РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ростовский государственный университет путей сообщения»
ФГБОУ ВПО РГУПС

На правах рукописи

ЧУБЕЙКО СЕРГЕЙ ВАЛЕРЬЕВИЧ

Аналитические и имитационные методы дискретно-событийного моделирования в задачах анализа надежности и производительности компьютерных систем

Специальность: 05.13.18 – «Математическое моделирование, численные методы и комплексы программ»

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

Научный руководитель – доктор технических наук, профессор
Бутакова Мария Александровна

Ростов-на-Дону – 2014

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования

Расширение функциональных возможностей, усложнение и увеличение числа задач, решаемых современными сетевыми информационно-управляющими системами (ИУС), обеспечивающими автоматизированное управление, контроль, информационную поддержку принятия решений в условиях реального времени в сложных технических системах, требуют развития и совершенствования методов их компьютерного моделирования, предшествующих этапам проектирования, разработки и эксплуатации. В современных условиях усложнения сетевых ИУС наиболее остро ощущается недостаток методов и инструментов оценки производительности и надежности, особенно для сетевого программного обеспечения (ПО) в условиях возникновения предельных информационных нагрузок, которые существенно влияют как на производительность, так и на надежность. Такие информационные нагрузки часто имеют непродолжительный, «всплесковый» характер, поэтому в контексте диссертации называются пиковыми информационными нагрузками.

Современное управляющее сетевое ПО разрабатывается по принципу событийного управления системами, поэтому среди методов моделирования, которые являются наиболее адекватными для решения указанных задач, можно выделить методы дискретно-событийного моделирования. В его основу положена следующая концепция: состояния моделируемой системы изменяются под воздействием некоторых событий, в общем случае безотносительно их точной привязки к временной шкале. Существенными являются лишь факты наличия возникновения этих событий и взаимодействие их между собой, то есть синхронизация (некоторое событие предшествует другому, некоторое событие вызывает возникновение другого, либо других событий и так далее).

Для сетевых ИУС, как многопользовательских, многозадачных, многомашинных и многопроцессорных систем, характерным является еще один аспект событийности – конкуренция за сетевые распределенные вычислительные ресурсы с целью увеличения производительности, минимизации простоев и тому подобное. Оба этих аспекта – синхронизация и конкуренция – делают сетевые компьютерные системы существенно нелинейными, что усложняет их аналитическое и имитационное моделирование, а предложенные в диссертации методы и численные алгоритмы можно рассматривать как возможную линеаризацию с целью упрощения моделирования таких систем. Завершенным инструментом дискретно-событийного моделирования является разработанное в диссертации специализированное ПО, реализованное на функциональном и императивном языках программирования.

Подтверждением актуальности темы исследования является участие автора в грантах, поддержанных Российским фондом фундаментальных исследований:

– 11-07-13110-офи-м-2011-РЖД «Методы, модели и алгоритмы оценки качества функционирования и синтеза надежного ПО информационно-управляющих систем на железнодорожном транспорте»;

– 12-08-00798-а «Математическое и программное обеспечение интеллектуальной обработки неполных и слабоструктурированных данных в информационно-управляющих системах с повышенными требованиями к надежности и качеству функционирования»;

– 13-01-00325-а «Гибридные модели интеллектуального управления и принятия диагностических решений в контролируемых динамических дискретных системах»;

– 13-01-00637-а «Модели интерполяций стохастических систем в условиях большой неопределенности: интеллектуальный анализ данных, алгоритмы и методы принятия оптимальных решений».

Степень разработанности проблемы

В контексте рассматриваемых в диссертации задач дискретно-событийное моделирование рассматривается в двух направлениях.

В первом направлении оно основано на преобразованиях функций со значениями в некоторых алгебраических полукольцах, причем обычные операции сложения и умножения заменяются на другие операции, связанные законом дистрибутивности. Если вместо обычного сложения используются операции нахождения максимума либо минимума, а вместо обычного умножения – операции обычного сложения, то получается подход, который в научной литературе называется (*max*, +), либо (*min*, +) алгеброй или идемпотентной математикой. Наиболее значительные результаты в этой области получены в нашей стране в работах В.Н. Колокольцова, Н.К. Кривулина, Г.Л. Литвинова, В.П. Маслова и их коллег. За рубежом в данной области значительные результаты получены Ф. Бачелли, П. Бутковичем, М. Гондраном, Ж.П. Квадра, Г. Коэном, У.М. Макэнени, М. Мину и другими.

Во втором направлении дискретно-событийное моделирование основано на математических моделях роста надежности ПО вследствие обнаружения и исправления ошибок в ПО и его обновления. Наиболее важные результаты в рассматриваемой области получены в работах Д.С. Боуэна, А. Гоэла, З. Джелинского, Л.Д. Ла Падула, М. Липова, Г. Майерса, П.Л. Моранда, Дж. Муса, Т.А. Тейера, Н.Ф. Шнейдевинда, М. Шумана и других.

Целью диссертационной работы является развитие математических методов дискретно-событийного моделирования высоконагруженных сетевых систем с пиковыми информационными нагрузками, имитационное моделирование компьютерных систем с гарантированным обслуживанием, разработка численных методов оценки информационной нагрузки и производительности сетевых компьютерных систем, разработка дискретно-событийного подхода к оценке надежности программного обеспечения, разработка алгоритмов моделирования неоднородных многомерных точечных случайных процессов с не-

прерывным временем для дискретно-событийного моделирования и прогнозирования надежности программного обеспечения, а также реализация соответствующих им программных средств.

Для достижения поставленной цели решаются **следующие задачи:**

1.Развитие двух направлений дискретно-событийного моделирования: 1) с применением алгебраических структур (*max*,+) и (*min*,+) подхода; 2) с визуализацией схем дискретно-событийных систем на основе выделения типовых структурных элементов системы в совокупности с реализуемыми ими вычислительными процедурами (*max*,+) и (*min*,+) алгебры. Решение этих задач позволяет синтезировать структуру всей моделируемой системы в виде временного событийного графа, снабженного рекуррентными уравнениями, моделирующими динамику системы.

2. Разработка аналитических моделей и численных оценок пиковых информационных нагрузок в сетевых системах. Целью является дискретно-событийное моделирование систем с индикаторами пиковых нагрузок, которые в диссертации рассматриваются как информационные нагрузки, однако возможно распространение их моделей на другие классы технических систем. Естественным практическим применением таких моделей в рассматриваемых технических системах является анализ их производительности, выполняемый с целью оптимизации управления информационными и транспортными потоками, многопользовательским доступом к ресурсам систем, а также управления сопутствующими технико-экономическими показателями этих систем.

3. Развитие методов дискретно-событийного моделирования надежности ПО сетевых систем, учитывающих структуру программных конструкций и классы ошибок. Используемый подход основан на принципе «роста» надежности ПО, который основывается на утверждении, что любая программная система содержит некоторое число ошибок, которые в процессе тестирования,

верификации и эксплуатации устраняются, а само ПО подвергается обновлению. Разработка методов оценки надежности ПО моделями типа «вход-конструкция-выход» с несколькими классами ошибок.

4. Разработка численных методов дискретно-событийного моделирования для новых моделей и подходов к оценке производительности и надежности компьютерных систем, алгоритмов моделирования неоднородных точечных многомерных случайных процессов с непрерывным временем, как инструмента моделирования процессов возникновения ошибок в ПО. В качестве программных средств реализации методов и алгоритмов выбраны императивный и функциональный подходы к программированию, являющиеся наиболее эффективными в отношении решения задач дискретно-событийного моделирования.

Объектами исследований являются модели сетевых систем и процессов. С технической точки зрения объектами являются процессы оценки производительности и надежности сетевых ИУС, их численные характеристики, характеристики надежности ПО. **Методы исследования** относятся к дискретно-событийному моделированию, имитационному моделированию, теории идемпотентной математики, методам сетеметрии, теории случайных точечных процессов, методам теории оценки производительности сетевых систем, методам теории оценки надежности ПО. **Предметом исследования** являются модели оценки производительности систем и надежности ПО распределенных сетевых ИУС, позволяющие реализовать вычислительные эксперименты.

Объект, предмет и методы исследования соответствуют специальности 05.13.18, так как содержанием диссертации является разработка фундаментальных основ и применение математического моделирования, численных методов и комплексов программ для решения научных и технических, фундаментальных и прикладных проблем и соответствует п. 4 «Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента», п. 5

«Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента» и п. 8 «Разработка систем компьютерного и имитационного моделирования» паспорта научной специальности.

Научная новизна. Основными новыми научными результатами являются

в области математического моделирования

– математические модели оценки пиковых информационных нагрузок компьютерных систем при различных свойствах базовых дискретно-событийных процессов;

– модель анализа производительности высоконагруженной сетевой системы с граничными значениями показателей производительности;

– модель оценки надежности ПО на основе методов дискретно-событийного моделирования, учитывающая алгоритмическое особенности ПО и разные классы ошибок;

– имитационная модель сетевой системы с гарантированным обслуживанием на основе (*min*,+) подхода;

в области численных методов

– численные методы расчета вариантов исполнения (последовательного, конкурентного и синхронизированного) событий в дискретно-событийной системе на основе интервальных временных событийных графов;

– алгоритмы имитационного моделирования на основе одномерных и двумерных неоднородных точечных случайных процессов с непрерывным временем с одним или двумя датчиками;

– численный метод расчета (*min*,+) интегральной свертки;

в области программных комплексов

– программная реализация на императивном языке программирования событийного ориентированного и процессно-ориентированного исполнения событий в дискретно-событийном моделировании;

– программная реализация на функциональном языке программирования имитационных моделей и оценок производительности сетевых систем с гарантированным обслуживанием;

– программная реализация на функциональном языке программирования основного уравнения баланса сетевой системы;

– программная реализация на императивном языке программирования оценки прогнозирования надежности сетевого ПО.

Основные результаты, выносимые на защиту.

1. Метод модульного дискретно-событийного моделирования на основе интервальных временных событийных графов.

2. Модели и числовые оценки пиковой информационной нагрузки и анализа производительности компьютерной сетевой системы с граничными значениями характеристик обслуживания.

3. Дискретно-событийный подход к оценке надежности программных компонентов с разными алгоритмическими конструкциями и классами ошибок.

4. Численные алгоритмы имитационного моделирования процессов возникновения ошибок, базирующиеся: на основе одномерных неоднородных случайных процессов с одним датчиком; на основе одномерных неоднородных случайных процессов с двумя датчиками; двумерных неоднородных случайных процессов на круге и в области, ограниченной некоторой функцией.

5. Методы имитационного моделирования и их программные реализации для технических приложений, связанных с задачами оценки надежности и производительности компьютерных систем: на основе (*min*,+) фильтраций, метод расчета (*min*,+) интегральной свертки и его реализация на функциональном языке программирования; метод имитационного дискретно-событийного моделирования производительности сетевых систем с гарантированным обслуживанием и его реализация на функциональном языке программирования; метод имитационного моделирования и прогнозирования надежности ПО на

основе бутстреппинга и его реализация на императивном языке программирования.

Достоверность полученных в диссертации результатов обеспечена математическими средствами анализа предложенных алгоритмов, анализом и верификацией исходных текстов программ, проведенными вычислительными экспериментами, как с модельными, так и с реальными данными, внедрением результатов диссертационного исследования.

Практическая значимость диссертации заключается в применении её результатов для решения широкого круга практических задач, связанных с оценкой производительности и надежности сетевых систем. Результаты диссертации нашли реализацию в виде новых программных средств, имеющихся исходных текстов программ и практического внедрения в деятельности телекоммуникационной компании ООО «Альянс-Телеком».

Апробация результатов работы

Основные положения и результаты диссертации были апробированы и обсуждались на следующих международных научно-практических конференциях:

– XII Всероссийском симпозиуме по прикладной и промышленной математике, 2011 г., г. Казань;

– Всероссийской научно-практической конференции «Транспорт-2011», 2011 г., г. Ростов-на-Дону;

– Первой научно-технической конференции «Интеллектуальные системы управления на железнодорожном транспорте (ИСУЖТ-2012)», 2012 г., г. Москва;

– Международной заочной научно-практической конференции «Теоретические и практические аспекты развития науки», 2013 г., г. Санкт-Петербург;

– Всероссийской научно-практической конференции «Транспорт-2013», 2013 г., г. Ростов-на-Дону;

Международной научно-практической конференции «Строительство-2013», 2013 г., г. Ростов-на-Дону.

Публикации. Полученные в диссертации теоретические и практические результаты нашли свое отражение в 13 печатных работах, 5 из которых опубликованы в изданиях, рекомендованных ВАК РФ.

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения, списка литературных источников из 73 наименований, заключения, приложения. Общий объем диссертации 131 страница.

1 ТЕОРЕТИЧЕСКИЕ И ПРАКТИЧЕСКИЕ АСПЕКТЫ ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ В КОНТЕКСТЕ ПОСТАВЛЕННЫХ ЗАДАЧ

1.1 Общий подход к дискретно-событийному моделированию систем

В диссертационной работе рассматриваются аспекты моделирования широкого класса систем, причем именно признаки событийности являются наиболее существенными для адекватного составления математических моделей. Признаки событийности и построенные на них методы моделирования [37, 68, 69] существенно отличают рассматриваемый далее подход от методов моделирования, общепринятых, например, в теории систем и теории автоматического управления, основными инструментами которых являются интегро-дифференциальные уравнения, методы теории вероятностей, математической статистики и случайных процессов. В теории систем и теории автоматического управления обычным описанием исследуемой системы является описание «вход-выход», а изменение выхода относительно входа, то есть пространство состояний системы задается передаточными функциями в матричном виде. Большинство систем, являющихся объектами моделирования являются нелинейными и в данном случае существенные усилия моделирования направлены на линеаризацию систем, выполняемую путем решения систем дифференциальных уравнений в матричном виде. Результатами моделирования являются восстановленное фазовое пространство моделируемых систем и характеристики звеньев управления и регулирования.

Другим подходом является имитационное моделирование [15] использующее методы теорий систем и сетей массового обслуживания (СМО и СеМО) [23, 25], в которых рассматриваются различные модели входных, выходных потоков и правил обслуживания, построенных на базе соответствующих законов распределения случайных величин и процессов. В данном случае методами моделирования являются генераторы некоторых типов случайных

процессов (имитирующих моменты времени поступления заявок на обслуживание), а результатами моделирования являются (часто усредненные) времена пребывания заявки в очереди, системе, времена обслуживания, вероятности пребывания в очереди, вероятности обслуживания (за некоторый период) и другие вероятностные и статистические характеристики.

Заметим следующее, что и в общей теории систем, и в теориях СМО и СеМО неявно предполагается использование процессного времени. Это означает, что в первом случае принимается, что процессы в системах протекают по возможности мгновенно (хотя в некоторых случаях допустима их инерционность), а во втором случае принимается, что процессы подчинены некоторому вероятностному закону. Однако во втором случае, иногда рассматривают потоки *general* (общего) типа, в которых основным соотношением является рекуррентное уравнение Линдли [24], что по сути близко к рассматриваемому нами далее подходу с идейной стороны, но не со стороны реализации методов моделирования. В целом, процессное время означает, что изменение состояний системы, а также её модели можно отметить на некоторой временной шкале, если шкала является непрерывной, то естественным будет непрерывное моделирование состояний системы, иначе – дискретное моделирование состояний системы, а также соответствующий им математический аппарат.

В основу дискретно-событийного моделирования, развиваемого появления известной системы *GPSS* [2] и сетей Петри [11, 53] положена другая концепция: состояния системы изменяются под воздействием некоторых событий, в общем случае безотносительно их точной привязки к временной шкале. Существенными являются лишь факты наличия возникновения этих событий и взаимодействие их между собой, то есть синхронизация (некоторое событие предшествует другому, некоторое событие вызывает возникновение другого, либо других событий и так далее). Примером таких информационных систем являются сетевые компьютерные системы. Для сетевых компьютерных систем, как многопользовательских многозадачных, многомашинных, многопроцессорных систем, характерным является еще один аспект событийности –

конкуренция за сетевые распределенные вычислительные ресурсы, с целью увеличения производительности, минимизации простоев и тому подобное. Оба этих аспекта – синхронизация и конкуренция делает сетевые компьютерные системы существенно нелинейными, что усложняет их аналитическое и имитационное моделирование, а рассматриваемый далее в работе подход можно рассматривать как возможную линеаризацию таких систем.

Проведем грань между дискретно-событийным моделированием и другими методами моделирования более четко. Как уже упоминалось, большинство систем моделируется по принципу «вход-состояние-выход». Принимая общеизвестные обозначения векторов: $\mathbf{u}(t)$– входа, $\mathbf{x}(t)$ – состояния, $\mathbf{y}(t)$ – выхода, динамика моделируемой системы описывается уравнениями:

$$\mathbf{x}'(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{1.1}$$

$$\mathbf{y}(t) = \mathbf{g}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{1.2}$$

с начальными условиями $t > 0$.

Уравнение (1.1) означает составление множества состояний моделируемой системы, а если принять что множество таких состояний равно *n*, а множество входных сигналов равно *m*, то получается, что необходимо моделировать *n* уравнений состояний

$$\begin{aligned} x_1'(t) &= f_1\left(x_1(t),...,x_n(t),u_1(t),...,u_m(t),t\right) \\ &\text{.....} \\ x_n'(t) &= f_n\left(x_1(t),...,x_n(t),u_1(t),...,u_m(t),t\right) \end{aligned},$$

и *k* выходных уравнений системы

$$\begin{aligned} y_1(t) &= g_1\left(x_1(t),...,x_n(t),u_1(t),...,u_m(t),t\right) \\ &\text{.....} \\ y_k(t) &= g_k\left(x_1(t),...,x_n(t),u_1(t),...,u_m(t),t\right) \end{aligned},$$

с начальными условиями $x_1(t_0) = x_1,...,x_n(t_0) = x_n$.

В зависимости от вида функций f и g, способа фиксации моментов времени t в (1.1–1.2) осуществляется непрерывное (или дискретное) моделирование нелинейной (или линейной) динамической системы.

Еще раз следует заметить, что в изменение состояний динамической системы в таких случаях моделирования всегда привязано ко времени, какими его измерениями мы бы ни пользовались, непрерывными или дискретными. Как сказали бы философы – «время первично, а события – вторичны». В дискретно-событийном моделировании важен факт фиксации события (или группы событий), а в какое время, либо интервал времени, либо через какой промежуток времени эти события фиксируются уже не столь важно. Получается, что «событие – первично, а время – вторично». В работе [40] имеются иллюстрации к вышесказанным положениям, которые приведены в доработанном виде на рис.1.1. В верхней части показано пространство состояний непрерывной системы, в средней части – дискретной системы, а в нижней части – дискретно-событийной системы. Ясно, что пространство состояний дискретно-событийной системы является дискретным и составляет события $s_1,...,s_5$, а переключение между этими состояниями происходит в соответствии наступлением некоторых событий $e_1,...,e_4$. Естественным образом, при динамической смене состояний системы может происходить возврат к предыдущим состояниям, поэтому моделирование пространства состояний будет составлять при упорядочении по хронологии цепочек событий и совпадении момента времени и события (или группы событий) последовательность пар («время», «состояние»). В данном случае для рис. 1.1. это

$$e_0 = s_2,\ \{e_1,e_2,e_3,e_4\} = \{(t_1,s_5),(t_2,s_4),(t_3,s_1),(t_4,s_3)\}. \qquad (1.3)$$

Событийное функционирование обнаруживается у широкого класса современных систем. В области информационных систем и технологий событийными являются объектно-ориентированные программные системы, сетевые компьютерные системы, интерактивные, диалоговые системы и другие.

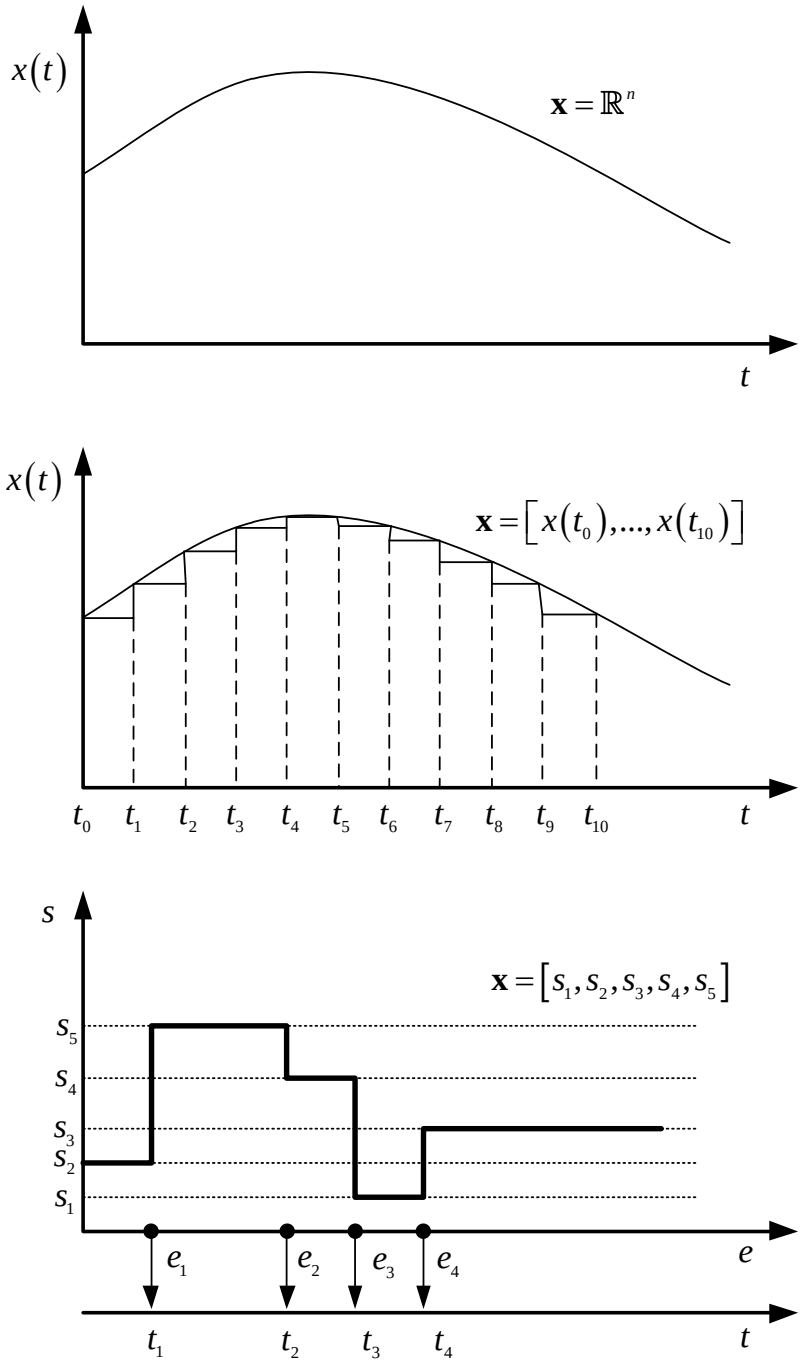

Рис. 1.1 – Непрерывная, дискретная и дискретно-событийная система

В формальном виде дискретно-событийная система представляет собой некоторую разновидность временно́го автомата [40], который представляется в следующем виде.

О п р е д е л е н и е 1 . 1 .

Модель дискретно-событийной системы представляет собой кортеж

$$DS = \left(X, E, f, \Gamma, x_0 \right), \tag{1.4}$$

где X *– конечное множество, пространство состояний системы;* E *– конечное множество событий;* f *– функция смены состояний,* $f : X \times E \to X$; Γ *– конечное множество активных (и исполняемых в текущий момент) событий;* x_0 *– начальное состояние.* □

В связи с тем, что время, как таковое не присутствует в модели (1.4), но при имитационном моделировании все же необходимо воспроизводить хронологию событий по мере упорядоченности их между собой, то (1.4) дополняется «модельными часами», связанными с множеством событий. Такие «модельные часы» представляют собой конечное множество

$$\mathbf{V} = \{\mathbf{v}_i : i \in E\},\ \mathbf{v}_i = \{v_{i,1}, v_{i,2}, ...\},\ i \in E,\ v_{i,k} \in \mathbb{R}^+,\ k = 1,2,... , \quad (1.5)$$

где $v_{i,k}$ – время жизни (продолжительность) события.

Для последовательности событий $\{e_1, e_2, ..., e_k, e_{k+1}, ...\}$ можно «включить модельные часы» (1.5), получить $\mathbf{V} = \{\mathbf{v}_i : i = 1, ..., m\}$ и генерировать события $e_{k+1} = h(x_k, \mathbf{v}_1, ..., \mathbf{v}_m)$. Динамика состояний дискретно-событийной системы при этом определяется уравнением

$$x_{k+1} = f(x_k, \mathbf{v}_1, ..., \mathbf{v}_m).$$

Таким образом, очень существенной задачей является формирование (генерация) списков событий, то есть пар вида (1.3), в зависимости от которой можно выделить событийное-ориентированное и процессно-ориентированное исполнение событий. Рассмотрим их более подробно.

Событийно-ориентированное моделирование в дискретно-событийно системе проиллюстрировано на рис.1.2.

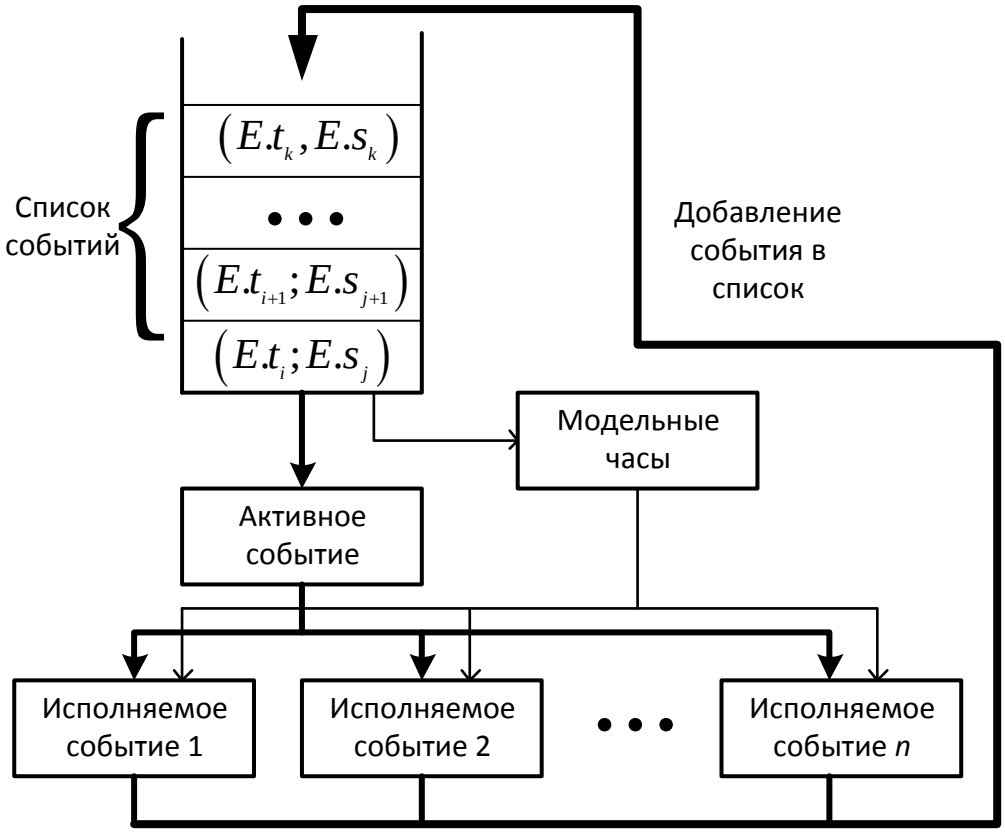

Рис. 1.2 – Событийно-ориентированное моделирование

Список событий является динамической структурой, а модельные часы содержат время последнего исполняемого события. В алгоритмическом виде моделирование заключается в следующей последовательности действий:

Алгоритм 1.1 – Событийно-ориентированное моделирование

1. Установить модельные часы в 0. Инициализировать начальный список событий, расположив их в хронологическом порядке следования. Элемент списка событий имеет структуру $(E.t_i; E.s_j)$ и характеризуется временем и типом состояния (t_i, s_j).

2. Выбирать событие E из начала списка. Если список пуст, то завершить моделирование.

3. Установить модельные часы в $E.t_i$. Проверить длительность события и при превышении времени, отведенного на моделирование его завершить.

4. В соответствии с типом события и состояния $E.s_j$ исполнить подпрограмму-обработчик события.

5. Обновлять список событий, системные переменные и структуры, помещать новое событие в список событий.

6. Продолжать моделирование, переходя к п.2.

Элементы возможной программной реализации алгоритма 1.1 показаны в примере 1.1.

Пример 1.1 – Основной событийно-ориентированный обработчик

```
event_oriented()
 {
 sim_time = 0; // Начальное время = 0
 list_init();  // Инициализация списка событий
 done = FALSE; // Признак завершения моделирования
 while(!done)  // Основной обработчик событий
  {
   next_event(status,time);//Выбор события из списка
    stime = time; // Фиксация времени
    if (stime > max_sim_time) // Проверка превышения времени
     {
      done = TRUE; // Установка признака завершения
      break; // Выход по завершению
     }
    exec_event(status); // Обработка события
    }
  }
```

Обработчик устанавливает начальное время, инициализирует список событий и в цикле выбирая следующее событие, переустанавливает модельные часы, проверяя, чтобы не было превышения максимально допустимого времени моделирования вызывает подпрограмму `exec_event(status)`. При её реализации имеет смысл запрограммировать хотя бы простейшее планирование списка событий в виде динамической очереди *FIFO* (*First In First Out*).

Во втором подходе, процессно-ориентированном исполнении событий есть возможность исполнения группы событий при их логическом объединении в процессы, как показано на рис. 1.3.

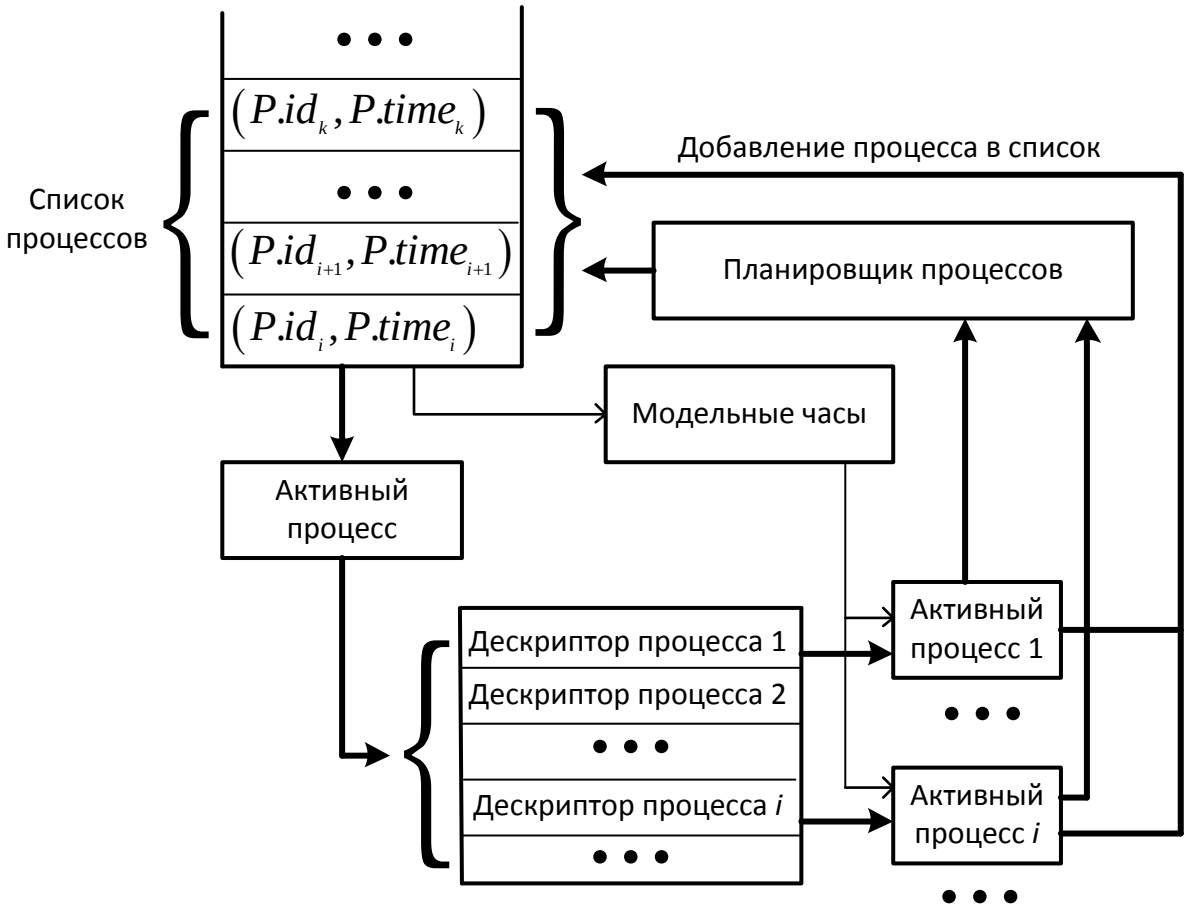

Рис. 1.3 – Процессно-ориентированное моделирование

Основным отличием данного вида от предыдущего является не только возможность объединять процессы в группы, но и также планировать переключение между процессами, разделяя при этом совместные ресурсы моделируемой системы. Общая схема такого моделирования представлена в виде алгоритма 1.2.

Алгоритм 1.2 – Процессно-ориентированное моделирование

1. Установить модельные часы в 0. Инициализировать начальный список событий, расположив их в хронологическом порядке следования. Элементы списков событий группируется в процессы, имеющие структуру $(P.id_i; P.time_i)$ и характеризуется идентификатором и временем $(id; time)$. Запустить основной цикл моделирования.

2. Создать активный процесс P из списка планируемых к исполнению событий. Если список пустой, то завершить моделирование.

3. Проверять превышение максимального времени, отведенного на моделирование. Если есть превышение, то моделирование завершить.

4. Присвоить процессу дескриптор исполнения и передать его планировщику процессов.

5. Выполнять планирование процессов: исполнение, ожидание, переключение, завершение.

6. В рамках процесса исполнять обработчики событий, генерировать новые события.

7. Перейти к п.2.

В примере 1.2 приведена схема возможной программной реализации.

Пример 1.2 – Схема процессно-ориентированного обработчика

```
event_job(descriptor) // Планировщик процессов (пример)
 {
  while (TRUE)
  {
   wait(resource); // Ожидать свободные ресурсы
   exec_event(status, time);// Исполнять обработчик события
   switch(descriptor)); // Переключать на другой процесс
   signal(release_event); // Завершить процесс
   }
  }
 process_init() // Инициализация процесса
   {
   for (i=0;i<N;i++)
       event_job(descriptor); // Планировать процесс
   if (stime < max_sim_time)
       {
        next_event(status, time) // Выбор события из списка
        stime = time; Фиксация времени
       }
       else break;
}
 process_oriented()/*Основной процессно-ориентированный
                    обработчик*/
  {
   stime = 0;
   process_init();
   simulate(stime);
}
```

Возможной простейшей стратегией планировщика может являться *FCFS* (*First Come First Served*), «первый пришел – первый обслужен», а также реализация семафорного переключения контекста процессов.

В итоге этого параграфа следует заметить, что научные исследования в области дискретно-событийного моделирования сейчас достаточно широко развиты в различных направлениях [35, 49, 51, 69], а также служат как существенное дополнение методов имитационного моделирования систем. Существуют также программные реализации дискретно-событийных систем моделирования, например, *AnyLogic*, *SimPy*, *SimEvents* и другие программные системы.

1.2. Постановка задач исследования

Как становится ясным из предыдущих параграфов главы наиболее важным аспектом дискретно-событийного моделирования является формирование последовательности событий (а также их групп). В общем эту задачу можно рассматривать как задачу генерации дискретных процессов специального вида и сформулировать следующим образом. Дано конечное множество событий $E = \{e_1, e_2, ..., e_n\}$. Предполагается, что каждое из событий $e_{i_k} \in E$ происходит в некоторый момент времени t_{i_k}. Для моделирования требуется генерировать дискретный процесс, состоящий из цепочек событий вида

$$\varphi = \{e_{i_1}, e_{i_2}, ..., e_{i_M}\}, ... \qquad (1.6)$$

где событие $e_{1_k} \in E$ происходит в момент времени t_{1_k}; $e_{2_k} \in E$ происходит в момент времени t_{2_k}, $e_{i_k} \in E$ происходит в момент времени t_{i_k}, и так далее, причем $t_{1_k} < t_{2_k} < ... < t_{i_k}$.

Очевидно, что дискретный процесс (1.6) может быть формализован с помощью различного математического аппарата, а также могут быть детализированы взаимоотношения между самими событиями. Все эти характеристики составляют спецификацию дискретно-событийной системы. Можно выделить

[53] следующие средства описания спецификаций для рассматриваемых систем:

1) по инструментарию моделирования

1.1) графические средства;

1.2) алгебраические средства;

1.3) формально-языковые и программные средства;

2) по детализации и взаимодействиям событий:

2.1) параллельные;

2.2) конкурирующие;

2.3) конфликтующие;

2.4) синхронизированные;

2.5) взаимоисключающие;

2.6) планируемые;

2.7) совместные;

2.8) периодические.

Первый пункт характеристик спецификации детализируется следующим образом:

1.1.) графические средства:

1.1.1) диаграммы «состояний-переходов», автоматные диаграммы;

1.1.2) сети Петри;

1.1.3) потоковые диаграммы;

1.1.4) диаграммы релейно-лестничной логики;

1.1.5) *Grafcet* диаграммы;

1.2) алгебраические средства:

1.2.1) булева алгебра;

1.2.2) алгебраическая теория автоматов;

1.2.3) темпоральная логика;

1.2.4) (*max*,+) и (*min*,+) алгебра;

1.3) формально-языковые и программные средства:

1.3.1) теория формальных языков;

1.3.2) языки конкурентного и параллельного программирования;

1.3.3) языки с логикой реального времени

Приведенная классификация не является исчерпывающей, а также следует заметить, что реализация этих всех возможностей в одной спецификации дискретно-событийной системы моделирования практически почти не осуществима.

Рассмотрим положения, проясняющие поставленные в диссертации задачи, которые вполне соответствуют изложению по главам.

Первой задачей, поставленной в рамках данной диссертации является развитие моделей и методов дискретно-событийного моделирования. Подходы к решению этих задач для моделирования сетевых компьютерных систем были рассмотрены в работе автора [27]. Основное внимание в развиваемом далее подходе уделяется двум вполне самостоятельным направлениям дискретно-событийного моделирования: 1) с применением алгебраических структур (*max*,+) и (*min*,+) подхода; 2) с визуализацией схем дискретно-событийных систем на основе выделения типовых структурных элементов системы в совокупности с реализуемыми ими вычислительными процедурами (*max*,+) и (*min*,+) алгебры. Решение этих задач позволяет синтезировать структуру всей моделируемой системы в виде временного событийного графа, снабженного рекуррентными уравнениями, моделирующими динамику системы.

Второй задачей, поставленной в диссертации является разработка моделей оценки пиковых (предельных) информационных нагрузок в сетевых системах. Целью является дискретно-событийное моделирование систем с индикаторами максимальных нагрузок, которые в диссертации рассматриваются как информационные нагрузки, однако позволяют распространение на другие широкие классы технических систем. Естественным практическим применением таких моделей в рассматриваемых технических системах является анализ

их производительности, выполняемый с целью оптимизации управления информационными и транспортными потоками, многопользовательским доступом к ресурсам систем, а также управления сопутствующими технико-экономическими показателями этих систем.

Третьей задачей диссертации является развитие дискретно-событийного моделирования надежности ПО. Как известно, при вводе в эксплуатацию сложных программно-управляемых программных комплексов выполняется тестирование и верификация ПО различными методами и подходами. В диссертации развивается подход, который не относится к указанным видам и стратегиям оценки надежности. Используемый подход основан на принципе «роста» надежности ПО, который основывается на утверждении, что любая, даже программная система, прошедшая тестирование и верификацию все же содержит некоторое число ошибок. В процессе эксплуатации программной системы ошибки проявляются различным образом, а для их устранения система либо останавливается, либо подвергается обновлению, либо некоторым аналогичным действиям. Число ошибок уменьшается относительно некоторой первоначальной величины, происходит рост надежности ПО и требуется смоделировать такой событийный и естественно дискретный случайный процесс. Требуется также учитывать особенности алгоритмических конструкций, используемых в ПО.

Четвертой задачей диссертации является разработка численных процедур дискретно-событийного моделирования для новых моделей и подходов, которые предложены в диссертации. Среди них особое внимание уделяется числовыми оценкам вариативности случайных точечных процессов, которые в свою очередь являются основой для разработки методов дискретно-событийного моделирования задач оценки надежности и производительности систем.

Последней, пятой задачей диссертации является разработка программных комплексов, реализующих предложенные алгоритмы вычислений. В ка-

честве парадигм программного комплекса выбраны как традиционный императивный подход, так и функциональный подход к программированию, который на взгляд автора диссертации является эффективным в отношении решения задач дискретно-событийного моделирования.

1.3 Методы (*max*,+) и (*min*,+) алгебры в дискретно-событийном моделировании

Естественно, основной интерес в дискретно-событийном моделировании представляет моделирование сложных систем, которые в общем случае являются нелинейными системами. В контексте рассматриваемых в диссертации задач упростить моделирование можно путем линеаризации моделируемой системы, которое основано на следующем преобразовании. Рассматривается пространство функций со значениями в некоторых полукольцах, причем обычные операции сложения и умножения заменяются на другие операции, связанные законом дистрибутивности. При выборе в качестве таких новых операций вместо обычного сложения – операции нахождения «максимума» (либо «минимума»), а вместо обычного умножения – операции обычного сложения получается подход, который в научной литературе называется (*max,* +), либо (*min*,+) алгеброй. Наиболее значительные результаты в этой области получены в нашей стране в работах В.П. Маслова , В.Н. Колокольцова [18], Г.Л. Литвинова [13], Н.К. Кривулина [12] и их коллег, а также называется он называется тропической или идемпотентной математикой. За рубежом в данной области значительные результаты получены Ф. Бачелли, Г. Коэном, Ж.П. Квадра [36], П Бутковичем [39], М. Гондраном, М. Мину [50], У.М. Макэнени [60] и другими.

Основной структурой в рассматриваемом подходе является (*max*,+) полукольцо, которое являющееся числовым множеством (с элементами из множества $\mathbb{R}$, включая $-\infty$ и не включая $+\infty$), снабженным операциями «максимума», играющего роль «сложения» и «сложения», играющего роль «умноже-

ния». Указанные операции (*max*,+) в специализированной литературе обозначаются соответственно $\oplus$ и $\otimes$ (то есть имеют другое значение в отличие от общепринятых – прямой суммы и прямого произведения матриц), то есть $x \oplus y = \max\{x, y\}$, $x \otimes y = x + y$. Данное (*max*,+) полукольцо является идемпотентным [18], то есть сложение $a \oplus a = a$ является идемпотентным, нулем считается $\varepsilon = -\infty$, единицей, считается $e = 0$, которая нейтральна по операции $\otimes$, а для каждого ненулевого элемента по этой операции имеется обратный элемент. Для обеих операций выполняется коммутативность и ассоциативность, а операция $\otimes$ дистрибутивна относительно $\oplus$. Такая алгебраическая структура часто называется диоидом [50]. Аналогичный подход с использованием (*min*,+) алгебраических структур рассмотрен в работе автора [26].

Данные свойства указанной алгебраической структуры оказываются существенно полезными для дискретно-событийного моделирования сетевых компьютерных систем с точки зрения оценки и достижения их максимальной производительности. Рассмотрим простой пример дискретно-событийного моделирования автоматизированной системы управления контейнерными перевозками (АСУ КП) на железнодорожном транспорте. В моделируемой системе имеются автоматизированные рабочие места (АРМ) работников АСУ КП на двух железнодорожных станциях: 1) узловой станции Ростов-Товарная; 2) припортовой станции Новороссийск. Посредством единой сети передачи данных ОАО «РЖД» АРМы АСУ КП работают в сетевом режиме и между ними передается информация, сопутствующая технологическому процессу обработки документов. Будем рассматривать 4 события (конечно, в существенно упрощенной форме), соответствующие работе оператора АРМ АСУ КП: 1) подготовка пакета исходной документации с телеграмм-натурным листом поезда; 2) ввод и передача информации в сетевом режиме АРМ АСУ КП на первой станции; 3) прием и распечатка входящей документации с телеграмм-натурным листом поезда; 4) подтверждение и корректировка принятых данных в АРМ АСУ КП на второй станции. Естественным образом, устанавлива-

ется порядок синхронизации событий, так, например, подтверждение и корректировка данных могут произойти не ранее, чем данные вообще будут получены, а подготовка документации может происходить не ранее прибытия поезда с контейнерами. Для того, чтобы не ограничиваться одним направлением и подчиняться общим принципам моделирования систем на рис.1.4 показаны технологические процессы с АРМ АСУ КП на станциях и названы в целом «подготовкой данных». Различное время технологических операций на станциях, а также приема и передачи в разных направлениях может в быть обусловлена различной производительностью АРМов, различным опытом операторов и другими аналогичными причинами.

Для дальнейшей определенности и конкретных расчетов положим, что время подготовки документации на железнодорожные платформы с контейнерами в АРМ АСУ КП занимает у оператора $x_A = 3$ минуты; ввод данных на каждый вагон формируемого поезда из 60 вагонов на станции Ростов-Товарная составляет $y_{AB} = 12$ минут при отправке телеграмм-натурного листа на станцию Новороссийск. Время, затрачиваемое на формирование принятого на припортовой станции Новороссийск телеграмм-натурного листа контейнерного поезда составляет $x_B = 2$ минуты, а время, затрачиваемое на согласование корректировок со станцией Ростов-на-Дону и внесение их в АРМ АСУ КП на станции Новороссийск составляет $y_{BA} = 10$ минут.

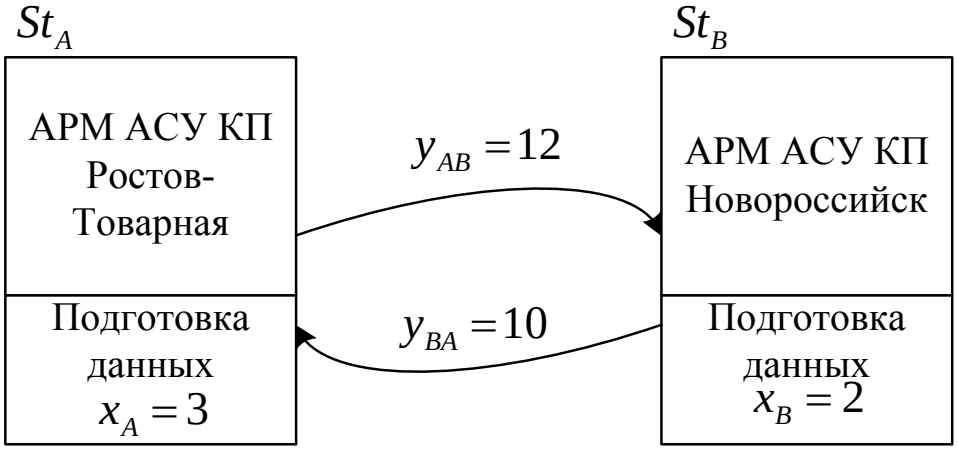

Рис. 1.4 – Графическая иллюстрация упрощенного технологического процесса контейнерной перевозки между станциями St_A и St_B

Поставим цели моделирования, состоящие в нахождении максимальных нагрузок на сетевые компьютерные АРМы АСУ КП при выполнении следующих ограничений:

1) нормативы времени на технологические операции, выполняемые операторами АРМ АСУ КП на станциях являются максимально допустимыми;

2) операторы АРМов приступают к формированию пакетов поездной документации только после прибытия поезда, однако между технологическими операциями задержек не возникает;

3) интенсивность обработки документаций в АРМах должна учитывать максимальную производительность контейнерных перевозок, а следовательно, и максимально возможное количество поездов, курсирующих на участке между St_A и St_B.

Смоделируем времена t_A отправления пакетов с телеграмм-натурными листами поездной документации со станции St_A. Пусть для начального условия $t_A(0)=0$, а, учитывая заданные ранее значения времени подготовки, ввода, обработки и корректировки данных следует, что очередной пакет поездных документов отправится не ранее, чем будет сформирован, то есть

$$t_A(1) \geq t_A(0) + x_A + \Delta, \tag{1.7}$$

а очередной прибывший пакет поездных документов будет обработан не ранее, чем будет переданы предыдущие корректировки на станцию St_A со станции St_B:

$$t_A(1) \geq t_B(0) + y_{BA} + \Delta, \tag{1.8}$$

где $t_B(0)$ – начальный момент времени отправления корректировок со станции St_B, который можно принять $t_B(0)=0$, Δ – некоторая временная задержка в технологических процессах на транспорте. Выражения (1.7) и (1.8) можно записать в более общем виде:

$$t_A(k+1) \geq t_A(k) + x_A + \Delta, \tag{1.9}$$

$$t_A(k+1) \geq t_B(k) + y_{BA} + \Delta, \tag{1.10}$$

Приняв $\Delta = 0$ и подставив значения из примера из (1.9) и (1.10), получим, что максимальная задержка отправления составит

$$t_A(k+1) = \max\left(t_A(k)+3, t_B(k)+10\right), \tag{1.11}$$

а для станции St_B аналогично (1.7) – (1.11)

$$t_B(k+1) = \max\left(t_A(k)+12, t_B(k)+2\right). \tag{1.12}$$

Так же как и в работе [51] для $k = 0,1,2,...$ по (1.11) и (1.12) рассчитаем последовательность $t_A = \{0,10,22,32,...\}$, и $t_B = \{0,12,22,34,...\}$.

Заметим, что выражения (1.9) – (1.10) и соответствующие им выражения (1.11) – (1.12) с подставленными в них максимальными значениями временных интервалов x_A, x_B, y_{AB}, y_{BA} можно переписать в общем виде следующим образом.

Пусть имеется n станций t_i, $t = 1,...,n$, а времена, необходимые на технологические операции по перемещению контейнерной документации между i-й и j-й станциями обозначим z_{ij}, тогда (1.11) и (1.12) имеет вид

$$t_i(k+1) = \max\left(t_1(k)+z_{i1}, t_2(k)+z_{i2}, ..., t_n(k)+z_{in}\right) = \max_{j=1,...,n}\left(z_{ij}+t_j(k)\right). \tag{1.12}$$

Выражение (1.12) можно сравнить с выражением для линейной рекуррентной последовательности [8] следующим образом:

$$t_i(k+1) = \sum_{j=1,...,n}\left(z_{ij}+t_j(k)\right); \tag{1.13}$$

$$t_i(k+1) = \max_{j=1,...,n}\left(z_{ij}+t_j(k)\right). \tag{1.14}$$

В (1.14) применим метод $(max,+)$-алгебры, а для того, чтобы подчеркнуть дальнейшую возможность использования методов линейной алгебры в $(max,+)$-алгебре, выражение (1.14) запишем как

$$t_i(k+1) = \bigoplus_{j=1}^{n}\left(z_{ij} \otimes t_j(k)\right). \tag{1.15}$$

Также, по аналогии с линейной алгеброй, выражение (1.15) можно записать в матричном виде:

$$t(k+1)=\mathbf{Z}\odot t(k), \qquad (1.16)$$

где $\odot$ – $(max,+)$-операции, $\mathbf{Z}$ – матрица, содержащая коэффициенты преобразований.

Рассчитаем в соответствии со значениями в приведенном выше примере по (1.15) матрица коэффициентов преобразований будет содержать времена технологических операций, поэтому $\mathbf{Z}=\begin{pmatrix}3 & 10\\ 12 & 2\end{pmatrix}$, $x(0)=\begin{pmatrix}0\\ 0\end{pmatrix}$, таким образом, $x(1)=\begin{pmatrix}(3\otimes 0)\oplus(10\otimes 0)\\ (12\otimes 0)\oplus(2\otimes 0)\end{pmatrix}=\begin{pmatrix}10\\ 12\end{pmatrix}$. Далее, принимая во внимание, что

$$x(2)=\mathbf{Z}\odot x(1)=\mathbf{Z}\odot\mathbf{Z}\odot x(0)=\begin{pmatrix}3 & 10\\ 12 & 2\end{pmatrix}\odot\begin{pmatrix}3 & 10\\ 12 & 2\end{pmatrix}\odot\begin{pmatrix}0\\ 0\end{pmatrix}=$$

$$=\begin{pmatrix}(3\otimes 3)\oplus(10\otimes 12) & (3\otimes 10)\oplus(10\otimes 2)\\ (12\otimes 3)\oplus(2\otimes 12) & (12\otimes 10)\oplus(2\otimes 12)\end{pmatrix}\odot\begin{pmatrix}0\\ 0\end{pmatrix}=\begin{pmatrix}22 & 13\\ 15 & 22\end{pmatrix}\odot\begin{pmatrix}0\\ 0\end{pmatrix}=$$

$$=\begin{pmatrix}(22\otimes 0) & \oplus & (13\otimes 0)\\ (15\otimes 0) & \oplus & (22\otimes 0)\end{pmatrix}=\begin{pmatrix}22\\ 22\end{pmatrix}$$

и так далее.

Рассмотренный подход с различными вариантами (*max*,+) и (*min*,+) идемпотентных полуколец в дальнейшем распространяется [46] для решения ряда задач оптимального управления, динамического программирования, принятия решений с помощью анализа иерархий, на основе Марковских процессов и для других задач. Вариантами таких полуколец являются:

– упомянутое в параграфе полукольцо $\mathbb{R}_{\max}=\{\mathbb{R}\cup\{-\infty\},\max,+\}$,

– (*min*,+) полукольцо $\mathbb{R}_{\min}=\{\mathbb{R}\cup\{+\infty\},\min,+\}$, в котором нулем является $+\infty$, единицей является 0;

– полные (*min*,+) и (*min*,+) полукольца $\mathbb{R}_{\max}=\{\mathbb{R}\cup\{\pm\infty\},\max,+\}$ и $\mathbb{R}_{\min}=\{\mathbb{R}\cup\{\pm\infty\},\min,+\}$, для которых $-\infty+(+\infty)=+\infty+(-\infty)=-\infty$;

– «тропическое» полукольцо $\mathbb{N}_{\min}=\mathbb{N}\cup\{+\infty,\min,+\}$;

– полукольцо $\mathbb{R}_{\max,\min} = \{\mathbb{R} \cup \{\pm\infty\}, \max, \min\}$;

– полукольцо Маслова $\mathbb{R}_h = \{\mathbb{R} \cup \{-\infty\}, \oplus_h, +\}$,

где $a \oplus_h b = h\log\left(e^{a/h} + e^{b/h}\right)$,

а также булево полукольцо $B = \left(\{true, false\}, \vee, \wedge\right)$.

Приведенный пример хоть и отражает специфику подхода к моделированию динамики системы, но не позволяет отразить взаимодействие событий. Наиболее адекватным в данном случае является применение формализмов, которые называются временными событийными графами [38]. Они являются подклассом сетей Петри и имеют для каждого места один входной и один выходной переход, а условия разметки имеют емкость не больше единицы. Временной событийный граф является двудольным размеченным графом (графом) и формально определяется следующим образом.

Определение 1.2.

Временной событийный граф G *имеет маркировку и является кортежем* $G = (P, T, A, \omega, M)$, (1.17)

где P *– конечное множество мест, «places»;* T *– конечное множество переходов, «transitions»;* $A \subseteq (P \times T) \cup (T \times P)$ *конечное множество ребер между местами (условиями возникновения событий) и переходами (событиями, изменяющими состояние системы) а также между переходами и местами;* $\omega : (P \times T) \cup (T \times P) \to \mathbb{N}^n$, *весовая функция, приписывающая вес ребрам;* $M \in \mathbb{N}^{|P|}$ *– начальная маркировка графа.*

Накладываются условия:

1) каждое ребро имеет единичный вес, $\sum_{i=1}^{|T|} \omega(p, t_i) = \sum_{i=1}^{|T|} \omega(t_i, p) = 1$;

2) множество всех входных переходов ${}^\bullet p_i \to \{t_j \in T \mid (t_j, p_i) \in A\}$ *для* p_i *-го места и выходных переходов* $p_i^\bullet \to \{t_j \in T \mid (p_i, t_j) \in A\}$ *для* p_i *равны 1,* $|{}^\bullet p_i| = |p_i^\bullet| = 1$.□

На основе определений 1.1. и 1.2 можно сформулировать определение дискретной динамической системы на основе временного событийного графа.

Определение 1.3.

Дискретная динамическая система на основе временного событийного графа является парой $(G,\mathbf{s})$, *где* G *– временной событийный граф,* $\mathbf{s}$ *– конечное множество состояний системы, начиная от начального* $s(0)$ *со сменой состояний в моменты времени, описываемой уравнениями*

$$s_i(k) = \underset{j,\omega(p_j,t_i)=1}{\left\langle \mathbf{M}_{in}^{ax} \right\rangle} \left(\tau_j(k) + \delta_j\right)$$

$$\tau_j(k+m) = s_i(k),$$

где $\left\langle \mathbf{M}_{in}^{ax} \right\rangle = \{\max, \min\}$; $s_i(k)$ *– наименьшее (или наибольшее) время* k *-го события активации перехода* t_j; $\tau_j(k)$ *– наименьшее (или наибольшее) время* k *–го события перехода* m *маркеров;* δ_j *– временная задержка в месте* p_j. □

Воспользуемся примером, приведенным в начале параграфа, чтобы разъяснить моделирование с использованием определения 1.3. На рис. 1.5. представлен временной событийный граф для рис. 1.4.

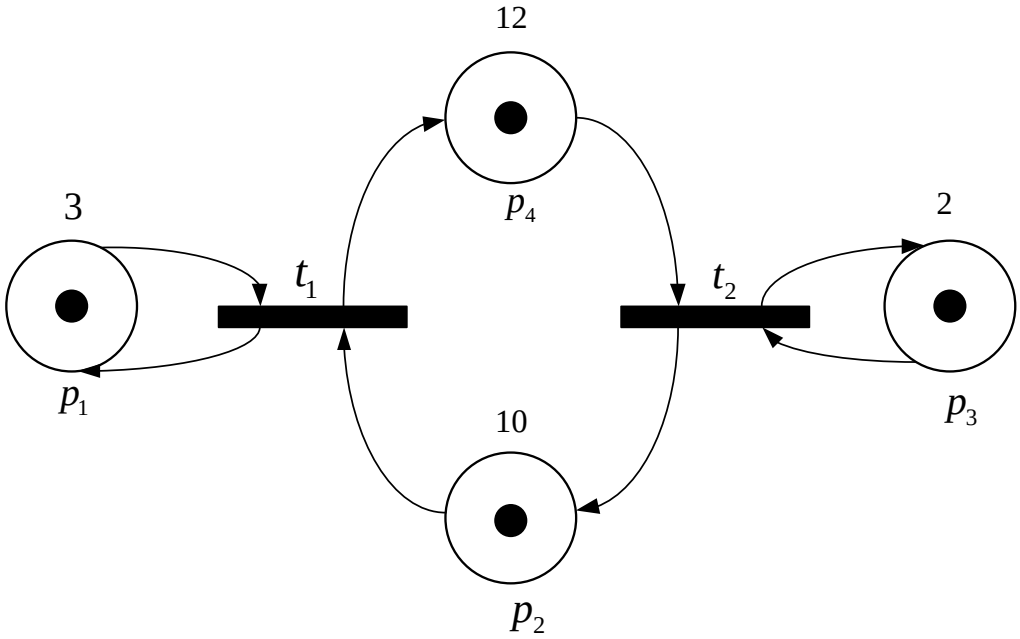

Рис. 1.5 – Временной событийный граф
для динамической системы рис. 1.4

Также, как и в предыдущем примере, динамика системы описывается уравнениями

$$s_1(k+1) = \max\left(s_1(k)+3, s_2(k)+10\right);$$

$$s_2(k+1) = \max\left(s_2(k)+2, s_3(k)+12\right);$$

$$s_3(k+1) = s_1.$$

На рис. 1.5 представлен пример достаточно простого временного событийного графа, однако структура реальных систем, подлежащих моделированию может быть гораздо сложнее.

Одним из существенный факторов, влияющих на адекватность составления моделей реальных систем (в данном случае компьютерных сетевых систем) является ограниченность времени, отводимого на доступ к разделяемым сетевым ресурсам, требование функционирования в реальном времени, сроки принятия управляющих решений и тому подобное. Данные факторы развивают подход в направлении дискретно-событийного моделирования с интервальным представлением (*max*,+) и (*min*,+) идемпотентных полуколец [14]. Интервальное идемпотентное полукольцо определяется как декартово произведение полуколец $\mathbb{R}_{\max}$ и $\mathbb{R}_{\min}$, то есть $\mathbb{R}_{\max}^{\min} = \left(\left(\mathbb{R} \cup \{-\infty\}\right) \times \left(\mathbb{R} \cup \{+\infty\}\right), \oplus, \otimes\right)$ с интервалами $\left[\underline{p}_1, \overline{p}_1\right]$ и $\left[\underline{p}_2, \overline{p}_2\right]$, а операции определяются как $\left[\underline{p}_1, \overline{p}_1\right] \oplus \left[\underline{p}_2, \overline{p}_2\right] = \left[\max\left(\underline{p}_1, \overline{p}_1\right), \min\left(\underline{p}_2, \overline{p}_2\right)\right]$,

$\left[\underline{p}_1, \overline{p}_1\right] \otimes \left[\underline{p}_2, \overline{p}_2\right] = \left[\underline{p}_1 + \overline{p}_1, \underline{p}_2 + \overline{p}_2\right]$. Нейтральный элемент $\varepsilon = \left[-\infty, +\infty\right]$, единичный элемент $e = \left[0, 0\right]$. В соответствии с таким интервальным представлением были предложены интервальные P-временные сети Петри [67], а также интервальные расширения временных событийных графов [58]. Добавление интервалов к временным событийным графам можно описать следующим образом. К каждому месту графа добавляется интервал $\left[\underline{p}, \overline{p}\right]$, где $\underline{p} \le \overline{p}$. Нижнее значение $\underline{p}$ интервала обычно связывается с минимальным временем, затрачиваемым на обработку данных, которые предшествуют возникновению

события, а верхнее значение $\overline{p}$ интервала обычно означает максимальное время обработки данных перед возникновением события, а при превышении которого оно может быть выполнено, либо, напротив, пропущено.

Формально, интервальный временной граф получается добавлением к определению 1.2 еще одной функции I, выполняющей отображение интервалов с неотрицательными граничными значениями.

Определение 1.4.

Интервальный временной событийный граф G_I *это кортеж*

$G = (P, T, A, \omega, M, I)$,

где P, T, A, ω, M *– такие как в определении 1.3,*

$I : P \to \mathbb{R}^{\min}_{\max}$, $p \to \left\{ \left[\underline{p}, \overline{p} \right]; \underline{p} \in \mathbb{R}^+, \overline{p} \in \mathbb{R}^+ \cup \{+\infty\}, \underline{p} \le \overline{p} \right\}$. □

Для упрощения процесса дискретно-событийного моделирования сложных систем в следующем параграфе разрабатывается модульный (блоковый) подход к описанию структуры моделируемых систем.

1.4 Модульный синтез схем дискретно-событийных систем на основе интервальных временных событийных графов

Рассмотренные в предыдущем параграфе примеры моделирования имеют сильные и слабые стороны. Например, временные событийные графы хорошо визуализируются и отражают последовательность событий. Однако они, как подкласс сетей Петри, во многих случаях не позволяют получить аналитическое решение для моделируемой динамической системы. Напротив, подход общей теории систем мало подвержен визуализации последовательности событий.

В предлагаемом далее подходе объединяются аналитические возможности теории систем, визуальные возможности временных событийных графов и модульный подход к имитационному моделированию аналогично теории систем и сетей массового обслуживания.

Динамика моделируемой системы описывается рекуррентными уравнениями:

$$x_i(k+1) = \bigoplus_{T_j \in T} a_{ij} \otimes x_j(k-m), \tag{1.18}$$

где $a_{ij} \in \mathbb{R}_{\max}^{\min}$,

$$a_{ij} = \bigoplus_{p_l \in P, M(p_l)=m} I_{p_l},$$

если $\{p_l \in P, M(p_l) = m\} = \varnothing$, то $a_{ij} \in \varepsilon$.

На рис. 1.6 показан пример интервального временного событийного графа.

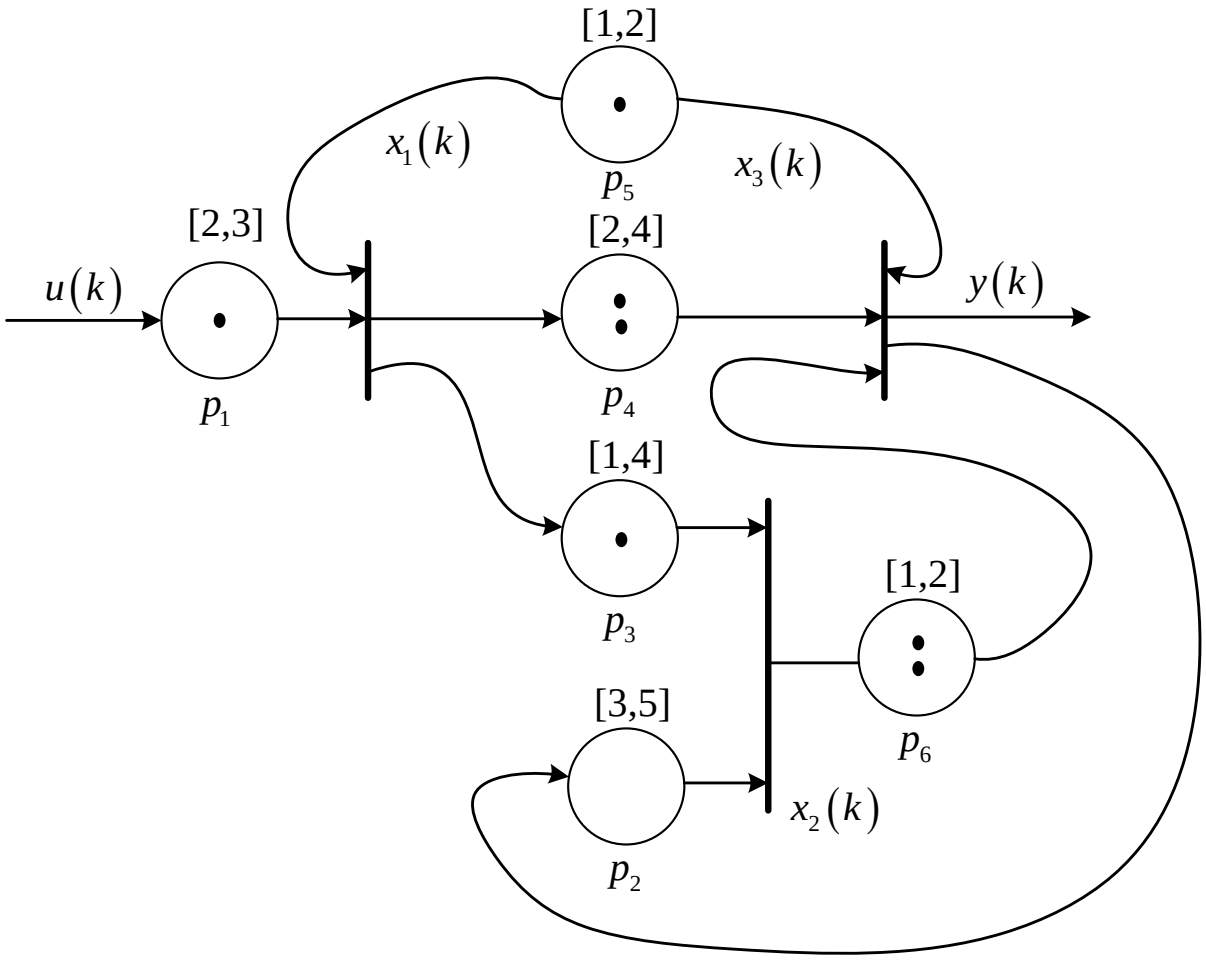

Рис. 1.6 – Интервальный временной событийный граф

Учитывая количество фишек в каждом месте ($p_1 = 1,\ p_2 = 0,\ p_3 = 1,\ p_4 = 2,\ p_5 = 1,\ p_6 = 2$) уравнения, описывающие динамику системы, согласно (1.17):

$$x_1(k)=[1,2]\otimes x_3(k-1)\oplus[2,3]\otimes u(k-1);$$

$$x_2(k)=\left([1,4]\otimes x_1(k-1)\right)\oplus\left([3,5]\otimes x_3(k)\right);$$

$$x_3(k)=\left([2,4]\otimes x_1(k-2)\right)\oplus\left([1,2]\otimes x_2(k-2)\right);$$

$$y(k)=x_3(k).$$

Рассмотрим подробнее типовые модули схем дискретно-событийного моделирования, полученные на основе интервальных временных событийных графов. Будем рассматривать модули, из которых синтезируются дискретно-событийные системы в виде блоков «вход-состояние-выход» со следующими обозначениями, представленными на рис. 1.7.

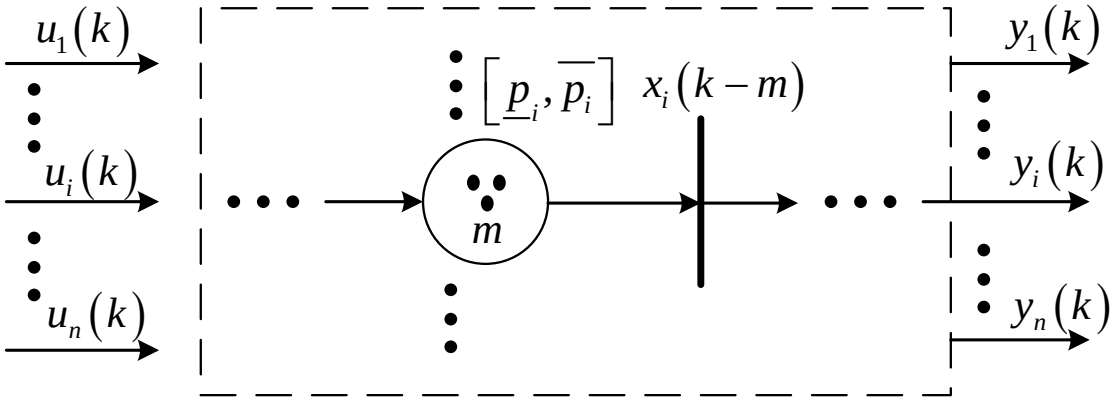

Рис. 1.7 – Дискретно-событийная система с интервальными временными событийными графами

На рис. 1.7: $u_i(k)$ – k-й момент времени i-го выходного воздействия. Динамика модулей в общем случае описывается системой уравнений:

$$\begin{cases} x_i(k+1)=\left\langle \mathbf{M}_{ax}^{in}\right\rangle\left(A_{i1}\otimes x_1(k),...,A_{in}\otimes x_n(k),B_{i1}\otimes u_1(k),...,B_{im}\otimes u_m(k)\right), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i=1,...,n; \quad (1.19) \\ y_i(k+1)=\left\langle \mathbf{M}_{ax}^{in}\right\rangle\left(C_{i1}\otimes x_1(k),...,C_{in}\otimes x_n(k),\right),\ i=1,...,p, \end{cases}$$

где $\left\langle \mathbf{M}_{ax}^{in}\right\rangle=\{\min,\max\}$; $A=\{A_{ij}\}$ – интервальная матрица состояния, $B=\{B_{ij}\}$ – интервальная матрица входа; $C=\{C_{ij}\}$ – интервальная матрица выхода, имеющая размерность $p\times n$.

В матричном виде выражение (1.19) записывается как система

$$\begin{cases} x(k+1) = \left(A \otimes x(k)\right)\left\langle \mathbf{M}_{ax}^{in} \right\rangle \left(B \otimes u(k)\right); \\ y(k) = C\left\langle \mathbf{M}_{ax}^{in} \right\rangle x(k), \end{cases}$$

где $\left\langle \mathbf{M}_{ax}^{in} \right\rangle = \{\min, \max\}$.

Модуль последовательного исполнения событий показан на рис. 1.8.

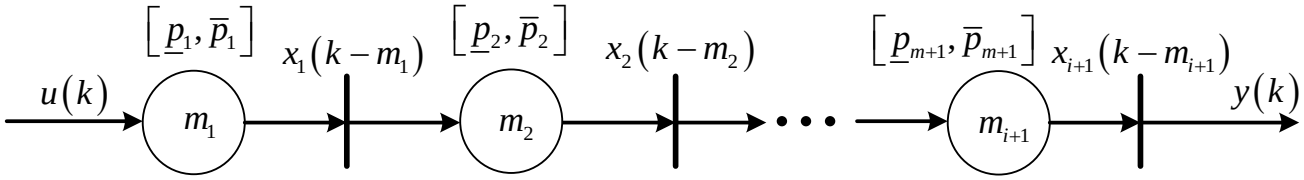

Рис. 1.8 – Интервальный временной событийный граф последовательного исполнения событий

Вычислительная схема 1.1.

$$x_1(k+1)\left\langle \mathbf{M}_{ax}^{in} \right\rangle \left(u(k), \left[\underline{p}_1, \overline{p}_1\right] \otimes x_1(k-m_1), \left[\underline{p}_2, \overline{p}_2\right] \otimes x_2(k-m_2)\right);$$

$$x_2(k+1)\left\langle \mathbf{M}_{ax}^{in} \right\rangle \left(\left[\underline{p}_2, \overline{p}_2\right] \otimes x_2(k-m_2), \left[\underline{p}_1, \overline{p}_1\right] \otimes x_1(k+1-m_1)\right);$$

$$\vdots$$

$$x_{i+1}(k+1)\left\langle \mathbf{M}_{ax}^{in} \right\rangle \left(\left[\underline{p}_{m+1}, \overline{p}_{m+1}\right] \otimes x_{i+1}, \left[\underline{p}_i, \overline{p}_i\right] \otimes x_i(k+1-m_i)\right);$$

$$y(k) = x_{i+1}(k).$$

Модуль конкурентного исполнения событий показан на рис. 1.9.

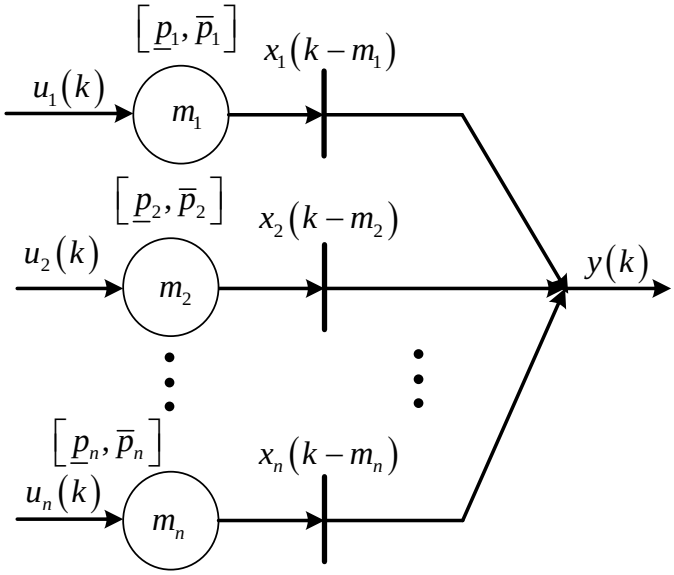

Рис. 1.9 – Интервальный временной событийный граф конкурентного исполнения событий

Вычислительная схема 1.2.

Для $i=1,2,...,n$

$$x_i(k+1)\left\langle \mathbf{M}_{ax}^{in}\right\rangle\left(\left[\underline{p}_1,\overline{p}_1\right]\otimes x_i(k-m_i),u_i(k),x_{n+1}(k)\right);$$

$$x_{n+1}(k+1)\left\langle \mathbf{M}_{ax}^{in}\right\rangle\left(\left[\underline{p}_1,\overline{p}_1\right]\otimes x_1(k+1-m_1),...,\left[\underline{p}_n,\overline{p}_n\right]\otimes x_n(k+1-m_n)\right);$$

$$y(k)=x_{n+1}(k).$$

Модуль синхронизированного исполнения событий показан на рис. 1.10.

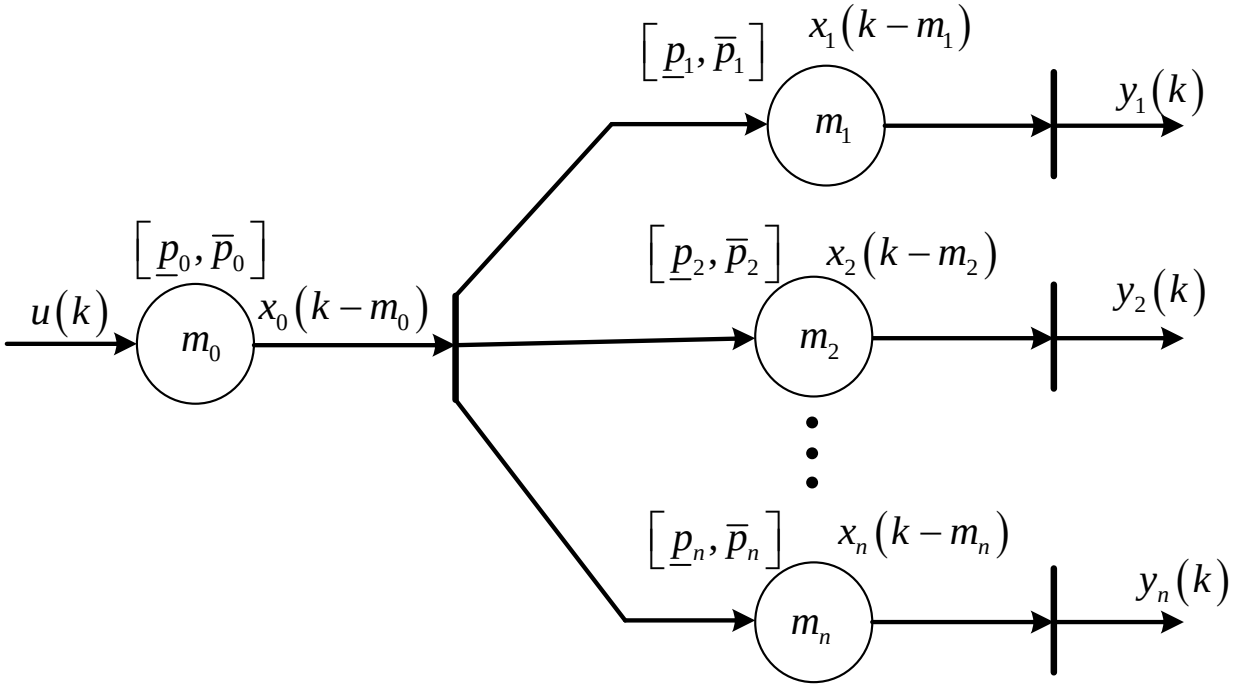

Рис. 1.10 – Интервальный временной событийный граф синхронизированного исполнения событий

Вычислительная схема 1.3.

$$x_0(k+1)=\left\langle \mathbf{M}_{ax}^{in}\right\rangle(\left[\underline{p}_0,\overline{p}_0\right]\otimes x_0(k-m_0),u(k),\left[\underline{p}_1,\overline{p}_1\right]\otimes x_1(k-m_1),$$
$$\left[\underline{p}_2,\overline{p}_2\right]\otimes x_2(k-m_2),...,\left[\underline{p}_n,\overline{p}_n\right]\otimes x_n(k-m_n));$$

Для $i=1,2,...,n$

$$x_i(k+1)\left\langle \mathbf{M}_{ax}^{in}\right\rangle\left(\left[\underline{p}_i,\overline{p}_i\right]\otimes x_i(k-m_i),\left[\underline{p}_0,\overline{p}_0\right]\otimes x_0(k+1-m_0)\right);$$

$$y_i(k)=x_i(k).$$

1.5. Выводы

1. В главе выполнен обзор методов дискретно-событийного моделирования систем, рассмотрены отличия этого подхода от известных методов аналитического и компьютерного моделирования некоторых классов сетевых систем.

2. Подробно рассмотрены методы и алгоритмы событийно-ориентированного моделирования и процессно-ориентированного моделирования сетевых систем, а также приведены вычислительные схемы их возможной программной реализации.

3. Выполнена постановка задач исследования, среди которых выделены следующие: применение алгебраических структур (*max*,+) и (*min*,+) подхода к дискретно-событийному моделированию; визуализация схем дискретно-событийных систем на основе выделения типовых структурных элементов системы в совокупности с реализуемыми ими вычислительными процедурами (*max*,+) и (*min*,+) алгебры; разработка дискретно-событийных моделей оценки производительности с индикаторами предельных и максимальных (пиковых) информационных нагрузок в сетевых системах; разработка методов дискретно-событийного моделирования надежности ПО в сетевых системах на основе процессов её роста; разработка соответствующих алгоритмов, вычислительных схем и ПО.

4. Предложен метод моделирования на основе модульного подхода к дискретно-событийному моделированию, схемы систем, полученные на основе интервальных временных событийных графов.

2 МЕТОДЫ АНАЛИТИЧЕСКОГО И ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ ВЫСОКОНАГРУЖЕННЫХ КОМПЬЮТЕРНЫХ СИСТЕМ

2.1 Источники высоких информационных нагрузок и модельные характеристики их вариативности

ИУС, применяемые во многих областях промышленности, производства и транспорта являются сетевыми вычислительными системами. В частности, ИУС на железнодорожном транспорте относятся к сложным территориально распределенным системам, строящимся на основе высокопроизводительных вычислительных комплексов и телекоммуникационных сетей. Они по своей организации разделяются на несколько уровней, начиная от микропроцессорных систем управления устройствами автоматики и телемеханики до систем аналитической обработки финансовой и управленческой информации.

Многие из подсистем в составе ИУС железнодорожном транспорте функционируют в реальном времени, что связано с требованиями обеспечения безопасности в первую очередь для пассажиров, а также для безопасной транспортировки грузов и недопущения опасных и катастрофических последствий влияния этих систем на внешнюю среду. Все технологические процессы перевозок на железнодорожном транспорте регламентированы нормативными документами, большая часть которых передается между подсистемами в электронном виде по корпоративной сети ОАО «РЖД». Также существует интенсивный информационный обмен между вычислительными комплексами автоматизации технологических процессов на станциях, перегонах, сортировочных горках и других объектах железнодорожной инфраструктуры.

Центральным звеном в рассматриваемом информационном обмене являются потоки телекоммуникационных сообщений в виде IP - телетрафика, объемы которого только в отдельных подсистемах ИУС на железнодорожном транспорте. Методы, относящиеся к построению моделей информационных

потоков ИУС были рассмотрены в работе автора [20]. Информационные потоки характеризуются большой информационной нагрузкой и интенсивностью. Например, в автоматизированной системе оперативного управления перевозками и системе интегрированной обработки дорожной ведомости только в сегменте информационно-вычислительного центра Северо-Кавказской железной дороги составляет около 2,1 Гбайт и 1,5 Гбайт в сутки соответственно. Суммарный объем телетрафика в указанном сегменте сети по в целом нескольким десяткам подсистем (на транспорте это системы СИРИУС, ЭТРАН, ОСКАР, САИ ПАЛЬМА и других) в настоящее время оценивается порядка 51,5 Гбайт в сутки, что очевидно позволяет относить некоторые из этих подсистем к высоконагруженным ИУС.

Очевидно, что ИУС такого назначения являются критичными к технологическим режимам функционирования как аппаратного, так и ПО. Например, на железнодорожном транспорте в связи с динамичностью технологических процессов организации движения, меняющейся интенсивностью обслуживания пассажиров, существенным обстоятельством является поддержка качества функционирования ИУС в условиях изменения нагрузок. Информационные нагрузки в них могут достигать предельных значений из-за возникновения пиковых факторов. Качество функционирования ИУС зависит, в частности от структуры и интенсивности информационных потоков, которые применительно на транспорте рассматривались в работе [4].

Среди причин, приводящими к высоким информационным нагрузкам, являются предельные режимы функционирования и пиковые факторы – явления, действие которых может привести к нарушению работоспособности аппаратного и ПО ИУС или ее элементов, а, следовательно, к резкому снижению качества обслуживания. Эти явления для транспортных систем ранее рассматривались в работах [5,7]. Пиковые факторы оказывают влияние на сетевые подключения её элементов или групп элементов.

Источники пиковых факторов можно классифицировать по различным классификационным схемам. Часто обобщением пиковых факторов в информационных нагрузках являются дестабилизирующие факторы, однако в диссертации основное внимание уделяется непреднамеренным воздействиям, которые могут существенно повлиять на работоспособность сетевых систем. Существенными признаками классификации информационных пиковых факторов являются:

1) место расположения источника дестабилизирующих факторов относительно информационно-управляющей системы, определяющее возможности сети в части его ликвидации или уменьшения его влияния (внутренние или внешние);

2) пространственно-временные показатели воздействий, характеризующие масштабы возможных одновременных разрушений или нарушений работоспособности элементов сети (локальные или массовые);

3) характер воздействий, определяющий «интеллект» источника и, следовательно, методы борьбы с его влияниями (преднамеренные или непреднамеренные).

В работе [6] предложена классификация пиковых факторов ИУС, которая представлена на рис. 2.1 на основе перечисленных признаков с указанием свойств ИУС, на которые влияют эти факторы.

Задачи, связанные с моделированием качества обслуживания в сетевых системах, которые не имеют информационных перегрузок и пиковых факторов хорошо изучены. Распространенными и адекватными методами моделирования на основе информационных потоков являются методы теории СМО и СеМО. В работах [4, 16] был предложен ряд аналитических и имитационных моделей информационных потоков в ИУС на железнодорожном транспорте, как имеющих, так и не имеющих свойства Марковости, эргодичности и стационарности входных и выходных потоков. Как и во всех подходах к моделированию на основе теории массового обслуживания инструментом моделирования является случайный процесс, описывающий входной и выходной поток, а

числовыми характеристиками являются усредненные показатели обслуживания (либо ожидания), такие как вероятность обслуживания (отказа в обслуживании), среднее время пребывания (ожидания) заявки в системе обслуживания, среднее число заявок, ожидающих (находящихся в системе) обслуживания и другие. Естественно, конечной целью такого моделирования является оценка производительности системы и формирование рекомендаций для её улучшения.

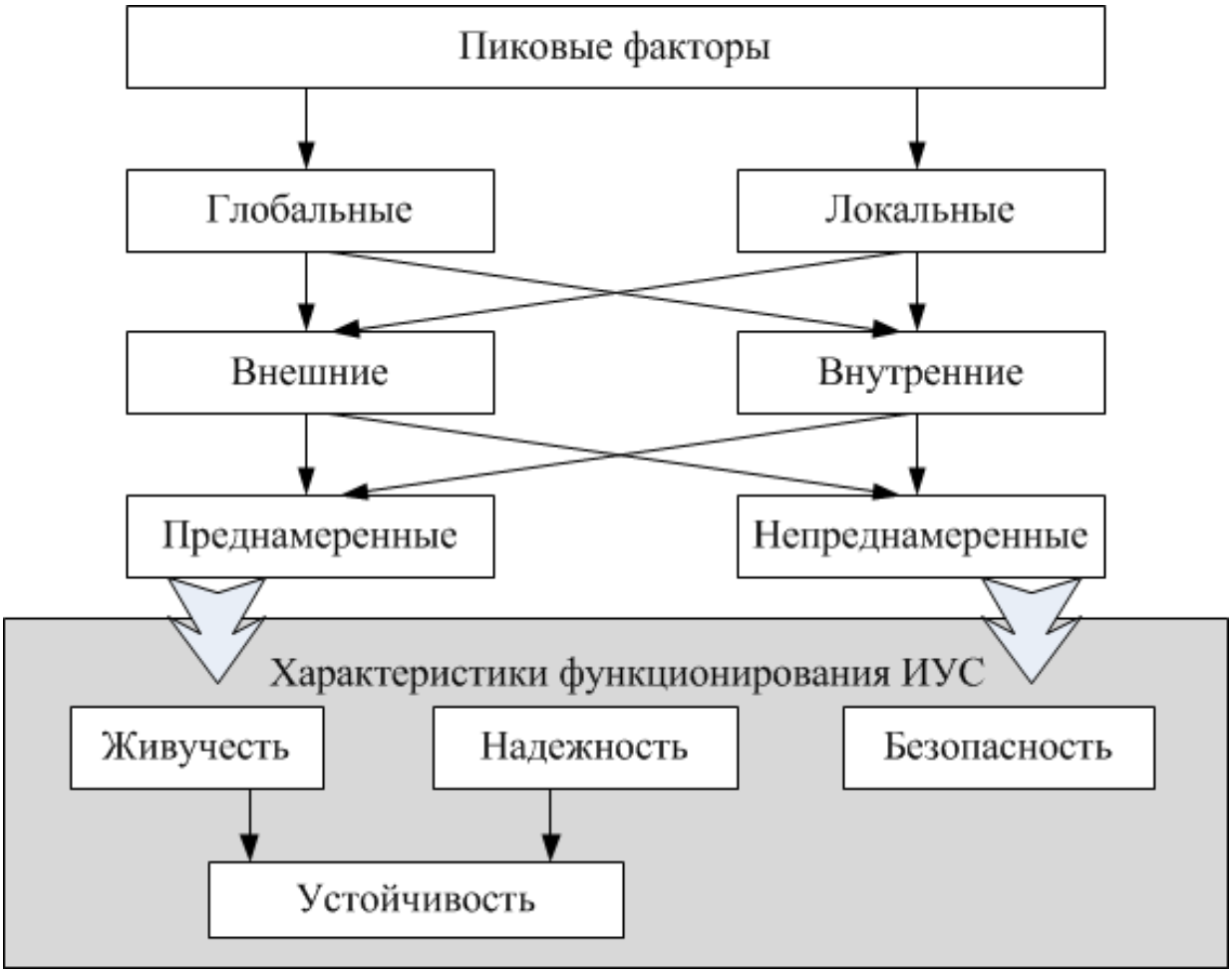

Рис. 2.1 – Классификация пиковых факторов ИУС

Предметом изучения в контексте данного исследования являются пиковые факторы, возникающие в моделируемых информационных потоках, которые, как уже ранее указывалось, могут приводить к возникновению пиковых и предельных информационных нагрузок. Исследованию подходов моделированию и измерению пиковых факторов информационных потоках на нынешнее время посвящено не так уж много научных работ, что свидетельствует, по

мнению авторов, скорее не об отсутствии интереса к данной тематике, а об отрыве разрабатываемых в теории методов от задач исследования реальных систем. Тем не менее, элементы моделирования и оценки пиковых факторов в информационных потоках присутствуют в современных монографиях по математическим методам в инженерии [45] по моделированию и оценке производительности компьютерных сетей [47] и научных работах, например, в работе [61] с обширной библиографией по данной тематике.

Одним из способов оценки пиковых факторов в информационных потоках на основе теории очередей является подход, предложенный в работе [44]. При этом рассматривается процесс $X(t)$ поступления заявок на обслуживание, в соответствии с которым некоторое количество элементов обслуживания $S(t)$ оказываются занятыми в момент времени t. Очевидно, что процесс обслуживания можно описать распределением времени обслуживания $F(x)$. Функционал оценки пиковой информационной нагрузки $z_X[F]$ представляет собой отображение $F(x)$ в скалярную величину

$$z_X[F] = \frac{Var[S(t)]}{E[S(t)]}. \qquad (2.1)$$

В (2.1) накладывается ограничение на независимость дисперсии $Var[S(t)]$ и математического ожидания $E[S(t)]$ от времени t в связи со стационарностью $X(t)$ и инвариантностью временных сдвигов. Очевидно, что соотношение (2.1) является довольно простой и эффективной мерой оценки пиковых факторов, но эта мера не лишена недостатков. Во-первых, в рассматриваемом функционале никак не отражается структура информационного потока поступления заявок на обслуживание, которая может существенно влиять на оценку пиковой составляющей информационной нагрузки ПО ИУС. Во-вторых, из контекста работы [44] становится понятным, что исходной моделью входного потока все же является случайный точечный процесс с восстановлениями, который накладывает ограничения, связанные с необходимостью

учета во входном потоке некоторого распределения времени поступления информации.

Для преодоления этих ограничений предлагается следующий подход, который далее рассмотрим подробнее. Одним из адекватных способов моделирования в нашем случае будет использование случайного точечного процесса, которых подходящим образом описывает поступление пакетного трафика [31], который в дальнейшем обрабатывается сетевым ПО ИУС. Случайный точечный процесс имеет вид

$$X(t)=\sum_{i=0}^{+\infty}\delta(t-T_i), \qquad (2.2)$$

где $\{T_i\}$, $i=0,1,...$ обозначает последовательность времени поступления пакетов с информацией, $\delta(\bullet)$ - дельта-функция Дирака.

Значение процесса $X(t)$ в некоторый момент времени равно 0, если информация не поступала. Также принимается $T_0=0$. Соответствующий точечному процессу $X(t)$ считающий процесс обозначим

$$N(t)=\sum_{i=0}^{\infty}u(t-T_i), \qquad (2.3)$$

где $u(t)$ является единичной ступенчатой функцией:

$$u(t)=\begin{cases}0, \text{ если } t<0;\\ 1, \text{ в остальных случаях.}\end{cases}$$

$N(t)$ имеет смысл, трактуемый как количество сетевых пакетов, поступивших на обработку ПО на интервале времени $(0,t]$, $t\geq 0$. Обозначим количество поступивших сетевых пакетов на интервале $(t_1,t_2]$ как $N_X(t_1,t_2)=N(t_2)-N(t_1)$. Интенсивность процесса поступления сетевых пакетов обозначим $\lambda_X \dfrac{1}{E[A_n]}$, где последовательность $\{A_n=T_n-T_{n-1}\}$ является последовательностью времен поступления между сетевыми пакетами. Если наложить на последовательность $\{A_n\}$ условие стационарности, то считающий процесс $N_\Delta(\tau,\tau+t)$ будет

эквивалентным по распределению ранее указанному процессу $N_X(t_1,t_2)$. При таких условиях ковариационная плотность точечного процесса $X(t)$ [1] определяется

$$k_{\Delta X}(t_1,t_2)=\lim_{\Delta t_1,\Delta t_2\to 0}\frac{Cov\left[N_\Delta(t_1,t_1+\Delta t_1),N_\Delta(t_2,t_2+\Delta t_2)\right]}{\Delta t_1,\Delta t_2}. \qquad (2.4)$$

В связи с тем, что предел (2.4) может не существовать, ковариационная плотность точечного процесса может быть определена как обобщенная функция из соотношения

$$Cov\left[N_\Delta(t_1,t_1+\Delta t_1),N_\Delta(t_2,t_2+\Delta t_2)\right]=\int_{t_1}^{t_1+\Delta t_1}\int_{t_2}^{t_2+\Delta t_2}k_{\Delta X}(\tau_1,\tau_2)d\tau_2 d\tau_1,$$

где $t_1,t_2,\Delta t_1,\Delta t_2,\tau_1,\tau_2\in\mathbb{R}$.

Характеристики λ_X и $k_{\Delta X}$ являются существенными для моделирования информационной нагрузки ИУС. В соответствии с описанным выше стационарным точечным процессом $X(t)$ с интенсивностью информационной загрузки λ_X и ковариационной плотностью $k_{\Delta X}$ рассмотрим обслуживание поступающих сетевых пакетов информации. Пусть для каждого поступления некоторого объема сетевых данных выделяются ресурсы, и $S(t)$ обозначает некоторое количество таких ресурсов, например, сетевых сокетов ПО. Связанный с этим процесс обслуживания обозначим в смысле распределения времени обслуживания через $F_S(x)$ со средней интенсивностью обслуживания μ_{F_S}.

Обратим теперь внимание на факт, что выражение (2.1) для функционала оценки пиковой информационной нагрузки имеет смысл общей вариативности процесса обслуживания поступающих сетевых пакетов трафика (в данном контексте – изменение усредненных соотношений обработанной и поступившей информации на данный момент времени). Для того чтобы учиты-

вать вариативность процесса поступления сетевого трафика в некоторых момент времени t предлагается уточнение формулы (2.1), и функционал оценки пиковой информационной нагрузки записывается в более общей форме:

$$z\left(X, F_S, t\right) = \frac{Var\left[S\left(X, F_S, t\right)\right]}{E\left[S\left(X, F_S, t\right)\right]}, \tag{2.5}$$

Так, формула (2.5) учитывает динамические режимы функционирования ИУС, при которых могут изменяться потоки поступающей на обработку информации. Она оказывается более адекватным способом моделирования и оценки пиковой информационной нагрузки, но, однако, требует уточнения в зависимости от ряда исходных условий моделирования.

Рассмотрим в данном параграфе еще несколько характеристик оценки вариативности информационной нагрузки в ИУС, оказывающих существенное влияние на дальнейшее построение моделей оценки пиковых информационных нагрузок. В работе [61] рассмотрены две характеристики вариативности сетевых пакетных информационных потоков, которые называются индексами дисперсии интервалов поступления данных и объемов данных. Их суть состоит в следующем. Рассмотрим случайную последовательность $\left\{X_n\right\}_{n=0}^{\infty}$, несущую смысл временных интервалов пакетного поступления данных. В дальнейшем, интервалы можно агрегировать, начиная с некоторого $i-$го номера, и дисперсию суммы n случайных переменных можно вычислить как

$$Var\left[X_{i+1} + ... + X_{i+n}\right] = n\ Var\left[X\right] + 2\sum_{j=1}^{n-1}\sum_{k=1}^{j} Cov\left[X_j, X_{j+k}\right]. \tag{2.6}$$

Нормализовав выражение (2.6) получаем индекс дисперсии интервалов

$$J_n = \frac{Var\left[X_{i+1} + ... + X_{i+n}\right]}{n\ E^2\left[X\right]}. \tag{2.7}$$

Используя автокорреляционную функцию $\rho_n = Cov\left[X_i, X_{i+n}\right] / Var\left[X\right]$ и квадратичный коэффициент вариативности временных интервалов

$c_j^2 = Var[X] / E^2[X]$ формулу (2.7) можно выразить в терминах автокорреляционных коэффициентов относительно временного лага n следующим образом

$$J_n = c_j^2 \left[1 + 2 \sum_{j=1}^{n-1} \left(1 - \frac{j}{n} \right) \rho_j \right]. \qquad (2.8)$$

Практически аналогично выражению (2.8) для оценки вариативности объемов данных d_i (числа битов, байтов и т.д.) на i – том временном интервале определяется индекс дисперсии следующим образом

$$I_t = \frac{Var\left[\sum_{i=1}^{n} d_i \right]}{E\left[\sum_{i=1}^{n} d_i \right]} = \frac{Var[d_t]}{E[d_t]} \left[1 + 2 \sum_{j=1}^{n-1} \left(1 - \frac{j}{n} \right) \rho_j \right]. \qquad (2.9)$$

2.2. Модели оценки пиковых информационных нагрузок компьютерных систем

На основе идей вышеприведенного подхода в данном параграфе развиваются методы моделирования, пригодные для оценки пиковых информационных нагрузок сетевых ИУС. Для начала запишем коэффициенты вариативности информационной нагрузки (2.7-2.9) в терминах случайных точечных и считающих процессов. Пусть $X(t)$ – это случайный процесс с дискретным временем, имеющий конечные математическое ожидание $E[X_t] = E_X$ и дисперсию $Var[X_t] = Var_X = \sigma_X^2$. Коэффициент вариации процесса

$$c_X = \frac{\sigma_X}{E_X}, \qquad (2.10)$$

Индекс дисперсии процесса

$$J_X = c_X^2 \sum_{k=-\infty}^{\infty} \rho_{X,k}. \qquad (2.11)$$

Для целей построения точных аналитических моделей оценки пиковой информационной нагрузки исключим рассмотрение семейств вероятностных

распределений с бесконечными моментами первого и высших порядков, а также ограничимся моделью экспоненциального потока обслуживания информационной нагрузки с μ – интенсивностью обслуживания.

Тогда выражение (2.5) принимает вид

$$z_X(\mu) = \lim_{t\to\infty} \frac{Var_X\left(S_\mu(t)\right)}{E_X\left(S_\mu(t)\right)}. \qquad (2.12)$$

Очевидно, что (2.12) при $\mu \to 0$ имеет вид $z_X(0) = \lim_{\mu\to 0} z_X(\mu)$, а с учетом (2.10) и (2.11) может быть определено как

$$z_X(0) = \frac{1 + c_X^2 \sum_{k=-\infty}^{\infty} \rho_{X,k}}{2} = \frac{1 + J_X}{2} \qquad (2.13)$$

Аналитическую модель функционала оценки пиковой нагрузки в общем случае, при $\mu \neq 0$ можно получить, если рассматривать случайный процесс X как вероятностный процесс с восстановлением (и без последействия). Для этого определим случайный точечный процесс $T = \{T_k\}$, $k \in \mathbb{Z}^+$ на основе процесса X как $T_k = \sum_{i=1}^{k} X_i$ и соответствующий ему считающий процесс $N_X(t_1, t_2)$, а также математическое ожидание этого процесса

$$M_X(t) = E\left[N_X(0, T_k)\right]. \qquad (2.14)$$

Тогда оценку пиковой информационной нагрузки можно выполнить, как

$$z_X(\mu) = 1 - \frac{1}{\mu E_X} + M_X(\mu), \qquad (2.15)$$

где $M_X(\mu) = \int_0^{\infty} e^{-\mu t} dM_X(t)$, $\mu \geq 0$ это преобразование Лапласа-Стилтьеса.

Для улучшения оценки (2.15) рассмотрим модификацию (2.14) в виде $M_X(t) = E\left[N_X(T_k, T_k + t)\right]$, а также применим свойство эргодичности (усреднение по времени) к рассматриваемому случайному процессу X.

В связи с этим представляется возможным рассматривать процесс X, как имеющий n - мерную функцию плотности вероятности и имеет место

У т в е р ж д е н и е 2.1.

Пусть случайный процесс $X=\{X_k\}$, $k\in\mathbb{Z}^+$ *, обладает свойством эргодичности и имеет аналитически определяемую* n *-мерную функцию плотности вероятности. Тогда*

$M_X(t)=\sum_{N=1}^{\infty}\Pr\left(\sum_{i=1}^{N}X_i\le t\right)$*, а пиковая информационная нагрузка оценивается по выражению (2.15).* □

Д о к а з а т е л ь с т в о .

Рассмотрим одну траекторию $X(s)$ случайного процесса X. Число событий, подсчитанных на интервале $\left[T_k(s),T_k(s)+t\right]$ считающим процессом N_X будет определяться как

$$N_{X(s)}\left(T_k(s),T_k(s)+t\right)=\sum_{i=1}^{\infty}u\left(T_{k+i}(s)\le T_k(s)+t\right)=\sum_{i=1}^{\infty}u\left(\sum_{j=1}^{i}X_{k+j}(s)\le t\right),$$

где u – это индикаторная функция, как в выражении (2.3).

Рассмотрим теперь математическое ожидание по всем траекториям случайного процесса X и используем свойство эргодичности, тогда

$$E_s\left[u\left(\sum_{j=1}^{i}X_{k+j}(s)\le t\right)\right]=E\left[\Pr\left(\sum_{j=1}^{i}X_{k+j}\le t\,|T_k\right)\right]=\Pr\left(\sum_{j=1}^{i}X_{k+j}\le t\right)=\Pr\left(\sum_{j=1}^{i}X_j\le t\right)$$

и для считающего процесса получаем

$E_s\left[N_{X(s)}\left(T_k(s),T_k(s)+t\right)\right]=\sum_{N=1}^{\infty}\Pr\left(\sum_{i=1}^{N}X_i\le t\right)$, а учитывая

$M_X(t)=E\left[N_X\left(T_k,T_k+t\right)\right]$ получаем

$M_X(t)=E_{k,s}\left[N_{X(s)}\left(T_k(s),T_k(s)+t\right)\right]=$

$=E_k\left[\sum_{N=1}^{\infty}\Pr\left(\sum_{i=1}^{N}X_i\le t\right)\right]=\sum_{N=1}^{\infty}\Pr\left(\sum_{i=1}^{N}X_i\le t\right).$□

Далее рассмотрим модификацию и построим аналитическую модель оценки пиковой информационной нагрузки в условиях скрытой Марковской модели, то есть в условиях, когда рассматриваемый случайный процесс X не

является наблюдаемым, то есть скрытым, а наблюдаемым является случайный процесс $Y=\{Y_k\}$, $k\in\mathbb{Z}^+$, однако, для любого k вероятностное распределение Y_k полностью определяется (управляется) X_k. Обычно, скрытая Марковская модель (*Hidden Markov Model*) представляет собой двумерный случайный процесс с дискретным временем, обозначается как $HMM(X,Y)$ и определяется следующим набором параметров $HMM(\mathbf{X},\pi,\mathbf{P},\mathbf{Y},\mathbf{Q})$. Первые три параметра относятся к определению процесса X и обозначают $\mathbf{X}$ – множество состояний скрытого процесса X, π – вектор распределения вероятностей начального состояния модели, $\mathbf{P}$ – матрица вероятностей переходов из одного состояния в другое. Четвертый и пятый параметр определяют соответственно: $\mathbf{Y}$ – множество состояний наблюдаемого процесса, $\mathbf{Q}$ – множество функций распределения или плотности вероятностей, устанавливающие связь между X и Y. Докажем утверждение 2.2.

У т в е р ж д е н и е 2.2.

Пусть случайный процесс Y определяется скрытой Марковской моделью $HMM(X,Y)=HMM(\mathbf{X},\pi,\mathbf{P},\mathbf{Y},\mathbf{Q})$ с $N=|\mathbf{X}|$ скрытыми состояниями. Также определена квадратная матрица $\mathbf{Z}(y)$, $N-$го порядка, имеющая элементами функции вида $\mathbf{Z}(y):[0,\infty)\to[0,\infty)^{N\times N}$ и задаваемые как

$$z_{i,j}(y)=p_{i,j}\int_0^y Q_j(x)dx,\ i,j\in\{1,2,...,N\}. \tag{2.16}$$

Тогда оценка пиковой нагрузки процесса Y, управляемого скрытой Марковской моделью определяется как

$$z_Y(\mu)=\pi\left(I-Z(\mu)\right)^{-1}\mathbf{1}_N-\frac{1}{\mu E_Y}, \tag{2.17}$$

где $Z(\mu)$ – преобразование Лапласа-Стилтьеса $Z(y)$, I – единичная квадратная матрица N-го порядка, $\mathbf{1}_N$ – единичный вектор длины N, $E_Y=E[Y_k]$ – математическое ожидание случайного процесса Y.□

Д о к а з а т е л ь с т в о .

Из определения *HMM* и выражения (2.16) можно сделать вывод, что вероятности $\Pr(Y_i \le t)$ можно рассчитать по формуле $\Pr(Y_i \le t) = \pi Z(t)\mathbf{1}_N$, а для суммы значений процесса Y – в виде свертки $\Pr\left(\sum_{i=1}^{N} Y_i \le t\right) = \pi Z^{*N}(t)\mathbf{1}_N$. Используя утверждение 2.1 заключаем, что $M_Y(t) = \pi\left(\sum_{i=1}^{\infty} Z^{*i}(t)\right)\mathbf{1}_N$. Выполнив преобразование Лапласа-Стилтьеса для левой и правой частей последнего выражения получим $M_Y(\mu) = \pi Z(\mu)\left(I - Z(\mu)\right)^{-1}\mathbf{1}_N$. Для получения оценки пиковой нагрузки применив выражение (2.16) получим

$$z_Y(\pi) = 1 + \pi Z(\mu)\left(I - Z(\mu)\right)^{-1}\mathbf{1}_N - \frac{1}{\mu E_Y} = \left(I - Z(\mu)\right)\left(I - Z(\mu)\right)^{-1}\mathbf{1}_N +$$

$$+\pi Z(\mu)\left(I - Z(\mu)\right)^{-1}\mathbf{1}_N - \frac{1}{\mu E_Y} = \pi\left(I - Z(\mu)\right)^{-1}\mathbf{1}_N - \frac{1}{\mu E_Y}.\square$$

Из утверждения 2.2 также можно сделать вывод о том, что если случайный процесс Y не зависит и не управляется процессом X, то оценка пиковой информационной нагрузки по формуле (2.17) упрощается и может быть определена как

$$z_Y(\mu) = \frac{1}{1 - F_Y(\mu)} - \frac{1}{\mu E_Y}, \qquad (2.18)$$

где $F_Y(\mu)$ это преобразование Лапласа-Стилтьеса для кумулятивной вероятностной функции $F_Y(y) = \Pr(Y_k \le y)$.

Формулу (2.19) можно проверить, подставив в утверждение 2.2 $N = 1$, а также $Z(y) = F_Y(y)$.

В следующем параграфе новыми методами решается задача построения аналитической модели высоконагруженной сетевой системы с обеспечением граничных условий характеристик обслуживания, а, следовательно, и производительности.

2.3. Модель анализа производительности сетевой системы с граничными показателями

Модели теории СМО и СеМО на протяжении многих лет являются одними из наиболее распространенных моделей вычислительных систем и информационных сетей. Тем не менее, возрастающая сложность топологии информационных сетей, увеличение объемов передаваемой информации, разнородность и «взрывной» характер трафика, наличие в нем свойств самоподобия и долговременной зависимости, интенсивно изучаемых на нынешнем этапе моделирования информационных сетей, делают все более затруднительными применение в этой области Марковских СеМО. Еще одним обстоятельством, ограничивающим применение хорошо изученных методов теории СМО и СеМО, является активное использование в реально существующих распределенных информационных системах и сетях принципов обеспечения некоторого гарантированного уровня качества услуг, предоставляемых пользователям сетей.

В связи с этим, в течение последнего десятилетия развивается подход к моделированию информационных систем и сетей на основе моделей очередей с обеспечением качества обслуживания, а некоторые новые методы в этом направлении предложены в работе автора диссертации [30]. Достаточно новым является подход «*network calculus*» [54, 72], пока не имеющий соответствующего устоявшегося русскоязычного термина. Очевидно, что буквальный перевод термина «*network calculus*» – «*сетевое исчисление*», не отражает сути моделей, положенных в основу этого метода моделирования, поэтому, приближенно, названный термин можно охарактеризовать как «методы моделирования инфокоммуникационных сетей с учетом требований производительности и качества обслуживания», либо как ранее «*методом сетеметрии*».

Отметим особенности вышеназванного метода моделирования, несколько отличающего его от широко известных в СМО и СеМО методов:

– входной поток имеет «взрывной характер», большие уклонения, что может означать возможность описания входного потока заявок на обслуживание вероятностными распределениями, имеющими бесконечное математическое ожидание;

– функционирование элементов обслуживания в общем случае описывается нелинейными соотношениями «вход-выход», что может привести к невозможности адекватного подбора вероятностного распределения длительности обслуживания из числа известных;

– длина очереди элемента обслуживания не является бесконечной, но может динамически изменяться для обеспечения некоторых заранее заданных гарантированных характеристик качества обслуживания;

– дисциплина обслуживания может динамически изменять как правила отказа требований в обслуживании, так и правила выбора требований на обслуживание в соответствии с некоторыми, заранее заданными граничными значениями, обеспечивающими гарантированный уровень качества обслуживания.

В соответствии с данными особенностями обслуживания математический аппарат метода сетеметрии состоит из: 1) моделей входного потока и его фильтрации; 2) моделей описания нелинейных устройств обслуживания и методов их линеаризации; 3) моделей управления параметрами обслуживания и методами описания соотношений, характеризующих выходной поток. Заметим также, что модели входного потока и модели обслуживания описываются как кумулятивные процессы.

Центральным элементом рассматриваемого подхода являются параметрические функции, описывающие какой объем информации могут обработать различные сетевые и вычислительные устройства за некоторый интервал времени. Если принять во внимание естественный факт, что любое реальное сетевое устройство, как и любая вычислительная система, обладает ограниченным набором ресурсов (памяти, пропускной способности и других характеристик) и что существуют некоторые предельные характеристики, превышение

которых ведет к ухудшению, либо к отказу в обработке информации устройством, то ему можно вполне адекватно сопоставить математическую модель в виде неубывающей в широком смысле функции, которую в работах [54, 72] называют «*service curve*». По сути, «*service curve*» представляет собой импульсный отклик системы, представляемой средствами, аналогичными алгебраической теории систем и его обоснованно можно называть *реакцией обслуживания системы*. Такое представление играет весьма важную роль в решении задачи моделирования и оценки производительности систем, поскольку предлагает линейную модель, применимую к сложным сетевым, зачастую нелинейным устройствам.

Рассмотрим некоторые математические аспекты описания вышеуказанных моделей. Пусть существует сетевое устройство в составе ИУС, например, сетевой роутер, либо коммутатор. Данное устройство представимо в виде модели из общей алгебраической теории систем [19] по типу «вход-выход»:

$$S: X \to Y, \tag{2.19}$$

где X и Y входной и выходной объекты системы S соответственно.

Для функциональной зависимости (2.19) рассматривается некоторое множество C состояний системы, формирующее данную зависимость в виде функции реакции системы $R: C \times X \to Y$. В нашем подходе, связанном лишь с оценкой производительности подробно рассматривать структуру и способы вычисления множества C нецелесообразно, поэтому применяя идемпотентную алгебру на диоиде $\left(\mathbb{R}^{+}, \wedge, +\right)$ [54] можно рассмотреть систему (2.19) в виде зависимости $S(t): x(t) \to y(t)$, где $x(t)$ и $y(t)$ функции описывающие кумулятивное накопительное поступление и обработку информации соответственно. Такие функции являются исходящими из нуля, неубывающими в широком смысле и они связаны соотношением

$$y(t) = (z \otimes x)(t) = \inf_{0 \le s \le t} \left\{ z(t-s) + x(s) \right\},$$

где $\otimes$ – обозначает свертку функций z и x относительно интервалов времени s и t.

Основными характеристиками рассматриваемой системы, формирующими функциональную связь входа и выхода являются следующие:

–функция объема обслуживания в смысле разности объемов поступившей и обработанной информации на момент времени t, включая информацию, находящуюся в очередях обслуживания, для системы с одной очередью обслуживания:

$$b(t) = x(t) - y(t);$$

– *функция виртуальной задержки обслуживания*:

$$d(t) = \inf_{0 \le t \le \tau} \{\tau \ge 0 : x(t) \le y(t+\tau)\};$$

– *функция объема поступления входного трафика*, которую называют α -кривой:

$$x(t) - x(s) \le \alpha(t-s) \Leftrightarrow x(t) \le \inf_{0 \le s \le t} \{\alpha(t-s) + x(s)\} = (\alpha \otimes x)(t);$$

– *функция реакции обслуживания*, β -кривую, являющуюся основной характеристикой, задающей модель, по которой оценивается производительность системы:

$$y(t) - x(t_0) \le \beta(t-t_0) \Leftrightarrow y(t) \le \inf_{0 \le s \le t} \{\beta(t-s) + x(s)\} = (\beta \otimes x)(t).$$

Перейдем теперь к заданию граничных условий для характеристик обслуживания в предложенной модели.

Обозначим $A(t)$ – кумулятивный входящий поток, $A^*(t)$ – кумулятивный исходящий поток на интервале времени $(0,t]$, $S(t)$ – кумулятивный поток обслуживания, $A(0) = A^*(0) = S(0) = 0$, $0 \le s \le t$, $A(s,t) = A(t) - A(s)$, $A^*(s,t) = A^*(t) - A^*(s)$, $S(s,t) = S(t) - S(s)$.

Обозначим $F = \{ f(\bullet) : \forall 0 \le x \le y, 0 \le f(x) \le f(y) \}$ – множество неотрицательных в широком смысле неубывающих функций,

$\hat{F} = \{ f(\bullet) : \forall 0 \le x \le y, 0 \le f(y) \le f(x) \}$ – множество неотрицательных в широком смысле невозрастающих функций.

У т в е р ж д е н и е 2 . 3 .

Для входного кумулятивного потока $A(t)$ *можно построить фильтрующий функционал* $A \sim < \alpha, \beta >$, $\alpha \in F$, $\beta \in \hat{F}$, *обеспечивающий заданные границы входного потока если для* $\forall t \ge 0$, $\forall x \ge 0$

$$\Pr\left\{ \sup_{0 \le s \le t} \sup_{0 \le u \le s} \left[A(u,s) - \beta(s-u) > x \right] \right\} \le \alpha(x) . \square$$

У т в е р ж д е н и е 2 .4.

Для входного кумулятивного потока можно построить функционал обслуживания $S \sim < \gamma, \delta >$, $\gamma \in F$, $\gamma \in \hat{F}$ *обеспечивающий заданные границы кумулятивного потока обслуживания* $S(t)$ *если для* $\forall t \ge 0$, $\forall x \ge 0$

$$\Pr\left\{ \sup_{0 \le s \le t} \left[A \otimes \gamma - A^*(s) \right] > x \right\} \le \delta(x), \text{ где } (f \otimes g)(t) = \inf_{0 \le s \le t} \{ f(s) + g(t-s) \} . \square$$

Утверждения 2.3 и 2.4 предназначены для определения граничных значений характеристик обслуживания при решении задачи обеспечения заданного уровня производительности системы.

2.4. Алгоритм численного моделирования систем с пиковыми информационными нагрузками

Целью данного параграфа является разработка алгоритма, позволяющего получить численную аппроксимацию выражения (2.15). В качестве основы будет использован экспоненциальный процесс восстановления (см., например, работу Д.Р. Кокса, В.Л. Смита, Теория восстановления, 1967, 299 с.). Точечный процесс $A_x(t)$ аппроксимируется экспоненциальным процессом восстановления:

$$A_x(t) \approx \alpha\left(1 - \exp(-\beta t)\right), \tag{2.20}$$

где $\alpha = \lim_{t \to \infty} A_x(t)$, β – некоторая константа.

Нахождение значений α и β является целью дальнейших вычислений. Из формулы (2.20) очевидно, что

$$M_X(t) \approx \lambda_X t + \alpha\left(1 - \exp(-\beta t)\right), \tag{2.21}$$

а соответствующая (2.21) аппроксимация функции интенсивности

$$m_X(t) \approx \lambda_X + \alpha\beta\exp(-\beta t). \tag{2.22}$$

Преобразование Лапласа для выражения (2.22) имеет вид

$$m_X(s) \approx \frac{\lambda}{s} + \frac{\alpha\beta}{s+\beta} \tag{2.23}$$

Обратимся теперь к (2.15) и, подставив в него выражение (2.23) получим, что

$$z_X(\mu) \approx z_A(\mu) = 1 + \frac{\alpha\beta}{\mu+\beta}. \tag{2.24}$$

В выражении (2.24) получена аппроксимация $z_A(\mu)$ для оценки пиковой нагрузки $z_X(\mu)$ и для определения параметров α и β предлагается следующий алгоритм.

Алгоритм 2.1. Числовая оценка пиковой информационной нагрузки

Часть 1. Расчет α.

1.1. Ввод последовательности наблюдаемых значений процесса $\{X_i\}_{i=1}^{n}$.

1.2. Расчет автокорреляционной функции процесса $\rho_X(1) = Cov[X_i, X_{i+1}] / Var[X]$.

1.3. Расчет коэффициента вариации процесса по формуле (2.10)

$$c_X = \frac{\sigma_X}{E_X}.$$

1.4. Расчет индекса дисперсии процесса по формуле (2.11)

$$J_X = c_X^2 \sum_{k=1}^{n} \rho_{X,k}.$$

1.5. Расчет оценки пиковой информационной нагрузки по формуле (2.13)

$$z_X(0) = \frac{1 + c_X^2 \sum_{k=-\infty}^{\infty} \rho_{X,k}}{2} = \frac{1 + J_X}{2}.$$

1.6. Расчет $\alpha = z_X(0) - 1$.

Часть 2. Расчет β.

Входные параметры: рассчитанные в части 1 $\rho_X(1)$ и $z_X(0)$.

2.1. Ввод начального значения μ_0, h .

2.2. Расчет $\rho_X^2(1)$.

2.3. Рекурсивный расчет оценочных значений $\hat{z}_X(\mu_j)$

2.3.1. $\hat{z}_X(\mu_j) = \sum_{k=0}^{n} \binom{\mu_j}{k} \Delta^k \hat{z}_X(0)$.

2.3.2. $$\Delta^k \hat{z}_X(\mu_j) = \begin{cases} \hat{z}_X(0), & k=0 \\ \Delta^k \hat{z}_X(\mu_j + h) - \Delta^k \hat{z}_X(\mu_j), & k=1 \\ \Delta\left[\Delta^k \hat{z}_X(\mu_j)\right], & k>1 \end{cases}$$

2.4. Расчет усредненной оценки β по формуле

$$\beta = \frac{1}{(1 - \rho_X^2(1))} \sum_{j=1}^{n} \frac{\mu_j \left(1 - \hat{z}_X(\mu_j)\right)}{\hat{z}_X(\mu_j) - z_X(0)}.$$

3. Конец алгоритма

Данный алгоритм имеет линейную сложность и обладает сходимостью.

2.5. Выводы

1. В главе проанализированы источники высоких информационных нагрузок сетевых систем, являющихся объектами для исследуемого в диссертации дискретно-событийного моделирования.

2. Среди факторов высоких информационных нагрузок выделены пиковые факторы, для которых выполнена классификация и предложен подход к

оценке пиковой информационной нагрузки функционалом специализированного вида.

3. Предложенная оценка пиковой информационной нагрузки предназначена для индикации превышения нагрузки в информационных потоках системы, описываемых случайными точечными процессами, адекватно подходящими для задач дискретно-событийного моделирования сетевых информационных систем.

4. Разработаны модификации моделей оценки пиковой информационной нагрузки для информационных потоков, описываемых многомерными случайными точечными процессами со свойством эргодичности и для случайных процессов, управляемых скрытыми Марковскими моделями

5. На основе рассматриваемых в первой главе методов дискретно-событийного моделирования с (*max*,+) и (*min*,+) алгебраическими структурами и методов сетеметрии предложена модель анализа производительности сетевой системы с установкой граничных значений показателей производительности для решения задач моделирования систем с гарантированным обслуживанием.

6. Разработан алгоритм численного расчета функционала оценки пиковой информационной нагрузки на основе аппроксимации случайных точечных процессов экспоненциальными процессами восстановления.

3 ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ В ЗАДАЧАХ ОЦЕНКИ НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1. Классификация моделей надежности программного обеспечения

Развитие области надежности ПО сетевых систем берет начало с 1970х годов. В это же время была разработана большая часть из моделей надежности ПО [34, 59, 66, 70]. Основными критериями классификации моделей выступали два достаточно простых метода исследования:

1. Исследование количества ошибок за определенный период времени (измерение времени «в режиме настенных часов» или измерение времени относительно исполнения процессов устройствами компьютера).
2. Исследование временных отрезков между ошибками.

Считалось, что модели, классифицированные данным способом взаимно не пересекаются и могут включать в себя частные случаи в каждом исследовании [43]. Одной из первых сложных моделей развития признана модель Мусса и Окумото [63]. Для построения данной модели использовался набор атрибутов, включавших в себя:

- временной отрезок;
- общее количество ошибок, которые возможно отследить за бесконечный или конечный отрезок времени;
- распределение количества ошибок, произошедших за время t (по Пуассоновскому или биноминальному типу);
- класс ошибки или функциональная форма активности ошибок за определенное время (применяется только для испытаний с конечными временными отрезками);
- тип ошибки или вид функции интенсивности ошибок, которые возможны за бесконечное время применяется только для испытаний с бесконечными временными отрезками). [63]

С развитием области разработки и практического применения ПО схема классификации моделей надежности ПО значительно расширилась. Разработаны не только новые критерии классификации, но и за счет объединения и пересечения нескольких критериев в различный моделях усложнилась структура классификации [62].

К наиболее часто употребляемым методикам и факторам надежности ПО относятся: стадия жизненного цикла разработки ПО (классификация моделей зависит от этапа, на котором рассчитывается надежность ПО), возможность раннего прогнозирования ошибок, ориентированность на информацию либо архитектуру (классификация производиться на основании проверки правильности входных/выходных данных, либо на проверке функционального наполнения ПО), рост надежности ПО в процессе выявления и исправления ошибок и т.д.

Классификация по данным факторам представляется наиболее полной, позволяя представить не только сами модели, но и взаимосвязь между ними. Иллюстрация этого положения представлена на рис 3.1.

Количество моделей надежности на сегодняшний день превышает сотню и продолжает расти [9,34]. В зависимости от точки зрения на понятие надежности и глубины исследования выявляются различные критерии и нюансы. Поэтому построение исчерпывающей схемы представляется делом весьма трудоемким, а восстановление всех связей между моделями и критериями может основательно запутать процесс. Представленная схема классификации моделей надежности ПО, как и рассмотренные критерии, не является единственной.

Существует схема, классифицирующая все модели надежности на аналитические и эмпирические. Эмпирические модели надежности, в свою очередь, подразделяют на модели сложности и модели, определяющие время, необходимое на «доводку» программы.

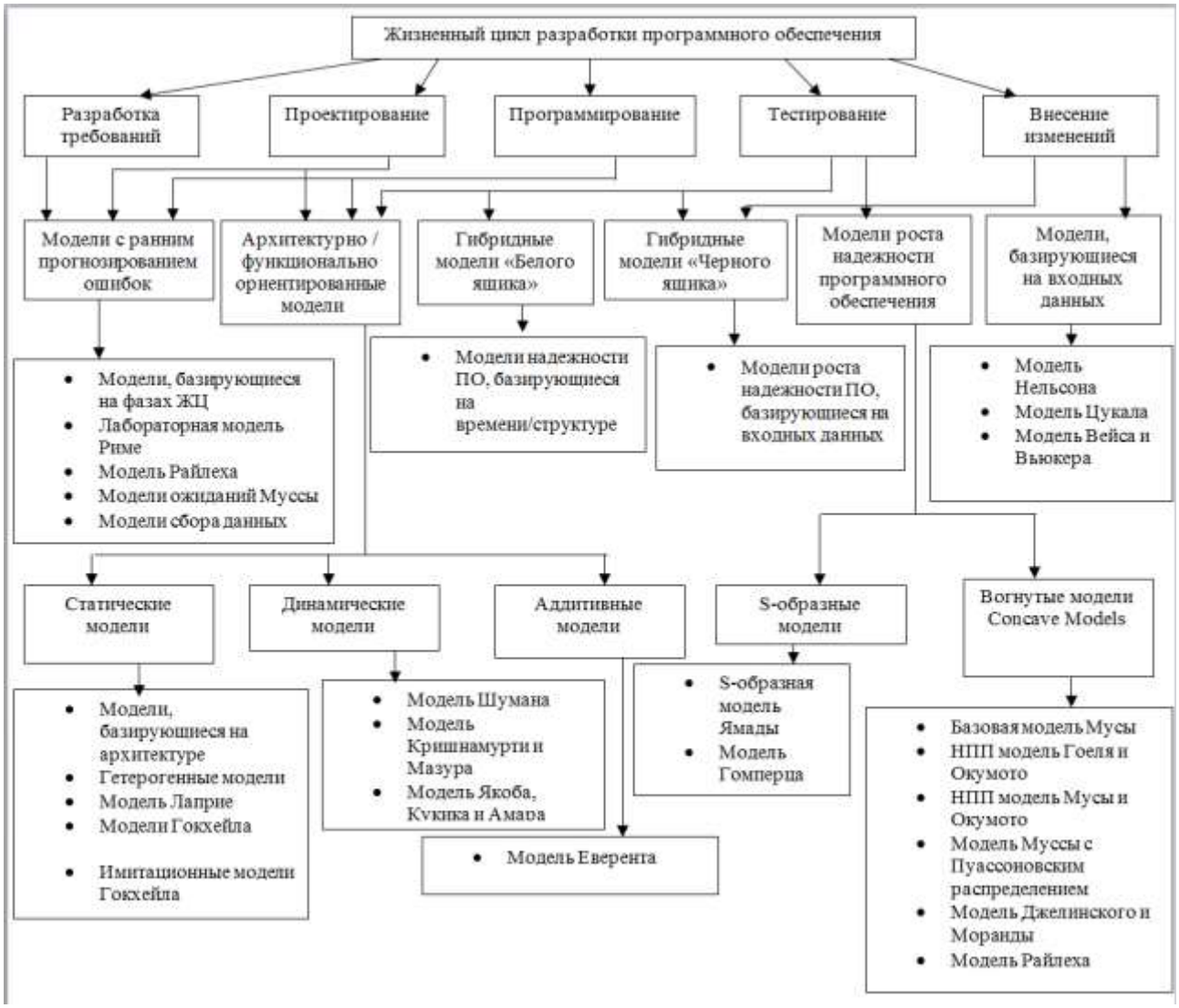

Рис. 3.1. – Классификация моделей надежности ПО

Так же аналитические модели надежности делятся на динамические (дискретные и непрерывные) и статические (по области ошибок, по области данных). В качестве примера пересечения различных квалификаций можно привести модель Шумана. Классификация на рис. 3.1 относит модель Шумана к моделям рассматриваемым на этапе проектирования ПО и к архитектурно ориентированным моделям. Другая схема классификации относит модель Шумана к аналитическим динамическим дискретным моделям.

Для большей наглядности применения представленного материала рассмотрим некоторые формулы расчета надежности ПО для различных моделей.

Формула расчета надежности одной из базовых моделей Муссы выглядит следующим образом [63]:

$$P(f_i = x) = \exp\left[-\left(\mu_{(t_i)} - \mu_{(t_{i-1})}\right)\right]\left[\left(\mu_{(t_i)} - \mu_{(t_{i-1})}\right)\right]^x / x! \qquad (3.1)$$

где $\mu_{(t_i)}$ – среднее число ошибок на момент времени t_i, $x = 0, 1, \ldots$

Модель Джелинского – Моранды рассматривает временные периоды между сбоями ПО.

$$f(X_i \mid T_{i-1}) = z(X_i \mid T_{i-1})\exp\left(-z(X_i \mid T_{i-1})X_i\right) = \\ = \phi\left[N-(i-1)\right]\exp\left(-\phi\left[N-(i-1)\right]X_i\right) \qquad (3.2)$$

где $X_i = T_i - T_{i-1}$ – время между программными сбоями, $i = 1, \ldots, n$.

Соответственно X_i, является независимой экспоненциально распределенной случайной величиной со значением функции равной:

$$\frac{1}{\phi\left(N-(i-1)\right)} = 1/\,z(X_i \mid T_{i-1}). \qquad (3.3)$$

Модель Шумана существенно похожа на упомянутую модель Джелинского – Моранды. Функция надежности в ней выражается в следующем виде:

$$Z(t) = k\left[\frac{N}{I} - n_c(\tau)\right], \qquad (3.4)$$

где T – время функционирования ПО; I – количество инструкций (команд) в анализируемом ПО; τ – время анализа (тестирования, дебаггинга) ПО; $n_c(\tau)$ – количество ошибок, удаленных из ПО за время анализа τ; k – коэффициент пропорциональности.

Модель Шика-Волвертона, принимая во внимание все допущения модели Джелинского –Моранды, добавляет следующие допущения:

1) ПО имеет N фиксированных ошибок перед началом анализа;

2) время до обнаружения некоторой ошибки a, обозначаемое T_a имеет распределение Вейбулла с параметрами α и β;

3) число обнаруживаемых на различных интервалах ошибок в ПО не зависит от общего количества ошибок в ПО.

Модель имеет вид, где интенсивность

$$\lambda(e) = Nf_a(t) = N\alpha\beta t^{a^{-1}}\exp\left(-\beta t^{\alpha}\right)$$

среднее значение

$$\mu(t) = NF_a(t) = N\left(1 - \exp\left(-\beta t^{\alpha}\right)\right),$$

функция надежности

$$Z\left(t \middle| t_{i-1}\right) = \left(N - i + 1\right)\alpha\beta\left(t + t_{i-1}\right)^{\alpha-1}, \qquad (3.5)$$

где $t_{i-1} \le t + t_{i-1} < t_i$.

В данном параграфе была представлена одна из возможных схем классификации моделей надежности ПО, также рассмотрены наиболее часто встречаемые критерии надежности ПО и показаны формулы расчета для некоторых моделей надежности.

3.2. Динамические и статические модели оценки надежности программного обеспечение

Как уже упоминалось в предыдущем параграфе, рассматриваемые модели можно классифицировать на модели динамические [21] и модели статические [22].

Динамические модели оценки надежности ПО принято делить на дискретные и непрерывные. Примерами непрерывных динамических моделей могут служить: модель Джелинского - Моранды и модель переходных вероятностей Маркова. Основными постулатами модели Джелинского - Моранды считаются: 1) время между двумя последовательными ошибками имеет экспоненциальное распределение; 2) интенсивность ошибок программы пропорциональна их количеству, не выявленному при тестировании и эксплуатации. Таким образом, вероятность безотказного функционирования ПО в зависимости от времени t_i,равна (3.6):

$$P\left(t_i\right) = \exp(-\lambda_i t_i), \qquad (3.6)$$

где интенсивность ошибок определяется выражением (3.7):

$$\lambda_i = C_D\left(N - \left(i - 1\right)\right). \qquad (3.7)$$

Здесь C_D – коэффициент пропорциональности, N – первоначальное количество ошибок в программе, $(i-1)$ – точка отсчета времени с момента последнего отказа.

Используя метод максимума правдоподобия по (3.6), обозначая через *k* номер выявленной ошибки, можно установить вид функции правдоподобия:

$$F = \prod_{i=1}^{k-1} C_D\left(N-i+1\right)\exp(-C_D\left(N-i+1\right)t_i). \tag{3.8}$$

Функция правдоподобия логарифмического вида определяется как:

$$L = \ln F = \sum_{i=1}^{k-1}\left[\ln\left(C_D\left(N-i+1\right)\right) - C_D\left(N-i+1\right)t_i\right]. \tag{3.9}$$

Таким образом, можно установить условия нахождения экстремумов:

$$\frac{\partial L}{\partial C_D} = \sum_{i-1}^{k-1}\left[\frac{1}{C_D} - \left(N-i+1\right)t_i\right] = 0, \tag{3.10}$$

$$\frac{\partial L}{\partial N} = \sum_{i-1}^{k-1}\left[\frac{1}{N-i+1} - C_D t_i\right] = 0. \tag{3.11}$$

Получаем (3.12):

$$C_D = \frac{\sum_{i=1}^{k-1}\frac{1}{N-i+1}}{\sum_{i=1}^{k-1} t_i}. \tag{3.12}$$

Подставляя (3.12) в (3.10). Получаем (3.13):

$$\left(k-1\right)\cdot\frac{\sum_{i=1}^{k-1} t_i}{\sum_{i=1}^{k-1}\frac{1}{N-i+1}} = \sum_{i=1}^{k-1}\left(N-i+1\right)t_i. \tag{3.13}$$

Примером дискретных моделей могут служить: модель Муса, модель Шумана, модель Шика-Волвертона. Модель, разработанная Шиком и Волвертоном, применяется для случая, когда в рассматриваемом интервале времени возникает более одной ошибки. Вероятность обнаружения ошибки возрастает

с течением времени. Интенсивность обнаружения ошибок – постоянна в течение интервала времени t_i и пропорциональна количеству ошибок оставшихся в программе по истечению $(i-1)$:

$$\lambda_i = C\left(N - n_{i+1}\right)\left(T_{i-1} + t_i/2\right), \tag{3.14}$$

где C – коэффициент пропорциональности; N – число первоначальных ошибок, n_{i-1} – число ошибок оставшихся в программе по истечению $(i-1)$ интервала; T_{i-1} – суммарное время затраченное на тестирование в течении $(i-1)$ этапов; t_{i-1} – среднее время выполнения программы в текущем интервале. Остальные расчеты аналогичны расчетам для модели Джелинского-Моранды.

Модель Муса оценивает надежность ПО на этапе внедрения в эксплуатацию по результатам этапа тестирования. Обозначают суммарное время тестирования – T ,число отказов – n. И получают формулу средней наработки до отказа по модели Муса:

$$\tau = \tau_0 \exp\left(\frac{CT}{n\tau_0}\right), \tag{3.15}$$

где τ_0 – средняя наработка до отказа до начала тестирования, C – коэффициент, учитывающий уплотнение тестового времени в сравнении с временем реальной эксплуатации.

Существует ряд базовых статических моделей надежности ПО таких, как модель Миллса, модель Липова и модель последовательности испытаний Бернулли.

Для модели Миллса, перед началом испытаний в ПО вносят ряд известных ошибок и фиксируют их в журнале искусственных ошибок. Предполагается, что искусственно внесенные и собственные ошибки имеют равные вероятностные шансы быть обнаруженными. На основании статистики, проводимых испытаний, и журнала об искусственных ошибках проводят оценку надежности. По выражению (3.16) определяют первоначальное количество ошибок в программе – N .

$$N = \frac{Sn}{V}, \tag{3.16}$$

где S – число искусственно добавленных ошибок, n – число найденных собственных ошибок, V – общее количество найденных в результате испытаний ошибок. Если в процессе тестовых испытаний в программе были найдены все S искусственных ошибок и n собственных ошибок, то количество ошибок $N = n$. Формула (3.17) характеризует уровень доверия к прогнозу, где C – мера доверия к модели.

$$C = \frac{S}{S + K + 1}. \tag{3.17}$$

Вероятность обнаружения n собственных ошибок и V внесенных ошибок рассчитывается как:

$$Q(n,V) = \frac{m}{n+V} \cdot q^{n+V} \cdot (1-q)^{m-n-\frac{V \cdot \frac{N}{n} - \frac{S}{V}}{\left(\frac{N+S}{n+V}\right)}}, \tag{3.18}$$

где m – количество тестов, q – вероятность обнаружения ошибки в каждом из m тестов, рассчитываемая по формуле $q = (n+V)/n$, S – число искусственно добавленных ошибок, N – количество собственных ошибок.

Для использования модели Липова, должен выполнятся ряд условий: $N \geq n \geq 0$, $S \geq V \geq 0$, $m \geq n+V \geq 0$.

Для описания модели последовательности испытаний Бернулли рассматривают класс простых программ, имеющих единственный вход и единственный выход, т.е. не содержащих бесконечных циклических операций. Все возможные результаты испытания программы разделяют на два блока: правильные и ошибочные. Предполагают, что любой результат можно отнести к одному из двух блоков. Классическая вероятностная модель последовательности испытаний Бернулли базируется на том, что пространство элементарных событий содержит 2^n точек, где n – число испытаний, вероятность ошибоч-

ного результата испытания – p, вероятность правильного результата испытания – $(1-p)$. Вероятность того, что n испытаний приведут к ошибочному результату, выражается биноминальной формулой (3.19).

$$B(p,n,k) = C_n^k \cdot p^k \cdot (1-p)^{(n-k)}, \tag{3.19}$$

где C_n^k – число сочетаний; величина p неизвестна, но известны n и k; величина B, как функция p имеет максимум при $p = k / n$.

Надежность ПО будет рассчитывается по (3.20):

$$R = \frac{n-k}{n}, \tag{3.20}$$

значения которой лежат в интервале $[0,1]$. Чем ближе к единице, тем выше надежность ПО и наоборот. Все испытания являются независимыми и результаты предыдущего испытания не влияют на результаты последующего.

3.3. Дискретно-событийный подход к оценке надежности программного обеспечения

В целом, дискретно-событийный подход, опирается на упомянутые в предыдущем параграфе идеи роста надежности ПО, однако в отличие от известных моделей будет использоваться покомпонентная технология моделирования событий возникновения ошибок на входе и выходе программного компонента. Процессы обнаружения и устранения ошибок моделируются случайными точечными процессами, а времена обнаружения ошибок сопоставляются с событиями, при возникновении которых требуется рассчитать вероятностные оценки надежности программных компонентов, которые в зависимости от реализуемых ими алгоритмов будут иметь разные формулы для расчета.

Процедуры моделирования в предлагаемом подходе можно разделить на две группы. Первая группа предназначена для генерации процессов, имитирующих появление ошибок в ПО. Сгенерированные модельные значения разделяются на несколько классов и составляют исходные данные для следующей группы процедур, которая предназначена для оценки системной надежности

компонентного ПО. Модельные значения, которые разделены на классы, сопоставляются с различными классами ошибок в событийной модели типа «вход-структура-выход». При этом анализируемый с точки зрения надежности программный компонент также относится к одному из классов в зависимости от используемых в нем программных конструкций. Каждому варианту программной конструкции соответствует отдельный вариант расчета оцениваемой надежности ПО.

Рассмотрим подробнее модели первой группы. Определим следующие переменные:

1) $N(t)$ – кумулятивное (накопленное) число ошибок ПО, выявленное на стадиях разработки и тестирования до временного момента t;

2) T_i – i-ый интервал времени возникновения, либо обнаружения ошибки $(T_0 = 0,\ i = 1,2,...)$;

3) X_i – интервал времени между $(i-1)$ и i-й ошибками $(X_0 = 0,\ i = 1,2,...)$.

На рис. 3.1 показано возникновение события $\{N(t) = i\}$, которое означает, что было обнаружено i ошибок к моменту времени t.

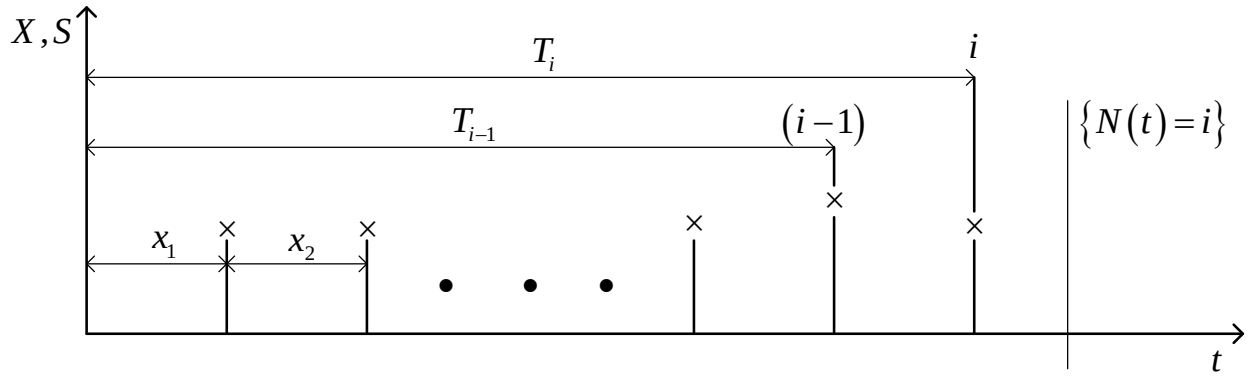

Рис. 3.1 – Дискретно-событийный процесс возникновения и обнаружения ошибок ПО

Таким образом, имеем следующий принцип моделирования:

$$T_i = \sum_{k=1}^{i} x_k,\ x_i = T_i - T_{i-1}, \qquad (3.21)$$

Обозначим $S_i(x)$ – уровень ошибок, т.е. величину, пропорциональную числу не обнаруженных (и, следовательно, не исправленных) ошибок для каждого из интервалов времени x_i, $i=1,2,...$, общий подход к дискретно-событийному моделированию можно определить как

$$S_i(x)=M_i(x)\lambda_i(x),\ i=1,2,...,N,\ x\geq 0,\ \lambda(x)>0, \qquad (3.22)$$

где $M_i(x)$ – функция, задающая начальные условия модели, $\lambda(x)$ – функция интенсивности ошибок.

Естественно, в законе моделирования (3.22) функции $M_i(x)$ и $\lambda_i(x)$ могут быть заданы различными зависимостями, начиная от константы и до выражения, имеющего сложную форму. Сама же функция надежности ПО, обозначим ее $R_n(x)$, исходя из (3.22) также может иметь разный вид, например:

$$R_n(x)=\exp\left[-\int_0^x S_i(x)dx\right],\ i=1,2,...,$$

а числовая характеристика, определяющая среднее время между ошибками может иметь вид

$$E[X_n]=\int_0^\infty R_n(x)dx.$$

Последовательность процедур моделирования, составляющих первую группу методов моделирования, следующая.

1. *Установление исходных положений моделирования.*

1.1 Ошибки ПО возникают в случайные интервалы времени, отсчитываемые последовательно на одной временной оси, имеется некоторое число скрытых ошибок ПО, которые можно обнаружить на этапах разработки и тестирования.

1.2 Ошибка в ПО вызывает ошибочное состояние всей системы и требует ее исправления для восстановления функциональности системы.

1.3 Процесс исправления ошибки выполняется немедленно, новых ошибок при этом не вносится.

2. *Выбор математического аппарата моделирования.*

Выбираем случайный считающий процесс $N(t)$, $t \geq 0$, подсчитывающий кумулятивное число ошибок в ПО, детектированных ко времени t.

Условия и ограничения следующие:

$N(0) = 0$.

2.1 $\{N(t),\ t \geq 0\}$ имеет независимые приращения.

2.3 $Pr\{N(t+\Delta t) - N(t) = 1\} = h(t\Delta t)\mathrm{o}(\Delta t)$ – где $h(t)$ – функция мгновенной интенсивности детектирования ошибок.

2.4 $Pr\{N(t+\Delta t) - N(t) \geq 2\} = \mathrm{o}(\Delta t)$ – в один момент времени обнаруживается не более одной ошибки.

3. *Генерация модельного процесса*.

Генерируем неоднородный случайный пуассоновский процесс

$$Pr\{N(t) = n\} = \frac{\{H(t)\}^n}{n!}\exp[-H(t)],\ n = 0,1,2,... \qquad (3.23)$$

где $H(t)$ – ожидаемое кумулятивное число детектированных ошибок,

$$E[N(t) = n] = H(t) = \int_0^t h(x)dx,$$

где $h(x)$ – интенсивность обнаружения и исправления ошибок.

4. *Переход к численному и алгоритмическому моделированию процессов.*

В упрощенной форме выражение (3.23) моделируется дискретным неоднородным пуассоновским процессом со средним кумулятивным количеством детектированных ошибок D_n следующим образом:

$$Pr\{N_x = n\} = \frac{[D_n]^x}{x!}\exp[-D_n],\ (x, n = 0,1,2,...). \qquad (3.24)$$

В дальнейшем, вместо выражения (3.24) могут использоваться другие различные дискретные аналоги выражения (3.23).

Рассмотрим один из таких аналогов, построенный на основе экспоненциальной модели роста, которая обсуждалась в предыдущем параграфе. Пусть H_n обозначает кумулятивное число обнаруженных ошибок за некоторые n

периодов эксплуатации ПО, его тестирования, отладки и тому подобное. Процесс роста надежности ПО описывается разностным уравнением:

$$H_{n+1} - H_n = \sigma\beta(\alpha - H_n), \qquad (3.25)$$

где σ – некоторая константа, α – ожидаемое число ошибок на бесконечно длинном интервале времени, β – интенсивность обнаружения ошибок.

Для численной оценки параметров α и β из (3.25) составим уравнение регрессии

$$Y_n = A + BH_n, \qquad (3.26)$$

где $Y_n = H_{n+1} - H_n$, $A = \delta\alpha\beta$, $B = -\delta\beta$.

Имея наблюдаемые, статистические данные по числу обнаруженных ошибок, назовем их A и B можно получить оценки требуемых параметров из (3.25): $\alpha = -\dfrac{A}{B}$ и $\beta = -\dfrac{B}{\delta}$.

Завершающий этап генерации процессов возникновения ошибок вынесен в отдельный следующий 3.4 параграф диссертации.

Перейдем теперь к процедурам второй группы. Здесь разработана компонентная модель оценки надежности ПО, учитывающая несколько классов ошибок.

5. *Формальное представление компонентной модели оценки надежности ПО.*

О п р е д е л е н и е 3 .1.

Компонентную модель оценки надежности ПО, учитывающую структуру программных конструкций и классы ошибок определим следующим образом. Имеется три класса ошибок:

1) $F_A = \{F_{A_1}, F_{A_2}, ..., F_{A_K}\}$ *– некритические, не воздействующие на выход своего и вход другого блока;*

2) $F_B = \{F_{B_1}, F_{B_2}, ..., F_{B_L}\}$ *– переходящие, воздействующие на выход своего и вход другого блока;*

3) $F_C = \{F_{C_1}, F_{C_2}, ..., F_{C_M}\}$ *– критические, воздействующие на выход своего блока и не воздействующие на вход другого блока.*

Оценкой надежности компонента с различной программной конструкцией будем считать вероятность возникновения события, связанного с проявлением ошибки любого вышеуказанного класса, рассматривая программный компонент как систему «вход-структура-выход»

$\Pr(I,O)$, $I \in F_B$, $O \in (F_B \cup F_C)$, *причем*

$\sum_{O \in (F_B \cup F_C)} \Pr(I,O) = 1$, $\forall I \in F_B$. □

6. *Варианты расчетов возникновения события, сигнализирующего появление ошибки ПО*

В связи с заявленным типом модели «вход-структура-выход» рассмотрим варианты расчета вероятности возникновения события, вызванного ошибкой в зависимости от структуры программного компонента

6.1. *Последовательная структура*

На рис. 3.2 показана последовательная структура исполнения алгоритмов $A_1, A_2, ..., A_n$ программного компонента.

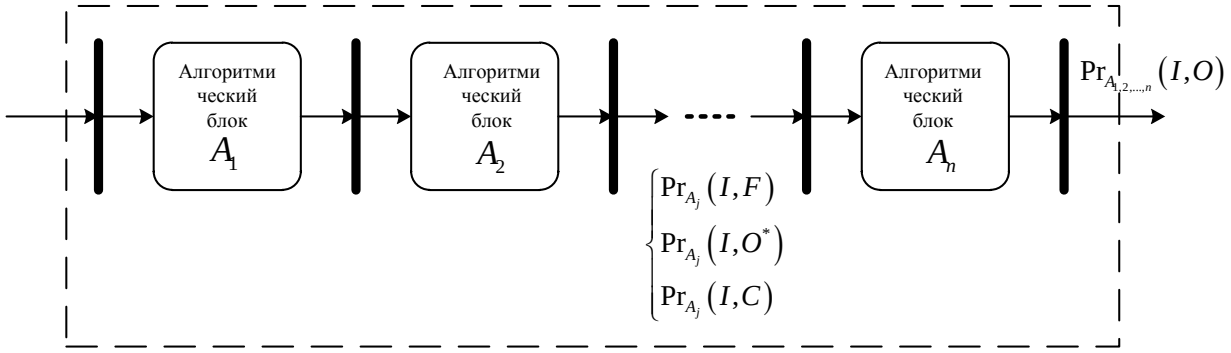

Рис. 3.2. Последовательная структура программного компонента

Вычислительная схема 3.1:

1. Генерация вероятностей $\Pr_{A_j}(I,F)$, $\Pr_{A_j}(I,O^*)$, $\Pr_{A_j}(I,C)$, $I \in F_B$, $F \in F_A$, $O^* \in F_B$ $C \in F_C$, $j = 1,...,n$.

2. Для имитации возникновения некритической ошибки на выходе A_n:

$\Pr_{A_{1,2,...,n}}(I,O) = \Pr_{A_j}(I,F)$, где $F \in F_A$.

3. Для имитации возникновения переходящей ошибки на выходе A_n:

$\Pr_{A_{1,2,\ldots,n}}(I,O) = \sum_{O^*} \Pr_{A_{1,2,\ldots,n-1}}(I,O^*)\Pr_{A_n}(O^*,O)$, где $O^* \in F_B$.

4. Для имитации возникновения критической ошибки на выходе A_n:

$\Pr_{A_{1,2,\ldots,n}}(I,O) = \Pr_{A_{1,2,\ldots,n-1}}(I,C) + \sum_{O^* \in F_B} \Pr_{A_{1,2,\ldots,n-1}}(I,O^*)\Pr_{A_n}(O^*,O)$,

где $O^* \in F_B\ C \in F_C$

6.2. *Разветвляющаяся структура*

Если программный компонент имеет структуру разветвления, имеющую c_n ветвей, из которых первые $n-1$ являются ветвями «если-то», а n -я ветвь является общим «иначе», то расчет будет производиться по нижеследующей вычислительной схеме. Иллюстрация показана на рис. 3.3.

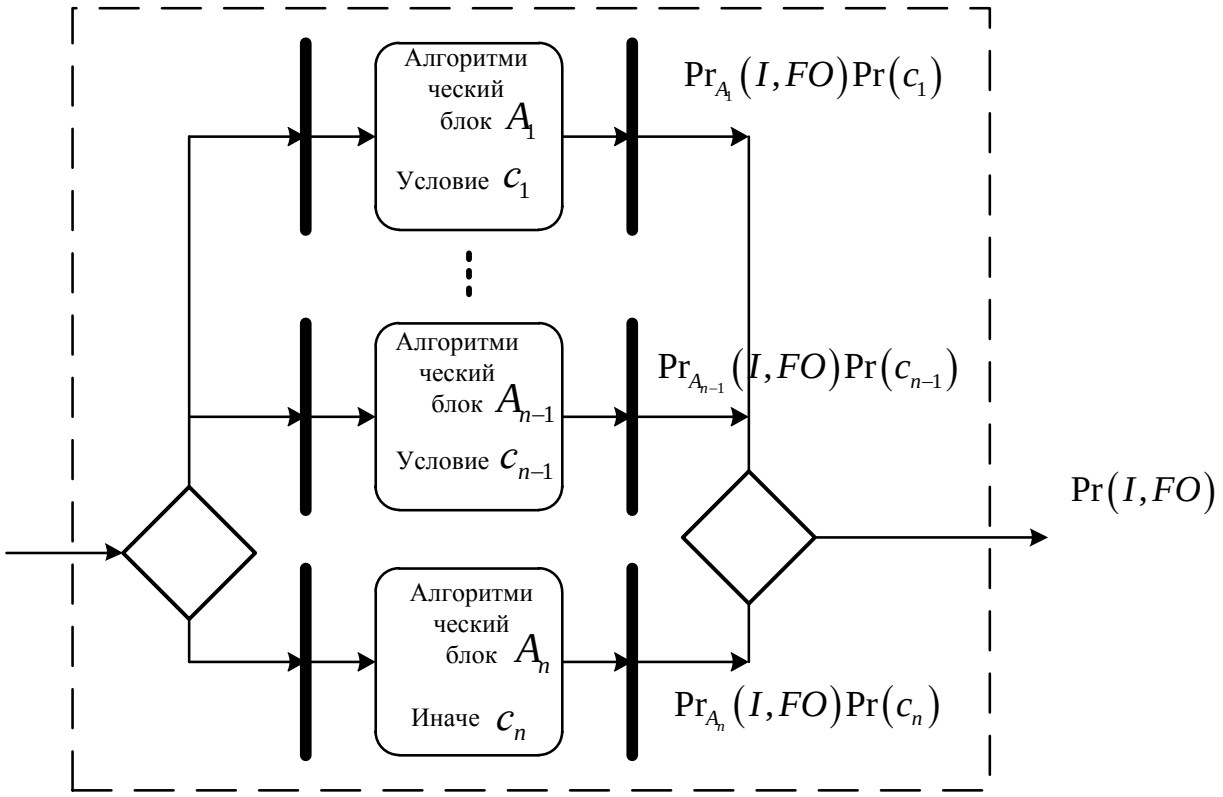

Рис.3.3 Разветвляющаяся структура программного компонента

Вычислительная схема 3.2:

1. Генерация вероятностей $\Pr_{A_j}(I,FO)$, $I \in F_B, FO \in (F_B \cup F_C)$, $j = 1,\ldots,n$, $\Pr(c_i)$, $i = 1,\ldots,n-1$.

2. Для имитации возникновения ошибки рассчитывается

$$\Pr(I,O)=\sum_{i=1}^{n-1}\Pr_{A_i}(I,FO)\Pr(c_i)+\Pr_{A_n}(I,FO)\left(1-\sum_{i=1}^{n-1}\Pr(c_i)\right)$$

6.3. *Циклическая структура*

Если программный компонент имеет циклическую структуру, то её можно трансформировать в n -кратное повторение последовательной структуры исполнения алгоритма $\underbrace{A_1,A_1,...,A_1}_{n\ раз}$ программного компонента. Таким образом, для циклической структуры оказывается подходящей вычислительная схема из п. 6.1.

6.4. *Параллельная структура*

Если программный компонент предусматривает параллельное исполнение n алгоритмов $A_1,A_2,...,A_n$, то моделируется возникновение событий, связанных с классами некритичных и критичных ошибок. Структура исполнения алгоритмов в этом случае представлена на рис. 3.4.

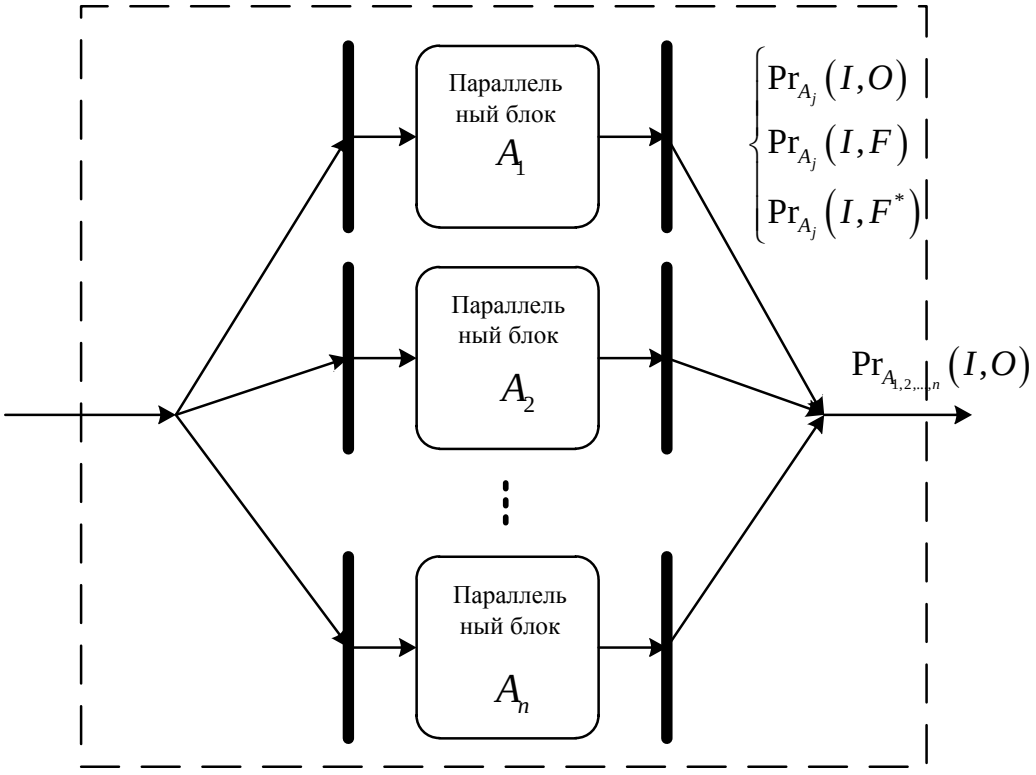

Рис. 3.4. Параллельная структура программного компонента

Вычислительная схема 3.3:

1. Генерация вероятностей $\Pr_{A_j}(I,O)$, $\Pr_{A_j}(I,F)$, $\Pr_{A_j}(I,F^*)$, $I,\in F_B$, $F^* \in F_A$, $F \in (F_A \cup F_C)$ $j=1,...,n$.

2. Если $n-1$ параллельных ветвей имеют корректный выход, а n-я ветвь имеет некритическую ошибку, то рассчитывается

$$\Pr_{A_{1,2,...,n}}(I,O) = \Pr_{A_{1,2,...,n-1}}(I,O)\Pr_{A_n}(I,F^*).$$

3. Если $n-1$ параллельных ветвей имеют некритическую ошибку, а n-я ветвь имеет критическую ошибку, либо, наоборот, то рассчитывается

$$\Pr_{A_{1,2,...,n}}(I,O) = \left[\sum\nolimits_{F\in(F_A\cup F_C)} \Pr_{A_{1,2,...,n-1}}(I,F)\right]\Pr_{A_n}(I,F).$$

3.4 Алгоритмы дискретно-событийного моделирования процессов возникновения ошибок в сетевом программном обеспечении

Основные результаты данного параграфа опубликованы в работе автора [9]. Большинство систем, применяемых в качестве средств автоматизации технологических процессов на производстве, в промышленности, на транспорте и в других отраслях являются программно-управляемыми. Качество функционирования таких систем зависит не только от качества аппаратных средств, применяемых в их составе, но и от качества ПО, важнейшей характеристикой которого является надежность. Вопросы анализа и моделирования надежности ПО рассмотрены в более пятидесяти моделях, которые обсуждаются в работе [73]. Также ранее рассматривалась общая постановка задачи оценки качества ПО в условиях возникновения пиковых информационных нагрузок [6].

Несмотря на такое внушительное количество имеющихся в наличии математических моделей, предназначенных для оценки надежности ПО, следует отметить наличие близкого сходства между многими из них. Это сходство заключается в применении наиболее распространенных в математической теории надежности систем вероятностных распределений экспоненциального вида для моделирования средних значений интенсивностей ошибок в ПО и

пуассоновских процессов для моделирования процедур повышения надежности ПО. Гораздо меньшее количество моделей построено на других видах вероятностных распределений или имеет в основе нестационарные модели случайных процессов. В связи с этим фактом основной целью данного параграфа диссертации является рассмотрение ряда случайных процессов и алгоритмов их моделирования, применимых в качестве базовой составляющей методов оценки надежности ПО, которые используют описание процессов обнаружения программных ошибок на этапах его разработки и тестирования.

С общеинженерной точки зрения надежность ПО рассматривается как его способность сохранять свой уровень качества функционирования за определенный период времени. Большинство математических моделей трактуют надежность ПО как вероятность корректного исполнения функций, заложенных в спецификацию ПО при разных входных наборах значений и программном окружении различных пользователей. Специфика исследований в области надежности ПО заключается также в том, что оно не подвергается старению в процессе эксплуатации, а большинство ошибок ПО возникает из-за дефектов проектирования и реализации. Также активизация и проявление ошибок зависит от входных данных, и при некоторых из них ошибки могут оставаться латентными. Если некоторые программные модули в процессе эксплуатации ПО не используются, либо не исполняются, то даже при наличии в них ошибок не возникает возможности их проявления. Напротив, наибольшее количество ошибок обнаруживается в процессе тестирования программ, и если ошибка успешно исправляется, то в ПО наблюдается увеличение, рост надежности. В связи с этим многие вероятностные модели классифицируются как модели роста надежности ПО [3]. Заметим также, что существуют модели, которые строятся на изучении отказов, которые могут последовать из-за возникновения ошибок в ПО, что является актуальным для информационно-управляющих систем, так как программные ошибки могут повлечь некорректное функционирование и отказы аппаратуры.

Ошибки ПО могут классифицироваться по различным принципам, например, по частоте возникновения, степени влияния на функционирование всей системы, трудоемкости исправления, стоимости восстановления системы после ошибки, либо последовавшего за ней отказа системы и другим характеристикам. По своему содержанию ошибки ПО также можно разделить на многочисленные категории, такие как ошибки объявления и инициализации структур данных, ошибки обращения к структурам данных, ошибки вычисления, ошибки преобразования типов данных, ошибки ввода-вывода, ошибки управления ходом вычислений, ошибки в организации интерфейсов с пользователем и другие.

Алгоритмы моделирования на основе одномерных случайных процессов

Вероятностные модели надежности ПО, имеющие своей базой случайные процессы, описываются следующим образом.

Пусть $N(t)$ обозначает число ошибок (событий), проявившихся на интервале времени $(0,t]$, и пусть T_i обозначает время проявления i-й ошибки. Наблюдаемая последовательность времен фиксации появившихся ошибок составляет точечный случайный процесс, а $\{N(t),\ t \geq 0\}$ является соответствующим ему считающим случайным процессом. Считающий случайный процесс является случайным процессом с непрерывным временем $(N(t))_{t\geq 0}$ с пространством состояний $\mathbb{Z}_+$ и является также неотрицательной и неубывающей функцией времени t. В таком виде $N(t)$ показывает число событий, имевших место до наступления времени t. Вследствие этого время наступления событий является случайной величиной из $\mathbb{R}_+$ в моменты, когда $N(t)$ скачкообразно изменяет свои значения. Для ПО данные факты означают, что считающий случайный процесс $N(t)$ имеет смысл некоторой целочисленной случайной величины, которая равна количеству проявившихся (например, при тестировании) ошибок вплоть до момента времени t. Заметим, что хотя возможно

рассматривать начальное состояние считающего процесса отличным от нуля, обычно подразумевается, что траектория процесса выходит из нуля: $N(0)=0$. Также при рассмотрении процессов моделирования ошибок в ПО полагается, что если в некоторый момент времени не может проявиться более, чем одна ошибка, то величина скачка считающего процесса $N(t)$ равна единице. С другой стороны, данное обстоятельство понимается как возможность только конечного числа событий, возникающих на некотором конечном интервале времени. Как было указано ранее, $N(t)$ является неубывающей функцией, а переменная T_i, обозначающая время возникновения (либо проявления) i-й ошибки есть случайная величина $T_i=\inf\{t\geq 0|N(t)=i\}$, $i\geq 1$. Приняв, что $T_0=0$, естественным образом определяются величины времени между интервалами возникновения ошибок $X_i=T_i-T_{i-1}$, $i\geq 1$. Последовательности случайных величин $\{T_n\}_{n\geq 0}$ и $\{X_n\}_{n\geq 0}$ составляют случайные процессы (не являющиеся считающими) с дискретным временем со значениями из $\mathbb{R}_+$. Ясно, что для любого $n\geq 1$ вероятностное распределение T_n определяет вероятностное распределение для X_n и наоборот. Также, если нам известен процесс $\{T_n\}_{n\geq 0}$, можно определить $N(t)$ как $N(t)=\max\{i\in \boldsymbol{N}|T_i\leq t\}$. Более того, для $n\geq 1$, $k\geq 1$ и $0<l_1<...<l_k$ процессы $\{N(t)\}_{t\geq 0}$, $\{T_n\}_{n\geq 0}$, $\{X_n\}_{n\geq 0}$ требуют для моделирования лишь одно из следующих вероятностных распределений:

– совместное распределение $N(l_i),...,N(l_k)$;

– совместное распределение $T_1,...,T_n$;

– совместное распределение $X_1,...,X_n$.

Считающие процессы могут также быть определены посредством функции среднего значения процесса, которую обозначим $\Lambda(t)$, являющегося математическим ожиданием числа событий вплоть до момента времени t:

$$\Lambda(t)=E\left[N(t)\right].$$

Для класса процессов, рассматриваемых в данной работе функция $\Lambda(t)$ имеет смысл кумулятивной интенсивности обнаружения ошибок, а производная этой функции по времени является мгновенной, либо просто интенсивностью $\lambda(t) = \Lambda'(t)$.

ПО с точки зрения надежности относится к восстанавливаемым системам, поэтому наиболее распространенной моделью является случайный пуассоновский процесс. Рассмотрим более детально алгоритм его моделирования. Процесс $N(t)$, $t \geq 0$ является однородным пуассоновским процессом с интенсивностью $\lambda > 0$, если

1) $N(0) = 0$;

2) события $N(s_2) - N(s_1)$ и $N(t_2) - N(t_1)$ на неперекрывающихся временных интервалах $(s_1, s_2]$ и $(t_1, t_2]$ являются независимым друг от друга, то есть выполняется требование независимости приращений процесса;

3) распределение числа событий на некотором интервале зависит только от длины интервала, то есть выполняется требование стационарности приращений;

4) $\lim\limits_{\Delta t \to 0} \dfrac{\Pr(N(\Delta t) = 1)}{\Delta t} = \lambda$;

5) $\lim\limits_{\Delta t \to 0} \dfrac{\Pr(N(\Delta t) \geq 2)}{\Delta t} = 0$.

В пуассоновском процессе число событий на интервале длины t является случайной величиной, имеющей пуассоновское вероятностное распределение. Одним из методов имитационного моделирования пуассоновского процесса является генерация интервалов времени между возникновением ошибок X_i, $i = 1, 2, ..., n$ в соответствии с методом инверсии по формуле $X_i = -\dfrac{1}{\lambda} \ln U_i$, где U_i являются случайными величинами, имеющими равномерное распределение $U(0,1)$. Для n временных интервалов возникновения ошибок

$T_j = \sum_{i=1}^{j} X_i,\ \ j = 1,...,n.$ Число ошибок на интервале $(0,t)$ определяется выражением $\inf\left\{n : \sum_{i=1}^{n} X_i > t\right\} - 1.$ Алгоритм имитационного моделирования пуассоновского процесса следующий.

Алгоритм 3.1. Имитационное моделирование пуассоновского процесса на интервале $(0,t)$.

1. s=0. $i = 0$. Ввод t.
2. Генерация случайной равномерной величины $U \sim U(0,1)$.
3. $X = -\frac{1}{\lambda}\ln U$.
4. $s = s + X$.
5. Если $s > t,$ то конец алгоритма, п. 8.
6. $i = i + 1;\ T_i = s$.
7. Переход к п.2.
8. Конец алгоритма.

Как следует из алгоритма 3.1, формула генерации пуассоновского процесса $T_i = \left(T_{i-1} - \frac{1}{\lambda}\ln U_i\right),\ i = 1,2,...,n$, $U_i \sim U(0,1)$.

Рассмотрим теперь случайный процесс, который является одним из основных в моделях роста надежности ПО – неоднородный пуассоновский процесс [56]. С помощью данного процесса определяются модели надежности ПО как с непрерывным временем, так и с дискретным. В моделях роста надежности ПО с непрерывным временем, например, одной из самых известных, называемой моделью Гоэла–Окумото [48], устанавливается, что в результате тестирования ПО ошибки проявляются либо в экспоненциальной, либо в форме кривой, имеющей S-образный вид, либо имеют некоторый смешанный вид [34]. Для моделей, разработанных на основе неоднородного пуассоновского процесса, характерны следующие особенности:

– программно-управляемая система является подверженной ошибкам в результате исполнения программ и необнаруженным в них ошибкам;

– количество выявляемых ошибок в любой момент времени пропорционально количеству не выявленных ошибок;

– не выявленные ошибки в одинаковой мере влияют на интенсивность обнаруживаемых ошибок;

– любые ошибки являются независимыми друг от друга, а также от их разновидностей;

– после обнаружения ошибка устраняется немедленно, поэтому отношение числа обнаруженных и устраненных ошибок является постоянной величиной.

Некоторые из данных условий могут выполняться не всегда и не в полной мере, также как при устранении обнаруженных ошибок с добавлением исправлений и нового программного кода могут вноситься новые ошибки. Часто подвергается критике и предположение о независимости и равнозначности ошибок ПО, тем не менее модели надежности на основе неоднородных пуассоновских процессов составляют наиболее широкий класс моделей.

Пуассоновский процесс $\{N(t), t \geq 0\}$ является неоднородным с функцией интенсивности $\lambda(t),\ t \geq 0,$ если

1) $N(0) = 0;$

2) $\{N(t), t \geq 0\}$ имеет независимые приращения;

3) $\lim\limits_{\Delta t \to 0} \dfrac{\Pr(N(t+\Delta t) - N(t) = 1)}{\Delta t} = 0.$

Важным следствием перечисленных свойств является то, что число ошибок на интервале $(s,t],\ t > s$ имеет пуассоновское распределение с параметром

$$S_s^t \lambda(u)du,\ \Pr(N(t) - N(s) = k) = \frac{\left(S_s^t \lambda(u)du\right)^k}{k!} \exp\left(-\int_s^t \lambda(u)du\right).$$

Очевидно, что в отличие от пуассоновского процесса, данный неоднородный случайный процесс не требует условия стационарности приращений,

а при $\lambda(t)=\lambda$ неоднородный пуассоновский процесс становится однородным пуассоновским процессом.

Наиболее распространенный метод имитационного моделирования неоднородного пуассоновского процесса состоит в генерации считающего процесса, который на интервалах $(0,t_1]$, $(t_1,t_2]$ и далее $0\le t_i\le T$ подсчитывает количество событий, сгенерированных в соответствии с пуассоновским процессом с интенсивностью λ с вероятностью $\lambda(t)/\lambda$. Процедура в общем виде состоит из следующих этапов: 1) определяется интервал моделирования $0\le t\le T$ и вводятся интенсивности обнаружения ошибок λ; 2) генерируются события в соответствии с однородным пуассоновским процессом с интенсивностью λ; 3) считается количество событий, произошедших к моменту времени t и сохранять его с вероятностью $\lambda(t)/\lambda$.

Алгоритм 3.2. Имитационное моделирование неоднородного пуассоновского процесса

1. $t=0$. $n=0$. Ввод λ, T.
2. Генерация случайной величины, распределённой по равномерному закону $U\sim U(0,1)$.
3. $X=-\dfrac{1}{\lambda}\ln U$;
4. $t=t+X$.
5. Если $t>T$, то конец алгоритма, п. 9.
6. Если $\lambda(t)/\lambda\ge U$, то $n=n+1$; $T_n=t$.
7. Сохранить T_n.
8. Перейти к п. 2.
9. Конец алгоритма.

В результате исполнения алгоритма 3.2 формируется последовательность $T_1,T_2,...,T_n$ времен обнаружения ошибок, подчиняющихся неоднородному пуассоновскому процессу. Для того, чтобы обеспечить лучшее качество

генерации, можно применить несколько модифицированный алгоритм 3.3 с двумя датчиками равномерного распределения случайных величин.

Алгоритм 3.3. Имитационное моделирование неоднородного пуассоновского процесса (2 датчика).

1. $t=0.\ n=0.$ Ввод $\lambda,\ T$.

2. Генерация случайной величины $U_1 \sim U(0,1)$, имеющей равномерное распределение на отрезке $(0,1)$.

3. $t = t - \ln\dfrac{U_1}{\lambda}.$

4. Цикл пока $t < T$

4.1. Генерация случайной величины $U_2 \sim U(0,1)$, равномерное распределение на отрезке $(0,1)$.

4.2. Если $U_2 \le \lambda(t)/\lambda,$ то $n = n+1;\ T_n = t.$

4.3. Генерация $U_1 \sim U(0,1)$.

4.4. $t = t - \ln\dfrac{U_1}{\lambda}.$

5. Конец алгоритма.

Алгоритмы моделирования, основанные на двумерных случайных процессах

Хотя основным инструментом выявления ошибок ПО остается его тестирование, время тестирования является не единственным фактором, который может дать исчерпывающую информацию об обнаруживаемых ошибках. С другой стороны, также очевидно, что ошибки могут обнаруживаться не только одним, а двумя и более коллективами разработчиков, тестировщиками ПО, а также, возможно, его пользователями. Таким образом, одномерный случайный процесс уже не будет адекватно отражать смысл моделирования, и требуется его распространение на большее количество измерений. Еще одним фактором, служащим для основания применения многомерных случайных процессов в качестве моделей оценки надежности ПО является применение в

них разных метрик надежности, например, в работе [33] рассматривается модель, построенная на двух метриках: кумулятивном количестве ошибок ПО и тестовом покрытии. При этом модель надежности ПО рассматривается как двумерный стохастический процесс типа неоднородного пуассоновского процесса вида $\{N(s,u), s \geq 0, u \geq 0\}$, где S – количество обнаруженных ошибок, а U – величина тестового покрытия:

$$\Pr(N(s,u) = n) = \frac{(m(s,u))n}{n!} \exp(-m(s,u)), \quad n = 0,1,2...$$

где $m(s,u)$ – математическое ожидание процесса, $m(s,u) = \int_0^s \int_0^u \lambda(\zeta,\xi) d\zeta d\xi.$

Дадим формальное определение двумерного однородного пуассоновского процесса. Считающий процесс $\{N(t), t \geq 0\}$ является двумерным однородным случайным пуассоновским процессом на $C \in \mathbb{R}^2$ с интенсивностью $\lambda \geq 0$, если

1) число событий в области $R \subseteq C$ имеет пуассоновское распределение с параметром $\Lambda(R) = \int_R \lambda dV = \lambda A(R)$, где $A(R)$ – это площадь области R;

2) число событий на любом множестве неперекрывающихся областей является независимым.

Генерация двумерного однородного пуассоновского процесса с интенсивностью λ в области, имеющей форму круга с радиусом $r > 0$, опирается на теорему из работы [57].

Т е о р е м а 3.1.

Пусть $(R_1,\Theta_1),(R_2,\Theta_2),...,(R_N,\Theta_N)$ полярные координаты $N > 0$ событий пуассоновского процесса на круге $C = \{(x,y): x^2 + y^2 \leq r^2\}$. Тогда для $N = n$ упорядоченные радиусы $R_{(1)}, R_{(2)},..., R_{(n)}$ являются порядковыми статистиками плотности $f(z) = 2z / r^2, \ z \in [0,z]$. Величины $\Theta_1,\Theta_2,...,\Theta_n$ являются

независимыми, одинаково распределенными на $[0,2\pi]$ случайными величинами, независимыми от $R_1, R_2, ..., R_n$.□

В общем виде процедура генерации включает: генерацию n событий по пуассоновскому закону распределения с параметром $\pi r^2\lambda$, далее генерацию радиусов $R_{(1)}, R_{(2)}, ..., R_{(n)}$ как порядковых статистик из плотности $f(z)$, а затем генерацию величин углов $\Theta_1, \Theta_2, ..., \Theta_n$ равномерно на отрезке $[0,2\pi]$.

Алгоритм 3.4. Имитационное моделирование двумерного однородного пуассоновского процесса на круге.

1. Ввод r, λ.
2. Генерация n событий пуассоновского процесса с интенсивностью $\pi r^2\lambda$.
3. Генерация равномерно распределенных величин $U_1, U_2, ...U_n \sim U(0,1)$.
4. Вычислить $R_1 r\sqrt{U_1}, R_2 r\sqrt{U_2}, ..., R_n r\sqrt{U_n}$.
5. Отсортировать $R_1, R_2, ..., R_n$ в возрастающем порядке и получить $R_{(1)}, R_{(2)}, ..., R_{(n)}$.
6. Генерировать n равномерно распределенных величин $U_{n+1}, U_{n+2}, ...U_{2n} \sim U(0,1)$.
7. Вычислить $\Theta_1 = 2\pi U_{n+1}, \Theta_2 = 2\pi U_{n+2}, ..., \Theta_n = 2\pi U_{2n}$.
8. Сформировать пары $(R_{(1)}, Q_1), (R_{(2)}, Q_2), ..., (R_{(n)}, Q_n)$.

В работе [65] предлагается способ имитационного моделирования, подходящий для генерации процесса при условии, что область моделирования имеет менее регулярную форму. Рассмотрим алгоритм моделирования для области, ограниченной первым квадрантом и проходящей в нем положительнозначной функцией $f(x)$, то есть

$$C = \{(x,y): x \ge 0, y \ge 0, y \le f(x), x \le T\}.$$

Алгоритм использует свойство порядковых статистик $X_{(1)}, X_{(2)},..., X_{(n)}$ такое, что $\int_{X_{(i-1)}}^{X_{(i)}} f(x)dx$ имеет экспоненциальное распределение со средним значением $1/\lambda$. Соответствующие координаты X_i имеют равномерное распределение на интервале $\left[0, f\left(X_{(i)}\right)\right]$.

Алгоритм 3.5. Имитационное моделирование двумерного однородного пуассоновского процесса в области, ограниченной функцией $f(x)$ в первом квадранте.

1. Ввод λ, T.
2. $j=1.\ n=0.\ \omega=0.\ X_0=0.$
3. Генерация равномерно распределенных случайных величин $U_j \sim U(0,1)$.
4. $\omega_j = -(1/\lambda)\ln(1-U_j)$.
5. $\omega = \omega + \omega_j$.
6. Если $\omega \le \int_0^t f(x)dx$, тогда $n = n+1$, переход к п.3.
7. Найти $x_{(i)}$, $i=1,2,...,n$, удовлетворяющие условию $\int_{x(i-1)}^{x(i)} f(x)dx = \omega$.
8. Генерация $U_{n+1}, U_{n+2}, ...U_{2n} \sim U(0,1)$ равномерно распределенных случайных величин.
9. $y_i = U_{n+i} f(x_i)$.
10. Формировать пары $\left(x_{(i)}, y_i\right)$, $i=1,2,...,n$.
11. Конец алгоритма.

Перейдем к более детальному рассмотрению определения и имитационных алгоритмов моделирования двумерных неоднородных пуассоновских процессов. Считающий процесс $\{N(t), t \ge 0\}$ является двумерным неоднородным пуассоновским процессом на $C \in \mathbb{R}^2$ с функцией интенсивности $\lambda(x,y) > 0$, если

1) число событий в области $R \subseteq C$ является распределенным по пуассоновскому закону с параметром $\Lambda(R) = \iint_R \lambda(x, y)\,dxdy$;

2) число событий на любом множестве неперекрывающихся областей является независимым.

Процедура генерации процесса использует идею формирования координат (x, y) событий в области C в соответствии с функцией плотности вероятности следующего вида:

$$f(x, y) = \frac{\lambda(x, y)}{\iint_C \lambda(x, y)\,dxdy},\ (x, y) \in C.$$

Тогда, учитывая первое из вышеперечисленных свойств двумерных неоднородных пуассоновских процессов, основные этапы процедуры генерации состоят в следующем:

1) генерация случайной величины n, распределенной по пуассоновскому закону со средним значением $\iint_C \lambda(x, y)\,dxdy$;

2) генерация n случайных величин (x_i, y_i), $i = 1, 2, ..., n$, которые распределены в области C с плотностью $f(x, y) = \frac{\lambda(x, y)}{\iint_C \lambda(x, y)\,dxdy}$.

Заметим, что данная процедура существенно зависит от трудоемкости нахождения интеграла по области C. Более эффективный и конкретизированный метод имитационного моделирования двумерного неоднородного пуассоновского процесса можно предложить, опираясь на следующую теорему [57].

Т е о р е м а 3.2.

Пусть (x_i, y_i), $i = 1, 2, ..., n$ являются декартовыми координатами двумерного пуассоновского процесса на с интенсивностью $\lambda^*(x, y) \geq 0$. Если i-е событие независимо удаляется с вероятностью $1 - \lambda(x, y) / \lambda^*(x, y)$, где

$0 \le \lambda(x,y) \le \lambda^*(x,y)\ \ \forall (x,y) \in C$ является пуассоновским процессом на C с интенсивностью $\lambda(x,y)$. Требуются также два следующих свойства рассматриваемого процесса:

1) для координат (X_i, Y_i) двумерного случайного процесса абсцисса X_i соответствует одномерному неоднородному пуассоновскому процессу с интенсивностью $\lambda_X(x) = \int_{C(x)} \lambda(x,y)dy$, где $C(x) = \{y : (x,y) \in C\}$;

2) для координат (X_i, Y_i) двумерного случайного процесса ордината Y_i имеет вероятностное распределение с плотностью $\lambda(x,y) / \lambda_X(x)$.□

Алгоритм 3.6. Имитационное моделирование двумерного неоднородного случайного пуассоновского процесса.

1. $i = 0$. Ввод n.

2. Генерировать x_i по алгоритму 3.2 (алгоритму 3.3) для одномерного неоднородного пуассоновского процесса с интенсивностью $\lambda_X(x)$.

3. Генерировать y_i в соответствии в соответствии с плотностью $\lambda(x_i, y) / \lambda_X(x_i)$.

4. Формировать последовательность $\{(x_i, y_i)\}$.

5. Если $i \le n$, то $i = i + 1$. Перейти к п. 2.

6. Конец алгоритма.

Эффективность данного алгоритма зависит от вида функций интенсивности $\lambda(x,y)$ и $\lambda_X(x)$.

В заключении следует отметить, что представленные алгоритмы имеют линейную сложность и достаточно просто реализуются в любой современной среде программирования.

3.5. Выводы

1. Основные результаты главы относятся к разработке методов дискретно-событийного моделирования случайных неоднородных точечных процессов, являющихся базой для моделирования роста надежности ПО сетевых систем, вследствие процессов обнаружения и исправления ошибок.

2. Выполнена классификация моделей надежности ПО, а также проанализированы наиболее известные динамические и статические модели оценки надежности.

3. Предложен дискретно-событийный подход к оценке надежности программного обеспечения использующий покомпонентную технологию моделирования событий возникновения ошибок на входе и выходе программного компонента. Процессы обнаружения и устранения ошибок моделируются случайными точечными процессами, а времена обнаружения ошибок сопоставляются с событиями, при возникновении которых требуется рассчитать вероятностные оценки надежности программных компонентов, которые в зависимости от реализуемых ими алгоритмических структур имеют разные формулы для расчета.

4. Предложен ряд численных алгоритмов дискретно-событийного моделирования процессов оценки надежности ПО, базирующихся на имитационном моделировании процессов возникновения ошибок в сетевом ПО. Эти алгоритмы предназначены для моделирования случайных точечных процессов и соответствующих им считающих случайных процессов с непрерывным временем: на основе одномерных неоднородных случайных процессов с одним датчиком; на основе одномерных неоднородных случайных процессов с двумя датчиками; двумерных неоднородных случайных процессов на круге и в области, ограниченной некоторой функцией.

4 ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ И НАДЕЖНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ

4.1 Метод и программное обеспечение имитационного моделирования компьютерных систем на основе (*min*,+) фильтраций

Под компьютерными системами будем понимать совокупность распределенных программно-управляемых систем обменивающихся информационным трафиком посредством стека *IP*-протокола. Такие сетевые системы предоставляют основной спектр нынешних Интернет-услуг и предъявляют различные требования к их качеству, которое принято называть *QoS* (*Quality of Service, качество обслуживания*). Современные механизмы *QoS* опираются на принципы либо интегрированной архитектуры обслуживания (*Integrated Services, IntServ*), которая описана в руководстве *IETF RFC1633*, либо на принципы архитектуры дифференцированного предоставления обслуживания (*Differentiated Services, DiffServ*), которая описана в руководстве *IETF RFC2474*. Архитектура *IntServ* обеспечивает сетевое обслуживание в двух режимах: 1) контролируемой загрузки (*Controlled Load, CL*) для обеспечения гарантированного трафика к сетевым адаптивным приложениям «мягкого» реального времени; 2) гарантированного обслуживания (*Guaranteed Service, GS*) с обеспечением заранее заданной полосы пропускания к сетевым неадаптивным приложениям «жесткого» реального времени.

В случае *CL* для настройки услуг не используются управляющие параметры *QoS*, такие как уровень потерь пакетов и время задержки поступления данных. Доставка трафика сетевому приложению зависит лишь от времени задержки ретрансляции данных сетевым оборудованием, а также от всплескового характера самого трафика. Время минимальной задержки в режиме *CL*, будет стремиться к максимальной величине всплескового (в количественном отношении) объема трафика. В случае обеспечения уровня услуг *GS* сетевому

приложению выделяется гарантированная полоса пропускания трафика и регулируется граничное значение времени задержки обслуживания. Величина максимального времени задержки зависит от параметров канала передачи данных, от дисциплины, интенсивности обслуживания и определена в руководстве *IETF RFC2212* и алгоритмом маркерной корзины (*Token Bucket*):

$$D = \begin{cases} \dfrac{(b-M)(p-R)}{R(p-r)} + \dfrac{(M+C_{tot})}{R} + D_{tot}, & p > R \geq r \\ \dfrac{(M+C_{tot})}{R} + D_{tot}, & r \leq p \leq R \end{cases} \qquad (4.1)$$

где r – интенсивность поступления маркеров в корзину, байтов/с;

p – максимальная интенсивность поступления данных, байтов/с;

R – интенсивность обслуживания, байтов/с;

b – размер маркерной корзины, байтов;

M – максимальный размер поступающего блока данных (*IP*-дейтаграммы), байтов;

C – максимальная девиация размера поступающего блока данных, байтов;

D –максимальная девиация времени при обработке данных устройствами обслуживания, с.

В дополнение к рассмотренной, архитектура *DiffServ* осуществляет разделение трафика на классы и приоритеты. В целом, при управлении качеством обслуживания в *IP*-сетях основное внимание уделяется обеспечению минимальных (или максимальных) гарантированных значений параметров управления трафиком: полосой пропускания, временной задержкой, девиацией временной задержки и уровнем потерь блоков данных. Поэтому одними из современных адекватных математических моделей сетевых систем с гарантированным обслуживанием являются модели, построенные на (*min*,+) или (*max*,+) алгебраических структурах [55]. Данное направление базируется на подходах теории алгебраических структур – полуколец с каноническим упорядочиванием (идемпотентных полуколец, диоидов), которые рассматривались в

предыдущих главах диссертации [49] идемпотентного анализа [18], моделирования дискретных динамических систем [35]. Моделированию и оценке гарантированной производительности и надежности сетевых систем посвящены работы с участием автора [3, 10, 28, 29].

Основной алгебраической структурой в рассматриваемом подходе является диоид, ранее подробно рассмотренная в параграфе 1.3. Еще раз отметим, что диоид позволяет предложить достаточно эффективный подход к аналитическому и имитационному моделированию сетевых систем, наряду с ТМО. Существенные отличия подходов, в целом, состоят в том, что в ТМО для расчета числовых характеристик необходимо построить аналитические или адекватные им приближенные модели входных потоков и элементов обслуживания, а в алгебраическом подходе сетеметрии исследуются граничные условия для кумулятивных функций, описывающих как входной поток, так и поток обслуживания. Таким образом в последнем подходе, элемент обслуживания служит фильтром входного потока, придавая выходному потоку ту, или иную заданную форму. Отметим, что функции, описывающие граничные условия могут быть как детерминированными, так и случайными [55].

Далее рассматриваем детерминированные функции фильтрации входного трафика. На рис. 4.1 показана иллюстрация к такому подходу, где x – входной фильтрованный поток (входная кумулятивная функция), y – поток обслуживания (выходная кумулятивная функция), а часть обозначений соответствует формуле (4.1). Буфер сетевых данных принимает входной поток p, который на физическом уровне *IP*-сети является последовательным байтовым потоком, составляющим накапливаемый (следовательно, кумулятивный) объем данных, и, в целом – входной трафик. Далее поток подвергается α - фильтрации, то есть в техническом отношении при информационной перегрузке, переполнении входных буферов, различных неисправностях программно-аппаратного обеспечения, при выполнении функций *QoS*, разделении и трафика на классы и приоритеты происходит отсекание части трафика, подобно прохождению сигналов через фильтры. Так формируется входной

фильтрованный поток x, поступающий на устройство обслуживания, которое в свою очередь также имеет β - фильтр с характеристиками, зависящими от применяемого в сетевом устройстве алгоритма маркерной корзины (*Token Bucket*). В итоге формируется выходной поток y.

Математической моделью рассматриваемого (*min*,+) подхода являются уравнения:

$$\text{для } s,t \in \mathbb{Z}^+, s \le t, \alpha : \mathbb{R}^+ \cup \{+\infty\}$$

$$x(t) - x(s) \le \min_{0 \le s \le t} \{\alpha(t-s)\}; \tag{4.2}$$

$$\text{для } \forall t \in \mathbb{Z}^+, \exists s \in \mathbb{Z}^+, s \le t$$

$$y(t) \ge \min_{0 \le s \le t} \{x(s) + \beta(t-s)\}. \tag{4.3}$$

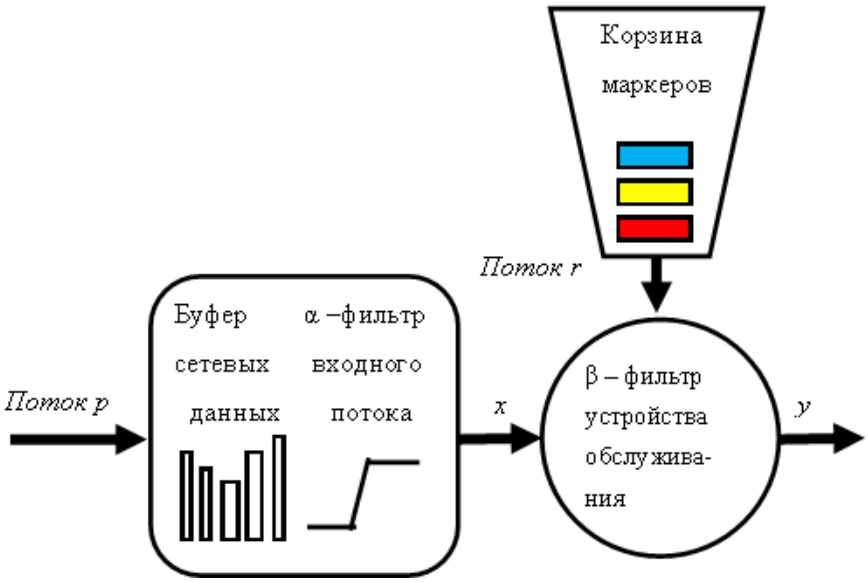

Рис. 4.1 – Имитационная модель сетевой системы с (*min*,+) фильтрацией

Поиск решения уравнений (4.2) и (4.3) приводит к известной из теории цифровой обработки информации задаче о вычислении интеграла свертки:

$$(x \circ y)(t) = \int_{0 \le s \le t} x(s) y(t-s) ds, \tag{4.4}$$

а в терминах рассматриваемого подхода

$$(x \circ y)(t) = \min_{0 \le s \le t} \{x(s) + y(t-s)\}. \tag{4.5}$$

Метод расчета (min,+) интегральной свертки и его реализация на функциональном языке программирования

В работе автора [29] предложен подход к реализации алгоритмов сетеметрии на функциональном языке программирования *J* и были даны необходимы описания входных кумулятивных функций. Рассмотрим дальнейшее описание имитационных алгоритмов расчета (*min*,+) интегральной свертки на функциональном языке программирования *J* [71].

В вышеуказанном языке программирования есть оператор **min**, обладающая всеми необходимыми для названных ранее операций «сложение, $\oplus$» и «умножение, $\otimes$» свойствами. Оператор **min** может быть применим, как к скалярным, так и к матричным величинам. Для скалярных величин можно в среде программирования *J* ввести с клавиатуры, например: **4 min 5, ((35 min 24) min 4).** Можно проверить свойство дистрибутивности операции: **((3 min 2) + 4) = (3 + 4) min (2 + 4)**, результатом будет вывод на экран числа 1, что означает истинность выражения (если результат равен 0, то выражение не является истинным). Матричные операции в (*min,*+) алгебре

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \oplus \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} \min(p_1, q_2) \\ \min(p_2, q_2) \end{pmatrix} \text{ и } \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \oplus \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} = \begin{pmatrix} \min(p_1 + m_{11}, p_2 + m_{12}) \\ \min(p_2 + m_{21}, p_2 + m_{22}) \end{pmatrix}$$

в языке программируются соответственно: **P min"0 Q** и **P (min"1 .+) M.**

Для расчетов (поточечного) минимума двух функций $(f+g)(t)$ и их свертки $(f \circ g)(t)$ в языке *J* нужно запрограммировать соответственно: **min@(f@t, g@t)** и **min@(f@s, g@(t-s)).**

Теперь опишем главный результат параграфа – программную реализацию основного уравнения баланса сетевой модели. Уравнение Линдли [41] (аналогично ТМО):

$$y(t+1) = \left(y(t) + x(t+1) - x(t) - c\right)^{+}$$

На функциональном языке *J* реализовано следующей программой:

Программа 4.1. Уравнение баланса сетевой модели.

```
cocurrent < 'LINDLEY'
ylist =: rhs0
c =: >@lhs1
xlist =: >@lhs2
ind =: <:@#@ylist
xnext =: ind { xlist
yprev =: {:@ylist
ynext =: max0 @ xnext + yprev + c
cocurrent <'base'
lindley_z_ =: ylist_LINLEY_,ynext_LINDLEY_
```

Анализ в области имитационного моделирования сетевых систем показал, что методы, построенные на основе алгебраического подхода с использованием кумулятивных функций входных и выходных потоков представляются очень перспективными и вполне соответствуют технологическим особенностям оборудования, применяемого в современных *IP*-сетях. Выбор в качестве программной реализации весьма нераспространенного функционального языка программирования *J* и реализованные на нем алгоритмы имитационного моделирования также показали эффективность исполнения алгоритмов за счет внутренней рекурсивной организации самого языка программирования. В связи с этим, можно сделать вывод об адекватности применения математического и программного инструментария для имитационного моделирования сетевых систем для оценки их производительности.

4.2 Имитационное моделирование систем с гарантированным обслуживанием его программная реализация

Как уже было указано, одним из наиболее адекватных методов моделирования сетевых устройств является в настоящее время ТМО. Она к настоящему времени довольно развита в направлении применимости к задачам исследования производительности и причин неэффективности компьютерных сетей. Од-

нако, возрастание трафика компьютерных сетей, возникновение информационных перегрузок, сбои в работе оборудования существенно ограничивают применимость математических методов теории систем и сетей массового обслуживания в «чистом» виде. Причиной этому является условие «марковости» процессов в моделируемой системе, но по указанным выше факторам наиболее адекватными инструментами моделирования сетевых систем будут являться именно немарковские модели и процессы. Вследствие этого методы моделирования в рассматриваемой области развиваются в направлении разработки случайных процессов специального вида: автомодельных (фрактальных), скачкообразных процессов Леви [42], процессов, построенных на основе вероятностных распределений с медленно убывающими и тяжелыми хвостами и других.

Еще одной, не менее важной задачей, является разработка математических моделей для сетевых систем (т.е. активного сетевого оборудования, маршрутизаторов, коммуникаторов и др.) при условии выполнения данным оборудованием принципов гарантированного качества обслуживания *QoS*. Современные, применяемые в компьютерных сетях программно-аппаратные комплексы, например, фирмы *Cisco* имеют возможности поддержки качества обслуживания. *QoS Cisco* обеспечивают приоритетную работу важных приложений, одновременно разделяя ресурсы с менее важными приложениями. Функции качества обслуживания *Cisco* включают: классификацию и маркировку сетевых дейтаграмм с помощью установки значений поля *IP*-приоритета или поля кода дифференцированного обслуживания (*Differentiated Services Code Point, DSCP*); управление интенсивностью трафика с помощью функций дозирования трафика и схемы «корзины маркеров» (*token bucket*); предотвращения перегрузки после достижения максимального значения длины очереди по принципу «отбрасывания хвоста» (*tail drop*), а также интеллектуального сбора пакетов на основании маркировки в момент возникновения нагрузки (*WRED, Weighted Random Earlier Detection*); управление очередностью обслуживания по композитному алгоритму взвешенной справедливой очередности

(*Class-Based Weighted Fair Queue, CBWFQ*) и другие, во многом интеллектуальные методы поддержки гарантированного обслуживания.

Для составления математических моделей сетевых устройств с гарантированным обслуживанием выберем методы сетеметрии. Также, как и ТМО методы сетеметрии предназначены для оценки производительности систем, однако, отличие состоит в том, что во втором подходе процесс поступления трафика моделируется некоторой убывающей (в широком смысле, и может вероятностной) функцией, ограниченной сверху и имеющей смысл кумулятивного трафика, а процесс обслуживания моделируется также функцией (неубывающей и возможно вероятностной), но ограниченной снизу, которая несет смысл кумулятивного процесса обслуживания.

Рассмотрим аспекты реализации моделей систем с гарантированным обслуживанием при сетеметрическом подходе на языке функционального программирования *J*, как и в предыдущем параграфе. Выбор данного языка обусловлен его широкими возможностями по обработке матричной и векторной информации, а также богатым набором математических функций.

Элементы детерминированной модели трафика состоят в следующем. Пусть последовательность *a* индексирована интервалами времени *t*=1,2,…, запись *a*(t) обозначает количество поступивших на интервал времени сетевых дейтаграмм (можно использовать количество трафика в байтах). Поток входных заявок будет представлять собой кумулятивную последовательность $A=\{A(t), t=0,1...\}$, где $A(t)$ – число поступлений заявок на интервал [0,*t*], $A(0)=0:\ A(t)=\sum_{s=1}^{t} a(s)$

Последовательность $A(t)$ ограничена сверху функцией *f* следующим образом: $A(t)-A(s)\le f(t-s),\ \ 0\le s\le t$.

На языке *J* приведем примеры программ. Определим входную последовательность поступления дейтаграмм как вектор *a*1

```
a1=: 3 5 2 4 5 2 3 1 3 2.
```

Выражение на языке *J* для кумулятивной последовательности A_1, соответствующей последовательности *a*1:

Программа 4.2

```
] A1 =: 0, + / \a1
```

Результат:

```
0 3 8 10 14 19 21 24 25 28 30.
```

Определим $A_1(t)$ как функцию на языке *J*, для чего используем глагол языка **seq**:

Программа 4.3

```
seq_z =: {~
```

Результат

```
A1=: 0 3 8 10 14 19 21 24 25 28 30 & seq
```

Теперь возможно определение значений функции, например $A_1(3)$

```
 A1 3
10.
```

Входной поток в случае канала связи с фиксированной скоростью передачи данных *CBR* (*constant bit rate*), ограничиваемый функцией $f(t) = ct$ определяется

$$A_1(t) - A_1(s) \le c(t-s), \forall 0 \le s \le t.$$

Время на языке *J*, то есть *t* и *s* определим как глаголы **t_z_** и **s_z_** :

```
t_z_=:]
s_z_=:i.@ >:@t
```

Продемонстрируем результат вычисления при c=3:

Программа 4.4

```
  * / @ ((A1@t - A1@s) <: c * (t - s)) & > i.11
```

Результат

```
1 1 0 0 0 0 0 0 0 0 1
```

По результату видно, что канал *CBR* не допускает «взрывной» характер трафика с определенной выше ограничивающей функцией $F(t) = ct = 3t.$

Для функции гарантированного обслуживания в случае упомянутой ранее схемы *DSCP* имеем:

$B_1(t) - B_1(t) \geq f_s(t - t_0), 0 \leq t_0 \leq t,$ где t_0 – начало обслуживания, $B(t)$ – отставание на момент времени t . На t_0 $B_1(t_0) - A_1(t_0) = 0,$ тогда

$$B_1(t_0) - A_1(t_0) \geq f_s(t - t_0).$$

В терминах сетеметрии

$$B_1(t) \geq \min_{t_0 \leq s \leq t}(A_1(s) + f_s(t - s)).$$

На языке *J* для интенсивности обслуживания $f_s = \lambda(t) = 3$ для канала *CBR*

Программа 4.5

```
fs =: 3 & pr
]B1 =: min(A1@s + fs@(t - s))      & > i.11
```

Результат

```
0 3 6 9 12 15 18 21 24 27 30.
```

Метрики производительности можно рассчитать по следующим формулам и их реализациям на языке *J*:

1) максимальная задержка в обслуживании:

$$d = \max_{s \geq 0}(f(s) - f_s(s))$$

Программа 4.6

```
d =: max@(f@s - fs@s)
```

2) минимальная задержка в обслуживании

$$\tau_z = \min(\tau \geq 0; f(s) \leq f_s(s + \tau))$$

Программа 4.7

```
tau_z_ =: lhso(i.7)(tau;(f@s,: fs@(s + tau)))"0
```

4.3. Имитационное моделирование в задачах прогнозирования надежности программного обеспечения

В настоящее время средства разработки ПО позволяют выполнять автоматическую генерацию исходного текста, использовать шаблоны проектирования программ, повторно использовать компоненты, объекты, выполнять рефакторинг, обладают другими возможностями, ускоряющими цикл проектирования ПО. В тоже время все эти широкие возможности увеличивают не только количество программных подсистем в разрабатываемом проекте, но и усложняют взаимосвязи между подсистемами, неизбежно усложняя процесс поиска ошибок в проекте и снижая его надежность. В связи с этим, основным инструментарием обеспечения надежности программного проекта на данное время являются методы тестирования на разных стадиях жизненного цикла программ, особенно на завершающей стадии с целью обнаружения ошибок при некорректно предоставляемых данных и построение упрощенных формальных спецификаций сложного «большого» проекта, отражающих его основные свойства и требования на том или ином формально-логическом языке с целью логического обоснования корректности исходного программного проекта. Однако существует, хотя и не так интенсивно развивается математическая теория надежности ПО, строящаяся наподобие математической теории систем. Значительным плюсом такого подхода к надежности ПО является возможность не только констатации фактов вида «в ПО есть ошибки, либо с такой-то вероятностью нет ошибок», а вопрос ставится более глубоко – как на основании имеющихся и измеряемых в процессе работы ПО показателей, не имея доступа к исходному тексту программ, математически обоснованно рассчитать вероятность появления, или количество ошибок, через какое-либо будущее время работы программы? Одному из таких способов посвящен далее предлагаемый метод имитационного моделирования.

Предлагаемая модель будет основываться на математической модели из третьей главы, в основе которой лежит неоднородный пуассоновский процесс

$\{N(t), t \geq 0\}$, который характеризуется следующими допущениями. Во-первых, запуски программы происходят в неперекрывающиеся интервалы времени $\left[(0,t_1),(t_1,t_2),...,(t_{i-1},t_i),...,(t_{m-1},t_m)\right]$. Во-вторых, количество ошибок $(f_1, f_2,..., f_m)$ на каждом интервале не зависит от других интервалов, а каждая ошибка устраняется, как только будет выявлена. В третьих, кумулятивное количество ошибок, обнаруженных на момент времени t, имеет пуассоновское распределение со средним $m(t)$. И, наконец, каждая ошибка может быть обнаружена с одинаковой вероятностью, имеет одинаковые негативные последствия, интервалы обнаружения ошибок имеют одинаковую размерность (сутки, неделя, месяц и т.п.).

Функция $m(t)$ является неубывающей и ограниченной сверху, то есть

$m(t) = 0$, при $t = 0$,

$m(t) = a$, при $t \to \infty$, (4.6)

где a – количество ошибок в ПО, которое в принципе в нем содержится и может быть обнаружено.

Если принять, что количество ошибок в ПО, обнаруженных на интервале времени Δt пропорционально количеству ошибок, имеющихся в ПО, но пока не обнаруженных, то с учетом (4.6)

$$m(t+\Delta t) - m(t) = b\{a - m(t)\}\Delta t + o(\Delta t). \quad (4.7)$$

Из выражения (4.7) при $\Delta t \to 0$ получаем дифференциальное уравнение

$$m'(t) = ab - bm(t), \quad (4.8)$$

решая которое с условиями (4.6) получаем, что среднее количество ошибок

$$m(t) = a(1 - \exp(-bt)). \quad (4.9)$$

Интенсивность ошибок из выражений (4.8) и (4.9) получаем как

$$\lambda(t) = m'(t) = abe(-bt). \quad (4.10)$$

Процесс моделирования ошибок с функцией интенсивности $\lambda(t)$ выполняется в соответствии со стохастическим процессом:

– если не возникает ошибок за время Δt, то

$$P\{N(t,\Delta t)-N(t)=0\}=1-\lambda(t)\Delta t+o(\Delta t); \quad (4.11)$$

–если за время Δt возникает одна ошибка

$$P\{N(t,\Delta t)-N(t)=1\}=\lambda(t)\Delta t+o(\Delta t); \quad (4.12)$$

–если за время Δt возникает две и более ошибки

$$P\{N(t,\Delta t)-N(t)=2\}=o(\Delta t). \quad (4.13)$$

Таким образом, возникновение f_i ошибок на интервале времени $(0,t_i)$ будет

$$P(N(t_i)=f_i)=\frac{m(t_i)^{f_i}e^{-m(t_i)}}{f_i!}, \quad (4.14)$$

где i – некоторый интервал времени в процессе которого программа загружена и исполняется;

f_i – число ошибок, которые возникли за период времени i;

t_i – кумулятивное время исполнения программы, включая интервал времени i, то есть совокупность всех времен исполнения программы;

$m(t_i)$ – функция средней интенсивности ошибок для времени, $m(t_i)=E[N(t_i)]=\int_0^t\lambda(s)ds$, $\lambda(s)$ – функция интенсивности.

Для дальнейших численных расчетов среднюю интенсивность ошибок можно приравнять λ_i – среднему количество ошибок, обнаруживаемых на i-ом интервале, и тогда выражение (4.14) будет представлять собой известную модель [52]:

$$P(N(t_i)=f_i)=\frac{\lambda_i{}^{f_i}\exp(-\lambda_i)}{f_i!}.$$

Учитывая, что рассматривается неоднородный пуассоновский процесс, а, следовательно, для каждого t_i-го интервала времени функция интенсивности может изменяться, продолжим рассуждения следующим образом. Параметр λ_i определяется по выражению (4.9) как

$$\lambda_i = \lambda\left[\left(1-\exp\left(-\mu t_i\right)\right)-\left(1-e\left(-\mu t_{i-1}\right)\right)\right] = \lambda\left(\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)\right), \qquad (4.15)$$

где μ – интенсивность устранения ошибок.

Задача моделирования состоит в определении параметров λ и μ – интенсивностях обнаружения и устранения ошибок в ПО в выражении (4.13), причем для каждого t_i-го интервала времени.

Очевидно, что в нашем случае, при наличии выборки данных, содержащей кумулятивное количество обнаруженных ошибок к некоторому t_i-му интервалу времени, для определения параметров λ и μ подходящим оказывается метод максимального правдоподобия [17].

Функция правдоподобия имеет вид

$$L\left(\lambda,\mu\left|\textit{выборка}\right.\right) = \prod_{i=1}^{I}\exp\left(-\lambda_l\right)\frac{\lambda_i{}^{f_i}}{f_i!}, \qquad (4.16)$$

где I – общее число временных интервалов наблюдения ошибок.

Прологарифмируем выражение (4.16):

$$\bar{L}\left(\lambda,\mu\left|\textit{выборка}\right.\right) = \ln L\left(\lambda,\mu\left|\textit{выборка}\right.\right) = -\sum_{i=1}^{I}\lambda_i + \sum_{i=1}^{I} f_i \ln\left(\frac{\lambda_i}{f_i!}\right). \qquad (4.17)$$

Подставив вместо λ_i в выражении (4.17) функцию из выражения (4.15) получаем

$$\begin{aligned}\bar{L}\left(\lambda,\mu\left|\textit{выборка}\right.\right) = &-\sum_{i=1}^{I}\ \lambda\left(\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)\right)+\sum_{i=1}^{I} f_i \ln\lambda + \\ &+\sum_{i=1}^{I} f_i\lambda\left(\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)\right)-\sum_{i=1}^{I} f_i \ln\left(f_i!\right)\end{aligned} \qquad (4.18).$$

Для того чтобы найти максимум для данной функции возьмём частную производную по переменной λ:

$$\frac{d\bar{L}}{d\lambda} = -\sum_{i=1}^{I}\left(\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)\right)+\sum_{i=1}^{I}\frac{f_i}{\lambda}. \qquad (4.19)$$

Просуммировав выражение (4.19) получаем

$$\frac{d\bar{L}}{d\lambda} = \left(1-e^{-\mu t_I}\right)+\left(\frac{f_+}{\lambda}\right),$$

где f_+ – кумулятивное число ошибок для всех i-ых периодов времени.

Приравнивая к нулю производную (4.19), получаем выражение для оценки интенсивности обнаружения ошибок

$$\hat{\lambda} = \frac{f_+}{\left(1-\exp\left(-\mu t_i\right)\right)} \tag{4.20}$$

Подставив из выражения (4.20) $\hat{\lambda}$ в выражение (4.18), взяв производную по переменной μ и приравняв её нулю, получаем

$$\sum_{i=1}^{I}\left[\frac{f_+\left(\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)\right)}{\left(1-\exp\left(-\mu t_i\right)\right)} - f_i\right]\cdot\left[\frac{-t_{i-1}\exp\left(-\mu t_{i-1}\right)+t_i\exp\left(-\mu t_i\right)}{\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)}\right]=0.$$

(4.21)

Рассмотрим вопрос, связанный с нахождением начального значения μ. Пусть

$$x = \exp\left(-\mu t_i\right) \tag{4.22}$$

и пусть

$$s = \frac{1}{f_+ t_i}\sum_{i=1}^{I} f_i\left[\frac{-t_{i-1}\exp\left(-\mu t_{i-1}\right)+t_i\exp\left(-\mu t_i\right)}{\exp\left(-\mu t_{i-1}\right)-\exp\left(-\mu t_i\right)}\right]. \tag{4.23}$$

Из выражений (4.21) – (4.23) видно, что s и x связаны соотношениями $s=[x/(1-x)]$, $x=[s/(1+s)]$. Начальное значение x_0 можно отыскать по формуле

$$x = f_{t_{1+}} / f_{t_{2+}}, \tag{4.24}$$

то есть, разделив количество ошибок, полученных за первую половину времени исполнения программы, на количество ошибок, полученных за вторую половину времени исполнения программы. Далее подставляем полученное значение x в формулу (4.22) и получаем начальное значение μ. Далее для уточнения μ используется итеративный процесс формула (4.25):

$$\mu_{n+1} = \hat{\lambda}\sum_{i=1}^{I}\left[\left(t_i\exp\left(-\mu_n t_i\right)-t_{i-1}\exp\left(-\mu_n t_{i-1}\right)\right)+t_i\right]. \tag{4.25}$$

На основании вышеизложенных соотношений далее предложены алгоритмы прогнозирования показателей надежности ПО.

Алгоритмы нахождения оценок параметров и прогнозирования показателей надежности ПО

Для оценки λ применим метод размножения выборок, известный как метод бутстреппинга [32]. Исходной информацией является выборка наблюдаемых значений (массив данных), в нашем случае содержащий три столбца (номер дня, время исполнения программы для проявления ошибок, число ошибок (пример приведен в табл. 4.1)).

Таблица 4.1 Тестовые данные

День	Время исполнения, ч	Количество ошибок
1	6	4
2	32	5
3	54	5

Таким образом, при методе бутстреппинга, имея некоторую выборку конечной размерности n, то есть эмпирическое распределение и, используя датчики случайных чисел, из неё можно сформировать любое число размноженных выборок. В нашем случае вероятностное распределение в формируемых выборках должно подчиняться пуассоновскому распределению. Рассмотрим алгоритм формирования случайных чисел, распределенных по закону Пуассона.

Алгоритм 4.1. Формирование бутстреп-выборок

Входные параметры:

a. Массив наблюдаемых значений $M = \{m_1, m_2, ..., m_i, ..., m_n\}$ количества ошибок за время исполнения программы;

b. t =0 – текущее время, T – общее время моделирования;

c. I =0 – количество событий за время моделирования T.

d. λ – интенсивность ошибок.

e. j – текущий элемент массива, n – размерность выборки.

1. Пока $j \le n$

2. Генерация случайного числа с равномерным распределением $U \sim U(0,1)$ на отрезке $(0,1)$.

3. $t = t - \frac{1}{\lambda}\ln U$.

4. Если $t > T$, то завершить моделирование и выдать результаты.

5. Генерация случайного числа с равномерным распределением $U \sim U(0,1)$ на отрезке $(0,1)$.

6. Если $U \le \lambda(t)/\lambda, I = I + 1, S_j(I) = t$.

7. Сформировать строки массива $M_1 = \{S_1(I), m_2, ..., m_j, ..., m_n\}$, заменяя поочередно значение в исходной выборке сгенерированным значением, получая тем самым массивы $M_2 = \{m_1, S_2(I), ..., m_j, ..., m_n\}$, …,

$M_j = \{m_1, m_2, ..., S_j(I), ..., m_n\}$, $M_m = \{m_1, m_2, ..., m_j, ..., S_n(I)\}$.

8. Конец алгоритма.

Алгоритм 4.2. Нахождение оценки μ

Входные параметры:

a. Начальное значение x_0 (формула 4.24).

b. TF – общее кумулятивное количество ошибок за все время исполнения программы.

c. Массив времени исполнения программы T – второй столбец из табл. 4.1.

d. Массив сформированных размноженных выборок M из алгоритма 4.1

e. n – размерность массивов T и M.

f. *eps* – точность.

g. *time* – общее время исполнения программ.

1. $Mu := (((-1)\cdot \ln x_0)) / time)$
2. Повторять
2.1. $MuNew := Mu$
2.2. Цикл от 1 до n (расчеты по формуле (4.20))
2.2.1. $t_1 = T[i]e^{-1\cdot Mu\cdot T[i]}$
2.2.2. $t_2 = T[i-1]e^{-1\cdot Mu\cdot T[i-1]}$
2.2.3. $t_3 = e^{-1\cdot Mu\cdot T[i-1]} - e^{-1\cdot Mu\cdot T[i]}$
2.2.4. $t_4 = t_4 + \left(M[i]\cdot\left(\left(t_1 - t_2\right)/t_3\right)\right)$
2.2.5. Конец цикла по n
2.3. Если $t_4 < 0.01$ тогда $t_4 = 0.01$
2.4. $s = \left(t_4 / TF\cdot time\right)$
2.5. $x = \left(s/\left(1+s\right)\right)$
2.6. $t_4 = 0$
2.7. $MuNew = (((-1)\cdot \ln\left(x\right)/time$
2.8. Конец цикла повторять пока $\left|MuNew - Mu\right| \le eps$
3. Результат вывод Mu
4. Конец алгоритма.

Опишем теперь общую процедуру прогнозирования ожидаемого количества ошибок, которые могут проявиться при последующем исполнении программ на интервале времени τ, следующим за кумулятивным интервалом наблюдения t_I. В соответствии с рассматриваемым нами неоднородным пуассоновским процессом среднее ожидаемое количество ошибок

$$m(\tau, t_I, \lambda, \mu) = E\left[N\left(t_I + \tau\right) - N(t_I)\right] = \lambda \exp\left(-\mu t_I\right)\cdot\left(1 - \exp\left(-\mu\tau\right)\right). \quad (4.26)$$

Оценка параметров по предложенным алгоритмам 4.1 и 4.2 позволяет записать выражение (4.26) с использованием оцененных параметров $\hat{\lambda}$ и $\hat{\mu}$:

$$m(\tau, t_I, \lambda, \mu) = m\left(\tau, t_I, \hat{\lambda}, \hat{\mu}\right) = \hat{\lambda}\exp\left(-\hat{\mu}t_I\right)\cdot\left(1-\exp\left(-\hat{\mu}\tau\right)\right).$$

Вычислительная схема 4.1. Прогнозирование ошибок ПО

Шаги процедуры состоят в следующем.

1. Получить оценки $\hat{m}(b) = m\left(\tau, t_I, \hat{\lambda}(b), \hat{\mu}(b)\right)$, где $b = 1,2,...,B$, размноженные методом бутстреппинга выборки, всего B оценок средней интенсивности ошибок.

2. Отсортировать по возрастанию полученные оценки средней интенсивности ошибок $\hat{m}_i(b)$.

3. Определить требуемые доверительные интервалы (обычно нижнее значение $\alpha = 0.05$, а верхнее значение $\alpha = 0.95$) и отсечь не входящие в эти интервалы оценки.

4. Используя оценку среднего значения и известные свойства функции распределения Пуассона можно ожидать, что на интервале времени $t_I, t_I + \tau$ количество ошибок будет не более $\sum_{k=0}^{K}\frac{\exp\left(-\hat{m}_i\right)\cdot\left(\hat{m}_i\right)^k}{k!}$.

Реализация ПО

Предложенный метод прогнозирования был реализован в виде программы, написанной на языке программирования Паскаль в свободной среде программирования PascalABC.NET [64]. Программа состоит их следующих функций:

FirstXValue – определения начального значения x_0 (по выражению 4.24);

FindMu – определения оценки μ (по алгоритму 4.2);

GenerateLambdas – определения оценки λ по методу размножения выборок (алгоритм 4.1);

AssignPoissons – нахождения случайных чисел по распределению Пуассона (элемент алгоритма 4.1).

Исходные данные читаются из файла, имеющего структуру вида таблицы, приведем копию экрана содержимого файла на рис. 4.2.

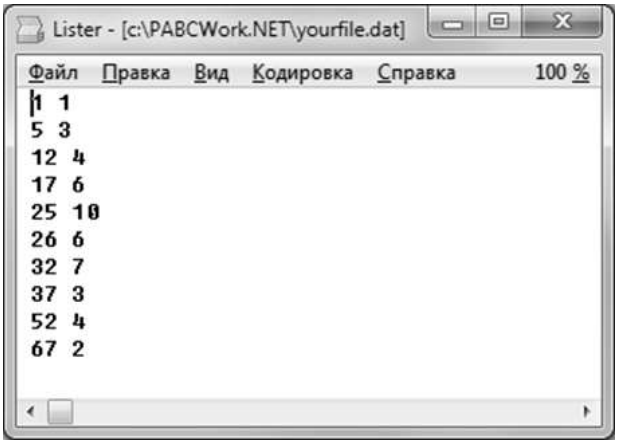

Рис.4.2 – Содержимое тестового файла

Первый столбец означает кумулятивное время исполнения программы (например, в днях), а второй столбец обозначает количество ошибок, зафиксированных в определенные временные интервалы, например, через 5 дней были обнаружены (и исправлены) 3 ошибки в ПО.

При запуске программы с данным тестовым файлом можно увидеть результаты на рис. 4.3, например, прогноза проявления ошибок в ПО с 95% доверительным интервалом на будущие 10 дней.

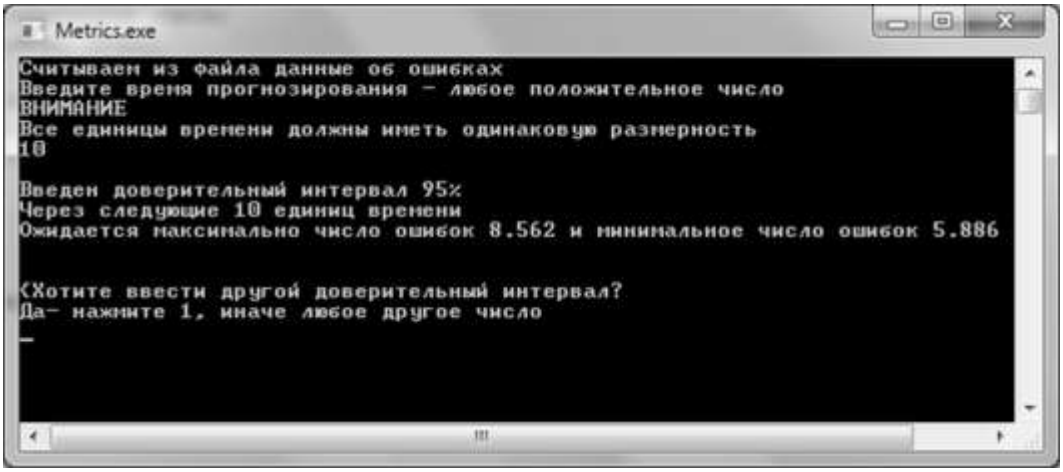

Рис. 4.3 – Результаты прогноза на следующие 10 дней
с тестовым файлом из рис. 4.2

При необходимости можно изменить доверительный интервал, например, установить 75%, получив при этом следующие результаты, на рис. 4.4.

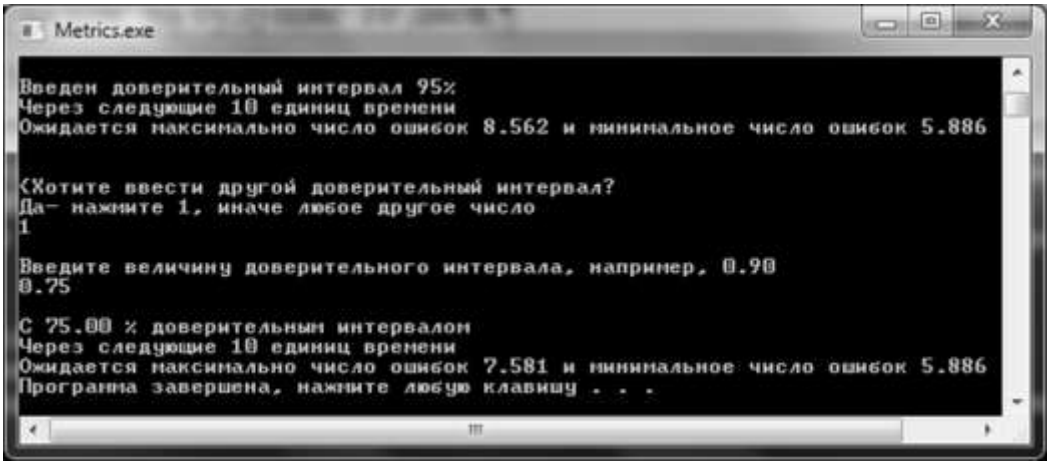

Рис. 4.4 – Результаты прогнозирования при тестовом файле из рис. 4.2 и доверительном интервале 75%

Ниже приведены исходные тексты программ для перечисленных функций.

Программа 4.8. Функция *FirstXValue*

```
function FirstXValue: real;
var
  SearchValue, FirstHalfCount, SecondHalfCount: real;
  i: integer;
begin
  SearchValue := (TotalTime / 2);
  FirstHalfCount := 0;
  SecondHalfCount := 0;
  i := 1;
  repeat
    i := (i + 1);
  until (Times[i] >= SearchValue);
  FirstHalfCount :=  (FaultsEnd[i - 1] + (FaultsEnd[i] -
FaultsEnd[i - 1]) * (SearchValue - Times[i - 1]) / (Times[i] -
Times[i - 1]));
  SecondHalfCount := (FaultsEnd[LengthofDataSet] -
FirstHalfCount);
  FirstXValue := ((SecondHalfCount / FirstHalfCount) *
(SecondHalfCount / FirstHalfCount));
end;
```

Программа 4.9. Функция *FindMu*

```
function FindMU(XOne, TFaults: real; F: DATA): real;
var
  MUCurrent, MUNew, s, x, t1, t2, t3, t4: real;
  i: integer;
begin
  s := 0;
  x := 0;
  i := 0;
  t1 := 0;
```

```
  t2 := 0;
  t3 := 0;
  t4 := 0;
  MUCurrent := (((-1) * LN(XOne)) / TotalTime);
  MUNew := MUCurrent;
  repeat
    MUCurrent := MUNew;
    for i := 1 TO LengthofDataSet DO
    begin
      t1 := Times[i] * EXP((-1) * MUCurrent * Times[i]);
      t2 := Times[i - 1] * EXP((-1) * MUCurrent * Times[i - 1]);
      t3 := EXP((-1) * MUCurrent * Times[i - 1]) - EXP( (-1) *
MUCurrent * Times[i]);
      t4 := t4 + (F[i] * ((t1 - t2) / t3));
    end;
    if (t4 < (0.01)) THEN t4 := 0.01;
    s := (t4 / (TFaults * TotalTime));
    x := (s / (1 + s));
    t4 := 0;
    MUNew := (((-1) * LN(X)) / TotalTime);
  until ((ABS(MUNew - MUCurrent)) <= epsilon);
  FindMU := MUCurrent;
end;
```

Программа 4.10. Функция *GenerateLambdas*

```
procedure GenerateLambdas;
var
  i: integer;
begin
  for i := 2 TO LengthofDataSet DO
  begin
    LAMi[i] := (LAMBDA * exp(-1 * MU * Times[i - 1]) - EXP(-1 *
MU * Times[i]));
  end;
  LAMi[1] := (LAMBDA * (1 - EXP(-I * MU * Times[1])));
end;
```

Программа 4.11. Функция *AssignPoissons*

```
procedure AssignPoissons;
var
  A, B, W, X, U: real;
var
  i: integer;
begin
  for i := 1 to LengthofDataSet DO
  begin
    W := EXP((-1) * LAMi[i]);
    X := 0;
    A := W;
    B := A;
    U := RANDOM(1000) / 1000;
    while (U > A) AND (A <= 0.9999) DO
```

```
    begin
      X := X + 1;
      B := (B * LAMi[i]) / X;
      A := A + B;
    end;
    tempfaults[i] := X;
    TotItFaults := TotItFaults + tempfaults[i];
  end;
end;
```

Программа 4.12. Основная программа

```
program SoftwareReliability (cumfail,output);

uses
  crt;

const
  epsilon = 1.0E-03;
  LengthofDataSet = 10;
  iteration = 10;

type
  DATA = ARRAY [0..LengthofDataSet] of real;
  RUNS = ARRAY [0..iteration] of real;

var
  Faults, Times, FaultsEnd, LAMi, tempfaults: DATA;
  lamboot, muboot, lamprob, muprob, mean: RUNS;

  XFirst, MU, LAMBDA, TotalTime, TotalFaults: real;
  TotItFaults, A, B, key, percentile, temp: real;
  i, j, k, low, high, T, other: integer;
  Cumfail: text;


{********************THE MAIN PROGRAM**********************}
begin
  clrscr;
  XFirst := 0;
  MU := 0;
  LAMBDA := 0;
  TotalTime := 0;
  TotalFaults := 0;
  TotItFaults := 0;
  T := 0;
  i := 0;
  j := 0;
  k := 0;
  for i := 1 TO iteration DO
  begin
    lamboot[i] := 0;
    muboot[i] := 0;
```

```
    mean[i] := 0;
  end;
  for i := 1 TO LengthofDataSet DO
  begin
    Times[i] := 0;
    Faults[i] := 0;
    FaultsEnd[i] := 0;
    LAMi[i] := 0;
    tempfaults[i] := 0;
  end;

  Writeln('Считываем из файла данные об ошибках');
  ASSIGN(Cumfail, 'yourfile.dat');
  RESET(Cumfail);
  while not EOF(Cumfail) DO
  begin
    j := (j + 1);
    READLN(Cumfail, A, B);
    Times[j] := A;
    FaultsEnd[j] := B;
  end;

  for i := 1 TO LengthofDataSet DO
  begin
    Faults[i] := FaultsEnd[i] - FaultsEnd[i - 1];
  end;
  TotalTime := Times[LengthofDataSet];
  TotalFaults := FaultsEnd[LengthofDataSet];

  {***************Function and Procedure calls************** }
  RANDOMIZE;
  XFirst := FirstXValue;
  MU := FindMU(XFirst, TotalFaults, Faults);
  LAMBDA := (TotalFaults / (1 - EXP( (-1) * MU * TotalTime)));
  GenerateLambdas;
  for k := 1 TO iteration DO
  begin
    TotItFaults := 0;
    AssignPoissons;
    muboot[k] := FindMU(XFirst, TotItFaults, tempfaults);
    lamboot[k] := (TotItFaults / (1 - EXP((-1) * muboot[k] * To-
talTime)));
  end;
  low := ROUND(iteration * (0.05));
  high := ROUND(iteration * (0.95));
  Writeln('Введите время прогнозирования - любое положительное
число');
  Writeln('ВНИМАНИЕ');
  Writeln('Все единицы времени должны иметь одинаковую размер-
ность');
  Readln(T);
  Writeln;
  for i := 1 TO iteration DO
```

```
  begin
    mean[i] := lamboot[i] * (EXP((-1) * muboot[i] *
Times[LengthofDataSet]) ) * (1 - EXP((-1) * muboot[i] * T));
  end;

  for j := 1 to iteration - 1 do
    for i := 1 to iteration - j do
      if Mean[i] > Mean[i + 1] then
      begin
        temp := Mean[i];    Mean[i] := Mean[i + 1];     Mean[i +
1] := temp;
      end;

  Writeln('Введен доверительный интервал 95%');
  Writeln('Через следующие ', T, ' единиц времени');
  Writeln('Ожидается максимально число ошибок ', mean[high]:0:3,
' и минимальное число ошибок ', mean[1]:0:3);
  Writeln;
  Writeln;
  Writeln('{Хотите ввести другой доверительный интервал?');
  Writeln('Да- нажмите 1, иначе любое другое число');
  Readln(other);
  Writeln;
  if (other = 1) THEN begin
    Writeln('Введите величину доверительного интервала, напри-
мер, 0.90');
    Readln(percentile);
    Writeln;
    low := ROUND(iteration * (1 - percentile));
    high := ROUND(iteration * percentile);
    Writeln('C ', (percentile * 100):2:2, ' % доверительным ин-
тервалом');
    Writeln('Через следующие ', T, ' единиц времени');
    Writeln('Ожидается максимально число ошибок ',
mean[high]:0:3, ' и минимальное число ошибок ', mean[low]:0:3);
  end;
  CLOSE(Cumfail);
end.
```

4.4. Выводы

1. В главе предложены новые методы имитационного моделирования для технических приложений, связанных с задачами оценки надежности и производительности.

2. Предложен метод имитационного дискретно-событийного моделирования сетевых систем на основе (*min*,+) фильтраций, метод расчета (*min*,+) интегральной свертки и его реализация на функциональном языке программирования.

3. Предложен метод имитационного дискретно-событийного моделирования производительности сетевых систем с гарантированным обслуживанием и его реализация на функциональном языке программирования.

4. Предложен метод имитационного моделирования и алгоритм прогнозирования надежности ПО путем модификации модели, построенной на неоднородном пуассоновском процессе, в котором применен способ размножения выборок, известный как бутстреп-метод. Реализована программа для оценки прогнозирования надежности ПО на императивном языке программирования.

ЗАКЛЮЧЕНИЕ

Основные результаты работы относятся к развитию математических моделей и методов дискретно-событийного моделирования, подходящих для решения задач оценки производительности и надежности компьютерных систем и ПО. Получены результаты в следующих направлениях развития: интервальное уточнение схем дискретно-событийного моделирования, дискретно-событийное моделирование высоконагруженных сетевых систем с индикаторами пиковых значений нагрузок, методы дискретно-событийного моделирования надежности ПО в условиях различий его структуры и классов ошибок.

Получены следующие научные результаты, характеризующиеся научной новизной:

1. Метод моделирования на основе модульного подхода к дискретно-событийному моделированию, схемы систем, полученные на основе интервальных временных событийных графов.

2. Классификация и подход к оценке пиковой информационной нагрузки функционалом специализированного вида. Оценка пиковой информационной нагрузки ведется с помощью индикации ее превышения в информационных потоках системы, описываемых случайными точечными процессами, адекватно подходящими для задач дискретно-событийного моделирования сетевых информационных систем.

3. Модификации моделей оценки пиковой информационной нагрузки для информационных потоков, описываемых многомерными случайными точечными процессами со свойством эргодичности, и для случайных процессов, управляемых скрытыми Марковскими моделями.

4. Модель анализа производительности сетевой системы с установкой граничных значений показателей производительности для решения задач моделирования систем с гарантированным обслуживанием.

5. Алгоритм численного расчета функционала оценки пиковой информационной нагрузки на основе аппроксимации случайных точечных процессов экспоненциальными процессами восстановления.

6. Дискретно-событийный подход к оценке надежности программного обеспечения, использующий покомпонентную технологию моделирования событий возникновения ошибок на входе и выходе программного компонента. Процессы обнаружения и устранения ошибок моделируются случайными точечными процессами, а времена обнаружения ошибок сопоставляются с событиями, при возникновении которых требуется рассчитать вероятностные оценки надежности программных компонентов, которые в зависимости от реализуемых ими алгоритмических структур имеют разные формулы для расчета.

7. Численные алгоритмы дискретно-событийного моделирования процессов оценки надежности ПО, базирующиеся на имитационном моделировании процессов возникновения ошибок в сетевом ПО. Эти алгоритмы предназначены для моделирования случайных точечных процессов и соответствующих им считающих случайных процессов с непрерывным временем: на основе одномерных неоднородных случайных процессов с одним датчиком; на основе одномерных неоднородных случайных процессов с двумя датчиками; двумерных неоднородных случайных процессов на круге и в области, ограниченной некоторой функцией.

8. Метод имитационного дискретно-событийного моделирования сетевых систем на основе (*min*,+) фильтраций, метод расчета (*min*,+) интегральной свертки и его реализация на функциональном языке программирования.

9. Метод имитационного дискретно-событийного моделирования производительности сетевых систем с гарантированным обслуживанием и его реализация на функциональном языке программирования.

10. Метод имитационного моделирования и алгоритм прогнозирования надежности ПО путем модификации модели, построенной на неоднородном пуассоновском процессе, в котором применен способ размножения выборок, известный как бутстреп-метод. Реализована программа для оценки прогнозирования надежности ПО на императивном языке программирования.

СПИСОК ЛИТЕРАТУРЫ

1. Бартлетт, М.С. Введение в теорию случайных процессов. – М.: Изд-во иностранной литературы, 1958.

2. Бражник А. Н. Имитационное моделирование: возможности GPSS WORLD. — СПб..: Реноме, 2006. – 439 с.

3. Бутакова М.А. Имитационное моделирование процессов возникновения ошибок для оценки надежности ПО / Бутакова М.А., Чубейко С.В. // Известия высших учебных заведений. Северо-Кавказский регион. Технические науки. – 2012. – №5 (168). – С.29-34.

4. Бутакова М.А. Модели информационных потоков в системах массового обслуживания на транспорте. Монография – Ростов н/Д: Изд-во РГУ, 2006. – 228 с.

5. Бутакова М.А. Стохастические модели потоков с пиковыми нагрузками // Обозрение прикладной и промышленной математики. – М.: Т. 13, вып.2, – 2006. – С. 216 – 220.

6. Бутакова М.А., Гуда А.Н., Чубейко С.В. Моделирование и оценка качества функционирования сетевого ПО информационно-управляющих систем на транспорте в условиях предельных нагрузок // Вестник Донского государственного технического университета, Т. 11, № 6 (57), 2011. С.875-883.

7. Бутакова М.А., Лябах Н.Н. Анализ критических ситуаций в транспортных системах // Известия вузов. Сев.-Кав. Регион. Сер. Техн. науки. № 2, 2004. С. 70 – 72.

8. Глухов М.М., Елизаров В.П., Нечаева А.А. Алгебра. Т. 2. – М.: Гелиос АРВ. – 2003. – 416 с.

9. Гуда А.Н. Методы, модели и алгоритмы оценки качества функционирования и синтеза надежного программного обеспечения информационно-

управляющих систем на железнодорожном транспорте / Гуда А.Н., Бутакова М.А., Чернов А.В., Чубейко С.В. // Труды Первой научно-технической конференции «Интеллектуальные системы управления на железнодорожном транспорте (ИСУЖТ-2012)», 15-16 ноября 2012 г. – М.: ОАО «НИИАС». – 2012. С. 270-274.

10. Гуда А.Н. Прогнозирование надежности ПО на основе модели неоднородного пуассоновского процесса и бутстреп-методов / Гуда А.Н., Чубейко С.В. // Программные продукты и системы. – 2012. – №3. – С. 131-135.

11. Котов В.Е. Сети Петри.- М.: Наука, 1984. – 160 с.

12. Кривулин Н. К. Методы идемпотентной алгебры в задачах моделирования и анализа сложных систем. СПб.: Изд-во С.-Петерб. ун-та, 2009. – 256 с.

13. Литвинов Г.Л. Деквантование Маслова, идемпотентная и тропическая математика: краткое введение // Записки научных семинаров ПОМИ, т. 326, 2005. – С. 145-182.

14. Литвинов Г.Л., Маслов В.П., Соболевский А.Н. Идемпотентная математика и интервальный анализ // Вычислительные технологии, т.6, №6, 2001. – с. 47-70.

15. Лоу А.М., Кельтон В.Д. Имитационное моделирование. Классика CS 3-е изд. СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.

16. Лябах Н.Н., Бутакова М.А. Системы массового обслуживания: развитие теории, методология моделирования и синтеза: Монография. Южн. науч. центр РАН, Рост. гос. ун-т путей сообщения.–Ростов н/Д, 2004. – 200 с.

17. Магнус Я.Р., Катышев П.К., Пересецкий А.А. Эконометрика. Начальный курс. — М.: Дело, 2007. — 504 с.

18. Маслов В. П., Колокольцов В. Н. Идемпотентный анализ и его применение в оптимальном управлении. — М.: Наука, 1994. – 144 с.

19. Месарович М., Такахара Я. Общая теория систем: математические основы. М.: Мир, 1978.

20. Москат Н.А., Чубейко С.В. Моделирование телекоммуникационного трафика высокопроизводительных распределенных информационно-вычислительных систем железнодорожного транспорта // Труды Международной научно-практической конференции «Транспорт 2013». Часть 2. Технические науки – Ростов н/Д: Рост. гос. ун-т путей сообщения, 2013. – С. 65.

21. Паращенко И.Г., Чубейко С.В. Динамические модели оценки эффективности программного обеспечения // «Строительство-2013»: Экономика и управление: проблемы, перспективы, тенденции: материалы Международной научно-практической конференции. – Ростов н/Д: Рост. гос. строит. ун-т, 2013. – С. 234-236.

22. Паращенко И.Г., Чубейко С.В. Статистические методы оценки надежности программного обеспечения // «Строительство-2013»: Экономика и управление: проблемы, перспективы, тенденции: материалы Международной научно-практической конференции. – Ростов н/Д: Рост. гос. строит. ун-т, 2013. – С. 237-239.

23. Рыжиков Ю.И. Имитационное моделирование. Теория и технологии. М.: Альтекс-А, 2004. – 384 с.

24. Таха Х.А. Введение в исследование операций. 7-е издание: Пер. с англ. – Издательский дом «Вильямс», 2005. – 912 с.

25. Финаев В.И. Алгоритмизация и имитационное моделирование с применением аппарата систем массового обслуживания. Учебное пособие. Изд-во ТРТУ, Таганрог, 2003. – 155 с.

26. Чубейко С.В. Алгоритмы и программное обеспечение для имитационного моделирования сетевых систем на основе (min,+) фильтраций // Современные проблемы науки и образования. – 2013. – № 6; URL: http://www.science-education.ru/113-11545 (дата обращения: 13.01.2014).

27. Чубейко С.В. Дискретно-событийное моделирование сетевых компьютерных систем в условиях высоких нагрузок // Фундаментальные исследования, № 8 (часть 2), 2014.– С. 340-344.

28. Чубейко С.В. Модели и методы сетеметрии в задачах разработки высоконагруженных информационно-управляющих систем на транспорте // Труды Всероссийской научно-практической конференции «Транспорт-2011», май 2011 г. Часть 1. Естественные и технические науки. Рост. гос. ун-т путей сообщения. Ростов н/Д. – 2011. – С. 59-61.

29. Чубейко С.В. Моделирование систем с гарантированным обслуживанием: элементы теории и программная реализация // Сборник научных статей по итогам Международной заочной научно-практической конференции «Теоретические и практические аспекты развития науки», 14-15 октября 2013 г. – СПб.: Изд-во «КультИнформПресс». – 2013. – С 113-120.

30. Чубейко С.В., Бутакова М.А. Некоторые аспекты математических моделей очередей с обеспечением качества обслуживания // Обозрение прикладной и промышленной математики, т.18. вып. 1., 2011. С 165-166.

31. Шелухин, О.И. Самоподобие и фракталы. Телекоммуникационные приложения / О.И. Шелухин, А.В. Осин, С.М. Смольский // – М.: ФИЗМАТЛИТ, 2008. – 368 с.

32. Эфрон Б. Нетрадиционные методы многомерного статистического анализа: Сб. статей: Пер. с англ./Предисловие Ю. П. Адлера, Ю. А. Кошевника. — М.: Финансы и статистика, 1988. — 263 с: ил.

33. Anni Princy B., Sridhar S. An Efficient Software Reliability Growth Models with Two Types of Imperfect Debugging // European Journal of Scientific Research. Vol. 72, № 4, 2012. PP. 490 – 503.

34. Asad Ch.A., Ullah M.I., Rehman M.Y. An Approach for Software Reliability Model Selection // International Computer Software and Applications Conference (COMPSAC), 2004. PP. 534 – 539.

35. Baccelli F. Synchronization and linearity: An algebra for discrete event systems / Baccelli F., Cohen G., Olsder G.J., Quadrat J.-P. – Chichester: Wiley. – 1992. – 514 p.

36. Baccelli F., Cohen G., Olsder G. J., Quadrat J.-P. Synchronization and linearity: An algebra for discrete event systems. Chichester: Wiley, 1992. – 514 p.

37. Banks J. Discrete-event System Simulation. Pearson Prentice Hall, 2005. – 608 p.

38. Brunsch T., Raisch J., Hardouin L., Boutin O. Discrete-Event Systems in a Dioid Framework: Modeling and Analysis // Control of Discrete-Event Systems, LNCIS 433, Springer-Verlag London, 2013. – pp. 431-450.

39. Butkovic P. Max-linear Systems: Theory and Algorithms. Springer-Verlag, London, 2010. – 286 p.

40. Cassandras C.G., Lafortune S. Introduction to Discrete Event Systems. Second Edition. – Springer, 2008. – 782 p.

41. Chang C.-S. Performance Guarantees in Communication Networks. – Springer-Verlag, London. – 2000. – 392 p.]

42. Chernov A., Butakova M., Chubieko S., Klimanskaja E. Simulation Models and Algorithms based on Stochastic Jump Processes with Time Substitution // International Journal of Simulation- Systems, Science and Technology- IJSSST V14 - IJSSST: Vol. 14, No. 6. PP. 14-24. URL: http://ijssst.info/Vol-14/No-6/paper3.pdf

43. Coutinho J. S. Software Reliability Growth - IEEE Symposium on Computer Software Reliability, 1973.

44. Eckberg, A.E. Generalized Peakedness of Teletraffic Processes / A.E. Eckberg // Proceeding of the 10-th International Teletraffic Congress, Montreal, Canada, 1983.

45. Fische, G. Mathematics for Engineers // G. Fische, G. Hebuterne / Wiley, – 2008.

46. Gaubert S. Max Plus. Methods and applications of (max,+) linear algebra. Lecture Notes in Computer Science Volume 1200, 1997, pp 261-282.

47. Giambene, G. Queueing Theory and Communications: Networks and Applications // G. Giambene. / Springer, – 2005.

48. Goel A.L., Okumoto K. Time-Dependent Error Detection Rate Model for Software Reliability and other Performance Measures // IEEE Transactions on Reliability, R-28(3), 1979. PP. 206 – 211.

49. Gondran M. Graphs, Dioids and Semirings. New Models and Algorithms. / Gondran M., Minoux M. – Springer Science+Business Media, LLC. – 2008. 384 p.

50. Gondran M., Minoux M. Graphs, Dioids and Semirings. New Models and Algorithms. Springer, New York, 2008. – 401 p.

51. Heidergott B. Max Plus at Work. Modeling and Analysis of Synchronized Systems: A course on Max-Plus Algebra and Its Applications // Heidergott B., Jan Olsder G., van der Woude J. – Princeton University Press, Princeton. – 2006. – 232 p.

52. Hoang Pham. System Software Reliability. Springer-Verlag Limited. – London, 2006. P.440.

53. Hrúz B, Zhou M.C. Modeling and Control of Discrete-event Dynamic Systems: with Petri Nets and Other Tools. Springer London, 2007. – 341 p

54. J. Y. Le Boudec, P. Thiran. Network Calculus. New York: Springer-Verlag, 2001, vol. LNCS 2050, Lecture Notes in Computer Science.

55. Jiang Y. Stochastic Network Calculus / Jiang Y., Liu Y. – Springer-Verlag, London. – 2008. – 240 p.

56. Lai R., Garg M. A Detailed Study of NHPP Software Reliability Models // Journal of Software, Academy Publisher, vol. 7, № 6, 2012. PP. 1296 – 1306.

57. Lewis P.A.W., Shedler G.S. Simulation of nonhomogenous Poisson process with log-linear rate function. Biometrika, № 63, 1976. PP. 501 – 505.

58. Lhommeau M., Hardouin L., Cottenceau B., Jaulin L. Interval analysis and dioid: application to robust controller design for timed event graphs // Autimatica, v.40(11), 2004. – pp.1923-1930.

59. Luy M.R. Handbook of sotware reliability engineering - IEEE Computer Society Press, 1996.

60. MCEneaney W.M. Max-Plus Methods for Nonlinear Control And Estimation. Birkhauser, Boston, 2006. – 258.

61. Molnar, S. Peakedness Characterization in Teletraffic // S. Molnar, G. Miklos / Proceeding PICS '98 Proceedings of the IFIP TC6/WG6.3 Seventh International Conference on Performance of Information and Communication Systems. PP. 97 – 110.

62. Moranda P.L., Jelinski Z. Final Report on Software Reliability Study - McDonnell Douglas Astronautics Company, 1972.

63. Musa J. D., Okumoto, K. Software Reliability Models: Concepts, Classification, Comparisons, and Practice - Electronic Systems Effectiveness and Life Cycle Costing, 2000.

64. pascalabc.net [электронный ресурс, проверено 02.07.2014 г.].

65. Ross S. Simulation. Elsevier, Burlington, M.A. 2006.

66. Shooman M.L. Operational Testing and Software Reliability Estimation During Program Developments - IEEE Computer Society, 1973.

67. Spacek P., Komenda J. Modeling of interval P-timed Petri nets using dioid algebra // In Proc. of 10-th Int. Workshop on Discrete Event Systems (WODES'10), 2010, – pp. 312-317.

68. Tyszer J. Object-oriented Computer Simulation of Discrete-event Systems? Springer US, 1999. – 258 p.

69. Wainer G.A. Discrete Event Modeling and Simulation: A Practitioner's Approach. – CRC Press, 2009. – 486 p.

70. William F. Software reliability modeling survey - Naval Surface Warfare Center, 1996.

71. www.jsoftware.com [электронный ресурс, проверено 05.09.2014 г.].

72. Y. Jiang, Y. Liu. Stochastic Network Calculus. New York: Springer, 2008.

73. Yadav A., Khan R.A. Critical Review on Software Reliability Models // International Journal of Recent Trends in Engineering. Vol.2, №3, 2009. PP.114-116.

ПРИЛОЖЕНИЕ. АКТЫ О ВНЕДРЕНИИ

АКТ

о практическом внедрении результатов диссертации С.В. Чубейко «Аналитические и имитационные методы дискретно-событийного моделирования в задачах анализа надежности и производительности компьютерных систем» в деятельность ООО «Альянс-Телеком»

Настоящим актом подтверждается, что результаты диссертационной работы имеют практическое внедрение в деятельности телекоммуникационной компании ООО «Альянс-Телеком», находящейся в г. Ростов-на-Дону, по адресу 344018, пер. Доломановский 132а литер «г».

В основной деятельности телекоммуникационной компании ООО «Альянс-Телеком» используются следующие результаты:

– математическая модель пиковых (максимальных) информационных нагрузок сетевых систем;

– дискретно-событийная модель высоконагруженной сетевой системы;

– имитационная модель сетевой системы с гарантированным обслуживанием;

– программная реализация на функциональном языке программирования имитационных моделей и оценок производительности сетевых систем в гарантированным обслуживанием;

– программная реализация на функциональном языке программирования основного уравнения баланса сетевой системы;

– программная реализация на императивном языке программирования оценки прогнозирования надежности сетевого ПО.

Указанные результаты используются для практической оценки производительности телекоммуникационных систем, а также для обеспечения качественного и гарантированного доступа абонентов в Интернет.

Директор ООО
«Альянс-Телеком» Шамараков И.П.