Реферат

Отчет 121 с., 6 рис., 12 табл., 15 источников, 1 прил.

ДИСПЕТЧИРОВАНИЕ, ЖЕЛЕЗНОДОРОЖНЫЕ ПЕРЕВОЗКИ, РЕСУРС-ДЕЙСТВИЕ-ОПЕРАЦИЯ, ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, КОРРЕКТИРОВКА, МОДУЛЬ

Объектом разработки является подсистема диспетчирования перевозок серы железнодорожным транспортом на основе РДО.

Ресурс, действие, операция (РДО) – программный комплекс, предназначенный для имитационного моделирования сложных дискретных систем с целью проведения их анализа и синтеза.

Цель работы – создание на основе РДО подсистемы диспетчирования перевозок серы железнодорожным транспортом.

При создании подсистемы было проведено концептуальное исследование, в результате которого было выявлено место подсистемы диспетчирования в составе АСУ, предложена ее принципиальная архитектура, составлено техническое задание на модули подсистемы.

В результате дальнейшей работы был разработан действующий прототип подсистемы, апробированный на тестовом примере.

Полученные результаты демонстрирует возможность создания полнофункциональной подсистемы по разработанному техническому заданию на основе РДО.

Содержание

Перечень сокращений, символов и специальных терминов с их определением

ARIS	*Architecture of Integrated Information Systems* (методология и программный продукт для моделирования бизнес-процессов компании)
MFC	*Microsoft Foundation Classes* (Библиотека фирмы Microsoft, предназначенная для разработки приложений Microsoft Windows)
ODBC	*Open Data Base Connectivity* (программный интерфейс доступа к БД, разработанный фирмой Microsoft
OLE DB	*Object Linking and Embedding, Database* (набор интерфейсов, которые позволяют приложениям обращаться к данным, хранимым в разных источниках информации с помощью унифицированного доступа)
SQL	*Structured Query Language* (структурированный язык запросов)
UML	*Universal Modeling Language* (универсальный язык моделирования)
АС	Автоматизированная система
АСУ ПС	Автоматизированная система управления перевозками серы
БД (БЗ)	База данных (База знаний)
ВП	Вагонный парк
ГОВ	График отправки вагонов
ОАО ГТП	ОАО «Газпромтранс»

ПП	Программный продукт
ПЭВМ	Персональная электронно-вычислительная машина
РДО	Ресурсы-действия-операции – система имитационного моделирования
РЖД	Российские железные дороги
ТЗ	Техническое задание

Введение

Данный дипломный проект посвящен разработке подсистемы диспетчирования перевозок серы железнодорожным транспортом для ОАО ГПТ. Рассматриваемая подсистема входит в состав АСУ ПС. Техническое задание на АСУ ПС было разработано специалистами ОАО ГПТ при участии специалистов кафедры «Компьютерные системы автоматизации производства» МГТУ им. Баумана. Основным инструментом для разработки является среда имитационного моделирования РДО.

Таким образом, тема дипломной работы содержит два понятия из мира автоматизации производства: диспетчирование и имитационное моделирование.

Под *процессом диспетчирования* мы понимаем процесс непрерывного контроля и оперативного регулирования хода производства с использованием определенных технических средств. В традиционной концепции многоконтурного управления предприятием (OLAP-ERP-MES-SCADA) системы диспетчирования (Dispatching Production Units Systems) входят в состав производственных исполнительных систем (MES). Системы диспетчирования обычно используются совместно с системами оперативного планирования (Operations Scheduling Systems) и обеспечивают точность выполнения запланированных процессов.

В отличие от традиционной системы диспетчирования, выполняющей сравнение «факт-план», в данной работе представлена подсистема диспетчирования, использующая технологию имитационного моделирования, что позволяет прогнозировать выполнение перевозок с текущего момента времени до конца планового периода. Таким образом, диспетчер получает возможность принимать решения, опираясь на результаты имитационного эксперимента.

Предпроектное исследование

Описание деятельности ОАО ГПТ

Основными видами деятельности компании ОАО ГПТ являются:

- Предоставление собственного железнодорожного состава
- Подача-уборка вагонов собственным локомотивным парком
- Транспортно-экспедиторское облуживание

ОАО ГПТ, являясь грузоотправителем и владельцем подвижного состава, обеспечивает значительные объемы перевозок серы, поставляемой на химические заводы России и на экспорт. Компания ОАО ГПТ осуществляет доставку жидкой и гранулированной серы с предприятий ОАО «Газпром» потребителям собственными вагонами и арендованными.

Пунктов производства серы в настоящее время два (Каргала и Аксарайская-2) и планируется увеличение производительности одного из них. Объемы производства серы достаточны для обеспечения потребителей серой с любого завода-производителя.

Имеются ограничения на скорости погрузки как гранулированной, так и жидкой серы. Погрузка жидкой серы производится на специальных эстакадах заводов-производителей. Определенное количество вагонов (36 вагонов с двух сторон эстакады) одновременно загружается в течение нормативного времени. Погрузка гранулированной серы осуществляется специальными устройствами и имеет следующую производительность:

- на станции Каргала могут загружать 20 вагонов в сутки твердой серы и 90 вагонов жидкой;
- на станции Аксарайская-2 могут загружать 140 вагонов твердой и 120 вагонов жидкой серы в сутки.

Погруженные вагоны в соответствии с заявкой (форма 12) подаются на станцию РЖД и с этого момента времени находятся под управлением служб РЖД. По истечении нормативного времени доставки, известного для каждого пункта назначения, вагоны поступают под разгрузку на станции потребителя.

Разгрузка также производится на эстакадах, количество мест одновременной разгрузки вагонов для каждого пункта назначения известно, известны также и нормативные времена разгрузки в этих пунктах. В процессе работы по доставке серы потребителям используются как «свои», так и «чужие» вагоны, т.е. вагоны, взятые в аренду.

В каждом пункте погрузки серы имеются подъездные пути и пути, на которых могут стоять как груженые составы в ожидании отправки на станцию РЖД, так и пустые вагоны в ожидании погрузки серы и резервный парк вагонов. Объемы подъездных путей ограничены, и одновременно на них может стоять ограниченное количество порожних вагонов, ожидающих погрузку, избыточных вагонов, груженых вагонов, ожидающих отправку на станцию РЖД, а также могут располагаться вагоны с другими типами грузов, которые перевозятся ГПТ.

В настоящее время имеется N пунктов назначения, в которые поставляется сера. В каждом пункте назначения формируется желательный график приема серы, включающий количество жидкой и гранулированной серы и желательные сроки ее поставки. График приема формируется с учетом производственных потребностей и возможностей пункта потребления. Желательный график согласуется с представителями ГПТ за определенный срок (до 10 дней) до начала месяца и после согласования с возможностью некоторых изменений принимается к исполнению. Однако могут подаваться «корректировки» и за 3 дня до отправки груза, но с более высокой стоимостью.

В пункты доставки требуется перевезти 250-300 тыс. тонн гранулированной серы и 40-50 тыс. тонн жидкой серы. Имеется два типа вагонов. Полувагоны используются для перевозки гранулированной серы и цистерны, которые используются для перевозки жидкой серы. Вместимость вагонов и цистерн составляет 66-68 тонн. Всего имеется X вагонов и Y цистерн.

Разгрузка поступивших вагонов производится сотрудниками организаций потребителей, и у ОАО ГПТ нет реальных механизмов влияния на длительность разгрузки и время отправки назад к местам производства серы уже разгруженных вагонов. Однако за простой вагонов больше нормативного срока ОАО ГПТ может выставить потребителям штрафы. Поэтому задерживать разгруженные вагоны нет смысла. Пустые вагоны к местам производства серы посылаются обычно группами, а не «россыпью», поскольку для групп имеются скидки, зависящие от размера группы.

Еще более выгодно посылать вагоны как нагруженные, так и пустые отдельными поездами, так называемыми прямыми маршрутами. Обычно такие поезда состоят из 44 вагонов для доставки гранулированной серы и из 60-66 вагонов для доставки жидкой серы. Общая грузоподъемность поезда по требованиям РЖД не должна превышать 3000 тонн. На такие поезда даются значительные скидки до 10%, а их среднесуточная скорость составляет 500 км/сутки, в то время как среднесуточная скорость движения вагонов, посылаемых заказчику не в отдельном маршруте («россыпью») составляет 330 км/сутки.

Однако имеется возможность ускорить движение, как отдельных маршрутов (поездов), так и отдельных вагонов, посылаемых «россыпью». Их скорость может быть увеличена вдвое, но при этом стоимость доставки грузов также возрастает в два раза. С каждого завода в сутки может отправляться до 2-3 поездов с гранулированной серой и через сутки поезд с жидкой серой. Статистика по задержкам поставок, задержек по погрузкам и отгрузкам серы и отклонениям от плана пока не собирается.

Вагоны после прохождения определенного количества километров отправляются на профилактический ремонт. Профилактические осмотры вагонов обычно проводятся на специальных станциях. Профилактические ремонты проводятся на станциях, где есть депо и с которыми ОАО ГПТ заключил соответствующие договора. Такие станции считаются известными.

Маршруты доставки (т. е. перечень станций движения) вагонов в основном определяются службами РЖД. Однако при доставке грузов за границу в некоторых случаях можно задавать пункты пересечения границы.

Достаточно часто уже в процессе выполнения перевозок возникают ситуации, когда заказчики требуют увеличить количество перевозок груза, т.е. требуется организовать дополнительные перевозки грузов. Для этого требуется оценить текущие возможности, получаемую выгоду и скорректировать расписание отправки поездов с грузами.

Если вагоны с серой были поставлены с завода на станцию РЖД раньше, чем нормативный срок доставки вагонов потребителям, то ОАО ГПТ выставляет штрафной счет РЖД и, фактически, не несет убытков. Если же вагоны доставлены позже, чем нормативный срок доставки вагонов потребителю, то происходит выяснение причин задержек. Если завод-производитель был готов производить погрузку, а вагонов не было, то ответственность ложится на ОАО ГПТ. Если же вагоны под погрузку были готовы, а продукция в этот момент была не готова, то ответственность ложится на завод-производитель. Если же вагоны с серой были доставлены на станцию РЖД вовремя, а задержка произошла в процессе доставки, то это является виной РЖД со всеми вытекающими последствиями.

Однако задержки даже по вине РЖД увеличивают период оборачиваемости вагонов. Это по возможности надо также учитывать. Все вагоны, которые должны отправляться потребителям, до погрузки должны быть оплачены, т. е. до погрузки должна быть заплачена стоимость поставляемой серы и стоимость ее доставки. В случае, когда должна производиться погрузка поезда, а некоторые вагоны не оплачены, приостанавливается погрузка до полной оплаты груза и ответственности за это и последующее опоздание ОАО ГПТ не несет. В случае погрузки вагонов, доставляемых «россыпью» некоторые вагоны могут грузиться и отправляться без предоплаты, по договоренности.

Краткое описание АСУ ПС и требования к системе в целом

Объектом автоматизации АСУ ПС является процесс составления декадных планов перевозок и диспетчеризации перевозок серы от предприятий ОАО Газпром потребителям по их заявкам средствами ОАО ГПТ и РЖД.

АСУ должна создаваться на основе открытой архитектуры, с учетом следующих требований:

• возможность эволюции аппаратно-программных средств без перепрограммирования существующих приложений;

• возможность эволюции аппаратно-программных средств без перепрограммирования существующих приложений;

• коллективное использование информационных ресурсов;

• единая интегрированная среда для всех пользователей системы;

• единое централизованное управление, администрирование и техническое обслуживание информационно-коммуникационных ресурсов сети;

• преимущественное использование готовых стандартных программно-технических компонент в противовес разработке специализированных решений;

• использование стандартных (промышленных) интерфейсов взаимодействия между этими компонентами;

• обеспечение независимости предоставляемых пользователям возможностей от их географического расположения.

Проведение работ по созданию «АСУ ПС» преследует следующие основные цели: рисунок 1. Предлагаемый состав АСУ ПС с точки зрения функциональной структуры представлен на рисунке 2.

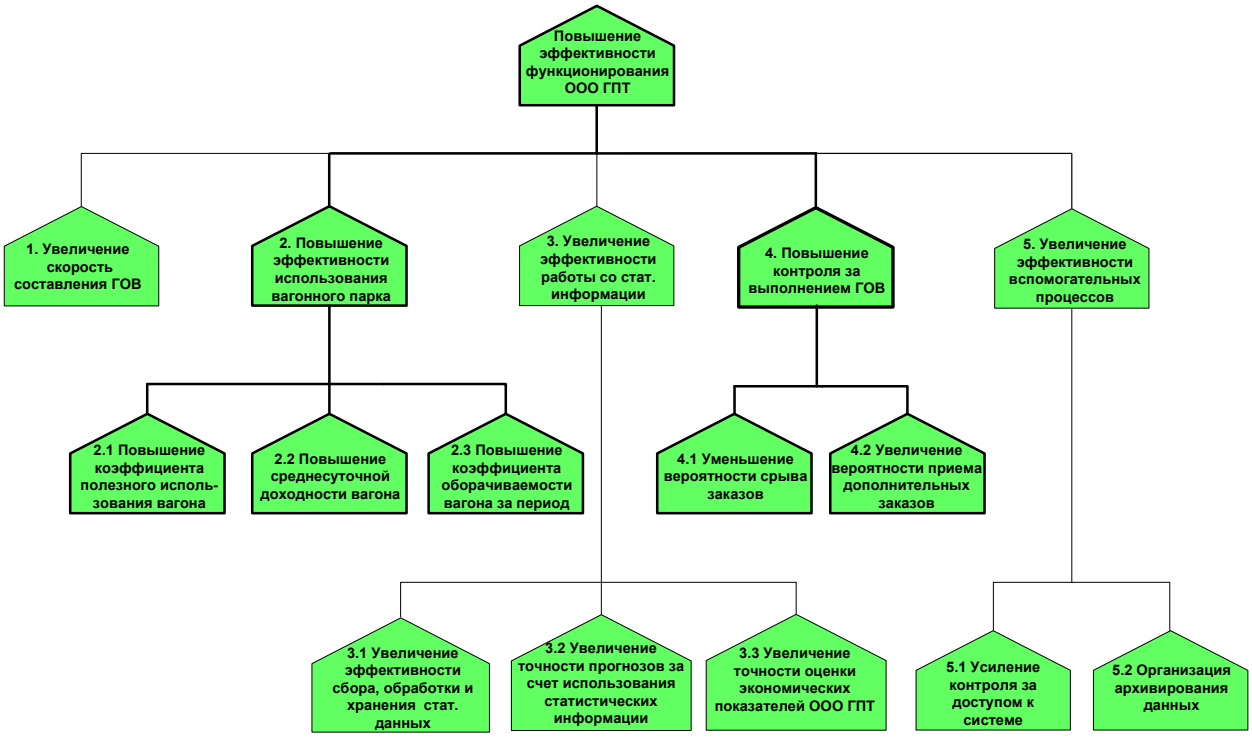

Рис. 1. Цели создания АСУ ПС

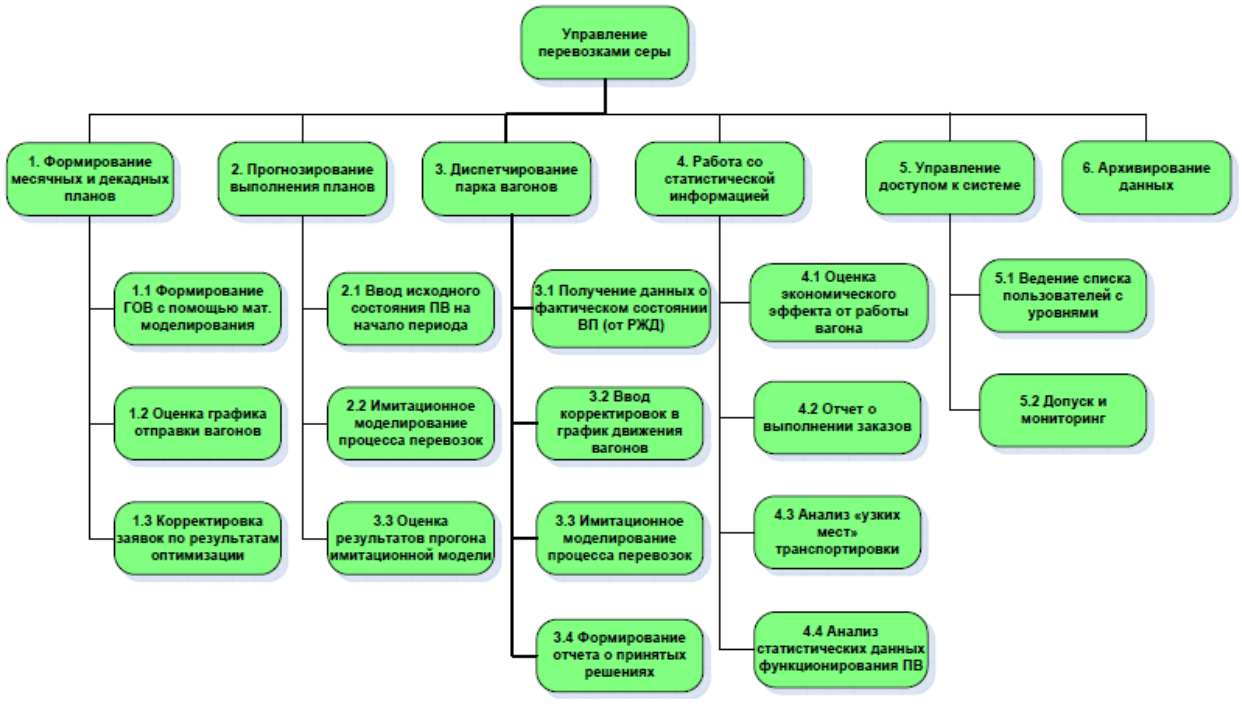

Рис. 2. Функциональный состав АСУ ПС

Представленные на рисунке 2 функции реализуются в соответствии с ТЗ на АСУ ТП следующими подсистемами:

1. Подсистема формирования декадных и месячных планов
2. Подсистема прогноза выполнения ГОВ
3. Подсистема диспетчирования ПВ
4. Подсистема интеграции с АС «Транспортировка»

5. Подсистема анализа и статистики
6. Подсистема управления доступом
7. Подсистема архивирования

Ниже приведем краткое описание данных подсистем, за исключением подсистемы диспетчирования, месту которой в составе АСУ ПС посвящен следующий раздел.

Функция «Формирование месячных и декадных планов» (рис.2) реализуется подсистемой формирования декадных и месячных планов. Данная подсистема служит для достижения целей 1,2 (рис. 1). В задаче формирования месячных и декадных планов требуется определить план отправки вагонов в соответствии с заказами потребителей в необходимом количестве в заданные сроки и при этом повысить:

• оборачиваемость вагонов

• доход от использования каждого вагона

и сократить:

• использование «чужих» вагонов

• требуемый парк вагонов для обеспечения всех необходимых перевозок

• запаздывания от утвержденного графика доставки серы потребителям

Для решения задачи составления оптимального расписания предполагается использовать методы математического программирования. В качестве численных критериев оптимизации авторами ТЗ предлагается использовать следующие:

- Среднесуточная доходность вагона:

ДС / (Nгодн * Т),

где ДС – денежные средства;

ДС = Σ (Vгруза * Стар) - Σ (Ш1 + Ш2 – Ш3) , где:

Vгруза – количество перевезенного груза,

Стар - тариф перевозки с учетом комиссионных,

Ш1 – штраф, выплачиваемый ГПТ РЖД за несвоевременную подачу вагонов;

Ш2 - штраф, выплачиваемый за невыполнение заявок на перевозку;

Ш3 – штраф, выплачиваемый грузополучателем ГПТ за несвоевременную выгрузку (простой вагонов сверх норматива) вагонов.

Nгодных = Nинвен – Nнеисп, где:

Nгодных – количество вагонов (собственных и арендованных) которые могут быть использованы для перевозки грузов в период планирования;

Nинвен – инвентарное количество вагонов,

Nнеисп – количество неисправных вагонов (в ремонте и сломанных).

Вариант:

ДС = Σ(ДСполуч - ДСржд), где:

ДСполуч – денежные средства, полученные от грузополучателя за перевозку грузов и штрафы;

ДСржд - денежные средства, затраченные ГПТ за перевозку грузов и штрафы РЖД.

Вспомогательные показатели оценки функционирования вагонного парка:

- Показатель (коэффициент) оборачиваемости вагона за период планирования:

Коб = (Vгруза) / (Sваг * Nгодных), где:

Sваг – стат. нагрузка - средний вес, перевозимый в вагоне.

- Коэффициент полезного использования:

Кисп. = Σ (Li * Ni * Xi)/ Σ (Li * Ni), где:

Li – длина перевозки i-ой группы вагонов;

Ni - количество вагонов i-ой группы;

Xi – загруженность (Xi = 1, если перевозился груз, Xi = 0, если перевозились пустые вагоны).

Функция «Прогноз выполнения ГОВ» (рис.2) реализуется подсистемой прогноза выполнения ГОВ. Данная система предназначена для достижения целей 2,4 (рис.1). Результатом прогноза является состояние парка вагонов и выполнение графика перевозок через определенное время, вычисляемое с использованием нормативных или статистических данных.

Функция «Обработка и анализ статистики» реализуется подсистемой обработки и анализа статистики. Данная подсистема предназначена для достижения цели 3 (рис.1). В настоящее время статистика по перевозкам не собирается.

Функция «Управление доступом» реализуется подсистемой управления доступом, которая предназначена для защиты от несанкционированного доступа и разграничения полномочий.

Функция «Архивирование» реализуется подсистемой архивирования, предназначенной для хранения данных о функционировании системы за определенный период.

Подсистема интеграции с АС «Транспортировка» предназначена для объединение АСУ ТП и существующих на предприятии средств автоматизации в единое информационное пространство.

Место подсистемы диспетчирования в АСУ ПС.

Подсистема диспетчирования реализует функцию «Диспетчирование перевозок» (рис. 2) и служит для достижения целей 4.1, 4.2 и косвенно целей 2.1, 2.2, 2.3 (рис. 1)

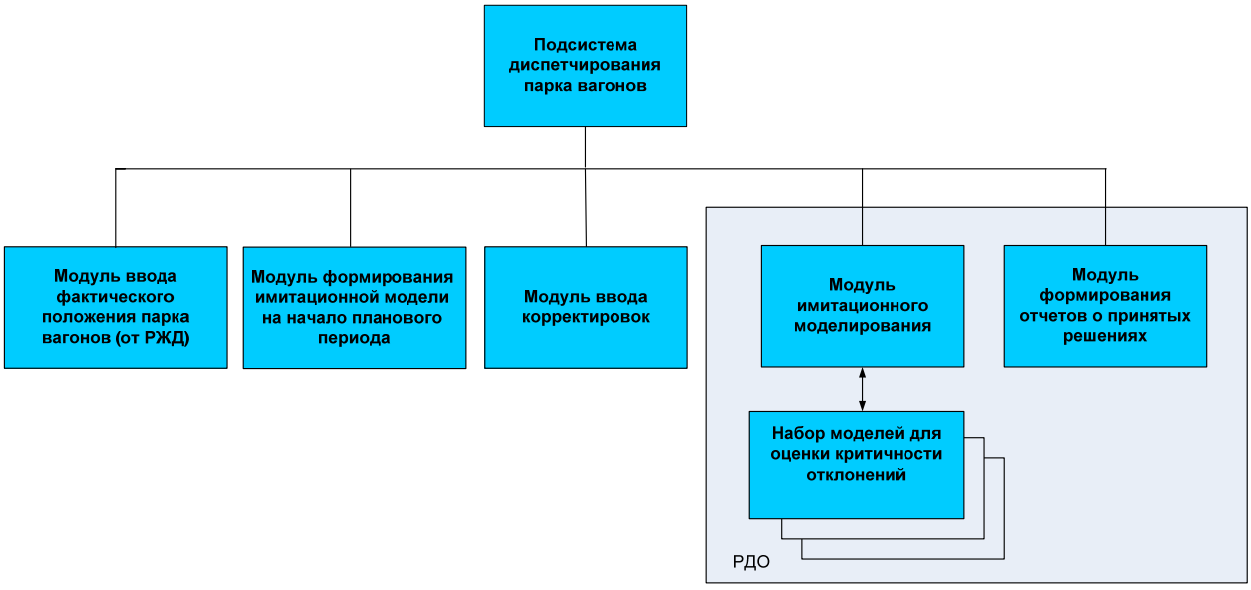

Рис. 3 Состав подсистемы диспетчирования

Наиболее логичным составом подсистемы диспетчирования с точки зрения функциональной структуры представляется модульный состав, представленный на рисунке 3.

Модуль ввода фактического положения парка вагонов предназначен для ввода в подсистему данных от РЖД о положении вагонного парка на определенную дату. Данные от РЖД предоставляются в виде базы данных. Данный модуль предлагается реализовать в виде приложения Windows, реализующего доступ к базам данных при помощи одной из стандартных технологий.

Модуль формирования имитационной модели на начало планового периода предназначен для автоматизированного процесса формирования файлов описания ресурсов и параметров модели по выходным данным подсистемы формирования расписания на начало планового периода и подсистемы анализа и статистики.

Модуль ввода корректировок предназначен для ввода пользовательских корректировок.

Оставшиеся модули подсистемы представляют собой реализацию имитационной модели в среде РДО, дополненную средствами формирования отчетов.

Отметим, что модуль имитационного моделирования помимо подсистемы диспетчирования входит также в подсистему прогноза выполнения ГОВ. Имитационные модели, входящие в данные модули описывают одну и ту же предметную область.

Реализация подсистемы диспетчирования на языке ИМ РДО.

При моделировании выполнения графика перевозок серы в ГПТ предполагается использование *среды имитационного моделирования РДО.* (Ресурс-Действие-Операция) разработанной в Московском государственном техническом университете (МГТУ им. Н.Э.Баумана) на кафедре "Компьютерные системы автоматизации производства". Модель РДО

основана на методе «сканирования активностей». Для описания функционирования в РДО – методе используются модифицированные продукционные правила, которые позволяют описывать знания, которые использует человек при принятии решений.

В среде РДО моделирование ведется на основе ресурсов и действий над ними.

Ресурсы представляют собой объекты реального мира, над которыми совершаются действия.

Операция описывает:

- правила выполнения действий данного типа (operation)
- список релевантных ресурсов (то есть ресурсов, участвующих в описываемой операции), предусловия начала операции (задаваемые по значениям параметров релевантных ресурсов)
- конверторы преобразования состояния ресурсов, участвующих в операции, в ее начале и в конце
- выражения, по которым вычисляются длительности выполнения действий, описываемых данной операцией.

Описание типов ресурсов определяет структуру глобальной базы данных программы (модели). Операции определяют содержимое базы знаний программы (модели). Они представляют собой знания о функционировании моделируемой системы (знания о предметной области), записанные в виде модифицированных продукционных правил в соответствии с синтаксисом языка.

Результаты имитационного моделирования определяются на произвольное время и представляются на упрощенной карте РЖД в виде местонахождения групп вагонов с указанием их состояния и отклонения от утвержденного ГОВ.

Использование технологии ODBC

Для разработки модуля ввода фактического положения парка вагонов необходимо использовать одну из возможных технологий работы с базами данных в ОС Windows. Это необходимо для получения доступа к данным, предоставляемым РЖД. При написании приложения баз данных с применением MFC поддерживается два принципиальных подхода:

OLE DB предоставляет способ доступа к локальным и удаленным БД с применением технологии ActiveX, обеспечивающей эффективный способ доступа к БД, без дополнительных расходов, накладываемых MFC.

ODBC (Open Data Base Connectivity) предоставляет стандартный, ориентированный на функции, системно-независимый интерфейс доступа к данным, поддерживаемый множеством поставщиков продуктов БД.

В данной работе будем использовать интерфейс ODBC. Данный интерфейс определяет набор вызовов функций для операции с БД, которые являются системно-нейтральными. Также данный подход обладает большей универсальностью по отношению к типам БД. Однако, использование данного подхода менее эффективно в плане быстродействия приложения.

Выводы по результатам предпроектного исследования.

В ходе предпроектного исследования мы определили место разрабатываемой подсистемы в составе АСУ ПС, определили основные технологии создания программного обеспечения. Теперь перейдем к этапу концептуального проектирования, на котором мы установим взаимосвязи между модулями разрабатываемой подсистемы и выработаем требования к ним, для написания технического задания.

Концептуальное проектирование

Цели разработки подсистемы диспетчирования

Возвращаясь к ранее рассмотренной ARIS-диаграмме целей создания АСУ ПС (рис. 1) определим цели создания подсистемы диспетчирования.

Основной целью является повышение контроля выполнения графика отправки вагонов. Подцелями являются:

- Уменьшение вероятности срыва сроков доставки
- Увеличение вероятности приема дополнительных заказов

Также отметим, что использование подсистемы диспетчирования приводит к повышению эффективности использования вагонного парка (рис.1), поскольку увеличиваются:

- Коэффициент полезного использования вагона
- Среднесуточная доходность вагона
- Коэффициент оборачиваемости вагона

Разработка функциональной IDEF-0 модели системы

На листе 2 представлена функциональная модель подсистемы диспетчирования в соответствии с нотацией IDEF-0.

Цель создания модели: Детализация функций, процессов и состава разрабатываемой подсистемы.

Точка зрения: Разработчик подсистемы.

На верхнем уровне функциональная модель подсистемы диспетчирования представлена одним функциональным блоком: «Диспетчирование перевозок серы».

На *вход* поступают данные о фактическом местоположении вагонов от РЖД и корректировки, предлагаемые диспетчером. В качестве *управления* выступают требования к разрабатываемой системе и плановое расписание движения вагонов. *Механизмом* является разработанное программное и аппаратное обеспечение. На *выходе* подсистема диспетчирования формирует

отчет о результатах моделирования с учетом корректировок, предложенных диспетчером.

При декомпозиции функционального блока «Диспетчирование перевозок» выделим следующие функциональные блоки:

- Ввод данных о фактическом положении групп вагонов (данные РЖД)
- Формирование исходных данных для моделирования
- Имитационное моделирование процесса перевозок
- Формирование отчета о результатах моделирования

Будем рассматривать данный уровень декомпозиции совместно с информационной моделью ввода корректировок в имитационную модель (нотация IDEF1x), изображенной на листе 3.

1. Ввод данных о фактическом положении групп вагонов

Данный функциональный блок описывает процесс «выгрузки» данных о фактическом положении групп вагонов из базы данных РЖД в файл, который в дальнейшем используется подсистемой диспетчирования для построения имитационной модели.

Входными данными поля базы данных РЖД, содержащие данные о фактическом положении групп вагонов.

Фактическое положение групп вагонов представлено сущностью ФАКТ_ПОЛОЖЕНИЯ_ГРУППЫ_ВАГОНОВ. Ключевым атрибутом является Номер_группы_вагонов. Дислокация группы задается пунктом отправления вагонов, пунктом назначения вагонов и положением группы по факту, принимающим значение от нуля до единицы. К примеру, если положение по факту равно 0.5, то группа вагонов находится посередине между пунктом отправления и пунктом назначения.

В качестве *управления* выступают требования к подсистеме диспетчирования в целом и к модулю ввода данных о фактическом положении парка вагонов.

Механизмом являются модуль ввода фактического положения групп вагонов.

Выходными данными является файл фактического положения групп вагонов. Данный файл может быть представлен той же информационной сущностью, что и входные данные. Они поступают на вход следующего функционального блока – формирование исходных данных для моделирования.

2. Формирование исходных данных для моделирования.

Данный функциональный блок описывает формирование имитационной модели по составленному на начало периода графику отправки вагонов, учитывающей фактическое положение групп вагонов по данным РЖД и корректировки, предлагаемые диспетчером. Декомпозиция данного функционального блока представлена на листе 3.

2.1Формирование исходного файла ресурсов модели.

Данный функциональный блок описывает формирование имитационной модели, ресурсы которых (вагоны, заявки и т.д.) находятся в состоянии, соответствующем началу планового периода.

Входными данными являются статистические данные, полученные от подсистемы анализа и статистики.

В качестве *управления* выступают:

- Расписание движения вагонов на начало планового периода
- Требования диспетчера к типу формируемой модели (нормативная, среднестатистическая, случайная)

Механизмом является модуль формирования исходной модели на начало планового периода

Выходными данными являются файлы ресурсов и параметров модели, соответствующих началу планового периода. Выходные данные представлены сущностью ИМИТАЦИОННАЯ_МОДЕЛЬ_НА_НАЧАЛО_ПЛАНОВОГО_ПЕРИОДА

2.2Учет фактического положения групп вагонов

Данный функциональный блок описывает процесс внесения корректировок в модель, учитывающих фактическое положение групп вагонов по данным РЖД.

Входными данными являются:

- Файлы ресурсов и параметров модели
- Файл, содержащий фактическое положение групп вагонов

В качестве *управления* выступают требования к модулю ввода корректировок.

Механизмом является модуль ввода корректировок.

Выходными данными являются файлы ресурсов и параметров модели, с учетом фактического положения групп вагонов.

2.3Учет корректировок диспетчера

Данный функциональный блок описывает процесс внесения любых корректировок модель диспетчером посредством системы соответствующих экранных форм.

Входными данными являются:

- файл ресурсов модели с учетом фактического положения групп вагонов
- корректировки диспетчера

Корректировки диспетчера представлены сущностью КОРРЕКТИРОВКА. Ключевой атрибут - Номер_корректировки. Зависимые сущности:

- КОРРЕКТИРОВКА_ОПИСАНИЯ_ВАГОННОГО_ПАРКА
- КОРРЕКТИРОВКА_ОПИСАНИЯ_ЖД_СТАНЦИЙ
- КОРРЕКТИРОВКА_ОПИСАНИЯ_ГРУПП_ВАГОНОВ
- КОРРЕКТИРОВКА_ОПИСАНИЯ_ГРАФИКА_ОТПРАВКИ_ВАГОНОВ

представляют типы возможных корректировок, вносимых в модель.

В качестве *управления* выступают требования к модулю ввода корректировок.

Механизмом является модуль ввода корректировок.

Выходными данными являются файлы имитационной модели с учетом корректировок и фактического положения вагонов. Выходные данные представлены сущностью ИМИТАЦИОННАЯ_МОДЕЛЬ_С_УЧЕТОМ_КОРРЕКТИРОВОК. Также выходными данными являются записи в журнале корректировок о корректировках внесенных диспетчером.

Заметим, что предыдущий функциональный блок также предназначен для внесения корректировок в модель, однако, в данном случае корректировки вносятся пользователем вручную. Порядок следования блоков 2.2 и 2.3 непринципиален.

3. Имитационное моделирование процесса перевозок.

Данный функциональный блок описывает процесс интеллектуального имитационного моделирования процесса перевозок серы на языке РДО. Период времени, на который делается прогноз, задается диспетчером.

Входными данными являются файлы описания ресурсов модели с учетом текущего положения групп вагонов и корректировок диспетчера.

Механизмом является модуль имитационного моделирования (имитационная модель) на языке РДО.

Управлением являются требования к подсистеме диспетчирования.

Выходные данные представляют собой:

- файл *.pmd с собираемыми в ходе моделирования показателями
- отображение вагонного парка на упрощенной карте РЖД

4. Формирование отчета по результатам моделирования.

Данный функциональный блок описывает процесс формирования отчета о прогнозе выполнения графика отправки вагонов с учетом текущего положения групп вагонов и корректировок диспетчера.

Входными данными является файл с собранными в ходе моделирования показателями.

Механизмом является модуль формирования отчетов о результатах моделирования.

Управлением являются требования к модулю формирования отчетов о результатах моделирования.

Построение логической информационной модели предметной области

Логическая информационная модель описывает сущности входящие в систему транспортировки и связи между ними (Лист 4).

ЗАЯВКА

В соответствии с ТЗ подсистема формирования месячных и декадных планов формирует три типа заявок. Каждая заявка имеет уникальный номер, по которому ее можно однозначно идентифицировать в системе. Каждой заявке ставится в соответствие группа вагонов, предназначенная для ее выполнения. Общим атрибутом для всех заявок является Количество_вагонов. Типы заявок имеют атрибуты, соответствующие форме заявок, описанной в ТЗ. Каждая заявка в каждый момент времени находится в определенном состоянии. Диаграммы состояний заявок разрабатываются на этапе технического проектирования.

ЗАЯВКА_НА_ОТПРАВКУ_ВАГОНОВ_С_ГРУЗОМ

Заявка на отправку вагонов с грузом включает в себя следующие атрибуты:

- Вес_груза
- Тип_доставки (маршрутом или россыпью)
- Скорость_доставки (обычная или срочная)
- Тип_груза (гранулированная или жидкая сера)

Каждая заявка имеет ровно один пункт отправления серы, ровно один пункт назначения серы и ровно одну дату отправления груза.

ЗАЯВКА_НА_ОТПРАВКУ_ПОРОЖНИХ_ВАГОНОВ

Заявка на отправку порожних вагонов содержит ровно один пункт отправления порожних вагонов, ровно один пункт назначения порожних вагонов, ровно одну дату отправления порожних вагонов. Также данный тип

заявок имеет зависимую сущность «ТИП_ВАГОНОВ_В_ЗАЯВКЕ», содержащая атрибуты, описывающие тип вагонов и принадлежность вагонов.

ЗАЯВКА_НА_ОБРАБОТКУ_ВАГОНОВ

Заявка на обработку вагонов имеет атрибут один атрибут – Код_операции. Данный атрибут предназначен для описания типа операции: погрузка, разгрузка, формирование и т.д. Заявка на обработку вагонов имеет ровно один пункт обработки, ровно одну дату начала операции и ровно одну дату окончания операции. Также данный тип заявок имеет зависимую сущность - «ТИП_ВАГОНОВ_В_ЗАЯВКЕ».

ВАГОН

Каждый вагон в системе имеет уникальный номер, по которому его можно однозначно идентифицировать. Каждый вагон имеет в качестве зависимой сущности «ПРОБЕГ_ВАГОНА». Данная сущность содержит атрибуты, описывающие общий пробег вагона, пробег вагона с грузом и пробег вагона с момента последнего профилактического осмотра. Каждый вагон в каждый момент имеет определенное состояние. Диаграммы конечных состояний для сущности «ВАГОН» разработаны на этапе технического проектирования. Вагоны имеют определенный тип. Типы вагонов описаны с помощью зависимой сущности «ТИП_ВАГОНА», содержащей атрибуты: Название_типа и Грузоподъемность. Вагоны могут состоять или не состоять в группе.

ВАГОН_В_ГРУППЕ

В качестве мигрирующего ключа вагон в группе имеет ключ «Номер_группы_вагонов» Дислокация вагона в группе определяется дислокацией группы.

СВОБОДНЫЙ_ВАГОН

Свободный вагон, в отличие от вагона в группе, характеризуется собственной дислокацией. Дислокация свободного вагона определяется именем станции, на которой находится свободный вагон.

ГРУППА_ВАГОНОВ

Группа вагонов является зависимой сущностью для заявки, т.к. формируется для ее выполнения. Группа может включать не более 40 вагонов (не больше состава) или не включать вагонов совсем (если она только начала формироваться). Группа характеризуется уникальным номером и дислокацией группы. Дислокация группы задается пунктом отправления вагонов, пунктом назначения вагонов и положением группы по факту, принимающим значение от нуля до единицы. К примеру, если положение по факту равно 0.5, то группа вагонов находится посредине между пунктом отправления и пунктом назначения. Группа вагонов может находиться в одном из состояний, диаграммы состояний для групп вагонов разработаны на этапе технического проектирования.

СТАНЦИЯ

Станции характеризуются уникальным атрибутом – именем станции. В соответствии с ТЗ будем выделять станции трех типов: станции-поставщики, станции-потребители и станции ГПТ, на которых можно проводить профилактический ремонт вагонов. Станции имеют неидентифицирующие отношения со следующими сущностями:

- Дислокация свободных вагонов
- Пункт отправления группы
- Пункт назначения группы
- Пункт отправления серы
- Пункт назначения серы
- Пункт отправления порожних вагонов
- Пункт назначения порожних вагонов
- Пункт обработки

ФАКТ

Сущность «Факт» описывает связь между положением группы по расписанию (исходя из пункта отправления, пункта назначения и скорости движения) и фактическим положением группы в системе. Факт в качестве атрибутов имеет уникальный номер и время записи факта.

Техническое задание

Техническое задание разработано в соответствии с ГОСТ 19.201-78 (Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.)

Общие сведения

Разрабатываемая в рамках дипломного проектирования подсистема диспетчирования перевозок серы железнодорожным транспортом входит в состав АСУ ПС. Техническое задание на АСУ ПС было разработано специалистами ОАО ГПТ при участии специалистов кафедры РК-9 МГТУ им. Баумана. Полная версия данного технического задания находится на диске в приложении к пояснительной записке. Основным инструментом для разработки является среда имитационного моделирования РДО. Сроки проведения начального этапа работ по созданию подсистемы: 04.02.2010 – 08.06.2010 г.г.

В ходе предпроектного исследования и концептуального проектирования был выявлен модульный состав разрабатываемой подсистемы. Вследствие этого, следующие разделы технического задания по ГОСТ 19.201-78 относятся к подсистеме в целом:

- Назначение и цели создания системы
- Характеристика объектов автоматизации
- Требования к системе
- Порядок контроля и приемки системы
- Требования к документированию

И к каждому из модулей подсистемы:

- Требования к системе
- Состав и содержание работ по созданию системы

Назначение и цели создания системы

Создание подсистемы диспетчирования преследует следующие цели и подцели, определенные на этапе предпроектного исследования [стр. 17]:

- Повышение контроля выполнения ГОВ
- Повышение эффективности использования вагонного парка

Подсистема диспетчирования предназначена для выполнения следующих функций:

- Получение данных о фактическом положении вагонного парка
- Ввод корректировок в ГОВ
- Имитационное моделирование процесса перевозок серы
- Формирование отчета о прогнозе выполнения ГОВ

Характеристика объектов автоматизации

Объектом автоматизации подсистемы диспетчирования является процесс перевозок серы от предприятий ОАО Газпром потребителям по их заявкам средствами ОАО ГПТ и РЖД [стр. 10].

Требования к системе

Требования к системе в целом

Требования к подсистеме диспетчирования учитывают требования, сформулированные для АСУ ПС в целом [стр. 10]:

Подсистема диспетчирования должна создаваться на основе открытой архитектуры с учетом следующих требований:

- Возможность эволюции программно-аппаратных средств без перепрограммирования существующих приложений
- Преимущественное использование готовых стандартных программно-технических компонент в противовес разработке специализированных решений
- Использование стандартных (промышленных) интерфейсов взаимодействия между этими компонентами

Данные требования предъявляются ко всем модулям разрабатываемой подсистемы.

Требования к модулю ввода фактического положения групп вагонов

Данный модуль предназначен для получения данных о фактическом положении групп вагонов из базы данных РЖД. Модуль должен представлять собой пользовательское приложение Windows, реализованное в среде программирования Visual Studio 2005. Доступ к базе данных должен быть реализован посредством технологии ODBC.

Модуль должен осуществлять доступ к информации, содержащейся в базе данных, согласно информационной модели подсистемы диспетчирования [Лист_3] (сущность ФАКТ_ПОЛОЖЕНИЯ_ГРУППЫ_ВАГОНОВ):

- Номер группы вагонов
- Пункт отправления группы вагонов
- Пункт назначения группы вагонов
- Местоположение

Выходные данные должны быть представлены в виде текстового файла.

Пример выходных данных:

1 Аксарайская_1 Потребитель_2 0,4

2 Москва Петушки 0,1

……

…….

Требования к модулю формирования имитационной модели на начало планового периода

Данный модуль предназначен для формирования файлов имитационной модели перевозок на языке РДО, описывающих ресурсы и параметры на начало планового периода. При формировании данных файлов обязательно должно быть учтено следующее (*входные данные*):

- График отправки вагонов на период, сформированный подсистемой формирования месячных и декадных планов. В соответствии с ТЗ на АСУ ПС ГОВ имеет следующий вид:

График отправки вагонов с грузом:

№ группы вагонов	Пункт отправления серы	Пункт назначения серы	Вес груза (т)	Тип груза (жидкая, гранули-рованная)	Количество вагонов (шт)	Тип доставки (маршрутом, россыпью)	Скорость доставки (обычная, срочная)	Дата отправления груза	Текущая дислокация

График отправки порожних вагонов:

№ группы вагонов	Пункт отправления порожних вагонов	Пункт назначения порожних вагонов	Тип вагона	Принадлежность вагона (свой, арендованный)	Количество вагонов (шт)	Дата отправки порожних вагонов	Текущая дислокация

График обработки вагонов:

№ группы вагонов	Пункт обработки	Код (наименование операции)	Тип вагона (полувагон, цистерна)	Принадлежность вагона (свой, арендованный)	Количество вагонов (шт)	Дата начала операции	Дата окончания операции

- Состояние объектов, входящих в систему транспортировки (вагоны, группы вагонов, станции) на начало планового периода. Объекты и их состояния должны соответствовать разработанной логической информационной модели предметной области [Лист 4].
- Статистические данные, предоставляемые подсистемой анализа и статистики
  - Скорость движения вагонов (с грузом, порожних)
  - Скорость обработки вагонов (погрузка, разгрузка, профилактический ремонт)

На основании статистических данных пользователь системы должен иметь возможность формирования следующих типов имитационной модели:

- Нормативная

Длительность и результат всех операций соответствуют нормативным документам

- Среднестатистическая

Длительность и результат всех операций соответствует данным, полученным на основании обработки статистики.

  - Оптимистическая (минимальная длительность)
  - Пессимистическая (максимальная длительность)
  - Среднестатистическая (среднестатистическая длительность)

- Случайная

Длительность и результат всех операций генерируется генераторами случайных чисел по данным, полученным на основании обработки статистики.

- Прочие данные необходимые для однозначного, непротиворечивого, адекватного описания процесса транспортировки (Например: расстояния между станциями РЖД)

Требования к входящим данным могут быть уточнены при написании имитационной модели.

Выходные данные представляют собой следующие файлы описания ресурсов и параметров модели:

- Файл ресурсов модели *.rss

Данный файл должен содержать описание станций, вагонов, групп вагонов, пунктов графика отправки и обработки вагонов, входящих в модель.

- Файл описания функций и констант *.fun

Данный файл должен содержать статистические данные о скорости движения и обработки вагонов, изменяемые по требованию пользователя.

- Файл описания показателей *.pmd

Данный файл должен содержать описание всех показателей, собираемых в ходе моделирования, которые будут включаться в отчет о результатах прогноза.

- Файл описания кадров анимации *.frm

Данный файл должен содержать описание объектов анимации, которые символизируют отображение всех объектов моделируемой системы на упрощенной карте РЖД.

Структура данных файлов должна быть уточнена при разработке модуля имитационного моделирования (имитационной модели).

Требования к модулю ввода корректировок

Данный модуль предназначен для реализации диалога формирования файлов имитационной модели, учитывающих корректировки диспетчера и фактическое положения групп вагонов по данным РЖД. Иными словами, данный модуль подготавливает данные для передачи в РДО-имитатор. Модуль должен представлять собой пользовательское приложение Windows, реализованное в среде программирования Visual Studio 2005.

Входными данными являются:

- Данные о текущем положении групп вагонов (от РЖД)

Текстовый файл, сформированный модулем ввода фактического положения групп вагонов [стр. 26]

- Файлы ресурсов и параметров модели на начало планового периода

Файлы имитационной модели на языке РДО (*.rss, *.fun, *.pmd, *.frm), сформированные модулем формирования имитационной модели на начало планового периода.

- Корректировки диспетчера

Диспетчер должен иметь возможность вносить любые изменения в ресурсы модели посредством соответствующих экранных форм.

1. Корректировки параметров существующих ресурсов модели

(Например, увеличение скорости выполнения определенных заявок по согласованию с РЖД и т.д.)

2. Изменение набора ресурсов модели

(Например, ввод дополнительного пункта расписания отправки группы порожних вагонов, исключение из модели вагонов в связи с выводом из эксплуатации и т.д.)

Должна быть предусмотрена система учета корректировок. Каждая вносимая (удаляемая) корректировка должна делаться соответствующая запись в журнале корректировок (Суть корректировки, а также, кем и когда она была сделана)

Выходными данными являются файлы имитационной модели на языке РДО, учитывающие сделанные корректировки и текущее положение групп вагонов.

Требования к модулю имитационного моделирования на языке РДО

Данный модуль предназначен для реализации процесса моделирования в РДО-имитаторе. Модуль должен представлять собой пользовательское приложение Windows, реализованное в среде программирования Visual Studio 2005.

Модуль должен реализовывать следующие возможности:

- Возможность выбора между однократным прогоном модели или многократного прогона с последующей статистической обработкой результатов прогонов.
- Отображение укрупненного состояния выполнения заявок в ходе моделирования; отображение положения вагонного парка на упрощенной карте РЖД.

Входными данными являются файлы имитационной модели на языке РДО, учитывающие сделанные корректировки и текущее положение групп вагонов, сформированные модулем ввода корректировок [стр.28]

Файлы описания операций (*.pat, *.opr) и типов ресурсов (*.rtp) не изменяются в ходе нормальной работы системы и включены в состав рассматриваемого модуля.

Выходными данными является файл с собранными в ходе прогона показателями - *.pmv, который передается в модуль формирования отчета о результатах моделирования.

Требования к модулю формирования отчетов о результатах моделирования

Отчет должен в максимально наглядной форме информировать пользователей о результатах прогноза. Предлагается реализовать данный модуль совместно с модулем имитационного моделирования. В качестве

программной платформы для реализации следует применять систему генерации отчетов Fast Report.

Отчет должен формироваться каждый раз при окончании прогона модели и должен включать следующие показатели:

- Разбивка прогноза выполнения пунктов ГОВ по группам. Будем выделять три группы:

1. «Зеленая» группа – пункт расписания выполняется в срок или выполняется с отставанием от графика на a%.
2. «Желтая» группа – пункт расписания выполняется с незначительным опозданием; Отставание более a% и менее b% от полного времени выполнения операции.
3. «Красная» группа – пункт расписания выполняется со значительным опозданием; Отставание более b% от полного времени выполнения операции.

a и b – параметры имитационной модели

- Статистика по вагонам: общий пробег, пробег с грузом, пробег с момента последнего профилактического осмотра.

Необходимую вспомогательную информацию:

- время и дата создания отчета
- вариант прогноза (однократный, статистическая обработка)
- плановый период
- краткий перечень проведенных корректировок

Отметим, что данный состав отчета является предварительным и может меняться в зависимости от требований конечного пользователя (диспетчер, плановый отдел).

Входные данные представляют собой файл (*.pmv) с собранными в ходе прогона показателями, сформированный модулем имитационного моделирования.

Выходные данные представляют собой файл отчета, сгенерированный системой Fast Report.

Состав и содержание работ по созданию системы

Состав этапов и работ разработан с учетом ГОСТ 34.601-90 (Стадии создания автоматизированных систем) В данной таблице перечислены работы, проведенные по созданию подсистемы диспетчирования в рамках дипломного проектирования:

Этапы	**Работы**
Предпроектное исследование	1. Анализ деятельности ОАО ГПТ 2. Определение места разрабатываемой подсистемы в составе АСУ ПС 3. Сравнительный анализ и выбор технологий решения основных задач
Концептуальное проектирование	1. Определение целей разработки подсистемы диспетчирования 2. Построение информационной и функциональной моделей подсистемы диспетчирования 3. Построение логической информационной модели предметной области 4. Разработка технического задания
Техническое проектирование	1. Разработка алгоритма доступа к БД посредством технологии ODBC (модуль ввода фактического состояния групп вагонов) 2. Разработка диаграммы конечных состояний ресурсов модели на языке UML (модуль имитационного моделирования на языке РДО)
Рабочее проектирование	1. Разработка имитационной модели на языке РДО (модуль имитационного моделирования на языке РДО) 2. Создание модуля формирования отчетов о результатах моделирования
Контроль и приемка	1. Разработка тестового примера и апробирование разработанной системы

Техническое проектирование

Разработка диаграмм конечных состояний ресурсов модели

На этапе технического проектирования предлагается разработать диаграммы конечных состояний, для сущностей, входящих в систему: вагонов, групп вагонов, заявок. Диаграммы разрабатываются в соответствии со стандартом языка UML. В UML автомат является пакетом, в котором определено множество понятий, необходимых для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом состояний и переходов. Состояние и переход - основные понятиями, входящие в формализм автомата. Главное различие между ними заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. В общем случае автомат представляет динамические аспекты моделируемой системы в виде ориентированного графа, вершины которого соответствуют состояниям, а дуги - переходам.

Разработка диаграммы конечных состояний вагона.

Разработанная диаграмма конечных состояний вагона представлена на листе 6 данного дипломного проекта. Данная диаграмма описывает состояния, в которых может находиться вагон по техническому заданию. Диаграмма не учитывает спонтанных поломок вагона. На данной диаграмме не показаны начальное и конечное состояния, т.к. вагоны рассматриваются как постоянный ресурс в системе.

Разработка диаграммы конечных состояний заявки.

Под заявками будем подразумевать пункты расписания, составленного подсистемой формирования месячных и декадных планов (в отличие от заявок потребителей). В начале все заявки имеют состояние «Не выполняется, не опаздывает», выполнение заявки начинается в срок, заранее заявки не выполняются. Когда наступает время выполнения заявки, она

может перейти в одно из трех состояний, соответствующих принятой разбивки по группам [стр.35] в зависимости от величины задержки. Остальные переходы показаны на листе 6 дипломного проекта. Выделим конечные состояния «Закончена с опозданием», «Закончена с незначительным опозданием» и «Закончена вовремя», как наиболее важные, поскольку по тому в каком из них окажется заявка по истечении сроков ее выполнения можно качественно судить о результатах прогноза выполнения заявки.

Разработка диаграммы конечных состояний групп вагонов.

Связывающим состоянием является состояние «Сформирована на станции», в это состояние группа вагонов переходит после того как была сформирована. Если группа находится в этом состоянии, это означает, что она готова к выполнению действий, соответствующих расписанию: погрузке, разгрузке, профилактическому ремонту и расформированию.

Разработка алгоритма доступа к БД по средствам технологии ODBC.

На листе 8 представлена диаграмма классов библиотеки MFC, предназначенных для работы с базами данных с использованием механизма ODBC.

Любое приложение, которое работает с такими базами данных, обязано иметь в своем составе, по крайней мере, два класса: ***CDatabase*** и ***CRecordset***.

Класс CDatabase.

Объекты данного класса используются для соединения с базами данных, посредством которого можно манипулировать источником данных.

Для создания объекта CDatabase служит конструктор `CDatabase::CDatabase()`

После того, как объект создан необходимо установить соединение с определенным источником данных.

Для этого может быть использована функция

`BOOL CDatabase::Open(LPCTSTR IpszDSN, BOOL bExclusive, BOOL bReadOnly, LPCTSTR IpszConnect, BOOL bUseCursorLib)`

Параметр `IpszDSN` определяет имя источника данных, которое должно быть зарегистрировано с помощью программы ODBC Administrator.

Параметр `bExclusive` определяет открывается ли источник данных для совместного использования.

Параметр `bReadOnly` позволяет предоставить доступ к источнику данных в режиме «только для чтения».

Параметр `IpszConnect` определяет строку, описывающую соединение, которая содержит информацию об источнике данных, идентификаторе пользователя, имеющего к нему доступ, пароль и другую информацию.

Параметр `bUseCursorLib` указывает на необходимость загрузки динамической библиотеки ODBC Cursor Library, позволяющей работать с базами данных.

Закрытие доступа к БД осуществляется функцией `BOOL CDatabase::Close()`

Наиболее часто в классе `CDatabase`, используются нижеследующие функции, предоставляющие данные о соединении, драйвере и источнике данных:

`CStringS CDatabase::GetConnect()` вызов этой функции позволяет получить описывающую соединение строку, которая использовалась во время вызова функции `Open()`

`BOOL CDatabase::IsOpen()` позволяет определить, имеется или нет текущее соединение объекта `CDatabase` с базой данных.

`BOOL CDatabase::CanUpdate()` устанавливает, может ли пользователь обновлять базу данных.

Класс CRecordset.

Данный класс позволяет пользователю работать с набором записей. В классе `CRecordset` определены следующие основные компоненты данных:

`UINT CRecordset::m_nFields` содержит число полей данных в результирующем наборе – число столбцов, получаемых из источника данных.

`CDatabase CRecordset::m_pDatabase` содержит указатель на объект класса CDatabase, посредством которого результирующий набор соединяется с источником данных.

Для конструирования класса `CRecordset` используются следующие функции:

Конструктор `CRecordset::CRecordset(CDatabase* pDatabase = NULL)` служит для инициализации и создания объекта класса `CRecordset`.

Функция `BOOL CRecordset::Open(UINT nOpenType, LPCTSTR lpszSQL, DWORD dwOptions)` служит для открытия доступа к результирующему набору. Параметр **`nOpenType`** определяет тип доступа к источнику данных и может принимать следующие значения:

`CRecordset::dynaset`

Результирующий набор с возможностью двунаправленного просмотра. При этом режиме изменения, вносимые в базу данных другими пользователями, отображаются сразу же. К сожалению, декларированное поведение результирующего набора при этом режиме не поддерживается

`CRecordset::snapshot`

Статический результирующий набор с возможностью двунаправленного просмотра. Изменения, вносимые в базу данных другими пользователями, отображаются только после повторного открытия результирующего набора.

`CRecordset::dynamic`

Результирующий набор с возможностью двунаправленного просмотра. При этом режиме изменения, вносимые в базу данных другими пользователями, отображаются при выполнении следующей операции. Многие драйверы ODBC не поддерживают этот режим доступа

`CRecordset::forwardOnly`

Результирующий набор "только для чтения" с возможностью просмотра "только вперед".

Параметр **IpszSQL** — указатель на строку, содержащую одно из значений: NULL, имя таблицы, оператор SQL, не обязательно с предложениями WHERE или ORDER BY, или оператор CALL, определяющий имя предопределенного запроса или сохраненной процедуры.

Функция `void CRecordset::Close()` служит для закрытия результирующего набора.

Класс `CRecordset` предоставляет следующие основные функции для предоставления доступа к атрибутам результирующего набора:

`const CStringS CRecordset::GetTableName()`

Позволяет получить имя таблицы, на которой основывается запрос результирующего набора, или пустую строку.

`const CStringS CRecordset::GetSQL()`

Позволяет получить оператор SQL, который используется для выборки записей результирующего набора

`BOOL CRecordset::IsOpen()`

Позволяет определить, открыт ли уже результирующий набор.

`BOOL CRecordset::IsBOF()`

Позволяет определить, является ли текущая запись первой в наборе данных.

```
BOOL CRecordset::IsEOF()
```

Позволяет определить, является ли текущая запись последней в наборе данных.

```
BOOL CRecordset::IsDeleted()
```

Позволяет определить, была ли текущая запись удалена.

Основные функции перемещения по результирующему набору:

```
void CRecordset::Move (long nRows,WORD wFetchType =
SQL_FETCH_RELATIVE)
```

Функция имеет два параметра: nRows — количество строк, на которое необходимо переместиться вперед (положительное значение) или назад (отрицательное) и wFetchType — определяет набор строк, которые функция должна выбрать.

```
void CRecordset::MoveFirst()
```

Делает текущей первую запись результирующего набора.

```
void CRecordset::MoveLast()
```

Делает текущей последнюю запись результирующего набора.

```
void CRecordset: MoveNext()
```

Делает текущей первую запись следующего набора строк.

```
void CRecordset::MovePrev()
```

Делает текущей первую запись предыдущего набора строк.

Класс `CRecordset` также содержит переопределяемые методы, которые должны быть определены программистом в классе-обертке. Они рассмотрены ниже при описании класса `CUserSet.`

Класс CUserSet

Данный класс является пользовательской оберткой класса CRecordset и включает в себя переопределяемые функции, настраиваемые на решение конкретных задач. Также класс CUserSet должен включать в себя параметры (поля результирующего набора), используемые при выгрузке данных из конкретной БД [стр. 30]:

m_Group_number: Long Номер группы вагонов

m_Start_point: String Пункт отправления группы вагонов

m_Destination: String Пункт назначения группы вагонов

m_Location: Double Положение группы вагонов между пунктом отправления и пунктом назначения

Переопределяемые функции:

1. void CRecordset::**DoFieldExchange** (CFieldExchange* pFX)

Вызывается для организации обмена данными между полями результирующего набора и соответствующими столбцами текущей записи в источнике данных. В качестве параметра функция принимает указатель на объект CFieldExchange, который автоматически создается и передается библиотекой MFC. Обмен данными осуществляется с помощью механизма RFX (Record Field Exchange, Обмен полями записи), который работает в обоих направлениях: от полей данных результирующего набора к записям источника данных и наоборот. Ниже приведен фрагмент, демонстрирующий переопределение функции DoFieldExchange.

```
void CDBSet::DoFieldExchange(CFieldExchange* pFX)
{
	RFX_Long(pFX, _T("[Group_number]"), m_Group_number);
	RFX_Text(pFX, _T("[Start_point]"), m_Start_point);
	RFX_Text(pFX, _T("[Destination]"), m_Destination);
	RFX_Double(pFX, _T("[Location]"), m_Location);
}
```

2. CString CRecordset::**GetDefaultConnect**()

Библиотека MFC вызывает данную функцию, чтобы получить строку содержащую источник данных, на котором базируется результирующий набор.

3. `CString CRecordset::GetDefaultSQL ()`

Библиотека MFC вызывает эту функцию, чтобы получить строку, содержащую оператор SQL, на котором базируется результирующий набор. Это должно быть или имя таблицы, или непосредственно оператор SELECT.

К примеру:

```
CString CMVTSSet::GetDefaultSQL()
{
    return _T("[Groups_Location]");
}
```

Класс CRecordView

Объекты этого класса предоставляют для изображения записей базы данных в элементах управления форму, которая непосредственно соединена с объектом CRecordset. Объекты CRecordView используют механизм `DDX(Dialog Data Exchange`, Обмен данными с блоком диалога) и `RFX (Record Field Exchange`, Обмен полями записей) для автоматического перемещения данных между элементами управления формы и полями результирующего набора.

Класс содержит следующие основные функции:

Конструктор `CRecordView::CRecordView(UINT nIDTemplate)`, параметр `nIDTemplate` – идентификатор шаблона блока диалога.

`BOOL CRecordView::IsOnFirstRecord()`

Позволяет определить, является ли текущая запись первой в результирующем наборе, ассоциированном с данной формой.

`BOOL CRecordView::IsOnLastRecord()`

Позволяет определить, является ли текущая запись последней в результирующем наборе, ассоциированном с данной формой.

Данный класс также содержит виртуальные функции, требующие переопределения. Данные функции рассмотрены ниже, при рассмотрении класса `CUserView.`

Класс CUserView

Класс-обертка для `CRecordView`, в котором переопределяются следующие виртуальные функции:

`CRecordset* CRecordView::OnGetRecordset()`

Возвращает указатель на объект CRecordset, ассоциированный с формой, позволяя тем самым работать с некоторым результирующим набором.

`BOOL CRecordView::OnMove(UINT nIDMoveCommand)`

Позволяет программисту перемещаться по записям результирующего набора.

`nIDMoveCommand` задает направление перемещения и может принимать следующие значения:

`ID_RECORD_FIRST` Переход к первой строке в результирующем наборе

`ID_RECORD_LAST` Переход к последней строке в результирующем наборе

`ID_RECORD_NEXT` Переход к следующей строке в результирующем наборе

`ID_RECORD_PREV` Переход к предыдущей строке в результирующем наборе

```
void CMVTSView::DoDataExchange(CDataExchange* pDX)
```

Функция предназначена для обмена данными между результирующим набором и полями экранной формы. К примеру:

```
void CMVTSView::DoDataExchange(CDataExchange* pDX)
{
     CRecordView::DoDataExchange(pDX);
     DDX_FieldText(pDX,     IDC_EDIT1,     m_pSet->m_Group_number,
m_pSet);
     DDX_FieldText(pDX,     IDC_EDIT2,     m_pSet->m_Start_point,
m_pSet);
     DDX_FieldText(pDX,     IDC_EDIT3,     m_pSet->m_Destination,
m_pSet);
     DDX_FieldText(pDX, IDC_EDIT4, m_pSet->m_Location, m_pSet);
}
```

Рабочее проектирование

Разработка имитационной модели на языке РДО

В состав имитационной модели, написанной на языке РДО, входят следующие файлы:

1. Файл *.rtp определяет структуру глобальной базы знаний модели
2. Файл *.rss определяет начальное состояние глобальной базы данных модели
3. Файл *.pat содержит знания о функционировании моделируемой системы (знания о предметной области), записанные в виде модифицируемых продукционных правил в соответствии с синтаксисом языка
4. Файл *.opr определяет содержимое базы знаний модели
5. Файл *.dtp описывает способы использования образцов для моделирования процесса и принятия решений на уровне событий
6. Файл *.fun определяет символьные константы и функции модели
7. Файл *.frm определяет содержимое кадров анимации модели
8. Файл *.pmd содержит описание показателей, которые необходимо собрать в ходе прогона модели
9. Файл *.smr содержит ряд необходимых для управления прогоном данных и режимов

Файлы *.rss, *.fun, *.frm, *.pmd, *.smr формируются модулем формирования модели на начало планового периода и изменяются модулем ввода корректировок.

Файлы *.rtp, *.pat, *.opr, *.dpt остаются неизменными в ходе работы подсистемы диспетчирования.

При разработке модели установим следующее соответствии между реальным и модельным временем: одна единица модельного времени соответствует одному часу реального времени.

Типы ресурсов модели

Вагоны

Вагоны являются постоянным ресурсом модели. Количество и параметры вагонов определяются пользователем в модуле формирования модели на начало планового периода [стр.31] и могут быть скорректированы при помощи модуля ввода корректировок [стр.28].

$Resource_type Вагоны: permanent
$Parameters

Каждый вагон имеет уникальный номер:

Номер_вагона : integer

В соответствии с техническим заданием доставка серы может осуществляться двумя типами вагонов: собственными вагонами ОАО ГПТ и арендованными.

Принадлежность : (свой, арендованный) = свой

Параметр «Загруженность» определяет степень загруженность вагона (0 – вагон порожний, 1 – вагон загружен полностью)

Загруженность : real[0..1] = 0

Параметр «Состояние» определяет состояние вагона в соответствии с диаграммой конечных состояний, разработанной на этапе технического проектирования.

Состояние : (порожний_на_станции, порожний_отсоединение, порожний_не_в_группе, порожний_присоединение, порожний_в_пути, прибыл_на_погрузку, на_погрузке, погружен_на_станции, погружен_отсоединение, погружен_не_в_группе, погружен_присоединение, погружен_в_пути, прибыл_на_разгрузку, на_разгрузке, требует_ремонт, прибыл_на_ремонт, на_ремонте, в_пути_на_ремонт, требует_ремонт_на_станции, требует_ремонт_отсоединение, требует_ремонт_не_в_группе, требует_ремонт_присоединение) = порожний_не_в_группе

В соответствии с техническим заданием в системе транспортировки присутствуют различные типы вагонов с различной грузоподъемностью.

Тип_вагона : (полувагон, цистерна)
Грузоподъемность : integer

Пробег вагона задается тремя параметрами: «Пробег» - общий пробег вагона, «Пробег с грузом» - параметр, необходимый для вычисления КПД вагона, «Пробег с момента ремонта» - параметр, необходимый для определения времени отправки вагона на очередной профилактический ремонт.

Пробег : real = 0
Пробег_с_грузом : real = 0
Пробег_с_момента_ремонта : real = 0

Вагоны могут включаться в группы. Если вагон не входит не в одну из групп, то параметр «Номер группы» принимает значение -1.

Номер_группы : integer = -1

Для вагонов, не состоящих в группах, задается параметр «Дислокация свободного вагона», принимающий значение имени станции, на которой находится вагон.

Дислокация_свободного_вагона : such_as Станции.Имя

Параметры «Абсцисса местоположения» и «Ордината местоположения» предназначены для отображения вагона на кадре анимации модели, отображающем состояние вагонного парка и разбивку вагонов по группам.

Абсцисса_местоположения : integer
Ордината_местоположения : integer
$End

Станции

Станции являются постоянным ресурсом модели.

$Resource_type Станции : permanent
$Parameters

Каждая станция имеет уникальное имя.

Имя : (Каргала, Аксарайская_2, Потребитель_1, Потребитель_2, Станция_тех_обслуживания)

В соответствии с техническим заданием будем выделять три типа станций – поставщики, потребители и станции ГПТ.

Тип_станции : (поставщик, потребитель, ГПТ)

Параметры «Абсцисса» и «Ордината» предназначены для отображения станций на упрощенной карте железных дорог.

Абсцисса : integer
Ордината : integer

Параметр «Количество вагонов» определяет количество порожних вагонов, готовых для формирования в группы.

Количество_вагонов : integer
$End

Группы вагонов

Группы вагонов являются переменным ресурсом модели.

$Resource_type Группы_вагонов : temporary
$Parameters

Параметр «Вес груза» определяет общее количество серы в тоннах, находящееся во всех вагонах групп.

Вес_груза : integer = 0
Количество_вагонов : integer = 0

Каждая группа имеет уникальный номер.

Номер_группы_вагонов : integer

Каждой группе в каждый момент времени соответствует один пункт ГОВ, для выполнения которого она предназначена.

Номер_заявки : integer

Параметр «Состояние» определяет состояние группы вагонов в соответствии с диаграммой конечных состояний, разработанной на этапе технического проектирования.

Состояние : (не_сформирована, формируется, присоединение_вагона, на_станции, расформируется, отсоединение_вагона, в_пути, конец_итерации, на_погрузке, погрузка_вагона, на_разгрузке, разгрузка_вагона, на_ремонте, ремонт_вагона) = не_сформирована

Параметры «Пункт отправления» и «Пункт назначения» принимают одно из определенных в модели имен станций.

Пункт_отправления : such_as Станции.Имя
Пункт_назначения : such_as Станции. Имя

Параметр «Положение по факту» задает местоположение группы вагонов относительно пункта назначения и пункта отправления. К примеру, если группа находится в пункте отправления, Положение_по_факту = 0. В пункте назначения: Положение_по_факту = 1. Если группа вагонов находится в пути: 0 < Положение_по_факту < 1.

Положение_по_факту : real[0..1] = 0

Координаты пунктов назначения и отправления группы используются при отображении группы на упрощенной карте ЖД.

Абсцисса_пункта_отправления : integer = 0
Ордината_пункта_отправления : integer = 0
Абсцисса_пункта_назначения : integer = 0
Ордината_пункта_назначения : integer = 0

Параметр «Количество отремонтированных вагонов» используется при оценке времени до окончания профилактического ремонта вагонов.

Количество_неотремонтированных_вагонов : integer = 0
$End

Пункт расписания обработки вагонов

Все пункты ГОВ являются постоянными ресурсами модели.

$Resource_type Пункт_расписания_обработки_вагонов: permanent
$Parameters

Каждая заявка имеет уникальный номер.

Номер_заявки : integer

Каждой заявке ставиться в соответствие группа вагонов для ее выполнения.

Номер_группы_вагонов : integer

Параметры «Состояние» и «Опоздание» определяют состояние заявки в соответствии с диаграммой конечных состояний, разработанной на этапе технического проектирования.

Состояние : (не_выполняется, выполняется, закончена) = не_выполняется
Опоздание : (зеленая, желтая, красная) = зеленая

Параметр «Статус проверки» используется операциями проверки выполнения ГОВ.

Статус_проверки : (требует_проверки, проверяется) = требует_проверки
Количество_обработанных_вагонов : integer = 0

Остальные параметры соответствуют форме пункта расписания обработки вагонов, представленного в техническом задании [стр. 31]

Количество_вагонов : integer
Вес_груза : integer = -1
Код_операции : (погрузка, разгрузка, ремонт, формирование_порожних, расформирование_порожних)
Пункт_обработки : such_as Станции.Имя
Принадлежность_вагонов : such_as Вагоны.Принадлежность
Тип_вагонов : such_as Вагоны.Тип_вагона
Дата_начала_операции : real
Дата_окончания_операции : real

$End

Пункт отправки порожних вагонов

$Resource_type Пункт_расписания_отправки_порожних_вагонов: permanent
$Parameters

Номер_заявки : integer
Номер_группы_вагонов : integer
Состояние : (не_выполняется, выполняется, закончена) = не_выполняется
Опоздание : (зеленая, желтая, красная) = зеленая
Количество_вагонов : integer
Принадлежность_вагонов : such_as Вагоны.Принадлежность
Тип_вагонов : such_as Вагоны.Тип_вагона
Пункт_отправления_порожних_вагонов : such_as Станции.Имя
Пункт_назначения_порожних_вагонов : such_as Станции.Имя
Дата_прибытия : real = 0
Дата_прибытия : real = 0
Статус_проверки : (требует_проверки, проверяется) = требует_проверки

Параметр «Факт» содержит информацию о местоположении группы порожних вагонов в ходе имитационного эксперимента. Данный параметр используется для сравнения с плановым, который рассчитывается по следующей зависимости:

Плановое местоположение = (**Time_now** – *Дата_отправления*)/ (*Дата_прибытия - Дата_отправления);* при *Дата_отправления* < **Time_now** < *Дата_прибытия,* **Time_now –** текущее время имитационной модели.

Параметр *Факт* равен параметру *Местоположение* группы вагонов, выполняющей данный пункт ГОВ.

Факт : real = 0
$End

Пункт отправки вагонов с грузом

$Resource_type Пункт_расписания_отправки_вагонов_с_грузом: permanent
$Parameters

Номер_заявки : integer
Номер_группы_вагонов : integer
Состояние : (не_выполняется, выполняется, закончена) = не_выполняется
Опоздание : (зеленая, желтая, красная) = зеленая
Количество_вагонов : integer
Вес_груза : real
Тип_доставки : (маршрутом, россыпью) = маршрутом
Скорость_доставки : (обычная, срочная) = обычная
Тип_груза : (жидкая, гранулированная)

Пункт_отправления_серы : such_as Станции.Имя
Пункт_назначения_серы : such_as Станции.Имя
Дата_отправления : real
Дата_прибытия : real
Статус_проверки : (требует_проверки, проверяется) = требует_проверки
Факт : real = 0
$End

Шаблоны операций модели

Содержание базы знаний модели (структура точек принятия решений) представлено на листе 9 дипломного проекта. Ниже приведем описание операций и правил, моделирующих:

- процесс отправки порожних групп
- процесс проверки выполнения отправки порожних групп
- процесс формирования группы вагонов

Отправка порожней группы

Блок-схема алгоритма моделирования отправки порожней группы представлена на листах 9, 10 дипломного проекта. Отправка порожних групп моделируется следующими продукционными правилами и операциями:

$Decision_point** Отправка_группы_порожних_вагонов : **some
$Condition NoCheck
$Activities
_Начало_отправки_порожней_группы : Начало_отправки_порожней_группы
_Начало_отправки_порожних_вагонов : Начало_отправки_порожних_вагонов
_Продвижение_порожней_группы : Продвижение_порожней_группы
_Прибытие_порожних_вагонов : Прибытие_порожних_вагонов
_Прибытие_порожней_группы : Прибытие_порожней_группы
$End

1. Продукционное правило «Начало отправки порожней группы» переводит ресурсы «Пункт расписания» и «Группа вагонов» в состояния «выполняется» и «конец_итерации» в случае, если соответствующие ресурсы подобраны.

$Pattern** Начало_отправки_порожней_группы : **rule
$Relevant_resources
Пункт_расписания : Пункт_расписания_отправки_порожних_вагонов
Keep
*Пункт_назначения : Станции **NoChange***
*Группа_порожних_вагонов : Группы_вагонов **Keep***
$Body
Пункт_расписания

```
            Choice from
                  Пункт_расписания.Состояние = не_выполняется and
                  Пункт_расписания.Дата_отправления <= Time_now
            Convert_rule
                  Состояние set выполняется
      Пункт_назначения
            Choice from
                  Пункт_расписания.Пункт_назначения_порожних_вагонов =
            Пункт_назначения.Имя
      Группа_порожних_вагонов
            Choice from
Группа_порожних_вагонов.Вес_груза = 0 and
Группа_порожних_вагонов.Количество_вагонов =
Пункт_расписания.Количество_вагонов and
Группа_порожних_вагонов.Номер_группы_вагонов =
Пункт_расписания.Номер_группы_вагонов and Группа_порожних_вагонов.Состояние =
на_станции and Группа_порожних_вагонов.Пункт_отправления =
Пункт_расписания.Пункт_отправления_порожних_вагонов
            Convert_rule
Пункт_назначения  set     Пункт_расписания.Пункт_назначения_порожних_вагонов
Номер_заявки                  set     Пункт_расписания.Номер_заявки
Абсцисса_пункта_назначения    set     Пункт_назначения.Абсцисса
Ордината_пункта_назначения    set     Пункт_назначения.Ордината
Состояние                     set     конец_итерации
$End
```

2. Продукционное правило «Начало отправки порожних вагонов» переводит вагоны, входящие в группу, в состояние «порожний_в_пути».

```
$Pattern Начало_отправки_порожних_вагонов :       rule
$Relevant_resources
      Пункт_расписания  : Пункт_расписания_отправки_порожних_вагонов
NoChange
      Группа_вагонов    :     Группы_вагонов    NoChange
      Вагон             :     Вагоны                  Keep
$Body
      Пункт_расписания
            Choice from
                  Пункт_расписания.Состояние = выполняется
      Группа_вагонов
            Choice from
                  Группа_вагонов.Состояние = конец_итерации and
                  Группа_вагонов.Номер_заявки = Пункт_расписания.Номер_заявки
      Вагон
            Choice from
                  Вагон.Номер_группы = Группа_вагонов.Номер_группы_вагонов and
                  Вагон.Состояние = порожний_на_станции
            Convert_rule
                  Состояние set порожний_в_пути
$End
```

3. Операция «Продвижение порожней группы» моделирует продвижение группы вагонов с течением модельного времени. (Увеличение параметра *Положение_по_факту*) Данная операция выполняется до тех пор, пока группа не прибудет на станцию (*Положение_по_факту* = 1). Функция «Продвижение группы порожних вагонов» описана ниже, в разделе «Функции и константы модели».

```
$Pattern Продвижение_порожней_группы : operation
$Relevant_resources
Группа_вагонов :Группы_вагонов Keep Keep
Пункт_расписания :Пункт_расписания_отправки_порожних_вагонов NoChange
    Keep
$Time = Единица_продвижения_модельного_времени
$Body
    Группа_вагонов
        Choice from
            Группа_вагонов.Положение_по_факту < 1 and
            Группа_вагонов.Состояние = конец_итерации
        Convert_begin
            Состояние set в_пути
        Convert_end
            Состояние set конец_итерации
            Положение_по_факту set
    Продвижение_группы_порожних_вагонов(Группа_вагонов.Положение_по_факту,
Группа_вагонов.Пункт_отправления, Группа_вагонов.Пункт_назначения)
    Пункт_расписания
        Choice from
            Пункт_расписания.Номер_заявки = Группа_вагонов.Номер_заявки
        and
            Пункт_расписания.Состояние = выполняется
        Convert_end
            Факт set
    Продвижение_группы_порожних_вагонов(Пункт_расписания.Факт,
Пункт_расписания.Пункт_отправления_порожних_вагонов,
Пункт_расписания.Пункт_назначения_порожних_вагонов)
$End
```

4. Продукционное правило «Прибытие порожних вагонов» моделирует прибытие вагонов на станцию назначения. Правило выполняется, если параметр соответствующей группы *Положение_по_факту = 1*. Правило производит увеличение значения пробега каждого вагона.

```
$Pattern Прибытие_порожних_вагонов :        rule
$Relevant_resources
        Пункт_расписания :      Пункт_расписания_отправки_порожних_вагонов
        Keep
        Группа_вагонов      :   Группы_вагонов                  NoChange
        Вагон               :   Вагоны                          Keep
$Body
        Пункт_расписания
                Choice from
                        Пункт_расписания.Состояние = выполняется
                Convert_rule
                        Факт set 1
        Группа_вагонов
                Choice from
                        Группа_вагонов.Состояние = конец_итерации and
                        Группа_вагонов.Положение_по_факту = 1
        Вагон
                Choice from
                        Вагон.Номер_группы = Группа_вагонов.Номер_группы_вагонов and
                        Вагон.Состояние = порожний_в_пути
                Convert_rule
                Пробег                  set     Вагон.Пробег + Длина_пути
                        (Пункт_расписания.Пункт_отправления_порожних_вагонов,
                        Пункт_расписания.Пункт_назначения_порожних_вагонов)
                Пробег_с_момента_ремонта        set
        Вагон.Пробег_с_момента_ремонта +
Длина_пути(Пункт_расписания.Пункт_отправления_порожних_вагонов,
Пункт_расписания.Пункт_назначения_порожних_вагонов)
                Состояние   set     порожний_на_станции
$End
```

5. Продукционное правило «Прибытие порожней группы» переводит ресурсы «Пункт_расписания» и «Группа вагонов» в состояния «закончена» и «на станции» после того, как изменены состояния всех вагонов, входящих в группу.

```
$Pattern Прибытие_порожней_группы :        rule
$Relevant_resources
        Пункт_расписания :      Пункт_расписания_отправки_порожних_вагонов
        Keep
        Группа_вагонов      :   Группы_вагонов
        Keep
$Body
        Пункт_расписания
                Choice from
                        Пункт_расписания.Состояние = выполняется
                Convert_rule
                        Состояние   set     закончена
        Группа_вагонов
                Choice from
```

Группа_вагонов.Номер_группы_вагонов =
Пункт_расписания.Номер_группы_вагонов ***and***
Группа_вагонов.Состояние = конец_итерации ***and***
Группа_вагонов.Положение_по_факту = 1 ***and***
Группа_вагонов.Номер_заявки = Пункт_расписания.Номер_заявки

Convert_rule

Состояние ***set*** *на_станции*
Пункт_отправления ***set*** *Группа_вагонов.Пункт_назначения*
Абсцисса_пункта_отправления ***set***
Группа_вагонов.Абсцисса_пункта_назначения
Ордината_пункта_отправления ***set***
Группа_вагонов.Ордината_пункта_назначения
Положение_по_факту ***set*** *0*

$End

Проверка выполнения отправки порожних групп

Операции проверки состояния выполнения пунктов ГОВ в разрабатываемой модели выполняются через интервал времени, задаваемый константами: *Интервал_проверки_расписания_на_опоздание*, *Интервал_проверки_погрузки_группы_вагонов* и т.д. начиная с момента начала операции по плану до момента ее фактического завершения. Результатом проверки является отнесение пункта ГОВ в одну из трех групп: зеленую, желтую или красную [стр.35].

Функция *состояние_выполнения_отправки_группы_порожних_вагонов* будет рассмотрена ниже в разделе «Функции и константы модели».

$Pattern *Проверка_выполнения_отправки_группы_порожних_вагонов:* ***operation***
$Relevant_resources
Пункт_расписания :Пункт_расписания_отправки_порожних_вагонов ***Keep*** ***Keep***
$Time *= Интервал_проверки_выполнения_отправки_группы_с_грузом*
$Body

Пункт_расписания

Choice from

Пункт_расписания.Статус_проверки = требует_проверки ***and*** *Пункт_расписания.Состояние <> закончена*

Convert_begin

Статус_проверки ***set*** *проверяется*

Convert_end

Опоздание ***set*** *Состояние_выполнения_отправки_группы_порожних_вагонов (Пункт_расписания.Состояние,Пункт_расписания.Факт, Пункт_расписания.Дата_прибытия, Пункт_расписания.Дата_отправления, Пункт_расписания.Пункт_назначения_порожних_вагонов, Пункт_расписания.Пункт_отправления_порожних_вагонов)*

Статус_проверки ***set*** *требует_проверки*

$End

Формирование группы вагонов

Формирование группы вагонов моделируется одной операцией и двумя продукционными правилами:

$Decision_point *Формирование_группы_порожних_вагонов : **some***
$Condition NoCheck
$Activities
_Создание_группы_порожних_вагонов : Создание_группы_порожних_вагонов
_Окончание_формирования_группы_порожних_вагонов : Окончание_формирования_группы_порожних_вагонов
_Присоединение_порожнего_вагона : Присоединение_порожнего_вагона
$End

1. Продукционное правило «Создание группы порожних вагонов» переводит ресурсы «Пункт расписания» и «Группа вагонов» в состояния «выполняется» и «формируется» в случае, если соответствующие ресурсы подобраны.

$Pattern *Создание_группы_порожних_вагонов : **rule***
$Relevant_resources
*Пункт_расписания : Пункт_расписания_обработки_вагонов **Keep***
*Пункт_обработки : Станции **NoChange***
*Группа_вагонов : Группы_вагонов **Keep***
$Body
Пункт_расписания
Choice from
*Пункт_расписания.Код_операции = формирование_порожних **and***
*Пункт_расписания.Дата_начала_операции <= **Time_now and***
Пункт_расписания.Состояние = не_выполняется
Convert_rule
*Состояние **set** выполняется*
*Количество_обработанных_вагонов **set** 0*
Пункт_обработки
Choice from
Пункт_обработки.Имя = Пункт_расписания.Пункт_обработки
Группа_вагонов
Choice from
*Группа_вагонов.Номер_группы_вагонов = Пункт_расписания.Номер_группы_вагонов **and***
Группа_вагонов.Состояние = не_сформирована
Convert_rule
*Номер_заявки **set** Пункт_расписания.Номер_заявки*
*Состояние **set** формируется*
*Пункт_отправления **set** Пункт_расписания.Пункт_обработки*
*Абсцисса_пункта_отправления **set** Пункт_обработки.Абсцисса*
*Ордината_пункта_отправления **set** Пункт_обработки.Ордината*

```
            Пункт_назначения            set
            Пункт_расписания.Пункт_обработки
            Абсцисса_пункта_назначения  set  Пункт_обработки.Абсцисса
            Ордината_пункта_назначения  set
      Пункт_обработки.Ордината
$End
```

2. Операция «Присоединение порожнего вагона» моделирует процесс присоединения порожнего вагона к формируемой группе. Время выполнения операции определяется константой - *Время_присоединения_вагона.* В результате выполнения операции количество вагонов в группе увеличивается на 1.

```
$Pattern    Присоединение_порожнего_вагона    :    operation
$Relevant_resources
      Группа_вагонов      :Группы_вагонов                          Keep        Keep
      Пункт_расписания    :Пункт_расписания_обработки_вагонов      NoChange    Keep
      Вагон               :Вагоны                                  Keep        Keep
      Станция             :Станции                                 NoChange    Keep
$Time = Время_присоединения_вагона
$Body
      Группа_вагонов
            Choice from
                  Группа_вагонов.Состояние = формируется
            Convert_begin
                  Состояние            set   присоединение_вагона
            Convert_end
                  Количество_вагонов   set   Группа_вагонов.Количество_вагонов + 1
                  Состояние            set   формируется
      Пункт_расписания
            Choice from
      Пункт_расписания.Количество_вагонов > Группа_вагонов.Количество_вагонов
and   Пункт_расписания.Номер_заявки = Группа_вагонов.Номер_заявки
            Convert_end
      Количество_обработанных_вагонов    set
      Пункт_расписания.Количество_обработанных_вагонов + 1
      Вагон
            Choice from
      Вагон.Загруженность = 0 and Вагон.Принадлежность =
      Пункт_расписания.Принадлежность_вагонов and    Вагон.Состояние =
      порожний_не_в_группе and Вагон.Тип_вагона = Пункт_расписания.Тип_вагонов
      and
      Вагон.Номер_группы = -1 and    Вагон.Дислокация_свободного_вагона =
      Пункт_расписания.Пункт_обработки
      Convert_begin
            Номер_группы         set   Группа_вагонов.Номер_группы_вагонов
            Состояние            set   порожний_присоединение
      Convert_end
            Состояние            set   порожний_на_станции
```

Абсцисса_местоположения ***set***
*Группа_вагонов.Количество_вагонов*Расстояние_между_вагонами*
Станция
Choice from
Станция.Имя = Пункт_расписания.Пункт_обработки
Convert_end
Количество_вагонов ***set*** *Станция.Количество_вагонов - 1*
$End

3. Данное продукционное правило срабатывает, если количество вагонов в группе стало равно количеству вагонов в заявке. В результате выполнения операции параметры «Состояние» группы вагонов и пункта расписания принимают значения «на_станции» и «Закончена».

$Pattern *Окончание_формирования_группы_порожних_вагонов : rule*
$Relevant_resources
Группа_вагонов : Группы_вагонов ***Keep***
Пункт_расписания : Пункт_расписания_обработки_вагонов ***Keep***
$Body
Группа_вагонов
Choice from
Группа_вагонов.Состояние = формируется
Convert_rule
Состояние ***set*** *на_станции*
Пункт_расписания
Choice from
Пункт_расписания.Код_операции = формирование_порожних ***and***
Пункт_расписания.Номер_заявки = Группа_вагонов.Номер_заявки
and
Пункт_расписания.Количество_вагонов =
Группа_вагонов.Количество_вагонов
Convert_rule
Состояние ***set*** *закончена*
Количество_обработанных_вагонов ***set*** *0*
$End

Функции и константы модели

Описание констант модели

Описание констант в разрабатываемой модели имеет следующий вид:

```
$Constant
    Скорость_доставки_маршрутом                 :    real  =   22.92
    Единица_продвижения_модельного_времени      :    real  =   0.1
    Интервал_проверки_расписания_на_опоздание   :    real  =   0.1
    Интервал_проверки_погрузки_группы_вагонов   :    real  =   0.1
    Интервал_проверки_разгрузки_группы_вагонов  :    real  =   0.1
    Интервал_проверки_ремонта_группы_вагонов    :    real  =   0.1
    Время_присоединения_вагона                  :    real  =   0.3
    Время_отсоединения_вагона                   :    real  =   0.3
    Время_погрузки_вагона                       :    real  =   0.78
    Время_разгрузки_вагона                      :    real  =   1
    ……..
$End
```

Отметим, что такие параметры модели как: «Скорость доставки маршрутом», «Время погрузки вагона» и т.д. могут быть заданы средним значением, полученным от подсистемы анализа и статистики, в случае среднестатистической модели или нормальным распределением в случайной модели. Выбор типа модели осуществляется пользователем при работе с модулем формирования модели на начало планового периода и может быть изменен в модуле ввода корректировок.

Описание функций модели

Функция ***Длина_пути*** предназначена для определения длины пути между пунктом назначения и пунктом отправления. Описание данной функции формируется модулем формирование модели на начало планового периода в зависимости от количества указанных пользователем станций и расстояний между ними. Описание функции может быть скорректировано в модуле ввода корректировок.

```
$Function   Длина_пути  :       real
$Type  =    table
$Parameters
        Станция_х   :       such_as     Станции.Имя
        Станция_у   :       such_as     Станции.Имя
$Body
        {Станция_х = Каргала Аксарайская_2 Потребитель_1 Потребитель_2 Станция_тех_об}
        {Станция_у}
        {Каргала}       0           300         200         200         100
        {Аксарайская_2}300          0           300         300         200
        {Потребитель_1}200          300         0           200         100
        {Потребитель_2}200          300         200         0           100
        {Станция_тех_об}100         200         100         100         0
$End
```

Следующие функции являются неизменяемыми в том смысле, что при работе оператора с подсистемой диспетчирования их описание добавляется к файлу *.fun имитационной модели в неизменном виде. Функция ***Продвижение_группы_порожних_вагонов*** осуществляет увеличение параметра «*Положение по факту*» группы вагонов при выполнении соответствующей операции.

```
$Function Продвижение_группы_порожних_вагонов   :       real
$Type = algorithmic
$Parameters
        Положение_по_факту_     :       such_as Группы_вагонов.Положение_по_факту
        Пункт_отправления_      :       such_as Группы_вагонов.Пункт_отправления
        Пункт_назначения_       :       such_as Группы_вагонов.Пункт_назначения
$Body
Calculate_if   Пункт_отправления_ = Пункт_назначения_
               Продвижение_группы_порожних_вагонов = 0
Calculate_if   Положение_по_факту_ + Скорость_доставки_маршрутом *
Единица_продвижения_модельного_времени/Длина_пути (Пункт_отправления_,
Пункт_назначения_) >= 1
               Продвижение_группы_порожних_вагонов      = 1
Calculate_if   Положение_по_факту_ + Скорость_доставки_маршрутом *
Единица_продвижения_модельного_времени/Длина_пути (Пункт_отправления_,
Пункт_назначения_) < 1
               Продвижение_группы_порожних_вагонов      =
Скорость_доставки_маршрутом *
Единица_продвижения_модельного_времени/Длина_пути(Пункт_отправления_,
Пункт_назначения_) + Положение_по_факту_
$End
```

Функция ***Продвижение_группы_вагонов_с_грузом*** аналогична предыдущей, ее описание приведено в приложении.

Функция ***Вес_груза_в_вагоне_на_погрузке*** предназначена для определения веса груза при моделировании погрузке очередного вагона. К

примеру, при перевозке 100 тонн груза в двух вагонах грузоподъемностью 66 тонн, во второй вагон требуется погрузить 34 тонны.

$Function *Вес_груза_в_вагоне_на_погрузке : **integer***
$Type = ***algorithmic***
$Parameters
*Вес_текущий : **such_as** Группы_вагонов.Вес_груза*
*Вес_требуемый : **such_as** Пункт_расписания_обработки_вагонов.Вес_груза*
*Тип_вагона : **such_as** Пункт_расписания_обработки_вагонов.Тип_вагонов*
$Body
Calculate_if *Тип_вагона = полувагон **and** (Вес_требуемый - Вес_текущий) > Грузоподъемонсть_полувагона*
Вес_груза_в_вагоне_на_погрузке = Грузоподъемонсть_полувагона
Calculate_if *Тип_вагона = полувагон **and** (Вес_требуемый - Вес_текущий) <= Грузоподъемонсть_полувагона*
Вес_груза_в_вагоне_на_погрузке = Вес_требуемый - Вес_текущий
Calculate_if *Тип_вагона = цистерна **and** (Вес_требуемый - Вес_текущий) > Грузоподъемность_цистерны*
Вес_груза_в_вагоне_на_погрузке = Грузоподъемность_цистерны
Calculate_if *Тип_вагона = цистерна **and** (Вес_требуемый - Вес_текущий) <= Грузоподъемность_цистерны*
Вес_груза_в_вагоне_на_погрузке = Вес_требуемый - Вес_текущий
$End

Следующие функции используются для распределения пунктов ГОВ по группам [стр.35] при выполнении соответствующих операций проверки:

- ***Состояние_формирования_группы_порожних_вагонов***
- ***Состояние_расформирования_группы_порожних_вагонов***
- ***Состояние_погрузки_группы_вагонов***
- ***Состояние_разгрузки_группы_вагонов***
- ***Состояние_ремонта_группы_порожних_вагонов***
- ***Состояние_выполнения_отправки_группы_порожних_вагонов***
- ***Состояние_выполнения_отправки_группы_вагонов_с_грузом***

Ниже рассмотрим функцию

С***остояние_выполнения_отправки_группы_порожних_вагонов***, описание оставшихся функций находятся на диске в приложении.

$Function *Состояние_выполнения_отправки_группы_порожних_вагонов :*
such_as *Пункт_расписания_отправки_порожних_вагонов.Опоздание*
$Type = ***algorithmic***
$Parameters
*Состояние_заявки_текущее :**such_as** Пункт_расписания_отправки_порожних_вагонов.Состояние*

*Положение_по_факту : **such_as***
Группы_вагонов.Положение_по_факту

*Дата_окончания_операции : **real***

*Дата_начала_операции : **real***

*Пункт_назначения : **such_as** Станции.Имя*

*Пункт_отправления : **such_as** Станции.Имя*

$Body

// Заявка не выполняется

***Calculate_if Time_now** <= Дата_начала_операции + (Дата_окончания_операции - Дата_начала_операции)*Зеленая_желтая_граница **and** Состояние_заявки_текущее = не_выполняется*

Состояние_выполнения_отправки_группы_порожних_вагонов = зеленая

***Calculate_if Time_now** > Дата_начала_операции + (Дата_окончания_операции - Дата_начала_операции)*Желтая_красная_граница **and** Состояние_заявки_текущее = не_выполняется*

Состояние_выполнения_отправки_группы_порожних_вагонов = красная

***Calculate_if Time_now** <= Дата_начала_операции + (Дата_окончания_операции - Дата_начала_операции)*Желтая_красная_граница **and Time_now** > Дата_начала_операции + (Дата_окончания_операции - Дата_начала_операции)*Зеленая_желтая_граница **and** Состояние_заявки_текущее = не_выполняется*

Состояние_выполнения_отправки_группы_порожних_вагонов = желтая

// Заявка выполняется

***Calculate_if** Состояние_заявки_текущее = выполняется **and Time_now** + Длина_пути (Пункт_назначения, Пункт_отправления)*(1.0-Положение_по_факту) / Скорость_доставки_маршрутом <= Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)*Зеленая_желтая_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = зеленая

***Calculate_if** Состояние_заявки_текущее = выполняется **and Time_now** + Длина_пути(Пункт_назначения, Пункт_отправления)*(1.0-Положение_по_факту)/ Скорость_доставки_маршрутом > Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)* Желтая_красная_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = красная

Calculate_if *Состояние_заявки_текущее = выполняется* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1.0-Положение_по_факту) /Скорость_доставки_маршрутом <= Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)* Желтая_красная_граница* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1- Положение_по_факту)/Скорость_доставки_маршрутом > Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)*Зеленая_желтая_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = желтая

// Выполнение заявки закончено

Calculate_if *Состояние_заявки_текущее = закончена* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1.0- Положение_по_факту)/Скорость_доставки_маршрутом <= Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)* Зеленая_желтая_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = зеленая

Calculate_if *Состояние_заявки_текущее = закончена* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1.0- Положение_по_факту)/Скорость_доставки_маршрутом > Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)* Желтая_красная_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = красная

Calculate_if *Состояние_заявки_текущее = закончена* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1.0- Положение_по_факту)/Скорость_доставки_маршрутом <= Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)* Желтая_красная_граница* ***and Time_now*** *+ Длина_пути(Пункт_назначения, Пункт_отправления)*(1- Положение_по_факту)/Скорость_доставки_маршрутом > Дата_окончания_операции + (Дата_окончания_операции - Дата_начала_операции)*Зеленая_желтая_граница*

Состояние_выполнения_отправки_группы_порожних_вагонов = желтая

$End

Отметим, что данная функция использует следующие параметры:

Зеленая_желтая_граница : real

Желтая_красная_граница : real

Данные параметры задают правило, по которому заявка относится к одной из групп: красной, желтой или зеленой. К примеру, пусть *Зеленая_желтая_граница = 0,05* и *Желтая_красная_граница = 0,2*. Таким

образом, если задержка выполнения операции не превышает 5% от планового времени выполнения операции, цвет группы – «зеленый». Если задержка выполнения операции находится в пределах от 5% до 20% процентов выполнения операции, цвет группы – «желтый». Если задержка превышает 20% от планового времени выполнения операции, цвет группы - «красный».

Пример использования разработанной подсистемы

Рассмотрим процесс работы подсистемы диспетчирования на следующем примере.

Исходные данные

Пусть имеется карта железных дорог с пятью станциями, доступными на начало планового периода.

- Каргала (тип станции – поставщик)
- Аксарайская-2 (тип станции – поставщик)
- Потребитель-1 (тип станции – потребитель)
- Потребитель-2 (тип станции – потребитель)
- Станция технического обслуживания (тип станции – станция ГПТ)

Расстояния между станциями и количество вагонов на станциях показано на рисунке 4.

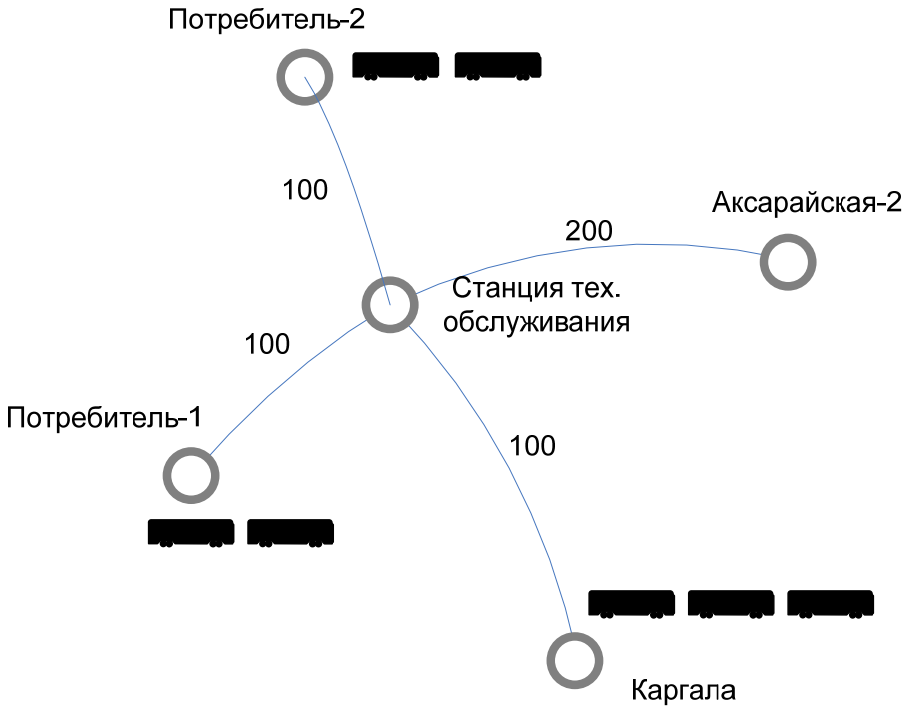

Рис. 4

Для простоты, в рамках данного примера, будем считать, что осуществляется доставка только гранулированной серы. Тип вагонов – полувагон. Все вагоны на начало планового периода – порожние.

Имеется график отправки вагонов на начало планового периода, сформированный подсистемой формирования месячных и декадных планов:

График отправки вагонов с грузом:

№ Группы вагонов	Пункт отправления	Пункт назначения	Вес груза (т)	Тип груза	Количество вагонов	Скорость доставки	Время отправления
1	Каргала	Потребитель-1	150	гранулир.	3	обычная	6
4	Аксарайская-2	Потребитель-2	200	гранулир.	4	обычная	27

График отправки порожних вагонов:

№ Группы вагонов	Пункт отправления	Пункт назначения	Тип вагона	Принадлежность	Количество вагонов	Скорость доставки	Время отправления
2	Потребитель-1	Аксарайская	полувагон	свой	2	обычная	3
3	Потребитель-2	Аксарайская	полувагон	свой	2	обычная	3

График обработки вагонов:

№ Группы вагонов	Пункт обработки	Наименование операции	Тип вагона	Принадлеж-ность	Количество вагонов	Время начала	Время окончания
1	Каргала	формирование	полувагон	свой	3	1	4
1	Каргала	погрузка	полувагон	свой	3	4	6
2	Потребитель-1	формирование	полувагон	свой	2	1	3
3	Потребитель-2	формирование	полувагон	свой	2	1	3
1	Потребитель-1	разгрузка	полувагон	свой	3	20	24
2	Аксарайская-2	расформирование	полувагон	свой	2	19	21
3	Аксарайская-2	расформирование	полувагон	свой	2	19	21
4	Аксарайская-2	формирование	полувагон	свой	4	21	24
4	Аксарайская-2	погрузка	полувагон	свой	4	24	27
4	Потребитель-2	разгрузка	полувагон	свой	4	41	45

Будем считать, что диспетчер строит прогноз в момент времени Тмод = 12 единиц модельного времени. Одна единица модельного времени соответствует одному часу реального времени. Окончание планового периода Тпп = 45 единиц модельного времени. На момент времени Тмод диспетчеру известны следующие факты:

1. По данным РЖД группа вагонов №3 задерживается. Положение по факту = 0,1 (30 км от станции Потребитель_2). Причина задержки – неисправное состояние одного из порожних вагонов (вина ОАО ГПТ).
2. На станцию тех. обслуживания поступили новые вагоны в количестве 2 шт.

Решение задачи диспетчирования

1. Формирование имитационной модели на начало планового периода.

Данная задача решается модулем формирования имитационной модели на начало планового периода [стр. 31]. В рамках дипломного проекта осуществляется формирование файлов вручную, пользователем системы. Данная имитационная модель находится на диске в приложении к РПЗ.

2. Формирование имитационной модели на время составления прогноза.

Далее следует формирование файлов имитационной модели, соответствующей началу планового периода. Будем считать, что за исключением факта задержки группы №3 и поступления новых вагонов на станцию тех. обслуживания, других отклонений от графика нет. Данные отклонения учитываются путем ввода корректировок в модуле ввода корректировок. В рамках дипломного проекта ввод корректировок в файлы модели для Тмод = 12 осуществляется вручную пользователем системы. Имитационная модель с учетом корректировок находится на диске в приложении к РПЗ.

3. Построение прогноза выполнения графика отправки вагонов на конец планового периода.

Осуществим нормативный прогон имитационной модели. Результаты моделирования представлены ниже:

Группа 1: Опоздания_по_группам	
1 Отправка_п_группы_1	зеленая
2 Отправка_п_группы_2	красная
3 Отправка_груза_1	зеленая
4 Отправка_груза_2	желтая
5 Погрузка_1	зеленая
6 Погрузка_2	красная
7 Разгрузка_1	зеленая
8 Разгрузка_2	красная
Группа 2: Состояние_выполнения	
1 Сост_отп_п_гр_1	закончена
2 Сост_отп_п_гр_2	закончена
3 Сост_отпр_гр_1	закончена
4 Сост_отпр_гр_2	закончена
5 Сост_погр_1	закончена
6 Сост_погр_2	закончена
7 Сост_разгр_1	закончена
8 Сост_разгр_2	не_выполняется

Рис. 5 Результаты имитационного эксперимента.

Очевидно, что срыв выполнения графика вагонов связан с отставанием от графика третьей группы вагонов.

4. Формирование имитационной модели с учетом корректировок, предлагаемых диспетчером.

В данном случае возможны два варианта решения задачи диспетчирования:

- Увеличение скорости движения группы №3
- Формирование на станции тех. обслуживания новой группы порожних вагонов и отправки их в пункт Потребитель_2.

Рассмотрим второй вариант.

Добавим в модель следующие пункты ГОВ:

В график обработки вагонов:

№ Группы вагонов	Пункт обработки	Наименование операции	Тип вагона	Принадлеж-ность	Количество вагонов	Время начала	Время окончания
5	Станция тех. обслуживания	формирование	полувагон	свой	2	12	14
5	Аксарайская	расформирование	полувагон	свой	2	19	21

В график отправки порожних вагонов:

№ Группы вагонов	Пункт отправления	Пункт назначения	Тип вагона	Принадлежн ость	Количество вагонов	Скорость доставки	Время отправления
5	Станция тех. обслуживания	Аксарайская-2	полувагон	свой	2	обычная	14

Также изменим пункт назначения в графике отправки порожних вагонов для группы №3 с Аксарайская-2 на “Станция технического обслуживания”, так как доставка данных вагонов до поставщика была заменена поставкой новых вагонов со станции технического обслуживания. Имитационная модель с учетом корректировок, предложенных диспетчером, находится на диске в приложении к РПЗ.

5. Построение прогноза выполнения графика отправки вагонов на конец планового периода с учетом корректировок, предложенных диспетчером.

Осуществим нормативный прогон имитационной модели. Результаты моделирования представлены ниже:

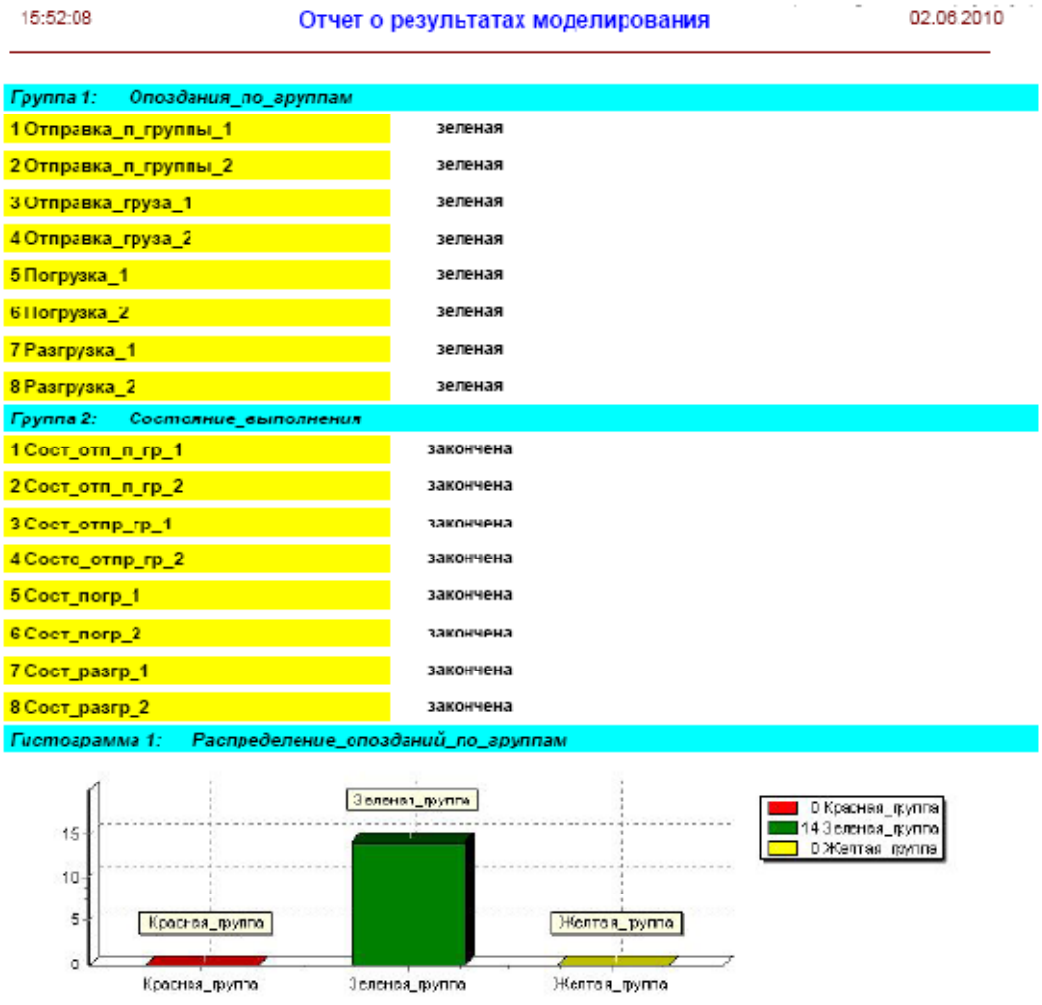

Рис. 6 Результаты имитационного эксперимента с учетом корректировок, сделанных диспетчером.

Таким образом, из рисунка 6 видно, что все пункты расписания выполнены на конец планового периода без опозданий.

Организационно-экономическая часть

В данной части дипломной работы производится расчет затрат на разработку «Подсистемы диспетчирования перевозок серы на основе РДО». Организация и планирование процесса разработки программного продукта при традиционном методе планирования [3] предполагает следующие этапы:

- Формирование состава выполняемых работ и группировка их по стадиям разработки
- Расчет трудоемкости выполнения работ
- Определение продолжительности выполнения отдельных этапов разработки
- Построение календарного графика выполнения разработки
- Контроль выполнения календарного графика

Разработка информационных систем является сложным процессом. Отметим следующие факты, характеризующие процесс разработки рассматриваемой подсистемы:

- Работа по созданию подсистемы диспетчирования должна быть проведена в ходе работ по созданию АСУ ПС, составной частью которой она является.
- Предполагается командная работа нескольких разработчиков, являющихся экспертами в разных областях знаний.
- Итерационный характер разработки. В ходе работ требования к подсистеме и АСУ ПС могут дополняться и уточняться. Могут возникнуть задачи, наличие которых не удалось выявить до начала технического и рабочего этапов проектирования.
- Иерархичность подсистемы. Разрабатываемая подсистема состоит из нескольких взаимосвязанных модулей.

Принимая во внимание вышеперечисленные утверждения, можно сделать вывод о том, что данная оценка стоимости разработки является предварительной и может быть скорректирована по ходу работ.

Расчет действителен на второй квартал 2010 года (цены на оборудование, расходные материалы, уровень заработной платы исполнителей и т.д.).

Формирование состава выполняемых работ и группировка их по стадиям разработки

Перечень стадий по разработке подсистемы соответствует ГОСТ 34.601-90. Разбивка работ по стадиям разработки приведена в таблице 1.

Таблица 1 – Укрупненный состав работ по стадиям разработки программного продукта.

Стадия разработки программного продукта	**Состав выполняемых работ**
Техническое задание	Постановка задач, выбор критериев эффективности. Разработка технико-экономического обоснования разработки. Определение состава пакета прикладных программ, состава и структуры информационной базы. Выбор языков программирования. Предварительный выбор методов выполнения работы. Разработка календарного плана выполнения работ.
Эскизный проект	Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритмов реализации решения задач. Разработка пояснительной записки. Консультации с разработчиками постановки задач. Согласование и утверждение эскизного проекта.
Технический проект	Разработка алгоритмов решения задач. Разработка пояснительной записки. Согласование и утверждение технического проекта. Разработка структуры программы. Разработка программной документации и передача ее для включения в технический проект. Уточнение структуры, анализ и определение формы представления входных и выходных данных. Выбор конфигурации технических средств.

Рабочий проект	Комплексная отладка задач и сдача в опытную эксплуатацию. Разработка проектной документации. Программирование и отладка программ. Описание контрольного примера. Разработка программной документации. Разработка, согласование программы и методики испытаний. Предварительное проведение всех видов испытаний.
Внедрение	Подготовка и передача программной документации для сопровождения с оформлением соответствующего акта. Передача программной продукции в фонд алгоритмов и программ. Проверка алгоритмов и программ решения задач, корректировка документации после опытной эксплуатации программного продукта.

Оценка трудоемкости разработки программного обеспечения

По степени новизны разрабатываемый проект относится к категории Б – разработка программной продукции не имеющей аналогов, в том числе разработка пакетов прикладных программ.

По степени сложности алгоритма функционирования проект относится к 1 группе – программная продукция, реализующая оптимизационные и моделирующие алгоритмы.

По **виду представления исходной информации и способу её контроля** разрабатываемая подсистема диспетчирования относится к *группе 11*, так как исходная информация хранится в информационной базе в виде связанных таблиц, имеющих различную структуру, а также часть входной информация может быть получена в результате диалога с пользователем. Таким образом, необходимо осуществлять контроль входной информации представленной в различной форме, учитывая её взаимовлияние. По **виду представления выходной информации** подсистема диспетчирования относиться к *группе 21* – требуется вывод на печать документов многоуровневой структуры.

Трудоемкость разработки системы $\tau_{ПП}$ может быть определена как

сумма величин трудоемкости выполнения отдельных стадий разработки из выражения:

$\tau_{ПП} = \tau_{ТЗ} + \tau_{ЭП} + \tau_{ТП} + \tau_{РП} + \tau_{В}$, где

$\tau_{ТЗ}$ – трудоёмкость разработки технического задания на систему;

$\tau_{ЭП}$ – трудоёмкость разработки эскизного проекта системы;

$\tau_{ТП}$ – трудоёмкость разработки технического проекта системы;

$\tau_{РП}$ – трудоёмкость разработки рабочего проекта системы;

$\tau_{В}$ – трудоёмкость внедрения разработанной системы.

Трудоемкость разработки технического задания на систему

$\tau_{тв} = T^{з}_{рз} + T^{з}_{рп}$, где

$T^{з}_{рз}$ - затраты времени разработчика постановки задач на разработку ТЗ;

$T^{з}_{рп}$ - затраты времени разработчика программного обеспечения на разработку ТЗ.

$T^{з}_{рз} = t_{з} \cdot K^{з}_{рз}$;

$T^{з}_{рп} = t_{з} \cdot K^{з}_{рп}$, где

$t_{з}$ - норма времени на разработку ТЗ на программный продукт в зависимости от функционального назначения и степени новизны разрабатываемого продукта;

$K^{з}_{рз}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии ТЗ;

$K^{з}_{рп}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ.

Для разрабатываемой подсистемы диспетчирования:

$t_{з} = 57$ человеко-дней, т.к. система относится к группе новизны Б и решает задачи диспетчеризации (Таблица 2 [3]):

$K^{э}_{рз}$ = 1,0; $K^{э}_{рп}$ = 0,35, так как разработчик постановки задач работает над постановкой задач и разработкой самостоятельно, а разработчик ПО работает над разработкой ТЗ, консультируясь с разработчиком постановки задач.

$\tau_{тв} = 57*(1,0 + 0,35) = 77$ человеко-дней.

Трудоемкость разработки эскизного проекта системы

$\tau_{эп} = T^{э}_{рз} + T^{э}_{рп}$, где

$T^{э}_{рз}$ - затраты времени разработчика постановки задач на разработку ЭП;

$T^{э}_{рп}$ -затраты времени разработчика программного обеспечения на разработку ЭП.

$T^{э}_{рз} = t_{э} \cdot K^{э}_{рз}$;

$T^{э}_{рп} = t_{э} \cdot K^{э}_{рп}$, где

$t_{э}$ - норма времени на разработку ЭП на программный продукт в зависимости от функционального назначения и степени новизны разрабатываемого продукта;

$K^{э}_{рз}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии ЭП;

$K^{э}_{рп}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ЭП.

Для разрабатываемой подсистемы диспетчирования:

$t_{э}$ = 117 человеко-дней, $K^{э}_{рз}$ = 1,0; $K^{э}_{рп}$ = 0,3. Критерии выбора нормы и коэффициентов те же, что и для $t_{з}$ (Таблица 3 [3]);

$\tau_{эп} = 117*(1,0 + 0,3) = 153$ человеко-дня.

Трудоемкость разработки технического проекта

$\tau_{тп} = (t^{т}_{рз} + t^{т}_{рп}) \cdot K_{в} \cdot K_{р}$, где

$t^{т}_{рз}$, $t^{т}_{рп}$ - нормы времени, затрачиваемого на разработку ТП разработчиком постановки задач и разработчиком программного обеспечения соответственно;

$K_{в}$ - коэффициент учёта вида используемой информации;

$K_{р}$ - коэффициент учёта режима обработки информации.

$$K_{в} = \frac{K_{п} n_{п} + K_{нс} n_{нс} + K_{б} n_{б}}{n_{п} + n_{нс} + n_{б}}$$, где

$K_{п}, K_{нс}, K_{б}$ - значения коэффициентов учёта вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно;

$n_{п}, n_{нс}, n_{б}$ - количество наборов переменной, нормативно-справочной информации и баз данных соответственно.

Для разрабатываемой подсистемы диспетчирования:

$t^{т}_{рз} = 131$ человеко-дней, $t^{т}_{рп} = 38$ человеко-дней (Таблица 4 [ссылка]).

Система решает задачи диспетчирования и имеет 13 форм входной информации:

- информация из БД от РЖД о текущем положении парка вагонов
- информация о графике движения вагонов от подсистемы формирования ГОВ
- корректировки диспетчера
- ИМ модель на языке РДО – 9 текстовых файлов различного формата
- обновляемая статистическая информация для моделирования от подсистемы сбора статистики;

и 3 формы выходной информации:

- отчёты о прогнозе выполнения ГОВ, формируемые модулем формирования отчетов

- вспомогательная выходная информация: анимация модели, графики
- документация на систему.

$K_р$ = 1,45; группа новизны - Б, а режим обработки информации – в реальном времени. (Таблица 17 [3]):

Для группы новизны Б по таблице 18 [3]:

$K_п = 1{,}2,\ n_п = 11;$

$K_{нс} = 1{,}08,\ n_{нс} = 1;$

$K_б = 3{,}12,\ n_б = 1.$

$$K_в = \frac{1{,}2 \cdot 11 + 1{,}08 \cdot 1 + 3{,}12 \cdot 1}{11 + 1 + 1} = 1{,}34$$

$\tau_{тт} = (131 + 38)*1{,}45*1{,}34 = 329$ человеко-дней.

Трудоемкость разработки рабочего проекта

$\tau_{рп} = K_к \cdot K_р \cdot K_я \cdot K_з \cdot K_{иа} \cdot (t^р_{рз} + t^р_{рп})$, где

$t^р_{рз}, t^р_{рп}$ - норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком программного обеспечения соответственно;

$K_к$ - коэффициент учета сложности контроля информации;

$K_я$ - коэффициент учета уровня используемого алгоритмического языка программирования;

$K_з$ - коэффициент учета степени использования готовых программных модулей;

$K_{иа}$ - коэффициент учета вида используемой информации и сложности алгоритма программного продукта.

$$K_{иа} = \frac{K'_п n_п + K'_{нс} n_{нс} + K'_б n_б}{n_п + n_{нс} + n_б}$$, где

$K'_П, K'_{НС}, K'_Б$ - значения коэффициентов учета сложности алгоритма программного продукта и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

Для разрабатываемой подсистемы диспетчирования:

$K_к$ = 1,16; степень сложности контроля входной информации - 11, степень сложности контроля выходной информации - 21 (Таблица 19 [3]);

$K_р$ = 1,52; группа новизны - Б, а режим обработки информации – в реальном времени (Таблица 17 [3]);

$K_я$ = 1; C++ - Алгоритмический язык высокого уровня, язык РДО – интерпретируемый язык (Таблица 20 [3]);

$K_з$ = 0,8; Используются шаблоны работы с БД (Таблица 21 [3]);

Для группы новизны Б по таблице 22 [3]:

$K'_п = 1{,}62,\ n_п = 11;$

$K'_{нс} = 0{,}97,\ n_{нс} = 1;$

$K'_б = 0{,}6,\ n_б = 1.$ Группа сложности алгоритма работы с БД – 3 (алгоритм, реализующий стандартные методы).

$t^р_{рз} = 43$ человеко-дня, $t^р_{рп} = 231$ человеко-день. (Таблица 23 [3]);

$$K_{иа} = \frac{1{,}62 \cdot 11 + 0{,}97 \cdot 1 + 0{,}6 \cdot 1}{11 + 1 + 1} = 1{,}49$$

$\tau_{рп} = 1{,}16*1{,}52*1*0{,}8*1{,}49*(43 + 231) = 576$ человеко-дней.

Трудоемкость внедрения программного продукта

$$\tau_в = (t^в_{рз} + t^в_{рп}) \cdot K_к \cdot K_р \cdot K_з$$

$t^в_{рз}, t^в_{рп}$ - норма времени, затрачиваемого разработчиком постановки задач и разработчиком программного обеспечения соответственно на выполнение процедур внедрения программного продукта.

Для разрабатываемой системы информационной поддержки:

$K_к$ = 1,16; степень сложности контроля входной информации - 11, степень сложности контроля выходной информации - 21 (Таблица 19 [3]);

$K_р$ = 1,39; группа новизны - Б, а режим обработки информации – в реальном времени (Таблица 17 [3]);

$K_з$ = 0,6; РДО и СУБД представляют собой готовые программные решения (Таблица 21 [3]);

$t^в_{рз} = 36$ человеко-дней, $t^в_{рп} = 50$ человеко-дней (Таблица 40 [3]).

$\tau_в = (36 + 50)*1,16*1,39*0,6 = 84$ человеко-дня.

Расчет суммарной трудоемкости разработки

Поскольку предполагается, что работа будет вестись тесно-взаимодействующей командой разработчиков, предлагается объединить во времени выполнение технического и рабочего этапов. Это позволить снизить время выполнения технического этапа на 15% [4, стр.10]

Таким образом, суммарная трудоемкость разработки и внедрения подсистемы диспетчирования составляет:

$\tau_{сп} = \tau_{тв} + \tau_{эп} + \tau_{тп} + \tau_{рп} + \tau_в = 77 + 153 + 329*0,85 + 576 + 84 = 1170$ человеко-дней.

В таблице 2 приведена разбивка трудоемкости по этапам разработки:

Таблица 2. Распределение трудоемкости по этапам разработки.

Стадия	Трудоемкость, дни
Техническое задание	77
Эскизный проект	153
Технический проект	329
Рабочий проект	576
Внедрение	84
Итого:	**1 219**

Оценка численности разработчиков программного обеспечения

Продолжительность выполнения работ по всем этапам рассчитывается по формуле:

$$T_i = \frac{\tau_i + Q}{n_i},$$

где τ_i — трудоёмкость i-ой работы, (человеко-дни);

Q — трудоёмкость дополнительных работ, выполняемых исполнителем, (человеко-дни);

n_i — количество исполнителей, выполняющих i-ую работу.

В ходе дальнейших расчетов будем полагать, что дополнительные работы по этапам отсутствуют. Будем также полагать, фирма-разработчик имеет достаточное количество людских ресурсов.

$T_{тз}$ = 77/2 = 34 дня;

Для написания технического задания привлечены системный архитектор (100% занятость), системный аналитик (100% занятость).

$T_{эп}$ = 153/3,5 = 44 дня;

Для работы над эскизным проектом привлечены системный архитектор (100% занятость), системный аналитик (100% занятость), 1 разработчик со 100% занятостью и 1 разработчик с 50% занятостью.

$T_{тп}$ = 280/4 = 70 дней;

Для работы над техническим проектом привлечены системный архитектор (50% занятость), системный аналитик (100% занятость), 2 разработчика со 100% занятостью и 1 разработчик с 50% занятостью.

$T_{рп}$ =576/5 = 116 дней;

Для работы над рабочим проектом привлечены системный архитектор (100% занятость), системный аналитик (100% занятость), 3 разработчика (100% занятость).

$T_{внедр}$ = 84/2,5 = 34 дня;

Для работы над внедрением проекта привлечены системный архитектор (50% занятость), системный аналитик (100% занятость), 1 разработчик (100% занятость).

Таблица 3. Количество работников на этапах разработки

№ Стадии разработки	Стадия	Квалификационные требования	Состав разработчиков	Число участников	Средняя занятость в проекте, %	Продолжитель-ность этапа, дни
1	Техническое задание	Требуется наивысшая квалификация и опыт работы на всех стадиях разработки системы	Системный архитектор	1	100%	34
			Системный аналитик	1	100%	
2	Эскизный проект	Уровень квалификации позволяющий разрабатывать и оценивать альтернативные варианты решения задач	Системный архитектор	1	100%	44
			Системный аналитик	1	100%	
			Разработчик	2	75%	
3	Технический проект	Знание и опыт работы с современными средствами разработки ПО	Системный архитектор	1	50%	70
			Системный аналитик	1	100%	
			Разработчик	3	83%	
4	Рабочий проект	Опыт кодирования алгоритмов в современных средах программирования	Системный архитектор	1	100%	116
			Системный аналитик	1	100%	
			Разработчик	3	100%	
5	Внедрение	Уровень квалификации должен позволять производить проверка и корректировка созданного ПО.	Системный архитектор	1	50%	34
			Системный аналитик	1	100%	
			Разработчик	1	100%	

Оценка трудозатрат разработчиков

В соответствии с таблицей 3 получаем следующее распределение трудозатрат по каждому исполнителю:

Таблица 4. Распределение трудозатрат по должностям.

Должность участника	Сумма часов
Разработчик	626
Системный аналитик	298
Системный архитектор (для всех участников проекта)	246
Всего:	1170

Оценка стоимости разработки и внедрения подсистемы диспетчирования

Расчет основной заработной платы

Расчет проводится по формуле:

Сзо = Зi/d*ti, где

Зi - месячный оклад i-го исполнителя, руб.

d - среднее количество рабочих дней в месяце, d = 21,8 дней.

Ti - трудоемкость работ, выполняемых i-м исполнителем, человеко-дни.

Таблица 5. Расчет затрат на оплату труда

Должность участника	Сумма часов	Оклад, руб./мес.	Затраты, руб.
Разработчик	626	35 000	1 005 046
Системный аналитик	298	60 000	820 183
Системный архитектор (для всех участников проекта)	246	80 000	902 752
Всего:			2 727 982

Расчет дополнительной заработной платы

В данной статье также учитываются выплаты непосредственным исполнителям за время, не проработанное на производстве, в том числе: оплата очередных отпусков, компенсация за недоиспользованный отпуск и др. Дополнительная заработная плата рассчитывается по формуле:

$C_{зд} = C_{зо} \cdot A_{д}$,

где $A_{д}$ - коэффициент отчислений на дополнительную заработную плату.

$A_{д} = 0.2$

$C_{зд} = 2\ 727\ 982 \cdot 0{,}2 = 545\ 596$ руб.

Единый социальный налог и процент страхования от несчастного случая

В разделе учитываются отчисления в бюджет социального страхования по установленному законодательством тарифу от суммы основной и дополнительной заработной платы. Расчет производится следующим образом:

$C_{сс} = (A_{есн} + A_{снс})\ (C_{зо}+C_{зд})$,

где

$A_{есн}$ – единый социальный налог.

$A_{есн} = 0{,}26$,

$A_{снс} = 0{,}004$

$C_{сс} = (0{,}26+0{,}004)\cdot(2\ 727\ 982 + 545\ 596) = 864\ 225$ руб.

$C_{сс} = 864\ 225$ руб.

Амортизационные отчисления

В данном разделе рассчитываются амортизационные отчисления на оборудование и ПО, необходимое для осуществления процесса разработки.

Годовой фонд времени работы оборудования с учетом простоя на ремонт и профилактику составляет 1 925 часов. Ставка амортизации составляет 20%.

Таблица 6. Амортизационные отчисления

Наименование актива	Количество	Балансовая стоимость	Время работы, часов	Амортизационные отчисления, руб
ПЭВМ эксперта	1	30 000	246	767
ПЭВМ аналитика	1	30 000	298	929
ПЭВМ разработчика	3	30 000	210	1 964
Сервер для совместной работы	1	50 000	1 170	6 078
Сетевой Принтер	1	10 000	1 170	1 216
ОС Windows 7	5	10 000	234	1 216
Microsoft Office Proffessional 2007	5	10 000	234	1 216
Microsoft Visual Studio 2008	5	10 000	234	1 216
Итого:				14 599

Накладные расходы

В данную статью входят другие затраты, входящие в состав себестоимости продукции (работ, услуг), но не относящиеся к ранее перечисленным элементам затрат.

$C_n = A_н \cdot C_{зо}$,

где:

$A_н$ - коэффициент накладных расходов. Принимаем: $A_н = 2{,}0$

$C_n = 2 \cdot 2\,727\,982 = 5\,455\,964$ руб.

В статье «Накладные расходы» будем также учитывать стоимость системы имитационного моделирования РДО, входящую в стоимость подсистемы диспетчирования, которую примем равной: 100 000 руб.

Таким образом: $C_n = 5\,555\,964$ руб.

Общие затраты на создание подсистемы

Таблица 7. Затраты на разработку подсистемы диспетчирования

№ Статьи	Наименование статьи	Сметная стоимость, руб.
1	Затраты на основную заработную плату	2 727 982
2	Затраты на дополнительную заработную плату	545 596
3	Единый социальный налог и процент страхования от несчастного случая	864 225
4	Амортизационные отчисления	14 599
5	Накладные расходы	5 555 964
Итого:		9 708 366

Определение цены разработки и внедрения подсистемы диспетчирования

Цена разработки и внедрения определяется следующим образом:

$Ц = K \times C + Пр$

где C - затраты на разработку ПП.

K - коэффициент учета затрат на изготовление опытного образца ПП как продукции производственно-технического назначения (K = 1,1)

$П_Р$ - нормативная прибыль, рассчитываемая по формуле:

$П_Р = C \cdot \rho_Н / 100$,

где $\rho_н$ - норматив рентабельности, 30 %;

$Пр$ = 9 708 366 · 30 / 100 = 2 912 510 руб.

$Ц$ = 1,1 · 9 708 366 + 2 912 510 = 13 592 000 руб.

Цена создания разрабатываемой программы: **13 592 000** рублей.

Выводы по организационно-экономической части

В рамках организационно-экономической части были произведены расчеты для определения затрат на создания подсистемы диспетчирования на основе РДО. Исследованы и рассчитаны следующие статьи затрат: материальные затраты; основная заработная плата исполнителей - дополнительная заработная плата исполнителей; отчисления на социальное страхование; амортизационные отчисления; накладные расходы.

В результате расчетов было получено общее время выполнения проекта, которое составило 298 рабочих дней, получены данные по суммарным затратам на создание подсистемы для моделирования в среде РДО с использованием процессного подхода, которые составили 9 708 366 рублей.

В результате расчетов была определена цена создания данной подсистемы, которая составила **13 592 000** рублей.

Требования к безопасности при работе с подсистемой диспетчирования

Введение

Разрабатываемая подсистема диспетчирования, входит в состав АСУ ПС (Автоматизированная система управления перевозками серы). АСУ ПС будет являться коммерческим продуктом, разработанным для нужд ОАО ГПТ. Подсистема диспетчирования предполагает работу оператора ПЭВМ в течение всей смены.

Работа с ПЭВМ регламентируется требованиями санитарно-эпидемиологических правил и нормативами «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» от 30 июня 2003 года (далее СанПин 2.2.2/2.4.1340-03). Каждый тип требований подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке.

- Выделим требования к организации условий труда диспетчера
- Выделим вредные и опасные производственные факторы
- Произведем расчеты вредных и опасных факторов.

Требования к организации работы

В соответствии с СанПиН 2.2.2/2.4.1340-03 требования к ПЭВМ и организации работы можно разделить на следующие группы:

- Требования к ПЭВМ (В рамках данной работы рассматриваться не будут)
- Требования к помещениям для работы с ПЭВМ
- Требования к микроклимату, содержанию аэрионов и вредных химических веществ в воздухе на рабочих местах, оборудованных ПЭВМ

- Требования к уровням шума и вибрации на рабочих местах, оборудованных ПЭВМ
- Требования к освещению на рабочих местах, оборудованных ПЭВМ
- Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ
- Требования к организации рабочих мест пользователей ПЭВМ
- Требования к организации рабочих мест пользователей ПЭВМ
- Требования к организации рабочих мест пользователей ПЭВМ
- Требования к пожаробезопасности на рабочих местах, оборудованных ПЭВМ
- Требования к электробезопасности на рабочих местах, оборудованных ПЭВМ

Требования к помещениям для работы с ПЭВМ

Помещения для эксплуатации ПЭВМ должны иметь естественное и искусственное освещение. Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электроннолучевой трубки (ЭЛТ) должна составлять не менее 6 $м^2$, с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 $м^2$.

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно-отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5.

Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения.

Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

Требования к микроклимату, содержанию аэроинов и вредных химических веществ в воздухе на рабочих местах, оборудованных ПЭВМ.

В производственных помещениях, в которых работа с использованием ПЭВМ является основной (диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.) и связана с нервно-эмоциональным напряжением, должны обеспечиваться оптимальные параметры микроклимата для категории работ 1а и 1б в соответствии с действующими санитарно-эпидемиологическими нормативами микроклимата производственных помещений (Таблица 1).

Таблица 8. Оптимальные величины показателей микроклимата производственных помещений (СанПиН 2.2.4.548-96).

Период года	Категория работ по уровням энергозатрат, Вт	Температура воздуха, С°	Температура поверхностей, С°	Относитель-ная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Iа (до 139)	22 – 24	21 – 25	60 – 40	0,1
	Iб (140 - 174)	21 – 23	20 – 24	60 – 40	0,1
	IIа (175 - 232)	19 – 21	18 – 22	60 – 40	0,2
	IIб (233 - 290)	17 – 19	16 – 20	60 – 40	0,2
	III (более 290)	16 - 18	15 – 19	60 – 40	0,3
Тёплый	Iа (до 139)	23 –25	22 – 26	60 – 40	0,1
	Iб (140 - 174)	22 –24	21 – 25	60 – 40	0,1
	IIа (175 - 232)	20 – 22	19 – 23	60 – 40	0,2
	IIб (233 - 290)	19 – 21	18 – 22	60 – 40	0,2
	III (более 290)	18 - 20	17 – 21	60 – 40	0,3

На других рабочих местах следует поддерживать параметры микроклимата на допустимом уровне, соответствующем требованиям указанным выше нормативов. В помещении, оборудованных ПЭВМ, должна проводиться ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПЭВМ.

Уровни положительных и отрицательных аэроионов в воздухе помещений, где расположены ПЭВМ, должны соответствовать действующим санитарно-эпидемиологическим нормативам (Таблица 2)

Таблица 9. Нормативы по аэрионам в воздухе помещений СанПиН 2.2.4.1294-03

Нормируемые показатели	Концентрация	Концентрация	Коэффициент униполярности Y
	n+ ион/см^3)	n- (ион/см^3)	
Минимально допустимые	n+ >= 400	n- >= 400	0,4 =< Y =< 1,0
Максимально допустимые	n+ < 50000	n- < 50000	

Содержание вредных химических веществ в производственных помещениях, в которых работа с использованием ПЭВМ является основной

(диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.), не должно превышать предельно допустимых концентраций загрязняющих веществ в атмосферном воздухе населенных мест в соответствии с действующими гигиеническими нормативами ГН 2.1.6.1338-03.

Таблица 10. Нормативы по содержанию вредных химических веществ в воздухе помещений по ГН 2.1.6.1338-03.

N	Наименование вещества	N CAS	Формула	Предельно допустимые концентрации мг/м3		Лимитирующий показатель	Класс опасности
				Максимальная разовая	Среднесуточная		
1	Азота диоксид	10102-44-0	NO2	0,085	0,04	рефл.-рез.	2
2	Азот (II) оксид	10102-43-9	NO	0,4	0,06	рефл.	3
8	Углерод оксид	630-08-0	CO	5	3	рез.	4
9	Свинец и его неорганические соединения	7439-92-1	Pb	0,001	0,0003	рез.	1

Для обеспечения соблюдения требований к микроклимату в помещениях, оборудованных ПЭВМ рекомендуется:

- устраивать системы вентиляции
- использовать фильтры очистки воздуха для удаления из воздуха вредных веществ перед его подачей на рабочие места.

Требования к уровням шума и вибрации на рабочих местах, оборудованных ПЭВМ

В производственных помещениях при выполнении основных или вспомогательных работ с использованием ПЭВМ уровни шума на рабочих местах не должны превышать предельно допустимых значений, установленных для данных видов работ в соответствии с действующими санитарно-эпидемиологическими нормативами.

При выполнении работ с использованием ПЭВМ в производственных помещениях уровень вибрации не должен превышать допустимых значений вибрации для рабочих мест (категория 3, тип «в») в соответствии с действующими санитарно-эпидемиологическими нормативами.

Снизить уровень шума в помещениях с ВДТ и ПЭВМ можно с помощью акустической обработки потолка звукопоглощающими материалами с максимальными коэффициентами звукопоглощения в области частот 63 - 8000 Гц.

Дополнительным звукопоглощением служат однотонные занавеси из плотной ткани, гармонирующие с окраской стен и подвешенные в складку на расстоянии 15 - 20 см от ограждения. Ширина занавеси должна быть в 2 раза больше ширины окна.

Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

Требования к освещению на рабочих местах, оборудованных ПЭВМ

Рабочие столы следует размещать таким образом, чтобы мониторы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк. Освещение не должно создавать бликов

на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м2.

Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м2 и яркость потолка не должна превышать 200 кд/м2.

Показатель ослепленности для источников общего искусственного освещения в производственных помещениях должен быть не более 20.

Показатель дискомфорта в административно-общественных помещениях - не более 40.

Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м2, защитный угол светильников должен быть не менее 40 градусов.

Светильники местного освещения должны иметь не просвечивающий отражатель с защитным углом не менее 40 градусов.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1 - 5:1, а между рабочими поверхностями и поверхностями стен и оборудования - 10:1.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп. В светильниках местного

освещения допускается применение ламп накаливания, в том числе галогенных.

Для освещения помещений с ПЭВМ следует применять светильники с зеркальными параболическими решетками, укомплектованными электронными пускорегулирующими аппаратами (ЭПРА). Допускается использование многоламповых светильников с электромагнитными пускорегулирующими аппаратами (ЭПРА), состоящими из равного числа опережающих и отстающих ветвей.

Применение светильников без рассеивателей и экранирующих решеток не допускается.

При отсутствии светильников с ЭПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует включать на разные фазы трехфазной сети.

Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении видеодисплейных терминалов. При периметральном расположении компьютеров линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

Коэффициент запаса для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах пользователей представлены в таблице 4.

Таблица 11. Временные допустимые уровни ЭМП по СанПиН 2.2.4.1294-03

Наименование параметров		ВДУ
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

Требования к организации рабочих мест пользователей ПЭВМ

Общие требования к организации рабочих мест пользователей ПЭВМ

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

Рабочие места с ПЭВМ в помещениях с источниками вредных производственных факторов должны размещаться в изолированных кабинах с организованным воздухообменом.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию. Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Требования к пожаробезопасности на рабочих местах, оборудованных ПЭВМ

Наиболее вероятной причиной пожара в помещении с работающей ПЭВМ является неисправность электрооборудования и электросетей. При эксплуатации ПЭВМ возможны возникновения следующих аварийных ситуаций: короткие замыкания, перегрузки, повышение переходных

сопротивлений в электрических контактах, перенапряжение, возникновение токов утечки. При возникновении аварийных ситуаций происходит резкое выделение тепловой энергии, которая может явиться причиной возникновения пожара. Требования к пожаробезопасности зданий и сооружений определяются согласно СНиП 21.01-97.

Для снижения вероятности возникновения пожара необходимо проводить различные профилактические мероприятия:

- Организационные – правильная эксплуатация электрооборудования, правильное содержание зданий и помещений;
- Технические — соблюдение противопожарных правил и норм, норм при проектировании зданий, при устройстве отопления, вентиляции освещения, правильное размещение оборудования;
- Мероприятия режимного характера – запрещение курения в неустановленных местах и т.д.;
- Эксплуатационные — своевременные профилактические осмотры и ремонт неисправного электрооборудования.

Для снижения вероятности возникновения и распространения пожара на ранней стадии необходимо:

- установить пожарную сигнализацию с системой оповещения работников, дежурного по объекту и, желательно, автоматическое оповещение противопожарных служб;
- иметь в наличии несколько ручных огнетушителей

Требования к электробезопасности на рабочих местах, оборудованных ПЭВМ

Питание ПЭВМ и приборов осуществляется через сеть с частотой 50 Гц и напряжением 220 В (+/-10...15%).

Для обеспечения работы пользователей ЭВМ необходимо исключить возможность случайного соприкосновения людей с токонесущими частями оборудования. Это достигается путем изоляции токоведущих частей ЭВМ и приборов и размещения их в недоступных зонах.

Для обеспечения электробезопасности согласно ПУЭ (правилам устройства электроустановок) необходимо выполнить заземление электроустановок:

- при номинальном напряжении более 50 В переменного тока (действующее значение), и более 120 В постоянного (выпрямленного) тока - во всех электроустановках;
- при номинальных напряжениях выше 25 В переменного тока, но ниже 50 В, и выше 60 В, но до 120 В, постоянного тока в помещениях с повышенной опасностью, особо опасных и в наружных электроустановках;

В данном случае ПЭВМ и другие электроприборы используются в помещении без повышенной опасности, но при постоянном токе превышающем 120 В, поэтому необходимо выполнить зануление электроустановок.

Для заземляющих устройств в первую очередь должны быть использованы естественные заземлители:

- водопроводные трубы, проложенные в земле;
- металлические конструкции зданий и сооружений, имеющие надежное соединение с землей;
- металлические оболочки кабелей (кроме алюминиевых);
- обсадные трубы артезианских скважин.

Запрещается в качестве заземлителей использовать трубопроводы с горючими жидкостями и газами, трубы теплотрасс.

Естественные заземлители должны иметь присоединение к заземляющей сети не менее чем в двух разных местах.

В качестве искусственных заземлителей применяют:

- стальные трубы с толщиной стенок 3.5 мм, длиной 2 - 3 м;
- полосовую сталь толщиной не менее 4 мм;
- угловую сталь толщиной не менее 4 мм;
- прутковую сталь диаметром не менее 10 мм, длиной до 10 м и более.

Все элементы заземляющего устройства соединяются между собой при помощи сварки, места сварки покрываются битумным лаком. Допускается присоединение заземляющих проводников к корпусам электрооборудования с помощью болтов.

Необходимо соблюдать требования техники безопасности при работе с высоким напряжением и следующие меры предосторожности:

- не оставлять ПЭВМ и другое оборудование под напряжением без наблюдения персонала;
- не подключать разъёмы кабелей ПЭВМ при включении напряжения в сети.

Определение вредных и опасных производственных факторов

Определив общий список требований к организации работы с ПЭВМ в соответствии с СанПиН 2.2.4.1294-03, сформируем список *основных* опасных и вредных факторов при работе диспетчера.

Вредный производственный фактор - фактор среды и трудового процесса, который может вызвать снижение работоспособности, патологию (профессиональное заболевание), привести к нарушению здоровья потомства.

Опасный производственный фактор - фактор среды и трудового процесса, который может вызвать резкое ухудшение здоровья, травму, смерть.

Таблица 12. Опасные и вредные производственные факторы.

Опасные производственные факторы	Вредные производственные факторы
Электрический ток	Недостаточная освещенность
	Загазованность и запыленость воздуха
	Монотонный и малоподвижный характер работы

Расчеты вредных и опасных факторов

Расчет общего искусственного освещения на рабочем месте диспетчера

Для расчета общего равномерного освещения используем метод светового потока, учитывающий световой поток, отраженный от потолка и стен [см. Юдин Е. Я., Белов С. В. Охрана труда в машиностроении]

Исходные данные для расчета:

Длина помещения	6
Ширина помещения	8
Высота помещения	3
Тип ламп	Люминес-центные
Потолок	Побеленный
Стены	Чистые, светлые

Требуется определить количество ламп и их параметры.

Потребный световой поток группы ламп светильника Фл (лм) находится по формуле:

$$F_{л} = \frac{100 \times E_{н} \times k_{з} \times S \times z}{N \times \eta},$$

где Ен - минимальная нормированная освещенность, для нашего случая Ен = 300 лк;

kз - коэффициент запаса, находится по таблицам; учитывает старение ламп, запыление и загрязнение светильников. Принимаем kз = 1,4.

S - площадь освещаемой поверхности. Для данного помещения S = 48 м кв.

z - коэффициент минимальной освещенности, равный отношению средней освещенности к минимальной. В нашем случае – люминесцентные лампы, z = 1,1.

η - коэффициент использования светового потока ламп (%), зависящий от типа светильника, коэффициента отражения потолка Rоп и стен Rос и индекса i формы помещения.

Показатель помещения:

$$i = \frac{A \times B}{h \times (A + B)},$$

где A и B - ширина и длина помещения; h - высота подвеса светильников над рабочей поверхностью. Для нашего случая:

$$i = \frac{6 \times 8}{3 \times (6 + 8)} \approx 1.125$$

Для помещения Rоп = 70% (побеленный потолок), Rос = 50% (чистые светлые стены) находим коэффициент использования светового потока для светильников с люминесцентными лампами типа ЛБ - η = 34%.

Находим потребный световой поток для люминесцентных ламп:

$$F_л * N = \frac{100 \times 300 \times 1.4 \times 48 \times 1.1}{34} = 65220 лм$$

Параметры люминесцентных ламп общего назначения для ламп типа ЛБ:

Мощность, Вт	Сила тока, А	Напряжение, В	Световой поток, лм
30	0,35	104± 10,4	2180
40	0,43	103± 10,3	3200
65	0,67	110± 10,0	4800
80	0,87	102± 10,2	5400

В помещении оператора будем использовать люминесцентные лампы со световым потоком 4800 лм. Определим потребное количество светильников:

N = 65220/4800 = 13,58 = 14 шт.

Вывод: Для создания минимальной нормированной освещенности в помещении оператора подсистемы диспетчирования требуется оборудовать помещение 14тью светильниками с мощностью ламп 4800лм.

Заключение

В рамках работы по созданию подсистемы диспетчирования перевозок серы на ЖД транспорте были получены следующие результаты:

1. Проведено предпроектное исследование объекта автоматизации – процесса доставки серы потребителям.
2. Определено место подсистемы диспетчирования в составе АСУ ПС, определены цели создания подсистемы и функциональный состав.
3. Была разработана концепция подсистемы диспетчирования – предложен модульный состав подсистемы.
4. Был осуществлен выбор технологии реализации взаимодействия с БД.
5. Было составлено техническое задание отдельно на каждый модуль подсистемы диспетчирования.
6. На техническом этапе работ были разработаны:
   - Диаграмма состояний подсистемы диспетчирования в нотации UML.
   - Структура модуля ввода фактического положения ВП
   - Алгоритмы моделирования и диаграммы состояний ресурсов имитационной модели
7. На рабочем этапе была разработана имитационная модель на языке РДО совместно с модулем формирования отчетов о моделировании.
8. Проведено апробирование разработанного функционала системы на тестовом примере.
9. В организационно-экономической части дипломного проекта проведен расчет затрат на разработку подсистемы диспетчирования.
10. Приведены требования безопасности при работе пользователей с разработанной системой, а также проведен типовой расчет освещенности помещения оператора ПЭВМ.

Все поставленные цели дипломного проекта достигнуты, требуемый функционал системы реализован.

Полученные результаты демонстрируют возможность создания полнофункциональной подсистемы по разработанному техническому заданию.

Список использованной литературы

1. Емельянов В. В., Ясиновский С. И. Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. – М.: «АНВИК», 1998. – 427 с., ил. 136.
2. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II: Пер. с англ. – М.: Мир, 1987. – 646 с.
3. Арсеньев В.В., Сажин Ю.Б. Методические указания к выполнению организационно-экономической части дипломных проектов по созданию программной продукции. М.: изд. МГТУ им. Баумана, 1994. 52 с.
4. Юдин Е. Я., Белов С. В. Охрана труда в машиностроении: Учебник для О-92 машиностроительных вузов/ Е. Я. Юдин, С. В. Белов и др., Под ред. Е. Я. Юдина, С. В. Белова – 2-е изд., перераб. и доп. – М.: Машиностроение, 1983, 432 с., ил.
5. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. СанПиН 2.2.2/2.4.1340–03 [http://www.mhts.ru/biblio/].
6. Санитарные правила и нормы. Гигиенические требования к микроклимату производственных помещений. СанПиН 2.2.4.548-96 [http://www.mhts.ru/biblio/].
7. СанПиН 2.2.4.1294-03. Санитарно-эпидемиологические правила и нормативы. Гигиенические требования к аэроионному составу воздуха производственных и общественных помещений [http://www.mhts.ru/biblio/].
8. Предельно допустимые концентрации (ПДК) вредных веществ в воздухе рабочей зоны. ГН 2.2.5.1313–03[http://www.mhts.ru/biblio/].
9. Горнев В. Ф., Грибанов Н.Г., Овсянников М.В., Методические указания к дипломному проектированию для студентов кафедры РК-9

«Компьютерные системы автоматизации производства». Учебное пособие. - М.: Каф. РК9 МГТУ им. Н.Э.Баумана, 2004, 41с.

10. Справка по языку РДО (в составе программы) [http://rdo.rk9.bmstu.ru/forum/viewforum.php?f=15]
11. Леоненков. Самоучитель UML. [http://khpi-iip.mipk.kharkiv.edu/library/case/leon/]
12. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78.
13. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-90. Условные обозначения и правила выполнения.
14. Норенков И.П. Основы автоматизированного проектирования. – М: «МГТУ им. Баумана», 2006. – 447 с., ил.
15. Бьерн Страуструп. Язык программирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2007 г. – 1104 с.: ил.

Приложение 1. Листинг файла реализации модуля ввода фактического положения парка вагонов.

Файл MVTS.h

```
// MVTS.h : main header file for the MVTS application
//
#pragma once

#ifndef __AFXWIN_H__
	#error "include 'stdafx.h' before including this file for PCH"
#endif

#include "resource.h"		// main symbols
// CMVTSApp:
// See MVTS.cpp for the implementation of this class
//
class CMVTSApp : public CWinApp
{
public:
	CMVTSApp();
// Overrides
public:
	virtual BOOL InitInstance();
// Implementation
	afx_msg void OnAppAbout();
	DECLARE_MESSAGE_MAP()
};
extern CMVTSApp theApp;
```

Файл MVTSDoc.h

```
#pragma once
#include "MVTSSet.h"
class CMVTSDoc : public CDocument
{
protected: // create from serialization only
	CMVTSDoc();
	DECLARE_DYNCREATE(CMVTSDoc)
// Attributes
public:
	CMVTSSet m_MVTSSet;
// Operations
public:
// Overrides
public:
	virtual BOOL OnNewDocument();
// Implementation
public:
	virtual ~CMVTSDoc();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// Generated message map functions
protected:
	DECLARE_MESSAGE_MAP()
};
```

Файл MVTSSet.h

```
#pragma once

class CMVTSSet : public CRecordset
{
public:
	CMVTSSet(CDatabase* pDatabase = NULL);
	DECLARE_DYNAMIC(CMVTSSet)

	long		m_Group_number;
	CStringW	m_Start_point;
	CStringW	m_Destination;
	double		m_Location;

// Overrides
	// Wizard generated virtual function overrides
	public:
	virtual CString GetDefaultConnect();	// Default connection string
	virtual CString GetDefaultSQL();	// default SQL for Recordset
	virtual void DoFieldExchange(CFieldExchange* pFX);	// RFX support
// Implementation
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif
};
```

Файл MVTSView.h

```
#include <vector>

#pragma once

// Описание текущего положения группы
class TrainLocation
{
public:
	long Group_number;
	CStringW Start_point;
	CStringW Destination;
	double Location;
};

class CMVTSSet;

class CMVTSView : public CRecordView
{
protected: // create from serialization only
	CMVTSView();
	DECLARE_DYNCREATE(CMVTSView)

public:
	enum{ IDD = IDD_MVTS_FORM };
	CMVTSSet* m_pSet;

// Attributes
public:
	CMVTSDoc* GetDocument() const;

// Operations
public:

// Overrides
```

```
public:
	virtual CRecordset* OnGetRecordset();
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	virtual void OnInitialUpdate(); // called first time after construct
	virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
	virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
	virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);

// Implementation
public:
	virtual ~CMVTSView();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClickedButton1();
};

#ifndef _DEBUG  // debug version in MVTSView.cpp
inline CMVTSDoc* CMVTSView::GetDocument() const
   { return reinterpret_cast<CMVTSDoc*>(m_pDocument); }
#endif
```

MVTS.cpp

```
#include "stdafx.h"
#include "MVTS.h"
#include "MainFrm.h"

#include "MVTSSet.h"
#include "MVTSDoc.h"
#include "MVTSView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMVTSApp

BEGIN_MESSAGE_MAP(CMVTSApp, CWinApp)
	ON_COMMAND(ID_APP_ABOUT, &CMVTSApp::OnAppAbout)
	// Standard print setup command
	ON_COMMAND(ID_FILE_PRINT_SETUP, &CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()


// CMVTSApp construction

CMVTSApp::CMVTSApp()
{
	// TODO: add construction code here,
	// Place all significant initialization in InitInstance
}
```

```
// The one and only CMVTSApp object

CMVTSApp theApp;


// CMVTSApp initialization

BOOL CMVTSApp::InitInstance()
{

	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
	// Set this to include all the common control classes you want to use
	// in your application.
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);

	CWinApp::InitInstance();

	// Initialize OLE libraries
	if (!AfxOleInit())
	{
		AfxMessageBox(IDP_OLE_INIT_FAILED);
		return FALSE;
	}
	AfxEnableControlContainer();
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));
	LoadStdProfileSettings(4);
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CMVTSDoc),
		RUNTIME_CLASS(CMainFrame),       // main SDI frame window
		RUNTIME_CLASS(CMVTSView));
	if (!pDocTemplate)
		return FALSE;
	AddDocTemplate(pDocTemplate);
	// Parse command line for standard shell commands, DDE, file open
	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);

	if (!ProcessShellCommand(cmdInfo))
		return FALSE;

	// The one and only window has been initialized, so show and update it
	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();
	// call DragAcceptFiles only if there's a suffix
	//  In an SDI app, this should occur after ProcessShellCommand
	return TRUE;
}

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	enum { IDD = IDD_ABOUTBOX };
```

```
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// App command to run the dialog
void CMVTSApp::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}
```

MVTSDoc.cpp

```
#include "stdafx.h"
#include "MVTS.h"

#include "MVTSSet.h"
#include "MVTSDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMVTSDoc

IMPLEMENT_DYNCREATE(CMVTSDoc, CDocument)

BEGIN_MESSAGE_MAP(CMVTSDoc, CDocument)
END_MESSAGE_MAP()


// CMVTSDoc construction/destruction

CMVTSDoc::CMVTSDoc()
{
	// TODO: add one-time construction code here

}

CMVTSDoc::~CMVTSDoc()
{
}

BOOL CMVTSDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;
```

```
	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)

	return TRUE;
}

// CMVTSDoc diagnostics

#ifdef _DEBUG
void CMVTSDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CMVTSDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif
```

MVTSView.cpp

```
#include "stdafx.h"
#include "MVTS.h"

#include "MVTSSet.h"
#include "MVTSDoc.h"
#include "MVTSView.h"
#include <stdio.h>
#include < stdlib.h >

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

IMPLEMENT_DYNCREATE(CMVTSView, CRecordView)

BEGIN_MESSAGE_MAP(CMVTSView, CRecordView)
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, &CRecordView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_DIRECT, &CRecordView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW, &CRecordView::OnFilePrintPreview)
	ON_BN_CLICKED(IDC_BUTTON1, &CMVTSView::OnBnClickedButton1)
END_MESSAGE_MAP()

// CMVTSView construction/destruction

CMVTSView::CMVTSView()
	: CRecordView(CMVTSView::IDD)
{
	m_pSet = NULL;

}

CMVTSView::~CMVTSView()
{
}
```

```
void CMVTSView::DoDataExchange(CDataExchange* pDX)
{
	CRecordView::DoDataExchange(pDX);
	DDX_FieldText(pDX, IDC_EDIT1, m_pSet->m_Group_number, m_pSet);
	DDX_FieldText(pDX, IDC_EDIT2, m_pSet->m_Start_point, m_pSet);
	DDX_FieldText(pDX, IDC_EDIT3, m_pSet->m_Destination, m_pSet);
	DDX_FieldText(pDX, IDC_EDIT4, m_pSet->m_Location, m_pSet);
}

BOOL CMVTSView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	return CRecordView::PreCreateWindow(cs);
}

void CMVTSView::OnInitialUpdate()
{
	m_pSet = &GetDocument()->m_MVTSSet;
	CRecordView::OnInitialUpdate();

}

// CMVTSView printing

BOOL CMVTSView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}

void CMVTSView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add extra initialization before printing
}

void CMVTSView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: add cleanup after printing
}

// CMVTSView diagnostics

#ifdef _DEBUG
void CMVTSView::AssertValid() const
{
	CRecordView::AssertValid();
}

void CMVTSView::Dump(CDumpContext& dc) const
{
	CRecordView::Dump(dc);
}

CMVTSDoc* CMVTSView::GetDocument() const // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CMVTSDoc)));
	return (CMVTSDoc*)m_pDocument;
}
#endif
```

```
// CMVTSView database support
CRecordset* CMVTSView::OnGetRecordset()
{
	return m_pSet;
}

// CMVTSView message handlers

void CMVTSView::OnBnClickedButton1()
{
	FILE *stream;
	OPENFILENAME smr_name;
	char szFile[100] ;
	ZeroMemory( &smr_name , sizeof( smr_name));
	smr_name.lStructSize = sizeof ( smr_name );
	smr_name.hwndOwner = NULL  ;
	smr_name.lpstrFile = (LPWSTR) szFile ;
	smr_name.lpstrFile[0] = '\0';
	smr_name.nMaxFile = sizeof( szFile );
	smr_name.lpstrFilter = (LPCWSTR) L"All Files\0*.*\0\0";
	smr_name.nFilterIndex = 1;
	smr_name.lpstrFileTitle = NULL ;
	smr_name.nMaxFileTitle = 0 ;
	smr_name.lpstrInitialDir=NULL ;
	smr_name.Flags = OFN_PATHMUSTEXIST|OFN_FILEMUSTEXIST ;
	GetOpenFileName(&smr_name);

	char file_path[20] ;
	char buffer[20] ;
	int i = 0;

	while (smr_name.lpstrFile[i])
		file_path[i++] = smr_name.lpstrFile[i];
	file_path[i] = '\0' ;

	stream = fopen(file_path, "w" );

	//
	TrainLocation Train_Location_Buffer;
	std::vector <TrainLocation> V_Group_location;
	m_pSet->MoveFirst() ;
	while (!m_pSet->IsEOF())
	{
		Train_Location_Buffer.Group_number  = m_pSet->m_Group_number;
		fprintf(stream,"%d ",m_pSet->m_Group_number) ;
		Train_Location_Buffer.Start_point = m_pSet->m_Start_point;

		for (i = 0; i < 20; i++)
			buffer[i] = 0;
		for (i = 0; i<=Train_Location_Buffer.Start_point.GetLength();i++)
			buffer[i] = Train_Location_Buffer.Start_point[i] ;
		buffer[i] = '\0' ;
		fprintf(stream,"%s ", buffer) ;

		Train_Location_Buffer.Destination = m_pSet->m_Destination;

		for (i = 0; i < 20; i++)
			buffer[i] = 0;
		for (i = 0; i<=Train_Location_Buffer.Destination.GetLength();i++)
			buffer[i] = Train_Location_Buffer.Destination[i] ;
		buffer[i] = '\0' ;
		fprintf(stream,"%s ", buffer) ;
```

```
            Train_Location_Buffer.Location = m_pSet->m_Location;
            fprintf(stream,"%f\n",m_pSet->m_Location) ;

            V_Group_location.push_back(Train_Location_Buffer);
            m_pSet->MoveNext();
        }
}
```