МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Политехнический институт
Кафедра «Робототехника и автоматизация производства»

На правах рукописи

ГРИШИН Константин Анатольевич

УДК 004.051

ОРГАНИЗАЦИЯ РАБОТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЕРАРХИЧЕСКОЙ ЦИФРОВОЙ СИСТЕМЫ УПРАВЛЕНИЯ СЛОЖНЫМИ ОБЪЕКТАМИ

Диссертация на соискание ученой степени
кандидата технических наук по специальности
2.3.5. Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Научный руководитель: д.т.н., профессор
Ларкин Евгений Васильевич

Тула, 2022

Содержание

ВВЕДЕНИЕ 5
1. ЦИФРОВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ГОФРОАГРЕГАТАМИ И ПРОБЛЕМЫ ИХ РАЗРАБОТКИ 11
1.0. Введение 11
1.1. Технические средства гофроиндустрии 11
1.2. Обобщенная функциональная схема гофроагрегата..... 14
1.2.1. Функциональная схема и задачи, решаемые технологическим процессом 14
1.2.2. Компьютерные системы управления объектами 24
1.2.3. Особенности работы программного обеспечения 31
1.2.4. Случайность времени обработки данных 31
1.2.5. Особенности работы человека-оператора 35
1.3. Общий принцип реализации команд цифровой системой управления..... 37
1.3.1. Учет требований теоремы Котельникова..... 38
1.3.2. Учет времени запаздывания..... 42
1.3.3. Шум полинга и перекос данных..... 45
1.3.4. Обоснование выбора теории полумарковских процессов для моделирования и оптимизации циклограмм управления гофроагрегата..... 46
1.4. Выводы 49
2. ИСПОЛЬЗОВАНИЕ ТЕОРИИ ПОЛУМАРКОВСКИХ ПРОЦЕССОВ ДЛЯ МОДЕЛИРОВАНИЯ ПРОГРАММ УПРАВЛЕНИЯ ГОФРОАГРЕГАТОМ 51
2.0. Введение 51
2.1. Модель программы как ординарный полумарковский процесс 51
2.2. Определение и способы задания полумарковского процесса 57
2.2.1. Определение полумарковского процесса 57
2.2.2. Представление полумарковского процесса в виде взвешенного

графа 59
2.2.3. Блуждания по полумарковской цепи 60
2.3. Модель циклограммы как полумарковский процесс 63
2.3.1. Полумарковская матрица и структура циклограммы 63
2.3.2. Свойства полумарковской модели циклограммы 65
2.4. Определение временных интервалов блуждания по эргодическим полумарковским процессам 66
2.4.1. Характеристическая полумарковская матрица 66
2.4.2. Временной интервал блуждания по выделенной траектории 68
2.4.3. Временной интервал блужданий по одной из возможных траекторий 70
2.4.4. Полумарковский процесс, сформированный из траекторий блуждания 73
2.4.5. Методики определения временных интервалов блуждания по эргодическому полумарковскому процессу 74
2.5. Оценка степени «марковости» процессов 79
2.5.1. Регрессионный критерий и критерий Пирсона 80
2.5.2. Корреляционный критерий 81
2.5.3. Параметрические критерии 82
2.5.4. Случай равномерного закона распределения 89
2.6. Выводы 91
3. МОДЕЛИ ДИСПЕТЧЕРИЗАЦИИ. 93
3.0. Введение 93
3.1. Параллельный полумарковский процесс 94
3.1.1. Понятие параллельного полумарковского процесса 94
3.1.2. Простейший *M*-параллельный полумарковский процесс 97
3.1.3. Сложный *M*-параллельный полумарковский процесс 103
3.2. Генерация команд управления в диалоговом режиме 111
3.2.1. Полумарковский процесс генерации команд 111

3.2.2. Генерация команд 114
3.3. Моделирование исполнения команд 125
3.3.1. Модель циклической дисциплины диспетчеризации 126
3.3.2. Модели квазистохастических дисциплины диспетчеризации 131
3.4. Выводы 136
4. ОЦЕНКА ХАРАКТЕРИСТИК ПРОГРАММ УПРАВЛЕНИЯ ГОФРОАГРЕГАТОМ 137
4.0. Введение 137
4.1. Использование сетей Петри-Маркова в задачах моделирования процессов диспетчеризации 138
4.1.1. Назначение программы 141
4.1.2. Использование программы в режиме редактирования сети 145
4.1.3. Формирование/редактирование структуры сети Петри-Маркова 146
4.1.4. Задание/редактирование параметров сети Петри-Маркова 151
4.1.5. Метод имитационного моделирования 155
4.1.6. Структура классов объектов 158
4.1.7. Разработка программного обеспечения 165
4.1.8. Визуальный интерфейс программы 166
4.1.9. Описание отдельных функций 173
4.1.10. Описание формата файла сети Петри-Маркова 176
4.2. Выводы 183
ЗАКЛЮЧЕНИЕ 184
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 186
ПРИЛОЖЕНИЕ 202

ВВЕДЕНИЕ

Актуальность темы. Цифровизация производства является основной тенденцией развития промышленности в настоящее время. Цифровые технологии управления сложным технологическим оборудованием предполагают замену жестко программируемых аналоговых контроллеров на универсальные или специализированные ЭВМ Фон Неймановского типа, в которых требуемые законы управления реализуются программно. Подобные цифровые регуляторы обладает новыми, по сравнению с аналоговыми контроллерами, свойствами, обусловленными последовательной интерпретацией операторов алгоритма управления, развивающейся в реальном физическом времени. При управлении сложными объектами формируется множество контуров, каждый из которых обеспечивает стабилизацию или программное изменение соответствующего технологического параметра за счет обработки сигналов сенсоров и вывода управляющего воздействия на приводы. В то же время в промышленности накоплен богатый опыт реализации континуальных регуляторов, который не может быть без дополнительных исследований распространен на цифровые технологии, поскольку последовательная интерпретация алгоритмов управления приводит к таким явлениям, как перекос данных, и чистое запаздывание в цепи обратной связи, что оказывают существенное влияние на качественные характеристики управления.

Величины параметров перекоса данных и чистого запаздывания зависят, во-первых, от общей структуры системы управления (с одним контроллером или с множеством контроллеров, организованных в иерархическую структуру), а во-вторых, от качества диспетчеризации задач и/или от качества согласования функционирования множества котроллеров. Уменьшение величины перекоса данных и чистого запаздывание достигается за счет использования при управлении сложными объектами ЭВМ с иерархической организацией, в которой на функционально-логическом уровне замыкаются обратные связи контуров, а на тактическом уровне обеспечивается организация работы контроллеров, реализующих обратные связи в системе. Организация работы подобных ЭВМ на этапе проектирова-

ния программного обеспечения затруднена тем, что на сегодняшний день в инженерной практике недостаточно применяются методы априорной оценки временной сложности алгоритмов управления, создающих запаздыванию и перекос данных, а следовательно получаемые инженерные решения не могут считаться оптимальными.

Таким образом, потребности в эффективных программных комплексах, организующих работу цифровой системы управления сложными многоконтурными объектами, построенной на базе Фон Неймановских контроллеров, и отсутствие общей теории реализации подобных комплексов объясняет необходимость и ***актуальность*** исследований, проведенных в диссертации.

Объектом исследования диссертационной работы является программное обеспечение цифровых систем управления, реализующее циклограммы управления функционированием узлов и блоков сложных многоконтурных агрегатов, а также программы, обеспечивающие согласование работы узлов и блоков во времени.

Предметом исследования диссертационной работы являются методы оценки временной вычислительной сложности программ управления, и методы согласования функционирования управляющих программ во времени.

Примером сложного агрегата, на котором реализованы разработанные в диссертации методы, является гофроагрегат, в котором одновременно выполняется множество операций по предварительной подготовке заготовок к склейке гофрированного картона, склейка, сушка, транспортировка, нарезка и т.п.

В диссертации использован подход, к оценке временной сложности программ и согласования их работы во времени, основанный на ***аналитических методах*** математического моделирования. Математические модели оценки временной сложности основываются на теории полумарковских процессов, согласование работы программ во времени базируются на теории систем. Разработка программного продукта производилось в среде программирования С++.

Вопросы полумарковских процессов отражены в работах В.С. Королюка, А.Ф. Турбина, Д.С. Сильвестрова, Р. Ховарда. В работах Е.В Ларкина. решались

задачи прикладного применения теории полумарковских процессов для оценки временных характеристик программного продукта. Вопросами разработки системного программного обеспечения занимались Карл Вернер, Ванг Хуа Юнг. Оптимизацией программного продукта занимались Дональд Кнут, Крис Касперски, Джон Луис Бентли и др.

Цель диссертационной работы состоит в повышении качества управления сложными многоконтурными агрегатами за счет разработки системного программного обеспечения иерархических цифровых систем управления.

В соответствии с поставленной целью в диссертации решены следующие ***задачи.***

1. Выявлены общие закономерности работы программ управления сложными многоконтурными объектами, реализуемых на ЭВМ Фон Неймановского типа, и подбор фундаментальных теорий, которые могли бы быть положены в основу метода их аналитического математического моделирования.

2. Разработан метод оценки временной вычислительной сложности программ по их полумарковским моделям.

3. Разработан метод объединения ординарных полумарковских процессов в *M*-параллельный полумарковский процесс.

4. Разработан метод оценки временной вычислительной сложности *M*-параллельного полумарковского процесса.

5. Создан метод диспетчеризации в системах цифрового управления с одной управляющей ЭВМ и в многоконтроллерных системах.

6. Проведено конфигурирование иерархической цифровой системы управления гофроагрегатом.

7. Для цифровой системы управления разработан комплекс программ по управлению узлами и блоками агрегата.

8. Разработана рекомендации по организации работы комплекса программ управления.

Научная новизна диссертации заключается в следующем.

1. Построена модель *M*-параллельного дискретного полумарковского процесса, основанная на синергетическом соединении ординарных полумарковских процессов, что позволяет исследовать соревнование субъектов и строить схемы диспетчеризации, обеспечивающие оптимальную загрузку управляемого оборудования.

2. Разработан метод приведения континуальных плотностей распределения к дискретному виду, что позволяет распространить результаты, полученные для дискретных распределений, на случай *M*-параллельных полумарковских процессов с континуальными законами распределения.

3. Разработана метод диспетчеризации программ управления, обеспечивающий оптимальное распределение задач при управлении многоконтурными объектами.

Практическая ценность работы заключается в том, что методы создания программного обеспечения иерархической системы управления апробированы на задаче управления гофроагрегатом, что позволяет рекомендовать их для практического применения в широкой инженерной практике.

Достоверность полученных теоретических результатов подтверждается корректным применением аналитических моделей математического моделирования, использующие теорию полумарковских процессов, теорию оценки эффективности, теорию управления, а также экспериментальными исследованиями временных и стохастических характеристик циклограмм по их полумарковским моделям.

Научные положения, выносимые на защиту.

1. Модель *M*-параллельного дискретного полумарковского процесса, основанная на синергетическом соединении ординарных полумарковских процессов, что позволяет исследовать соревнование субъектов и строить схемы диспетчеризации, обеспечивающие оптимальную загрузку управляемого оборудования.

2. Метод приведения континуальных плотностей распределения к дискретному виду, что позволяет распространить результаты, полученные для дис-

кретных распределений, на случай *M*-параллельных полумарковских процессов с континуальными законами распределения.

3. Метод диспетчеризации программ управления, обеспечивающий оптимальное распределение задач при управлении многоконтурными объектами.

Реализация и внедрение результатов. Предложенные в диссертации методы и методики реализованы автором в учебном процессе кафедры РТиАП и подразделения УНИР.

Работа выполнена при поддержке гранта Российского фонда фундаментальных исследований № 19-38-90066 «Аспиранты» и госзадания № 2.3121.2017/ПЧ на тему: «Параллельные полумарковские процессы в системах управления мобильными роботами»,

Апробация работы. Основные положения диссертации докладывались на следующих конференциях и семинарах.

1. Международная конференция The 7th International Conference on Fuzzy Systems and Data Mining (South Korea, Seoul, 2021)

2. Ежегодные научно-практические конференции профессорско-преподавательского состава Тульского государственного университета (Россия, Тула, 2016 – 2021).

По теме диссертации опубликовано 17 работ, включенных в список литературы, в том числе: 15 работ в сборниках, рекомендуемых ВАК РФ для публикаций материалов кандидатских диссертаций, 1 работа, опубликованная в изданиях, номинированных в базе данных «Scopus», а также 1 патент на полезную модель.

Структура и объем работы. Диссертационная работа состоит из введения, четырех разделов, изложенных на 202 страницах машинописного текста и включающих 64 рисунков и 5 таблиц, заключения, списка использованной литературы из 160 наименований и 1 приложения.

Краткое содержание диссертации.

Во ***введении*** к диссертации отражена актуальность темы, определены объект, предмет, методы и задачи исследования, дана общая характеристика работы, обозначены основные положения, выносимые на защиту, а также приведены

краткие аннотации разделов диссертации.

В ***первом разделе*** проведен анализ существующих цифровых систем управления и их программного обеспечения, обеспечивающего синхронную работу отдельных управляющих контуров сложных технологических объектов.

Приводится типовая структурная схема цифровой системы управления сложными технологическими процессами на примере системы управления гофроагрегатом. Отмечается, что типовая структура является трехуровневой иерархической.

Во ***втором разделе*** представлена модель работы программного обеспечения одного контроллера как ординарный полумарковский процесс, получены выражения для определения плотности распределения времени выполнения последовательности действий при реализации циклограмм управления; для указанных плотностей определены математическое ожидание и дисперсия времени выполнения. Получено выражение для определения плотности распределения временного интервала выполнения одного из возможных действий при реализации циклограмм управления.

В ***третьем разделе*** сформулировано понятие *M*-параллельного полумарковского процесса и определены операции, на нем. Показано, что после получения *M*-параллельного полумарковского процесса анализа временных характеристик работы иерархической ЭВМ сводится к выполнению операций.

Полученные характеристики ординарных и *M*-параллельного полумарковских процессов при решении задачи управления опросом ЭВМ Фон Неймановского типа функционально-логического уровня в иерархической системе управления.

В ***четвертом разделе*** разработано программное обеспечение организации работы иерархической цифровой системы управления гофроагрегата, позволяющее оценить численно вероятностные и временные характеристики блужданий по полумарковским процессам.

В ***заключении*** сделаны выводы по работе.

Приложение содержит код программы.

1. ЦИФРОВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ГОФРОАГРЕГАТАМИ И ПРОБЛЕМЫ ИХ РАЗРАБОТКИ

1.0. Введение

Гофроагрегат представляет собой ряд скомпонованных в одну линию отдельных машин и узлов, на которых осуществляется не только процесс производства гофрированного картона, но и отдельные операции по его переработке.

Гофроагрегат включает в себя две части: мокрую и сухую. На первой ведется само производство гофрокартона, а вторая необходима для резки материала и подготовки его к транспортировке.

Мокрая часть также включает два отсека: клеильно-сушильный и гофрировальный. Последний содержит несколько элементов: гофрирующий мост; нагреватель; накопительный мост; устройство для разматывания рулонов.

Составляющие элементы сухой части предназначены для нарезки листов готового материала на заданные размеры и последующей укладке: листоукладчик (транспортер); рилевочно-резательное оборудование; станок поперечно-резательный.

1.1. Технические средства гофроиндустрии

Fosber. Секции гофроагрегатов выполнены с учетом основных потребностей гофропроизводств и передовых технологических решений:

Склейка полотна осуществляется в режиме non-stop благодаря уникальной конструкции склеивающей головки в бумагосоединительном узле;

Мощная конструкция станины гофропресса (35 тонны) позволяет исключить вибрацию и обеспечить высокое качество формирования гофры, стабильность параметров и долгий срок службы узла;

Рис. 1.1. Гофроагрегат Fosber

Периферическим нагрев гофровалов решает проблему образования и удаления конденсата;

Смена заказа выполняется всего за 25 секунд за счет роботизированного управления продольно-резательным узлом и попеременной работы двух пар ножей и рилевок;

Вырубка брака осуществляется сегментированным ножом, что сохраняет постоянное натяжение полотна и максимально уменьшает долю брака;

Управление работой линии осуществляется с одного пульта, что дает широкие возможности контроля: одновременная настройка всех узлов, диагностика их работы, синхронизация с системой отдела продаж.

TCY. Гофроагрегат для производства пятислойного гофрированного картона предназначена для получения гофрированного картона в 3 слоя или в пять слоев с максимальным форматом в 2200 мм без обрезки. Путем последовательной склейки 2 полотен бумаги с полотном предварительно гофрированной бумаги, в случае изготовления 3-х слойного картона. Последовательной склейки 3 полотен бумаги с 2 полотнами предварительно гофрированной бумаги, в случае изготоления пятислойного гофрократона.

Рис. 1.2. Гофроагрегат для производства пятислойного гофрокартона Модель: TCY-CCPL-2200-5L

J. S. Machine. Складская система J.S. Machine предназначена для складирования рулонов и готовой продукции – гофролиста. Данная система очень простая, ее принцип заключается в том, что когда логистический робот приносит готовую продукцию, складская система считывает штрих-код, расставляет все по местам и с помощью компьютера отдает оператору необходимую продукцию согласно штрих-коду. Такие системы уже есть, но мы старались заточить свои именно под гофропроизводство, поэтому убрали из них много ненужного для данной конкретно сферы. Соответственно, в связи с этим и цена на них будет ниже. Мы готовы разрабатывать для своих клиентов складские системы в индивидуальном порядке, разных размеров, параметров и требований.

Рис. 1.3. Гофроагрегат JETS400

Петромаш-Сервис. Первый гофроагрегат представлял собой набор автономных устройств, которые позволяли производить двухслойный гофрокартон. Лайнер и флютинг склеивались в рамках отдельного технологического процесса. Операции рилевки, а также продольной и поперечной резки выполнялись на отдельных машинах, как правило, ручного управления. Для изготовления трехслойного гофрокартона рулоны перемещались на склеивающую машину.

Рис. 1.4. Гофроагрегат для производства 5-ти слойного гофрокартона МОДЕЛЬ WJ-120-1400 F-II

1.2. Обобщенная функциональная схема гофроагрегата

1.2.1. Функциональная схема и задачи, решаемые технологическим процессом

Если абстрагироваться от среды, в которой оперирует гофроагрегат, то можно построить функциональную схему, которая приведена на рис. 1.5. Технологический процесс, включает пульт управления и контроллер, панель управления и машинную часть.

На пульте управления оператор гофроагрегата во взаимодействии с ЭВМ генерирует серию команд, обеспечивающих целенаправленное функционирование агрегата. [35, 36] При этом связь с центральным пультом осуществляется по

кабелю или по локальной сети. Оператор, в зависимости от решаемой тактической задачи, текущего состояния агрегата и навыков по управлению подобными системами вводит в ЭВМ, расположенную на пункте управления, очередную команду. Введенная команда кодируется с помощью ЭВМ и передается через систему связи. [71, 72, 182]

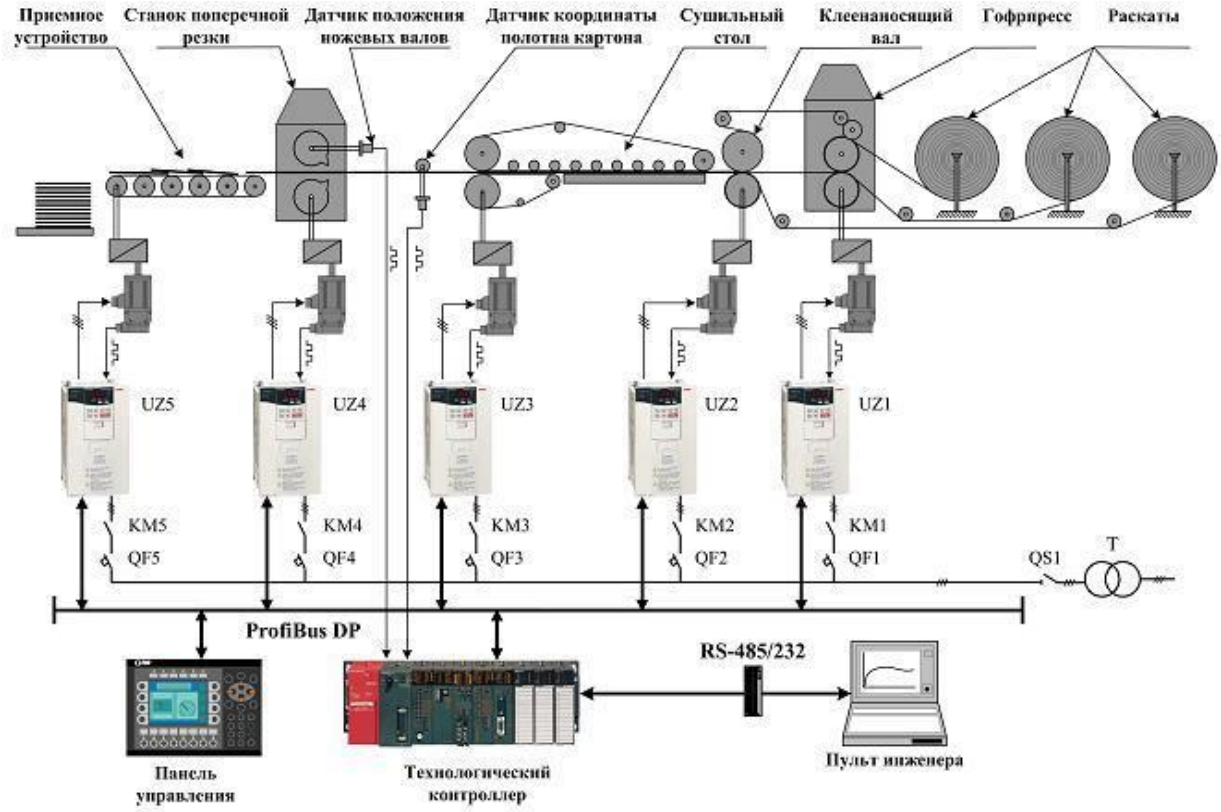


Рис. 1.5. Функциональная схема гофроагрегата

Принятое сообщение поступает на контроллер и декодируется. При необходимости передача/прием команды производится повторно для повышения надежности управления. В соответствии с поступившей командой ЭВМ по заранее зашитому в нее алгоритму выбирает узел, блок или механизм, который является текущим объектом управления, необходимым для перевода роботизированной платформы в состояние, обеспечивающее решение поставленной решаемой тактической задачи. Для текущего объекта управления определяется его текущее состояние и выбирается команда, которая должна переводить заданный узел в сле-

дующее состояние. Команда кодируется, и код через контроллер вводится в исполнительный блок и/или механизм. Исполнение команды контролируется сенсором, данные с которого через интерфейс вводятся в ЭВМ, кодируются и через систему связи передаются на ЭВМ пункта управления. На пункте управления данные, поступившие с платформы, декодируются, формируется сообщение оператору и через интерфейс отображаются на экране монитора. Таким образом, замыкается обратная связь, контролирующая исполнение очередной команды.

Рассмотрим подробно информационно-измерительные системы гофроагрегата, анализирующие информацию при производстве. Такие системы называют сенсорными. [31, 61, 62, 163, 164, 165, 181]

Сенсорные системы МР в соответствии с решаемыми задачами можно разделить на три группы:

1) системы, которые определяют общую картину окружающей среды с последующим обособлением различных объектов;

2) системы, которые определяют физико-химические свойства внешней среды и объектов в ее составе;

Первая группа включает в себя локаторы различного типа, вторая группа — сенсоры для измерения геометрических параметров, химического состава, плотности, температуры, оптических свойств и тому подобных.

Наиболее ответственным участком линии по производству гофрокартона является его «мокрая» часть (рис. 1.6.). Рассмотрим в действии ее основные узлы.

При помощи подогревателей повышается температура картона. В результате этого он становится более пластичным и послушным. Далее его подают на гофропресс. Он придает этому материалу требуемый внешний вид. Бумажное полотно направляется на разогретые гофровалы. В результате этого образуются волны или гофры. Изменяя форму гофровалов, можно получать разные профили гофрокартона. На верхушки полученных гофров с помощью специального клеевого вала наносится клей. Существуют несколько разновидностей современных гофропрессов:

вакуумный;

нагнетательный;

стандартныый;

кассетный.

При обработке материала на гофропрессах следует наблюдать за равномерностью натяжения бумаги и контролировать нанесение клея. Важно следить за уровнем влажности, давлением и температурой бумаги. Необходимо оперативно и своевременно менять бобины.

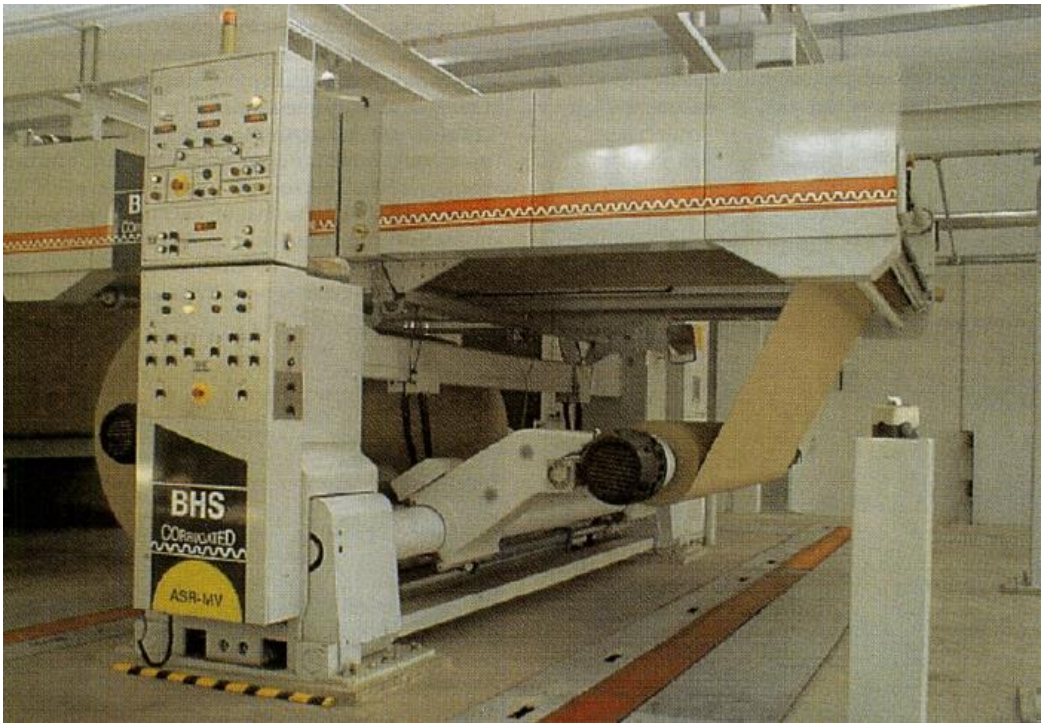


Рис. 1.6. «Мокрая» часть гофроагрегата

На завершающем этапе гофрокартон подается на сушильный стол. Тут происходит окончательное схватывание клея. В результате материал получает свою окончательную форму. На сушильном столе при помощи специальной системы выполняется прижим гофрокартона и контроль за его короблением. Это обеспечивает максимальную сцепку его элементов и высокое качество получаемого материала. При подготовке линии к работе важно следить, чтобы сушильно-охлаждающий стол был постоянно накрыт чистым гигроскопичным сукном.

Гофрокартон, изготовленный в мокрой части гофроагрегата, поступает в

сухую часть (рис. 1.7.); здесь его режут в продольном направлении, наносят рилевки или биговки в машинном направлении, режут в поперечном направлении, затем укладывают в стопы для транспортировки листов заказчику или на перерабатывающие линии, где из него производят ящики, лотки, вкладыши или витрины.

Как часть беспрерывного технологического процесса гофроагрегат должен быть оснащен системой смены заказов, обеспечивающей беспрерывную работу, и оборудованием для удаления отходов.

Ротационную резку ставят на выходе после пресса для изготовления двухслойного гофрированного картона. Она выполняет двойную функцию:

а) вырезает переваренный или непроклеенный снизу (расслоившийся) картон, получаемый при запуске машины или при возникновении каких-то проблем в мокрой части машины. Для реализации этой функции обычно устанавливают выталкиватель отходов. Устройство оснащено подъемным клапаном, направляющим бракованный картон в зазор между сверхскоростными разгонными валами, которые по очереди загружают листы в тележку для отходов. Тележка — на колесиках; вручную ее можно вывезти с линии, опорожнить, а затем поместить на место;

б) при смене заказов используют симплексную или многофункциональную систему резки.

Существует несколько типов ротационных резок: с одним зубчатым ножом, вставленным в упругую резину или полиуретановую основу (которая может занимать всю поверхность вала, а может быть представлена полоской на валу, совмещенной с ножом или опорной лентой); с одним ротационным ножом, установленным напротив контрножа (симплексная резка), или двумя ротационными ножами, режущими друг напротив друга (дуплексная резка).

Симплексные и дуплексные резки режут в соответствии с заданными настройками длины листа, управляемыми регулируемым приводом. При безостановочной системе смены заказов, описанной в данной главе ниже, на некоторых системах используют сегментные контрножи на симплексных резках, чтобы можно было выбрать траекторию реза по ширине полотна.

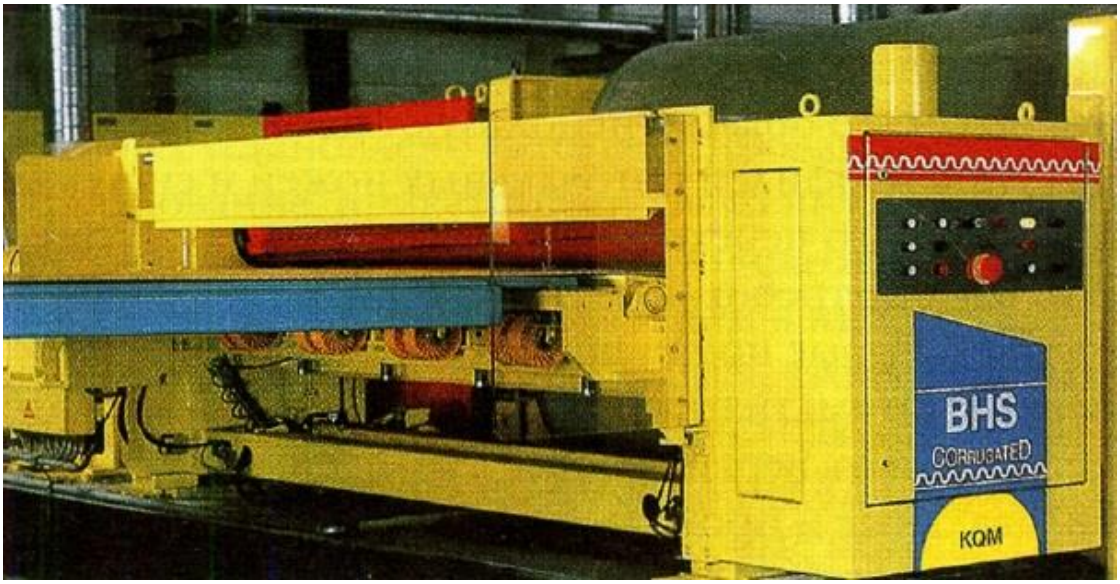


Рис. 1.7. «Сухая» часть гофроагрегата

При любом конструкторском решении нож должен быть оснащен достаточно мощным приводом, который разгоняется от нуля до скорости движения картона, режет, а также тормозит до полного останова за один оборот. Таким образом, инерция ротационного барабана с ножом, размер двигателя и электрическое управление определяют максимальную скорость картона, при которой работает резка.

Обычно не рекомендуют срезать отходы на высокой скорости. Симплексная резка рассчитана на традиционный способ смены заказа в сухой части при достаточно высокой скорости на гофроагрегате, до 250-300 м/мин или выше. Иногда в сухой части срезают несколько листов на рабочей скорости, чтобы дать время основным системам по ходу движения на мокрой части перестроиться на производство нового заказа.

На симплексной резке проще выполнить переход, чем на многофункциональной несколькими ножами. Полотно натянуто при помощи основных систем, установленных по ходу движения машины, которые вращаются с повышенной скоростью, поэтому точность настроек и регулировок ножа не настолько существенна, так как картон будет разрезан в любом случае.

Следующий рез при последовательной многофункциональной резке может зависнуть (так как на бумаге отсутствует натяжение), если нож и контрнож плохо подогнаны и находятся в плохом положении, как раз поперек полотна, или если отбракованные листы плотно зажаты и протянуты сверхскоростными протяжными валами.

Процессы, выполняемые гофроделательным агрегатом, заключаются в следующем. Рулоны бумаги и картона устанавливаются на раскаты 1 и 2. Современные бесштанговые конструкции раскатов имеют рулонодержатели с гидравлическим или механическим приводом перемещения по высоте и по ширине закрепляемого рулона (рис. 1.8.). Для обеспечения и контроля натяжения полотна при подаче его в машину используют пневматические, дисковые и электрические тормоза. Склейка концов рулонов производится в специальном узле (сплайдере). Его функция заключается в стыковке концов старого и нового рулонов вместе в процессе работы машины. Современные сплайдеры оснащены системами автоматического управления величиной натяжения с мгновенным реагированием.

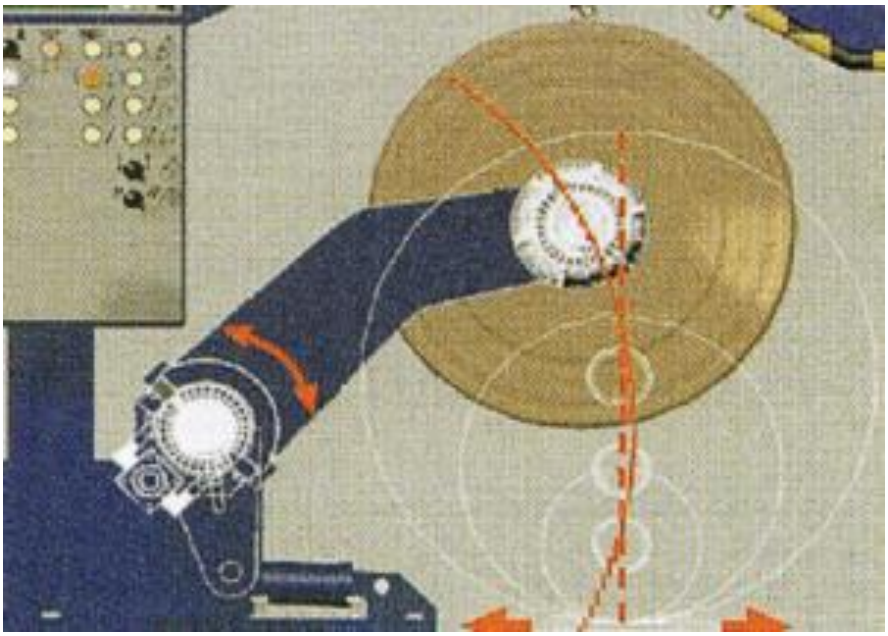


Рис. 1.8. Система загрузки рулонов

Разматываемое из рулона полотно бумаги через подогреватель 3 и увлажнитель 4 подается к нагреваемым паром рифленым валам узла гофрирования 6. Как

показала практика, увлажнение паром для большинства видов бумаги для гофрирования не требуется. Исключение составляют бумаги из полухимической древесной или сильно клееной макулатурной массы, а также лайнер, полученный из крафтцеллюлозы. Увлажнение такой бумаги с одновременным ее нагревом несколько размягчает содержащиеся в ней проклеивающие вещества и способствует улучшению проникновения клея внутрь бумаги при склеивании. Кроме того, бумага становится более эластичной, увеличивается ее способность к удлинению в процессе гофрирования и, следовательно, устраняется основная причина образования трещин. При переувлажнении бумага плохо воспринимает клей, становится рыхлой и не обеспечивает требуемую жесткость гофров. Оптимальной считается влажность бумаги перед гофрированием 7-8 %. Иногда допускается увеличение влажности до 9 %. Влажность картона для плоских слоев перед склей-кой должна быть несколько ниже влажности бумаги и не превышать 7 %.

Подогреватели для бумаги и картона представляют собой стальные барабаны диаметром 900-930 мм и длиной, превышающей рабочую ширину агрегата. Барабан-подогреватель рассчитан на нагрев поверхности паром до температуры 180-200 0С. Для регулирования степени охвата барабана полотном и в целях регулирования влажности картона, поступающего на склейку, имеются два металлических вала. Минимальный охват окружности барабана - 900, максимальный - 2700. Для увлажнения картона и бумаги перед гофрированием применяются паровые увлажнители трубчатого или камерного типа.

Бумага (флютинг), проходя между зубчатыми валами гофрировальной машины (гофропресса), приобретает волнообразный профиль. Клей при помощи клеевых валиков 5 наносится на гребни волн флютинга. Сразу после гофропресса и нанесения клея флютинг объединяется с предварительно подготовленным картоном (лайнером), образуя после склеивания двуслойный гофрокартон. Полученный двуслойный гофрокартон через накопительный мост 8 подается к клеильному устройству 5, где клей наносится на вершины гофров с другой стороны флютинга.

С отдельного раската соответственно подготовленный второй слой лайнера подается и склеивается с двухслойным гофрокартоном. Так как гофрокартон из

трех и более слоев не сгибается без деформации, термосклеивание и сушка его производится под нажимом роликов между плоской конвейерной лентой 9 и сушильными плитами 10. Далее на соответствующих устройствах обрезаются кромки, осуществляется продольная резка 11, рилевка 12 и поперечная резка 13, где готовый гофрированный картон нарезается на отдельные листы требуемой длины. Двухслойный картон можно наматывать в рулон. Листы гофрокартона с помощью приемного конвейера 14 и стопоукладчика 15 штабелируются и транспортируются на отлежку для охлаждения и окончательного схватывания клея.

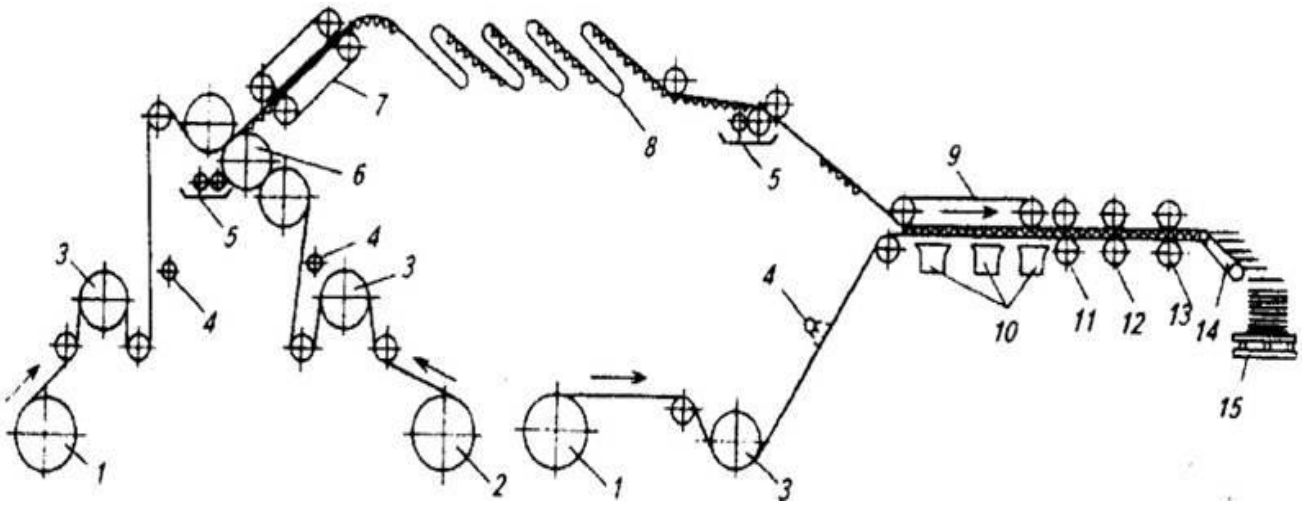


Рис. 1.9. Кинематическая схема гофроагрегата

При выполнении манипуляционных операций основные задачи сенсорного обеспечения решаются с помощью силового очувствления. Вопросы силомоментного очувствления являются актуальными при выполнении гофроагрегатом сборочных операций (вставление стержня в отверстие, резьбовые соединения и т. п.), операций по перемещению хрупких или легкодеформируемых предметов, операции механической обработки (шлифование, полировка, резка, абразивная чистка и т.п.), манипуляций с различными органами управления (рукоятками, тумблерами, кнопками, вентилями и т.д.).

Включение в контур управления гофроагегата человека-оператора позволяет автоматизировать лишь часть операций позиционно-силового управления. Этот подход позволяет последовательно наращивать степень автоматизации по мере

совершенствования сенсорной системы и отработки соответствующих алгоритмов управления.

Приведённые выше факты позволяют сделать вывод о том, что современные гофроагрегаты обладают развитыми разнообразными датчиками, имеющими различное назначение и принцип действия. Это обуславливает необходимость организации их согласованного и взаимодополняющего функционирования с одной стороны, и обеспечение совместного решения общих задач с другой.

В качестве примеров типовых задач, в которых использование мультисенсорных систем оказывается более эффективным, можно привести такие, как:

— повышение достоверности результатов измерений за счёт дублирования измерений с помощью нескольких сенсоров, использующих различные физические принципы восприятия окружающей среды;

— получение более широкого диапазона значений измеряемой величины за счёт использования различных датчиков в разных поддиапазонах измерения;

— комплексирование данных, поступающих от разных сенсорных подсистем (например от широкодиапазонного датчика с низкой точностью, и высокоточного датчика, работающего в узком диапазоне измерений);

— определение параметров объекта манипулирования или свойств внешней среды как функции от данных, формируемых несколькими датчиками;

— использование мультисенсорной информации в задаче распознавания объектов.

Комплексирование нескольких сенсорных подсистем предполагает решение ряда сложных инженерных задач. Необходимо обеспечить их совместимость друг с другом, исключить взаимное влияние (помехи) одного датчика, на другие. При разработке алгоритмов управления и обработки сенсорной информации необходимо тщательно продумать вопросы временно́го согласования (синхронизации), поскольку в большинстве случаев ожидается, что интервалы получения информации с сенсоров являются постоянными величинами.

С точки зрения упрощения аппаратной реализации и снижения общей стоимости необходимо обеспечить максимальную унификацию используемых компо-

нентов сенсорных подсистем и интерфейсов, обеспечивающих их подключение к гофроагрегату.

Используемые на практике интерфейсы можно разбить на три большие группы: аналоговые, цифровые дуплексные и цифровые сетевые. [31, 43, 66, 67, 70]

Аналоговые интерфейсы использовались с датчиками первого поколения, имевшими аналоговый выход. В их состав входят аналого-цифровые преобразователи, обеспечивающие преобразование непрерывных сигналов, формируемых сенсорами, в цифровую форму, пригодную для обработки средствами цифровой вычислительной техники.

Цифровые дуплексные интерфейсы появились позже и предназначены для датчиков следующего поколения, имеющих цифровой выход (протоколы RS-232, RS-485, ARINC 429).

В цифровых сетевых интерфейсах передача сенсорной информации осуществляется через коммуникационную сеть (Ethernet, MIL STD 1553B, CAN). Обычно такие интерфейсы используются при разработке сложных многосенсорных систем, требующих высокого быстродействия и надёжного обмена данными.

1.2.2. Компьютерные системы управления объектами

Как следует из функциональной схемы гофроагрегата, приведенной на рис. 1.12, существующие гофроагрегаты способны решать свои задачи только под непосредственным контролем человека оператора. Именно оператор принимает решение о выполнении тех или иных действий. [18, 165] Подобный принцип управления предполагает разделение системы управления на иерархические уровни, на каждом из которых решается свой определенный круг задач. Иерархические уровни и решаемые на них задачи представлены в табл. 1.1.

Таблица 1.

Уровни иерархии системы управления МР

Уровень иерархии	**Решаемые задачи**	**Принципы Реализации**
Стратегический	1) Формирование модели поведения гофроагрегата. 2) Контроль общего состояния технологического процесса, 3) Генерация совокупности/последовательности команд для реализации модели поведения при изготовлении гофрокартона. 4) Контроль за исполнением команд, внесение корректив в модель поведения	В настоящее время - диалоговый режим с человеком-оператором.

Уровень иерархии	**Решаемые задачи**	**Принципы Реализации**
Тактический	1) Получение команд со стратегического уровня. 2) Генерация генетического алгоритма реализации команд с разделением их на команды тактического уровня, реализуемые узлами и блоками. 3) Согласование и синхронизация функционирования оборудования. 4) Сбор информации от сенсоров о состоянии оборудования, подготовка сообщений, передаваемых по каналу связи человеку-оператору.	ЭВМ
Функционально-логический	1) Реализация алгоритмов управления узлами и агрегатами с замыканием обратных связей. 2) Реализация алгоритмов сжатия/распаковки данных для передачи информации о состоянии на пункт управления и получения команд с пункта управления. 3) Контроль энергетической уста-	Возможны следующие реализации: 1) Управление ЭВМ. 2) Управление с использованием сети микроконтроллеров. 3) Аналоговые контуры управления. 4) Смешанный тип

Уровень иерархии	Решаемые задачи	Принципы Реализации
	новки.	управления (часть функций реализуется ЭВМ, часть реализуется на микроконтроллерах, а часть контуров управления являются аналоговыми.

Таким образом, на верхнем, *стратегическом* уровне иерархии технологический процесс получает самую общую внешнюю команду от оператора. В соответствии с ней генерируется последовательность решаемых задач. При этом осуществляется контроль исполнения команды, и могут быть внесены коррективы в состав и последовательности задач в соответствии с обстановкой. Этот уровень соответствует уровню искусственного интеллекта. Задача интеллектуального управления к настоящему времени не решена, поэтому данный уровень обеспечивается внешним оператором, расположенным пункте управления.

На следующем, *тактическом* уровне, ЭВМ получает команды на решение задач (от внешнего оператора). Определяется генетический алгоритм, соответствующий данной команде. Алгоритм модифицируется в соответствии с текущей обстановкой и состояниями узлов и агрегатов и передается в систему управления узлами и агрегатами для ее реализации.

На *функционально-логическом* уровне происходит реализация алгоритмов управления узлами и агрегатами. При этом замыкаются обратные связи, осуществляется синхронизация работы оборудования по времени, энергетике и другим ресурсам, осуществляется обработка данных с сенсоров и т.п.

Таким образом, стратегические решения должны приниматься человеком-оператором (постановка задач), а задачи тактического уровня (конкретная ситуация - конкретное действие) и функционально-логического уровня (непосред-

ственное управление оборудованием) должны решаться автономной системой управления технологического процесса. Вариант структуры, при которой задачи функционально-логического уровня решается с помощью микроконтроллеров, приведен на рис. 1.10.

В этом случае система управления включает ряд локальных контуров, каждый из которых состоит из контроллера, привода, исполнительной (механической) части и сенсора. Вход привода, и выход сенсора, включенного в локальный контур, подключен контроллеру.

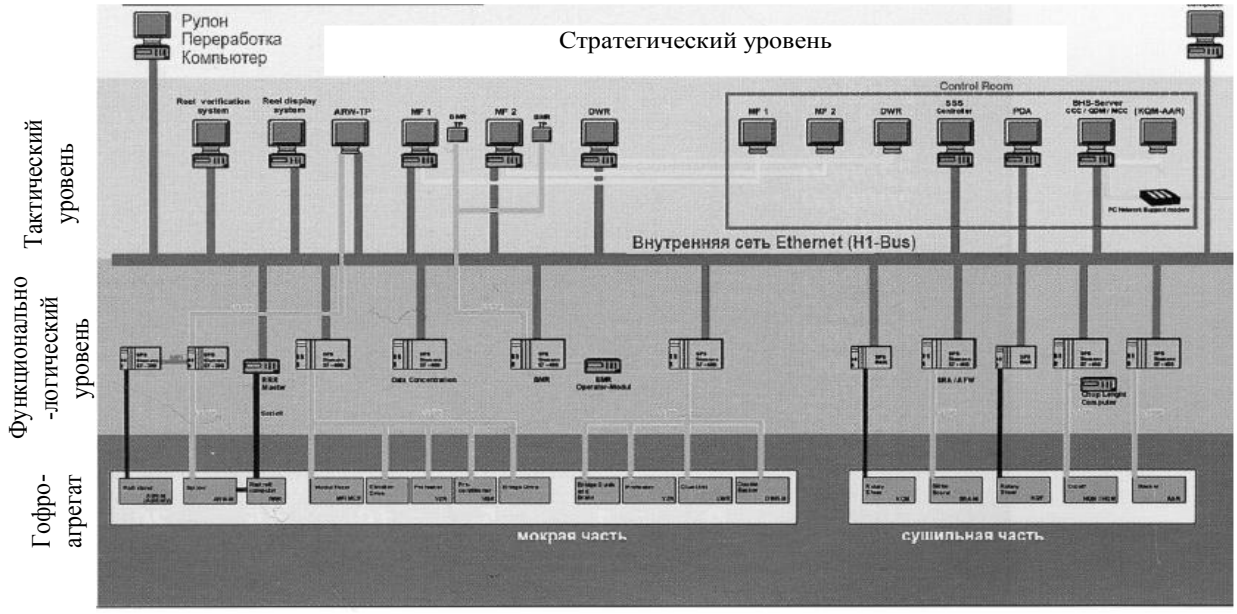


Рис. 1.10. Обобщенная схема системы управления гофроагрегата с уровнями иерархии

Таким образом, внутренняя шина гофроагрегата частично разгружается за счет того, что пара контроллеров, осуществляющих, соответственно ввод данных от сенсора и вывод управляющего воздействия данных на привод заменяется одним контроллером, осуществляющим ввод/вывод данных. Кроме того, частично разгружается процессор ЭВМ за счет того, что величина управляющего воздействия рассчитывается непосредственно для данного контура управления, минуя центральную ЭВМ.

Сжатие данных также осуществляется с помощью микроконтроллера. При этом в микроконтроллер поступает информация как от сенсоров контроля окружающий обстановки, так и от тепловых сенсоров. Сжатые данные через шину передаются на ЭВМ, и затем через аппаратуру передачи данных на пункт управления.

Схема реализации локальных аналоговых контуров управления приведена на рис. 1.11.

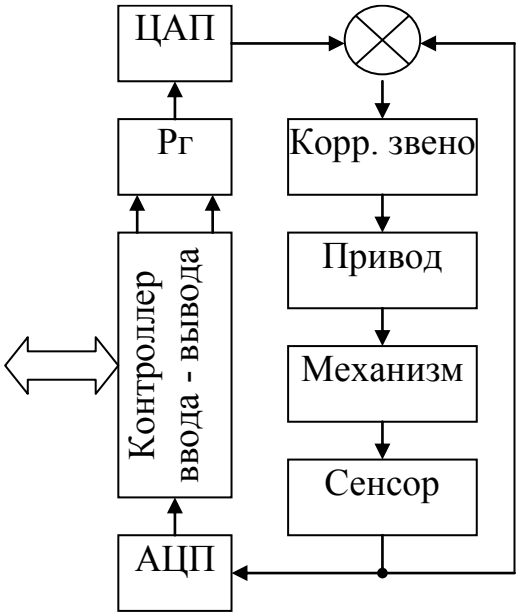


Рис. 1.11. Схема локального аналогового контура управления

В локальный аналоговый контур управления входят привод, механизм, сенсор, регистр, цифро-аналоговый преобразователь (ЦАП), аналого-цифровой преобразователь (АЦП) и контроллер ввода-вывода. Структура представляет собой классическую систему управления с обратной связью и работает следующим образом. Цифровой код, соответствующий требуемому значению выходного параметра исполнительного механизма (скорость, перемещение рабочего органа), вводится через контроллер в регистр Рг и удерживается там до следующего ввода требуемого значения. Выходы регистра подключены ко входам цифро-налогового преобразователя, и при появлении цифрового кода на выходе ЦАП с временной

задержкой, определяемой только быстродействием аналоговой части ЦАП, появляется выходная величина, соответствующая удерживаемому коду в регистре.

Сигнал с выхода цифро-аналогового преобразователя сравнивается с сигналом сенсора, при этом формируется разностный сигнал, который корректируется в соответствии с выбранным при проектировании законом управления. Коррекция сигнала осуществляется на физическом уровне, без временной задержки, что соответствует общему принципу обработки сигналов на физическом уровне. Скорректированный аналоговый сигнал отрабатывается приводом, в результате чего перемещается исполнительный механизм. Это перемещение измеряется с помощью сенсора. Сигнал сенсора, с одной стороны, подается на схему сравнения в аналоговом виде, а с другой стороны преобразуется в цифровую форму. Цифровой код, в свою очередь, может быть считан через контроль ввода-вывода по интерфейсу, соединяющему контур управления с ЭВМ.

Таким образом, с одной стороны осуществляется управление состоянием механизма без управляющей ЭВМ, а с другой стороны, осуществляется информационный обмен между контуром управления и компьютером, что необходимо для определения текущего состояния технологического процесса.

Рациональное разделение системы управления на иерархические уровни, выбор способа реализации управления на нижнем, функционально-логическом и тактическом уровнях является инженерной задачей, к настоящему времени не решенной. Методология создания надежных иерархически разделенных систем управления должна опираться на фундаментальный математический аппарат, отражающий физические явления и процессы, протекающие в системах управления и в самом агрегате, как объекте управления. [20, 21 29, 53, 5682, 83, 84]

1.2.3. Особенности работы программного обеспечения

Структура технологического процесса может быть представлена в виде множества единиц оборудования, управляемого ЭВМ, или автономным контроллером. Каждая единица оборудования в совокупности с ЭВМ, или автономным

контроллером, формирует контур управления, включающий объект управления, сенсор, привод и счетно-решающий прибор (ЭВМ или автономный контроллер в случае цифрового управления, или корректирующее звено в случае аналогового управления), через который замыкается обратная связь. Вследствие того, что контроль оборудования осуществляется непрерывно, процесс управления может быть представлен в виде множества циклограмм обработки данных на ЭВМ с последующим исполнением команд механической частью технологического процесса. Типовая циклограмма обработки данных при цифровом управлении показана на рис. 1.12. [33, 48, 62, 133, 134]

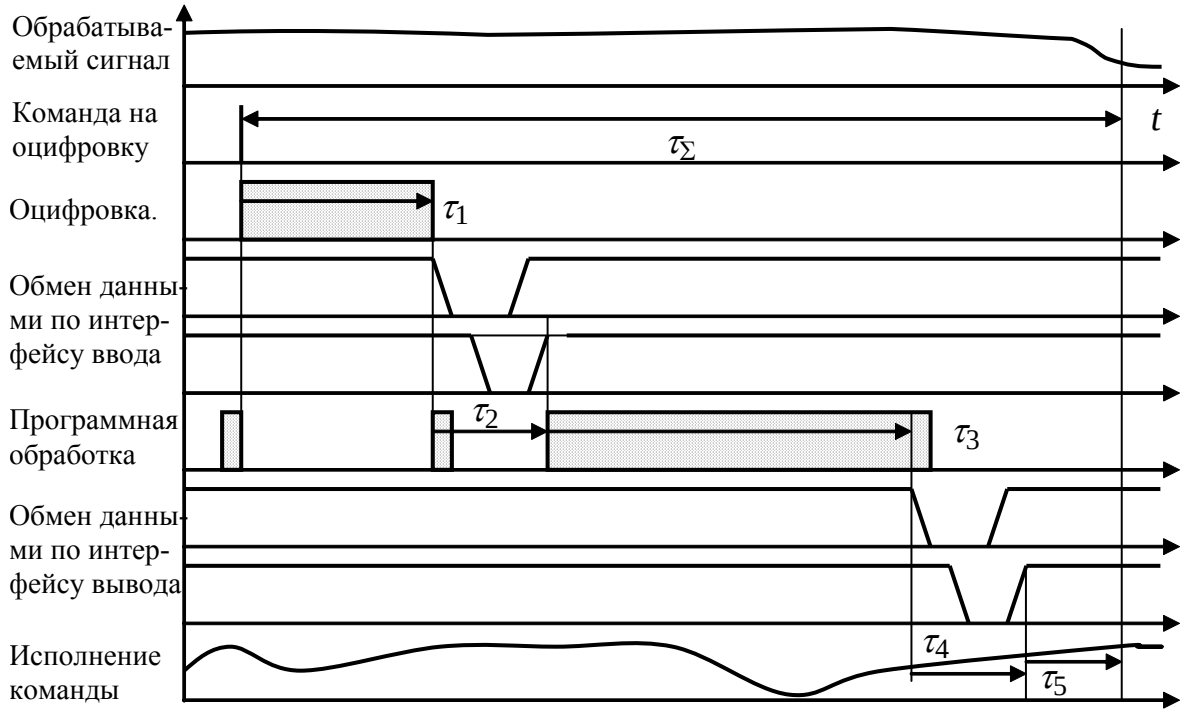


Рис. 1.12. Типовая циклограмма цифрового управления узлами и блоками технологического процесса

Типовой цикл включает последовательное выполнение следующих операций:

оцифровка аналогового сигнала, поступающего с сенсора (временная задержка τ_1);

ввод цифрового кода в ЭВМ через контроллер обмена данными (временная задержка τ_2);

обработка данных по соответствующему алгоритму управления (временная задержка τ_3);

вывод результатов в контроллер привода (временная задержка τ_4);

исполнение команды приводом (временная задержка τ_5).

Рассмотрим в отдельности каждый из временных интервалов из множества $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$.

τ_1 - время оцифровки сигнала является детерминированным и определяется техническими характеристиками аналого-цифрового преобразователя;

τ_2 - время обмена данными по интерфейсу ввода, в системах с квитированием является случайным и зависит от текущего физического состояния линий интерфейса, температуры, влажности;

τ_3 - время программной обработки, случайно [3, 11, 92, 102,];

τ_4 - время обмена данными по интерфейсу вывода, в системах с квитированием является случайным;

τ_5 - время формирования силового сигнала зависит от многих факторов, в том числе и от текущего момента сопротивления на исполнительном механизме, температуры, влажности и т.п. факторов.

При аналоговом управлении циклограммы принимают вид, показанный на рис. 1.13. Согласно данной циклограмме в аналоговом контуре развиваются три независимых процесса:

процесс ввода управляющего воздействия в регистр Рг (τ_1);

процесс опроса состояния сенсора (τ_2);

процесс оцифровки сигнала сенсора (τ_3).

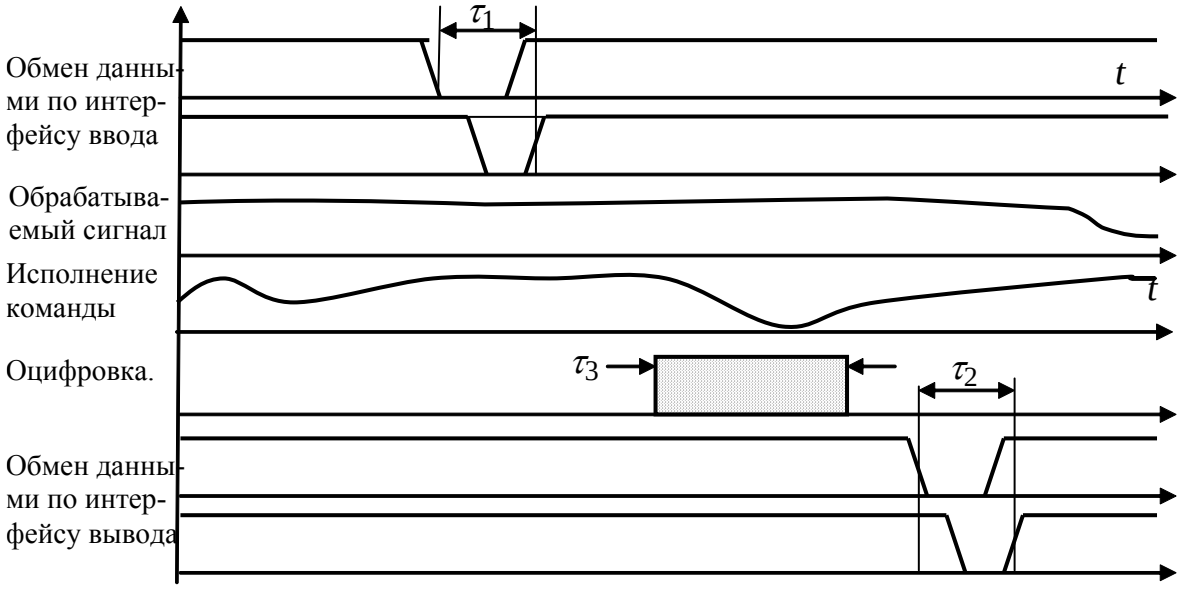


Рис. 1.13. Типовая циклограмма аналогового управления

Случайность времени обмена по интерфейсам ввода-вывода зависит от текущего физического состояния линий интерфейса, температуры, влажности и т.п. факторов. Этот временной интервал не оказывает влияния на характер управления, и определяет только время занятости общей шины при обмене. Время преобразования аналогового сигнала сенсора в цифровой код определяется типом преобразователя. Вследствие того, что ввод данных в ЭВМ в данном случае реализует только функцию мониторинга состояния агрегата, этот фактор также оказывает малое влияние на качество его функционирования.

1.2.4. Случайность времени обработки данных

Для подтверждения случайности времени программной обработки данных рассмотрим простейший детерминированный алгоритм обработки данных (рис. 1.14). При реализации циклограммы вводится и обрабатывается случайная величина X, распределенная по закону $f(x)$ с область. определения $-1 \le x \le 1$. [2, 15, 29, 53] Расчет функции y производится в формате с фиксированной точкой, в частности, в регистрах общего назначения, либо за четыре такта машинного времени (при $x \le 0$) с вероятностью $\int_{-1}^{0} f(x)dx$, либо за двадцать тактов машинного времени

(при $x > 0$) с вероятностью $\int_0^1 f(x)dx$.

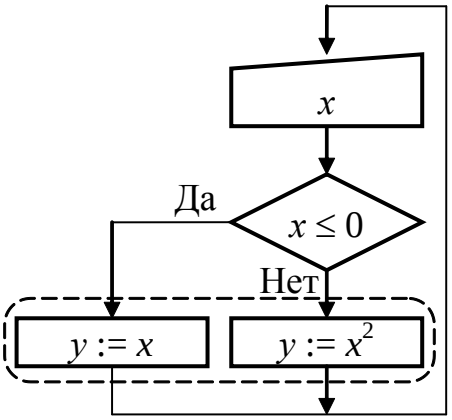


Рис. 1.14. К вопросу о случайности времени обработки

Таким образом, время расчета величины y (состояние циклограммы обведено пунктирной линией) является случайной величиной, при этом источником случайности являются обрабатываемые данные. Если рассматривать вычисление каждой функции $y := x$ и $y := x^2$ как одно из состояний циклограммы, то можно сделать вывод, что попадание в каждое из этих состояний также случайно. Вероятность попадания в каждое состояние определяется плотностью распределения величины X и условиями, определенными в состоянии принятия решения.

Таким образом, каждая из приведенных на рис. 1.14 задержек, включая задержку на обработку данных по детерминированному алгоритму, характеризуется случайным временем (время аналого-цифрового преобразования можно рассматривать как случайное, определяемое вырожденным законом распределения $\delta(t - \tau_1)$), а следовательно и общее время цикла каждого цикла управления является случайным. При имеющейся возможности переключения из текущего состояния циклограммы в одно из сопряженных с ним, это переключение осуществляется для внешнего наблюдателя случайным образом, что предполагает стохастический характер последовательности переключений.

1.2.5. Особенности работы человека-оператора

Функционирование человека-оператора, управляющего гофроагрегатом, производится в соответствии с информационной моделью, приведенной на рис. 1.15. Информация, необходимая для принятия решений, формируется гофроагрегатом и передается через аппаратуру передачи данных на пункт управления. Кроме того, к человеку-оператору поступают команды управления с более высокой степенью иерархии, направленные на корректировку действий, или самого человека оператора [90, 98, 99, 101, 105, 108, 109, 111, 112].

Первичная информация содержится в общем задании на решение целевой задачи технологического процесса, предъявляется человеку-оператору на экране монитора (воспроизведение окружающей обстановки, состояние гофроагрегата), транспарантах, приборах и т.п. Эта информация может быть представлена как множество индикаторов $И_1$, ..., $И_n$, ..., $И_N$, которые доводят до человека-оператора некоторую информацию о состоянии гофроагрегата и окружающей среды. Деятельность оператора включает следующие этапы.

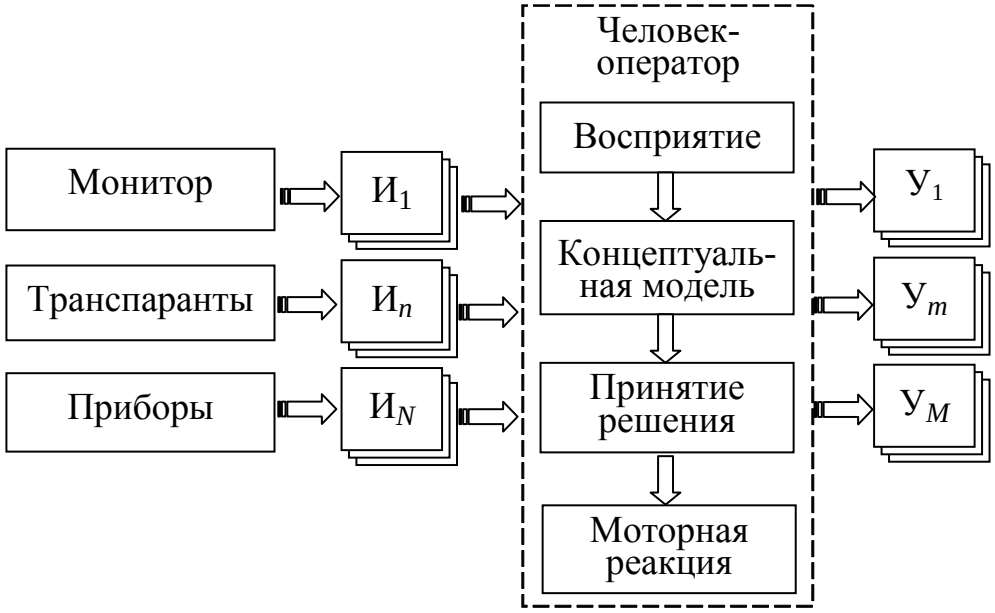


Рис. 1.15. Информационная модель работы человека-оператора

Восприятие информации, заключается в обнаружении в общем потоке визуальной, слуховой и т.п. информации отдельных фрагментов, отвечающих стоящей перед человеком-оператором задаче; ознакомления с выделенными признаками и опознания объекта восприятия. Для осуществления этого этапа необходимо, чтобы все данные о состоянии гофроагрегата, как на приборах, так и на экране монитора представлялись в форме, удобной для восприятия.

Оценка информации, ее анализ и обобщение на основе заранее заданных или сформированных в процессе обучения критериев, производится путем сопоставления воспринятой информационной модели со сложившейся у оператора внутренней образно-концептуальной моделью состояния.

Принятие решения о действиях, акт, формируемый на основе проведенного анализа информационной и образно-концептуальной моделей обстановки. На этом этапе человек-оператор, на основании воспринятой информации, опыта предыдущего управления, и стоящей перед оператором задачей принимает решения по управлению гофроагрегата.

Исполнение принятого решения выражается в моторной реакции, т.е. выполнении определенного действия по нажатию кнопки, клавиши, педали, перемещению джойстика и т.п.

Как следует из анализа действий оператора за пультом управления [3, 131, 135, 136,] они характеризуются тремя аспектами:

деятельность обучаемого оператора также может рассматриваться как конгломерат взаимодействующих элементарных процессов восприятия, формирования концептуальной модели, принятия решения и моторной реакции по исполнению принятого решения;

элементарный процесс сводится к целенаправленной последовательности действий, каждое из которых можно рассматривать как состояние субъекта, развивающееся во времени, переход из одного состояния в другое для внешнего наблюдателя является случайным, хотя для самого человека-оператора он представляется достаточно логичным;

для каждого из действий последовательности можно определить временной интервал, и вероятность перехода к другому действию данной последовательности, временной интервал является случайным и зависит от многих факторов, в том числе состояния и обученности человека-оператора, дополнительных нерелевантных воздействий и т.п.

Таким образом, человек-оператор при управлении гофроагрегатом оперирует по алгоритму, идентичному алгоритму функционирования управляющей ЭВМ, а следовательно, при моделировании его действий может быть применен тот же самый математический аппарат.

1.3. Общий принцип реализации команд цифровой системой управления

Независимо от выбора конфигурации системы управления оборудованием во всех процессах управления участвует ЭВМ/контроллер фон-неймановского типа. [122, 137, 143] Обобщенная функциональная схема включения ЭВМ в контур управления приведена на рис. 1.16, где показаны:

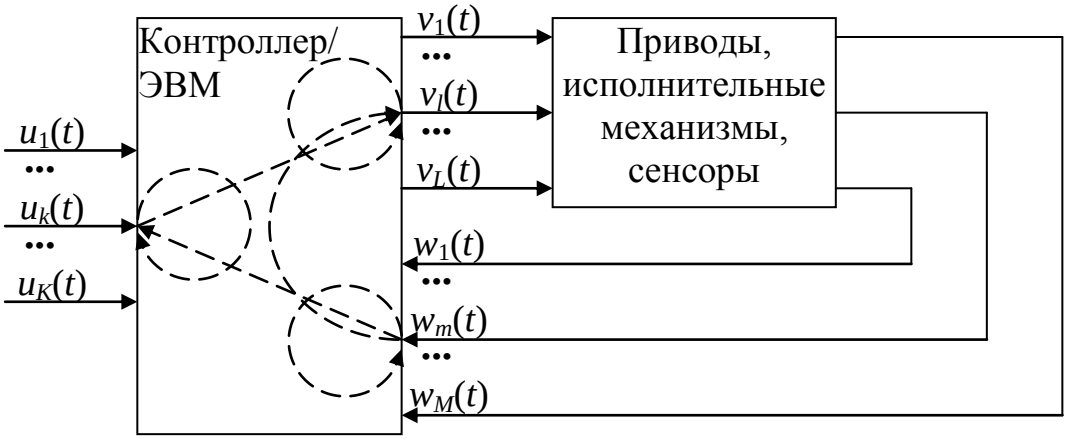


Рис. 1.16. Типовая функциональная схема
реализации команд системой управления

$\boldsymbol{u} = [u_1(t), ..., u_k(t), ..., u_K(t)]^\theta$ - вектор опорных сигналов, определяющих значение регулируемых параметров (могут формироваться как внутри контроллера, так и поступать извне);

$\boldsymbol{v} = [v_1(t), ..., v_l(t), ..., v_L(t)]^\theta$ - вектор управляющих сигналов подаваемых на приводы исполнительных механизмов и движителей;

$\boldsymbol{w} = [w_1(t), ..., w_m(t), ..., w_M(t)]^\theta$ - вектор сигналов с сенсоров обратной связи, определяющих состояние МР.

ЭВМ (контроллер) интерпретирует управляющий алгоритм, который организует опрос перечисленных сигналов в соответствии с логикой функционирования гофроагрегата. Рассмотрим вопрос о временных интервалах, которые нужно обеспечивать для достижения поставленной цели управления.

1.3.1. Учет требований теоремы Котельникова

Объединим векторы ***u, v, w*** в единый вектор сигналов, определяющих состояние технологического процесса:

$$\boldsymbol{s}(t) = \boldsymbol{u}(t) \cup \boldsymbol{v}(t) \cup \boldsymbol{w}(t) = [s_1(t), ..., s_n(t), ..., s_N(t)]^\theta, \qquad (1.1)$$

где θ - операция транспонирования; $N = K + L + M$.

Для обработки на ЭВМ сигналы вектора $\boldsymbol{s}(t)$ подвергаются дискретизации и квантованию по уровню [41, 51, 60, 89]. В первом приближении любой физический прибор, дискретизирующий составляющие вектора $\boldsymbol{s}(t)$ может быть представлен как идеальный мультипликативный дискретизатор, в котором преобразование многомерного сигнала выражается как произведение вектора $\boldsymbol{s}(t)$ на вектор-строку дискретизирующиз функций $\sigma(t)$:

$$\hat{\boldsymbol{s}}(t)=\boldsymbol{s}(t)\otimes\boldsymbol{c}(t)=\begin{pmatrix} s_1(t)\sum\limits_{k=-\infty}^{\infty}\delta(t-kT_1) \\ s_n(t)\sum\limits_{k=-\infty}^{\infty}\delta(t-kT_n) \\ s_N(t)\sum\limits_{k=-\infty}^{\infty}\delta(t-kT_m) \end{pmatrix}=\begin{pmatrix} \hat{s}_1(t) \\ \hat{s}_n(t) \\ \hat{s}_N(t) \end{pmatrix}, \qquad (1.2)$$

где $\boldsymbol{c}(t)=\left[\sum\limits_{i=-\infty}^{\infty}\delta(t-iT_1),\ \ ...\ \ \sum\limits_{i=-\infty}^{\infty}\delta(t-iT_n),\ \ ...\ \ \sum\limits_{i=-\infty}^{\infty}\delta(t-iT_N)\right]^{\theta}$ - вектор дискретизирующих функций; $\hat{\boldsymbol{s}}(t)=[\hat{s}_1(t),...,\hat{s}_n(t),...,\hat{s}_N(t)]^{\theta}$ - вектор дискретных сигналов; T_n - n-й период дискретизации, $1 \le n \le N$; $\delta(...)$ - δ-функция Дирака.

Вид функции $\hat{s}_n(t)$, получающейся в результате дискретизации функции $s_i(t)$, приведен на рис. 1.17.

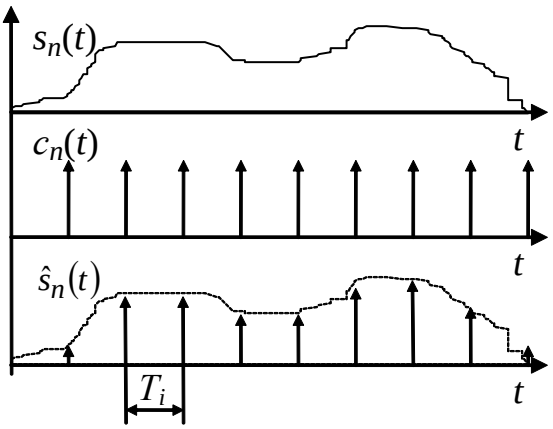


Рис. 1.17. Дискретизация сигнала идеальным дискретизатором

Рассмотрим условия, которым должны удовлетворять $\boldsymbol{s}(t)$ и $\boldsymbol{c}(t)$. Для этого определим Фурье-спектр дискретного сигнала как

$$\hat{\boldsymbol{S}}(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\hat{\boldsymbol{s}}(t)\exp(-j\omega t)dt=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\boldsymbol{s}(t)\otimes\boldsymbol{c}(t)\exp(-j\omega t)dt=\boldsymbol{S}(\omega)*\boldsymbol{C}(\omega), \qquad (1.3)$$

где * - знак операции свертки; $j=\sqrt{-1}$..

Вектор $\boldsymbol{C}(\omega)$ имеет вид

$$\boldsymbol{C}(\omega)=[\tilde{N}_1(\omega),...,\tilde{N}_n(\omega),...,\tilde{N}_N(\omega)]^{\theta}=\frac{1}{\sqrt{2\pi}}\times$$

$$=\times\int_{-\infty}^{\infty}\left(\sum_{i=-\infty}^{\infty}\delta(t-iT_1),\quad ...\quad \sum_{i=-\infty}^{\infty}\delta(t-iT_n),\quad ...\quad \sum_{i=-\infty}^{\infty}\delta(t-iT_N)\right)^{\theta}\exp(-j\omega t)dt. \qquad (1.4)$$

Для определения вектора $\boldsymbol{C}(\omega)$ представим дискретизирующую функцию как бесконечную последовательность прямоугольных импульсов

$$c_n(t)=\begin{cases}\tau_n^{-1} & \text{when } iT_n-\tau_n/2\le t<iT_n+\tau_n/2;\\ 0 & \text{when } iT_n+\tau_i/2\le t<(i+1)T_n-\tau_n/2,\end{cases} \qquad (1.5)$$

где τ_n - ширина импульса. T_n - период следования импульсов.

Спектральная плотность для несмещенного импульса прямоугольной формы вида

$$c_{n0}=\begin{cases}\tau_n^{-1} & \text{ïðè } -\tau_n/2\le t\le\tau_n/2;\\ 0 & \text{ïðè } t<-\tau_n/2,\ t>\infty.\end{cases}$$

определяется как

$$\tilde{N}_{n0}(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\frac{\tau_n}{2}}^{\frac{\tau_n}{2}}\frac{1}{\tau_n}\exp(-j\omega t)dt=\frac{\sin\omega\frac{\tau_n}{2}}{\sqrt{2\pi}\omega\tau_n}.$$

Спектральная плотность последовательности прямоугольных импульсов определяется в виде

$$\tilde{N}_n(\omega)=\frac{\sin\omega\frac{\tau_n}{2}}{\sqrt{2\pi}\omega\tau_n}\sum_{i=-\infty}^{\infty}\exp(j\omega iT_n). \qquad (1/6)$$

При $\tau_n\to 0$ выражение преобразуется в следующее:

$$\tilde{N}_n(\omega)=\sqrt{\frac{2}{\pi}}\sum_{i=-\infty}^{\infty}\exp(j\omega iT_n) \qquad (1.7)$$

Выражение (1.7), в свою очередь, можно рассматривать как сумму гармоник с периодами iT_n, i = 1, ±1, ±2, ±3, ...

Таким образом,

$$\boldsymbol{C}(\omega)=\left(\sum_{i=-\infty}^{\infty}\delta(\omega-i\Omega_1),\quad \ldots\quad \sum_{i=-\infty}^{\infty}\delta(\omega-i\Omega_n),\quad \ldots\quad \sum_{i=-\infty}^{\infty}\delta(\omega-i\Omega_N)\right), \qquad (1.8)$$

$$\Omega_n T_n = 2\pi, \qquad (1.9)$$

где T_n - период дискретизации *n*-го сигнала вектора $\boldsymbol{s}(t)$.

Таким образом, из (1.3) и (1.8) следует, что спектральные характеристики вектора сигналов $\hat{\boldsymbol{s}}(t)$ имеют вид:

$$\hat{\boldsymbol{S}}(\omega)=\left(\sum_{i=-\infty}^{\infty}s_1(\omega-i\Omega_1),\quad \ldots\quad \sum_{i=-\infty}^{\infty}s(\omega-i\Omega_n),\quad \ldots\quad \sum_{i=-\infty}^{\infty}s(\omega-i\Omega_N)\right). \qquad (1.10)$$

Это, в свою очередь, означает, что спектры размножаются по оси частот со сдвигом на величину Ω_n. Из (1.10) следует, что если сигнал $s_n(t)$ имеет область ненулевых значений спектра $|\omega|\le\frac{\Omega}{2}$ и восстанавливается с помощью идеального восстанавливающего фильтра, имеющего область ненулевых значения $|\omega|\le\frac{\Omega}{2}$, то дискретизация сигнала производится без потерь. (теорема Котельникова). На практике спектры сигналов $s_1(t),\ldots,s_n(t),\ldots,s_N(t)$ имеют бесконечные области ненулевых значений, поэтому даже при идеальном восстанавливающем фильтре возникают ошибки дискретизации. Эти ошибки влияют на качественные характеристики управления узлами и блоками агрегата. Величина ошибки может быть оценена как

$$\varepsilon_n=\left[\int_{-\infty}^{-\frac{\Omega_n}{2}}|S_n(\omega)|d\omega+\int_{\frac{\Omega_n}{2}}^{\infty}|S_n(\omega)|d\omega\right]+\sum_{\substack{i=-\infty\\ i\neq 0}}^{\infty}\int_{-\frac{\Omega_n}{2}}^{\frac{\Omega_n}{2}}|S_n(\omega-i\Omega_n)|d\omega, \qquad (1.11)$$

где часть, заключенная в квадратные скобки, оценивает долю спектра несмещенной спектральной характеристики, попавшую за пределы восстанавливающего фильтра; часть под знаком суммы оценивает долю спектров смещенных спектральных характеристик, попавших внутрь восстанавливающего фильтра.

Вследствие того, что транзакции на прием/передачу данных к бортовой ЭВМ формируются программно, временной интервал, обеспечивающий величину

ошибки не более, чем определено зависимостью (1.11), должен выдерживаться при интерпретации алгоритма контроллером.

1.3.2. Учет времени запаздывания

Рассмотрим контур управления, в котором обратная связь замыкается через ЭВМ или цифровой контроллер (см. рис. 1.16). Выделим среди множества контуров управления *l*-й контур и построим его структурную схему (рис. 1.18). Пусть в выделенном контуре объект управления является линейным, а программа ЭВМ реализует линейный закон управления. [59, 97, 158, 159] На рис. 1.18:

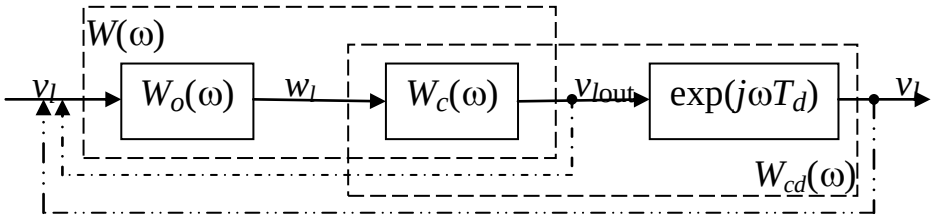


Рис. 1.18. Выделенный контур управления

линейная передаточная функция объекта управления представлена в виде

$$W_o(\omega) = \frac{w_l}{v_l}, \tag{1.12}$$

где ω - круговая частота;

линейный закон управления, реализованный в программе ЭВМ, имеет вид

$$W_{\tilde{n}}(\omega) = \frac{w_{l\text{out}}}{w_l}; \tag{1.13}$$

линейный закон управления с учетом чистого запаздывания, вносимого управляющей ЭВМ за счет последовательной интерпретации операторов алгоритма управления, может быть представлен в виде

$$W_{cd}(\omega) = \frac{v_l}{w_l} = = W_c(\omega) \cdot \exp(j\omega T_d), \tag{1.14}$$

где T_d - время чистого запаздывания.

Передаточная функция от входа объекта управления до выхода ЭВМ без учета запаздывания имеет вид

$$W(\omega)=\frac{v_{l\text{out}}}{v_l}=A(\omega)\cdot\exp\left[j\varphi(\omega)\right], \qquad (1.15)$$

где $A(\omega)=|W(\omega)|$ - амплитудная частотная характеристика передаточной функции;

$\varphi(\omega)=\arccos\frac{\mathrm{Im}[W(\omega)]}{A(\omega)}$ - фазово-частотная характеристика.

Вид типовой амплитудной и фазовой частотных характеристик приведен на рис. 1.19.

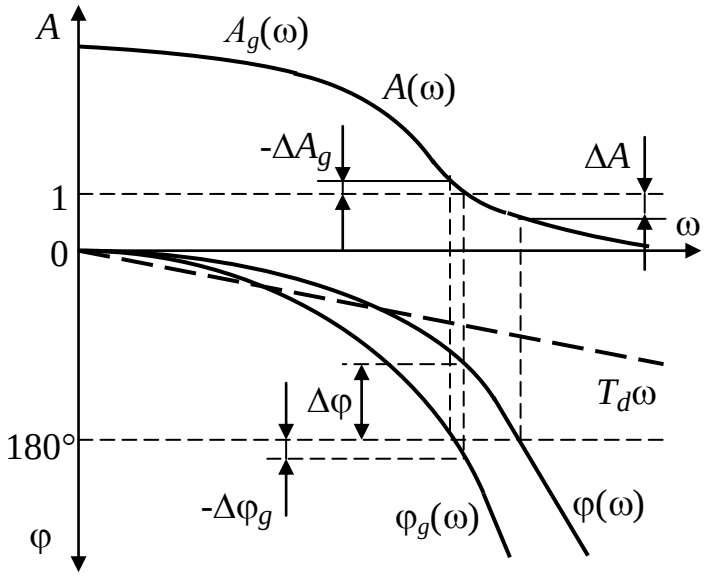


Рис. 1.19. Амплитудная и фазовая частотные характеристики контура управления без запаздывания (φ) и с запаздыванием (φ_g)

На характеристиках имеется две особых точки, по которым оценивается устойчивость системы управления, это точка пересечения амплитудной характеристикой $A(\omega)$ прямой $A=1$ и точка пересечения фазовой характеристикой $\varphi(\omega)$ прямой $\varphi=180^\circ$.

По координате ω точки пересечения прямой $A=1$ и амплитудно-частотной характеристики $A(\omega)$ может быть определен запас по фазе $\Delta\varphi$ разомкнутой системы без запаздывания. По координате ω точки пересечения прямой $\varphi=180^\circ$ и

фазово-частотной характеристики $\varphi(\omega)$ может быть определен запас по амплитуде ΔA разомкнутой системы без запаздывания.

Предположим, что линейная система без запаздывания является устойчивой и запасы по амплитуде и фазе существуют. Добавим в контур управления чистое запаздывание, создаваемое работой контроллера Фон Неймановского типа.

Звено с чистым запаздыванием имеет коэффициент передачи, равный единице во всей области частот, а фазово-частотная характеристика, в принятой на рис. 1.46 системе координат, имеет вид наклонной прямой. Коэффициент наклона определяется временем чистого запаздывания. Таким образом, амплитудная характеристика системы не меняется, т.е. $A_g(\omega) = A(\omega)$, а сдвиг по фазе увеличивается, т.е.

$$\varphi_d(\omega) = \varphi_d(\omega) + T_d\omega. \qquad (1.16)$$

В связи с этим возможна ситуация, когда запас по амплитуде и запас по фазе становятся отрицательными, и равными, соответственно $-\Delta A_g$ и $-\Delta\varphi_g$.

Приведенные выкладки показывают, что при проектировании контуров управления оборудованием необходимо в каждом контуре учитывать время обработки данных на ЭВМ и закладывать запас по амплитуде и фазе с учетом случайного времени запаздывания. По результатам анализа может быть построена матрица временных задержек вида:

$$\boldsymbol{T}_g = \begin{pmatrix} T_{d(1,1)} & \dots & T_{d(1,l)} & \dots & T_{d(1,L)} \\ & & \dots & & \\ T_{d(k,1)} & \dots & T_{d(k,l)} & \dots & T_{d(k,L)} \\ & & \dots & & \\ T_{d(K,1)} & \dots & T_{d(K,l)} & \dots & T_{d(K,L)} \\ T_{d(K+1,1)} & \dots & T_{d(K+1,l)} & \dots & T_{d(K+1,L)} \\ & & \dots & & \\ T_{d(K+m,1)} & \dots & T_{d(K+m,l)} & \dots & T_{d(K+m,L)} \\ & & \dots & & \\ T_{d(K+M,1)} & \dots & T_{d(K+M,l)} & \dots & T_{d(K+M,L)} \end{pmatrix}. \qquad (1.17)$$

Матрица имеет размеры $L \times (K + M)$. Строки с первой по *K*-ю матрицы описывают случайное время задержки при обработке опорных сигналов вектора ***u***

= $[u_1(t), ..., u_k(t), ..., u_K(t)]^{\theta}$. Строки с ($K$+1)-й по ($K$+$M$)-ю описывают случайное время задержки при сигналов обратной связи вектора $\boldsymbol{w} = [w_1(t), ..., w_m(t), ..., w_M(t)]^{\theta}$. Подобная сложная структура матрицы случайных задержек времени обработки данных на ЭВМ порождает проблему временных согласований сигналов управления.

1.3.3. Шум полинга и перекос данных

Одним из возможных способов ввода/вывода данных из ЭВМ является полинг, т.е. ввод в основную управляющую программу операторов ввода/вывода. Выше было показано, что любые временные интервалы, генерируемые программно, являются случайными. Случайность временного интервала опроса сенсоров, или приводов приводит к тому, что истинный момент опроса смещается относительно теоретического момента на некоторый интервал, который является случайным. [123]

В ЭВМ Фон Неймановского типа, при опросе периферийных устройств, реализуется принцип: один запрос - одно слово данных. Это порождает т.н. «перекос данных», т.е. смещение моментов опроса элементов вектора данных относительно теоретического момента опроса вектора. Названная ситуация приведена на рис. 1.20, где показаны: s_n - n-е вводимые/выводимые данные; c_n - n-е запрсы на ввод/вывод данных; t_d - теоретические моменты запросов; t_{d1}, t_{d2} - реальные моменты запросов, обозначенные жирной штриховой линией.

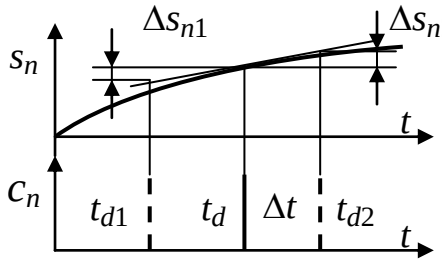


Рис. 1.20. Перекос данных

Смещение времени запроса порождает изменение вводимых данных. Величина изменения определяется как

$$\Delta s_n = \frac{ds_n}{dt}\Delta t, \tag{1.18}$$

где Δt - время отклонения.

Как следует из (1.18) величина отклонения зависит от производной от сигнала s_n и времени отклонения. Сигнал s_n является случайной функцией по определения, интервал Δt также является случайной величиной. Поэтому появление случайного фактора в виде интервала Δt порождает шум полинга, или шум перекоса данных. Этот шум является индивидуальным для каждой величины s_n.. Сигнал может быть представлен в виде:

$$\boldsymbol{s}_d(t) = \boldsymbol{s}(t) + r_d(t), \tag{1.19}$$

где $r_d(t)$ - вектор шума полинга; Δt_n - случайное время перекоса, индивидуальное для каждого обрабатываемого сигнала;

$$r_d(t) = \begin{pmatrix} \frac{ds_1(t)}{dt}\Delta t_n \\ \cdots \\ \frac{ds_n(t)}{dt}\Delta t_n \\ \cdots \\ \frac{ds_N(t)}{dt}\Delta t_N \end{pmatrix}. \tag{1.20}$$

1.3.4. Обоснование выбора теории полумарковских процессов для моделирования и оптимизации циклограмм управления гофроагрегата

Как показано выше, цифровое управление объектами различной сложности порождает весьма высокие требования к временной вычислительной сложности алгоритмов управления, реализующих циклограммы управления гофроагрега. Под вычислительной сложностью в данном случае понимается промежуток времени между появлением на объекте ситуации, требующей адекватного реагирова-

ния со стороны управляющей ЭВМ и выдачей соответствующего возникшей ситуации управляющего воздействия на объект. В свою очередь, временная вычислительная сложность реализаций управляющих алгоритмов определяется целым рядом факторов, к которым, кроме банальных архитектуры ЭВМ, частоты тактирования и структуры команд, относятся: операционная среда, дисциплина диспетчеризации, организация транзакций, математические модели процессов для которых строятся управляющие алгоритмы и т.п.

Гофроагрегат относится к категории сложных технических систем, управляемых ЭВМ. Проектирование систем указанного класса осуществляется на нескольких последовательных уровнях. В частности к ним относят

структурный, на котором ведется укрупненный синтез гофроагрегата, выбирается архитектура, определяется состав аппаратных средств и синтезируются связи между узлами и блоками системы;

функционально-логический, на котором решаются вопросы распределения функций между основными компонентами системы;

алгоритмический, на котором решаются общие семантические вопросы синтеза циклограмм управления отдельными узлами и блоками, а также гофроагрегатом в целом;

организационный, на котором решается вопрос организации взаимодействия узлов и блоков в рамках реализации глобальной циклограммы управления технологическим процессом в целом.

На более низких уровнях происходит формирование конкретных инженерно-конструкторских решений, реализуемых в программных кодах.

Наибольшей значимостью и степенью влияния на функциональные возможности и технические характеристики технологического процесс обладают структурный, функционально-логический и организационный уровни данной иерархии. Это связано с тем, что решения, принимаемые на указанных уровнях, в первую очередь определяют общее свойства создаваемой системы, а также направление и характер работ на всех последующих этапах проектирования.

В общем случае, тактические задачи, которые ставятся перед гофроагрегатом, сводятся к тому, чтобы из текущего состояния за счет реализации одной из циклограмм, заложенных в ЭВМ, переключиться в требуемое состояние за заданный промежуток времени. Циклограммам, управляемым техническим средствам и человеку-оператору присуще свойство «недетерминированности», т.е. относительной свободы выбора продолжения процесса при достижении поставленной цели. Этот факт выражается в вероятностном переходе от одного состояния субъекта к другому при сохранении общей цели функционирования гофроагрегата. Возможности перехода из одного состояния к другому определяются структурой модели, а основным фактором, определяющим эффективность перехода, является фактор времени. Таим образом, модели должны быть структурно-параметрическими.

Существует два вида структурно-параметрических моделей: модели в виде взвешенных графов с детерминированными значениями весов, и модели в виде графов со случайными значениями весов. Из последних, при структурно-параметрическом анализе наибольшее распространение получили полумарковские процессы. Применение полумарковских процессов для моделирования функционирования технологического процесса при выполнении команд оператора вытекает из наличия в циклограммах множества ярко выраженных состояний $A = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\}$, например, состояний обработки информации или обмена данными, состояния передачи данных через контроллер на исполнительный механизм и т.п., а также случайности времени пребывания в указанных состояниях, определяемой плотностью распределения $f_{j(a)}(t)$, и стохастическим характером $p_{j(a),1(a)}, ..., p_{j(a),j(a)}, ..., p_{j(a),J(a)}$ переключения из текущего состояния $a_{j(a)}$ в сопряженные состояния $a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}$.

Классический анализ полумарковских процессов сводится к решению следующих задач:

определение плотности распределения времени пребывания процесса в подмножестве состояний;

определение плотности распределения времени достижения некоторого подмножества состояний из состояний другого подмножества;

определение вероятности достижения некоторого подмножества состояний из состояний другого подмножества в течение некоторого, наперед заданного интервала времени.

Следует отметить тот факт, что классическая теория полумарковских процессов при решении перечисленных задач предполагает исследование одного процесса. Поэтому с использованием методов указанной теории достаточно просто могут быть построены и исследованы циклограммы, описывающие управление отдельными узлами и блоками структуры. Однако в указанной структуре имеется как минимум два субъекта, а именно человек-оператор и гофроагрегат, которые функционируют одновременно и параллельно, каждый по своей циклограмме. Кроме того, на пункте управления существует диалоговая ЭВМ, функционирующая параллельно с человеком-оператором, а на гофроагрегате располагается множество единиц оборудования, каждая из которых управляется своей циклограммой. Таким образом, математическая модель гофроагрегата может строиться как единый параллельный полумарковский процесс, учитывающий весь комплекс особенностей функционирования технологического процесса. В литературе подобные процессы исследованы недостаточно, что и объясняет целесообразность проведения исследований в данном направлении.

1.4. Выводы

1) На основании обзора существующих и перспективных гофроагрегатов, сформулирована общая задача проектирования систем управления гофроагрегата, как систем, обеспечивающих реализацию оптимальных циклограмм с заданными свойствами.

2) Разработана обобщенная функциональная схема технологического процесса и показано, что указанный тип технических средств включает ЭВМ, управляемые систему, целевое оборудование, контролер, и средства связи с пунктом

управления; в свою очередь, пункт управления содержит диалоговую ЭВМ, средства управления и индикации, а также рабочее место человека-оператора.

3) Проведен анализ циклограмм и показано, что время пребывания в состояниях циклограмм является случайной величиной, определенной с точностью до плотности распределения, а переключение из текущего в сопряженные состояния для внешнего наблюдателя также является случайным процессом, определенным с точностью до вероятностей переключения.

4) Для данных условий функционирования сделан вывод о том, что адекватным формализмом для аналитического описания последовательностей смены состояний в узлах и блоках гофроагрегата является полумарковский процесс.

5) Проведен анализ работы человека-оператора при управлении гофроагрегатом и показано, что основные операции, восприятия информации, создания концептуальной модели состояния технологического процесса, принятия решения и исполнения решения в виде моторной реакции воздействия на орган управления идентичны операциям, выполняемым управляющей циклограммой, а следовательно могут быть представлены в виде полумарковской модели.

6) Показано, что исследование циклограмм сводится к решению задачи определения временного интервала и вероятности достижения одного подмножества состояний из другого подмножества состояний.

2. ИСПОЛЬЗОВАНИЕ ТЕОРИИ ПОЛУМАРКОВСКИХ ПРОЦЕССОВ ДЛЯ МОДЕЛИРОВАНИЯ ЦИКЛОГРАММ УПРАВЛЕНИЯ ГОФРОАГРЕГАТОМ

2.0. Введение

Как показано в разделе 1, любой гофроагрегат представляет собой сложный комплекс, в который входят технические средства, обеспечивающие:

энергетическая установка, трансмиссия, движители;

наблюдение за ходом изготовления гофрокартона;

контроль уровня температуры;

решение целевых задач;

управление оборудованием (ЭВМ или сеть контроллеров);

связь с пунктом управления;

Каждая из названных функций сводится к выполнению последовательность действий, которые разворачиваются во времени. Каждое действие может быть охарактеризовано случайным временным интервалом, измеряемым от начала действия до его окончания, а также случайным переходом к выполнению другого действия последовательности, если имеется альтернатива продолжения. Целью моделирования может быть определение временных и вероятностных характеристик выполнения последовательности действий. В силу перечисленных особенностей функционирования бортового оборудования для его моделирования использовать теорию полумарковских процессов [1, 7, 8, 76, 87, 92, 94, 140, 166].

2.1. Модель технологического процесса как ординарный полумарковский процесс

Определим вероятностное пространство борелевской тройкой

$$\mathrm{B} = (\Omega, \Theta, P), \qquad (2.1)$$

где $\Omega = \{\omega_1, ..., \omega_n, ..., \omega_N\}$ - множество элементарных событий; Θ носитель алгебры с сигнатурой $\{\cup, \cap, \bar{}\}$, включающей операции объединения $\cup$, пересечения $\cup$ и дополнения $\bar{}$; P - вероятностная мера [23, 24, 25, 26].

Множество элементарных событий может быть представлено в виде объединения непересекающихся подмножеств

$$\Omega = \Omega^a \cup \Omega^t; \ \Omega^a \cap \Omega^t = \varnothing, \tag{2.2}$$

где $\Omega^a = \left\{\omega^a_{1(a)}, ..., \omega^a_{j(a)}, ..., \omega^a_{J(a)}\right\}$ - счетное дискретное подмножество элементарных событий, формирующих состояния технологического процесса; $\Omega^t = \left\lfloor\omega^t_{1(t)}, ..., \omega^t_{j(t)}, ...\right\rfloor$ - бесконечное упорядоченное подмножество элементарных событий, формирующих временные интервалы, составляющее континуум; $\varnothing$ - пустое множество.

Под состоянием

$$a_{j(a)} = \alpha\left(\omega^a_{j(a)}\right) \tag{2.3}$$

понимается выполнение оборудованием технологического процесса некоторого действия. В этом состоянии гофроагрегат пребывает от начала выполнения действия до его окончания. Функция $\alpha\left(\omega^a_{j(a)}\right)$ от элементарного события является дискретной одноместной и взаимно однозначной, такой, что одному элементарному событию подмножества $\omega^a_{j(a)}$ соответствует одно состояние $a_{j(a)}$, и наоборот, одному состоянию $a_{j(a)}$ соответствует одно элементарное событие $\omega^a_{j(a)}$.

Применение функции (2.3) к множеству Ω^a дает *множество* состояний технологического процесса;

$$\alpha\left(\Omega^a\right) = z\left\{\omega^a_{1(a)}, ..., \omega^a_{j(a)}, ..., \omega^a_{J(a)}\right\} = \left\{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\right\} = A. \tag{2.4}$$

где $a_{j(a)}$ - $j(a)$-е физическое состояние технологического процесса; $J(a)$.- мощность множества Ω^a.

Подмножеству Ω^a ставится в соответствие вероятностная мера

$$p_{j(a)} = P\lfloor a : a_{j(a)} \in A \rfloor, \tag{2.5}$$

которая характеризует вероятность пребывания процесса в одном из состояний множества A для внешнего по отношению к технологическому процессу наблюдателю. Тот факт, что система может находиться в одном и только в одном из состояний, накладывает следующее ограничение на вероятности (2.5):

$$\sum_{j(a)=1(a)}^{J(a)} p_{j(a)} = 1. \tag{2.6}$$

Определим событие как смену состояния, или переключение из $a_{j(a)}$ в $a_{n(a)}$. Множество возможных переключений может быть получено путем возведения A во вторую декартову степень:

$$\begin{gathered}(A)^2 = \left\{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\right\}^2 = \\ = \{\{(a_{1(a)}, a_{1(a)}), ..., (a_{1(a)}, a_{n(a)}), ..., (a_{1(a)}, a_{J(a)})\}, \\ ...,\{(a_{j(a)}, a_{1(a)}), ..., (a_{j(a)}, a_{n(a)}), ..., (a_{k(a)}, a_{J(a)})\}, \\ ...,\{(a_{J(a)}, a_{1(a)}), ..., (a_{J(a)}, a_{n(a)}), ..., (a_{J(a)}, a_{J(a)})\}\} = S.\end{gathered} \tag{2.7}$$

Назовем кортеж

$$s_{j(a),n(a)} = \lfloor a_{j(a)}, a_{n(a)} \rfloor \in S \tag{2.8}$$

переключением из состояния $a_{j(a)}$ в состояние $a_{n(a)}$. Каждой паре $\lfloor a_{j(a)}, a_{n(a)} \rfloor$, $1(a) \le j(a), n(a) \le J(a)$ из (2.7) может быть поставлена в соответствие вероятностная мера

$$p_{k(a),j(a)} = P\left\lfloor s\left(a^-, a^+\right): \; s_{j(a),n(a)} = \left[a_{j(a)}, a_{n(a)}\right] \in S \right\rfloor, \tag{2.9}$$

где a^- - состояние технологического процесса до переключения; a^+ - состояние технологического процесса после переключения.

Разделим множество состояния технологического процесса на два непересекающихся подмножества

$$A = E \cup \overline{E}, \tag{2.10}$$

где E - подмножество поглощающих состояний; $\overline{E}$ - подмножество непоглощающих состояний.

Для вероятностной меры (2.9) справедливо следующее выражение:

$$\sum_{n(a)=1(a)}^{J(a)} p_{j(a),n(a)} = \begin{cases} 1, \text{ when } a_{j(a)} \in \overline{E}; \\ 0, \text{ when a } _{j(a)} \in E. \end{cases} \quad (2.11)$$

Выражение (2.11) означает, что переключение из непоглощающих состояний образует полную группу несовместных событий, а переключение из поглощающих состояний невозможно.

Рассмотрим континуум Ω^t. свяжем это бесконечное упорядоченное множество с физическим временем. Введем функцию

$$t_{j(t)} = \tau\left\lfloor \omega^t_{j(t)} \right\rfloor, \quad (2.12)$$

и свяжем эту функцию с реальным физическим временем. В частности $t_{j(t)}$ - это момент времени, соответствующий элементарному событию $\omega^t_{j(t)}$. Функция (2.12) является континуальной, одноместной и взаимно однозначной, такой, что одному элементарному событию $\omega^t_{j(t)}$ соответствует один момент времени $t_{j(t)}$, и наоборот, одному моменту времени $t_{j(t)}$ соответствует одно элементарное событие $\omega^t_{j(t)}$. Применение зависимости (2.12) к континууму Ω^t дает

$$t = \tau\left(\Omega^t\right), \quad (2.13)$$

где t - физическое время.

Величина t имеет следующие особенности;

в контексте задачи моделирования состояний технологического процесса началом отсчета времени каждый является момент смены состояний технологического процесса;

время отсчитывается от одного события переключения состояний технологического процесса до другого события переключения состояний технологического процесса (рис. 2.1);

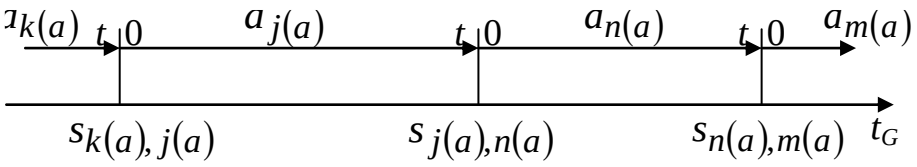


Рис. 2.1. Формирование фактора времени, t_G - глобальное время

с учетом условия о начале отсчета времени и правила его отсчета, на время накладывается ограничение $t \geq 0$.

В соответствии с особенностями формирования временных интервалов можно утверждать, что последовательность событий переключения состояний технологического процесса формируется поток временных интервалов, каждый из которых является случайной непрерывной величиной. В этом случае для $j(t)$-го момента времени $j(a)$-го интервала потока, если известно, что следующим переключением будет $s_{j(a),n(a)}$, может быть определена вероятностная мера

$$f_{j(a),n(a)}(t_{j(t)})dt = P\lfloor t : t(s_{k(a),j(a)}) = 0, t(s_{j(a),n(a)}) = t_{j(t)} \rfloor, \qquad (2.14)$$

где $t(s_{k(a),j(a)})$ - момент переключения в состояние $a_{j(a)}$; $t(s_{j(a),n(a)})$ - момент переключение из состояния $a_{j(a)}$, если принято решение, что следующим состоянием будет $a_{n(a)}$.

Вследствие того, что момент $t_{j(t)}$ был выбран произвольно, (2.14) может быть преобразовано следующим образом:

$$f_{j(a),n(a)}(t)dt = P\lfloor t : t(s_{k(a),j(a)}) = 0, t(s_{j(a),n(a)}) = t \rfloor, \qquad (2.15)$$

где $f_{j(a),n(a)}(t)$ - плотность распределения времени пребывания в состоянии $a_{j(a)}$ с последующим переключением в состояние $a_{n(a)}$.

Вследствие того, что всегда, при всех переключениях, $t(s_{k(a),j(a)}) = 0$, плотность распределения не зависит от предыстории переключений. Вследствие того, что $t(s_{i(a),n(a)}) \neq 0$, плотность распределения, в общем случае, зависит от того, в какое состояние технологического процесса переключится следующий раз.

Обычно рассматриваться поток случайных временных интервалов $\tau(\omega^t)$, который обладает особенно простыми свойствами:

стационарностью, если вероятность попадания того или иного числа интервалов на участок времени длиной θ зависит только от длины этого участка и не зависит от того, где именно на оси t расположен этот участок;

отсутствием последействия, которое трактуется как независимость временного интервала, оставшегося до очередного переключения состояния от начала наблюдения процесса в глобальном времени t_G.

ординарностью, если вероятность попадания на элементарный участок двух или более временных интервалов $\tau(\omega^t)$ пренебрежимо мала по сравнению с вероятностью попадания одного из них.

Если поток переключений обладает всеми тремя свойствами, то он называется стационарным пуассоновским потоком[26, 167, 178, 179, 249, 259]. Для такого потока число временных интервалов, попадающих на любой фиксированный интервал времени, будет распределено по закону Пуассона:

$$P_m = \frac{\lambda^m}{m!}\exp(-\lambda_{j(a),j(n)}), \tag{2.16}$$

где $\lambda_{j(a),n(a)}$ - параметр закона Пуассона; m - целочисленный параметр, $m \in \{1, 2, 3, ...\}$.

Плотность распределения значений временных интервалов $\tau(\omega^t)$ для подобного потока определяется в виде:

$$f_{j(a),n(a)}(t) = 1(t)\lambda_{j(a),n(a)}\exp\left[-\lambda_{j(a),n(a)}t\right], \tag{2.17}$$

где $\lambda_{j(a),n(a)}$ - плотность потока переключений; $1(t)$ - единичная функция Хевисайда.

Очевидно, что если на временные интервалы $\tau(\omega^t)$ не накладываются ограничения по отсутствию последействия, то они могут быть распределены по произвольному закону, отличному от (2.17). При этом единственным ограничением, накладываемым на плотность распределения произвольного закона, является то,

что его область определения лежит в положительной полуплоскости, а сама функция - в первом квадранте, т.е.

$$f(t)\begin{cases}\geq 0, \text{ when } 0 \leq t_{\min} \leq t \leq t_{\max}; \\ = 0, \text{ when } t < 0;\end{cases} \quad \int_{t\min}^{t\max} f(t)dt = 1. \qquad (2.18)$$

На верхний предел области ненулевых значений плотности распределения $t_{\max}$ никаких дополнительных ограничений, кроме (2.18), не накладывается. Для некоторых физических объектов возможна ситуация, когда $t_{\max} \to \infty$.

Такой процесс, в котором отсутствуют требования к пуассоновскому характеру потока переключений, называется полумарковским процессом [76, 87, 94]. Ввиду отсутствия ограничений (2.17), класс систем, для моделирования которых может быть применен полумарковский процесс существенно расширяется. В частности, В этот класс попадают и системы цифрового управления технологического процесса, что показано в разделе 1.

2.2. Определение и способы задания полумарковского процесса

2.2.1. Определение полумарковского процесса

Как следует из вышеизложенного, полумарковский процесс полностью определяется как [34, 42, 46, 78 174, 175]

$$\mu = \{A, \boldsymbol{r}, \boldsymbol{h}(t)\}, \qquad (2.19)$$

где A - множество состояний, совпадающее с множеством состояний технологического процесса; $\boldsymbol{r}$ - матрица смежности, определяющая множество связей между состояниями; $\boldsymbol{h}(t)$ - полумарковская матрица $\boldsymbol{h}(t)$, представляющая собой прямое (поэлементное) произведение матрицы вероятностей $\boldsymbol{p}$ и матрицы $\boldsymbol{f}(t)$ плосостей распределения

$$\boldsymbol{h}(t) = [h_{j(a),n(a)}(t)] = \boldsymbol{p} \otimes \boldsymbol{f}(t) = [p_{j(a),n(a)} \cdot f_{j(a),n(a)}(t)], \qquad (2.20)$$

$$\boldsymbol{p} = [p_{j(a),n(a)}]; \qquad (2.21)$$

$$\boldsymbol{f}(t) = [f_{j(a),n(a)}(t)]; \qquad (2.22)$$

$$A = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\}; \tag{2.23}$$

$$\boldsymbol{r} = [r_{j(a),n(a)}]; \tag{2.24}$$

$$r_{j(a),n(a)} = \begin{cases} 1, \text{ when } p_{j(a),n(a)} \neq 0; \\ 0, \text{ otherwise.} \end{cases} \tag{2.25}$$

В контексте решаемой задачи элемент $h_{j(a),n(a)}(t)$ полумарковской матрицы (2.10) представляет собой взвешенную плотность распределения пребывания технологического процесса в состоянии $a_{j(a)}$ с последующим переключением в состояние $a_{n(a)}$.

Следует отметить, что если переключение $s_{j(a),n(a)} = [a_{j(a)}, a_{n(a)}]$ является невозможным событием, то

$$p_{k(a),j(a)} = P\left\{s\left(a^-, a^+\right): s_{j(a),n(a)} = \left[a_{j(a)}, a_{n(a)}\right] \in S\right\} = 0;$$

$$f_{j(a),n(a)}(t) = \lim_{\tau \to \infty} \delta(t - \tau). \tag{2.26}$$

Из (2.20) следует, что матрицы стохастическая и плотностей распределения могут быть получены в соответствии со следующими выражениями:

$$\boldsymbol{p} = \int_0^\infty \boldsymbol{h}(t) dt; \tag{2.27}$$

$$\boldsymbol{f}(t) = \left[\frac{h_{j(a),n(a)}(t)}{p_{j(a),n(a)}} \right]. \tag{2.28}$$

Полумарковский процесс кроме стохастической матрицы может быть охарактеризован следующие числовыми характеристиками, наиболее часто используемыми в теории вероятностей:

математическими ожиданиями времени пребывания в состояниях множества A:

$$\boldsymbol{T} = \int_0^\infty t\boldsymbol{f}(t) dt = \left\| T_{j(a),n(a)} \right\|; \tag{2.29}$$

дисперсиями времени пребывания в состояниях множества A

$$\boldsymbol{D} = \int_0^{\infty} t^2 \boldsymbol{f}(t)dt - \boldsymbol{T} \otimes \boldsymbol{T} = \lfloor D_{j(a),n(a)} \rfloor. \qquad (2.30)$$

2.2.2. Представление полумарковского процесса в виде взвешенного графа

Наглядно процесс (2.19) может быть изображен в виде ориентированного графа, показанного на рис. 2.2.

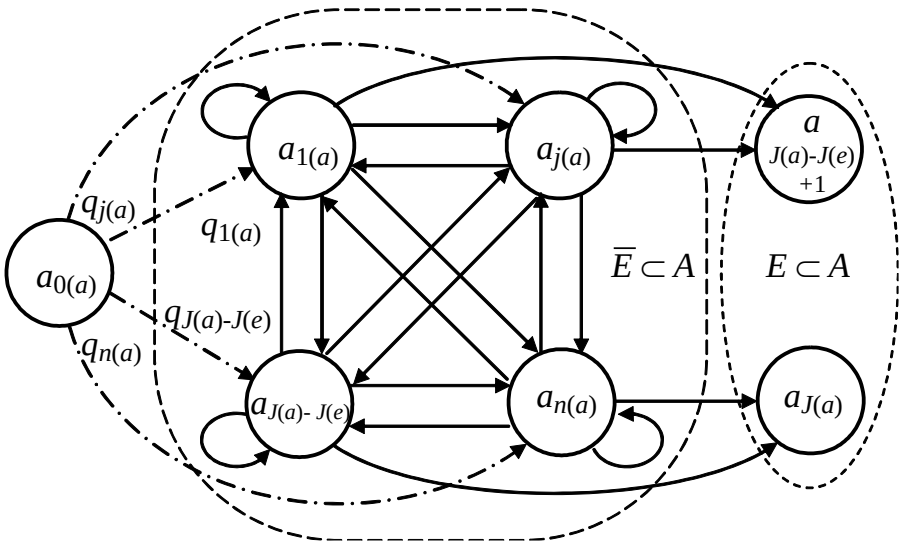


Рис. 2.2. Полумарковский процесс, моделирующий алгоритм с оператором «конец»

В соответствии с (2.10) множество A состояний полумарковского процесса разделено на два непересекающихся подмножества: подмножества непоглощающих состояний $\overline{E}$ и подмножество поглощающих состояний E. Для удобства состояния подмножества $\overline{E}$ имеют индексы, начиная с первого и оканчивая ($J(a)$-$J(e)$)-м. Состояния подмножества E имеют индексы, начиная с ($J(a)$-$J(e)$ +1)-го и оканчивая $J(a)$-м, т.е.

$$E = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)-J(e)}\};$$
$$E = \{a_{J(a)-J(e)+1}, ..., a_{j(e)}, ..., a_{J(a)}\}. \qquad (2.31)$$

Вследствие подобного разделения матрица смежности $\boldsymbol{r}$, полумарковская матрица $\boldsymbol{h}(t)$ и стохастическая матрица $\boldsymbol{p}$ имеют структуру, в которой строки с

($J(a)$-$J(e)$ +1)-й по $J(a)$-ю являются нулевыми [54, 104, 110].

Полумарковский процесс (2.19) обладает одним существенным свойством, которое вытекает из свойства реальных систем управления технологического процесса, а именно из любого состояния подмножества $\overline{E}$ существует хотя бы один путь в одно из состояний подмножества E.

Справедливо следующее утверждение.

Утверждение 2.1. Безусловная плотность распределения времени пребывания полумарковского процесса в непоглощающих состояниях $a_{j(a)}$ до его переключения в состояния $a_{n(a)} \in O\lfloor a_{j(a)} \rfloor$, где $O\lfloor a_{j(a)} \rfloor$ - выходная функция состояния $a_{n(a)}$, т.е. множество состояний, в которые можно попасть из $a_{j(a)}$, определяется зависимостью

$$f_{j(a)}(t) = \sum_{n(a)=1(a)}^{J(a)} h_{j(a),n(a)}(t). \qquad (2.32)$$

Доказательство, Действительно, переключения из $a_{j(a)}$ в одно из состояний подмножества $O\lfloor a_{j(a)} \rfloor$ составляет полную группу независимых несовместных событий, поэтому вероятности указанных переключений суммируются, что и дает зависимость (2.32).

2.2.3. Блуждания по полумарковской цепи

Последовательность смены состояний технологического процесса для внешнего наблюдателя может быть представлена как блуждание по полумарковской цепи (рис. 2.3) [86, 87].

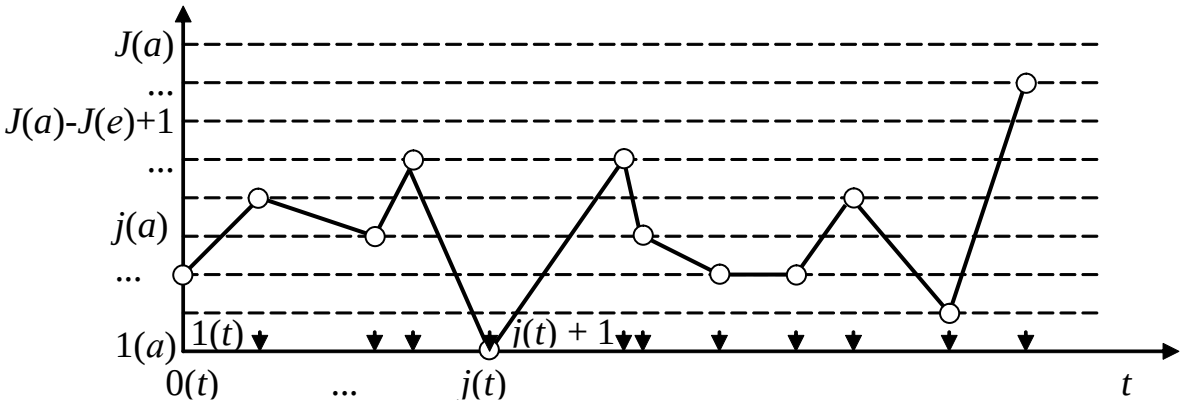


Рис. 2.3. Блуждания по полумарковской цепи

При блужданиях процесс пребывает в состоянии $a_{j(a)}$ в течение случайного времени, а затем с вероятностью $p_{j(a),n(a)}$ переключается в состояние $a_{n(a)}$. Элемент $h_{j(a),n(a)}(t)$ полумарковской матрицы определяет временные и вероятностные характеристики между двумя переключениями. Время, в течение которого процесс пребывает в состоянии $a_{j(a)}$, определено с точностью до плотности распределения $f_{j(a),n(a)}(t)$[55, 57]. Состояния, в которые последовательно попадает процесс при блужданиях, ниже будет называться траекторией блуждания. Очевидно, что для каждой реализации полумарковского процесса траектория блуждания детерминирована и строго определяется логикой управления технологическим процессом. Для внешнего же, по отношению к процессу, наблюдателя каждая конкретная траектория реализации является случайной.

Блуждания по полумарковскому процессу имеют начало. При этом, поскольку не существует никаких ограничений на старт процесса из состояний множества *A*, целесообразно ввести стохастическую меру начала переключений в полумарковском процессе. Начало процесса определяется вектором вероятностей

$$\boldsymbol{q} = [q_{j(a)}]. \tag{2.32}$$

где

$$\sum_{j(a)=1(a)}^{J(a)} q_{j(a)} = 1;\ q_{j(a)} = 0 \text{ при } a_{j(a)} \in E. \tag{2.33}$$

Введение вектора $\boldsymbol{q}$ означает, что в процессе появляется «нулевое», или стартовое состояние $a_{0(a)}$. При этом формируется полумарковский процесс (рис. 2.2, штрихпунктирные линии)

$$\mu = \{A, \boldsymbol{r}, \boldsymbol{h}(t)\} \rightarrow \mu' = \{A', \boldsymbol{r}', \boldsymbol{h}'(t)\}. \quad (2.34)$$

В полумарковском процессе μ':

$$A' = \{a_{0(a)}, a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)}\}; \quad (2.35)$$

$$\boldsymbol{r}'(t) = \left[r'_{j(a),n(a)}\right] = \begin{pmatrix} 0 & r_{0(a),1(a)} & \cdots & r_{0(a),n(a)} & \cdots & r_{0(a),J(a)} \\ 0 & r_{1(a),1(a)} & \cdots & r_{1(a),n(a)} & \cdots & r_{1(a),J(a)} \\ & & & \cdots & & \\ 0 & r_{j(a),1(a)} & \cdots & r_{j(a),n(a)} & \cdots & r_{j(a),J(a)} \\ & & & \cdots & & \\ 0 & r_{J(a),1(a)} & \cdots & r_{J(a),n(a)} & \cdots & r_{J(a),J(a)} \end{pmatrix}; \quad (2.36)$$

$$r_{0(a),n(a)} = \begin{cases} 0, \text{ если } q_{n(a)} = 0; \\ 1, \text{ если } q_{n(a)} \neq 0. \end{cases}$$

$$\boldsymbol{h}'(t) = \begin{pmatrix} 0 & q_{1(a)}\delta(t) & \cdots & q_{n(a)}\delta(t) & \cdots & q_{J(a)}\delta(t) \\ 0 & h_{1(a),1(a)}(t) & \cdots & h_{1(a),n(a)}(t) & \cdots & h_{1(a),J(a)}(t) \\ & & & \cdots & & \\ 0 & h_{j(a),1(a)}(t) & \cdots & h_{j(a),n(a)}(t) & \cdots & h_{j(a),J(a)}(t) \\ & & & \cdots & & \\ 0 & h_{J(a),1(a)}(t) & \cdots & h_{J(a),n(a)}(t) & \cdots & h_{J(a),J(a)}(t) \end{pmatrix}, \quad (2.37)$$

где $\delta(t)$ - δ-функция Дирака.

Очевидно, что в полумарковском процессе $h'(t)$, определяемом матрицей (2.21), множество состояний $A' = \{a_{0(a)}, a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)},\}$ разбивается на три класса:

$$A' = B \cup \bar{E} \cup E, \quad (2.38)$$

где $B = \{a_{0(a)}\}$; $\bar{E} = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)-J(e)}\}$; $E = \{a_{J(a)-J(e)+1}, ..., a_{j(e)}, ..., a_{J(a)}\}$.

Отметим, что вследствие ограничения (2.33) существует хотя бы один путь из состояния $a_{0(a)}$. в состояния подмножества $\bar{E}$. Потребуем, чтобы из состояния $a_{0(a)}$ существовал хотя бы один путь в любое из состояний подмножества $\bar{E}$. Тогда из состояния $a_{0(a)}$ существует хотя бы один путь, ведущий в одно из состояний

подмножества E через любое состояние подмножества $E = \{a_{1(a)}, ..., a_{j(a)}, ..., a_{J(a)\text{-}J(e)}\}$. Это, в свою очередь, означает, что в правильно спроектированном алгоритме отсутствуют операторы или циклы, изолированные от других операторов, а значит, все состояния алгоритма являются актуальными, т.е. вносят свой вклад во временные и вероятностные характеристики циклограммы.

2.3. Модель циклограммы как полумарковский процесс

2.3.1. Полумарковская матрица и структура циклограммы

При реальном управлении узлами и блоками технологического процесса реализуется принцип обратной связи (см. раздел 1)[132, 133, 134]. Реализация указанного принципа сводится к периодическому контролю управляющей подсистемой состояния узлов и блоков платформы [125, 135]. Это означает тот факт, что после переключения полумарковского процесса в состояния подмножества E происходит переключение с вероятностью, равной единице и за время, определяемое вырожденным законом распределения с нулевым математическим ожиданием, в состояние $a_{0(a)}$. Из состояния $a_{0(a)}$ за время, определяемое вырожденным законом распределения с нулевым математическим ожиданием процесс переключается в одно из состояний подмножества $\bar{E}$ с вероятностями, определяемыми вектором $\boldsymbol{q}$. Таким образом, формируется циклограмма управления. Вид типовой циклограммы приведен на рис. 2.5.

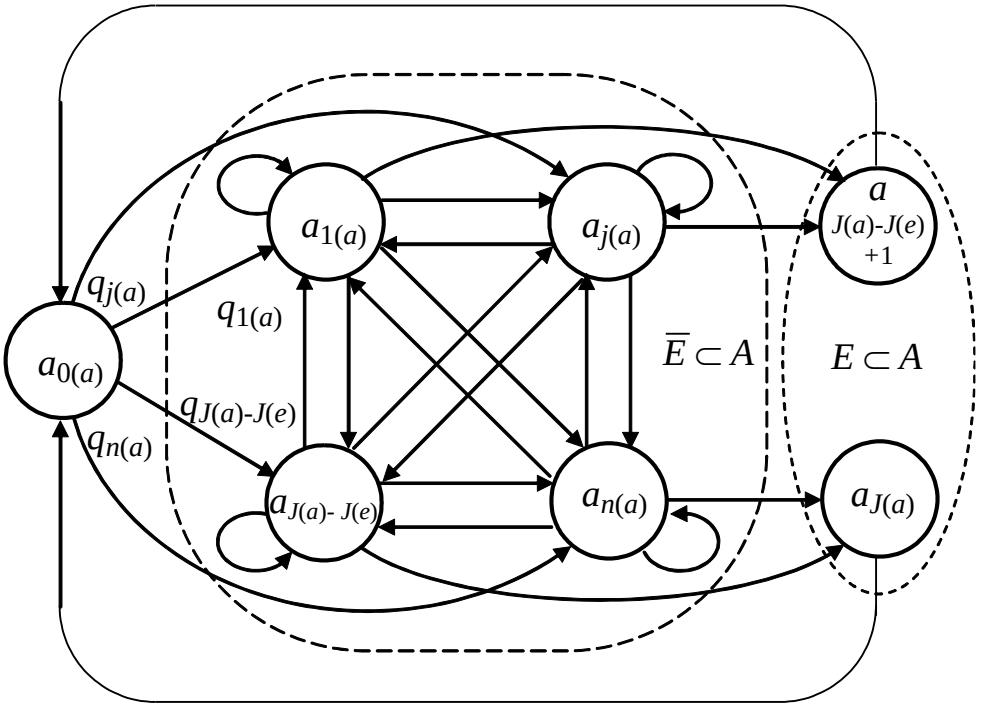


Рис. 2.5. Структура типовой циклограммы управления

При формировании типовой структуры циклограммы

$$\mu' = \{A', \boldsymbol{r}', \boldsymbol{h}'(t)\} \to \mu = \{A'', \boldsymbol{r}'', \boldsymbol{h}''(t)\}, \tag{2.39}$$

где $A'' = A'$;

$$\boldsymbol{h}''(t) = \lfloor h''_{j(a),n(a)}(t) \rfloor, \tag{2.40}$$

$$\boldsymbol{r}''(t) = \lfloor r''_{j(a),n(a)} \rfloor; \tag{2.41}$$

$$h''_{j(a),n(a)}(t) = \begin{cases} 0 \text{ when } n(a) = 0,\ 0 \le j(a) \le J(a) - J(e), \\ \text{or when } j(a) = 0,\ J(a) - J(e) + 1 \le n(a) \le J(a), \\ \text{or when } J(a) - J(e) + 1 \le j(a) \le J(a),\ 1(a) \le n(a) \le J(a); \\ h_{j(a),n(a,)} \text{ when } 1(a) \le j(a) \le J(a) - J(e),\ 1(a) \le n(a) \le J(a); \\ q_{n(a)} \cdot \delta(t) \text{ when } j(a) = 0,\ 1(a) \le n(a) \le J(a) - J(e); \\ \delta(t) \text{ when } n(a) = 0,\ J(a) - J(e) + 1(a) \le j(a) \le J(a). \end{cases} \tag{2.42}$$

$$r''_{j(a),n(a)} = \begin{cases} 0 \text{ when } n(a) = 0,\ 0 \le j(a) \le J(a) - J(e), \\ \text{or when } j(a) = 0,\ J(a) - J(e) + 1 \le n(a) \le J(a), \\ \text{or when } J(a) - J(e) + 1 \le j(a) \le J(a),\ 1(a) \le n(a) \le J(a); \\ r_{j(a),n(a,)} \text{ when } 1(a) \le j(a) \le J(a) - J(e),\ 1(a) \le n(a) \le J(a); \\ r'_{0(a),n(a)} \text{ when } j(a) = 0,\ 1(a) \le n(a) \le J(a) - J(e); \\ 1 \text{ when } n(a) = 0,\ J(a) - J(e) + 1(a) \le j(a) \le J(a). \end{cases} \tag{2.43}$$

2.3.2. Свойства полумарковской модели циклограммы

Свойство 2.1. Подмножества $\overline{E} \subset A$ и $E \subset A$ непусты.

Действительно, если $\left|\overline{E}\right| = 0$, то циклограмма не содержит ни одного действия, если $\left|E\right| = 0$, то у тела циклограммы нет конечного действия, за которым следует начальное действие.

Свойство 2.2. Число $J(a)$ состояний полумарковского процесса больше двух и конечно [229, 230, 231, 268].

Действительно, вследствие свойства 2.1, в полумарковском процессе (2.26) иметься более одного состояния, принадлежащего непустому подмножеству $\overline{E} \subset A$ и более одного состояния, принадлежащего непустому подмножеству $E \subset A$. Кроме того, $\overline{E} \cap E = \varnothing$. Поэтому если $\left|\overline{E}\right| \geq 1$, $\left|E\right| \geq 1$, то $\left|\overline{E} \cup E\right| \geq 2$, что доказывает первую часть свойства. Максимальное же количество состояний и структурных связей между ними ограничено физическими возможностями реальных запоминающих устройств, предназначенных для хранения цифровых образов циклограмм [235], что доказывает вторую часть свойства.

Свойство 2.3. Любое из $1(a) \leq j(a) \leq J(a)$ состояний полумарковского процесса $\boldsymbol{h}''(t)$ достижимо из состояния $a_{0(a)}$[236].

Действительно, полумарковский процесс $\boldsymbol{h}'(t)$, который является исходным для построения модели гистограммы, включает два типа состояний: первый тип - это состояния $a_{j(a)} \in \overline{E}$, достижимые из $a_{0(a)}$ непосредственно; второй тип $a_{j(a)} \in E$, каждое из которых достижимо из подмножества $\overline{E}$. Таким образом, состояния $a_{j(a)} \in E$ достижимы из $a_{0(a)}$ через состояния $a_{j(a)} \in \overline{E}$. Других состояний в полумарковском процессе $\boldsymbol{h}''(t)$ нет, что и доказывает свойство.

Свойство 2.4. Полумарковский процесс $\boldsymbol{h}''(t)$ является возвратным.

Действительно, в соответствии со свойством 2.3, любое из $1(a) \leq j(a) \leq J(a)$ состояний полумарковского процесса $\boldsymbol{h}''(t)$ достижимо из состояния $a_{0(a)}$. В силу

особенностей матрицы смежности (2.28), из всех состояний $a_{j(a)} \in E$ имеется дуга $\lfloor a_{j(a)}, a_{0(a)} \rfloor\ a_{0(a)}$. Таким образом, $\grave{a}_{j(a)} \to \grave{a}_{j(a)}$ для всех $0(a) \le j(a) \le J(a)$. Если все состояния полумарковского процесса $\boldsymbol{h}''(t)$ являются возвратными, то указанный полумарковский процесс является возвратным, что и доказывает свойство.

Отметим, что граф, представляющий структуру полумарковского процесса $\boldsymbol{h}''(t)$ является сильносвязанным графом

Свойство 2.5. На взвешенные плотности распределения $h''_{j(a),n(a)}(t)$ накладываются ограничения

$$h''_{j(a),n(a)}(t) = 0\text{, если } t \le 0,\ 1(a) \le j(a) \le J(a) - J(e). \qquad (2.44)$$

Действительно, любое действие в роботизированной платформе реализуется на реальных аппаратных средствах, а следовательно оно совершается за ненулевой время, что и доказывает свойство.

Свойство 2.6. Взвешенные плотности распределения полумарковского процесса (элементы матрицы) $\boldsymbol{h}''(t)$ некоррелированы между собой.

Утверждение 2.2. Полумарковский процесс $\boldsymbol{h}''(t)$ является эргодическим.

Доказательство. Действительно, в соответствии со свойством 2.4 полумарковский процесс $\boldsymbol{h}''(t)$ является возвратным. В соответствии со свойствами (2.2) и (2.6), среди состояний $a_{j(a)} \in E$ имеется хотя бы одно с ненулевым временем пребывания. В соответствии с классификацией полумарковских процессов, приведенной в [44, 45, 46, 237] полумарковский процесс $\boldsymbol{h}''(t)$ является эргодическим.

2.4. Определение временных интервалов блуждания по эргодическим полумарковским процессам

2.4.1. Характеристическая полумарковская матрица

Ограничение (2.18) является более сильным, чем условие Дирихле, следовательно $\boldsymbol{h}''(t)$ может быть применено преобразование Фурье

$$\Im\left[\boldsymbol{h}''(t)\right]=\int_0^{\infty}\boldsymbol{h}''(t)\exp(-i\varpi t)dt=\boldsymbol{H}''(i\varpi)=\left[H''_{j(a),n(a)}(i\varpi)\right], \qquad (2.45)$$

где $i=\sqrt{-1}$; ϖ - круговая частота.

Матрица $\boldsymbol{H}''(i\omega)$ представляет собой характеристическую матрицу полумарковского процесса $\boldsymbol{h}''(t)=\left\|h''_{j(a),n(a)}(t)\right\|$. Обратно полумарковская матрица из характеристической восстанавливается по следующей зависимости:

$$\boldsymbol{h}''(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\boldsymbol{H}''(i\varpi)\cdot\exp i\varpi t\cdot d\varpi\,. \qquad (2.46)$$

2.4.2. Временной интервал блуждания по выделенной траектории

Утверждение 2.3. Взвешенное время последовательности переключений полумарковского процесса по выделенной траектории блуждания определяется в виде

$$h_{0,N}(t)=\Im^{-1}\left\{\prod_{n=0}^{N-1}\Im\left[h_{n,n+1}(t)\right]\right\}. \qquad (2.47)$$

Доказательство. Выделим некоторую реализацию траектории блуждания и представим ее в виде локального полумарковского процесса

$$\boldsymbol{h}(t)=\begin{bmatrix}
0 & h_{0,1}(t) & 0 & \dots & 0 & \dots & 0 & 0 & 0\\
0 & 0 & h_{0,1}(t) & \dots & 0 & \dots & 0 & 0 & 0\\
0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0\\
 & & & & \dots & & & & \\
0 & 0 & 0 & \dots & h_{n,n+1}(t) & \dots & 0 & 0 & 0\\
 & & & & \dots & & & & \\
0 & 0 & 0 & \dots & 0 & \dots & 0 & h_{N-2,N-1}(t) & 0\\
0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & h_{N-1,N}(t)\\
0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0
\end{bmatrix}. \qquad (2.48)$$

Вид процесса приведен на рис. 2.6 а.

В соответствии с правилами суммирования случайных величин, плотность распределения суммы случайных временных интервалов, определенных плотно-

стями $h_{n\text{-}1,n}(t)$ и $h_{n,n+1}(t)$ определяется как свертка указанных плотностей, т.е.

$$h_{n-1,n+1}(t) = h_{n-1,n}(t) * h_{n,n+1}(t) = \int_0^\infty h_{n-1,n}(\tau) h_{n,n+1}(t-\tau) d\tau, \qquad (2.49)$$

где τ - вспомогательная переменная.

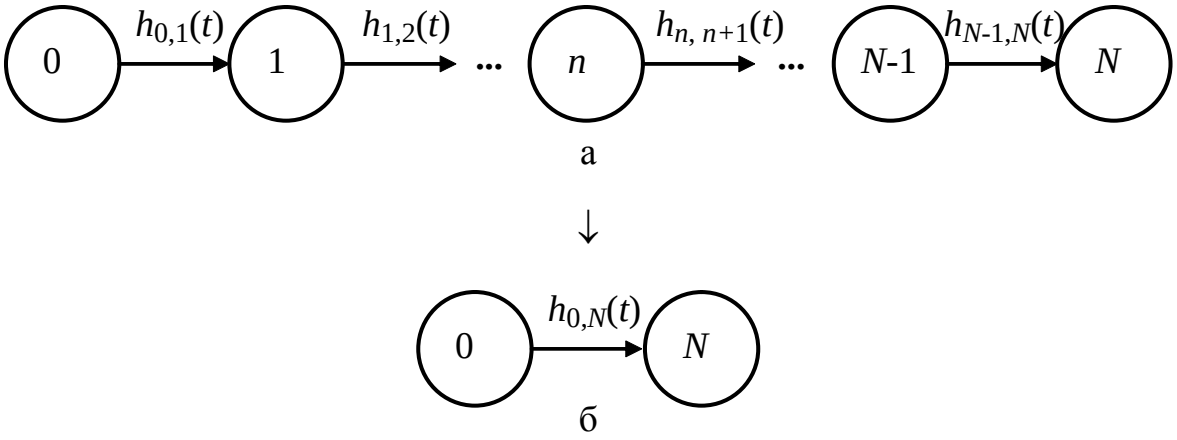


Рис. 2.6. Полумарковский процесс (2.48)

Если применить к обеим частям свертки преобразование Фурье, то в соответствии с теоремой о свертке,

$$\Im\left[h_{n-1,n}(t) * h_{n,n+1}(t)\right] = h_{n-1,n}(i\omega) \cdot h_{n,n+1}(i\omega).$$

Таким образом,

$$h_{n-1,n+1}(t) = \Im^{-1}\left\{\Im\left[h_{n-1,n}(t) * h_{n,n+1}(t)\right]\right\} = \Im^{-1}\left\{\Im\left[h_{n-1,n}(t)\right] \cdot \Im\left[h_{n,n+1}(i\omega)\right]\right\}$$

Распространение полученного результата на всю цепочку состояний, определенную матрицей (2.34), дает выражение (2.33), что и требовалось доказать.•

Процесс (2.34) преобразуется в процесс $\begin{bmatrix} 0 & h_{0,N}(t) \\ 0 & 0 \end{bmatrix}$, показанный на рис. 2.4 б. Для взвешенной плотности распределения $h_{0,N}(t)$ может быть найдена вероятность блуждания по траектории (2.34), плотность распределения времени блуждания, а также математическое ожидание и дисперсия времени блуждания:

$$p_{0,N} = \int_0^\infty \Im^{-1}\left\{\prod_{n=0}^{N-1} \Im\left[h_{n,n+1}(t)\right]\right\} dt; \qquad (2.50)$$

$$f_{0,N}(t)=\frac{h_{0,N}(t)}{p_{0,N}}; \tag{2.51}$$

$$T_{0,N}=\int_0^{\infty} t f_{0,N}(t)dt=\sum_{n=1}^{N} T_{n-1,n}\,; \tag{2.52}$$

$$D_{0,N}=\int_0^{\infty}\left(t-T_{0,N}\right)^2 f_{0,N}(t)dt=\sum_{n=1}^{N} D_{n-1,n}\,, \tag{2.53}$$

где $T_{n-1,n}$ и $D_{n-1,n}$ - соответственно математическое ожидание и дисперсия времени пребывания полумарковского процесса (2.34) в состоянии n - 1.

2.4.3. Временной интервал блужданий по одной из возможных траекторий

Утверждение 2.4. Взвешенное время блужданий по одной из возможных траекторий со стохастическим выбором определяется в виде

$$h_{0,N}(t)=\sum_{m=1}^{M} h_{0,N,m}(t). \tag{2.54}$$

Доказательство. Пусть существует M возможных траекторий блуждания, для каждой из которых определены взвешенные плотности $h_{0,N,M}(t)$. Локальный полумарковский процесс в этом случае принимает вид

$$\hat{\boldsymbol{h}}(t)=\begin{bmatrix} 0 & h_{0,N,1},\ldots,h_{0,N,m},\ldots,h_{0,N,M} \\ 0 & 0 \end{bmatrix}. \tag{2.55}$$

Вид процесса приведен на рис. 2.7.

Каждая из реализаций траектории блуждания может быть получена в соответствии с утверждением 2.2. Блуждания по различным траекториям при конкретных реализациях представляет собой несовместные некоррелированные события, которые стохастически суммируются. Из стохастического суммирования следует зависимость (2.54).

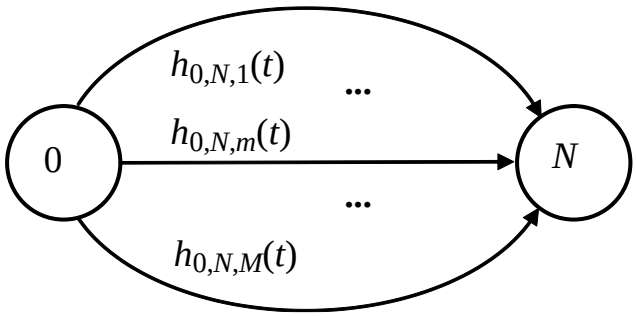


Рис. 2.7. Полумарковский процесс (2.55)

Для взвешенной плотности распределения (2.54) может быть найдена вероятность блуждания по множеству траекторий, плотность распределения времени блуждания, а также математическое ожидание и дисперсия времени блуждания:

$$\hat{p}_{0,N} = \int_0^{\infty} \sum_{m=1}^{M} h_{0,N,m}(t)dt = \sum_{m=1}^{M} p_{0,N,m}; \tag{2.56}$$

$$\hat{f}_{0,N}(t) = \frac{\sum_{m=1}^{M} h_{0,N,m}(t)}{\hat{p}_{0,N}}; \tag{2.57}$$

$$\hat{T}_{0,N} = \int_0^{\infty} t \frac{\sum_{m=1}^{M} h_{0,N,m}(t)}{\hat{p}_{0,N}} dt = \frac{\sum_{m=1}^{M} T_{0,N,m} \cdot p_{0,N,m}}{\hat{p}_{0,N}}; \tag{2.58}$$

$$\hat{D}_{0,N} = \int_0^{\infty} \left(t - \hat{T}_{0,N}\right)^2 \frac{\sum_{m=1}^{M} h_{0,N,m}(t)}{\hat{p}_{0,N}} dt = \frac{\sum_{m=1}^{M} \left(D_{0,N,m} + T_{0,N,m}^2\right) \cdot p_{0,N,m}}{\hat{p}_{0,N}} - \hat{T}_{0,N}^2, \tag{2.59}$$

где $T_{0,N,m}$ и $D_{0,N,m}$ - соответственно математическое ожидание и дисперсия времени блуждания полумарковского процесса (2.40) по m-й возможной траектории.

2.4.4. Полумарковский процесс, сформированный из траекторий блуждания

Рассмотрим полумарковский процесс ${}^{N}\boldsymbol{h}''(t)$, каждое состояние которого формируется из траекторий, полученных на процессе $\boldsymbol{h}''(t)$ в соответствии с утверждениями (2.2) и (2.3). Подобный процесс считается пребывающем в некотором состоянии до тех пор, пока в исходном процессе $\boldsymbol{h}''(t)$ не произойдет N переключений.

Утверждение 2.5. Полумарковский процесс ${}^{N}\boldsymbol{h}''(t)$ имеет $J(a) + 1$ состояний.

Доказательство. Все состояния полумарковского процесса ${}^{N}\boldsymbol{h}''(t)$ формируются из состояний процесса $\boldsymbol{h}''(t)$, который является возвратным, и более того, эргодическим. Поэтому в течение N переключений можно считать, что процесс ${}^{N}\boldsymbol{h}''(t)$ находится в том состоянии процесса $\boldsymbol{h}''(t)$, которое было до первого переключения, а затем сразу переключается в состояние, в которое процесс $\boldsymbol{h}''(t)$ переключился в результате блуждания по реализации траектории из N переключений. Таким образом, состояниями процесса ${}^{N}\boldsymbol{h}''(t)$ будут состояния множества

$$ {}^{N}A'' = \{{}^{N}a_{0(a)}, {}^{N}a_{1(a)}, ..., {}^{N}a_{j(a)}, ..., {}^{N}a_{J(a)}\}; {}^{N}a_{j(a)} \in A'. \qquad (2.60)$$

Утверждение 2.6. Полумарковская матрица процесса ${}^{N}\boldsymbol{h}''(t)$ может быть определена по зависимости

$$ {}^{N}\boldsymbol{h}''(t) = \Im^{-1}\left\lfloor \{\Im[\boldsymbol{h}''(t)]\}^{N} \right\rfloor = \left\lfloor {}^{N}h''_{j(a),n(a)}(t) \right\rfloor. \qquad (2.61)$$

Доказательство. Утверждение может быть доказано методом математической индукции.

База индукции: ${}^{1}\boldsymbol{h}''(t) = \boldsymbol{h}''(t)$, и в соответствии с утверждениями 2.2 и 2.3,

$$ {}^{2}\boldsymbol{h}''(t) = \boldsymbol{h}''(t) * \boldsymbol{h}''(t) = \Im^{-1}\{\Im[\boldsymbol{h}''(t)] \cdot \Im[\boldsymbol{h}''(t)]\} = \Im^{-1}\left\lfloor \{\Im[\boldsymbol{h}''(t)]\}^{2} \right\rfloor.$$

После перемножения матриц $\Im[\boldsymbol{h}''(t)]$ по правилам умножения (поэлементное умножение строки на столбец), каждый элемент произведения будет представлять собой сумму из $J(a) + 1$ произведений двух элементов характеристических матриц. Каждое произведение соответствует утверждению 2.2, т.е.

описывает характеристическую функцию траектории, включающей два состояния, которые полумарковский процесс $\boldsymbol{h}''(t)$ проходит за два переключения. Первый сомножитель представляет собой характеристическую функцию исходного текущего состояние, а второй сомножитель - есть характеристическая функция состояния, в которое процесс $\boldsymbol{h}''(t)$ попадает из исходного. Сумма произведений соответствует утверждению 2.3. т.е. описывает множество из $J(a) + 1$ возможных траекторий, которые полумарковский процесс $\boldsymbol{h}''(t)$ проходит за два переключения.

Шаг индукции. Предположим, что

$$^{N-1}\boldsymbol{h}''(t) = \Im^{-1}\left[\{\Im[\boldsymbol{h}''(t)]\}^{N-1}\right].$$

В соответствии с утверждениями 2.2 и 2.3,

$$^{N}\boldsymbol{h}''(t) = ^{N-1}\boldsymbol{h}''(t) * \boldsymbol{h}''(t) = \Im^{-1}\left\{\Im\left[^{N-1}\boldsymbol{h}''(t)\right]\cdot \Im[\boldsymbol{h}''(t)]\right\} = \Im^{-1}\left[\{\Im[\boldsymbol{h}''(t)]\}^{N}\right].$$

Каждое произведение соответствует утверждению 2.2, т.е. описывает характеристическую функцию траектории, включающей N - 1 состояний, которые полумарковский процесс $\boldsymbol{h}''(t)$ проходит за N - 1 переключение, плюс состояние, которое он проходит за N-е переключение. Сумма из $[J(a) + 1]^N$ произведений соответствует утверждению 2.3. т.е. описывает множество возможных траекторий, которые полумарковский процесс $\boldsymbol{h}''(t)$ проходит за N переключений. Таким образом, утверждение доказано.

Из (2.61) могут быть получены матрицы вероятностей и плотностей распределения времени пребывания в состояниях полумарковского процесса $^{N}\boldsymbol{h}''(t)$, $^{N}A'' = \{^{N}a_{0(a)}, ^{N}a_{1(a)}, ..., ^{N}a_{j(a)}, ..., ^{N}a_{J(a)}\}$; $^{N}a_{j(a)} \in A'$.

$$^{N}\boldsymbol{p}'' = \int_{0}^{\infty} \Im^{-1}\left[\{\Im[\boldsymbol{h}''(t)]\}^{N}\right]dt = \left[^{N}p''_{j(a),n(a)}\right]; \qquad (2.62)$$

$$^{N}\boldsymbol{f}''(t) = \left[\frac{^{N}h''_{j(a),n(a)}(t)}{^{N}p''_{j(a),n(a)}}\right] = \left[^{N}f''_{j(a),n(a)}(t)\right]; \qquad (2.63)$$

$$ {}^{N}\boldsymbol{T}'' = \left[\int_0^{\infty} t \frac{{}^{N}h''_{j(a),n(a)}(t)}{{}^{N}p''_{j(a),n(a)}} dt \right] = \left[{}^{N}T''_{j(a),n(a)} \right]; \tag{2.64} $$

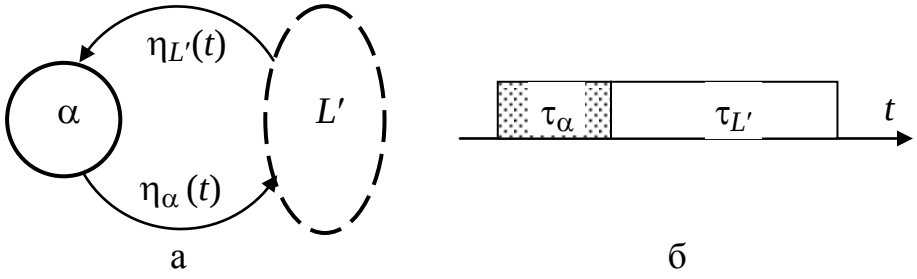
$$ {}^{N}\boldsymbol{D}'' = \left[\int_0^{\infty} \left[t - {}^{N}T''_{j(a),n(a)} \right]^2 \frac{{}^{N}h''_{j(a),n(a)}(t)}{{}^{N}p''_{j(a),n(a)}} dt \right] = \left[{}^{N}D''_{j(a),n(a)} \right]; \tag{2.65} $$

2.4.5. Методики определения временных интервалов блуждания по эргодическому полумарковскому процессу

Рассмотрим простой эргодический полумарковский процесс L, включающий возвратное состояние α и подпроцесс L', в который можно переключиться из α, и из которого в результате внутренних блужданий можно переключиться в α (рис. 2.6 а)

$$ L = \begin{pmatrix} 0 & \eta_{\alpha}(t) \\ \eta_{L'}(t) & 0 \end{pmatrix}, \tag{2.66} $$

где - $\eta_{\alpha}(t)$ - плотность распределения времени пребывания процесса в состоянии α; $\eta_{L'}(t)$ - плотность распределения времени блуждания по подпроцессу L' с последующим переключением в состояние α.



а б

Рис. 2.8. К вопросу о вероятности пребывания процесса L в состояниях α_1 и α_2

Математические ожидания времени пребывания полумарковского процесса в состояниях α_1 и α_2 равны, соответственно:

$$\tau_\alpha = \int_0^\infty \eta_\alpha(t)tdt \text{ и } \tau_{L'} = \int_0^\infty \eta_{L'}(t)tdt. \quad (2.67)$$

В соответствии с утверждением 2.2, среднее время возврата в состояние 1 (2) равно $\tau = \tau_\alpha + \tau_{L'}$ (рис. 2.8 б). Для внешнего по отношению к полумарковскому процессу наблюдателя вероятность того, что в произвольный момент времени процесс находится в состоянии α_1, равна

$$\pi_\alpha = \frac{\tau_\alpha}{\tau}. \quad (2.68)$$

Вероятность того, что в произвольный момент времени будут происходить блуждания по подпроцессу L' определяется как $\pi_{L'} = \frac{\tau_{L'}}{\tau}$.

Таким образом, при анализе состояний технологического процесса необходимо определять:

время пребывания полумарковского процесса, моделирующего функционирование процесса, в отдельно взятом состоянии;

время блуждания по подпроцессу;

время возврата полумарковского процесса в заданное состояние.

Методика 2.1. Время пребывания полумарковского процесса, моделирующего функционирование платформы, в отдельно взятом состоянии, вследствие того, что переключения из любого состояния составляют полную группу несовместных событий определяется по зависимости:

$$f_{j(a)}(t) = \sum_{n(a)=0(a)}^{J(a)} h_{j(a),n(a)}(t). \quad (2.69)$$

Числовые характеристики рассчитываются по формулам.

$$T_{j(a)} = \sum_{n(a)=0(a)}^{J(a)} T_{j(a),n(a)} p_{j(a),n(a)}; \quad (2.70)$$

$$D_{j(a)} = \sum_{n(a)=0(a)}^{J(a)} \left[D_{j(a),n(a)} + T^2_{j(a),n(a)}\right] p_{j(a),n(a)} - T^2_{j(a)}. \quad (2.71)$$

Методика 2.2. Определение времени блуждания по подпроцессу.

Рассмотрим самый общий случай, когда структура состояний эргодического полумарковского процесса описывается полным ориентированным графом с петлями (рис. 2.9 а). Случай является более общим, чем процесс $\boldsymbol{h''}(t)$, поэтому будем представлять его в виде

$$\boldsymbol{h'''}(t)=\left\|h'''_{j(a),n(a)}\right\|,\ h'''_{j(a),n(a)}\neq 0,\ 0(a)\leq j(a)\leq J(a). \qquad (2.72)$$

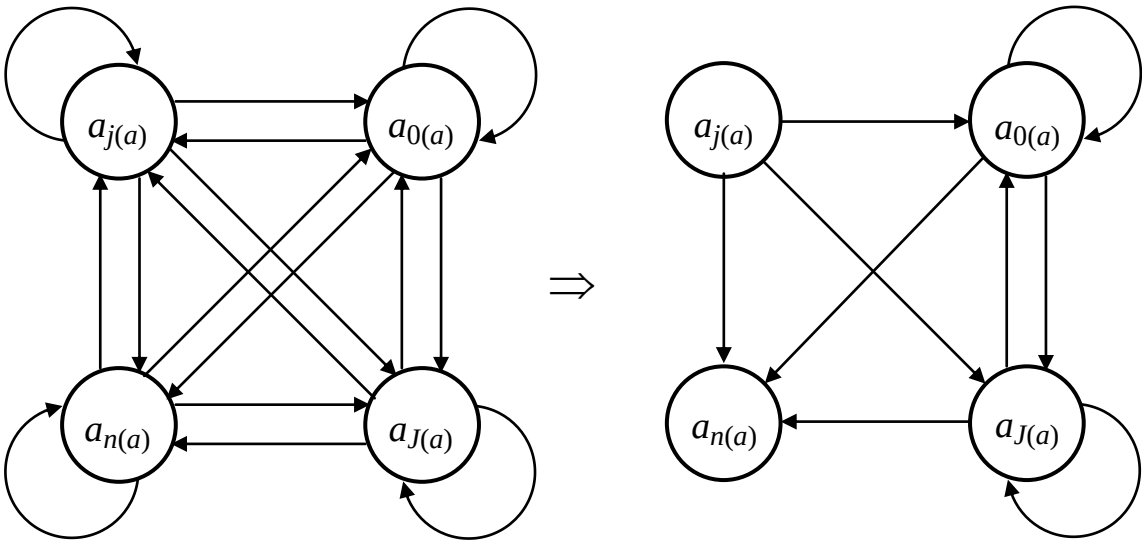


Рис. 2.9. Преобразование графа состояний полумарковского процесса при определении времени блуждания

Предположим, что точкой входа в подпроцесс является состояние $a_{j(a)}$, а точкой выхода из подпроцесса является состояние $a_{n(a)}$. Единственным ограничением на траектории при определении времени блуждания является то, что в ни в состояние $a_{n(a)}$, ни в состояние $a_{j(a)}$ процесс не должен попадать дважды. Для того, чтобы удовлетворить этому ограничению должен быть сформирован граф состояний, приведенный на рис. 2.9 б.

Для этого из полумарковской матрицы и матрицы смежности должны быть удалены ссылки на состояние $a_{j(a)}$ во всех строках, матриц, а из строки, определяющей состояние $a_{n(a)}$ должны быть удалены все ссылки на все состояния матриц. Матрица смежности процесс принимают вид:

$$\boldsymbol{r}''' = \begin{pmatrix} r_{0(a),0(a)} & \ldots & 0 & \ldots & r_{0(a),n(a)} & \ldots & r_{0(a),J(a)} \\ & & & & \ldots & & \\ r_{j(a),0(a)} & \ldots & 0 & \ldots & r_{j(a),n(a)} & \ldots & r_{j(a),J(a)} \\ & & & & ,,, & & \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ & & & & \ldots & & \\ r_{J(a),0(a)} & \cdots & 0 & \cdots & r_{J(a),n(a)} & \cdots & r_{J(a),J(a)} \end{pmatrix}; \qquad (2.73)$$

полумарковская матрица принимает вид

$$\boldsymbol{h}'''(t) = \begin{pmatrix} h_{0(a),0(a)}(t) & \ldots & 0 & \ldots & h_{0(a),n(a)}(t) & \ldots & h_{0(a),J(a)}(t) \\ & & & & \ldots & & \\ h_{j(a),0(a)}(t) & \ldots & 0 & \ldots & h_{j(a),n(a)}(t) & \ldots & h_{j(a),J(a)}(t) \\ & & & & ,,, & & \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ & & & & \ldots & & \\ h_{J(a),0(a)}(t) & \cdots & 0 & \cdots & h_{J(a),n(a)}(t) & \cdots & h_{J(a),J(a)}(t) \end{pmatrix}. \qquad (2.74)$$

Полумарковский процесс $h'''(t)$ не является эргодическим. Он имеет единственное начальное состояние $a_{j(a)}$ и единственное поглощающее состояние. Все возможные траектории блуждания по полумарковскому процессу составляют полную группу несовместных событий. Поэтому плотность распределения времени достижения состояния $a_{j(a)}$ из состояния $a_{n(a)}$ определяется по зависимости

$$f'''_{j(a),n(a)}(t) = \Im^{-1}\left({}^{r}\boldsymbol{I}_{j(a)} \cdot \sum_{k=1}^{\infty} \left\{ \Im\left[\boldsymbol{h}'''(t) \right] \right\}^{k} \cdot {}^{c}\boldsymbol{I}_{n(a)} \right), \qquad (2.75)$$

где ${}^{r}\boldsymbol{I}_{j(a)}$ - вектор-строка, $j(a)$-й элемент которого равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{n(a)}$ - вектор-столбец, $n(a)$-й элемент которого равен единице, а остальные элементы равны нулю.

Из (2.75) могут быть получены числовые характеристики:

$$p'''_{j(a),n(a)} = 1;$$

$$T'''_{j(a),n(a)} = \int_{0}^{\infty} t \cdot f'''_{j(a),n(a)}(t)\, dt; \qquad (2.76)$$

$$D'''_{j(a),n(a)} = \int_0^{\infty} \left[t - T'''_{j(a),n(a)} \right]^2 \cdot f'''_{j(a),n(a)}(t)\,dt.$$

Методика 2.3. Определение времени возврата в состояние

Методика составлена для общего случая полумарковского процесса (2.54).

В общем случае можно допустить, что определяется время возврата в состояние $a_{j(a)}$. Разделим вершину на вершину $a_{j(a)b}$ входа в подпроцесс и вершину $a_{j(a)e}$, моделирующую поглощающее состояние. С учетом того, что при блужданиях процесс не должен попадать в состояние $a_{j(a)}$ дважды, должен быть из графа, приведенного на рис. 2.10 а должен быть сформирован граф состояний, приведенный на рис. 2.10 б.

Матрицы смежности и полумарковская принимают вид

$$ {}_{be}\boldsymbol{r}''' = \begin{bmatrix} r_{0(a),0(a)} & \cdots & r_{0(a),j(a)-1} & 0 & r_{0(a),j(a)+1} & \cdots & r_{0(a),J(a)} & r_{0(a),j(a)} \\ \cdots & & \cdots & & \cdots & & \cdots & \cdots \\ r_{j(a)-1,0(a)} & \cdots & r_{j(a)-1,j(a)-1} & 0 & r_{j(a)-1,j(a)+1} & \cdots & r_{j(a)-1,J(a)} & r_{j(a)-1,j(a)} \\ r_{j(a),0(a)} & & r_{j(a),j(a)-1} & & r_{j(a),j(a)+1} & \cdots & r_{j(a),J(a)} & r_{j(a),j(a)} \\ r_{j(a)+1,0(a)} & & r_{j(a)+1,j(a)-1} & 0 & r_{j(a)+1,j(a)+1} & \cdots & r_{j(a)+1,J(a)} & r_{j(a)+1,j(a)} \\ \cdots & & \cdots & & \cdots & & \cdots & \cdots \\ r_{J(a),0(a)} & & r_{J(a),j(a)-1} & 0 & r_{J(a),j(a)+1} & \cdots & r_{J(a),J(a)} & r_{J(a),j(a)} \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{2.77}$$

$$ {}_{be}\boldsymbol{h}'''(t) = \begin{bmatrix} h_{0(a),0(a)} & \cdots & h_{0(a),j(a)-1} & 0 & h_{0(a),j(a)+1} & \cdots & h_{0(a),J(a)} & h_{0(a),j(a)} \\ \cdots & & \cdots & & \cdots & & \cdots & \cdots \\ h_{j(a)-1,0(a)} & \cdots & h_{j(a)-1,j(a)-1} & 0 & h_{j(a)-1,j(a)+1} & \cdots & h_{j(a)-1,J(a)} & h_{j(a)-1,j(a)} \\ h_{j(a),0(a)} & & h_{j(a),j(a)-1} & & h_{j(a),j(a)+1} & \cdots & h_{j(a),J(a)} & h_{j(a),j(a)} \\ h_{j(a)+1,0(a)} & & h_{j(a)+1,j(a)-1} & 0 & h_{j(a)+1,j(a)+1} & \cdots & h_{j(a)+1,J(a)} & h_{j(a)+1,j(a)} \\ \cdots & & \cdots & & \cdots & & \cdots & \cdots \\ h_{J(a),0(a)} & & h_{J(a),j(a)-1} & 0 & h_{J(a),j(a)+1} & \cdots & h_{J(a),J(a)} & h_{J(a),j(a)} \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{2.78}$$

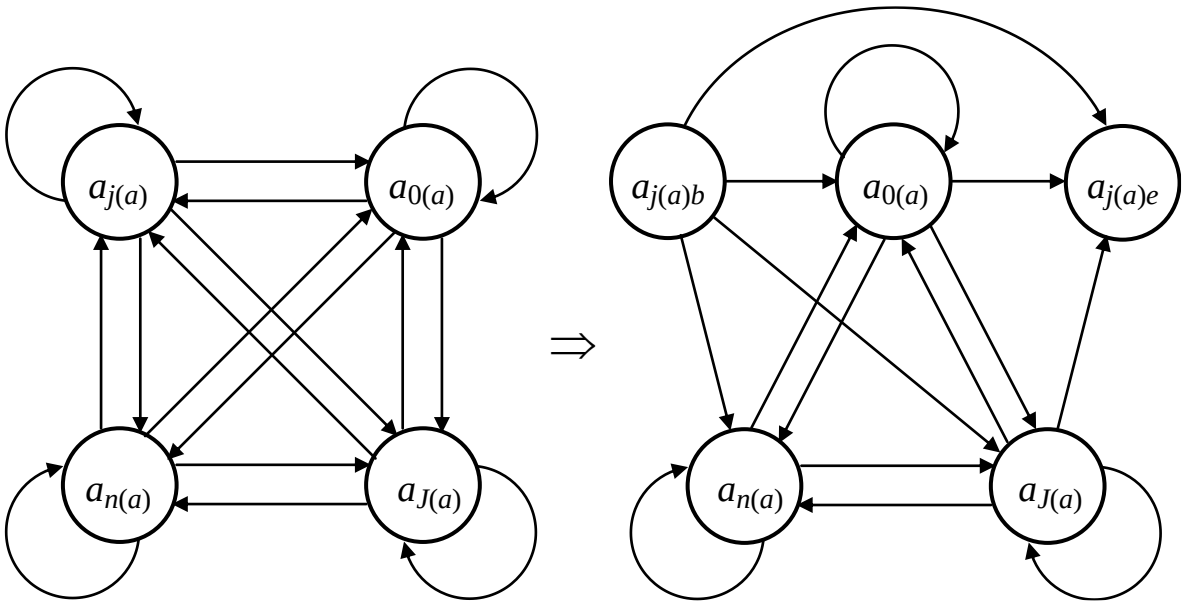


Рис. 2.10. Преобразование графа состояний полумарковского процесса при определении времени возврата

Преобразования матриц в них должны быть добавлены справа один столбец и снизу одна строка с номером $J(a) + 1$. В получившихся матрицах размером $(J(a) + 1) \times (J(a) + 1)$ $j(a)$-е строка и столбец будут определять одну из разделенных вершин, например вершину $a_{j(a)b}$, а $(J(a) + 1)$-е строка и столбец будут определять другую разделенную вершину, а именно, $a_{j(a)e}$.

В вершину $a_{j(a)b}$ не входит ни одна дуга, поэтому элементы $j(a)$-го столбца будут равны нулю. Из вершины $a_{j(a)e}$ не выходит ни одна дуга, поэтому элементы $(J(a) + 1)$-й строки будут равняться нулю. В вершину $a_{j(a)e}$ входят все дуги, которые в исходном графе входили в вершину $a_{j(a)}$, поэтому $j(a)$-й столбец должен быть перенесен на $(J(a) + 1)$-е место. В целом элементы матрицы будут определяться следующим выражением:

Полумарковский процесс $_{be}\boldsymbol{h}'''(t) =$ не является эргодическим. Он имеет единственное начальное состояние $a_{j(a)b}$ и единственное поглощающее состояние $a_{j(a)e}$. Все возможные траектории блуждания по полумарковскому процессу составляют полную группу несовместных событий. Поэтому плотность распределения времени достижения состояния $a_{j(a)e}$ из состояния $a_{j(a)b}$ определяется по зависимости

$$f'''_{j(a),\,j(a)}(t) = \Im^{-1}\left({}^{r}\boldsymbol{I}_{j(a)} \left(\sum_{k=1}^{\infty} \{\Im[{}_{be}\boldsymbol{h}'''(t)]\}^{k} \right) {}^{c}\boldsymbol{I}_{J(a)+1} \right), \qquad (2.79)$$

где ${}^{r}\boldsymbol{I}_{j(a)}$ - $(J(a) + 1)$-мерный вектор-строка, $j(a)$-й элемент которого равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{J(a)+1}$ - $(J(a) + 1)$-мерный вектор-столбец, $(J(a) + 1)$-й элемент которого равен единице, а остальные элементы равны нулю.

Из (2.79) могут быть получены числовые характеристики:

$$p'''_{j(a),\,j(a)}(t) = 1;$$

$$T'''_{j(a),\,j(a)}(t) = \int_0^{\infty} t f'''_{j(a),\,j(a)}(t)\,dt; \qquad (2.80)$$

$$D'''_{j(a),\,j(a)} = \int_0^{\infty} \left(t - T'''_{j(a),\,j(a)}\right)^2 f'''_{j(a),\,j(a)}(t)\,dt.$$

Плотность распределения времени возврата и основные числовые характеристики дают достаточно полное описание динамики состояний гофроагрегата при решении поставленных задач.

2.5. Оценка степени «марковости» процессов

При решении задач построения оптимальных циклограмм управления МР крайне важным представляется вопрос о степени приближения потока переключений к пуассоновсуому потоку. С одной стороны, абстракция «пуассоновский поток является достаточно грубой, а с другой стороны, применение этого понятия приводит к значительному сокращению выкладок при аналитическом решении задач, или вычислительной сложности алгоритмов при компьютерном моделировании МР. Поэтому актуальными остаются исследования критериев соответствия плотности распределения времени между событиями $g(t)$ в потоке переключений, плотности распределения интервалов времени между событиями в пуассоновском

потоке, описываемой экспоненциальным законом распределения. Критерий должен быть простым, и в то же время должен давать точную оценку степени «марковости процессов. Рассмотрим и сравним различные критерии оценки.

2.5.1. Регрессионный критерий и критерий Пирсона

Интервалы времени между событиями в пуассоновском потоке характеризуются экспоненциальным законом распределения [79, 80, 88, 185, 207, 262]

$$f(t)=\frac{1}{T}\exp\left(-\frac{t}{T}\right), \tag{2.81}$$

где T – математическое ожидание времени между событиями в потоке.

Регрессионный критерий (критерий наименьших квадратов) основан на оценке интеграла квадрата разности между анализируемым $g(t)$ и экспоненциальным (1) законами [107, 116, 180, 186, 213, 223]:

$$\varepsilon_r=\int_0^{\infty}[g(t)-f(t)]^2\,dt\ . \tag{2.82}$$

Очевидно, что если $g(t)\to f(t)$, то $\varepsilon_r\to 0$. Пусть

$g(t)=\delta(t-T)$,

где $\delta(t-T)$ – смещенная δ-функция Дирака, для которой

$$\delta(t-T)=\begin{cases}0 \text{ when } t\neq T;\\ \infty \text{ when } t=T;\end{cases}\quad \int_0^{\infty}\delta(t-T)dt=1. \tag{2.83}$$

Тогда

$$\varepsilon_r=\int_0^{\infty}[\delta(t-T)-f(t)]^2\,dt=\varepsilon_{r1}+\varepsilon_{r2}+\varepsilon_{r3}, \tag{2.84}$$

где

$$\varepsilon_{r1}=\int_0^{\infty}\delta^2(t-T)dt=\lim_{a\to 0}\int_{T-a}^{T+a}\left(\frac{1}{2a}\right)^2 dt=\infty;$$

$$\varepsilon_{r2} = -2\int_0^{\infty} \delta\left(t - T_g\right) \cdot \frac{1}{T_f} \exp\left(-\frac{t}{T}\right) dt = -\frac{2}{eT} \ ;$$

$$\varepsilon_{r3} = \int_0^{\infty} \frac{1}{T^2} \exp\left(-\frac{2t}{T}\right) dt = \frac{1}{2T}.$$

Таким образом, статистика регрессионного критерия изменяется от 0 (поток без последействия) до ∞ (поток с жесткой детерминированной связью между событиями).

В том случае, если временные интервалы между событиями определяются экспериментально и плотность распределения $g(t)$ представляет собой статистический ряд вида

$$g(t) = \begin{pmatrix} t_0 \le t < t_1 & \dots & t_{i-1} \le t < t_k & \dots & t_{J-1} \le t < t_J \\ n_1 & \dots & n_i & \dots & n_J \end{pmatrix}, \qquad (2.85)$$

где n_i - количество результатов измерения, лежащих в интервале $t_{i-1} \le t < t_i$, то для оценки близости плотности и гистограммы может быть использован *критерий Пирсона* (критерий χ^2) [1, 230, 186, 199], который принимает вид

$$\chi^2 = \sum_{j=1}^{J} \frac{\left\{ n_j - \left[\exp\left(-\frac{t_{j-1}}{T}\right) - \exp\left(-\frac{t_j}{T}\right) \right] \cdot N \right\}^2}{\left[\exp\left(-\frac{t_{j-1}}{T}\right) - \exp\left(-\frac{t_j}{T}\right) \right] \cdot N} \ . \qquad (2.86)$$

Критерий (6) достаточно громоздок и применим в ограниченном количестве случаев (для больших репрезентативных выборок).

2.5.2. Корреляционный критерий

Корреляционный критерий имеет вид [120]

$$\varepsilon_c = \int_0^{\infty} g(t) \cdot \frac{1}{T} \exp\left(-\frac{t}{T}\right) dt. \qquad (2.87)$$

При $g(t) = \frac{1}{T} \exp\left(-\frac{t}{T}\right)$ критерий достигает максимума:

$$\varepsilon_{c1} = \int_0^\infty \frac{1}{T}\exp\left(-\frac{t}{T}\right)\cdot\frac{1}{T}\exp\left(-\frac{t}{T}\right)dt = \frac{1}{2T},$$

а при $g(t)=\delta(t-T)$ критерий достигает минимума

$$\varepsilon_{c2} = \int_0^\infty \delta(t-T)\exp\left(-\frac{t}{T}\right)dt = \frac{1}{eT}.$$

Однако, если критерий – это индикатор отсутствия последействия, то его статистика должна быть безразмерной и укладываться в интервал $0 \le \tilde{\varepsilon}_c \le 1$ Нуль должен достигаться в первом случае (отсутствие последействия), единица должна достигаться во втором случае (детерминированная связь между событиями). Это происходит, если значение ε_c, рассчитанное как корреляция по зависимости (7), будет пересчитано по формуле

$$\tilde{\varepsilon}_c = \frac{e(1-2T\varepsilon_c)}{e-2}. \qquad (2.88)$$

2.5.3. Параметрические критерии

Простейший вариант *параметрического критерия* основан на следующем свойстве экспоненциальной плотности распределения [1, 8, 30]:

$$T = \sqrt{D}, \qquad (2.89)$$

где D – дисперсия, определяемая по зависимости

$$D = \int_0^\infty \frac{(t-T)^2}{T}\exp\left(-\frac{t}{T}\right)dt.$$

Очевидно, что подобными свойствами обладают многие плотности распределения, например, взвешенная пара вырожденных законов, $g(t)=0{,}5\delta(t-\tau_1)+0{,}5\delta(t-\tau_2)$, если $\tau_1=0$, $\tau_2>0$. Это затрудняет практическое использование зависимости (9).

Для установления более сложного критерия рассмотрим процесс генерации событий в потоке как «соревнование», в котором участвуют два субъекта: внешний наблюдатель и генератор. Если в момент старта одновременно запускаются

случайные процессы, характеризующие временные интервалы между стартом и наблюдением и между двумя событиями, то «соревнование» может быть описано с помощью 2-параллельного полумарковского процесса [1, 8]

$$\mathbf{M} = [A, \mathbf{h}(t)], \tag{2.90}$$

где $A = \{a_{w1}, a_{w2}, a_{g1}, a_{g2}, \}$ – множество состояний; a_{w1}, a_{g1} – стартовые состояния; a_{w2}, a_{g2} – поглощающие состояния; $\mathbf{h}(t)$ – полумарковская матрица;

$$\mathbf{h}(t) = \begin{bmatrix} \begin{bmatrix} 0 & w(t) \\ 0 & 0 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} 0 & g(t) \\ 0 & 0 \end{bmatrix} \end{bmatrix}; \ \mathbf{0} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.91}$$

Рассмотрим ситуацию, когда первый субъект выигрывает «соревнование» в момент времени τ и ожидает, когда второй субъект достигнет финиша. Для определения времени ожидания по полумарковскому процессу (2.91) (рис. 2.11 а) может быть построен ординарный полумарковский процесс (рис. 2.11 б) вида

$$\mathbf{M}' = [\mathrm{A}', \mathbf{h}'(t)], \tag{2.92}$$

где $\mathrm{A}' = \mathrm{A} \cup \mathrm{B}$ – множество состояний; $\mathrm{A} = \{\alpha_1, \alpha_2, \alpha_3\}$ – подмножество состояний, моделирующее начало и окончания блужданий по полумарковскому процессу; α_1 – стартовое состояние; α_2 – поглощающее состояние, моделирующее выигрыш второго субъекта; α_3 – поглощающее состояние, моделирующее окончание ожидания первым субъектом финиширования второго, проигравшего субъекта; $\mathrm{B} = \{\beta_1, ..., \beta_i, ...\}$ – бесконечное множество состояний, задающих временные интервалы для различных ситуаций завершения дистанции вторым, проигравшим, субъектом; $\mathbf{h}'(t) = \{h'_{m,n}(t)\}$ – полумарковская матрица, задающая временные интервалы процесса.

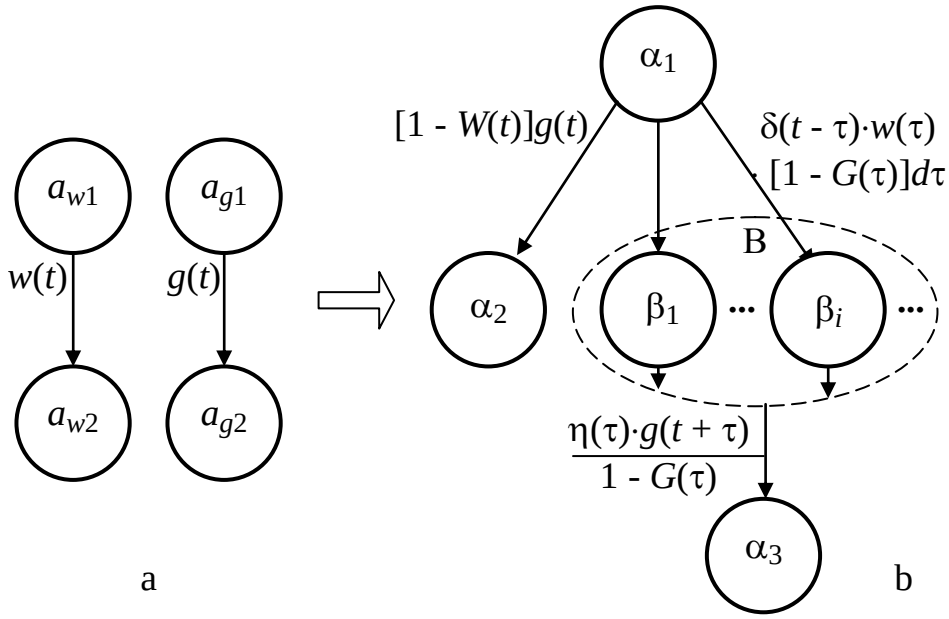


Рис. 2.11. К расчету времени ожидания события

Элементы $h'_{m,n}(t)$ определяются следующим образом:

$h'_{1,2}(t)$ определяется как взвешенная плотность распределения времени финиширования второго субъекта, если он является «победителем» «соревнования»,

$$h'_{12}(t) = g(t)[1 - W(t)], \tag{2.93}$$

где $W(t) = \int_0^t w(\theta)d\theta$ – функция распределения; θ – вспомогательная переменная; $h'_{1,2+i}(t)$, $i = 1, 2, \ldots$, определяются как взвешенные плотности распределения времени финиширования первого субъекта в точности во время τ, если он является «победителем соревнования» и ожидает второго субъекта;

$$h'_{1,2+i}(t) = \delta(t - \tau) \cdot w(\tau)[1 - G(\tau)]d\tau, \tag{2.94}$$

где $\delta(t - \tau)$ – вырожденный закон распределения, определяющий время τ, финиширования второго субъекта; $G(t) = \int_0^t g(\theta)d\theta$; $w(\tau)[1 - G(\tau)]d\tau$ – вероятность финиширования первого субъекта в точности во время τ, если он является «победителем соревнования»;

$$\frac{1(t) \cdot g(t + \tau)}{1 - G(\tau)},$$

где 1(t) – единичная функция Хевисайда – плотность распределения времени пребывания полумарковского процесса (12) в состоянии B, которая получается путем отсечения от смещенной плотности $g(t+\tau)$ значений с отрицательным аргументом.

Таким образом, вероятность попадания процесса в подмножество B равна $p_{\alpha_0\beta} = \int_0^\infty [1-G(\tau)]w(\tau)d\tau = \int_0^\infty W(t)g(t)dt$. Взвешенная плотность распределения времени ожидания первым субъектом финиширования второго субъекта равна $h_{w\to g}(t) = \eta(t)\int_0^\infty w(\tau)g(t+\tau)d\tau$. Чистая плотность распределения определяется следующим образом

$$f_{w\to g}(t) = \frac{\eta(t)\int_0^\infty w(\tau)g(t+\tau)d\tau}{\int_0^\infty W(t)dG(t)}. \qquad (2.95)$$

Следует отметить, что операция (2.95) не является коммутативной, т.е. в общем случае

$$f_{g\to w}(t) = \frac{\eta(t)\int_0^\infty g(\tau)w(t+\tau)d\tau}{\int_0^\infty G(t)dW(t)} \neq f_{w\to g}(t)$$

Рассмотрим поведение $f_{w\to g}(t)$ для двух видов функции $g(t)$: когда указанная функция описывает поток событий без последействия, т.е. $g(t) = \frac{1}{T}\exp\left(-\frac{t}{T}\right)$, и когда поток событий является строго детерминированным, т.е. $g(t) = \delta(t-T)$.

Выражение (15) для первого случая принимает вид:

$$f_{w\to g}(t)=\frac{\eta(t)\int_0^{\infty} w(\tau)\frac{1}{T}\exp\left[-\frac{t+\tau}{T}\right]d\tau}{1-\int_{t=0}^{\infty}\left[1-\exp\left(-\frac{t}{T}\right)\right]dW(t)}=\frac{1}{T}\exp\left(-\frac{t}{T}\right). \qquad (2.96)$$

Таким образом, плотность $f_{w\to g}(t)$ отражает свойство отсутствия последействия в строго марковских процессах с непрерывным временем, которое может быть сформулировано следующим образом. Если плотность распределения времени между любыми двумя событиями в системе распределена по экспоненциальному закону, то для внешнего наблюдателя время, оставшееся до наступления очередного события, будет также распределено по экспоненциальному закону, независимо от момента начала наблюдения.

Выражение (2.95) для второго случая принимает вид:

$$f_{w\to g}(t)=\frac{\eta(t)w(T_w-t)}{W(T_w)}. \qquad (2.97)$$

Пусть $w(t)$ имеет область определения $T_{w\min}\le \arg w(t)\le T_{w\max}$ и математическое ожидание $T_{w\min}\le T_w\le T_{w\max}$. В зависимости от местоположения $w(t)$ и $g(t)$ на оси времени, возможны следующие ситуации:

а) $T<T_{w\min}$. В этой ситуации выражение (2.95) не имеет смысла.

б) $T_{w\min}\le T\le T_{w\max}$. В этой ситуации плотность распределения выражается зависимостью (2.97), область определения $f_{w\to g}(t)$ определяется как $0\le \arg\left\lfloor f_{w\to g}(t)\right\rfloor\le T-T_{w\min}$, и $\int_0^{\infty} t f_{w\to g}(t)dt\le T$.

в) $T>T_{w\max}$. В этой ситуации $f_{w\to g}(t)=w(T-t)$, $T-T_{w\max}\le \arg\left\lfloor f_{w\to g}(t)\right\rfloor\le T-T_{w\min}$, и $\int_0^{\infty} t f_{w\to g}(t)dt\le T$.

Таким образом, математическое ожидание функции $f_{w\to g}(t)$ для пуассоновского потока событий остается неизменным, а для детерминированного потока событий уменьшается, и это уменьшение определяется видом функции $w(\tau)$. Это

обстоятельство позволяет определить вид простого критерия, основанного на использовании математического ожидания плотности распределения времени ожидания.

Пусть плотность распределения времени наблюдения определяется вырожденным законом распределения с математическим ожиданием, равным T, т.е. $w(t)=\delta(t-T)$ (соответствует детерминированному потоку событий). Для этого случая плотность распределения времени ожидания δ-функцией Дирака события, когда завершится событие $g(t)$, определяется по зависимости

$$f_{\delta\to g}(t)=\frac{\eta(t)\cdot g(t+T)}{\int\limits_{T}^{\infty} g(t)dt}. \qquad (2.98)$$

Математическое ожидание в (18) имеет вид

$$T_{\delta\to g}=\int\limits_{0}^{\infty} t\frac{g(t+T)}{\int\limits_{T}^{\infty} g(t)dt}dt. \qquad (2.99)$$

Статистика критерия, основанного на определении времени ожидания, имеет вид

$$\varepsilon_w=\left(\frac{T-T_{\delta\to g}}{T}\right)^2, \qquad (2.100)$$

где T – математическое ожидание анализируемой плотности распределения времени между соседними событиями; $T_{\delta\to g}$ – математическое ожидание плотности распределения $f_{\delta\to g}(t)$, рассчитываемое по зависимости (2.98).

Для экспоненциального закона

$$,\varepsilon_w=\left(\frac{T-T_{\delta\to g}}{T}\right)^2=\left(\frac{T-T}{T}\right)^2=0, \qquad (2.101)$$

что означает отсутствие последействия. Для строго детерминированной связи между событиями, выражаемой δ-функцией Дирака $g(t)=\delta(t-T)$

$$f_{\delta\to g}(t)=\delta(t),\ \text{и}\ \varepsilon_w=\left(\frac{T-0}{T}\right)^2=1, \tag{22}$$

что означает детерминированную связь между событиями, или «абсолютное последействие».

Исследуем поведение критерия $\sqrt{\varepsilon_w}$. Для этого определим математическое ожидание функции $g(t)$ в виде (рис. 2.12)

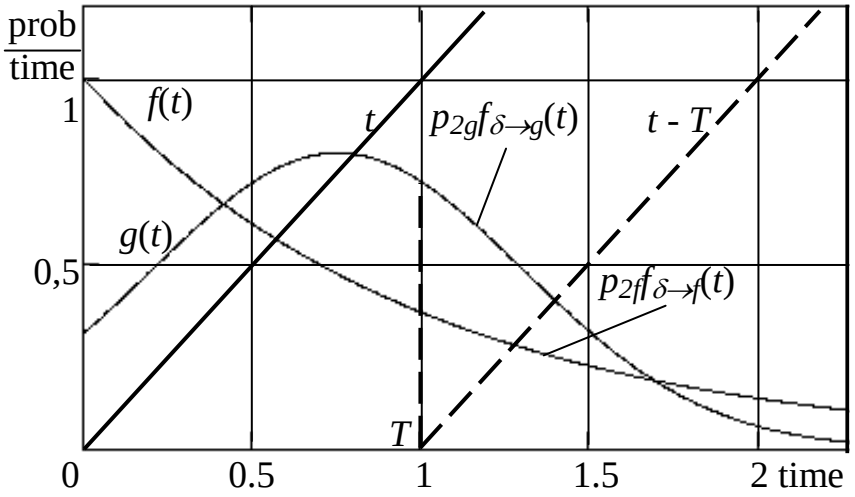


Рис. 2. 12. К расчету математического ожидания числа событий в потоке

$$\begin{aligned}\int_0^\infty tg(t)dt &= \int_0^T tg(t)dt + \int_0^\infty tg(t+T)dtT + T\int_0^\infty g(t+T)dt = \\ &= p_{1g}T_{1g} + p_{2g}T_{\delta\to g} + p_{2g}T = T,\end{aligned} \tag{2.103}$$

где $p_{1g}=\int_0^T g(t)dt;\ p_{2g}=\int_T^\infty g(t)dt$.

Очевидно, что в (2.103) $T_2=T_{\delta\to g}$. Если $g(t)=f(t)$, то из уравнения

$$p_{1f}T_{1f} + p_{2f}T_{\delta\to f} + p_{2f}T = T, \tag{2.104}$$

где $p_{1f}=\frac{e-1}{e};\ p_{2f}=\frac{1}{e};\ T_{1f}=T\frac{e-2}{e-1}$,

следует, что

$$T_{\delta\to f}=T. \tag{2.105}$$

Равенство (2.105) подтверждает справедливость зависимостей (2.96) и

(2.101). При $g(t) \neq f(t)$ из (2.103) следует

$$T_{\delta \to g} = \frac{p_{1g}(T - T_{1g})}{1 - p_{1g}}. \tag{2.106}$$

Значение $T_{\delta \to g}$, в зависимости от соотношения значений T_{1g} и p_{1g} может быть как $T_{\delta \to g} > T$, так и $T_{\delta \to g} < T$ (случай $T_{\delta \to g} = T$ представлен зависимостями (2.104), (2.105)).

2.5.4. Случай равномерного закона распределения

В качестве примера рассмотрим случай, когда поток событий формируется в результате «соревнования» *K* субъектов с равновероятными и одинаковыми законами распределения. Модель формирования потока может быть представлена в виде *K*-параллельного полумарковского процесса

$$\mathbf{M}^K = \left\lfloor \mathrm{A}^K, \mathbf{h}^K(t) \right\rfloor, \tag{2.107}$$

где $A^K = \{a_{11}, ..., a_{k1}, ..., a_{K1}, a_{12}, ..., a_{k2}, ..., a_{K2}\}$ – множество состояний; $a_{11}, ..., a_{k1}, ..., a_{K1}$ – подмножество стартовых состояний; $a_{12}, ..., a_{k2}, ..., a_{K2}$ – подмножество поглощающих состояний; $\mathbf{h}^K(t)$ – полумарковская матрица;

$$\mathbf{h}^K(t) = \begin{bmatrix} \begin{bmatrix} 0 & v_1(t) \\ 0 & 0 \end{bmatrix} & & \cdots & & \mathbf{0} \\ & \cdots & & & & \\ & & \begin{bmatrix} 0 & v_k(t) \\ 0 & 0 \end{bmatrix} & & \cdots & \\ & & & \cdots & & \\ \mathbf{0} & & \cdots & & \begin{bmatrix} 0 & v_K(t) \\ 0 & 0 \end{bmatrix} \end{bmatrix}; \tag{2.108}$$

$$v_1(t) = ... = v_k(t) = ... v_K(t) = v(t) = \begin{cases} 1, \text{ when } 0 \le t \le 1; \\ 0, \text{ otherwise.} \end{cases} \tag{2.109}$$

K-параллельный процесс запускается изо всех состояний подмножества $a_{11}, ..., a_{k1}, ..., a_{K1}$ одновременно. Событие генерируется, когда один из ординар-

ных процессов, например k-й, достигает своего поглощающего состояния, $a_{k1}, 1 \le k \le K$. В соответствии с теоремой Б.Григелиониса [8], при $K \to \infty$ поток событий, генерируемых параллельно независимыми генераторами, стремится к пуассоновскому.

Плотность распределения интервала времени между началом процесса и достижением хотя бы одним процессом поглощающего состояния, для рассматриваемого конкретного случая определяется зависимостью

$$g_K(t) = \frac{d\left\{1 - [1 - V(t)]^K\right\}}{dt}, \tag{2.107}$$

где

$$V(t) = \int_0^t v(\tau) d\tau = \begin{cases} 2t, \text{ when } 0 \le t \le 1; \\ 0, \text{otherwise.} \end{cases}$$

Для случая (2.106)

$$g_K = \begin{cases} K(1-t)^{K-1}, \text{ when } 0 \le t \le 1; \\ 0, \text{ otherwise.} \end{cases} \tag{2.108}$$

Математическое ожидание для (22.108) определяется по зависимости

$$T_K = \int_0^1 tK(1-t)^{K-1} dt = \frac{1}{K+1}[\text{time}]. \tag{2.109}$$

Экспоненциальный закон, определяющий пуассоновский поток событий, имеет вид:

$$f_K(t) = (K+1)\exp[-(K+1)t]\left[\frac{\text{prob}}{\text{time}}\right]. \tag{2.110}$$

Для усеченного закона

$$\tilde{T}_K = \frac{K}{(K+1)^2}[\text{time}]. \tag{2.111}$$

$$\lim_{K\to\infty} \varepsilon_{lg}^K = \lim_{K\to\infty} \frac{T_K - \tilde{T}_K}{T_K} = \lim_{K\to\infty} \frac{1}{K+1} = 0, \tag{2.102}$$

т.е. с увеличением K закон приближается к экспоненциальному, что соответствует теореме Б. Григелиониса [8], и подтверждается введенным критерием,

основанном на вычислении функции ожидания. Вид плотностей распределения приведен на рис. 2.13.

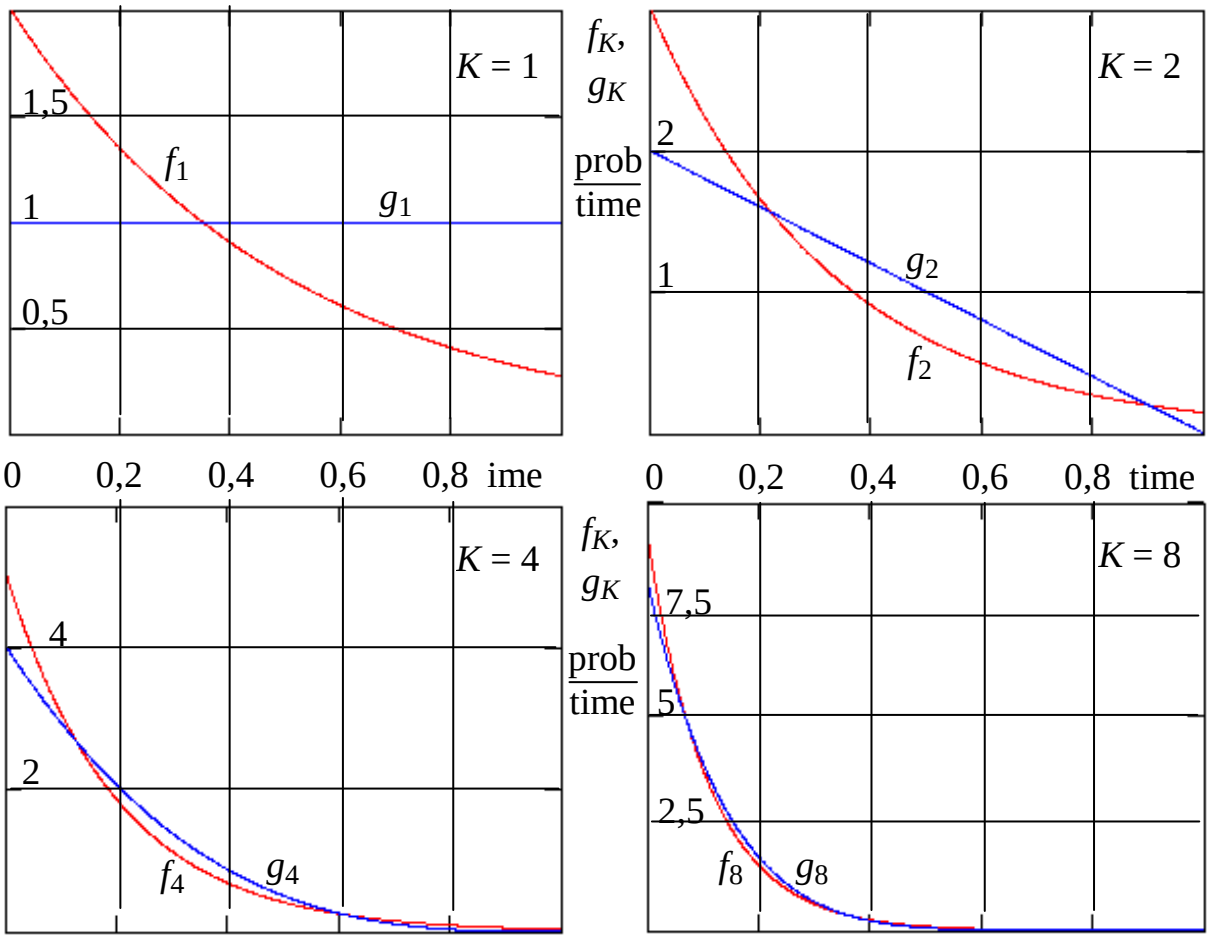


Рис. 2.13. Вид плотностей распределения для различного числа субъектов соревнования

2.6. Выводы

1) Определено понятие полумарковского процесса, который используется в качестве инструментария для моделирования состояний технологического процесса, показано, что полумарковские процессы являются математическим подобием циклограмм управления отдельными узлами и блоками роботов, а состояния полумарковских процессов связаны с выполнением оборудованием действий.

2) Исследованы основные свойства полумарковских процессов, вытекающие из свойств циклограмм управления узлами и блоками, показано, что полумарковские процессы являются возвратными и эргодическими.

3) Получены выражения для определения плотности распределения времени выполнения последовательности действий при реализации циклограмм управления; для указанных плотностей определены математическое ожидание и дисперсия времени выполнения.

4) Получено выражение для определения плотности распределения временного интервала выполнения одного из возможных действий при реализации циклограмм управления; для указанных плотностей определены математическое ожидание и дисперсия времени выполнения.

5) Из выражений для определения времени выполнения последовательности действий и времени выполнения одного из возможных действий получены матричные выражения для определения временных и вероятностных характеристик блуждания по полумарковской цепи.

6) С использованием понятия расщепления состояния получено выражение для плотности распределения времени блуждания из одного состояния эргодического полумарковского процесса в другое, для плотности распределения найдены зависимости для расчета математического ожидания и дисперсии.

3. МОДЕЛИ ДИСПЕТЧЕРИЗАЦИИ

3.0. Введение

Как было показано в [3, 4, 6, 14, 26, 38], управление гофроагрегатом сводится к генерации команд оператором с расположенного дистанционно пульта управления. Задачей системы цифрового управления второго (тактического) уровня иерархии является дешифрация команд, разделение команд по приоритетам и распределение команд по контроллерам (или передача на ЭВМ) третьего, функционально-логического уровня для реализации соответствующей циклограммы управления. Для правильного конструирования систем второго уровня иерархии необходимо иметь информацию о характеристиках потока команд, генерируемых оператором, а также информацию об ограничениях на временные интервалы, объективно определяемых физическими процессами, протекающими в блоках и узлах оборудования. Кроме того, весь поток команд, поступающих от оператора, должен быть разделен по приоритетам на команды мгновенного реагирования, и команды, исполнение которых может быть проведено с относительно большим запаздыванием.

Из замкнутой цепочки управления: оператор - пультовая ЭВМ - средства передачи данных - ЭВМ - исполнительные органы - гофроагрегат- сенсоры - ЭВМ - средства передачи данных - пультовая ЭВМ - оператор в настоящем разделе будет сформирована модель управления гофроагрегатом, описывающая генерацию команд и распределение команд по исполнительным органам технологического процесса. Из возможных способов распределения команд будет исследовано распределение с циклической и квазистохастической диспетчеризацией, а также исполнение срочных команд через режим прерывания.

3.1. Параллельный полумарковский процесс

3.1.1. Понятие параллельного полумарковского процесса

Как было показано в разделе 1, в состав технологического процесса входит значительное количество управляемых узлов и блоков, которые функционируют одновременно. Для правления каждым из них необходима реализация своей циклограммы, каждая из которых может быть представлена в виде ординарного полумарковского процесса. Очевидно, что множество циклограмм формирует новый полумарковский процесс, который включает множество ординарных процессов, эволюционирующих параллельно. Параллельная эволюция наделяет процесс качественно новыми свойствами, которые должны использоваться при управлении технологического процесса.

Вернемся к борелевской тройке (2.1), и представим множество элементарных событий Ω^a, формирующих состояния технологического процесса как объединение непересекающихся подмножеств:

$$^{M}\Omega^{a} = \bigcup_{m=1}^{M} {}^{m}\Omega_{m}^{a}, \tag{3.1}$$

где

$$\Omega_m^a = \left\{\omega_{1(a,m)}^a, ..., \omega_{j(a,m)}^a, ..., \omega_{J(a,m)}^a\right\}; \tag{3.2}$$

$$\Omega_m^a \bigcap \Omega_n^a = \begin{cases} \varnothing, \text{ when } m \neq n; \\ \Omega_m^a, \text{ when } m = n. \end{cases} \tag{3.3}$$

Применение функции (2.3) к множеству Ω_m^a дает множество состояний оборудования гофроагрегата;

$$\alpha\left(\Omega_{1m}^a\right) = \alpha\left\{\omega_{1(a,m)}^a, ..., \omega_{j(a,m)}^a, ..., \omega_{J(a,m)}^a\right\} = \left\{a_{1(a,m)}, ..., a_{j(a,m)}, ..., a_{J(a,m)}\right\} = A_m. \tag{3.4}$$

где $a_{j(a,m)}$ - $j(a,m)$-е физическое состояние m-й единицы оборудования гофроагрегата; $J(a,m)$.- мощность множества Ω_m^a.

Подмножеству Ω_m^a ставится в соответствие вероятностная мера

$$p_{j(a,m)} = P\left\lfloor {}^m a_m : a_{j(a,m)} \in A_m \right\rfloor, \tag{3.5}$$

которая характеризует вероятность пребывания m-й единицы бортового оборудования технологического процесса в одном из состояний множества A_m для внешнего по отношению к технологического процесса наблюдателю. Тот факт, что m-я единица бортового оборудования может находиться в одном и только в одном из состояний, накладывает следующее ограничение на вероятности (2.5):

$$\sum_{j(a,m)=1(a,m)}^{J(a,m)} p_{j(a,m)} = 1. \tag{3.6}$$

Состояние m-й единицы бортового оборудования меняется в определенные моменты времени, поэтому можно ввести понятие переключения переключением оборудования из состояния $a_{j(a,m)}$ в состояние $a_{n(a,m)}$:

$$s_{j(a,m),n(a,m)} = \left\lfloor a_{j(a,m)}, a_{n(a,m)} \right\rfloor \in S_m, \tag{3.7}$$

где S_m - множество возможных переключений;

$$\begin{aligned} S_m = \{ & s_{1(a,m),1(a,m)}, \ldots, s_{1(a,m),n(a,m)}, \ldots, s_{1(a,m),J(a,m)}, \ldots, \\ & s_{j(a,m),1(a,m)}, \ldots, s_{j(a,m),n(a,m)}, \ldots, s_{j(a,m),J(a,m)}, \ldots, \\ & s_{J(a,m),1(a,m)}, \ldots, s_{J(a,m),n(a,m)}, \ldots, s_{J(a,m),J(a,m)}, \ldots \}. \end{aligned} \tag{3.8}$$

Каждой паре $\left\lfloor a_{j(a),m}, a_{n(a,m)} \right\rfloor$, $1(a,m) \le j(a,m), n(a,m) \le J(a,m)$ из (89) может быть поставлена в соответствие вероятностная мера

$$p_{k(a,m),j(a,m)} = P\left\lfloor s\left(a^-, a^+\right):\ s_{j(a,m),n(a,m)} = \left[a_{j(a,m)}, a_{n(a,m)}\right] \in S_m \right\rfloor, \tag{3.9}$$

где a^- - состояние технологического процесса до переключения; a^+ - состояние технологического процесса после переключения.

Переключения оборудования технологического процесса развиваются в едином времени, определенном зависимостью (2.13), Если ввести допущение, что

переключения каждой из M единиц оборудования не зависят от текущего состояния остальных единиц, то может быть построена диаграмма, определяющая фактор времени (рис.3.1).

Отсчет времени начинается для каждого из M параллельных процессов в момент переключения этого процесса из состояния $a_{j(a,m)}$ в состояние $a_{n(a,m)}$. На оси глобального времени t_G, таким образом формируется поток переключений, который получается путем объединения потоков переключений единиц оборудования технологического процесса. В литературе [1, 37, 67, 76, 79, 94, 102, 116] объединение потоков называется суммированием, и доказывается, что суммарный поток является пуассоновским.

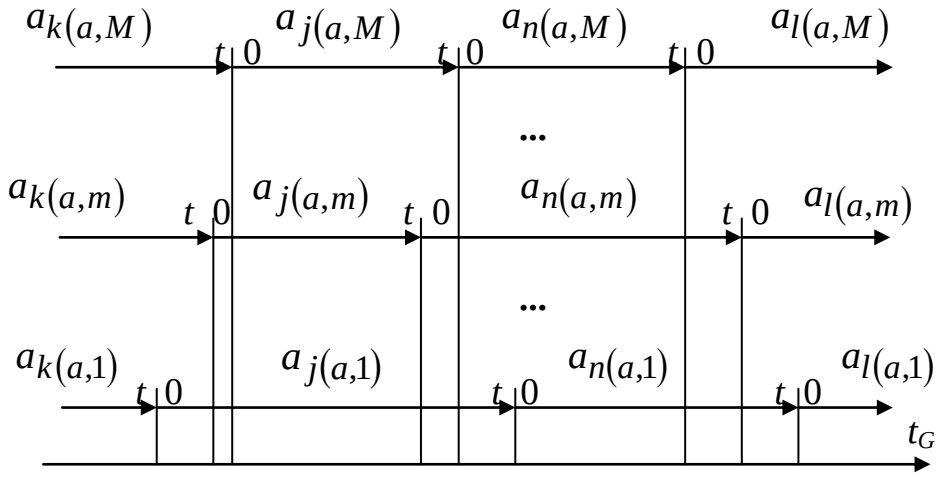


Рис. 3.1. Формирование фактора времени

Для каждой единицы оборудования последовательность событий переключения состояний формирует поток временных интервалов, каждый из которых является случайной непрерывной величиной. В этом случае для $j(t)$-го момента времени $j(a,m)$-го интервала потока, если известно, что следующим переключением будет ${}^{m}s_{j(a,m),n(a,m)}$, может быть определена вероятностная мера

$$f_{j(a,m),n(a,m)}(t_{j(t)})dt = P\lfloor t : t(s_{k(a,m),\,j(a,m)}) = 0,\, t(s_{j(a,m),n(a,m)}) = t_{j(t)} \rfloor, \quad (3.10)$$

где $t\left(s_{k(a,m),j(a,m)}\right)$ - момент переключения в состояние $a_{j(a,m)}$; $t\left(s_{j(a),n(a)}\right)$ - момент переключение из состояния $a_{j(a,m)}$, если принято решение, что следующим состоянием будет $a_{n(a),m}$.

Плотность распределения (3.10) не зависит от предыстории переключений. а зависит от того, в какое состояние технологического процесса переключится следующий раз.

3.1.2. Простейший *M*-параллельный полумарковский процесс

Можно считать, что Простейший *M*-параллельный полумарковский процесс состоит из *M* подпроцессов $\boldsymbol{r}_m = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \boldsymbol{h}_m(t) = \begin{bmatrix} 0 & f_m(t) \\ 0 & 0 \end{bmatrix}$, $1 \le m \le M$. и имеет вид (рис. 3.2);

$$^{M}\mu_M = \left\{^{M}A, ^{M}\mathbf{r}, ^{M}\mathbf{h}(t)\right\}, \tag{3.11}$$

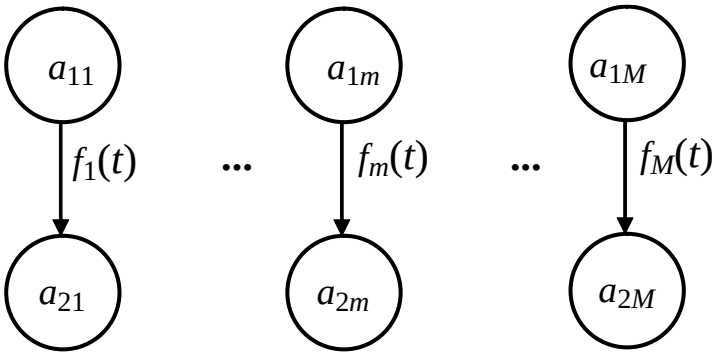


Рис. 3.2. Простейший *M*-параллельный полумарковский процесс

где состояния $\{a_{11}, ..., a_{1m}, ..., a_{1M}\}$ моделируют старт полумарковского процесса; состояния $\{a_{21}, ..., a_{2m}, ..., a_{2M}\}$ являются поглощающими: $f_m(t)$ - плотность распределения времени пребывания процесса в состоянии a_{1m} до его переключения в состояние a_{m2}, $1 \le m \le M$; $^{M}\boldsymbol{r}$ - матрица смежности; $^{M}\mathbf{h}(t)$ - полумарковская матрица;

$$^{M}A = \{a_{11}, ..., a_{1m}, ..., a_{1M}\} \cup \{a_{21}, ..., a_{2m}, ..., a_{2M}\}; \quad (3.12)$$

$$^{M}\boldsymbol{r} = \begin{bmatrix} 0 & 1 & & & & & & \\ 0 & 0 & & & & & 0 & \\ & & \ldots & & & & & \\ & & & 0 & 1 & & & \\ & & & 0 & 0 & & & \\ & & & & & \ldots & & \\ & 0 & & & & & 0 & 1 \\ & & & & & & 0 & 0 \end{bmatrix}; \quad (3.13)$$

$$^{M}\boldsymbol{h}(t) = \begin{bmatrix} 0 & f_1(t) & & & & & & \\ 0 & 0 & & & & & 0 & \\ & & \ldots & & & & & \\ & & & 0 & f_m(t) & & & \\ & & & 0 & 0 & & & \\ & & & & & \ldots & & \\ & 0 & & & & & 0 & f_M(t) \\ & & & & & & 0 & 0 \end{bmatrix}. \quad (3.14)$$

Для простейшего *M*-параллельного полумарковского процесса справедливо следующее утверждение.

Утверждение 3.1. Взвешенные плотности распределения времени переключения в поглощающие состояния первым, $h_{wm}(t)$, $1 \le m \le M$. удовлетворяют условию

$$\sum_{m=1}^{M} h_{wm}(t) = \frac{d}{dt}\left\{1 - \prod_{m=1}^{M}[1 - F_m(t)]\right\}, \quad (3.15)$$

где $F_m(t)$ - функции распределения, соответствующие плотностям $f_m(t)$, $F_m(t) = \int_0^t f_m(\tau)d\tau$.

Утверждение доказывается методом математической индукции. *Базой индукции* является случай, когда $M = 2$ (2-параллельный процесс). Для случая, когда первый процесс из двух побеждает в «соревновании может быть построен ординарный полумарковский процесс, приведенный на рис. 3.3.

В полумарковском процессе, приведенном на рис. 3.1,

α - стартовое состояние, соответствующее одновременному началу подпроцесов $\boldsymbol{h}_1(t)$ и $\boldsymbol{h}_2(t)$;

α_{w1} - поглощающее состояние, соответствующее случаю, когда первым завершился $\boldsymbol{h}_1(t)$, а $\boldsymbol{h}_2(t)$ еще не достиг поглощающего состояния $a_{2,2}$;

α_{w2} - поглощающее состояние, соответствующее случаю, когда первым завершился подпроцесс $\boldsymbol{h}_2(t)$, а подпроцесс $\boldsymbol{h}_1(t)$ еще не достиг поглощающего состояния $a_{1,2}$;

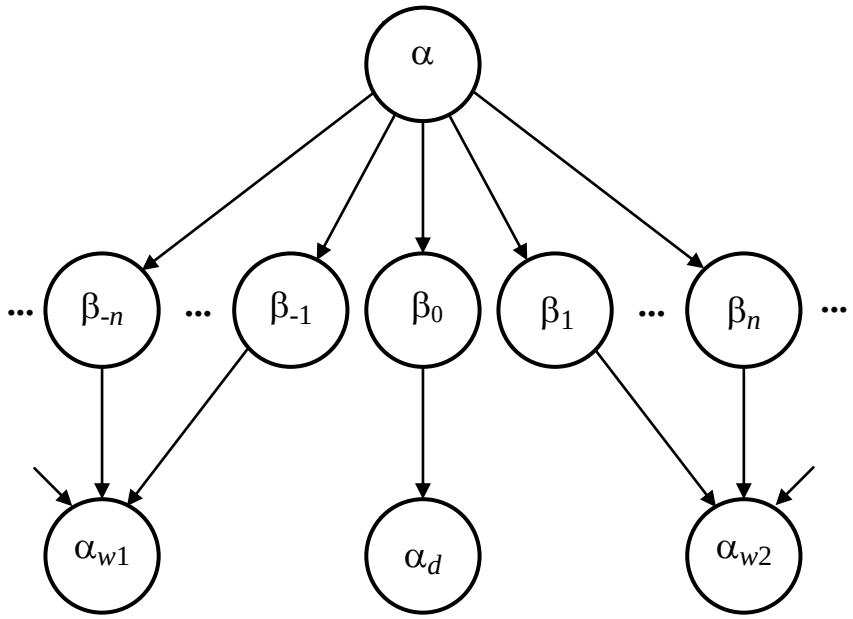


Рис. 3.3. Ординарный полумарковский процесс
описывающий «победу» в «соревновании» первого процесса

α_d - поглощающее состояние, соответствующее случаю, когда подпроцессы $\boldsymbol{h}_1(t)$ и $\boldsymbol{h}_2(t)$ достигают своих поглощающих состояний $a_{1,2}$ и $a_{2,2}$, соответственно, одновременно;

β_{-n} - состояние, моделирующее достижение подпроцессом $\boldsymbol{h}_1(t)$ состояния $a_{1,2}$ при условии, что подпроцесс $\boldsymbol{h}_2(t)$ еще не достиг состояния $a_{2,2}$, за время $n\Delta_t$, где Δ_t - квант времени;

β_n - состояние, моделирующее достижение подпроцессом $\boldsymbol{h}_2(t)$ состояния $a_{2,2}$ при условии, что подпроцесс $\boldsymbol{h}_1(t)$ еще не достиг состояния $a_{1,2}$, за время $n\Delta_t$;

β_n - состояние, моделирующее одновременное достижение подпроцессами $\boldsymbol{h}_1(t)$ и $\boldsymbol{h}_2(t)$ состояний $a_{1,2}$ и $a_{2,2}$ за время $n\Delta_t$.

Определим плотности распределения времени достижения поглощающих состояний α_{w1}, α_{w2}, α_d из начального состояния α:

вероятность того, что за время $t = n\Delta_t$ будет достигнуто состояние α_{w1} с временной задержкой, определяемой состоянием β_{-n}, равна $P_{-n}(t = n\Delta_t) = [1 - F_2(n\Delta_t)] \cdot f_1(n\Delta_t)\Delta_t$, где $F_2(n\Delta_t)$ - функция распределения, соответствующая плотности $f_2(t)$;

вероятность того, что за время $n\Delta_t$ будет достигнуто состояние α_{w2} с временной задержкой, определяемой состоянием β_n, равна $P_{-n}(t = n\Delta_t) = [1 - F_1(n\Delta_t)] \cdot f_2(n\Delta_t)\Delta_t$, где $F_1(t)$ - функция распределения, соответствующая плотности $f_1(t)$;

вероятность того, что за время $n\Delta_t$ будет достигнуто состояние α_d с временной задержкой, определяемой состоянием β_0, равна $P_0(t = n\Delta_t) = f_1(n\Delta_t) \cdot f_2(n\Delta_t)\Delta_t^2$, и имеет больший порядок малости, чем вероятности $P_{-n}(t = n\Delta_t)$ и $P_n(t = n\Delta_t)$.

Взвешенная плотность распределения времени достижения состояний α_{w1} и α_{w2} определяется по зависимостям

$$h_{w1}(t) = \lim_{\substack{n\to\infty \\ \Delta_t \to 0}} \frac{P_{-n}(t = n\Delta_t)}{\Delta_t} = [1 - F_2(t)] \cdot f_1(t); \qquad (3.16)$$

$$h_{w2}(t) = \lim_{\substack{n\to\infty \\ \Delta_t \to 0}} \frac{P_n(t = n\Delta_t)}{\Delta_t} = [1 - F_1(t)] \cdot f_2(t). \qquad (3.17)$$

Сумма взвешенных плотностей распределения (3.16) и (3.17) равна единице, поскольку других исходов «соревнования» быть не может.

Из предположения, что для (*M*-1)-параллельного полумарковского процесса справедливо выражение $\sum_{m=1}^{M-1} h_{wm}(t) = \frac{d}{dt}\left\{1 - \prod_{m=1}^{M-1}[1 - F_m(t)]\right\}$, следует справедливость (3.15).

Из (3.15) может быть получены вероятность того, что процесс $\boldsymbol{h}_m(t)$ достигнет состояния a_{m2} первым из *M*-параллельных процессов:

$$p_{wm} = \int_0^{\infty} f_m(t) \cdot \prod_{\substack{n=1 \\ n \neq m}}^{M} [1 - F_n(t)] dt . \qquad (3.18)$$

Плотность распределения времени достижения процессом $\boldsymbol{h}_m(t)$ состояния a_{m2} первым из *M*-параллельных процессов определяется по зависимости:

$$f_{wm}(t) = \frac{f_m(t) \cdot \prod_{\substack{n=1 \\ n \neq m}}^{M} [1 - F_n(t)]}{p_{wm}} . \qquad (3.19)$$

Числовые характеристики плотности распределения (2.121) определяются по зависимостям

$$T_{wm} = \int_0^{\infty} t \frac{f_m(t) \cdot \prod_{\substack{n=1 \\ n \neq m}}^{M} [1 - F_n(t)]}{p_{wm}} dt ; \qquad (3.20)$$

$$D_{wm} = \int_0^{\infty} (t - T_{wm})^2 \frac{f_m(t) \cdot \prod_{\substack{n=1 \\ n \neq m}}^{M} [1 - F_n(t)]}{p_{wm}} dt , \; 1 \le m \le M.$$

Утверждение 3.2. Взвешенные плотности распределения $h_{\overline{w}m}(t)$, $1 \le m \le M$. времени переключения процесса (3.21) в поглощающие состояния $a_{m,2}$, $1 \le \le m \le M$, удовлетворяют условию

$$\sum_{m=1}^{M} h_{\overline{w}m}(t) = \frac{d}{dt} \prod_{m=1}^{M} F_m(t), \tag{3.21}$$

Доказывается утверждение 3.2 аналогично утверждению 3.1, методом математической индукции.

Из утверждения 3.2 следует, что

$$f_{\overline{w}m}(t) = \frac{f_m(t) \cdot \prod\limits_{\substack{n=1 \\ n \neq m}}^{M} F_n(t)}{p_{\overline{w}j}(a)}. \tag{3.22}$$

$$p_{\overline{w}m} = \int_0^{\infty} f_m(t) \cdot \prod_{\substack{n=1 \\ n \neq m}}^{M} F_n(t) dt. \tag{3.23}$$

Зависимости (2.103) и (2.104) определяют плотность распределения времени и вероятность того, что m-й полумарковский процесс достигнет своего состояния $a_{m,2}$ последним из M-параллельных процессов.

Рассмотрим случай, когда важен факт достижения поглощающих состояний $a_{m,2}$ любыми K из M-параллельных процессов, $K < M$. Создадим множество N_M M-значных двоичных чисел натурального ряда, m-й разряд которых, σ_m, закреплен за полумарковским процессом $\boldsymbol{h}_m(t)$, и может принимать два значения:

$$\sigma_m = \begin{cases} 0, \text{if } m\text{ - th process gets } a_{m2}; \\ 1, \text{if } m\text{ - й th process not gets } a_{m2}. \end{cases} \tag{3.24}$$

Выберем из множества N_M подмножество $N_M^K \subset N_M$ двоичных M-разрядных чисел, которые имеют K единиц и M - K нулей:

$$N_M^K = \left\{ n_1, ..., n_{c_M^K}, ..., n_{C_M^K} \right\}, \tag{3.25}$$

где C_M^K - K-й биномиальный коэффициент M-й степени; $\tilde{n}_M^K$ - номер числа в множестве (3.24);

$$C_M^K = \frac{M!}{K \cdot (M-K)!}. \tag{3.26}$$

$$n_{c_M^K} = \left\langle \sigma_1^{c_M^K}, ..., \sigma_m^{c_M^K}, ..., \sigma_M^{c_M^K} \right\rangle. \quad (3.27)$$

Введем функцию $\Phi\left(f_m, \sigma_m^{c_M^K} \right)$, которая принимает вид

$$\Phi\left(f_m, \sigma_m^{c_M^K} \right) = \begin{cases} F_m(t), \text{ when } \sigma_m^{c_M^K} = 1; \\ [1 - F_m(t)], \text{ when } \sigma_m^{c_M^K} = 0. \end{cases} \quad (3.28)$$

С учетом (3.28) общая зависимость для определения функции распределения времени достижения состояний $a_{m,2}$ любыми K из M-параллельных процессов будет иметь вид:

$$F_M^K(t) = \sum_{c_M^K=1}^{C_M^K} \prod_{m=1}^{M} \Phi\left(f_m, \sigma_m^{c_M^K} \right). \quad (3.29)$$

Дифференцируя (3.29) по времени, получим плотность распределения времени достижения состояний $a_{m,2}$ любыми K из M-параллельных процессов:

$$f_M^K(t) = \frac{d \sum_{c_M^K=1}^{C_M^K} \prod_{m=1}^{M} \Phi\left(f_m, \sigma_m^{c_M^K} \right)}{dt}. \quad (3.30)$$

Для плотности (3.30) могут быть найдены характеристики:

$$T_M^K = \int_0^\infty t \cdot d \sum_{c_M^K=1}^{C_M^K} \prod_{m=1}^{M} \Phi\left(f_m, \sigma_m^{c_M^K} \right). \quad (3.31)$$

$$D_M^K = \int_0^\infty \left(t - T_M^K\right)^2 d \sum_{c_M^K=1}^{C_M^K} \prod_{m=1}^{M} \Phi\left(f_m, \sigma_m^{c_M^K} \right).$$

3.1.3. Сложный *M*-параллельный полумарковский процесс

Сложный *M*-параллельный полумарковский процесс состоит из ординарных полумарковских процессов самого общего вида (рис. 3.4). Потребуем, чтобы сложный *M*-параллельный полумарковский процесс состоял из процессов (3.32);

$$^M\mu' = \left\{ ^M A', ^M \mathbf{r}', ^M \mathbf{h}'(t) \right\}, \quad (3.32)$$

где ${}^{M}A'$ - множество состояний; ${}^{M}\mathbf{r}'$ - матрица смежности; ${}^{M}\mathbf{h}'(t)$ 0 полумарковская матрица.

Введем понятия структурного и функционального состояний. Под структурным состоянием ниже будет пониматься состояние отдельного полумарковского процесса, которое управляет выполнением определенного действия МР. В общем случае можно считать, что *m*-й полумарковский процесс включает *J*(*a*, *m*)+1 структурных состояний:

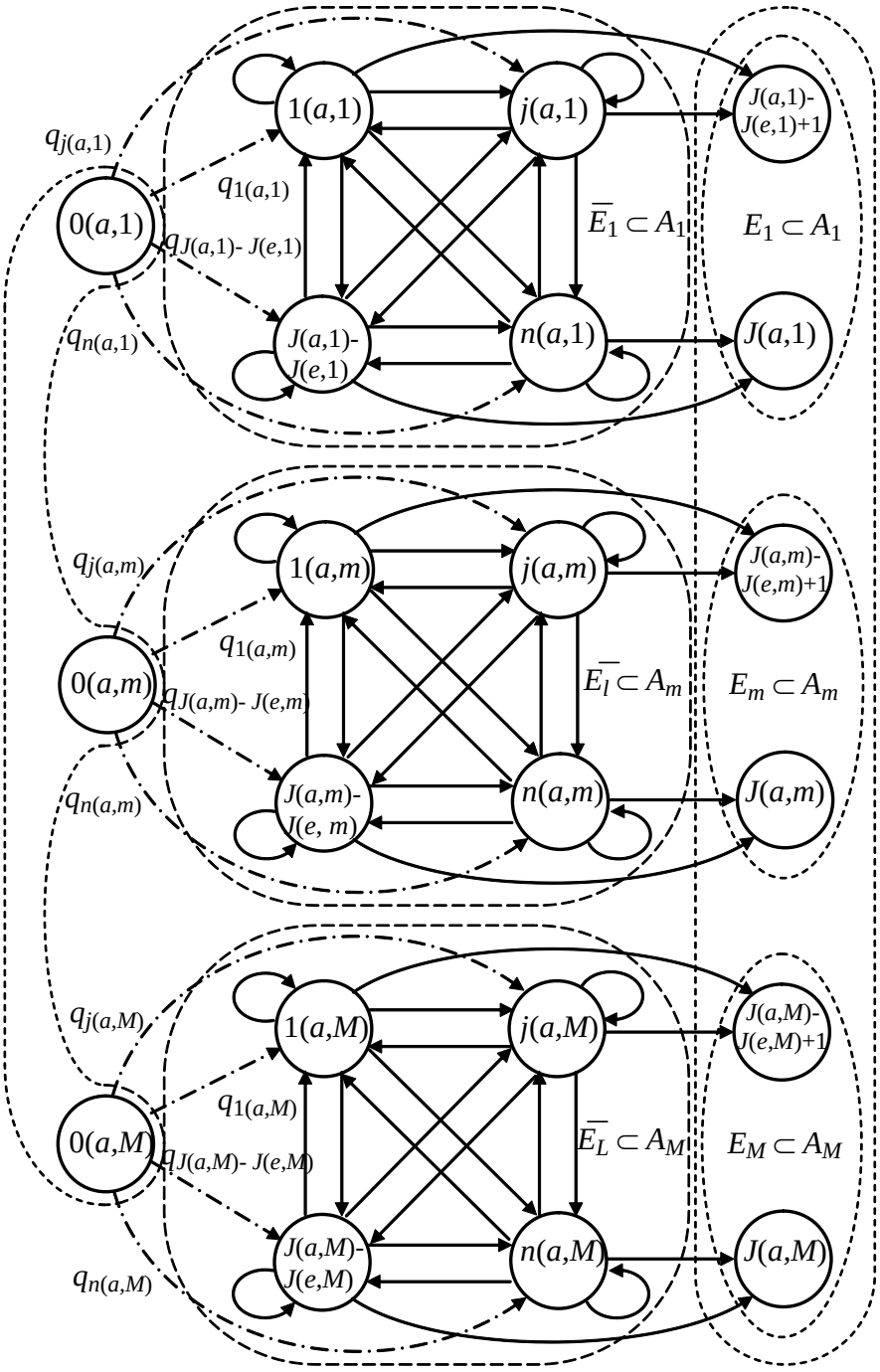


Рис. 3.4. Структура сложного *M*-параллельного полумарковского процесса

$$A'_m = \{a_{0(a,m)}, a_{1(a,m)}, ..., a_{j(a,m)}, ..., a_{J(a,m)}\},\ 1 \le m \le M; \qquad (3.33)$$

$A_m \cap A_n = \varnothing$ при $m \ne n$.

Таким образом, общее количество структурных состояний сложного процесса определяется суммой $N(M) = \sum_{m=1}^{M} |A_m| = M + \sum_{m=1}^{M} J(a,m)$.

Функциональное состояние M-параллельного полумарковского процесса может быть определено как вектор $\alpha_{j(\alpha)}$в пространстве функциональных состояний.

Определим *множество состояний* сложного M-параллельного полумарковского процесса с учетом того, что согласно (3.33) каждое множество A'_m структурных состояний делится на непересекающиеся подмножества

$$A'_m = B'_m \cup \bar{E}'_m \cup E'_m, \qquad (3.34)$$

$$B'_m \cap \bar{E}'_m = \varnothing;\ B'_m \cup E'_m = \varnothing;\ \bar{E}'_m \cup E'_m = \varnothing,$$

где B'_m - стартовое состояние E'_m - подмножество поглощающих состояний $\bar{E}'_m$ - подмножество поглощающих состояний;

$$B'_m = \{a_{0(a,m)}\}; \qquad (3.35)$$

$$\bar{E}'_m = \{a_{1(a,m)}, ..., a_{j(a,m)}, ..., a_{J(a,m)-J(e,m)}\}; \qquad (3.35)$$

$$E'_m = \{a_{J(a,m)-J(e,m)+1}, ..., a_{j(e,m)}, ..., a_{J(a,m)}\}. \qquad (3.36)$$

Справедливо следующее утверждение.

Утверждение 3.3. Стартовое функциональное состояние является единственным.

Доказательство. Выделим из структурных состояний стартовые состояния:

$$^{M}a_9 = \left[a_{0(a,1)}, ..., a_{0(a,m)}, ..., a_{0(a,M)}\right]. \qquad (3.37)$$

В соответствии с (3.37), время пребывания m-го ординарного процесса в состоянии $a_{0(a,m)}$ определяется несмещенной δ-функцией Дирака, следовательно,

при старте все процессы переключаются из состояния ${}^{M}a_9$ одновременно, что и доказывает утверждение.

Все остальные функциональные состояния ${}^{M}A$ A_α могут быть получены как произведение множеств A'_m, что с учетом утверждения 3.3 дает

$$ {}^{M}A' = {}^{M}a_0 \cup \coprod_{m=1}^{M} {}^{D}\left(\overline{E}'_m \cup E'_m\right), \tag{3.38}$$

где $\prod^{D}$ - знак группового декартова произведения множеств.

Из (3.38) следует, что ${}^{M}A'$ помимо ${}^{M}a_0$ включает в себя;

подмножество непоглощающих состояний ${}^{M}\overline{E}$, функциональные состояния которого формируются только из структурных непоглощающих состояний;

подмножество поглощающих состояний ${}^{M}E$, функциональные состояния которого формируются только из структурных поглощающих состояний;

подмножество полупоглощающих состояний ${}^{M}\hat{E}$ в функциональные состояния которого входят как структурных поглощающих состояния, так и структурные непоглощающих состояния.

Перечисленные подмножества определяются следующим образом;

$$ {}^{M}\overline{E} = \coprod_{m=1}^{M} {}^{D}\overline{E}'_m ; \tag{3.39}$$

$$ {}^{M}E = \coprod_{m=1}^{M} {}^{D}E'_m ; \tag{3.40}$$

$$ {}^{M}\hat{E} = \left[\left({}^{M}A' \setminus {}^{M}E'\right) \setminus {}^{M}\overline{E}'\right] \setminus {}^{M}a_0 \tag{3.41}$$

Общее количество функциональных состояний сложного *M*-параллельного полумарковского процесса определяется зависимостью

$$ J(\alpha) = \left|{}^{M}A'\right| = 1 + \prod_{m=1}^{M} J(a,m). \tag{3.42}$$

Определим *матрицу смежности* сложного *M*-параллельного полумарковского процесса. Для этого исключим из матриц смежности $\mathbf{r}'_m$, $1 \le m \le M$, строки

$\mathbf{r}'_{0(a,m)}$ с номером $0(a,m)$ и столбцы с номером $0(a,m)$ (они являются нулевыми), преобразовав их таким образом снова (см. (3.34)) в матрицы

$$\mathbf{r}_m = \lfloor r_{j(a,m),n(a,m)} \rfloor. \tag{3.43}$$

Элемент, находящийся на пересечении $j(a,m)$-й строки и $n(a,m)$-го столбца матрицы (3.43) означает, что в графе, представляющем структуру процесса, имеется дуга, ведущая из вершины с номером строки в вершину с номером столбца. Введем операцию декартова произведения $J(k)\times J(k)$ матрицы смежности $\mathbf{r}_k = \lfloor r_{j(a,k),n(a,k)} \rfloor$ на $J(m)\times J(m)$ матрицу смежности $\mathbf{r}_m = \lfloor r_{j(a,m),n(a,m)} \rfloor$;

$$\mathbf{r}_k \times \mathbf{r}_m = \lfloor r_{[j(k),\,j(m)],[n(k),n[m]]} \rfloor, \tag{3.44}$$

где

$$r_{[j(k),\,j(m)],[n(k),n[m]]} = \begin{cases} 1, \text{ when } j(k) = n(k),\ j(l) \neq n(m) \text{ и } r_{j(m),n[m]} = 1, \\ \text{or when } j(k) \neq n(k),\ j(m) = n(m), \text{ и } r_{j(k),n[k]} = 1; \\ 0 \text{ otherwise.} \end{cases} \tag{3.45}$$

Для того, чтобы получить декартово произведение исходных матриц с неисключенными строкой и столбцом, $\mathbf{r}'_k \times \mathbf{r}'_m$, необходимо в матрицу, сформированную в соответствии с зависимостью (3.45) добавит нулевой столбец, состоящий из нулей и нулевую строку, формируемую следующим образом. Строки $\mathbf{r}'_{0(a,m)}$ и $\mathbf{r}'_{0(a,k)}$ декартово перемножаются, $\mathbf{r}'_{0(a,m)} \times \mathbf{r}'_{0(a,k)}$, в результате чего формируется вектор из двоек. В соответствии с определением матрицы смежности, элементы двоек могут принимать значения, либо ноль, либо единица. Номер двойки соответствует номеру столбца в декартовом произведении. Если в двойке все элементы равны единице, то на пересечении нулевой строки и столбца, соответствующей двойки ставится единица, и ноль в противном случае.

С учетом введенной операции декартова произведения, матрица смежности ${}^{M}\mathbf{r}_\alpha = \lfloor {}^{M} r_{j(\alpha),n(\alpha)} \rfloor$ будет иметь вид

$${}^{M}\mathbf{r}_\alpha = \prod_{m=1}^{M} {}^{D}\mathbf{r}_m, \tag{3.46}$$

где D - символ, означающий операцию группового декартова произведения матриц смежности.

Для того, чтобы получить декартово произведение исходных матриц с неисключенными строкой и столбцом, $\mathbf{r}'_k \times \mathbf{r}'_m$, необходимо в матрицу, сформированную в соответствии с зависимостью (3.45) добавит нулевой столбец, состоящий из нулей и нулевую строку, формируемую следующим образом. Строки $\mathbf{r}'_{0(a,1)}$, ..., $\mathbf{r}'_{0(a,m)}$, ..., $\mathbf{r}'_{0(a,M)}$ декартово перемножаются, $\mathbf{r}'_{0(a,1)} \times ... \times \mathbf{r}'_{0(a,m)} \times ... \times \mathbf{r}'_{0(a,M)}$, в результате чего формируется вектор из М-к. В соответствии с определением матрицы смежности, элементы двоек могут принимать значения, либо ноль, либо единица. Номер двойки соответствует номеру столбца в декартовом произведении. Если в двойке все элементы равны единице, то на пересечении нулевой строки и столбца, соответствующей двойки ставится единица, и ноль в противном случае.

Сформированная матрица смежности ${}^{M}\mathbf{r}'$ будет иметь размеры $\left[\left(\prod_{m=1}^{M} J(a,m)\right)+1\right] \times \left[\left(\prod_{m=1}^{M} J(a,m)\right)+1\right]$.

Для определения значений элементов стохастической матрицы воспользуемся утверждениями 3.1 и 3.2 определяющими вероятностные и временные характеристики между переключениями в сопряженные состояния в параллельных полумарковских процессах. В полумарковской матрице

$$\mathbf{h}_k(t) \times \mathbf{h}_m(t) = \left(h_{[j(k),\,j(m)],[n(k),n(m)]}(t)\right) \qquad (3.47)$$

в строке с номером имеет место [$J(k)$+ $J(m)$]-2-параллельный полумарковский процесс, определяемый векторами $\left\lfloor h_{j(k),1(k)}, ..., h_{j(k),n(k)}, ..., h_{j(k),J(k)} \right\rfloor$.

Плотность распределения времени переключения $J(k)$-параллельного процесса определяется по зависимости:

$$f_{j(k)}(t) = \sum_{n(k)=1(k)}^{J(k)} h_{j(k),n(k)}(t). \qquad (3.48)$$

Плотность распределения времени переключения $J(m)$-параллельного процесса определяется по зависимости:

$$f_{j(m)}(t) = \sum_{n(m)=1(m)}^{J(m)} h_{j(m),n(m)}(t) \quad . \tag{3.49}$$

Вероятности того, что в $[j(k), j(m)]$-й строке в первую очередь переключение произойдет в *k*-м/*m*-м процессе, определяются зависимостями

$$q^k_{j(k),\,j(m)} = \int_0^\infty f_{j(k)}(t) \cdot \left[1 - F_{j(m)}(t)\right] dt \,. \tag{3.50}$$

$$q^m_{j(k),\,j(m)} = \int_0^\infty f_{j(m)}(t) \cdot \left[1 - F_{j(m)}(t)\right] dt \,. \tag{3.51}$$

Таким образом,

$$\mathbf{p}_k(t) \times \mathbf{p}_m(t) = \left\lfloor p_{[j(k),\,j(m)],[n(k),n(m)]} \right\rfloor, \tag{3.52}$$

где

$$p_{[j(k),\,j(m)],[n(k),n(m)]} = \begin{cases} p_{j(m),n(m)} \cdot q^m_{j(k),\,j(m)}, \text{ when } j(k) = n(k),\ j(m) \neq n(m); \\ 0 \text{ otherwise}; \\ p_{j(k),n(k)} \cdot q^k_{j(k),\,j(m)} \text{ when } j(k) \neq n(k),\ j(m) = n(m). \end{cases}$$

С учетом введенной операции матрица вероятностей примет вид

$$^M\mathbf{p}_\alpha = \prod_{m=1}^{M} {}^D\mathbf{p}_m \,. \tag{3.53}$$

Матрица плотностей распределения, соответствующая декартову произведению матриц, будет иметь вид

$$\mathbf{f}_k(t) \times \mathbf{f}_m(t) = \left\lfloor f_{[j(k),\,j(m)],[n(k),n(m)]}(t) \right\rfloor, \tag{3.54}$$

где

$$f_{[j(k),\,j(m)],[n(k),n(m)]}(t) = \begin{cases} f_{j(m),n(m)}(t), \text{ when } j(k) = n(k),\ j(m) \neq n(m); \\ f_{j(k),n(k)}(t), \text{ when } j(k) \neq n(k),\ j(m) = n(m); \\ 0 \text{ otherwise}. \end{cases} \tag{3.55}$$

С учетом введенной операции матрица плотностей распределения примет вид

$$^M\mathbf{f}_\alpha(t) = \prod_{m=1}^{M} {}^D\mathbf{f}_m(t). \tag{3.56}$$

Полумарковская матрица сложного *M*-параллельного полумарковского процесса имеет вид

$$
\begin{aligned}
{}^{M}\mathbf{h}_{\alpha}(t) &= {}^{M}\mathbf{p}_{\alpha} \otimes {}^{M}\mathbf{f}_{\alpha}(t) = \\
&= \left[{}^{M}p_{j(\alpha),m(\alpha)} \cdot {}^{M}f_{j(\alpha),m(\alpha)}(t) \right] = \left[{}^{M}h_{j(\alpha),m(\alpha)}(t) \right],
\end{aligned}
\tag{3.57}
$$

где ${}^{M}h_{j(\alpha),m(\alpha)}(t)$ - взвешенная плотность распределения сложного *M*-параллельного полумарковского процесса при переключении из состояния

Полумарковская матрица (3.57) будет иметь размеры $\left[\prod_{m=1}^{M} J(a,m)\right] \times \left[\prod_{m=1}^{M} J(a,m)\right]$. Полумарковские матрицы $\mathbf{h}_{m}(t)$, $1 \le m \le M$, получаются из матриц $\mathbf{h}'_{m}(t)$ путем исключения строк $\mathbf{h}'_{0(a,m)}(t)$ с номером $0(a,m)$ и столбцов с номером $0(a,m)$ (они являются нулевыми). Для того, чтобы получить декартово произведение исходных матриц с неисключенными строками $\mathbf{h}'_{0(a,m)}$, необходимо в матрицу, сформированную в соответствии с зависимостью (3.57) добавить нулевой столбец, состоящий из нулей, и нулевую строку, формируемую как алгебраическое декартово произведение векторов

$$
\mathbf{h}'_{0(\alpha)}(t) = \prod_{m=1}^{M} {}^{D}\mathbf{h}'_{0(a,m)}(t). \tag{3.58}
$$

В результате формируется полумарковская матрица ${}^{M}\mathbf{h}'_{\alpha}(t)$ размером $\left[1 + \prod_{m=1}^{M} J(a,m)\right] \times \left[1 + \prod_{m=1}^{M} J(a,m)\right]$. Строка $\mathbf{h}'_{0(\alpha)}(t)$ описывает переключения из стартового состояния сложного *M*-параллельного полумарковского процесса.

Таким образом, из сложного *M*-параллельного полумарковского процесса, состоящего из ординарных полумарковских процессов самого общего вида сформирован процесс ${}^{M}\mu'$ (3.58), который является ординарным полумарковским процессом с функциональными состояниями. Для определения стохастических и временных характеристик блужданий по ординарному полумарковскому процес-

су с функциональными состояниями могут быть использованы метода, изложенные в п. 2.3.3 и 2.3.4.

3.2. Генерация команд управления в диалоговом режиме

3.2.1. Полумарковский процесс генерации команд

Структура полумарковского процесса, моделирующего генерацию команд в интерактивном режиме, приведена на рис. 3.5.

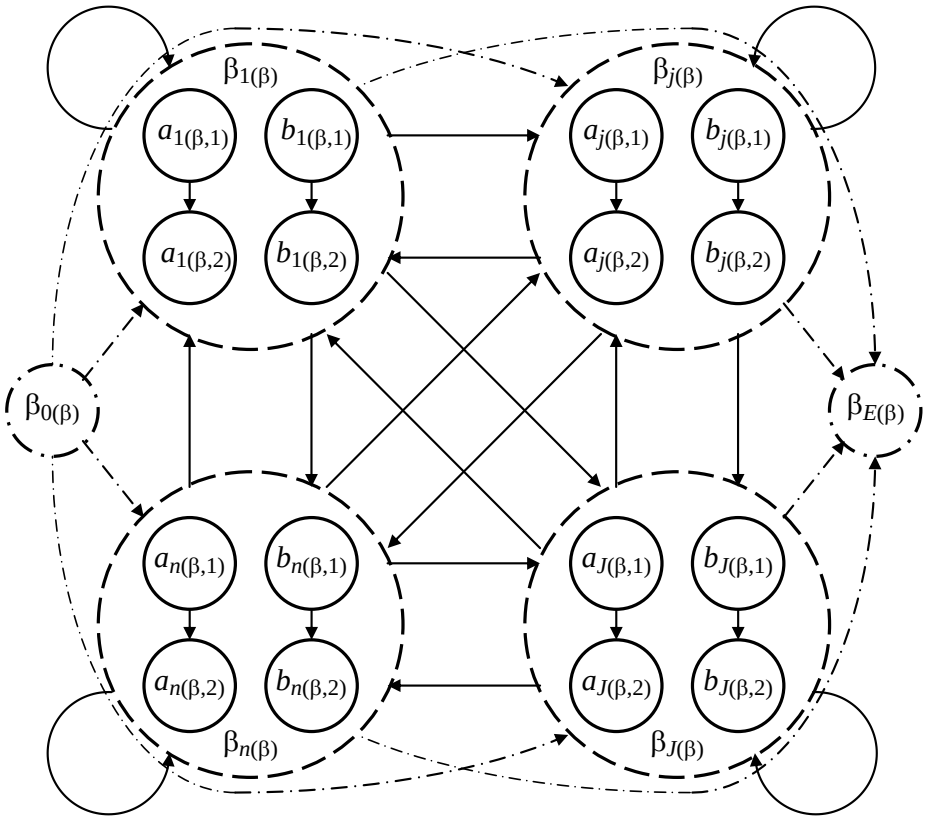


Рис. 3.5. Модель полумарковского генератора команд

Модель полумарковского генератора команд включает в себя $J(\beta)$2-параллельных полумарковских процессов (см. раздел 2), каждый из которых включает состояния $\{a_{j(\beta,1)}, a_{j(\beta,2)}, b_{j(\beta,1)}, b_{j(\beta,2)}\}$, $1(\beta) \le j(\beta) \le J(\beta)$, и описывается матрицей смежности

$$ {}^{2}\boldsymbol{r}_{j(\beta)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.59} $$

и полумарковской матрицей

$$ {}^{2}\boldsymbol{h}_{j(\beta)}(t) = \begin{bmatrix} 0 & f_{a[j(\beta)]}(t) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & f_{b[j(\beta)]}(t) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.60} $$

где $f_{a[j(\beta)]}(t)$ - плотность распределения времени пребывания процесса в состоянии $a_{j(\beta,1)}$ до его переключения в состояние $a_{j(\beta,2)}$; $f_{b[j(\beta)]}(t)$ - плотность распределения времени пребывания процесса в состоянии $b_{j(\beta,1)}$ до его переключения в состояние $b_{j(\beta,2)}$.

После переключения элементарных процессов $f_{a[j(\beta)]}(t)$ и $f_{b[j(\beta)]}(t)$ в состояния $a_{j(\beta,2)}$ и $b_{j(\beta,2)}$, происходит выход полумарковского генератора команд из состояния $\beta_{j(\beta)}$ и переключение его в сопряженное состояние $\beta_{n(\beta)}$, $1(\beta) \le j(\beta), n(\beta) \le J(\beta)$. В принципе, в полумарковском генераторе команд предусмотрены начальное $\beta_{0(\beta)}$ и поглощающее $\beta_{E(\beta)}$ состояния (показаны штрихпунктирной линией), но в установившемся режиме старт процесса не рассматривается, а переключение в состояние $\beta_{E(\beta)}$ считается маловероятным событием.

Таким образом, плотность распределения времени выхода из $j(\beta)$-го простейшего 2-параллельного полумарковского процесса определяется по зависимости

$$ f_{j(\beta)}(t) = \frac{d}{dt}\left[F_{a[j(\beta)]}(t) \cdot F_{b[j(\beta)]}(t)\right] \tag{3.61} $$

В общей модели генератора команд плотность складывается из условных плотностей распределения времени пребывания в состоянии $\beta_{j(\beta)}$ с последующим переключением в состояния $\beta_{n(\beta)}$, $1(\beta) \le j(\beta) \le J(\beta)$:

$$f_{j(\beta)} = \sum_{n(\beta)=1(\beta)}^{J(\beta)} h_{j(\beta),n(\beta)}(t);\;\; h_{j(\beta),n(\beta)}(t) = p_{j(\beta),n(\beta)} \cdot f_{j(\beta),n(\beta)}(t), \quad (3.62)$$

где $p_{j(\beta),n(\beta)}$ - вероятность переключения из состояния $\beta_{j(\beta)}$ в состояние $\beta_{n(\beta)}$; $f_{j(\beta),n(\beta)}(t)$ - плотность распределения времени переключения из состояния $\beta_{j(\beta)}$ при условии переключения процесса в состояние $\beta_{n(\beta)}$;

$$\sum_{n(\beta)=1(\beta)}^{J(\beta)} p_{j(\beta),n(\beta)} = 1. \quad (3.63)$$

Вследствие того, что $\int_0^\infty f_{j(\beta)}(t)dt = 1$, элементы $h_{j(\beta),n(\beta)}(t)$ формируют полумарковский процесс

$$\mathrm{B} = \{\beta_{1(\beta)}, ..., \beta_{j(\beta)}, ..., \beta_{J(\beta)}\},$$

$$\boldsymbol{r}_\beta = \left[r_{j(\beta),n(\beta)}\right];$$

$$\boldsymbol{h}_\beta(t) = \left[h_{j(\beta),n(\beta)}(t)\right]; \quad (3.64)$$

$$\boldsymbol{p}_\beta = \int_0^\infty \left[h_{j(\beta),n(\beta)}(t)\right]dt = \left[p_{j(\beta),n(\beta)}\right];$$

$$\boldsymbol{f}_\beta(t) = \left[\frac{h_{j(\beta),n(\beta)}(t)}{p_{j(\beta),n(\beta)}}\right] = \left[f_{j(\beta),n(\beta)}(t)\right];$$

$$\boldsymbol{T}_\beta = \int_0^\infty t\left[f_{j(\beta),n(\beta)}(t)\right]dt = \left[T_{j(\beta),n(\beta)}\right];$$

$$\boldsymbol{D}_\beta = \int_0^\infty \left[\left[t - T_{j(\beta),n(\beta)}\right]^2 f_{j(\beta),n(\beta)}(t)\right]dt = \left[D_{j(\beta),n(\beta)}\right];$$

где B - множество состояний; $\boldsymbol{r}_\beta$ - матрица смежности; $\boldsymbol{h}_\beta(t)$ - полумарковская матрица; $\boldsymbol{p}_\beta$ - стохастическая матрица; $\boldsymbol{f}_\beta(t)$ - матрица плотностей распределения; $\boldsymbol{T}_\beta$ - матрица математических ожиданий; $\boldsymbol{D}_\beta$ - матрица дисперсий; $r_{j(\beta),n(\beta)}=1,\ 1(\beta) \le j(\beta), n(\beta) \le J(\beta)$.

В общем случае полумарковский процесс $\boldsymbol{h}_{\beta}(t)$ соответствует следующим свойствам:

любое из состояний $\beta_{n(\beta)}$ достижимо из состояния $\beta_{j(\beta)}$, $1(\beta) \le j(\beta), n(\beta) \le J(\beta)$;

полумарковский процесс $\boldsymbol{h}_{\beta}(t)$ является возвратным;

взвешенные плотности распределения времени пребывания в состояниях определены на положительном полубесконечном интервале, т.е. $\arg\left[h_{j(\beta),n(\beta)}(t)\right] > 0$;

полумарковский процесс $\boldsymbol{h}_{\beta}(t)$ является эргодическим.

В генераторе структура полумарковского процесса такова, что обеспечивает его эргодичность.

3.2.2. Генерация команд

Выделим в эргодическом полумарковском процессе $\boldsymbol{h}_{\beta}(t)$ состояния, переключение в которые инициирует генерацию команды управления гофроагрегатом. Без нарушения общности рассуждений можно считать, что выделенные (Selected) состояния полумарковского процесса имеют индексы с наименьшими значениями, т.е.

$$\mathrm{B} \supset \mathrm{B}_S = \left\{\beta_{1(S)}, ..., \beta_{j(S)}, ..., \beta_{J(S)}\right\},\ J(S) < J(\beta). \qquad (3.65)$$

Таковая индексация всегда может быть обеспечена, поскольку B представляет собой неупорядоченное множество, т.е. список. Матрицы $\boldsymbol{r}_{\beta}$ и $\boldsymbol{h}_{\beta}(t)$ при изменении индексов состояний могут быть получены из исходных путем соответствующих перестановок строк и столбцов.

Очевидно, что каждое переключение из состояния $\beta_{j(S)} \in \mathrm{B}_S$ в состояние $\beta_{n(S)} \in \mathrm{B}_S$ формирует поток команд на технологического процесса.

Для определения времени переключения полумарковского процесса из $\beta_{j(S)} \in \mathrm{B}_S$ в $\beta_{n(S)} \in \mathrm{B}_S$, в соответствии с методикой, приведенной в [28, 47, 93], разделим $j(S)$-е состояние подмножества B_S, на два: начальное $\beta_{j(S,b)}$, отмеченное индексом $j(S, e) = j(S)$, и поглощающее $\beta_{j(S,e)}$, отмеченное индексом $j(S, e) = J(\beta) + j(S)$. Таким образом, из исходного эргодического полумарковского процесса (1) формируется неэргодический полумарковский процесс

$$\boldsymbol{r}'_{\beta} = \lfloor r'_{j(\beta),n(\beta)} \rfloor,$$
$$\boldsymbol{h}'_{\beta}(t) = \lfloor h'_{j(\beta),n(\beta)}(t) \rfloor, \qquad (3.66)$$

в котором

$$\mathrm{B}' = \{\beta_{1(S)}, ..., \beta_{j(S)}, ..., \beta_{J(S)}, \beta_{J(S)+1}, ..., \beta_{j(\beta)}, ..., \beta_{J(\beta)}, \beta_{J(\beta)+1}, ..., \beta_{J(\beta)+J(S)}, ..., \beta_{J(\beta)+J(S)}\}; \qquad (3.67)$$

$$\boldsymbol{r}' = \begin{pmatrix} 0 & .. & 0 & r_{1(S),J(S)+1} & .. & r_{1(S),J(\beta)} & r_{1(S),J(\beta)+1} & .. & r_{1(S),J(\beta)+J(S)} \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & r_{J(S),J(S)+1} & .. & r_{J(S),J(\beta)} & r_{J(S),J(\beta)+1} & .. & r_{J(S),J(\beta)+J(S)} \\ 0 & .. & 0 & r_{J(S)+1,J(S)+1} & .. & r_{J(S)+1,J(\beta)} & r_{J(S)+1,J(\beta)+1} & .. & r_{J(S)+1,J(\beta)+J(S)} \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & r_{J(\beta),J(S)+1} & .. & r_{J(\beta),J(\beta)} & r_{J(\beta),J(\beta)+1} & .. & r_{J(\beta),J(\beta)+J(S)} \\ 0 & .. & 0 & 0 & .. & 0 & 0 & .. & 0 \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & 0 & .. & 0 & 0 & .. & 0 \end{pmatrix}; \qquad (3.68)$$

$$\boldsymbol{h}'(t) = \begin{pmatrix} 0 & .. & 0 & h_{1(S),J(S)+1} & .. & h_{1(S),J(\beta)} & h_{1(S),J(\beta)+1} & .. & h_{1(S),J(\beta)+J(S)} \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & h_{J(S),J(S)+1} & .. & h_{J(S),J(\beta)} & h_{J(S),J(\beta)+1} & .. & h_{J(S),J(\beta)+J(S)} \\ 0 & .. & 0 & h_{J(S)+1,J(S)+1} & .. & h_{J(S)+1,J(\beta)} & h_{J(S)+1,J(\beta)+1} & .. & h_{J(S)+1,J(\beta)+J(S)} \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & h_{J(\beta),J(S)+1} & .. & r_{hJ(\beta),J(\beta)} & r_{hJ(\beta),J(\beta)+1} & .. & h_{J(\beta),J(\beta)+J(S)} \\ 0 & .. & 0 & 0 & .. & 0 & 0 & .. & 0 \\ & .. & & & .. & & & .. & \\ 0 & .. & 0 & 0 & .. & 0 & 0 & .. & 0 \end{pmatrix}; \qquad (6.69)$$

Мощность множества B′ равна $|\mathrm{B}'| = J(\beta) + J(S)$. Матрицы $\boldsymbol{r}'$ и $\boldsymbol{h}'_{\beta}(t)$ имеют размеры $[J(\beta) + J(S)] \times [J(\beta) + J(S)]$.

Развитие полумарковского процесса (6) представляет собой блуждание по состояниям в соответствии с матрицей смежности $\boldsymbol{r}'_{\beta}$ (7). Каждая случайная последовательность переключений начинается в одном из начальных состояний подмножества $\{\beta_{1(S)}, ..., \beta_{j(S)}, ..., \beta_{Jj(S)}\} \subset \mathrm{B}'$ и оканчивается в одном из поглощающих состояний подмножества $\{\beta_{J(\beta)+1}, ..., \beta_{J(\beta)+n(S)}, ..., \beta_{J(\beta)+J(S)}\} \subset \mathrm{B}'$, $1(S) \le j(S), n(S) \le J(S)$

Взвешенная плотность распределения времени блуждания от момента попадания в состояние $\beta_{j(S)} \in \mathrm{B}_S$ до момента попадания в состояние $\beta_{n(S)} \in \mathrm{B}_S$ определяется по зависимости

$$h''_{j(S),n(S)}(t) = \sum_{k=1}^{\infty} \Im^{-1}\left[{}^{r}\boldsymbol{I}_{j(S)} \{\Im[\boldsymbol{h}'_{\beta}(t)]\}^{k} {}^{c}\boldsymbol{I}_{n(S)} \right], \qquad (3.70)$$

где ${}^{r}\boldsymbol{I}_{j(S)}$ - вектор-строка размером $J(\beta) + J(S)$, $j(S)$-й элемент которого равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{n(S)}$ - вектор-столбец размером $J(\beta) + J(S)$, $[J(\beta) + n(S)]$-й элемент которого равен единице, а остальные элементы равны нулю.

Операция (6.70) должна быть выполнена для всех $1(S) \le j(S), n(S) \le J(S)$.

В результате формируется полумарковский процесс, включающий только состояния, моделирующие генерацию команд на технологического процесса со множеством состояний

$$S = \{s_{1(S)}, ..., s_{j(S)}, ..., s_{J(S)}\}. \qquad (3.71)$$

Полумарковский процесс описывается следующей системой зависимостей:

$$\boldsymbol{r}_S = \lfloor r''_{j(S),n(S)} \rfloor;$$

$$\boldsymbol{h}_S(t) = \lfloor h''_{j(S),n(S)}(t) \rfloor; \qquad (3.72)$$

$$\boldsymbol{p}_S = \int_0^{\infty} \left[h''_{j(S),n(S)}(t) \right] dt = \left[p''_{j(S),n(S)} \right];$$

$$\boldsymbol{f}_S(t) = \left[\frac{h''_{j(S),n(S)}(t)}{p''_{j(S),n(S)}} \right] = \left[f''_{j(S),n(S)}(t) \right];$$

$$\boldsymbol{T}_S = \int_0^{\infty} t \left[f''_{j(S),n(S)}(t) \right] dt = \left[T''_{j(S),n(S)} \right];$$

$$\boldsymbol{D}_S = \int_0^{\infty} \left[\left[t - T''_{j(S),n(S)} \right]^2 f_{j(S),n(S)}(t) \right] dt = \left[D''_{j(S),n(S)} \right];$$

где S - множество состояний; $\boldsymbol{r}_S$ - матрица смежности; $\boldsymbol{h}_S(t)$ - полумарковская матрица; $\boldsymbol{p}_S$ - стохастическая матрица; $\boldsymbol{f}_S(t)$ - матрица плотностей распределения; $\boldsymbol{T}_S$ - матрица математических ожиданий; $\boldsymbol{D}_S$ - матрица дисперсий; $r_{j(S),n(S)}=1,\ 1(S) \le j(S),\ n(S) \le J(S).$

Справедливо следующее утверждение:

Утверждение 3.1. Если $\boldsymbol{h}_\beta(t)$ является эргодическим полумарковским процессом, то $\boldsymbol{h}_S(t)$ - также эргодический полумарковский процесс.

Действительно, в силу эргодичности процесса $\boldsymbol{h}_\beta(t)$ все состояния $\mathrm{B} \supset \mathrm{B}_S = \{\beta_{1(S)}, ..., \beta_{j(S)}, ..., \beta_{J(S)}\}$ являются возвратными. В силу того, что плотности распределения $h''_{j(s),n(s)}(t)$ определяются зависимостью (6.12), них не все функции определяются вырожденными законами нулевым математическим ожиданием. Таким образом, справедливость утверждения доказана.

В силу того, что в общем случае $r_{j(S),n(S)}=1,\ 1(S) \le j(S),\ n(S) \le J(S)$, структура состояний полумарковского процесса $\boldsymbol{h}_S(t)$ моделируется полным графом с петлями. Каждое его переключение из состояния $s_{j(S)}$ в сопряженное состояние $s_{n(S)}$, или по петле в то же самое состояние $s_{j(S)}$, моделирует генерацию одной команды (пока без разделения на типы команд).

В соответствии с результатами, изложенными в [187, 188], вероятности пре-

бывания процесса $\boldsymbol{h}_S(t)$ в состоянии $s_{j(S)}$, для внешнего, по отношению к генератору команд, наблюдателю, определяются по зависимости:

$$\pi_{j(S)} = \frac{T''_{j(S)}}{\tau_{j(S)}}, \tag{3.73}$$

где $T''_{j(S)}$ - время пребывания процесса $\boldsymbol{h}_S(t)$ в состоянии $s_{j(S)}$; $\tau_{j(S)}$ - время возврата процесса $\boldsymbol{h}_S(t)$ в состояние $s_{j(S)}$.

Время пребывания процесса $\boldsymbol{h}_S(t)$ в состоянии $s_{j(S)}$ определяется выражением:

$$T''_{j(S)} = \sum_{n(S)=1(S)}^{J(S)} T''_{j(S),n(S)} \cdot p''_{j(S),n(S)}. \tag{3.74}$$

Для определения времени возврата процесса $\boldsymbol{h}_S(t)$ в состояние разделим состояние $s_{j(S)}$ на стартовое состояние и поглощающее. В этом случае множество состояний примет вид

$$S' = \{s_{1(S)}, ..., s_{j(S)}, ..., s_{J(S)}, s_{J(S)+1}\}. \tag{3.75}$$

Мощность множества S', $|S'| = J(S)+1$

Примем, что в множестве (6.75) состояние $s_{j(S)}$ является стартовым, а состояние $s_{J(S)+1}$ - поглощающим. В этом случае матрица смежности и полумарковская матрица будут иметь размеры $J(S)+1 \times J(S)+1$ и примут вид:

$$\boldsymbol{h}'_S(t) = \begin{pmatrix} h''_{1(S),1(S)} & \cdots & h''_{1(S),j(S)-1} & 0 & h''_{1(S),j(S)+1} & \cdots & h''_{1(S),J(S)} & h''_{1(S),j(S)} \\ & & & & \cdots & & & \\ h''_{j(S)-1,1(S)} & \cdots & h''_{j(S)-1,j(S)-1} & 0 & h''_{j(S)-1,j(S)+1} & \cdots & h''_{j(S)-1,J(S)} & h''_{j(S)-1,j(S)} \\ h''_{j(S),1(S)} & \cdots & h''_{j(S),j(S)-1} & 0 & h''_{j(S),j(S)+1} & \cdots & h''_{j(S),J(S)} & h''_{j(S),j(S)} \\ h''_{j(S)+1,1(S)} & \cdots & h''_{j(S)+1,j(S)-1} & 0 & h''_{j(S)+1,j(S)+1} & \cdots & h''_{j(S)+1,J(S)} & h''_{j(S)-1,j(S)} \\ & & & & \cdots & & & \\ h''_{J(S),1(S)} & \cdots & h''_{J(S),j(S)-1} & 0 & h''_{j(S),j(S)+1} & \cdots & h''_{J(S),J(S)} & h''_{J(S),j(S)} \\ 0 & & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}. \tag{3.76}$$

Плотность распределения времени возврата в состояние $s_{j(S)}$ определяется

по зависимости

$$f_{j(S)}(t)=\sum_{k=1}^{\infty}\Im^{-1}\left[{}^{r}\boldsymbol{I}_{j(S)}\{\Im[\boldsymbol{h}'_{S}(t)]\}^{k}\,{}^{c}\boldsymbol{I}_{J(S)+1}\right], \qquad (3.77)$$

где ${}^{r}\boldsymbol{I}_{j(S)}$ - вектор-строка размером $J(S)+1$, $j(S)$-й элемент которого равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{J(S)+1}$ - вектор-столбец размером $J(S)+1$, $[J(S)+1]$-й элемент которого равен единице, а остальные элементы равны нулю.

Выражение (3.77) имеет физический смысл плотности распределения, а не взвешенной плотности распределения, поскольку в полумарковском процессе $\boldsymbol{h}'_{(S)}(t)$ имеется единственное поглощающее состояние, $s_{J(S)+1}$, и все траектории блуждания, начинаясь в $s_{j(S)}$, рано или поздно попадают в него.

Из (3.73), (3.74), (3.77) следует, что

$$\pi_{j(S)}=\frac{\sum\limits_{n(S)=1(S)}^{J(S)}T''_{j(S),n(S)}\cdot p''_{j(S),n(S)}}{\int\limits_{0}^{\infty}t\sum\limits_{k=1}^{\infty}\Im^{-1}\left[{}^{r}\boldsymbol{I}_{j(S)}\{\Im[\boldsymbol{h}'_{S}(t)]\}^{k}\,{}^{c}\boldsymbol{I}_{J(S)+1}\right]dt}. \qquad (3.78)$$

Зависимость (3.78) определяет вероятности пребывания полумарковского процесса в состояниях $s_{j(S)}$ $1(S)\le j(S)\le J(S)$ для внешнего, по отношению к генератору команд, наблюдателя.

Утверждение 3.4. Если известны вероятности $\pi_{j(S)}$ пребывания в состояниях $s_{j(S)}$, $1(S)\le j(S)\le J(S)$, генератора команд, то плотность распределения времени между двумя событиями генерации очередной команды определяется выражением

$$g(t)=\sum_{j(S)=1(S)}^{J(S)}\pi_{j(S)}\cdot\sum_{n(S)=1(S)}^{J(S)}h''_{j(S),n(S)}(t), \qquad (3.79)$$

где $h''_{j(S),n(S)}(t)$ - взвешенная плотность распределения времени блуждания от момента попадания в состояние $\beta_{j(S)}\in \mathrm{B}_S$ до момента попадания в состояние

$\beta_{n(S)} \in \mathrm{B}_S$ определяемое по зависимости (3.65).

Действительно, пусть в какой-то момент времени произошло событие переключения состояния полумарковского процесса (6.67) $\boldsymbol{h}_S(t)$. С вероятностью $\pi_{j(S)}$ переключение произошло в состояние $\beta_{j(S)} \in \mathrm{B}_S$. Следующее переключение процесса $\boldsymbol{h}_S(t)$ из состояния $\beta_{j(S)} \in \mathrm{B}_S$ произойдет через время, определяемое по зависимости типа $\sum_{n(S)=1(S)}^{J(S)} h''_{j(S),n(S)}(t)$, откуда и следует (3.79).

Выражение (3.79) определяет временные интервалы между командами в общем потоке команд, формируемом в результате интерактивного взаимодействия оператора и аппаратных средств, безотносительно к тому, как эти команды разделяются по типам команд, в соответствии с которыми технологического процесса переключается в различные состояния. В случае если поток команд, генерируемых полумарковским генератором $\boldsymbol{h}_S(t)$, является неоднородным, что показано на рис. 3.6, то поток команд с интервалами $g(t)$ формируется в результате объединения потоков по каждому типу команд.

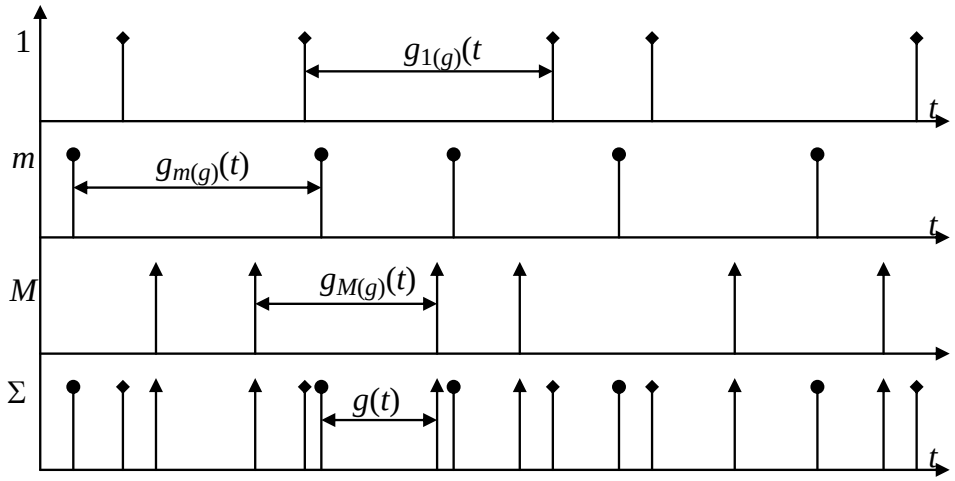


Рис. 3.6. Генерация потока команд с разделением по типу

Рассмотрим случай, когда генерируется $M(g)$ команд, и определим плотности распределения $g_{m(g)}(t)$, времени между событиями в каждом отдельном потоке генерации $m(g)$-й команды, $1 \le m(g) \le M(g)$. Для этого разделим множество состо-

яний генератора $S = \{s_{1(S)}, ..., s_{j(S)}, ..., s_{J(S)}\}$ на подмножества $S_{m(g)} = \{s_{1[S,m(g)]}, ..., s_{j[S,m(g)]}, ..., s_{J[S,m(g)]}\}$, $1 \le m(g) \le M(g)$, которые обладают следующими свойствами: $S_{m(g)} \cap S_{n(g)} = \varnothing$, если $m(g) \ne n(g)$, и $\bigcup_{m(g)=1(g)}^{M(g)} S_{m(g)} = S$, т.е. множества являются непересекающимися. Этот случай показан на рис. 3.7.

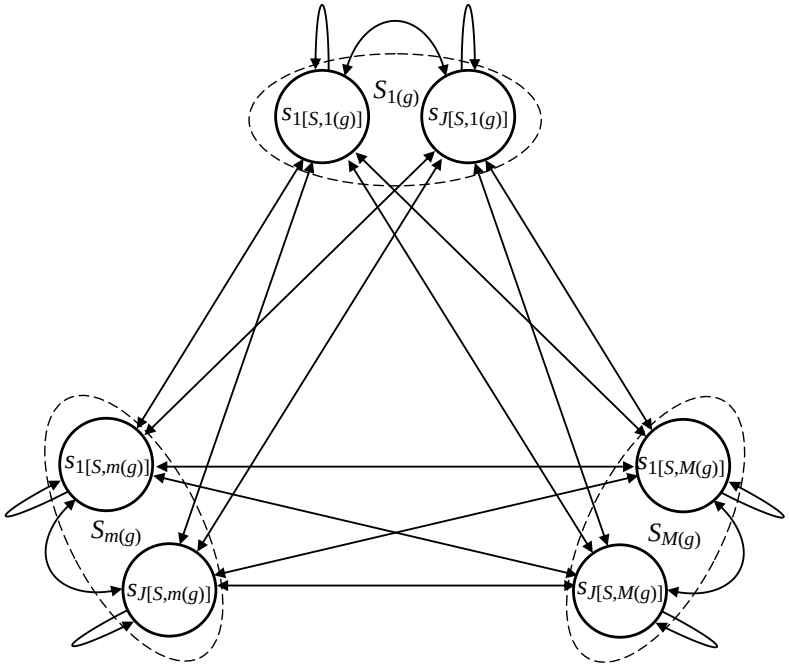


Рис. 3.7. Структура неоднородного генератора команд до преобразования

Без нарушения общности можно считать, что команда первого типа генерируется при переключении в состояния $s_{j(S)}$, у которых $1(S) \le j(S) \le J(S,1)$, ..., команда $m(g)$-го типа генерируется при переключении в состояния $s_{j(S)}$, у которых $\sum_{n(g)=1(g)}^{m(g)-1} J[S,n(g)] \le j(S) \le \sum_{n(g)=1(g)}^{m(g)} J[S,n(g)]$, ..., команда $M(g)$-го типа генерируется при переключении в состояния $s_{j(S)}$, у которых $\sum_{n(g)=1(g)}^{M(g)-1} J[S,n(g)] \le j(S) \le \sum_{n(g)=1(g)}^{M(g)} J[S,n(g)] = J(S)$. Таким образом, пересчет индексов состояний может быть проведен следующим образом;

$$j\left[S,m(g)\right]=j(S)-\sum_{n(g)=1}^{m(g)-1}J\left[S,n(g)\right], \tag{3.80}$$

где $j(S)$ - индекс, используемый для нумерации элементов множества *S*; $j\left[S,m(g)\right]$ - индекс для нумерации элементов подмножества $S_{m(g)}$.

Полумарковский процесс $\boldsymbol{h}_S(t)$ из структуры, определяемой множеством состояний *S* и матрицей смежности $\boldsymbol{r}_S$, приведенной на рис. 3.7, за счет объединения состояний, переключение в которые генерирует поток команд одного типа, может быть преобразован в процесс, структура которого приведена на рис. 3.8. Процесс имеет множество состояний

$$S_g=\left\{S_{1(g)},...,S_{m(g)},...,S_{M(g)}\right\}, \tag{3.81}$$

где $S_{m(g)}=\left\{s_{1\left[S,m(g)\right]},...,s_{j\left[S,m(g)\right]},...,s_{J\left[S,m(g)\right]}\right\}$, $1(g)\le m(g)\le M(g)$.

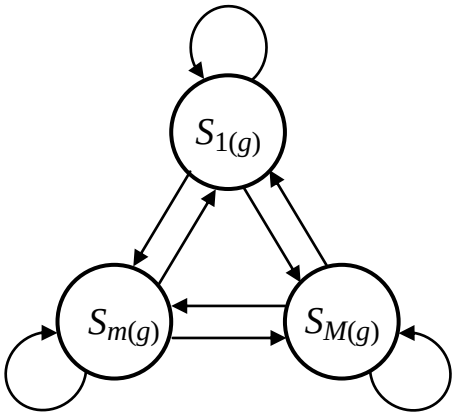


Рис. 3.8. Структура неоднородного генератора команд после преобразования

Процесс описывается матрицей смежности и полумарковской матрицей

$${}^{g}\boldsymbol{r}=\left[\,{}^{g}r_{m(\mathrm{g}),n(\mathrm{g})}\,\right];$$

$${}^{g}\boldsymbol{h}(t)=\left[\,{}^{g}h_{m(\mathrm{g}),n(\mathrm{g})}(t)\right]=\left[\,g_{m(\mathrm{g}),n(\mathrm{g})}(t)\cdot{}^{g}p_{m(\mathrm{g}),n(\mathrm{g})}\,\right],$$

где ${}^{g}r_{m(g),n(g)}=1$, $1\le m(g),\ n(g)\le M(g)$;

$$ {}^{g}h_{m(g),n(g)}(t) = \sum_{j(S)=\sum_{l(g)=1(g)}^{m(g)-1} J[S,l(g)]}^{\sum_{l(g)=1(g)}^{m(g)} J[S,l(g)]} \; \sum_{n(S)=\sum_{l(g)=1(g)}^{n(g)-1} J[S,l(g)]}^{\sum_{l(g)=1(g)}^{n(g)} J[S,l(g)]} h''_{j(S),n(S)}(t). \quad (6.82) $$

Вероятности пребывания полумарковского процесса ${}^{g}\boldsymbol{h}(t)$ в состояниях $S_{n(g)} \in S_g$ могут быть определены по вышеприведенной методике, основанной на разделении вершин $S_{n(g)}$ графа со структурой ${}^{g}\boldsymbol{r}$. Однако, с учетом того, что вероятности $\pi_{j(S)}$ пребывания в состояниях $s_{j(S)}$, для внешнего, по отношению к генератору команд, наблюдателя уже найдены и определяются по зависимости (6.80), а состояния $S_{n(g)}$ полумарковского процесса ${}^{g}\boldsymbol{h}(t)$ являются несовместными, они могут быть получены как следующие суммы вероятностей:

$$ \pi_{n(g)} = \sum_{j(S)=\sum_{l(g)=1(g)}^{n(g)-1} J[S,l(g)]}^{j(S)=\sum_{l(g)=1(g)}^{n(g)} J[S,l(g)]} \pi_{l(S)}. \quad (3.83) $$

Для внешнего наблюдателя взвешенные плотности распределения времени переключения полумарковского процесса ${}^{g}\boldsymbol{h}(t)$ со структурой ${}^{g}\boldsymbol{r}$ в состояния $S_{m(g)}$, порождающие $m(g)$-й поток команд равны

$$ {}^{g}h_{m(g)}(t) = \sum_{n(g)=1(g)}^{M(g)} {}^{g}\pi_{n(g)} \, {}^{g}h_{n(g),m(g)}(t). \quad (3.84) $$

Таким образом, окончательно полумарковский процесс генератора команд может быть представлен в виде модели, структура которой имеет вид, приведенный на рис. 3.9.

Приведенная структура имеет множество состояний

$$ \sigma_g = \{\sigma_{0(g)}, ..., \sigma_{1(g)}, ..., \sigma_{m(g)}, ..., \sigma_{M(g)}\} \quad (3.85) $$

и описывается матрицей смежности

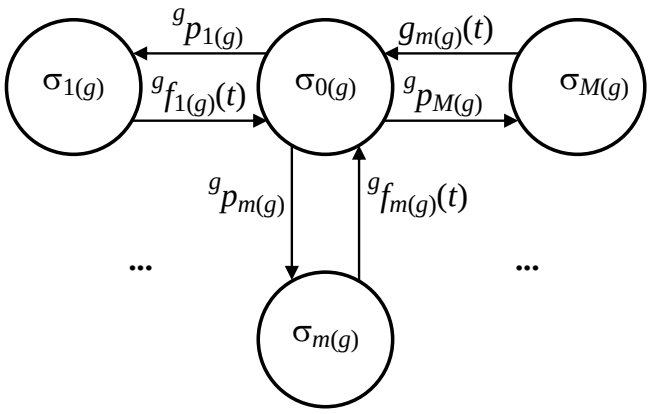


Рис. 3.9. Итоговая модель генератора команд

$$\sigma_{\boldsymbol{r}} = \begin{bmatrix} 0 & 1 & \dots & 1 & \dots & 1 \\ 1 & 0 & \dots & 0 & \dots & 0 \\ \dots & & & \dots & & \\ 1 & 0 & \dots & 0 & \dots & 0 \\ \dots & & & \dots & & \\ 1 & 0 & \dots & 0 & \dots & 0 \end{bmatrix}. \qquad (3.86)$$

Полумарковский процесс имеет вид

$$\sigma_{\boldsymbol{h}}(t) = \begin{bmatrix} 0 & \delta(t)\cdot{}^{g}p_{1(g)} & \dots & \delta(t)\cdot{}^{g}p_{m(g)} & \dots & \delta(t)\cdot{}^{g}p_{M(g)} \\ {}^{g}f_{1(g)}(t) & 0 & \cdots & 0 & \dots & 0 \\ \dots & & & \dots & & \\ {}^{g}f_{m(g)}(t) & 0 & \dots & 0 & \dots & 0 \\ \dots & & & \dots & & \\ {}^{g}f_{M(g)}(t) & 0 & \dots & 0 & \dots & 0 \end{bmatrix}, \qquad (3.87)$$

где

$${}^{g}p_{m(g)} = \int_{0}^{\infty} {}^{g}h_{m(g)}(t)dt\,; \qquad (3.88)$$

$${}^{g}f_{m(g)}(t) = \frac{{}^{g}h_{m(g)}(t)}{\int_{0}^{\infty} {}^{g}h_{m(g)}(t)dt}. \qquad (3.89)$$

Для ${}^{g}f_{m(g)}(t)$ могут быть получены математическое ожидание и дисперсия:

$$ {}^{g}T_{m(g)} = \int_{0}^{\infty} \frac{{}^{g}h_{m(g)}(t)}{\int_{0}^{\infty} {}^{g}h_{m(g)}(t)dt} t dt; \qquad (3.90) $$

$$ {}^{g}D_{m(g)} = \int_{0}^{\infty} \frac{{}^{g}h_{m(g)}(t)}{\int_{0}^{\infty} {}^{g}h_{m(g)}(t)dt} \left(t - {}^{g}T_{m(g)}\right)^2 dt. \qquad (3.91) $$

Отметим, что в полумарковском процессе (3.87) речь идет о стохастическом суммировании потоков, в отличие от приведенного на рис. (3.7) прямого суммирования. При этом, если вероятности, определяемые зависимостью (3.88), являются априорными вероятностями появления команды $m(g)$-го типа, то вероятности

$$ {}^{g}p_{m(g),n(g)} = \int_{0}^{\infty} \sum_{j(S)=\sum_{l(g)=1(g)}^{m(g)-1} J[S,l(g)]}^{\sum_{l(g)=1(g)}^{m(g)} J[S,l(g)]} \sum_{n(S)=\sum_{l(g)=1(g)}^{n(g)-1} J[S,l(g)]}^{\sum_{l(g)=1(g)}^{n(g)} J[S,l(g)]} h''_{j(S),n(S)}(t)dt. \qquad (3.92) $$

дают апостериорные вероятности появления команды $n(g)$-го типа, при условии, что предыдущая команда была $m(g)$-го типа. Соответственно и плотность распределения $g(t)$, определенная зависимостью (3.79), представляет собой априорную оценку интервала между командами вообще, а плотности распределения времени между командами, определенные зависимостями (3.89) характеризуют интервал при условии, что с соответствующей вероятностью был выбран $m(g)$-й тип команды.

3.3. Моделирование исполнения команд

Одним из возможных способов исполнения команд, поступающих из центрального пункта управления, является введение режима прерываний. Указанный режим характеризуется тем, что при поступлении команды с пункта управления прерывается реализация текущей циклограммы управления технологическим процессом, и циклограмма перезапускается с другого места. При оценке времен-

ных и вероятностных характеристик выполнения команды через прерывания будем считать, что команды генерируются полумарковским генератором, а циклограмма моделируется полумарковским процессом, причем оба процесса являются независимыми друг от друга. Существование в системе прерываний представляет собой характерный пример параллельного полумарковского процесса.

Рассмотрим модели диспетчеризации:

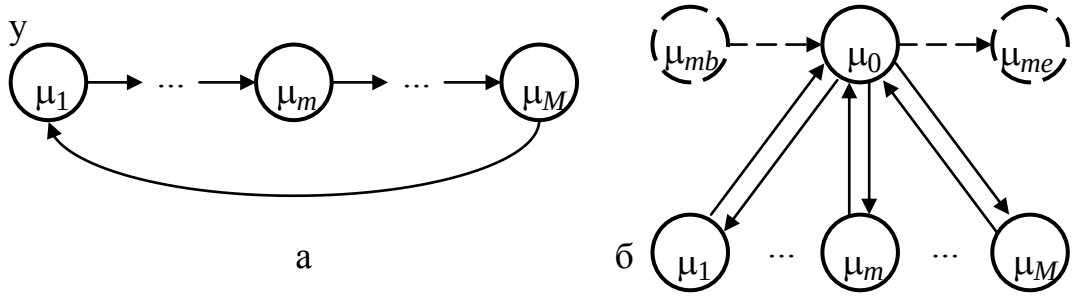


Рис.10. Модели циклической (а) и квазистохастической (б) дисциплин диспетчеризации

3.3.1. Модель циклической дисциплины диспетчеризации

Структура модели циклической дисциплины диспетчеризации приведена на рис. 3.11.

Модель формируется из полумарковских процессов $\boldsymbol{h}'_l(t)$, $1 \le l \le L$, которые имеют вид (3.90):

$$\boldsymbol{h}'_l(t) = [h'_{j(a,l),n(a,l)}(t)], \tag{3.90}$$

у которого

$$A'_l = \{a_{0(a,l)}, a_{1(a,l)}, ..., a_{j(a,l)}, ..., a_{J(a,l)}\}; \tag{3.91}$$

$$A'_l = B_l \cup \bar{E}_l \cup E_l, \tag{3.92}$$

где $B_l = \{a_{0(a,l)}\}$; $\bar{E}_l = \{a_{1(a,l)}, ..., a_{j(a,l)}, ..., a_{J(a,l)-J(e,l)}\}$; $E_l = \{a_{J(a,l)-J(e,l)+1}, ..., a_{j(e,l)}, ..., a_{J(a,l)}\}$.

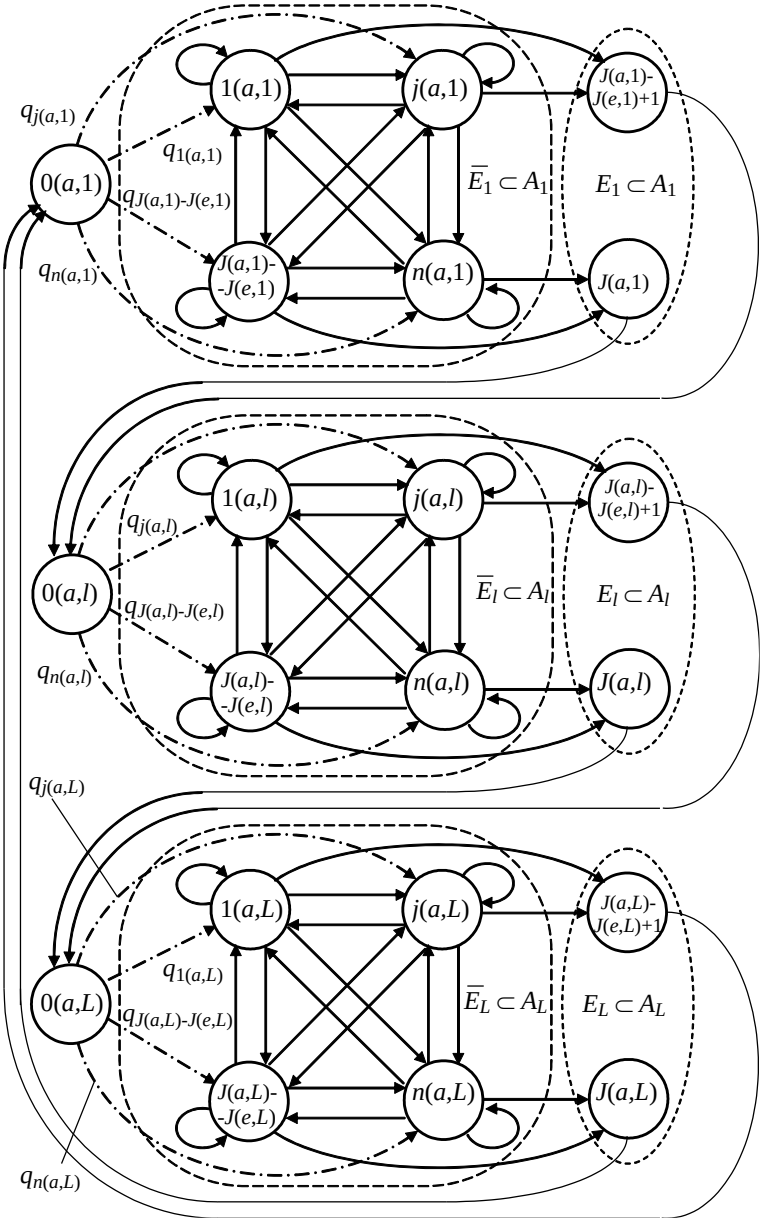


Рис. 3.11 Модель циклической диспетчеризации

–

$$\boldsymbol{r}'_l(t) = \left[r'_{j(a,l),n(a,l)}\right] = \begin{pmatrix} 0 & r_{0(a,l),1(a,l)} & \cdots & r_{0(a,l),n(a,l)} & \cdots & r_{0(a,l),J(a,l)} \\ 0 & r_{1(a,l),1(a,l)} & \cdots & r_{1(a,l),n(a,l)} & \cdots & r_{1(a,l),J(a,l)} \\ & & & \cdots & & \\ 0 & r_{j(a,l),1(a,l)} & \cdots & r_{j(a,l),n(a,l)} & \cdots & r_{j(a,l),J(a,l)} \\ & & & \cdots & & \\ 0 & r_{J(a,l),1(a,l)} & \cdots & r_{J(a,l),n(a,l)} & \cdots & r_{J(a,l),J(a,l)} \end{pmatrix}; \quad (3.93)$$

$$r_{0(a,l),n(a,l)} = \begin{cases} 0, \text{ если } q_{n(a,l)} = 0; \\ 1, \text{ если } q_{n(a,l)} \neq 0; \end{cases}$$

$$\mathbf{h}'_l(t)=\begin{pmatrix} 0 & q_{1(a,l)}\delta(t) & \dots & q_{n(a,l)}\delta(t) & \dots & q_{J(a,l)}\delta(t) \\ 0 & h_{1(a,l),1(a,l)}(t) & \dots & h_{1(a,l),n(a,l)}(t) & \dots & h_{1(a,l),J(a,l)}(t) \\ & & & \dots & & \\ 0 & h_{j(a,l),1(a,l)}(t) & \dots & h_{j(a,l),n(a,l)}(t) & \dots & h_{j(a,l),J(a,l)}(t) \\ & & & \dots & & \\ 0 & h_{J(a,l),1(a,l)}(t) & \dots & h_{J(a,l),n(a,l)}(t) & \dots & h_{J(a,l),J(a,l)}(t) \end{pmatrix}. \quad (3.94)$$

Сформированный из полумарковских процессов (3.95) обобщенный циклический полумарковский процесс имеет вид:

$$\mathbf{h}_c(t)=\begin{bmatrix} \mathbf{0} & \mathbf{h}_{c,1}(t) & \mathbf{q}_{c,1}(t) & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ & & & \dots & & & \dots & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{h}_{c,l}(t) & \mathbf{q}_{c,l}(t) & \dots & \mathbf{0} \\ & & & \dots & & & & \\ \mathbf{q}_{c,L}(t) & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{h}_{c,L}(t) \end{bmatrix} =$$

$$=\lfloor h_{j(c),n(c)} \rfloor, \quad (3.95)$$

$$\boldsymbol{r}_c(t)=\begin{bmatrix} \boldsymbol{0} & \boldsymbol{r}_{c,1} & {}_r\boldsymbol{q}_{c,1} & \dots & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ & & & \dots & & & \dots & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{r}_{c,l}(t) & {}_r\boldsymbol{q}_{c,l} & \dots & \boldsymbol{0} \\ & & & \dots & & & & \\ {}_r\boldsymbol{q}_{c,L} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{r}_{c,L} \end{bmatrix} = \lfloor r_{j(c),n(c)} \rfloor, \quad (3.96)$$

где $\boldsymbol{0}$ - матрица или вектор, элементы которого равны нулю; $\boldsymbol{h}_{c,1}(t)$ - матрица $\boldsymbol{h}'_1(t)$ без первого столбца, размером (строки×столбцы) $[J(a,1) + 1] \times J(a,1)$; $\boldsymbol{h}'_{c,l}(t)=\boldsymbol{h}'_l(t)$; $\boldsymbol{q}_{c,l}(t)=\lfloor q_{j(c,l)}(t) \rfloor$, $1 \le l \le L$ - 1 - вектор-столбец размером $[J(a,l) + 1] \times 1$, элементы которого равны:

$$\mathbf{q}_{j[c,l]}(t)=\begin{cases} 0, \text{ если } 0(a,l) \le j[c] \le \sum_{m=1}^{l-1} J(a,m) + j(e,l); \\ \delta(t), \text{ если } \sum_{m=1}^{l-1} j(a,m) + j(e,l) \le j[c] \le \sum_{m=1}^{l} J(a,m), \end{cases} 1 \le l \le L;$$

$\boldsymbol{r}_{c,1}(t)$ - матрица $\boldsymbol{r}'_1(t)$ без первого столбца, размером $[J(a,1) + 1] \times J(a,1)$; $\boldsymbol{r}'_{c,l}(t)=\boldsymbol{r}'_l(t)$; ${}_r\boldsymbol{q}_{c,l}(t)=\lfloor {}_r q_{j(c,l)}(t) \rfloor$, $1 \le l \le L$ - 1 - вектор-столбец размером $[J(a,l) + 1] \times 1$, элементы которого равны:

$$ {}_r\mathbf{q}_{j[c,l]}(t)=\begin{cases}0, \text{если } 0(a,l)\le j[c]\le \sum\limits_{m=1}^{l-1}J(a,m)+j(e,l);\\ 1, \text{если } \sum\limits_{m=1}^{l-1}j(a,m)+j(e,l)\le j[c]\le \sum\limits_{m=1}^{l}J(a,m),\end{cases} \quad 1\le l\le L. $$

Докажем следующие свойства циклической модели.

Утверждение 3.5. Обобщенный полумарковский процесс $\boldsymbol{h}_c(t)$, реализованный в циклической модели, является возвратным.

Действительно, в каждом из l полумарковских процессов $\boldsymbol{h}'_l(t)$, $1 \le l \le L$, любое из $1(a, l) \le j(a, l) \le J(a, l)$ состояний достижимо из состояния $a_{0(l,a)}$, в том числе и состояния подмножества $E_l = \{a_{J(a,l)\text{-}J(e,l)+1}, ..., a_{j(e,l)}, ..., a_{J(a,l)}\}$. В свою очередь, в силу особенностей матрицы смежности (5.7) из состояний $E_l = \{a_{J(a,l)\text{-}J(e,l)+1}, ..., a_{j(e,l)}, ..., a_{J(a,l)}\}$ достижимо состояние $a_{0(a,l+1)}$ для всех $1 \le l \le L$ - 1 а из состояний $\{a_{J(a,L)\text{-}J(e,l)+1}, ..., a_{j(e,\,L)}, ..., a_{J(a,\,L)}\}$ достижимо состояние $a_{0(1,a)}$. Таким образом, $\grave{a}_{j(a,l)} \to \grave{a}_{j(a,l)}$ для всех $0(a, l) \le j(a, l) \le J(a, l)$ и $1 \le l \le L$. Если все состояния полумарковского процесса $\boldsymbol{h}_c(t)$ являются возвратными, то указанный полумарковский процесс является возвратным, что и доказывает свойство.

Утверждение 3.6. Полумарковский процесс $\boldsymbol{h}_c(t)$, реализованный в циклической модели, является эргодическим.

Действительно, в соответствии с утверждением 3.5 полумарковский процесс $\boldsymbol{h}_c(t)$ является возвратным. В соответствии со свойствами (3.95) и (3.96), среди состояний процесса $\boldsymbol{h}_c(t)$ имеется хотя бы одно с ненулевым временем пребывания. Таким образом, полумарковский процесс $\boldsymbol{h}_c(t)$ является эргодическим [].

Для эргодического полумарковского процесса могут быть определены временные интервалы времени возврата в каждый элементарный процесс и вероятности пребывания в элементарных процессах для внешнего наблюдателя.

Для этого определим время достижения процессом $\boldsymbol{h}'_l(t)$ одного из состояний подмножества E из состояния $a_{0(a)}$. Вследствие того, что в процессе $\boldsymbol{h}'_l(t)$ отсутствуют состояния, недостижимые из $a_{0(a)}$ и состояния, из которых недостижи-

мы состояния подмножества E, все возможные траектории блуждания из $a_{0(a)}$ в E составляют полную группу несовместных событий, поэтому оценка времени достижения будет иметь характер плотности распределения, а не взвешенной плотности распределения. Тогда плотность распределения времени достижения E из $a_{0(a)}$ имеет вид:

$$f_{0(l),E(l)}(t) = \Im^{-1}\left({}^{r}\boldsymbol{I}_{0(l)} \left\{ \sum_{k=1}^{K} \Im\left[\boldsymbol{h}'_{l}(t) \right] \right\}^{k} {}^{c}\boldsymbol{I}_{E(l)} \right), \tag{3.97}$$

где ${}^{r}\boldsymbol{I}_{0(l)}$ - вектор-строка размером $J(a,l)+1$, $0(l)$-й элемент которой равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{E(l)}$ - вектор столбец размером $J(a,l)+1$, элементы с $[J(e) + 1]$-го по $J(a)$-й равны единице, а все остальные элементы равны нулю.

Для (3.97) могут быть найдены числовые характеристики, математическое ожидание и дисперсия:

$$T_{0(l),E(l)} = \int_{0}^{\infty} t f_{0(l),E(l)}(t)\, dt; \tag{3.98}$$

$$D_{0(l),E(l)} = \int_{0}^{\infty} \left[t - T_{0(l),E(l)} \right]^{2} f_{0(l),E(l)}(t)\, dt. \tag{3.99}$$

Плотность распределения и математическое ожидание времени возврата в начало цикла определяется в виде

$$f_{c}(t) = \Im^{-1}\left\{ \prod_{l=1}^{L} \Im\left[f_{0(l),E(l)}(t) \right] \right\}; \tag{3.100}$$

$$T_{c} = \sum_{l=1}^{L} T_{0(l),E(l)}. \tag{3.101}$$

Для внешнего наблюдателя вероятность того, что в текущий момент времени роботизированная платформа находится в l-м состоянии равна

$$\tilde{p}_{l} = \frac{T_{0(l),E(l)}}{T_{c}}. \tag{3.102}$$

3.3.2. Модели квазистохастических дисциплины диспетчеризации

Структура моделей квазистохастических дисциплин диспетчеризации приведена на рис. 3.12. С учетом упрощающего результата (3.97), модель может быть преобразована к виду, приведенному на рис. 3.13.

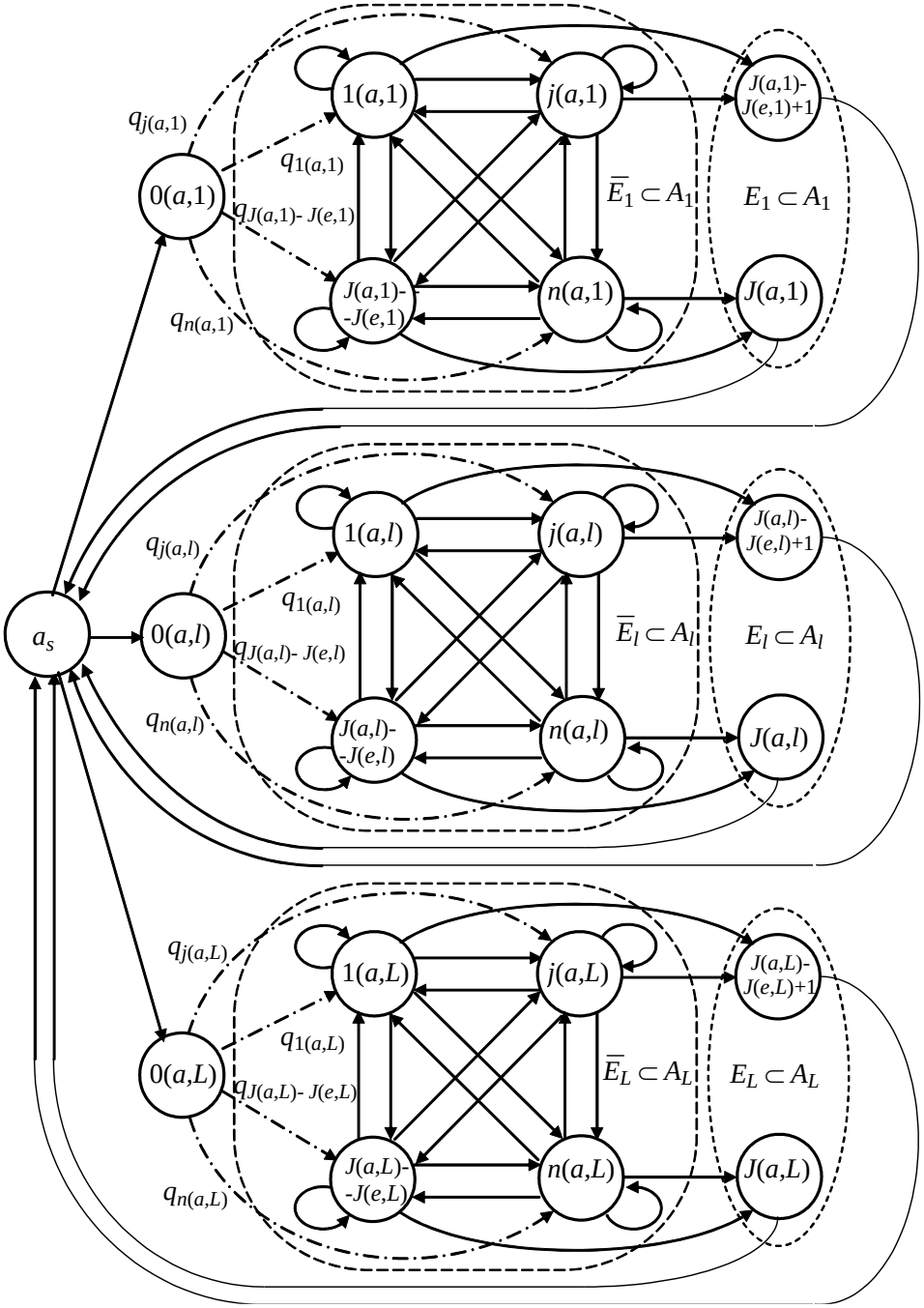


Рис. 3.12. Модель квазистохастической диспетчеризации

Модель квазистохастической дисциплины формируется из моделей (1) и состояния, которое моделирует работу диспетчера. Это состояние имеет номер $l=0$. Полумарковский процесс имеет вид

$$\boldsymbol{h}_S(t)=\begin{bmatrix} 0 & p_{d1}\cdot f_{d1}(t) & \dots & p_{dl}\cdot f_{dl}(t) & \dots & p_{dL}\cdot f_{dL}(t) \\ f_{0(1),E(1)}(t) & 0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ f_{0(l),E(l)}(t) & 0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ f_{0(L),E(L)}(t) & 0 & \dots & 0 & \dots & 0 \end{bmatrix}, \quad (3.103)$$

$$\boldsymbol{r}_S(t)=\begin{bmatrix} 0\,1 & \dots & 1 & \dots & 1 \\ 1\,0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1\,0 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1\,0 & \dots & 0 & \dots & 0 \end{bmatrix}, \quad (3.104)$$

где $f_{dl}(t)$ - плотность распределения времени работы программы-диспетчера, если известно, что далее будет произведено переключение в состояние l; p_{dl} - вероятность выбора состояния l для продолжения управления роботизированной платформой; $\sum_{l=1}^{L} p_{dl}=1$.

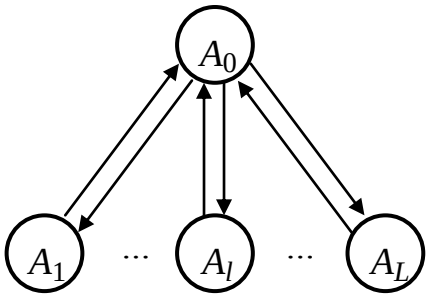


Рис. 3.13. Упрощенная модель квазистохастической диспетчеризации

Утверждение 3.6. Обобщенный полумарковский процесс $\boldsymbol{h}_S(t)$, реализованный в квазистохастической модели, является возвратным.

Действительно, В соответствии с матрицей (3.103), из состояния 0 процесс может переключиться в каждое из l состояний, $1 \le l \le L$. С другой стороны, из

каждого из l состояний можно переключиться в нулевое состояние. Таким образом, $0 \to l$ и $l \to 0$ для всех $1 \le l \le L$, что и доказывает свойство.

Утверждение 3.7. Обобщенный полумарковский процесс $\boldsymbol{h}_S(t)$, реализованный в квазистохастической модели, является эргодическим.

Действительно, в соответствии со свойством 5.1 полумарковский процесс $\boldsymbol{h}_S(t)$ является возвратным. В соответствии со свойствами (5.5) и (5.6), среди состояний процесса $\boldsymbol{h}_S(t)$ имеется хотя бы одно с ненулевым временем пребывания. Таким образом, полумарковский процесс $\boldsymbol{h}_S(t)$ является эргодическим [16,24,107].

Плотность распределения и среднее время возврата в программу диспетчеризации определяются по зависимостям:

$$f_0(t) = \Im^{-1}\left\{\sum_{l=1}^{L} p_l \cdot \Im\left[f_{0(l),E(l)}(t)\right] \cdot \Im\left[f_{dl}(t)\right]\right\}; \qquad (3.105)$$

$$T_0 = \sum_{l=1}^{L} p_l \cdot \left[T_{0(l),E(l)} + T_{dl}\right], \qquad (3.106)$$

где $T_{dl} = \int_0^\infty t \cdot f_{dl}(t)dt$ - математическое ожидание времени работы программы диспетчеризации.

Для оценки плотности распределения и математического ожидания времени возврата полумарковского процесса $\boldsymbol{h}_S(t)$ разделим состояние l на два состояния, стартовое и поглощающее (рис. 3.14).

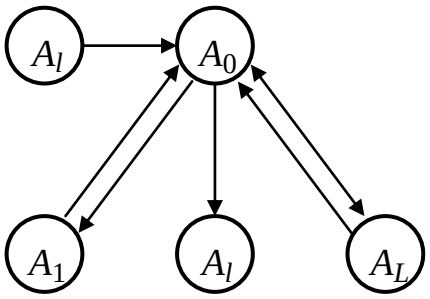


Рис. 3.14. Квазистохастическая структура

с расщепленным l-м состоянием

Для расщепления матрица (3.103) должна быть преобразована следующим образом:

– в матрицу должны быть добавлены крайний правый столбец и крайняя нижняя строка, которые описывают поглощающее состояние, сформированное из состояния l в результате расщепления, в этом случае l-я строка и l-й столбец будут описывать стартовое состояние;

– элемент $p_{dl} \cdot f_{dl}(t)$ полумарковской матрицы (3.103) должен быть перенесен в крайний правый столбец, а все оставшиеся элементы этого столбца должны быть приравнены к нулю;

– элементы l-го столбца должны быть приравнены к нулю;

– элементы крайней нижней строки также должны быть приравнены к нулю.

Полумарковский процесс приобретает следующий вид:

$$\mathbf{h}'_s(t) = \begin{bmatrix} 0 & p_{d1} \cdot f_{d1}(t) & \dots & 0 & \dots & p_{dL} \cdot f_{dL}(t) & p_{dl} \cdot f_{dl}(t) \\ f_{0(1),E(1)}(t) & 0 & & 0 & & 0 & 0 \\ & & \dots & \dots & \dots & & \dots \\ f_{0(l),E(l)}(t) & 0 & & 0 & & 0 & 0 \\ & & \dots & \dots & \dots & & \dots \\ f_{0(L),E(L)}(t) & 0 & & 0 & & 0 & 0 \\ 0 & 0 & \dots & 0 & \dots & 0 & 0 \end{bmatrix}; \quad (3.107)$$

$$\boldsymbol{r}'_s(t) = \begin{bmatrix} 0\,1 \dots 0 \dots 1\,1 \\ 1\,0 \dots 0 \dots 0\,0 \\ \\ 1\,0 \dots 0 \dots 0\,0 \\ \\ 1\,0 \dots 0 \dots 0\,0 \\ 0\,0 \dots 0 \dots 0\,0 \end{bmatrix}. \quad (3.108)$$

Плотность распределения времени возврата в состояние l определяется следующим выражением:

$$f_l(t) = \Im^{-1}\left({}^{r}\boldsymbol{I}_l \left\{ \sum_{k=1}^{\infty} \Im[\boldsymbol{h}'_s(t)] \right\}^{k} {}^{c}\boldsymbol{I}_{L+1} \right), \quad (3.109)$$

где ${}^{r}\boldsymbol{I}_{l}$ - вектор строка размером $L + 2$, l-й элемент которого равен единице, а остальные элементы равны нулю; ${}^{c}\boldsymbol{I}_{L+1}$ - вектор-столбец размером $L + 2$, последний элемент которого равен единице, а остальные элементы равны нулю.

Математическое ожидание времени возврата в процесс $\boldsymbol{h}_{l}(t)$ будет определяться как

$$T_{l} = T_{dl} + T_{0(l),E(l)} + \frac{\sum\limits_{\substack{n=1,\\ n\neq l}}^{N} p_{dn} \cdot \left(T_{0(l),E(l)} + T_{dn}\right)}{p_{dl}}; \tag{3.110}$$

Для внешнего наблюдателя вероятность того, что в текущий момент времени встроенная цифровая система управления находится в состоянии работы диспетчера, определяется по зависимости

$$\tilde{p}_{0} = \frac{\sum\limits_{l=1}^{L} p_{dl} \cdot T_{dl}}{T_{0}}; \tag{3.111}$$

Вероятность того, что системой осуществляется контроль над l-м блоком, равна:

$$\tilde{p}_{l} = \frac{T_{0(l),E(l)}}{T_{l}}. \tag{3.112}$$

Одной из важных задач, которую решает операционная система, является проблема, связанная с определением, когда и каким процессам следует выделять ресурсы процессора — задача эффективного использования процессора. Составленные модели циклической и приоритетной дисциплин диспетчеризации позволяют решить эту задачу для программного обеспечения роботизированных платформ.

3.4. Выводы

1) Сформулировано понятие 2-параллельного полумарковского процесса, получены зависимости для вероятности переключения каждого из элементарных полумарковских процессов первым и последним.

2) Сформулировано понятие *M*-параллельного полумарковского процесса, получены выражения для вероятностей переключения каждого из элементарных полумарковских процессов первым и последним.

3) Построен сложный *M*-параллельный полумарковский процесс, как процесс, имеющий больше двух состояний, среди которых можно выделить как минимум одно поглощающее и более одного непоглощающего состояния.

4) Показана сводимость сложного *M*-параллельного полумарковского процесса к ординарному полумарковскому процессу с функциональными состояниями, что позволяет применить к исследованию стохастических и временных характеристик методы, разработанные для ординарных полумарковских процессов.

5) Разработана концепция и сформирована модель генератора команд на гофроагрегат, характерной особенностью которой является интерактивный режим взаимодействия оператора и технических средств, что позволяет использовать при моделировании результаты, полученные при моделировании 2-параллельных процессов.

6) Решена задача расчета параметров потока команд на гофроагрегат, генерируемых в диалоговом режиме, предложено аппроксимировать интервал времени между командами в потоке экспоненциальным законом распределения, что позволяет существенно упростить выкладки по расчету состояний агрегата при управлении от генератора команд.

7) Получены зависимости для определения состояний технологического процесса при циклической и квазистохастической дисциплинах диспетчеризации.

4. ОЦЕНКА ХАРАКТЕРИСТИК ПРОГРАММ УПРАВЛЕНИЯ ГОФРОАГРЕГАТОМ

4.0. Введение

Особенностью сложных агрегатов является их многофункциональность. При централизованном управлении различными функциями возникает задача построения общей модели состояний. Как уже отмечалось, в современных роботизированных платформах циклограмма реализуется в виде программного кода бортовой ЭВМ. При управлении отдельными устройствами программный продукт, реализующий циклограмму, естественным образом разделяется на ряд программных модулей, каждый из которых осуществляет свою функцию в общей задаче управления. Наличие подобного разделения позволяет вести автономную промышленную разработку и отработку программных модулей, включаемых в пакет, на основе универсальных инструментальных средств.

Программные модули имеют унифицированную структуру, что позволяет строить диспетчерские программы опроса модулей. На рис. 4.1 показана общая структура программного обеспечения.

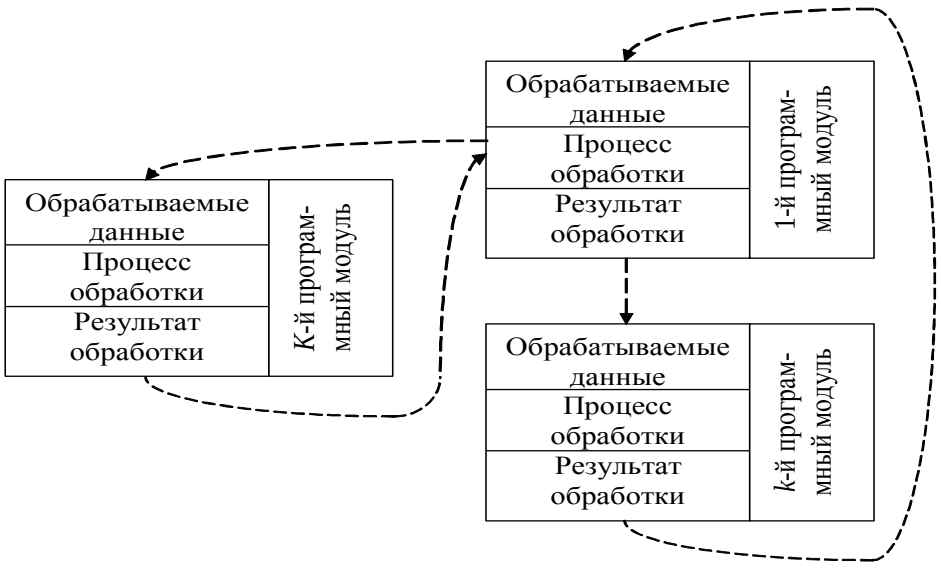


Рис. 4.1. Взаимодействие программ пакета при функционировании управляющей ЭВМ

Стрелками на приведенном рисунке показана типовая передача управления между программами, формирующими пакет. При разбиении программного обеспечения на модули определяют требования к организации: обрабатываемых данных, процесса обработки, результатов обработки. Требования к организации обрабатываемых данных и результатов обработки формируются в виде протокола межмодульного обмена, а требования к процессу обработки формируются в виде алгоритмов программных модулей.

Выполнение каждой программы требует вполне определенных затрат процессорного времени, под которым понимается отрезок между началом и окончанием интерпретации соответствующего программного модуля. Обращение к датчикам сенсорной подсистемы и исполнительным устройствам бортового оборудования реализуется в виде полинга.

Работа осуществляется под управлением операционной системы ЭВМ, которая осуществляет т.н. диспетчеризацию обработки данных. Организация очередей задач определяется дисциплиной диспетчеризации, которая определяет их порядок формирования и обслуживания. В агрегатах с однопроцессорной ЭВМ как программа диспетчеризации, так и прикладные программы интерпретируются одним и тем же процессором, поэтому для повышения эффективности его использования алгоритм диспетчеризации и дисциплина обслуживания очереди должны быть максимально простыми. Именно по этой причине из трех возможных типов диспетчеризации: динамической, циклической и приоритетной - рекомендуется использовать две последних. Динамические дисциплины используются при нестабильных потоках задач, чего нет в рассматриваемых системах, и сложны в реализации.

4.1. Использование сетей Петри-Маркова в задачах моделирования процессов диспетчеризации

Как было показано ранее, *M-L*- параллельные полумарковские процессы с синхронизацией, применяемые, в том числе, в задачах разработки диспетчеров различных типов, оптимизирующих программный код, могут быть сведены к

Петри-Марковским моделям. Однако непосредственое применение ранее разработанной среды моделирования сетей Петри-Маркова оказывается затруднительным, что обуславливает целесообразность модификациии программного обеспечения путём введения дополнительных структур данных, соответствующих диспетчерам различных типов.

Разобъем взаимодействующие элементы при решении задачи диспетчеризации на *L+1* группу: непосредственно диспетчер и *L* параллельно функционирующих программных модулей, осуществляющих соответствующие функции в общей задаче управления роботизированной платформой. Действия в процессе решения задачи управления каждой группы и их пересечение во времени в виде сети Петри-Маркова могут быть представлены следующим образом (рис. 4.2).

Основными элементами сети здесь выступают диспетчер (обозначен индексом 9) и *L* блоков, моделирующих работу соответствующих программных модулей роботизированной платформы (индекс 1, 2 и 3 на рисунке). Каждый из перечисленных блоков относится к собственной подсети. Таким образом, общая сеть Петри-Маркова состоит из *L+1* подсети, в каждой из которых содержится собственный маркер, позволяющий конкретизировать состояние подсети в произвольный момент времени.

В рассматриваемой задаче диспетчеризации параллельных процессов подсети программных модулей не взаимодействуют между собой, а работают независимо друг от друга, взаимодействуя только с диспетчером. Для реализации подобного взаимодействия каждая подсеть содержит по два непримитивных двухвходовых перехода и по одному дополнительному блоку синхронизаторов, обозначенных на рисунке как синхронизаторы подсети (индексы 4, 5 и 6). В подсеть, к которой относится диспетчер, входят также *L* дополнительных блоков синхронизаторов диспетчера (индексы 7, 8 и 9). Для повышения наглядности элементы подсети диспетчера выделены на рисунке утолщёнными линями, а подсети, описывающие программные модули роботизированной платформы – штриховыми линями.

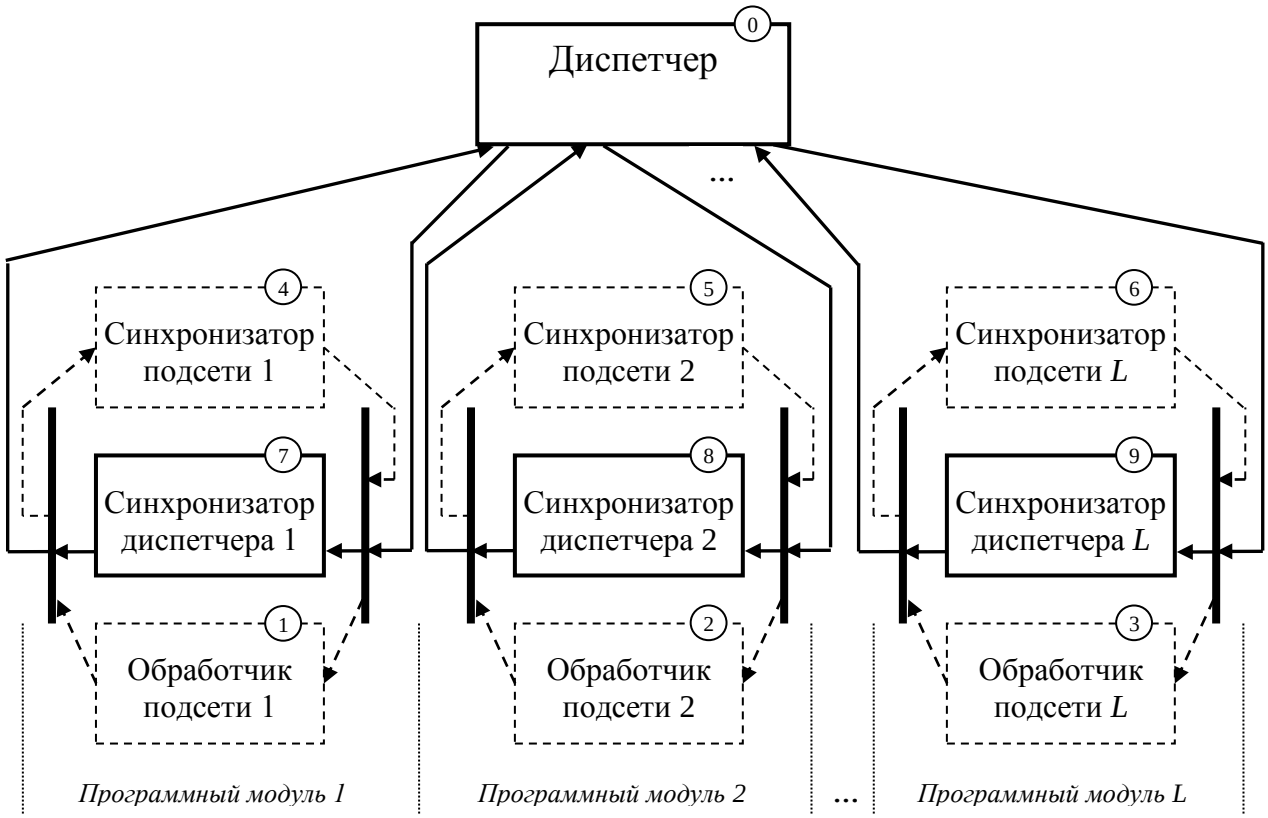


Рис. 4.2. Сеть Петри-Маркова в задаче диспетчеризации

Синхронизаторы подсетей являются начальными позициями в своих подсетях, то есть содержат собственный маркер в начальный момент моделирования. Аналогично блок диспетчера является начальной позицией подсети, реализующей механизм диспетчеризации, и также содержит собственный маркер в начале моделирования.

Позиции синхронизаторов диспетчера, служат для соблюдения формальных ограничений, налагаемых аппаратом сетей Петри-Маркова, и настраиваются таким образом, чтобы не вносить дополнительных задержек в перемещение маркера синхронизации. Позиции синхронизаторов подсетей, в зависимости от требований к моделируемой системе, могут иметь как нулевую, так и ненулевую, в том числе случайную, задержку в передаче маркеров соответствующих подсетей.

Общая логика алгоритма работы рассматриваемой сети следующая.

1. В начальный момент времени моделирования маркеры подсетей программных модулей перемещаются из блоков синхронизаторов подсетей во вход-

ные непримитивные переходы (показанные на рисунке справа от блоков обработчиков).

2. Диспетчер, в зависимости от выбранной дисциплины диспетчеризации, выдаёт свой маркер по одной из исходящих дуг в переход соответствующего программного модуля. В случае циклической диспетчеризации первым модулем выбирается модуль, описываемый подсетью 1. При приоритетной диспетчеризации выбирается модуль, имеющий наибольший приоритет. В случае с квазистохастической диспетчеризацией выбор модуля осуществляется псевдослучайным образом, в соответствии с априорными сведениями о характере распределения вероятностей вызова программных модулей.

3. Входной переход, в котором собрались оба маркера – собственный маркер программного модуля и маркер диспетчера – срабатывает. Собственный маркер переходит блок обработчика подсети, имитируя процесс работы соответствующего программного модуля, а маркер диспетчера с нулевой задержкой через позицию соответствующего синхронизатора диспетчера переходит в выходной переход (показанный на рисунке слева от блока обработчика).

4. Через время, определяемое параметрами блока обработчика подсети, в этот же переход попадает собственный маркер подсети. Это вызывает срабатывание выходного перехода и передачу маркера диспетчера в блок диспетчера, а маркера подсети – вновь в блок синхронизатора подсети.

5. Далее цикл повторяется для другого программного модуля, в соответствии с выбранной дисциплиной диспетчеризации.

4.1.1. Назначение программы

Программное обеспечение «Среда моделирования сетей Петри-Маркова» (далее программа, программное обеспечение) предназначено для визуального проектирования сетевых структур и последующего расчёта временных и вероятностных характеристик блужданий по полумарковским процессам методом имитационного моделирования в соответствии с теорией сетей Петри-Маркова. Раз-

работанная программа позволяет конечному пользователю выполнять следующие действия.

1. Осуществлять визуальное проектирование структуры сети Петри-Маркова путём

– добавления новых вершин типа «Позиция» или «Переход»;

– установления связей между вершинами (программа различает связи, в которых вершина-источник имеет тип «Позиция» от связей, в которых вершина-источник имеет тип «Переход», и контролирует соответствующий тип вершины-приёмника);

– удаления вершин и связей, ранее включённых в структуру сети по ошибке;

2. Указывать параметрические свойства сети посредством редактирования параметров каждого элемента (перечень параметров определяется типом редактируемого элемента сети);

3. Выполнять процедуру имитационного моделирования по разработанной сети Петри-Маркова, с накоплением статистики для расчёта оценок временных и вероятностных характеристик системы.

Программа позволяет сохранять для последующего использования созданную сеть (её структуру и параметры) в виде текстового файла, разработанный формат которого допускает прямое редактирование сети в любом текстовом редакторе. Этот файл также может быть вновь загружен в программу для последующего редактирования или моделирования.

Разработанное программное обеспечение осуществляет контроль возможных ошибок со стороны пользователя как при формировании структуры сети, так и при задании параметров отдельных вершин (например, контролируется факт равенства единице суммы вероятностей срабатывания каждого выхода из позиции).

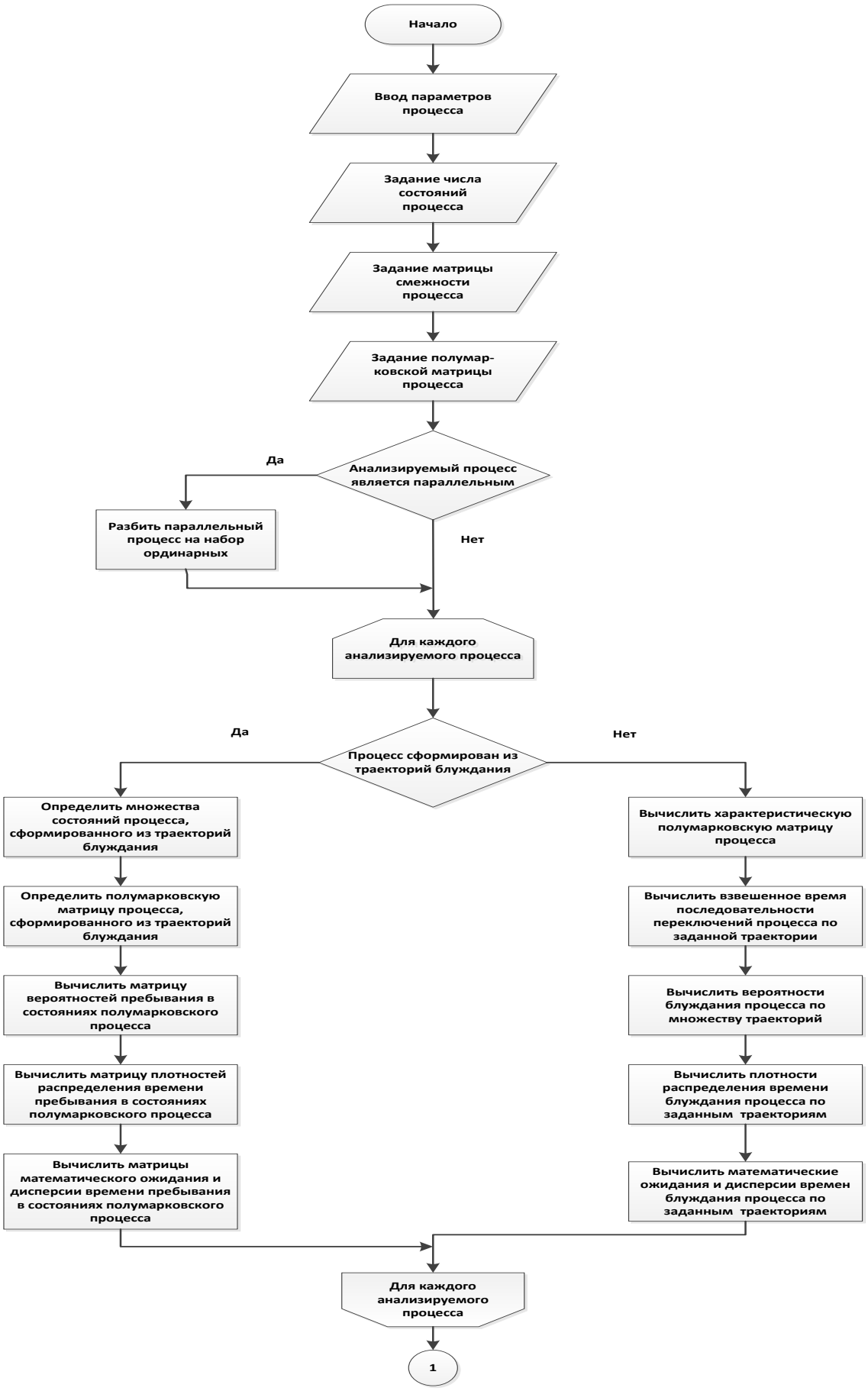


Рис. 4.3. Модуля для определения характеристик блужданий (начало)

Рис. 4.4. Модуль для определения характеристик блужданий (конец)

4.1.2. Использование программы в режиме редактирования сети

Основной исполняемый файл программы — PMEdit2021.exe — должен быть запущен из Проводника (или иным способом) для начала работы с программой. Её основное окно имеет вид, показанный на рис. 4.5.

Программа имеет стандартный для приложений Windows интерфейс, основу которого составляет панель инструментов (слева) и поле для редактирования сети Петри-Маркова (справа). Содержание панели инструментов меняется в зависимости от текущего режима работы программы [143].

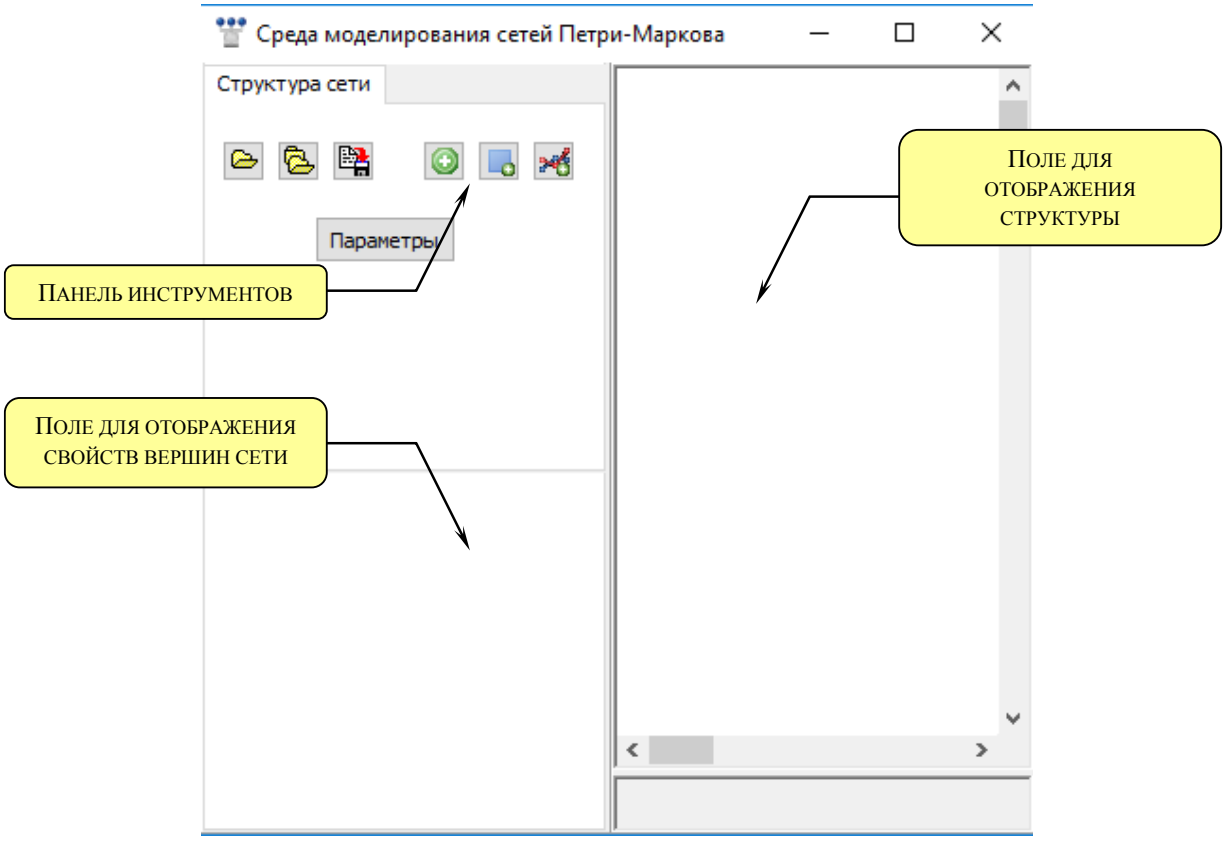


Рис. 4.5. Внешний вид основного окна программы

Можно выделить три основных режима работы:

1) формирование/редактирование структуры сети Петри-Маркова;

2) задание/редактирование параметров элементов сети Петри-Маркова;

3) имитационное моделирование на созданной сети.

4.1.3. Формирование/редактирование структуры сети Петри-Маркова

Режим формирования новой сети является режимом «по умолчанию» при запуске программы. Признаком того, что программа находится в этом режиме, является заголовок основной панели инструментов «Структура сети».

Как указывалось выше, сеть Петри-Маркова может рассматриваться как ориентированный граф, вершины которого могут относится к одному из двух типов:

1) позиции — отображаются кругами;

2) переходы — отображаются прямоугольниками зелёного цвета.

Цвет позиций в режиме формирования структуры выбран красным. При переходе в другие режимы работы программы цвета позиций могут изменяться в зависимости от того, к какой подсети относится конкретная позиция. Цвет перехода всегда остаётся зелёным.

Для добавления позиции на экран необходимо нажать на кнопку  «Добавить позицию», после чего следует переместить указатель мыши в правую половину окна (на поле для отображения сети) и в точке, где планируется разместить элемент, нажать кнопку мыши. При этом на экране в этой точке должен появиться круг красного цвета, обозначающий новую позицию. Внутри круга будет отображён порядковый номер позиции (нумерация позиций начинается с нуля).

Позицию можно перемещать по экрану с помощью мыши (кнопку необходимо нажать и удерживать при перемещении).

Аналогично для добавления очередного перехода необходимо нажать на кнопку  «Добавить переход». После этого следует переместить указатель мыши в правое поле отображения сети и в точке, где планируется разместить переход, нажать кнопку мыши. При этом на экране в этой точке должен появиться квадрат зелёного цвета, обозначающий новый переход, также с указанием его порядкового номера. Переход можно перемещать по экрану с помощью мыши (кнопку необходимо нажать и удерживать при перемещении).

На рис. 4.6. показан пример размещения нескольких позиций и переходов в режиме формирования структуры СПМ.

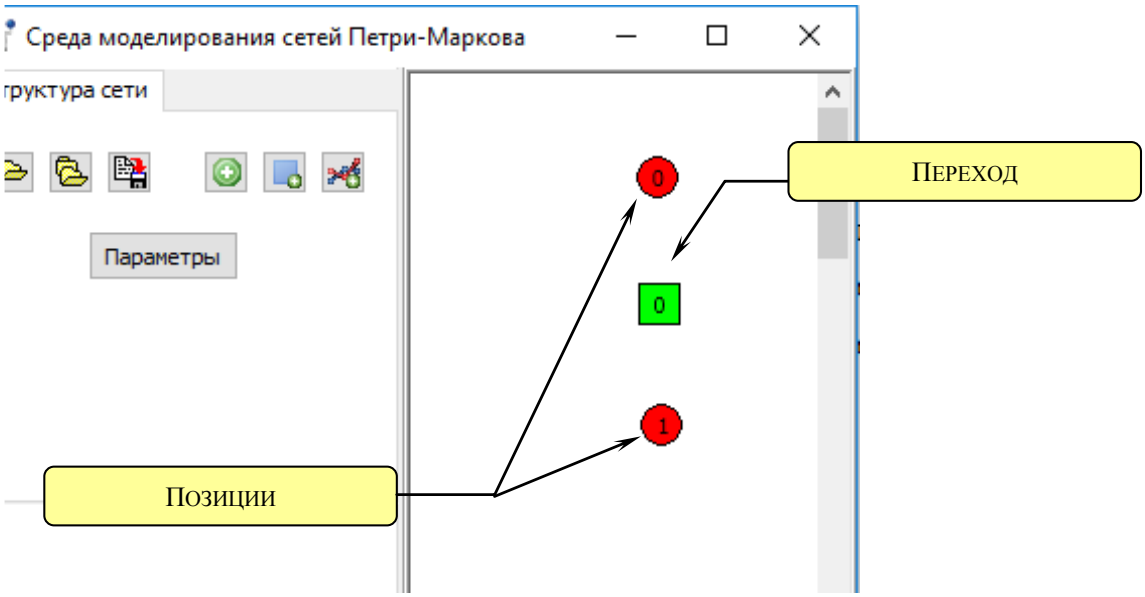


Рис. 4.6. Пример добавления двух позиций и трёх переходов

Для указания связей между позициями и переходами используются дуги. Для создания новой дуги необходимо нажать кнопку  «Добавить дугу», а затем последовательно выделить начальную вершину сети (для которой дуга является исходящей) и конечную вершину (для которой дуга является входящей). Дуги, для которых вершинами-источниками являются позиции, отображаются красным цветом, а для которых вершинами-источниками являются переходы - зелёным цветом (рис. 4.12).

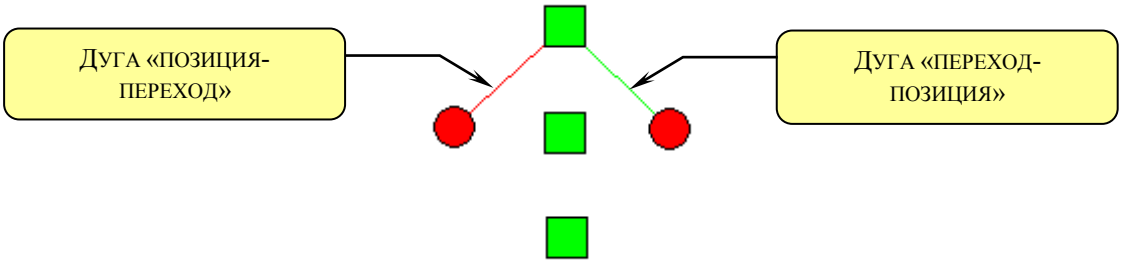


Рис. 4.7. Примеры дуг различных типов

Дуги сложной формы формируются из отрезков прямых с помощью нескольких щелчков мышью в соответствующих точках поля редактирования СПМ (рис. 4.8). При этом в местах соединения отрезков отображаются промежуточные узлы того же цвета, что и соответствующая дуга.

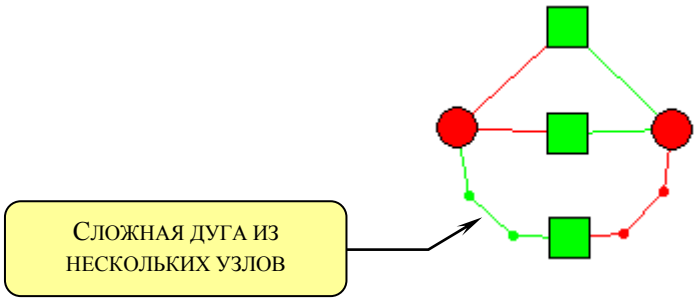


Рис. 4.8. Пример дуги, содержащей
несколько промежуточных узлов

Отрезки, составляющие дугу сложной формы, можно перераспределять по экрану, путём перемещения мышью соответствующих промежуточных узлов.

При ошибочном размещении каждый элемент сети (включая дуги сложной формы) может быть удалён. Для этого на соответствующей вершине необходимо щёлкнуть правой кнопкой мыши и в контекстном меню выбрать команду «Удалить переход» или «Удалить позицию» (рис. 4.9).

Команда «№ перехода» или «№ позиции» из контекстного меню отображает на экране диалоговое окно, в котором приводится информация об идентификационном номере соответствующей вершины сети (рис. 4.10).

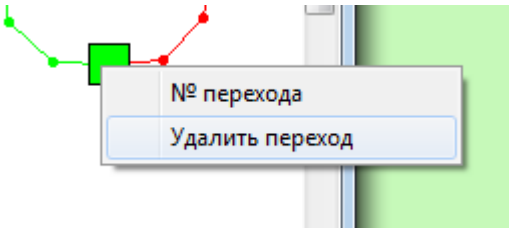


Рис. 4.19. Пример вызова контекстного меню

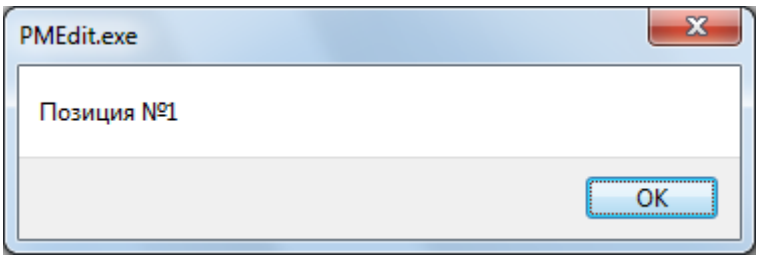


Рис. 4.10. Диалоговое окно отображения идентификационного номера вершины сети

При удалении позиции или перехода автоматически удаляются все дуги, непосредственно связанные с этой вершиной. В случае если вершина соединялась дугой, содержащей несколько узлов, дополнительные звенья такой дуги должны быть удалены в порядке, описанном выше, при этом правую кнопку мыши следует нажимать на соответствующих узлах удаляемой дуги.

Для сохранения структуры сети в файле на внешнем носителе необходимо использовать команду «Сохранить структуру сети» (кнопка). При её выборе на экране отображается стандартное диалоговое окно выбора места сохранения файла и его имени (рис. 4.11). Сохраняемые файлы по умолчанию имеют расширение *.pmn (Petri-Markov Net). Описание данного формата приведено далее, в соответствующем подразделе отчёта.

Сохранённые файлы, содержащие только структуру СПМ, могут быть прочитаны с диска с помощью команды «Загрузить структуру сети» (кнопка). При её выборе на экране появляется окно, аналогичное показанному на рис. 4.12, в котором оператор может выбрать загружаемый файл.

Если файл сохранялся из режима задания/редактирования параметров, и включал не только структуру, но и параметры элементов сети (см. ниже), то он может быть загружен с помощью команды «Загрузить структуру и параметры сети» (кнопка). При её выборе на экране появляется окно, аналогичное показанному на рис. 4.11, но из файла, при этом, считаются сведения не только о структуре, но и о параметрах загружаемой сети.

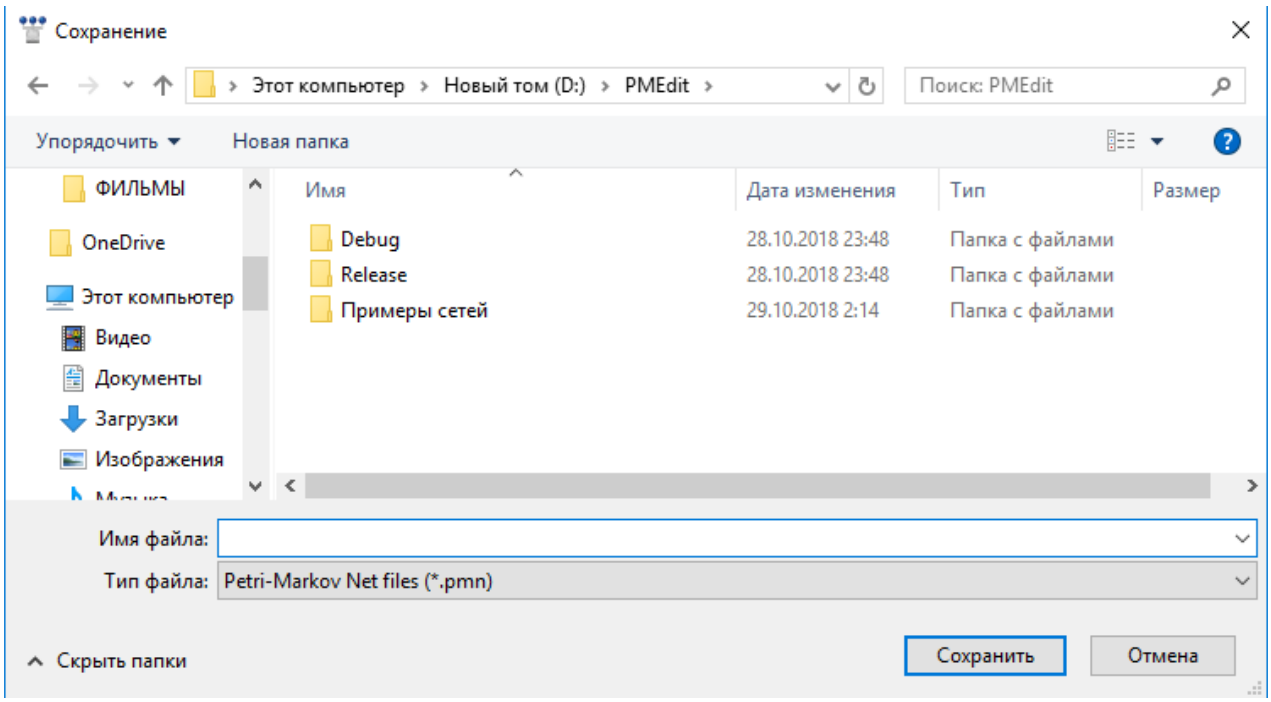


Рис. 4.11. Диалоговое окно сохранения файла

После завершения создания/редактирования структуры сети Петри-Маркова следует нажать кнопку «Параметры» для перехода в режим редактирования параметров отдельных вершин сети. На экране, при этом, отобразится диалоговое окно с предупреждающим сообщением (рис. 4.12).

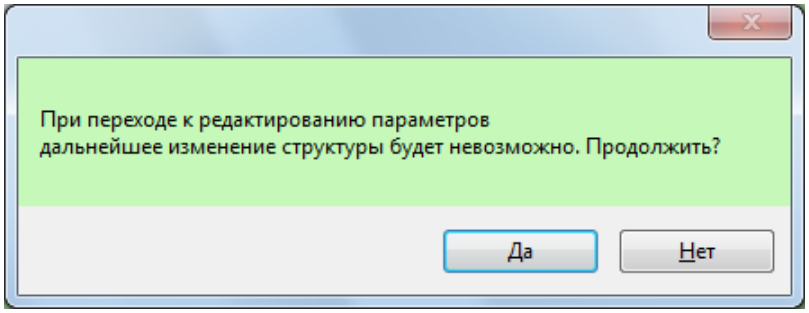


Рис. 4.12. Предупреждающее сообщение при переходе к режиму редактирования

В случае если оператор выбирает «Нет», программа возвращается в исходный режим создания/редактирования структуры сети. При подтверждении «Да» программа переходит в режим редактирования параметров и дальнейшее изменение структуры сети становится невозможным.

4.1.4. Задание/редактирование параметров сети Петри-Маркова

При переходе к режиму задания/редактирования параметров сети внешний вид панели инструментом изменяется (рис. 4.13).

В верхней части панели инструментов в этом режиме работы программы остаётся лишь кнопка сохранения данных в файл  , однако её функционал изменятся в сравнении с режимом формирования/редактирования структуры сети. Теперь она выполняет команду «Сохранить структуру и параметры сети», то есть помимо сведений о структуре СПМ, позволяет сохранить в файл информацию о свойствах каждого элемента сети.

Кнопки «Структура» и «Модель» служат для переключения программы соответственно в режим редактирования структуры СПМ и в режим имитационного моделирования. Следует обратить внимание, что при возвращении в режим редактирования структуры сведения о параметрах элементов сети могут быть утеряны (при удалении элементов) или не заполнены (при добавлении новых элементов).

В нижней части панели инструментов в этом режиме отображается поле для задания свойств текущего выбранного элемента сети.

При выборе левой кнопкой мыши некоторой позиции её текущие свойства будут отображены в нижней части панели инструментов (рис. 4.13).

Поле «Позиция № *NN*» — не редактируемое поле, предназначенное для отображения уникального идентификационного номера позиции, который присваивается автоматически на этапе формирования структуры сети.

Поле «Подсеть» задаёт номер подсети, к которой относится данная позиция. Номер представляет собой целое неотрицательное число (по умолчанию 0). Предполагается, что позиции, непосредственно связываемые между собою через примитивные переходы, должны относиться к одной подсети. Для упрощения визуального обнаружения ошибок отнесения позиции к неправильной подсети, и для лучшего восприятия общей структуры сети человеком-оператором цвет пози-

ции после изменения номера подсети меняется с красного на иной, в соответствии со следующей схемой:

0 – пурпурный;

1 – бирюзовый;

2 – тёмно-синий;

3 – бордовый;

4 – жёлтый;

5 – розовый;

6 – голубой;

7 – зелёный;

8 – серебряный;

иной номер — красный.

Переключатель «Тип позиции» определяет один из трёх возможных типов, к которому относится текущая позиция. По умолчанию всем позициям присваивается тип «промежуточная». В каждой подсети одна из позиций должна быть определена как начальная. Именно в начальной позиции в процессе моделирования создаётся маркер, который и определяет функционирование данной подсети. Кроме того, в каждой подсети может быть одна (или даже несколько) конечная позиция, при достижении которой маркер далее не перемещается и процесс моделирования работы данной подсети останавливается. В некоторых случаях конечную позицию в той или иной подсети можно не указывать, но по крайней мере одна конечная позиция во всей сети должна существовать.

«Вероятности выхода в переход» — таблица, количество строк в которой соответствует количеству исходящих из позиции дуг (красного цвета). Первый столбец содержит идентификационные номера переходов, с которыми позиция связана с дугами. Второй столбец содержит вероятности срабатывания каждого из этих переходов.

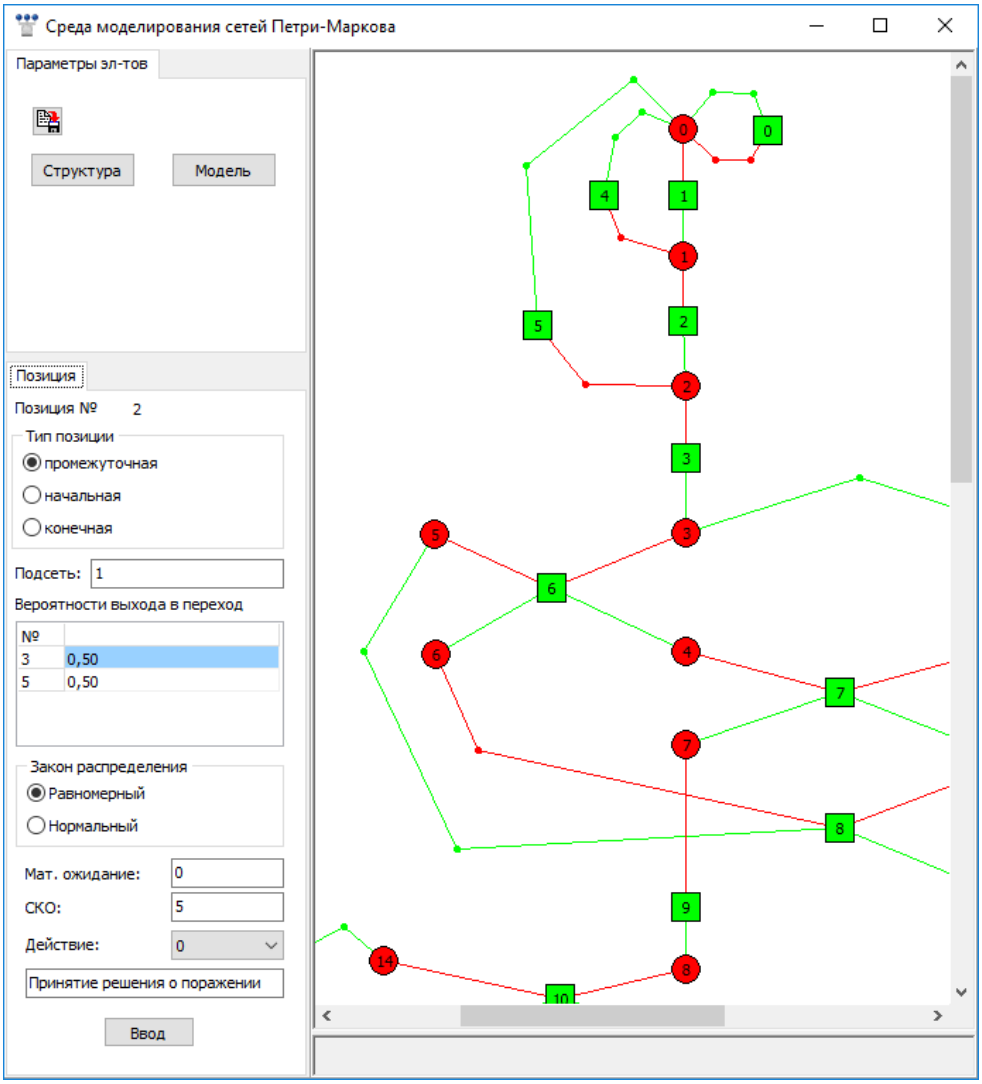


Рис. 4.13. Внешний вид программы в режиме редактирования параметров

Редактирование вероятностей осуществляется непосредственно в таблице, щелчком мыши на соответствующей строке и последующим вводом с клавиатуры необходимого значения. Вероятности вводятся и отображаются с точностью до 2 знаков.

Для подтверждения внесённых изменений необходимо нажать кнопку «Ввод» в нижней части окна.

Для всех позиций за исключением конечной сумма вероятностей по всем выходам должна быть равна 1. В случае нарушения этого требования внесённые изменения не будут зафиксированы, а на экране появится предупреждающее сообщение. Для конечных позиций, не имеющих выходных дуг, проверка корректности этого критерия не выполняется.

Для каждой исходящей дуги можно определить тип закона распределения времени пребывания фишки в позиции перед выходом в переход и его параметры. Тип закона выбирается переключателем «Закон распределения», а параметры — математическое ожидание и среднеквадратическое отклонение — непосредственным вводом в поля «Мат. ожидание» и «СКО».

Все вносимые изменения должны подтверждаться нажатием на кнопку «Ввод», в противном случае они не будут сохранены.

Свойства некоторого перехода показаны на рис. 4.14.

«Переход № *NN*» — не редактируемое поле; отображает уникальный идентификационный номер перехода, автоматически присвоенный ему при редактировании структуры сети.

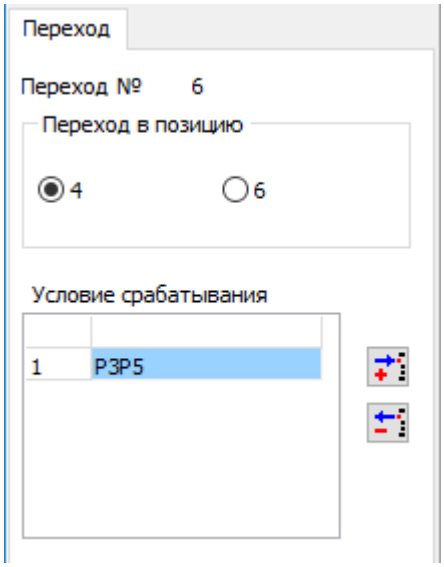


Рис. 4.14. Свойства перехода

Переключатель «Переход в позицию» — задаёт номер позиции, для перехода в которую осуществляется редактирование условие срабатывания перехода. Состав отображаемых идентификаторов позиций динамически изменяется для каждого выбираемого перехода в зависимости от структуры сети.

Таблица «Условие срабатывания» содержит набор логических условий срабатывания перехода по отношению к выбранной позиции.

Условие срабатывания задаётся в виде логической функции, заданной в совершенной дизъюнктивной нормальной форме.

Количество и состав входных переменных, образующих отдельную конъюнкцию, определяются количеством и составом позиций, из которых в данный переход могут поступать фишки. Каждая такая входная переменная кодируется символом P, за которым следует идентификационный номер позиции. Например, P3. Инверсия отдельной переменной, входящей в конъюнкцию, задаётся символом p, за которым следует идентификационный номер позиции. Например, p5.

Отдельная строка таблицы соответствует отдельной конъюнкции. Для добавления новой конъюнкции необходимо нажать кнопку  «Добавить конъюнкцию». При этом в таблице «Условие срабатывания» будет автоматически добавлена новая строка.

Для удаления ошибочно внесённой конъюнкции следует выбрать соответствующую строку с помощью указателя мыши и нажать на кнопку  «Удалить конъюнкцию».

После завершения задания параметров элементов СПМ и сохранения структуры и параметров получившейся сети на диск можно переключить программу в режим моделирования путём нажатия на кнопку «Модель» в верхней части панели инструментов.

4.1.5. Метод имитационного моделирования

При переходе к режиму имитационного моделирования внешний вид панели инструментом изменяется (рис. 4.21).

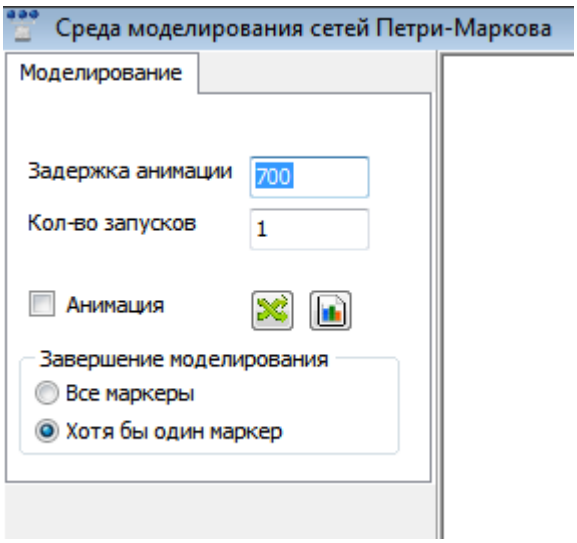


Рис. 4.15. Внешний вид панели инструментов
в режиме имитационного моделирования

В процессе моделирования программа может воспроизводить анимированно перемещение маркеров по узлам подсетей. В этом случае переключатель «Анимация» должен быть активирован (установлена отметка), а поле «Задержка анимации» задаёт минимальный интервал времени (в миллисекундах) между последовательными перемещениями маркеров из одной позиции в другую. В случае, если переключатель «Анимация» не выбран, параметр «Задержка анимации» игнорируется, а процесс моделирования протекает максимально быстро.

Поле «Кол-во запусков» определяет число пробных прогонов модели в процессе сбора статистики моделирования. По умолчанию эта величина равна единице.

Переключатель «Завершение моделирования» определяет логику, в соответствии с которой программа завершает процедуру имитационного моделирования. По умолчанию используется режим «Хотя бы один маркер», в котором процедура завершается при первом попадании любого маркера в конечную позицию. В случае выбора режима «Все маркеры» процедура имитационного моделирования будет продолжаться до тех пор, пока все маркеры, циркулирующие в сети, не окажутся в соответствующих конечных вершинах своих подсетей.

В основу алгоритма имитационного моделирования, применённого в про-

грамме, положен дискретно-событийный подход. Его суть сводится к представлению процесса моделирования потоком дискретных событий - перемещений маркеров по узлам СПМ [112].

Логика работы СПМ предполагает случайных характер времени пребывания маркера в позиции, при этом это время может быть достаточно большим. Таким образом, вместо обычно применяемого постоянного небольшого шага моделирования в рассматриваемом методе предлагается использовать переменный шаг времени, величину которого программа вычисляет исходя из анализа ближайшего следующего события. После определения ближайшего по времени следующего события глобальное время модели должно быть скорректировано на соответствующую дискретную величину.

Метод имитационного моделирования представлен следующей последовательностью действий.

1. Обнулить счётчик глобального модельного времени

2. Выбрать в каждой из подсетей, образующих общую СПМ, начальные позиции, и создать в каждой найденной начальной позиции по одному маркеру.

3. Для каждого созданного маркера выполнить шаги 4-6:

4. В соответствии с заданными вероятностями P выхода из позиции определить случайным образом дугу, по которой маркер выйдет из позиции в переход.

5. Для выбранной дуги в соответствии с заданным законом распределения PLaw и параметрами математического ожидания MeanVal и среднеквадратического отклонения MRS определить случайное время пребывания dT данного маркера в текущей позиции.

6. Если маркеры рассмотрены не все, выбрать следующий и перейти к шагу 4, иначе — к шагу 7.

7. Из всех маркеров определить тот, который имеет наименьшее время пребывания в позиции dTmin — зафиксировать индекс выбранного маркера для последующего выполнения полушага (перемещения из позиции в переход).

8. Уменьшить для всех маркеров оставшееся время пребывания в позициях на величину dTmin и увеличить на эту же величину глобальное время моделирования.

9. Изъять выбранный на шаге 7 маркер из соответствующей позиции и по ранее выбранной дуге переместить его в переход.

10. Для перехода, получившего новый маркер, проверить выполнение условия срабатывания перехода.

11. Если условие срабатывания не выполнено, переход к шагу 16.

12. Выбрать все маркеры, определившие условие срабатывания перехода.

13. Для каждого выбранного маркера выполнить второй полушаг из перехода в связанную позицию, соблюдая требование соответствия индексов подсетей.

14. Зафиксировать необходимую статистику (время пребывания маркера в позиции, время ожидания маркера в переходе и т.п.).

15. Если маркер достиг конечной позиции и условием завершения является достижение хотя бы одним маркером конечной позиции, то выдать сообщение об успешном окончании моделирования и завершить моделирование, иначе — переход к шагу 16.

16. Если существуют маркеры, находящиеся в позициях, отличных от конечной, то вернуться к шагу 7, иначе — переход к шагу 17.

17. Если все маркеры находятся в конечной позиции — выдать сообщение об успешном окончании моделирования и завершить моделирование, иначе — переход к шагу 18.

18. Выдать сообщение о «зависании» системы и завершить моделирование.

4.1.6. Структура классов объектов

Математический аппарат сетей Петри-Маркова задаёт обладающие высокой степенью формализации объекты (элементы сетей) и правила их взаимодействия между собой, что делает целесообразным применение объектно-

ориентированного подхода при разработке программного обеспечения [111]. Разные типы элементов СПМ (позиции, переходы, дуги, маркеры) могут быть представлены в программе как объекты соответствующих классов, динамически создаваемые и связываемые друг с другом в соответствии с логикой построения СПМ.

Рассмотрим основные классы объектов, реализованные в программе.

Основными элементами СПМ являются позиция и переход, которые в программе представлены в виде классов PMPosition и PMTransition соответственно. Структура сети задаётся с помощью дуг, программным аналогом которых является класс PMArc. Маркеры, перемещаемые по сети в процессе имитационного моделирования, в программе представлены классом PMMarker.

Все перечисленные объекты реализуют логику работы сетей Петри-Маркова, и не содержат средств для визуализации. Это обусловлено стремлением снизить возможные затраты на портируемость разработанного программного обеспечения на другие платформы в случае появления подобной необходимости.

Отображение перечисленных выше объектов на экране компьютера осуществляется с помощью стандартных классов библиотеки Visual Components Library — TForm, TShape и т.п. Это, с одной стороны, позволяет сосредоточиться на отработке внутренней логики программы, минимизируя усилия на разработку интерфейса, а, с другой стороны, позволит, при необходимости, реализовать интерфейс на другой платформе другими средствами, сохранив ядро моделирования в неизменном виде.

Класс PMPosition включает в свою структуру следующие свойства и методы.

Свойство ind (целый) – номер позиции. Для сокращения времени создания объектов класса PMPosition в программе реализован статический массив POS, размер которого заведомо больше возможного числа позиций в сети (в текущей версии принят равным 1000 элементов). Индекс элемента массива, в котором хранится данный экземпляр класса, содержится в поле ind.

Свойство SubNet (целый) – индекс подсети, указывающий, к какой из подсетей относится данная позиция. Этот индекс является едиными для всех позиций, относящихся к данной подсети.

Свойство TypePos (перечислимый) – тип позиции (0 – промежуточная, 1 – начальная, 2 – конечная). Особенностью начальной позиции является то, что перед началом моделирования во всех начальных позициях создаются маркеры, которые в последующем циркулируют в пределах соответствующих подсетей, не покидая их.

Свойство NArcOut (целый) – текущее количество дуг, выходящих из данной позиции.

Свойство ArcInd[MaxArc] (массив целых) – индексы переходов, в которые выходят соответствующие дуги. Данный массив используется для описания структурных связей данной позиции с соседними переходами. В нём хранятся индексы переходов, с которыми данная позиция связана выходными дугами. Количество дуг, хранящихся в этом массиве, задаётся полем NArcOut.

Свойство P[MaxArc] (массив вещественных) – вероятности выхода в соответствующие дуги. Если позиция содержит более одной выходной дуги, то в процессе имитационного моделирования маркер, находящийся в позиции, случайным образом покидает её по одной из них. Вероятность выбора конкретной дуги хранится в массиве P. Его размер совпадает с размером ArcInd.

Свойство PLaw[MaxArc] (массив перечислимых) – типы законов распределений, связанных с дугами (0 – равномерное распределение, 1 – нормальное распределение).

Свойство MeanVal[MaxArc] (массив вещественных) – массив математических ожиданий соответствующих законов распределения.

Свойство MRS[MaxArc] (массив вещественных) – массив среднеквадратических отклонений соответствующих законов распределения.

С каждой дугой также связан определённый тип закона распределения времени пребывания фишки в позиции, закодированный целым числом (хранится в массиве PLaw), а также параметры этого закона, такие как математическое ожи-

дание (массив MeanVal) и среднеквадратическое отклонение (массив MRS). Размеры всех этих массивов также равны NArcOut. В текущей версии программы реализованы равномерное и нормальное распределение, для описания которых достаточно указанных статистических параметров (нормальное распределение вычисляется приближённо, как сумма нескольких равномерно распределённых величин).

Свойство Descript (строковый) – краткое текстовое описание текущей позиции. Используется для повышения наглядности в процессе создания/редактирования сети (описание выводится в окне подсказки при наведении указателя мыши на позицию).

Помимо перечисленных полей в структуру класса PMPosition входит несколько методов.

Метод int randArcOutNum() — функция, генерирующая случайным образом порядковый номер в массиве дуг (для определения дуги, по которой фишка уйдёт из данной позиции) с учётом заданных вероятностей выхода по каждой из дуг. Функция возвращает порядковый номер из массива ArcInd индексов переходов.

Метод int randTime(int i) — функция, генерирующая время пребывания фишки в позиции, в соответствии с ранее выбранной траекторией дальнейшего его движения (определённой функцией randArcOutNum()) и типом и параметрами закона распределения PLaw, MeanVal и MRS. Вход int i — порядковый номер выбранной дуги в массиве ArcInd; выход — время пребывания.

Объект типа «переход» реализован в виде класса PMTransition. Этот класс включает в себя следующие свойства и методы.

Свойство ind (целый) — индекс перехода. Объекты класса PMTransition хранятся в статическом массиве TRA достаточно большого для практических применений размера (в текущей версии – 1000 элементов). Номер элемента массива, в котором хранится данный экземпляр класса, дублируется в поле ind.

Свойство ArcInInd[MaxArc] (массив целых) — массив индексов позиций, из которых в данный переход приходят соответствующие входные дуги.

Свойство NArcIn (целый) — текущее количество дуг, входящих в данный переход (определяет число заполненных элементов в массиве NArcIn).

Свойство ArcOutInd[MaxArc] (массив целых) – массив индексов позиций, в которые попадают маркеры из данного перехода после его срабатывания.

Свойство NArcOut (целый) — текущее количество дуг, выходящих из данного перехода (определяет число заполненных элементов в массиве ArcOutInd).

Массивы ArcInInd и ArcOutInd в совокупности со свойствами NArcIn и NArcOut используются, таким образом, для описания структурных связей данного перехода с позициями сети. Двойственность данных массивов отличает переходы от позиций. В позициях индексы входных дуг и связанных с ними переходов не сохраняются, поскольку с точки зрения процедуры моделирования работы сети безразлично, по какой входной дуге в позицию поступила фишка.

Свойство nMarkers[MaxArc] (массив целых) – массив для хранения числа маркеров, пришедших по соответствующим входным дугам

Свойство Logic[MaxArc] (массив строк) – массив для хранения конъюнкций, определяющих срабатывание перехода (количество элементов в нём совпадает с числом элементов в ArcOutInd). Логические условия должны задаваться в дизъюнктивной (ДНФ) или совершенной (СДНФ) дизъюнктивной нормальной форме.

Свойство NMarkers[NArcIn] (массив целых) – вспомогательный массив, предназначенный для учёта количества маркеров, пришедших в переход из соответствующей позиции. Размер совпадает с массивом входных дуг ArcInInd. Массив используется в процедуре определения условий срабатывания перехода.

Метод PMTransition() – конструктор по умолчанию, который осуществляет обнуление вспомогательного массива nMarkers при создании объекта типа «переход».

При проектировании сетей, содержащих несколько подсетей, предлагается все переходы внутри подсети рассматривать как примитивные, т.е. имеющие один вход и один выход. Они, фактически, не содержат никакой логики срабатывания, и выполняют функцию элементарных соединителей последовательностей пози-

ций. Условие срабатывания такого перехода по умолчанию формулируется в виде `Pxx`, где `xx` – номер предшествующей переходу позиции, из которой в переход приходит маркер.

В случае, когда подсети должны взаимодействовать между собой, используются непримитивные переходы, в которых количество входов и выходов больше единицы.

В этом случае предполагается, что количество входов непримитивного перехода равно количеству выходов и, соответственно, равно количеству взаимодействующих в этой точке подсетей. Условие срабатывания любого непримитивного перехода всегда одинаково — переход срабатывает после того, как на все его входы поступили маркеры из соответствующих подсетей, т.е. все условия срабатывания формулируются как конъюнкции вида `P0P1P2`….

Срабатывание перехода заключается в передаче маркера из входной позиции в выходную. При этом маркер не должен покидать своей подсети, т.е. он попадает в тот выход, который связан с позицией из соответствующей подсети.

Таким образом, непримитивные переходы выполняют функцию синхронизации процессов, протекающих в различных подсетях.

В общем случае допускается для записи логического условия срабатывания перехода использовать дизъюнктивную нормальную форму, в которой имена некоторых позиций из числа связанных дугами с рассматриваемым переходом опущены. В этом случае программа рассматривает такие пропущенные позиции, как не влияющие на условие срабатывания перехода. Аналогично, если в записи выражения присутствует имя позиции, не связанной с данным переходом, оно игнорируется, как не влияющее на условие срабатывания данного перехода.

В программе каждая конъюнкция представляет собой строку символов вида `Pxx` или `pxx`, в которой `xx` определяет номер соответствующей позиции, а `P` или `p` задаёт соответственно наличие или отсутствие в переходе маркеров из этой позиции. Между собой конъюнкции разделяются знаком «|».

Класс PMMarker включает в свою структуру следующие свойства и методы.

Свойство SubNet (целый) – номер подсети, в которой был порождён данный маркер. Это поле контролируется при прохождении маркера через каждый непримитивный переход, таким образом маркер из непримитивного перехода всегда переходит по дуге, ведущей в позицию той же подсети, в которой он был порождён.

Свойство TimeProc (вещественный) – совокупное время пребывания маркера в позициях по итогам имитационного моделирования (суммарное время обработки/действия).

Свойство TimeWait (вещественный) – совокупное время пребывания маркера в переходах (суммарные потери времени на синхронизацию процессов в различных подсетях, с которыми взаимодействует подсеть, породившая маркер).

Свойство nPosEnt (целый) – пройденное число полушагов вида «переход-позиция».

Свойство nTraEnt (целый) – пройденное число полушагов вида «позиция-переход».

Поля TimeProc, TimeWait, nPosEnt и nTraEnt используются для накопления статистики в процессе моделирования.

Свойство dT (вещественный) – при нахождении в позиции поле интерпретируется как сгенерированное случайным образом время пребывания фишки в позиции; при нахождении в переходе — начальное время попадания фишки в переход.

Свойство inPos (логический) – признак пребывания в текущий момент маркера в позиции (true) или в переходе (false).

Свойства indPos (целый) и indTra (целый) – индексы, интерпретируемые в зависимости от состояния inPos, следующим образом: если inPos равен true, т.е. маркер находится в позиции, то indPos соответствует индексу текущей позиции, а indTra – индексу перехода, в который должен попасть маркер; если inPos равен false, т.е. маркер находится в переходе, то indPos соответствует индексу позиции, из которой пришёл маркер, а indTra – индексу текущего перехода, в котором находится маркер.

Метод PMMarker – конструктор по умолчанию.

Объекты класса PMMarker хранятся в отдельном массиве Mark, размер которого равен 1000 элементов.

Объект типа «дуга» реализован в виде класса PMArc. Этот класс включает в себя следующие свойства.

Свойство type0 (целый) – для сложной дуги, включающей несколько узлов, определяет тип объекта, из которого вышел первый отрезок дуги (1 — позиция; 2 — переход).

Свойство ind1 (целый) – тип объекта, который является началом данного отрезка дуги (1 — позиция, 2 — переход, 3 — узел дуги).

Свойство type1 (целый) – тип объекта, который является началом данного отрезка дуги (1 — позиция, 2 — переход, 3 — узел дуги).

Свойство ind2 (целый) – идентификационный номер объекта, который является окончанием данного отрезка дуги.

Свойство type2 (целый) – тип объекта, который является окончанием данного отрезка дуги (1 — позиция, 2 — переход, 3 — узел дуги).

Методы в данном классе отсутствуют

4.1.7. Разработка программного обеспечения

Разработка программы осуществлялась на языке C++ с использованием интегрированной среды быстрого проектирования CodeGear C++ Builder.

Структурно программа включает в себя следующие модули:

PMEdit.cbproj — файл описания проекта — служебный файл, необходимый для компиляции отдельных модулей и сборки в единый исполняемый файл;

PMEdit.cpp — исходный текст функции WinMain, определяющей точку старта программы;

PMEdit.res — файл, содержащий некоторые общие ресурсы программы;

Unit1.cpp, Unit1.h, Unit1.dfm — исходный файл, файл-заголовок и файл-форма, содержащие определение основного класса программы TForm1 и её основного функционала;

Unit2.cpp, Unit2.h, Unit2.dfm — исходный файл, файл-заголовок и файл-форма, содержащие определение класса окна статистики TForm2.

4.1.8. Визуальный интерфейс программы

Внешний вид программы определяется классом TForm1, порождённым от стандартного компонента-формы TForm. Он определяет, как внешний вид основного окна приложения, так и общую логику работы программы.

В табл. 4.1 приведено описание визуальных и невизуальных компонентов класса TForm1.

Таблица 4.1

Описание визуальных и невизуальных компонентов класса TForm1

Наименование	**Тип**	**Описание**
Button1	TButton *	Кнопка «Модель» для переключения в режим имитационного моделирования (связана с TabSheet5)
Button2	TButton *	Кнопка «Структура» для возврата в режим редактирования структуры сети (связана с TabSheet4)
Button3	TButton *	Кнопка «Параметры» для перехода в режим редактирования параметров сети (связана с TabSheet3)
CheckBox1	TCheckBox *	Переключатель «Анимация» (определяет необходимость анимации процесса моделирования)

Наименование	Тип	Описание
Edit1	TEdit *	Поле ввода «Мат. ожидание» (связано с TabSheet1)
Edit2	TEdit *	Поле ввода «СКО» (связано с TabSheet1)
Edit4	TEdit *	Поле ввода «Задержка анимации» (определяет величину задержки, в миллисекундах, между событиями при включённом переключателе «Анимация»)
Edit5	TEdit *	Поле ввода «Кол-во запусков» (определяет количество пусков процедуры имитационного моделирования в процессе сбора статистических данных)
Edit6	TEdit *	Поле ввода «Действие» (задаёт краткое текстовое описание действия, соответствующего редактируемой позиции)
Edit7	TEdit *	Поле ввода «Подсеть» (задаёт номер подсети, к которой относится редактируемая позиция)
Image1	TImage *	Изображение, предназначенное отображения системы дуг в соответствии с заданной структурой сети Петри-Маркова; используется совместно с Image2
Image2	TImage *	Скрытое изображение, в которое осуществляется последовательная отрисовка системы дуг в соответствии с заданной структурой сети Петри-Маркова; после завершения рисования осуществляется быстрое копирование картинки в Image1 (для предотвращения нежелательного мерцания при перерисовке)

Наименование	Тип	Описание
Label7	TLabel *	Метка для отображения № редактируемого перехода (связана с TabSheet2)
Label8	TLabel *	Метка для отображения № редактируемой позиции (связана с TabSheet1)
N1	TMenuItem *	Команда «№ позиции» из контекстного меню PopupMenu1
N2	TMenuItem *	Команда «Удалить позицию» из контекстного меню PopupMenu1
N3	TMenuItem *	Команда «№ перехода» из контекстного меню PopupMenu2
N4	TMenuItem *	Команда «Удалить переход» из контекстного меню PopupMenu2
N5	TMenuItem *	Команда «№ узла дуги» из контекстного меню PopupMenu3
N6	TMenuItem *	Команда «Удалить узел дуги» из контекстного меню PopupMenu3
OpenDialog1	TOpenDialog *	Компонент для отображения диалогового окна «Открыть файл»
PageControl1	TPageControl *	Компонент управления страницами, группирующий элементы, отвечающие за работу со свойствами позиций и переходов
PageControl2	TPageControl *	Компонент управления страницами, группирующий элементы, отвечающие за реализацию действий пользователя по редактированию структуры сети
Panel1	TPanel *	Вспомогательный элемент, на котором размещаются все компоненты, относящиеся к панели инструментов

Наименование	Тип	Описание
Panel2	TPanel *	Вспомогательный элемент, на котором размещается поле для формирования СПМ
PopupMenu1	TPopupMenu *	Контекстное меню, связанное с позицией
PopupMenu2	TPopupMenu *	Контекстное меню, связанное с переходом
PopupMenu3	TPopupMenu *	Контекстное меню, связанное с узлом дуги
RadioGroup1	TRadioGroup *	Переключатель «Закон распределения» (связан с TabSheet1)
RadioGroup2	TRadioGroup *	Переключатель «Переход в позицию» для выбора номера позиции, для выхода в которую будет редактироваться условие срабатывания (связан с TabSheet2)
RadioGroup3	TRadioGroup *	Переключатель «Завершение моделирования» для выбора способа завершения имитационного моделирования (достижение конечной вершины хотя бы одним маркером или всеми маркерами).
RadioGroup4	TRadioGroup *	Переключатель «Тип позиции» для выбора типа редактируемой позиции (начальная, промежуточная или конечная) (связан с TabSheet1)
SaveDialog1	TSaveDialog *	Компонент для отображения диалогового окна «Сохранить файл»
ScrollBox1	TScrollBox *	Вспомогательный компонент, на котором размещается изображение Image1 для отрисовки структуры сети Петри-Маркова; обеспечивает скроллинг изображения в случаях, когда его размер превышает размеры экрана

Наименование	Тип	Описание
SpeedButton1	TSpeedButton *	Кнопка «Добавить позицию» (связана с TabSheet3)
SpeedButton10	TSpeedButton *	Кнопка «Загрузить структуру и параметры сети» (связана с TabSheet3)
SpeedButton11	TSpeedButton *	Кнопка «Запустить моделирование» (связана с TabSheet5)
SpeedButton13	TSpeedButton *	Кнопка «Статистика моделирования» (связана с TabSheet5)
SpeedButton2	TSpeedButton *	Кнопка «Добавить переход» (связана с TabSheet3)
SpeedButton3	TSpeedButton *	Кнопка «Добавить дугу» (связана с TabSheet3)
SpeedButton4	TSpeedButton *	Кнопка «Загрузить структуру сети» (связана с TabSheet3)
SpeedButton5	TSpeedButton *	Кнопка «Сохранить структуру сети» (связана с TabSheet3)
SpeedButton6	TSpeedButton *	Кнопка «Добавить конъюнкцию» для добавления дополнительной конъюнкции в таблицу StringGrid2 (связана с TabSheet2)
SpeedButton7	TSpeedButton *	Кнопка «Удалить конъюнкцию» для удаления текущей выбранной строки из таблицы StringGrid2 (связана с TabSheet2)
SpeedButton8	TSpeedButton *	Кнопка «Ввод» для подтверждения внесённых изменений (связана с TabSheet1)
SpeedButton9	TSpeedButton *	Кнопка «Сохранить структуру и параметры сети» (связана с TabSheet4)
StringGrid1	TStringGrid *	Таблица «Вероятности выхода в переход» (связана с TabSheet1)

Наименование	Тип	Описание
StringGrid2	TStringGrid *	Таблица «Условие срабатывания» для отображения и редактирования условия срабатывания перехода в ранее выбранную с помощью RadioGroup2 позицию (связана с TabSheet2)
TabSheet1	TTabSheet *	Страница «Позиция», содержащая компоненты, отображающие свойства позиции (связана с PageControl1)
TabSheet2	TTabSheet *	Страница «Переход», содержащая компоненты, отображающие свойства перехода (связана с PageControl1)
TabSheet3	TTabSheet *	Страница «Структура сети», содержащая компоненты по редактированию структуры сети (связана с PageControl2)
TabSheet4	TTabSheet *	Страница «Параметры эл-тов», содержащая компоненты по сохранению параметров элементов сети в файл (связана с PageControl2)
TabSheet4	TTabSheet *	Страница «Моделирование», содержащая компоненты для работы в режиме имитационного моделирования (связана с PageControl2)

Наряду с визуальными и невизуальными компонентами класс TForm1 содержит ряд дополнительных членов, предназначенных для реализации логики работы программы. Перечень и описание этих элементов приведены в табл. 4.2.

Таблица 4.2

Перечень и описание элементов,
предназначенных для реализации логики работы программы

Наименование	**Тип**	**Описание**
POSShape	TShape* []	Массив для хранения указателей на визуальные формы типа «Позиция»
POS	MyPosition []	Массив для хранения семантики объектов типа «Позиция»
NPos	Int	Текущее количество позиций в сети Петри-Маркова
TRAShape	TShape* []	Массив для хранения указателей на визуальные формы типа «Переход»
TRA	MyTransition []	Массив для хранения семантики объектов типа «Переход»
NTra	Int	Текущее количество переходов в сети Петри-Маркова
NODShape	TShape* []	Массив для хранения узлов дуг
NNod	Int	Текущее количество узлов дуг
ARC	MyArc []	Массив для хранения информации о дугах, соединяющих переход/позицию/узел; кодирует структуру сети Петри-Маркова
NArc	Int	Текущее количество дуг в сети Петри-Маркова
SelObjInd	Int	Идентификационный номер объекта (элемента сети), выделенного с помощью указателя мыши
SelObjType	Int	Тип выделенного объекта (1 — позиция, 2 — переход, 3 — узел дуги)
AddPos	Bool	Логический признак перехода в режим добавления позиции в сеть

Наименование	Тип	Описание
AddTrans	Bool	Логический признак перехода в режим добавления перехода в сеть
AddArc	Bool	Логический признак перехода в режим добавления дуги в сеть
BasePointType	Int	Тип объекта, который является начальной точкой комплексной дуги, содержащей несколько узлов
FirstPointType	Int	Тип объекта, который считается начальной точкой текущего отрезка дуги
FirstPointInd	Int	Идентификационный номер объекта, который считается начальной точкой текущего отрезка дуги
MouseX	Int	Горизонтальная координата указателя мыши в точке нажатия/отпускания
MouseY	Int	Вертикальная координата указателя мыши в точке нажатия/отпускания
PopupInd	Int	Идентификационный номер объекта, на котором с помощью правой кнопки мыши открыли контекстное меню

4.1.9. Описание отдельных функций

Ниже перечислены основные функции класса TForm1 с кратким описанием назначения каждой из них.

POSMouseDown — обработчик события нажатия кнопкой мыши на объект типа «Позиция»; обрабатывает нажатия на объект в режимах: а) перемещения позиции по экрану; б) соединения дуги с позицией; в) выбора позиции, для которой будут отображаться свойства; г) отображения контекстного меню.

POSMouseUp — обработчик события отпускания кнопки мыши для объекта типа «Позиция».

POSMouseMove — обработчик движения мыши над поверхностью объекта типа «Позиция»; предназначен для реализации перемещения позиций по экрану.

Аналогичные тройки функций определены для остальных объектов:

— TRAMouseDown, TRAMouseUp, TRAMouseMove — для объектов типа «Переход»;

— NODMouseDown, NODMouseUp, NODMouseMove — для объектов типа «Узел дуги».

Image1MouseUp — обработчик события нажатия кнопкой мыши на свободное место в области отображения сети (на объект Image1). Предназначен для реализации операций добавления позиции, перехода или узла дуги.

FormCreate — однократно вызывается при запуске программы; настраивает размеры изображения для работы с создаваемой сетью Петри-Маркова; инициализирует заголовки таблиц и видимость отдельных страниц.

N1Click, N2Click, N3Click, N4Click, N5Click, N6Click — обработчики соответствующих пунктов контекстного меню.

Edit1KeyPress — обработчик нажатия клавиши на клавиатуре (связан с парой редакторов Edi1 и Edit2); предназначен для отслеживания нажатия клавиш «,» и «.» с целью замены их на текущий установленный в операционной системе разделитель целой и дробной частей в вещественных числах.

Обработчики нажатий на соответствующие кнопки:

SpeedButton1Click — добавить позицию;

SpeedButton2Click — добавить переход;

SpeedButton3Click — добавить дугу;

SpeedButton4Click — загрузить структуру (и параметры) сети;

SpeedButton5Click — сохранить структуру сети;

SpeedButton6Click — добавить конъюнкцию;

SpeedButton7Click — удалить конъюнкцию;

SpeedButton8Click — подтверждение ввода данных

Button2Click — переход в режим редактирования структуры сети;

Button3Click — переход в режим редактирования параметров сети.

RadioGroup2Click — выбор перехода в позицию при отображении свойств перехода;

StringGrid2SetEditText — сохранение изменений, внесённых в таблицу условий срабатывания перехода StringGrid2;

PrintLogic(AnsiString L) — печать в таблицу StringGrid2 множества конъюнкций, определяющих условие срабатывания выбранного перехода;

StringGrid1Click — обработчик нажатия на строку таблицы StringGrid1; определяет конкретный выход из позиции в переход, для которого будут отображаться сведения в полях «Закон распределения» (RadioGroup1), «Мат. ожидание» (Edit1) и «СКО» (Edit2).

PrintNet — функция отрисовки всей совокупности дуг в сети Петри-Маркова.

DeleteArc(int ind) — функция удаления дуги с идентификационным номером ind; после удаления оставшиеся дуги перенумеровываются, ссылки на них в соответствующих массивах корректируются.

DeletePos(int ind) — функция удаления позиции с идентификационным номером ind; связанные с удаляемой позицией дуги также удаляются; оставшиеся позиции и дуги перенумеровываются, ссылки на них в соответствующих массивах корректируются.

DeleteTrans(int ind) — функция удаления перехода с идентификационным номером ind; связанные с удаляемым переходом дуги также удаляются, оставшиеся переходы и позиции перенумеровываются; ссылки в соответствующих массивах корректируются.

DeleteNode(int ind) — функция удаления узла дуги с идентификационным номером ind; связанные с удаляемым узлом дуги также удаляются; оставшиеся узлы и дуги перенумеровываются; ссылки на них в соответствующих массивах корректируются.

Полный листинг разработанного программного обеспечения приведён в приложении.

4.1.10. Описание формата файла сети Петри-Маркова

Для обеспечения возможности сохранения на диске сети Петри-Маркова и последующего её считывания в память программы предложен следующий формат файла, отражённые в табл. 4.3.

Таблица 4.3

Структура файла для хранения СПМ

Наименование поля	**Описание**	**Тип**
Строка 1 — Общее описание состава СПМ		
<Кол-во позиций>	Общее число позиций в СПМ	целое
<Кол-во перехо-дов>	Общее число переходов в СПМ	целое
<Кол-во узлов дуг>	Общее число узлов дуг в СПМ	целое
<Кол-во дуг>	Общее число дуг в СПМ	целое
<Свойства>	Признак наличия в файле свойств сети	1 – есть, 0 – нет
Строка 2 —Описание позиций		
<X-позиции>	Горизонтальная координата позиции	целое
<Y-позиции>	Вертикальная координата позиции	целое
Строка 3 — Описание переходов		
<X-перехода>	Горизонтальная координата перехода	целое
<Y-перехода>	Вертикальная координата перехода	целое
Строка 4 — Описание узлов дуг		
<X-узла>	Горизонтальная координата узла	целое
<Y-узла>	Вертикальная координата узла	целое

Строка 5 — Описание структуры связей дуг сети		
<Тип0>	Тип объекта, который являлся начальной точкой для сложной дуги, состоящей из нескольких отрезков	1 – позиция, 2 – переход
<Индекс1>	Идентификационный номер объекта, являющегося начальным для текущего отрезка дуги	целое
<Тип1>	Тип объекта, являющегося начальным для текущего отрезка дуги	1 – позиция, 2 – переход, 3 – узел дуги
<Индекс2>	Идентификационный номер объекта, являющегося конечным для текущего отрезка дуги	целое
<Тип2>	Тип объекта, являющегося конечным для текущего отрезка дуги	1 – позиция, 2 – переход, 3 – узел дуги
Строка 6 — Описание свойств сохраняемых позиций		
<Индекс>	Идентификационный номер сохраняемой позиции	целое
<ИсхДуг>	Количество исходящих дуг из данной позиции	целое
<Действие>	Индекс функции из внешней библиотеки	целое
<ИндексПодсети>	Индекс подсети, к которой относится описываемая позиция	целое
<ТипПозиции>	Тип описываемой позции (0 — промежуточная, 1 — начальная, 2 — конечная)	целое
Строка 7 — Текстовое описание позиции		

<Описание>	Текстовое описание позиции	строка
Строка 8 — Описание выходов из позиции в тот или иной переход		
<ИндексДуги >	Идентификационный номер дуги, выходящей из текущей позиции в переход	целое
<Вероятность>	Вероятность выхода по этой дуге	вещественное
<Тип закона>	Тип закона распределения времени пребывания фишки в позиции при выборе выхода по этой дуге	целое
<Матожидание>	Математическое ожидание закона распределения	вещественное
<СКО>	Среднеквадратическое отклонение закона распределения	вещественное
Строка 9 — Описание свойств сохраняемых переходов		
<Индекс>	Идентификационный номер сохраняемого перехода	целое
<ИсхДуг>	Количество исходящих дуг из данного перехода	целое
<ВхДуг>	Количество входящих дуг в данный переход	целое
Строка 10 — Описание логики срабатывания перехода		
<ИндексДуги>	идентификационный номер дуги, выходящей из текущего перехода в позицию	целое
<СДНФ>	логическая функция в совершенной дизъюнктивной нормальной форме, определяющая условие срабатывания перехода по данному выходу	строка
Строка 11 — Описание входящих в переход дуг		

<ИндексДуги>	индекс входящей в переход дуги	целое

Рассмотрим в качестве примера простую сеть Петри-Маркова, состоящую из 4 позиций и 2 переходов. Позиции 0 и 1 относятся к подсети 1, а позиции 2 и 3 – к подсети 2. Начальными позициями в подсетях являются позиция 0 (подсеть 1) и позиция 2 (подсеть 2). Позиция 3 является конечной для подсети 2, позиция 1 является промежуточной в подсети 1 (в этой подсети нет конечной позиции – из позиции 1 существует петля обратной связи в позицию 0 через примитивный переход 1).

Подсети между собой связаны непримитивным переходом 0.

Структура подобной сети Петри-Маркова в представленном формате файла будет иметь вид:

```
4 2 0 6 1
135 30
138 209
288 26
289 208
207 117
67 119
1 0 1 0 2
2 0 2 1 1
1 1 1 1 2
2 1 2 0 1
1 2 1 0 2
2 0 2 3 1
0 1 0 1 1
Начальная вершина 1-й подсети
0 1.000000 0 5.000000 2.000000
1 1 0 1 0
Промежуточная позиция 1-й подсети
1 1.000000 0 4.000000 1.000000
2 1 0 2 1
Начальная позиция 2-й подсети
0 1.000000 0 3.000000 3.000000
3 1 0 2 2
Конечная позиция 2-й подсети
2 1.000000 0 2.000000 0.000000
```

```
0 2 2
1 P0P2
3 P0P2
0
2
1 1 1
0 P1
1
```

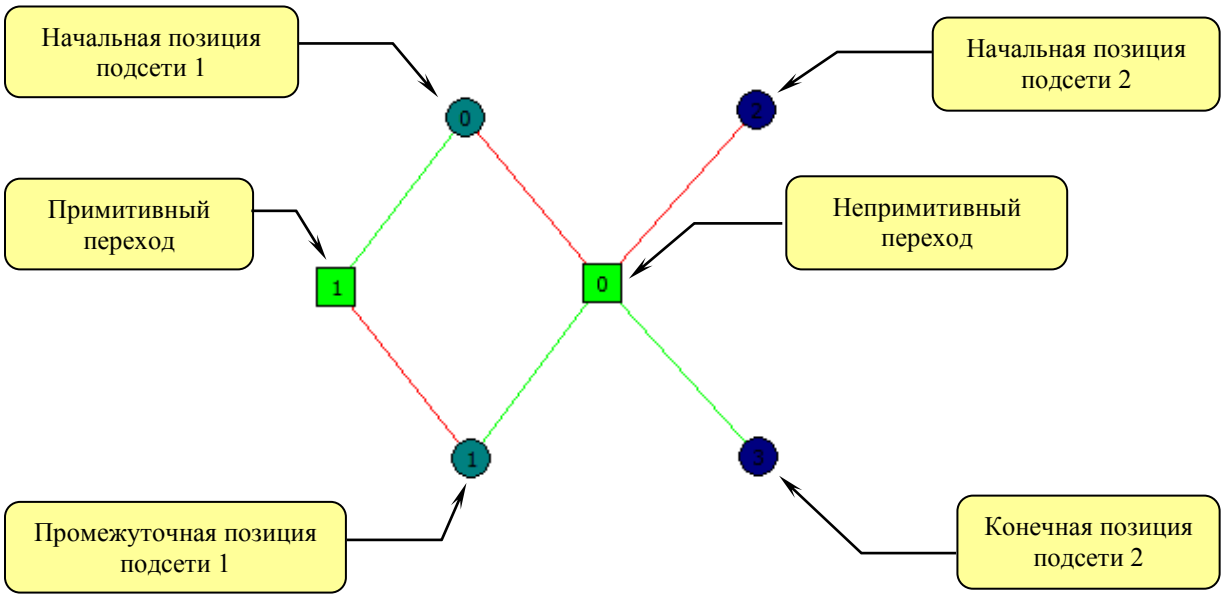


Рис. 4.16. Пример СПМ для определения формата файла pmn

Данный файл может быть при необходимости скорректирован в любом текстовом редакторе, либо загружен в разработанную программу и скорректирован через стандартный интерфейс пользователя.

Этот модуль автоматически регулирует все технологические параметры, касающиеся температуры, нанесения клея, усилия полотна, давления валиков и пара для всех объединенных в задании профилей бумаги и валов. При этом в регулировочных характеристиках учитываются такие переменные величины, как изменение скорости и положение консоли для обвивания.

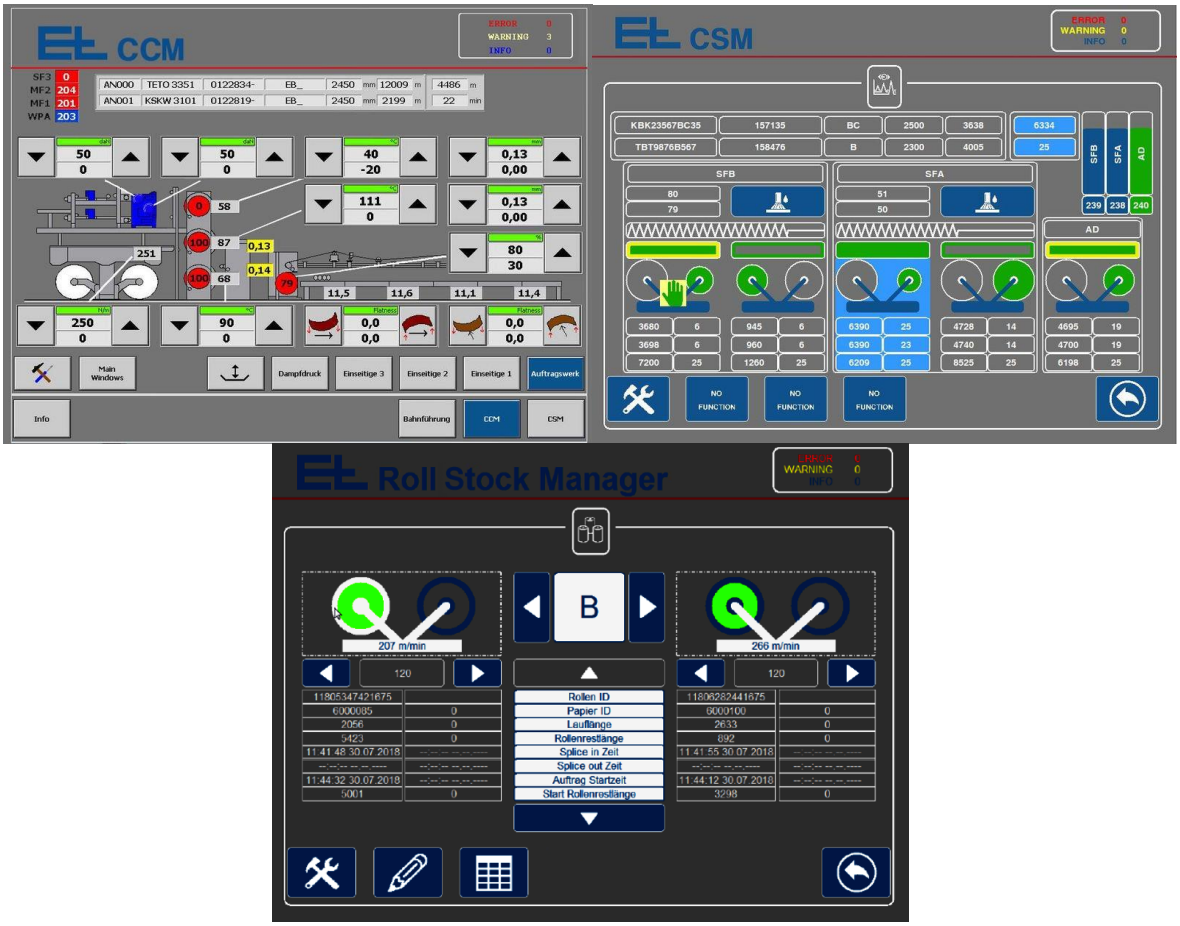


Рис. 4.17. Структура программной платформы

В геометрии регулирования учитываются как заданные в системе сорта бумаги, так и покрытия, препринт и водоустойчивый клей. За счет этого оптимизируется качество производства без расслоения или коробления, а также расход энергии для пара и расход клея. Управляет производительностью автоматически регулирует через интерфейс к сухому участку и системе обработки заказов все параметры, относящиеся к скорости сращивания, запасу на мостах и выбраковыванию мест сращивания для заказов, представленных в системе менеджмента. Все заложенные в системе сорта бумаги снабжаются собственными характеристиками (или их группами) и выбираются автоматически. Это позволяет гармонизировать и автоматически оптимально согласовывать скорости машин между мокрым и сухим участками. Обеспечивается производство именно требуемого количества изделия, без меняющегося пере- и недопроизводства. Повышается производительность, минимизируются отходы этапа запуска, улучшается менеджмент остатков на рулонах. правления производительностью

автоматически регулирует через интерфейс к сухому участку и системе обработки заказов все параметры, относящиеся к скорости сращивания, запасу на мостах и выбраковыванию мест сращивания для заказов, представленных в системе менеджмента. Все заложенные в системе сорта бумаги снабжаются собственными характеристиками (или их группами) и выбираются автоматически.

Таблица 4.4

Оценка быстродействия программы

Описание	**Значение**
Исходное состояние	0
Первый круг оценки быстродействия	0,0233
Второй круг оценки быстродействия	0,0521
Третий круг оценки быстродействия	0,0789
Четвертый круг оценки быстродействия	0.1001
Пятый круг оценки быстродействия	0,1019
Шестой круг оценки быстродействия	0,1198
Седьмой круг оценки быстродействия	0.1198
Восьмой круг оценки быстродействия	0.1198

В результате получаем, что после пятого круга значение оценки не меняется. Значение быстродействия программы составляет 12%. Это позволяет гармонизировать и автоматически оптимально согласовывать скорости машин между мокрым и сухим участками. Обеспечивается производство именно требуемого количества изделия, без меняющегося пере- и недопроизводства. Повышается производительность, минимизируются отходы этапа запуска, улучшается менеджмент остатков на рулонах.

4.2. Выводы

1) Разработан метод прямого расчета временных и вероятностных характеристик комплексного блуждания между состояниями и возврата в состояние с использованием только операций с числовыми матицами: стохастической, математических ожиданий и дисперсий, характеризующих элементы исходной полумарковской матрицы.

2) Предложен метод имитационного моделирования на основе дискретно-событийного подхода, позволяющий рассчитать временные и вероятностные характеристики блужданий по полумарковским процессам.

3) Разработано программное обеспечение, реализующее предложенный метод, и позволяющее оценить численно вероятностные и временные характеристики блужданий по полумарковским процессам, достижимость отдельных позиций и ряд других свойств.

ЗАКЛЮЧЕНИЕ

1) На основании анализа существующих цифровых систем управления сложными технологическими объектами и их программного обеспечения сделан вывод о том, что перспективной схемой их организации является иерархическая структура, в которой работа по управлению отдельными контурами должна быть синхронизирована, что требует на этапе проектирования программного обеспечения оценивать производительность ЭВМ Фон Неймановского типа.

2) На основании анализа типовой циклограммы управления, включающей ввод данных от сенсоров, расчет управляющего воздействия и вывод данных на исполнительные приводы технологической установки сформулированы основные проблемы, возникающие при переходе на цифровые технологии, а именно, погрешности оцифровки сигналов сенсоров, шум полинга, перекос данных, задержка сигнала обратной связи.

3) Определено понятие полумарковского процесса, как базовой теории для оценки временных характеристик управляющих ЭВМ Фон Неймановского типа, и показано, что полумарковские процессы являются математическим подобием алгоритмов управления отдельными узлами и блоками, а состояния полумарковских процессов связаны с интерпретацией соответствующих операторов алгоритма.

4) Исследованы основные свойства полумарковских процессов, вытекающие из свойств алгоритмов управления узлами и блоками, показано, что полумарковские процессы являются возвратными и эргодическими.

5) Получено выражение для оценки плотности распределения времени достижения одного из операторов из другого оператора при интерпретации алгоритмов управления, а также плотности распределения времени возврата в один из операторов,

6) Разработана методика прямого расчета математического ожидания времени достижения одного из операторов из другого оператора и времени возврата в оператор по стохастический матрице и матрице математических ожиданий.

7) Сформулировано понятие *M*-параллельного полумарковского процесса и показано, как основного подхода к моделированию иерархических цифровых систем управления, и функциональных состояний.

8) Получены зависимости для формирования множества функциональных состояний, матрицы смежности и полумарковской матрицы как декартовых произведений соответствующих неупорядоченных и упорядоченных в матрицы множеств.

9) Разработаны модели циклической и квазистохастической дисциплины опроса контроллеров функционально-логического уровня, получены зависимости для оценки времени между транзакциями.

10) Квази-стохастическая дисципллина диспетчеризации применена при разработке системного программного обеспечения ЭВМ тактического уровня, что позволило ускорить отклик получения данных.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Абдуллаев Д.А., Амирсаидов У.Б. Моделирование локальных вычислительных сетей с учетом вероятностно-временных характеристик //Автоматика и телемеханика. - 1994. - N 3. - С. 151 - 160.

2. Акименко Т.А. Компьютерное моделирование процесса нагрева поверхности мишени лазерным лучом. Известия ТулГУ. Серия Технические науки. Вып. 9. Тула: Изд-во ТулГУ, 2018.- с.585-591

3. Акименко Т.А., Аршакян А.А., Ларкин Е.В. Управление информационными процессами в робототехнических комплексах специального назначения. - Тула: Изд-во ТулГУ, 2012. - 150 с.

4. Акименко Т.А., Аршакян А.А., Рудианов Н.А. Управление группами роботизированных платформ // Известия ТулГУ. Сер. Технические науки. - Вып. 8. - 2015. - С. 200 - 208.

5. Акименко Т.А., Рыбалкина Ю.С., Филиппова Е.В. Оптическая модуляция излучения в тепловизионной системе. Известия ТулГУ. Серия Технические науки. Вып. 4. Тула: Изд-во ТулГУ, 2018.- с.472-478

6. Акименко Т.А., Тароватов А.А., Филиппова Е.В. Компьютерное моделирование движения мобильного колесного робота с учетом рельефа местности // Известия ТулГУ. Технические науки. 2013. - Вып. 7. Ч. 2. - Тула: Изд-во ТулГУ. - С. 123 - 130.

7. Акименко Т.А., Филиппова Е.В. Исследование статических характеристик и пространственной динамики тепловизионной системы наблюдения. Известия ТулГУ. Серия Технические науки. Вып. 9. Тула: Изд-во ТулГУ, 2018.- с.497-500

8. Андреев В.П. Разработка новых принципов построения информационно-измерительных систем технического зрения мобильных роботов: Дис. … доктора техн. наук. 05.11.16 (Москва, 2011 г.). - 300 с.

9. Анисимов В.В. 0 предельном поведении полумарковского процесса с расщепляющимся множеством состояний; Докл. АН СССР. - 1972, № 4, С. 777 - 779.

10. Анисимов В.В. Предельные теоремы для полумарковских процессов со счетным множеством состояний; Докл. АН СССР. - 1970, № 3, С. 503 - 505.

11. Анисимов В.В. Случайные процессы с дискретной компонентой. - Киев Вища школа, 1988. - 184 с.

12. Антонов М.А., Гришин К.А. Модель мобильного робота как ординарный полумарковский процесс. Известия Тульского государственного университета. Технические науки. 2018. № 2. С. 95-100.

13. Анучин О.Н. Инерциальные системы ориентации и навигации для морских подвижных объектов. - С.Пб.: ЦНИИ «Электроприбор», 2003. - 387 с.

14. Арсенишвили Г.Л. Некоторые вопросы из теории полу марковских процессов r-го порядка // Вопросы разработки и внедрения средств вычислительной техники. - Тбилиси: 1970, С. 128—132

15. Арсенишвили Г.Л., Ежов И.И. 0 распределении времени пребывания в заданной области полумарковским процессом r-го порядка. // Труды Института прикладной математики Тбилисского университета. - 1969. - № 2, - С. 151,-157

16. Арсенишвили Г.Л., Ежов И.И. Об одной предельной теореме для полумарковских процессов r-го порядка // Сообщ. АН ГрузССР. - 1969. - № 1. - С. 25-28.

17. Артамонов Г.Т., Тюрин В.Д. Анализ информационно-управляющих систем со случайным интервалом квантования сигнала во времени. М.: Энергия, 1977.

18. Артемьев В. М., Ивановский А. В. Вероятностный анализ импульсных систем при случайном законе распределения интервалов квантования // Автоматика и вычислительная техника. Минск: Высшая школа, 1981. Вып. 10. С. 2-12.

19. Артемьев В. М., Ивановский А. В. Дискретные системы управления со случайным периодом квантования. М.: Энергоатомиздат, 1986.

20. Аршакян А.А., Ларкин Е.В. Параметры потока транзакций, генерируемых по принципу поллинга // Известия ТулГУ . Серия Технические науки. - 2016. - Вып. 2. - Тула: Изд-во ТулГУ, - С..40 - 48.

21. Аршакян А.А., Ларкин Е.В., Рудианов Н.А. Интерактивный генератор команд // Известия ТулГУ. Сер. Технические науки. - Вып. 7. Ч. 2. - 2015. - С. 251 - 262.

22. Аршакян А.А., Ларкин Е.В., Рудианов Н.А. Оценка установившихся режимов функционирования групповых систем мониторинга // Известия ТулГУ. Сер. Технические науки. - Вып. 3. - 2015. - С. 115 - 123.

23. Байцер Б. Микроанализ производительности вычислительных систем. - М.: Радио и связь, 1983. - 360 с.

24. Баклан В. В., Эргодическая теорема для марковских процессов с дискретным вмешательством случая // Украинский математический журнал, - 1967. - № 5. -С. 123 - 12.

25. Барский А.Б. Параллельные процессы в вычислительных системах: Планирование и организация. - М.: Радио и связь, 1990. - 255 с.

26. Басараб М.А. Алгоритмы решения задачи быстрого поиска пути на географических картах. / Басараб М.А., Домрачева А.Б., Купляков В.М. – М.:МГТУ.Инженерный журнал: наука и инновации, 2013. – 21 с.

27. Беллман Р., Кук К. Л. Дифференциально-разностные уравнения / Пер. с англ. М., 1967. 548 с.

28. Богуславский Л.Б., Ляхов А.И. Методы оценки производительности многопроцессорных систем. - М.: Наука, 1992. - 213 с.

29. Бояринов Ю.Г. Анализ систем и процессов на основе нечетких полумарковских моделей // Вестник Саратовского государственного технического университета, Вып. 4, Т. 4, - 2011. С. 207 – 212.

30. Бояринов Ю.Г., Борисов В.В. Анализ систем и процессов на основе нечетких полумарковских моделей // Информационные технологии. - 2011. - № 11. С. 31 – 36.

31. Бояринов Ю.Г., Глушко С.И. Полумарковские модели систем с нечеткими параметрами // Программные продукты и системы. - 2012. - № 2. С. 146 – 149.

32. Броди С.М. Исследование систем массового обслуживания с помощью полумарковских процессов // Кибернетика. - 1965. - № 6. - С. 55—58.

33. Броди С.М., Шпак В.Д. Применение ассоциированных полумарковских процессов в анализе надежности систем // Кибернетика. - 1970. - № 5. - С. 90 - 96.

34. Вухалев В. А. Анализ точности динамических систем со случайной структурой, описываемой условной марковской цепью //Изв. АН СССР. Техн. кибернетика. 1976. № 2. С. 179-186.

35. Валах В.Я. Королюк В.С, Стохастические автоматы со случайным временем реакции и их функционирование в случайных средах // Автоматы, гибридные и управляющие машины // М.: Наука, 1972, С. 38 - 45.

36. Верден А. Р. О статистическом расчете линейных импульсных систем со случайными интервалами повторения импульсов//Тр. I конг. ИФАК. Т. 2. М.: Изд-во АН СССР, 1961. С. 299-412.

37. Веселерский В. А. Цифровые автоматические системы. М.: Наука, 1976.

38. Воротников С.А. Информационные устройства робототехнических систем: Учеб. пособие. / С.А. Воротников. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. — 384 с.; ил.

39. Гане В. А., Куклев Е. А., Степанов В. Л. Системы управления при скачкообразных воздействиях. Минск: Наука и техника, 1985.

40. Герцбах И.Б. Оптимальное управление полумарковским процессом при наличии ограничений на вероятности состояний // Кибернетика. - 1970. - № 5. - С. 56 - 61.

41. Гришин К.А. Описание типов и характеристик программаторов для программируемых логических интегральных схем // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2018. Вып. 9. С. 607-610.

42. Гришин К.А. Определение дискретного М-параллельного полумарковского процесса // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2021. Вып. 2. С. 109-114.;

43. Гришин К.А. Структура 2-параллельного полумарковского процесса с альтернативными маршрутами // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2020. Вып. 9. С. 201-206.;

44. Гришин К.А. Устойчивость систем с задержкой, основанная на втором методе Ляпунова // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2020. Вып. 9. С. 47-50.;

45. Гришин К.А. Аппроксимация композиции плоскостей законом распределения // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2017. Вып. 9: в 2 ч. Ч. 1. С. 67-70.

46. Гришин К.А. Интерфейсы передачи данных и сопряжение устройств // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2018. Вып. 9. С. 604-606.

47. Гришин К.А. Метод последовательных упрощений полумарковского процесса // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2021. Вып. 2. С. 69-73.

48. Гришин К.А. Методика определения временных интервалов блуждания по эргодическим полумарковским процессам // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2016. Вып. 9. С. 77-81.

49. Гришин К.А. Модель квази-стахонической дисциплины диспетчеризации // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2017. Вып. 2. С. 32-36.

50. Гришин К.А. Модель роботизированной платформы как ординарный полумарковский процесс // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2016. Вып. 9. С. 70-76

51. Гришин К.А. Оценка степени "марковости" процессов с помощью регрессионного, корреляционного критерий и критерия Пирсона. Известия Тульского государственного университета. Технические науки. 2018. № 2. С. 56-59.

52. Гришин К.А. Петри-марковское моделирование типовых структур избыточных систем // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2017. Вып. 9: в 2 ч. Ч. 1. С. 70-74.

53. Гришин К.А. Полумарковская модель циклограммы управления роботизированной платформой // Известия Тульского государственного университета. Технические науки. Тула: Изд-во ТулГУ, 2017. Вып. 2. С. 37-40.

54. Грязев М.В., Стась Г.В., Кусакина Е.В. Методические положения оценки вероятности крупных аварий в угольной промышленности. Известия Тульского государственного университета. Науки о Земле. 2018. № 2. С. 127-136.

55. Губенко Л.Г., Штатланд Э.С, Об управляемых полумарковских процессах // Кибернетика. - 1972. - № 2. - С. 26 - 29.

56. Давыдов О.И., Пряничников В.Е. Управление движением мобильного робота по данным ультразвуковых сенсоров // Информационно-измерительные и управляющие системы. - 2015. - Т. 13. - № 7. - С. 57 – 67.

57. Данилевский В.А. Картонная и бумажная тара. – М.: Лесная промышленность, 1979.- 216 с.

58. Девятериков Е.А., Михайлов Б.Б. Управление движением мобильного робота с использованием данных визуального одометра // Экстремальная робототехника. - 2013. - Т. 1. - № 1. - С. 303 – 313.

59. Деч Г. Руководство к практическому применению преобразования Лапласа и Z-преобразования. - М.: Наука, 1971. - 288 с.

60. Джексон Р.Г. Новейшие датчики. - М.: Техносфера, 2008. - 400 с.

61. Дитрих Я. Проектирование и конструирование. Системный подход. - М.: Мир, 1981. - 454 с.

62. Добринский Е.П., Бушуев Д.А., Рубанов В.Г. Методы управления группой мобильных роботов в ограниченном пространстве // Математические методы в технике и технологиях. - 2014. - № 3. - С. 16 – 20.

63. Добрыдень В.А., Оптимальное наблюдение полумарковского процесса // Изв. АН СССР. Техн. кибернетика. - 1971. - № 4. - С. 47 - 49.

64. Ежов И.И. Цепи Маркова с дискретным вмешательством случая, образующим полумарковский процесс // Украинский математический журнал, - 1966. - № 1. - С. 48 - 65.

65. Ежов И.И. Эргодическая теорема для вероятностных процессов с полумарковским вмешательством случая // Украинский математический журнал. - 1968. № 3 - С. 384 - 388.

66. Ежов И.И., Королюк В.С, Полумарковские процессы и их приложения // Кибернетика. - 1967. - № 5. С. 58 - 65.

67. Ерофеенко В.Г., Козловская И.С. Основы математического моделирования. - Минск: БГУ, 2002. - 195 с.

68. Есиков Д.О. Методика выбора метода решения задач обеспечения устойчивости функционирования распределённых информационных систем. Электронные информационные системы. 2018. № 1 (16). С. 65-78.

69. Есиков О.В., Есиков Д.О., Акиншина Н.Ю. Общие принципы выбора параметров многоагентных алгоритмов стохастического поиска для решения отдельных задач дискретной оптимизации. Приборы и системы. Управление, контроль, диагностика. 2018. № 5. С. 28-37.

70. Жучков П.А., Саунин В.И. Тепловой и гидравлический режимы работы бумагоделательных и картоноделательных машин. – М.: Лесная промышленность, 1972. – 408 с.

71. Зарубин В.С. Математическое моделирование в технике. - М.: Изд-во МГТУ им. Н.Э.Баумана, 2001. - 496 с.

72. Зеленцов Б.П., Максимов В.П., Шувалов В.П. Модель функционирования линии связи в условиях недостоверного контроля технического состояния // Вестник СибГУТИ. - 2015. - № 3 (31), С. 35 – 43.

73. Иванов Н.Н. Математическое прогнозирование надежного выполнения наборов задач с симметричными распределениями времени выполнения // Открытое образование. - 2011. - № 2-2, С. 52 – 55.

74. Иванов Ю.В. Гироскопические системы измерения вертикальной качки. - Тула: ТулГУ, 2004. - 184 с.

75. Ивутин А.Н., Ларкин Е.В. Временные и вероятностные характеристики транзакций в цифровых системах управления // Известия ТулГУ. Сер. Технические науки. Вып. 1. - Тула: Изд-во ТулГУ, 2013. - С. 252 - 258.

76. Ивутин А.Н., Ларкин Е.В. Обобщенная полумарковская модель алгоритма управления цифровыми устройствами // Известия ТулГУ. Сер. Технические науки. Вып. 1. - Тула: Изд-во ТулГУ, 2013. - С. 221 - 228.

77. Ивутин А.Н., Ларкин Е.В. Прогнозирование времени выполнения алгоритма // Известия ТулГУ. Технические науки. Вып. 3. - Тула: Изд-во ТулГУ, 2013. - С. 301 - 315.

78. Ивутин А.Н., Ларкин Е.В. Системы контроля и диагностики программного обеспечения // Известия ТулГУ. Сер. Технические науки. 2014. - Вып. 9. Ч. 2. - Тула: Изд-во ТулГУ, - С. 35 - 41.

79. Игнатущенко В.В., Клушин Ю.С. Прогнозирование выполнения сложных программных комплексов на параллельных компьютерах: Прямое сттохастическое моделирование // Автоматика и телемеханика. - 1994. - N 11. - С. 142 - 157.

80. Игнатьев В.М., Ларкин Е.В. Анализ производительности ЭВМ: Учебное пособие. - Тула: ТГТУ, 1994. - 104 с.

81. Игнатьев В.М., Ларкин Е.В. Временные характеристики алгоритмов в системах с прерываниями // Проектирование ЭВМ. Рязань, РГРТА, 1994, - С. 29-40.

82. Игнатьев В.М., Ларкин Е.В. Сети Петри-Маркова. - Тула: ТулГУ, 1997. - 163 с.

83. Информационная система поддержки принятия решений при мониторинге акустической безопасности профессиональной деятельности авиационных специалистов. Богомолов А.В., Драган С.П., Зинкин В.Н., Загребина С.А., Свиридюк Г.А., Ларкин Е.В. Международная конференция по мягким вычислениям и измерениям. 2018. Т. 2. С. 202-205

84. Информационно-измерительная техника и технологии / В.И.Калашников, С.В. Нефедов, А.Б. Путилин и др. Под ред. Г.Г. Раннева. - М.: Высшая школа, 2002. - 454 с.

85. Казаков И.Е. Статистическая динамика систем с переменной структурой. М.: Наука, 1977.

86. Казаков И.Е. Статистическая теория систем управления в пространстве состояний. М.: Наука, 1975.

87. Каляев И.А. Децентрализованные системы компьютерного управления /И.А. Каляев, Э.В. Мельник - Ростов н/Д: Издательство ЮНЦ РАН, 2011.-196 с.

88. Каляев И.А., Мельни Э.В. Децентрализованные системы компьютерного управления. - Ростов н/Д: Издательство ЮНЦ РАН, 2011.-- 196 с.

89. Каштанов В.А. Стратегия технического обслуживания на основе полумарковских процессов с конечным множеством состояний // Надежность и качество сложных систем. - 2013. - № 1. - С. 41 – 46.

90. Клёнов С. Г. Полумарковские модели, алгоритмы и комплекс программ оптимизации технического обслуживания сложных технологических систем с учётом старения оборудования: Дис. ... канд. техн. наук: 05.13.18 : М.: 2003. - 153 с. РГБ ОД, 61:04-5/1074.

91. Клинцов Г., Ларкин Е.В. Основные свойства вычислительных алгоритмов, выполняемых на реальных ЭВМ // Известия ТулГУ . Серия Технические науки. - 2016. - Вып. 1. - Тула: Изд-во ТулГУ, - С. 61 - 68.

92. Ковалева Л.М. О времени пребывания двух независимых полумарковских процессов в заданном состоянии // Украинский математический журнал. - 1968, - № 6. - С. 837 - 841.

93. Кокс Л.Р., Смит У.Л. Теория очередей. - М.: Мир, 1966. - 218 с.

94. Кононов Б.В., Ландау Г.Е., Погребов Е.М.,Гофрированный картон.М.: Лесная промышленность, 1971.- 189 с.

95. Копытов Е.Ю., Любченко А.А. Оценка периодичности профилактического обслуживания технических систем на основе модели полумарковского про-

цесса // Доклады Томского государственного университета систем управления и радиоэлектроники. 2011. № 2-2 (24). - С. 300 – 305.

96. Кормилкин А.А. Алгоритмы многокритериальной оптимизации параметров систем управления мобильными робототехническими комплексами: Дис. … кандидата техн. наук. 05.13.01 (Москва, 2010). - 150 с.

97. Королюк В.С. Время пребывания полумарковского процесса в фиксированном множестве состояний // Украинский математический журнал. - 1965 - № 3. - С. 123 - 128.

98. Королюк В.С. Об асимптотическом поведении времени пребывания полумарковского процесса в подмножестве состояний // Украинский математический журнал. - 1969. - № 6. С. 842 - 846.

99. Королюк В.С., Полищук Л.И., Томусяк А.А. Об одной предельной теореме для полумарковских процессов // Кибернетика. - 1969. - № 4, С. 144 - 145.

100. Королюк В.С., Томусяк А.А. 0 некоторых стационарных характеристиках полумарковских процессов. - Кибернетика. - 1971. - № .5. - С. 65 - 68.

101. Королюк В.С., Томусяк А.А. Описание функционирования резервированных систем посредством полумарковских процессов // Кибернетика. - 1965. - № 5. 0 С. 55 -59.

102. Королюк В.С., Турбин А.Ф. Полумарковские процессы и их применения. - Киев: Наукова думка, 1976. - 184 с.

103. Королюк В.С., Турбин А.Ф. Об асимптотическом поведении времени пребывания полумарковского процесса в приводимом подмножестве состояний. Теория вероятностей и мат. стат. Межвед. науч. *сб.,* 1970, вып. 2, 133—143 (РЖМат, 1971, 3В37)

104. Королюк В.С., Турбин А.Ф. Об одном методе доказательства предельных теорем для некоторых функционалов от полумарковских процессов // Украинский математический журнал. - 1972, - № 2. - С. 234 - 340.

105. Котов В.В., Котова Н.А., Ларкин Е.В. Генерация Петри-Марковских моделей в задачах оптимизации когнитивных технологий обучения // Известия

ТулГУ. Технические науки. 2013. - Вып. 9. Ч. 1. - Тула: Изд-во ТулГУ. - С. 298-303.

106. Котов В.В., Котова Н.А., Ларкин Е.В. Метод имитационного моделирования систем с использованием сетей Петри-Маркова // Известия ТулГУ. Сер. Технические науки. - Вып. 9. - 2015. - С. 164 - 170.

107. Котова Н.А., Ларкин Е.В. Проектирование информационных систем роботов с использованием сетей Петри-Маркова: Учебное пособие. - Тула: Изд-во ТулГУ, 2008. - 158 с.

108. Котюк А.Б. Датчики в современных измерениях. - М.: Радио и связь: Горячая линия - Телеком. - 2006. - 96 с.

109. Краснов М.П., Киселев А.И., Макаренко Г.И. Функции комплексного переменного. Операционное исчисление. Теория устойчивости. - М.: Наука, 1971. - 304 с.

110. Красовский Н.Н. Некоторые задачи теории устойчивости движения. М., Государственное изд. физ.-мат. литературы, 1959. 211 с.

111. Кузнецов С.В. Математические модели процессов и систем технической эксплуатации авионики как марковские и полумарковские процессы // Научный вестник Московского государственного технического университета гражданской авиации. - 2015. - № 213 (3), - С. 28 – 33.

112. Кузнецов С.В. Математические модели процессов и систем технической эксплуатации бортовых комплексов и функциональных систем авионики // Научный вестник Московского государственного технического университета гражданской авиации, 2017. - Т. 20, № 1, - С. 132 – 140.

113. Куконин А.Г., Ларкин Е.В. Исследование временных характеристик алгоритмов интерактивного взаимодействия пользователя с ЭВМ // Автоматизация технологической подготовки производства. Минск: ИТК АН БССР, 1986. - С. 89 - 96.

114. Лариошкин И.Н., Акименко Т.А. Параметры тепловизионных камер. Известия ТулГУ. Серия Технические науки. Вып. 2. Тула: Изд-во ТулГУ, 2018.- с.239-243

115. Ларкин Е.В. «Соревнование» в M-L-параллельном полумарковском процессе // Известия Тульского государственного университета. Технические науки. - 2016. - № 2. С. 133 – 141.

116. Ларкин Е.В. Многостадийные соревновательные игры // Известия ТулГУ. Сер. Технические науки. - 2016. - Вып. 5. - Тула: Изд-во ТулГУ, - С. 52 – 66.

117. Ларкин Е.В. Моделирование параллельных полумарковских процессов. Известия ТулГУ. Серия Технические науки. Вып. 2. Тула: Изд-во ТулГУ, 2018.- с.3-11

118. Ларкин Е.В., Антонов М.А., Басс А.В. Преобразователь последовательного интерфейса в параллельный. Известия ТулГУ. Серия Технические науки. Вып. 9. Тула: Изд-во ТулГУ, 2018.- с.43-49

119. Ларкин Е.В., Горбачев Д.В., Привалов А.Н. О приближении поток событий к пуассоновскому потоку // Чебышевский сборник. Научно-теоретический математический журнал. - 2017. - Т. XVIII. - Вып. 2 (62). - С. 222 - 234.

120. Ларкин Е.В., Гришин К.А., Антонов М.А. Модель циклограммы управления мобильным роботом как полумарковский процесс. Известия Тульского государственного университета. Технические науки. 2018. № 2. С. 188-195.

121. Ларкин Е.В., Ивутин А.Н. Определение временных интервалов в алгоритмах управления // Известия Томского политехнического университета. - № 5. - Т. 324. - 2014. - С. 6 - 12.

122. Ларкин Е.В., Ивутин А.Н., Костомаров Д.С. Методика формирования сети Петри-Маркова для моделирования когнитивных технологий // Известия ТулГУ. Технические науки. - 2013. - Вып. 9. Ч. 1. - Тула: Изд-во ТулГУ. - С. 303 - 311.

123. Ларкин Е.В., Котов В.В., Котова Н.А. Оценка эффективности программного обеспечения робота с использованием сетей Петри-Маркова // Известия ТулГУ. Технические науки. - 2013. - Вып. 9. Ч. 2. - Тула: Изд-во ТулГУ, - С. 156 - 163.

124. Ларкин Е.В., Привалов А.Н. О дискретном подходе к моделированию парных эстафет. Известия ТулГУ. Серия Технические науки. Вып. 9. Тула: Изд-во ТулГУ, 2018.- с. 28-42

125. Ларкин Е.В., Привалов А.Н. О задаче распределения информации по уровням хранения в вычислительных модулях тренажерной системы // Известия ТулГУ. Сер. Технические науки. - Вып. 8. - 2015. - С. 175 - 183.

126. Ларкин Е.В., Привалов А.Н. Проектирование программного обеспечения вычислительных средств тренажерных систем. - Тула: Изд-во ТулГУ, 2010. - 259 с. ISBN 978-5-7679-1701-3

127. Ларкин Е.В., Привалов А.Н. Создание программного обеспечения тренажерных систем на основе унифицированных программных модулей // Вестник компьютерных и информационных технологий. - 2010. -№ 4. -С. 50 - 56.

128. Ларкин Е.В., Привалов А.Н., Акименко Т.А. Возможности инженерной психологии при исследовании когнитивных процессов // Известия ТулГУ. Технические науки. С. 311 - Вып. 9. Ч. 1. - Тула: Изд-во ТулГУ, 2013. - С. - 319.

129. Ларкин Е.В., Привалов А.Н., Ивутин А.Н. Моделирование когнитивного процесса тренинга в эргатических системах. - Saarbrucken Deutchland: LAP LAMBERT Academic Publishing GmbH & Co., 2013. - 232 Pp. ISBN 978-3-659-33370-5

130. Ларкин Е.В., Привалов А.Н., Ивутин А.Н. Оптимизация многоуровневых систем с применением сетей Петри-Маркова // Телекоммуникации. - 2016. - № 12. - С. 2 - 10.

131. Ларкин Е.В., Сычугов А.А. К вопросу о соревновании случайных процессов // Известия ТулГУ. Технические науки. - 2013. - Вып. 3. - Тула: Изд-во ТулГУ. - С. 275 - 282.

132. Ларкин Е.В., Сычугов А.А. Соревновательные игры // Известия ТулГУ. Технические науки. - 2013. - Вып. 7. Ч. 2. - Тула: Изд-во ТулГУ. - С. 108 - 116.

133. Мазалов В.В., Винниченко Е.В. Моменты остановки и управляемые случайные блуждания. - Новосибирск: Наука, 1992. - 104 с.

134. Мальцев Г.Н., Вознюк В.В., Туктамышев М.Р. Моделирование конфликта сложных радиотехнических систем методом параллельных развивающихся стохастических процессов // Информационно-управляющие системы. - 2013. - № 5 (66), С. 26 – 33.

135. Математические модели, динамические характеристики и анализ систем автоматического управления. Т. 1. Методы классической и современной теории автоматического управления / Ред. К.А. Пупков и Н.Д. Егупов. - М.: Изд-во МГТУ им. Н.Э.Баумана, 2004. - 656 с.

136. Метод построения нечеткой полумарковской модели функционирования сложной системы // Борисов В.В., Бояринов Ю.Г., Дли М.И., Мищенко В.И. Программные продукты и системы. - 2010 - № 3, С. 26 – 31.

137. Морозов В.Г. О топологических методах исследования конечных полумарковских автоматов // Сб. Автоматы, гибридные и управляющие машины. - М,, Наука, 1972. - С. 50 - 59 .

138. Окулов С.М. Динамическое программирование. / Окулов С.М., Пестов О.А. – М.:Бином. Лаборатория знаний, 2012. -296 с.

139. Ориентация и навигация подвижных объектов: Современные информационные технологии / Б.С. Алешин и др. Ред. Б.С. Алешина, К.К. Веремеенко,, А.И. Черноморского. - М.: Физматлит, 2006. - 424 с.

140. Орлов С.А. О-66 Программная инженерия. Учебник для вузов. 5-е издание обновлённое и дополненное. Стандарт третьего поколения. – СПб.: Питер, 2016. – 640 с.: ил. – (Серия «Учебник для вузов»).

141. Отчет о научно-исследовательской работе «Параллельные полумарковские процессы в системах управления мобильными роботами» (промежуточный, этап 1), № 2.3121.2017/ПЧ

142. Параллельные полумарковские процессы в задачах группового управления объектами. Привалов А.Н., Ларкин Е.В. В сборнике: Алгебра, теория чисел и дискретная геометрия: современные проблемы и приложения Материалы XV Международной конференции, посвященной столетию со дня рождения профессора Николая Михайловича Коробова. 2018. С. 54-56.

143. Парамонов П.+П. Методология информационного проектирования систем авионики: Дис. … доктора техн. наук: 05.11.16, С-Пб, 2003. - 273 с.

144. Патент РФ на полезную модель № 176316, «Тест-объект для измерения разрешения тепловизоров»/ Акименко Т.А., Ларкин Е.В., Филиппова Е.В./ Решение о выдаче, заявка от 25.07.2017

145. Патент РФ на полезную модель № 185002, «Устройство буферизации потока данных»/ Антонов М.А., Гришин К.А., Ларкин Е.В. Заявка от 18.06.2018 № 2018122329.

146. Пельпор Д.С. Гироскопические системы ориентации и стабилизации: Справочное пособие. - Л.: Машиностроение, 1982. - 165 с.

147. Привалов А.Н., Ларкин Е.В. Моделирование информационных процессов тренажерных систем: Концепция, методология, модели. - Saarbrucken Deutchland: LAP LAMBERT Academic Publishing GmbH & Co., 2012. - 230 Pp. ISBN 978-3-8473-3699-0.

148. Промышленный робот с информационной системой управления // Е.В.Ларкин, Т.А.Акименко, А.А.Аршакян, А.Н.Будков, - Известия ТулГУ. Технические науки. - 2013. - Вып. 4. - Тула: Изд-во ТулГУ, - С. 133 - 138.

149. Проталинский И.О. Модели и алгоритмы управления группой мобильных роботов : Дис. ... канд. техн. наук. 05.13.01 (Саратов, 2013 г.). - 150 с.

150. Пытьев Ю.П. Методы математического моделирования измерительно-вычислительных систем. - М.: Физматлит, 2002. - 384 с.

151. Репин М.Ю. Квадратичные функционалы Ляпунова для систем с запаздыванием // Прикладная математика и механика. 1965. Т. 29. С. 564-566.

152. Робастное управление мобильными роботами с использованием технического зрения // М.В.Фаронов, А.А.Пыркин, И.Б.Фуртат, С.А.Колюбин, М.О.Суров, А.А.Ведяков. - Известия высших учебных заведений. Приборостроение. 2012. - Т. 55. - № 12. - С. 63 – 65.

153. Робототехника и техническая кибернетика. 2014. - № 3. С. 39 – 42.

154. Розанов Ю.А. Стационарные случайные процессы. - М.: Наука, 1990. - 271 с.

155. Руководство по применению микромеханического трёхосевого гироскопа L3G4200 [электронный ресурс] // MICROSIN.NET. URL: http://microsin.net/adminstuff/hardware/l3g4200d-mems-motion-sensor.html (дата обращения 10.10.2018).

156. Сабо Ю. И. Методология системного проектирования авионики с отказоустойчивыми свойствами: Дис. … доктора техн. наук: 05.11.16, Тула, 2004. - 359 с.

157. Сапаговас И. О сходимости сумм марковских процессов восстановления к процессу Пуассона // Liet. mat. rinkinys, Литовский математический сборник. - 1966. - № 2. - С. 271 - 277.

158. Сапаговас И. О сходимости сумм марковских процессов восстановления к многомерному процессу Пуассона // Liet. mat. rinkinys, Литовский математический сборник. - 1969. - № 4. С. 817 - 826.

159. Фляте Д.М. Направления развития производства бумаги для гофрирования // Целлюлоза, бумага и картон: Обзорная информация. ВНИПИЭИлеспром. – 1990.- Вып.8.- 36 с.

160. Grishin K.A. Hierarchical digital control system performance/Larkin E., Akimenko T.A., Grishin K.A.// Frontiers in Artificial Intelligence and Applications (FAIA), Seoul, 2021, 70-76.

ПРИЛОЖЕНИЕ

Листинг основных функций программного обеспечения для расчета характеристик блужданий по полумарковским процессам

ПА.1. Файл Unit1.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include <stdio.h>

#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------


THintWindow* RevealHint (TControl* Control);
void RemoveHint (THintWindow* Hint);

//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Image1->Picture->Bitmap->PixelFormat = pf24bit;
  Image1->Picture->Bitmap->Height = 1500;
  Image1->Picture->Bitmap->Width  = 1000;

  Image1->Canvas->Brush->Color = clWhite;
  Image1->Canvas->FillRect(Image1->ClientRect);


  Image2->Picture->Bitmap->PixelFormat = pf24bit;
  Image2->Picture->Bitmap->Height = 1500;
  Image2->Picture->Bitmap->Width  = 1000;


  //Настройка закладок в соответствии с текущим режимом работы
  //На этапе создания структуры свойства объектов не отображаются
  TabSheet1->TabVisible = false;
  TabSheet2->TabVisible = false;

  //Текущий режим работы -- создани структуры сети
  TabSheet3->TabVisible = true;

  //На этапе создания структуры параметры объектов
  //менять/задавать/отображать нельзя
  TabSheet4->TabVisible = false;

  //На этапе создания структуры запускать моделирование нельзя
  TabSheet5->TabVisible = false;

  //Настройка заголовков таблиц
  StringGrid1->Cells[0][0] = "№";
  StringGrid1->Cells[0][1] = "Вероятность выхода";

  //Инициализация генератора псевдослучайных чисел
  Randomize();
}
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
  : TForm(Owner)
```

```
{
  AddPos   = false; //Сейчас не добавляем позицию
  AddTrans = false; //не добавляем переход
  AddArc   = false; //не добавляем дугу

  MouseX = -1;
  MouseY = -1;

  SelObjInd  = -1;  //Ни один объект не выделен
  SelObjType = -1;  //Ни один объект не выделен

  NPos = 0; //Текущее количество позиций -- 0
  NTra = 0; //Текущее количество переходов -- 0
  NArc = 0; //Текущее количество дуг -- 0
  NNod = 0; //Текущее количество узлов дуг -- 0

  FirstPointInd = -1; //Мы не выбираем объект для дуги

  NMark = 1;

  EditingMode = true; //Система находится в режиме редактирования

}
//---------------------------------------------------------------------------
void TForm1::PrintNet()
{
  Image2->Canvas->Brush->Color = clWhite;
  Image2->Canvas->FillRect(Image2->ClientRect);

  int x, y;
  //Сначала отрисуем дуги
  for(int i=0; i<NArc; i++){
    switch(ARC[i].type1){
      case 1:
        x = POSShape[ARC[i].ind1]->Left+10;
        y = POSShape[ARC[i].ind1]->Top+10;
        break;

      case 2:
        x = TRAShape[ARC[i].ind1]->Left+10;
        y = TRAShape[ARC[i].ind1]->Top+10;
        break;

      case 3:
        x = NODShape[ARC[i].ind1]->Left+2;
        y = NODShape[ARC[i].ind1]->Top+2;
        break;
    }

    x += ScrollBox1->HorzScrollBar->Position;
    y += ScrollBox1->VertScrollBar->Position;

    Image2->Canvas->MoveTo(x, y);

    switch(ARC[i].type2){
      case 1:
        x = POSShape[ARC[i].ind2]->Left+10;
        y = POSShape[ARC[i].ind2]->Top+10;
        break;

      case 2:
        x = TRAShape[ARC[i].ind2]->Left+10;
        y = TRAShape[ARC[i].ind2]->Top+10;
```

```
        break;

      case 3:
        x = NODShape[ARC[i].ind2]->Left+2;
        y = NODShape[ARC[i].ind2]->Top+2;
        break;
    }

    x += ScrollBox1->HorzScrollBar->Position;
    y += ScrollBox1->VertScrollBar->Position;

    Image2->Canvas->Pen->Color = (ARC[i].type0 == 1 ? clRed : clLime);
    Image2->Canvas->LineTo(x, y);
  }

  Image1->Canvas->CopyRect(Image1->ClientRect, Image2->Canvas,
              Image1->ClientRect);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
  AddPos = !AddPos;
  //Изменим "жирность" кнопки добавления позиции
  if(AddPos) SpeedButton1->Font->Style = SpeedButton1->Font->Style << fsBold;
  else       SpeedButton1->Font->Style = SpeedButton1->Font->Style >> fsBold;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Image1MouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
  if(AddPos){ //Режим добавления новой позиции
    AddPos = false;

    //Уберём "жирность" с кнопки добавления позиции
    SpeedButton1->Font->Style = SpeedButton1->Font->Style >> fsBold;

    if(POSShape[NPos]) delete POSShape[NPos];
    POSShape[NPos] = new PMShape(this);
    POSShape[NPos]->Parent = ScrollBox1;
    POSShape[NPos]->Shape  = stCircle;
    POSShape[NPos]->Brush->Color = TColor(clRed);
    POSShape[NPos]->Height = 21;  //2*10+1
    POSShape[NPos]->Width  = 21;
    POSShape[NPos]->ShowHint = true;

    POSShape[NPos]->Top  = Y-10 - ScrollBox1->VertScrollBar->Position;
    POSShape[NPos]->Left = X-10 - ScrollBox1->HorzScrollBar->Position;

    POSShape[NPos]->Tag = NPos;

    POSShape[NPos]->OnMouseDown = POSMouseDown;
    POSShape[NPos]->OnMouseUp   = POSMouseUp;
    POSShape[NPos]->OnMouseMove = POSMouseMove;
    NPos++;
  }
  else
  if(AddTrans){ //Режим добавления нового перехода
    AddTrans = false;

    //Уберём "жирность" с кнопки добавления перехода
    SpeedButton2->Font->Style = SpeedButton2->Font->Style >> fsBold;

    if(TRAShape[NTra]) delete TRAShape[NTra];
```

```
  TRAShape[NTra] = new PMShape(this);
  TRAShape[NTra]->Parent = ScrollBox1;
  TRAShape[NTra]->Shape  = stRectangle;
  TRAShape[NTra]->Brush->Color = TColor(clLime);
  TRAShape[NTra]->Height = 21;  //2*10+1
  TRAShape[NTra]->Width  = 21;


  TRAShape[NTra]->Top  = Y-10 - ScrollBox1->VertScrollBar->Position;
  TRAShape[NTra]->Left = X-10 - ScrollBox1->HorzScrollBar->Position;

  TRAShape[NTra]->Tag = NTra;

  TRAShape[NTra]->OnMouseDown = TRAMouseDown;
  TRAShape[NTra]->OnMouseUp   = TRAMouseUp;
  TRAShape[NTra]->OnMouseMove = TRAMouseMove;

  NTra++;
}
else
if(AddArc){ //Режим добавления новой дуги
  if(FirstPointInd == -1){ //Выбирается первая точка
    //Если попали сюда, значит при добавлении первой точки дуги
    //промахнулись по позиции/переходу и попали в пустое изображение
    MessageBeep(MB_ICONASTERISK);
    return;
  }
  else{
    //Если попали сюда, значит промахнулись по объекту-приёмнику дуги
    //Следовательно нужно сформировать промежуточный узел
    if(NODShape[NNod]) delete NODShape[NNod];
    NODShape[NNod] = new PMShape(this);
    NODShape[NNod]->Parent = ScrollBox1;
    NODShape[NNod]->Shape  = stCircle;
    NODShape[NNod]->Pen->Color   = BasePointType == 1 ? clRed : clLime;
    NODShape[NNod]->Brush->Color = BasePointType == 1 ? clRed : clLime;
    NODShape[NNod]->Height = 5;  //2*2+1
    NODShape[NNod]->Width  = 5;


    NODShape[NNod]->Top  = Y-2 - ScrollBox1->VertScrollBar->Position;
    NODShape[NNod]->Left = X-2 - ScrollBox1->HorzScrollBar->Position;

    NODShape[NNod]->Tag = NNod;


    NODShape[NNod]->OnMouseDown = NODMouseDown;
    NODShape[NNod]->OnMouseUp   = NODMouseUp;
    NODShape[NNod]->OnMouseMove = NODMouseMove;

    //Теперь добавим новую дугу
    ARC[NArc].type0 = BasePointType;

    ARC[NArc].ind1  = FirstPointInd;
    ARC[NArc].type1 = FirstPointType;

    ARC[NArc].ind2  = NNod;
    ARC[NArc].type2 = 3; //Окончанием дуги был узел дуги

    //Т.к. продолжаем рисовать дугу, то теперь предыдущим объектом
    //стал узел
    FirstPointInd = NNod;
    FirstPointType = 3;
```

```
      NArc++;  //Теперь одной дугой стало больше
      NNod++;  //и одним узлом больше

      PrintNet();
    }
    //AddArc не присваиваем false, т.к. рисование дуги продолжается
  }
  else
  if(MouseX > -1 && MouseY > -1){ //Возможно передвигаем какой-то объект
  }
}
//---------------------------------------------------------------------------
//Удалить дугу с индексом ind и перенумеровать остальные
void TForm1::DeleteArc(int ind)
{
  if(ind >= NArc) return; //Неправильный индекс удаляемой дуги

  for(int i = ind; i<NArc-1; i++)
    ARC[i] = ARC[i+1];

  NArc--;
}
//---------------------------------------------------------------------------
//Удалить позицию с индексом ind, связанные с ней дуги
//и перенумеровать остальные + скорректировать ссылки в дугах
void TForm1::DeletePos(int ind)
{
  if(ind >= NPos) return; //Неправильный индекс удаляемой позиции

  int i = 0;

  //Сначала удалим все исходящие дуги, связанные с позицией
  while(i<NArc){
    if(ARC[i].type1 == 1 && ARC[i].ind1 == ind) DeleteArc(i);
    else i++;
  }

  //Затем удалим все входящие дуги, связанные с позицией
  i = 0;
  while(i<NArc){
    if(ARC[i].type2 == 1 && ARC[i].ind2 == ind) DeleteArc(i);
    else i++;
  }

  //А теперь удалим саму позицию и перенумеруем оставшиеся
  delete POSShape[ind];
  for(i=ind; i<NPos-1; i++){
    POSShape[i] = POSShape[i+1];
    POSShape[i]->Tag = i;
  }
  NPos--;
  POSShape[NPos] = NULL;

  //Переиндексируем все дуги, указывавшие на объекты после удаляемой позиции
  for(i=0; i<NArc; i++){
    if(ARC[i].type1 == 1 && ARC[i].ind1 > ind)
      ARC[i].ind1--;
    if(ARC[i].type2 == 1 && ARC[i].ind2 > ind)
      ARC[i].ind2--;
  }
}
//---------------------------------------------------------------------------
```

```
//Удалить переход с индексом ind, связанные с ним дуги
//и перенумеровать остальные + скорректировать ссылки в дугах
void TForm1::DeleteTrans(int ind)
{
  if(ind >= NTra) return; //Неправильный индекс удаляемого перехода

  int i = 0;

  //Сначала удалим все исходящие дуги, связанные с переходом
  while(i<NArc){
    if(ARC[i].type1 == 2 && ARC[i].ind1 == ind) DeleteArc(i);
    else i++;
  }

  //Затем удалим все входящие дуги, связанные с переходом
  i = 0;
  while(i<NArc){
    if(ARC[i].type2 == 2 && ARC[i].ind2 == ind) DeleteArc(i);
    else i++;
  }

  //А теперь удалим сам переход и перенумеруем оставшиеся
  delete TRAShape[ind];
  for(i=ind; i<NTra-1; i++){
    TRAShape[i] = TRAShape[i+1];
    TRAShape[i]->Tag = i;
  }
  NTra--;
  TRAShape[NTra] = NULL;

  //Переиндексируем все дуги, указывавшие на объекты после удаляемого перехода
  for(i=0; i<NArc; i++){
    if(ARC[i].type1 == 2 && ARC[i].ind1 > ind) ARC[i].ind1--;
    if(ARC[i].type2 == 2 && ARC[i].ind2 > ind) ARC[i].ind2--;
  }
}
//---------------------------------------------------------------------------
//Удалить переход с индексом ind, связанные с ним дуги
//и перенумеровать остальные + скорректировать ссылки в дугах
void TForm1::DeleteNode(int ind)
{
  if(ind >= NNod) return; //Неправильный индекс удаляемого узла

  int i = 0;

  //Сначала удалим все исходящие дуги, связанные с узлом
  while(i<NArc){
    if(ARC[i].type1 == 3 && ARC[i].ind1 == ind) DeleteArc(i);
    else i++;
  }

  //Затем удалим все входящие дуги, связанные с узлом
  i = 0;
  while(i<NArc){
    if(ARC[i].type2 == 3 && ARC[i].ind2 == ind) DeleteArc(i);
    else i++;
  }

  //А теперь удалим сам узел и перенумеруем оставшиеся
  delete NODShape[ind];
  for(i=ind; i<NNod-1; i++){
    NODShape[i] = NODShape[i+1];
    NODShape[i]->Tag = i;
```

```
  }
  NNod--;
  NODShape[NNod] = NULL;

  //Переиндексируем все дуги, указывавшие на объекты после удаляемого узла
  for(i=0; i<NArc; i++){
    if(ARC[i].type1 == 3 && ARC[i].ind1 > ind) ARC[i].ind1--;
    if(ARC[i].type2 == 3 && ARC[i].ind2 > ind) ARC[i].ind2--;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
  AddTrans = !AddTrans;
  //Изменим "жирность" кнопки добавления перехода
  if(AddTrans)
    SpeedButton2->Font->Style = SpeedButton2->Font->Style << fsBold;
  else
    SpeedButton2->Font->Style = SpeedButton2->Font->Style >> fsBold;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::POSMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
  switch(Button){
    case mbLeft:
      //Запомним координаты на тот случай, если нужно двигать объект
      MouseX = X; MouseY = Y;

      if(AddArc){ //Добавляем дугу
        if(FirstPointInd == -1){ //Выбирается первая точка
          FirstPointInd  = ((PMShape*)Sender)->Tag;
          FirstPointType = 1; //Это позиция
          BasePointType  = 1;
        }
        else{ //Это выбирается уже вторая точка
          if(BasePointType == 2){
            int SecondPointIndex = ((PMShape*)Sender)->Tag;

            //Началом комлпексной дуги был переход
            ARC[NArc].type0 = BasePointType;

            ARC[NArc].ind1  = FirstPointInd;
            //Началом дуги был переход (или узел дуги)
            ARC[NArc].type1 = FirstPointType;

            ARC[NArc].ind2  = SecondPointIndex;
            ARC[NArc].type2 = 1; //Окончанием дуги была позиция

            NArc++;

            AddArc = false;  //Дугу больше не добавляем
            SpeedButton3->Font->Style = SpeedButton3->Font->Style >> fsBold;
            FirstPointInd = -1; //Первая точка не выбрана
          }
          else{ //Первый объект был не переход,
                //поэтому нельзя соединять позицию с позицией
            MessageBeep(MB_ICONASTERISK);
            return;
          }
        }
      }
      else{  //Если не добавляем дугу, а просто "потрогали"
```

```
            //позицию левой кнопкой
          //Работаем со свойствами только в случае, если находимся
          //в режиме редактирования параметров
          if(TabSheet4->TabVisible){
            TabSheet1->TabVisible = true;  //Отображаем закладку про позицю
            TabSheet2->TabVisible = false; //Закладку про переход скрыть

            PopupInd = ((PMShape*)Sender)->Tag;
            Label8->Caption = AnsiString(PopupInd); //Номер позиции

            //Признак того, что эта позиция является начальной в данной подсети
            RadioGroup4->ItemIndex = POS[PopupInd].TypePos;
            Edit7->Text = POS[PopupInd].SubNet; //Номер подсети

            //Считать из базы свойства позиции и отобразить их
            StringGrid1->RowCount = POS[PopupInd].NArcOut+1;
            for(int i=0; i<POS[PopupInd].NArcOut; i++){
              StringGrid1->Cells[0][i+1] = POS[PopupInd].ArcInd[i];
              StringGrid1->Cells[1][i+1] = FormatFloat("0.00", POS[PopupInd].P[i]);
            }

            RadioGroup1->ItemIndex = POS[PopupInd].PLaw[0];
            Edit1->Text = POS[PopupInd].MeanVal[0];
            Edit2->Text = POS[PopupInd].MRS[0];

            ComboBox1->ItemIndex = POS[PopupInd].Action;

            Edit6->Text = POS[PopupInd].Descript;
          }
        }

        break;

      case mbRight:
        TPoint cPt;
        GetCursorPos(&cPt);
        PopupInd = ((PMShape*)Sender)->Tag;
        PopupMenu1->Popup(cPt.x, cPt.y);// открываем контекстное меню
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::POSMouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
  if(Button == mbLeft){
    MouseX = -1; MouseY = -1;
    if(!AddArc) PrintNet();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::POSMouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
  if(MouseX < 0 || MouseY <0 || AddPos || AddTrans || AddArc) return;
  ((PMShape*)Sender)->Left += (X - MouseX);
  ((PMShape*)Sender)->Top  += (Y - MouseY);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TRAMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
  switch(Button){
    case mbLeft:
```

```
  //Запомнить координаты на тот случай, если нужно двигать объект
  MouseX = X; MouseY = Y;

  if(AddArc){ //Добавляем дугу
    if(FirstPointInd == -1){ //Выбирается первая точка
      FirstPointInd  = ((PMShape*)Sender)->Tag;
      FirstPointType = 2; //Это переход
      BasePointType  = 2;
    }
    else{ //Это выбирается уже вторая точка
      if(BasePointType == 1){
        int SecondPointIndex = ((PMShape*)Sender)->Tag;
        //Началом комплексной дуги была позиция
        ARC[NArc].type0 = BasePointType;

        ARC[NArc].ind1  = FirstPointInd;
        //Началом дуги была позиция (или узел дуги)
        ARC[NArc].type1 = FirstPointType;

        ARC[NArc].ind2  = SecondPointIndex;
        ARC[NArc].type2 = 2; //Окончанием дуги был переход

        NArc++;

        AddArc = false;  //Дугу больше не добавляем
        SpeedButton3->Font->Style = SpeedButton3->Font->Style >> fsBold;
        FirstPointInd = -1; //Первая точка не выбрана
      }
      else{ //Первый объект был не позиция, поэтому
            //нельзя соединять переход с переходом
        MessageBeep(MB_ICONASTERISK);
        return;
      }
    }
  }
  else{  //Если не добавляем дугу, а просто "потрогали"
         //переход левой кнопкой
    //Работаем со свойствами только в случае, если находимся
    //в режиме редактирования параметров
    if(TabSheet4->TabVisible){
      TabSheet2->TabVisible = true;  //Отображаем закладку про переход
      TabSheet1->TabVisible = false; //Закладку про позицю скрыть

      PopupInd = ((PMShape*)Sender)->Tag;
      Label7->Caption = AnsiString(PopupInd); //Переход №

      //Считать из базы свойства перехода и отобразить их
      RadioGroup2->Items->Clear();
      for(int i=0; i<TRA[PopupInd].NArcOut; i++)
        RadioGroup2->Items->Add(TRA[PopupInd].ArcOutInd[i]);

      RadioGroup2->ItemIndex = 0;

      //Печать конъюнкций в СтрингГрид2
      PrintLogic(TRA[PopupInd].Logic[0]);
    }
  }
  break;

case mbRight:
  TPoint cPt;
  GetCursorPos(&cPt);
  PopupInd = ((PMShape*)Sender)->Tag;
```

```
      PopupMenu2->Popup(cPt.x, cPt.y);// открываем контекстное меню
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TRAMouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
  if(Button == mbLeft){
    MouseX = -1; MouseY = -1;
    if(!AddArc) PrintNet();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TRAMouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
  if(MouseX < 0 || MouseY <0 || AddPos || AddTrans || AddArc) return;
  ((PMShape*)Sender)->Left += (X - MouseX);
  ((PMShape*)Sender)->Top  += (Y - MouseY);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NODMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
  switch(Button){
    case mbLeft:
      //Запомним координаты на тот случай, если нужно двигать объект
      MouseX = X; MouseY = Y;
      break;

    case mbRight:
      TPoint cPt;
      GetCursorPos(&cPt);
      PopupInd = ((PMShape*)Sender)->Tag;
      PopupMenu3->Popup(cPt.x, cPt.y);// открываем контекстное меню
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NODMouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
  if(Button == mbLeft){
    MouseX = -1; MouseY = -1;
    if(!AddArc) PrintNet();
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NODMouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
  if(MouseX < 0 || MouseY <0 || AddPos || AddTrans || AddArc) return;
  ((PMShape*)Sender)->Left += (X - MouseX);
  ((PMShape*)Sender)->Top  += (Y - MouseY);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N1Click(TObject *Sender)
{
  ShowMessage("Позиция №" + AnsiString(PopupInd));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N2Click(TObject *Sender)
{
  DeletePos(PopupInd);
```

```
  PrintNet();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
  if(!AddArc){
    AddArc = true; //Переключились в режим задания дуги
    SpeedButton3->Font->Style = SpeedButton3->Font->Style << fsBold;
  }
  else{
    AddArc = false;
    SpeedButton3->Font->Style = SpeedButton3->Font->Style >> fsBold;
    FirstPointInd = -1; //Мы больше не выбираем объект для дуги
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N3Click(TObject *Sender)
{
  ShowMessage("Переход №" + AnsiString(PopupInd));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N4Click(TObject *Sender)
{
  DeleteTrans(PopupInd);
  PrintNet();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N6Click(TObject *Sender)
{
  DeleteNode(PopupInd);
  PrintNet();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N5Click(TObject *Sender)
{
  ShowMessage("Узел №" + AnsiString(PopupInd));
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton5Click(TObject *Sender)
{
  if(!SaveDialog1->Execute()) return;

  FILE* fout;

  char fname[255];
  int  fnamelen;
  fnamelen = WideCharToMultiByte (CP_ACP,     // кодорвка ANSI
                      WC_COMPOSITECHECK,      //
                      SaveDialog1->FileName.c_str(), // строка Unicode
                      -1,                     // -1 -- строка завершается 0
                      fname,                  // Строка-приёмник
                      sizeof(fname),          // Размер буфера
                      NULL,
                      NULL);

  if((fout = fopen(fname, "wt")) == NULL)
  {
    ShowMessage("Cannot open output file.\n");
    return;
  }

  //Определим, будут ли в файле сохраняться свойства объектов,
  //или только структура
```

```
  int PROP = (Sender == SpeedButton5) ? 0 : 1;

  //Сначала количество элементов сети
  fprintf(fout, "%d %d %d %d %d\n", NPos, NTra, NNod, NArc, PROP);

  //Теперь выводим координаты всех позиций
  for(int i=0; i<NPos; i++){
    fprintf(fout, "%d %d\n",
      POSShape[i]->Left + ScrollBox1->HorzScrollBar->Position,
      POSShape[i]->Top  + ScrollBox1->VertScrollBar->Position);
  }

  //Теперь выводим координаты всех переходов
  for(int i=0; i<NTra; i++){
    fprintf(fout, "%d %d\n",
      TRAShape[i]->Left + ScrollBox1->HorzScrollBar->Position,
      TRAShape[i]->Top  + ScrollBox1->VertScrollBar->Position);
  }

  //Теперь выводим координаты всех узлов
  for(int i=0; i<NNod; i++){
    fprintf(fout, "%d %d\n",
      NODShape[i]->Left + ScrollBox1->HorzScrollBar->Position,
      NODShape[i]->Top  + ScrollBox1->VertScrollBar->Position);
  }

  //Теперь выводим все дуги
  for(int i=0; i<NArc; i++){
    fprintf(fout, "%d %d %d %d %d\n",
      ARC[i].type0, ARC[i].ind1, ARC[i].type1, ARC[i].ind2, ARC[i].type2);
  }

  //Если сохраняем только структуру, то нужно закрыть файл и завершить функцию
  if(Sender == SpeedButton5){
    fclose(fout);
    return;
  }

  //Иначе переходим к сохранению свойств системы
  //Сначала сохраняем свойства позиций
  for(int i=0; i<NPos; i++){
    fprintf(fout, "%d %d %d %d %d\n", POS[i].ind, POS[i].NArcOut,
            POS[i].Action, POS[i].SubNet, POS[i].TypePos);
    fprintf(fout, "%s\n", POS[i].Descript.c_str());
    for(int j=0; j<POS[i].NArcOut; j++){
      fprintf(fout, "%d %f %d %f %f\n", POS[i].ArcInd[j], POS[i].P[j],
              POS[i].PLaw[j], POS[i].MeanVal[j], POS[i].MRS[j]);
    }
  }

  //Затем сохраняем свойства переходов
  for(int i=0; i<NTra; i++){
    fprintf(fout, "%d %d %d\n", TRA[i].ind, TRA[i].NArcOut, TRA[i].NArcIn);
    for(int j=0; j<TRA[i].NArcOut; j++)
      fprintf(fout, "%d %s\n", TRA[i].ArcOutInd[j], TRA[i].Logic[j].c_str());

    for(int j=0; j<TRA[i].NArcIn; j++)
      fprintf(fout, "%d\n", TRA[i].ArcInInd[j]);
  }

  fclose(fout);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
  if(!OpenDialog1->Execute()) return;

  //Уничтожим всю существующую сеть
  for(int i=0; i<NPos; i++){
    if(POSShape[i]) delete POSShape[i];
    POSShape[i] = NULL;
  }

  for(int i=0; i<NTra; i++){
    if(TRAShape[i]) delete TRAShape[i];
    TRAShape[i] = NULL;
  }

  for(int i=0; i<NNod; i++){
    if(NODShape[i]) delete NODShape[i];
    NODShape[i] = NULL;
  }

  FILE* fin;

  char fname[255];
  int  fnamelen;
  fnamelen = WideCharToMultiByte (CP_ACP,
                      WC_COMPOSITECHECK,
                      OpenDialog1->FileName.c_str(),
                      -1, fname, sizeof(fname),
                      NULL, NULL);

  if((fin = fopen(fname, "rt")) == NULL)
  {
    ShowMessage("Cannot open input file.\n");
    return;
  }

  int PROP; //Признак наличия свойств объектов в загружаемом файле
            //(если 1 -- то структура+свойства, если 0 -- только структура)

  //Сначала количество элементов сети
  fscanf(fin, "%d %d %d %d %d\n", &NPos, &NTra, &NNod, &NArc, &PROP);

  int x, y;
  //Теперь выводим координаты всех позиций
  for(int i=0; i<NPos; i++){
    fscanf(fin, "%d %d\n", &x, &y);

    if(POSShape[i]) delete POSShape[i];
    POSShape[i] = new PMShape(this);
    POSShape[i]->Parent = ScrollBox1;
    POSShape[i]->Shape  = stCircle;
    POSShape[i]->Brush->Color = TColor(clRed);
    POSShape[i]->Height = 21;  //2*10+1
    POSShape[i]->Width  = 21;

    POSShape[i]->Top  = y - ScrollBox1->VertScrollBar->Position;
    POSShape[i]->Left = x - ScrollBox1->HorzScrollBar->Position;

    POSShape[i]->Tag = i;

    POSShape[i]->OnMouseDown = POSMouseDown;
    POSShape[i]->OnMouseUp   = POSMouseUp;
    POSShape[i]->OnMouseMove = POSMouseMove;
```

```
}

//Теперь выводим координаты всех переходов
for(int i=0; i<NTra; i++){
  fscanf(fin, "%d %d\n", &x, &y);

  if(TRAShape[i]) delete TRAShape[i];
  TRAShape[i] = new PMShape(this);
  TRAShape[i]->Parent = ScrollBox1;
  TRAShape[i]->Shape  = stRectangle;
  TRAShape[i]->Brush->Color = TColor(clLime);
  TRAShape[i]->Height = 21;  //2*10+1
  TRAShape[i]->Width  = 21;


  TRAShape[i]->Top  = y - ScrollBox1->VertScrollBar->Position;
  TRAShape[i]->Left = x - ScrollBox1->HorzScrollBar->Position;

  TRAShape[i]->Tag = i;

  TRAShape[i]->OnMouseDown = TRAMouseDown;
  TRAShape[i]->OnMouseUp   = TRAMouseUp;
  TRAShape[i]->OnMouseMove = TRAMouseMove;
}

//Теперь выводим координаты всех узлов
for(int i=0; i<NNod; i++){
  fscanf(fin, "%d %d\n", &x, &y);

  if(NODShape[i]) delete NODShape[i];
   NODShape[i] = new PMShape(this);
   NODShape[i]->Parent = ScrollBox1;
   NODShape[i]->Shape  = stCircle;

   NODShape[i]->Height = 5;  //2*4+1
   NODShape[i]->Width  = 5;

   NODShape[i]->Top  = y - ScrollBox1->VertScrollBar->Position;
   NODShape[i]->Left = x - ScrollBox1->HorzScrollBar->Position;

   NODShape[i]->Tag = i;

   NODShape[i]->OnMouseDown = NODMouseDown;
   NODShape[i]->OnMouseUp   = NODMouseUp;
   NODShape[i]->OnMouseMove = NODMouseMove;
}

//Теперь выводим все дуги
for(int i=0; i<NArc; i++){
  fscanf(fin, "%d %d %d %d %d\n",
      &ARC[i].type0, &ARC[i].ind1, &ARC[i].type1, &ARC[i].ind2, &ARC[i].type2);
}

//Устанавливаем правильные цвета всех узлов
for(int j=0; j<NArc; j++){
  if(ARC[j].type1==3){
  //Когда маркер создаётся, его нужно отрисовать
  NODShape[ARC[j].ind1]->Pen->Color = ARC[j].type0 == 1 ? clRed : clLime;
  NODShape[ARC[j].ind1]->Brush->Color = ARC[j].type0 == 1 ? clRed : clLime;
  }
}

char temp_buf[256];
```

```
if(PROP && (Sender == SpeedButton10)){
  //В файле присутствуют свойства объектов сети, их тоже нужно прочитать

  //Сначала считываем свойства позиций
  for(int i=0; i<NPos; i++){
    fgets(temp_buf, 256, fin);
    sscanf(temp_buf, "%d %d %d %d %d", &POS[i].ind, &POS[i].NArcOut,
           &POS[i].Action, &POS[i].SubNet, &POS[i].TypePos);

    fgets(temp_buf, 256, fin);
    for(int j = 0; temp_buf[j]!= '\0'; j++)
      if(temp_buf[j] == '\n') temp_buf[j] = '\0';
    POS[i].Descript    = AnsiString(temp_buf);
    POSShape[i]->Hint  = POS[i].Descript;


    for(int j=0; j<POS[i].NArcOut; j++){
      fgets(temp_buf, 256, fin);
      sscanf(temp_buf, "%d %f %d %f %f", &POS[i].ArcInd[j], &POS[i].P[j],
             &POS[i].PLaw[j], &POS[i].MeanVal[j], &POS[i].MRS[j]);
    }
  }

  char buf[255];
  //Затем считываем свойства переходов
  for(int i=0; i<NTra; i++){
    fscanf(fin, "%d %d %d", &TRA[i].ind, &TRA[i].NArcOut, &TRA[i].NArcIn);
    for(int j=0; j<TRA[i].NArcOut; j++){
      fscanf(fin, "%d %s", &TRA[i].ArcOutInd[j], buf);
      TRA[i].Logic[j] = AnsiString(buf);
    }

    for(int j=0; j<TRA[i].NArcIn; j++)
      fscanf(fin, "%d\n", &TRA[i].ArcInInd[j]);
  }


  TabSheet3->TabVisible = false;
  TabSheet4->TabVisible = true;

  //Раскрасим позиции в цвета, соответствующие номерам подсетей
  for(int i=0; i<NPos; i++){
    switch(POS[i].SubNet){
      case 0:  POSShape[i]->Brush->Color = TColor(clPurple);  break;
      case 1:  POSShape[i]->Brush->Color = TColor(clTeal);    break;
      case 2:  POSShape[i]->Brush->Color = TColor(clNavy);    break;
      case 3:  POSShape[i]->Brush->Color = TColor(clMaroon);  break;
      case 4:  POSShape[i]->Brush->Color = TColor(clYellow);  break;
      case 5:  POSShape[i]->Brush->Color = TColor(clFuchsia); break;
      case 6:  POSShape[i]->Brush->Color = TColor(clAqua);    break;
      case 7:  POSShape[i]->Brush->Color = TColor(clGreen);   break;
      case 8:  POSShape[i]->Brush->Color = TColor(clSilver);  break;

      default: POSShape[i]->Brush->Color = TColor(clRed); break;
    }
  }

}
else{
  //В файле была только структура, свойств нет, поэтому их не читаем
  TabSheet3->TabVisible = true;
  TabSheet4->TabVisible = false;
}
```

```
  fclose(fin);

  PrintNet();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
  if(Key == ',' || Key == '.') Key = DecimalSeparator;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  if(Application->MessageBox(L"При переходе к редактированию структуры сети\n"
                    L"информация о параметрах будет утеряна. Продолжить?",
                    L"", MB_YESNO) != IDYES) return;

  TabSheet3->TabVisible = true;
  TabSheet4->TabVisible = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  if(Application->MessageBox(L"При переходе к редактированию параметров\n"
                             L"дальнейшее изменение структуры будет невозможно."
                             L" Продолжить?", L"", MB_YESNO) != IDYES) return;

  TabSheet3->TabVisible = false;
  TabSheet4->TabVisible = true;


  try{
    //Сформируем параметры по умолчанию
    //Сначала обработаем все позиции
    int i, j, k, n;

    for(i=0; i<NPos; i++){
      POS[i].ind = i;
      POS[i].NArcOut = 0;
      for(j=0; j<NArc; j++){
        if(ARC[j].type1 == 1 && ARC[j].ind1 == i){
          //Нашли дугу, исходящую из данной позиции
          if(ARC[j].type2 == 2){ //Если исходящая дуга непосредственно соединена
          //с переходом то информацию можно сразу внести в свойства позиции
            POS[i].ArcInd[POS[i].NArcOut] = ARC[j].ind2;
          }
          else{ //Иначе дуга связана с узлом и нужно пройтись по цепочке узлов,
                //пока не встретим переход
            n = j;
            do{
              for(k=0; k<NArc; k++){
                //Поиск дуги, у которой начальный элемент --
                // -- узел с заданным номером
                if(ARC[k].type1 == 3 && ARC[k].ind1 == ARC[n].ind2) break;
              }

              if(k==NArc) throw(AnsiString("Некорректная структура сети"));

              //Сейчас в k находится номер дуги, исходящей из промежуточного узла
              if(ARC[k].type2 == 2){
                POS[i].ArcInd[POS[i].NArcOut] = ARC[k].ind2;
                break;
              }
```

```
          n = k;
        }
        while(true);
      }
      POS[i].NArcOut++;
    }
  }

  for(j=0; j<POS[i].NArcOut; j++){
    POS[i].P[j] = 1.0/POS[i].NArcOut;  //Вероятности выхода в соотв. дуги
    POS[i].PLaw[j] = 0;                //Тип закона по умолчанию равномерный
    POS[i].MeanVal[j] = 0.0;           //Матожидание соответствующего закона
    POS[i].MRS[MaxArc] = 0.0;          //СКО соответствующего закона
  }
}

//Теперь -- все переходы
//Сначала аналогично найдём все исходящие дуги
for(i=0; i<NTra; i++){
  TRA[i].ind = i;
  TRA[i].NArcOut = 0;
  for(j=0; j<NArc; j++){
    if(ARC[j].type1 == 2 && ARC[j].ind1 == i){
      //Нашли дугу, исходящую из данного перехода
      if(ARC[j].type2 == 1){ //Если исходящая дуга непосредственно
                             //соединена с позицией, то информацию из неё
                             //можно сразу внести в свойства перехода
        TRA[i].ArcOutInd[TRA[i].NArcOut] = ARC[j].ind2;
      }
      else{ //Иначе дуга связана с узлом и нужно пройтись по цепочке узлов,
            //пока не встретим позицию
        n = j;
        do{
          for(k=0; k<NArc; k++){
            //Поиск дуги, у кот. начальный элемент - узел с заданным номером
            if(ARC[k].type1 == 3 && ARC[k].ind1 == ARC[n].ind2) break;
          }

          if(k==NArc) throw(AnsiString("Некорректная структура сети"));

          //Сейчас в k находится номер дуги, исходящей из промежуточн. узла
          if(ARC[k].type2 == 1){
            TRA[i].ArcOutInd[TRA[i].NArcOut] = ARC[k].ind2;
            break;
          }

          n = k;
        }
        while(true);
      }
      TRA[i].NArcOut++;
    }
  }

  //Теперь все входящие в переход дуги
  TRA[i].NArcIn = 0;
  for(j=0; j<NArc; j++){
    if(ARC[j].type2 == 2 && ARC[j].ind2 == i){
      //Нашли дугу, входящую в данный переход
      if(ARC[j].type1 == 1){ //Если входящая дуга непосредственно соединена
                             //с позицией то информацию из неё можно сразу
                             //внести в свойства перехода
```

```
            TRA[i].ArcInInd[TRA[i].NArcIn] = ARC[j].ind1;
          }
          else{ //Иначе дуга связана с узлом и нужно пройтись по цепочке узлов,
                //пока не встретим позицию
            n = j;
            do{
              for(k=0; k<NArc; k++){
                //Ищем дугу, у кот. конечный элемент -- узел с заданным номером
                if(ARC[k].type2 == 3 && ARC[k].ind2 == ARC[n].ind1) break;
              }

              if(k==NArc) throw(AnsiString("Некорректная структура сети"));

              //Сейчас в k находится номер дуги, исходящей из промежуточн. узла
              if(ARC[k].type1 == 1){
                TRA[i].ArcInInd[TRA[i].NArcIn] = ARC[k].ind1;
                break;
              }

              n = k;
            }
            while(true);
          }
          TRA[i].NArcIn++;
        }
      }

      //Заполним перечень выходов
      for(j=0; j<TRA[i].NArcOut; j++){
        TRA[i].Logic[j] = "";
        for(k=0; k<TRA[i].NArcIn; k++)
          TRA[i].Logic[j] += AnsiString("P") + AnsiString(TRA[i].ArcInInd[k]);
      }
    }
  }
  catch(AnsiString &buf){
    ShowMessage(buf);
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::RadioGroup2Click(TObject *Sender)
{
  int tra = Label7->Caption.ToInt();
  int pos = RadioGroup2->ItemIndex;

  PrintLogic(TRA[tra].Logic[pos]);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton8Click(TObject *Sender)
{
  int pos = Label8->Caption.ToInt();
  int tra = StringGrid1->Row-1;

  //Нужно проверить сумму вероятностей выходов, чтобы она не отличалась от 1
  //и если это так, то переписать их все в соответствующие ячейки
  int i;
  float P;
  for(i=0, P=0.0; i<StringGrid1->RowCount-1; i++){
    P += StringGrid1->Cells[1][i+1].ToDouble();
  }

  if(P != 1 && RadioGroup4->ItemIndex != 2){ //Позиция не конечная
    ShowMessage("Сумма вероятностей выходов в переходы не равна единице.\n"
```

```
                "Необходимо ввести коррекцию");
    return;
  }

  POS[pos].SubNet   = Edit7->Text.ToInt();
  POS[pos].TypePos  = RadioGroup4->ItemIndex;
  //скорректируем цвет позиции в соответствии с номером подсети
  switch(POS[pos].SubNet){
    case 0:  POSShape[pos]->Brush->Color = TColor(clPurple); break;
    case 1:  POSShape[pos]->Brush->Color = TColor(clTeal); break;
    case 2:  POSShape[pos]->Brush->Color = TColor(clNavy); break;
    case 3:  POSShape[pos]->Brush->Color = TColor(clMaroon); break;
    case 4:  POSShape[pos]->Brush->Color = TColor(clYellow); break;
    case 5:  POSShape[pos]->Brush->Color = TColor(clFuchsia); break;
    case 6:  POSShape[pos]->Brush->Color = TColor(clAqua); break;
    case 7:  POSShape[pos]->Brush->Color = TColor(clGreen); break;
    case 8:  POSShape[pos]->Brush->Color = TColor(clSilver); break;

    default: POSShape[pos]->Brush->Color = TColor(clRed); break;
  }



  for(i=0; i<StringGrid1->RowCount-1; i++){
    POS[pos].P[i] = StringGrid1->Cells[1][i+1].ToDouble();;
  }

  POS[pos].PLaw[tra]    = RadioGroup1->ItemIndex;
  POS[pos].MeanVal[tra] = Edit1->Text.ToDouble();
  POS[pos].MRS[tra]     = Edit2->Text.ToDouble();

  POS[pos].Action       = ComboBox1->ItemIndex;

  POS[pos].Descript     = Edit6->Text;
  POSShape[pos]->Hint   = POS[pos].Descript;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid1Click(TObject *Sender)
{
  int pos = Label8->Caption.ToInt();
  int tra = StringGrid1->Row-1;

  RadioGroup1->ItemIndex = POS[pos].PLaw[tra];
  Edit1->Text = POS[pos].MeanVal[tra];
  Edit2->Text = POS[pos].MRS[tra];
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton6Click(TObject *Sender)
{
   AnsiString l = "";
   int tra = Label7->Caption.ToDouble();
   int pos = RadioGroup2->ItemIndex;

   int k;

   for(k=0; k<TRA[tra].NArcIn; k++)
       l += AnsiString("P") + AnsiString(TRA[tra].ArcInInd[k]);

   //Добавим ещё одну конъюнкцию
   TRA[tra].Logic[pos] = TRA[tra].Logic[pos] + "|" + l;

   //Печать конъюнкций в СтрингГрид2
```

```
   PrintLogic(TRA[PopupInd].Logic[0]);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::PrintLogic(AnsiString L)
{
  AnsiString l;
  int p1;
  int nRow;

  //Подсчитаем количество конъюнкций, чтобы сформировать нужное число строк
  l = L;
  nRow = 1; //Даже если ни разу не попали в цикл, точно знаем, что строка
            //конъюнкций не пустая и одну строчку всё равно нужно печатать
  while((p1=l.Pos("|")) != 0){
    l = l.SubString(p1+1, l.Length()-(p1+1));
    nRow++;
  }

  //Сформируем нужное число строк
  StringGrid2->RowCount = nRow+1; //+1 для строки-заголовка

  //Теперь, собственно, реальное заполнение этих строк
  l = L;
  nRow = 1; //1 - потому что индексирование строк начинается с 1
            //(учитывая, что нулевая строка -- заголовок)
  while((p1=l.Pos("|")) != 0){
    //Это не последняя строка
    StringGrid2->Cells[0][nRow] = nRow;
    StringGrid2->Cells[1][nRow] = l.SubString(0, p1-1);
    l = l.SubString(p1+1, l.Length()-p1);
    nRow++;
  }
  //Это последняя (или единственная подстрока)
  StringGrid2->Cells[0][nRow] = nRow;
  StringGrid2->Cells[1][nRow] = l;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StringGrid2SetEditText(TObject *Sender, int ACol,
      int ARow, const AnsiString Value)
{
   if(StringGrid2->EditorMode) return;

   AnsiString l = "";
   int tra = Label7->Caption.ToDouble();
   int pos = RadioGroup2->ItemIndex;

   int k;

   l = StringGrid2->Cells[1][1];
   for(k=2; k<StringGrid2->RowCount; k++){
     l = l + "|" + StringGrid2->Cells[1][k];
   }

   TRA[tra].Logic[pos] = l;

   ShowMessage("Изменения внесены в базу");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton7Click(TObject *Sender)
{
  if(Application->MessageBox(L"Конъюнкция будет удалена. Продолжить?",
                             L"Предупреждение", MB_YESNO) != IDYES) return;
```

```
  int i, j;
  int ARow = StringGrid2->Row;

  if(ARow == 0) return;

  for(i=ARow; i<StringGrid2->RowCount; i++)
    StringGrid2->Rows[i] = StringGrid2->Rows[i+1];

  StringGrid2->RowCount = StringGrid2->RowCount-1;
  StringGrid2SetEditText(Sender, 0, ARow, "");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TabSheet5->TabVisible = true;
  TabSheet4->TabVisible = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton11Click(TObject *Sender)
{

  GTime = 0; //Обнулим глобальное модельное время

  //Логика создания маркеров:
  //создаётся всегда по 1 маркеру в начальной позиции каждой подсети
  NMark = 0;
  for(int i=0; i<NPos; i++){
    if(POS[i].TypePos == 1){ //Признак начальной позиции какой-то подсети
      Mark[NMark].SubNet = POS[i].SubNet;
      Mark[NMark].indPos = POS[i].ind;  //Индекс позиции
      NMark++;
    }
  }

  if(NMark == 0){
    ShowMessage("В моделируемой сети нет ни одной начальной позции.\n"
                "Моделирование невозможно");
    return;
  }

  int NRun = Edit5->Text.ToInt(); //Количество запусков цикла моделирования
  if(NRun>10000) NRun = 10000;    //Проверка на максимальное число повторений
                                  //(определяется размерами массивов для сбора
                                  //статистики)
  int Run;                        //Текущий цикл моделирования

  if(NMark>1000){
    NMark = 1000;
    ShowMessage("В данной версии программы количество маркеров "
                "не должно превышать 1000");
  }


  EditingMode = false; //Редактирование завершено. Система перешла в режим
                       //моделирования. Состояние этой переменной определяет тип
                       //отображаемой информации:
                       //  - в режиме редактирования отображаются номера
                       //    позиций/переходов,
                       //  - в режиме моделирования -- количество маркеров
                       //    в позиции/переходе

  int DelayTime = Edit4->Text.ToInt(); //Время паузы между кадрами анимации
  bool Animation = CheckBox1->Checked; //Необходимость выдерживать эту паузу
```

```
int i, j, k, n;

nShot1 = 0; nShot2 = 0;
nWin1  = 0; nWin2  = 0;

for(Run = 0; Run < NRun; Run++){

  GTime = 0; //Обнулим глобальное модельное время

  NMark = 0;
  for(int i=0; i<NPos; i++){
    if(POS[i].TypePos == 1){ //Признак начальной позиции какой-то подсети
      Mark[NMark].SubNet = POS[i].SubNet;
      Mark[NMark].indPos = POS[i].ind;  //Индекс позиции
      NMark++;
    }
  }


  NMarkEnd = 0; //Количество маркеров в конечной позиции

  //Обнулим и перересуем все позиции и переходы
  for(i=0; i<NPos; i++){
    //Если позиция начальная -- в ней есть маркер
    if(POS[i].TypePos == 1) POSShape[i]->Label = 1;
    else                    POSShape[i]->Label = 0;
    POSShape[i]->Repaint();
  }
  for(i=0; i<NTra; i++){ TRAShape[i]->Label = 0; TRAShape[i]->Repaint();}

  for(i=0; i<NMark; i++){
    Mark[i].TimeProc = 0; //Общее время пребывания в позициях
    Mark[i].TimeWait = 0; //Общее время пребывания в переходах

    Mark[i].nPosEnt = 0;  //Число попаданий в позиции
    Mark[i].nTraEnt = 0;  //Число попаданий в переходы

    Mark[i].dT = 0; //Сгенерированное при попадании в позицию время пребывания
    Mark[i].inPos = true;
    Mark[i].indTra = 0;  //Индекс перехода
  }

  //Сбросим информацию во всех переходах
  for(i=0; i<NTra; i++)
    for(j=0; j<TRA[i].NArcIn; j++)TRA[i].nMarkers[j] = 0;

  int nArc;  //Вспомогательная переменная для хранения выбранной дуги
             //(индекса в массиве POS::ArcInd)

  //Пройдём по всем имеющимся в модели маркерам для инициализации их
  //первоначального времени пребывания в позиции
  for(i=0; i<NMark; i++){
    if(!Mark[i].inPos) continue; //Если текущий маркер по какой-то причине
                                 //находится не в позиции, а в переходе --
                                 //игнорируем его

    //Сгенерируем случайным образом, по какой дуге пойдёт маркер дальше
    nArc = POS[Mark[i].indPos].randArcOutNum();

    //Теперь определим время пребывания маркера в позиции
    Mark[i].dT = POS[Mark[i].indPos].randTime(nArc);
```

```
  //Обработаем статистику маркера по позиции
  //Учтём текущее время обработки в общем времени пребывания в позициях
  Mark[i].TimeProc += Mark[i].dT;
  //а также увеличим количество попаданий в позиции
  Mark[i].nPosEnt++;

  //И сформируем индекс перехода, в который по выбранной ранее дуге
  //попадёт маркер
  Mark[i].indTra = POS[Mark[i].indPos].ArcInd[nArc];

}

//Запускаем дискретный шаг моделирования
bool StopModeling = false;

int NMarkInPos; //Вспомогательная переменная, показывающая текущее
                //количество маркеров в позициях. Предназначена для
                //обнаружения случая, когда все маркеры собрались в
                //переходах и система "зависла"

float dTmin;    //Вспомогательная переменная для определения минимального
                //времени, оставшегося до завершения обработки какого-то
                //маркера

do{
  NMarkInPos = 0; //Подсчитаем количество маркеров, находящихся в позициях
                  //(для определения того, что система не "зависла")

  //Проходим по всем маркерам
  dTmin = -1;
  for(i=0; i<NMark; i++){
    //Если текущий маркер не в позиции, или в конечной позиции --
    //игнорируем его
    if(!Mark[i].inPos || POS[Mark[i].indPos].NArcOut == 0) continue;
    //Если это первый найденный в позиции маркер, то его время и будет
    //в этот момент минимальным из всех найденных
    if(dTmin == -1) dTmin = Mark[i].dT;
    else
    if(Mark[i].dT < dTmin) dTmin = Mark[i].dT;
    NMarkInPos++; //Увеличим количество маркеров, найденных в позициях
  }

  //Если маркеров в позициях нет -- система "зависла"
  //и следует прекратить моделирование
  if(NMarkInPos == 0) break;


  GTime += dTmin; //Дискретным образом изменяем глобальное модельное время

  //Уменьшаем время срабатывания всех маркеров относительно изменившегося
  //глобального модельного времени
  for(i=0; i<NMark; i++){
    //Если текущий маркер не в позиции, или в конечной позиции --
    //игнорируем его
    if(!Mark[i].inPos || POS[Mark[i].indPos].NArcOut == 0) continue;
    Mark[i].dT -= dTmin;
  }


  int indTra; //Локальная переменная для хранения индекса перехода,
              //с которым идёт работа

  //При определении минимального времени срабатывая не фиксируется индекс
```

```
//маркера, на котором достигается этот минимум, т.к. теоретически в
//дискретном времени может обнаружиться несколько маркеров с одинаковым
//временем срабатывания, тогда всех их нужно будет переместить из
//соответствующих позиций в соответствующие переходы, и для каждого
//определить условие срабатывания перехода, при положительном исходе
//которого снова переместить их уже в следующие позиции
for(i=0; i<NMark; i++){
  //Если маркер находится в переходе, или в конечной позиции,
  //или время срабатывания у него не равно текущему времени -- этот
  //маркер не рассматривается как кандидат на переход
  if(!Mark[i].inPos || POS[Mark[i].indPos].NArcOut == 0 || Mark[i].dT > 0)
    continue;

  //Если попали сюда, значит текущий i-ый маркер -- тот (или один из тех),
  //что закончили обработку в позиции и должны переместиться в переход

  //ВНИМАНИЕ! Если по каждой позиции будет собираться статистика
  //пребывания в ней маркеров, то в этом месте её нужно обрабатывать !!!

  //Вынимаем маркер из позиции и переправляем его в переход (индекс
  //перехода уже сформирован ранее)
  Mark[i].inPos = false;

  indTra = Mark[i].indTra;

  //Определим, в какой ячейке массива TRA::ArcInInd хранится ссылка
  //на позицию, из которой пришёл маркер
  for(j=0; j<TRA[indTra].NArcIn; j++)
    if(TRA[indTra].ArcInInd[j] == Mark[i].indPos) break;

  //В j хранится номер ячейки массива TRA::ArcInInd, в которой хранится
  //ссылка на позицию, из которой пришёл маркер. Он же является номером
  //ячейки массива TRA::nMarkers, в которой фиксируется факт поступления
  //ещё одного маркера из позиции.
  //Увеличим это число
  TRA[indTra].nMarkers[j]++;

  //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  POSShape[Mark[i].indPos]->Label--;
  POSShape[Mark[i].indPos]->Repaint();

  TRAShape[indTra]->Label++;
  TRAShape[indTra]->Repaint();

  if(Animation){
    Application->ProcessMessages();
    Sleep(DelayTime);
  }
  //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  //В самом маркере учтём очередное его попадание в новый переход
  Mark[i].nTraEnt++;
  Mark[i].dT = GTime; //Зафиксируем время попадания в переход (имея ввиду,
                      //что глобальное модельное время уже скорректировано)
                      //Это время понадобится для определения времени
                      //пребывания (ожидания) маркера в переходе

  //=============================================
  //  ПРОЦЕДУРА ПРОВЕРКИ
  //  СРАБАТЫВАНИЯ НЕПРИМИТИВНОГО ПЕРЕХОДА
  //
  //  Здесь считается, что в непримитивном переходе количество входов
  //  определяется количеством взаимодействующих подсетей.
```

```
//  Количество выходов также должно быть равно количеству
//  взаимодействующих подсетей. Все выходы срабатывают при одном
//  условии -- собрались маркеры из всех взаимодействующих подсетей.
//  Каждый маркер уходит одновременно в тот выход, который соответсвует
//  "его" подсети. Анализ конъюнкций, связанных с выходами,
//  не выполняется (за ненадобностью)
//=============================================

//Проверим -- если по каждой входной дуге поступил хотя бы один маркер,
//значит переход сработал, и на каждый соответствующий выход нужно
//эти маркеры перекинуть

bool TraAct = true;
for(k=0; k<TRA[indTra].NArcIn; k++){
  //Если в переходе отсутствуют маркеры из позиции, которая входит
  //в условие срабатывания -- переход не сработал, дальше можно
  //не анализировать
  if(TRA[indTra].nMarkers[k] == 0){TraAct = false; break;}
}


//Если переход сработал -- надо перемещать соответствующие маркеры
//в соответствующие выходные позиции соответствующих подсетей
if(TraAct){
   //Для срабатывания перехода нужно
   //1. Каждый маркер, находившийся в этом переходе, перекинуть
   //   на тот выход, который ведёт в вершину, относящуюся к той же
   //   подсети, что и маркер
   //2. Скорректировать статистику
   //3. Уменьшить количество маркеров в переходе по каждому
   //   из входов на 1


   //Найдём все маркеры, находящиеся в сработавшем переходе
   //и переместим их в соответствующие позиции соответствующих подсетей
   for(n=0; n<NMark; n++){
     //Найден подходящий маркер
     if(!Mark[n].inPos && Mark[n].indTra == indTra){
       //Определим, в какую подсеть нужно отправить маркер
       //Для каждого из имеющихся выходов из перехода проверим
       for(j=0; j<TRA[indTra].NArcOut; j++)
         if(POS[TRA[indTra].ArcOutInd[j]].SubNet == Mark[n].SubNet){
           Mark[n].indPos = TRA[indTra].ArcOutInd[j];

           //ВНИМАНИЕ! Если стурктура сети нарушена, и
           //из непримитивного перехода нет выхода в нужную подсеть,
           //макрер останется в переходе.

           //Маркер теперь не в переходе, а в позиции
           Mark[n].inPos = true;

           //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
           POSShape[Mark[n].indPos]->Label++;
           POSShape[Mark[n].indPos]->Repaint();

           TRAShape[Mark[n].indTra]->Label--;
           TRAShape[Mark[n].indTra]->Repaint();

           if(TRAShape[Mark[n].indTra]->Label<0)
             ShowMessage("Авария");

           //Вызовем функцию анимации, соответствующие попаданию в
           //данную позицию. Сначала отобразим над соответствующей
```

```
//позицией в окне структуры сети текст из свойства Hint,
//который поясняет назначение этой позиции
if(Animation){
  THintWindow* Hint;
  Hint = RevealHint(POSShape[Mark[n].indPos]);
  Sleep(2000);
  RemoveHint (Hint);
}

 //Если требуемая функция загружена из библиотеки,
 //то вызовем её
 if(pf[POS[Mark[n].indPos].Action])
   (pf[POS[Mark[n].indPos].Action])(Form1, Animation);

 if(Animation){
   Application->ProcessMessages();
   Sleep(DelayTime);
 }
 //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 //Рассчитаем статистику по пребыванию маркера в переходе
 Mark[n].TimeWait += (GTime-Mark[n].dT);


 //Проверим, не достиг ли маркер конечной позиции
 //(признак -- тип позиции равен 2)
 if(POS[Mark[n].indPos].TypePos == 2){
   NMarkEnd++;

   //Признак того, что завершать моделирование необходимо
   //при первом же случае достижения любым маркером
   //конечной позиции
   if(RadioGroup3->ItemIndex == 1){
     if(Animation){
       ShowMessage(
         "Один из маркеров достиг конечной позиции"
                  );
     }

     //Скорректирем количество побед
     //И время, потребовавшееся для завершения дуэли
     if(Mark[n].Tag == 0){
       Time1[nWin1] = Mark[n].TimeProc + Mark[n].TimeWait;
       nWin1++;
     }
     else{
       Time2[nWin2] = Mark[n].TimeProc + Mark[n].TimeWait;
       nWin2++;
     }

     StopModeling = true;
   }

   if(NMarkEnd == NMark){
     ShowMessage("Все маркеры достигли конечной позиции");
     StopModeling = true;
   }
 }


 //Сгенерируем случайным образом, по какой дуге
 //пойдёт маркер дальше
 nArc = POS[Mark[n].indPos].randArcOutNum();
```

```
                    //Теперь определим время пребывания маркера в позиции
                    Mark[n].dT = POS[Mark[n].indPos].randTime(nArc);

                    //Обработаем статистику маркера по позиции

                    //Учтём текущее время обработки в общем времени
                    //пребывания в позициях
                    Mark[n].TimeProc += Mark[n].dT;
                    //а также увеличим количество попаданий в позиции
                    Mark[n].nPosEnt++;

                    //И сформируем индекс перехода, в который по выбранной
                    //ранее дуге попадёт маркер
                    Mark[n].indTra = POS[Mark[n].indPos].ArcInd[nArc];
                   ////////////////
                   break;
                 }
              }
            }
          //Поскольку все маркеры из сработавшего перехода выведены,
          //необходимо скорректировать их количество в свойствах самого перехода
          for(k=0; k<TRA[indTra].NArcIn; k++) TRA[indTra].nMarkers[k]--;
          //break;
        }
    }
    }
    while(!StopModeling);


    if(NMarkInPos == 0) ShowMessage("Все маркеры сосредоточились в переходах,"
                          " которые не могут сработать.\nСистема 'зависла'");

    float AveTimeInPos = 0; //Среднее время пребывания в позиции
    int   NPosEnt = 0;      //Количество входов в позиции
    float AveNPos = 0;      //Среднее длина тракетории (количество позиций)

    float AveTimeInTra = 0;

    for(i=0; i<NMark; i++){
      AveNPos += Mark[i].nPosEnt;
    }

    if(Animation)
      ShowMessage("Средняя длина траектории: " +
                                    FormatFloat("0.00", AveNPos/NMark));
  }

  float aveTime1, aveTime2;

  aveTime1 = 0;
  if(nWin1>0){
    for(i=0; i<nWin1; i++) aveTime1 += Time1[i];
    aveTime1 /= nWin1;
  }

  aveTime2 = 0;
  if(nWin2>0){
    for(i=0; i<nWin2; i++) aveTime2 += Time2[i];
    aveTime2 /= nWin2;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton12Click(TObject *Sender)
```

```
{

    PMShape *sh = new PMShape(this);
    sh->Parent = ScrollBox1;
    sh->Shape  = stCircle;
    sh->Brush->Color = TColor(clRed);
    sh->Height = 21;  //2*10+1
    sh->Width  = 21;

    sh->Top  = 100 - ScrollBox1->VertScrollBar->Position;
    sh->Left = 100 - ScrollBox1->HorzScrollBar->Position;

    sh->OnMouseDown = POSMouseDown;
    sh->OnMouseUp   = POSMouseUp;
    sh->OnMouseMove = POSMouseMove;

    EditingMode = false;

    sh->Label = 10;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton13Click(TObject *Sender)
{
  Form2->ShowModal();
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------
// Демонстрирует всплывающую подсказку для определенного элемента
// управления (Control), возвращает ссылку на hint-объект,
// поэтому в дальнейшем подсказка может быть спрятана вызовом
// RemoveHint (смотри ниже).
//---------------------------------------------------------------------
THintWindow* RevealHint (TControl* Control)
{
  THintWindow* Result;
  AnsiString ShortHint;

  char AShortHint[256];
  TPoint HintPos;
  TRect HintBox;

  //Создаем окно
  Result = new THintWindow (Control);

  //Получаем первую часть подсказки до '|'
  ShortHint = GetShortHint(Control->Hint);

  //Вычисляем месторасположение и размер окна подсказки
  HintPos = Control->ClientOrigin;

  //Позиционирование подсказку выше элемента управления.
  //Это может быть изменено, если по какой-то причине необходима
  //другая позиция окна с подсказкой.
  HintPos.y -= (Control->Height + 6);

  HintBox = Bounds(0, 0, Screen->Width, 0);
  DrawText(Result->Canvas->Handle, StrPCopy(AShortHint, ShortHint), -1,
           &HintBox, DT_CALCRECT | DT_LEFT | DT_WORDBREAK | DT_NOPREFIX);

  OffsetRect(HintBox, HintPos.x, HintPos.y);

  HintBox.Right += 6;
```

```
  HintBox.Bottom += 2;

  // Теперь показываем окно
  Result->ActivateHint(HintBox, ShortHint);
  Result->Update();
  return Result;
} // RevealHint


//---------------------------------------------------------------------
// Освобождаем дескриптор окна всплывающей подсказки, выведенной
// предыдущим RevealHint.
//---------------------------------------------------------------------
void RemoveHint (THintWindow* Hint)
{
  Hint->ReleaseHandle();
  Hint->Free();
  Hint = NULL;
}
//---------------------------------------------------------------------------
```

П 5.2. Файл Unit1.dfm

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = #1057#1088#1077#1076#1072'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1103'
'#1089#1077#1090#1077#1081' '#1055#1077#1090#1088#1080'-
'#1052#1072#1088#1082#1086#1074#1072
  ClientHeight = 900
  ClientWidth = 1496
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  Position = poDesigned
  ShowHint = True
  WindowState = wsMaximized
  OnCreate = FormCreate
  PixelsPerInch = 96
  TextHeight = 13
  object Panel1: TPanel
    Left = 0
    Top = 0
    Width = 217
    Height = 900
    Align = alLeft
    TabOrder = 0
    object PageControl1: TPageControl
      Left = 1
      Top = 392
      Width = 215
      Height = 507
      ActivePage = TabSheet1
      Align = alBottom
      TabOrder = 0
```

```
      object TabSheet1: TTabSheet
        Caption = #1055#1086#1079#1080#1094#1080#1103
        object Label1: TLabel
          Left = 3
          Top = 2
          Width = 58
          Height = 13
          Caption = #1055#1086#1079#1080#1094#1080#1103' '#8470
        end
        object Label4: TLabel
          Left = 10
          Top = 332
          Width = 82
          Height = 13
          Caption = #1052#1072#1090'. '#1086#1078#1080#1076#1072#1085#1080#1077':'
        end
        object Label5: TLabel
          Left = 10
          Top = 356
          Width = 26
          Height = 13
          Caption = #1057#1050#1054':'
        end
        object Label6: TLabel
          Left = 3
          Top = 141
          Width = 162
          Height = 13
          Caption = #1042#1077#1088#1086#1103#1090#1085#1086#1089#1090#1080'
'#1074#1099#1093#1086#1076#1072' '#1074' '#1087#1077#1088#1077#1093#1086#1076
        end
        object SpeedButton8: TSpeedButton
          Left = 65
          Top = 440
          Width = 65
          Height = 22
          Caption = #1042#1074#1086#1076
          OnClick = SpeedButton8Click
        end
        object Label8: TLabel
          Left = 86
          Top = 3
          Width = 12
          Height = 13
          Caption = '__'
        end
        object Label11: TLabel
          Left = 10
          Top = 382
          Width = 53
          Height = 13
          Caption = #1044#1077#1081#1089#1090#1074#1080#1077':'
        end
        object Label13: TLabel
          Left = 3
          Top = 119
          Width = 47
          Height = 13
          Caption = #1055#1086#1076#1089#1077#1090#1100':'
        end
        object RadioGroup1: TRadioGroup
          Left = 3
          Top = 256
```

```
          Width = 190
          Height = 65
          Caption = ' '#1047#1072#1082#1086#1085'
'#1088#1072#1089#1087#1088#1077#1076#1077#1083#1077#1085#1080#1103' '
          ItemIndex = 0
          Items.Strings = (
            #1056#1072#1074#1085#1086#1084#1077#1088#1085#1099#1081
            #1053#1086#1088#1084#1072#1083#1100#1085#1099#1081)
          TabOrder = 0
        end
        object StringGrid1: TStringGrid
          Left = 3
          Top = 160
          Width = 190
          Height = 89
          ColCount = 2
          DefaultColWidth = 32
          DefaultRowHeight = 15
          RowCount = 2
          Options = [goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine,
goRangeSelect, goEditing]
          TabOrder = 1
          OnClick = StringGrid1Click
          ColWidths = (
            32
            151)
        end
        object Edit1: TEdit
          Left = 113
          Top = 328
          Width = 80
          Height = 21
          TabOrder = 2
          Text = '0,0'
          OnKeyPress = Edit1KeyPress
        end
        object Edit2: TEdit
          Left = 113
          Top = 352
          Width = 80
          Height = 21
          TabOrder = 3
          Text = '1,0'
          OnKeyPress = Edit1KeyPress
        end
        object Edit6: TEdit
          Left = 10
          Top = 406
          Width = 183
          Height = 21
          Hint = #1054#1087#1080#1089#1072#1085#1080#1077'
'#1076#1077#1081#1089#1090#1074#1080#1103
          TabOrder = 4
        end
        object ComboBox1: TComboBox
          Left = 113
          Top = 379
          Width = 80
          Height = 21
          Hint =
            #1048#1085#1076#1077#1082#1089' '#1092#1091#1085#1082#1094#1080#1080'
'#1080#1079' '#1074#1085#1077#1096#1085#1077#1081'
'#1073#1080#1073#1083#1080#1086#1090#1077#1082#1080',
```

```
'#1072#1085#1080#1084#1080#1088#1091#1102#1097#1077#1081'
'#1076#1077#1081#1089#1090#1074#1080#1077', '#1089#1074#1103#1079 +
          #1072#1085#1085#1086#1077' '#1089' '#1076#1072#1085#1085#1086#1081'
'#1087#1086#1079#1080#1094#1080#1077#1081
        Style = csDropDownList
        TabOrder = 5
        Items.Strings = (
          '0')
      end
      object Edit7: TEdit
        Left = 56
        Top = 116
        Width = 137
        Height = 21
        TabOrder = 6
        Text = '1'
      end
      object RadioGroup4: TRadioGroup
        Left = 0
        Top = 22
        Width = 193
        Height = 88
        Caption = ' '#1058#1080#1087' '#1087#1086#1079#1080#1094#1080#1080' '
        ItemIndex = 0
        Items.Strings = (
          #1087#1088#1086#1084#1077#1078#1091#1090#1086#1095#1085#1072#1103
          #1085#1072#1095#1072#1083#1100#1085#1072#1103
          #1082#1086#1085#1077#1095#1085#1072#1103)
        TabOrder = 7
      end
    end
    object TabSheet2: TTabSheet
      Caption = #1055#1077#1088#1077#1093#1086#1076
      ImageIndex = 1
      object Label2: TLabel
        Left = 3
        Top = 8
        Width = 60
        Height = 13
        Caption = #1055#1077#1088#1077#1093#1086#1076' '#8470
      end
      object Label3: TLabel
        Left = 8
        Top = 112
        Width = 118
        Height = 13
        Caption = #1059#1089#1083#1086#1074#1080#1077'
'#1089#1088#1072#1073#1072#1090#1099#1074#1072#1085#1080#1103
      end
      object SpeedButton6: TSpeedButton
        Left = 174
        Top = 144
        Width = 23
        Height = 22
        Hint = #1044#1086#1073#1072#1074#1080#1090#1100'
'#1082#1086#1085#1098#1102#1085#1082#1094#1080#1102
        Glyph.Data = {
          76010000424D760100000000000076000000280000002000000010000000100
          0400000000000000010000130B0000130B0000100000000000000000000000000
          8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
          FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00333333333333
          33333333FF33333333FF333993333333300033377F33333337773339933333333
          300033F77FFF3333377739999993333333333777777F3333333F399999933333
```

```
            3300377777733333337733399333333330033377F33333333773339933333333
            3333333773333333333F33333333333333300333333333F33333773333333C3333
            330033333337FF3333773333333CC333333333FFFFF77FFF3FF33CCCCCCCCCC3
            993337777777777F77F33CCCCCCCCCC399333777777777773773333333333CC333
            333333333337733333FF3333333C333330003333333733333777333333333333
            3000333333333333377733333333333333333333333333333333}
          NumGlyphs = 2
          OnClick = SpeedButton6Click
        end
        object SpeedButton7: TSpeedButton
          Left = 174
          Top = 172
          Width = 23
          Height = 22
          Hint = #1059#1076#1072#1083#1080#1090#1100'
'#1082#1086#1085#1098#1102#1085#1082#1094#1080#1102
          Glyph.Data = {
            76010000424D760100000000000076000000280000002000000010000000100
            0400000000000001000130B0000130B0000100000000000000000000000000000
            8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
            FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00333333333333
            3333333333333333FF3333333333333000333333333333377733333333333333
            300033FFFFFF3333377739999993333333333777777F3333333F399999933333
            3300377777733333337733333333333333300333333333333337733333333333
            3333333333333333333F33333333333333300333333F33333333773333C3333333
            330033337F3333333377333CC3333333333333F77FFFFFFF3FF33CCCCCCCCCC3
            993337777777777F77F33CCCCCCCCCC399333777777777773773333CC3333333
            333333377F33333333FF3333C333333330003333733333333777333333333333
            3000333333333333377733333333333333333333333333333333333333333333}
          NumGlyphs = 2
          OnClick = SpeedButton7Click
        end
        object Label7: TLabel
          Left = 88
          Top = 8
          Width = 12
          Height = 13
          Caption = '__'
        end
        object StringGrid2: TStringGrid
          Left = 3
          Top = 128
          Width = 159
          Height = 113
          ColCount = 2
          DefaultColWidth = 32
          DefaultRowHeight = 15
          RowCount = 2
          Options = [goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine,
goRangeSelect, goEditing]
          TabOrder = 0
          OnSetEditText = StringGrid2SetEditText
          ColWidths = (
            32
            114)
        end
        object RadioGroup2: TRadioGroup
          Left = 3
          Top = 27
          Width = 194
          Height = 70
          Caption = ' '#1055#1077#1088#1077#1093#1086#1076' '#1074'
'#1087#1086#1079#1080#1094#1080#1102' '
```

```
          Columns = 2
          ItemIndex = 0
          Items.Strings = (
            '1'
            '2'
            '3'
            '4')
          TabOrder = 1
          OnClick = RadioGroup2Click
        end
      end
    end
    object PageControl2: TPageControl
      Left = 1
      Top = 1
      Width = 215
      Height = 195
      ActivePage = TabSheet3
      Align = alTop
      TabOrder = 1
      object TabSheet3: TTabSheet
        Caption = #1057#1090#1088#1091#1082#1090#1091#1088#1072'
'#1089#1077#1090#1080
        ExplicitHeight = 188
        object SpeedButton4: TSpeedButton
          Left = 6
          Top = 16
          Width = 23
          Height = 22
          Hint = #1047#1072#1075#1088#1091#1079#1080#1090#1100'
'#1089#1090#1088#1091#1082#1090#1091#1088#1091' '#1089#1077#1090#1080
          Glyph.Data = {
            76010000424D760100000000000076000000280000002000000010000000100
            0400000000000001000012B0000120B0000120B000010000000000000000000000000000
            8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
            FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00555555555555
            5555555555555555555555555555555555555555555555555555555555555555
            5555555555555555555555555555555555555FFFFFFFFFF555550000000000
            5555557777777775F55500B8B8B8B8B05555775F555555575F550F0B8B8B8B8
            B05557F75F555555575F50BF0B8B8B8B8B0557F575FFFFFFFF7F50FBF0000000
            000557F557777777775550BFBFBFBFB0555557F555555557F55550FBFBFBFBF0
            555557F555555FF7555550BFBFBF00055555575F555577755555550BFBF05555
            55555575FFF755555555557000075555555555577775555555555555555555555
            5555555555555555555555555555555555555555555555555555555555555555}
          NumGlyphs = 2
          OnClick = SpeedButton4Click
        end
        object SpeedButton5: TSpeedButton
          Left = 64
          Top = 16
          Width = 23
          Height = 22
          Hint = #1057#1086#1093#1088#1072#1085#1080#1090#1100'
'#1089#1090#1088#1091#1082#1090#1091#1088#1091' '#1089#1077#1090#1080
          Glyph.Data = {
            76010000424D760100000000000076000000280000002000000010000000100
            0400000000000001000013B0000130B0000130B000010000000000000000000000000000
            8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
            FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00333333330070
            7700333333337777777733333333008088003333333377F73377333333330088
            88003333333377FFFF7733333333000000003FFFFFFF7777777000000000000
            0000777777777777770FFFFFFF0FFFFFF07F3333337F3333370FFFFFFF0FFF
            FFF07F3FF3FF7FFFFFF70F00F0080CCC9CC07F773773777777770FFFFFFFF039
```

```
            99337F3FFFF3F7F777F30F0000F0F09999937F7777373777777F0FFFFFFFF999
            99997F3FF3FFF77777770F00F000003999337F773777773777F30FFFF0FF0339
            99337F3FF7F3733777F30F08F0F0337999337F7737F73F7777330FFFF0039999
            93337FFFF7737777773330000003333333333777777733333333333}
          NumGlyphs = 2
          OnClick = SpeedButton5Click
        end
        object SpeedButton1: TSpeedButton
          Left = 112
          Top = 16
          Width = 23
          Height = 22
          Hint = #1044#1086#1073#1072#1074#1080#1090#1100'
'#1087#1086#1079#1080#1094#1080#1102
          Glyph.Data = {
            36040000424D3604000000000000360000002800000010000000100000000100
            2000000000000000400000130B0000130B00000000000000000000FFFFFF00FFFF
            FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF00FFFFFF0041924E233D8F497D3A8C44DB368940F332873CF32F84
            37DB2C81337D287F3023FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF0049995853459653E6419950FF7DC28FFF96D0A6FF96CFA6FF78BE
            89FF368D42FF2C8134E6297F3053FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00519F61534D9C5DF464B478FFA8DBB5FF87CC98FF66BC7DFF64BA7CFF86CB
            98FFA5D9B4FF58AA6BFF2C8134F4297F3053FFFFFF00FFFFFF00FFFFFF0059A6
            6B2256A366E56AB97DFFA8DBB2FF60BC77FF5CBA73FF59B870FF59B56FFF58B5
            6FFF5BB774FFA5D9B3FF5AAA6CFF2C8234E5297F3022FFFFFF00FFFFFF005DA9
            707E53AB68FFAADDB4FF64C179FF5FBE71FF60BC77FFFFFFFFFFFFFFFFFFFF59B8
            70FF58B56EFF5CB774FFA6DAB4FF388F43FF2C82347EFFFFFF00FFFFFF0061AC
            75DB8ACC98FF89D396FF6BC67AFF63C170FF55AB65FFFFFFFFFFFFFFFFFFFF59B8
            70FF59B870FF5BB972FF85CC97FF7BBE8DFF308539DBFFFFFF00FFFFFF0065AF
            7AF6A9DDB3FF7DCF8AFF75CC81FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFFF59B870FF67BE7DFF9CD4ABFF34883DF6FFFFFF00FFFFFF0069B2
            7EF6B6E2BEFF8BD597FF7AC986FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFFF59B870FF69C17EFF9DD4AAFF388B42F6FFFFFF00FFFFFF006DB5
            83DBACDDB6FFA6DFAFFF81CB8CFF7CC986FF6EBD79FFFFFFFFFFFFFFFFFFFF5BAC
            6AFF60BC77FF5CBA73FF8BD199FF80C592FF3C8E47DBFFFFFF00FFFFFF0070B8
            877E85C797FFD2EED7FF95D9A0FF8AD394FF7FC889FFFFFFFFFFFFFFFFFFFF79CD
            85FF6BC37CFF6FC77EFFACDFB5FF459E57FF40914C7EFFFFFF00FFFFFF0073BA
            8A2270B887E5AADAB7FFD8F1DCFF92D89DFF88CD93FF84CC8EFF8BD496FF8AD4
            95FF83D28EFFAFE0B7FF6BB97DFF489856E544945122FFFFFF00FFFFFF00FFFF
            FF0073BB8B5370B887F4AFDCBBFFDCF2E0FFB6E4BDFF9BDBA5FF96D9A0FFA5DF
            AFFFC0E8C5FF79C28AFF509E5FF44C9B5B53FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF0073BB8B5371B887E694CEA4FFC3E6CBFFCFEBD4FFC9E9CEFFAFDD
            B8FF6DB97FFF58A569E654A16553FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF00FFFFFF0074BB8B2371B9887D6EB684DB6AB380F367B17CF363AE
            77DB60AB737D5CA86E23FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFF
            FF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00FFFFFF00
          }
          OnClick = SpeedButton1Click
        end
        object SpeedButton2: TSpeedButton
          Left = 141
          Top = 16
          Width = 23
          Height = 22
          Hint = #1044#1086#1073#1072#1074#1080#1090#1100'
'#1087#1077#1088#1077#1093#1086#1076
          Glyph.Data = {
            36030000424D3603000000000000360000002800000010000000100000000100
            1800000000000003000000000000000000000000000000000000000000000000
            0000000000000000000000000000000000000000000000000D381D176533196B
            371765330D381D0000000000000000000000000000000000000000000000000000
```

```
    00000000000C371C288C5364BA8D95D2B264BA8D288C530D3A1E000000000000
    DA946BD99268D89066D88E64D68C62D58961D5895F216D3A62BA8B60BA87FFFF
    FF60B98767BC8F1867350000000000000DC986FE3AF8FE3AD8DE2AB8BE1AA8AE1
    A888E0A787317B4C9CD4B6FFFFFFFFFFFFFFFFFF95D2B2196B37000000000000
    DD9B73E4B192E4AF91E3AE8FE3AC8DE1AB8BE1A98949885E90D3B192D6B1FFFF
    FF65BC8C67BC8F1867350000000000000DE9F77E5B495E4B393E4B192E3AF90E3
    AE8EE2AC8D999C7961AB8195D4B4BAE6D06ABB8F2D8F570D3A1E000000000000
    E1A27BE6B798E6B596E5B494E4B292E4B191E3AF8FE3AD8D9FA07E5F956F4F8E
    66488459163A2300000000000000000000E1A67FE8BA9BE7B899E6B697E6B596E5
    B394E4B192E4AF91E3AE8FE3AD8DE2AB8BD88E66000000000000000000000000000
    E3AA81E9BC9EE8BB9CE8B99AE7B899E6B697E6B495E4B394E4B192E3AF90E3AE
    8FD9926A000000000000000000000000E3AC85EABFA0EABE9FE8BC9DE8BA9CE8
    B99AE6B798E6B696E5B494E4B393E4B191DA966C000000000000000000000000
    E5AF86EBC1A2EAC0A2EABEA0E9BD9EE8BC9DE8BA9BE7B899E6B698E6B596E5B3
    94DC9A70000000000000000000000000E5B289EBC3A5EBC2A3EBC0A2EABFA1EA
    BEA0E9BD9EE8BB9DE8BA9BE7B899E6B697DE9D750000000000000000000000000
    E6B38CECC4A6EBC4A5EBC2A4EBC2A3EAC0A2EABFA0EABE9FE8BC9EE8BB9CE8B9
    9ADEA27800000000000000000000000E6B38CE6B38CE6B38CE5B28AE5B289E5
    AF86E5AE86E3AC85E3AB83E3A980E1A77FE1A57C0000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000}
  OnClick = SpeedButton2Click
end
object SpeedButton3: TSpeedButton
  Left = 170
  Top = 16
  Width = 23
  Height = 22
  Hint = #1044#1086#1073#1072#1074#1080#1090#1100' '#1076#1091#1075#1091
  Glyph.Data = {
    36030000424D360300000000000036000000280000001000000010000000100
    18000000000000030000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000D381D176533196B
    371765330D381D000000000000000000000000000000000000000000000000000
    00000000000C371C288C5364BA8D95D2B264BA8D288C530D3A1E57341CAA6636
    53311A000000000000000000000000000000000001A683662BA8B60BA87FFFF
    FF60B98767BC8F186735B26B39D1AC93B26B38000000000000000000000000000
    0000000000317B4C9CD4B6FFFFFFFFFFFFFFFFFF95D2B2196B375B371EB86F3C
    B96F3B5A351C0000000000000000221DA32D25E942825E90D3B192D6B1FFFF
    FF65BC8C67BC8F1867350000000000000130B069C5E327F4C280604020B0A2D36
    30ED9496FE3F5BA861AB8195D4B4BAE6D06ABB8F2D8F570D3A1E000000000000
    0000000402016B49685C48C03D37DB322DC93731EC312BE42C493D538B674F8E
    663C7C54163A230000004142AA5A5AF44847CB4847D63E3CC1523B6AAE683A30
    1D0F000000221E90241FA50000000000004B2B1595562A4829146364F6B2B5FE
    5B5BF61212340000000000321E10BC713CB56C3A593521372FC547317B7847
    24AD6532C0A18C94552A4749AB6364F64243AA0000000000000000000000000B6
    6E3BD0B19CBD9F708A55483E35DE3F29485F381C9B5B2E4C2C160000000000000
    0000000000000000000000000000005B371EB56D3B58351C00000014124C3530
    D60404130000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000002321852B27AF000002000000000000000000
    0000000000000000000000000000000000000000000000000000000000000101
    053834D13834E11816690000000000000000000000000000000000000000000000
    0000000000000000000000000000003E3BD89799FC3632D4000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0021206B3F3CD51D1B6A0000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000}
  OnClick = SpeedButton3Click
end
object SpeedButton10: TSpeedButton
  Left = 35
  Top = 16
  Width = 23
```

```
          Height = 22
          Hint = #1047#1072#1075#1088#1091#1079#1080#1090#1100'
'#1089#1090#1088#1091#1082#1090#1091#1088#1091' '#1080'
'#1087#1072#1088#1072#1084#1077#1090#1088#1099' '#1089#1077#1090#1080
          Glyph.Data = {
            76010000424D760100000000000076000000280000002000000010000000100
            0400000000000001000012 0B0000120B0000100000000000000000000000000
            8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
            FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00555555555555
            55555555FFFFFFFFFF55555000000000555555577777777775F55500B8B8B8B8
            B05555775F555555575F550F0B8B8B8B8B05557F75F555555575550BF0B8B8B8
            B8B0557F575FFFFFFFF7550FBF0000000000557F557777777777500BFBFBFBFB
            0555577F555555557F550B0FBFBFBFBF05557F7F555555FF75550F0BFBFBF000
            55557F75F555577755550BF0BFBF0B0555557F575FFF757F55550FB700007F05
            55557F557777557F55550BFBFBFBFB0555557F555555557F55550FBFBFBFBF05
            55557FFFFFFFFF7555550000000000555555577777777775555555 0FBFB055555
            5555575FFF7555555555570000755555555555577775555555555}
          NumGlyphs = 2
          OnClick = SpeedButton4Click
        end
        object SpeedButton12: TSpeedButton
          Left = 152
          Top = 56
          Width = 23
          Height = 22
          Visible = False
          OnClick = SpeedButton12Click
        end
        object Button3: TButton
          Left = 55
          Top = 56
          Width = 75
          Height = 25
          Hint = #1055#1077#1088#1077#1081#1090#1080' '#1074'
'#1088#1077#1078#1080#1084'
'#1088#1077#1076#1072#1082#1090#1080#1088#1086#1074#1072#1085#1080#1103'
'#1087#1072#1088#1072#1084#1077#1090#1088#1086#1074
          Caption = #1055#1072#1088#1072#1084#1077#1090#1088#1099
          TabOrder = 0
          OnClick = Button3Click
        end
      end
      object TabSheet4: TTabSheet
        Caption = #1055#1072#1088#1072#1084#1077#1090#1088#1099' '#1101#1083'-
'#1090#1086#1074
        ImageIndex = 1
        ExplicitHeight = 188
        object SpeedButton9: TSpeedButton
          Left = 14
          Top = 16
          Width = 23
          Height = 22
          Hint = #1057#1086#1093#1088#1072#1085#1080#1090#1100'
'#1089#1090#1088#1091#1082#1090#1091#1088#1091' '#1080'
'#1087#1072#1088#1072#1084#1077#1090#1088#1099' '#1089#1077#1090#1080
          Glyph.Data = {
            76010000424D760100000000000076000000280000002000000010000000100
            0400000000000001000013 0B0000130B0000100000000000000000000000000
            8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
            FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00333333330070
            7700333333337777777733333333008088003333333377F73377333333330088
            88003333333377FFFF7733333333000000003FFFFFFF7777777000000000000
            0000777777777777777 0FFFFFFF0FFFFFF07F3333337F3333370FFFFFFF0FFF
```

```
            FFF07F3FF3FF7FFFFFF70F00F0080CCC9CC07F773773777777770FFFFFFFF039
            99337F3FFFF3F7F777F30F0000F0F09999937F7777373777777F0FFFFFFFF999
            99997F3FF3FFF77777770F00F000003999337F773777773777F30FFFF0FF0339
            99337F3FF7F3733777F30F08F0F0337999337F7737F73F7777330FFFF0039999
            93337FFFF773777777333000000333333333377777733333333333}
          NumGlyphs = 2
          OnClick = SpeedButton5Click
        end
        object Button2: TButton
          Left = 13
          Top = 49
          Width = 75
          Height = 25
          Hint = #1042#1077#1088#1085#1091#1090#1100#1089#1103' '#1074'
'#1088#1077#1078#1080#1084'
'#1088#1077#1076#1072#1082#1090#1080#1088#1086#1074#1072#1085#1080#1103'
'#1089#1090#1088#1091#1082#1090#1091#1088#1099
          Caption = #1057#1090#1088#1091#1082#1090#1091#1088#1072
          TabOrder = 0
          OnClick = Button2Click
        end
        object Button1: TButton
          Left = 113
          Top = 49
          Width = 75
          Height = 25
          Hint = #1055#1077#1088#1077#1081#1090#1080' '#1074'
'#1088#1077#1078#1080#1084'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1103'
'#1088#1072#1073#1086#1090#1099' '#1089#1080#1089#1090#1077#1084#1099
          Caption = #1052#1086#1076#1077#1083#1100
          TabOrder = 1
          OnClick = Button1Click
        end
      end
      object TabSheet5: TTabSheet
        Caption =
#1052#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1077
        ImageIndex = 2
        ExplicitHeight = 171
        object SpeedButton11: TSpeedButton
          Left = 118
          Top = 70
          Width = 23
          Height = 22
          Hint = #1047#1072#1087#1091#1089#1090#1080#1090#1100'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1077
          Glyph.Data = {
            36030000424D360300000000000036000000280000001000000010000000100
            1800000000000003000012B0000120B0000000000000000000040204040204
            0402040402040402040402040402040402040402040402040402040402040402
            0404020404020404020404020404020404020404020404020404020404020404
            0204040204040204040204040204040204040204040204040204040204040204
            0402040402040402040402040402040402040402040402040402040402040404CA
            8404020404020404020404045E34048644046E3C0402040402040402040402040404
            0204040204043A1C04864404864404DEA404C284040204040204048644O4DE9C
            04DE9C04DA94045A34040204040204040204049654O4DA9C04D69C04D69C04D6
            9C04D69C04BE8404020404864404DE9C04DA9404DE9C04E29C047A3C04020404
            A66404D29C04D29C04D29404D69C04D29404D29C04BE7C040204045E34049254
            24C28C14DAA404D69C04E2A404A26404CE9C04CE9C3CDEBC049E640486446CEA
            CC04BA7C040204040204040204040204040204040414AA741CE2AC049E6404CE9C04
            CA9C4CE2BC045A2C04020404020404BA8404020404020404020404020404020404
            0402040402040049E6404CA9C04CA9C54E2C404823C040204040204040204040402
```

```
            04040204040204040204040204040204040204A26C04CA9C04C69C64E2CC04
            9A5C04D2A4047A4404020404020404045E3404020404020404020404020404042214
            04A26C04C69C04C29C74E2D4048E4C24D2AC04C69C04CEA404763C04020404D2
            AC045A2C04020404020404048A4404C6A404BE9C04BE9C8CE6DC04562C0402042C
            AA7C2CCEAC04C29C04C69C04CAA404C69C04CA9C045A2C040204048A4404BA9C
            04BE9C94E6DC04562C0402040402040402042CAA7C54D6BC04BE9C04BE9C04C2
            9C04C29C9CEEE4040204048A44A4EAE434AE7C044624040204040204040204 04
            020404020404048A446CCAB4A4EADC9CEEE4ACEEE4045A2C040204040204040204
            0402040402040402040402040402040402040402040402040402040402040402 04BCF2
            F404562C04020404020404020404020404020404020404020404020404020404
            020404020404020404020404020404045E34040204040204040204}
          OnClick = SpeedButton11Click
        end
        object Label10: TLabel
          Left = 8
          Top = 6
          Width = 102
          Height = 13
          Caption = #1047#1072#1076#1077#1088#1078#1082#1072'
'#1072#1085#1080#1084#1072#1094#1080#1080
        end
        object SpeedButton13: TSpeedButton
          Left = 147
          Top = 70
          Width = 23
          Height = 22
          Hint = #1057#1090#1072#1090#1080#1089#1090#1080#1082#1072'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1103
          Glyph.Data = {
            36030000424D3603000000000000360000002800000010000000100000000100
            1800000000000003000000000000000000000000000000000000000000000000
            0C0C0C0B0B0B0B0B0B0B0B0B0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0909
            09090909000000000000000000030303FCFCFCFBFBFBF9F9F9F9F9F9F7F7F7F7
            F7F7F7F7F7F7F5F5F5F5F5F5F5F5F5F5F5F5FCFCFC090909000000000000030303
            FCFCFCBFBBB7BFB0A2C0B0A2BCB7AE9FB3A79EB3A3A6B7B1A1B0C09EAFBFB1BC
            C4F9F9F9090909000000000000030303FCFCFCC3A890E67C2AE67C2DABA0631F
            923F198E2F3D9B7A1F78E61974E6609EE1FBFBFB090909000000000000040404
            FCFCFCBDA089F6690CFD6A0CA28F550C862400850F348F6A0B66F30060F85EA0
            F5FCFCFC0A0A0A000000000000040404FCFCFCCCA98FFD6609FD6408B09C580C
            892200830B34976F0B65FC005BFD5D9EF6FCFCFC0A0A0A000000000000040404
            FCFCFCB79A85F35904FD5B04A1834F0B7C1A007C063284600756F30053F85B98
            F7FCFCFC0A0A0A000000000000040404FCFCFCD1A98EFE5403FE5000B2954E0C
            841700790242996D6095EB578FE990B5EDFCFCFC0B0B0B000000000000040404
            FCFCFCBDA08DFE7B2EFE8130AE8F5F0B730D007702598F5AF9F9F9F9F9F9F9F9
            F9FCFCFC0B0B0B000000000000040404FCFCFCC8C0BCFBFBFBFBFBFBDADCCF03
            7603037603549C54FBFBFBF9F9F9F9F9F9FCFCFC0B0B0B000000000000040404
            FCFCFCAEAFAFF8F8F8F8F8F8E7E8E793C69391CA91BCCEBCF9F9F9F2F2F2F1F1
            F1FCFCFC0B0B0B000000000000040404FCFCFCE1E1E1FCFCFCFCFCFCFCFCFCFC
            FCFCFCFCFCFCFCC6C6C6CCCCCCC9C9C9DADADA0A0A0A000000000000040404
            FCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFBFBFBCBCBCBE8E8E8FCFC
            FC6C6C6C000000000000000000040404FCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFC
            FCFCFCFCFCF7F7F7CFCFCFFCFCFC757575000000000000000000000000040404
            FCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCFCF7F7F7DADADA7C7C7C0000
            00000000000000000000000000000000BABABAB4B4B4B4B4B4B4B4B4B4B4B4B4
            B4B4B4B4B4B8B8B8808080000000000000000000000000000000000000}
          OnClick = SpeedButton13Click
        end
        object Label12: TLabel
          Left = 8
          Top = 32
          Width = 84
          Height = 13
```

```
          Caption = #1050#1086#1083'-'#1074#1086'
'#1079#1072#1087#1091#1089#1082#1086#1074
        end
        object Edit4: TEdit
          Left = 118
          Top = 4
          Width = 60
          Height = 21
          TabOrder = 0
          Text = '700'
        end
        object CheckBox1: TCheckBox
          Left = 8
          Top = 69
          Width = 97
          Height = 17
          Caption = ' '#1040#1085#1080#1084#1072#1094#1080#1103
          TabOrder = 1
        end
        object RadioGroup3: TRadioGroup
          Left = 3
          Top = 98
          Width = 190
          Height = 66
          Caption = ' '#1047#1072#1074#1077#1088#1096#1077#1085#1080#1077'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1103' '
          ItemIndex = 1
          Items.Strings = (
            #1042#1089#1077' '#1084#1072#1088#1082#1077#1088#1099
            #1061#1086#1090#1103' '#1073#1099' '#1086#1076#1080#1085'
'#1084#1072#1088#1082#1077#1088)
          TabOrder = 2
        end
        object Edit5: TEdit
          Left = 118
          Top = 30
          Width = 60
          Height = 21
          TabOrder = 3
          Text = '1'
        end
      end
    end
  end
  object Panel2: TPanel
    Left = 217
    Top = 0
    Width = 1279
    Height = 900
    Align = alClient
    Caption = 'Panel2'
    TabOrder = 1
    object Splitter1: TSplitter
      Left = 1
      Top = 866
      Width = 1277
      Height = 3
      Cursor = crVSplit
      Align = alBottom
      ExplicitTop = 1
      ExplicitWidth = 423
    end
    object ScrollBox1: TScrollBox
```

```
      Left = 1
      Top = 1
      Width = 1277
      Height = 865
      Align = alClient
      TabOrder = 0
      object Image1: TImage
        Left = 0
        Top = 0
        Width = 193
        Height = 193
        AutoSize = True
        OnMouseUp = Image1MouseUp
      end
      object Image2: TImage
        Left = 320
        Top = 0
        Width = 185
        Height = 193
        AutoSize = True
        Visible = False
      end
    end
    object ScrollBox2: TScrollBox
      Left = 1
      Top = 869
      Width = 1277
      Height = 30
      Align = alBottom
      TabOrder = 1
      object Shape1: TShape
        Left = 5
        Top = 6
        Width = 11
        Height = 11
        Brush.Color = clYellow
        Shape = stCircle
        Visible = False
      end
      object Shape2: TShape
        Left = 37
        Top = 6
        Width = 11
        Height = 11
        Brush.Color = clRed
        Pen.Color = clBlue
        Shape = stCircle
        Visible = False
      end
    end
  end
  object PopupMenu1: TPopupMenu
    Left = 240
    Top = 312
    object N1: TMenuItem
      Caption = #8470' '#1087#1086#1079#1080#1094#1080#1080
      OnClick = N1Click
    end
    object N2: TMenuItem
      Caption = #1059#1076#1072#1083#1080#1090#1100'
'#1087#1086#1079#1080#1094#1080#1102
      OnClick = N2Click
    end
```

```
  end
  object PopupMenu2: TPopupMenu
    Left = 320
    Top = 312
    object N3: TMenuItem
      Caption = #8470' '#1087#1077#1088#1077#1093#1086#1076#1072
      OnClick = N3Click
    end
    object N4: TMenuItem
      Caption = #1059#1076#1072#1083#1080#1090#1100'
'#1087#1077#1088#1077#1093#1086#1076
      OnClick = N4Click
    end
  end
  object PopupMenu3: TPopupMenu
    Left = 400
    Top = 312
    object N5: TMenuItem
      Caption = #8470' '#1091#1079#1083#1072' '#1076#1091#1075#1080
      OnClick = N5Click
    end
    object N6: TMenuItem
      Caption = #1059#1076#1072#1083#1080#1090#1100' '#1091#1079#1077#1083'
'#1076#1091#1075#1080
      OnClick = N6Click
    end
  end
  object OpenDialog1: TOpenDialog
    DefaultExt = 'pmn'
    Filter = 'Petri-Markov Net files (*.pmn)|*.pmn|All files (*.*)|*.*'
    Left = 264
    Top = 256
  end
  object SaveDialog1: TSaveDialog
    DefaultExt = 'pmn'
    Filter = 'Petri-Markov Net files (*.pmn)|*.pmn|All files (*.*)|*.*'
    Left = 352
    Top = 256
  end
end
```

ПА.3. Файл Unit1.h

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Buttons.hpp>
#include <Classes.hpp>
#include <ComCtrls.hpp>
#include <Controls.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <Forms.hpp>
#include <Graphics.hpp>
#include <Grids.hpp>
#include <ImgList.hpp>
#include <jpeg.hpp>
#include <math.h>
#include <Menus.hpp>
```

```
#include <msxmldom.hpp>
#include <StdCtrls.hpp>
#include <XMLDoc.hpp>
#include <xmldom.hpp>
#include <XMLIntf.hpp>
//---------------------------------------------------------------------------

const int MaxArc = 100;

//---------------------------------------------------------------------------
// Объявления классов объектов,
// моделирующих структурные элементы сетей Петри-Маркова
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//
//                            Класс PMPosition
//
//        Определяет свойства и методы объекта СПМ типа "Позиция"
//
//---------------------------------------------------------------------------
class PMPosition
{
  public:

    int   SubNet;          //Индекс подсети
                           //(если СПМ состоит из нескольких подсетей)
    int   TypePos;         //Тип позиции: 0 - обычная, 1 - начальня,
                           //             2 - конечная (в своей подсети)

    int   NArcOut;         //Текущее количество исходящих дуг
    int   ArcInd[MaxArc];  //Индексы переходов, в которые выходят
                           //соответствующие дуги
    float P[MaxArc];       //Вероятности выхода в соответствующие дуги
    int   PLaw[MaxArc];    //Типы законов распределений, связанных с дугами
    float MeanVal[MaxArc]; //Матожидание соответствующего закона
    float MRS[MaxArc];     //СКО соответствующего закона

    int ind;               //Номер позиции (совпадает с индексом в массиве)

    int   Action;          //Связанное с позицией действие (индекс функции,
                           //которая будет осуществлять дополнительную
                           //анимацию в процессе моделирования)
    AnsiString Descript;   //Текстовое описание действия

    //Функция, генерирующая случайным образом порядковый номер в массиве дуг
    //(для определения дуги, по которой маркер уйдёт из данной позиции) с учётом
    //заданных вероятностей выхода по каждой из дуг
    //ВНИМАНИЕ: функция возвращает не индекс перехода, в который уходит маркер,
    //а порядковый номер из массива ArcInd индексов переходов
    int randArcOutNum()
    {
      int   p = Random(1000);
      float prob = 0.0;
      int   i;

      for(i = 0; i<NArcOut; i++){
         prob += P[i]*1000.0;
         if(prob>=p) return i;
      }
      return NArcOut-1;
    };
```

```
    //Функция, генерирующая время пребывания маркера в позиции, в соответствии
    //с ранее выбранной траекторией дальнейшего его движения (определённой
    //функцией randArcOutNum()) и типов и параметрами закона распределения PLaw,
    //MeanVal и MRS
    //Вход int i -- порядковый номер выбранной дуги в массиве ArcInd
    float randTime(int i)
    {
      float t;

      switch(PLaw[i]){
        case 0: //Равномерное распределение
          //Ширина интервала при равномерном распределении 2*СКО*корень(3)
          t = Random(2.0*MRS[i]*1.7321*1000.0)/1000.0 +
                                                 (MeanVal[i] - MRS[i]*1.7321);
          if (t<0) t = 0.001;
          break;

        case 1: //Нормальное распределение
          t = 0;
          for(int i=0; i<10; i++)
            t += Random(2.0*MRS[i]*1.7321*1000.0)/1000.0 +
                 (MeanVal[i] - MRS[i]*1.7321);

            t /= 10.0;
          break;

        default: //Вырожденное распределение
          t = 1;
          break;
      }

      return t;
    };
};
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//
//                            Класс PMTransition
//
//        Определяет свойства и методы объекта СПМ типа "Переход"
//
//---------------------------------------------------------------------------
class PMTransition
{
  public:
    int   NArcOut;            //Текущее количество исходящих дуг
    int   ArcOutInd[MaxArc];  //Индексы позиций, в которые переводят
                              //соответствующие исходящие дуги

    int   nMarkers[MaxArc];   //Массив для хранения числа маркеров,
                              //пришедших по соответствующим входным дугам

    int   NArcIn;             //Текущее количество входящих дуг
    int   ArcInInd[MaxArc];   //Индексы позиций, из которых приходят
                              //соответствующие входящие дуги

    AnsiString Logic[MaxArc]; //Массив для хранения конъюнкций

    int ind; //Номер позиции

    PMTransition()
    {
```

```
      //Обнулим количество маркеров, т.к. в начальный момент времени
      //маркеров в переходе быть не может
      for(int i=0; i<MaxArc; i++) nMarkers[i] = 0;
    };
};
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
//
//                           Класс PMArc
//
//        Определяет свойства и методы объекта типа "Дуга"
//     Используется для соединения элементов сетей Петри-Маркова.
//
//---------------------------------------------------------------------------
class PMArc
{
  public:
    int type0; //Тип объекта, который является прародителем дуги,
               //если дуга состоит из ряда узлов

    int ind1;  //Индекс объекта, который является началом дуги
    int type1; //Тип объекта, который является началом дуги
               //(1 -- позиция, 2 -- переход, 3 -- дуга)

    int ind2;  //Индекс объекта, который является окончанием дуги
    int type2; //Тип объекта, который является окончанием дуги
               //(1 -- позиция, 2 -- переход, 3 -- дуга)
};
//---------------------------------------------------------------------------


//---------------------------------------------------------------------------
//
//                           Класс PMMarker
//
//        Определяет свойства и методы объекта СПМ типа "Маркер"
//
//---------------------------------------------------------------------------
class PMMarker
{
  public:

    //Переменные для формирования результирующей статистики
    float TimeProc; //Общее время пребывания в позициях (обучение)
    float TimeWait; //Общее время пребывания в переходах (ожидание готовности)

    int nPosEnt;  //Число попаданий в позиции
    int nTraEnt;  //Число попаданий в переходы


    //Переменные для поддержки процедуры моделирования
    float dT;     //Сгенерированное при попадании в позицию время пребывания
                  //в ней. Либо (при попадании в переход) время попадания
                  //в переход -- для последующего вычисления времени пребывания
                  //в переходе

    bool inPos;   //Признак пребывания в позиции (true) или в переходе (false)

    int  indPos;  //Индекс позиции
    int  indTra;  //Индекс перехода
    //Индексы позиции и перехода, в зависимости от состояния inPos
    //означают следующее:
```

```
    //если inPos == true, т.е. маркер находится в позиции
    //indPos соответствует индексу текущей позиции, а indTra -- индексу
    //перехода, в который должен попасть маркер
    //
    //если inPos == false, т.е. маркер находится в переходе
    //indPos соответствует индексу позиции, из коорой пришёл маркер,
    //a indTra -- индексу текущего перехода, в котором находится маркер

    int Tag;      //Дополнительное поле для хранения какой-нибудь информации,
                  //связанной с маркером

    int SubNet;   //Номер подсети, к которой относится данный маркер

    PMMarker(){
      TimeProc = 0; //Обнулим время обработки
      TimeWait = 0; //и время ожидания
      nPosEnt  = 0; //Хотя маркер после создания находится в нулевой позиции,
                    //факт первого попадания в позицию мы учтём, когда будем
                    //рассчитывать время его пребывания в ней
      nTraEnt  = 0; //В переходы маркер пока не попадал

      inPos    = true; //В начальный моделирования момент маркер пребывает
                       //в позиции, а не в переходе
      indPos   = 0;    //Текущая позиция -- нулевая
      indTra   = 0;
      int SubNet = 0;  //Признак того, что маркер не относится ни к какой
                       //подсети (будем считать, что подсети нумеруются с 1)
    };
};
//---------------------------------------------------------------------------
class PMShape;
//---------------------------------------------------------------------------


//---------------------------------------------------------------------------
//
//                           Класс TForm1
//                       Основное окно программы
//
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:  // IDE-managed Components
  TButton *Button1;
  TButton *Button2;
  TButton *Button3;
  TCheckBox *CheckBox1;
  TComboBox *ComboBox1;
  TEdit *Edit1;
  TEdit *Edit2;
  TEdit *Edit4;
  TEdit *Edit5;
  TEdit *Edit6;
  TEdit *Edit7;
  TImage *Image1;
  TImage *Image2;
  TLabel *Label1;
  TLabel *Label10;
  TLabel *Label11;
  TLabel *Label12;
  TLabel *Label13;
  TLabel *Label2;
  TLabel *Label3;
```

```
TLabel *Label4;
TLabel *Label5;
TLabel *Label6;
TLabel *Label7;
TLabel *Label8;
TMenuItem *N1;
TMenuItem *N2;
TMenuItem *N3;
TMenuItem *N4;
TMenuItem *N5;
TMenuItem *N6;
TOpenDialog *OpenDialog1;
TPageControl *PageControl1;
TPageControl *PageControl2;
TPanel *Panel1;
TPanel *Panel2;
TPopupMenu *PopupMenu1;
TPopupMenu *PopupMenu2;
TPopupMenu *PopupMenu3;
TRadioGroup *RadioGroup1;
TRadioGroup *RadioGroup2;
TRadioGroup *RadioGroup3;
TRadioGroup *RadioGroup4;
TSaveDialog *SaveDialog1;
TScrollBox *ScrollBox1;
TScrollBox *ScrollBox2;
TShape *Shape1;
TShape *Shape2;
TSpeedButton *SpeedButton1;
TSpeedButton *SpeedButton10;
TSpeedButton *SpeedButton11;
TSpeedButton *SpeedButton12;
TSpeedButton *SpeedButton13;
TSpeedButton *SpeedButton2;
TSpeedButton *SpeedButton3;
TSpeedButton *SpeedButton4;
TSpeedButton *SpeedButton5;
TSpeedButton *SpeedButton6;
TSpeedButton *SpeedButton7;
TSpeedButton *SpeedButton8;
TSpeedButton *SpeedButton9;
TSplitter *Splitter1;
TStringGrid *StringGrid1;
TStringGrid *StringGrid2;
TTabSheet *TabSheet1;
TTabSheet *TabSheet2;
TTabSheet *TabSheet3;
TTabSheet *TabSheet4;
TTabSheet *TabSheet5;
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
void __fastcall FormCreate(TObject *Sender);
void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);
void __fastcall N1Click(TObject *Sender);
void __fastcall N2Click(TObject *Sender);
void __fastcall N3Click(TObject *Sender);
void __fastcall N4Click(TObject *Sender);
void __fastcall N5Click(TObject *Sender);
void __fastcall N6Click(TObject *Sender);
void __fastcall NODMouseDown(TObject *Sender, TMouseButton Button,
```

```
                TShiftState Shift, int X, int Y);
  void __fastcall NODMouseMove(TObject *Sender, TShiftState Shift, int X, int Y);
  void __fastcall NODMouseUp(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);
  void __fastcall POSMouseDown(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);
  void __fastcall POSMouseMove(TObject *Sender, TShiftState Shift, int X, int Y);
  void __fastcall POSMouseUp(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);
  void __fastcall PrintLogic(AnsiString L);
  void __fastcall RadioGroup2Click(TObject *Sender);
  void __fastcall SpeedButton11Click(TObject *Sender);
  void __fastcall SpeedButton12Click(TObject *Sender);
  void __fastcall SpeedButton13Click(TObject *Sender);
  void __fastcall SpeedButton1Click(TObject *Sender);
  void __fastcall SpeedButton2Click(TObject *Sender);
  void __fastcall SpeedButton3Click(TObject *Sender);
  void __fastcall SpeedButton4Click(TObject *Sender);
  void __fastcall SpeedButton5Click(TObject *Sender);
  void __fastcall SpeedButton6Click(TObject *Sender);
  void __fastcall SpeedButton7Click(TObject *Sender);
  void __fastcall SpeedButton8Click(TObject *Sender);
  void __fastcall StringGrid1Click(TObject *Sender);
  void __fastcall StringGrid2SetEditText(TObject *Sender, int ACol, int ARow,
                const AnsiString Value);
  void __fastcall TRAMouseDown(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);
  void __fastcall TRAMouseMove(TObject *Sender, TShiftState Shift, int X, int Y);
  void __fastcall TRAMouseUp(TObject *Sender, TMouseButton Button,
                TShiftState Shift, int X, int Y);

private:  // User declarations
public:    // User declarations

   PMShape*  POSShape[1000]; //Массив для хранения указателей на визуальные
                             //формы типа "Позиция"
   PMPosition POS[1000];     //Массив для хранения семантики объектов
                             //типа "Позиция"
   int      NPos;            //Текущее количество позиций в сети Петри-Маркова

   PMShape* TRAShape[1000];  //Массив для хранения указателей на визуальные
                             //формы типа "Переход"
   PMTransition TRA[1000];   //Массив для хранения семантики объектов
                             //типа "Переход"
   int NTra;                 //Текущее количество переходов в сети Петри-Маркова

   PMShape* NODShape[1000];  //Массив для хранения узлов дуг
   int NNod;                 //Текущее количество узлов дуг

   PMArc ARC[1000];   //Массив для хранения информации о дугах,
                      //соединяющих переход/позицию/узел
   int NArc;          //Текущее количество дуг в сети Петри-Маркова

   int SelObjInd;     //Индекс выделенного объекта
   int SelObjType;    //Тип выделенного объекта

   bool AddPos;       //Признак перехода в режим добавления позиции
   bool AddTrans;     //Признак перехода в режим добавления перехода
   bool AddArc;       //Признак перехода в режим добавления дуги

   int BasePointType;  //Тип объекта, который считается начальной точкой
                       //комплексной дуги, содержащей несколько узлов
   int FirstPointType; //Тип объекта, который считается начальной точкой дуги
```

```
   int FirstPointInd;  //Номер объекта, который считается начальной точкой дуги


   int MouseX, MouseY; //Координаты мыши в точке нажатия/отпускания

   int PopupInd;       //Индекс объекта, на котором открыли popup-меню


   bool EditingMode;   //Признак того, что система находится в режиме
                       //редактирования

   float    GTime;       //Глобальное модельное время
   PMMarker Mark[1000];  //Массив для хранения маркеров
   int NMark;            //Текущее количество маркеров в модели
   int NMarkEnd;         //Количество маркеров в конечной позиции


   void PrintNet();           //Функция отрисовки всей совокупности дуг
                              //в сети Петри-Маркова

   void DeleteArc(int ind);   //Удалить дугу с индексом ind
                              //и перенумеровать остальные
   void DeletePos(int ind);   //Удалить позицию с индексом ind, связанные с ней
                              //дуги и перенумеровать остальные +
                              //скорректировать ссылки в дугах
   void DeleteTrans(int ind); //Удалить переход с индексом ind, связанные с ним
                              //дуги и перенумеровать остальные +
                              //скорректировать ссылки в дугах
   void DeleteNode(int ind);  //Удалить узел с индексом ind, связанные с ним
                              //дуги и перенумеровать остальные +
                              //скорректировать ссылки в дугах

   //Массив указателей на функции из внешней библиотеки,
   //осуществляющие анимацию действий
   void(__stdcall * pf[100]) (TForm1*, bool);
   int NAct;       //Количество функций-действий внутри загруженной библиотеки
   HINSTANCE dllp; //Дескриптор загружаемой внешней библиотеки

   int nShot1, nShot2;
   int nWin1,  nWin2;
   float Time1[10000];
   float Time2[10000];


  __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------


//---------------------------------------------------------------------------
//
//                           Класс PMShape
//
// Потомок стандартного класса TShape. Используется для визуализации
// объектов СПМ (позиций и переходов)
//
//---------------------------------------------------------------------------
class PMShape : public TShape
{
 protected:
   virtual void __fastcall Paint(void)
   {
     TShape::Paint();
```

```
    AnsiString s;
    //Если система в режиме редактирования -- печатаем номера позиций/переходов
    if(((TForm1*)Owner)->EditingMode){
      s = AnsiString(Tag);
      Canvas->TextOut(11-Canvas->TextWidth(s)/2, 10-Canvas->TextHeight(s)/2, s);
    }
    else{ //Иначе -- печатаем количество маркеров в позиции/переходе
      s = Label ? AnsiString(Label) : AnsiString("");
      Canvas->TextOut(11-Canvas->TextWidth(s)/2, 10-Canvas->TextHeight(s)/2, s);
    }
  };

 public:
  int Label; //Текст для вывода

  __fastcall virtual PMShape(TComponent* owner) : TShape(owner)
  {
    Label = 0;
  };

};
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

ПА.4. Файл Unit2.cpp

```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Chart"
#pragma link "TeEngine"
#pragma link "TeeProcs"
#pragma link "Series"
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormShow(TObject *Sender)
{
  StringGrid1->ColCount = 7;
  StringGrid1->RowCount = Form1->NMark+2; //1 дополнительная -- под заголовок
                                          //и 1 дополнительная -- под "итого"

  StringGrid1->Cells[0][0] = "№";
  StringGrid1->Cells[1][0] = "T_mod";
  StringGrid1->Cells[2][0] = "T_proc";
  StringGrid1->Cells[3][0] = "T_wait";
  StringGrid1->Cells[4][0] = "N_pos";
  StringGrid1->Cells[5][0] = "T_per_pos";
```

```
StringGrid1->Cells[6][0] = "T_per_tra";

int i;

//Средние характеристики по модели в пересчёте на маркер
float Tmodel = 0.0; //Среднее время пребывания маркера в модели
float Tproc  = 0.0; //Среднее время пребывания маркера в позициях
float Twait  = 0.0; //Среднее время пребывания маркера в переходах
float AveLen = 0.0; //Средняя длина тракетории, по которой движутся маркеры
float Tpos   = 0.0; //Среднее время пребывания маркера в одной позиции
float Ttra   = 0.0; //Среднее время пребывания маркера в одном переходе

for(i=0; i<Form1->NMark; i++){
  //Номер маркера
  StringGrid1->Cells[0][i+1] = i;
  StringGrid1->Cells[1][i+1] = FormatFloat("0.00", Form1->Mark[i].TimeProc +
                                                   Form1->Mark[i].TimeWait);
  Tmodel += Form1->Mark[i].TimeProc + Form1->Mark[i].TimeWait;

  StringGrid1->Cells[2][i+1] = FormatFloat("0.00", Form1->Mark[i].TimeProc);
  Tproc += Form1->Mark[i].TimeProc;

  StringGrid1->Cells[3][i+1] = FormatFloat("0.00", Form1->Mark[i].TimeWait);
  Twait += Form1->Mark[i].TimeWait;

  StringGrid1->Cells[4][i+1] = Form1->Mark[i].nPosEnt;
  AveLen += Form1->Mark[i].nPosEnt;

  StringGrid1->Cells[5][i+1] = FormatFloat("0.00",
                           Form1->Mark[i].TimeProc / Form1->Mark[i].nPosEnt);
  Tpos += float(Form1->Mark[i].TimeProc) / Form1->Mark[i].nPosEnt;

  StringGrid1->Cells[6][i+1] = FormatFloat("0.00",
                           Form1->Mark[i].TimeWait / Form1->Mark[i].nTraEnt);
  Ttra += float(Form1->Mark[i].TimeWait) / Form1->Mark[i].nTraEnt;
}

StringGrid1->Cells[0][Form1->NMark+1] = "В среднем:";
StringGrid1->Cells[1][Form1->NMark+1] =
       FormatFloat("0.00", Tmodel/Form1->NMark);
StringGrid1->Cells[2][Form1->NMark+1] =
       FormatFloat("0.00", Tproc/Form1->NMark);
StringGrid1->Cells[3][Form1->NMark+1] =
       FormatFloat("0.00", Twait/Form1->NMark);
StringGrid1->Cells[4][Form1->NMark+1] =
       FormatFloat("0.00", AveLen/Form1->NMark);
StringGrid1->Cells[5][Form1->NMark+1] =
       FormatFloat("0.00", Tpos/Form1->NMark);
StringGrid1->Cells[6][Form1->NMark+1] =
       FormatFloat("0.00", Ttra/Form1->NMark);

Application->HintHidePause = 4500;


//Заполнение гистограмм
float Tmin, Tmax, step;
int h1[10], h2[10], h_all[10];

Tmin = Tmax = (Form1->nWin1 > 0) ? Form1->Time1[0] : Form1->Time2[0];

//int i;
for(i=0; i<Form1->nWin1; i++){
  if(Form1->Time1[i] < Tmin) Tmin = Form1->Time1[i];
```

```
    if(Form1->Time1[i] > Tmax) Tmax = Form1->Time1[i];
  }

  for(i=0; i<Form1->nWin2; i++){
    if(Form1->Time2[i] < Tmin) Tmin = Form1->Time2[i];
    if(Form1->Time2[i] > Tmax) Tmax = Form1->Time2[i];
  }

  Tmin = int(Tmin);
  Tmax = (Tmax - int(Tmax) > 0) ? int(Tmax)+1 : int(Tmax);

  step = (Tmax - Tmin) / 10;

  for(i=0; i<10; i++){
    h1[i] = 0;
    h2[i] = 0;
    h_all[i] = 0;
  }

  for(i=0; i<Form1->nWin1; i++){
    h1[int((Form1->Time1[i] - Tmin)/step)]++;
    h_all[int((Form1->Time1[i] - Tmin)/step)]++;
  }

  for(i=0; i<Form1->nWin2; i++){
    h2[int((Form1->Time2[i] - Tmin)/step)]++;
    h_all[int((Form1->Time2[i] - Tmin)/step)]++;
  }

  Chart1->Series[0]->Clear();
  for(i=0; i<10; i++){
    Chart1->Series[0]->Add(h1[i], "", clGreen);
  }

  Chart2->Series[0]->Clear();
  for(i=0; i<10; i++){
    Chart2->Series[0]->Add(h2[i], "", clBlue);
  }

  Chart3->Series[0]->Clear();
  for(i=0; i<10; i++){
    Chart3->Series[0]->Add(h_all[i], "", clYellow);
  }

}
//---------------------------------------------------------------------------
void __fastcall TForm2::StringGrid1MouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
  int ACol, ARow;
  static int C = 0, R = 0;

  StringGrid1->MouseToCell(X, Y, ACol, ARow);
  if((ACol != C) || (ARow != R)){
    C = ACol; R = ARow;
    Application->CancelHint();

    if(ARow == 0 ){
      switch (ACol)
      {
        case 0:
          StringGrid1->Hint = "Индекс маркера";
          break;
```

```
    case 1:
      StringGrid1->Hint = "Общее время пребывания маркера в модели\n"
                          "(без учёта пребывания в конечной позиции)";
      break;

    case 2:
      StringGrid1->Hint = "Общее время пребывания маркера в"
                          " позициях (кроме конечной)";
      break;

    case 3:
      StringGrid1->Hint = "Общее время пребывания маркера "
                          "в непримитивных переходах";
      break;

    case 4:
      StringGrid1->Hint = "Общее число пройденных маркером позиций";
      break;

    case 5:
      StringGrid1->Hint = "Среднее время пребывания маркера в позиции";
      break;

    case 6:
      StringGrid1->Hint = "Среднее время пребывания маркера в переходе";
      break;
  }
}
else
if(ARow < (StringGrid1->RowCount-1)){
  switch (ACol)
  {
    case 0: StringGrid1->Hint = "Индекс маркера"; break;
    case 1:
      StringGrid1->Hint = "Общее время пребывания маркера " +
            AnsiString(ARow-1) +
            " в модели\n(без учёта пребывания в конечной позиции)";
      break;

    case 2:
      StringGrid1->Hint = "Общее время пребывания маркера " +
      AnsiString(ARow-1) + " в позициях (кроме конечной)";
      break;

    case 3:
      StringGrid1->Hint = "Общее время пребывания маркера " +
      AnsiString(ARow-1) + " в непримитивных переходах";
      break;

    case 4:
      StringGrid1->Hint = "Общее число пройденных маркером " +
      AnsiString(ARow-1) + " позиций";
      break;

    case 5:
      StringGrid1->Hint = "Среднее время пребывания маркера " +
      AnsiString(ARow-1) + " в позиции";
      break;

    case 6:
      StringGrid1->Hint = "Среднее время пребывания маркера " +
      AnsiString(ARow-1) + " в переходе";
      break;
```

```
      }
    }
    else{
      switch (ACol)
      {
        case 0:
          StringGrid1->Hint = "Строка итогов";
          break;

        case 1:
          StringGrid1->Hint = "Среднее общее время пребывания маркеров "
          "в модели\n(без учёта пребывания в конечной позиции)";
          break;

        case 2:
          StringGrid1->Hint = "Среднее общее время пребывания маркеров "
          "в позициях (кроме конечной)";
          break;

        case 3:
          StringGrid1->Hint = "Среднее общее время пребывания маркеров "
          "в непримитивных переходах";
          break;

        case 4:
          StringGrid1->Hint = "Среднее общее число пройденных маркерами позиций";
          break;

        case 5:
          StringGrid1->Hint = "Время пребывания маркера в позиции "
          "в среднем по модели";
          break;

        case 6:
          StringGrid1->Hint = "Время пребывания маркера в переходе "
          "в среднем по модели";
          break;
      }
    }
  }

}
//---------------------------------------------------------------------------
void __fastcall TForm2::StringGrid1DrawCell(TObject *Sender, int ACol, int ARow,
      TRect &Rect, TGridDrawState State)
{
  StringGrid1->Canvas->Brush->Color = clWhite;
  StringGrid1->Canvas->FillRect(Rect);


  char szANSIString[256];
  int szANSIStringLen;
  szANSIStringLen = WideCharToMultiByte (CP_ACP,
                      WC_COMPOSITECHECK,
                      StringGrid1->Cells[ACol][ARow].c_str(),
                      -1,
                      szANSIString,
                      sizeof(szANSIString),
                      NULL,
                      NULL);

  DrawText(StringGrid1->Canvas->Handle,
           szANSIString,
```

```
            szANSIStringLen, &Rect,
            DT_CENTER | DT_VCENTER | DT_SINGLELINE);
}
//---------------------------------------------------------------------------

void __fastcall TForm2::BitBtn2Click(TObject *Sender)
{
  AnsiString file1 = "temp.txt";
  TStringList *Table1 = new TStringList;

  for(int i = 0; i<StringGrid1->RowCount; i++)
  {
    Table1->Add(StringGrid1->Rows[i]->DelimitedText);
  }
  Table1->SaveToFile(file1) ;   // просто сохраняем

  delete Table1;
}
//---------------------------------------------------------------------------
```

ПА.5. Файл Unit2.dfm

```
//---------------------------------------------------------------------------
object Form2: TForm2
  Left = 0
  Top = 0
  Caption = #1057#1090#1072#1090#1080#1089#1090#1080#1082#1072'
'#1084#1086#1076#1077#1083#1080#1088#1086#1074#1072#1085#1080#1103
  ClientHeight = 446
  ClientWidth = 513
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  Position = poOwnerFormCenter
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
  object Label1: TLabel
    Left = 64
    Top = 384
    Width = 52
    Height = 13
    Caption = #1055#1086#1076#1089#1077#1090#1100' 1'
  end
  object Label2: TLabel
    Left = 232
    Top = 384
    Width = 52
    Height = 13
    Caption = #1055#1086#1076#1089#1077#1090#1100' 2'
  end
  object Label3: TLabel
    Left = 400
    Top = 384
    Width = 68
    Height = 13
    Caption = #1054#1073#1097#1077#1077' '#1074#1088#1077#1084#1103
  end
```

```
  object StringGrid1: TStringGrid
    Left = 0
    Top = 0
    Width = 513
    Height = 154
    Align = alTop
    ColCount = 7
    DefaultColWidth = 70
    DefaultRowHeight = 20
    Options = [goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine, goRang-
eSelect, goColSizing]
    ParentShowHint = False
    ShowHint = True
    TabOrder = 0
    OnDrawCell = StringGrid1DrawCell
    OnMouseMove = StringGrid1MouseMove
  end
  object Panel1: TPanel
    Left = 0
    Top = 405
    Width = 513
    Height = 41
    Align = alBottom
    BevelOuter = bvNone
    TabOrder = 1
    object BitBtn1: TBitBtn
      Left = 277
      Top = 6
      Width = 75
      Height = 25
      Cancel = True
      Caption = 'OK'
      Default = True
      DoubleBuffered = True
      Glyph.Data = {
        DE010000424DDE010000000000007600000028000000240000001200000001000
        0400000000068010000000000000000000010000000000000000000000000000
        8000008000000080800080000000800080008080000C0C0C000808080000000
        FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00333333333333
        3333333333333333333333300003333333333333333333333F33333333333
        0000333344333333333333333388F3333333330000033334224333333333333
        338338F3333333300003334222243333333333833338F3333333000033342
        2222433333333383333338F3333333000034222A22224333333338F338F333
        8F33333300003222A3A2224333333338F3838F338F33333300003A2A333A2224
        33333338F83338F338F33333000033A33333A22243333333833333F338F3333
        0000333333333A22243333333333338F338F33300003333333333A222433333
        333333338F338F33000033333333333A22243333333333338F338F300003333
        33333333A22243333333333338F338F0000333333333333A22433333333333
        3338F38F000033333333333333A223333333333333338F830000333333333333
        333A33333333333333338330000333333333333333333333333333333333333
        0000}
      ModalResult = 1
      NumGlyphs = 2
      ParentDoubleBuffered = False
      TabOrder = 0
    end
    object BitBtn2: TBitBtn
      Left = 181
      Top = 6
      Width = 75
      Height = 25
      Caption = #1057#1086#1093#1088#1072#1085#1080#1090#1100
      DoubleBuffered = True
```

```
    ModalResult = 1
    NumGlyphs = 2
    ParentDoubleBuffered = False
    TabOrder = 1
    OnClick = BitBtn2Click
  end
end
object Chart1: TChart
  Left = 8
  Top = 160
  Width = 163
  Height = 218
  Legend.Visible = False
  Title.Text.Strings = (
    'TChart')
  Title.Visible = False
  View3D = False
  TabOrder = 2
  ColorPaletteIndex = 13
  object Series1: TBarSeries
    Marks.Arrow.Visible = True
    Marks.Callout.Brush.Color = clBlack
    Marks.Callout.Arrow.Visible = True
    Marks.Visible = False
    XValues.Name = 'X'
    XValues.Order = loAscending
    YValues.Name = 'Bar'
    YValues.Order = loNone
  end
end
object Chart2: TChart
  Left = 177
  Top = 160
  Width = 163
  Height = 218
  Legend.Visible = False
  Title.Text.Strings = (
    'TChart')
  Title.Visible = False
  View3D = False
  TabOrder = 3
  ColorPaletteIndex = 13
  object BarSeries1: TBarSeries
    Marks.Arrow.Visible = True
    Marks.Callout.Brush.Color = clBlack
    Marks.Callout.Arrow.Visible = True
    Marks.Visible = False
    XValues.Name = 'X'
    XValues.Order = loAscending
    YValues.Name = 'Bar'
    YValues.Order = loNone
  end
end
object Chart3: TChart
  Left = 346
  Top = 160
  Width = 163
  Height = 218
  Legend.Visible = False
  Title.Text.Strings = (
    'TChart')
  Title.Visible = False
  View3D = False
```

```
    TabOrder = 4
    ColorPaletteIndex = 13
    object BarSeries2: TBarSeries
      Marks.Arrow.Visible = True
      Marks.Callout.Brush.Color = clBlack
      Marks.Callout.Arrow.Visible = True
      Marks.Visible = False
      XValues.Name = 'X'
      XValues.Order = loAscending
      YValues.Name = 'Bar'
      YValues.Order = loNone
    end
  end
end
```

ПА.6. Файл Unit2.h

```
//---------------------------------------------------------------------------
#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#include <Buttons.hpp>
#include <Classes.hpp>
#include <Controls.hpp>
#include <ExtCtrls.hpp>
#include <Forms.hpp>
#include <Grids.hpp>
#include <StdCtrls.hpp>
#include "Chart.hpp"
#include "Series.hpp"
#include "TeEngine.hpp"
#include "TeeProcs.hpp"
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:  // IDE-managed Components
  TBarSeries *BarSeries1;
  TBarSeries *BarSeries2;
  TBarSeries *Series1;
  TBitBtn *BitBtn1;
  TBitBtn *BitBtn2;
  TChart *Chart1;
  TChart *Chart2;
  TChart *Chart3;
  TLabel *Label1;
  TLabel *Label2;
  TLabel *Label3;
  TPanel *Panel1;
  TStringGrid *StringGrid1;

  void __fastcall FormShow(TObject *Sender);
  void __fastcall StringGrid1MouseMove(TObject *Sender, TShiftState Shift, int X,
          int Y);
  void __fastcall StringGrid1DrawCell(TObject *Sender, int ACol, int ARow,
          TRect &Rect, TGridDrawState State);
  void __fastcall BitBtn2Click(TObject *Sender);
private:
public:
  __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
```

```
//---------------------------------------------------------------------------
#endif
```