Московский Государственный Институт Электронной Техники

Технический Университет

Факультет: МПиТК

Кафедра: ИПОВС

Пояснительная записка

*к дипломному проекту*

Тема: Программа регистрации процесса производства для автоматизированной системы управления предприятием электронной промышленности

Дипломант Ляшко М.А. //

Руководитель проекта Брусникин Г.Н. //

Консультант Брусникин Г.Н. //

Консультант по технологической части Брусникин Г.Н. //

Консультант по организационно-

экономической части Пискунова Н.Н. //

Консультант по производственно-

экологической части Никулина И.М. //

Содержание

Введение

1. Состояние и тенденции развития АСУ ПП

1.1 Опыт отечественной науки - ситуационные системы управления

1.1.2 Применения ситуационного управления

1.2 Manufacturing executing systems - перспективные автоматизированные системы управления производственными процессами

1.2.1 Общая характеристика MES-систем

1.2.2 Блоки MES-систем

1.2.3 Достоинства MES-систем

1.2.4 Перспективы развития MES-систем

1.3 Выводы

2. Постановка задачи

2.1 Введение в предметную область. Анализ работы цехов ОАО Ангстрем

2.1.1 Особенности технологического цикла производства партий пластин

2.1.1.1 Изделия, партии, технологические маршруты, маршрутные листы

2.1.1.2 Регистрация процесса производства партий пластин

2.1.1.3 Незавершенное производство

2.1.2 Общая схема управления

2.1.3 Задержки в транспорте и обработке партий пластин

2.2 Анализ требований к системе

2.3 Рекомендации к разработке автоматизированной системы управления производственным процессом

2.4 Предлагаемая архитектура автоматизированной системы управления производственным процессом

2.4.1 Общая схема управления

2.4.2 Взаимодействие программы регистрации и персонала

2.4.3 Реализация

2.4.4 Управление обработкой партий

2.4 4.1 Понятие дисциплины очереди

2.4.4.2 Предварительная схема дисциплины очереди

2.5 Техническое задание на дипломный проект

2.6 Выбор платформы и инструмента разработки программы

3. Разработка алгоритмов и программ

3.1 Этапы объектно-ориентированного подхода

3.2 Характерные свойства системы

3.3 Выбранные объекты

3.4 Алгоритм выполнения технологической операции

3.5 Алгоритм формирования дисциплины очереди

3.6 Использование библиотеки Microsoft Foundation Classes

4. Объектно-ориентированное программирование

4.1 Введение

4.2 Понятие жизненного цикла программного обеспечения

4.3 Модели жизненного цикла программного обеспечения

4.4 Анализ

4.5 Проектирование

4.5.1 Методы проектирования

4.5.2 Объектно-ориентированная модель

4.5.3 Процесс объектно-ориентированного проектирования

4.6 Эволюция

4.7 Сопровождение

4.8 Заключение

5. Методика отладки и результаты работы программы

5.1 Особенности тестирования программных продуктов

5.2 Типичный процесс тестирования программного обеспечения

5.3 Особенности задачи в приложении к тестированию программ

5.3.1 Особенности среды программирования

5.3.2 Основные факторы, влияющие на надежность разрабатываемой системы

5.3.2.1 Контроль структуры программы

5.3.2.2 Контроль чтения и записи переменных

5.4 Результаты работы программы

6. Организационно-экономическая часть

6.1 Введение

6.2 Сетевая модель, ее основные элементы, правила построения

6.3 Расчет параметров сетевой модели

6.4 Основы оптимизации сетевого графика

6.5 Анализ и оптимизация сетевой модели

6.6 Выводы

7. Производственная и экологическая безопасность

7.1 Введение

7.2 Рабочее место программиста

7.3 Вредные факторы на рабочем месте программиста и пользователя ЭВМ

7.4 Нерациональное освещение

7.5 Расчет общего освещения

7.6 Электроопасность

7.7 Требования по пожарной безопасности

7.8 Меры по снижению уровня шума

7.9 Защита от вредных излучений

7.10 Параметры микроклимата в машинном зале

7.11 Психофизиологические факторы

7.12 Планировка рабочего места программиста и организация работы с компьютером

7.13 Выводы

Заключение

Список использованной литературы

Приложение

## Введение

Данный дипломный проект выполнен на актуальную для предприятий, производящих полупроводниковые кристаллы, тему, тесно связанную с реальными планами производства, решая практическую задачу по разработке и внедрению программного обеспечения автоматизированного управления производственным процессом. По своему содержанию дипломный проект соответствует современному уровню науки и техники.

Современное полупроводниковое производство представляет собой сложный организационно-технологический комплекс, который включает в себя одновременную обработку многих сотен партий полупроводниковых пластин, проходящих по различным сложным, имеющим циклический характер, технологическим маршрутам с участием большого числа специализированных единиц оборудования, расположенных на различных производственных участках. Для эффективной работы производства нужно обеспечить управление транспортом и обработкой партий пластин, включающее учет технологически необходимых времен обработки и возможных временных задержек, учет приоритетов изготовления, мониторинг качества и сбор статистики, учет реставраций, связанных с исправление возможных несоответствий, учет возможных сбоев в работе технологического оборудования и т.п.

Система управления процессом производства должна быть построена таким образом, чтобы обеспечить разумный компромисс между желанием максимально загрузить имеющееся оборудование (что неизбежно приводит к возникновению очередей на обслуживание при прохождении партий полупроводниковых пластин по технологическому маршруту) и необходимостью обеспечить короткое время выполнения заказа (для чего необходимо, чтобы изготавливаемая партия пластин как можно меньше простаивала в очередях).

Система управления производством должна быть оперативной: для типичного цеха необходимо принимать порядка 105 элементарных решений о транспорте партий и порядка 109 решений по обработке пластин в месяц. Оперативность системы управления подразумевает, что она должна базироваться на современных компьютерно-сетевых технологиях, а сложность решаемых задач накладывает требование "интеллектуальности" такой системы управления.

В процессе решения задачи создания программного продукта детально были рассмотрены особенности разработки программного обеспечения для промышленных программных продуктов. Они применяются для решения самых разных задач, таких, например, как системы с обратной связью, которые управляют или сами управляются событиями физического мира и для которых ресурсы времени и памяти ограничены; системы управления и контроля за реальными процессами (например, диспетчеризация воздушного или железнодорожного транспорта). Системы подобного типа обычно имеют большое время жизни.

В технологической части разработки программных систем и программной документации освещены этапы решения задачи на ЭВМ, принципы тестирования программ и их отладка. Целый раздел посвящен вопросам надежности программного обеспечения.

В организационно-экономическом разделе рассчитан сетевой график разработки автоматизированной системы управления, определен критический путь, резервы времени событий и работ, проведена оптимизация сетевого графика по количеству исполнителей с целью уменьшения затрат на разработку системы.

В производственно-экологическом разделе организации рабочего места программиста и пользователя ЭВМ рассмотрены вредные факторы, присутствующие на рабочем месте, психофизиологические факторы, требования к освещенности помещения, шуму, электро- и пожаробезопасности, спланировано само рабочее место и организация работы программиста с компьютером.

Специальная часть:

Программа регистрации процесса производства для автоматизированной системы управления.

## 1. Состояние и тенденции развития АСУ ПП

## 1.1 Опыт отечественной науки - ситуационные системы управления

В 1970-1980-х годах в отечественной прикладной науке возникло мощное направление исследований - разработка систем ситуационного управления [1-4]. Функционировал постоянный межотраслевой научно-технический семинар, разрабатывались теоретические и практические методы ситуационного управления. При разработке систем ситуационного управления активно использовались теоретические методы искусственного интеллекта [5], теории автоматов [6], нечеткой логики [7,8] и т.п.

1.1.1 Общие принципы ситуационного управления

Охарактеризуем кратко общие принципы работы ситуационного управления [1]. Общая схема ситуационного управления иллюстрируется Рис.1. Описание текущей ситуации, сложившейся на объекте управления, дается на вход *Анализатора*. Его задача состоит в оценке сообщения и определении необходимости вмешательства системы управления в процесс, протекающий в объекте управления. Если текущая ситуация не требует такого вмешательства, то *Анализатор* не передает ее на дальнейшую обработку. В противном случае описание текущей ситуации поступает в *Классификатор*. Используя информацию, хранящуюся в нем, *Классификатор* относит текущую ситуацию к одному или нескольким классам, которым соответствуют одношаговые решения. Эта информация подается в *Коррелятор*, в котором хранятся все *логико-трансформационные правила* (*ЛТП*). Список *ЛТП* задает возможности системы управления воздействовать на процессы, протекающие в объекте управления. *Коррелятор* определяет то *ЛТП*, которое должно быть использовано. Если такое правило единственное, то оно выдается для использования. Если же таких правил несколько, то выбор лучшего из них производится после обработки предварительных решений в *Экстраполяторе*, после чего выдается решение о воздействии на объект. Если *Коррелятор* или *Классификатор* не могут принять решения по поступившему описанию текущей ситуации, то срабатывает *Блок случайного выбора* и выбирается одно из воздействий, оказывающих не слишком большое влияние на объект.

Такие системы управления должны быть неизбежно открытыми. Они должны иметь возможность корректировать свои знания об объекте и методах управления им. В работе такой системы управления имеется два этапа:

1) этап обучения и настройки и 2) этап работы. На первом этапе собираются и обобщаются данные от технологов и формируются классы ситуаций и *ЛТП*. На втором этапе неизбежно проводится дообучение системы управления.

Рис.1. Общая схема ситуационного управления

Объект

управления

Система

ситуационного

управления

Описание

текущей

ситуации

Анализатор

Классификатор

Коррелятор

Блок случайного выбора

Экстраполятор

## 1.1.2 Применения ситуационного управления

Принципы ситуационного управления нашли широкую сферу применения. Приведем некоторые примеры практических разработок, в которых использовались принципы ситуационного управления [1]:

Оперативное диспетчерское управление погрузочно-разгрузочными работами в морском порту (Одесса, Калининград).

Оперативное управление перемещением по буровым специальных установок для производства тампонажных работ и буровых установок по пунктам бурения (Грозный).

Комплекс задач АСУ гражданской авиации (Рига).

Оперативное управление производственными участками на приборостроительных предприятиях (Устинов).

Оперативная диагностика заболеваний и назначение лечения (Баку).

В целом принципы ситуационного управления близки к задачам управления производством партий в микроэлектронном технологическом процессе, и естественно использовать сложившиеся в ситуационном управлении методы и подходы при разработке системы управления производством партий в ОАО Ангстрем.

Прототипами системы управления производством партий в ОАО Ангстрем могут быть активно разрабатываемые в настоящее время за рубежом перспективные автоматизированные системы управления производственными процессами, Manufacturing executing systems (фирма Consilium и др.) [9,10] и системы диспетчерского управления и сбора данных (SCADA - системы). Эти системы кратко характеризуются в следующих разделах.

## 1.2 Manufacturing executing systems - перспективные автоматизированные системы управления производственными процессами

## 1.2.1 Общая характеристика MES-систем

MES-системы возникли из потребности оптимизации сложных производственных процессов, таких как производство микроэлектронных интегральных схем, полупроводниковое производство, производственные процессы в фармацевтической, текстильной промышленности, в производстве продуктов питания.

MES-системы представляют собой компьютерно-сетевые архитектуры (Рис.2,3), способствующие эффективному управлению производственным процессом [9]. MES-системы предоставляют информацию, которая обеспечивает оптимизацию производства продукции от стадии заказа до выпуска конечной продукции. Используя точный сбор текущих данных, MES-система динамически отслеживает весь производственный процесс: инициирует операции, при необходимости оперативно вмешивается в процесс и ведет сбор данных об этом процессе. Функционирование MES-системы обеспечивает оперативное отслеживание постоянно меняющихся условий и устранение лишних затрат. MES-система осуществляет необходимое информационное обеспечение для управления производством в интерактивном режиме.

MES-системы разрабатываются рядом фирм. Фирмы объединены в ассоциацию MESA. Эти фирмы изготовляют различные наборы программных продуктов, в соответствии с требованиями предприятий-заказчиков.

## 

## 1.2.2 Блоки MES-систем

Различают 11 типов блоков, из которых может быть изготовлена конкретная MES система для того или иного производства. Перечислим эти блоки:

Составление расписания операций (Operations/Detail Scheduling) - установление временной последовательности действий для рассматриваемого производственного процесса на данном предприятии при заданных ресурсах. Составление расписания подразумевает определенную оптимизацию производственного процесса;

Распределение ресурсов и слежение за ресурсами (Resource Allocation and Status) - распределение заданий персоналу, машинам и установкам и отслеживание того, что они делают сейчас, и того, что они только что сделали;

Диспетчеризация элементов производства (Dispatching Production Units) - подача команд для транспорта материалов или заказов на определенные участки завода для того, чтобы начать какие-либо процессы или действия;

Управление документооборотом (Document Control) - управление информацией и распределение информации о продуктах, процессах, установках и т.д.;

Отслеживание генеалогии продуктов (Product Tracking and Genealogy) - мониторинг прогресса развития создаваемых продуктов и партий, осуществляемый с целью отслеживания полной истории изготавливаемых на заводе продуктов;

Анализ эффективности производства (Performance Analysis) - сравнение результатов завода с метриками, устанавливаемыми корпорацией, потребителями и организациями, регулирующими рыночные отношения;

Управление персоналом (Labor Management) - управление использованием персонала с точки зрения квалификации, распределения обязанностей и потребностей производства;

Управление оборудованием (Maintenance Management) - планирование и выполнение работ по поддержке работоспособности установок и другого оборудования в соответствии с производственными целями завода;

Управление текущими производственными процессами (Process Management) - управление потоком работ на заводе в соответствии запланированными и текущими процессами;

Управление качеством (Quality Management) - отслеживание и анализ характеристик продуктов и сопоставление их с желаемым идеалом;

Сбор и обработка данных (Data Collection/Acquisition) - мониторинг, обработка и упорядочивание данных о процессах, материалах и операциях, получаемых от персонала, машин и систем управления.

## 1.2.3 Достоинства MES-систем

Практика применения MES-систем демонстрирует следующие их достоинства:

уменьшение длительности производственного цикла, в среднем на 45%;

сокращение времени ввода (entry time), на 75% или больше;

уменьшение незавершенного производства (НЗП), на 24%;

уменьшение бумажного документооборота между сменами, в среднем на 61%;

сокращение времени проектирования (lead time), в среднем на 27%;

уменьшение работы с бумажной документацией и потерь на копировальные работы, в среднем на 56%;

уменьшение дефектности производимой продукции, в среднем на 18%.

## 

## 1.2.4 Перспективы развития MES-систем

Перспективы развития MES-систем связаны с общими тенденциями развития компьютерных сетей. Программные продукты MES-систем пишутся на объектно-ориентированных языках. Имеется определенная тенденция к использованию MES-систем в географически распределенных предприятиях (при этом производитель становится как бы ближе к заказчику и к субподрядчикам) на основе использования передовых достижений компьютерно-сетевых технологий (Рис.2,3). В США сформирован поддерживаемый государством консорциум Solution for MES-Adaptable Replicable Technology (NIIIP/SMART), который предназначен для разработки стандартов для создания инфраструктуры распределенных предприятий будущего.

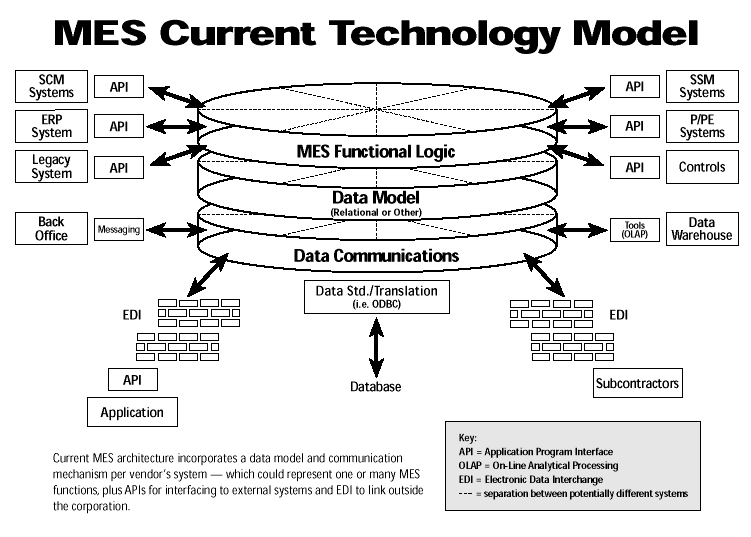


Рис.2. Современная компьютерно-сетевая архитектура MES - системы

API - интерфейс прикладного программирования.

Application - пользовательское приложение.

Data Model - модель представления данных.

Data Communications - передача данных.

Database - база данных.

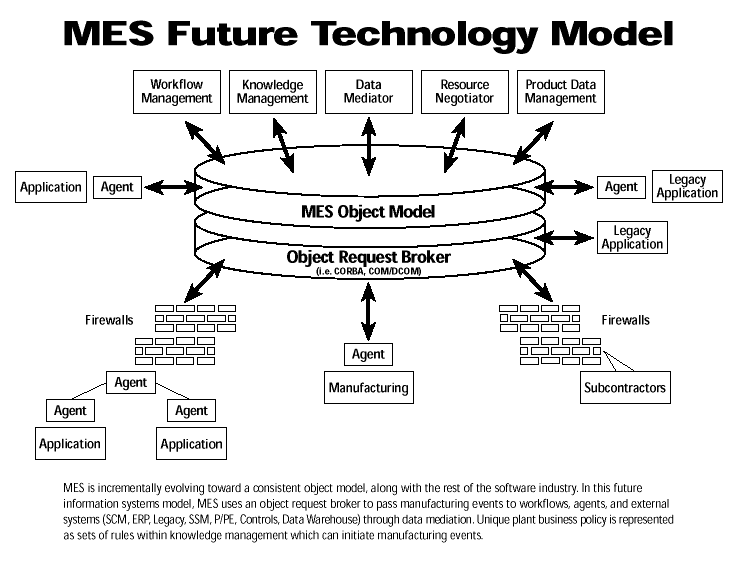


Рис.3. Схема, иллюстрирующая перспективные тенденции развития MES-систем

Object Model - объектная модель.

Object Request Broker - обработчик запросов.

FireWall - система защиты от несанкционированного доступа.

API - интерфейс прикладного программирования.

Application - пользовательское приложение.

## 1.3 Выводы

Современное микроэлектронное производство представляет собой сложный процесс. На уровне отдельного цеха этот процесс включает в себя обработку партий полупроводниковых пластин на установках, которые расположены на различных участках. Необходима транспортировка партий между участками, необходим учет времен обработки партий на установках, возможных временных задержек обработки, необходим контроль качества партий, исправление дефектов обработки, учет возможных выходов из строя оборудования.

В настоящее время интенсивно развиваются компьютеризированные системы управления производственными процессами. Более того, предприятия, не использующие современные технологии, рискуют оказаться в роли "вечно отсталых" и, в конечном итоге, могут не выдержать рыночной конкуренции. Поэтому можно с уверенностью сказать: разработка АСУПП на ОАО "Ангстрем" необходима. При разработке такой системы целесообразно использовать подходы, методы и принципы построения ситуационных систем управления, принципы создания современных компьютерно-сетевых MES-систем и SCADA-систем.

## 2. Постановка задачи

Предлагается схема системы управления производством партий интегральных схем на предприятии электронной промышленности. Предложена общая структура системы управления транспортом и обработкой партий полупроводниковых пластин в цехе. Разработана объектная модель компьютерно-сетевой системы управления транспортом и обработкой партий.

## 2.1 Введение в предметную область. Анализ работы цехов ОАО Ангстрем

В этом разделе производится анализ существующей системы управления производством партий ОАО Ангстрем.

## 2.1.1 Особенности технологического цикла производства партий пластин

Охарактеризуем особенности технологического процесса производства партий. Предварительно введем понятия (изделия, партии, технологические маршруты, маршрутные листы), которые потребуются при описании производственного процесса.

## 2.1.1.1 Изделия, партии, технологические маршруты, маршрутные листы

*Изделие* - полупроводниковая микросхема, выполняющая заданные разработчиком функции. Каждое изделие характеризуется индивидуальным технологическим маршрутом, который заранее прорабатывается.

*Партия* - группа пластин, помещенных в специальный контейнер, имеющая номер и собственный маршрутный лист. Партия в полупроводниковом производстве обрабатывается и транспортируется в соответствии маршрутным листом. Все множество партий можно разбить на группы по принадлежности к определенному изделию (Рис.4). Принадлежность партии к определенному изделию находит отражение в номере. Номер партии состоит из двух частей: первая часть - номер изделия, вторая - порядковый номер в пределах данного изделия.

*Технологический маршрут* - последовательность операций, при выполнении которой на поверхности кремниевой пластины формируется заданная структура. Каждое изделие характеризуется индивидуальным технологическим маршрутом. Прохождение партии по маршруту отслеживается в настоящей системе управления с помощью маршрутного листа.

*Маршрутный лист* - документ, неразрывно связанный с партией. Маршрутный лист представляет собой таблицу со следующими атрибутами: номер операции, название операции, время поступления партии на участок, параметры режима обработки. На маршрутном листе, закрепляемом за партией, ставится номер партии. Каждой партии соответствует свой маршрутный лист, который прикреплен к коробке с пластинами. На протяжении всего периода существования партии маршрутный лист следует за ней. В нем регистрируется прохождение партии по технологическому маршруту. Как правило, маршрутные листы тиражируются типографским методом.

Каждому изделию ставится в соответствие индивидуальный технологический маршрут. В пределах одного изделия все партии имеют одинаковый маршрутный лист. Элементарная единица управления - партия, т.е. система управления отслеживает прохождение каждой партии по маршруту. Партий, представляющих конкретное изделие (изготавливаемых по данному маршруту), может быть много (до 500). Номер партии пластин составляется из номера изделия и номера партии в пределах изделия (рис.4).

Рис.4. Схема нумерации партий

Изделие 1

Изделие 2

Партия 001-001

Партия 001-002

Партия 001-003

Партия 001-00N

Партия 002-001

Партия 002-002

Партия 002-003

Партия 002-00K

…

…

…

## 2.1.1.2 Регистрация процесса производства партий пластин

Процесс производства партий контролируется посредством регистрации потоков партий в регистрационных журналах, в бланках незавершенного производства и в маршрутных листах. Регистрация необходима для контроля прохождения партий, в частности, при необходимости партию можно быстро найти.

Каждый участок имеет свой регистрационный журнал. В нем персонал установок отмечает партии, обработанные на данном участке. По нему считают проработку партий за смену, за месяц.

Для "нулевых" партий (запускаемых на начало обработки на входе цеха 201) есть свой регистрационный журнал. Для полностью обработанных партий (на выходе цеха 201) также есть отдельный регистрационный журнал.

Маршрутный лист - документ, неразрывно связанный с партией. Каждой партии соответствует свой маршрутный лист, который прикреплен к коробке с пластинами. На протяжении всего периода существования партии маршрутный лист следует за ней. Он представляет собой таблицу, содержащую: номер операции, название, время поступления партии, параметры режима обработки, колонку для росписи или табельного номера.

Записи в маршрутные листы вносят операторы участков. Операторы регистрируют выполнение технологических операций: по окончании обработки партии на конкретной операции оператор ставит свой табельный номер и параметры режима обработки. Ни распреды, ни диспетчер, ни начальник смены не вносят никаких записей в маршрутный лист.

Бумажный "документооборот" (регистрация в журналах, бланках незавершенного производства и маршрутных листах) до определенной степени избыточен, что обеспечивает надежность контроля производственного процесса.

## 2.1.1.3 Незавершенное производство

Важное понятие, характеризующее производственный процесс - незавершенное производство (НЗП).

Незавершенное производство - это продукция (партии кремниевых пластин), не прошедшая до конца весь технологический цикл.

Учитывается НЗП в регистрационных журналах, бланках незавершенного производства и в маршрутных листах. В настоящее время незавершенное производство велико - количество партий НЗП в 4 раза превышает месячную пропускную способность цеха. Наличие большого НЗП приводит к тому, что растет длительность производственного цикла.

Большая величина НЗП связана с тем, что административный персонал цеха и завода, желая увеличить загрузку оборудования, вводит в производственный процесс все новые и новые партии. При этом уже обрабатываемые партии скапливаются в "узких местах" производственного процесса, что создает "рецидивы" накопления НЗП.

## 2.1.2 Общая схема управления

Начальник цеха, плановый отдел

Мастер

Нач. смены

Диспетчер

Планово-диспетчерская служба

Устанавливает априорные приоритеты партий для каждого изделия и формирует план выпуска на месяц

Распред

Распред

Распред

Выдает распоряжения и координирует работу

Партия

Партия

Партия

Осуществляет транспортировку

Участок ИЛ

Оператор

Установка

Участок ФК

Оператор

Установка

Участок УНЭС

Оператор

Установка

Координирует работу и планирует загрузку

Рис.5. Общая схема управления

## 2.1.3 Задержки в транспорте и обработке партий пластин

В данное время существует много задержек в транспорте и обработке партий. Эти задержки можно разделить на следующие группы:

обеденные перерывы.

совещания административного персонала цеха.

сдача и приемка смен.

оформлением списка, где какие партии находятся, и поиск партий.

сбои оборудования.

необходимость принимать повторные решения при сбое оборудования.

и т.п.

Первые три группы задержек можно устранить посредством целенаправленной организации рабочего дня персонала (см. ниже).

Остальные группы задержек являются следствием несовершенства системы управления и контроля. Контролировать их людьми очень трудно, поэтому вводится избыточная загрузка цеха партиями, чтобы обеспечить работой все участки в случае сбоев некоторых. Эта загрузка приводит к большому НЗП, что в свою очередь увеличивает длительность цикла. Автоматизированная система управления производственным процессом призвана сократить эти задержки. Электронная система позволит повысить четкость отслеживания партий и планирования работ, в результате этого можно будет уменьшить НЗП, не опасаясь перерывов в работе участков. Уменьшение НЗП в свою очередь приведет к уменьшению длительности производственного цикла.

Оценим количественно задержки в транспорте партий и задержки в управлении транспортом и производством партий.

Задержки в транспорте партий, обусловленные неоптимальностью работы распредов (в расчете на сутки). Пересменка - 2\*20 минут = 40 минут, обеденные перерывы 30\*4 = 2 часа. Составление списка партий, поиск партий (полагаем пока условно, в дальнейшем эти цифры можно будет уточнить) - 2 часа в сутки.

Итого: задержки в транспорте партий составляют примерно 5 часов в сутки.

Устранить эти задержки в электронной системе управления можно следующим образом (для определенности рассматриваем пример подцеха 131): Задержки, связанные с пересменками и обеденными перерывами, можно устранить, вводя дополнительно одного распреда и вводя скользящий график работы. А именно, считаем, что первый распред начинает работу в 7-00, второй - в 7-30, третий - в 8-00. Смена первого, второго и третьего распредов заканчивается в 19-00, 19-30 и 20-00 соответственно. Аналогично смещаются относительно друг друга и обеденные перерывы распредов. В результате, в подцехе все время будут находиться, по крайней мере, два распреда.

Задержки, связанные с составлением списков должны устраниться в электронной системе автоматически, так как ведется учет всех партий. Составление списка (если это потребуется будет сводиться к распечатке соответствующего документа).

Оценим задержки в управлении (в сутки): совещания административного персонала цеха - 2\*30 мин = 1 час, обеденные перерывы - 4\*30мин =2 часа, сдача и приемка смен (полагаем пока условно, в дальнейшем эти цифры можно будет уточнить) - 2\*30 мин = 1 час, сбои оборудования, необходимость принимать повторные решения при сбое оборудования и т.п. (полагаем пока условно, в дальнейшем эти цифры можно будет уточнить) = 1 час.

Итого - 5 часов в сутки не принимается решение о транспорте партий и обработке партий.

Эти задержки можно устранить следующим образом. Сократить или заменить краткой электронной конференцией (или телеконференцией) совещания административного персонала цеха. В крайнем случае, вынести эти совещания за рамки смены, т.е. проводить совещания не во время своей смены, а течение последних 30 минут предыдущей смены. Обеденные перерывы распределить между административным персоналом, так чтобы не было перерывов в управлении транспортом партий.

Наличие электронной системы позволит обеспечить указанное устранение задержек, так как большинство решений о транспорте партий будет приниматься именно электронной системой и нагрузка на административный персонал радикально снизится. Сокращение (или даже полное устранение совещаний) будет возможно, так как многие вопросы могут быть сняты точным контролем и обеспечением информации на рабочие места административного персонала через терминалы компьютерно-сетевой системы управления.

Электронная система обеспечит более строгий учет за транспортировкой и обработкой партий. При этом снимаются проблемы поиска потерявшихся партий, необходимости принятия повторных решений. Система позволит оперативно и гибко реагировать на сбои оборудования, меняя очередность прохождения партий на "узких" участках.

Кроме того, система позволит собирать точную статистику по обработке партий, пропускной способности участков, дефектности; появится возможность прогнозирования, как на краткосрочную, так и на долгосрочную перспективу.

## 2.2 Анализ требований к системе

Современное микроэлектронное производство представляет собой сложный процесс. На уровне отдельного цеха этот процесс включает в себя обработку партий полупроводниковых пластин на установках, которые расположены на различных участках. Необходима транспортировка партий между участками, необходим учет времен обработки партий на установках, возможных временных задержек обработки партий. Важно учесть приоритеты изготовления партий. Необходим контроль качества партий, исправление дефектов обработки партий, учет возможных выходов из строя установок, обрабатывающих партии.

В этом процессе важно также найти компромисс между максимальной загруженностью установок цеха и минимальным циклом изготовления партий. Влияние загруженности установок на длительность цикла изготовления сводится к следующему: для увеличения загрузки установок нужно запускать в производственный процесс как можно большее число партий, но при этом партии будут скапливаться в "узких местах", в результате чего возникают большие очереди на установках, и длительность цикла изготовления отдельной партии будет возрастать.

Помимо этого, необходимо осуществлять мониторинг качества работы участков, собирать и обрабатывать статистику о процессе производства партий.

Итак, необходима система управления производством партий. Эта система должна обеспечивать принятие решений о транспорте и обработке партий. При этом количество принимаемых решений в единицу времени достаточно велико: оценки для типичного цеха дают величины порядка 108 элементарных принятий решений в месяц. Следовательно, система управления должна быть оперативной. Более того, эта система должна быть достаточно "интеллектуальной", с тем, чтобы учесть указанные выше требования. Оперативность и "интеллектуальность" системы управления подразумевают, что она должна базироваться на современных компьютерно-сетевых технологиях и использовать "интеллектуальные" схемы принятия решения.

В настоящем сообщении предлагаются принципы построения автоматизированной компьютерно-сетевой системы управления транспортом и обработкой партий отдельного цеха.

При разработке этих принципов мы основывались на общих методах и подходах, предложенных в системах ситуационного управления [1-4], и использовали идеи активно развивающегося сейчас за рубежом направлений Manufacturing executing systems (MES-системы) и Системы диспетчерского управления и сбора данных (SCADA-системы).

Общие требования к автоматизированной системе управления производственным процессом состоят в следующем. АСУ ПП должна:

Осуществлять контроль прохождения партий по технологическому маршруту.

Оптимизировать прохождение партий по различным маршрутам, чтобы все они обрабатывались в срок, с наилучшим выходом годных, и предприятие работало с наибольшей производительностью.

Собирать и обрабатывать статистику о качестве работы участков, принимаемых решениях и их результатах, и на основании этого изменять свое поведение и информировать руководство о возможных причинах неоптимальной работы предприятия.

В рамках подготовительного этапа была проведена следующая работа:

Проведен анализ принципов построения аналогичных систем управления (результаты анализа отражены в обзоре литературы).

Проанализирована структура технологического процесса на ОАО "Ангстрем", сложившаяся к настоящему времени.

Предложена архитектура компьютерно-сетевой АСУ ПП, предназначенной для управления технологическим процессом производства партий.

Произведен объектно-ориентированный анализ технологического процесса производства партий. В результате анализа разработана объектная модель производственного процесса. Объектная модель представляет собой концептуальную основу разрабатываемой компьютерно-сетевой системы управления производством партий (см. технологическую часть).

## 2.3 Рекомендации к разработке автоматизированной системы управления производственным процессом

На основе проведенного анализа можно рекомендовать следующие общие принципы построения разработки автоматизированной системы управления производством партий (АСУ ПП) ОАО "Ангстрем".

Система управления должна включать в себя:

Реляционную базу данных для хранения всей информации о производственном процессе. База данных нужна для того, чтобы информация о состоянии производства была представлена в некотором стандартном формате.

Центральную программу управления. Эта программа должна быть предназначена для:

Принятия решений о транспорте и обработке партий.

Отслеживания и сокращения задержек с целью сокращения производственного цикла.

Сбора статистики (пропускная способность участков, средний объем НЗП, задержки).

Контролирования объема НЗП.

Формирования очереди к потенциально узким участкам.

Своевременного предупреждения "рецидивов" накопления НЗП.

Планирования на краткосрочную и долгосрочную перспективу.

Предполагается, что эта программа управления должна быть расположена на центральном компьютере и помогать принимать решения диспетчеру и персоналу участков.

Локальные терминалы на участках, предназначенные для обмена информацией между персоналом участков и центральным компьютером.

Имитационную программу, предназначенную для анализа производственного процесса и поиска оптимальных режимов управления производством. Системы регистрации и мониторинга производства партий.

Общение программного обеспечения с пользователями должно происходить посредством максимально удобного и интуитивно понятного интерфейса, например, типа MS Windows. Анализ развития компьютерных технологий и опыта зарубежных фирм по разработке MES - и SCADA-систем показывает, что АСУ ПП должна базироваться на компьютерно-сетевых технологиях и использовать методы объектно-ориентированного программирования.

## 

## 2.4 Предлагаемая архитектура автоматизированной системы управления производственным процессом

В этом разделе кратко излагаются предложения по созданию системы управления производством партий.

## 2.4.1 Общая схема управления

Общая схема системы управления представлена на Рис.6. Система предназначена для управления производством партий в отдельном цехе.

Центральный блок управления

Имитационная модель предприятия

Реляционная база данных

Программа управления производством

Управляемое производство

Участок №1

Терминал №1

Участок №2 Терминал №2

Участок №N Терминал №N

Установка №1

Установка №2

Установка №3

Группа контроля

Тренинг

Рис.6. Архитектура системы управления производством партий в цехе

Стрелками показаны информационные потоки. Двойными стрелками показаны информационные потоки между программными модулями в центральном блоке управления. Блоки "Тренинг" и "Имитационная модель" не входят в систему управления цехом, а только лишь связаны с ней.

Блоки системы управления описаны ниже.

*Группа контроля* осуществляет общее руководство управлением производством партий в цехе. В группу контроля входит диспетчер и, возможно, другой административный персонал. Группа контроля интерактивно взаимодействует с программой управления производством. Программа осуществляет управление потоками партий в режиме "автопилота". Группа контроля имеет право вмешиваться в режим управления. В случае необходимости группа контроля принимает управление на себя и выполняет управление "в ручном режиме".

*Программа управления производством* предназначена для управления потоками партий в режиме реального времени. Эта программа отслеживает весь производственный процесс.

Предварительно, модули и функции программы управления можно описать следующим образом:

Программа должна содержать *модуль принятия решения*, который обеспечивает решения о транспорте партий и загрузке установок.

Программа должна содержать *модуль* *контроля участков.* Этот модуль отслеживает соответствие между инструкциями программы управления и работой участков.

Программа отслеживает весь процесс производства вплоть до конкретного исполнителя.

В *реляционной базе данных* собирается информация о процессе производства партий. Эта информация используется как в программе управления производством, так и в имитационной модели.

*Имитационная модель* предназначена для исследования функционирования и оптимизации самой системы управления.

Предполагается, что разработанная программа (или ее упрощенная версия) будет использоваться для *тренинга* персонала. Для отдельных цехов блоки "Имитационная модель" и "Тренинг" могут отсутствовать.

## 2.4.2 Взаимодействие программы регистрации и персонала

*Персонал участка* выполняет операции в соответствии с инструкциями программы и ставит в известность программу о выполненных операциях. Программа регистрации процесса производстваставит обработанную партию в очередь на все установки, соответствующие дальнейшему маршруту партии. Приоритеты рассчитываются только в случае конфликтов. При расстановке приоритетов учитываются мощности установок (скажем, партия с большим приоритетом должна обрабатываться на более мощной установке). Программе известно нормативное время обработки партии, и если потери времени *существенны*, то информация об этом поступает в группу контроля.

На каждом участке есть терминал, через который осуществляется взаимодействие между программой и персоналом участка.

Предполагается, что регистрация выполнения операций осуществляется с помощью внесения оператором участка информации в базу данных.

Как и в имеющейся системе, транспорт партий осуществляется распредами, а анализ забракованных партий - ведущим технологом.

## 2.4.3 Реализация

Предлагается следующая схема технической реализации (схема может быть уточнена в процессе дальнейшей работы). Программа управления производством реализуется на центральной ЭВМ. Реляционная база данных и имитационная модель реализуются на отдельном сервере либо на центральной ЭВМ. На участках с длительными технологическими операциями есть терминал, который обеспечивает информационное сообщение между персоналом участка и программой управления производством. Рабочие места диспетчера и ведущего технолога также оснащаются терминалами.

## 2.4.4 Управление обработкой партий

## 

## 2.4 4.1 Понятие дисциплины очереди

Управление транспортом партий происходит в соответствии с определенной дисциплиной очереди.

Дисциплина очереди - это набор правил, в соответствии с которыми осуществляется управление порядком обработки партий на установках.

Существует два типа очередей:

Очереди НЗП, таких очередей несколько, отдельная конкретная очередь соответствует определенной установке. Партии, стоящие в этих очередях, "ждут" подачи на начало соответствующего микроцикла.

Очередь на установках, партии в этих очередях "ждут" подачи на операцию внутри микроцикла.

Ожидание в очереди 1-го типа может быть длительным (часы, сутки, недели). Ожидание в очереди 2-го типа должно быть кратковременным (минуты, десятки минут).

Управление очередями 2-го типа простое, оно осуществляется по методу FIFO (первым пришел - первым вышел), т.е. в первую очередь обрабатываются те партии, которые первыми попали в очередь.

Основное управление осуществляется между микроциклами. Это управление осуществляет группа контроля при помощи программы управления производством. Эта группа анализирует загруженность цеха, и определяет момент, когда способен принять партию из НЗП на обработку. Именно здесь важно использовать продуманную дисциплину очереди.

Ниже приведены схемы дисциплины очереди.

## 

## 2.4.4.2 Предварительная схема дисциплины очереди

При формировании дисциплины очереди необходимы знания об определенных параметрах партии и производственного процесса. Выделим эти параметры.

Основные параметры (атрибуты партий, атрибуты установок, характеристики производственного процесса), которые требуется учесть при управлении потоками партий:

Время задержки прохождения партией данной операции (отрицательные времена тоже важны), Δ *t*зад;

Априорная важность партии, *Imp* (обычная партия *Imp* = 1, важная партия *Imp* = 2,3,…) априорную важность партии устанавливает группа контроля по заданию руководства;

Процент прохождения маршрута, Pr;

Прогноз следствий невыполнения;

Коэффициент нагрузки, *r* = *I*заявок / *I*обслуживания (*I* - интенсивность). *r* зависит от НЗП. Средняя длина очереди зависит от *r:* *l* = [1 - *r*] -1 Разумно выбирать *r* ≤ 0,5 - 0,6, в этом случае длина очереди *l* = ≤ 2 - 2,5.

Предварительная схема управления потоками партий такова.

Если партия прошла *n*-й микроцикл, то в программу управления производством поступает рапорт о выполнении микроцикла и заявка на выполнение *n* +1-ого микроцикла.

Считаем, что схема имеет вид правил:

Если …., то …., иначе.

Некоторые из правил требуют выполнения определенных расчетов. Схема принятия решения основана на учете указанных выше параметров.

Опишем конкретный вариант упрощенной схемы принятия решения.

*Если* есть свободная установка для выполнения первой операции *n* +1-ого микроцикла, и эта установка одна, *то* заявка подается на эту установку.

Иначе

*Если* свободных устройств несколько, *то* заявка подается на установку с наибольшим приоритетом для данной партии.

Иначе

*Если* все устройства, соответствующие первой операции *n* +1-ого микроциклазаняты, *то* партия подается в очередь на все эти установки.

При возникновении свободного устройства обрабатывается партия с наибольшим текущим приоритетом.

В простейшем случае текущий приоритет есть Δ *t*зад \* *Imp.*

Еще раз подчеркнем, что необходим отдельный анализ по разработке дисциплины очереди. Отметим, что известен ряд методов формирования систем управления, подобных разрабатываемой системе. Одна из серьезных проработок: методы ситуационного управления, описанные выше. Второй пример - схемы классифицирующих систем (Classifier Systems), представляющие собой самообучающиеся системы принятия решения. Эти системы основаны на использовании правил типа "Если …., то …. ". Совокупность таких правил оптимизируется как путем обучения (переоценка приоритетов правил), так и путем эволюции (эволюционный поиск новых правил). Еще одну возможность предоставляют системы поддержки принятия решений, основанные на активно применяемом в последнее время нейросетевом подходе.

## 2.5 Техническое задание на дипломный проект

Исходя из изложенного выше материала, сформулируем техническое задание на дипломное проектирование.

Необходимо разработать программу регистрации процеса производства партий полупроводниковых пластин для использования в автоматизированной системе управления.

Программа должна обеспечивать контроль и регистрацию производственного процесса производства партий пластин. Вести учет за прохождением партий полупроводниковых пластин по технологическому маршруту. Разработку провести, используя объектно-ориентированный подход.

К программе регистрации предъявляются следующие требования:

Регистрировать передачу партии полупроводниковых пластин на участок.

Фиксировать время начала и окончания выполнения технологической операции.

Осуществлять управление очередностью изготовления партий на установках с длительными операциями.

Формировать плановый срок для каждой операции обработки и следить за его соблюдением.

Оценивать потери, с их разложением на каждом участке для каждой смены.

Программа должен функционировать в среде MS Windows 95,98,NT. Предполагается использовать программу на участках с длительными технологическими операциями (нанесение эпитаксиальных структур, диффузия, металлизация).

## 2.6 Выбор платформы и инструмента разработки программы

В качестве операционной системы для реализации программного обеспечения была выбрана среда Windows’98 (Windows NT). Можно выделить следующие причины, обосновывающие этот выбор:

Распространенность этих операционных систем;

удобство графического интерфейса дает возможность при минимизации затрат на его реализацию максимизировать удобство работы оператора с программой. Интерфейс стандартизован, соответственно оператор не потратит много времени на освоение программы и в дальнейшем количество возможных ошибок в его действиях будет минимизировано;

возможность работы с большими массивами данных, реализация чего в среде Windows 3.1 или в среде MS-DOS представляет нетривиальную и трудновыполнимую задачу;

32-разрядность систем Windows’95 и Windows NT увеличивает скорость выполнения вычислительных задач.

В качестве среды программирования была выбрана среда Visual C++ фирмы MicroSoft, сочетающая в себе следующие преимущества:

простота и надежность создания и отладки программы;

использование всех преимуществ операционных систем Windows’98 и Windows NT, включая 32-разрядность, многозадачность, удобный интерфейс и прочее;

использование обработки исключений (exceptions), что позволяет повысить надежность работы программного продукта;

наличие и доступность большого количества компонент, реализующих многие стандартные функции;

наличие мощных, удобных утилит для отладки и тестирования программного обеспечения;

наличие большой системы справочной информации, руководств разработчика, облегчающей создание программного обеспечения.

## 3. Разработка алгоритмов и программ

## 3.1 Этапы объектно-ориентированного подхода

К созданию системы был выбран объектно-ориентированный подход. Этот подход предусматривает спиральную модель создания информационных систем, которая состоит из следующих этапов:

Анализ требований.

Проектирование.

Реализация (программирование).

Тестирование.

Спиральная модель подразумевает, что эти этапы циклически повторяются, на каждом цикле в программу вносятся исправления и новые возможности. Этот способ создания программного обеспечения позволяет достаточно быстро получить первую рабочую версию программы, а на следующих циклах реализовывать дополнительно возникающие требования и оптимизировать существующие части программы, не нарушая ее целостности и работоспособности.

Объектно-ориентированный анализ

Ниже приводятся результаты первого этапа - анализа требований к проектируемой системе. При анализе использовались стратегии и образцы, разработанные Питером Коудом, а также методология OMT (Object Modeling Technique). Результатами анализа являются назначение системы, характерные свойства, объектная модель.

Назначение системы:

Обеспечить контроль и регистрацию производственного процесса производства партий пластин. Вести учет за прохождением партий полупроводниковых пластин, как материальных ценностей.

## 3.2 Характерные свойства системы

Система должна:

Регистрировать передачу партии полупроводниковых пластин на участок.

Составлять график изготовления для каждой партии и следить за его выполнением.

Фиксировать результаты контроля и измерений для последующего анализа.

Фиксировать время начала и окончания выполнения технологической операции.

Определять очередность и времена запуска партий на обработку.

Осуществлять управление очередностью изготовления партий на установках с длительными операциями.

Формировать плановый срок для каждой операции обработки и следить за его соблюдением.

Оценивать потери, с их разложением на каждом участке для каждой смены.

## 3.3 Выбранные объекты

В процессе анализа были выявлены следующие объекты:

Группа контроля - лицо или группа лиц, осуществляющее планирование запуска партий из НЗП в производство и материально ответственное за партии, хранящиеся в НЗП.

Транспорт - служба, занимающаяся транспортировкой партий между участками.

Представитель участка - лицо или группа лиц, материально ответственное за партии находящиеся на участке.

Ведущий технолог - лицо, принимающее решения в случае забраковки партий на контроле.

Установка - устройство, осуществляющее операции над партиями.

Участок - административная единица, выполняющая некий набор операций над партиями.

Участок НЗП - участок, на котором партии хранятся между микроциклами, в НЗП поступают новые партии и готовые партии.

Шкаф - место на участке НЗП, в котором партии хранятся между микроциклами.

Контроль - 1) участок, на котором осуществляется операция контроля,

2) операция контроля, на которой выявляется брак.

Партия - объект, подвергающийся операциям на участках в соответствии с маршрутным листом.

Маршрутный лист - документ, содержащий список операций, которые партия должна пройти в процессе изготовления.

Операция - действие, изменяющее готовность партии, этап изготовления партии согласно маршрутному листу.

Хранение - операция, которой подвергаются партии между микроциклами.

Очередь - место на технологическом участке, где находятся партии в ожидании очередной операции.

Реставрация - изменение в технологическом маршруте, возникающее в связи с браком, осуществляется ведущим технологом.

Микроцикл - часть технологического маршрута; между микроциклами партия может храниться длительное время.

Забраковка - признание партии непригодной для дальнейшей обработки, ее удаление из плана производства и, возможно, запуск новой идентичной партии.

Входной считыватель штрихкода - устройство, фиксирующее поступление партии на участок.

Выходной считыватель штрихкода - устройство, фиксирующее передачу партии с участка на транспорт.

Технологический участок - административная единица, выполняющая некий набор операций над партиями на установках, составляющих данный участок.

Терминал - средство общения между системой управления и персоналом.

Цех - помещение, в котором осуществляется производство.

Участок ВТ - место нахождения Ведущего технолога.

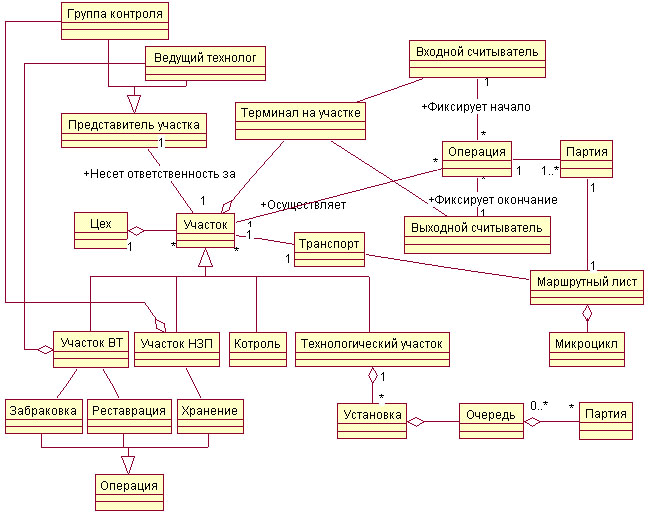


Рис.7. Объектная модель, характеризующая объекты системы управления и связи между ними

Объектная модель показывает планируемую структуру программы, с точки зрения объектно-ориентированного подхода. В этой модели показаны объекты, важные для проектируемой системы и связи между ними. Рис.8 поясняет нотацию (систему обозначений) Unified, в которой приведена модель.

Рис.8. Обозначения, принятые в объектной модели

Участок

Имя объекта

Атрибуты объекта (определяются на этапе проектирования)

Методы объекта (определяются на этапе проектирования)

– агрегация: означает, что один объект является частью другого

– ассоциация: означает, что один объект "знает" о существовании другого, т.е. объекты каким-либо образом взаимодействуют друг с другом

– наследование: означает, что объект-наследник обладает свойствами объекта предка, а также своими собственными свойствами, отличающими его от предка и других наследников

В процессе проектирования и разработки программы управления к модели будут добавляться новые объекты и связи, для объектов будут определяться атрибуты и методы, взаимодействие между объектами будет описано через сценарии. Конечным результатом усложнения модели будет программа управления производством. Все объекты, приведенные в данной модели, будут являться частями будущей программы.

## 3.4 Алгоритм выполнения технологической операции

В данной программе схема работы представлена двумя основными алгоритмами. Рис.9 описывает алгоритм выполнения технологической операции.



Рис.9. Алгоритм выполнения технологической операции

## 3.5 Алгоритм формирования дисциплины очереди

Рис.10 конкретизирует этап выбора партии полупроводниковых пластин для обработки.

Управление транспортом партий происходит в соответствии с определенной дисциплиной очереди.

Дисциплина очереди - это набор правил, в соответствии с которыми осуществляется управление порядком обработки партий на установках.

Проведенный анализ выявил 4 правила, в соответствии с которыми из всего множества партий пластин выбирается одна для обработки. Правила перечислены в порядке убывания приоритета. Каждое последующее правило применяется, когда выполнены все предыдущие.

Выбрать из всего множества партии с минимальным межоперационным временем;

Выбрать из оставшегося списка партии пластин с высоким априорным приоритетом;

Выбрать из оставшегося списка партии, близкие к завершению технологического цикла;

Использовать правило "FIFO - первым вошел - первым вышел".

Все рассмотренные правила представлены алгоритмом.

Рис.10. Алгоритм формирования дисциплины очереди



## 3.6 Использование библиотеки Microsoft Foundation Classes

При разработке программы использовались преимущества библиотеки классов MFC, предоставляющей программисту набор мощных инструментов. Библиотека инкапсулирует наиболее важные структуры и функции интерфейса прикладного программирования в группе классов, пригодных для многократного использования.

Использование объектно-ориентированного подхода дает многочисленные преимущества в сравнении с традиционным стилем программирования:

Инкапсуляция кода и данных внутри класса;

Наследование;

Разрешение коллизий имен функций;

Уменьшается общий объем текстов.

Основные возможности библиотеки MFC:

Поддержка всех функций, управляющих элементов и ссобщений Windows, графических примитивов GDI, меню и окон диалога;

Использование соглашения об именах API Windows, при котором по имени класса становится ясным его предназначение;

Избавление от необходимости применять оператор switch/case, который часто является источником ошибок. Каждому сообщению ставится в соответствие функция-член класса;

Архитектура, основанная на широком использовании обработки исключительных ситуаций, делает код приложения более устойчивым к ошибкам.

На рис.11 приведена иерархия классов программы. Ниже будут приведены пояснения для основных классов.

Рис.11. Иерархия классов программы регистрации процесса производства



Cobject - базовый класс, широко используемый при разработке приложений Windows.

CCmdTarget -

CWnd - родительский класс для всех окон.

CWinApp - родительский класс для приложения.

CDialog - родительский класс для окон диалога.

Как видно из иерархии классов, программа состоит из четырех основных классов:

CAngstremApp - основной класс. Разделен на два файла - заголовочным для него является файл cangstremapp. h, файл реализации - cangstrem. cpp.

CMainFrame - этот класс, производный от CFrameWnd, используется для управления главным окном приложения.

CAngstremView - является наследником класса CView. Класс CView, производный от CWnd, служит базовым для пользовательских классов отображения. Отображение (view) служит промежуточным звеном между документом и пользователем. Обычно отображение - это дочернее окно главного окна.

CAngstremDoc - класс используется для хранения данных.

Технологическая часть:

Технология разработки программных систем

Выполнил:

Ляшко М.А.

Консультант:

Брусникин Г.Н.

## 4. Объектно-ориентированное программирование

## 

## 4.1 Введение

В основе разработки программного обеспечения, как и любой отрасли промышленного производства, лежит технологический процесс. Качество программного продукта, его стоимость, сроки создания определяются тем, насколько хороша принятая на фирме технология разработки программ и насколько точно она соблюдается. Это утверждение, очевидное для всех отраслей промышленности еще со времен цеховых мастеров, не легко пробивает себе дорогу в мир разработки программного обеспечения. Это связано с тем, что программирование - это интеллектуальный высокотехнологичный труд, очень тяжело поддающийся формализации. Но законы развития промышленного производства одинаковы для всех отраслей промышленности.

Технология программирования - это совокупность методов и средств разработки (написания) программ и порядок применения этих методов и средств. На ранних этапах развития программирования, когда программы писались в виде последовательностей машинных команд, какая-либо технология программирования отсутствовала. Первые шаги в разработке технологии состояли в представлении программы в виде последовательности операторов. Написанию последовательности машинных команд предшествовало составление операторной схемы, отражающей последовательность операторов и переходы между ними. Операторный подход позволил разработать первые программы для автоматизации составления программ - так называемые составляющие программы.

С увеличением размеров программ стали выделять их обособленные части и оформлять их как подпрограммы. Часть таких подпрограмм объединялась в библиотеки, из которых подпрограммы можно было включать в рабочие программы и затем вызывать из рабочих программ. Это положило начало процедурному программированию - большая программа представлялась совокупностью процедур-подпрограмм. Одна из подпрограмм являлась главной и с нее начиналось выполнение программы.

В 1958 году были разработаны первые языки программирования, Фортран и Алгол-58. Программа на Фортране состояла из главной программы и некоторого количества процедур - подпрограмм и функций. Программа на Алголе-58 и его последующей версии Алголе-60 представляла собой единое целое, но имела блочную структуру, включающую главный блок и вложенные блоки подпрограмм и функций. Компиляторы для Фортрана обеспечивали раздельную трансляцию процедур и последующее их объединение в рабочую программу, первые компиляторы для Алгола предполагали, что транслируется сразу вся программа, раздельная трансляция процедур не обеспечивалась.

Процедурный подход потребовал структурирования будущей программы, разделения ее на отдельные процедуры. При разработке отдельной процедуры о других процедурах требовалось знать только их назначение и способ вызова. Появилась возможность перерабатывать отдельные процедуры, не затрагивая остальной части программы, сокращая при этом затраты труда и машинного времени на разработку и модернизацию программ.

Следующим шагом в углублении структурирования программ стало так называемое структурное программирование, при котором программа в целом и отдельные процедуры рассматривались как последовательности канонических структур: линейных участков, циклов и разветвлений. Появилась возможность читать и проверять программу как последовательный текст, что повысило производительность труда программистов при разработке и отладке программ. С целью повышения структурности программы были выдвинуты требования к большей независимости подпрограмм, подпрограммы должны связываться с вызывающими их программами только путем передачи им аргументов, использование в подпрограммах переменных, принадлежащих другим процедурам или главной программе, стало считаться нежелательным.

Процедурное и структурное программирование затронули прежде всего процесс описания алгоритма как последовательности шагов, ведущих от варьируемых исходных данных к искомому результату. Для решения специальных задач стали разрабатываться языки программирования, ориентированные на конкретный класс задач: на системы управления базами данных, имитационное моделирование и т.д.

Далее совершенствование аппаратных средств и многократное усложнение программного обеспечения обусловило появление так называемого объектно-ориентированного программирования, ставшего действительно гигантским шагом вперед в развитии всего программирования в целом. В этой главе будет рассматривается весь этап создания программного продукта от постановки задачи и анализа до его внедрения и сопровождения с применением именно объектно-ориентированных технологий. Однако, опыт программирования показывает, что любой методологический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу.

## 4.2 Понятие жизненного цикла программного обеспечения

Одним из базовых понятий методологии разработки информационных систем является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО). ЖЦ ПО - это непрерывный процесс, который начинается с момента принятия решения о необходимости создания программного обеспечения и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207 (ISO - International Organization of Standardization). Он определяет структуру жизненного цикла, содержащую процессы, действия и задачи, которые должны быть выполнены.

## 4.3 Модели жизненного цикла программного обеспечения

Стандарт ISO/IEC 12207 не предлагает конкретную модель жизненного цикла и методы разработки ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий его разработки.

Под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Модель ЖЦ зависит от специфики разрабатываемой информационной системы и специфики условий, в которых система создается и функционирует. К настоящему моменту наибольшее распространение получили следующие две основные модели ЖЦ: каскадная модель и спиральная модель.

В изначально существовавших однородных ИС приложения представляли собой единое целое. Для разработки такого типа приложений применялся каскадный способ*.* Его основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на другой происходит только после того, как будет полностью завершена работа на текущем (рис.12).

Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.



Рис.12. Каскадная схема разработки программного обеспечения

Преимущества применения каскадного способа заключается в следующем:

На каждом этапе формируется законченный набор проектной документации, отвечающей критериям полноты и согласованности;

выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты;

каскадный подход хорошо зарекомендовал себя при построении информационных систем, для которых в самом начале можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их технически как можно лучше. В эту категорию попадают сложные расчетные системы, системы реального времени и др.

В то же время этот подход обладает рядом существенных недостатков, вызванных прежде всего тем, что реальный процесс создания достаточно сложного ПО никогда полностью не удавалось уложить в такую жесткую схему, постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимал вид, представленный на рис.13. Вторым недостатком каскадного подхода является существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к информационной системе “заморожены" в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена.



Рис.13. Схема реального процесса разработки программного обеспечения

Для преодоления вышеперечисленных проблем была предложена спиральная модель ЖЦ, в которой делается упор на начальные этапы ЖЦ: анализ и проектирование с использованием объектно-ориентированного подхода. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, постепенно углубляются и последовательно конкретизируются детали проекта. Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если вся запланированная работа не закончена.



Рис.14. Спиральная модель жизненного цикла

## 4.4 Анализ

Первый шаг в разработке прикладной информационной системы заключается в точном формулировании целей внедрения машинной обработки. Требования к системе еще не создают полной картины. В постановке задачи, разумеется, должны принять участие как представители организации-заказчика, так и те, кто будет заниматься проектированием системы. Необходимо, чтобы этот процесс был гибко организован и продолжался в течении достаточно длительного времени, поскольку на любом этапе разработки или внедрения могут вскрываться ранее не предусмотренные трудности, требующие внесения определенных изменений в проект. В то же время с целью упрощения разработки в договор следует включить пункт, который запрещал бы радикальный пересмотр требований, на стадии реализации системы. В этом же пункте могут быть оговорены условия внесения несущественных изменений. Все договоренности должны оформляться в официальном порядке.

Разработка информационной системы на базе ЭВМ, включает такой обязательный элемент, как определение формы, в которой должна быть представлена информация, хранящаяся в хранилище. Кроме того, необходимо установить какие варианты данных будут использоваться на различных этапах, начиная от сбора исходной информации и заканчивая выдачей результатов. Очевидно, формы, применяемые на этапах сбора данных, их корректировки или выборки с целью проведения той или иной обработки, могут отличаться друг от друга. Варианты данных должны охватывать все входящие в автоматизированную систему и исходящие из нее информационные потоки. Постановка задачи должна охватывать весь процесс сбора, хранения, использования и окончательного представления данных. В ней должны также учитываться ограничения на время обработки, финансовые затраты и вычислительные ресурсы. Кроме того, в постановке задачи следует предусмотреть возможность последующей модернизации системы. Оценка будущих потребностей должна осуществляться совместно как проектировщиками системы, так и ее потенциальными пользователями, поскольку необходимо принимать компромиссные решения, касающиеся соотношения между универсальностью и эффективностью, гибкостью организации данных и стремлением обеспечить текущие потребности.

## 4.5 Проектирование

Этап проектирования в разработке информационной системы представляет собой наиболее ответственный процесс. Результат этого этапа накладывает самый большой отпечаток на всю систему. И грамотное проведение этого процесса в последствие намного облегчит труд разработчиков на следующих этапах. Поэтому наиболее подробно рассмотрим этот этап в разработке программного обеспечения информационной системы.

В любой инженерной дисциплине под проектированием обычно понимается некий унифицированный подход, с помощью которого мы ищем пути решения определенной проблемы, обеспечивая выполнение поставленной задачи. В контексте инженерного проектирования цели проектирования определяются как создание системы, которая:

удовлетворяет заданным (возможно, неформальным) функциональным спецификациям;

согласована с ограничениями, накладываемыми оборудованием;

удовлетворяет явным и неявным требованиям по эксплуатационным качествам и ресурсопотреблению;

удовлетворяет явным и неявным критериям дизайна продукта;

удовлетворяет требованиям к самому процессу разработки, таким, например, как продолжительность и стоимость, а также привлечение дополнительных инструментальных средств.

По предположению Страуструпа: "Цель проектирования - выявление ясной и относительно простой внутренней структуры, иногда называемой архитектурой... Проект есть окончательный продукт процесса проектирования". Проектирование подразумевает учет противоречивых требований. Его продуктами являются модели, позволяющие нам понять структуру будущей системы, сбалансировать требования и наметить схему реализации.

## 4.5.1 Методы проектирования

Ясно, что не существует такого универсального метода, который бы провел инженера-программиста по пути от требований к сложной информационной системе до их выполнения. Проектирование такой информационной системы отнюдь не сводится к слепому следованию некоему набору рецептов. Скорее это постепенный и итеративный процесс. И тем не менее использование методологии проектирования вносит в процесс разработки определенную организованность. Как уже утверждалось выше, инженеры-программисты с момента зарождения программирования до с настоящих дней разработали десятки различных методов, которые можно классифицировать по трем категориям:

процедурное программирование;

метод структурного проектирования "сверху вниз";

метод потоков данных;

объектно-ориентированное проектирование.

Структурное проектирование "сверху вниз" основано на алгоритмической декомпозиции. При таком подходе к проектированию происходит обычное разделение алгоритмов, где каждый модуль системы выполняет один из этапов общего процесса. Структурный подход в проектировании в течение большого промежутка времени был практически лидером среди имеющихся методов разработки программного обеспечения. Но в связи с резким скачком развития аппаратного обеспечения, появлением очень мощных вычислительных комплексов, и как результат - многократное усложнение программного обеспечения, структурное проектирование перестало быть эффективным. Значение структурного подхода осталось прежним, но было замечено, что структурный подход не работает, если объем программы превышает приблизительно 100000 строк. Структурный подход не позволяет выделить абстракции и обеспечить ограничение доступа к данным; он также не предоставляет достаточных средств для организации параллелизма. Структурный метод не может обеспечить создание предельно сложных систем, и он, как правило, неэффективен в объектных и объектно-ориентированных языках программирования. Однако, в некоторых случаях его применение в этих языках оправдывает себя.

В методе потоков программная система рассматривается как преобразователь входных потоков в выходные. Метод потоков данных, как и структурный метод, с успехом применялся при решении ряда сложных задач, в частности, в системах информационного обеспечения, где существуют прямые связи между входными и выходными потоками системы и где не требуется уделять особого внимания быстродействию. Более подробно остановимся на третьем методе проектирования, т.к он на данный момент является самым современным и эффективным средством проектирования информационных систем. Объектно-ориентированное проектирование - это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы.

В данном определении содержатся две важные части: объектно-ориентированное проектирование, во-первых основывается на объектно-ориентированной декомпозиции, во-вторых использует многообразие приемов представления моделей, отражающих логическую (классы и объекты) и физическую (модули и процессы) структуру системы, а также ее статические и динамические аспекты.

Именно объектно-ориентированная декомпозиция отличает объектно-ориентированное проектирование от структурного; в первом случае логическая структура системы отражается абстракциями в виде классов и объектов, во втором - алгоритмами.

## 4.5.2 Объектно-ориентированная модель

Объектно-ориентированная технология основывается на так называемой объектной модели. Основными ее принципами являются: абстрагирование, инкапсуляция, модульность, иерархичность, типизация, параллелизм и сохраняемость. Каждый из этих принципов сам по себе не нов, но в объектной модели они впервые применены в совокупности.

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя. Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности поведения от несущественных.

Инкапсуляция - это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Модульность - это свойство системы, которая была разложена на внутренне связные, но слабо связанные между собой модули. Таким образом, принципы абстрагирования, инкапсуляции и модульности являются взаимодополняющими. Объект логически определяет границы определенной абстракции, а инкапсуляция и модульность делают их физически незыблемыми.

Значительное упрощение в понимании сложных задач достигается за счет образования из абстракций иерархической структуры. Таким образом иерархия - это упорядочение абстракций, расположение их по уровням.

Типизация является способом защиты от использования объектов одного класса вместо другого, или по крайней мере, управлять таким использованием. Типизация заставляет выражать абстракции так, чтобы язык программирования, используемый в реализации, поддерживал соблюдение принятых проектных решений.

В то время, как объектно-ориентированное программирование основано на абстракции, инкапсуляции и наследовании, параллелизм главное внимание уделяет абстрагированию и синхронизации процессов. Каждый объект (полученный на конкретном уровне абстракции) может представлять собой отдельный поток управления (абстракцию процесса). Такой объект называется активным. Система, построенная на основе ООП, может быть представлена, как совокупность взаимодействующих объектов, часть из которых является активной и выступает в роли независимых вычислительных центров.

Любой программный объект существует в памяти и живет во времени. Существуют объекты, которые присутствуют лишь во время вычисления выражения, но есть и такие, как базы данных, которые существуют независимо от программы. Этот спектр сохраняемости объектов охватывает:

промежуточные результаты вычисления выражений;

локальные переменные в вызове процедур;

собственные переменные, глобальные переменные и динамически создаваемые данные;

данные, сохраняющиеся между сеансами выполнения программы;

данные, сохраняемые при переходе на новую версию программы;

данные, которые вообще переживают программу.

В использовании объектно-ориентированной модели можно выделить следующие преимущества:

объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования.

Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования не только программ, но и проектов, что в конце концов ведет к созданию среды разработки. Объектно-ориентированные системы часто получаются более компактными, чем их не объектно-ориентированные эквиваленты. А это означает не только уменьшение объема кода программ, но и удешевление проекта за счет использования предыдущих разработок, что дает выигрыш в стоимости и времени.

Использование объектной модели приводит к построению систем на основе стабильных промежуточных описаний, что упрощает процесс внесения изменений. Это дает системе возможность развиваться постепенно и не приводит к полной ее переработке даже в случае существенных изменений исходных требований.

Объектная модель уменьшает риск разработки сложных систем, прежде всего потому, что процесс интеграции растягивается на все время разработки, а не превращается в единовременное событие. Объектный подход состоит из ряда хорошо продуманных этапов проектирования, что также уменьшает степень риска и повышает уверенность в правильности принимаемых решений.

## 4.5.3 Процесс объектно-ориентированного проектирования

Процесс объектно-ориентированного проектирования не сводится к одностороннему движению "сверху вниз" или "снизу вверх". Хорошо структурированные сложные системы можно создать методом "возвратного проектирования".

В дальнейшем будем называть этот процесс микропроцессом. В этом методе основное внимание уделяется процессу поступательного итеративного развития путем совершенствования различных, но, тем не менее, совместимых между собой логических и физических моделей системы. Возвратное проектирование составляет необходимую основу процесса объектно-ориентированного проектирования.

Микропроцесс объектно-ориентированной разработки приводится в движение потоком сценариев и архитектурных продуктов, которые порождаются и последовательно уточняются в макропроцессе. Микропроцесс, по большей части, - повседневный труд отдельного разработчика или небольшого коллектива разработчиков.

Микропроцесс относится в равной степени к программисту и архитектору программной системы. С точки зрения программиста, микропроцесс предлагает руководство в принятии бесчисленного числа ежедневных тактических решений, которые являются частью процесса создания и подгонки архитектуры системы. С точки зрения архитектора, микропроцесс является основой для развития архитектуры и опробования альтернатив. В микропроцессе традиционные фазы анализа и проектирования умышленно перемешаны.



Рис.15. Микропроцесс проектирования

Как показано на рис.15, микропроцесс обычно состоит из следующих видов деятельности:

выявление классов и объектов на данном уровне абстракции;

выяснение семантики этих классов и объектов;

выявление связей между этими классами и объектами;

спецификация интерфейса и реализация этих классов и объектов.

Рассмотрим их более подробно:

1. Выявление классов и объектов.

*Цель:* Выявления классов и объектов состоит в том, чтобы найти границы предметной области. Кроме того, эта деятельность является первым шагом в продумывании объектно-ориентированной декомпозиции разрабатываемой системы. Этот шаг в анализе применяется тогда, когда обнаруживаются абстракции, составляющие словарь предметной области и, тем самым, ограничивая задачу, решая, что важно, а что - нет. Такие действия необходимы при проектировании, когда мы изобретаем новые абстракции, которые являются составными частями решения. Переходя к программной реализации, мы применяем процедуру выявления, чтобы изобрести более простые абстракции, из которых строятся более сложные, и обнаружить общие черты существующих абстракций, дабы упростить архитектуру системы.

*Результаты:* Главным результатом этого шага является обновляющийся по мере развития проекта словарь данных.

2. Идентификация семантики классов и объектов.

*Цель:* Определить поведение и атрибуты каждой абстракции, выявленной на предыдущем шаге. При этом мы уточняем намеченные абстракции, продуманно и измеримо распределяя между ними обязанности.

*Результаты:* Во-первых, уточняется словарь данных, с помощью которого мы изначально присвоили обязанности абстракциям. В ходе проектирования можно выработать спецификации к каждой абстракции, перечисляя имена операций в протоколе каждого класса. Затем, необходимо выразить интерфейсы этих классов на языке реализации. Например, для C++ это означает создание. h-файлов, в Ada - спецификаций пакетов, в CLOS - обобщенных функций для каждого класса, в Smalltalk - это объявление, но не реализация методов каждого класса. В добавление к этим, по сути тактическим решениям, мы составляем диаграммы объектов и диаграммы взаимодействий, передающие семантику сценариев, создаваемых в ходе макропроцесса. Эти диаграммы формально отражают раскадровку каждого сценария и, таким образом, описывают явное распределение обязанностей среди взаимодействующих объектов. На этом шаге впервые появляются конечные автоматы для представления некоторых абстракций.

3. Выявление связей между классами и объектами.

*Цель:* Уточнить границы каждой обнаруженной ранее в микропроцессе абстракции и опознать все сущности, с которыми она взаимодействует. Это действие формализует концептуальное и физическое размежевание между абстракциями, начатое на предыдущем шаге.

*Результаты:* Основными результатами этого шага являются диаграммы классов, объектов и модулей. Хотя в конце концов необходимо выразить решения, принятые при анализе и проектировании, на языке программирования, диаграммы дают более широкий обзор архитектуры и, кроме того, позволяют раскрыть отношения, которые с трудом формулируются на используемом языке реализации. Нет ни возможности, ни необходимости создавать исчерпывающий набор диаграмм, которые определили бы все возможные виды связей между нашими абстракциями. Нужно сосредоточиться лишь на самых важных отношениях, причем подразумевается, что в число важных входят те связи между абстракциями, которые отражают фундаментальные архитектурные решения или выражают детали, необходимые для реализации.

4. Реализация классов и объектов.

*Цель:* На этапе анализа реализация классов и объектов нужна, чтобы довести существующие абстракции до уровня, достаточного для обнаружения новых классов и объектов на следующем уровне абстракции; они сами будут в дальнейшем поданы на новую итерацию микропроцесса. Этот шаг намеренно выполняется позже всех, так как микропроцесс концентрирует внимание на поведении и откладывает насколько возможно решения о представлении. Такая стратегия оберегает разработчика от недозрелых решений, которые могут не оставить шансов на облегчение и упрощение архитектуры, и оставляет свободу выбора реализации (например, из соображений эффективности), гарантируя сохранение существующей архитектуры.

*Результаты:* Чтобы раскрыть связи между логическим и физическим в нашей реализации системы, вводятся диаграммы модулей, которые можно затем использовать, чтобы наглядно показать отображение архитектуры в ее программную реализацию. Далее можно применить специфические инструментальные средства (например CASE-средства), которые позволяют либо генерировать код из диаграмм, либо восстанавливать диаграммы по реализации. В этот шаг входит и обновление словаря данных, включая новые классы и объекты, которые были выявлены или изобретены при реализации существующих абстракций. Эти новые абстракции являются частью исходной информации для следующего цикла микропроцесса.

Контролирующим по отношению к микропроцессу является макропроцесс. Макропроцесс предписывает ряд измеримых результатов и действий, которые позволяют команде разработчиков оценить риск, внести заблаговременные изменения в микропроцесс и сосредоточиться на коллективном анализе и проектировании. Макропроцесс - это деятельность всего коллектива в масштабе от недель до месяцев.

## 4.6 Эволюция

Основная цель данного этапа - наращивать и изменять реализацию, последовательно совершенствуя ее, чтобы в конечном счете создать готовую систему. Эволюция архитектуры в значительной степени состоит в попытке удовлетворить нескольким взаимоисключающим требованиям ко времени, памяти и т.д. - одно всегда ограничивает другое. При эволюции системы на практике ожидаются следующие типы изменений:

добавление нового класса или нового взаимодействия между классами;

изменение реализации класса;

изменение представления класса;

реорганизация структуры классов;

изменение интерфейса класса.

Первичным результатом эволюции является серия исполнимых релизов, представляющих итеративные усовершенствования изначальной архитектурной модели. Вторичным продуктом следует признать выявление поведения, которое используется для исследования альтернативных подходов и дальнейшего анализа "темных углов" системы. В начале процесса разработки основные действующие релизы передаются разработчиками контролерам качества, которые тестируют их по сценариям, составленным при анализе, и накапливают информацию о полноте, корректности и устойчивости работы релиза. Это раннее накопление данных помогает при выявлении проблем качества, которые будут учтены в следующих релизах. Позднее действующие релизы передаются конечным (альфа и бета) пользователям некоторым управляемым способом. Это означает, что разработчики тщательно выверяют требования к каждому релизу и определяют аспекты, которые желательно проверить и оценить.

## 4.7 Сопровождение

Этот этап разработки представляет собой управление эволюцией продукта в ходе его эксплуатации. Он в значительной степени продолжает предыдущие фазы, за исключением того, что вносит меньше архитектурных новшеств. Вместо этого делаются более локализованные изменения, возникающие по мере учета новых требований и исправления старых ошибок. Кроме тех требований, которые по каким-либо причинам не были учтены, вероятно, уже вскоре после выпуска работающей системы, разработчики и конечные пользователи должны обменяться множеством пожеланий и предложений, которые они хотели бы увидеть воплощенными в следующих версиях системы. Чем больше с системой работает пользователей, тем больше может выявиться новых ошибок. Составляется список, в который заносятся обнаруженные дефекты и новые требования, которые будут учтены при планировании новых релизов в соответствии с их приоритетом.

Типичный порядок действий на этом этапе таков:

упорядочивают по приоритетам предложения о крупных изменениях и сообщения об ошибках, связанных с системными проблемами;

составляют список этих изменений и принять их за функциональные точки в дальнейшей эволюции;

если позволяют ресурсы, планируют в следующем релизе менее интенсивные, более локализованные улучшения;

приступают к разработке следующего эволюционного релиза программы.

## 4.8 Заключение

В завершение этой главы можно сказать, что объектно-ориентированный метод проектирования со времени своего появления и до настоящего момента стал основным и, с уверенностью можно утверждать, лучшим средством в разработке программного обеспечения информационных систем. Этот метод не дает универсальных рецептов однозначно ведущих к успеху в разработке ПО. Как отмечает Страуструп, "не существует рецептов, которые могли бы заменить ум, опыт и хороший вкус в проектировании и программировании... Различные фазы программного проекта, такие, как проектирование, программирование и тестирование, неотделимы друг от друга". Однако, несмотря на это, процесс объектно-ориентированного анализа и проектирования определен достаточно хорошо, чтобы быть предсказуемым и воспроизводимым в умелых руках.

Необходимо подчеркнуть, что каждый проект уникален, и, следовательно, разработчик, если он придерживается объектно-ориентированной методикой разработки, сам должен поддерживать баланс между неформальностью микропроцесса и формальностью макропроцесса, описанных выше.

## 5. Методика отладки и результаты работы программы

## 5.1 Особенности тестирования программных продуктов

Программы как объекты тестирования имеют ряд особенностей, которые отличают процесс тестирования от традиционного, применяемого для проверки аппаратуры и других технических изделий. С этой позиции основными особенностями программ являются:

отсутствие полностью определенного эталона, по которому должны соответствовать все результаты тестирования проверяемого программного обеспечения;

высокая сложность программ и принципиальная невозможность построения тестовых наборов, достаточных для их исчерпывающей проверки;

невысокая степень формализации критериев качества процесса тестирования и достигаемого при этом качества программного обеспечения как объекта тестирования;

наличие в программах вычислительных компонент, а также компонент, характеризующихся стохастическим и динамическим поведением.

Неоднократно экспериментально установлено, что в любом сложном программном обеспечении в процессе эксплуатации обнаруживаются ошибки, даже если проведено самое тщательное тестирование. Тем самым объективно утверждается, что невозможно формализовать и обеспечить абсолютную полноту всех эталонных значений, а также провести всеобъемлющее исчерпывающее тестирование и гарантированно устранить все ошибки в сложных программных продуктах. Поэтому тестирование проводится в объемах, минимально необходимых для проверки программ в некоторых ограниченных пределах изменения параметров и условий функционирования. Ограниченность ресурсов тестирования привела к необходимости тщательного упорядочения методов и конкретных значений параметров с целью получения при тестировании наибольшей глубины проверок программ. Анализ многих проектов показывает, что до начала тестирования число ошибок в сложных программах составляет порядка 1-2% от общего числа объектных команд в программе, т.е. в программном обеспечении объемом 100 тысяч команд в процессе тестирования обычно выявляется 1-2 тысячи ошибок. При тщательном системном проектировании и программировании на языках высокого уровня начальное число ошибок в несколько раз меньше. Таким образом самое тщательное тестирование сложных программных комплексов позволяет получить программы с вероятностью ошибки в каждой команде порядка 10-4-10-5, т.е. несколько ошибок может остаться не выявленными.

## 5.2 Типичный процесс тестирования программного обеспечения

Процесс тестирования программ обычно включает:

создание совокупности тестовых эталонных значений и правил, которым должна соответствовать программа по выполняемым функциям, структуре, правилам описания, значениям исходных и соответствующих им результирующих данных;

статическое тестирование текстов разработанных программ и данных на выполнение всех заданных правил построения и описания без исполнения объектного кода;

тестирование программы с её исполнением в объектном коде и с разными уровнями детализации: детерминированное, стохастическое, и тестирование в реальном масштабе времени;

диагностику и локализацию причин отклонения результатов тестирования от заданных эталонных значений и правил;

разработку изменения программы с целью исключения причин отклонения результатов от эталонных;

реализацию корректировки программы, обеспечивающую соответствие программы заданному эталону.

Статическое тестирование является наиболее формализованным и автоматизируемым методом проверки корректности программ. В качестве эталонов применяются правила структурного построения программных модулей и обработки данных, конкретизированные для проекта в целом. Кроме того, могут использоваться некоторые частные правила обработки данных, зафиксированные в спецификациях на отдельные компоненты программ. Проверка степени выполнения этих правил проводится без исполнения объектного кода программы путём формального анализа текста программы на языке программирования. Операторы и операнды текста программ при этом анализируются в символьном виде, поэтому такой метод называют также символическим тестированием. Развитие и углубление символического тестирования может доводиться до уровня формальной верификации программы на соответствие её текста детальной спецификации совокупности утверждений, полностью определяющей связи между входными и выходными данными этой программы.

Наиболее трудоёмкими и детализирующими являются методы детерминированного тестирования. При детерминированном тестировании контролируется каждая комбинация исходных эталонных данных и соответствующая ей комбинация результатов функционирования программы. Это позволяет выявлять отклонение результатов от эталона с конкретным фиксированием всех значений исходных и результирующих данных, при которых это отклонение обнаружено.

В сложных программах невозможно перебрать все комбинации исходных данных и проконтролировать результаты функционирования программы на каждой из них. В таких случаях применяется стохастическое тестирование, при котором исходные тестовые данные задаются множествами случайных величин с соответствующими распределениями и для сравнения полученных результатов используются также распределения случайных величин. В результате при стохастическом тестировании возможно более широкое варьирование исходных данных, хотя отдельные ошибки могут быть не обнаружены, если они мало искажают средние статистические значения или распределения. Стохастическое тестирование применяется в основном для обнаружения ошибок, а для диагностики и локализации ошибок приходится переходить к детерминированному тестированию с использованием конкретных значений параметров из области изменения ранее использовавшихся случайных величин.

Последующее расширение области изменения исходных данных возможно при применении тестирования в реальном масштабе времени. В процессе такого тестирования проверяется исполнение программ и обработка исходных данных с учётом времени их поступления, длительности и приоритетности обработки, динамики использования памяти и взаимодействия с другими программами и т.д. При обнаружении отклонений результатов исполнения программ от предполагавшихся эталонных для локализации ошибки приходится фиксировать время и переходить к детерминированному тестированию.

## 5.3 Особенности задачи в приложении к тестированию программ

## 5.3.1 Особенности среды программирования

Среда программирования Microsoft Visual Studio 6.0 и язык программирования в ней Microsoft Visual C++ имеют ряд особенностей, влияющих на тестирование программ:

Visual C++ является языком программирования высокого уровня, что сильно увеличивает значимость статического (символьного) тестирования;

среда программирования разработана для поддержки высокой степени использования стандартных поставляемых в модулях процедур. Эти модули являются уже оттестированными и испытанными. Считается, что вызываемые процедуры обладают высокой надёжностью и не содержат ошибок;

Visual Studio имеет развитую систему автоматической символьной проверки. Она на этапе написания программы следит за корректностью вводимого текста и обнаруживает все синтаксические ошибки;

в комплект Visual Studio входит множество утилит для отладки, значительно облегчающих процесс тестирования программного обеспечения;

большую часть структуры программы в Visual C++ with Microsoft Foundation Classes занимают стандартные, автоматически создаваемые объекты, управляющие работой программы в ответ на действия пользователя. Поэтому ошибки в основном содержатся в алгоритмах процедур, написанных программистом.

## 5.3.2 Основные факторы, влияющие на надежность разрабатываемой системы

Представленная программа регистрации процеса прохождения партий по технологическому маршруту, в силу предъявляемых к нему требований, создана для внесения строго определенной информации. Исходя из этого, можно выделить главные факторы, влияющие как на надежность данного программного обеспечения, так и на надежность любого программного обеспечения:

корректность структуры программы;

корректность обработки данных;

устойчивость к ошибочному вводу данных пользователем.

Устойчивость к ошибочному вводу данных пользователем осуществлена при помощи стандартных программных средств, реализованных в библиотеке MFC. Вводятся следующие ограничения на поля данных:

недопустимо непустое значение поля данных;

недопустимо значение поля данных, не входящее в заданный интервал допустимых значений.

В случае ввода неверных данных пользователю выводится на экран соответствующее сообщение.

## 5.3.2.1 Контроль структуры программы

Корректность структуры программ определяется корректностью структуры модулей и корректностью структуры групп программ, построенных из модулей. Для оценки корректности структуры программ используется несколько частных показателей, различающихся степенью охвата тестами структурных компонент программы при тестировании. Проверка корректности структурных компонент производится статически по исходным текстам программ или динамически при исполнении программы в объектном коде. При обоих методах по мере углубления критериев тестовых проверок возрастает объем и сложность тестов, а следовательно и достигаемая степень корректности программ. Рассмотрим один из наиболее часто используемых критериев выделения тестируемых маршрутов для проверки корректности структуры программы.

Суть критерия состоит в выборе и анализе базовых маршрутов в программе, формируемых и оцениваемых на основе определения цикломатического числа исходного графа, определяющего структуру программы или ее фрагмента. Критерий наиболее широко применяется для оценки сложности и корректности тестирования программных модулей. Для определения цикломатического числа *Z* исходного графа фрагмента программы используется полное число его вершин *Nп*, связывающих их дуг *Y* и связных компонент *L*: *Z = Y - Nп + L*. Вычисление цикломатического числа осуществляется по величинам, определяемым по максимально связному графу, который получается из исходного путем замыкания дуги из конечной вершины в начальную. В результате в максимально связном графе появляются маршруты между любыми двумя вершинами. При этом предполагается, что исходный граф корректен, т.е. не содержит "висячих" и "тупиковых" вершин. В таком графе цикломатическое число равно максимальному количеству его линейно-независимых маршрутов. Величина *L* соответствует количеству связных компонент всего исходного графа, и обычно для графов не содержащих полностью независимых частей *L=1*. Иначе говоря, *L* измеряется количеством замыкающих дуг, необходимых для преобразования исходного графа в максимально сильно связный граф.

Таким образом, критерий учитывает каждый линейно-независимый цикл в максимально связном графе программы. Каждый линейно-независимый маршрут или цикл отличается от всех остальных хотя бы одной вершиной или дугой, т.е. его структура не может быть полностью образована компонентами других маршрутов.

**1**) qtyCompl=operations.Qty();

**2**) if (!flags) {

**3**) while (counter<qtyCompl)

{ **4**) routes.status[counter]=1;

**5**) counter++; }

}

**6**) operations.AddOp();

Рис.16. Фрагмент исходного текста программы

Рассмотрим на примере фрагмента кода разрабатываемого программного обеспечения тестирование корректности структуры этого фрагмента (см. рис.16). На основе данного фрагмента кода выделим операторы управления и построим графовую модель программы по управлению. Цифрами на рис.16 условно обозначены вершины графа. Рис.17 отображает схему этого графа.



Рис.17. Граф программы по управлению

Далее определим цикломатическое число графа по исходным данным:

Nп = 6

Y = 8

L = 1

Z = Y - Nп + L = 3

В данном случае для максимально связного графа количество линейно-независимых маршрутов для которых следует подготовить тесты равно цикломатическому числу графа. Эти маршруты показаны на рис.18.

1 - 2 - 3 - 6

1 - 2 - 3 - 4 - 5 - 3 - 6

1 - 2 - 6

Рис.18. Линейно-независимые маршруты

В итоге для проверки корректности структуры данного фрагмента кода необходимо провести статическое и динамическое тестирование четырех полученных маршрутов.

## 

## 5.3.2.2 Контроль чтения и записи переменных

Корректность обработки данных определяется степенью тестирования процесса обработки представительной выборки значений переменных в диапазонах их изменения. Для сложных программ принципиально невозможно провести исчерпывающее тестирование при всех значениях переменных, которые могут появиться при реальном функционировании программ.

Контроль чтения и записи переменных включает в себя две задачи: построение диагностической модели программы с учетом информационных связей и просмотр модели с целью выявления недопустимых сочетаний операций считывания и записи величин. Схема структурных связей в виде направленного графа программы используется как база для построения на модели программы информационных связей между операторами. Пример построения графа фрагмента программы приведен в предыдущем подразделе. Схема контроля детально показана на рис. 19.

Контроль записи и чтения переменных сводится к последовательному просмотру частей программы, которые образуют маршруты, ведущие от места считывания некоторой величины ко входам программы. При просмотре устанавливается:

существует ли запись этой величины вообще;

на всех ли путях, ведущих ко входам, есть запись соответствующей величины перед считыванием с той же области памяти;

не забивается ли считываемая величина записью другой величины, т.е. нет ли между записью и чтением дополнительной записи в ту же область памяти другой величины.



Рис. 19. Схема контроля чтения и записи переменных

Также по всем маршрутам, ведущим из начальной вершины графа в конечную, выполняется контроль наличия запрещенных сочетаний операций по всем наименованиям величин. Условиями правильной реализации являются:

предварительная запись в область памяти перед считыванием из нее информации;

наличие в индексной переменной значения индекса требуемого наименования.

В итоге все обнаруженные ошибки и сомнительные сочетания операций фиксируются и исправляются в тексте программы.

## 

## 5.4 Результаты работы программы

Ниже приведены результаты работы программы.

На рис.20 показано главное меню программы, содержащее помимо стандартных элементов управления (File, Windows, Help) специальные. Существенные объекты выделены в отдельное выпадающее меню Objects.

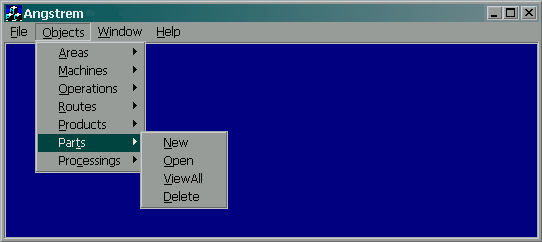


Рис. 20. Главное меню программы

На рис.21. показано окно вывода информации о процессе обработки, выполняемой над партией полупроводниковых пластин с заданным номером в пределах заданного изделия. При этом выводится вся необходимая информация о маршруте, параметрах и режимах технологической операции, рассчитывается время окончания технологической операции.

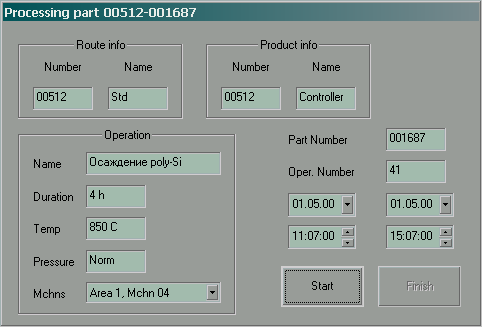


Рис.21. Окно вывода результатов выполнения технологической операции

На рис.22 показано окно вывода информации о техологических операциях, произведенных над партией полупроводниковых пластин с момента ее запуска в производство. В приведенной таблице отображаются не только номера и названия технологических операций, но и параметры обработки, время фактического начала и окончания обработки. Расположенные над таблицей фильтры позволяют выбрать диапазон, охватывающий либо весь период существования партии полупроводниковых пластин с момента ее запуска в производство, либо заданный интервал времени и отобразить операции, произведенные над партией пластин в этот период.

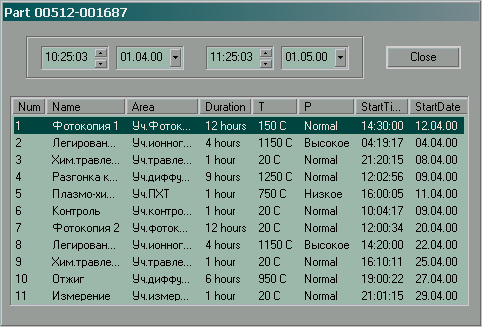


Рис.22. Окно вывода информации о технологических операциях, произведенных над партией полупроводниковых пластин

## 6. Организационно-экономическая часть

## 6.1 Введение

Особенности процесса создания и освоения новой техники, в том числе постановки сложной современной продукции в производстве, связаны с чрезвычайно большим количеством разнообразных операций и действий, выполняемых различными подразделениями и коллективами. Во многих случаях эта коллективы не соподчинены, имеют противоречивые интересы и не управляются из единого центра. В то же время все работы и процессы должны выполняться слаженно, согласованно и по возможности в кратчайшие сроки. Именно этого требуют задачи научно-технического прогресса, успешное решение которых чрезвычайно актуально для нашей отечественной промышленности.

Рассмотренные особенности планирования требуют, во-первых, надежной координации всех звеньев и подразделений, выполняющих тот или иной проектируемый комплекс работ, и, во вторых, использования методов оптимизации сроков, затрат и ресурсов, необходимых для реализации проекта. К наилучшим планово-координационным инструментам из известных в настоящее время относятся модели, на базе которых созданы системы сетевого планирования и управления (СПУ).

Работы по развитию и использованию СПУ получили большое распространение за рубежом. В нашей стране многими организациями и предприятиями также накоплен опыт применения систем СПУ. Они эффективно используются в разных отраслях при создании и освоении новой техники, при строительстве и пуске крупных объектов, ремонте уникальных агрегатов, координации крупных научно-исследовательских работ, реализации межотраслевых проектов. Следует в то же время отметить, что большим достижением использования СПУ в нашей стране является то, что с самого начала ее внедрения были приняты меры по обеспечению единообразия терминов, обозначений, методов расчетов, - словом, по максимальной унификации системы.

Система СПУ создана в первую очередь для планирования комплекса работ, носящих характер неопределенности. С помощью сетевых графиков и методов их расчета (рассмотренных ниже) успешно преодолеваются осложнения, связанные о неопределенностью в оценках времени, необходимого для выполнения отдельных работ, в сроках достижения промежуточных и конечного результатов, в потребных ресурсах.

Сетевые графики обеспечивают строгую увязку и координацию всех работ, входящих в комплекс проекта.

Определение величины затрат, связанных о выполнением исследовательских, конструкторских, технологических этапов, с изготовлением опытного образца, в значительной море облегчается при применении методов СПУ, поскольку становятся известными оценки времени с достаточной для стохастических процессов точностью. Подсчет количества и состава участников работы производится согласно ожидаемым затратам времени (или исходя из временных ограничений) на каждую работу. Определение после этого затрат на заработную плату, материалы, … осуществляется известными методами, а уровни накладных расходов берутся в соответствии с данными конкретного предприятия.

Особенно целесообразно использовать методологию СПУ, когда планируются комплексные разработки, имеющие вероятностный характер как по составу работ, так и по времени их выполнения.

Необходимым условием планирования и управления работами является возможность не только оценки текущего состояния, но и предсказания дальнейшего хода работ и, таким образом, воздействия на их выполнение (при этом имеется в виду, что весь комплекс работ выполнен в сжатые сроки и о наименьшими затратами).

График как элемент системы планирования и управления должен отражать те стороны выполняемых работ, которые являются существенными в отношении достижения конечных целей.

Основным элементом системы является сетевой график. Это плановая модель процесса, на которой можно провести эксперименты и выявить, к каким результатам приведет то или иное изменение сроков выполнения отдельных работ.

В системе используется графическое отражение логической последовательности, взаимосвязи и продолжительности работ.

Сетевые графики можно классифицировать по объему и степени детализации, по степени определенности, по числу достигаемых целей (результатов), по вероятности сроков выполнения отдельных приемов и работ.

По объему и степени детализации различают:

первичные,

частные,

комплексные сети.

По степени определенности:

детерминированные сети, когда известна конкретная цель разработки и основные процессы, ведущие к ее достижению,

стохастические сети, когда возможны альтернативные варианты хода выполнения работ.

По числу конечных целей модели подразделяются на:

одноцелевые, когда заключительный результат всего комплекса работ в сети является единственной целью данной работы.

многоцелевые, когда выполнение всего комплекса работ приводит к получению нескольких заключительных результатов, сеть в этом случае завершается несколькими событиями.

Исходя из признака вероятности сетевые модели могут создаваться как на основе нормативных данных о продолжительности включаемых в модель работ (детерминированные модели), так и на осаове вероятностной (предположительной) оценки их продолжительности (вероятностные сетевые модели). Кроме сроков выполнения работ, сетевье модели могут включать и другие важные данные: величину затрат на выполнение действий, наличие ресурсов и т.п.

СПУ состоит из следующих стадий:

предпроектной, включающей выбор объекта планирования;

создания сетевой модели, основанного на перечне всех работ и событий (промежуточных результатов);

оптимизации модели по установленным критериям (например, срокам или затратам);

управления с помощью сетевого графика на основе поступающей информации о ходе выполнения работ на данный период.

В данной работе рассматривается детерминированная одноцелевая модель. При исследовании рассматриваемой модели будут пройдены все стадии сетевого планирования и управления, включая выбор объекта планирования, создание сетевой модели, анализ и оптимизация.

## 6.2 Сетевая модель, ее основные элементы, правила построения

Основным плановым документом в системах СПУ является сетевой график, представляющий собой информационно-динамическую модель, в которой отражаются взаимосвязи и результаты всех работ, необходимых для достижения конечной цели разработки. Кроме того, модель содержит и другие характеристики (временные, стоимостные, ресурсные), относящиеся к отдельным работам или ко всему комплексу. В сетевом графике детально или укрупненно показывается последовательность, время, условие, выполнение которого обеспечит окончание всех работ не позже заданного (директивного) срока. В основе сетевого моделирования лежит изображение планируемого комплекса работ в виде графа. Граф - это схема, состоящая из заданных точек (вершин), соединенных определенной системой линий. Отрезки, соединящие вершины, называются. ребрами (дугами) графа. Ориентированным нарывается, такой граф, на котором стрелкой указаны направления всех его ребер (дуг), т.е. для каждой дуги указано, какая из двух ее вершин является начальной, а какая - конечной.

Математическая теория графов оперирует понятием пути, под которым понимается такая последовательность ребер, когда конец каждого предыдущего ребра совпадает о началом последующего.

Сетевой график - это ориентированный граф без контуров, ребра которого имеют одну или несколько числовых характеристик. Понятие контура означает конечный путь, у которого начальная вершина совпадает с конечной.

Таким образом, в отличие от ленточного графика, где основным является только один элемент - работа, в сетевом графике, как правило имеются два основных элемента - работа и событие.

Работами называются любые процессы, действия, приводящие к достижению определенных результатов (событий).

Кроме работ действительных, требующих затрат времени, в сетевых моделях могут содержаться так называемые фиктивные работы (зависимости). Фиктивной работой (зависимостью) называется связь между отдельными результатами работ (событиями), не требующая затрат времени вообще или требующая минимальных затрат времени. К таким работам относится, в частности, передача по телефону необходимой информации.

Работа в сетевом графике изображается стрелкой. Каждая стрелка означает затраты времени, необходимого для выполнения соответствующей работы. Величина затрачиваемого на работу времени указывается над стрелкой. Для отображения на графике фиктивных работ используются пунктирные стрелки, над которыми время на указывается.

Событиями называются результаты производственных работ. Каждое событие может быть отправным моментом для начала последующих работ. В отличие от работы, имеющей обычно протяженность во времени, событие представляет собой только момент свершения работы (или работ).

Первоначальное событие в сети, не имеющее предшествующих событий и отражающее начало выполнения всего комплекса работ, включенных в данную сеть, называется исходным и обозначается символом J. Событие, которое не имеет последующих событий и отражает конечную цель комплекса работ, включенных в данную сеть, называется завершающим и обозначается символом C.

Любая последовательность работ в сетевом графике, в которой конечное событие одной работы совладает с начальным событием следующей за ней работы, называется путем.

В сетевом графике следует различать несколько видов путей:

от исходного события до завершающего события - полный путь, или просто путь;

от исходного события до данного - путь, предшествующий данному событию;

от данного события до завершающего - путь, последующий за данным событием;

путь между исходным и завершающим событием, имеющим наибольшую продолжительность - критический путь.

Все события и работы, которые необходимо выполнить для peaлизации проекта, необходимо систематизировать при составлении перечня событий и работ.

В перечне указываются кодовые номера и наименования событий, начиная с нулевого или первого, определяющего решение о начале работ, перечисляются работы, выходящие из каждого события, даются их коды. Код работы состоит из кодов начального и конечного для данной работы событий, следующие графы перечня предусматривают простановку оценок времени выполнения работ.

Таблица 1. Перечень событий

|  |  |
| --- | --- |
| Номер | Событие |
| 0 | Решение о начале работ принято |
| 1 | Рабочая документация проанализирована |
| 2 | Опрос работников предприятия произведен |
| 3 | Существующие решения проанализированы |
| 4 | Полученная ннформация систематизирована |
| 5 | Аппарат имитационного моделирования создан |
| 6 | Разработаны возможные варианты дисциплины очереди |
| 7 | База данных разработана |
| 8 | Имитационная модель написана |
| 9 | Варианты протестированы |
| 10 | Имитационная модель сопряжена с базой данных |
| 11 | Пользовательский интерфейс разработан |
| 12 | Оптимальный вариант дисциплины очереди выбран |
| 13 | Имитационная модель цеха разработана |
| 14 | Имеющиеся ресурсы проанализированы |
| 15 | Аппаратная часть системы выбрана |
| 16 | Программные технологии системы выбраны |
| 17 | Система разбита на взаимосвязанные блоки |
| 18 | Разработка компьютерной системы завершена |

0

1

3

2

4

7

6

9

12

13

14

5

8

10

17

16

18

15

11

Рис.23. Сетевой график разработки автоматизированной системы управления производственным процессом на предприятии электронной промышленности

Таблица 2 Перечень работ

|  |  |
| --- | --- |
| Номер | Содержание работы |
| 0-1 | Анализ рабочей документации |
| 1-3 | Анализ существующих решений в управлении |
| 3-4 | Систематизация полученной информации |
| 0-2 | Опрос работников предприятия |
| 2-3 | Передача информации |
| 3-7 | Разработка базы данных |
| 7-8 | Передача информации |
| 4-6 | Разработка возможных вариантов дисциплины очереди |
| 6-9 | Тестирование вариантов дисциплины очереди |
| 9-12 | Выбор оптимального варианта |
| 12-13 | Разработка имитационной модели |
| 4-5 | Создание аппарата имитационного моделирования |
| 5-8 | Написание имитационной модели |
| 8-10 | Сопряжение имитационной модели с базой данных |
| 10-12 | Передача информации |
| 4-11 | Разработка пользовательского интерфейса |
| 11-12 | Передача информации |
| 13-14 | Анализ имеющихся программных и аппаратных ресурсов |
| 14-17 | Разбиение проектируемой системы на взаимосвязанные блоки |
| 17-18 | Разработка компьютерной системы |
| 14-15 | Выбор аппаратной части системы |
| 15-17 | Передача информации |
| 14-16 | Выбор программных технологий компьютерной системы |
| 16-17 | Передача информации |

## 6.3 Расчет параметров сетевой модели

К основным параметрам сетевой модели относятся критический путь, резервы времени событий и работ. Эти параметры являются исходными для получения ряда дополнительных характеристик, а также для анализа сети или, что то же самое, для анализа составленного плана разработки.

Расчет параметров сети имеет целью определение их численных величин: продолжительности критического пути, размеров резервов времени событий и работ и др. Этот расчет приводится далее.

В результате расчета параметров и анализа их величин появляется возможность оптимизации сетевого графика по выбранным или назначенным критериям. Расчет параметров и нахождение их определенных величин возможны для детерминированных и условно-детерминированных моделей.

Критический путь - это наиболее протяженная по времени цепочка работ, ведущих от исходного к завершающему событию. Изменение продолжительности любой работы, лежащей на критическом пути, соответственным образом меняет (приближает или отдаляет) момент наступления завершающего события, т.е. дату достижения конечной цели разработки.

Поскольку продолжительность критического пути определяет срок выполнения всего комплекса работ, в процессе управления ходом разработки внимание руководства сосредотачивается на главном направлении - на работах критического пути. Это позволяет более рационально и оперативно контролировать ограниченное число работ, влияющих на срок разработок, а также лучше использовать имеющиеся ресурсы (трудовые и материальные).

Критические пути резервами не располагают. На сетевом графике они изображаются обычно жирной линией.

В сетевых графиках имеются и другие пути, опирающиеся на исходное и завершающее события (полные пути), которые могут либо полностью проходить вне критического пути, либо частично совпадать с критической последовательностью работ. Эти пути называются ненапряженными. Они по продолжительности меньше критического пути.

Ненапряженные пути обладают важным свойством: на участках, не совпадающих с критической последовательностью работ, они имеют резервы времени.

Кроме критического пути, другими основными параметрами сетевой модели являются резервы времени наступления события и разновидности резервов времени работ. Резервы времени существуют в сетевом графике во всех случаях, когда имеются два или более путей разной продолжительности. Резерв времени события - это такой промежуток времени, на который может быть отсрочено наступление данного события без нарушения сроков завершения разработки в целом. Резерв времени события определяется как разность между поздним Тп и ранним Тр сроками наступления события:

R = Т п - Т р.

Поздний срок наступления события Тп - максимальный из допустимых моментов наступления данного события, при котором возможно соблюдение расчетного срока наступления завершающего события. Превышение Тп вызовет аналогичную задержу наступления завершающего события. Иными словами, если событие наступило в момент Тп, оно попало в критическую зону и последующие за ним работы должны находиться под таким же контролем, как и работы критического пути.

Ранний срок наступления события Тр - минимальный срок, необходимый для выполнения всех работ, предшествующих данному событию. Это время находится путем выбора максимального значения из продолжительностей всех путей, ведущих в данному событию.

Если обозначить предшествующее событие i, а последующее j, то ранний и поздний сроки свершения событий будут обозначаться соответственно Трi, Тпi, Трj, Тпj, т.е. рядом с индексом, определяющим характер срока свершения события (ранний срок - р, поздний - п), пишется номер события.

В каждой сети для некоторых событий, которые имеют нулевой резерв времени, наибольший допустимый срок равен наименьшему ожидаемому. Путь, соединяющий эти события, и является критическим, т.е. он соответствует максимальной продолжительности последовательно выполняемых работ, ведущих от исходного к завершающему событию. Исходное и завершающее события во всех случаях имеют нулевой резерв времени.

Зная ранние и поздние сроки наступления событий, можно для любой работы (i,j) определить также ранние и поздние сроки начала и окончания работы.

Ранний срок начала работы:

Т рн ij = Т р i

поздний срок ее начала:

Т пн ij = Т п j - t ij

ранний срок окончания работы:

Т ро ij = Т р i + t ij

поздний срок окончания работы:

Т по ij = Т п j.

Все названные сроки можно определять как календарные (задаваемые от начального момента календарной датой).

Рис.24. Параметры сети, отмеченные непосредственно на графе: продолжительность работ, количество исполнителей, ранние и поздние сроки наступления событий, резервы времени событий.

2

5

9

4

3

0

9

9

4

0

17

17

5

2

26

24

1

0

4

4

0

0

0

0

8

2

40

38

9

0

40

40

10

2

45

43

11

26

45

19

7

12

40

28

6

0

25

25

14

0

57

57

16

6

67

61

15

6

67

61

17

0

67

67

13

0

55

55

12

0

45

45

18

0

82

82

4

4

5

19

8

8

2

15

5

14

5

10

2

10

15

4

4

2

5

2

6

2

3

3

4

4

4

2

5

4

2

5

2

3

4

7

Резервами времени располагают не только события, но и пути (кроме критического), а также работы, лежащие на некритических путях. Для определения полного резерва времени пути R (Li) следует опять вернуться в тому условию, что длина критического пути в сетевом графике больше, чем длина любого другого полного пути. Разница между длиной критического пути t (Lкр) и длиной любого другого пути t (Li) называется полным резервом времени пути:

R (Li) = t (Lкр) - t (Li)

Полный резерв пути показывает, насколько могут быть увеличены продолжительности всех работ, принадлежащих пути Li, в сумме или, иными словами, каково предельно допустимое увеличение продолжительности этого пути.

При определении резерва времени работы следует учитывать, что работа может принадлежать нескольким путям одновременно. Полный резерв времени этой работы равен резерву времени максимального по продолжительности из путей, проходящих через эту работу и не может быть больше его.

Полный резерв времени работы - это максимальное количество времени, на которое можно увеличить продолжительность данной работы, не изменяя при этом продолжительности критического пути. Полный резерв времени работы:

R п ij = Т п j - Т р i - t ij

где i - начальное событие данной работы; j - конечное событие этой работы; Тпj и Трi - соответственно поздний и ранний срок свершения событий i и j.

Важным свойством полного резерва времени работы является то, что при использовании его частично или целиком для увеличения длительнооти какой-либо работы соответственно уменьшится резерв времени всех последующих работ, лежащих на этом пути. При использовании полного резерва времени целиком для одной работы резервы времени остальных работ, лежащих на максимальном пути, проходящем через нее, будут полностью исчерпаны, поскольку полный резерв времени работы принадлежит не только ей, но и всем работам, лежащим иа путях, проходящих через данную работу. Резервы времени работ, лежащих на других (не максимальных по продолжительности) путях, проходящих через эту работу, сократятся и будут равны разности между прежним резервом времени этих работ и использованным резервом времени работы, лежащей на максимальном пути.

У отдельных работ помимо полного резерва времени имеется свободный резерв времени Rс, который равен разности между ранними сроками наступления начального i и конечного j событий за вычетом продолжительности работы t ij:

R с ij = Т р j - Т р i - t ij

Свободный резерв времени - это максимальное количество времени, на которое можно увеличить продолжительность работы или отсрочить ее начало, не изменяя при этом ранних сроков начала последующих работ, при условии, что начальное событие этой работы наступило в свой ранний срок.

Если начальное событие i наступает не в свой ранний срок, а в какой-то фактический Тф, то работа (i,j) будет располагать свободным фактическим резервом времени:

R сф ij = Т р j - Т ф i - t ij

Используя свободный резерв времени, ответственные исполнители могут беспрепятственно маневрировать в его пределах сроком начала данной работы или ее продолжительностью. Это важное свойство свободного резерва времени должно учитываться исполнителями при независимом выполнении отдельно взятых работ, располагающих этим резервом.

Полные резервы времени работы так же позволяют маневрировать сроками начала и окончания работы, но в этом случае изменяются резервы последующих работ. При отсутствии резервов возможности подобного маневрирования нет.

В то же время необходимо правильно использовать имеющиеся резервы. Процесс оптимизации сетевых графиков, как показано в дальнейшем, в значительной мере связан о использованием резервов времени работ. Ответственные исполнители и руководители работ должны знать величину резервов и возможности их использования.

Полный резерв времени работы может быть распределен, как отмечалось, между работами, находящимися на пути, которому принадлежит данная работа. При этом и начальное, и конечное события этой работы могут располагать резервами времени.

Свободный резерв - это независимый резерв. Его использование на какой-либо работе не меняет величины свободных резервов остальных работ сети, так как при его исчислении в качестве плановых сроков начала выполнения всех работ приняты ранние сроки наступления событий.

Таблица 3. Параметры сетевой модели

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | j | Трi | Тпi | Трj | Тпj | Tij | Трнij | Тпнij | Троij | Тпоij | Rпij | Rсij |
| 0 | 1 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 4 | 4 | 0 | 0 |
| 1 | 3 | 4 | 4 | 9 | 9 | 5 | 4 | 4 | 9 | 9 | 0 | 0 |
| 3 | 4 | 9 | 9 | 17 | 17 | 8 | 9 | 9 | 17 | 17 | 0 | 0 |
| 0 | 2 | 0 | 0 | 4 | 9 | 4 | 0 | 5 | 4 | 9 | 5 | 0 |
| 2 | 3 | 4 | 9 | 9 | 9 | 0 | 4 | 9 | 4 | 9 | 5 | 5 |
| 3 | 7 | 9 | 9 | 28 | 40 | 17 | 9 | 21 | 28 | 40 | 12 | 0 |
| 7 | 8 | 28 | 40 | 38 | 40 | 0 | 28 | 40 | 28 | 40 | 12 | 10 |
| 4 | 6 | 17 | 17 | 25 | 25 | 8 | 17 | 17 | 25 | 25 | 0 | 0 |
| 6 | 9 | 25 | 25 | 40 | 40 | 15 | 25 | 25 | 40 | 40 | 0 | 0 |
| 9 | 12 | 40 | 40 | 45 | 45 | 5 | 40 | 40 | 45 | 45 | 0 | 0 |
| 12 | 13 | 45 | 45 | 55 | 55 | 10 | 45 | 45 | 55 | 55 | 0 | 0 |
| 4 | 5 | 17 | 17 | 24 | 26 | 7 | 17 | 19 | 24 | 26 | 2 | 0 |
| 5 | 8 | 24 | 26 | 38 | 40 | 14 | 24 | 26 | 38 | 40 | 2 | 0 |
| 8 | 10 | 38 | 40 | 43 | 45 | 5 | 38 | 40 | 43 | 45 | 2 | 0 |
| 10 | 12 | 43 | 45 | 45 | 45 | 0 | 43 | 45 | 43 | 45 | 2 | 2 |
| 4 | 11 | 17 | 17 | 19 | 45 | 2 | 17 | 43 | 19 | 45 | 26 | 0 |
| 11 | 12 | 19 | 45 | 45 | 45 | 0 | 19 | 45 | 19 | 45 | 26 | 26 |
| 13 | 14 | 55 | 55 | 57 | 57 | 2 | 55 | 55 | 57 | 57 | 0 | 0 |
| 14 | 17 | 57 | 57 | 67 | 67 | 10 | 57 | 57 | 67 | 67 | 0 | 0 |
| 17 | 18 | 67 | 67 | 82 | 82 | 15 | 67 | 67 | 82 | 82 | 0 | 0 |
| 14 | 15 | 57 | 57 | 61 | 67 | 4 | 57 | 63 | 61 | 67 | 6 | 0 |
| 15 | 17 | 61 | 67 | 67 | 67 | 0 | 61 | 67 | 61 | 67 | 6 | 6 |
| 14 | 16 | 57 | 57 | 61 | 67 | 4 | 57 | 63 | 61 | 67 | 6 | 0 |
| 16 | 17 | 61 | 67 | 67 | 67 | 0 | 61 | 67 | 61 | 67 | 6 | 6 |

## 

## 6.4 Основы оптимизации сетевого графика

Получение параметров сетевого графика позволяет перейти в следующему этапу сетевого планирования. На этом этапе, который раскрывает одну из основных идей СПУ, выполняется всесторонний анализ созданного графика и предпринимаются меры для его оптимизации. Конечная цель этих работ - приведение сетевого графика в соответствии с заданными сроками и одновременно возможностями организаций, осуществляющих разработку. Анализ сетевого графика, опиравшийся на результаты расчетов, включает оценку целесообразности и структуры графика, загрузки исполнителей работ на всех этапах выполнения проекта, возможности смещения начала работ некритической зоны и, наконец, оценку вероятности наступления события в заданный срок.

Предварительный анализ построенного графа включает просмотр топологии сети, контроль правильности выполнения построения и его структуры, установление целесообразности степени детализации работ.

Следующим этапом анализа может быть классификация и группирование работ по величинам резервов (полных и свободных). При этом следует подчеркнуть, что далеко не всегда величина полного резерва может достаточно точно характеризовать, насколько напряженным является выполнение той или иной работы некритической зоны. Все зависит от того, на какую последовательнооть работ распространяется вычисленный резерв, какова величина (протяженность) этой последовательности. Иными словами, важно не только абсолютное, но и относительное значение резерва работы или отрезка пути.

Определить сложность выполнения в срок каждой группы работ некритического пути можно с помощью коэффициента напряженности - это отношение продолжительности отрезка наибольшего из некритических путей, проходящих через данную работу, к продолжительности несовпадающего отрезка критического пути, проходящего через эту работу.

Если совпадающую с критическим путем величину отрезка пути обозначить t' (Lкр), длину критического пути - t (Lкр), а протяженность максимального из некритических путей, проходящих через данную работу, - t (Lmax), то коэффициент напряженности Kнij будет равен:



Используя зависимость, по которой определяется полный резерв времени пути, формулу можно привести к виду:



Как отмечалось, работы могут обладать одинаковым полным резервом, но степень их критичности (или коэффициент напряженности) будет различной. И наоборот, одинаковым полным резервам могут соответствовать совершенно различные коэффициенты напряженности.

Чем выше коэффициент напряженности, тем сложнее выполнить данную работу в установленные сроки. И наоборот, чем меньше коэффициент напряженности, тем большими относительными резервами обладает данный путь в сети.

По полученным таким образом коэффициентам напряженности все работы распределяются по зонам (критическая, подкритическая и резервная). Критическая зона сетевого графика - это совокупность путей, включая критический, абсолютные значения резервов времени которых меньше заданной величины. Можно сказать и иначе - это совокупность работ, имеющих коэффициенты напряженности, близкие к единице. Как показывает практика, объем критической зоны составляет 10 - 15% всего числа работ сетевой модели. Резервная зона - это совокупность работ, имеющих коэффициенты напряженности значительно меньше единицы.

## 

## 6.5 Анализ и оптимизация сетевой модели

Оптимизация сетевого графика представляет собой провесс улучшения организации выполнения комплекса работ с целью соблюдения установленного срока и выделения ресурсов. Она осуществляется за счет факторов:

перераспределения ресурсов - временных (использование резервов времени), материальных, финансовых, а также ресурсов рабочей силы;

интенсификации выполнения работ критического пути (дополнительное количество исполнителей и оборудования, материальное стимулирование, сверхурочные работы);

параллельного выполнения работ критического пути;

изменения в характере комплекса работ.

Оптимизация сетевого графика в зависимости от полноты решаемых задач может быть условно разделена на частную и комплексную. Видами частной оптимизации сетевого графика является минимизация:

времени выполнения разработки при заданной ее стоимости;

потребляемых (используемых одновременно) ресурсов;

стоимости всего комплекса работ при заданном времени выполнения проекта.

Комплексная оптимизация сетевого графина - это нахождение оптимума в соотношениях величин затрат и сроков выполнения проекта в зависимости от конкретных целей, ставящихся при его реализации. Решение задачи полной оптимизации всецело зависит от значимости и степени срочности выполняемого комплекса работ. В одних случаях целесообразно увеличение затрат по сравнению с их нормальней величиной в целях сокращения сроков, в других превалирует экономия затрат, а с некоторым увеличением сроков разработки можно мириться.

При минимизации времени выполнения работ (оптимизации по критерию "время") общий срок выполнения разработки следует сокращать а первую очередь за счет изменения продолжительности критических работ. Это шаг не связан с изменением топологии, сеть не вычерчивается заново, меняются лишь временные оценки. Следует иметь в виду, что при значительном совращении сроков выполнения критических работ уменьшаются и резервы времени некритических путей, в результате чего постепенно возрастает число критических путей. В перспективе все пути могут стать равнонапряженными и иметь одинаковую продолжительность.

Сокращение продолжительности критического пути может быть достигнуто ответственными исполнителями критических работ за счет внутренних резервов, а также путем привлечения дополнительных ресурсов. Внутренние резервы работ, подчиненных одному ответственному исполнителю, заключаются в интенсификации выполняемых действий в лучшей организации работ, повышении уровня их автоматизации. Напримар, системы автоматического проектирования (САПР) позволяют во много раз ускорить процессы расчетов, отбора лучших вариантов из большого числа просчитанных, быстро найти наилучшее решение. Во многих случаях интенсификация приводит к повышению затрат.

Выделение дополнительных ресурсов для критических работ, естественно, представляется более привлекательным для ответственных исполнителей этих работ, чем интенсификация. Откуда могут быть получены эти ресурсы? Обычно речь идет не о внешних для проекта в целом источниках увеличения ресурсов, а о перераспределении ресурсов (например, работников) между работами резервной и критической зон. Переброска работников с некритических работ на критические сопровождается увеличением сроков выполнения некритических работ и сокращением располагаемых ими резервов. Резервы этих работ будут сокращаться в связи с увеличением их продолжительности и одновременно с сокращением продолжительности критических работ.

Наглядно представить процесс изменения резервов времени и потребности в исполнителях на отдельных этапах выполнения проекта в целом можно с помощью так называемой карты проекта, в которую с этой целью преобразуется исходная сетевая модель. Перед таким преобразованием на самом сетевом графике вводится дополнительная информация: под стрелкой каждой работы даются сведения не только о ее продолжительности, но и о потребности в различиях категориях работников. Карта проекта отражает календарные сроки выполнения работ и необходимое количество работников каждой категории на соответствующем этапе. Такая карта может быть получена из обычной сети после нахождения критического пути.

Ход преобразования сетевой модели в карту проекта следующий. Проводится оценка потребного количества работников для выполнения каждой отдельной работы комплекса. На сетевом графике под стрелкой каждой работы указано число работников, необходимых для выполнения данной работы.

После этого сетевая модель преобразуется в календарный график (карту проекта). Критический путь на сетевом графике изображается жирной прямой линией в масштабе времени вдоль оси абсцисс. Тогда все остальные работы можно отложить выше и ниже критического пути. Они изображаются тонкими стрелками, длина которых пропорциональна продолжительности работ.

Пунктирные линии играют вспомогательную роль (соединение работы с событиями) и дают наглядное представление об имеющихся резервах времени. Если по каким-либо причинам некритическая работа начинается не сразу после наступления предшествующего события, ее обозначают пунктирной линией. Фиктивные работы вычерчиваются только пунктирными линиями. При отсутствии резерва они изображаются на карте вертикальной пунктирной линией.

Под календарным графиком изображаются диаграммы дотребнооти в работниках.

Маневрирование ресурсами работников, сокращение их числа или же уменьшение общей продолжительности комплекса работ возможно при условии двух предпосылок:

если специалисты, выполняющие параллельные работы, взаимозаменяемы;

если эти группы специалистов подчинены одному руководителю.

При отсутствии предпосылок дальнейшая (после рациональной перестройки сети) оптимизация и сокращение сроков могут потребовать выделения дополнительных ресурсов - работников и средств на работы критического пути.

Предположим, что либо мероприятиями но интенсификации работ критического пути. либо переброской исполнителей с работ некритического пути на критический, либо тем и другим образом критический путь сетевого графика совращён.

Следовательно изменяется карта проекта. Иными в этом случае стали и диаграммы потребности в исполнителях вследствие их перераспределения между работами критической и некритической зоны. Следующим этапом является минимизация потребности в исполнителях. Это достигается путем смещения сроков начала (и соответственно окончания) работ напряженных путей.

Работы подкритических путей располагают малыми возможностями для смещения их начала в ту или другую сторону. На графике эти работы образуют почти непрерывную цепочку сплошных горизонтальных линий, лишь местами имеющих пунктирные участки. В то же время работы путей, имеющих большие резервы, допускают значительные вариации во времени их начала. Величины смещений начала работ могут быть найдены по приведенным выше формулам сроков начала и окончания работ.

Рис.25. Карта проекта до оптимизации: календарный график

0

1

3

4

7

6

9

12

13

14

5

8

10

17

16

18

15

0

2

4

4

11

3

14

14

2

2

4

5

2

4

5

4

4

4

3

3

3

5

2

2

2

6



Рис.26. Карта проекта до оптимизации: диаграмма потребности в исполнителях



Рис.27. Карта проекта после оптимизации: диаграмма потребности в исполнителях.

## 6.6 Выводы

В результате рассмотрения методов сетевого планирования и управления можно сделать вывод о ряде преимуществ систем СПУ.

Внимание концентрируется на решающих работах, лежащих на критическом пути и относящихся к ограниченному числу ответственных исполнителей. Управление строится по принципу временного исключения из поля зрения руководства тех работ, которые в данный момент и в ближайшем будущем не могут влиять на общий ход разработки.

Практически реализуется принцип непрерывности планирования хода работ и управления им. Систематическая корректировка плана и своевременное принятие решений полностью подчинены задаче завершения разработки в заданные сроки.

Система СПУ обеспечивает возможность рационального маневрирования выделенными для данной разработки ресурсами времени и средств.

Упрощается и унифицируется отчетная документация.

Обеспечивается возможность использования ЭВМ для составления анализа, расчета и корректировки графиков выполнения работ.

Появляется возможность обосновать и математически оптимизировать план реализации намеченных сложных комплексов работ, что не позволяют другие методы планирования вероятностных процессов.

В рассмотренном сетевом графе для выполнения всего комплекса работ привлекалась только одна категория исполнителей работ - инженеры.

Анализ сетевого графика выявил работы критического пути, общая продолжительность которых составила 82 дня. Оптимизация сетевого графика ставила своей целью обеспечение равномерности загрузки исполнителей на всем протяжении критического пути. В результате были рассчитаны резервы времени событий и работ, в соответствии с которыми были изменены сроки начала некоторых работ, не принадлежащих критическому пути. Это позволило уменьшить максимальную потребность в исполнителях и сократить штатное расписание с 11 до 9 человек. В конечном счете затраты на выполнение всего комплекса работ сократились со 150150 руб. до 122850 руб. Экономия от оптимизации в абсолютном выражении составила 27300 руб., что составляет 18% от первоначальной стоимости.

Таблица 4. Результаты оптимизации

|  |  |  |
| --- | --- | --- |
| Параметры | После  оптимизации | До  оптимизации |
| Штатное расписание, чел. | 9 | 11 |
| Заработная плата инженера, руб. /мес. | 5000 | 5000 |
| Длительность проекта, дн. | 82 | 82 |
| Всего затраты на проект, руб. | 122850 | 150150 |
| Экономия,% | 18 |  |

## 7. Производственная и экологическая безопасность

Организация рабочего места программиста и пользователя ЭВМ.

Выполнил:

Ляшко М.А.

Консультант:

Никулина И.М.

## 7.1 Введение

На современном этапе развития электронной промышленности создание новой техники ставит задачу не только облегчить труд человека, но и привести к изменению его роли и места в производственном процессе. В условиях технического прогресса увеличивается количество объектов, которыми он должен управлять, возрастают скорости управляемых им процессов, широкое применение получает дистанционное управление. В связи с этим возрастает роль охраны труда, призванной не только облегчить труд человека, но и сделать условия труда комфортными.

При использовании человеком даже самой передовой технологии у него могут возникнуть соответствующие профессиональные заболевания, если работая, он будет пренебрегать даже элементарными правилами техники безопасности. Типичными ощущениями, которые испытывают к концу рабочего дня чрезмерно увлеченные и беспечные пользователи персональных компьютеров, являются: головная боль, резь в глазах, тянущие боли в мышцах шеи, рук и спины, зуд кожи на лице и т.д. Испытываемые день за днем, они могут привести к мигреням, частичной потере зрения, сколиозу, кожным воспалениям и другим нежелательным явлениям. Все это не случайно.

Вероятнее всего, человеку уже никогда не удастся полностью избежать пагубного влияния передовых технологий, но, как и во многих других случаях, сами пользователи персональных компьютеров, по крайней мере, могут свести их к минимуму. Большинство проблем решаются сами собой при правильной организации рабочего места, соблюдении правил техники безопасности и разумном распределении рабочего времени.

При рассмотрении вопросов охраны труда большое внимание уделяется производственному освещению, оздоровлению воздушной среды, защите от шума, электробезопасности, пожарной безопасности и др.

Инженер всегда обязан помнить, что первое условие научно-организованного труда состоит в том, чтобы обеспечить безопасность людей.

В процессе работы здоровью людей могут угрожать факторы, вызванные преимущественно неправильной организацией работы либо несоблюдением техники безопасности.

Повышенную утомляемость может вызвать повышенный уровень шума; высокий уровень излучения мониторов либо неконтрастность изображения на них могут привести, также к повышенной утомляемости либо ослаблению зрения. С целью избежания подобных недостатков возможно применение защитных экранов, обеспечение персонала мониторами по возможности с более низким уровнем радиации и надзор за качеством изображения на них.

Непосредственную опасность для жизни и здоровья людей представляют собой приборы и элементы оборудования, требующие для своей работы питания от сети с высоким напряжением. С целью избежания несчастных случаев при использовании человеком подобного оборудования либо контакте с ним, необходимо проведение среди персонала предприятия инструктажей по технике безопасности, а также, соблюдение и контроль соблюдения требований техники безопасности. Кроме того, уменьшить вероятность несчастных случаев или аварий можно путем проведения некоторых организационных и профилактических мер.

Эффективным средством профилактики несчастных случаев является наиболее удачное расположение оборудования, использование, по возможности, приборов и оборудования с наиболее оптимальными конструктивными решениями. Важным средством обеспечения безопасности служит надежная изоляция токонесущих частей, кабелей, а также, заземление корпусов всех приборов и металлических частей оборудования.

## 7.2 Рабочее место программиста

Стандартное автоматизированное рабочее место программиста имеет необходимые составные части:

системный блок, включающий в себя основную аппаратную логику, обеспечивающую нормальное функционирование компьютера;

монитор, являющийся основным средством вывода информации, через который выдается подавляющее количество всей выводимой информации, даже с учетом распространения мультимедиа;

клавиатуру как основное средство ввода, хотя в последнее время стремительно увеличивается значимость манипулятора типа "мышь";

вышеупомянутый манипулятор типа "мышь".

Иногда программист имеет дело с дополнительными периферийными устройствами, такими как принтер, модем, но гораздо реже, поэтому эти устройства не играют решающей роли в обеспечении экологической безопасности рабочего места программиста и исключены из дальнейшего рассмотрения.

Рабочие места программистов и операторов ПЭВМ располагаются в помещениях, подверженном влиянию совокупности производственных факторов, оказывающих влияние на здоровье и работоспособность человека в процессе труда.

## 7.3 Вредные факторы на рабочем месте программиста и пользователя ЭВМ

Даже исправный компьютер постоянно оказывает вредное воздействие на организм человека, которое может причинить большой вред организму человека, если пренебрегать эргономическими и экологическими требованиями. К вредным производственным факторам относятся побочное излучение в различных диапазонах, дополнительные шумы, возникающие при работе компьютера. Группа эргономических факторов включает в себя правильность расположения отдельных частей компьютера, цветовую гамму окружающей обстановки, правильно подобранную палитру интерфейса прикладных программ. Работа с системой связана с воздействием следующих поражающих факторов:

Поражение электрическим током питающей сети.

Пожароопасность.

Излучения экрана монитора.

Шум, возникающий при работе механических и электрических устройств системы.

Физические перегрузки (статические).

Психофизиологические перегрузки:

Утомление, связанное с монотонностью работы,

Перенапряжение зрительных анализаторов,

Умственное напряжение,

Эмоциональные перегрузки в большом коллективе.

Поражающие факторы 1 и 2 не являются источниками постоянно действующей опасности. Их безопасность гарантируется тщательным соблюдением правил техники безопасности эксплуатации системы. Каждый работающий должен ознакомиться с этими правилами, пройти соответствующий инструктаж и следить за соблюдением этих правил. Поражающие факторы 3-6 являются постоянно действующими, возникающими при каждодневной текущей работе.

## 7.4 Нерациональное освещение

Правильно спроектированное и выполненное производственное освещение обеспечивает возможность нормальной производственной деятельности. Сохранность зрения человека, состояние его центральной нервной системы и безопасность на производстве в значительной мере зависят от условий освещения.

Анализируя условия работы программиста получаем следующие требования к производственному освещению:

наименьшая допустимая освещенность от общего освещения составляет 200 лк;

при работе за компьютером желательно, чтобы освещенность рабочего места не превышала 2/3 нормальной освещенности помещения;

экран дисплея не должен быть ориентирован в сторону источников света (окон, настольных ламп и т.п.); при размещении рабочего места рядом с окном угол между экраном дисплея и плоскостью окна должен составлять не менее 90 градусов (для исключения бликов), прилегающую часть окна желательно зашторить;

не следует располагать дисплей непосредственно под источником освещения или вплотную с ним;

стена позади дисплея должна быть освещена примерно так же, как и его экран;

яркость для блестящих поверхностей более 0.2 кв. м не должна превышать 500 кд/кв. м;

показатель ослепленности не должен превышать 40 единиц;

коэффициент пульсаций 10 - 20%.

Специфика работы за ЭВМ, состоит в том, что работать приходится с так называемым самосветящимся объектом.

Свечение со стороны экрана, а также частая смена заставок на экране при большой продолжительности трудовой деятельности может отрицательно воздействовать на зрение.

Такой режим работы утомляет зрительные органы. Поэтому разработчику программного обеспечения следует учитывать этот фактор при проектировании программного обеспечения и его отладке за компьютером.

Для обеспечения нормальной естественной освещенности, площадь оконных проемов должна быть не менее 25% площади пола.

Работу программиста за компьютером можно отнести к классу точных работ. Контрастность текста на экране монитора является средней, фон светлый. Следовательно рекомендуемая освещенность при работе с дисплеем составляет 200 лк.

Рекомендуемые перепады яркости в поле зрения оператора должны лежать в пределах 1: 5 - 1: 10.

## 7.5 Расчет общего освещения

Рассчитаем общее освещение в машинном зале ПЭВМ методом коэффициента использования светового потока по уравнению:

Выбираем рекомендованное для машинного зала люминесцентное освещение.



Располагаем светильники рядами вдоль длинной стороны помещения. Будем использовать светильники типа УСП-35 с двумя лампами типа ЛБ-40.

Для обеспечения наилучших условий освещения, расстояние между рядами светильников L должно соответствовать отношению:



где h-высота подвеса светильников,

где H = 3.2 м - высота помещения,

hc = 0.2 м - свес светильника,

hp = 0.75 м - высота рабочей поверхности от пола.



h = 3.0-0.2-0.75 = 2.25 [м]

⇒ L = λ\*h = 3.1…3.4 [м]

Длина помещения А = 8 м

Ширина помещения В = 6 м

Количество рядов светильников N найдем из уравнения:

L \* (0.33\* 2 + N-1) = B



Количество рядов светильников N = 2 ряда.

Согласно нормам, нормируемая минимальная освещенность при общем освещении: Eн = 200 лк.

Так как запыленность воздуха меньше 1 мг/м³, то коэффициент запаса:

Кз = 1.5

Площадь помещения S = A\*B = 8\*6 = 48 [м²].

Так как мы предполагаем создать достаточно равномерное освещение, то коэффициент неравномерности освещения: z = 1.15.

Индекс помещения:



Коэффициенты отражения светового потока принимаем:

от потолкаρп = 70%,

от стенρс = 50%,

от полаρпола = 10%.

Тогда по таблице находим коэффициент использования светового потока:

η = 0.46.

Так как затенения предполагаем не создавать, то коэффициент затенения:

γ = 1.

По таблице находим световой поток лампы ЛБ-40:

Фл = 3120 лм.

Световой поток светильника: Фсв = 2\*Фл = 6240 [лм].

Количество светильников в одном ряду:



Расположение светильников:

Длина светильника lсв = 1.3 м

Количество светильников в ряду М = 3 шт

Количество рядов светильников N = 2 шт

Так как А - М\*lсв = 4.1<4\*L = 12.8 [м]

(где L - рассчитанное минимальное расстояние между светильниками), то расстояние между светильниками в одном ряду L2 можно сделать равным расстоянию от крайнего светильника в ряду до стены.

Тогда



Расстояние между рядами L1 при расстоянии крайнего ряда от стены 0.33\*L1:



Итак, для нормального освещения машинного зала ПЭВМ используем 6 светильников типа УСП-35 с двумя лампами типа ЛБ-40.

## 7.6 Электроопасность

Помещение машинного зала ПЭВМ не должно относится к категории помещений с повышенной электроопасностью, то есть:

Относительная влажность воздуха в помещении должна быть не более 75%.

Должна отсутствовать токопроводящая пыль.

Не должно быть повышенной температуры воздуха в помещении (температура постоянно или периодически, более одних суток, превышает +35 ºС).

Должна отсутствовать возможность одновременного прикосновения человека к имеющим соединение с землей металлическим конструкциям здания, оборудованию и т.д., с одной стороны, и к металлическим корпусам аппаратуры или токоведущим частям, с другой стороны.

Не должно быть токопроводящих полов.

Основными средствами защиты от поражения электрическим током при работе на компьютере являются защитное заземление и зануление.

Защитное заземление - это преднамеренное электрическое соединение с землей или ее эквивалентом металлических нетоковедущих частей электрического и технологического оборудования, которые могут оказаться под напряжением. Защитное заземление является простым, эффективным и широко распространенным способом защиты человека от поражения электрическим током при прикосновении к металлическим поверхностям, оказавшимся под напряжением. Обеспечивается это снижением напряжения между оборудованием, оказавшимся под напряжением, и землей до безопасной величины.

Опасность прикосновения человека к токоведущим частям электроустановки определяется величиной протекающего через тело человека тока. В общем случае величина этого тока зависит от схемы прикосновения человека к электросети, ее напряжения, схемы самой сети и рода тока, режима нейтрали сети. Для питания отдельных устройств используются однофазные сети переменного тока 220 В / 50 Гц.

Iраб. = Upaб. / R, где

Upaб. - рабочее напряжение сети;

R - электрическое сопротивление тела человека.

Iраб. = 220В / 1000Ом = 0,22А - ток, протекающий через тело человека.

Iпор. = 0,5 мА - допустимая величина тока.

Как видно из приведенной формулы, сила тока, протекающего через тело человека, зависит только от напряжения сети и сопротивления человека. В этом случае единственной мерой, повышающей безопасность обслуживающего персонала, может быть понижение рабочего напряжения сети. Однако по техническим условиям напряжение питания устройств в машинном зале составляет 220В.

Как показывает анализ случаев электротравматизма при эксплуатации промышленных установок, чаще всего имеет место однополюсное (однофазное) прикосновение в изолированных и глухозаземленных сетях.

Единственным фактором, при данном напряжении, ограничивающим силу тока протекающего через тело человека в однофазных сетях, является сопротивление тела человека.

Опасность эксплуатации таких электроустановок очевидна. Эффективной мерой защиты при питании оборудования машинного зала напряжением опасным для жизни человека является защитное заземление. Заземляющим устройством называют совокупность заземлителя (металлического проводника или группы проводников, находящихся в непосредственном соприкосновении с грунтом) и заземляющих проводников, соединяющих заземленные части электроустановки с заземлителем. Если корпус электрооборудования не имеет контакта с землей, то в случае перехода напряжения прикосновение к нему так же опасно, как и прикосновение к токоведущим частям электроустановки. Когда корпус заземляют, образуется ветвь тока, параллельная участку сети, в которую включается человек. Ток замыкания на землю перераспределяется: вследствие малого сопротивления заземляющего устройства - больший ток пойдет через заземляющую систему, и меньший - через человека. При исправном заземлении ток, проходящий через тело человека, становится неопасным.Т. к. помещение машинного зала относится к классу помещений без повышенной опасности, то для обеспечения безопасности персонала необходим обычный комплекс профилактических работ, включающий обеспечение надежного заземления всех металлических частей, причем наибольшая допустимая величина сопротивления заземления не должна превышать 4 Ом.

Зануление является одним из средств, обеспечивающих безопасную эксплуатацию электроустановок. Оно выполняется присоединением к неоднократно заземленному нулевому проводу корпусов и других конструктивных металлических частей электрооборудования, которые нормально не находятся под напряжением, но могут оказаться под ним при повреждении изоляции.

Наибольшее допустимое сопротивление заземляющих устройств и заземлителей в системе зануления при подключении ЭВМ типа IBM PC составляет 30 Ом.

Изоляция имеет важное значение в электроустановках, она защищает их от чрезмерной утечки токов, предохраняет людей от поражения током и исключает возникновение пожаров.

Правилами устройства электроустановок определено, что сопротивление изоляции сети на участке между двумя смежными предохранителями или за последними предохранителями между любыми проводами должно быть не менее 0.5 МОм.

Проводка в производственных помещениях выполняется изолированными проводами или кабелями, которые в местах, где возможны их механические повреждения, укладываются в металлические трубы. Помещения, в которых устанавливаются персональные ЭВМ должны соответствовать всем вышеуказанным требованиям.

## 7.7 Требования по пожарной безопасности

Помещение машинного зала ПЭВМ относится к категории В (пожароопасная) пожарной опасности помещений, так как в помещении находится электрооборудование, горючие вещества (мебель, пластиковые корпуса аппаратуры и др.). Поэтому помещение должно соответствовать нормативам по огнестойкости строительных конструкций, планировке зданий, этажности, оснащенности устройствами противопожарной защиты, установленным для этой категории помещений. Помещение машинного зала должно обладать I или II степенью огнестойкости (см. СНиП 2.01.02-85 “Противопожарные нормы”), то есть самой высокой.

## 7.8 Меры по снижению уровня шума

Шум оказывает большое влияние на человека и его работоспособность. Уже при уровнях шума в 40-70 дБ ослабляется внимание, ухудшается память, появляется быстрая утомляемость, головная боль. Наибольшей степенью воздействия на состояние человека обладают импульсные и нерегулярные шумы.

Согласно ГОСТ 12.1 003-83 ССБТ “Шум. Общие требования безопасности” допустимым значением уровня шума в помещении машинного зала ПЭВМ является 50 дБ. Основными источниками шума в помещении машинного зала являются матричные принтеры, которые могут создавать уровень шума близкий к 50 дБ. Для снижения уровня шума в помещении машинного зала рекомендуется:

Располагать помещение машинного зала вдали от внешних источников шума.

Использовать звукопоглощающие облицовочные материалы.

Использовать малошумящую вентиляцию.

Использовать струйные или лазерные принтеры вместо матричных.

## 7.9 Защита от вредных излучений

Основным источником эргономических проблем, связанных с охраной здоровья людей, использующих в своей работе персональные компьютеры, являются дисплеи (мониторы), особенно дисплеи с электронно-лучевыми трубками. Они представляют собой источники наиболее вредных излучений, неблагоприятно влияющих на здоровье операторов.

История исследования этого вопроса достаточно длительная и непростая, но полученные результаты носят пока еще преимущественно статистический характер и не имеют адекватного объяснения. Частотный состав (спектр) излучения монитора характеризуется наличием рентгеновских, ультрафиолетовых, инфракрасных и других электромагнитных колебаний. Опасность рентгеновского и части других излучений большинством ученых признается пренебрежимо малой, поскольку их уровень достаточно невелик и в основном поглощается покрытием экрана. Наиболее тяжелая ситуация связана, по-видимому, с полями излучений очень низких частот (ОНЧ) и крайне низких частот (КНЧ), которые, как выяснилось, способны вызывать биологические эффекты при воздействии на живые организмы. Было обнаружено, что электромагнитные поля с частотой порядка 60 Гц могут инициировать изменения в клетках животных (вплоть до нарушения синтеза ДНК). Особенно поразительным для исследователей оказался тот факт, что, в отличие, например, от рентгеновского излучения, электромагнитные волны обладают необычным свойством: опасность их воздействия при снижении интенсивности излучения не уменьшается, мало того, некоторые поля действуют на клетки тела только при малых интенсивностях или на конкретных частотах.

Специальные измерения показали, что мониторы действительно излучают магнитные волны, по интенсивности не уступающие уровням магнитных полей, способных обусловливать возникновение опухолей у людей.

Исследователи из Macworld обнаружили, что если на расстоянии 10 см перед мониторами, обычно используемыми с компьютерами Macintosh, напряженность магнитного поля составляет примерно от 5 до 23 мГс, то на расстоянии 70 см от экрана ни у одного из обследованных мониторов напряженность поля не превышала величины 1 мГс. (Интенсивность поля вне указанных пределов составляла 0.1 - 0.5 мГс)

Как это ни странно, но до сих пор нет нормативов для излучений КНЧ-магнитных полей, хотя в некоторых странах (в том числе в Швеции и Канаде) разработаны стандарты для излучений ОНЧ-магнитных полей. Большое число поставщиков - например, фирмы IBM, DEC и Philips - продают мониторы, удовлетворяющие указанным стандартам. Кроме того, любой монитор, работающий не на ЭЛТ, имеет то преимущество, что не излучает переменных компонент, связанных с наличием систем вертикального и горизонтального отклонения электронного луча.

Пользователям персональных компьютеров, желающим снизить уровень облучения переменными магнитными полями, следует расположить мониторы так, чтобы расстояние до них составляло величину, равную расстоянию вытянутой руки (с вытянутыми пальцами). Поскольку магнитные поля сзади и по бокам большинства мониторов значительно сильнее, чем перед экраном, пользователи должны располагать свои рабочие места на расстоянии не менее 1.22 м от боковых и задних стенок других компьютеров. Следует иметь ввиду, что магнитное излучение ничем не задерживается.

## 7.10 Параметры микроклимата в машинном зале

Метеорологическими условиями согласно ГОСТ 12.1 005-88 являются:

температура;

относительная влажность;

скорость движения воздуха;

атмосферное давление.

С целью обеспечения комфортных условий для операторов ПЭВМ и надежной работы оборудования, необходимо поддерживать следующие метеорологические условия (согласно СН 512-78):

Атмосферное давление в помещении машинного зала должно быть 1013.25±266 ГПа. При пониженном давлении воздуха ухудшается отвод теплоты от элементов ПЭВМ, снижаются изоляционные свойства воздуха.

Воздух, используемый для вентиляции машинного зала, должен очищаться от пыли. Запыленность воздуха не должна превышать 1 мг/м³, а размеры пылинок - 3 мкм.

Пыль, попадающая на платы комплекса, приводит к снижению теплообмена и способствует перегреву приборов.

В помещении машинного зала необходимо предусмотреть систему отопления.

Она должна обеспечивать достаточное, постоянное и равномерное нагревание воздуха в помещении в холодный период года, а также безопасность в отношении пожара и взрыва. При этом колебания температуры в течении суток не должны превышать 2-3 °С; в горизонтальном направлении - 2 °С на каждый метр длины; а в вертикальном - 1 °С на каждый метр высоты помещения.

Для отопления помещения машинного зала рекомендуется использовать водяные или воздушные системы центрального отопления.

Работающие люди и включенные компьютеры являются источниками избыточного тепла в помещении машинного зала.

Поэтому необходимо предусмотреть систему кондиционирования помещения.

Таблица 5. Параметры воздушной среды

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Параметры воздушной среды на рабочих местах | | | | | |
| Темпера-тура | Оптимальные | | | Допустимые | | |
| Наружного воздуха, °С | Темпера-тура, °С | Относитель-ная влажность,% | Скорость движения воздуха, не более, м/c | Темпера-тура, °С | Относитель-ная влажность,% | Скорость движения воздуха, не более, м/c |
| Ниже +10 | 20-22 | 40-60 | 0.2 | 18-22 | Не более 70 | 0.3 |
| Выше +10 | 20-25 | 40-60 | 0.3 | Не более, чем на 3 °С выше наружного воздуха в 13 ч дня самого жаркого месяца, но не выше 28 °С | 70 при 24 °С  и ниже;  65 при 25 °С;  60 при 26 °С;  55 при 27 °С;  50 при 28 °С. | 0.5 |

## 7.11 Психофизиологические факторы

Психофизиологические факторы в зависимости от характера действия делятся на следующие группы:

физические перегрузки (статические);

нервно-психические перегрузки:

умственное перенапряжение,

перенапряжение зрительных анализаторов,

монотонность труда,

эмоциональные перегрузки в большом коллективе.

Монотонность или монотония - психическое состояние человека, вызванное однообразием восприятий или действий. Под утомлением понимается процесс понижения работоспособности, временный упадок сил, возникающий при выполнении определенной физической или умственной работы. Для уменьшения влияния этих факторов необходимо применять оптимальные режимы труда и отдыха в течении рабочего дня:

общее время работы за дисплеем не должно превышать 50% всего рабочего времени программиста;

при обычной работе за компьютером необходимо делать 15-минутные перерывы через каждые два часа, а при интенсивной работе - через каждый час;

не следует превышать темп работы порядка 10 тысяч нажатий клавиш в час (примерно 1500 слов);

предпочтительнее использовать дисплеи с высокой разрешающей способностью (разрешением) и удобным размером экрана (лучше не применять CGA-мониторы и малоразмерные, менее 14" по диагонали, экраны);

лучше выбирать видеоадаптеры с высоким разрешением и, по возможности (если есть на рынке и цена приемлемая), частотой обновления экранного изображения не менее 70-72 Гц;

обязательно ставить на дисплеи экранные, в частности, поляризационные, фильтры, в несколько раз снижающие утомляемость глаз;

наконец, при вводе данных с клавиатуры рекомендуется не зажимать телефонную трубку между плечом и ухом.

Рабочая поза оказывает значительное влияние на эффективность работы человека. Основные требования к рабочим местам при выполнении работы сидя приведены в ГОСТ 12.2 033-78 "ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования". При организации рабочего места программиста необходимо придерживаться следующих рекомендаций:

рабочее место должно быть оборудовано так, чтобы исключать неудобные позы и длительные статические напряжения тела;

поскольку найти такое идеальное положение для тела, в котором можно было бы пребывать в течении всего дня, вряд ли возможно, для большинства людей комфортабельным может быть рабочее место, которое можно приспособить, как минимум, для двух позиций (при этом положение оборудования должно соответствовать выполняемой работе и привычкам пользователя).

К обслуживанию и работе на ЭВМ допускаются лица прошедшие медосмотр при поступлении на работу. Последующий медосмотр проводится раз в два года. Также необходимо соблюдать ограничения на работу с персональными компьютерами для служащих, страдающих заболеваниями опорно-двигательного аппарата, глаз (или нарушениями зрения), кожи.

## 

## 7.12 Планировка рабочего места программиста и организация работы с компьютером

Планировка рабочего места программиста должна удовлетворять требованиям удобства выполнения работы и экономии энергии и времени программиста, рационального использования площади помещения, соблюдения правил охраны труда. Рекомендуемая планировка рабочего места программиста, учитывающая антропометрические данные человека и габариты аппаратуры представлена на Рис.28.

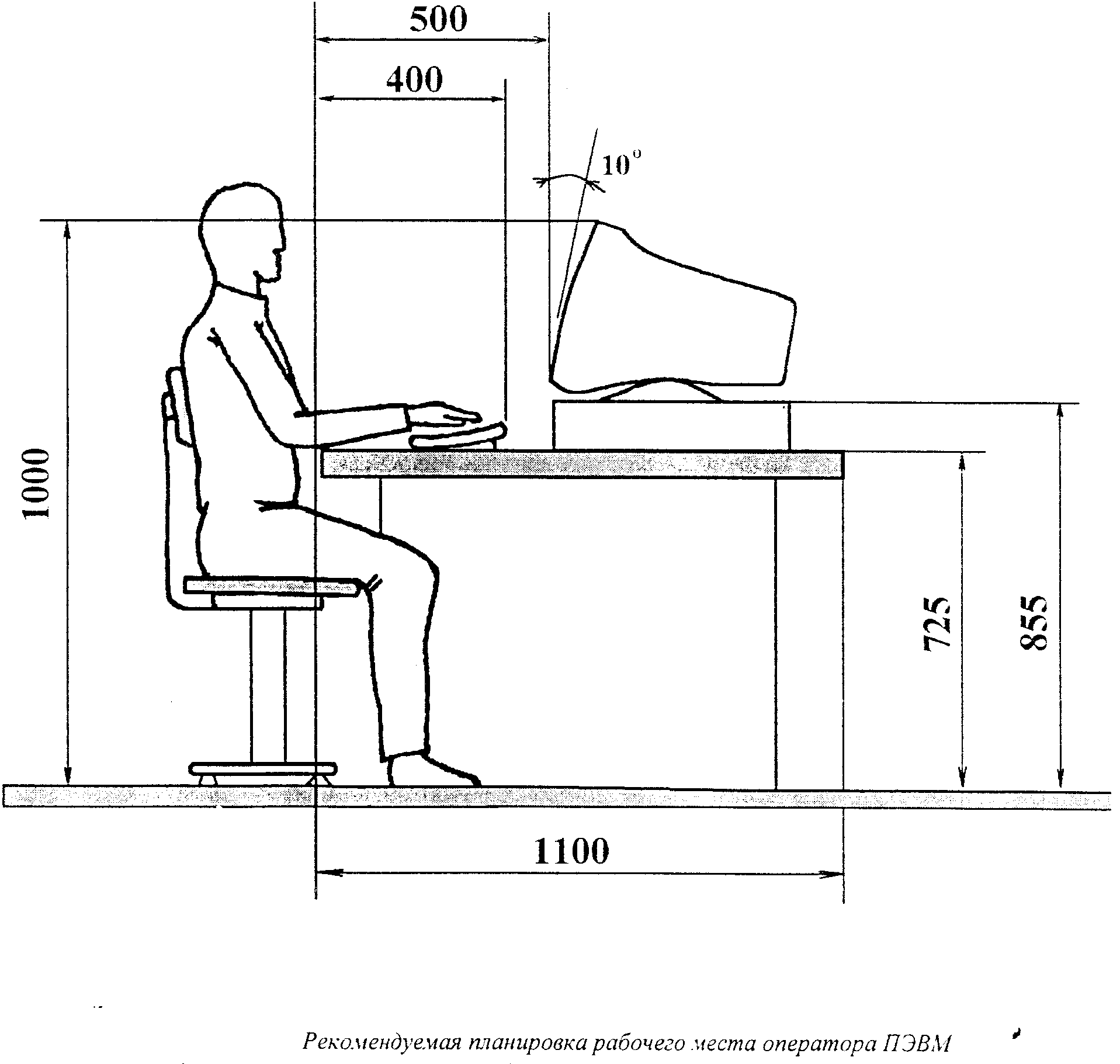


Рис.28. Рекомендуемая планировка рабочего места программиста.

Для облегчения чтения информации с документов рекомендуется использовать специальные держатели бумаг, крепящиеся к монитору компьютера. Для предотвращения перенапряжения зрительных анализаторов оператора и снижения монотонности труда, работу, связанную с использованием дисплея ПЭВМ, необходимо чередовать с работой, не требующей использования ПЭВМ, либо делать небольшие перерывы через каждые 45-90 минут. Общее время работы за экраном ЭВМ не должно превышать 6 часов в день.

## 7.13 Выводы

В этой главе дипломного проекта рассмотрены требования охраны труда и разработаны рекомендации по оптимизации санитарно-гигиенической обстановки при выполнении работ на ЭВМ.

В данной главе были рассмотрены вопросы:

Нерациональность освещения

Излучения экрана монитора

Электроопасность

Шум

Параметры микроклимата

Психофизиологические перегрузки.

Установлено, что уровень шума на рабочем месте не должен превышать 50 дб, объем производственного помещения на одного работающего должен составлять не менее 15 куб. м, площадь - не менее 4.5 кв. м, наименьшая допустимая освещенность - 200 лк.

Показана важность и необходимость организационных и санитарно-профилактических мероприятий по созданию условий безопасного труда.

В данной главе был проведен расчет искусственного освещения и подсчитано фактическое значение минимальной освещенности дисплейного класса - 200 лк.

Вопросы экологической безопасности не рассматривались, поскольку работа с ЭВМ не связана с использованием природных ресурсов, нет вредных выбросов в окружающую среду и не представляет опасности для окружающей среды.

## Заключение

Таким образом, в результате проведенных работ намечены пути построения компьютерно-сетевой системы управления производственным процессом предприятия электронной промышленности. В частности, было проанализировано текущее состояние производственного процесса, проанализированы известные методы создания аналогичных систем управления и разработаны предложения по проектированию системы управления производством партий на ОАО "Ангстрем".

Дальнейшие этапы работы должны состоять в следующем:

создание имитационной программы, моделирующей процесс управления производством партий;

разработка алгоритмов дисциплины очереди;

проектирование системы управления;

реализация, тестирование и оптимизация системы управления производством партий на ОАО "Ангстрем".

Необходимо подчеркнуть, что разработка компьютерно-сетевых систем управления производственными процессами весьма актуальна для отечественной промышленности. И возможно, разрабатываемая АСУ ПП станет базовым образцом для широкого класса "интеллектуальных" промышленных систем управления.

## Список использованной литературы

1. Поспелов Д.А. Ситуационное управление. Теория и практика. М.: Наука, 1986, 285 с.
2. Поспелов Д.А. Принципы ситуационного управления // Известия АН СССР. Серия Техническая кибернетика. 1971. N.2.
3. Клыков Ю.И. Ситуационное управление большими системами. М.: Энергия, 1974.
4. Поспелов Д.А. Большие системы. Ситуационное управление. М.: Знание, 1975.
5. Искусственный интеллект. - В 3-х кн. Справочник. М.: Радио и связь, 1990.
6. MES Explained: A High Level Vision for Executives. White Paper of MESA International (1997). Интернет-адрес: http://www.mesa.org/html/main. cgi? sub=7
7. Technical Articles of Consilium Company. Интернет-адрес: http://www.consilium.com/Publications/technica. htm
8. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика. М.: Мир, 1992.238 с.
9. С.С. Гайсарян. Объектно-ориентированные технологии проектирования прикладных программных систем. Интернет-адрес: http://www.citforum.ru/programming/oop\_rsis/index. shtml
10. P. Coad, M. Mayfield. Object Models: Strategies, Patterns and Applications (Yourdon Press Computing Series). 1996.
11. Буч.Г. "Объектно-ориентированный анализ и проектирование с примерами приложений на C++", М.,"Бином", 1998 г.

## Приложение

Фрагменты исходного кода

// Angstrem. h: main header file for the ANGSTREM application

//

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

#ifndef \_\_AFXWIN\_H\_\_

#error include 'stdafx. h' before including this file for PCH

#endif

#include "resource. h" // main symbols

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CAngstremApp:

// See Angstrem. cpp for the implementation of this class

//

class CAngstremApp: public CWinApp

{

public:

CAngstremApp ();

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CAngstremApp)

public:

virtual BOOL InitInstance ();

// }}AFX\_VIRTUAL

// Implementation

// {{AFX\_MSG (CAngstremApp)

afx\_msg void OnAppAbout ();

afx\_msg void OnObjectsPartsNew ();

afx\_msg void OnObjectsPartsOpen ();

afx\_msg void OnObjectsPartsViewall ();

afx\_msg void OnObjectsPartsDelete ();

afx\_msg void OnAppFindpart ();

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// {{AFX\_INSERT\_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

// Angstrem. cpp: Defines the class behaviors for the application.

//

#include "stdafx. h"

#include "Angstrem. h"

#include "MainFrame. h"

#include "ChildFrame. h"

#include "AngstremDoc. h"

#include "AngstremView. h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE [] = \_\_FILE\_\_;

#endif

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CAngstremApp

BEGIN\_MESSAGE\_MAP (CAngstremApp, CWinApp)

// {{AFX\_MSG\_MAP (CAngstremApp)

ON\_COMMAND (ID\_APP\_ABOUT, OnAppAbout)

ON\_COMMAND (ID\_OBJECTS\_PARTS\_NEW, OnObjectsPartsNew)

ON\_COMMAND (ID\_OBJECTS\_PARTS\_OPEN, OnObjectsPartsOpen)

ON\_COMMAND (ID\_OBJECTS\_PARTS\_VIEWALL, OnObjectsPartsViewall)

ON\_COMMAND (ID\_OBJECTS\_PARTS\_DELETE, OnObjectsPartsDelete)

ON\_COMMAND (ID\_APP\_FINDPART, OnAppFindpart)

// }}AFX\_MSG\_MAP

// Standard file based document commands

END\_MESSAGE\_MAP ()

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CAngstremApp construction

CAngstremApp:: CAngstremApp ()

{

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// The one and only CAngstremApp object

CAngstremApp theApp;

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CAngstremApp initialization

BOOL CAngstremApp:: InitInstance ()

{

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need.

#ifdef \_AFXDLL

Enable3dControls (); // Call this when using MFC in a shared DLL

#else

Enable3dControlsStatic (); // Call this when linking to MFC statically

#endif

// Change the registry key under which our settings are stored.

// TODO: You should modify this string to be something appropriate

// such as the name of your company or organization.

SetRegistryKey (\_T ("Local AppWizard-Generated Applications"));

LoadStdProfileSettings (); // Load standard INI file options (including MRU)

// Register the application's document templates. Document templates

// serve as the connection between documents, frame windows and views.

CMultiDocTemplate\* pDocTemplate;

pDocTemplate = new CMultiDocTemplate (

IDR\_ANGSTRTYPE,

RUNTIME\_CLASS (CAngstremDoc),

RUNTIME\_CLASS (CChildFrame), // custom MDI child frame

RUNTIME\_CLASS (CAngstremView));

AddDocTemplate (pDocTemplate);

// create main MDI Frame window

CMainFrame\* pMainFrame = new CMainFrame;

if (! pMainFrame->LoadFrame (IDR\_MAINFRAME))

return FALSE;

m\_pMainWnd = pMainFrame;

// Enable drag/drop open

m\_pMainWnd->DragAcceptFiles ();

// Enable DDE Execute open

EnableShellOpen ();

RegisterShellFileTypes (TRUE);

// Parse command line for standard shell commands, DDE, file open

CCommandLineInfo cmdInfo;

ParseCommandLine (cmdInfo);

// Dispatch commands specified on the command line

if (! ProcessShellCommand (cmdInfo))

return FALSE;

// The main window has been initialized, so show and update it.

pMainFrame->ShowWindow (m\_nCmdShow);

pMainFrame->UpdateWindow ();

return TRUE;

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CAboutDlg dialog used for App About

class CAboutDlg: public CDialog

{

public:

CAboutDlg ();

// Dialog Data

// {{AFX\_DATA (CAboutDlg)

enum { IDD = IDD\_ABOUTBOX };

// }}AFX\_DATA

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CAboutDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// {{AFX\_MSG (CAboutDlg)

// No message handlers

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

CAboutDlg:: CAboutDlg (): CDialog (CAboutDlg:: IDD)

{

// {{AFX\_DATA\_INIT (CAboutDlg)

// }}AFX\_DATA\_INIT

}

void CAboutDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CAboutDlg)

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CAboutDlg, CDialog)

// {{AFX\_MSG\_MAP (CAboutDlg)

// No message handlers

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

// App command to run the dialog

void CAngstremApp:: OnAppAbout ()

{

CAboutDlg aboutDlg;

aboutDlg. DoModal ();

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CNewPartDlg dialog

class CNewPartDlg: public CDialog

{

// Construction

public:

CNewPartDlg (CWnd\* pParent = NULL); // standard constructor

// Dialog Data

// {{AFX\_DATA (CNewPartDlg)

enum { IDD = IDD\_PARTNEW };

// NOTE: the ClassWizard will add data members here

// }}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CNewPartDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CNewPartDlg)

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

CNewPartDlg:: CNewPartDlg (CWnd\* pParent /\*=NULL\*/)

: CDialog (CNewPartDlg:: IDD, pParent)

{

// {{AFX\_DATA\_INIT (CNewPartDlg)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

void CNewPartDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CNewPartDlg)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CNewPartDlg, CDialog)

// {{AFX\_MSG\_MAP (CNewPartDlg)

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

void CAngstremApp:: OnObjectsPartsNew ()

{

// TODO: Add your command handler code here

CNewPartDlg newPartDlg;

newPartDlg. DoModal ();

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// COpenPartDlg dialog

class COpenPartDlg: public CDialog

{

// Construction

public:

COpenPartDlg (CWnd\* pParent = NULL); // standard constructor

// Dialog Data

// {{AFX\_DATA (COpenPartDlg)

enum { IDD = IDD\_PARTOPEN };

// NOTE: the ClassWizard will add data members here

// }}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (COpenPartDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (COpenPartDlg)

// NOTE: the ClassWizard will add member functions here

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

COpenPartDlg:: COpenPartDlg (CWnd\* pParent /\*=NULL\*/)

: CDialog (COpenPartDlg:: IDD, pParent)

{

// {{AFX\_DATA\_INIT (COpenPartDlg)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

void COpenPartDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (COpenPartDlg)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (COpenPartDlg, CDialog)

// {{AFX\_MSG\_MAP (COpenPartDlg)

// NOTE: the ClassWizard will add message map macros here

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

void CAngstremApp:: OnObjectsPartsOpen ()

{

// TODO: Add your command handler code here

COpenPartDlg openPartDlg;

openPartDlg. DoModal ();

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CViewAllPartsDlg dialog

class CViewAllPartsDlg: public CDialog

{

// Construction

public:

CViewAllPartsDlg (CWnd\* pParent = NULL); // standard constructor

// Dialog Data

// {{AFX\_DATA (CViewAllPartsDlg)

enum { IDD = IDD\_PARTSVIEWALL };

// NOTE: the ClassWizard will add data members here

// }}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CViewAllPartsDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CViewAllPartsDlg)

// NOTE: the ClassWizard will add member functions here

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

CViewAllPartsDlg:: CViewAllPartsDlg (CWnd\* pParent /\*=NULL\*/)

: CDialog (CViewAllPartsDlg:: IDD, pParent)

{

// {{AFX\_DATA\_INIT (CViewAllPartsDlg)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

void CViewAllPartsDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CViewAllPartsDlg)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CViewAllPartsDlg, CDialog)

// {{AFX\_MSG\_MAP (CViewAllPartsDlg)

// NOTE: the ClassWizard will add message map macros here

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

void CAngstremApp:: OnObjectsPartsViewall ()

{

// TODO: Add your command handler code here

CViewAllPartsDlg viewAllPartsDlg;

viewAllPartsDlg. DoModal ();

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CDeletePartDlg dialog

class CDeletePartDlg: public CDialog

{

// Construction

public:

CDeletePartDlg (CWnd\* pParent = NULL); // standard constructor

// Dialog Data

// {{AFX\_DATA (CDeletePartDlg)

enum { IDD = IDD\_PARTDELETE };

// NOTE: the ClassWizard will add data members here

// }}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CDeletePartDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CDeletePartDlg)

// NOTE: the ClassWizard will add member functions here

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

CDeletePartDlg:: CDeletePartDlg (CWnd\* pParent /\*=NULL\*/)

: CDialog (CDeletePartDlg:: IDD, pParent)

{

// {{AFX\_DATA\_INIT (CDeletePartDlg)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

void CDeletePartDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CDeletePartDlg)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CDeletePartDlg, CDialog)

// {{AFX\_MSG\_MAP (CDeletePartDlg)

// NOTE: the ClassWizard will add message map macros here

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

void CAngstremApp:: OnObjectsPartsDelete ()

{

// TODO: Add your command handler code here

CDeletePartDlg deletePartDlg;

deletePartDlg. DoModal ();

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CPartProcessingPage dialog

class CPartProcessingPage: public CPropertyPage

{

DECLARE\_DYNCREATE (CPartProcessingPage)

// Construction

public:

CPartProcessingPage ();

~CPartProcessingPage ();

// Dialog Data

// {{AFX\_DATA (CPartProcessingPage)

enum { IDD = IDD\_PART\_PROCESSING };

// NOTE - ClassWizard will add data members here.

// DO NOT EDIT what you see in these blocks of generated code!

// }}AFX\_DATA

// Overrides

// ClassWizard generate virtual function overrides

// {{AFX\_VIRTUAL (CPartProcessingPage)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CPartProcessingPage)

afx\_msg void OnRefresh ();

afx\_msg void OnStart ();

afx\_msg void OnFinish ();

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

IMPLEMENT\_DYNCREATE (CPartProcessingPage, CPropertyPage)

CPartProcessingPage:: CPartProcessingPage (): CPropertyPage (CPartProcessingPage:: IDD)

{

// {{AFX\_DATA\_INIT (CPartProcessingPage)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

CPartProcessingPage:: ~CPartProcessingPage ()

{

}

void CPartProcessingPage:: DoDataExchange (CDataExchange\* pDX)

{

CPropertyPage:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CPartProcessingPage)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CPartProcessingPage, CPropertyPage)

// {{AFX\_MSG\_MAP (CPartProcessingPage)

ON\_BN\_CLICKED (IDREFRESH, OnRefresh)

ON\_BN\_CLICKED (IDC\_START, OnStart)

ON\_BN\_CLICKED (IDC\_FINISH, OnFinish)

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CPartProcessingsPage dialog

class CPartProcessingsPage: public CPropertyPage

{

DECLARE\_DYNCREATE (CPartProcessingsPage)

// Construction

public:

CPartProcessingsPage ();

~CPartProcessingsPage ();

// Dialog Data

// {{AFX\_DATA (CPartProcessingsPage)

enum { IDD = IDD\_PART\_PROCESSINGS };

// NOTE - ClassWizard will add data members here.

// DO NOT EDIT what you see in these blocks of generated code!

// }}AFX\_DATA

// Overrides

// ClassWizard generate virtual function overrides

// {{AFX\_VIRTUAL (CPartProcessingsPage)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CPartProcessingsPage)

// NOTE: the ClassWizard will add member functions here

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

IMPLEMENT\_DYNCREATE (CPartProcessingsPage, CPropertyPage)

CPartProcessingsPage:: CPartProcessingsPage (): CPropertyPage (CPartProcessingsPage:: IDD)

{

// {{AFX\_DATA\_INIT (CPartProcessingsPage)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

CPartProcessingsPage:: ~CPartProcessingsPage ()

{

}

void CPartProcessingsPage:: DoDataExchange (CDataExchange\* pDX)

{

CPropertyPage:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CPartProcessingsPage)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CPartProcessingsPage, CPropertyPage)

// {{AFX\_MSG\_MAP (CPartProcessingsPage)

// NOTE: the ClassWizard will add message map macros here

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CPartRoutePage dialog

class CPartRoutePage: public CPropertyPage

{

DECLARE\_DYNCREATE (CPartRoutePage)

// Construction

public:

CPartRoutePage ();

~CPartRoutePage ();

// Dialog Data

// {{AFX\_DATA (CPartRoutePage)

enum { IDD = IDD\_PART\_ROUTE };

// NOTE - ClassWizard will add data members here.

// DO NOT EDIT what you see in these blocks of generated code!

// }}AFX\_DATA

// Overrides

// ClassWizard generate virtual function overrides

// {{AFX\_VIRTUAL (CPartRoutePage)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CPartRoutePage)

// NOTE: the ClassWizard will add member functions here

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

IMPLEMENT\_DYNCREATE (CPartRoutePage, CPropertyPage)

CPartRoutePage:: CPartRoutePage (): CPropertyPage (CPartRoutePage:: IDD)

{

// {{AFX\_DATA\_INIT (CPartRoutePage)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

CPartRoutePage:: ~CPartRoutePage ()

{

}

void CPartRoutePage:: DoDataExchange (CDataExchange\* pDX)

{

CPropertyPage:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CPartRoutePage)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CPartRoutePage, CPropertyPage)

// {{AFX\_MSG\_MAP (CPartRoutePage)

// NOTE: the ClassWizard will add message map macros here

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /

// CFindPartDlg dialog

class CFindPartDlg: public CDialog

{

// Construction

public:

CFindPartDlg (CWnd\* pParent = NULL); // standard constructor

// Dialog Data

// {{AFX\_DATA (CFindPartDlg)

enum { IDD = IDD\_FINDPART };

// NOTE: the ClassWizard will add data members here

// }}AFX\_DATA

// Overrides

// ClassWizard generated virtual function overrides

// {{AFX\_VIRTUAL (CFindPartDlg)

protected:

virtual void DoDataExchange (CDataExchange\* pDX); // DDX/DDV support

// }}AFX\_VIRTUAL

// Implementation

protected:

// Generated message map functions

// {{AFX\_MSG (CFindPartDlg)

virtual void OnOK ();

// }}AFX\_MSG

DECLARE\_MESSAGE\_MAP ()

};

CFindPartDlg:: CFindPartDlg (CWnd\* pParent /\*=NULL\*/)

: CDialog (CFindPartDlg:: IDD, pParent)

{

// {{AFX\_DATA\_INIT (CFindPartDlg)

// NOTE: the ClassWizard will add member initialization here

// }}AFX\_DATA\_INIT

}

void CFindPartDlg:: DoDataExchange (CDataExchange\* pDX)

{

CDialog:: DoDataExchange (pDX);

// {{AFX\_DATA\_MAP (CFindPartDlg)

// NOTE: the ClassWizard will add DDX and DDV calls here

// }}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP (CFindPartDlg, CDialog)

// {{AFX\_MSG\_MAP (CFindPartDlg)

// }}AFX\_MSG\_MAP

END\_MESSAGE\_MAP ()

void CFindPartDlg:: OnOK ()

{

// TODO: Add extra validation here

// считывание номера партии и изделия в

// видимые для PartSheet переменные

CDialog:: OnOK ();

}

void CAngstremApp:: OnAppFindpart ()

{

// TODO: Add your command handler code here

INT partNumber;

INT productNumber;

CFindPartDlg findPartDlg;

if (findPartDlg. DoModal () == IDOK)

{

// Создание объекта блока диалога с вкладками

CPropertySheet partSheet ("Part");

// Создание объекта для каждой вкладки

CPartProcessingPage partProcessingPage;

CPartProcessingsPage partProcessingsPage;

CPartRoutePage partRoutePage;

// Добавление вкладок в блок диалога

partSheet. AddPage (&partProcessingPage);

partSheet. AddPage (&partProcessingsPage);

partSheet. AddPage (&partRoutePage);

partSheet. DoModal ();

}

}

// // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // // /