError(tbName, "Слишком длинное наименование!");

}

else

epName.Clear();

Validate();

}

private void tbWay\_TextChanged(object sender, EventArgs e)

{

if (tbWay.TextLength > 84)

{

epWay.SetError(tbWay, "Слишком длинный маршрут!");

Validate();

return;

}

string [] subWays = tbWay.Text.Split('-');

bool rightWay=false;

PelengEntities pe = new PelengEntities();

List<string> listWays = (from m in pe.Подразделения

select m.НомерПодразделения).ToList();

foreach (string s in subWays)

{

if (s.Length == 0)

continue;

if (listWays.Contains(s))

rightWay = true;

else

{

rightWay = false;

break;

}

}

if (rightWay)

{

epWay.Clear();

}

else

{

epWay.SetError(tbWay, "Ошибка в кодировке подразделения!");

}

Validate();

}

private void tbWay\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar == 'К') | (e.KeyChar == 'к'))

{

e.KeyChar = 'К';

return;

}

if ((tbWay.TextLength > 0) & (tbWay.SelectionStart>0))

if (tbWay.Text.ElementAt(tbWay.SelectionStart - 1) == '-' & (e.KeyChar == '-'))

{

e.Handled = true;

}

if ((tbWay.TextLength > 0) & (tbWay.SelectionStart < tbWay.TextLength))

if (tbWay.Text.ElementAt(tbWay.SelectionStart) == '-' & (e.KeyChar == '-'))

{

e.Handled = true;

return;

}

if (!Char.IsNumber(e.KeyChar) & (e.KeyChar != '-') & (e.KeyChar != 8))

e.Handled = true;

}

private void tabMain\_Validating(object sender, CancelEventArgs e)

{

if (epName.GetError(tbName) == "" & epWay.GetError(tbWay) == "")

btnSave.Enabled = true;

else

btnSave.Enabled = false;

}

}

}