Маликов Рамиль Фарукович, профессор, доктор физико-математических наук, заведующий кафедрой информационных и полиграфических систем и технологий Башкирского государственного педагогического университета им. М. Акмуллы.

Научные интересы профессора Р.Ф.Маликова сконцентрированы в области нелинейной оптики, информатики и информационных технологий, математического моделирования процессов и систем, теории систем и системного анализа. Автор свыше 150 научных учебно-методических работ, в том числе более 10 книг. Отличник образования Республики Башкортостан (1998). Почетный работник ВПО РФ (2008). E-mail: rfmalikov@mail.ru

Основы разработки компьютерных моделей сложных систем

В данном учебном пособии рассмотрены основные технологии и принципы конструирования программных продуктов, проведен обзор направлений и подходов к разработке компьютерных моделей и программ моделирования сложных систем. Рассмотрены технологии построения компьютерных моделей сложных систем в средах автоматизации аналитического (MathCAD, MATLAB), технического (Vissim) и имитационного (Arena, Anylogic, HPsim) моделирования.

Предназначено для бакалавров и магистров, обучающихся по различным направлениям подготовки, а также аспирантов, инженеров, научных работников, специализирующимся в области математического моделирования.

Р.Ф. Маликов Основы разработки компьютерных моделей сложных систем

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. М.АКМУЛЛЫ

Р.Ф.МАЛИКОВ

Основы разработки компьютерных моделей сложных систем

Учебное пособие

Уфа 2012

УДК 681.3
ББК 32.973.2-018
М 19

Печатается по решению учебно-методического совета Башкирского государственного педагогического университета им. М.Акмуллы

Маликов Р.Ф. Основы разработки компьютерных моделей сложных систем [Текст]: учеб. пособие. – Уфа: Изд-во БГПУ, 2012. – 256с.

В данном учебном пособии рассматриваются основные технологии конструирования программных продуктов, принципы, направления и походы к разработке компьютерных моделей и программ моделирования сложных систем, а также системы автоматизации аналитического (MathCAD, MATLAB), технического (Vissim) и имитационного (Arena, Anylogic, HPsim) моделирования.

Предназначено для бакалавров и магистров, обучающихся по направлениям подготовки «Прикладная информатика», «Профессиональное обучение (информатика, ВТ и компьютерные технологии)», «Информационные системы», а также аспирантов, инженеров, научных работников, специализирующимся в области математического моделирования.

Рецензенты: В.Е. Гвоздев, д-р техн. наук, профессор УГАТУ
М.М. Валиев, д-р техн. наук, профессор БГАУ

ISBN 978-5-87978-829-7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные специалисты в области информатики, информационных систем и технологий, математического моделирования и компьютерных наук в рамках университетской подготовки получают подготовку по информатике в широком его смысле. Эта подготовка предусматривает обучение основам программирования на языках Pascal, Fortran и др., сред программирования Delphi и группы Visual Studio, современным языкам программирования (C++, JAVA), языкам объектно-ориентированного, дискретно-событийного программирования, UML–моделированию, параллельному программированию и др.

Тем не менее, опыт показывает, что они имеют недостаточные знания по инженерии проектирования и управления проектами, качеству, конфигурации и соответствующим стандартам.

В связи с этим предметом обучения современных студентов, будущих разработчиков программного обеспечения, менеджеров программных проектов, тестировщиков, верификаторов, контролеров качества и др. должны стать не только теоретические и прикладные методы проектирования, но и инженерные методы управления коллективом, планирования и оценивания качества выполняемых работ и укладывания в заданные сроки и стоимость проекта.

Компьютерные науки вообще и программная инженерия в частности — очень популярные и стремительно развивающиеся области знаний. В настоящее время разработкой программного обеспечения занимаются как профессиональные программисты, так и специалисты из смежных областей знаний (физики, математики, информатики и др.), а их использование в своей профессиональной работе становится массовой деятельностью. Необходимость программных разработок в различных областях знаний, в производстве и для повседневной жизни возрастает. В связи с постоянно возрастающими объемами таких разработок требуется готовить кадровый потенциал, способный решать проблемы создания новых программных продуктов на инженерной профессиональной основе, используя накопленный запас знаний в области программирования и управления системами. Особенно остро стоит проблема подготовки специалистов в области математического и компьютерного моделирования. Компьютерное моделирование используется в широком спектре отраслей науки, бизнеса, производства, сферах массового обслуживания. Развитие систем и средств автоматизации моделирования сложных систем, в

частности имитационного моделирования, требует анализа стратегий, направлений и подходов к разработке компьютерных моделей сложных систем.

В работе представлены понятия об основных государственных и международных стандартах по разработке программного обеспечения, стадии и этапы разработок, основные технологии конструирования программных продуктов, принципы, направления и походы к разработке компьютерных моделей и программ моделирования на современном этапе, а также некоторые системы разработки компьютерных моделей сложных систем.

Эти знания позволят сформировать компетентность инженера по разработке программных продуктов, в частности по разработке компьютерных программ и систем для математического моделирования реальных объектов, процессов и явлений.

К ним мы относим:

– знания в области системной инженерии;

– знания государственных и международных стандартов;

– умения проводить системный анализ в области предмета разработки;

– умения разрабатывать техническое задание;

– умения определять функциональные и нефункциональные требования к разрабатываемому программному обеспечению и согласования их с заказчиком;

– знания и умения проектирования алгоритмов, структур данных и программных структур программного комплекса;

– владения навыками алгоритмизации и кодирования программного обеспечения;

– владения методами отладки и тестирования ПО;

– знания основных технологий сопровождения и эксплуатации ПО.

Математическое и компьютерное моделирование – развивающаяся отрасль науки - является одним из мощных инструментов познания реального мира. Поэтому основные тенденции и подходы развития математического и компьютерного моделирования помогут определиться с инструментами компьютерного моделирования и выбрать оптимальный путь к созданию компьютерных моделей. В этих целях необходимо иметь:

– знания основных направлений разработки компьютерных моделей;

– знания основных подходов к разработке имитационных компьютерных моделей сложных систем;

– умение работать с системами компьютерной математики, схемотехнического и имитационного моделирования.

В первой главе рассматриваются различные стандарты, стратегии, методологии по разработке и проектированию программного обеспечения, примеры технических заданий. Здесь изложены некоторые понятия программной инженерии (Software engineering) в виде их применения на процессах проектирования, разработки, тестирования и оценки качества программных систем для задач моделирования.

Во второй главе приведены основные направления исследований по разработке компьютерных моделей динамических систем и имитационного моделирования сложных систем. Здесь проведен анализ систем и средств автоматизации аналитического и имитационного моделирования и приведены технологические характеристики некоторых современных систем моделирования.

В третьей главе рассмотрены различные подходы к разработке систем и компьютерных моделей реальных объектов. Основной акцент в данной главе делается на технологии разработки компьютерных имитационных моделей сложных систем. Проведена классификация языков и систем программирования, рассмотрены различные подходы к разработке схемотехнических и имитационных моделей сложных объектов для проведения исследования и анализа процессов и явлений, происходящих в этих системах.

Четвертая глава посвящена примерам разработки аналитических компьютерных моделей в системах MathCAD и MATLAB, моделей систем массового обслуживания в средствах имитационного моделирования GPSS, Arena и некоторым возможностям создания моделей в среде AnyLogic.

В конце каждой главы студентам предлагаются контрольные вопросы и или задания по изложенной теме. В конце пособия – рекомендуемая литература, список сокращений и глоссарий.

Данное пособие предназначено для студентов, аспирантов, преподавателей, инженеров, разработчиков компьютерных моделей и научных работников. Автор надеется, что изложенная информация поможет разработчикам компьютерных моделей, научным специалистам в области математического моделирования оперативно выбрать систему моделирования, построить адекватные прикладные модели, способы их решения, перейти к полномасштабному исследованию реального явления или процесса на модели, оценить решения моделей, представить и прогнозировать поведение и закономерности изучаемого объекта.

ГЛАВА 1. ЭЛЕМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Процесс разработки программных систем тесно связан с областью управления проектами, потому что любой программный продукт является уникальным результатом. От организации этого процесса напрямую зависят основные характеристики выполнения программного проекта – сроки выполнения, запланированный бюджет, качество выпускаемого продукта.

Но профессиональное управление проектами само по себе не может обеспечить достижения указанных характеристик. Немаловажную роль в этом играет архитектура программной системы, опыт и квалификация участников команды разработки, а также правильное документирование всех процессов разработки программного обеспечения.

В данной главе рассматриваются различные российские и международные стандарты, регламентирующие организацию процесса разработки, документирование отдельных этапов и предназначено для изучения последовательности действий программиста и исследователя для разработки и проектирования компьютерных моделей и программного обеспечения для моделирования реальных объектов. В целях ознакомления приводятся различные стратегии разработки программного обеспечения (модели жизненного цикла программного продукта). Более подробно эти стратегии проектирования, разработки и тестирования программного обеспечения можно изучить по книгам [1-12, 27-29, 33,34, 36-49].

1.1. Стандарты, стадии и этапы разработок программного обеспечения

Рассмотрим международные и различные государственные стандарты на разработку программного обеспечения или автоматизированной системы, документации, информационного, технического обеспечения и т.д. [14-26, 30-32].

Стандарты комплекса ГОСТ 19 представляют собой всеобъемлющий комплекс, который устанавливает целевое назначение, область распространения, классификацию и правила обозначения стандартов, входящих в комплекс Единой системы программной документации (ЕСПД).

Единая система программной документации – комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ и программной документации. В стан-

дартах ЕСПД устанавливают требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ.

Стандарты комплекса ГОСТ 34 были созданы для стандартизации и увязки межотраслевых документов в конце 80-х годов. Объектами стандартизации являются автоматизированные системы различных видов, не только программное обеспечение и база данных. Наиболее популярными считаются стандарты ГОСТ 34.601-90 (стадии создания автоматизированной системы), ГОСТ 34.602-89 (техническое задание на создание автоматизированной системы), ГОСТ 34.603-92 (виды испытаний автоматизированных систем) и методические указания РД 50-34.698-90 (требования к содержанию документов). Стандарты предусматривают стадии и этапы выполнения работ по созданию автоматизированной системы, но не предусматривают сквозных процессов в явном виде.

Кроме перечисленных стандартов в Российской Федерации действует серия стандартов в части документирования и оценки качества программного продукта, разработанных на основе прямого применения международных стандартов ISO 9000 – ISO 9004, так называемые ГОСТ Р [19-26]. Это самые новые стандарты. Некоторые из них напрямую адресованы руководителям проекта или директорам информационных служб.

Базовым стандартом регламентирования качества программного продукта ранее являлся международный стандарт ISO 9126: 1991 (ГОСТ Р ИСО/МЭК 9126 – 93) «Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению». За последние годы разработано ряд стандартов ISO 9126-1, ISO 9126-2, ISO 9126-3, ISO 9126-4, регламентирующих процессы и продукты жизненного цикла программных продуктов и баз данных, которые могут служить основой для систем обеспечения качества программных продуктов.

Международный стандарт ISO/IEC 12207: 1995-08-01 является базовым на процессы жизненного цикла разработки различных видов программного обеспечения и типов проектов автоматизированных систем. В этом стандарте не предусмотрено каких-либо этапов жизненного цикла информационной системы. Этот стандарт определяет лишь ряд процессов: приобретение, поставка, разработка и т.п. Согласно ISO/IEC 12207, каждый процесс подразделяется на ряд действий, каждое действие – на ряд задач. Стандарт принципиально не содержит описания конкретных методов действий, а тем более – заготовок решений и документаций. Стандарт не предписывает имена, форматы или точное содержание получаемой документации. Решение такого типа принимаются сторонами, использующими стандарт. Степень обязательности рас-

сматриваемого стандарта следующая: после решения организации о применении ISO 12207 в качестве торговых отношений является ее ответственность за указание минимального набора требуемых процессов и задач, которые обеспечивают согласованность с этим стандартом.

Международный стандарт ISO 14598 посвящен методологии и стандартизации оценки характеристик качества готовых программных средств и их компонентов на различных этапах жизненного цикла.

Методологические и системные задачи оценки комплексной защиты информационных систем определены в трех частях стандарта ISO 15408: 1999-1 – ISO 15408: 1999-3 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий».

В 1987 году американский институт Software Engineering Institute (SEI) выпустил краткий обзор процессов разработки ПО с описанием их уровней зрелости и опросник, предназначенный для выявления проблем разработки ПО. На основе этих исследований 1991 году был выпущен, а 1992 году в новой версии разработан стандарт Capability Maturity Model for Software (CMM). Этот стандарт представляет интерес для разработчиков, так как здесь представлена шкала оценки зрелости процессов разработки ПО в компании и для измерения их параметров.

Кроме вышеперечисленных стандартов существуют и другие методики разработки прикладных информационных систем[1]:

Custom Development Method (методика Oracle) – технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle. Применяется CDM для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий "быстрой разработки" (Fast Track) или "облегченного подхода", рекомендуемых в случае малых проектов.

Rational Unified Process (RUP) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках RUP – это создание и сопровождение моделей на базе UML.

[1] http://it-claim.ru/Education/Course/ISDevelopment/Lecture_5.pdf

Microsoft Solution Framework (MSF) сходна с RUP, также включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной, предполагает использование объектно-ориентированного моделирования. MSF в сравнении с RUP в большей степени ориентирована на разработку бизнес-приложений.

Extreme Programming (XP). Экстремальное программирование (самая новая среди рассматриваемых методологий) сформировалось в 1996 году. В основе методологии командная работа, эффективная коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

Информацию об этих стандартах и методологиях можно найти в сети Интернет.

Разработка автоматизированной системы и реализация технического, программного и информационного обеспечения для решения математической модели имеет свои этапы и особенности. Рассмотрим эти этапы с точки зрения понятия жизненного цикла программы. Программный продукт создается группой разработчиков, оно имеет время жизни, которое зависит от многих факторов и продлевается в результате модификаций. Жизненный цикл программ можно представить из двух основных периодов:

- разработки
- использования и сопровождения.

Период разработки состоит из трех стадий:

- предпроектной,
- проектирования
- внедрения.

Рассмотрим жизненный цикл автоматизированной информационной системы (программного обеспечения) согласно ГОСТ 34. Согласно этому ГОСТу разработка информационной системы разбивается на стадии и этапы.

1. Предпроектная стадия – так называемая стадия формирования требований к автоматизированной системе.

Основная цель предпроектной стадии является обоснованием необходимости и целесообразности создания программных систем. В результате предпроектного обследования объекта определяются цели разработки, эффективность и необходимость. При предпроектном исследовании осуществляется изучение и анализ существующих аналогичных программных продуктов. Выявляются цели, критерии эффективности, существующие ограничения для разработки программного продукта в целом и ее модулей, уточня-

ется перечень модулей, выполняемые ими функции, решаемые задачи. Эта стадия подразделяется на ряд этапов.

Этап разработки концепции автоматизированной системы. На этом этапе проводятся следующие работы:

1. Предварительное обследование, изучение объекта и обоснование необходимости разработки автоматизированной системы;
2. Проведение необходимых научно исследовательских работ;
3. Формирование требований заказчика и разработка вариантов концепции автоматизированной системы удовлетворяющей заказчику;
4. Разработка отчета о проделанной работе и заявки на разработку технического задания.

Этап разработки и утверждение технического задания на разработку автоматизированной системы.

Техническое задание представляет собой утвержденный в установленном порядке документ, определяющий цели, требования и основные исходные данные, необходимые для разработки системы, и содержащий предварительную оценку ее экономической эффективности.

Существует два разных ГОСТа, позволяющих разработать техническое задание:

При создании программного продукта или его компонента используется ГОСТ 19.201-78 «Единая система программной документации. Техническое задание. Требования к содержанию и оформлению».

При разработке автоматизированной информационной системы используется ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».

Техническое задание определяет требования к задачам программного обеспечения по компьютерному моделированию, техническому, информационному и математическому обеспечению программы моделирования и регламентирует организацию разработки, объемы работ и затраты.

При разработке технического задания устанавливаются очереди создания и определяется перечень модулей и математических задач, предусмотренных в составе каждой очереди.

Техническое задание – это официальный документ, определяющий требования к создаваемой системе:

- основание для разработки: постановление или приказ вышестоящей организации;

- описания среды использования программного обеспечения;
- состав модулей и математических задач с обоснованным указанием очередности их разработки и внедрения;
- укрупненные схемы алгоритмов решения математической задачи и задачи разработки интерфейса пользователя (блок-схемы, псевдокоды, диаграммы и т.д.);
- перечень предварительно выбранных технических средств;
- намечаемый размер затрат и укрупненный расчет экономической эффективности.

Для задач математического моделирования на этом этапе могут быть проведены следующие работы:

– определение платформы или типа операционной системы для разработки программы (MS DOS, Windows, Unix, Mac);

– оценка необходимости сетевого варианта программы;

– определение базовых сред и языков программирования Visual Studio (Visual Basic, Visual C++, Visual C#), Pascal, Fortran и сред разработки интерфейса программного продукта Delphi, Dreamviewer, FrontPage и др.;

– определение требований к средствам обработки результатов моделирования;

– определение требований к интерфейсу конечного пользователя.

После экспертизы и корректировки техническое задание утверждают в установленном порядке.

2. Стадия проектирования

После утверждения технического задания выделяют специализированные группы, каждая из которых ведет разработку одной или нескольких модулей (подсистем). Эти группы уточняют перечень задач по функциональным подсистемам, их постановку и алгоритмизацию. Группы работают вместе с разработчиками программного обеспечения, проводя взаимное согласование состава и характеристик входных и выходных сигналов. Отдельные группы специалистов создают разделы технического проекта, относящиеся к техническим средствам экономической эффективности. Результатом работы всех групп является технический проект.

Проектная стадия подразделяется на ряд этапов.

Этап разработки эскизного проекта автоматизированной системы. Эскизный проект – документированное описание предлагаемого программного обеспечения. Здесь разрабатывают предварительные проектные

решения по всей системе и по ее частям, определяются спецификации, создаются формальные модели (диаграммы потоков данных, «сущность-связь», переходов состояний, функциональные диаграммы). Его подготовка позволяет выполнить начальные этапы проектирования, представить заказчику в удобной форме намечаемые основные проектные решения. Если принято решение о разработке эскизного проекта, он должен быть согласован и утвержден заказчиком. На стадии подготовки технического проекта решения, содержащиеся в эскизном проекте, корректируют и детализируют.

Этап разработки технического проекта. Технический проект – представляет собой утвержденную в установленном порядке техническую документацию, содержащую общесистемные проектные решения (описания, схемы, расчеты, чертежи, числовые показатели и т.д.), алгоритмы решения задач, а также оценку экономической эффективности системы и перечень мероприятий по подготовке объекта к внедрению. Разработка технического проекта ведется на основании утвержденного технического задания в такой последовательности: общий технический проект; технический проект первой очереди; технический проект второй очереди. Разработка технического проекта второй очереди может проводиться независимо от степени завершенности работ по первой очереди. В отдельных сложных случаях, когда невозможно выявить рациональные проектные решения без сопоставления вариантов, на стадии технического проекта должны прорабатываться различные варианты; однако необходимость такой проработки нескольких вариантов должна быть указана в техническом задании на программное обеспечение.

Чаще всего общий технический проект состоит из следующих разделов:

- ведомость технического проекта, общая и пояснительная информация к техническому проекту;
- описание систем классификации и кодирования;
- общая структура программного обеспечения с указанием модулей и общих принципов функционирования системы, перечнем задач, решаемых в составе каждого модуля, и выходные параметры задач, схемы связей между модулями;
- общие принципы математического обеспечения программного продукта, перечень программного обеспечения и СУБД;
- структура комплекса технических средств, необходимого для функционирования программного обеспечения программы моделирования,

перечень аппаратных средств, на которых планируется работа проектируемого программного продукта;

- мероприятия по подготовке к внедрению программного обеспечения, обучения пользователей;
- проектная и экспертная оценка надежности программной системы, расчет экономической эффективности системы;
- график разработки и внедрения программного обеспечения.

Стадии и этапы работы по проектированию описаны в стандарте ГОСТ 34.601-90. Стадия проектирования может состоять из следующих этапов:

- проектирование программного обеспечения, где проводится разработка проектных решений по всей системе и по ее частям;
- проектирование интерфейса пользователя программным продуктом;
- разработка документации на автоматизированную систему и ее оформление.

Этап проектирования программного обеспечения включает следующие работы:

1) формирования требований к программному продукту, корректно и точно отражающего цели и задачи программы моделирования;
2) уточнение состава объектов ПО и структура связей блоков программного продукта или так называемые программные модули;
3) определение информационных систем проектирования, которые используются для разработки моделей состава и структуры связей между объектами программного продукта, методы проектирования, согласно которым разрабатываются алгоритмы обработки информации;
4) проектирование завершается разработкой технического проекта.

Этап проектирования интерфейса пользователя включает следующие виды работ:

1) разработка графического интерфейса для комфорта эксплуатации пользователем программного продукта. Здесь чаще всего используют среды программирования Delphi, Dreamviewer, FrontPage и др. Графический интерфейс пользователя представляется в виде системы спускающихся меню при использовании систем ввода информации: мыши и клавиатуры;
2) разработка экранных окон и форм, содержащих объекты управления, панели инструментов с пиктограммами, выходные модули. Отметим о стандартных требованиях к графическому интерфейсу:

– постоянное местоположение графических объектов на экране:

– линейка меню включает не больше 6 понятий (подменю), каждое из которых содержит не более 6 опций (правило «шести»);

– пункты меню должны содержать привычные и краткие названия.

Работа над проектированием интерфейса пользователя программного обеспечения для задач компьютерного моделирования чаще всего проводится в том случае, когда программный продукт готовится для передачи заказчику, тиражирования или внедрения в фонд алгоритмов и программ.

Этап реализации программного обеспечения или создание программного кода.

На этом этапе осуществляется создание программного обеспечения системы, выполняется разработка программных модулей – программирование, иначе говоря, создание программного кода, которое заключается:

– в разработке блока программ управления функционированием системы;

– в разработке блока программ, реализующих расчетные формулы и функциональные алгоритмы;

– в разработке блока обработки результатов моделирования.

Программисты разрабатывают по системным спецификациям схемы программ и программные спецификации, затем пишут и отлаживают программы; проводят отладку комплексов программ по модулям и задачам.

В настоящее время часто используются готовые программные решения, в которых процесс разработки ПО сводится к:

модульной настройке;

параметрической настройке;

разработке компонентных приложений.

Этап создания и оформления документации технического проекта.

Здесь осуществляется оформление документации в виде ведомости технического проекта и обоснования проектных решений и разработка документации на программное обеспечение для эксплуатации, которая в целом состоит из:

– описания применения, которая дает общую характеристику программного продукта с указанием сферы применимости, технических требований к базовому программному обеспечении и характеристик ЭВМ;

– руководства пользователя, который предназначен для конечного пользователя для освоения программного продукта;

– руководства программиста, в котором указываются особенности установки, состав и назначение блоков, правила эксплуатации программного продукта.

Приведем некоторые из автоматически генерируемых по проекту документов в соответствии с отечественными стандартами на создание автоматизированной системы (шапки документов, листы утверждения и согласования опущены с целью экономии места). Документирование программного обеспечения осуществляется в соответствии с ЕСПД (ГОСТ 19.XXX). ГОСТ 19.101-77 содержит виды программных документов для программного обеспечения различных типов. По теории полный пакет документации может включать следующие документы:

- Пояснительная записка (РД 50-34.698-90);
- Схема функциональной структуры (РД 50-34.698-90);
- Общее описание системы (РД 50-34.698-90);
- Описание автоматизирумых функций (РД 50-34.698-90);
- Описание постановки задачи;
- Описание информационного обеспечения (РД 50-34.698-90);
- Описание организации информационной базы (РД 50-34.698-90);
- Перечень входных сигналов и данных (РД 50-34.698-90);
- Перечень выходных сигналов/документов (РД 50-34.698-90);
- Описание программного обеспечения (РД 50-34.698-90);
- Техническое задание на программное изделие (ГОСТ 19.201-78) или на автоматизированную систему (ГОСТ 34.602-89);
- Описание программы (ГОСТ 19.402-78);
- Пояснительная записка (ГОСТ 19.404-79);
- Программа и методика испытаний (ГОСТ 19.301-79 или ГОСТ 34.603-92).

3. Стадия внедрения

Эта стадия состоит из этапов опытной эксплуатации и ввода в эксплуатацию и выполняется разработчиком совместно с заказчиком. Эта стадия обычно оказывается распределенной во времени, и на этой стадии проводится отладка программных модулей на работоспособность и тестирование программного продукта. В период опытной эксплуатации выявляют и корректируют недостатки предыдущих этапов разработки, выполняют автономный тест каждого модуля, затем весь комплект модулей проходит системный тест – тест внутренней приемки продукта, показывающий уровень его качества. Сюда входят тесты функциональности и тесты надежности программного продукта [33, 37, 40]. Системное тестирование, выполняемое разработчиками в реальных условиях, называется альфа-тестированием.

Последний тест проверки программного обеспечения – это приемосдаточные испытания специалистами заказчика. Такой тест предусматривает

показ программного обеспечения заказчику и должен содержать группу контрольных тестов, моделирующих реальные процессы. В конце опытной эксплуатации обучают пользователя, отлаживают технические средства и проверяют возможность работы программного обеспечения при полной нагрузке в реальном масштабе времени. Приемо-сдаточные испытания должны регламентироваться ГОСТ 34.603-92. Это иногда называют бета-тестированием.

В случае разработки пакета программ, имеющего универсальный востребованный характер, производится внедрение программного продукта, т.е. передача системы клиентам для использования в производстве. Если при использовании программного обеспечения (эксплуатации) и его сопровождение обнаруживается (возникают) новые условия не осознанные разработчиком на ранних стадиях, то осуществляется переход к новому периоду разработки.

4. Пользовательский период или стадия сопровождения.

Жизненный цикл разработанного программного продукта должен поддерживаться постоянно, исправляться. Тем самым отодвигать старение программного продукта. Старение программного продукта может быть уже на стадии проектирования из-за сложности, длительности срока разработки или недостаточного понимания значения предпроектной стадии. Разработанная система должна быть гибкой, и это свойство должно поддерживаться в течение жизненного цикла программного продукта. На этой стадии проводится выполнение работ в соответствии с гарантийными обязательствами и послегарантийное обслуживание. Последняя стадия жизненного цикла программы характеризуется такими потребительскими свойствами, как:

– функциональная полнота – это степень реализации предусмотренных к автоматизации процессов обработки данных;

– валидность – программа должна решать поставленную задачу;

– комфорт эксплуатации связан с разработкой удобного интерфейса, дизайна, наличия автоматизированной обучающей программы и средства оказания помощи при неправильных действиях;

– эффективность работы программы – оптимизированные ресурсы памяти, времени решения задачи, размеры установочного пакета;

– надежность – программа должна правильно работать при тестовых исходных данных, выдавать результаты моделирования или диагностику ошибок;

– изменяемость – возможность внесения изменений, как во время разработки программ, так и после выпуска программной продукции. Это обычно

достигается использованием методов модульного или объектно-ориентированного программирования;

– читабельность кода программ, легкость отладки и тестирования определяется культурой разработки программ, правильностью организации кодов и наличием средств отладки и контрольных примеров для тестирования;

– переносимость – это возможность переноса программного средства из одного типа вычислительной среды в другую: с одного компьютера на другой; с одной операционной системы в другую; с одной конфигурации на другую.

Эти потребительские свойства характерны для программных продуктов, ориентированных на научно-вычислительный эксперимент, с другой стороны эти свойства можно определить как критерии качества программного продукта для задач математического моделирования и использовать при экспертной оценке программного продукта.

Процесс функционирования программного продукта является важной стадией жизненного цикла, который может быть использован в ежедневной работе, причем использование предыдущей автоматизированной системы сокращается. Период сопровождения программного продукта совпадает с процессом функционирования системы. Сопровождение подразделяется на действия:

- корректирующие – выявление дефектов и ошибок ПП;
- адаптивное – изменение ПП в ответ на изменения в вычислениях;
- совершенствующее – развитие ПП путем добавления новых функций и свойств.

Степень адаптивности ГОСТ 34. При использовании стандарта ГОСТ 34 имеются следующие возможности:

- объединять этапы формирования требований и разработки концепции системы;
- отказаться от этапа эскизного проектирования и объединять этапы разработки и рабочей документации;
- отказаться от некоторых стадий разработки;
- объединять большинство документов технической документации и их разделов;
- вводить дополнительные документы, разделы документов и работы;
- динамически создавать частные технические задания, что позволяет достаточно гибко формировать жизненный цикл программного продукта.

Стадии и этапы, выполняемые организациями – участниками работ по созданию автоматизированной системы, устанавливаются в договорах и техническом задании, что близко к подходу ISO 12207.

Обеспечение качества согласно ГОСТ 34 определяется в техническом задании.

Степень обязательности ГОСТ 34 и ГОСТ 19: полная обязательность отсутствует, материалы стандартов являются методической поддержкой и в большей степени ориентированы на заказчика. В стандарте имеется набор требований к содержанию технического задания и проведению испытаний разработанной системы.

Ключевым документом взаимодействия сторон является техническое задание на создание автоматизированной системы, оно является основным исходным документом для создания и приемки и определяет важнейшие точки взаимодействия заказчика и разработчика.

1.2. Стратегии разработки (модели жизненного цикла) программного продукта

Международные и государственные стандарты определяют порядок разработок программного обеспечения или автоматизированной системы, документации, информационного, технического обеспечения и т.д.

Стандарт определяет структуру жизненного цикла (ЖЦ), содержащую процессы, действия и задачи, которые должны быть выполнены во время создания и использования программного обеспечения (ПО). В России создание ПО первоначально, в 70-е гг. регламентировалось стандартами серии ГОСТ 19.XXX – Единая система программной документации и ГОСТ 34.XXX – Комплекс стандартов на АС. Однако создание, сопровождение и развитие современного прикладного ПО высокого качества в этих стандартах отражено недостаточно, а отдельные их положения уже устарели. Эти стандарты вынуждены использовать предприятия, выполняющие государственные заказы при создании ПО для внутреннего применения. Однако в экспортных заказах зарубежные клиенты требуют соответствия технологии проектирования, производства и качества продукции современным международным стандартам.

Основным зарубежным нормативным документом, наиболее полно и подробно регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207.

В России часто в качестве нормативного документа, регламентирующего жизненный цикл разработки программного продукта, выбирается ГОСТ 19.102 – 77, в котором определены названия и краткое описание этапов разработки.

Модель ЖЦ зависит от специфики ПО и специфики условий, в которых оно создается и функционирует. Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Тем не менее, часто разработчики программного обеспечения прибегают к модели ЖЦ, называемой моделью «стихийного программирования», согласно которому разработка программного продукта (кодирование) ведется так, как видится и понимается задача программистом. При этом подходе слабо прорабатывается стадия проектирования ПП и не соблюдаются фазы разработки ПП, это приводит к многим недостаткам разработки ПП, в частности к проблемам модификации и тестирования программного продукта. Проектная документация пишется после разработки программы. Отметим, что при таком подходе мы имеем чаще всего ПП низкого качества.

Кроме российских и международных стандартов ISO/IEC 12207: 1995-08-01, ISO 9000 – ISO 9004, ISO 9126 - 1-4 существуют множество других стандартов: в 1992 году разработан и внедрен стандарт Capability Maturity Model for Software (CMM), методика компании Oracle - стандарт ***Custom Development Method*** (CDM), итеративная модель разработки программного обеспечения ***Rational Unified Process (RUP)*** и другие.

Нормативные международные и государственные стандарты позволили ввести различные стратегии конструирования программного обеспечения. В рамках этих стратегий разработаны различные модели (технологии) разработки программного обеспечения. Они являются основополагающими моделями, в зависимости от вида программного обеспечения разработчики используют ту или иную стратегию разработки [1-12, 34, 36-39, 42-44, 47, 49].

1.2.1. Водопадная или каскадная стратегия

Водопадная модель. Эта стратегия была разработана в 1970 году Уинстоном Ройсом и основана на классическом жизненном цикле программного продукта. По этой теории разработка программного обеспечения рассматри-

вается как линейная последовательность этапов, таких как системный анализ, анализ требований, проектирование, кодирование, тестирование и сопровождение [34, 36, 39, 42-44]. Переход на следующий, иерархически нижний этап проводится после полного завершения работ на текущем этапе (рис. 1.1).

На этапе *системного анализа* проводится декомпозиция всей системы будущего программного продукта на элементы (модули) и определяется роль каждого элемента (модуля) в компьютерной системе, взаимодействие элементов (модулей) друг с другом. Ко всем системным элементам определяются требования и назначения подмножества этих требований программному «элементу». Необходимость системного подхода явно проявляется, когда формируется интерфейс ПО с другими элементами (аппаратурой, людьми, базами данных). На этом же этапе начинается решение задачи планирования проекта ПО. В ходе планирования проекта определяются объем проектных работ и их риск, необходимые трудозатраты, формируются рабочие задачи и план-график работ.

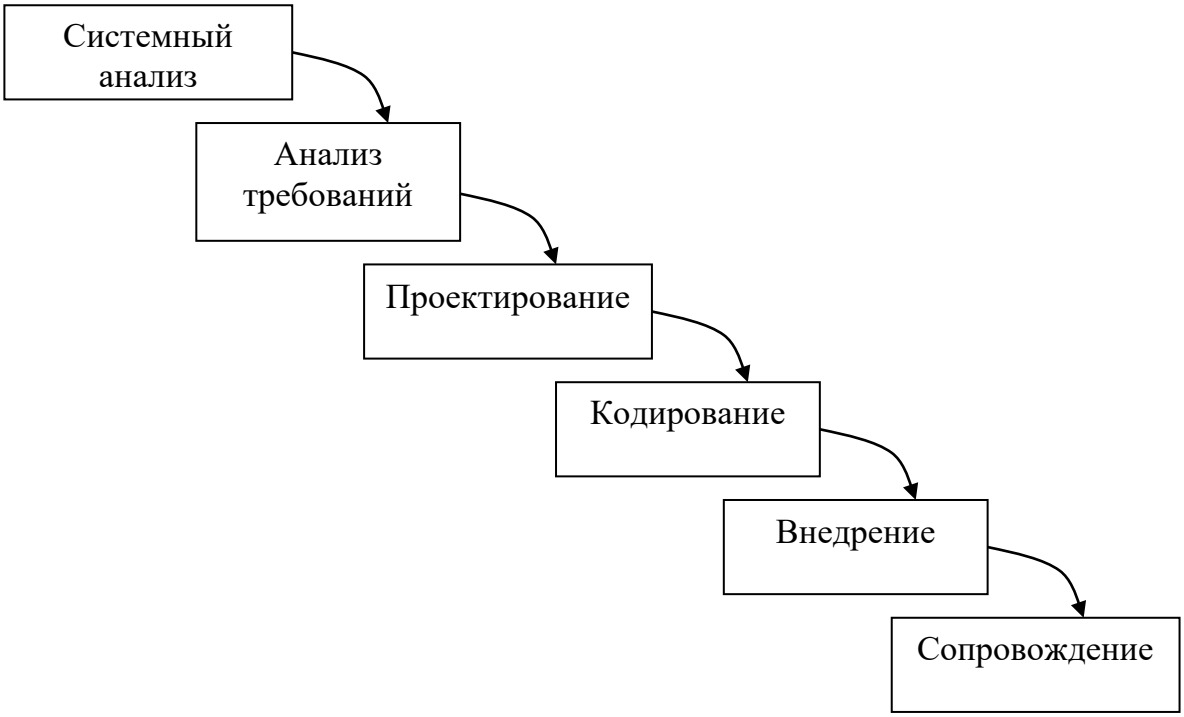


Рис. 1.1. Классический жизненный цикл разработки ПО

Этап *анализа требований* относится к элементу (модулю) программного обеспечения. Уточняются и детализируются его функции, характеристики и интерфейс. Здесь проводится сбор требований к ПО, их систематизация, выявление противоречий, недостающей информации и т.п. Анализ требований делится на три фазы: сбор, анализ и документирование.

Все определения документируются в *спецификации анализа.* Здесь же завершается решение задачи планирования проекта.

Этап *проектирования* состоит в создании представлений:

- архитектуры ПО;
- модульной структуры ПО;
- алгоритмической структуры ПО;
- структуры данных;
- входного и выходного интерфейса (входных и выходных форм данных).

На этом этапе создается описание свойств будущей системы. Исходные данные для проектирования содержатся в *спецификации анализа,* то есть в ходе проектирования выполняется трансляция требований к ПО во множество проектных представлений. При решении задач проектирования основное внимание уделяется качеству будущего программного продукта.

Этап *кодирования* состоит в переводе результатов проектирования в текст на языке программирования, т.е. процесс написания программного кода на определённом языке программирования с целью реализации алгоритмов, определённых на предыдущем этапе — проектировании.

Тестирование — выполнение программы для выявления дефектов в функциях, логике и форме реализации программного продукта. Здесь проводится проверка и испытание законченного продукта на предмет его качества: устойчивости к нагрузкам, дружественности к пользователю (юзабилити), безопасности (устойчивости к взломам), соответствию требованиям и т.п.

Внедрение – это процесс установки и настройки программного продукта, для конкретных условий использования. Также под внедрением подразумевают обучение пользователей работе с данным продуктом.

Сопровождение — процесс поддержки программного продукта. На данном этапе устраняются ошибки («баги»), вносятся изменения в эксплуатируемое ПО с целью улучшить продукт. Эта стадия в жизненном цикле, как правило, занимает большую часть времени.

Цели изменений:

- исправление ошибок;
- адаптация к изменениям внешней для ПО среды;
- усовершенствование ПО по требованиям заказчика.

Сопровождение ПО состоит в повторном применении каждого из предшествующих шагов (этапов) жизненного цикла к существующей программе, но не в разработке новой программы.

Как и любая инженерная схема, классический жизненный цикл имеет достоинства и недостатки.

Достоинства классического жизненного цикла: дает план и временной график по всем этапам проекта, упорядочивает ход конструирования.

Недостатки классического жизненного цикла:

1) реальные проекты часто требуют отклонения от стандартной последовательности шагов;
2) цикл основан на точной формулировке исходных требований к ПО (реально в начале проекта требования заказчика определены лишь частично);
3) результаты проекта доступны заказчику только в конце работы.

Каскадная модель с промежуточным контролем (водоворот). Модернизация каскадной модели усовершенствовалась до 1980 гг. Одна из модернизированных моделей является каскадная модель с промежуточным контролем (рис.1.2). Данная модель является почти эквивалентной по алгоритму предыдущей модели, однако при этом имеет обратные связи с каждым этапом жизненного цикла, при этом порождает очень весомый недостаток: многократное увеличение затрат на разработку программного продукта. Каскадная стратегия чаще всего применяется разработчиками при создании первоначальной версии программного обеспечения.

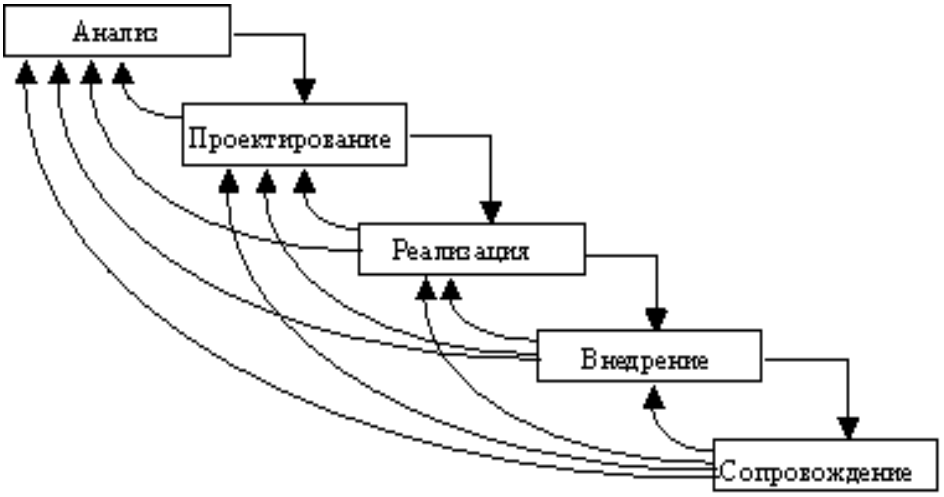


Рис. 1.2. Поэтапная модель с промежуточным контролем

1.2.2. Инкрементная стратегия

Инкрементная модель. В начале процесса определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. Первая версия реализует часть

запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система.

В начале работы над проектом определяются все основные требования к системе, после чего выполняется ее разработка в виде последовательности версий. При этом каждая версия является законченным и работоспособным продуктом. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система.

Данная модель жизненного цикла характерна при разработке сложных и комплексных систем, для которых имеется четкое видение (как со стороны заказчика, так и со стороны разработчика) того, что собой должен представлять конечный результат (информационная система).

Разработка версиями ведется в силу разного рода причин:

- отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
- отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
- появлением новых идей, решений и технологий разработки программного обеспечения;
- требований поэтапного внедрения и освоения продукта конечными пользователями.

Внедрение всей системы сразу может вызвать у ее пользователей неприятие и только «затормозить» процесс перехода на новые технологии. Образно говоря, они могут просто «не переварить большой кусок, поэтому его надо измельчить и давать по частям».

Достоинства и недостатки этой стратегии такие же, как и у классической. Но в отличие от классической стратегии заказчик может раньше увидеть результаты. Уже по результатам разработки и внедрения первой версии он может незначительно изменить требования к разработке, отказаться от нее или предложить разработку более совершенного продукта с заключением нового договора.

Инкрементная модель является классическим примером инкрементной стратегии конструирования. Она объединяет элементы последовательной водопадной модели с итерационной философией макетирования. Каждая линейная последовательность здесь вырабатывает поставляемый инкремент программного обеспечения.

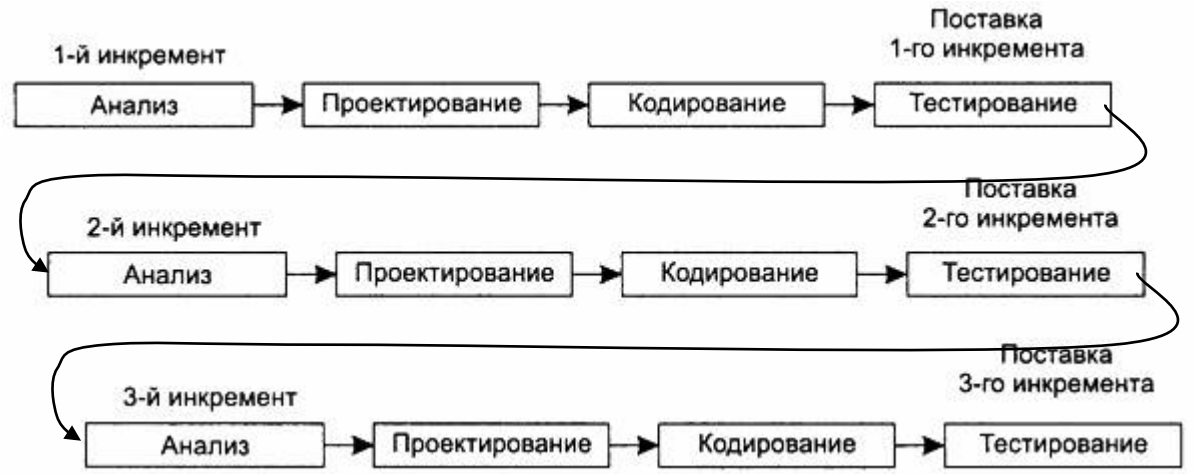


Рис.1.3. Инкрементная модель

Первый инкремент приводит к получению базового продукта, реализующего базовые требования (правда, многие вспомогательные требования остаются нереализованными). План следующего инкремента предусматривает модификацию базового продукта, обеспечивающую дополнительные характеристики и функциональность. По своей природе инкрементный процесс итеративен, но, в отличие от макетирования, инкрементная модель обеспечивает на каждом инкременте работающий продукт.

Примерами инкрементной стратегии разработки являются практически все системы компьютерного моделирования (Derive, Maple, MathCAD, Mathematica, MATLAB и его приложения SIMULINK и др.), системы обработки графической информации (Corel, Adobe Photoshop, Microcal Origin и др.) и многие другие программные продукты, разработанные по технологии разработки версий (инкрементов).

Модель экстремального программирования (разработка через тестирование). Одно из современных реализаций инкрементного подхода - экстремальное программирование XP. Оно ориентировано на очень малые приращения функциональности.

Экстремальное программирование (eXtreme Programming, XP) – облегченный (подвижный) процесс (или методология), главный автор которого Кент Бек (1999). XP – процесс ориентирован на группы малого и среднего размера, строящие программное обеспечение в условиях неопределенных или быстро изменяющихся требований. XP-группу образуют до 10 сотрудников, которые размещаются в одном помещении.

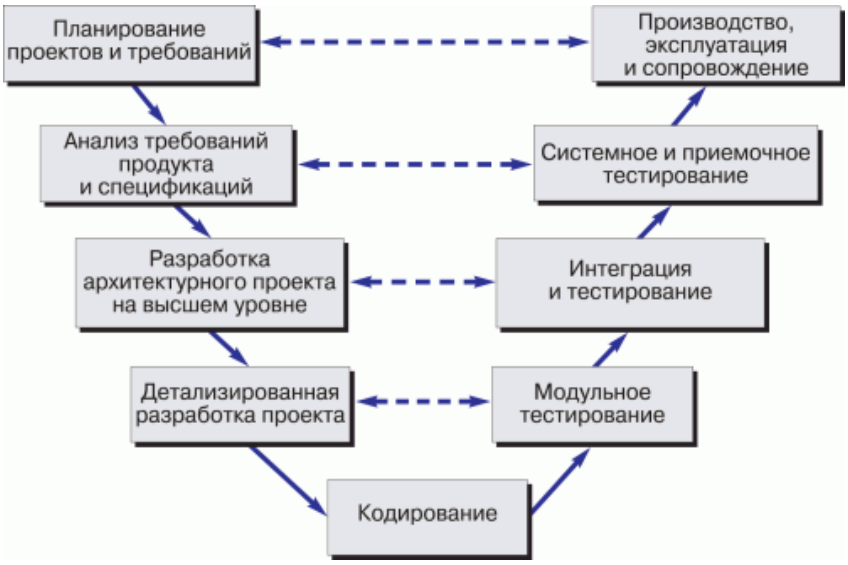


Рис. 1.4. Модель экстремального программирования

Основная идея XP – устранить высокую стоимость изменения, характерную для приложений с использованием объектов, паттернов и реляционных баз данных. Поэтому XP-процесс должен быть высоко динамичным процессом. XP-группа имеет дело с изменениями требований на всем протяжении итерационного цикла разработки, причем цикл состоит из очень коротких итераций. Четырьмя базовыми действиями в XP-цикле являются: кодирование, тестирование, выслушивание заказчика и проектирование. Динамизм обеспечивается с помощью четырех характеристик: непрерывной связи с заказчиком (и в пределах группы), простоты (всегда выбирается минимальное решение), быстрой обратной связи (с помощью модульного и функционального тестирования), смелости в проведении профилактики возможных проблем.

Модель на основе разработки прототипа. Данная модель основывается на разработке прототипов и прототипирования продукта. Прототипирование используется на ранних стадиях жизненного цикла программного обеспечения:

- прояснить неясные требования (прототип UI);
- выбрать одно из ряда концептуальных решений (реализация сценариев);
- проанализировать осуществимость проекта.

Классификация прототипов:

- горизонтальные и вертикальные;
- одноразовые и эволюционные;
- бумажные и раскадровки.

Горизонтальные прототипы — моделируют исключительно UI не затрагивая логику обработки и базу данных. Вертикальные прототипы — проверка архитектурных решений. Одноразовые прототипы — для быстрой разработки. Эволюционные прототипы — первое приближение эволюционной системы.

1.2.3. Эволюционная стратегия

К этой стратегии относится *спиральная модель*, предложенная Барри Боэмом (Barry Boehm) в 1988 году, она стала существенным прорывом в понимании природы разработки ПО. Она представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции, делающая упор на начальные этапы жизненного цикла: анализ и проектирование.

Система также строится в виде последовательности версий, но в начале процесса определены не все требования. Требования уточняются в результате разработки версий.

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла. По Боэму можно сформулировать десять наиболее распространённых (по приоритетам) рисков:

- дефицит специалистов;
- нереалистичные сроки и бюджет;
- реализация несоответствующей функциональности;
- разработка неправильного пользовательского интерфейса;
- «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей;
- непрекращающийся поток изменений;
- нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию;
- недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами;
- недостаточная производительность получаемой системы;
- «разрыв» в квалификации специалистов разных областей знаний.

Большая часть этих рисков связана с организационными и процессными аспектами взаимодействия специалистов в проектной команде. Каждый виток спирали соответствует созданию фрагмента или версии программного

обеспечения, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали.

Рис. 1.5. Спиральная модель: 1 – начальный сбор требований и планирование проекта; 2 – та же работа, но на основе рекомендаций заказчика; 3 – анализ риска на основе начальных требований; 4 – анализ риска на основе реакции заказчика; 5 – переход к комплексной системе; 6 – начальный макет системы; 7 – следующий уровень макета; 8 – сконструированная система; 9 – оценивание проекта заказчиком

Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации. Каждый виток разбит на 4 сектора:

1 – оценка и разрешение рисков,

2 – определение целей,

3 – разработка и тестирование,

4 – планирование.

На каждом витке спирали могут применяться разные модели процесса разработки ПО. В конечном итоге на выходе получается готовый продукт. Модель сочетает в себе возможности модели прототипирования и водопадной модели. Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем.

На рис.1.6 представлена усовершенствованная стратегия спиральной модели разработки программного обеспечения.

При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная задача — как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Основная проблема спирального цикла — определение момента перехода на следующий этап.

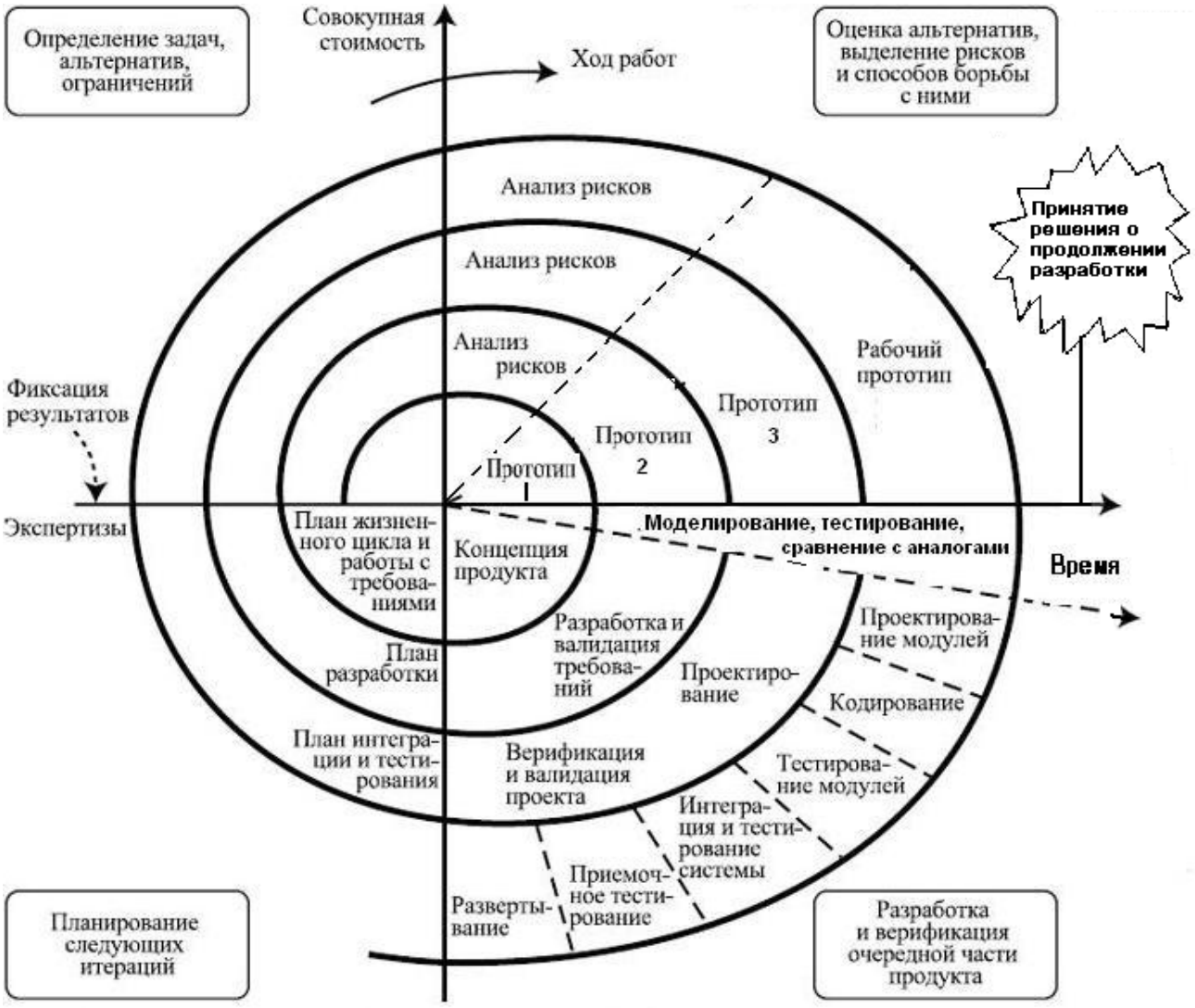


Рис. 1.6. Спиральная модель разработки программного обеспечения

Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Достоинства спиральной модели.

1. Наиболее реально (в виде эволюции) отображает разработку программного обеспечения.
2. Позволяет явно учитывать риск на каждом витке эволюции разработки;
3. Включает шаг системного подхода в итерационную структуру разработки.
4. Использует моделирование для уменьшения риска и совершенствования программного изделия.

Недостатки спиральной модели

1. Повышенные требования к заказчику.
2. Трудности контроля и управления временем разработки.

Спиральная стратегия разработки программного обеспечения близка к инкрементной стратегии, однако имеются и различия.

Модель быстрой разработки приложений (RAD модель). Одним из возможных итеративных подходов к разработке программного обеспечения в рамках спиральной модели жизненного цикла является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки программного обеспечения, содержащий 3 элемента:

- небольшую команду программистов (от 2 до 10 человек);
- короткий, но тщательно проработанный производственный график (от 2 до 6 месяцев);
- повторяющийся цикл, при котором разработчики, по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные через взаимодействие с заказчиком.

Жизненный цикл программного обеспечения по методологии RAD состоит из четырёх фаз:

- фаза определения требований и анализа;
- фаза проектирования;
- фаза реализации;
- фаза внедрения.

Применение RAD имеет и свои недостатки, и ограничения:

для больших проектов в RAD требуются существенные людские ресурсы (необходимо создать достаточное количество групп);

RAD применима только для таких приложений, которые могут декомпозироваться на отдельные модули и в которых производительность не является критической величиной;

RAD неприменима в условиях высоких технических рисков (т.е. при использовании новой технологии).

1.3. Стратегия объектно-ориентированного проектирования

В настоящее время объектно-ориентированный подход является одним из быстро развивающихся направлений в проектировании систем. Примером могут являться объектно-ориентированный анализ – методология разработки систем, предложенная Йорденом, объектно-ориентированное проектирование, объектно-ориентированное программирование, реализованное в многочисленных компиляторах C++, Object Pascal, Borland Pascal, Smalltalk.

Несмотря на различия, существующие в конкретных вариантах объектно-ориентированного подхода, все эти варианты объединяются несколькими основополагающими принципами:

- инкапсуляция – такое свойство, при котором объекты содержат описание атрибутов и действий одновременно,
- наследование – такой метод определения объектов, при котором производные объекты (потомки) наследуют свойства (атрибуты и действия) от своих родителей,
- полиморфизм – такое свойство объектов, при котором действие с одинаковыми именами вызывают различное поведение для различных объектов.

Основное преимущество объектно-ориентированного подхода состоит в том, что задача решается новым способом. Вместо формирования набора процедур, предназначенных для решения конкретной задачи, формируется набор объектов, свойственных данной предметной области. Если такой набор составляется грамотно, то не только конкретная задача может быть решена, но и потенциально закладывается фундамент для решения всех задач в данной предметной области. При структурном программировании такой подход стихийно складывался при разработке программ в виде библиотек подпрограмм по темам. Объектно-ориентированный подход дает новые механизмы, перечисленные выше (3 основные свойства объектно-ориентированного подхода), которые позволяют создавать действительно независимые от задачи описания предметной области в виде набора объектов.

Необходимо явно разделять две категории объектно-ориентированного подхода: объектно-ориентированное программирование (т.е. методика составления текстов программ на объектно-ориентированном языке) и объектно-ориентированный анализ и проектирование (т.е. создание проектов систем и описание предметной области в терминах объектно-ориентированного подхода – классов, атрибутов, методов).

В настоящее время объектно-ориентированное проектирование является наиболее прогрессивной технологией разработки программных систем. Тем не менее, объектно-ориентированный анализ, как средство описания предметной области и объектно-ориентированное проектирование как способ создания высокоуровневых проектов сегодня подвергается критике. Одним из результатов такой критики является создание итеративной модели жизненного цикла разработки программного продукта, которая является развитием спиральной стратегии. Примером итеративной модели жизненного цикла является технология разработки программного обеспечения Rational Objectory Process которая плавно перешла в технологию Rational Unified Process.

Исторически этот переход можно описать следующим образом, в Швеции в 1987 году Айваром Джейкобсоном (Ivar Jacobson) был создан процесс Objectory Process. Основанный на концепции использования прецедентов в объектно-ориентированном проектировании, этот процесс быстро получил признание в индустрии программных средств и был принят (полностью или в качестве составной части) множеством компаний по всему миру. А в 1992 году в виде учебника была издана упрощенная версия Objectory Process .

Версия Rational Objectory Process (ROP) 4.0 — это результат объединения процессов Rational Approach и Objectory Process (версия 3.8), происшедшего после слияния в 1995 году корпорации Rational Software Corporation и компании Objectory AB. От своего предка Objectory процесс унаследовал модель процесса и основополагающую концепцию прецедента. Со стороны Rational он получил нынешнюю формулировку итеративной разработки и архитектуры. Эта версия также объединила управление требованиями от корпорации Requisite, Inc. и подробный процесс тестирования от корпорации SQA, Inc., компаний, также объединившихся *с* Rational Software. И, наконец, данная версия процесса была первой, в которой использовался созданный язык UML (версия 0.8).

В 1997 году в Objectory процесс были добавлены требования и испытания дисциплины в подходе, в 1998 году добавлены две новые дисциплины: бизнес-моделирование, большая часть которой уже была в Objectory процесса, и дисциплины конфигурации и управления изменениями, а также доба-

вили новые методы тестирования производительности, UI Design, а также обновленную UML 1.1. В 1999 году они добавили дисциплины управления проектами и методы для реальной разработки программного обеспечения. Они внесли обновленный ROP для UML 1.3.

Таким образом, разработка и модернизация Rational Objectory Process (версии 4) привели к технологии Rational Unified Process (версия 5).

Rational Unified Process включает значительную часть материала из областей проектирования данных, моделирования производства, управления проектом и управления конфигурацией, причем последнее — это следствие объединения с процессом Pure-Atria. Rational Unified Process включает элементы метода Real-Time Object-Oriented Method, разработанного создателями метода ObjecTime, приобретенного корпорацией Rational Software в 2000 году. Кроме того, корпорация Rational Software для обеспечения инструментальной поддержки всех процессов жизненного цикла разработки и сопровождения программных средств RUP разработала для Rational Unified Process набор инструментов поддержки и рекомендует использование специализированных инструментальных средств IBM Rational:

– управление требованиями – IBM Rational RequisitePro;

– визуальное моделирование и генерация объектного кода – IBM Rational Rose, IBM Rational XDE;

– разработка — IBM Rational RapidDeveloper

– конфигурационное управление – IBM Rational ClearCase;

– управление изменениями – IBM Rational ClearQuest;

– автоматизированное документирование – IBM Rational SoDA;

– автоматизированное тестирование – IBM Rational TeamTest, IBM Rational TestFactory, IBM Rational Robot, IBM Rational PurifyPlus, IBM Rational SiteCheck и IBM Rational SiteLoad.

RUP использует итеративную модель разработки. В конце каждой итерации (в идеале продолжающейся от 2 до 6 недель) проектная команда должна достичь запланированных на данную итерацию целей, создать или доработать проектные артефакты и получить промежуточную, но функциональную версию конечного продукта. Итеративная разработка позволяет быстро реагировать на меняющиеся требования, обнаруживать и устранять риски на ранних стадиях проекта, а также эффективно контролировать качество создаваемого продукта.

Жизненный цикл разбит на внутренние циклы, результатом каждого из которых является собственная версия программной системы. Каждый цикл состоит из четырех фаз.

Начало (Inception)	Начальная стадия
Совершенствование (Elaboration)	Разработка
Построение (Construction)	Конструирование
Переход (Transition)	Ввод в эксплуатацию

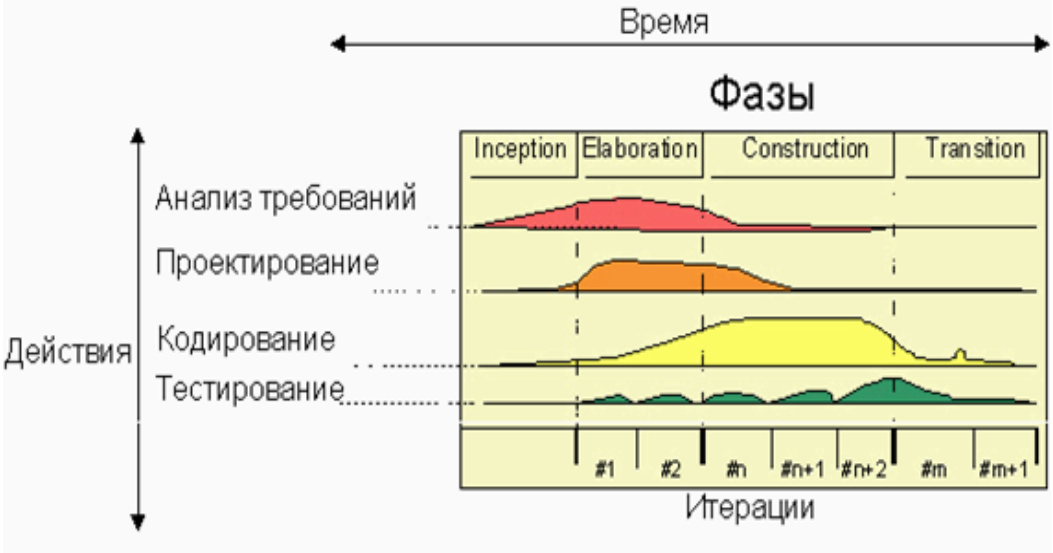


Рис. 1.7. Модель жизненного цикла UML

Динамическая структура RUP состоит из четырех фаз: Inception, Elaboration, Construction и Transition. Фазы могут подразделяться на итерации. Переход с фазы на фазу возможен только после выполнения задач фазы и представляет собой контрольную точку (milestone) процесса.

1. Начало (Inception)

На этом этапе формируются видение и границы проекта и устранение экономических рисков (в частности, именно здесь принимается решение о том, имеет ли смысл продолжать разработку).

Создается экономическое обоснование (business case), основное внимание в ней уделяется моделированию бизнес-процессов.

Определяются основные требования, ограничения и ключевая функциональность продукта. Создается базовая версия модели прецедентов.

Оцениваются риски. В этой фазе, как правило, осуществляется проектирование и реализация первого прототипа системы.

При завершении фазы ***Начало*** оценивается достижение цели жизненного цикла Lifecycle Objective Milestone, которое предполагает соглашение заинтересованных сторон о продолжении проекта.

2. Проектирование (Elaboration)

Эта фаза посвящена тщательной проработке требований и выбору основных проектных решений. В этой фазе концептуальный прототип превращается в реальную систему, позволяющую протестировать и оценить выбранные архитектурные решения. В результате к концу фазы команда готова к быстрой и качественной разработке основного объема кода системы. На этапе ***Проектирование*** проводится анализ предметной области и построение исполняемой архитектуры. Это включает в себя: документирование требований (включая детальное описание для большинства прецедентов использования); спроектированную, реализованную и оттестированную исполняемую архитектуру. Наличие тщательно проработанной устойчивой архитектуры гарантирует, что в дальнейшем не придется перерабатывать большие фрагменты системы; обновленное экономическое обоснование и более точные оценки сроков и стоимости; сниженные основные риски.

Успешное выполнение фазы ***Проектирование*** означает достижение вехи архитектуры жизненного цикла (Lifecycle Architecture Milestone).

3.Построение (Construction)

В фазе Construction основной задачей становится быстрая и экономичная разработка кода системы. К концу фазы система должна быть готова к передаче заказчику для бета-тестирования и/или приемо-сдаточных испытаний. Во время этой фазы происходит реализация большей части функциональности продукта. Фаза ***Построение*** завершается первым внешним релизом системы и вехой начальной функциональной готовности (Initial Operational Capability).

4. Внедрение (Transition)

Во время фазы ***Внедрение*** создается финальная версия продукта и передается от разработчика к заказчику. Это включает в себя программу бета-тестирования, обучение пользователей, а также определение качества продукта. В случае, если качество не соответствует ожиданиям пользователей или критериям, установленным в фазе Начало, фаза ***Внедрение*** повторяется снова. Выполнение всех целей означает достижение вехи готового продукта (Product Release) и завершение полного цикла разработки.

В отличие от каскадного подхода («водопада»), в RUP все дисциплины выполняются практически во всех фазах жизненного цикла ПС. Однако, в зависимости от фазы, меняются текущие цели проекта и соотношение между объемами работ, соответствующих различным дисциплинам.

Статическая структура RUP состоит из дисциплин, в которые группируются работы, задачи, артефакты и роли. Для описания осмысленной последовательности выполнения работ и задач используются рабочие процес-

сы (рис.1.8). Они описывают кто, что, как и когда выполняет в процессе. В RUP входят 6 основных дисциплин:

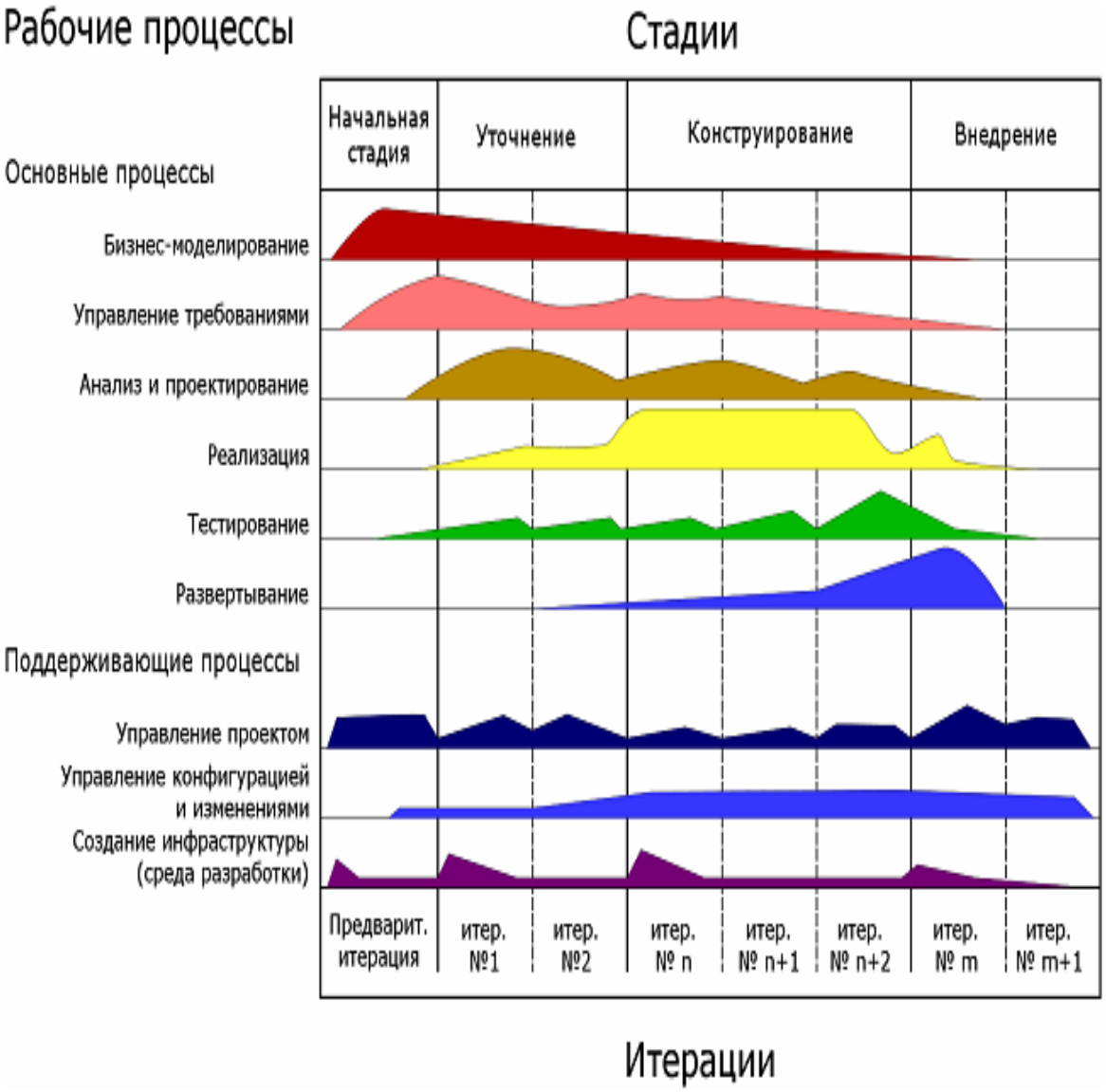


Рис. 1.8. Графическое представление процесса разработки по RUP

– бизнес-моделирование (Business modeling);
– управление требованиями (Requirements);
– анализ и Проектирование (Analysis and Design);
– реализация (Implementation);
– тестирование (Test);
– развертывание (Deployment).

И три вспомогательные:

– управление проектом (Project management);
– управление изменениями (Change management);

– среда (Environment).

Таким образом, Rational Unified Process поддерживает объектно-ориентированную технологию. Моделирование по методологии RUP является объектно-ориентированным и базируется на понятиях объектов, классов и зависимостей между ними. Эти модели, подобно многим другим техническим искусственным объектам (артефактам), в качестве единого стандарта для организации взаимодействия участников проекта используют Unified Modelling Language™ (UML) — универсальный язык моделирования.

Rational Unified Process поддерживает компонентно-ориентированный подход. Компоненты — это нетривиальные модули или подсистемы, которые выполняют конкретную функцию и могут быть использованы многократно. Как правило, компоненты соответствуют одной из промышленных спецификаций, таких как CORBA, COM/DCOM, ActiveX, Enterprise Java Beans и др.

1.3. Инструментальные средства проектирования

Практически любая современная крупная программная система разрабатывается с применением CASE-технологий по крайней мере на этапах анализа и моделирования, что связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ.

Целью систем анализа и проектирования является определение системных требований и свойств, создание проекта и архитектуры информационной системы, а также детальная «калька» проекта, включающая алгоритмы и определения структур данных.

Системы проектирования баз данных обеспечивают логическое моделирование данных, автоматическое преобразование моделей данных в Третью Нормальную Форму, автоматическую генерацию схем База данных и описаний форматов файлов на уровне программного кода.

В таблице 1.1 приведены некоторые CASE- средства анализа и проектирования программных систем.

Таблица 1.1

Средства проектирования

Система проектирования	Производитель	Нотация DFD	Поддержка	
			Поддержка	Поддержка диаграмм

	ПО		методологии	
Системы анализа и проектирования				
BPWin	Logic Works	Гейн-Сарсон	IDEF0/SADT IDEF3/SADT	Функциональной декомпозиции
ProKit *Workbench	MDIS	Гейн-Сарсон	Собственная STRADIS	Потоков данных в нотации Гейна-Сарсона, сущность-связь, структурные карты Константайна
CASE. Аналитик	Эйтекс	Гейн-Сарсон	структурного системного анализа Гейн-Сарсона	функциональной декомпозиции, потоков данных, управляющих потоков, структуры данных, сущность-связь
CASE /4/0	MicroTOOL	Йодан (расшир.)	анализа и проектирования систем реального времени Уорда-Меллера	Функциональной декомпозиции, потоков данных, переходов состояний, карты Джексона
Design/IDEF	Meta Software	-	IDEF0, IDEF1 IDEF1X, IDEF/CPN	сущность-связь
Visible Analyst Workbench	Visible Systems	Гейн-Сарсон, Йодан	Информационного моделирования Мартина	Функциональной декомпозиции, сущность-связь, потоков данных в нотации Йодана и Гейна-Сарсона, структурные карты Константайна
Rational Rose	Rational Software	UML, также поддерживается нотация Буча и OMT-2.	Собственная методология "Rational Objectory Process", UML	вариантов использования, взаимодействия объектов, последовательности взаимодействий, переходов состояний, классов, пакетов, компонентов, размещения
UML 2.x	Object Management Group	Собственная	UML	Классов, компонентов, композитной/ составной структуры, кооперации (UML 2.0), объектов, пакетов, профилей (UML 2.2), деятельности, состояний, прецедентов, коммуникации (UML 2.0) / кооперации (UML 1.x), обзора взаимодействия (UML 2.0), последовательности, синхронизации (UML 2.0)
ARIS Toolset	Software AG	Нотация ARIS eEPC	UML	сущность-связь, описания процессов, информационных потоков
Системы проектирования баз данных и файлов				
Designer/2000	Oracle	Гейн-Сарсон	Собственная CASE*Method	сущность-связь, потоков данных, иерархии функций, взаимодействия модулей структурные карты Джексона

ERWin	Logic Works	Гейн-Сарсон	IDEF3/SADT	сущность-связь
EasyCASE	Evergreen CASE Tools	Гейн-Сарсон, Йодан	структурного системного анализа Гейн-Сарсон	сущность-связь, потоков данных, переходов состояний, структурные карты в нотации Константайна
Vantage Team Builer	CAY-ENNE	Йодан	Структурного анализа и проектирования Чена и Йодана,	сущность-связь в нотации Чена, потоков данных в нотации Йодана, переходов состояний, структурные карты Константайна
Chen Toolkit	Chen & Asssociates	Чена	Чена	сущность-связь, потоков данных, переходов состояний
S-Designor	Sybase/ Powersoft		Информационного моделирования	сущность-связь
SILVERRUN	Computer Systems Advisers	произвольная	DATARUN	потоков данных BPM, сущность-связь ERX и RDM

Эти системы поддерживают те или иные методологии анализа и проектирования информационных систем, которые определяют руководящие указания для оценки и выбора проекта программного продукта, последовательность шагов выполнения работы, правила распределения и назначения операций и методов.

Кроме перечисленных систем проектирования разработаны различные инструментальные CASE–средства: разработки приложений – JAM (JYACC), PowerBuilder (Sybase), New Era (Informix), SQL Windows (Gupta) и др.; планирования и управления проектом – Microsoft Project, SE Companion; тестирования – Quality Works (Segue Software); документирования – SoDA (Rational Software), которые позволяют автоматизировать процессы проектирования, планирования, разработки, тестирования, документирования информационной системы моделирования.

Инструментальные средства, предназначенные для проектирования и моделирования информационных систем, могут быть отнесены к одной из следующих категорий:

- локальные, поддерживающие один-два типа моделей и методов (Design/IDEF, ProCap, S-Designor, “CASE. Аналитик”);
- малые интегрированные средства моделирования, поддерживающие несколько типов моделей и методов (ERwin, BPwin) [8, 41, 45] ;

- средние интегрированные средства моделирования, поддерживающие от 4 до 10—15 типов моделей и методов (Rational Rose, Paradigm Plus, Designer/2000) [46];
- крупные интегрированные средства моделирования, поддерживающие более 20 типов моделей и методов (ARIS Toolset).

Остановимся на малых интегрированных средствах проектирования информационных систем. Типичный представитель малых интегрированных средств моделирования – комплект программных продуктов Platinum Technology (CA/ Platinum/Logic Works), основанный на популярных пакетах BPwin и Erwin [8, 41, 46, 48].

BPWin. Компания LogicWorks, разработчик BPwin, сейчас входящий в Computer Associates, работает на рынке технологий моделирования уже более 20 лет. Для проведения анализа и реорганизации сложных систем и процессов Logic Works предлагает CASE-средство верхнего уровня – BPwin, который поддерживает 3 методологии:

- IDEF0 (функциональная модель),
- IDEF3 (WorkFlow Diagram) – только диаграммы процессов,
- DFD (DataFlow Diagram) – диаграммы потоков данных.

Функциональная модель предназначена для описания существующих систем и процессов (так называемая модель AS-IS) и идеального положения вещей – того, к чему нужно стремиться (модель TO-BE).

Интеграция выполняется как путем слияния нескольких моделей, так и посредством переключения на различные методологии в процессе разработки отдельных диаграмм информационной модели. Предусмотрено расширение возможностей анализа систем как в самом пакете BPwin (функционально-стоимостный анализ), так и с помощью экспорта данных в другие пакеты. BPwin автоматизирует задачи, связанные с построением моделей развития, обеспечивая семантическую строгость, необходимую для гарантирования правильности и непротиворечивости результатов.

ERwin. Поддерживает несколько разновидностей методологии информационного моделирования, основанной на ER-диаграммах (сущность – связь). Интеграция моделей BPwin с моделями ERwin выполняется путем обмена данными через функции экспорта/импорта.

Основной из трех методологий является IDEF0, она относится к семейству IDEF, которое было введено в 1973 году Россом под названием SADT (Structured Analysis and Design Technique). Хотя основной акцент использования малых интегрированных систем делается в применении к бизнес-

процессам на предприятиях, эти технологии применимы для декомпозиции и проектирования широкого класса систем, в том числе информационных систем, в частности для проектирования сайтов, порталов, систем и программных комплексов компьютерных установок для моделирования реальных объектов. Для информационных систем применение IDEF0 имеет своей целью определение требований и указание функций для последующей разработки системы, отвечающей поставленным требованиям и реализующей выделенные функции. Применительно к уже существующим системам IDEF0 может быть использована для анализа функций, выполняемых системой, и отображения механизмов, посредством которых эти функции выполняются.

Первая диаграмма в иерархии диаграмм IDEF0 всегда изображает функционирование информационной системы в целом (рис.1.9). Такая диаграмма называется контекстной. В контекст входит описание цели моделирования, области (описания того, что будет рассматриваться как компонент системы, а что как внешнее воздействие) и точки зрения (позиции, с которой будет строиться модель).

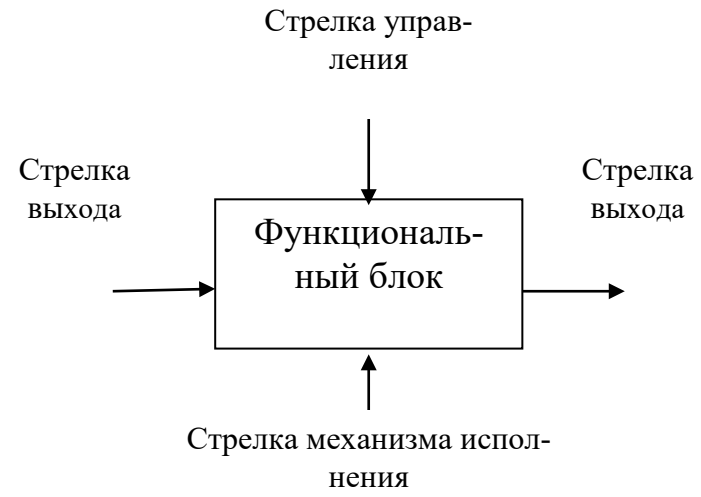


Рис. 1.9. Соединение стрелок со сторонами функционального блока

На рис.1.10 приведен пример проектирования этапов прикладной программы «Пастбище», предназначенной для моделирования пастбищной нагрузки при наличии мелкого и крупного рогатого скота.

На рис. 1.11 приведено оглавление диаграммы IDEF 0-3 для программы «Пастбище».

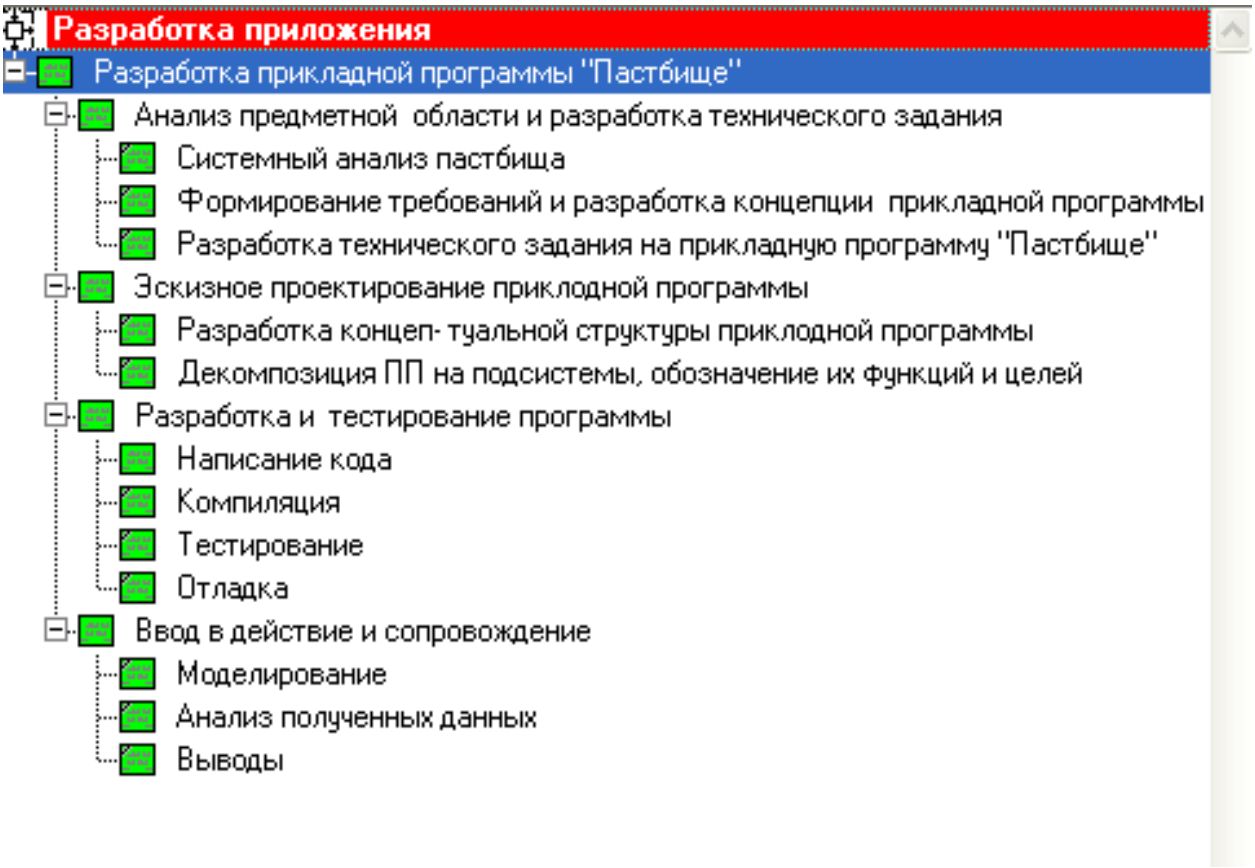


Рис.1.10. Проектирование этапов разработки программы

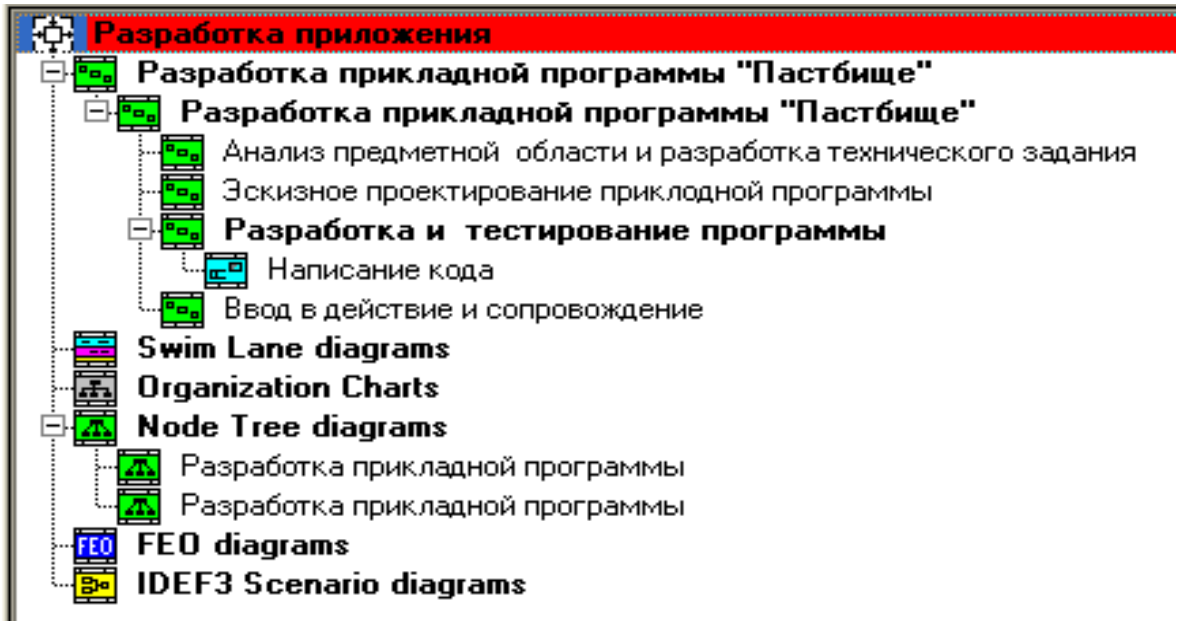


Рис.1.11. Оглавление диаграмм IDEF0 для программы «Пастбище»

На рис. 1.12 представлен пример контекстной модели, разработанной при проектировании программного компьютерного комплекса «Пастбище».

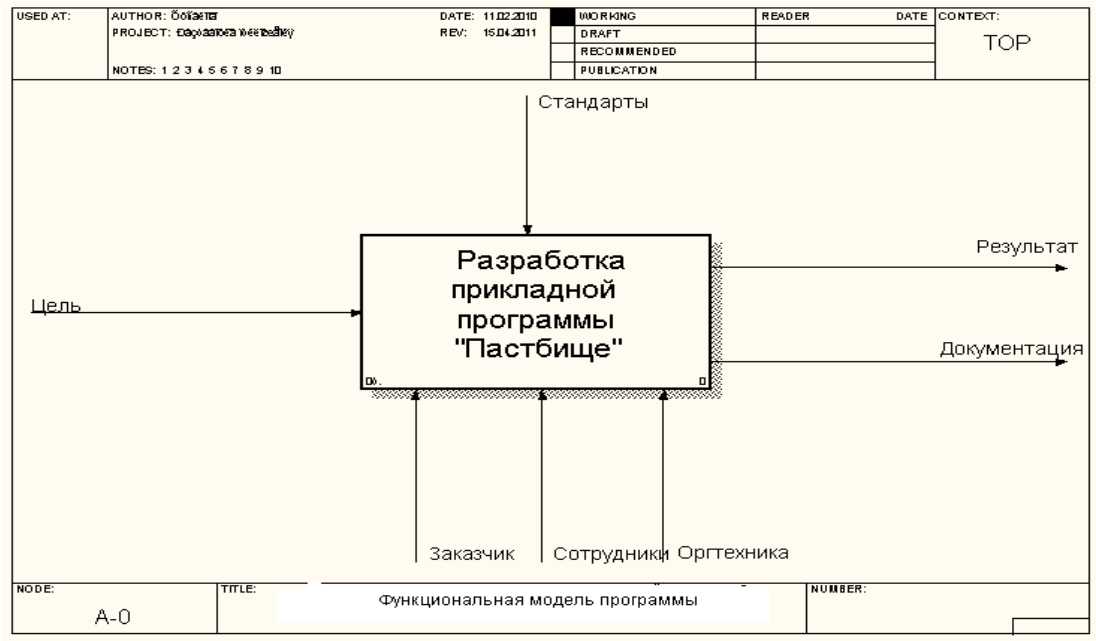


Рис.1.12. Функциональная модель первого уровня или контекстная модель «Пастбище»

Результатом применения IDEF0 к некоторой системе являются функциональные модели этой системы, состоящие из иерархически упорядоченных в несколько уровней диаграмм, связанных друг с другом с помощью ссылок.

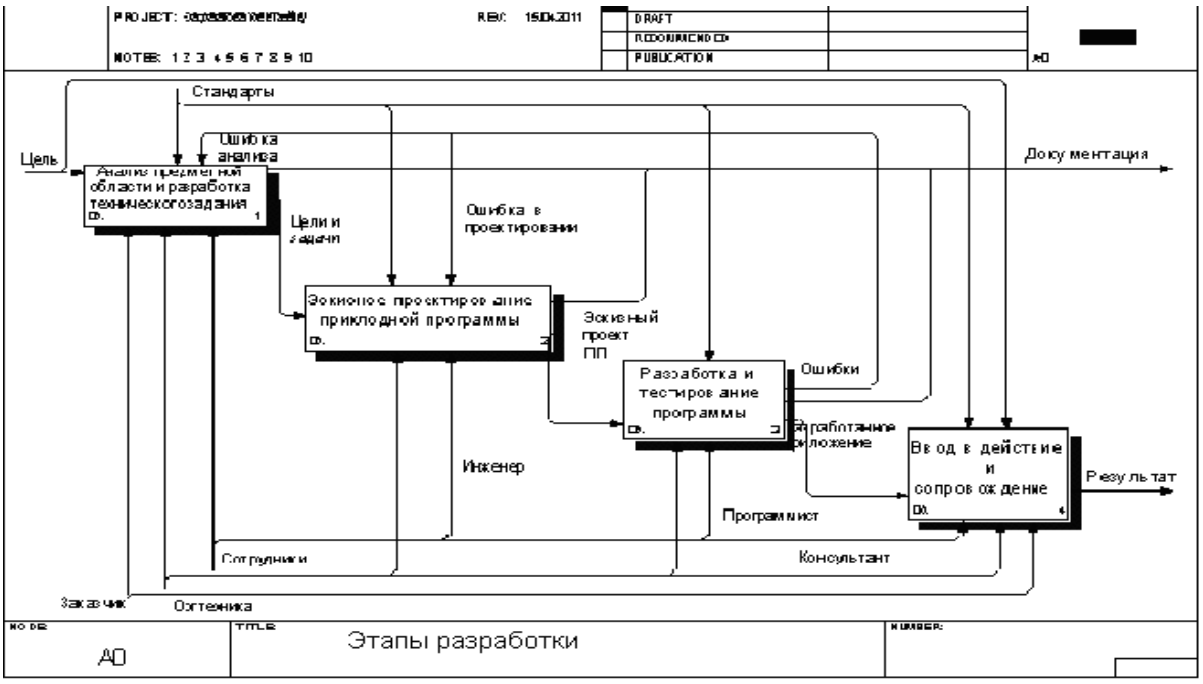


Рис.1.13. Функциональная модель второго уровня по этапам разработки программы

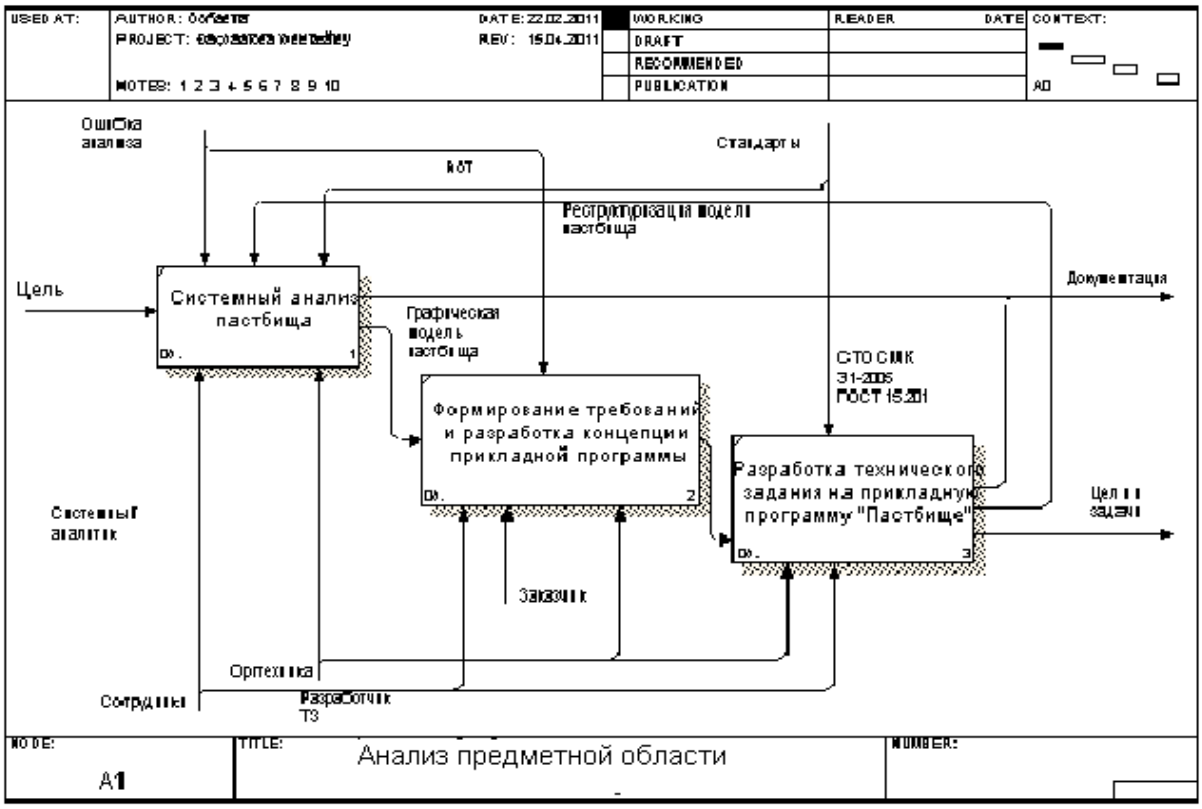


Рис.1.14. Функциональная модель третьего уровня по этапам разработки программы

Двумя наиболее важными компонентами, из которых строятся диаграммы IDEF0, являются работы (представленные на диаграммах в виде прямоугольников), данные и объекты (изображаемые в виде стрелок), связывающие между собой работы.

Недостатком Bpwin является незавершенность проработки интерфейса пользователя, что может осложнять работу проектировщика.

Созданию имитационных моделей всегда предшествует стадия проектирования программных продуктов. Для создания проектов имитационных моделей, особенно сложных систем, существуют достаточное количество CASE-средств [8, 41, 45, 46, 48]. Некоторые из них представлены в таблице 1.1.

При разработке имитационных моделей всегда стоит вопрос, какая часть разработки является наиболее трудоемкой и куда будут направлены основные усилия? Анализ и опыт разработок проектов показывает, что основная часть интеллектуальных усилий уходит:

- на понимание бизнеса клиента и сути и системного анализа поставленной задачи;
- на выбор информационной архитектуры решения;
- на разработку оптимизационных эвристик.

Создание собственно имитационной модели – ключевая, но, как правило, не самая большая часть проекта. Большая трудоемкость по времени может уйти на разработку интерфейса пользователя и интеграции решения с базами данных и т.п. Обработка существующих данных также занимает значительное время.

Контрольные вопросы

1. Какие российские стандарты существуют для разработки программного обеспечения?
2. Какие международные стандарты существуют для разработки программного обеспечения?
3. Каковы основные стадии разработки программного обеспечения по ГОСТ 34
4. Каковы основные цели предпроектной стадии разработки ПО?
5. На какие этапы подразделяется предпроектная стадия?
6. Что такое техническое задание на разработку ПО?
7. На сколько этапов можно подразделить проектную стадию разработки ПО?
8. В чем суть стадии внедрения?
9. Что такое период сопровождения?
10. В чем идея водопадной стратегии разработки ПО?
11. Каковы отличительные особенности инкрементной и эволюционной стратегии разработки ПО?
12. В чем суть итеративной модели жизненного цикла программного обеспечения?
13. Какие CASE - средства используются для анализа и проектирования программного обеспечения!
14. Какие системы проектирования баз данных и файлов востребованы на сегодняшний день?

ГЛАВА 2. ОСНОВНЫЕ НАПРАВЛЕНИЯ РАЗРАБОТКИ ПРОГРАММ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

Математическое моделирование для исследования характеристик систем условно подразделяется на аналитическое, имитационное и кибернетическое. В свою очередь, модели можно подразделить на классы аналитические, имитационные и схемотехнические (кибернетические).

Для **аналитического моделирования** характерно то, что процессы функционирования системы записываются в виде некоторых функциональных соотношений (алгебраических, дифференциальных, интегральных уравнений) [1, 8, 15, 28-31, 34, 38-41]. Эти соотношения называются аналитическими моделями. Аналитическая модель может быть исследована следующими методами:

1) аналитическим, если стремятся получить в общем виде явные зависимости для характеристик систем;

2) численным, если не удается найти решение уравнений в общем виде и их решают для конкретных начальных данных;

3) качественным, если при отсутствии решения находят некоторые его свойства.

При **имитационном моделировании** реализующий модель-алгоритм воспроизводит процесс функционирования системы во времени. Имитируются элементарные явления, составляющие процесс, с сохранением их логической структуры и последовательности протекания во времени. Основным преимуществом имитационных моделей по сравнению с аналитическими является возможность решения задач исключительной сложности. Метод имитационного моделирования обеспечивает имитацию любых сложных и многообразных процессов с большим количеством элементов, отдельные функциональные зависимости в таких моделях могут описываться громоздкими математическими соотношениями.

Имитационные модели позволяют легко учитывать наличие дискретных или непрерывных элементов, нелинейные характеристики, случайные воздействия и др. Поэтому этот метод широко применяется на этапе проектирования сложных систем.

Имитационное моделирование находит широкое применение в различных сферах, в социальных, экономических, политологических, технических и военных исследованиях:

– моделирование производственных систем;

– моделирование логистических процессов;

– в маркетинге;

– моделирование бизнес-процессов;

– моделирование экономических реформ;

– в социологии и политологии;

– моделирование социальных процессов и институтов;

– моделирование транспортных систем;

– моделирование технических, информационных и телекоммуникационных процессов и систем;

– моделирование различных глобальных мировых процессов.

Очень сложно перечислить все сферы применения имитационного моделирования.

Под **схемотехническим (кибернетическим) или аналоговым моделированием** понимаем моделирование сложных систем на основе информационного обмена между ее элементами, это класс моделей, который позволяет отразить управленческие аспекты поведения систем. Особенностью кибернетического моделирования является, кроме учета механизмов управления, возможность учета механизмов самоорганизации, обучения, адаптации и т.д. Этот подход широко применяется при построении моделей систем автоматического управления [5, 10, 31].

Под компьютерным моделированием понимают математическое моделирование с использованием средств вычислительной техники. С развитием вычислительной техники, на качественно новую ступень поднялась технология моделирования сложных систем и процессов, такая революция стала возможной в результате разработки методологии имитационного моделирования на ЭВМ и с появлением компьютерных систем аналитического и имитационного моделирования нового поколения.

Целью математического моделирования является анализ реальных объектов и процессов (в природе или технике) математическими и компьютерными методами. В свою очередь, это требует формализации математической модели процесса, подлежащего исследованию. Модель может представлять собой математическое выражение, содержащее переменные, поведение которых аналогично поведению реальной системы. Модель может включать элементы случайности, учитывающие вероятности возможных действий двух или большего числа «игроков», как, например, в теории игр; либо она может представлять реальные переменные параметры взаимосвязанных частей действующей системы. Таким образом, в результате математического моделирования реальный объект заменяется его аналитической или имитационной моделью в целях изучения закономерностей и прогнози-

рования поведения объекта с помощью компьютерных технологий и моделей.

Ниже в таблице 2.1 приведены этапы аналитического и имитационного моделирования.

Таблица 2.1

Этапы моделирования

№	Этапы аналитического моделирования	Этапы имитационного Моделирования
1	Построение математической модели	Разработка концептуальной модели (декомпозиция, структуризация связей). Подготовка исходных данных.
2	Выбор метода решения и построение алгоритма моделирования	Выбор средств моделирования
3	Разработка компьютерной вычислительной установки. Проверка адекватности и корректировка модели.	Разработка имитационной модели. Проверка адекватности и корректировка модели.
4	Планирование машинных экспериментов. Компьютерное исследование или вычислительный эксперимент	Планирование машинных экспериментов. Моделирование ("прогоны").
5	Обработка и анализ результатов вычислительного эксперимента	Анализ и визуализация результатов моделирования.

Более подробно этапы аналитического моделирования расписаны в книге [29-30], а этапы имитационного моделирования в [35-36]. На сегодняшний день математическое моделирование является одним из важнейших технологий исследования реальных объектов и процессов.

Анализ и проведенный обзор разработанных компьютерных моделей и систем моделирования позволяет выделить основные направления разработки программного обеспечения в отрасли математического и компьютерного моделирования:

1) разработка интерактивных компьютерных моделей имитации реальных объектов для обучающих электронных систем, предназначенных для обучения учащихся (школьников, студентов и др.), в частности для изучения объектов, их свойств и закономерностей в различных отраслях знаний – физике, биологии, химии и т.д.;
2) разработка сетевых виртуальных лабораторий по компьютерному моделированию, компьютерных систем и программ моделирования реальных объектов;

3) разработка программ моделирования и компьютерных комплексов для научных исследований в отраслях знаний – информатика, техника, физика, астрофизика, химия, биология, экология и т.д.;
4) создание инструментария, систем и средств автоматизации компьютерного моделирования для разработки математических и имитационных моделей;
5) разработка компьютерных имитационных моделей сложных систем, имеющих прикладное значение: для производства, экономики, сферы обслуживания, социальной сферы, экосистем, для различных отраслей медицины, образования, науки и др.;
6) разработка кибернетических (аналоговых) компьютерных моделей технических систем, это направление ориентировано на разработку моделей сложных систем управления в технике [4-7, 10, 16-18, 21, 25-27, 31, 32, 37].

2.1. Интерактивные компьютерные модели для электронных обучающих систем

Внешний интерфейс программного обеспечения для пользователя является одним из важных модулей любого компьютерного комплекса для моделирования и является одним из направлений разработки. Внешний интерес зависит от направлений и целей разработки компьютерных систем и программ компьютерного моделирования.

Рассмотрим некоторые характерные особенности разработки интерфейса по направлению разработки обучающих компьютерных систем с интерактивными программами компьютерного моделирования реальных объектов, т.е. предназначенные для системы образования.

Разработками таких компьютерных систем на российском рынке занимаются компьютерные фирмы, в частности: фирмы «Физикон», «1С-предприятие», «Кирилл и Мефодий», которые на профессиональном уровне разрабатывают электронные обучающие системы по физике, математике, по естественнонаучным и гуманитарным дисциплинам; ученые университетов России – разработчики уникальных компьютерных программ с интерактивными виртуальными моделями для моделирования реальных объектов. С программными продуктами этих фирм можно ознакомиться на сайтах:

http://www.physicon.ru/

http://obr.1c.ru/product.jsp?id=831

Интерфейс электронных обучающих систем и процессов моделирования реальных объектов разрабатывается чаще всего с помощью Web-технологий. Рассмотрим несколько примеров пользовательского интерфейса.

Пример 1. Рассмотрим электронный курс «Открытая Физика 2.5», разработанный для учащихся школ, лицеев, гимназий, колледжей, для абитуриентов, готовящихся к поступлению в вуз, студентов первых курсов технических и педагогических вузов и для самостоятельного изучения физики.

На рис. 2.1 приведены интерфейс и дизайн элемента обучающей системы «Открытая Физика 2.5» для раздела «Механика».

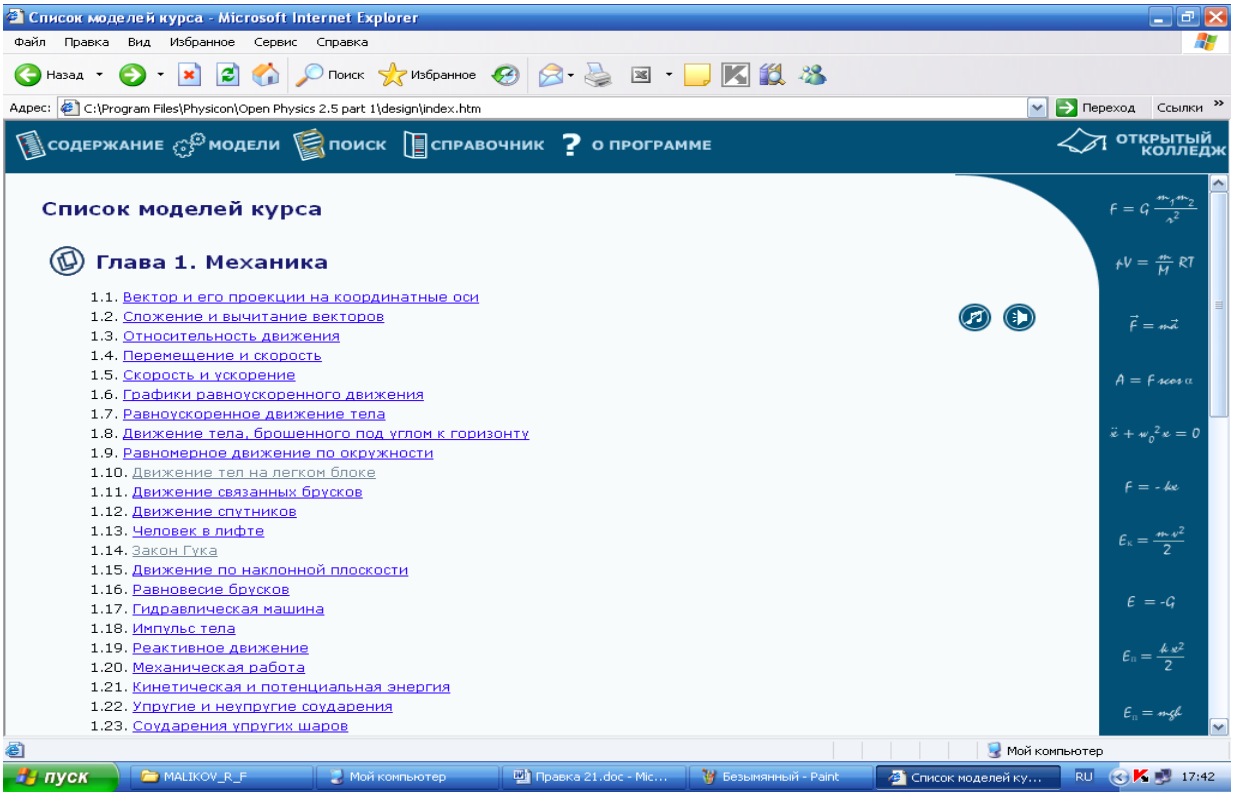


Рис.2.1. Интерфейс меню «Модели»

При разработке обучающих систем меню пользователя оформляется стандартно в виде вертикальной или горизонтальной линейки. В рассматриваемом электронном пособии приведена горизонтальная линейка меню пользователя, на котором отдельно выделены: справочная система, информация о программе, электронный учебник и меню работы с интерактивными компьютерными моделями выделены в отдельные меню; имеется также поисковая система по электронному пособию. Для раздела «Модели» приведен перечень интерактивных компьютерных моделей, к которым можно перейти по гиперссылке, и далее по данной компьютерной вычислительной установке провести исследования закономерностей реального объекта.

Дизайн экранного окна ввода исходных параметров модели и вывода результатов моделирования в данном пособии разработано по технологии совмещения (объединения) в одно экранное окно (рис.2.2). Это совмещение позволяет наглядно и мгновенно видеть результаты моделирования.

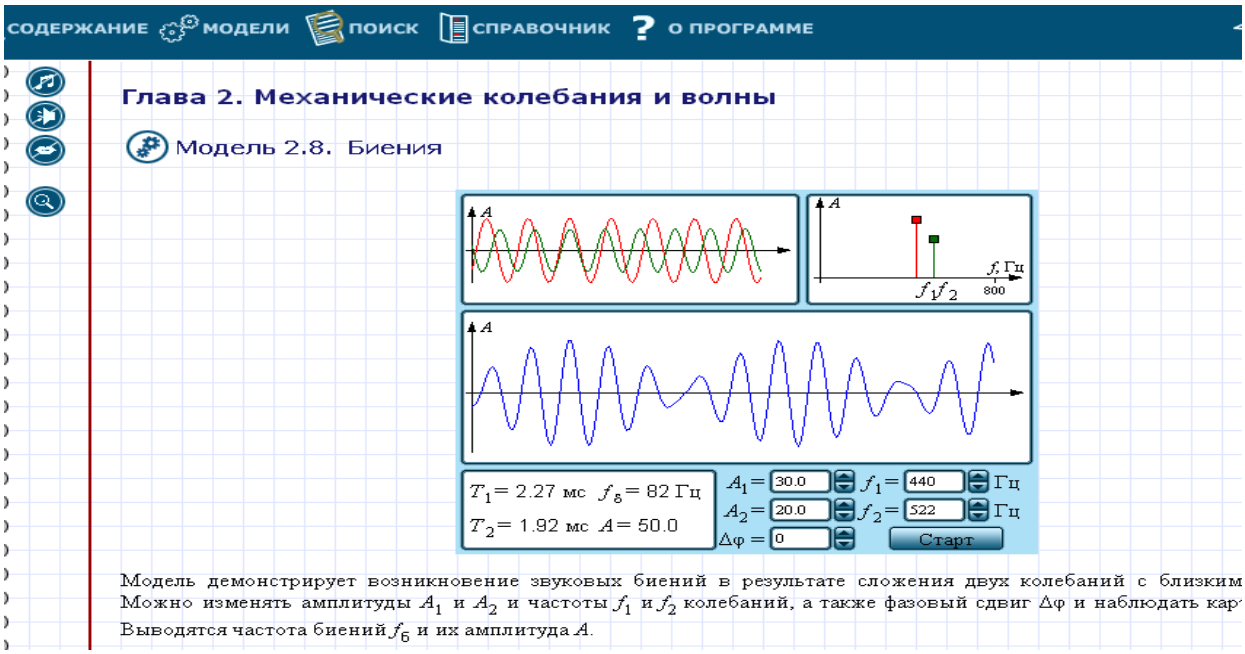


Рис. 2.2. Интерфейс экранного окна компьютерной вычислительной установки для изучения биений связанных маятников

Пример 2. Мультимедийный курс «Виртуальная лаборатория по общей физике» разработан в Томском государственном университете под руководством М.Толстика и представляет собой сборник компьютерных лабораторных работ-тренажёров по физике. Курс предназначен для студентов первого и второго курсов различных специальностей и школьников старших классах (рис.2.3).

Цели лабораторного практикума – углублённое изучение теоретического материала, знакомство с методиками измерения различных величин, привитие студентам навыков экспериментальной работы, изучение приборов, обучение сборке электрических схем и т.д.

Сборник содержит лабораторные работы по четырём разделам: механика, молекулярная физика, электричество и магнетизм, оптика.

Данный программный продукт имеет дружественный интерфейс, прост в использовании, снабжен подробным руководством пользователя и описанием программы. В данной учебной работе реализуются принципы наглядности, интерактивности.

Манипулятивный способ воздействия на компьютерные модели и задания данных делают уровень интерактивности весьма высоким, а виртуаль-

ные лабораторные эксперименты проходят максимально приближено к их реальным прототипам.

Рис. 2.3. Внешний и внутренний интерфейс мультимедийного курса «Виртуальная лаборатория по общей физике»

После проведения виртуального лабораторного эксперимента пользователь получает выходные данные на экране компьютера без возможности сохранить или редактировать их.

Пример 2. Лабораторный практикум по физике с использованием виртуальных приборов.

Кафедрой экспериментальной физики СПбГТУ разработан программный комплекс виртуального лабораторного практикума по всем разделам вузовского курса физики. Автор разработки – доцент В.П. Маслов. Интерфейс пользователя и программы, входящие в комплекс, выполнены в среде графического программирования LabVIEW, которая является универсальной системой программирования, ориентированной на решение задач управления инструментальными средствами измерения, сбора, обработки и представления экспериментальных данных.

Программы предусматривают наличие всех измерительных приборов, входящих в реальную экспериментальную установку. Все графики, необходимые при обработке результатов эксперимента, выводятся на экран монитора. Каждая программа содержит электронное описание теории исследуемого явления, реальной экспериментальной установки, методики проведения эксперимента и обработки результатов. Такого рода лабораторные работы могут быть использованы и для предварительной подготовки студентов.

Один из примеров передней панели виртуальной лабораторной работы представлен на рис.2.4.

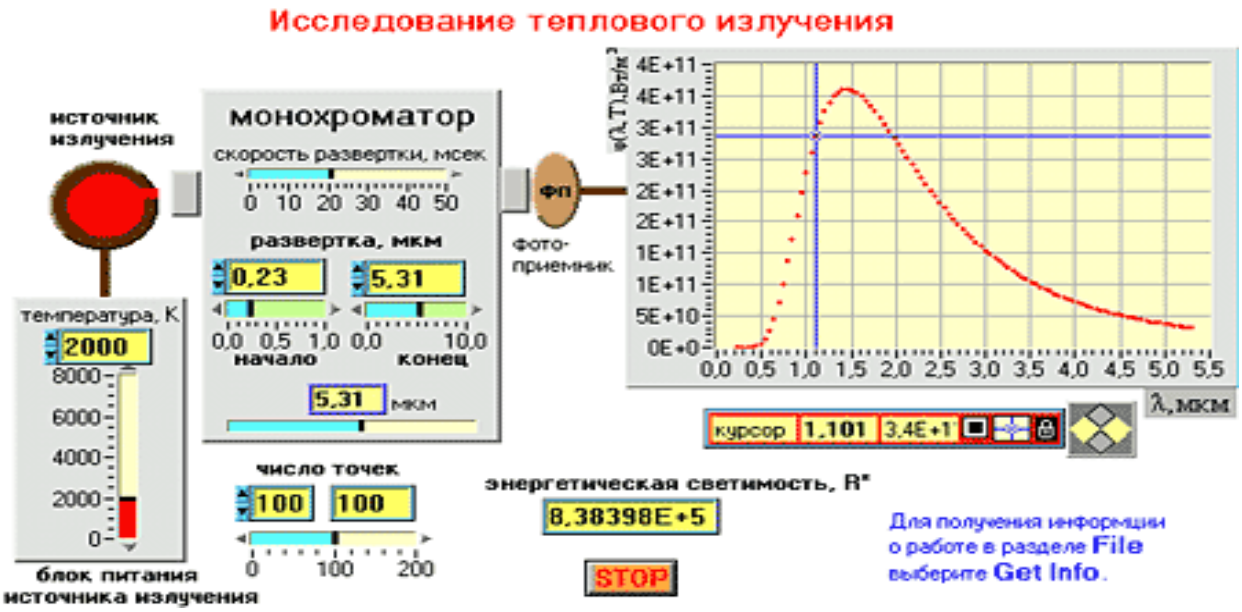


Рис. 2.4. Передняя панель виртуальной лабораторной работы «Исследование теплового излучения»

2.2. Сетевые компьютерные системы и комплексы для моделирования реальных объектов

В настоящее время развивается направление создания виртуальных компьютерных лабораторий, позволяющих проводить компьютерный эксперимент в дистанционном режиме. Например, интерактивное средство анализа и обработки данных солнечно-земной физики на основе MATLAB Web Server было создано в ИЗМИРАН в 2004 году в рамках проекта "Интерактивный WEB-ресурс для адаптивной обработки экспериментальных данных на основе инструментального средства MATLAB Web Server". Разработка и использование MATLAB Web сервера для расчетов на удаленном сервере для школьников, студентов и преподавателей естественных наук позволяет проводить интерактивные компьютерные эксперименты с помощью виртуальной лаборатории для организации дистанционного образования в области физики, химии, математики.

При использовании MATLAB Web Server технологии, для работы с лабораторией не требуется, чтобы пакет MATLAB был установлен на компьютере: лаборатория работает в дистанционном режиме, используя MATLAB Web-Server. При этом достаточно иметь доступ к компьютеру, подключенного к сети Интернет.

Например: http://mathmod.aspu.ru/?id=2

На основе технологии «клиент-сервер» и технологий Java были созданы виртуальные лаборатории по математическому моделированию в естественных науках, в частности:

- нелинейные процессы (MATLAB Web Server + VSSci)
- нелинейные процессы (Java Applets)
- базовые модели
- виртуальная лаборатория для моделирования живых систем
- виртуальная лаборатория для моделирования физических процессов
- физика и химия материалов и покрытий

В целом это комплекс программ, предоставляющий пользователям для исследования целый набор классических математических моделей физики, химии, биологии.

В компьютерном практикуме одной из виртуальных лабораторий представлено несколько крупных разделов: "Дифференциальные модели", "Отображения", "Клеточные автоматы", "Модели фазовых переходов", "Геометрические модели", "Дифференциальные модели с запаздыванием", "Рас-

пределенные системы". Во втором разделе рассматриваются как одномерные, так и двумерные отображения.

Виртуальная лаборатория состоит из нескольких частей, каждая из которых представляет собой целый комплекс разнообразных моделей. Набор моделей в основном соответствует учебному пособию [38].

Каждая модель – это отдельная электронная страница. Все модели имеют удобный графический интерфейс с различными элементами управления, с помощью которых пользователь имеет возможность изменять параметры моделей, что позволяет анализировать её поведение при различных условиях.

Первая часть лаборатории – «Дифференциальные модели». Пользователь имеет возможность изучить ряд базовых моделей физики, химии, биологии:

- Нелинейный маятник
- Модифицированная модель Вольтерры-Лотки (модель хищник-жертва)
- Модель межвидовой конкуренции
- Модель Холлинга-Тэннера
- Модель Лоренца
- Модель Ресслера
- Генератор Ван-дер-Поля
- Тримолекулярная модель (брюсселятор) и др.

При построении модели пользователь может не только задавать различные коэффициенты дифференциального уравнения, описывающие данную модель, но и начальные условия, непосредственно выбирая их на координатной плоскости. При этом фазовый портрет изображается вместе с особыми линиями, а также указывается тип особых точек

Другим средством разработки программ математического моделирования реальных объектов и процессов является стремительно развивающаяся Java-технология. Программы, созданные с использованием именно этой технологии, являются платформеннонезависимыми и Интернет-ориентированными, что уже является большим подспорьем для использования именно данной технологии в рамках дистанционного образования. С помощью JAVA-апплетов разработаны следующие компьютерные модели для виртуальной лаборатории:

- динамический хаос в простой механической системе;
- параметрические колебания маятника;
- механическая система.

На рис. 2.5 приведен интерфейс виртуальной установки исследования колебаний физического маятника с затуханием.

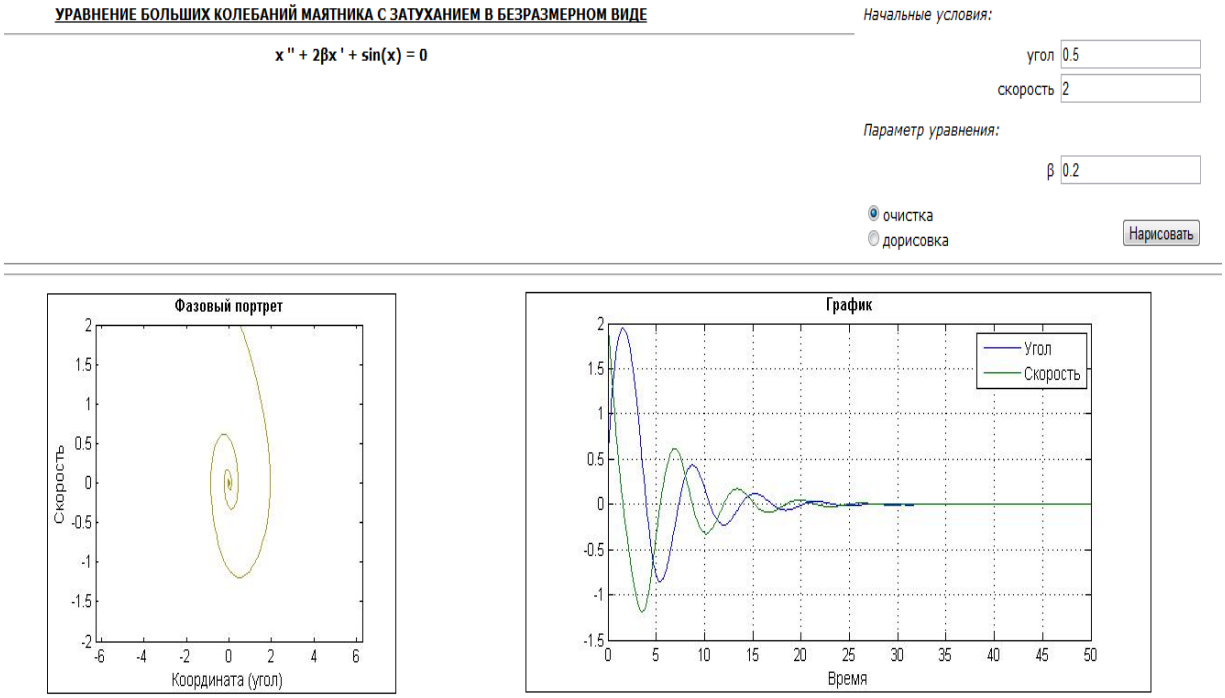


Рис.2.5. Интерфейс виртуальной установки для изучения колебаний физического маятника с затуханием

Пример 2. Рассмотрим другой пример создания виртуальной лаборатории с удаленным доступом с возможностями компьютерного моделирования на примере лаборатории с удаленным доступом «Физические основы электроники» (рис.2.6).

http://www.labfor.ru/index.php?act=labs&target=lab_el

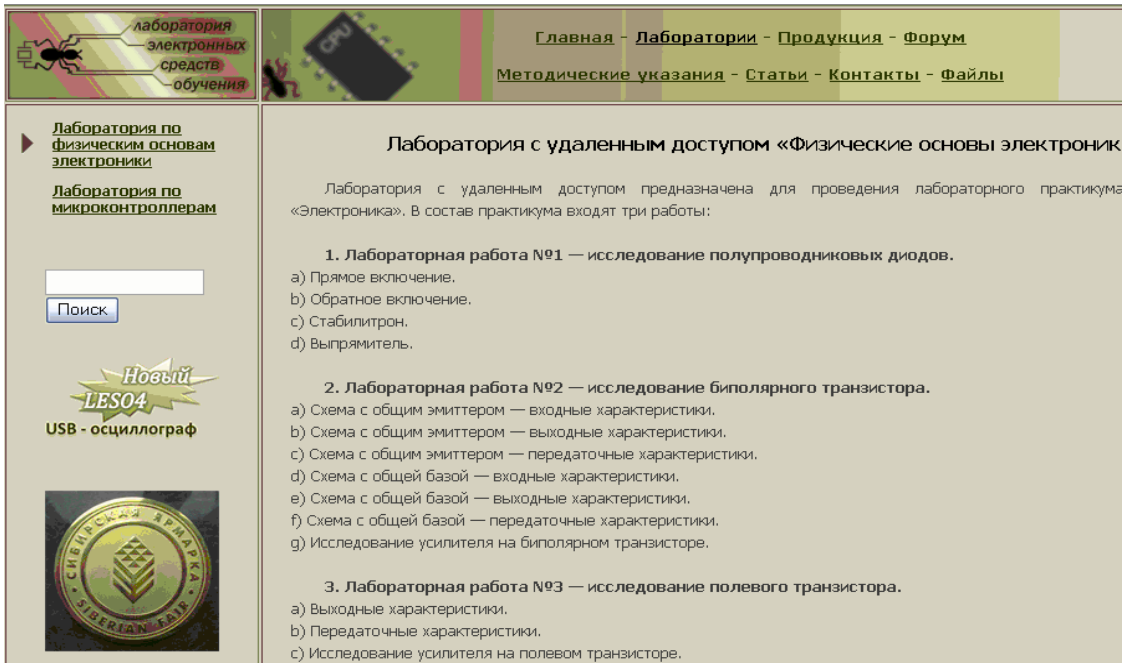


Рис.2.6. Интерфейс Web-страницы лаборатории с удаленным доступом

Для виртуальных автоматизированных лабораторий с удаленным доступом органы управления приборов и индикаторы измерителей находятся на экране персонального компьютера студента, подключенного к сети Internet. Для выполнения лабораторных работ с удаленным доступом нужно иметь экспериментальный стенд, оснащенный современным оборудованием и управляемый компьютером через Internet. Этот стенд может находиться в учебной лаборатории какого-нибудь университета, в научно-исследовательском институте или на предприятии. Этот подход позволяет изучать современные устройства, системы и сети, доступ к которым ограничен даже при очной форме обучения. Дополнительным достоинством этой технологии является возможность работы нескольких студентов на одном лабораторном стенде одновременно.

Установка состоит из сервера, подключенного к сети Internet. Сервер соединен с модулем USB-6008, который представляет собой недорогую систему сбора данных, производимую компанией National Instruments. Модуль подключается к серверу через интерфейс USB. Программное обеспечение виртуального комплекса состоит из двух частей — серверной и клиентской. За основу сетевого взаимодействия был выбран протокол гарантированной доставки TCP/IP.

Серверная часть представляет собой Win32 приложение, разработанное в среде Delphi 7. Программа принимает клиентов по протоколу TCP, управляет работой мультиплексора, коммутирующего исследуемую схему, а также управляет АЦП и ЦАП DAQ устройства NI-6008.

Клиентская часть выполнена в среде LabView 7.1. LabView был выбран из-за большой базы визуализированных приборов, делающих процесс работы студента интуитивно понятным, также в LabView есть встроенная поддержка протокола TCP/IP. На рис. 2.7 изображено окно исследования характеристик полевого транзистора.

Пример 3. На портале Киевского центра имитационного моделирования (http://www.simulation.kiev.ua) доступна виртуальная лаборатория - онлайн система OpenGPSS дискретно-событийного имитационного моделирования, работающая с языком GPSS, предназначенная для автоматического разделения заданий между узлами кластера, проведения параллельных вычислений и последующей сборки результатов.

Возможности работы в данной виртуальной лаборатории:

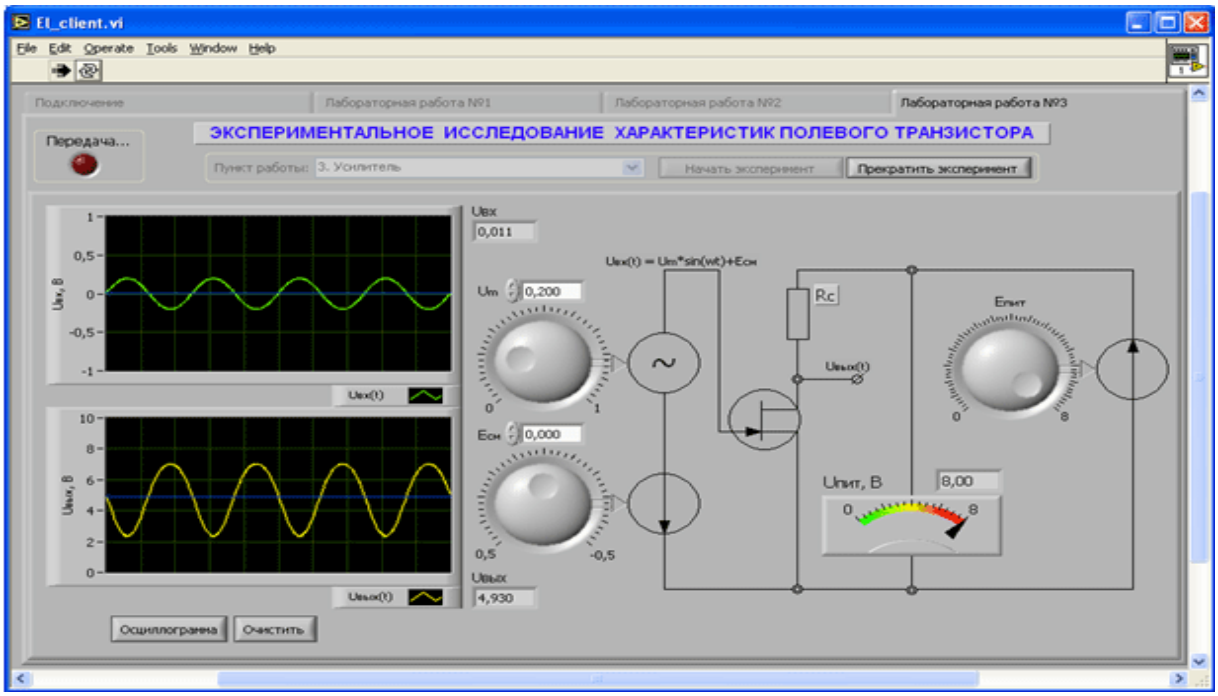


Рис. 2.7. Окно виртуального прибора по исследованию полевого транзистора

- работа с системой бесплатная;
- дискретно-событийная система рассматривает дискретные модели;
- работа в режиме онлайн через стандартный браузер;
- система совместима с языком имитационного моделирования GPSS, отличия от GPSS/PC связаны с различием архитектур систем и использованием новых технологий реализации;
- все разработанные ранее модели на языке GPSS можно запускать в системе OpenGPSS;
- система может выполнять несколько компьютерных прогонов одновременно для каждого пользователя.

Кто может работать в данной виртуальной лаборатории?

- преподаватели вузов для подготовки теоретического и практического обучения по курсам «Имитационное моделирование систем», «Моделирование систем» по разным направлениям подготовки;
- студенты для проведения лабораторных работ;
- организации по разработке имитационных моделей;
- заказчики моделей для проведения компьютерных прогонов, получения и анализа результата.

Виртуальные лаборатории такого типа позволяют создавать компьютерные модели и проводить компьютерные исследования на основе разработанной модели. Недостаток виртуальных лабораторий заключается в удаленности сервера, на котором находятся данные систем. Удаленность систем приводит к задержке и медленной работе компьютерных моделей.

2.3. Компьютерные вычислительные установки для научных исследований

Для разработки компьютерных установок по аналитическим, численным и вероятностным моделям используют языки и среды общего назначения: Pascal, Fortran, Visual Studio (Visual Basic, Visual C++, Visual C#), Forte for Solaris Developer Tools, Delphi Suite и Java JBuilder и др., а также технологии разработки компьютерных моделей в системах компьютерной математики Maple, MathCAD, Mathematica, MATLAB и др. Технологии разработки компьютерных моделей с помощью языков программирования и систем компьютерной математики можно изучить по книгам [1,2, 16-30, 39-43].

Выше были рассмотрены стадии и этапы разработки ПО и направления разработки программ компьютерного моделирования. Причем интуитивно предполагалось, что согласно технологии разработки программного обеспечения, разработку программ будут выполнять специализированные группы или несколько программистов, между которыми перераспределяется общая задача. Эти группы уточняют перечень задач по функциональным подсистемам, их постановку и алгоритмизацию. Одна часть разработчиков программного обеспечения работают, проводя взаимное согласование состава модулей и характеристик входных и выходных сигналов. Другая часть специалистов создают разделы технического проекта, относящиеся к техническим средствам и документации. Результатом работы всех групп является готовое программное обеспечение со всеми спецификациями и документацией.

Однако, рассматривая случай разработки программного обеспечения (компьютерных установок для моделирования) в области аналитического моделирования объекта исследования, чаще всего сталкиваемся с научным работником, работающим самостоятельно, и самое интересное в этом процессе то, что он одновременно является и заказчиком, и исполнителем. Поэтому он обходится минимумом удобств. При проектировании он отбрасывает многие дополнительные возможности оформления программного продукта, так как эта часть наиболее трудоемкая и на это уходит дополнительное время.

Отметим, что при таком подходе разработки программ в любом случае правила комментирования и первоначального оформления документации должны соблюдаться. Этап оформления пользовательского интерфейса идет по разным траекториям – версиям.

Версия 1. Так как исследование проходит в виде диалога «компьютер – исследователь», то разработчик программного обеспечения использует модульное строение программы численного решения математической задачи приведенной на рис. 2.8, но без интерфейса. Чаще всего используется блок задания исходных данных и вывода результатов исследования в программе моделирования. Все программы приложения разработаны согласно этой версии – подходу.

При разработке компьютерных установок, предназначенных для научных исследований в различных отраслях науки (физике, химии, биологии, экологии и др.), используется принцип модульной реализации (построения) программного комплекса и которая чаще всего состоит из следующих модулей – интерфейса, ядра, библиотеки, модулей пользователя. Каждый модуль состоит из спецификации и тела. Спецификации определяют правила использования модуля, а тело – способ реализации процесса обработки. Схема связей модулей показана на рис 2.8.

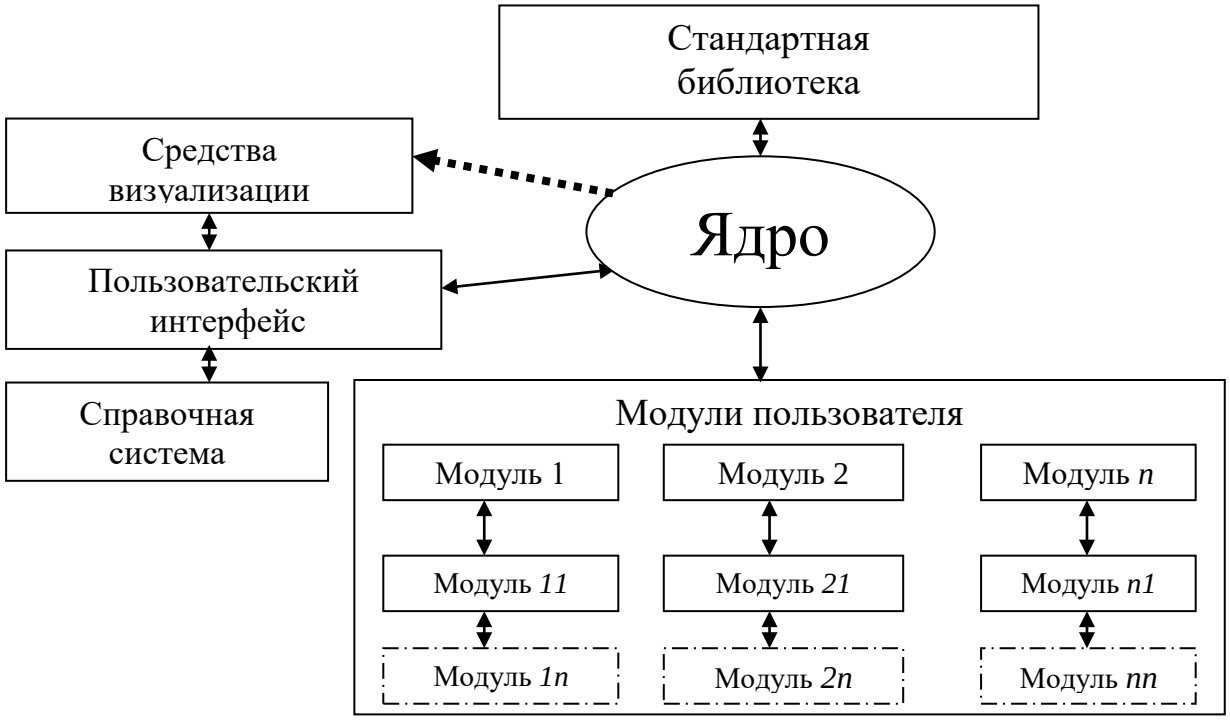


Рис.2.8 Модульное строение программ компьютерного моделирования

Версия 2. Здесь разрабатываются экранные формы представления исходных и выходных данных, используются готовые стандартные модули и простейшие элементы управления (рис.2.9).

Пользовательский интерфейс содержит:

– форму ввода данных;

– форму вывода с окном рисунка для вывода графика;

– форму управления координатной сеткой

Используются основные стандартные модули:

– процедуры решения дифференциальных уравнений

– процедура построения графика решения с процедурой построения координатной сетки.

Интерфейс – это часть программы, которая дает возможность пользователю использовать ядро и подключенные к нему библиотеки или модули. Оформление, дизайн внешнего вида, комфортность работы и удобство эксплуатации для пользователя заключается в этой части программного комплекса. В этой программе приводится меню пользователя, где задаются входные параметры задачи и выводятся результаты моделирования.

На рис. 2.9 приведен пример оформления графического интерфейса пользователя входных и выходных параметров программы моделирования прохождения нейтронов через пластинку.

Для разработки интерфейса чаще всего используются среды программирования Visual Studio (Visual Basic, Visual C++, Visual C#) – компании Microsoft, Forte for Solaris Developer Tools – компании Sun Microsystems Inc., Delphi Suite и Java JBuilder – компании Borland International и языки программирования Forth, Simula-67, OAK, Smalltalk и др.

Ядро компьютерной модели представляет собой программу, выполняющую основную вычислительную функцию решения задачи математического моделирования. Для задач вычислительного характера для написания ядра используются языки программирования Fortran, Cobol, Ada, PL/1, Фокал, Basic, Pascal и др. Пользователь работает с ядром посредством программы интерфейса. Ядро программы разрабатывается так, что любой алгоритм в нем работает достаточно быстро. Для расширения набора стандартных процедур используется библиотека языка кодирования и набор модулей пользователя.

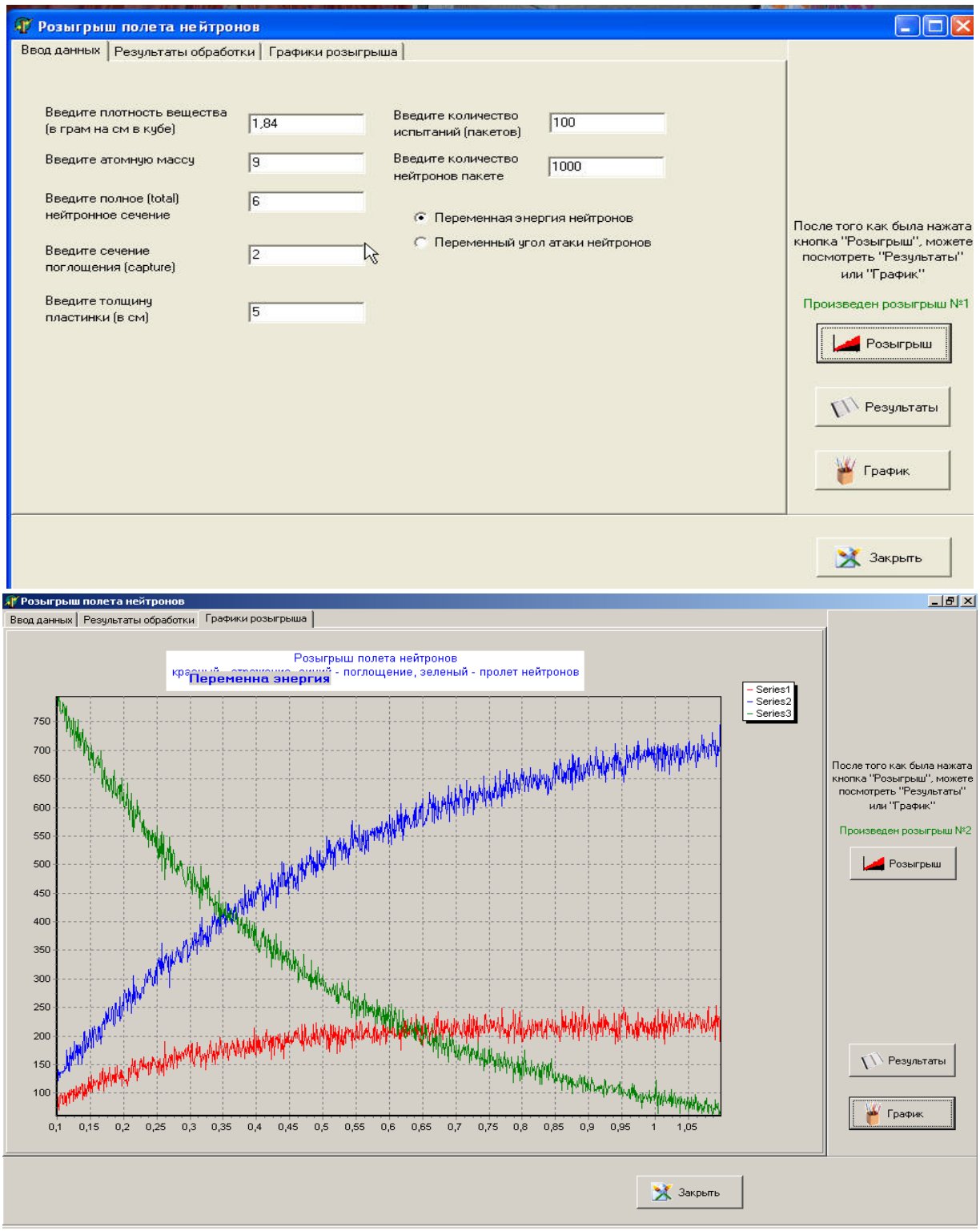


Рис. 2.9. Интерфейс входных и выходных данных программы моделирования прохождения нейтронов через пластинку (среда программирования Delphi)

Общая схема программы ядра состоит из трех последовательных блоков:

1. Блок задания входных постоянных параметров и исходных данных задачи.

3. Блок основного тела программы ядра, где проводится решение задачи на алгоритмическом языке программы.

3. Блок вывода результатов решения математической задачи.

Приведем пример решения одной задачи для моделирования маятника Фуко по методу Эйлера на языке Pascal.

1	{Программа моделирования колебаний маятника Фуко методом Эйлера}
	Program Mayatnik_Fuko;
	const pi=3.1415; g=9.8; n=2000;
	var m,gd,gm,I:integer;
	r,w,t,tt,dt,v,d,L,vx0,vy0,x0,y0:real;
	x,y,vx,vy:array [0..n] of real;
	mfuko:text;
	begin
	{Исходные данные}
	dt:=0.01; L:=50; tt:=10; w:=0.04;
	vx[0]:=0; vy[0]:=0; x[0]:=1; y[0]:=0;
	t:=n*dt; m:=round(tt/dt);
		Блок описания переменных программы и исходных данных (параметров и входных значений) математической задачи

2	{Основное тело программы}
	for i:=0 to m-1 do
	begin
	R:=sqrt(x[I]*x[i]+y[i]*y[i]);
	if r=0 then r:=1e-6;
	v:=sqrt(vx[i]*vx[i]+vy[i]*vy[i]);
	If v=0 then v:=1e-6;
	vx[i+1]:=vx[i]+dt*(2*vy[i]*w+w*w*x[i]-
	g*x[i]/L);
	vy[i+1]:=vy[i]+dt*(-2*vx[i]*w+w*w*y[i]-
	g*y[i]/L);
	x[I+1]:=x[i]+dt*vx[i];
	y[I+1]:=y[i]+dt*vy[i];
	end;
		Основное тело программы ядра – решение поставленной математической задачи

3	{Вывод результатов моделирования} assign (mfuko,'c:\mafuko.DAT'); rewrite (fuk); writeln(y[m]); for i:=0 to m do writeln(mfuko, x[i],' ',y[i]); close (mfuko); readln; end.	Блок вывода результатов решения математической задачи.

Интерфейс пользователя является самой дорогой и трудоемкой частью разработки программного комплекса. Поэтому в зависимости от цели и назначения программы математического моделирования разработка интерфейса всегда стоит под вопросом и разработчики-исследователи доволь-

ствуются минимумом удобств. Чаще всего разработка интерфейса проводится для случая, когда программный продукт готовится для передачи заказчику, тиражирования или внедрения в фонд алгоритмов и программ. Поэтому для удобства и обучения пользователя разрабатывается электронная справочная система или руководство пользователя, где представляются необходимые сведения для работы с программой (общие сведения, системные требования, описание компьютерной модели, средства навигации по вводу и выводу данных, необходимые модули программы и так далее).

Библиотека. В любой среде программирования существуют своя библиотека готовых стандартных программ (модулей) в виде процедур или функций. Пользователь часто использует эти готовые модули при решении своих задач. В свою очередь эти модули могут использовать другие дополнительные модули для выполнения своих процедур или функций.

Модули пользователей. Кроме стандартных библиотек, поставляемых с пакетом среды программирования, пользователи при разработке программы моделирования могут использовать свою личную библиотеку, т.е. собственные разработанные модули (процедуры и функции). Эти модули могут быть разработаны на различных базовых языках и откомпилированы в виде исполняемых *.exe или *.com файлов.

Средства визуализации. Результаты решений представляют в виде графиков. Причем полученные графики должны быть выведены для текстового редактора Word, в целях публикации. Для этих целей разрабатывают модуль построения и выдачи графика или используют готовые графические пакеты. В частности, можно использовать графические пакеты Microcal Origin, Advanced Grapher и др. Эти пакеты широко применяются исследователями всего мира, в том числе и в нашей стране для обработки и визуализации экспериментальных и численных данных.

Microcal Origin является приложением системы Windows. Пользовательский интерфейс данного программного продукта не отличается от других приложений. Этот графический пакет очень удобен для обработки результатов при проведении как реального, так и вычислительного эксперимента. Он обладает многими возможностями:

- построения двухмерных графиков в декартовой и полярной системе координат;
- построения множества графиков в одной координатной плоскости;
- построения множества графиков в нескольких координатных плоскостях;

- построения различных видов диаграмм;
- построения трехмерных графиков;
- вращения трехмерных графиков относительно любой из осей;
- регрессии и интерполяции графиков;
- статистической обработки результатов эксперимента и графиков;
- экспорта и импорта данных и графиков в другие приложения и графические пакеты.

Для работы в этом пакете необходимо иметь экспериментальные данные в виде текстового файла с расширением *.dat. Подготовка такого файла, например, на языке Паскаль осуществляется следующим образом:

```
FIL:TEXT;                          {описание текстового файла}
ASSIGN(FIL,'C:\FILTR.DAT');        {открытие и указание пути сохранения
                                   файла}
REWRITE(FIL);                      {обнуление файла}
```

Тело программы

```
FOR I:=1 TO N DO BEGIN
  WRITELN(FIL,I*H, ' ', U[I]);END;   {формирование файла }
CLOSE(FIL);                          {закрытие файла}
```

Построение графика проводится следующим путем. В меню **File** находится подменю **Import**, в котором имеется подменю **ASCII**. Нажатие указателя мыши по нему открывает файлы папки Origin. Здесь осуществляется поиск необходимого нам файла **ASCII**. Для приведенного примера этот файл закидывается в корневой каталог C: в виде **ASCII** файла FILTR.DAT. Открытие **ASCII** файла приводит появление в окне Origin значений в виде таблицы (рис.2.10). Построение графиков проводится с помощью меню **Plot** с различными возможностями выбора вида графиков. Для построения трехмерных графиков необходимо сформировать ASCII-файл в табличной форме, затем в меню **Edit** конвертировать его в матрицу с помощью подменю **Direct.** Матрица обычно закрашена в желтый цвет. Используя меню **Plot 3D,** можно построить трехмерный график с выбором вида графика (рис.2.11).

Меню **Matrix** позволяет инвертировать, транспонировать и интерполировать полученную матрицу. С помощью меню **3D** трехмерный график можно вращать относительно осей координат, наклонять. Режим **Options** в меню **3D** позволяет изменять параметры и расцветку трехмерного графика. Изменение значений и параметров координатных осей, цвета графиков про-

изводится в подменю, которое открывается быстрым двойным щелчком мыши по графику или его параметрам.

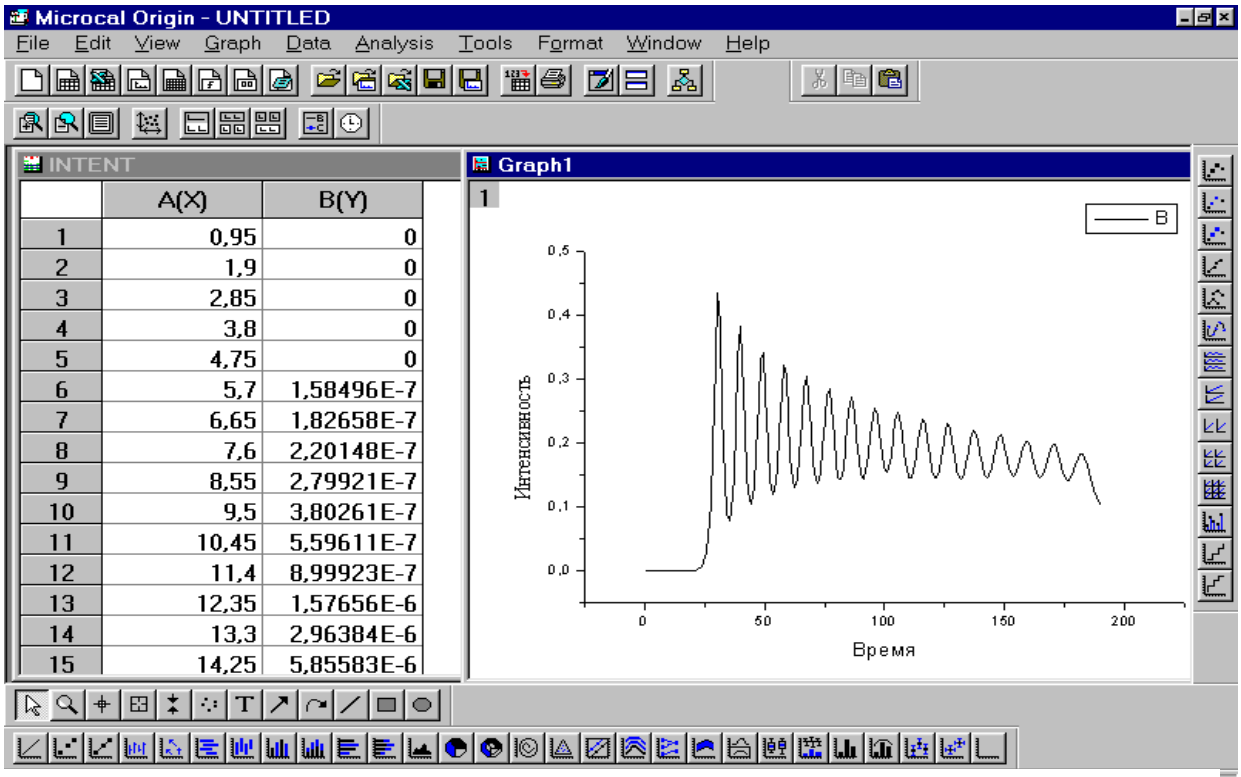


Рис.2.10. Построение двухмерных графиков

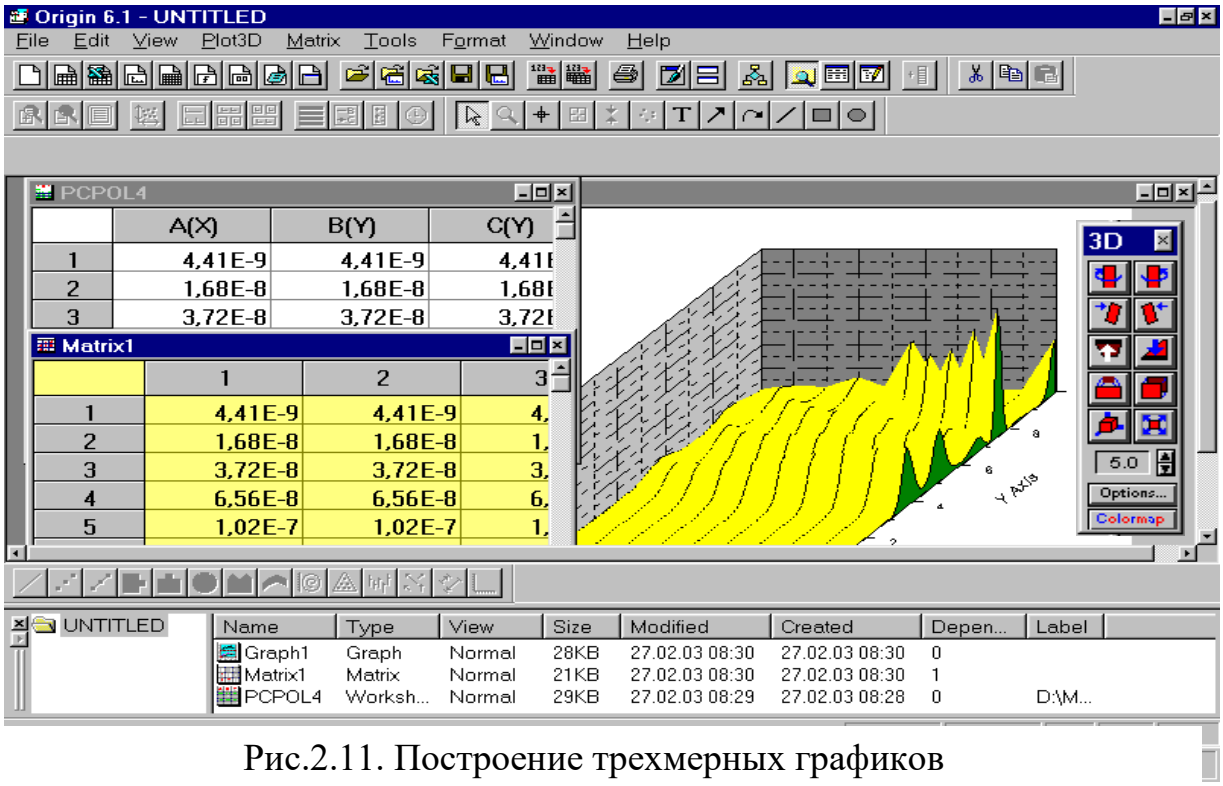


Рис.2.11. Построение трехмерных графиков

2.4. Инструментарии и средства автоматизации компьютерного моделирования

Одно из направлений развития вычислительных технологий в настоящее время — это разработка мощных математических пакетов и средств автоматизации компьютерного моделирования, позволяющих максимально упростить процессы подготовки задачи, проектирования компьютерной установки, планирования, проведения компьютерного эксперимента, представления и анализа результатов моделирования.

Создание универсальных программных средств символьной математики стало, в последнее время, основой нового научного направления в информатике, которое получило название — *компьютерная алгебра.* Наиболее известными и широко используемыми на практике являются такие математические системы, как Mathematica, Maple, Mathcad, MATLAB и др. Эти математические компьютерные системы, в общей совокупности, представляют собой непревзойденные средства решения самых сложных математических и инженерных задач.

Эру создания компьютерной символьной математики принято отсчитывать с начала 60-х годов. Именно тогда в вычислительной технике возникла новая ветвь компьютерной математики, названная компьютерной алгеброй. Речь шла о возможности создания компьютерных систем, способных осуществлять типовые алгебраические преобразования: подстановки в выражениях, упрощение выражений, операции со степенными многочленами (полиномами), решение линейных и нелинейных уравнений и их систем, вычисление их корней и т. д. При этом предполагалась возможность получения аналитических (символьных) результатов везде, где это только возможно.

Разработка таких систем была вызвана необходимостью использования математического аппарата пользователями, не имеющими профессионального математического образования. Большинство же пользователей заинтересовано в том, чтобы правильно выполнить конкретные несложные аналитические преобразования, алгебраические вычисления, вычислить в символьном виде производную или первообразную заданной функции, разложить ее в ряд Тейлора или Фурье, провести аппроксимацию и т. д. Тем более, что предметные области, представляющие интерес для пользователя (будь он математик, физик, биолог или химик), перегружены своим собственным математическим аппаратом. Иначе говоря, большинству пользователей нужны системы компьютерной алгебры в качестве простого и удобного инструмента для работы.

Это положение определило направление разработок западных фирм по созданию компьютерных систем символьной математики, ориентированных на широкие круги пользователей, не являющихся профессионалами в компьютерной алгебре. Учитывая невероятно большую сложность автоматизации решения задач в аналитическом виде (число математических преобразований и соотношений весьма велико, и некоторые из них неоднозначны в истолковании), первые подобные системы удалось создать лишь для больших ЭВМ. Но затем появились и системы, доступные для мини-ЭВМ.

Заметное развитие получили языки программирования для символьных вычислений Reduce, система muMath для малых ЭВМ, а в дальнейшем — интегрированные системы символьной математики для персональных компьютеров: Derive, MathCAD, Mathematica, Maple V и др.

В бывшем СССР большой вклад в развитие систем символьной математики внесла школа академика Глушкова. В конце 70-х годов были созданы малые инженерные ЭВМ класса «Мир», способные выполнять аналитические вычисления даже на аппаратном уровне. Был разработан и успешно применялся язык символьных вычислений «Аналитик». Эти работы отчасти предвосхитили развитие систем символьной математики. К огромному сожалению, они появились слишком рано для своего времени и не соответствовали «генеральной линии» развития советской вычислительной техники в те годы. Уклон в сторону развития больших ЭВМ серии ЕС, навязанный в СССР компьютерными чиновниками, отодвинул компьютеры «Мир» на задний план, а затем этот класс компьютеров просто прекратил свое существование и развитие.

Решающий скачок в компьютеризации общества произошел с началом массового производства и внедрения ПК. Долгое время их ограниченные возможности не позволяли реализовать на них серьезные системы символьной математики. Но к началу 90-х годов ситуация стала заметно меняться к лучшему. С одной стороны, аппаратные возможности ПК стали резко возрастать по мере быстрой смены поколений микропроцессоров. В итоге по скорости счета и объему оперативного запоминающего устройства (ОЗУ) ПК стали обходить «большие» ЭВМ класса ЕС, а сейчас оставили их далеко позади. Это создало реальные предпосылки к развертыванию работ по разработке систем компьютерной алгебры. Даже на сегодняшний день разрыв в производительности между новейшими ПК и многопроцессорными супер-ЭВМ остается достаточно большим.

Наибольшую известность получили три класса систем символьной математики: созданная на базе языка искусственного интеллекта Mu Lisp малая

система Derive, одна из самых мощных и поныне привлекательных систем Maple V (ядро написано на языке C) и системы Mathematica 1 и 2. Позже на базе ядра системы Maple V символьные вычисления были реализованы в популярных числовых системах Mathcad – версии Mathcad 2.0/4.0/5.0/ Plus5.0/ 6.0/Plus 6.0/7.0 /Plus7.0/8.0/8.0PRO/2000PRO/2000 Premium имеют замечательный пользовательский интерфейс и возможности, улучшающиеся от версии к версии. Блок символьной математики на базе ядра Maple V был добавлен и в одну из самых крупных матричных систем — MATLAB.

Система Derive и поныне является привлекательной своими невзыскательными требованиями к аппаратным ресурсам ПК — это единственная система, которая работает даже на ПК класса IBM PC XT без жесткого диска. Новая версия Derive 7 под Windows имеет современный интерфейс [22]. А по возможности графической визуализации результатов вычислений Derive все еще далеко отстает от них. То же можно сказать и о новой системе символьной математики MuPAD 1.4.

Система Maple V (компании Waterloo Maple Inc, Канада) — патриарх в семействе систем символьной математики. И поныне это весьма привлекательная система для математика-аналитика и научного работника [19]. Новейшие системы Maple V для Windows (реализации R5 и R6) по возможностям графики стоят на одном уровне с системами Mathematica 3/4. Считается, что они несколько превосходят системы Mathematica в части символьных преобразований, но такое превосходство на сегодня уже является весьма спорным. Полнота ядра системы, хранящего более 2700 математических функций (у реализации Maple 6 и Maple 9 их уже свыше 3000!) и правил их преобразования, вполне заслуживает внимания. Весьма привлекательное свойство этой системы — подробная встроенная помощь и множество примеров ко всем встроенным в нее функциям и прикладным пакетам. Эти примеры легко скопировать в окно редактирования системы и тут же решить.

Появление новых версий Mathematica 3 и 4 вновь резко поднимает планку оценки качества систем компьютерной алгебры [20]. Наступает новый этап интеграции математических систем как друг с другом, так и с современными текстовыми и табличными процессорами, такими как Word и Excel из офисных пакетов Microsoft Office.

Компания MathSoft, Inc., к примеру, за какие-то десять лет породила добрый десяток новых версий (инкрементов) популярной системы Mathcad — 2.0, 4.0, 5.0, Plus 4.0, 6.0, Plus 6.0, 7.0, Plus 7.0, 8.0, 8.0 PRO, Mathcad 2000 PRO/Premium, Mathcad — 12.0, 14.0 [24].

В этой среде описание записи и решения математических задач дается с помощью привычных математических формул и знаков. Такой же вид имеют и результаты вычислений. Поэтому среда MathCAD вполне оправдывает аббревиатуру CAD (Computer Aided Disign), говорящую о принадлежности к сложным и продвинутым средствам автоматизации аналитического моделирования.

Компания MathWorks (США) разработала систему автоматизации вычислений и моделирования под названием MATLAB [2, 11, 18, 23]. Достаточно интересны реализации матричных систем MATLAB 5.2/5.3, но с очень громоздкой системой поддержки вычислений на дисковом пространстве, например — MATLAB 5.2.1 занимает на жестком диске 1500 Мбайт памяти (Mathematica 4 требует на порядок меньше места), для версий MATLAB 6.*, 7. разработчики уменьшили дисковое пространство до 1000 Мбайт. Представляет интерес универсальная система математического моделирования Simulink, интегрированная с математической системой MATLAB, позволяющая проводить визуальное моделирование динамических систем. Громоздкость и дороговизна этой системы сказывается на широком применении в вузах и университетах.

Более открытой системой визуального математического моделирования является VisSim (Visual Simulation), созданная фирмой Visual Solutions Incorporation[17] . Эта система интегрируется с системами компьютерной математики Mathcad 2000/2001/2001i /11 и MATLAB. Отличительная особенность этой системы сокращение времени моделирования в несколько по сравнению с другими системами.

Большую популярность приобрели также специализированные системы схемотехнического моделирования MultiSIM (Electronics Workbench) [27], MicroCap, P – CAD, PSpice, Design LAB, OrCAD и др.

Особое место среди систем математического моделирования занимает среда проектирования виртуальных приборов LabVIEW. Среда графического программирования LabVIEW получает всё большее распространение в промышленности и образовании, при проведении научных исследований и выполнении проектных работ.

Этому способствуют её несомненные преимущества – высокая производительность при разработке программ, называемых виртуальными приборами (ВП), широкий набор функциональных возможностей языка и среды программирования. В последнее время изложению этих вопросов в отечественной литературе был посвящен целый ряд публикаций и книг [4, 7, 16, 26, 32, 37]. В них были рассмотрены основные элементы среды LabVIEW,

основы программирования на языке G в LabVIEW, а также дополнительные возможности среды.

Для моделирования сложных систем разрабатываются различные программные продукты, реализующие объектно-ориентированный подход к программированию. Эти системы имитационного моделирования позволяют строить компьютерные модели сложных систем в разных областях производства, логистики, транспорта, в общественных и социальных науках и других отраслях. Например:

- Powersim Studio (компании Powersim Software AS) система динамического моделирования (для системной динамики) применяется для решения сложных задач в широком диапазоне предметных областей – от истории, литературы, социологии до биологии, физики и экономики;
- iThink Analys (компания iSee systems) – объектно-ориентированный пакет программ, обеспечивающий графическую, вычислительную и информационную поддержку процедурам системного анализа сложных процессов. iThink позволяет описывать модели разнообразных систем;
- Pilgrim (МЭСИ, Россия) – интегрированная объектно-ориентированная система имитационного моделирования для подхода системной динамики, а также стохастического дискретного и пространственного моделирования;
- AnyLogic (российская компания XJ Technologies – « Экс Джей Технолоджис») — программное обеспечение для имитационного моделирования сложных систем и процессов, позволяющее поддерживать направление агентного моделирования, дискретно-событийного моделирования и разработки моделей системной динамики;
- GPSS (англ. General Purpose Simulation System — общецелевая система моделирования) — язык объектно-ориентированного программирования, используемого для имитационного моделирования систем массового обслуживания различных информационных процессов и разработки имитационных моделей в сети интернет;
- Arena – программное обеспечение для имитационного моделирования (компании Systems Modeling Corporation), позволяющее создавать подвижные компьютерные модели, используя которые можно адекватно представить очень многие реальные системы. Эта система ориентирована на разработку моделей производства, бизнес-процессов и в области систем массового обслуживания;

- Plant Simulation — программная среда имитационного моделирования систем и процессов, предназначенная для оптимизации материалопотоков, загрузки ресурсов, логистики и метода управления для всех уровней планирования от целого производства и сети производств до отдельных линий и участков;
- SimBioSys: C++ – оболочка агентно-базового эволюционного моделирования в биологических и общественных науках;
- система агентного моделирования SWARM и его расширения MAML (Multi-Agent Modelling Language) для моделирования искусственного мира;
- пакеты Ascape(Agent Landscape) и RePast (Recursive Porous Agent Simulation Toolkit), написанные на платформе языка Java для поддержки агентно-базового моделирования;
- NetLogo и MIMOSE (Micro- and Multilevel Modelling Software) информационные системы, предназначенные для создания имитационных моделей и технологий моделирования в общественных науках;
- SPSS, Statistica, Z-Tree – системы моделирования для исследования экономических, статистических явлений и процессов.

Также следует отметить пакеты Process Charter (Scitor Corp.), PROCESS MODEL (PROMODEL Corporation), SIMUL8 (Visual Thinking International), EXTEND (Imagine That, Inc.) и другие системы.

Системы компьютерного моделирования можно условно подразделить на системы аналитического, технического и имитационного моделирования

В таблице 2.2 даны технологические характеристики некоторых современных систем моделирования.

Таблица 2.2.

Современные системы моделирования

Система моделирования	Производитель ПО	Приложения	Моделирующая среда и поддержка			
			Графическая поддержка	Авторское моделирование, программирование моделей	Анимация (в реальном времени)	Поддержка анализа и визуализации результатов
Системы аналитического моделирования						
DERIVE	Soft Warehouse (США)	Аналитическое, численное моделирование	Двухмерная, трехмерная графика	+	+	+
	Waterloo	Аналитическое, чис-	Графический	+	+	

MAPLE	Maple Inc, Канада	ленное моделирование	пакет	язык Maple		
MATHEMAT-ICA	Wolfram Research, Inc. (США)	Аналитическое, численное моделирование	Графический пакет	+	+	
MathCAD	MathSoft, Inc	Аналитическое, численное моделирование	Графический пакет	+	+	
MATLAB	MathWorks (США)	Аналитическое, численное моделирование	Блок-схемы, графический пакет, потоковые диаграммы	+ Язык MATLAB	-	+
MvStudium	MvStudium Group (Россия)	Аналитическое, численное моделирование, визуальное интерактивное моделирование	Блок-схемы, графический пакет	объектно-ориентированное программирование	+	+
Системы технического моделирования						
Vissim	Visual Solutions (США)	Моделирование систем автоматического регулирования и управления, моделирование различных физических, химических, экономических и прочих явлений и систем.	Блок-схемы, графический язык моделирования	Объектно-ориентированное визуальное программирование	+	+
Simulink приложение MATLAB	MathWorks (США)	Моделирование систем автоматического регулирования и управления, моделирование различных физических, химических, экономических и прочих явлений и систем.	Блок-схемы, графический язык моделирования	Объектно-ориентированное визуальное программирование	+	+
Multisim	Interactive Image Technologies; Electronics Workbench; National Instruments (США).	схемотехническое моделирование аналоговых и цифровых радиоэлектронных устройств	Электронные схемы, графический язык моделирования	Объектно-ориентированное визуальное программирование	+	+
LabVIEW,	National Instruments (США).	Разработка виртуальных приборов, схемотехническое моделирование аналоговых и цифровых радиоэлектронных устройств	Электронные схемы, блок-схемы, графический язык моделирования	Виртуальное графическое программирование	+	+
Micro-Cap	Spectrum	схемотехническое мо-	Электронные	Объектно-	+	+

	Software	делирование аналоговых и цифровых электро- радиоэлектронных устройств	схемы, графический язык моделирования	ориентированное визуальное программирование		
System View	Elanix	Схемотехническое моделирование устройств на функциональном уровне	Схемотехнический язык моделирования	-	+	+
Design Lab +PSpice	MicroSim	Проектирование и моделирование принципиальных схем радиоэлектронных устройств	Электронные схемы, графический язык моделирования	-	-	+
Circuit Maker	MicroCode Engineering	Схемотехническое моделирование аналоговых и цифровых электро- радиоэлектронных устройств	Электронные схемы, графический язык моделирования	-	-	+
Системы имитационного моделирования						
ARENA	System Modeling Corporation	Производство, анализ бизнес-процессов, дискретное моделирование, складской учет, логистика, транспортные задачи	Блок-схемы	Объектно-ориентированное визуальное моделирование +	+	+
EXTEND	Imagine That, Inc.	Стратегическое планирование, бизнес-моделирование	Компоновочные блоки, непрерывные и дискретные модели	+ язык Modl	+	Анализ чувствительности
GPSS/H-PROOF	Wolverine Software Corporation	Общего назначения, производство, транспорт и др.	Блок-схемы	+	+	ANOVA
ITHINK ANALYST	High Performance System, Inc.	Управление финансовыми потоками, реинжиниринг предприятий, банков, инвестиционных компаний и др.	CASE-средства, имитационное моделирование, потоковые диаграммы	+	+	Анализ чувствительности

eM-Plant	Siemens AG (Siemens PLM Software), Германия.	Производство (автомобильная отрасль, электроника, судостроение, станкостроение, сборочные линии и т.д.), логистика, сбыт, консалтинг, симуляция бизнес процессов, здравоохранение, банковский бизнес	CASE-средства, имитационное моделирование, потоковые диаграммы	Объектно-ориентированное визуальное моделирование +	+	+
PROCESS MODEL	PROMODEL Corporation	Общее производство, реинжиниринг	Блок-схемы, дискретное моделирование	-	-	+
SIMUL8	Visual Thinking International	Универсальное средство имитации дискретных процессов	-	Объектно-ориентированное программирование	+	+
TAYLOR SIMULATION SOFTWARE	F&H SimulationInc.	Производство, стоимостный анализ	Блок-схемы, дискретное моделирование	-	+	+
WITNESS	Lanner Group Inc.	Бизнес-планирование, производство, финансы	+	+	+	+ Блок оптимизации
VENSIM	Ventana Systems	Модели системной динамики	Потоковые диаграммы	-	-	Анализ чувствительности, оптимизация
POWERSIM	Powersin Co.	Непрерывное моделирование	Потоковые диаграммы	-	+	-
DYNAMO	Expectation Software	Модели системной динамики вычислительного типа	Блок-схемы	-	-	-
ANYLOGIC	XJ Technologies	Системная динамика, Дискретно-событийное, агентное моделирование	Потоковые и UML диаграммы, дискретное моделирование	Java Объектно-ориентированное визуальное моделирование	2D, 3D	-
						Оптимизация OptQuest

Отметим, что на мировом рынке появился ряд конкурентоспособных российских разработок по имитационному моделированию сложных систем (таблица 2.3).

Таблица 2.3

Инструментальные системы моделирования, разработанные в России

1	Среда моделирования MvStudium	Позволяет быстро создавать визуальные интерактивные модели многокомпонентных непрерывных, дискретных и гибридных (непрерывно-дискретных) систем и проводить с ними активные вычислительные эксперименты. MvStudium поддерживает объектно ориентированное моделирование и возможность создания пользователем своих собственных компонентов с использованием входного языка. Поддерживается 2D и 3D-анимация.	Колесов Ю.Б. (Москва), Инихов Д.Б, Сениченков Ю.Б (Санкт-Петербург), Россия. http://www.mvstudium.com/
2	Среда моделирования Rand Model Designe	Среда объектно - ориентированного моделирования и проектирования на базе математического моделирования сложных природных и технических объектов. Продукт поддерживает технологии проектирования многокомпонентных иерархических событийно-управляемых систем - компонентное моделирование с ориентированными и неориентированными компонентами (связями).	Инихов Д.Б., Колесов Ю.Б., Сениченков Ю.Б., Москва – Санкт-Петербург, Россия. http://www.mvstudium.com/ http://www.randservice.com/
3	Среда моделирования OpenMVL	Открытая среда для моделирования сложных динамических систем (аналогичная OpenModelica). Среда представляет собой набор модулей, связанных с решением математических задач, возникающих при моделировании многокомпонентных сложных динамических систем.	Авторский коллектив разработчиков – Исаков А.А., Сениченков Ю.Б., Санкт-Петербург, Россия. Сайт: https://dcn.ftk.spbstu.ru/index.php?id=275
4	Система Object GPSS (GPSS – Future)	Инструментальное средство для написания моделей в стиле GPSS непосредственно на языке Delphi (Object Pascal).	Северодонецкий технологический институт, Северодонецк, Украина. (автор Королёв А.Г) http://objectgpss.narod.ru/

		Каждая модель на Object GPSS представляет собой Include–файл (Model.pas), содержащий описание всех объектов модели и набор из 6 процедур: Initial, CloseAllObj, ResetAll, ModelTxt, Report, Modeling. Система GPSS – Future развивает Object GPSS в плане более гибкой работы со списком будущих событий.	
5	Среда моделирования «МВТУ»	Среда интеллектуального САПР, предназначенная для детального исследования и анализа нестационарных процессов в системах автоматического управления, в ядерных и тепловых энергоустановках, в следящих приводах и роботах, в любых технических системах, описание динамики которых может быть реализовано методами структурного моделирования. Является альтернативой программным продуктам SIMULINK, VisSim, MATRIXx и др.	Козлов О.С., Кондаков Д.Е., Скворцов Л.М., Тимофеев К.А., Ходяковский В.В., Москва, Россия .МВТУ им.Н.Баумана http:// energy.power.bmstu.ru
6	Имитационная платформа Фантомат	Моделирование инструментов для моделирования и обучения	Департамент систем имитационного моделирования IBS http://www.fantomat.ibs.ru
7	Профессиональный инструмент моделирования AnyLogic	Среда моделирования позволяющая строить сложные имитационные модели. Поддерживает подходы динамических систем, дискретно-событийного моделирования, системной динамики, агентного моделирования. Поддерживается 2D и 3D-анимация	Экс Джей Технолоджис, 194021, Санкт-Петербург, ул. Политехническая, 21 www.xjtek.com
8	Интерактивная система ИМ ISS-2000	Среда моделирования дискретно-событийных процессов.	Национальный технический университет «Киевский политехнический институт», Украина, г. Киев (автор Томашевский В.И.).
9	Среда моделирования СМО	Среда моделирования дискретно-событийных процессов с визуализацией.	Науменко Вячеслав Андреевич, Россия. http://modelsmo.narod.ru/

Для эффективного применения моделирования при решении исследовательских, производственных, управленческих и образовательных задач

необходимо четкое понимание возможностей и ограничения выбранного аналитического инструмента моделирования из представленных в таблице 2.2 программных продуктов. Отдельно можно отметить популярные системы для моделирования локальных и глобальных вычислительных систем и сетей.

NanoCAD — базовая система автоматизированного проектирования и черчения (САПР-платформа). Разработана компанией ЗАО Нанософт (Россия).

BONeS (фирма Systems and Networks) – графическая система моделирования общего назначения для анализа архитектуры систем, сетей и протоколов. Описывает модели на транспортном уровне и на уровне приложений. Дает возможность анализа воздействия приложений типа клиент-сервер и новых технологий на работу сети;

Netmaker (фирма OPNET Technologies) – проектирование топологии, средства планирования и анализа сетей широкого класса. Состоит из различных модулей для расчета, анализа, проектирования, визуализации, планирования и анализа результатов;

Optimal Perfomance (фирма Compuware; Optimal Networks) – имеет возможности быстрого оценочного и точного моделирования, помогает оптимизировать распределенное программное обеспечение;

Prophesy (компания Abstraction Software) – простая система для моделирования локальных и глобальных сетей. Позволяет оценить время реакции компьютера на запрос, количество "хитов" на WWW-сервере, количество рабочих станций для обслуживания активного оборудования, запас производительности сети при поломке определенного оборудования;

Stressmagic (фирма NetMagic Systems) – поддержка стандартных тестов измерения производительности; имитация пиковой нагрузки на файл-сервер и сервер печати. Возможно моделирование взаимодействия различных пользователей с файл-сервером. Включает 87 тестов производительности;

NetWork II.5 - автономный пакет фирмы CASI для анализа производительности используемых компьютерных систем, для анализа приложений в сети. Позволяет проводить моделирование компьютерной архитектуры любого типа.

Семейство CANE (компания ImageNet) – проектирование и реинжиниринг вычислительной системы, оценка различных вариантов, сценарии "что, если". Моделирование на различных уровнях модели OSI. Развитая библиотека устройств, которая включает физические, электрические, температур-

ные и другие характеристики объектов. Возможно создание своих библиотек;

Семейство OPNET (фирма OPNET Technologies) – средство для проектирования и моделирования локальных и глобальных сетей, компьютерных систем, приложений и распределенных систем. Возможность импорта и экспорта данных о топологии и сетевом трафике. Анализ воздействия приложений типа клиент-сервер и новых технологий на работу сети. Моделирование иерархических сетей, многопротокольных локальных и глобальных сетей; учет алгоритмов маршрутизации. Объектно-ориентированный подход. Исчерпывающая библиотека протоколов и объектов включает следующие продукты:

NETBIZ (проектирование и оптимизация вычислительной системы);

MODELER (моделирование и анализ производительности сетей, компьютерных систем, приложений и распределенных систем);

ITGURU (оценка производительности коммуникационных сетей и распределенных систем).

Семейство COMNET (фирма Compuware; CACI Products Company) -- объектно-ориентированная система моделирования локальных и глобальных сетей. Позволяет моделировать уровни: приложений, транспортный, сетевой, канальный. Использует все известные на сегодня технологии и протоколы, а также системы клиент-сервер. Легко настраивается на модель оборудования и технологий. Возможность импорта и экспорта данных о топологии и сетевом трафике. Моделирование иерархических сетей, многопротокольных локальных и глобальных сетей; учет алгоритмов маршрутизации.

COMNET III – система стохастического дискретного событийного моделирования систем массового обслуживания. основная система семейства для моделирования ЛВС и глобальных сетей, а также их планирования. Пользователь с помощью графического интерфейса создает иерархическую модель сети, а затем система проводит моделирование сети с анимацией, при этом позволяет: создавать отчеты производительности; прогнозировать межконцевую задержку, пропускные способности и загрузку линий связи, буферов и процессоров; воспроизводить случайную модель трафика; видеть пики и колебания трафика; определять источники задержек и узких мест. Позволяет детально моделировать сети как СМО, построенные с использованием всех известных технологий и протоколов, как то: ATM , Frame Relay , FDDI , TCP/IP , клиент-сервер и т.д. Результатами моделирования являются оценки производительности различных вариантов построения исследуемой локальной или глобальной сети, учитывая при этом стоимостные характеристики.

ADVANCED FEATURES PACK – данный пакет предоставляет дополнительные возможности пакету COMNET III для точного моделирования распределенного программного обеспечения клиент-серверных архитектур.

COMNET Predictor – система быстрого временного анализа. Предоставляет возможность быстро оценить производительность локальных и глобальных сетей. Эта система более интеллектуальна, помогает планировать изменения, прогнозирует результат, генерирует отчеты и графики о производительности сети, определяет, какие ресурсы создают задержку в сети, а какие недоиспользуются. На основе импортированных данных по топологии, протоколам и трафику пользователю предоставляется возможность изменить такие параметры, как топология, трафик, состав оборудования, полоса пропускания, протоколы и быстро получить результат в виде отчетных графических форм.

COMNET Baseliner – система импорта данных. Эта система в одном пакете объединяет возможности мониторинга и моделирования сети, в результате чего для моделирования автоматически появляется исходная информация, содержащая данные о базовом уровня сети. В системе есть различные опции фильтрования, могут комбинироваться данные о трафике, которые собраны различными инструментами мониторинга, в единый трафик модели. Эта система предназначена для импорта данных о топологии и протоколам из установленных у пользователя систем управления и мониторинга сетей с целью создания базовых моделей для пакетов COMNET III и COMNET Predictor.

COMNET Enterprise Profiler – система мониторирования сети. Позволяет производить мониторирование и сбор статистики в сети без возможности администрирования. Может интегрироваться с другими системами мониторирования и управления.

Кроме перечисленных систем, имеются ряд других разработок моделированию вычислительных систем и сетей, которые можно посмотреть по ссылке http://bibliofond.ru/view.aspx?id=33848 (29 декабря 2011г).

2.5. Основные направления и перспективы развития имитационного моделирования

Имитационное моделирование в настоящее время является одним из основных видов моделирования и исследования для экономических, производственных, экологических, демо- и энтографических систем, систем массового обслуживания и др. Оно заключается в создании модели-имитатора работы сложных (чаще всего при наличии стохастических факторов) систем и

процессов при неполных знаниях о ряде процессов в моделируемых объектах. Одними из наиболее эффективных методов исследования указанных объектов и систем являются методы имитационного и комплексного моделирования. Имитационное моделирование (ИМ) — это метод исследования, который основан на том, что анализируемая динамическая система заменяется имитатором и с ним производятся эксперименты для получения информации об изучаемой системе. Роль имитатора выполняет компьютерная программа или комплекс программ для ЭВМ, которая называется имитационной моделью.

В имитационном моделировании формируются тематические направления исследований [33]:

- теоретические основы и методология имитационного и комплексного моделирования;
- методы оценивания качества моделей и полимодельных комплексов;
- методы и системы распределенного моделирования;
- моделирование глобальных процессов;
- разработка средств автоматизации и визуализации имитационного моделирования;
- системная динамика (с обязательным наличием имитационной составляющей);
- практическое применение моделирования и инструментальных средств автоматизации моделирования, принятие решений по результатам моделирования;
- имитационное моделирование в обучении и образовании.

Анализ перечисленных ранее особенностей формального описания и исследования сложных объектов и систем показывает, что при моделировании и управлении данными объектами и системами следует базироваться на концепциях и принципах, положенных в основу современных технологий системного (комплексного) моделирования. Более того, как показывает анализ, при решении актуальных в современных условиях проблем структурно - функционального синтеза облика гибридных интеллектуальных систем управления (ГИСУ) СОТО целесообразно рассматриваемые технологии системного моделирования, традиционно связанные с количественными вычислениями, дополнить интеллектуальными информационными технологиями, ориентированными на символьную обработку информации. К указанным информационным технологиям принято относить [33]:

− технологии экспертных систем (Expert Systems) или систем, основанных на знаниях (Knowledge-Based Systems);

− технологии нечёткой логики (Fuzzy Logic);

− технологии искусственных нейронных сетей (Artificial Neural Networks);

− технологии вывода, основанного на прецедентах CBR (Case Based Reasoning) - технологии;

− технологии естественно-языковых систем и онтологии;

− технологии ассоциативной памяти;

− технологии когнитивного картирования и операционного кодирования;

− технологии эволюционного моделирования;

− технологии мультиагентного моделирования.

Несмотря на рост производительности современных компьютеров, их мощности не хватает для моделирования задач, связанных с самолето- и автомобилестроением, логистикой, сборочным производством и т. п., когда имитационные прогоны моделей могут длиться часами. Одним из вариантов решения этой проблемы является использование параллельного и распределенного дискретно-событийного моделирования. В этой области есть даже стандарт HLA – High Level Architecture.

Вопросам практического использования результатов имитационного и комплексного моделирования сложных процессов и систем уделяется особое внимание на многих конференциях, в частности на конференциях ИММОД [33].

Если говорить о направлениях построения и использования имитационных моделей, то они следующие:

1. Модели, предназначенные для выявления функциональных соотношений, т.е. для определения природы и закономерностей между двумя или несколькими факторами с одной стороны и откликом системы с другой стороны.
2. Модели, предназначенные для прогноза, т.е. для оценки поведения системы при некотором сочетании рабочих условий, в том числе и по времени.
3. Модели для экспертной оценки предлагаемой структуры или конфигурации системы по предлагаемым некоторым критериям или совокупностью аксиом, сформулированным с помощью экспертов. Эта совокупность является как бы основой разработанной экспертной системы (ядром) – имитационной модели.

4. Модели, позволяющие сопоставлять конкурирующие системы, рассчитанные на выполнение определенной функции, или сопоставление различных предлагаемых рабочих методик, принципов. Эти модели сравнения альтернатив разрабатываются для принятия решений, разрабатываются в виде утилиты, запускаются регулярно при принятии оперативных решений.
5. Модели для оптимизации в целях определения сочетания действующих факторов и их величин, при котором обеспечивается наилучший отклик системы в целом. Может использоваться как инструмент для оценки и сравнения вариантов предполагаемых изменений или выработки оптимальной стратегии. Эти модели широко применяются при решении задач управления (планирования, проектирования) и часто называются интерактивными оптимизационными моделями (системами).
6. Модели виртуальных игр, предназначенные для обучения студентов, управленческого персонала, сотрудников. Разработка моделей виртуальной реальности – инструментов для компьютерных имитационных игр для различных отраслей, военных, экономических, государственных и межгосударственных отношений.
7. Модели для анализа чувствительности, т.е. для выявления тех факторов, которые в наибольшей степени влияют на общее поведение системы.
8. Модели, встроенные в производственный процесс, в технические и автоматизированные установки, запускающихся автоматически при выполнении соответствующих операций.
9. Модели, созданные для динамической визуализации (демонстрации) проектируемого объекта в целях аргументации разрабатываемого проекта для руководства или потенциальных заказчиков (воспроизводящую виртуальную иллюзию процесса развития системы).

Говоря о моделировании как исследовательском методе, и имитационном (компьютерном) моделировании как его разновидности, необходимо остановиться на основных областях применения, в которых могут решаться отраслевые задачи при помощи данного метода, о его преимуществах и особенно – ограничениях.

Активнее всего имитационным моделированием интересуются в металлургии, нефтегазовой отрасли, производстве стройматериалов, пищевых

продуктов, в различных областях массового обслуживания (аэропорты, транспорт, медицина, торговые предприятия, сферы обслуживания и др.).

Анализ результатов конференций ИММОД показывает, что сводный перечень основных областей применения выглядит следующим образом [33]:

- предоставление услуг связи, сети передачи данных;
- управление подвижными (космическими) объектами, воздушным и автотранспортным движением;
- организация промышленного производства (ГПС, обувное, мелкосерийное, сборка персональных компьютеров, разработка программного обеспечения);
- проектирование рыбообрабатывающих комплексов на судах промыслового флота;
- космическая сварка;
- информационное противоборство, блочное шифрование;
- динамика популяции зверей и животных;
- региональные экономические системы;
- лечебно-эвакуационные мероприятия в авиадивизии;
- налоговое и пенсионное законодательство, обращение граждан в органы государственного управления;
- подготовка специалистов по управлению железнодорожным транспортом и магистральными трубопроводами;
- исследования в области экосистем (водных, морских, лесных, степных и др.);
- исследования демографических и этнических систем.

Применение имитационного моделирования достаточно медленно развивается из-за сложности рассматриваемых объектов и трудоемкости создания имитационных моделей, из-за отсутствия профессиональных кадров в области имитационного моделирования и ряду других причин. Особенно сложно обстоит дело в областях социальных и гуманитарных наук, где слабое применение методов имитационного моделирования может быть объяснено из-за причин, изложенных выше, нелинейностью и многофакторностью социальных процессов, сложностью взаимосвязей, а также из-за необоснованных ожиданий, возлагаемых на модель, которые не могут оправдаться в реальности. Тем не менее, формализацией социальных систем в настоящее время занимаются, определяются математические подходы и получены некоторые результаты [3,12-15].

Появление новых современных программных продуктов существенно снижает требования к разработчику модели, и открывают для специалистов широкого профиля, не обладающих навыками программирования, возможность разработки моделей, в том числе и для достаточно сложных систем. В то же время увеличивает требования к постановщику задач. Вышесказанное предопределяет чрезвычайно широкие возможности по применению методов имитационного моделирования при изучении социальных явлений, образовательной деятельности, при обучении управленческих кадров и т.д.

Модель представляет собой упрощенное отображение реальности – это менее детальное, менее сложное, менее подробное воспроизведение реально существующего объекта или феномена. При этом, когда мы говорим о моделировании социальных феноменов, по сути, модель представляет не просто упрощение реальности, а отображение реальности через призму определенного теоретического подхода, или мнения эксперта.

В социальных науках чаще всего используется статистическое моделирование как разновидность имитационного. Здесь при разработке модели решаются задачи формализации экспертного знания и различных теоретических концепций, при разработке модели максимально исчерпывающе описываются положения теоретической концепции или результаты экспертного анализа (например, в форме мозгового штурма, социологического экспертного опроса и проч). Сама по себе подобная формализация является важным результатом, и моделирование в этом свете может рассматриваться в качестве своеобразного формального языка и выступать в качестве определенного аналога математики для естественных наук. Результатом применения статистического моделирования является возможность проведения прогноза. Прогнозирование можно считать одним из наиболее ценных приложений имитационного моделирования.

Одним из направлений разработки компьютерных моделей является создание игровых имитационных моделей своеобразных тренажеров, позволяющих проводить обучения по психологическим, педагогическим, управленческим и др. ситуациям. В основе данных тренажеров лежит имитационная модель, которая позволяет обучающимся применять определенный набор инструментов воздействия и управления и воспроизводит реакцию социальной системы на соответствующее управляющее воздействие. Направление разработки такого рода компьютерных моделей часто называют ситуационным моделированием.

Тренажеры такого рода (виртуальные игровые имитационные модели) уже получили широкое распространение в зарубежной практике управленче-

ского образования, где они известны как искусственное общество (artificialsociety), микромир (microworld). В российском управленческом образовании использование игр на основе имитационных моделей только начинается, однако, на наш взгляд, это одно из основных направлений его качественного развития [33] .

Контрольные вопросы

1. Что такое аналитическая модель и как проводится исследование аналитической модели?
2. В чем сущность имитационного моделирования?
3. В каких областях может быть использовано имитационное моделирование?
4. Каковы этапы аналитического и имитационного моделирования?
5. Каковы основные направления создания и средств разработки компьютерных моделей и программного обеспечения для моделирования?
6. Особенности разработки интерактивных компьютерных моделей для электронных обучающих систем?
7. Какова технология разработки сетевых компьютерных систем и комплексов для моделирования реальных объектов?
8. Какие инструментальные средства моделирования используются для построения аналитических компьютерных моделей?
9. Какие системы имитационного моделирования позволяют строить компьютерные модели сложных систем в разных областях производства, логистики, транспорта, в общественных и социальных науках и других отраслях?
10. Какие системы автоматизации используются для моделирования и проектирования локальных и глобальных сетей?
11. Какие тематические направления и области применения имитационного моделирования существуют на сегодняшний день?

ГЛАВА 3. ОСНОВНЫЕ ПОДХОДЫ К РАЗРАБОТКЕ КОМПЬЮТЕРНЫХ МОДЕЛЕЙ СЛОЖНЫХ СИСТЕМ

3.1. Понятие о компьютерной модели

Компьютерная модель (англ. computer model), или численная модель (англ. computational model) — это (1) компьютерная программа, работающая на отдельном компьютере, суперкомпьютере или множестве взаимодействующих компьютеров (вычислительных узлов), реализующая абстрактную модель некоторой системы; это (2) модель, выполненная с помощью компьютерных информационных, схематичных, электронных устройств и технологий и сетей; это (3) созданный за счет ресурсов компьютера виртуальный образ, качественно и количественно отражающий внутренние свойства и связи моделируемого объекта, иногда передающий и его внешние характеристики; это (4) модель, воспроизводящая моделируемый объект программными средствами на компьютере.

Разработке компьютерной модели предшествуют мысленные, вербальные, структурные, математические и алгоритмические модели.

Компьютерные модели подразделяются на аналитические и имитационные. Компьютерные модели различаются по видам применения: обучающие, научно-исследовательские, научно-технические для исследования процессов и явлений, реальных объектов.

Компьютерные аналитические и имитационные модели, разрабатываемые для прикладных применений, например, различные компьютерные игровые модели для общества, бизнеса, производства подразделяются на деловые, экономические, военные, образовательные. С помощью таких моделей можно разрешать конфликтные ситуации, оказывать психологическую помощь, проигрывать поведение объекта в различных ситуациях.

Имитационные модели не просто отражают реальность с той или иной степенью точности, а имитируют ее. Эксперимент с моделью либо многократно повторяется при разных исходных данных, чтобы изучить и оценить последствия каких-либо действий на реальную обстановку, либо проводится одновременно со многими другими похожими объектами, но поставленными в разные условия.

В настоящее время компьютерные модели подразделяются условно на два вида:

- структурно-функциональные, которые представляют собой условный образ объекта, описанный с помощью программных и компьютерных технологий;

- имитационные, представляющие собой программу или комплекс программ, позволяющий воспроизводить процессы функционирования объекта в разных условиях.

Существует множество программных комплексов, которые позволяют проводить построение и исследование моделей (моделирование). Каждая программная среда имеет свой инструментарий и позволяет работать с определенными видами информационных моделей. Поэтому перед исследователем возникает нелегкий вопрос выбора наиболее удобной и эффективной среды для решения поставленной задачи. Надо сказать, что одну и ту же задачу можно решить, используя различные среды.

От выбора программной среды зависит алгоритм построения компьютерной модели, а также форма его представления.

Например, это может быть блок-схема. Руководствуясь блок-схемой, задачу можно решить в разных средах. В среде программирования – это программа, записанная на алгоритмическом языке. В прикладных средах – это последовательность технологических приемов, приводящая к решению задачи.

3.2. Языки и инструментальные системы программирования

Языков программирования достаточно много, для разработки программ моделирования могут быть использованы как языки общего назначения (ЯОМ), так и языки имитационного моделирования (ЯИМ) [13].

К языкам общего назначения можно отнести следующие языки программирования:

- **языки императивного программирования** здесь манипулируют данными в пошаговом режиме, используя последовательности из нескольких низкоуровневых инструкций, или команд. Исходя из того, что программа, по сути, представляла собой набор директив, обращенных к компьютеру: Fortran (1954), ALGOL(1960), COBOL(1960), Pascal (1970), C (1972), Basic (1963) и их модификации;
- **языки функционального программирования –** это языки, в которых единственным действием является вызов функции, единственным способом расчленения программы на части является введение имени для функции, а единственным правилом композиции – оператор суперпозиции функции. Отличительной особенностью функционального подхода является то, что любая программа, написанная на таком языке,

может интерпретироваться как функция с одним или несколькими аргументами. Такой подход дает возможность прозрачного моделирования текста программ математическими средствами, а значит, весьма интересен с теоретической точки зрения. Сложные программы при таком подходе строятся посредством агрегирования функций. При этом текст программы представляет собой функцию, некоторые аргументы которой можно также рассматривать как функции. Таким образом, повторное использование кода сводится к вызову ранее описанной функции, структура которой, в отличие от процедуры императивного языка, математически прозрачна. К языкам такого типа относятся: Lisp (1958), РЕФАЛ (1968), Scheme (1975), FP(1977), ML (1978), Miranda (1985), Standart ML(1985), Haskell (1990, 1998), F#(2010);

- **языки логического программирования.** Программа строится в виде совокупности правил или логических высказываний. Кроме того, в программе допустимы логические причинно-следственные связи, в частности, на основе операции импликации. Эти языки программирования базируются на классической логике и применимы для систем логического вывода, в частности, для так называемых экспертных систем. На языках логического программирования естественно формализуется логика поведения, и они применимы для описаний правил принятия решений, например, в системах, ориентированных на поддержку бизнеса. Примеры: Prolog (1971), Prolog II (1980), IC-Prolog (1982), LogLisp (1982), LCA 91982), M-Prolog (1983), T- Prolog (1983), Concurrent Prolog (1983), PARLOG (1983), LQF (1984), LEAF (1985), GHC (1986), Goedel (1992), Mercury (1993);
- **языки объектно-ориентированного программирования** содержат конструкции, позволяющие определять объекты, принадлежащие классам и обладающие свойствами инкапсуляции, наследования и полиморфизма. В рамках данного подхода программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов). Примеры: Simula (1962), SmallTalk (1972), Beta (1975), C++(1983), Object Pascal (1984), Self (1986), Cecil (1992), Java (1995), C# (2000), Visual Basic, GPSS.

Кроме этих языков созданы языки программирования специализированные для некоторой предметной области, чаще всего используемые для разработки пользовательского интерфейса.

- **Языки форматирования текстов** для издательских систем, систем подготовки научных публикаций с большим количеством формул. Примеры: TeX, LaTeX, Troff, Nroff.
- **Языки разметки** для универсальных форматов представления структурированных данных. Примеры: SGML, KML, XML, HTML, MathML.
- **Языки скриптов или сценариев** для разработки командных пакетных файлов для командных интерпретаторов. В рамках данного подхода программа представляет собой совокупность возможных сценариев обработки данных, выбор которых инициируется наступлением того или иного события (щелчок по кнопке мыши, попадание курсора в определенную позицию, изменение атрибутов того или иного объекта, переполнение буфера памяти и т.д.). События могут инициироваться как операционной системой (в частности, Microsoft Windows), так и пользователем. Примеры: Tcl/Tk, VBScript, PowerScript, LotusScript, Javascript, Perl, Sh, Bash, Csh, Ksh.
- **Языки создания графики** для подготовки визуальной графической информации. Пример: PostScript, OpenGL.
- **Языки описания виртуальной реальности** для создания трехмерных изображений, так называемых виртуальных миров. Примеры: VRML, Dynamo и другие.

Хронологию, списки языков программирования и их сравнительный анализ можно получить из Википедии по адресу http: // ru.wikipedia.org/ wiki/ Сравнение_ языков_ программирования.

Системой программирования будем называть весь комплекс программных средств, предназначенных для кодирования, тестирования и отладки программного обеспечения и прилагаемые к ним руководства по использованию.

В состав системы программирования входят следующие программные инструменты и библиотеки:

- **текстовый редактор** – это редактор, который позволяет набрать текст программы на языке программирования;
- **транслятор** –программы, обеспечивающие перевод исходного текста программы на машинный язык (объектный код);
- **редактор связей (сборщик)** – это программа, которая объединяет объектные модули отдельных частей программы и добавляет к ним стандартные модули подпрограмм стандартных функций (файлы с расширением ***.lib***), которые содержатся в библиотеках, поставляемых вместе

с компилятором, в единую программу, готовую к исполнению, т.е. создает *исполнимый* ***.exe*** файл;

- **отладчик** для проверочных для проверочных запусков программ и исправления ошибок;
- **библиотеки** стандартных функций для периода отладки и периода исполнения.

Классификацию систем программирования можно провести по ориентации на поддержку процессов.

1. Инструментарий поддержки технологических процессов:

- «процесс-ориентированный» инструментарий, поддерживающий определенный технологический процесс. Например, системы информационной поддержки управления проектами в основном связаны с планированием и управлением проектом. Примеры: MS Project (корпорация Microsoft), Manage Pro (компания Performance Technology Solutions, LLC), Rational Plan Multi (компании Rational Software Corporation), Primavera Enterprise (компании Oracle) и др.
- «процесс-независимый» инструментарий, который можно использовать в нескольких технологических процессах. Например, системы формального преобразования и верификации программ: TXL, IntelliJ Renamer, RescueWare.

2. Инструментальные системы разработки программных продуктов, эти системы называют *средами быстрого проектирования*, в которых программирование, по сути, заменяется проектированием, чаще всего визуальным. В проектируемое окно готовые визуальные компоненты перетаскиваются с помощью мыши, затем свойства и поведение компонентов настраивается с помощью редактора. Текст программы, ответственный за работу этих элементов, генерируется автоматически с помощью *среды быстрого проектирования*, которая называется ***RAD***-средой. Подобный подход иногда называется *визуальным программированием*. Здесь выделяют четыре группы инструментальных систем:

- инструментальные среды программирования. Примеры Visual Studio (Visual Basic, Visual C++, Visual C#) – компании Microsoft, Forte for Solaris Developer Tools (компании Sun Microsystems Inc.), Delphi Suite и Java JBuilder (компании Borland International), AnyLogic (компании XJ-Technologies) и другие;
- средства автоматизации проектирования программ (CASE-средства) - инструментарий для системных аналитиков, разработчиков и про-

граммистов, позволяющий автоматизировать процесс проектирования и разработки программного обеспечения. Примеры: Oracle Designer (компании Oracle), Bpwin, Erwin (компании Computer Associates International Inc.), Rational Rose (компании Rational Software Corporation) и другие;

- инструментарий поддержки коллективной разработки ориентирован на решение проблем разделения технологических ресурсов, включая управление файлами и каталогами и необходимости синхронизации времени и места. Примеры для Windows: Perforce (компания Perforce Software), Rational Clear Quest (компания Rational Software Corporation), Revision Control System, Concurrent Versions Systems. Примеры для Unix: Source Code Control System, Sync tool, Bringover/Putback.

3.3. Общие подходы к разработке компьютерных моделей

В процессе создания компьютерной модели полезно разработать удобный графический интерфейс, который позволит визуализировать формальную модель, а также реализовать интерактивный диалог человека с компьютером на этапе исследования модели.

В настоящее время существуют множество вариантов выбора методов и технологий разработки программ. При разработке программного обеспечения для моделирования и создании компьютерной модели применяются следующие технологии:

– разработка программного обеспечения (компьютерной модели) на основе языков и систем программирования;

– разработка компьютерных моделей на основе систем и пакетов компьютерного моделирования;

– разработка компьютерной модели с использованием электронных таблиц или других приложений: систем компьютерного черчения, систем управления базами данных, геоинформационных систем и т. д.

В зависимости от области и принадлежности объекта моделирования и языка разработки программного кода применяют различные подходы к решению задачи:

- ранние неструктурные подходы, называемые в отечественной литературе «стихийным программированием»;
- структурный или модульный подход (задача разбивается на подзадачи, затем на алгоритмы, составляются их структурные схемы и осуществляется реализация);

- функциональный подход;
- логический подход;
- объектно-ориентированный подход;
- смешанный подход (некоторые подходы можно комбинировать);
- компонентно-ориентированный (программный проект рассматривается как множество компонент, такой подход принят, в частности, в .NET);
- чисто объектный подход (идеальный с математической точки зрения вариант). Этот подход частично реализован в системе компьютерного моделирования MultiSim, для конструирования и построения электро-радиотехнических схем.

При разработке имитационных моделей мы имеем дело с реальной системой, представляющей совокупность взаимодействующих подсистем и элементов, функционирующих во времени. Такой реальный объект имеет сложную структуру и может быть описан тройкой множеств

$$F = F\langle A, S, T\rangle, \qquad (3.1)$$

где A – множество элементов, включая элементы внешней среды, S – множество допустимых связей между элементами (структура модели), T – множество рассматриваемых моментов времени.

При построении имитационных моделей необходимо:

– алгоритмически описать функционирование отдельных элементов;

– представить реальную систему (процесс) как совокупность взаимодействующих элементов;

– описать процесс взаимодействия различных элементов между собой и с внешней средой.

Особое внимание надо уделить описанию состояний системы. Состояние системы описывается набором переменных, изменение этого набора свидетельствует о переходе в другое состояние, тогда имитационная модель представляет собой описание динамической последовательности изменения состояния системы согласно определенным операционным правилам. Эти переходы из одного состояния в другое могут произойти либо непрерывно, либо в дискретные моменты времени.

Анализ процессов функционирования системы позволяет провести условную классификацию подходов к моделированию и выделить основные математические схемы моделирования (см. таблицу 3.1).

В качестве детерминированных моделей, когда при исследовании случайный факт не учитывается, для представления систем, функционирующих

в непрерывном времени, используются дифференциальные, интегральные и др. уравнения, а для представления систем, функционирующих в дискретном времени — конечные автоматы и конечно разностные схемы.

Таблица 3.1.

Математические схемы моделирования

Процессы функционирования системы	Типовая математическая схемы	Обозначение
Непрерывно детерминированный подход	Стандартные дифференциальные уравнения	D -схема
Дискретно - детерминированный подход	Конечные автоматы	F-схема
Дискретно – стохастический подход	Вероятностные автоматы	P-схема
Непрерывно – стохастический подход	Системы массового обслуживания	Q-схема
Комбинированный подход	Агрегативная система	A-схема
Подход сетей Петри	Динамические дискретные системы	N - схема

В начале стохастических моделей (при учёте случайного фактора) для представления систем с дискретным временем используются вероятностные автоматы, а для представления систем с непрерывным временем — системы массового обслуживания (СМО). Большое практическое значение при исследовании сложных индивидуальных управленческих систем, к которым относятся АСУ, имеют так называемые агрегативные модели [21].

Для моделирования динамических дискретных систем (преимущественно асинхронных параллельных процессов) используется математический аппарат сетей Петри. Как и в системах массового обслуживания, в сетях Петри вводятся объекты двух типов: динамические — изображаются метками (маркерами) внутри позиций и статические — им соответствуют вершины сети Петри. Развитие теории сетей Петри проводилось по двум направлениям. Формальная теория сетей Петри занимается разработкой основных средств, методов и понятий, необходимых для применения сетей Петри. Прикладная теория сетей Петри связана главным образом с применением сетей Петри к моделированию систем, их анализу и получающимся в результате этого глубоким проникновением в моделируемые системы.

Моделирование в сетях Петри осуществляется на событийном уровне. Определяются, какие действия происходят в системе, какое состояние предшествовали этим действиям и какое состояние примет система после выполнения действия. Выполнение событийной модели в сетях Петри описывает

поведение системы. Анализ результатов выполнения может сказать о том, в каком состоянии пребывала или не пребывала система, какое состояние в принципе не достижимо.

3.4. Подходы к разработке аналитических компьютерных моделей на основе языков и систем программирования

Большое значение при разработке компьютерной модели на ЭВМ имеет вопрос правильного выбора языка или системы программирования. Язык программирования должен отражать внутреннюю структуру понятий при описании широкого круга понятий. Высокий уровень языка моделирования значительно упрощает программирование моделей.

Основными моментами при выборе языка моделирования являются:

- проблемная ориентация;
- возможности сбора, обработки, вывода результатов;
- быстродействие;
- простота отладки;
- доступность восприятия.

Рассмотрим подход, который применяется при разработке компьютерных численных моделей для дифференциальных уравнений и при применении метода Монте-Карло. Здесь чаще всего используются языки и системы императивного программирования (Pascal, Fortran, пакеты Visual Studio, среды Delphi Suite и др.).

Процесс разработки программы математического моделирования разбивается на несколько последовательных, следующих друг за другом этапов. В зависимости от сложности объекта моделирования и набора задач, требующих решения при создании программ моделирования, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

На этапе реализации программного обеспечения выполняется разработка программных модулей – программирование, иначе говоря, создание программного кода, которое заключается:

- в разработке блока программ управления функционированием системы;
- в разработке блока программ, реализующих расчетные формулы и функциональные алгоритмы (ядро программы);
- в разработке блока обработки результатов моделирования.

Программисты разрабатывают по системным спецификациям схемы программ и программные спецификации, затем пишут и отлаживают программы на одном из языков программирования; проводят отладку комплексов программ по модулям и задачам.

Подход к разработке программных компьютерных установок на основе языков и сред программирования чаще всего используется при построении компьютерных аналитических и численных статистических (имитационно-вероятностных) моделей для научно-исследовательских целей.

Под аналитическими компьютерными моделями мы понимаем программный продукт для решения интегро-дифференциальных уравнений численными методами. Под статистическими численными моделями мы будем понимать созданные компьютерные модели с применением теории статистических испытаний (метода Монте-Карло). Эти компьютерные модели могут создаваться как с помощью языков программирования общего назначения, так и с помощью систем компьютерной математики.

Этапы создания и моделирования аналитических и численных статистических моделей хорошо описаны и могут быть представлены в виде схемы (рис.3.4) .

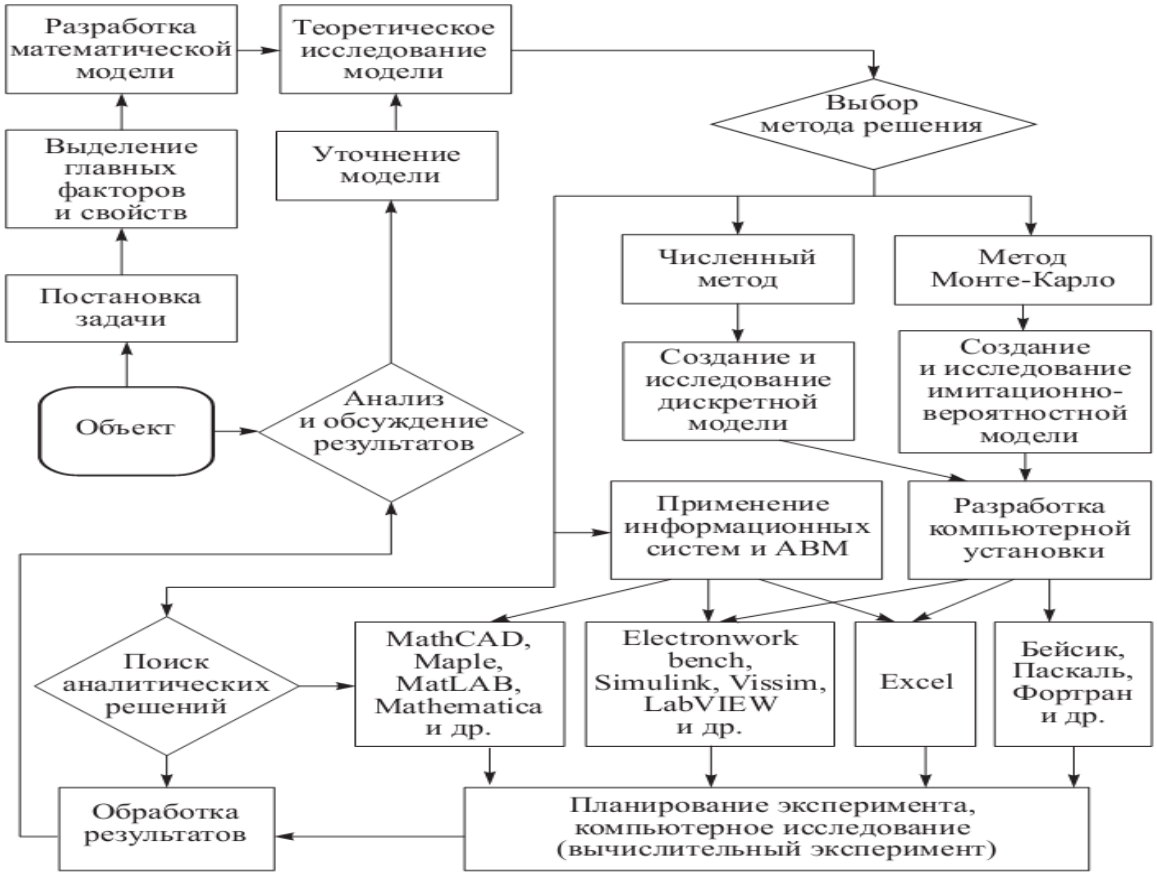


Рис. 3.4. Схема аналитического и компьютерного моделирования

В первой главе мы рассмотрели основные стадии и этапы в стадиях разработки программного обеспечения.

1. Предпроектная стадия – стадия формирования требований к автоматизированной системе.
   Этап разработки концепции автоматизированной системы.
   Этап разработки и утверждения технического задания.
2. Стадия проектирования и разработки программного обеспечения
   Этап разработки эскизного проекта автоматизированной системы.
   Этап разработки технического проекта.
   Этап проектирования программного обеспечения.
   Этап проектирования интерфейса.
   Этап реализации программного обеспечения (создание программного кода).
   Этап создания и оформления документации.
3. Стадия внедрения.
4. Период сопровождения или пользовательский период.

Эти стадии создания автоматизированных систем или программного продукта применимы также для разработки компьютерных установок для моделирования.

Здесь мы рассмотрим некоторые рекомендации, которые могут быть использованы разработчиками при создании программного обеспечения для моделирования (компьютерных моделей) с помощью языков и сред программирования.

Спецификация, алгоритмизация и кодирование программы. Разработка или проектирование программы не связано с тем языком, на котором будет окончательно записана конечная программ. В начале работы проводится определение спецификации процесса разработки. Спецификация процесса обычно представляется в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси-Шнайдермана. Существует ряд методик разработки алгоритма программы (методики проектирования программы). Эта работа с использованием спецификаций называется алгоритмизацией:

1) пошагового последовательного описания алгоритма решения задачи, описанного на любом алгоритмическом языке, в том числе текстовом языке;
2) разработки блок-схемы, достоинство которой заключается в том, что она не требует какой-то определенной детализации алгоритма и поэтому может использоваться на любых этапах разработки программ. Отметим, что

блок- схемы характеризуются хорошей наглядностью структуры алгоритма. Недостаток в том, что для сложных алгоритмов они становятся громоздкими. Кроме того их надо тщательно чертить, а также они неудобны для публикаций;

3) методика псевдокода. Каждый программист может придумать специальные символы или словесное описание каждого шага. В частности комбинация операторов паскаля и словесного описания задачи (подзадачи) в виде комментариев. Это прием позволяет получить хорошо прокомментированную предварительную программу;
4) проектирование программы моделирования с использованием специальных программ проектирования информационных систем и программных решений, например, пакет Bpwin, Erwin, Design/IDEF или Microsoft Solutions Framework в Visual Basic, Oracle Designer, Rational Rose, на которых можно спроектировать частично или весь алгоритм решения задачи.

После того, как алгоритм задачи спроектирован на алгоритмическом языке или псевдокоде, или в виде блок-схемы, начинается перевод его в программный код на одном из компьютерных языков программирования (Pascal, Visual Basic, Fortran и др.). Отметим, что многие начинающие программисты или модельщики (модельеры) пытаются писать программу решения задачи или алгоритма решения сразу на языке кодирования без прохождения этапа проектирования и алгоритмизации, по так называемому подходу стихийного программирования. Такой подход к разработке программного комплекса будет свидетельствовать о неграмотности и непрофессионализме разработчика.

Методы модульного программирования. Разработку программного комплекса по частям (модулям) называют модульным программированием. Модуль представляет собой завершенный фрагмент программы, характеризуется одним входом, где задается определенный набор входных данных и одним выходом, где имеем набор определяемых выходных. Структура модуля как системы черного ящика приводит к функциональной завершенности и логической независимости и является самостоятельным программным продуктом. Примером модульного строения программного продукта может быть разработка ПП для научных исследований (см. рис. 2.8).

Метод нисходящего проектирования (программирования). Разработка эффективного алгоритма решения в значительной мере является искусством. Практика развития программирования выработала методику разработки программ – метод пошаговой детализации или так называемый принцип нисходящего проектирования. Идея заключается в сведении решения сложной

задачи к решению простых задач. На практике это выглядит так: сложную задачу разбивают на 3-4 подзадачи, а в проектируемой программе намечается соответствующее число блоков (частей программы).

Здесь решается только проблема определения функциональных назначений каждого блока (что этот блок делает?). Эти подзадачи могут разбиться на более мелкие подзадачи, т.е. можно рассмотреть второй шаг детализации. Программисты уверяют, что этот принцип позволяет проверить правильность работы каждого блока и соответственно скорректировать общую схему программы. Отметим, что если на каждом шаге детализации используется принцип структурного программирования, то это обеспечит хорошую структурированность программы в целом.

При разработке программного кода проектирование всегда начинается со строительства модульной структуры программы в виде дерева. При использовании метода нисходящей разработки проектируются и реализуются программы, начиная с модуля самого верхнего уровня – головного, далее разрабатываются модули первого нижнего подуровня, затем второго и т.д. Далее производится их поочередное тестирование и отладка в таком же нисходящем порядке.

Метод восходящего проектирования (программирования). При таком подходе, так же как при нисходящем программировании, вначале строится древовидная модульная структура программы. Проектирование и разработка программного комплекса начинается с построения модуля самого нижнего уровня, затем следующего верхнего и т.д. Здесь программирование идет всех модулей, начиная с нижнего до верхнего, затем идет поочередное тестирование этих модулей. При таком программировании часто спецификации модулей определяются не полностью, и отработка тестирования всех модулей требует дополнительного отладочного материала и времени.

Подход структурного программирования. Идея структурного программирования была получена из результатов наблюдения чтения и понимания текста человеком. Эти исследования показали, что текст быстрее воспринимается, если человек читает фразы в порядке их следования в этом тексте. Если по ходу чтения его будут отсылать на фрагменты текста, находящихся на других страницах, то это резко затрудняет восприятие и понимание читаемого текста. При чтении программ «скачки» по тексту проводятся операторами переходов, которые могут отсылать как вперед, так и назад, в любое место программы. Поэтому структурное программирование часто называют «программированием без GO TO», чем меньше мы их используем, тем лучше. Структурное программирование предполагает использование

ограниченного числа базовых конструкций. Спецификации структурного программирования могут быть представлены также в виде: блок схем, методики псевдокода, Flow-форм, диаграмм Насси-Шнейдермана и др. и с помощью них разработаны алгоритмы решения задач.

При структурном программировании различают три вида вычислительного процесса: линейный, разветвленный и циклический, т.е. в качестве основных конструкций принимаются следующие структуры (рис.3.4).

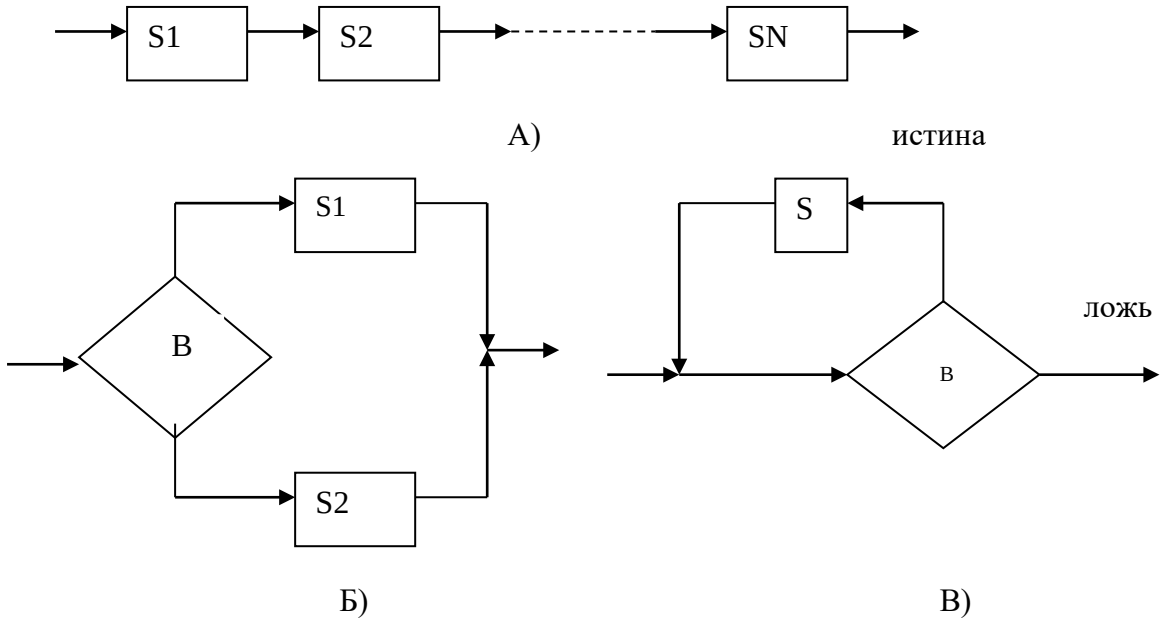


Рис.3.4. А) Следование. Эта структура представляет линейную последовательность блоков. Б) Ветвление. Это управляющая структура, которая в зависимости от выполнения заданного условия определяет выбор для исполнения одного из заданных в этой структуре блоков S1 и S3. В) Повторение типа «делать, пока». Эта структура представляет собой организацию цикла, выполняющуюся по условию

Здесь спецификация представлена в виде блок-схемы:

- следование;
- ветвление;
- цикл.

Отметим, что все структуры имеют один вход и один выход, это позволяет рекурсивное описание структуры, иначе говоря, каждый из блоков (прямоугольник) может быть не только отдельным оператором, но и любой из допустимых структур, т.е. допускается вложение структур.

Описание структурных алгоритмов можно представить в виде Flow-формы (рис.3.5) или в виде диаграммы Насси-Шнейдерамана (рис. 3.6)

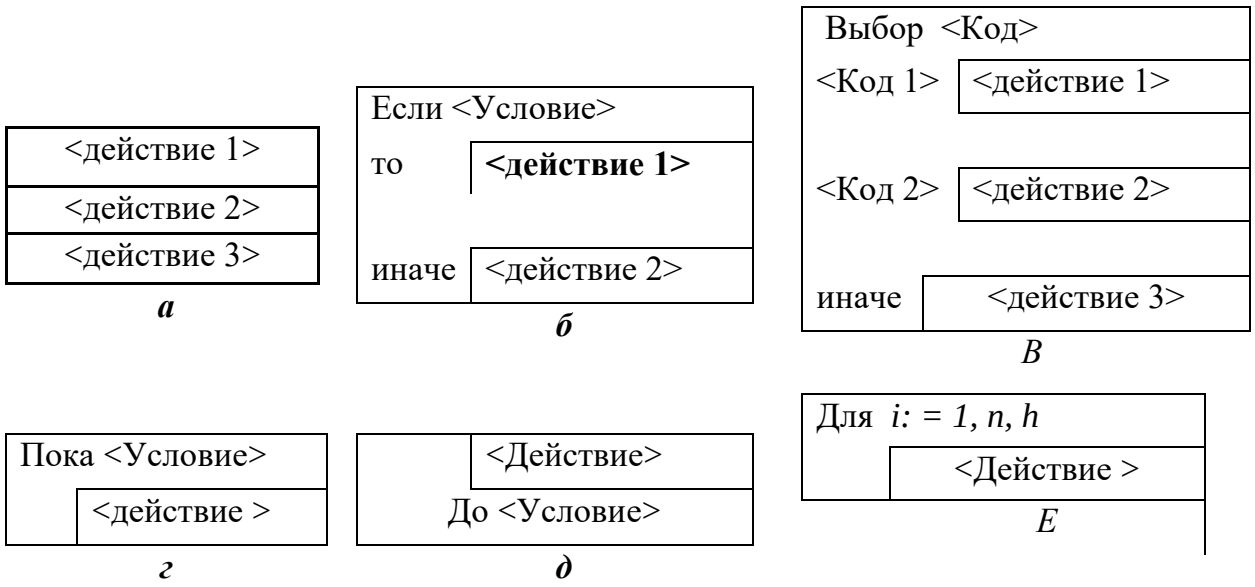


Рис. 3.5. Условные обозначения Flow- формы для основных конструкций: *а* – следование, *б* – ветвление, *в* – выбор, *г* - цикл- пока, *д* – цикл- до, *е* - счетный цикл

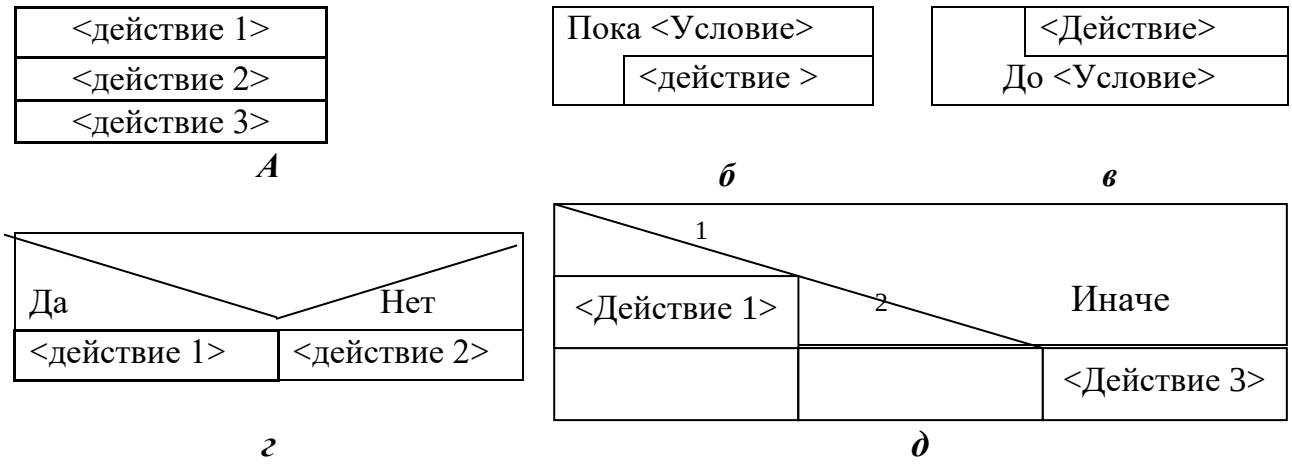


Рис. 3.6. Условные обозначения диаграмм Насси-Шнейдерамана для основных конструкций: *а* – следование, *б* – цикл - пока, *в* – цикл- до, *г* - ветвление, *д* – выбор

Каждый символ Flow-формы имеет вид прямоугольника и может быть вписан в любой внутренний прямоугольник любого другого символа.

При структурном подходе к программированию разрабатывают три типа моделей: модели функций, модели данных и модели потоков данных. Эти виды моделей представляются в виде следующих диаграмм:

- диаграмм потоков данных (DFD – Data flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе. Оно показывает, как каждый из процессов преобразует свои входные и выходные потоки данных и как эти процессы взаимодействуют ;

- диаграмм «сущность-связь» (ERD – Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области;
- диаграмм переходов состояний (STD – State Transition Diagrams), характеризующих поведение системы во времени;
- функциональных диаграмм (методика SADT), которые отражают взаимосвязи функций разрабатываемого программного обеспечения. Они создаются на ранних этапах проектирования систем для выявления составных частей и основных функций проектируемой системы.

О культуре оформления программ. Для наглядности чтения кода программы ядра используется расположение текста отдельных операторов по листу бумаги. А именно служебные слова, которые начинаются и заканчиваются тот или иной оператор записывают на одной вертикали, а все вложенные в него операторы записывают с некоторым отступом вправо (см. программу маятник Фуко).

Более эффективным средством облегчения чтения кода программы ядра, является ее комментирование. Эта возможность предусмотрена практически во всех языках программирования. Причем для профессионального программиста этот прием должен быть отработан до автоматизма. Начинающие программисты должны вырабатывать привычку комментировать программы кода по ходу их разработки и написания. Комментарии должны быть достаточно лаконичны, но так что при первом знакомстве с этой программой другой программист мог легко понять структуру программы, ее суть и логику ее работы, если просматривать содержащие в ней управляющие структуры и читать комментарии.

Отметим, что метод пошаговой детализации позволяет хорошо комментировать программу кода, если на первых шагах детализации назначению каждого выделяемого блока программы дается четкая, понятная словесная формулировка комментария.

«Некомментируемая программа – это, вероятно, наихудшая ошибка, которую может сделать программист, а также свидетельство делитантского подхода (пусть даже программист имеет десятилетний опыт работы); более того, это веская причина для увольнения программиста» [4].

Документирование программы. На первом этапе документирование программы ядра помещается в начале текста программы кода. Здесь задается общая информация о данной программе, содержащая следующие пункты:

1. Назначение программы.
2. Сведения об авторе программы.
3. Организация, в которой написана программа.
4. Дата написания программы.
5. Используемый метод решения задачи.
6. Указания по вводу и выводу.
7. Время, требуемое на выполнение программы.
8. Требуемый объем памяти.
9. Специальные указания оператору ЭВМ (в случае необходимости).

Полная документация на программное обеспечение для эксплуатации в виде описания применения, дающая общую характеристику программного продукта с указанием сферы применимости, технических требований к базовому программному обеспечению и характеристик ЭВМ, чаще всего излагается в технической документации или помещается в справке или помощи по меню интерфейса программного продукта.

Для оформления документов, обработки больших текстовых файлов, организации виртуальных трехмерных интерфейсов в Интернете, разработке баз данных используются языки HTML, Perl, Tcl/Tk, VRML, SQL, PL/SCL, Informix 4GL, Natural, DDL, DSDL, SEQUEL, QBE, IS и др.

3.5. Основные подходы к разработке имитационных моделей

Имитационная модель — это совокупность описания системы и внешних воздействий, алгоритмов функционирования системы или правил изменения состояния системы под влиянием внешних и внутренних возмущений. Эти алгоритмы и правила не дают возможности использования имеющихся математических методов аналитического и численного решения, но позволяют имитировать процесс функционирования системы и производить вычисления интересующих характеристик. Имитационная модель содержит элементы непрерывного и дискретного действия, поэтому может применяться для исследования динамических систем, когда требуется анализ узких мест, для исследования динамики функционирования, когда необходимо наблюдать на имитационной модели ход процесса в течении определенного времени.

В имитационном моделировании реальных объектов условно выделились четыре основных подхода: моделирование динамических систем, дискретно-событийное моделирование, агентное моделирование и системная динамика. В соответствии с этими направлениями реализуется непрерывный, дискретный, агентный подходы к моделированию и подход системной

динамики. В таблице 3.2 представлены основные подходы в имитационном моделировании.

Таблица. 3.2.

Подходы в имитационном моделировании

	Динамические системы	Системная динамика	Дискретно-событийное моделирование	Агентное моделирование
Процессы	Непрерывные	Непрерывные	Дискретные	Дискретные
Объекты	Переменные состояния	Накопители (неразличимы)	Заявки (пассивны)	Агенты (активны и индивидуальны)
Взаимодействие	Алгебро-дифференциальные уравнения	Потоки	Потоковые диаграммы Сети	Прямое и непрямое
Уровень абстракции	Низкий	Высокий	Средний	Любой

Анализ языков и средств автоматизации имитационного моделирования показывает, что существует множество программных продуктов, которые однозначно, а некоторые условно можно отнести к соответствующим подходам к имитационному моделированию (см. таблицу 3.3).

Таблица 3.3

Системы имитационного моделирования

Динамические системы	Системная динамика	Дискретное событийное моделирование	Агентное моделирование
Dynamo, PowerSim, MIMIC, АРТОН MIDAS, PACTOLUS, CSSL, СЛАМ, GASP, НЕДИС, МИКС, MATLAB+Simulink, Multisim VisSim, LabView, Easy5, MvStudium и др.	AnyLogic, Arena, SimBioSys, eM-Plant, Tecnomatix, Plant Simulation, SimuLab, Vensim, Powersim, Pilgrim, Dynamo, Stella, Ithink и др.	AnyLogic, Arena, Extend, Powersim Studio, Witness, ProModel, Pilgrim, Taylor Simulation, GPSS, SimScript, SIMULA, SIM-UL8, Modelling, SimProcess. AutoMod, Enterprise Dynamics, FlexSim и др.	AnyLogic, Swarm+MAML, SimAgent, SimBioSys, C++, Java, AgentSpeak, Oz, TeleScript, RePast, NetLogo, Ascape, Mason и др.

Системная динамика и дискретно-событийное моделирование – традиционные устоявшиеся подходы, агентное моделирование – относительно новый подход. Подход динамического моделирования позволяет увидеть по-

ведение модели во времени при движении в прошлое (для получения исторического результата) и в будущее (для выявления возможных исходов).

При изменении параметров модели можно выявить закономерности явления и находить оптимальные решения. Математически системная динамика и динамические системы оперируют в основном с непрерывными во времени процессами, тогда как дискретно-событийное и агентное моделирование – в основном с дискретными.

Из множества языков программирования можно выделить языки, относящиеся к имитационному моделированию. Их можно разбить на три основные группы, соответствующие трем типам математических схем: непрерывные (D-схемы), дискретные (Q-схемы) и комбинированные. Языки каждой группы предназначены для соответствующего представления системы *S* при создании ее компьютерной модели $K_м$.

В основе рассматриваемой классификации в языках имитационного моделирования лежит принцип формирования *системного времени.* Так как «системные часы» предназначены не только для продвижения системного времени в модели $К_м$, но также для синхронизации различных событий и операций в модели системы *S*, то при отнесении того или иного конкретного языка моделирования к определенному типу нельзя не считаться с типом механизма «системных часов». Существуют два основных способа изменения системного (модельного) времени:

1) по-шаговый (равномерный шаг времени). В этом способе используются фиксированные интервалы изменения модельного времени. Этот способ применяется: когда события распределены равномерно и можно подобрать шаг изменения временной координаты; когда сложно предсказать появление определенных событий; если закон изменения от времени описывается интегро-дифференциальными уравнениями и они решаются численными дискретными методами; когда событий очень много и они появляются группами;

2) по-событийный (неравномерный шаг времени). Здесь используются переменные интервалы изменения модельного времени, при этом величина шага измеряется интервалом до следующего события, применяется в тех случаях, когда все события распределены неравномерно по всей временной оси и появляются через значительные временные интервалы.

На рис.3.1 представлены языки и системы программирования для имитационного моделирования систем.

Рис.3.1. Языки и системы моделирования сложных систем

Непрерывный подход. Непрерывное представление системы *S* сводится к составлению уравнений, с помощью которых устанавливается связь между зависимыми и независимыми переменными модели. Примером такого непрерывного подхода является использование дифференциальных уравнений. Причем в дальнейшем дифференциальные уравнения могут быть применены

для непосредственного получения характеристик системы. Примеры: распространенным языком для имитационного моделирования непрерывных систем являются языки DYNAMO, POWERSIM, MIMIC, АРТОН, языки и среды императивного программирования.

Для моделирования непрерывных динамических систем получил распространение язык **CSMP**, который реализует пакетный режим взаимодействия с пользователем. Появились и другие языки и системы моделирования непрерывных процессов, такие как MIDAS, PACTOLUS, CSSL. К отечественным языкам и системам моделирования непрерывных динамических систем относятся МАСЛИН и МАСС (разработанные сотрудниками МЭИ).

Комбинированный подход. Представление системы *S* в виде типовой схемы, в которой участвуют как непрерывные, так и дискретные величины, называется *комбинированным.* Состояние модели системы *K(S)* описывается набором переменных, некоторые из которых меняются во времени непрерывно.

Законы изменения непрерывных компонент заложены в структуру, объединяющую дифференциальные уравнения и условия относительно переменных. Предполагается, что в системе могут наступать события двух типов:

1) события, зависящие от состояния системы;

2) события, зависящие от времени.

События первого типа наступают в результате выполнения условий, относящихся к законам изменения непрерывных переменных.

Для событий второго типа процесс моделирования состоит в продвижении системного времени от момента наступления события до следующего аналогичного момента.

Примеры языков для комбинированных систем: язык СЛАМ, объединяющий в одной оболочке языки GPSS и DYNAMO, т.е. позволяющий строить непрерывно-дискретные модели систем и «имеет практически неограниченные возможности». Примерами языков, реализующих комбинированное моделирование, являются GASP, НЕДИС и МИКС.

GASP является расширением языка ФОРТРАН. Здесь непрерывные алгоритмы моделируются дифференциальными уравнениями, а дискретные процессы представляются в виде событий, наступление которых зависит от процесса функционирования системы. Событие — переход системы из одного состояния в другое в соответствии с принятыми правилами.

НЕДИС — язык моделирования непрерывно-дискретных систем разработан сотрудниками Института кибернетики Академии наук Украины. НЕДИС создан на основе алгоритмических языков высокого уровня и отно-

сится к системам программирования универсального типа, т.е. языки GASP и НЕДИС относятся к процедурным языкам программирования.

МИКС (моделирование имитационное комбинированных систем) представляет собой удобное средство моделирования. Как и язык МАСЛИН, МАСС система МИКС имеет в своей основе блочно-ориентированный язык с непроцедурной технологией программирования, позволяющей легко и быстро моделировать исследуемую систему, осуществлять быстрое преобразование модели, воспроизводить реально действующие сигналы и организовать вычислительный эксперимент. Блочные языки и соответствующие программные модули позволяют легко реализовать динамическое распределение памяти посредством размещения во внешнее запоминающее устройство (ВЗУ) больших библиотек модулей, извлекать их по мере необходимости, пересылать их в оперативную память.

Для ***дискретного подхода*** можно выделить несколько принципиально различных групп языков имитационного моделирования (ЯИМ).

Первая группа ЯИМ подразумевает наличие списка событий, отличающих моменты начала выполнения операций. Продвижение времени осуществляется по событиям, в моменты наступления которых производятся необходимые операции, включая операции пополнения списка событий. Пример языка моделирования, ориентированного на транзакты (сообщения, заявки, запросы): Arena, Extend, ProModel, SimProcess, Pilgrim, Taylor, Witness, GPSS и др. Язык GPSS получил наиболее широкое распространение по сравнению с другими языками моделирования. Он включен в институтские учебные курсы по моделированию систем у нас в стране и изучается в аналогичных курсах во многих колледжах и университетах США и других стран.

При использовании ЯИМ второй группы после пересчета системного времени, в отличие от схемы языка событий, просмотр действий с целью проверки выполнения условий начала или окончания какого-либо действия производится непрерывно. Просмотр действий определяет очередность появления событий. Пример языка моделирования, ориентированного на события: SIMSCRIPT расширения языка Фортран.

Третья группа ЯИМ описывает системы, поведение которых определяется процессами. В данном случае под процессом понимается последовательность событий, связь между которыми устанавливается с помощью набора специальных отношений. Динамика заложена в независимо управляемых программах, которые в совокупности составляют программу процесса.

Пример языка моделирования, ориентированного на процессы: SIMULA, представляющий собой расширение языка АЛГОЛ.

Система Modelling является средством визуального проектирования дискретных и непрерывно-дискретных имитационных моделей (ДИМ и НДИМ). Главная область применения СИМ Modelling – разработка имитационных моделей и специализированных СИМ при изучении и проектировании сложных комплексных систем (систем массового обслуживания, АСОИУ и др.), где использование стандартных средств проектирования (MATLAB, GPSS и подобных) неудобно или неэффективно.

3.6. Подходы к разработке имитационных моделей сложных систем

Выясним основные признаки, которые приводят к решению разработки имитационной модели. Для этого рассмотрим начальный этап разработки проекта компьютерной модели. Здесь формулируется проблема, стоящая перед исследователями, и принимается решение о целесообразности применения метода имитационного моделирования, соответственно о его целевом назначении будущего проекта имитационной модели. Поставленные цели являются антиподами проблемы.

Разработка имитационной модели, на наш взгляд, тесно связана с признаком сложности представления системы. В этих целях введены понятия: «простая» и «сложная система». Определить различия между сложной и простой системой достаточно трудно. Понятие сложности может быть связано со сложностью алгоритма, отношений или сложностью поведения. Сложной можно считать систему, имеющую многоаспектную структуру связей или систему, в которой не достаточно информации для ее описания или систему, в которой большое количество элементов.

Анализ сложности рассматриваемого объекта предполагает применение теории систем и системного анализа, базовой частью которого является метод декомпозиции, предполагающий разделение целого (системы) на части (элементы): целей на подцели, задач на подзадачи, в конечном итоге этот подход приводит к построению древовидных структур – структуры сложной системы со всеми взаимосвязями и дерева целей.

Понятие «большая система» появилось при применении системного подхода и системных исследований для подчеркивания особенностей объектов и проблем. Отметим, что большие и сложные системы - это разные классы систем.

Программные системы являются разновидностью сложных систем. Разработка программной системы для простой и сложной системы отличаются своей масштабностью в объеме и времени. Поэтому тут используют метод декомпозиции сложности – разбивают на простые системы, используя идеологию: лучше решать много простых задач, чем одну сложную.

Суть одного из признаков программных сложных систем («издержка масштаба проекта») состоит в том, что чем крупнее программный проект, тем больше групп разработчиков задействованы в его реализации и тем больше информационных связей между группами. Нелинейный рост количества коммуникативных каналов приводит к нелинейному росту трудоемкости с увеличением размеров проекта. В зависимости от сложности системы подразделяют на легкие методологии разработки, ориентированные на решение локальных задач, и тяжелые методологии, ориентированные на создание сложных программных систем. Отличительные особенности методологии разработки малых и больших проектов приведено в таблице 3.4 [5].

Наука о сложных системах – системология определяет «сложную систему» как систему, способную принимать решения. Термин «сложная система» имеет многоаспектный и неопределенный характер, как и понятие «множество». К изучению теории систем и системного анализа сложных систем мы отправляем к авторам работ [1, 3, 5, 10,11, 17-18, 20, 22-25].

Таблица 3.4

Особенности разработки малых и больших проектов

	Малые проекты	Большие проекты
Стоимость внесения изменений	Невысокая	Высокая
Теснота связи с разработчиком	Сильная	Независимая
Время жизни ПП	Малое	Длительное
Стоимость разработки	Низкая	Высокая
Группа разработки	Небольшая	Большая
Период планирования	Короткий	Длительный
Требования к качеству	Низкие	Высокие
Получение результатов	За малое время	За большое время
Совершенствование проекта	Ограниченное	Длительное

Следующий признак, по которому мы должны применить подход имитационного моделирования, это наличие в рассматриваемой системе «сложных процессов». Под термином «сложный процесс» понимается такой процесс, прогноз которого в данный момент времени недоступен технологиям матема-

тического моделирования, иначе говоря, исследование рассматриваемого процесса лежит на грани возможностей математического моделирования. Другая характеристика сложного процесса – это протекающие процессы в системе. Эти процессы могут протекать одновременно, но быть разными по масштабу времени. В системе могут быть быстропротекающие и медленные процессы. Как учитывать эти процессы? Как организовать, учитывать влияние быстрого процесса на медленный процесс и наоборот? Выяснение влияний и анализ этих временных масштабов приводит к сложной и кропотливой работе по выбору или построению очередного способа или технологии создания имитационной модели.

Анализ признаков присутствия случайности и сложности системы показывает, что имитационная модель создается в тех случаях, когда система обладает некоторыми свойствами и качествами:

- присутствие в системе случайных внешних и внутренних характеристик;
- наличие в системе внешних управлений, которые должны задаваться извне эксплуататорами будущей имитационной модели для получения прогноза внутренних характеристик;
- система является сложной или громоздкой, и в ней реализуются сложные процессы;
- система настолько сложная, что модель приходится строить на грани возможности применения метода математического моделирования.

Таким образом, основными объектами имитационного и комплексного моделирования являются сложные процессы и системы, для которых характерно:

– комплексность и большая размерность (составных объектов и их параметров);

– иерархичность построения и наличие множества горизонтальных и вертикальных связей (в том числе и обратных);

– большая распределенность и удаленность объектов и систем в пространстве;

– наличие огромных потоков данных, информации и знаний, циркулирующих в контурах управления сложными объектами и системами.

В связи с этим, безусловно, возрастает роль и значение средств анализа функционирования и прогноза поведения сложных систем.

В них подобно исследуемым объектам и системам должны использоваться наиболее результативные теоретические подходы, они должны быть технологически современны, понятны и доступны исследователям.

Использование компьютерного имитационного моделирования для сложных систем позволяет:

определить внешние параметры изучаемого процесса или объекта;

– выявить закономерности, которые часто недоступны при наблюдении в естественных условиях;

– определить параметры, оптимизирующие протекание имитируемого процесса;

– выявить связи имитируемых явлений с заданными параметрами компьютерной модели и др.

Рассмотрим основные подходы к разработке имитационных моделей сложных систем [33-38].

Подход теории систем и системного анализа. Большинство изучаемых и подлежащих моделированию объектов являются сложными системами. Характерные признаки сложной системы – невозможность рассмотрения отдельно каждого элемента (без установления связей с другими элементами и внешней средой), неопределенность, проявляющаяся в большом числе возможных состояний системы, неопределенность достоверности исходной информации, разнообразие вариантов путей достижения конечной цели функционирования системы, адаптивность (приспосабливаемость системы к возмущающим факторам воздействия внешней среды). Эти особенности вызывают необходимость использования методологии системного анализа при создании имитационной модели сложного объекта. Для анализа сложных объектов и процессов рассматривают системные направления, включающие в себя следующие термины: системный подход, системные исследования, системный анализ. Эти термины часто употребляются как синонимы.

Системный подход. Этот термин начал применяться в первых работах, в которых элементы общей теории систем использовались для практических приложений. Заимствованные при этом понятия теории систем вводились не строго, не исследовался вопрос, каким классом систем лучше отобразить объект, какие свойства и закономерности этого класса следует учитывать при конкретных исследованиях и т. п. Иными словами, термин «системный подход» практически использовался вместо терминов «комплексный подход», «комплексные исследования».

Системные исследования. В работах под этим названием понятия теории систем используются более конструктивно: определяется класс систем,

вводится понятие структуры, а иногда и правила ее формирования и т. п. Это был следующий шаг в системных направлениях. В поисках конструктивных рекомендаций появились системные направления с разными названиями: системотехника, системология и др. Для их обобщения стал применяться термин «системные исследования». Часто в работах использовался аппарат исследования операций, который к тому времени был больше развит, чем методы конкретных системных исследований.

Системный анализ. В настоящее время системный анализ является наиболее конструктивным направлением. Здесь предлагается методология проведения исследования, делается попытка выделить этапы исследования и предложить методику выполнения этих этапов в конкретных условиях. В этих работах всегда уделяется особое внимание определению целей системы, вопросам формализации представления целей.

Системный анализ в широком смысле - это методология (совокупность методических приемов) постановки и решения задач построения и исследования систем, тесно связанная с математическим моделированием. В более узком смысле системный анализ – методология формализации сложных (трудно формализуемых, плохо структурированных) задач. Системный анализ возник как обобщение приемов, накопленных в задачах исследования операций и управления в технике, экономике, военном деле. Соответствующие модели и методы заимствовались из математической статистики, математического программирования, теории игр, теории массового обслуживания, теории автоматического управления. Системный анализ – это целенаправленная творческая деятельность человека, на основе которой обеспечивается представление исследуемого объекта в виде системы. Системный анализ характеризуется упорядоченным составом методических приемов исследования.

Системный анализ – более конструктивное направление, содержащее методику разделения процессов на этапы и подэтапы, систем на подсистемы, целей на подцели и т. д. В системном анализе выработана определенная последовательность действий (этапов) при постановке и решении задач, которую будем называть алгоритмом (методикой) системного анализа. Эта методика помогает более осмысленно и грамотно ставить и решать прикладные задачи. Если на каком-то этапе возникают затруднения, то нужно вернуться на один из предыдущих этапов и изменить (модифицировать) его. Если и это не помогает, то это значит, что задача оказалась слишком сложной и ее нужно разбить на несколько более простых подзадач, т. е. провести декомпозицию. Каждую из полученных подзадач решают по той же методике.

Метод системного анализа с успехом применяется к решению самых разных проблем в практически любой области – от проблем корпоративного управления и принятия управленческих решений, до моделей в области социологии, экономики, физики, информатики, биологии и т.п. Более того, в последние годы этот метод используется как один из основных подходов к анализу и построению структуры диссертационных исследований в любых отраслях.

В соответствии с методологией системного анализа существует определенная последовательность этапов моделирования.

1. Постановка задачи, формулировка и установление иерархии целей и подзадач. Изучение поведения системы в целом. Результатом этого этапа должно быть документированное содержательное описание объекта моделирования, иначе говоря, построение полной информационной модели.
2. Определение границ системы и внешней среды, т.е. выделение системы от окружающей среды. Определение входных параметров и выходных характеристик системы. Декомпозиция системы, определение и выделение основных компонент, элементов и подсистем (построение модели состава). Проведение анализа взаимосвязей подсистем, установление связей между компонентами, подсистемами, элементами (построение модели структуры связей). Принятие гипотез и допущений, физическая схематизация, иначе говоря, построение общей структуры системы с учетом всех подсистем и связей. Результатом деятельности разработчика компьютерной модели является создание полной концептуальной модели.
3. В зависимости от сложности объекта моделирования выбирается один из подходов имитационного моделирования (динамическая система, дискретно-событийное, мультиагентное, системная динамика). В рамках этого подхода проводится разработка математического описания объекта моделирования. Результатом этого этапа является разработка технического проекта компьютерной установки для моделирования.
4. Алгоритмизация и программная реализация, т.е. строится программный комплекс моделирования объекта исследования. Проводится отладка компьютерной имитационной модели.
5. Испытание, корректировка, проверка модели. Проводится верификация модели, комплексное тестирование компьютерной модели на адекватность объекту моделирования.

6. На этом этапе проводится стратегическое и тактическое планирование машинного эксперимента. Результатом является составленный план эксперимента и проведенный вычислительный эксперимент («прогоны» компьютерной модели с различными начальными данными).
7. Анализ результатов моделирования. Обработка, визуализация и интерпретация результатов машинного компьютерного эксперимента и использование или внедрение результатов исследования.

Статистический подход к разработке имитационной модели. Понятие «имитационное моделирование» первоначально появилось в теории случайных процессов и математической статистике как способ вычисления статистических характеристик случайных процессов путем многократного воспроизведения течения процесса с помощью модели этого процесса. Этот подход к исследованию реального процесса был назван методом статистических испытаний (методом Монте-Карло). Модели здесь строятся для явлений и систем объектов, входы и (или) функциональные соотношения между различными компонентами которой содержат элементы случайности или полностью случайных процессов, подчиняющиеся вероятностным законам. Реализация решения вероятностной модели реального объекта осуществляется на ЭВМ. Машинная имитация позволяет исследовать модель как в определенные моменты времени, так и в течение продолжительных периодов времени. Для нахождения устойчивых решений (характеристик) при численном статистическом моделировании требуется его многократное воспроизведение с последующей статистической обработкой. Здесь проводится имитация воздействия многочисленных случайных факторов на различные элементы модели. Каждое воздействие на процесс в модели представляется в виде «розыгрыша» случайного явления с помощью процедуры, дающей случайный результат. Множество таких реализаций в ходе одного варианта имитации дает одну реализацию (историю) процесса. Затем вычисляются средние статистические характеристики по многим историям.

Создание вероятностной модели, применение метода Монте-Карло для «имитации» и его решение на ЭВМ для определения характеристик и параметров исследуемого реального явления называют ***вероятностно-имитационным*** моделированием.

Подход разработки имитационных моделей на основе метода Монте-Карло является классическим подходом в молекулярной, статистической, нейтронной, квантовой физике, геофизике, газовой динамике, а также в теориях фильтрации, передачи и защиты информации и т.д. Разработанные модели чаще всего являются научно-исследовательскими моделями. При-

меры вероятностно-имитационных моделей можно рассмотреть в работах [2, 3].

Дискретно-событийный подход к разработке имитационных моделей. Этот подход широко используется в теории массового обслуживания, который изучает широкий класс случайных процессов в системах распространения информации, информационно-коммуникативных системах (компьютеры, интернет, связь и т.д.) и в различных отраслях массового обслуживания (железнодорожный, автомобильный транспорт, аэропорты, поликлиники, санаторные и лечебные учреждения, любые торговые предприятия, сферы обслуживания и др.). Приведем некоторые пояснения.

Система массового обслуживания (СМО) предназначена для обслуживания потока заявок, поступающих на канал обслуживания (прибор) в случайные моменты времени. Обслуживание заявки также длится случайное время. Процесс работы СМО является случайным с дискретными состояниями и непрерывным временем. Для стационарных и пуассоновских потоков существуют аналитические подходы (уравнения Колмогорова) к изучению этих процессов, позволяющих определять характеристики (среднее число заявок, среднее время обслуживания одной заявки, вероятности нахождения СМО в дискретных состояниях и др.) как одноканальных, так и многоканальных СМО. Если поток не пуассоновский, то получение аналитических формул не возможно, здесь используется «имитация» – использование ЭВМ для реализации случайного процесса СМО с последующей обработкой результатов методами математической статистики. Для моделирования реального процесса здесь создается некоторая программная установка, на которой проводятся вычислительные эксперименты.

Разработанные имитационные модели для отрасли массового обслуживания чаще всего используются непосредственно для анализа производственных задач.

Агрегативный подход к моделированию сложных систем Н.П. Бусленко. При системном подходе автоматизированные системы управления (АСУ) рассматривают как единую сложную систему совместно с управляющими подсистемами. Для обеспечения высокого качества управления необходимо хорошо знать свойства управляемых подсистем. Для того чтобы выявлять свойства управляемых подсистем, их реакцию на применяемые решения и мероприятия, а также оценивать качество принимаемых решений, необходимо использовать в работе АСУ результаты моделирования функционирования подсистем в тех или иных прогнозируемых условиях. Для реше-

ния этих задач можно применяют агрегатную модель Н.П.Бусленко. Агрегатная модель описывает объект управления в виде многоуровневой структуры из динамических систем заданных типов или агрегатов. При этом системы рассматриваются как обобщающий (самый общий и самый сложный) класс сложных систем и называются агрегативными. Агрегат используется для моделирования элементарных блоков сложных систем. Агрегативной системой называется любая совокупность агрегатов, если передача информации между ними происходит мгновенно и без искажений.

Агрегатом называется математическая модель вида:

$$A = \{T, Z, X, U, Y, H, G\}, \qquad (3.2)$$

где

T – интервал моделирования (обычно конечный);

Z – множество состояний (фазовое пространство);

X – множество входных сигналов;

U – множество управляющих (особенных) сигналов;

Y – множество выходных сигналов;

H – оператор переходов, который определяет текущее состояние по предыстории;

G – оператор выходов.

В общем случае все последовательности событий в агрегате являются реализациями случайных последовательностей с заданными законами распределения, оператор H также является случайным оператором.

Н.П. Бусленко при рассмотрении сложных систем выделяет два типа состояний:

1) обычные (неособые) состояния, в которых система находится почти все время;

2) особые состояния, характерные для системы в некоторые изолированные моменты времени, совпадающие с моментами получения входных и управляющих сигналов или выдачи выходного сигнала, в эти моменты состояние агрегата может измениться скачкообразно, а между особыми состояниями изменение координат происходит плавно и непрерывно.

Агрегативная система называется комплексом, если любой агрегат в ней соединен хотя бы с одним агрегатом системы. Агрегативная система называется m–фазной, если состоит из m последовательно соединенных комплексов. Агрегативная система называется m–канальной, если состоит из m параллельно соединенных комплексов. Агрегативная система называется строго иерархичной, если в ней можно выделить подчиненные и

управляющие комплексы, то есть если существует (управляющий) комплекс, входная информация которого служит управляющей информацией другого (подчиненного) комплекса.

Агрегат представляет собой математическую схему общего вида, частным случаем которой являются функции алгебры логики, релейно-контактные схемы, конечные автоматы, динамические системы, описываемые обыкновенными дифференциальными уравнениями и ряд других.

Модель агрегата может быть использована как модель всей непрерывно-дискретной системы или ее элемента. В этом случае система представляется сетью агрегатов с фиксированными каналами связей.

Среди агрегатов выделяются некоторые подклассы, для которых возможно более простое решение задачи моделирования. Классификация агрегатов определяется по виду оператора U. Например, кусочно-линейным агрегатом называется агрегат, у которого пространство состояний Z определенным образом структурировано и оператор U линейный.

К типичным примерам кусочно-линейных агрегатов можно отнести вероятностный автомат, математическую модель системы массового обслуживания или систему ОДУ, представленную конечно-разностными уравнениями.

Моделирование поведения агрегата и агрегативной системы заключается в построении последовательности переходов из одного особого состояния в другое, причисляя к множеству особых состояний z (0). Такой подход к моделированию, предложенный Н.П.Бусленко и основанный на принципе ”особых состояний”, фактически являлся первой попыткой учета дискретности в методах исследования непрерывно–дискретных систем.

Подход В.М. Глушкова для моделирования непрерывно-дискретных систем. Формализм описания включает в себя математическую модель непрерывно-дискретной системы, язык спецификации, а также набор процедур и функций реализации моделирующего алгоритма. В противоположность агрегативному подходу, моделирующий алгоритм В.М.Глушкова базируется на дискретном событийном подходе к моделированию сложных систем.

Непрерывно-дискретной системой называется математическая модель вида:

$$S = \{ T, P, e, E, K, F \}, \qquad (3.3)$$

где $T = \{t_i\}, \quad t_i \in R \geq 0$ - дискретная модель времени;

P – множество классов процессов:

e – множество классов событий (причин мгновенной смены поведения и структуры системы);

E – множество алгоритмов классов событий (подготовительных дискретных действий при переходе к новому поведению системы);

K – календарь планирования событий;

F – список уравнений, характеризующих локальные поведения процессов во временных интервалах между событиями.

В календарь планирования событий записываются отметки о событиях отдельными объектами, с его помощью описывается динамика системы. Планирование события подразумевает явное задание момента его наступления или задание условия его наступления. Модель календаря можно представить видом:

$$K = \{ \langle t_i, e_i \rangle, L \}, \qquad (3.4)$$

где L – условие его наступления (планирование события по условию, задается предикатом).

Структура процесса и его поведение описывается математической моделью:

$$P = \{ X, Y, V_3, V_d, B \}, \qquad (3.5)$$

где X , Y – каналы входа и выхода;

V_3 – множество статических переменных процесса, которые задаются алгебраическими выражениями и могут меняться только при исполнении алгоритмов событий;

V_d – множество "переменных-функций" - динамических переменных, которые задаются дифференциальными уравнениями из множества F ;

B – тело процесса, содержащее описания его всевозможных поведений.

Под моделированием поведения непрерывно-дискретной системы понимается построение множества последовательностей событий, приводящих к смене ее поведения и структуры, причисляя к событию начальное состояние системы. Глобальное поведение моделируется с помощью специального процесса-монитора, который продвигает системное время в соответствии с календарем планирования событий или в соответствии с анализом времени наступления события, которое планируется по условию. Процесс моделирования заканчивается, когда календарь событий оказывается пустым.

Для моделирования систем В.М.Глушкова разработана система моделирования НЕДИС.

Непрерывно–дискретная модель В.М.Глушкова (3.1) может быть описана в терминах агрегативного подхода и представлена в виде агрегата:

$$A_p = \{ T, Z, X, \Gamma, Y, H, G \} \qquad (3.6)$$

в котором

$T \subseteq R \geq 0$ – дискретная модель времени расширяется до непрерывной ,

$Z = V_S^{(P)} \cap V_d^{(P)}$ – фазовое пространство суть область значений переменных процесса P,

$X = e_1 \subset e$ – множество входных сигналов суть подмножество событий непрерывно–дискретной модели, алгоритмы которых $E(e_1)$ содержат операции изменения значений переменных Vs данного процесса,

$\Gamma = e_2 \cap e_{Init}$, $e_2 \subset e$ – множество управляющих сигналов суть подмножество событий непрерывно–дискретной модели, алгоритмы которых $E(e_2)$ содержат операции смены поведения и структуры (активизация, пассивизация, порождение, удаление, присвоение значений переменным V_d) данного процесса, плюс сигнал eInit , соответствующий событию начального запуска системы, в алгоритм которого входит инициализация переменных всех процессов, и в частности данного (для динамически порождаемого процесса определим например $E_{Init} = \{Z := 0\}$),

$Y = e(L) \subset e$ – множество выходных сигналов суть подмножество событий непрерывно–дискретной модели, планируемых по условию, для которых в L входят переменные данного процесса. Список этих условий определяет разбиение фазового пространства Z на систему подмножеств $\{Z_y\}$,

$H = \{V_x, V_g, W, U\}$ – оператор глобального поведения процесса P, в котором $V_x = \{E(e_1)\}$, $V_g = \{E(e_2)\}$ – алгоритмы событий множеств e_1 и e_2 соответственно; $W = \{E(e_L)\}$ – алгоритмы событий множества e_L ,

$U = F^{(P)}(V^{(P)}) \cap F_{Idle}$ — все возможные локальные поведения процесса P из списка F плюс дополнительно введенная функция F_{Idle} , описывающая ”никакое”поведение процесса до запуска, после удаления или в период его пассивности в непрерывно–дискретной модели (например, можно определить $F_{Idle} = \{Z = const\}$),

$G = G_? \cap G_!$ — оператор выходов, в котором $G_?$ есть оператор проверки условий $L^{(P)}$, $G_!$ — генератор выдачи сигналов типа активизации, пассивизации и пр. для других процессов. Фактически, оператор G частично выполняет функции процесса–монитора в непрерывно–дискретной модели В.М.Глушкова.

Всю непрерывно–дискретную систему можно представить A-комплексом с моделью каналов связей типа ”кажный с каждым”, в котором

число агрегатов соответствует максимальному числу процессов с учетом порождаемых во время функционирования.

Из данного построения видно, что множества входных, выходных и управляющих сигналов агрегата включают в себя описание всех возможных классов событий, приводящих к смене поведения и структуры непрерывно–дискретной системы и событие начального запуска и не содержат никаких других сигналов. Это означает, что каждому событию некоторого процесса непрерывно–дискретной системы соответствует единственное особое состояние соответствующего агрегата, и, следовательно, каждой цепочке событий непрерывно–дискретной системы соответствует единственная последовательность особых состояний, порождаемая А-комплексом, и никаких других последовательностей не порождается.

Таким образом, непрерывно–дискретная система может быть описана в терминах агрегативного направления, хотя, очевидно, при переходе от модели В.М.Глушкова к агрегативной системе теряется наглядность описания динамики системы, так как возможность динамического изменения структуры системы заменяется на фиксированную структуру системы с максимальным числом элементов.

Гибридный подход А.Пнуэли. Гибридное направление исследования непрерывно–дискретных систем возникло в начале 90-х годов на базе современной методологии спецификации и верификации сложных дискретных систем и систем реального времени, разработанной в теории реактивных систем. Основатели гибридного направления (А.Пнуэли, Д.Харел), вводя в базовую дискретную модель реактивной системы некоторые характеристики непрерывного поведения, определяют, таким образом новый класс сложных систем, который они называют ”гибридной реактивной системой”[35].

До 1992 года основной акцент в этом направлении был сделан на проблемах спецификации (построения компактного представления математической модели) непрерывно-дискретных систем в рамках дискретного подхода и аксиоматического доказательства некоторых качественных поведенческих свойств.

С появлением метода символьной верификации для систем реального времени [38], появилась надежда автоматизировать верификацию гибридных систем. В 1995-96 году создана система HyTech автоматической верификации гибридных систем, основанная на символьной верификации.

Исследование поведения гибридной системы сводится к статическому качественному анализу поведенческих свойств, без использования поточечного численного моделирования глобального поведения системы.

Гибридной системой называется следующая конструкция:

$$H = \{S, X, E, F, \varphi, \psi, \lambda\} \qquad (3.7)$$

где

S — конечное множество локаций,

X — конечное множество вещественных переменных,

E — конечное множество дуг. Дуга есть кортеж

$e = < s, a, \varphi_e, \lambda^{(s')}, s' > \in E$, $s, \dot{s} \in S$– исходная и целевая локации для дуги e, $a \in \Sigma$ Σ– алфавит меток переходов (алфавит событий),

φ – множество предикатов над X, описывающих условия перехода по дугам E,

λ – множество начальных значений переменных множества X для каждой локации,

F — оператор локального поведения внутри каждой локации (система дифференциальных включений, дифференциальных уравнений или аналитических функций),

ψ — множество предикатов над X, описывающих область значений переменных X в локациях $s \in S$ (инвариант bs).

Состоянием гибридной системы называется вектор значений переменных $x \in X$, к которому для удобства иногда приписывается локация $s \in S$, к которой этот вектор относится (т.е. для которой $\varphi_S(x) = True$). Поэтому s часто состоянием гибридной системы называется пара $<s, x>$, $s \in S$, $x \in X$.

Математической моделью, описывающей поведение гибридной системы, является система переходов:

$$\Sigma = \{T, Q, Q_0, Q_F \rightarrow\}, \qquad (3.8)$$

где T = {t } — бесконечная упорядоченная несходящаяся временная последовательность, Q — пространство состояний (в смысле определения состояния гибридной системы), Q_0, Q_F — множество начальных и конечных состояний, определяемых специальными предикатами над X,

→— отношение перехода между состояниями, которое определяется двумя правилами:

- дискретный переход при фиксированном времени (становится возможным как только $\varphi(X_0) = True$);
- временной переход – увеличение времени внутри локации с преобразованием непрерывных переменных (время относительное).

Моделирование глобального поведения системы переходов заключается в построении множества вычислений — бесконечных цепочек пар <s, t>.

Математическая модель гибридной системы эквивалентна модели агрегата. Действительно, система переходов Σ гибридной системы

$$H = \{S, X, E, F, \varphi, \psi, \lambda\}, \qquad (3.9)$$

может быть представлена агрегатом

$$A^* = \{T^*, Z^*, X^*, \Gamma^*, Y^*, H^*, G^*\}, \qquad (3.10)$$

в котором

$T^* = [0; \infty)$ — модель времени расширяется до непрерывной,

$Z^* = X$ — фазовым пространством является область значений вещественных переменных системы H , $z = (x , x , \ldots , x) \in Z$, фазовые координаты суть переменные системы H,

$\{z(0)\}* = Q$ — множество начальных состояний,

$X^* = \{X0\}$ — множество входных сигналов вырождается в единственный сигнал запуска агрегата (который инициализирует случайным образом некоторое начальное состояние из Q),

$\Gamma^* = 0$ — управляющие сигналы отсутствуют,

$Y^* = (y1, y2)$ — выходными сигналами являются вектора с компонентами y1 ∈ S, y2 ∈ T (компоненты элементов сценариев гибридной системы),

$H^* = \{W, U\}$ — оператор переходов, в котором W — оператор, описывающий преобразование вектора переменных при дискретном переходе по дуге (в момент выдачи выходного сигнала), соответствует оператору инициализации λ гибридной системы:

$$z(t + 0) = \lambda_{z(t)},$$

U — функция, описывающая непрерывные изменения в интервалах между моментами выдачи выходных сигналов (то есть в некоторой локации s ∈ S), соответствует F :

$z(t) = Fs\ [z(ti + 0), t], t \in [t_i, t_{i+1})$

$G = G_? \cup G_!$, где $G_?$ – оператор проверки принадлежности z (t) некоторому множеству из семейства $\{Z_y\}$. Разбиение Z на подмножества Z эквивалентно введению предикатов φ, ψ гибридной системы H и разбиению пространства состояний на локации S, $G_!$ – генератор выходного сигнала.

Способ разбиения пространства состояний на семейство $\{Z_y\}$ и соответствующее множество выходных сигналов определяет взаимно однозначное соответствие между начальной точкой в вычислении гибридной системы и особым состоянием агрегата, соответствующем приему сигнала запуска системы, и между каждой другой точкой в вычислении гибридной системы и особым состоянием, определяемым выдачей соответствующего выходного сигнала. Таким образом, из данного построения видно, что множество последовательностей особых состояний, порождаемых агрегатом, и множество вычислений гибридной системы эквивалентны.

Подход системной динамики Дж. Форрестера. Для анализа сложных систем с нелинейными обратными связями используется принцип системной динамики. Системная динамика — парадигма моделирования, где для исследуемой системы строятся графические диаграммы причинных связей и глобальных влияний одних параметров на другие во времени, а затем созданная на основе этих диаграмм модель имитируется на компьютере. По сути, такой вид моделирования более всех других парадигм помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями. С помощью системной динамики строят модели бизнес-процессов, развития города, модели производства, динамики популяции, экологии и развития эпидемии и другие. Метод основан Дж. Форрестером в 1950-х годах.

Системная динамика как метод имитационного моделирования включает в себя:

- структуризацию объекта;
- построение системной диаграммы объекта, где указываются связи между элементами;
- определение переменных для каждого элемента и темпов их роста;
- принятие гипотез о зависимости каждого темпа роста от переменных и формальное описание этих гипотез;
- процесс оценки введенных параметров с помощью имеющейся статистики.

Для построения и исследования моделей с помощью метода системной динамики разработан специальный язык программирования DYNAMO.

Модель Форрестера включает следующие элементы:

- уровни (ресурсы);
- потоки, перемещающие содержимое одного уровня к другому;

- функции решений, которые регулируют темпы потока между уровнями;
- каналы информации, соединяющие функции решений с уровнями.

Уровни характеризуют возникающие накопления внутри системы. Это могут быть заготовки, комплектующие и готовая продукция, страховые межоперационные запасы, производственные площади, численность работающих, финансовые ресурсы и т. п. Каждый уровень описывается его переменной величиной, зависящей от разности входящих и исходящих потоков. Темпы определяют существующие мгновенные потоки между уровнями в системе и отражают работу, в то время как уровни измеряют состояние, которого система достигает в результате выполнения некоторой работы.

Функции решений (или уравнения темпов) представляют собой формулировку правила поведения, определяющую, каким образом имеющаяся информация об уровнях приводит к выбору решений, связанных с величинами текущих темпов.

Базовая структура модели Форрестера, представленная на рисунке 3.15, показывает только одну сеть с элементарной схемой информационных связей между уровнями и темпами. Чтобы отразить деятельность всего промышленного предприятия, необходимы несколько взаимосвязанных сетей. Выделяют шесть типов сетей, представляющих существенно различные типы переменных: заказы, материалы, денежные средства, рабочую силу и оборудование, соединенных воедино с помощью сети информации. Любая из этих сетей может быть разбита еще на несколько отдельных частей.

Информационная сеть служит для остальных сетей соединительной тканью. Она переносит информацию от уровня к точкам решений, а также информацию о темпах в двух сетях к уровням в сети информации. В самой сети информации тоже существуют уровни и темпы. Например, информация о фактическом текущем темпе в потоке материалов усредняется для определения уровня среднего темпа потока материалов. Этот уровень относится к сети информации.

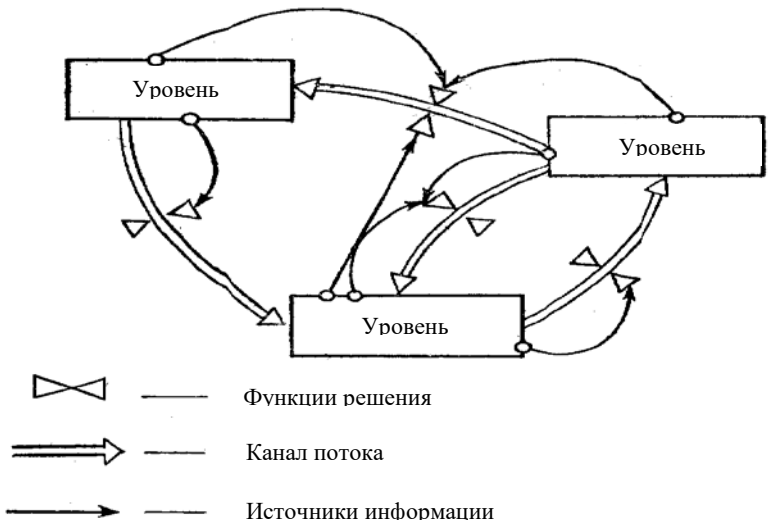


Рис. 3.15. Базовая структура модели Форрестера

Базовая структура модели дополняется системой уравнений, которые связывают характеристики уровней этой структуры. В основном эта система состоит из уравнений двух типов: уравнения уровней и уравнения темпов.

При построении уравнений временная ось разбивается на интервалы времени $\Delta\tau_{ij}$ между i-м и j-м моментами времени.

Новые значения уровней рассчитываются на конец интервала, и по ним определяются новые темпы (решения) для следующего интервала Δt_{ij+1}.

Уравнения уровня имеют вид:

$$Q_{ij} = Q_{ik} + \Delta\tau_{kj}\left(\sum_{x'} q^{k,j}_{x_1,i} - \sum_{x''} q^{k,j}_{x_2,i}\right), \qquad (3.11)$$

где $x_1 \in x', x_2 \in x'', x' \bigcup x'' = x$;

x - множество уровней, связанных с i-м уровнем;

Q_{ij} – значение i-го уровня в j-й момент времени;

$\Delta\tau_{kj}$ – величина интервала от момента времени k до момента времени j;

$q^{k,j}_{x_1,i}$ – темпы потоков, входящих в i-й уровень в интервале между моментами k и j;

$q^{k,j}_{x_2,i}$ – темпы потоков, выходящих из i-го уровня в интервале между моментами времени k и j.

Пример уравнения темпа:

$$q^{k,j}_{j} = S_k / \Delta t, \qquad (3.12)$$

где S_k - величина уровня, отражающего запаздывание в момент времени k; Δt - константа (среднее время), необходимое для преодоления запаздывания.

Этапы построения модели Форрестера:

1. Построение базовой структуры модели в виде специализированного графа;
2. Параметризация графа и построение соответствующей системы уравнений;
3. Описание полученной модели на одном из языков и систем программирования и проведение экспериментов.

К числу достоинств модели относятся: возможность отражать практически любую причинно-следственную связь; простая математическая форма; использование терминологии, синонимичной языку экономики и производства. Для основных фазовых переменных (так называемых системных уровней) используются дифференциальные уравнения одного вида:

$$\frac{dy}{dt} = y^{+} - y^{-}, \qquad (3.13)$$

где y^{+} – положительный темп скорости переменной y, включающий в себя все факторы, вызывающие рост переменной y;

y^{-} – отрицательный темп скорости, включающий в себя все факторы, вызывающие убывание переменной y.

Предполагается, что эти темпы расщепляются на произведение функций, зависящих только от "факторов" – комбинаций основных переменных, т.е., в свою очередь, самих являющихся функциями системных уровней: $y^{\pm} = g(y_1, y_2, \ldots, y_n) = f(F_1, F_2, \ldots, F_k) = f_1(F_1) f_2(F_2) \ldots f_k(F_k)$,

где $F_j = g_j(y_{i_1}, \ldots, y_{i_m})$ — факторы, причем $m = m(j) < n, \quad k < n, n - число\ уровней$, т.е. факторов меньше, чем основных переменных, и каждый фактор зависит не от всех системных уровней, а только от какой-то их части. Это позволяет упростить задачу моделирования.

Модель Форрестера – простая, ясная и полезная модель – иллюстрирует интересный подход к моделированию сложных нелинейных систем. Задуманная как учебный пример применения метода системной динамики, она стала неким образцом для последующих работ, привлекла внимание к проблеме мировой динамики, дала толчок к проведению других исследований, что привело к появлению целого направления, получившего название гло-

бального моделирования. Модель применяется для прогнозирования экологии, демографии, финансового анализа, экономики и социологии.

Программными средствами, поддерживающимися подход системной динамики (в частности модель Форрестера), являются: AnyLogic, Arena, SimBioSys, eM-Plant, Tecnomatix Plant Simulation, SimuLab, VenSim, PowerSim, Stella, Ithink, ModelMaker и др. Для построения моделей в них используются графическое представление зависимостей переменных, в виде так называемых «stock and flow diagrams».

Многоуровневый иерархический подход к анализу сложных систем (модель Мезаровича-Пестеля)[17]**.** Этот подход позволяет проводить анализ и расчет сложных систем на основе теории многоуровневых иерархических систем. В основе модели лежит идея "органического дифференцированного роста".

Основные принципы её построения:

1) Модель, отражающая сложные процессы взаимодействия человека с окружающей средой, должна основываться на теории многоуровневых иерархических систем.

2) Модель должна быть управляемой, т.е. включать в себя процесс принятия решений, что позволяет учесть возможность сознательного воздействия человека на развитие мировой системы.

3) Мир следует рассматривать не как единое однородное целое, а как систему взаимодействующих регионов, различающихся уровнем развития, населенностью и т.п.

Для оценки возможных вариантов развития будущего в модели используется метод анализа альтернативных сценариев. Сценарий представляет собой комбинацию возможных в будущем событий и альтернативных социально-политических решений. Реальное будущее лежит где-то в пределах рассматриваемого набора возможных сценариев. Использование метода Месаровича-Пестеля позволяет моделировать динамику каждого из сценариев и оценивать, к каким возможным последствиям в глобальном или региональных масштабах могут привести те или иные конкретные меры, направленные либо на достижение «предпочтительного будущего», либо на то, чтобы избежать развития каких-то нежелательных явлений или процессов.

Модель Месаровича-Пестеля принято считать «совершеннее» моделей Медоузов, модели «глобального равновесия» Форрестера. Месарович и Пестель противопоставили концепции “глобального равновесия” концепцию

“органичного роста” мира как единой системы взаимосвязанных частей, которые должны гармонически сочетаться. “Органичный рост” обеспечивается структурной дифференциацией элементов системы и функциональной взаимозависимостью между этими элементами и противопоставляется чисто количественному недифференцированному экспоненциальному росту.

М. Месаровичем и Э. Пестелем была построена на основе методики «иерархических систем» регионализированная модель, в которой мир разделен на 10 регионов с учетом экономических, социально-политических и идеологических различий. Каждый из этих регионов, в свою очередь, разделен на взаимодействующие иерархические сферы, или страты: экологическую, включающую антропогенно преобразуемую неживую природу и весь живой мир, кроме человека; технологическую – совокупность созданной техники и ее воздействие на природную среду; демоэкономическую, оказывающую влияние на развитие техники; социально-политическую, в которую входят «формальные организации» – правительства, официальные учреждения и т. п., а также «неформальные организации» – религиозные и политические движения, оказывающие влияние на деятельность формальных организаций; наконец, индивидуальную страту, которая охватывает условия физического и психологического развития человека.

Такая модель более реалистична и способна дать более детализированную и приемлемую для различных районов мира систему рекомендаций. В модели Месаровича и Пестеля заложено около ста тысяч соотношений (в более ранних моделях мира их было несколько сотен). Месарович и Пестель пришли к существенно иным выводам, чем Форрестер и группа Медоуза. Результаты их моделирования показали, что можно ожидать не одну глобальную, а несколько региональных катастроф. Варианты моделирования (или, как их называют, сценарии) предсказывают, прежде всего, продовольственный кризис в Юго-Восточной Азии вследствие отставания темпов роста производства продуктов питания от темпов роста народонаселения. По мнению Месаровича и Пестеля, стабилизация населения этого региона через 50 лет не даст возможности преодолеть продовольственный кризис, а стабилизация через 25–30 лет окажет положительное влияние в том случае, если экономике данного региона будет оказана соответствующая помощь.

Границы применения модели:

- невозможность полной формализации систем предпочтений и ценностных установок человека;
- невозможность полной формализации процедур принятия решений;

- невозможность полной формализации механизмов целеполагания социокультурной адаптации.

Агентный подход к разработке имитационных моделей — относительно новое (1990-2000-е гг.) направление в имитационном моделировании, которое используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами (как в других парадигмах моделирования), а наоборот, когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы. Цель агентных моделей — получить представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном, частном поведении ее отдельных активных объектов и взаимодействии этих объектов в системе. Агент — некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться.

Агентное моделирование является мощным инструментом исследования процессов эволюции сложных социальных систем. Агентные системы состоят из следующих основных компонентов:

- множество системных единиц, в котором выделяются подмножество активных единиц — агентов, манипулирующих подмножеством пассивных единиц — объектов;
- среда, т. е. некоторое пространство, в котором существуют агенты и объекты;
- множество задач (функций, ролей), которые поручаются агентам;
- множество отношений (взаимодействий) между агентами;
- множество организационных структур (конфигураций), формируемых агентами;
- множество действий агентов (например, различных операций над объектами или коммуникативных актов).

Взаимодействие агентов означает установление двухсторонних и многосторонних динамических отношений между агентами. Взаимодействия между агентами имеют определенную направленность — положительную или отрицательную, т. е. носят характер содействия или противодействия, притяжения или отталкивания, кооперации или конкуренции, сотрудничества или конфликта, координации или субординации и т. п.

Взаимосвязи и взаимозависимости между агентами характеризуются некоторой силой (интенсивностью). Взаимодействия между агентами динамичны.

Агентный подход позволяет исследовать задачи коллективного взаимодействия, эффективно решать задачи прогнозирования. Агентные системы позволяют исследовать процессы самоорганизации, дают возможность естественного описания сложных систем, обладают высокой гибкостью. В настоящее время выполняется несколько проектов по разработке многоагентной системы моделирования процессов кооперации и самоорганизации. По данному подходу разработаны языки имитационного моделирования: Swarm (Objective C) и его расширение MAML - Мультиагентный язык моделирования, SimAgent, SimBioSys, C++, Java, AgentSpeak, TeleScript, Oz и др.

Агентный подход позволяет исследовать задачи коллективного взаимодействия, эффективно решать задачи прогнозирования. Многоагентные системы позволяют исследовать процессы самоорганизации, дают возможность естественного описания сложных систем, обладают высокой гибкостью. В настоящее время выполняется несколько проектов по разработке многоагентной системы моделирования процессов кооперации и самоорганизации.

Когнитивный подход к анализу слабоструктурированных систем

Сложности анализа процессов принятия управленческих решений в таких областях как экономика, социология, экология и т.п. обусловлены рядом особенностей, присущих этим областям, а именно:

- взаимосвязанностью происходящих в них процессов (технико-экономических, социальных, политических и т.п.) и их многоаспектностью; в силу этого невозможно вычленение и детальное исследование отдельных явлений (например, только экономических или только социальных) – все происходящие внутри экономической (политической и т.п.) системы явления должны рассматриваться и исследоваться в совокупности;
- отсутствием достаточной количественной информации о динамике происходящих в моделируемой системе процессов, что вынуждает использовать наряду с количественной и качественную информацию при описании таких процессов;
- нестационарностью самих процессов, причем характер изменения тех или иных характеристик процессов зачастую неизвестен, что затрудняет построение их количественных моделей.

Такие системы называются слабоструктурированными (слабоформализованными). В них невозможен традиционный математический (экономиче-

ский, социометрический и т.п.) подход к анализу процессов для выработки комплексных (т.е. затрагивающих различные аспекты исследуемой системы) решений. Для моделирования сложных плохоформализуемых систем (например, социальных, технико-экономических, региональных и т.п.) используется когнитивный подход, который основывается на когнитивных аспектах. Эти аспекты включают в себя процессы восприятия, мышления, познания, объяснения и понимания. Схематическое, упрощенное описание картины мира, относящееся к проблемной ситуации, изображают в виде когнитивной карты.

С позиций когнитивного подхода процесс моделирования можно представить в виде схемы – рис.3.16.

Когнитивный анализ предусматривает последовательную причинно-следственную структуризацию информации о происходящих в исследуемой системе процессах. Выделяют следующие этапы описания системы:

- всякое событие, произошедшее в системе, вызывается определенными причинами (предпосылками), появление которых связано с движением материальных потоков (товары, деньги, ресурсы и т.п.) и нематериальных потоков (информационные взаимодействия). Движение каждого потока может быть описано в самом общем виде соответствующими цепочками причинно-следственных отношений, составляющих знания аналитика или его предположения о действующих в данной системе закономерностях.

Рис. 3.16. Процесс моделирования

- каждый из выделенных потоков описывается соответствующей совокупностью факторов. Объединение всех этих совокупностей составляет множество факторов, в терминах которых описываются процессы в си-

стеме;

- определяются взаимосвязи между факторами путем рассмотрения причинно-следственных цепочек, описывающих движение каждого потока. Считается, что факторы, входящие в первую часть «если...» цепочки «если - то.,..», влияют на факторы её второй части «то...», причем это влияние может быть либо усиливающим (положительным) либо тормозящим (отрицательным), либо переменного знака в зависимости от возможных дополнительных условий.

Для анализа и управления такого рода системами в настоящее время широко применяется когнитивный подход, который позволяет увидеть и осознать логику развития событий при большом количестве взаимозависимых факторов. Теоретические основы когнитивного подхода были заложены И. Максимовым (Институт проблем управления РАН).

Для принятия правильных управленческих решений необходимо отвечать на ряд весьма нетривиальных вопросов: «Что нужно сделать, чтобы улучшить состояние ситуации?», «Что будет с ситуацией через такое-то время, если ничего не предпринимать?» и т.д.

Успешно ответить на такие вопросы позволяют компьютерные средства познавательного (когнитивного) моделирования, основанные на когнитивном подходе. Специфика применения средств когнитивного моделирования заключается в их ориентированности на конкретные условия развития ситуации в той или иной стране, регионе, городе, поселке и т.д. Поэтому попытки применить в российских условиях известные зарубежные средства когнитивного моделирования пока не увенчались успехом.

Ключевым понятием в когнитивном моделировании является «когнитивная карта», представляющая собой взвешенный ориентированный граф, в котором вершины взаимнооднозначно соответствуют факторам, в терминах которых описывается предметная область, а дуги отображают непосредственные связи (взаимовлияния) между факторами. Взаимовлияния могут быть положительными, (увеличение/уменьшение одного фактора приводит к увеличению/уменьшению другого) и отрицательными (увеличение/уменьшение одного фактора приводит к уменьшению/увеличению другого). Для отображения степени влияния используют совокупность лингвистических переменных и соответствующую ей совокупность числовых значений из интервала [0, 1]:

«очень слабое» – 0,1, «умеренное» – 0,3, «существенное» – 0,5, «сильное» – 0,7 и «очень сильное» – 0,9 (допустимы и промежуточные значения).

Для составления когнитивной модели предметной области необходимо:

1) выделить список значимых факторов;

2) построить матрицу взаимовлияний;

3) определить начальные тенденции изменения факторов.

Имея, таким образом, составленную модель и выделив во всем множестве факторов два подмножества: «управляющих» и «наблюдаемых» факторов, мы сможем решить следующие задачи:

- моделировать саморазвитие ситуации (т.е. отвечать на вопрос: «Что будет, если сохранятся текущие тенденции изменения факторов?»);
- моделировать управляемое развитие ситуации (т.е. отвечать на вопрос: «Что будет, если приложить определенные управляющие воздействия?»);
- искать необходимые управления (т.е. отвечать на вопрос: «Какие управляющие воздействия приложить, чтобы получить желаемый результат?»).

Сила воздействия факторов друг на друга описывается с помощью лингвистических переменных типа "значительная", "умеренная", "слабая" и т.п. Можно сопоставить совокупности таких лингвистических переменных некоторую числовую шкалу так, что каждой переменной будет соответствовать некоторое число в этой шкале. В качестве такой шкалы можно выбрать интервал [0.1].

- Взаимовлияния факторов отображаются с помощью когнитивной карты, являющейся, моделью исследуемой системы в виде взвешенного орграфа Каждая вершина графа соответствует одному фактору или элементу картины мира. Дуги, связывающие вершины, соответствуют причинно-следственной связи между вершинами, связи могут быть положительными и отрицательными.
- Метод когнитивного моделирования относится методам мягкого моделирования (soft simulation). Ближайшими аналогами данного метода являются имитационное моделирование, метод системной динамики. Преимущество данного метода состоит в том, что метод может оперировать не только точными количественными значениями и формулами, но качественными значениями и оценками. Но также данный момент является и недостатком, т.к. результаты получаются качественные.

Когнитивное моделирование является "нулевым уровнем" моделирования. Когнитивное моделирование помогает быстро получить первичные ре-

зультаты, более подробно разобраться в моделируемой системе, выявить закономерности и потом перейти к более точным моделям (если такое представляется возможным и необходимым). Поэтому наиболее разумным будет применение когнитивного моделирования на верхнем уровне принятия решения при анализе сложных социально-экономических, политических, технических, техноэкономических систем.

В России данный метод применятся в МПС (2002 году в ИПУ РАН была построена модель железнодорожного транспорта), а также в администрациях некоторых областей. За границей данный метод применятся в ряде консалтинговых организациях.

Разработаны инструментальные средства ДК "Ситуация" и "КАНВА" (ИПУ РАН). "Ситуация" – закрытая система, ни какой информации по ней практически нет. КАНВА – простая система, реализующая только базовые методы.

Подходы и технологии разработки имитационных моделей сложных систем развиваются и соответственно этим подходам разработки создаются системы автоматизации строительства имитационных моделей. Отметим, что построение соответствующих моделей предполагает значительные трудозатраты. Для их уменьшения разрабатывались и предлагались многочисленные инструментальные средства и среды автоматизации моделирования (ИС-CAM) – GPSS, AnyLogic, BPsim, PowerSim, Simplex, Modul Vision, Triad.Net, CERT, ESimL, Simulab, NetStar, Pilgrim, МОСТ, КОГНИТРОН и т.д. Характеристики наиболее известных систем имитационного моделирования представлены в таблице 2.2. С помощью этих систем разрабатываются различные имитационные модели в металлургическом, нефтегазовом, строительном и других производствах, в различных отраслях массового обслуживания (железнодорожный, автомобильный транспорт, аэропорты, поликлиники, санаторные и лечебные учреждения, любые торговые предприятия, сферы обслуживания и др.), в комической и военной отрасли. Анализируются и моделируются бизнес-процессы, локальные и вычислительные сети, социальные и экологические процессы и системы и многое другое.

Контрольные вопросы

1. Какие языки программирования относятся к языкам общего назначения, а какие к языкам имитационного моделирования?
2. Как понимается выражение «инструментальные системы разработки ПО» и «среды быстрого проектирования»?
3. Как мы понимаем выражение «математическая схема моделирования»?

4. В чем заключается спецификация, алгоритмизация и кодирование программы?
5. Каковы методы модульного программирования?
6. Каковы основные подходы в имитационном моделировании?
7. В чем заключается идея непрерывного и комбинированного подходов?
8. В чем идея системного подхода к моделированию?
9. Какова идея статистического подхода к разработке имитационной модели?
10. Какие группы языков используются для моделирования по дискретному подходу?
11. В чем суть подхода В.П.Бусленко?
12. Что такое дискретно-событийный подход к разработке моделей?
13. В чем идея подхода Глушкова для моделирования непрерывно-дискретных систем?
14. В каких случаях для разработки имитационных моделей используется подход системной динамики?
15. В чем заключается идея гибридного подхода А.Пнуэли?
16. Что такое мультиагентная модель системы?
17. Какие основные подходы анализа и построения компьютерных моделей существуют?

ГЛАВА 4. СИСТЕМЫ АВТОМАТИЗАЦИИ МОДЕЛИРОВАНИЯ

4.1. Построение аналитических моделей на основе программных сред автоматизации моделирования

Средства автоматизации построения компьютерных моделей условно можно подразделить на системы аналитического, технического и имитационного моделирования, и отдельно можно выделить системы построения графических моделей (рис.4.1).

К системам аналитического моделирования можно отнести пакеты MvStudium, Derive, Mathematica, MathCad, Maple, MatLAB и др.

Эти пакеты разработаны различными фирмами и имеют свои особенности. Каждый из этих пакетов имеет свой интерфейс. В этих пакетах алгоритмизированы, систематизированы и заложены в виде процедур практически все известные методы аналитического и численного решения математических задач. Каждая из систем компьютерной математики имеет свой язык программирования, который может быть использован для разработки компьютерной модели.

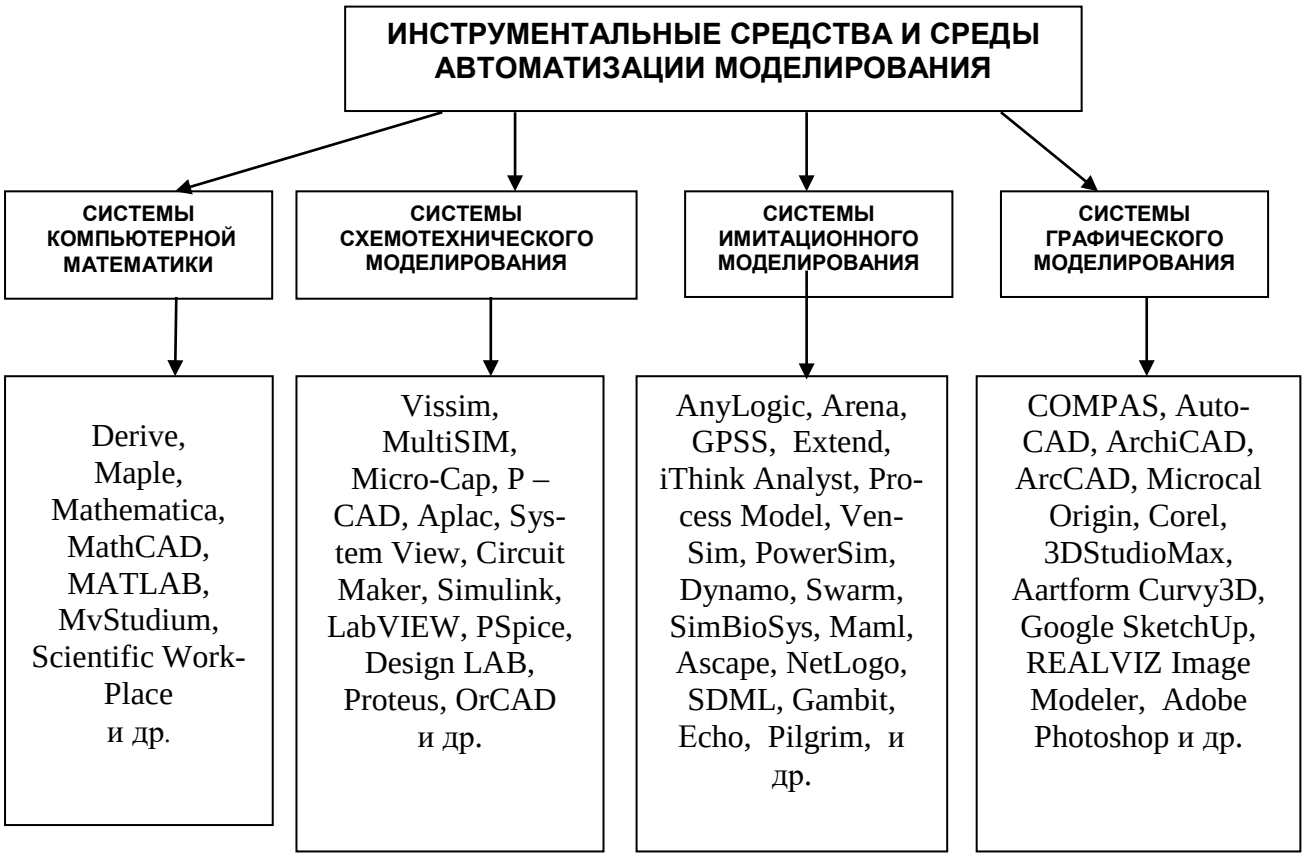


Рис. 4.1. Условная классификация систем автоматизации моделирования по типу решаемых задач

Эти системы называются также системами компьютерной математики, они нашли применение для решения большого круга задач с помощью программ, реализующих широко используемые математические методы решения разнообразных уравнений и систем, задач оптимизации, линейного программирования, для отладки типовых алгоритмов регулирования, для решения некоторых задач идентификации и проектирования.

Системы схемотехнического (аналогового) моделирования (аналоговые вычислительные машины - АВМ) позволяют решать различные виды математических моделей, представленных в виде дифференциальных уравнений с помощью натурного схемотехнического моделирования. В настоящее время получили развитие разработка информационных систем в виде компьютерных виртуальных конструкторов, к которым относятся Electronics Workbench, Simulink, Vissim, LabVIEW и др., решающие математические задачи с помощью схемотехнического имитационного моделирования [4, 11-13].

Системы графического моделирования. К этим системам можно отнести большое количество пакетов, связанных с построением и разработкой моделей графических образов: Corel, Adobe Photoshop, AutoCAD, ArchiCAD, Compass, 3DStudioMax, Aartform Curvy3D, Google SketchUp, VirtualGrid VRMesh, REALVIZ Image Modeler и др.

Системы имитационного моделирования. В настоящее время идет стремительное развитие направления разработки систем имитационного моделирования. Перечень программного обеспечения и систем имитационного моделирования можно посмотреть на сайте http://dic.academic.ru/dic.nsf.

Знание и применение систем компьютерной математики, технического и имитационного моделирования позволяют модельщикам оперативно выбрать систему моделирования, построить адекватные модели, способы их решения, перейти к полномасштабному исследованию реального явления или процесса на модели, оценить решения моделей и представить поведение и закономерности изучаемого явления.

Рассматриваемые методы математического и технологии компьютерного моделирования ориентированы на решение научных, экономических, технических и других прикладных задач.

Отметим, что результаты математического моделирования в значительной степени определяются тем, какие математические модели удалось построить, насколько они адекватны исследуемому процессу, насколько доступны для исследования.

4.1.1. Система компьютерной математики MathCAD

Система компьютерной математики MathCAD (разработка фирмы MathSoft, Inc.) является математическим редактором, позволяющим проводить разнообразные научные и инженерные расчеты, начиная от элементарной арифметики и заканчивая сложными реализациями численных методов.

Интерфейс данного пакета приведен на рис. 4.2.

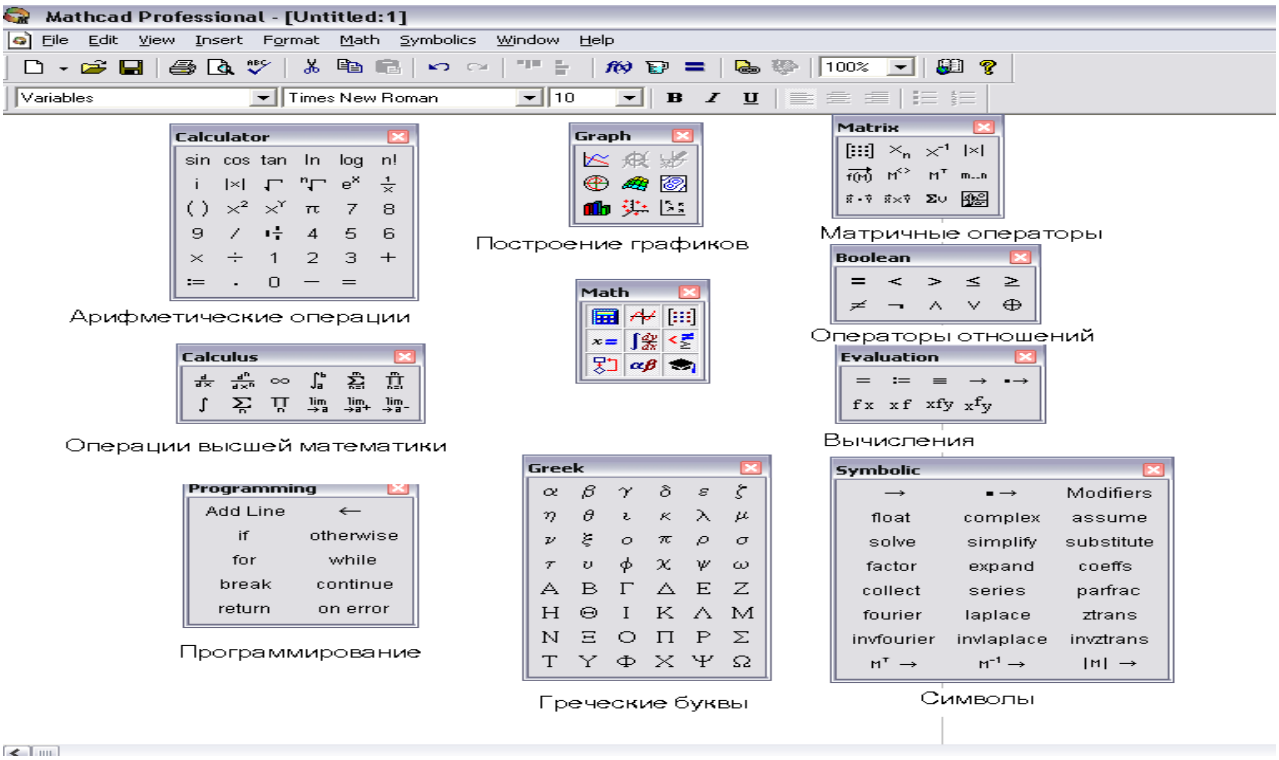


Рис.4.2. Пользовательский интерфейс СКМ MathCAD

Основная панель системы – это панель «Математика», имеющая 9 пиктограмм (кнопок), при нажатии которой выпадают панели:

- арифметических операций
- построения графиков
- матричных операторов
- операторов высшей математики
- матричных операторов
- операторов отношений
- операторов вычислений
- греческих букв
- операторов символьной математики

MathCAD включает в себя три редактора: формульный, текстовый и графический. В этой среде описание записи и решения математических задач дается с помощью привычных математических формул и знаков, так как это принято в математике. Система MathCAD относится к средствам аналитиче-

ского моделирования, она позволяет проводить два типа вычислений – символьный и численный. В первом случае мы получаем результат в виде математического выражения, во втором – в виде числа. В этой среде можно:

– решать всевозможные алгебраические задачи;

– осуществлять логические операции;

– проводить статистические вычисления и анализ;

– решать задачи с технологиями программирования;

– решать системы дифференциальных уравнений.

Рассмотрим построение аналитической компьютерной модели в MathCAD по технологии:

- применения языка программирования самой СКМ;
- использования готовых процедур (опций) СКМ.

Поясним применение этих технологий на примерах в СКМ MathCAD.

Пример 1. Применение языка программирования MathCAD. Рассмотрим задачу взаимодействия двух небесных тел. Движение любого тела в гравитационном поле описывается с помощью второго закона Ньютона

$$\vec{F} = m\vec{a}$$

и закона всемирного тяготения

$$F = G\frac{Mm}{r^2}\ .$$

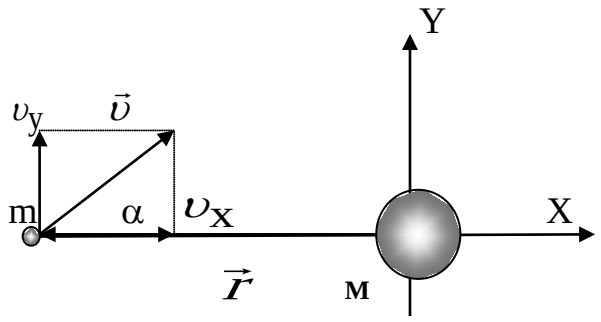


Рис.4.3

На основе этих законом можно получить уравнения, описывающие это взаимодействие:

$$\frac{d\upsilon_x}{dt} = -GM\frac{x}{(x^2+y^2)^{3/2}},$$

$$\frac{d\upsilon_y}{dt} = -GM\frac{y}{(x^2+y^2)^{3/2}}, \qquad (4.1)$$

$$\frac{dx}{dt} = \upsilon_x, \qquad \frac{dy}{dt} = \upsilon_y.$$

Начальные условия определяются двумя параметрами: начальной скоростью и углом α.

$$x(0) = x_0, \quad y(0) = 0,$$

$$\upsilon_{x0} = \upsilon_0\cos\alpha, \quad \upsilon_{y0} = \upsilon_0\sin\alpha. \qquad (4.2)$$

Ниже на листинге (рис.4.4) приведено численное решение на языке программирования MathCad.

Исправленный мет од Эйлера

$G := 6.67 \quad M := 2 \quad h := 0.033$

$v0 := 0.7 \quad n := 1156 \quad x_0 := -10 \quad y_0 := 0 \quad alfa := 30 \cdot \frac{\pi}{180}$

$Vx_0 := v0 \cdot \cos(alfa) \quad Vy_0 := v0 \cdot \sin(alfa)$

$$Fx(x,y) := -G \cdot M \cdot \frac{x}{\sqrt{x \cdot x + y \cdot y} \cdot (x \cdot x + y \cdot y)} \qquad Fy(x,y) := -G \cdot M \cdot \frac{y}{\sqrt{x \cdot x + y \cdot y} \cdot (x \cdot x + y \cdot y)}$$

$(x \; y) :=$

$x_0 \leftarrow x_0$

$y_0 \leftarrow y_0$

$Vx_0 \leftarrow v0 \cdot \cos(alfa)$

$Vy_0 \leftarrow v0 \cdot \sin(alfa)$

for $i \in 0..(n-1)$

$ax_i \leftarrow Fx(x_i, y_i)$

$ay_i \leftarrow Fy(x_i, y_i)$

$Vx_{i+1} \leftarrow Vx_i + h \cdot ax_i$

$Vy_{i+1} \leftarrow Vy_i + h \cdot ay_i$

$x_{i+1} \leftarrow x_i + h \cdot Vx_{i+1}$

$y_{i+1} \leftarrow y_i + h \cdot Vy_{i+1}$

$ax_{i+1} \leftarrow Fx(x_{i+1}, y_{i+1})$

$ay_{i+1} \leftarrow Fy(x_{i+1}, y_{i+1})$

$x_{i+1} \leftarrow x_i + \frac{h}{2} \cdot (Vx_{i+1} + Vx_i)$

$y_{i+1} \leftarrow y_i + \frac{h}{2} \cdot (Vy_{i+1} + Vy_i)$

$Vx_{i+1} \leftarrow Vx_i + \frac{h}{2} \cdot (ax_i + ax_{i+1})$

$Vy_{i+1} \leftarrow Vy_i + \frac{h}{2} \cdot (ay_i + ay_{i+1})$

$(x \; y)$

$stack(x^T, y^T) =$

	0	1	2	3	4	5
0						
1						

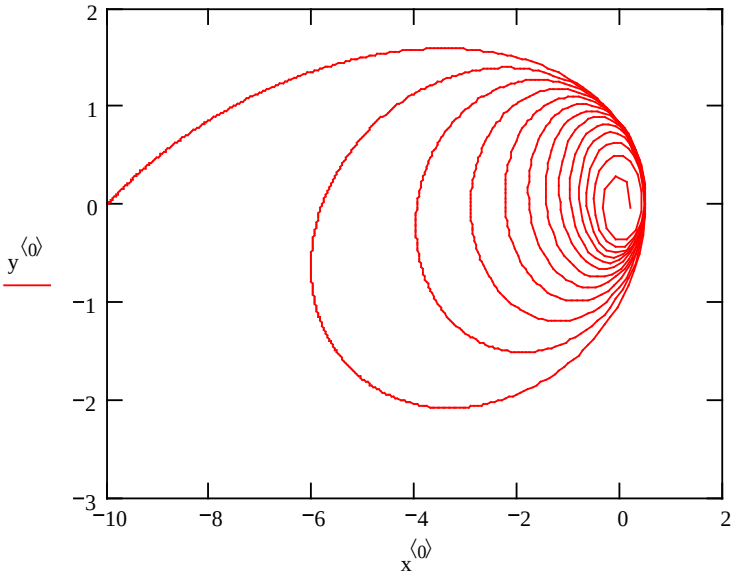


Рис.4.4

Здесь при программировании для создания (x y) используется опция матрица из блока *Matrix*, для проведения вертикальных линий необходимо воспользоваться опцией *Add Line* из блока *Programming*.

Пример 2. Использование готовых процедур (опций) MathCad. Здесь воспользуемся встроенной опцией (процедурой) численного решения ОДУ по методу Рунге - Кутта с равномерным шагом **rkfixed (y, a, b, n, f),** где y – определяемая функция, [a, b] – интервал, где ищется решение, n – число то-

чек разбиения интервала, f – правая часть уравнений. Эту опцию можно использовать для решения ОДУ или системы дифференциальных уравнений первого порядка.

Рассмотрим задачу полета сверхзвукового самолета. Уравнения движения для полета сверхзвукового самолета можно представить в виде

$$\frac{d\upsilon}{dy}=\frac{F_T-Q-mg\sin\alpha}{m\upsilon\sin\alpha}=f(\upsilon,\alpha),\qquad \frac{d\alpha}{dy}=\frac{N-mg\cos\alpha}{m\upsilon^2\sin\alpha}=\varphi(\upsilon,\alpha)$$

Где сила сопротивления воздуха и подъемная сила определяется как

$$Q=\frac{1}{2}k_1\rho S\upsilon^2 \qquad\qquad N=\frac{1}{2}k_2\rho S\upsilon^2$$

Здесь k_1 – коэффициент сопротивления воздуха, ρ - плотность воздуха, S - площадь крыльев, k_2 – коэффициент подъемной силы. Плотность воздуха подчиняется закону распределения Больцмана с учетом температуры

$$\rho=\rho_0\exp(-\frac{mgy}{kT})=\rho_0\exp(-\beta\cdot y\frac{T_0}{T_0-\alpha\cdot y})$$

Сила тяги является функцией высоты и скорости и будем определять ее по полуэмпирической формуле

$$F_T=(1+\frac{1}{2}\exp(-(0.001\upsilon-1.5)^2)(10^4-0.27y)$$

Компьютерная установка для моделирования данной задачи представлено на рис.4.5.

Листинг решения

$g := 9.8 \quad m := 8000 \quad k1 := 0.02 \quad k2 := 0.2 \quad p0 := 1.29 \quad k := 1.$

$s := 50 \quad T0 := 300 \quad k1 := 0.02 \quad n := 30000 \quad h := 0.01$

$b := \frac{0.125}{1000} \qquad a := \frac{0.65}{100} \qquad p(y) := p0\cdot e^{\frac{-b\cdot y\cdot T0}{T0-a\cdot y}}$

$Q(V,y) := \frac{1}{2}\cdot k1\cdot p(y)\cdot s\cdot V^2 \qquad N(V,y) := \frac{1}{2}\cdot k2\cdot p(y)\cdot s\cdot V^2$

$Ft(V,y) := \left[1+\frac{1}{2}\cdot e^{(0.001\cdot V-1.5)^2}\right]\cdot\left(10^4-0.27\cdot y\right)$

$FUNV(V,y,\alpha) := \frac{Ft(V,y)-Q(V,y)-m\cdot g\cdot\sin(\alpha)}{m\cdot V\cdot\sin(\alpha)}$

$$FUNA(V, y, \alpha) := \frac{N(V, y) - m \cdot g \cdot \cos(\alpha)}{m \cdot V^2 \cdot \sin(\alpha)}$$

$$D(t, z) := \begin{pmatrix} z_2 \cdot \cos(z_3) \\ z_2 \cdot \sin(z_3) \\ FUNV(z_2, z_1, z_3) \\ FUNA(z_2, z_1, z_3) \end{pmatrix} \qquad nach := \begin{pmatrix} 0 \\ 0 \\ 450 \\ \frac{15 \cdot \pi}{180} \end{pmatrix}$$

$UU := rkfixed(nach, 0, n \cdot h, n, D)$

$t := UU^{\langle 0 \rangle}$ $x := UU^{\langle 1 \rangle}$ $y := UU^{\langle 2 \rangle}$ $V := UU^{\langle 3 \rangle}$ $\alpha := UU^{\langle 4 \rangle}$

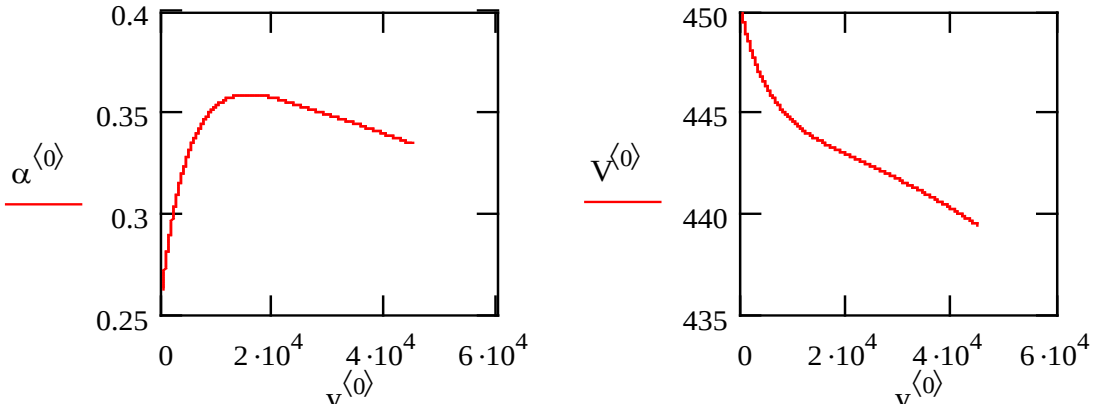


Рис.4.5

Аналитическая компьютерная модель, представленная на рис.4.5, позволяет провести исследования полета сверхзвукового самолета в зависимости от параметров и моделей среды, характеристик и силы тяги самолета. На этой же модели представлены результаты угла подъема и скорости самолета в зависимости от высоты полета.

Задание на моделирование:

1. Построить аналитические модели (рис.4.4 и рис.4.5) в среде моделирования MathCAD.
2. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

4.1.2. Система компьютерной математики MATLAB

Среди нескольких десятков математических пакетов (четыре из них перечислены ранее) особого внимания заслуживает система инженерных и научных расчетов MATLAB, которая в одинаковой мере ориентиро-

вана на применение как символических, так и численных методов. Включенное в MATLAB ядро Map1e хорошо справляется с символическими вычислениями, а сам MATLAB и его многочисленные инструментарии прекрасно работают с числами. Поэтому система MATLAB является одним из продвинутых средств разработки аналитических компьютерных моделей.

Система MATLAB предлагается разработчиками (фирма MathWorks, Inc.) как лидирующая на рынке, в первую очередь в системе военно-промышленного комплекса, в аэрокосмической отрасли и автомобилестроении, язык программирования *высокого уровня* для технических вычислений с большим числом стандартных пакетов прикладных программ.

Популярности системы способствует ее мощное расширение Simulink, предоставляющее удобные и простые средства, в том числе визуальное объектно-ориентированное программирование, для моделирования линейных и нелинейных динамических систем, а также множество других пакетов расширения системы. Это расширение позволяет строить не только аналитические компьютерные модели, но и имитационные модели (статистические, дискретно-событийные и др.)

Система MATLAB состоит из многих тысяч файлов, находящихся в множестве папок. Полезно иметь представление о содержании основных папок, поскольку это позволяет быстро оценить возможности системы — например, узнать, какие операторы, функции или графические команды входят в систему.

В MATLAB особое значение имеют файлы двух типов — с расширениями .mat и .m. Первые являются бинарными файлами, в которых могут храниться значения переменных. Вторые представляют собой текстовые файлы, содержащие внешние программы, определения команд и функций системы. Именно к ним относится большая часть команд и функций, в том числе задаваемых пользователем для решения своих специфических задач. Нередко встречаются и файлы с расширением .c (коды на языке Си), файлы с откомпилированными кодами MATLAB с расширением .tex и другие. Исполняемые файлы имеют расширение .exe.

Особое значение имеет папка MATLAB/TOOLBOX/MATLAB. В ней содержится набор стандартных m-файлов системы. Просмотр этих файлов позволяет детально оценить возможности поставляемой конкретной версии системы. Ниже перечислены основные подпапки с этими файлами (деление на категории условно, на самом деле все подпапки находятся в общей папке MATLAB/TOOLBOX/MATLAB).

Подпапка команд общего назначения:

General — команды общего назначения: работа со справкой, управление окном MATLAB, взаимодействие с операционной системой и т. д.

Подпапки операторов, конструкций языка и системных функций:

ops — операторы и специальные символы;

tang — конструкции языка программирования;

strfun — строковые функции;

iofun — функции ввода/вывода;

timefun — функции времени и дат;

datatypes — типы и структуры данных.

Подпапки основных математических и матричных функций:

elmat — команды создания элементарных матриц и операций с ними;

elfun — элементарные математические функции;

specfun — специальные математические функции;

matfun — матричные функции линейной алгебры;

datafun — анализ данных и преобразования Фурье;

polyfun — полиномиальные функции и функции интерполяции;

funfun — функции функций и функции решения обыкновенных дифференциальных уравнений;

soarfun — функции разреженных матриц.

Подпайки команд графики:

graph2d — команды двумерной графики;

graph3d — команды трехмерной графики;

specgraph — команды специальной графики;

graphics — команды дескрипторной графики;

uitools — графика пользовательского интерфейса.

Полный состав файлов каждой подпапки (их список содержится в файле соп-tents.m) можно вывести на просмотр с помощью команды help имя, где имя — название соответствующей подпапки.

Графические средства Handle Graphics (дескрипторная или описательная графика) позволяют создавать полноценные объекты графики высокого разрешения, как геометрического, так и цветового. Возможности этой графики поддерживаются объектно-ориентированным программированием, средства которого также имеются в языке программирования системы MATLAB.

Применение дескрипторной графики позволяет создавать и типовые элементы пользовательского интерфейса — кнопки, меню, информационные и инструментальные панели и т. д., то есть реализовать элементы визуально-

ориентированного программирования. Графики выводятся отдельно от текстов в отдельных окнах. На одном графике можно представить множество кривых, отличающихся цветом (при цветном дисплее) и отличительными символами (кружками, крестиками, прямоугольниками и т. д.). Графики можно выводить в одно или в несколько окон. Наконец, в статьях и книгах формата Notebook, реализованных при совместной работе системы MATLAB с популярным текстовым процессором Microsoft Word, графики могут располагаться вместе с текстом, формулами и результатами вычислений (числами, векторами и матрицами, таблицами и т. д.). Введен также ряд средств на основе графического интерфейса пользователя (GUI — Graphic User Interface), привычного для операционных систем Windows. Это панели инструментов, редактор и отладчик m-файлов, красочная демонстрация возможностей и т. д. Есть и возможность создавать свои средства пользовательского интерфейса.

С графическими возможностями можно познакомиться через учебные пособия [13] или через Help системы MATLAB в разделе **Graphics**. Здесь приведем 2 примера построения графиков с помощью процедур **plot(x,y)** и **plotyy(x,y1,x,y2);**

Пример 1. Построение графика с помощью процедуры **plot(x,y)**

```
x = 0:0.02:20; % интервал значений по x
y = 50*exp(-0.2*x).*sin(x); %  функция
figure(1); plot(x,y); %построение графика
```

Здесь сохранение графика проводилось с помощью **Edit → Copy Figure** в окне **Figure № 1**

```
x = 0:0.01:20; % интервал значений по x
y1 = 200*exp(-0.2*x).*sin(x); % первая функция
y2 = 0.8*exp(-0.5*x).*sin(10*x); % вторая функция
figure(2);[AX,H1,H2] = plotyy(x,y1,x,y2,'plot'); %построение графиков
```

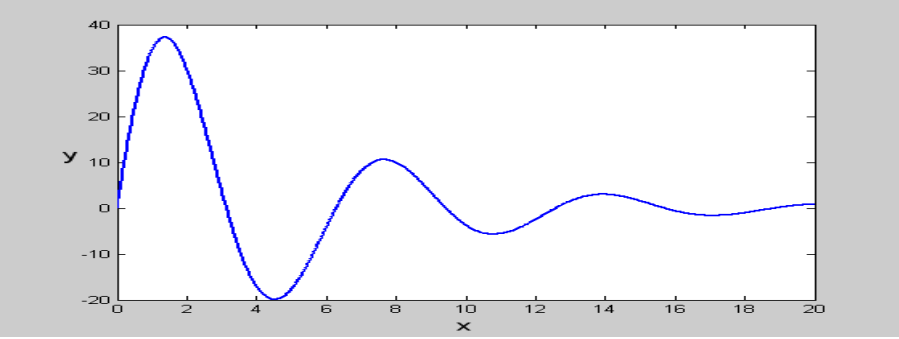


Рис.4.6

Пример 2. Выдача графиков с разными масштабами.

Для оформления графика можно использовать следующие операторы:

```
set(get(AX(1),'Ylabel'),'String','Left Y-axis'); % запись метки по оси y слева
set(get(AX(2),'Ylabel'),'String','Right Y-axis'); % запись метки по оси y справа
xlabel('Zero to 20 \musec.'); % метка по оси x
title('Labeling plotyy'); % обозначение графика
set(H1,'LineStyle','--'); % сплошная линия графика
set(H2,'LineStyle',':'); % пунктирная линия графика
```

Оформление внешнего вида графика (метки на осях, оглавление и др.) можно проводить также с помощью меню **Edit** в открытом окне нашего примера **Figure № 2.**

На рис.4.6 сохранение графика проводилось через **File** → **Export** в окне **Figure № 2** формате *.emf. Сохранение можно проводить с различным видом расширения *.emf, *.bmp, *.jpg, *.eps и другими, в зависимости от необходимости использования в том или ином документе.

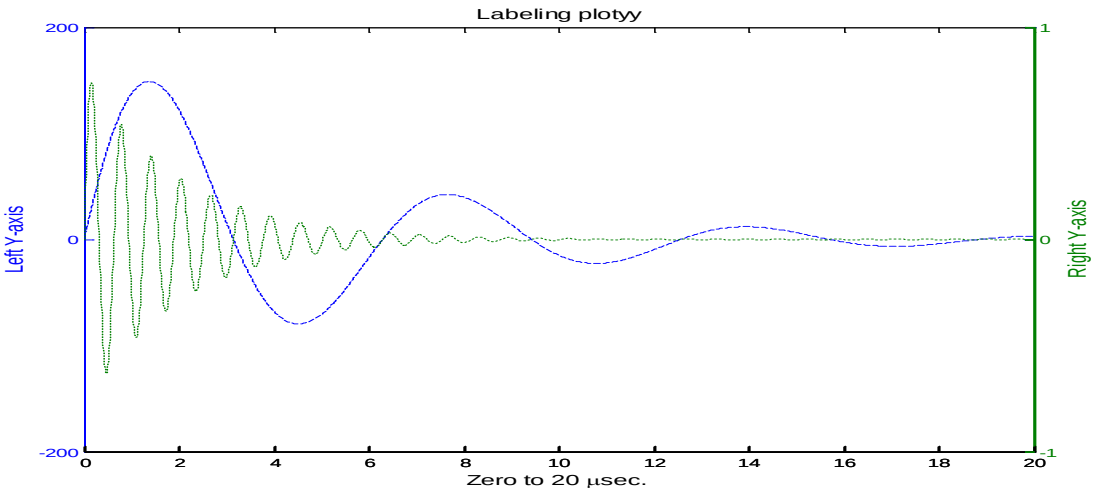


Рис.4.7

Технологии построения аналитических моделей. Анализ поведения многих систем и устройств в динамике, а также решение многих задач в теории колебаний и в поведении упругих оболочек обычно базируются на решении систем *обыкновенных дифференциальных уравнений* (ОДУ). Их, как правило, представляют в виде системы дифференциальных уравнений первого порядка в форме Коши:

$$\frac{dy}{dt} = f(y,t),$$

с граничными условиями $y(t_0\ t_{end}, \boldsymbol{b})$, где t_0 , t_{end} — начальные и конечные точки интервалов. Параметр t не обязательно означает время, хотя чаще всего дифференциальные уравнения решают во временной области. Вектор $\boldsymbol{b}$ задает начальные и конечные условия.

Рассмотрим технологии построения аналитических моделей на основе численных методов решения обыкновенных дифференциальных уравнений (ОДУ) и некоторые вспомогательные функции, полезные для решения систем ОДУ. Для решения систем ОДУ в MATLAB реализованы различные методы. Их реализации названы *решателями* ОДУ.

В MATLAB обобщенное название solver (решатель) означает один из возможных численных методов решения ОДУ. Решатели реализуют следующие методы решения систем дифференциальных уравнений.

ode45 (f, interval, y0 [, options]) — одношаговые явные методы Рунге-Кутта 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения. Во многих случаях он дает хорошие результаты;

ode23(f, interval, y0 [, options]) — одношаговые явные методы Рунге-Кутта 2-го и 4-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения;

ode113 (f, interval, y0 [, options]) — многошаговый метод Адамса-Башворта-Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения;

ode23tb (f, interval, y0 [, options]) — неявный метод Рунге-Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем.

Несмотря на сравнительно низкую точность, этот метод может оказаться более эффективным, чем ode15s;

ode15s (f, interval, y0 [, options]) — многошаговый метод переменного порядка (от 1 до 5, по умолчанию 5), использующий формулы численного дифференцирования. Это адаптивный метод, его стоит применять, если решатель ode45 не обеспечивает решения;

ode23s (f, interval, y0 [, options]) — одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка. Может обеспечить высокую скорость вычислений при низкой точности решения жесткой системы дифференциальных уравнений;

ode23t (f, interval, y0 [, options]) — метод трапеций с интерполяцией. Этот метод дает хорошие результаты при решении задач, описывающих колебательные системы с почти гармоническим выходным сигналом;

bvp4c служит для проблемы граничных значений систем дифференциальных уравнений вида $y'=f(t,y)$, $F(y(a), y(b), p)=0$ (краевая задача);

pdepe нужен для решения систем параболических и эллиптических дифференциальных уравнений в частных производных, введен в ядро системы для поддержки новых графических функций Open GL, пакет расширения Partial Differential Equations Toolbox содержит более мощные средства.

Входными параметрами этих функций являются:

- f – вектор функция для вычисления правой части уравнения или системы уравнений;
- interval – массив из двух чисел, определяющий интервал интегрирования дифференциального уравнения или системы. Для получения решений в конкретные моменты времени t0, tl,..., tfinal (расположенные в порядке уменьшения или увеличения) нужно использовать interval = [t0 tl ... tfinal];
- y0 – вектор начальных условий;
- options – параметры управления ходом решения дифференциального уравнения или системы аргумент, создаваемый функцией odeset (еще одна функция — odeget или bvpget (только для bvp4c)— позволяет вывести параметры, установленные по умолчанию или с помощью функции odeset /bvpset);.
- p1, p2,.,, — произвольные параметры, передаваемые в функцию F.

Все решатели могут решать системы уравнений явного вида $y' = f(t, y)$. Решатели ode15s и ode23t способны найти корни дифференциально-алгебраических уравнений $M(t)y' = f(t, y)$, где M называется матрицей массы. Решатели ode15s, ode23s, ode23t и ode23tb могут решать уравнения неявного вида $M(t,y)\, y' = f(t, y)$. И наконец, все решатели, за исключением ode23s, который требует постоянства матрицы массы и bvp4c, могут находить корни матричного уравнения вида $M(t, y)\, y' = f(t, y)$. ode23tb, ode23s служат для решения жестких дифференциальных уравнений . ode15s -жестких дифференциальных и дифференциально-алгебраических уравнений, ode23t -умеренно жестких дифференциальных и дифференциально-алгебраических уравнений.

T, Y – матрица решений Y, где каждая строка соответствует времени, возвращенном в векторе-столбце T.

Перейдем к описанию функций для решения систем дифференциальных уравнений:

[T,Y] = solver(@f, interval,y0) — где вместо solver подставляем имя конкретного решателя — интегрирует систему дифференциальных уравнений вида *y'=F(t,y)* на интервале interval с начальными условиями y0. @f — дескриптор (правая часть) ODE-функции. Каждая строка в массиве решений Y соответствует значению времени, возвращаемому в векторе-столбце T;

[T,Y] = solver(@f, interval, y0, options) — дает решение, подобное описанному выше, но с параметрами, определяемыми значениями аргумента options, созданного функцией odeset;

[T,Y] = solver(@f,interval,y0,options,pl,p2...) — дает решение, подобное описанному выше, передавая дополнительные параметры pi, p2,... в m-файл F всякий раз, когда он вызывается. Используйте options=[ ], если никакие параметры не задаются;

Параметры интегрирования (options) могут быть определены и в m-файле, и в командной строке с помощью команды odeset. Если параметр определен в обоих местах, определение в командной строке имеет приоритет.

Решатели используют в списке различные параметры. В приведенной ниже таблице они даны для решателей обычных (в том числе жестких) дифференциальных уравнений.

Способы решения дифференциальных уравнений в системе MATLAB, так же как и в других системах, могут быть разные, в частности с помощью использования численных методов (программированием) и использования готовых процедур самого пакета.

Метод использования численных методов. Рассмотрим задачу Коши:

$$\frac{dy}{dt} = -y^2 + \frac{1}{t^2}, \qquad y(1)=0$$

Для решения этого уравнения методом Эйлера создадим следующую функцию в М – файле.

```
% функция задающая решение по Эйлеру
function [y,t]=eiler(a,b,n,y0)
  h=(b-a)/n;
  y(1)=y0;
  for i=1:n
     t(i)=a+i*h;
  end
```

```
    for i=1:n-1
       y(i+1)=y(i)+h*(-y(i)*y(i)+1/(t(i)*t(i)));
    end
end
```

Эта функция должна быть сохранена, например, в следующем каталоге :

C:\MATLAB6p5\toolbox\matlab\funfun, т.е. там где хранятся все функции системы MATLAB. При начале вызова этой функции необходимо удостоверится о наличии директории, где располагаются ваши М-файлы

Current Directory | C:\MATLAB6p5\toolbox\matlab\funfun |

т.е. в каком каталоге находится необходимая функция. Если необходимые функции хранятся в другом каталоге, то вы должны указать соответствующую директорию.

```
>>[y,t]=eiler(0.01,100,1000,0.1)
>>figure(1);plot(t,y);
```

Найдем численное решение дифференциального уравнения с помощью одного из решателей ode113, ode23, ode45 и др. Они применяются для разных систем (мягкие, жесткие) и используют различные схемы численного интегрирования. Но для любой из них обязательно нужно задать систему, которую надо решить, начальные условия, интервал и шаг поиска решения.

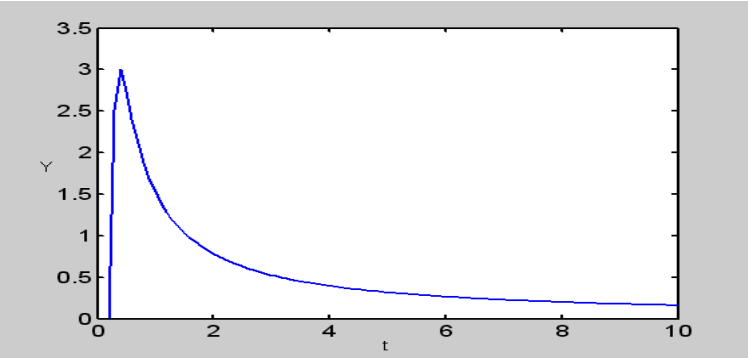


Рис.4.8

Систему уравнений нужно привести к нормальному виду и записать в файл функцию, которая по данным значениям аргумента x и всех функций $y_1, y_2, ..., y_n$ вычисляет производные в этой точке. Этот файл нужно назвать так, как называется функция (расширение дать m), и поместить в какой-либо каталог, доступный MATLAB.

Например:

```
function f=ff(t,y)
f=-y(1)*y(1)+1/(t*t);
```

end

Решение

>>[y1,t1]=ode23(@ff,[0.2 10], 0.001)

>> figure(3);plot(y,t,':',y1,t1);

Результаты решения по методу Эйлера и с помощью решателя **ode23** представлены на рис. 4.9.

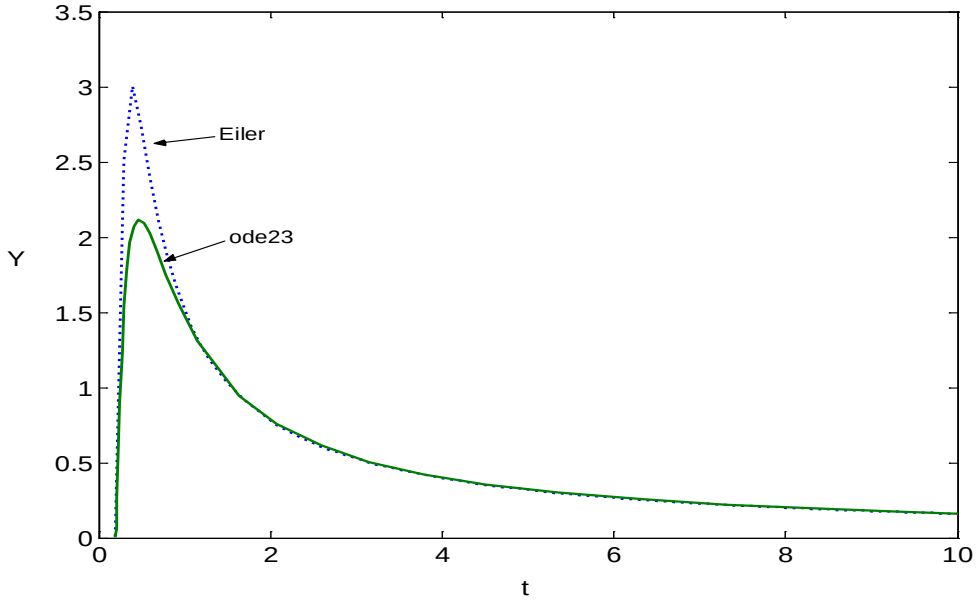


Рис.4.9

Решение систем дифференциальных уравнений. Рассмотрим задачу о маятнике Фуко. Математическую модель этой задачи можно представить в виде:

$$\frac{dy_1}{dt} = 2y_1\omega + \omega^2 y_3 - g\frac{y_3}{L},$$

$$\frac{dy_2}{dt} = 2y_2\omega + \omega^2 y_4 - g\frac{y_4}{L}, \quad (3.58)$$

$$\frac{dy_3}{dt} = y_1, \qquad \frac{dy_4}{dt} = y_2.$$

Здесь ω - частота колебания маятника Фуко, L – длина маятника, g – значение ускорения свободного падения.

Создадим М-файл

function dy=fuko(t,y);

g=9.8; L=50; w=0.04;

dy=zeros(4,1)

dy(1)=2*y(2)*w+w*w*y(3)-g*y(3)/L;

dy(2)=-2*y(1)*w+w*w*y(4)-g*y(4)/L;

```
  dy(3)=y(1);
  dy(4)=y(2);
end
```

Решение находим с помощью решателя ode23 и визуализируем в виде графиков (рис.4.10)

```
>> [T,Y]=ode23(@fuko,[0 100],[0;0;1;0])
>> figure(5); plot(T,Y,'-k');
>> figure(6); plot(Y(:,3),Y(:,4),'k');
```

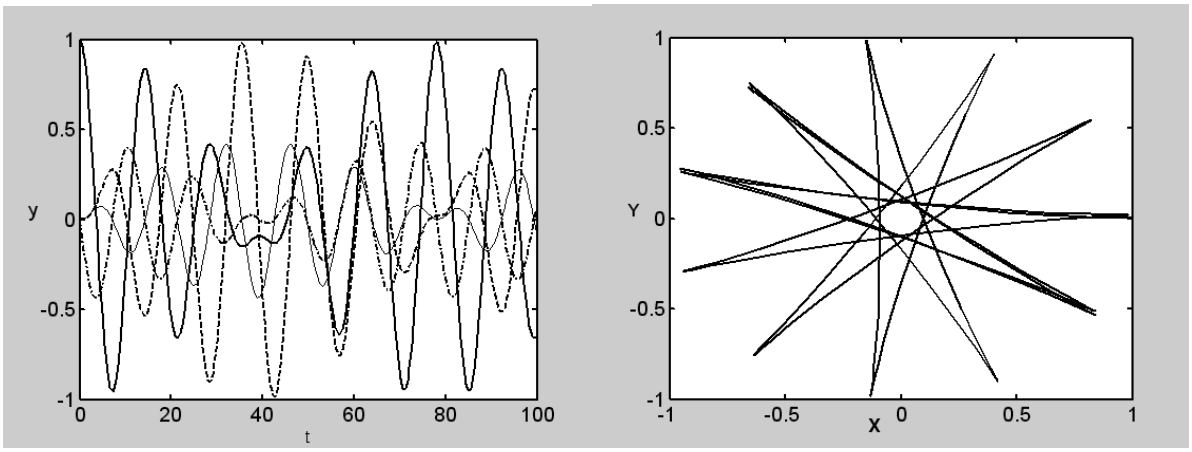


Рис.4.10

Аналитическое решение дифференциальных уравнений.

Рассмотрим следующее уравнение

$$y''+2y'+y = x * \sin(2 * x)$$

```
>> clear  all  % очистили память
eq='D2y+2*Dy+y-x*sin(2*x)' ; % уравнение
x0=-10;  % начальная точка
xk=10;  % конечная точка
y0=1; % начальное условие
y10=-1; % начальное условие
ic1=sprintf ( 'y (%d) =%d', x0,y0); % строка с начальным условием
ic2=sprintf ( 'Dy (%d) =%d' , x0,y10); % строка с начальным условием
fprintf (' Дифференциальное уравнение : \n % s=0\n ', eq);
fprintf (' Начальные условия : \n % s \n % s\n ', ic1, ic2);
```

```
 Дифференциальное уравнение :
 D2y+2*Dy+y-x*sin(2*x)=0
  Начальные условия :
```

y (0) =1

Dy (0) =-1

Нахождение аналитического решения

>> [y]= dsolve('D2y=-2*Dy-y+x*sin(2*x)','y(0)=1','Dy(0)=0','x')

y = -4/25*cos(2*x)*x-4/125*cos(2*x)-3/25*x*sin(2*x)+22/125 * sin(2* x) + 129/125*exp(-x)+21/25*exp(-x)*x

Для численного решения уравнения приведем наше уравнение 2-го порядка к системе двух уравнений 1-го порядка заменой:

$$y'_1 = y_2$$

$$y'_2 = -2 * y_2 - y_1 + x * \sin(2 * x)$$

Создадим М –файл

```
function dy=fff(x,y)
dy=zeros(2,1);
dy(1)=y(2);
dy(2)=-2*y(2)-y(1)+x*sin(2*x);
end
```

Для численного решения системы дифференциальных уравнений, кроме функции вычисления правых частей, нужно задать также начальные условия и массив точек, в которых будет вычисляться решение.

Начальные условия у нас есть — это переменные y0 и y10. Решаем систему уравнений решателем ode23.

```
[X,Y]=ode23(@fff,[-20 20],[1;-1])
```

В результате получаем: массив аргументов X — это точки, в которых найдено решение, и Y — двумерный массив функций. Каждый столбец массива Y — это одна из переменных $y_1, y_2, ..., y_n$ в точках X. В нашем случае 1-й столбец Y — это функция y(x), а 2-й — ее производная. Для построения графика нам нужен будет 1-й столбец массива УУ.

```
figure(6);plot(Y(:,1));
```

Надписываем заголовок и метки осей выбранным шрифтом. На рис.4.11 сохранение графика проводилось через **File → Export** в окне **Figure № 6** формате *.tif.

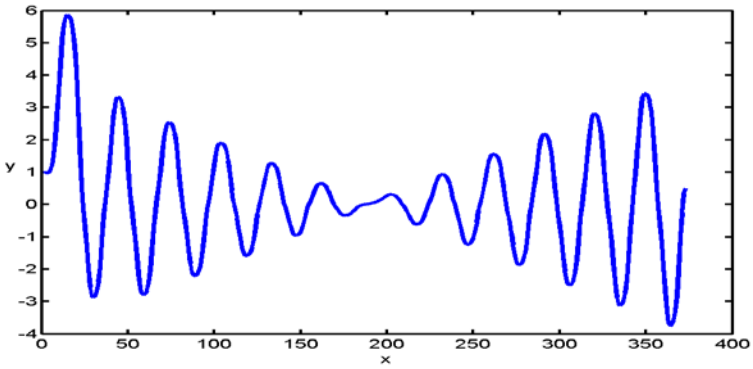


Рис.4.11

Технологии разработки компьютерных моделей с помощью систем компьютерной математики MATLAB описаны в книгах [1, 4, 7, 8, 18, 22].

Задание на моделирование:

1. Построить аналитические модели (рис.4.6 - 4.11) в среде моделирования MATLAB.
2. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

4.2. Система технического (кибернетического) моделирования VisSim

Наряду с развитием цифровых вычислительных машин формировалось направление аналоговых вычислительных машин (АВМ), с помощью которых решались различные физические и математические задачи. АВМ позволяли решать различные виды математических моделей, представленных в виде дифференциальных уравнений с помощью натурного схемотехнического моделирования. Скорость вычисления на таких машинах достигала до 50 млн. операций в секунду. Для 1960-х годов такая скорость вычислений была фантастической. АВМ использовали для математического и полунатурного моделирования ракет и ракетных комплексов, космических кораблей, самолетов, судов, энергетических установок и других объектов на всех этапах их создания. Кроме того, АВМ применялись для решения задач в медицине, биологии, химии и в других направлениях науки и техники.

Аналоговые ЭВМ не получили массового распространения и в настоящее время разрабатываются для решения специфических задач. С развитием цифровых вычислительных средств (ЭЦВМ-ЭВМ) начали развиваться программные средства и технологии моделирования на цифровых ЭВМ. Появление современных технических информационных систем объектно-

ориентированного моделирования в виде компьютерных виртуальных конструкторов, таких как MultiSim, Vissim, MvStudium, LabVIEW, приложение SIMULINK системы MATLAB и других систем, позволяют решать математические и инженерные задачи с помощью схемотехнического имитационного моделирования. Эти системы в некотором смысле можно назвать виртуальными аналогами АВМ. Этот подход к моделированию называют кибернетическим или аналоговым. Системы технического моделирования построены по принципу конструктора, по системе блоков. В системах технического моделирования можно решать как математические, так и инженерные задачи. В этих компьютерных системах можно собирать и конструировать виртуально любые электротехнические и радиотехнические схемы с использованием компьютерных аналогов электротехнических деталей и измерительных, а также визуальное моделирование и конструирование инженерных, технических имитаторов электронных приборов и логических устройств. С помощью систем технического моделирования создаются виртуальные имитационные модели (компьютерные установки).

Более того, проектированные и созданные виртуальные инженерные и производственные компьютерные объекты и установки можно использовать для натурного эксперимента и производственных испытаний в реальном масштабе времени.

В качестве примера кибернетического подхода к моделированию рассмотрим систему схемотехнического моделирования VisSim ориентированную на визуальное программирование и моделирование [4]. Она интегрируется с СКМ, в частности системами Mathcad и MATLAB. Данная система широко используется для решения инженерных задач в области автоматического регулирования и управления, а также при моделировании различных физических, химических, экономических и прочих явлений и систем.

При построении моделей в системе VisSim используются блоки, которые хранятся в библиотеке блоков и могут быть извлечены из нее и перенесены в окно модели. Блоки ориентированы на реализацию математических и логических операций. Это позволяет создавать математически прозрачные модели с любыми описывающими их зависимостями.

Система Vissim позволяет проводить интегрирование в цифровой форме – которая является более устойчивой операцией, чем дифференцирование, здесь необходимо преобразовывать исходные дифференциальные уравнения (ДУ) в операторной форме таким образом, чтобы оператор Лапласа входил в уравнение только с отрицательными степенями (требуется разде-

лить и левую и правую часть ДУ на оператор Лапласа в соответствующей степени). Это основной принцип построения моделей в программе VisSim.

Разработчиками VisSim рекламируется как самая быстрая система моделирования. И действительно, время моделирования систем и устройств в VisSim от 3 до 8 раз меньше, чем у других систем моделирования. Это большое достоинство системы, нередко оправдывающее переход к ее применению. Поэтому тема данного параграфа может быть актуальной. Запустив систему VisSim из Windows, вы увидите на экране окно (рис. 4.12). Над ним видна строка с основными элементами интерфейса. Опции главного меню, содержащиеся в этой строке, легко изучить самостоятельно; некоторые из них очень похожи на стандартные опции, принятые в текстовых редакторах Windows.

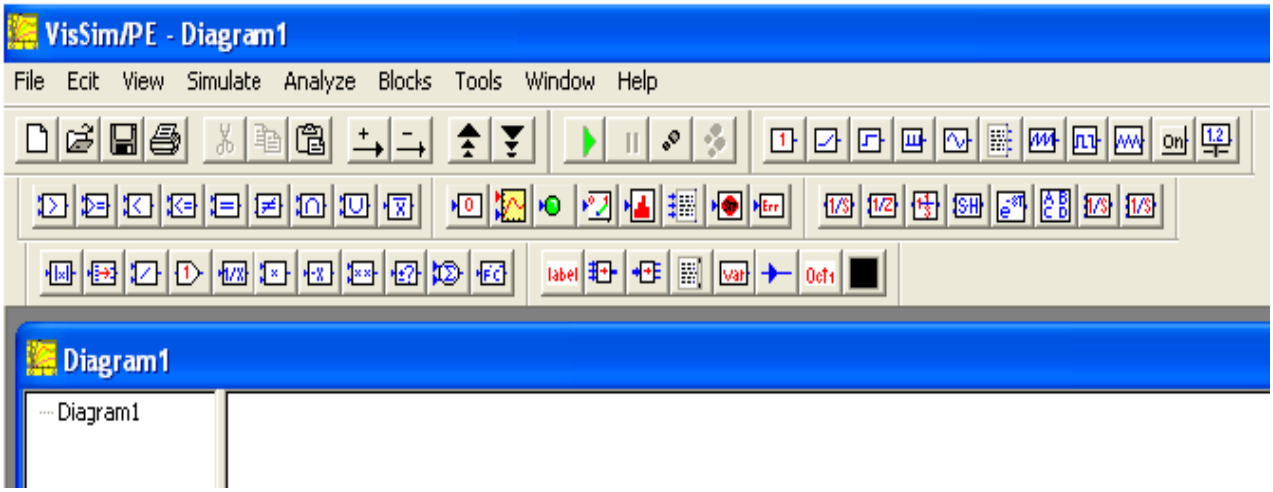


Рис. 4.12

Работа с документами VisSim обычно не требует обязательного использования возможностей главного меню, так как основные из них дублируются кнопками на панели инструментов, которые расположены в удобных, перемещаемых с помощью мыши наборных панелях–палитрах. Наборные панели появляются в окне редактирования документов при активизации кнопок. Они служат для вывода заготовок – шаблонов математических знаков (цифр, знаков арифметических операций, матриц, знаков интеграла, производных, приделов и др.). Система VisSim предназначена для решения задач математического моделирования, относящихся к следующим классам:

- линейные системы;
- нелинейные системы;
- непрерывные во времени системы;
- дискретные во времени системы;
- системы с изменяемыми во времени параметрами;

• гибридные системы;

• многоцелевые и многокомпонентные системы;

• одновходовые и одновыходные (одномерные) системы *SISO*;

• многовходовые и многовыходовые (многомерные) системы *M1MO*;

Система VisSim не имеет явной ориентации на какой-то класс моделирования. Это универсальная система, допускающая достаточно простое расширение и обеспечивающая легкую адаптацию под решение тех или иных конкретных задач пользователя. Тем не менее, можно считать, что наиболее удобна данная система для решения инженерных задач в области автоматического регулирования и управления, а также при моделировании различных физических, химических, экономических и прочих явлений и систем.

Для построения моделей в системе VisSim используются блоки, которые хранятся в библиотеке блоков и могут быть пернесены в окно модели и соединяться друг с другом.

Библиотека блоков, представленная в позиции Blocks (Блоки) меню (рис. 4.13) и инструментальными панелями, содержит следующие «тома»:

• **Animation** — блоки создания анимационных клипов;

•**Annotation** — блоки создания комментариев и определения переменных;

• **Arithmetic** — блоки арифметических и близких к ним операций;

• **Boolean** — блоки задания операций Булевой алгебры;

• **DDE** — блоки интерфейса;

• **Integration** — блоки задания операций интегрирования;

• **Linear Systems** — блоки задания параметров пространства состояний линейных систем и их передаточных функций;

• **MATLAB Interface** — блоки интеграции с матричной системой MATLAB;

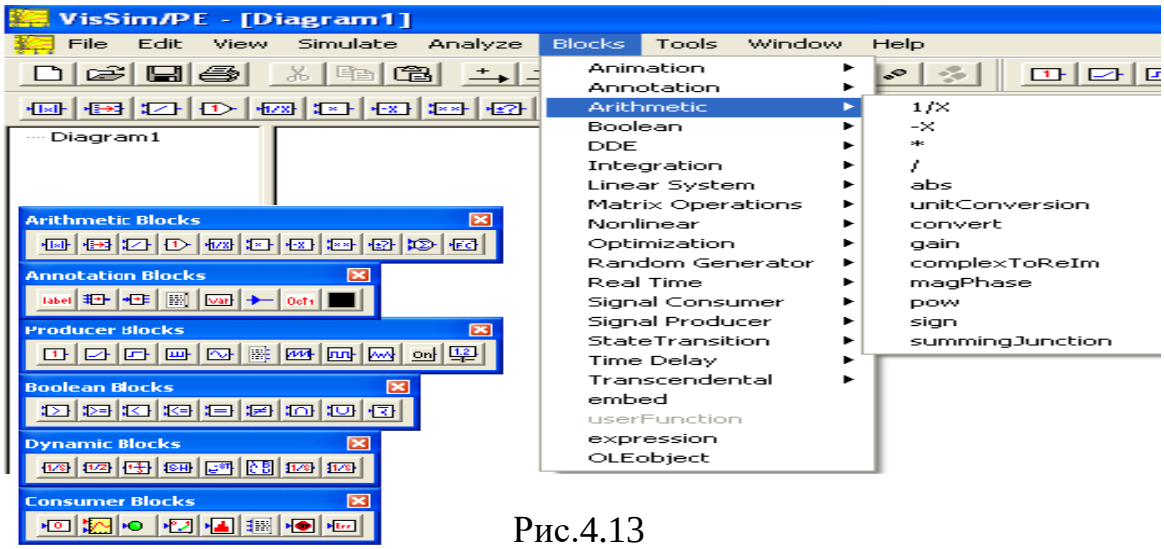


Рис.4.13

• **Matrix Operations** — блоки задания матричных операций;
• **Nonlinear** — блоки нелинейных операций и создания нелинейных систем;
• **Optimization** — блоки задания операций оптимизации;
• **Random Generator** — блоки генерации случайных чисел;
• **Real Time** — блоки для систем реального времени;
• **Signal Consumer** — блоки регистрации, индикации и построения графиков сигналов;
• **Signal Producer** — блоки создания сигналов;
• **Time Delay** — блоки создания временной задержки;
• **Transcendental** — блоки задания трансцендентных математических функций;
• **General** — функции общего характера.

На рис. 4.14 представлены блоки арифметических, интегро-дифференциальных и логических операций.

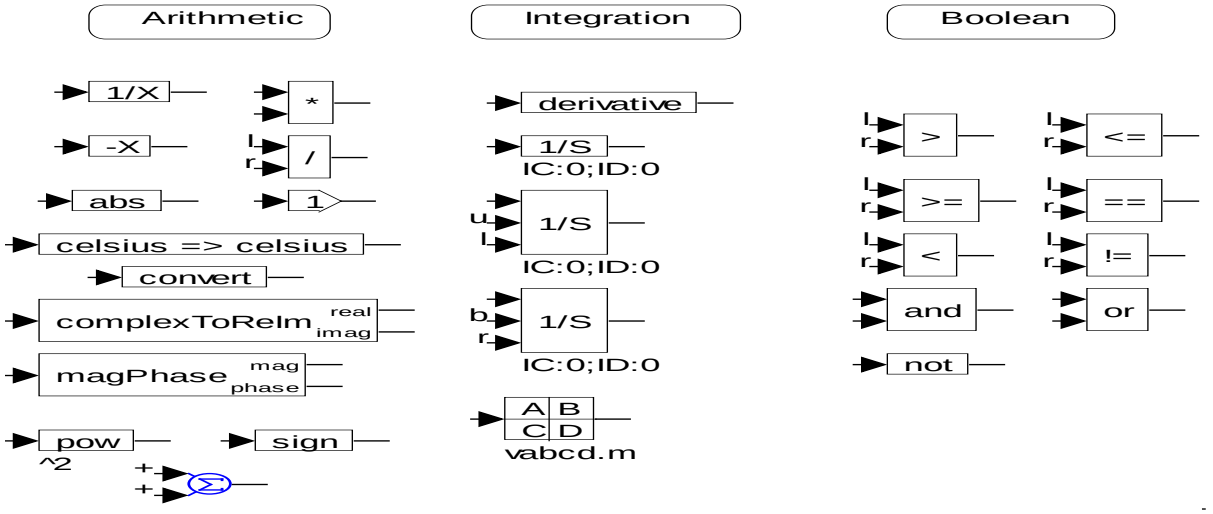


Рис.4.14

Приведем в целях знакомства работы с данной системой технологии моделирования различных дифференциальных уравнений [4]. Рассмотрим способ использования блока **variable** системы схемотехнического моделирования для решения дифференциальных уравнений на конкретных примерах.

Пример 1. Решение уравнений первого порядка .

Рассмотрим уравнение первого порядка

$$\frac{dy}{dt} = -y + r(t), \qquad r(t) = 1$$

Для построения схемотехнической модели дифференциального уравнения в левой части оставляем старшую производную, все остальные члены переносим в правую часть. Соберем схему с использованием суммы и значение присвоим переменной ***ydot,*** проведем интегрирование, результат просуммируем с начальным условием (в нашем расчете оно равно -9.2). Результат суммирования присвоим переменной Y, которую соединим осциллографом. На осциллографе будет выдан результат (рис.4.15).

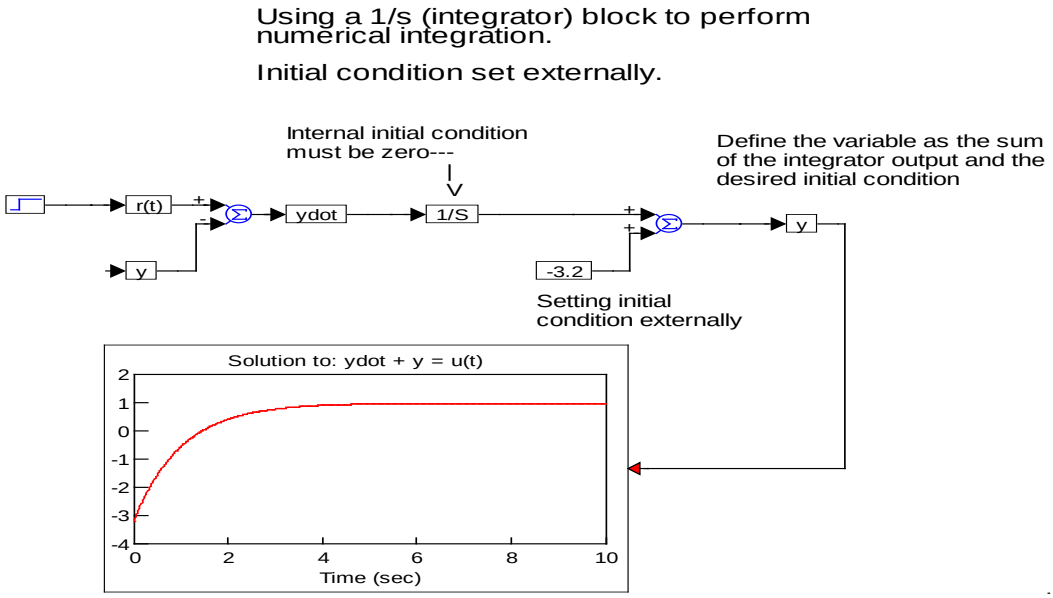


Рис.4.15

Пример 2. Решение уравнения второго порядка с помощью блока **variable** проводится аналогично примеру 1.

$$r(t) - y\frac{dy}{dt} - 2y = \frac{d^2y}{dt^2},\ r(t) = 1$$

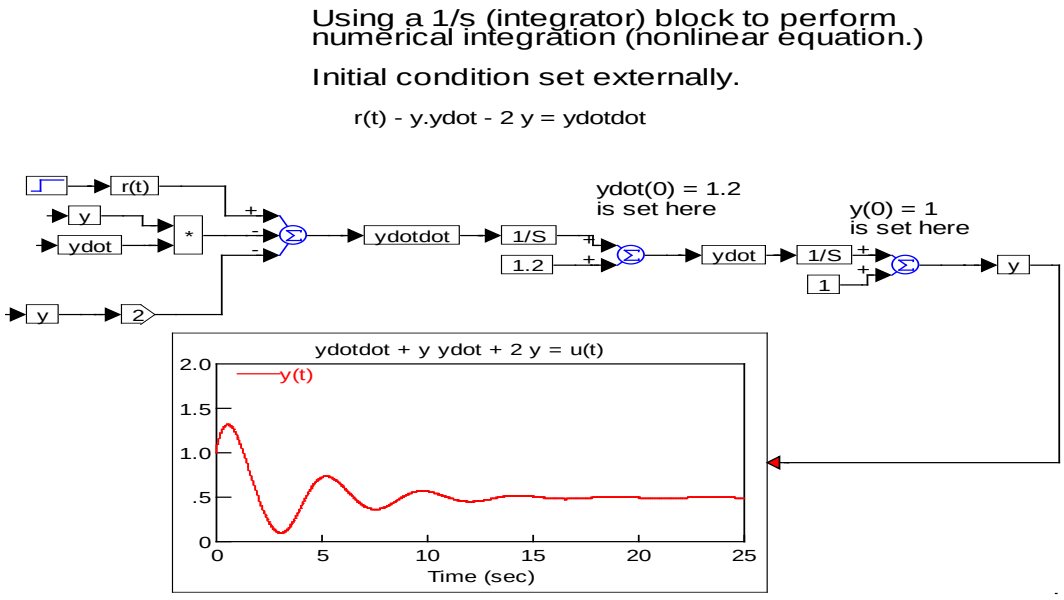


Рис.4.16

При решении уравнений можно опустить блок **variable,** в этом случае необходимо результаты операции присоединить соответствующим блокам.

Пример 3. Если исследуемая система описана ДУ, решаемая относительно одной из координат системы:

$$\frac{d^3x}{dt^3} + A_1\frac{d^2x}{dt^2} + A_2\frac{dx}{dt} + A_3x = f,$$

то его можно представить в следующем виде:

$$\frac{d^3x}{dt^3} = f - A_1\frac{d^2x}{dt^2} - A_2\frac{dx}{dt} - A_3x$$

$$s^3x = f - A_1s^2x - A_2s^1x - A_3x,$$

тогда соответствующая блок-схема будет выглядеть, как показано на рис.4.17

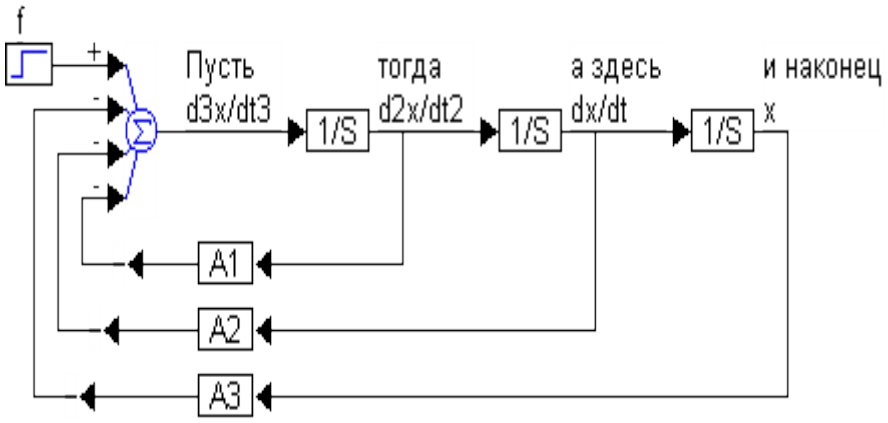


Рис.4.17

Где *x* - координата системы, относительно которой решено ДУ - это может быть ошибка системы *x(t)* или воздействие на объект *u(t),* или выходная координата - *y(t),* A1, ... , A3 - постоянные коэффициенты (если система не является зависимой от параметра) - суммы и произведения постоянных времени *Tj*, а так же коэффициенты усиления *Kn*, *f* - любое воздействие на систему - сигнал задания *g(t),* или помехи *fj(t).*

Пример 4. Если исследуемая система описана совокупностью дифференциальных уравнений первого порядка:

$$\left.\begin{aligned}\frac{dx_1}{dt} &= a_{11}x_1 + a_{21}x_2 + a_{31}x_3 + a_1 f_1\\ \frac{dx_2}{dt} &= a_{12}x_1 + a_{22}x_2 + a_{32}x_3 + a_2 f_2\\ \frac{dx_3}{dt} &= a_{13}x_1 + a_{23}x_2 + a_{33}x_3 + a_3 f_3\end{aligned}\right\},$$

тогда блок-схема системы уравнений будет выглядеть, так как показано на рис.4.18:

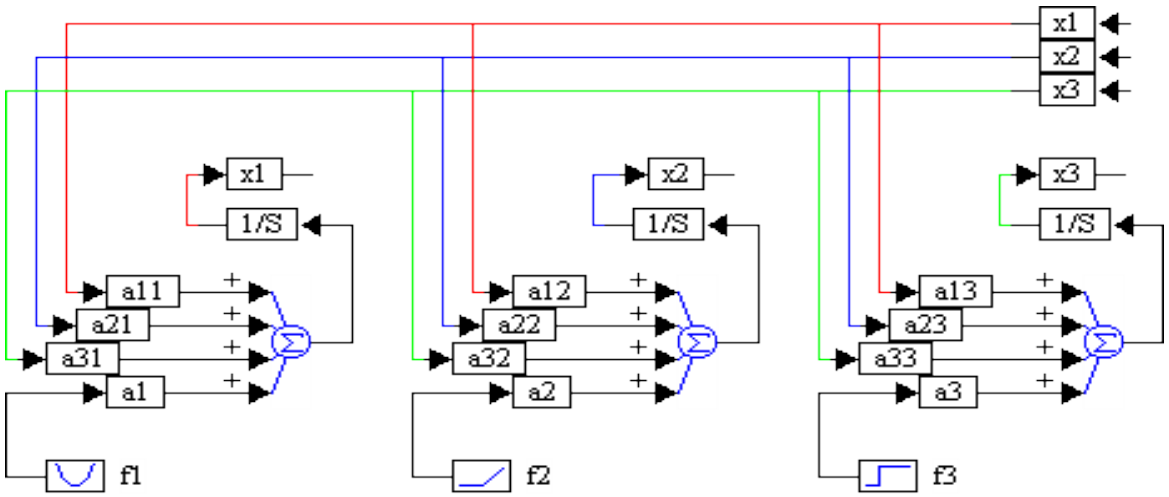


Рис.4.18

Где x1, x2, x3 - собственные координаты системы, x(t) – ошибка системы u(t) – воздействие на объект, выходная координата - y(t), a11, ... , a33 - постоянные коэффициенты (если система не является зависимой от параметра) - суммы и произведения постоянных времени Tj, коэффициентов усиления Kn;

- f1, f2, f3 - воздействия на систему - сигнал задания g(t), помехи $f_j(t)$.

Более подробно работы с системой VisSim мы рекомендуем работу [4] и сайт http://model. exponenta.ru /help / 0050.htm.

Задание на моделирование:

1. Построить схемотехнические модели (рис.4.15 - 4.18) в среде моделирования VisSim.
2. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

4.3. Среда объектно-ориентированного программирования GPSS

В рамках предлагаемой среды имитационного моделирования GPSS World можно рассмотреть основные подходы к моделированию систем различной природы. Для проведения практических занятий часто используют свободно распространяемую версию системы GPSS World Student Version. Выбранная система моделирования накладывает определенные условия на тип моделей, а точнее, реализуются так называемые модели массового обслуживания. И это вполне подходит для моделирования многих процессов, связанных с системами массового обслуживания, в том числе и для моделирования процессов, происходящих в компьютерах и вычислительных системах. Класс моделей можно расширить, используя другие системы моделирования, например Matlab/Simulink, VenSim, PowerSim, ModelMaker и другие.

Система GPSS (General Purpose System Simulator) предназначена для написания имитационных моделей систем с дискретными событиями. Наиболее удобно в системе GPSS описываются модели систем массового обслуживания, для которых характерны относительно простые правила функционирования составляющих их элементов. Программа GPSS World разработана компанией Minuteman Software. Используемая для лабораторных работ, версия программы GPSS World Student Version является свободно распространяемой и ее можно скачать с официального сайта разработчика www.minutemansoftware. com.

В системе GPSS моделируемая система представляется с помощью набора абстрактных элементов, называемых объектами. Каждый объект принадлежит к одному из типов объектов. Объект каждого типа характеризуется определенным способом поведения и набором атрибутов, определяемых типом объекта. Например, если рассмотреть работу порта, выполняющего погрузку и разгрузку прибывающих судов, и работу отделения банка, обслуживающего клиента, то можно заметить большое сходство в их функционировании (рис.4.19). В обоих случаях имеются объекты, постоянно присутствующие в системе (порт и кассир или банкомат), которые обрабатывают поступающие в систему объекты (корабли и посетители банка). В теории массового обслуживания эти объекты называются приборами и заявками. Когда обработка поступившего объекта заканчивается, он покидает систему. Если в момент поступления заявки прибор обслуживания занят, то заявка становится в очередь, где и ждет до тех пор, пока прибор не

освободится. Очередь также можно представлять себе как объект, функционирование которого состоит в хранении других объектов.

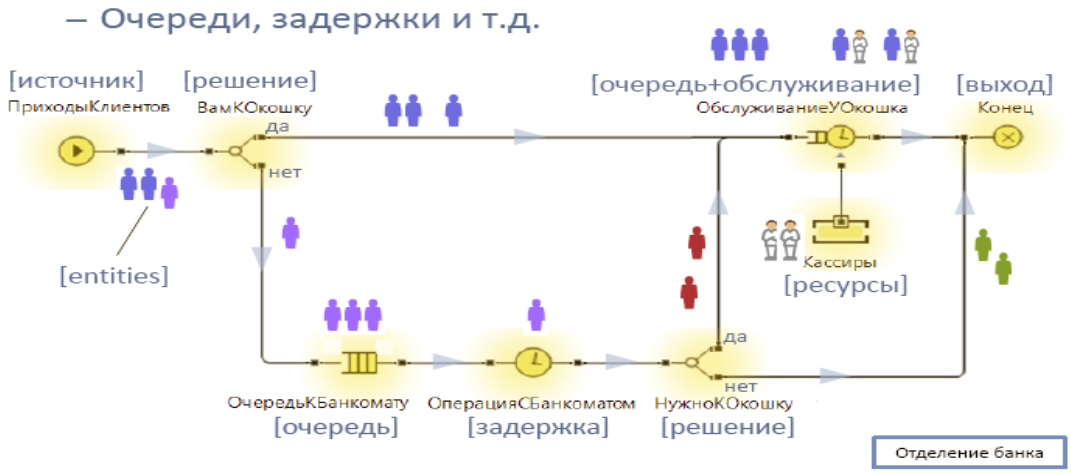


Рис. 4.19.

Каждый объект может характеризоваться рядом атрибутов, отражающих его свойства. Например, прибор обслуживания имеет некоторую производительность, выражаемую числом заявок, обрабатываемых им в единицу времени. Сама заявка может иметь атрибуты, учитывающие время ее пребывания в системе, время ожидания в очереди и т.д. Характерным атрибутом очереди является ее текущая длина, наблюдая за которой в ходе работы системы (или ее имитационной модели), можно определить ее среднюю длину за время работы (или моделирования). В языке GPSS определены классы объектов, с помощью которых можно задавать приборы обслуживания, потоки заявок, очереди и т.д., а также задавать для них конкретные значения атрибутов.

Современное развитие GPSS получило в 90-е годы, когда появились работы и книги, автором которых был профессор Борис Яковлевич Советов (ЛЭТИ). Серия учебников Б.Я.Советов, С.А.Яковлев "Моделирование систем" (1988-2012 г.г.) стала стандартом "де-факто" при разработке и проведении курсов по моделированию систем во многих вузах[27].

О возрождении интереса к языку говорит и появлении ряда русскоязычных Интернет-ресурсов. Кроме www.gpss.ru – это www.gpss-forum.narod.ru и www.simulation.org.ua.

Система GPSS World – комплексный моделирующий инструмент, охватывающий области как дискретного, так и непрерывного компьютерного моделирования и обладающий высоким уровнем интерактивности и визуального представления информации. На сегодняшний день GPSS World является приложением семейства операционных систем Windows и использует их очевидные преимущества – графический интерфейс пользователя и архи-

тектуру «документ-вид». Программное обеспечение представляет собой полноэкранный текстовый редактор, позволяющий работать сразу с несколькими объектами (текст модели, журнал, отчет и т. д.) Кроме того, графический интерфейс дает возможность интерактивно взаимодействовать с выполняющимися процессами моделирования, а также применять наглядные графические окна для отображения их работы.

Для реализации взаимодействия GPSS World предоставляет, во-первых, механизм потоков данных. Под потоком данных в GPSS World понимается последовательность текстовых строк, используемых в процессе моделирования. С его помощью можно считывать и записывать данные в файлы текстового формата. Для управления потоками данных служат специальные блоки и процедуры. Таким образом, с помощью потоков данных можно использовать текстовые файлы, в том числе и для обмена информацией с внешними приложениями.

Второй механизм – процедуры динамического вызова. Библиотека встроенных процедур GPSS World содержит набор процедур для вызова функций, хранящихся во внешних исполняемых файлах, включая динамически подключаемые библиотеки DLL. То есть пользователь может прибегнуть к библиотекам функций сторонних разработчиков (или собственной разработки), которые порой существенно расширяют возможности системы.

GPSS World – самая современная реализация языка GPSS, дополненная вспомогательным языком PLUS. Непосредственно язык GPSS включает в себя 53 типа блоков и 25 команд, большое количество системных числовых атрибутов. Кроме того, 12 типов операторов составляют язык PLUS — Programming Language Under Simulation. Эффективность PLUS во многом обеспечивается большой библиотекой процедур.

GPSS World – объектно-ориентированный язык. Объект «Модель» главным образом содержит операторы модели, а также набор настроек. Кроме того, он включает в себя закладки и циркулярный список синтаксических ошибок.

Объект «Процесс моделирования» создается при трансляции операторов объекта «Модель». Для изменения его состояния применяются интерактивные команды и блоки.

Одной из самых сильных сторон GPSS всегда были стандартные отчеты. Содержимое отчета настраивается, поэтому пользователь получает только необходимую информацию.

Текстовый объект – это способ представления обычного текстового файла в GPSS World. В основном они применяются совместно с командами

INCLUDE для подключения набора операторов, используемого в различных моделях. Кроме того, закрепив команду INCLUDE за горячей клавишей, можно интерактивно передавать объекту «Процесс моделирования» целые списки управляющих команд.

В GPSS World применяются полиморфные типы данных. Переменные могут принимать значения одного из четырех типов. Ячейки, элементы матриц, параметры транзактов и переменные пользователя могут принимать целочисленное, вещественное, строковое и неопределенное (unspecified) значение. Неопределенные значения используются при проведении дисперсионного анализа и указывают на отсутствующие данные. Значения времени могут быть целыми или вещественными. Преобразование типов происходит автоматически. Для работы со строковыми значениями в библиотеке процедур есть ряд специальных функций.

Вычислительные системы (ВС), которые будут рассматриваться как СМО, состоят из элементов, называемых объектами аппаратной категории (устройства, памяти и логические ключи). Этими элементами могут быть компьютеры, отдельные устройства компьютеров, устройства телеобработки и т.п. Динамическими объектами в СМО являются транзакты (сообщения, заявки), это решаемые в ВС задачи, которые представляют собой единицы исследуемых потоков. Функционирование СМО представляется как процесс прохождения транзактов через фиксированную структуру объектов аппаратной и ряда других категорий.

Блоки генерации и удаления транзактов

GENERATE A,B,C,D,E – блок генерации транзактов. A – средний интервал времени между последовательными транзактами; B – разброс интервала времени относительно A; C – время начальной задержки; D – количество генерируемых транзактов, по умолчанию неограниченно; E – приоритет транзактов, по умолчанию 0. TERMINATE A – блок удаления транзакта. A – величина уменьшение счетчика числа завершений.

Блоки занятия и освобождения приборов

SEIZE A – блок занятия прибора. A – имя прибора, подлежащего занятию транзактом.

RELEASE A – блок освобождения прибора. A – имя освобождаемого прибора.

ADVANCE A,B – блок задержки транзакта в цепи будущих событий. A – средний интервал времени задержки; B – разброс интервала времени относительно A.

Операторы и блоки вычислительной категории

Name VARIABLE X – оператор описания целой переменной. Name FVARIABLE X – оператор описания действительной переменной.

Name BVARIABLE X – оператор описания логической переменной.

Name –- имя переменной; X – выражение соответствующего типа.

SAVEVALUE A,B – блок сохранения значения сохраняемой величины. A – имя или номер изменяемой ячейки; B – значение, которое надо записать в ячейку.

Компиляция и запуск имитации

Для того чтобы создать модель необходимо запустить GPSS World. Выбрать пункт меню File → New и указать, что новый файл будет моделью (Model). После чего набрать текст программы на языке GPSS и откомпилировать ее, вызвав Command → Create Simulation. Если компилятор выдал сообщение об успешной компиляции:

Model Translation Begun.
Ready.

Пример 1: Процесс прохождения заявок (транзактов), поступление которых подчиняется равномерному закону со средним значением 8 и интервалом [6,10] единиц времени, а обработка - равномерному закону со средним 5 и интервалом [2,8]. Другими словами, поступление заявок будет подчиняться равномерному закону с интервалом 7 ± 2 мин, а обработка — равномерному закону со временем обработки 5 + 2 мин.

Программа для решения представлена в системе GPSS (рис.4.20).

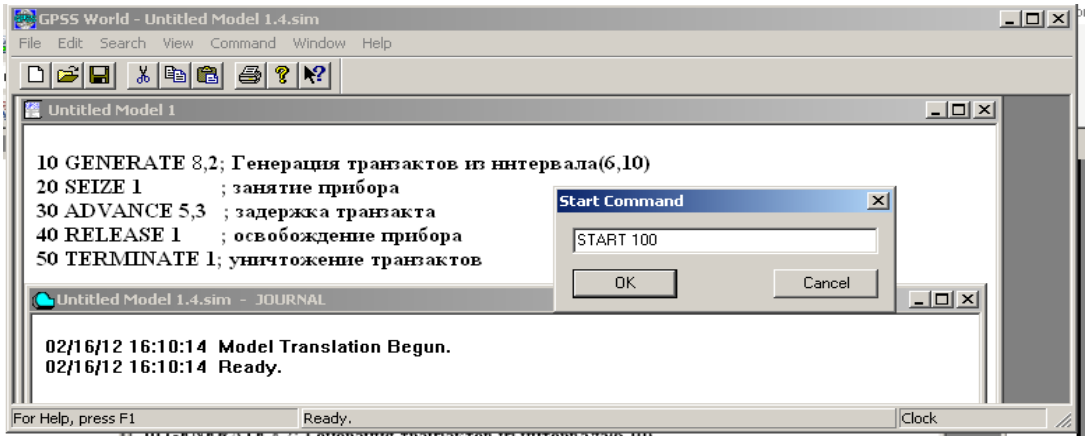


Рис.4.20

После набора программы можно провести компиляцию программы Command → Create Simulation, если ошибок нет, то запустить Start. В результате задания количества транзактов START 100 на рис. 4.21 имеем стандартный отчет (Report).

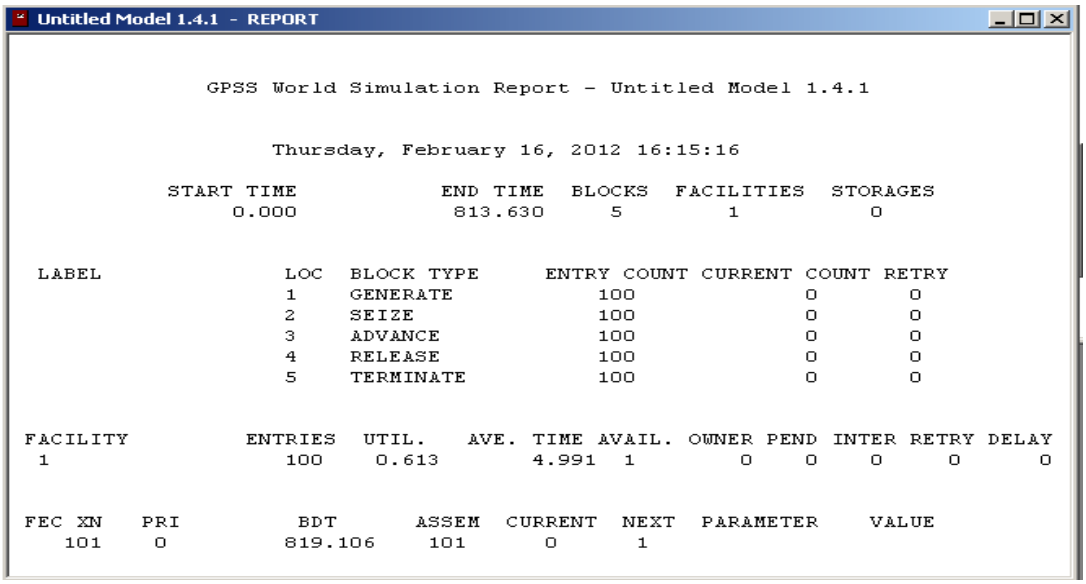


Рис.4.21

Блоки занятия и освобождения очереди

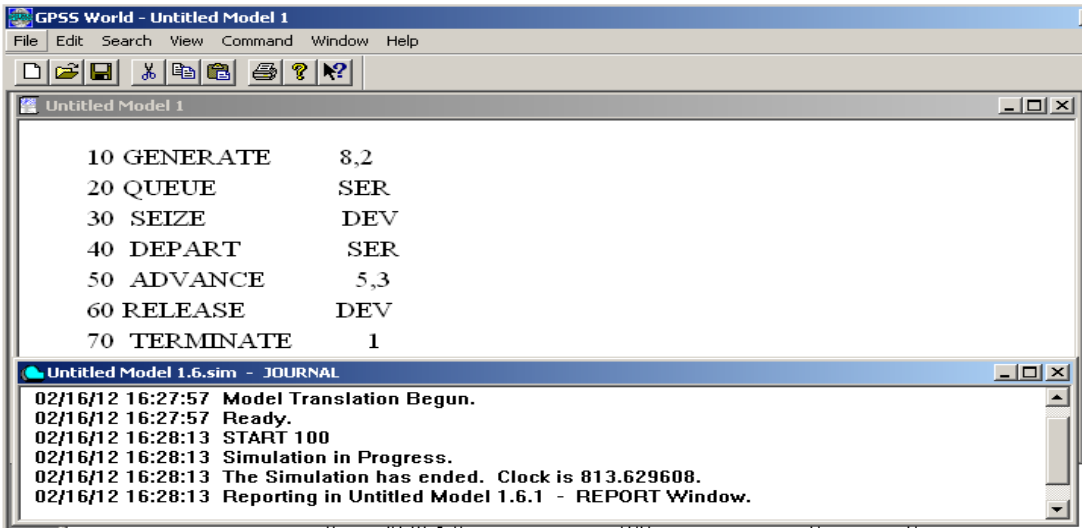
Транзакт помещается в очередь в том случае, если некоторое устройство не в состоянии обслужить его немедленно (например, устройство занято, либо память переполнена). Статистические данные об очередях могут быть получены с помощью двух типов блоков:

QUEUE A, B – блок занятия очереди. A – имя очереди; B – количество мест в очереди, занимаемое транзактом.

DEPART A,B – блок освобождения очереди. A – имя очереди; B – количество мест в очереди, освобождаемое транзактом.

Блок QUEUE может быть помещен перед любым блоком модели, в котором может возникнуть задержка. Отметим, что очередь к занятому устройству автоматически организуется пакетом моделирования независимо от того, есть в программе блок QUEUE или нет.

Пример 2: В условиях примера 1 необходимо получить статистические данные об очереди заявок, ожидающих обслуживание на приборе:



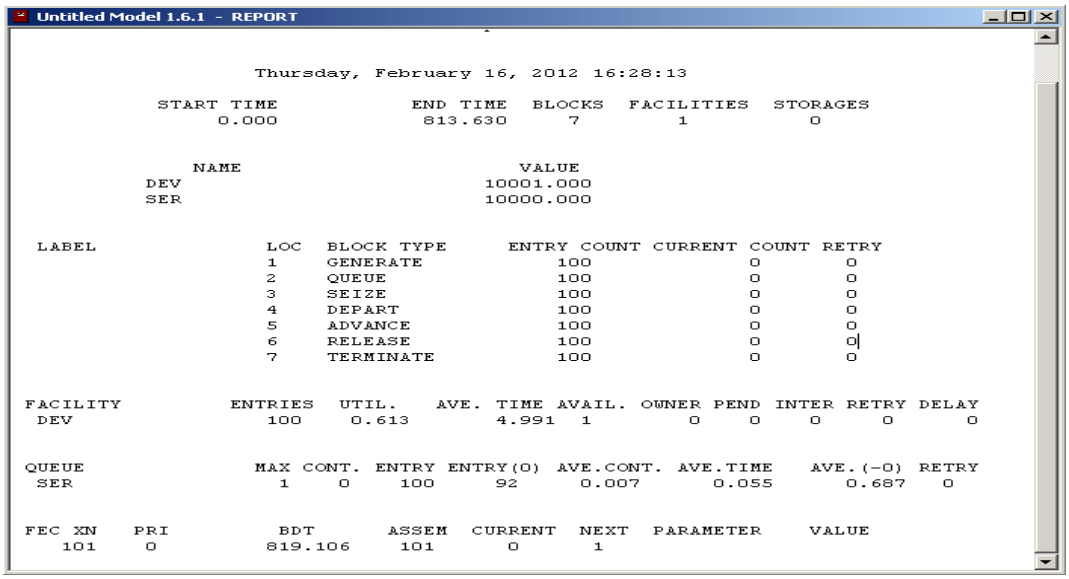


Рис.4.22

Построение гистограмм

Система GPSS позволяет строить дополнительные статистические таблицы для получения частотных распределений определенных аргументов, которыми могут быть некоторые СЧА (например, времени задержки транзакта в отдельных частях модели; длин очередей; содержимого памяти и т.п.). У каждой таблицы имеются определенные области значений аргумента. Число попаданий аргумента в каждую из этих областей регистрируется системой автоматически. В конце эксперимента результаты в таблицах выводятся на печать.

Name TABLE A,B,C,D – команда описания таблицы частотного распределения. Name – имя таблицы, A – имя переменной, значение которой табулируется, B – левая граница первого интервала таблицы, C – ширина интервалов, D – количество интервалов.

Name QTABLE A,B,C,D – команда описания таблицы времени пребывания в очереди. Name – имя таблицы, A – имя очереди, B – левая граница первого интервала таблицы, C – ширина интервалов, D – количество интервалов.

MARK A – блок отметки времени. A – номер параметра транзакта, в который заносится момент времени входа транзакта в данный блок.

TABULATE A,B – блок табулирования. A – имя таблицы, в которую заносится табулируемая величина, B – весовой коэффициент, задающий число раз занесения величины в таблицу при каждом входе в блок.

Пример 3: Получить таблицу распределения интервалов заявок по равномерному закону в интервале от 0 до 100:

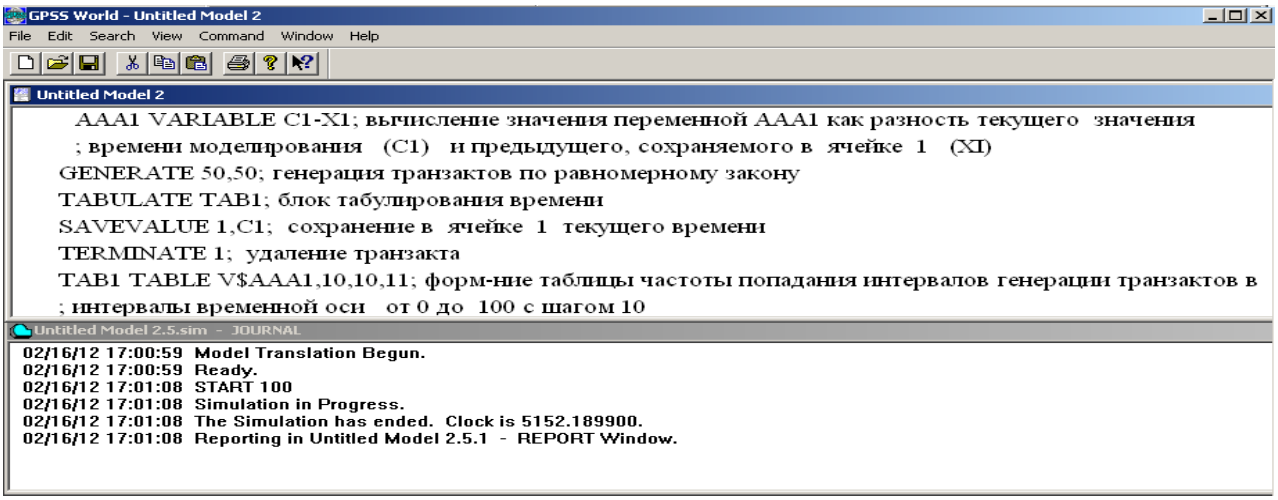


Рис.4.23. Транслированная программа

```
                    Thursday, February 16, 2012 17:01:08

          START TIME            END TIME  BLOCKS  FACILITIES  STORAGES
               0.000            5152.190     4         0          0


             NAME                          VALUE
          AAA1                          10000.000
          TAB1                          10001.000


 LABEL                LOC  BLOCK TYPE     ENTRY COUNT CURRENT COUNT RETRY
                      1    GENERATE           100              0       0
                      2    TABULATE           100              0       0
                      3    SAVEVALUE          100              0       0
                      4    TERMINATE          100              0       0


TABLE               MEAN     STD.DEV.        RANGE             RETRY FREQUENCY CUM.%
 TAB1              51.522    31.161                              0
                                          -        10.000              12     12.00
                                 10.000   -        20.000              10     22.00
                                 20.000   -        30.000              11     33.00
                                 30.000   -        40.000               5     38.00
                                 40.000   -        50.000               7     45.00
                                 50.000   -        60.000              10     55.00
                                 60.000   -        70.000               9     64.00
                                 70.000   -        80.000              14     78.00
                                 80.000   -        90.000               9     87.00
                                 90.000   -       100.000              13    100.00


SAVEVALUE                RETRY       VALUE
 1                         0        5152.190


FEC XN    PRI          BDT       ASSEM   CURRENT   NEXT   PARAMETER     VALUE
   101     0         5209.899     101       0       1
```

Рис.4.24. Результат моделирования в виде стандартного отчета

Статистика имитации. В процессе выполнения программы собирается стандартная статистическая информация, которая автоматически выводится на печать по окончании моделирования.

Содержание стандартного отчета:

- Заголовок содержит название модели, дату и время запуска модели.
- Начало (START TIME) и конец (END TIME) модельного времени, количество блоков (BLOCKS), устройств (FACILITIES) и памятей (STORAGES) в модели.
- Список имен введенных в модели, где каждому имени (NAME) приписывается число (VALUE). Значения начинаются с 10000.000.
- Список блоков, содержащий следующую информацию:

LABEL – метка блока,

LOC – позиция блока в модели (Location),

BLOCK TYPE – название блока на языке GPSS.

ENTRY COUNT – количество вошедших в блок транзактов с момента последней команды CLEAR или RESET,

CURRENT COUNT – количество транзактов в данный момент в блоке,

RETRY – количество транзактов, ожидающих входа в блок.

- Список устройств, содержащий информацию:

FACILITY – название или номер устройства,

ENTRIES – количество транзактов, занимавших это устройство блоками SEIZE или PREEMPT с момента применения команд CLEAR или RESET,

UTIL. – коэффициент использования устройства,

AVE.TIM E – среднее время одного обслуживания,

AVAIL. – доступность устройства в момент окончания имитации,

OWNER – количество транзактов, связанных с устройством,

PEND – количество транзактов, ожидающих устройство для захвата по блоку PREEMPT,

INTER – количество транзактов, прерванных при обработке блоками PREEMPT,

RETRY – количество транзактов, ожидающих по разным причинам освобождения устройства,

DELAY – количество транзактов, ожидающих освобождения устройства в режиме приоритетного входа по блоку PREEMPT.

- Список очередей с информацией:

QUEUE – название или номер очереди,

MAX – максимальное количество транзактов, содержащихся в очереди за весь период сбора статистики,

CONT. – текущее количество транзактов в момент окончания имитации,

ENTRY – общее количество входов в очередь,

ENTRY(0) – количество транзактов, вышедших из очереди без ожидания,

AVE.CONT – среднее значение транзактов в очереди,

AVE.TIME – среднее время ожидания в очереди,

AVE.(-O) – среднее время ожидания, исключая транзакты, которые не задерживались в очереди,

RETRY – количество транзактов, ожидающих вход в очередь по некоторым причинам.

- В таблицах печатается информация:

TABLE – имя или номер таблицы или Q-таблицы,

MEAN – среднее значение табулируемой величины,

STD.DEV – стандартное отклонение отклонение,

RANGE – левый и правый конец интервалов. Интервалы с нулевыми значениями не отображаются,

RETRY – количество транзактов, ожидающих по некоторым причинам в блоке TABULATE,

FREQUENCY – количество элементов, попавших в этот интервал,

CUM.% – накопительный процент попавших элементов.

Освоить программную среду GPSS и применить его для построения и исследования аналитических и имитационных моделей можно по книге Б.Я.Советов, С.А. Яковлев «Моделирование систем. Практикум», в котором подробно описаны методы и технологии работы в среде GPSS [27].

Задание на моделирование:

1. Построить модели систем массового обслуживания (рис.4.20 - 4.24) в среде объектно-ориентрованного программирования GPSS.
2. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

4.4. Система имитационного моделирования Arena

Охватить все технологии разработки имитационных моделей с помощью систем имитационного моделирования достаточно сложно. Рассмотрим в качестве примера технологию разработки модели СМО в системе имитационного моделирования Arena компании Rockwell Automation. Системами массового обслуживания (СМО) будем определять системы, которые обслуживают входящий поток заявок. На выходе имеем поток обслуженных заявок. В процессе обслуживания могут создаваться очереди конечной и бесконечной длины. Часть входящих заявок может получить отказ. Кроме того, различают одноканальные и многоканальные СМО.

Исходные данные для анализа: параметры распределения входящих и исходящих потоков, а также характеристики самой СМО, например, среднее время обслуживания. В результате расчетов определяют такие характери-

стики СМО, как среднее число заявок в системе, средняя продолжительность пребывания заявок в системе, среднее число заявок в очереди, средняя продолжительность пребывания заявок в очереди, средняя длина очереди и т.д.

Такие модели исследуют двумя методами, дающими близкие результаты. Аналитические методы теории СМО позволяют выполнять вероятностные расчеты и вычислять теоретические значения характеристик СМО.

Имитационное моделирование позволяет получить приблизительные оценки тех же параметров, причем с увеличением длительности моделирования они приближаются к теоретическим значениям. Имитационное моделирование можно использовать для исследования сложных систем, для которых непосредственное применение аналитической теории СМО основанной на уравнениях Колмогорова, затруднительно.

Система Arena компании Rockwell Automation является одним из лидеров на рынке программ имитационного моделирования. Этот программный продукт используют крупнейшие компании, включая General Motors, UPS, IBM, Nike, Xerox, Lufthansa, Ford Motor Company и другие. Основные области применения данного пакета: производство (моделирование конвейерного производства, определение узких мест…), логистика и складское хозяйство (оптимизация использования складских площадей), вооружение и безопасность, медицина (моделирование потока пациентов, распределение персонала…), и другие.

В пакете Arena используется процессор и язык имитационного моделирования SIMAN. Arena предоставляет пользователю удобный графический интерфейс с набором шаблонов моделирующих конструкций. Для создания модели в пакете Arena моделирующие конструкции сначала перетаскивают в окно модели, а затем соединяют, чтобы обозначить движение объектов в моделируемой системе. Затем моделирующие конструкции детализируются с помощью диалоговых окон или встроенных таблиц. В иерархии модели может быть неограниченное число уровней.

В базовом шаблоне Arena (Basic Process) представлены конструкции, позволяющие создавать дискретно-событийные модели: источники сущностей (create); уничтожители сущностей (dispose); действия (process), в которых можно задавать доступные ресурсы; конструкции для изменения свойств сущностей (assign); логические конструкции (decide).

Rockwell Arena выпускается только для операционной системы Windows. В Arena предусмотрен экспорт данных из Microsoft Excel и Microsoft Access.

Arena обеспечивает вывод на экран двухмерной и трехмерной (Arena 3DPlayer) анимации и позволяет выводить на экран динамическую графику (гистограммы и графики временной зависимости). На рис. 4.25 приведен внешний вид (интерфейс) системы Arena, в котором представлена логическая схема и анимационная картина работы газовой заправочной станции (пример из пакета Arena).

Число потоков случайных чисел в пакете Arena не ограничено. Более того, пользователь имеет доступ к 12 стандартным теоретическим распределениям вероятностей, а также к эмпирическим распределениям.

Данный пакет позволяет выполнять функционально-стоимостной анализ при использовании ABC-метода, благодаря чему можно учитывать дополнительные и обычные затраты, а также создавать временные отчеты. Результаты моделирования сохраняются в базе данных и отображаются на экране после прогона модели в виде отчета.

Пример 1. Рассмотрим систему документов, поступающих в некоторую компанию (фирму) [17].

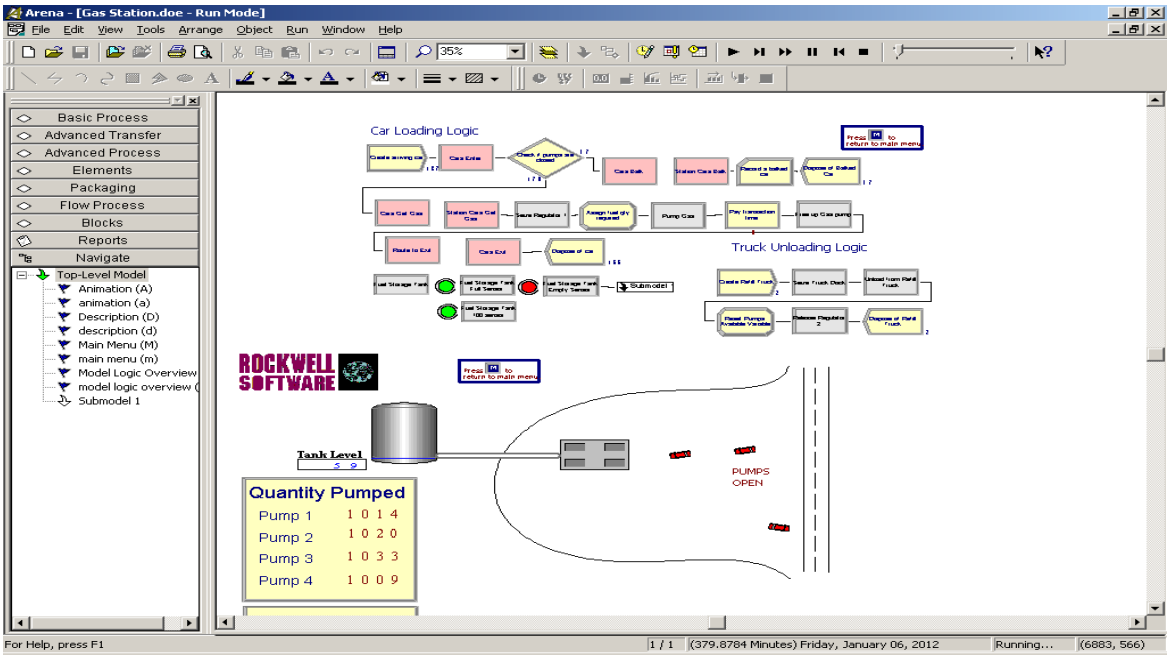


Рис.4.25

Процесс моделирования в системе **Arena** организован следующим образом. Сначала пользователь шаг за шагом строит в визуальном редакторе программы **Arena** модель. Затем система генерирует по ней соответствующий код на **SIMAN**, после чего автоматически запускается **Cinema animation**.

Имитационная модель в программе **Arena** состоит из блоков моделирования (модули) и операций (сущности). Сущности двигаются между модулями по мере их обслуживания.

Шаг. 1. С помощью графических модулей данной модели необходимо построить динамическую модель обработки заявок фирмы (с отказом), как это показано на рис. 4.26.

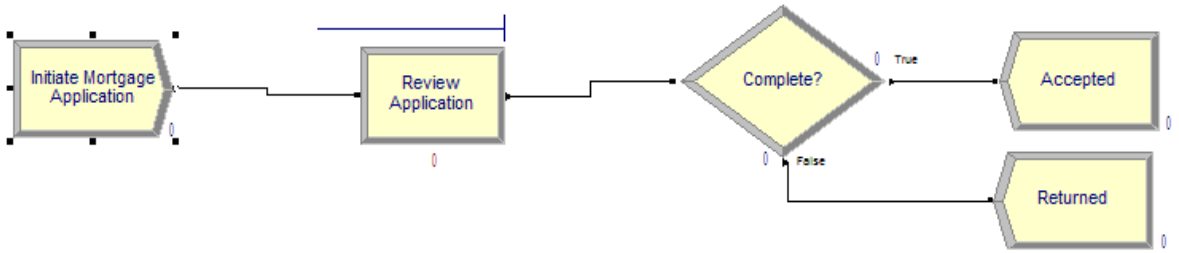


Рис. 4.26. Динамическая модель обработки заявок фирмы

Модуль **Initiate Mortgage Application** представляет собой входящий поток заявок, в нем задается интенсивность поступления заявлений, время, через которое прибудет первая заявка в модель от начала симуляции, количество заявок за одно прибытие, максимальное число заявок, которое может создать этот модуль (рис. 4.27):

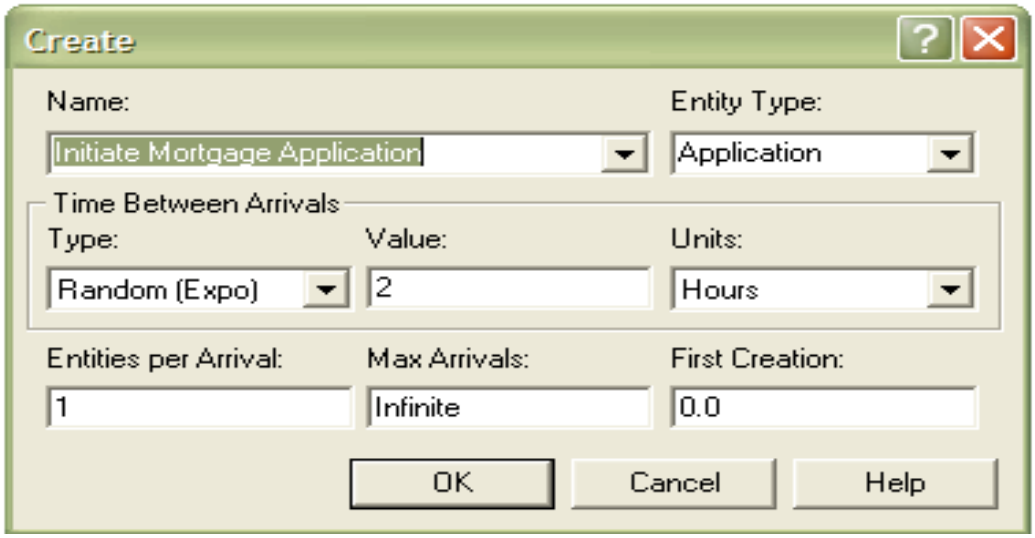


Рис. 4.27. Настройки модуля **Initiate Mortgage Application**

Модуль **Review Application** – здесь задаются ресурсы на обработку заявки, время на обработку одной заявки, а также мощность процесса, т.е. то, сколько одновременно могут обрабатываться заявок в системе, так как при существующем процессе обработки заявлений этим занимается 1 клерк (рис. 4.28).

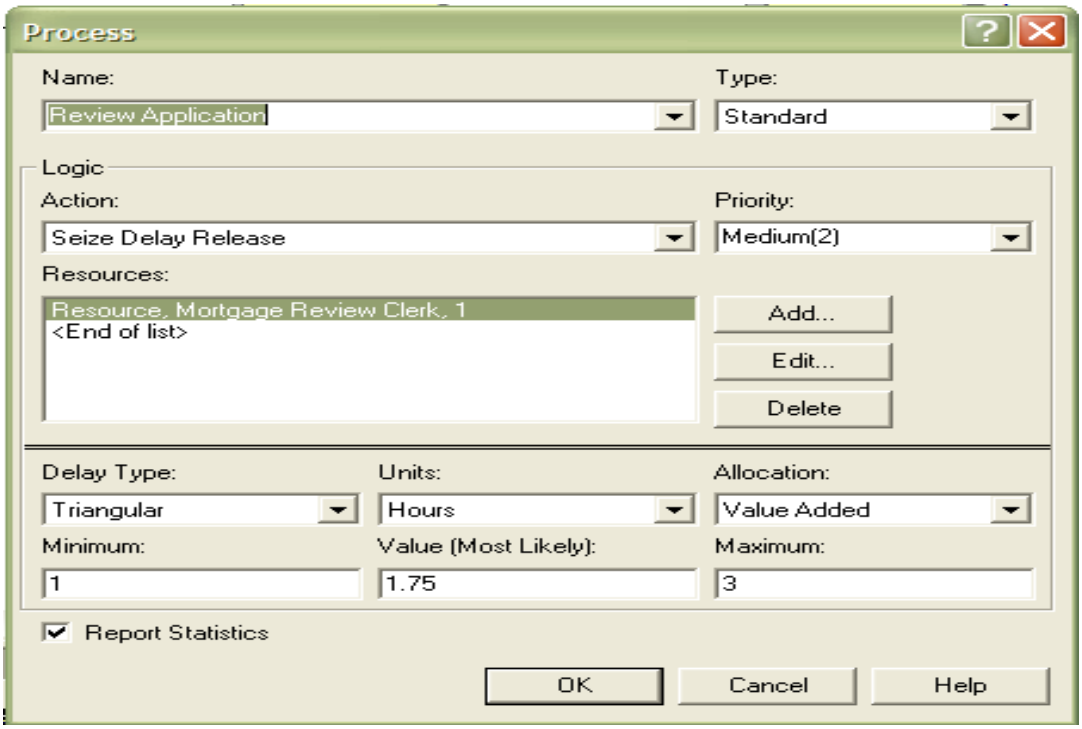


Рис. 4.28. Настройки модуля **Review Application**

Также вызвав панель ресурсов, необходимо указать затраты клерка в 12 $ в час (рис. 4.29).

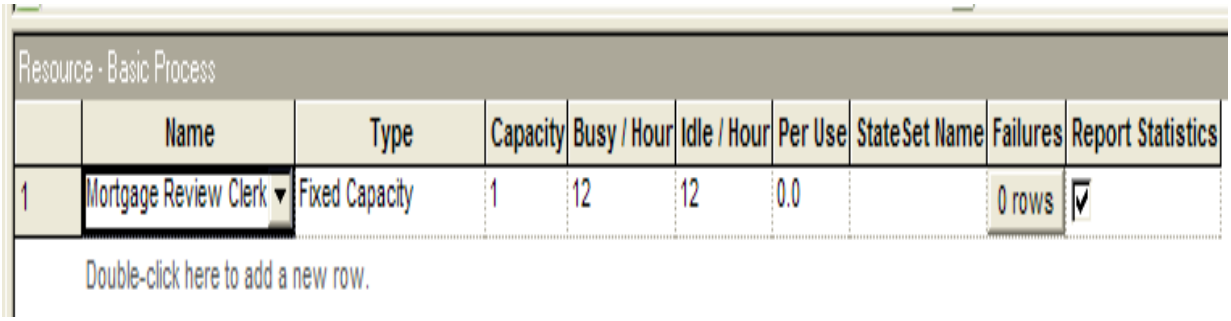


Рис. 4.29. Панель **Resorse**

Модуль **Complete?** (рис. 4.30) – проверяет обработанное заявление на ошибки, если значение модуля –**true**, то заявка переходит в модуль **Accepted** (рис. 4.31) , если значение, модуля **Complete?**, - **false** то обработанная заявка переходит в модуль **Returned** (рис. 4.32). В Модуле **Complete?** задается вероятность значения **true**, т.е. если процент числа ошибок 12% то вероятность того, что модуль **Complete?**, примет значение **true** – 88% (рис.4.33). Модули **Accepted** и **Returned** показывают общее количество обработанных заявок в системе и количество отказных заявок (рисунки 4.21 и 4.22).

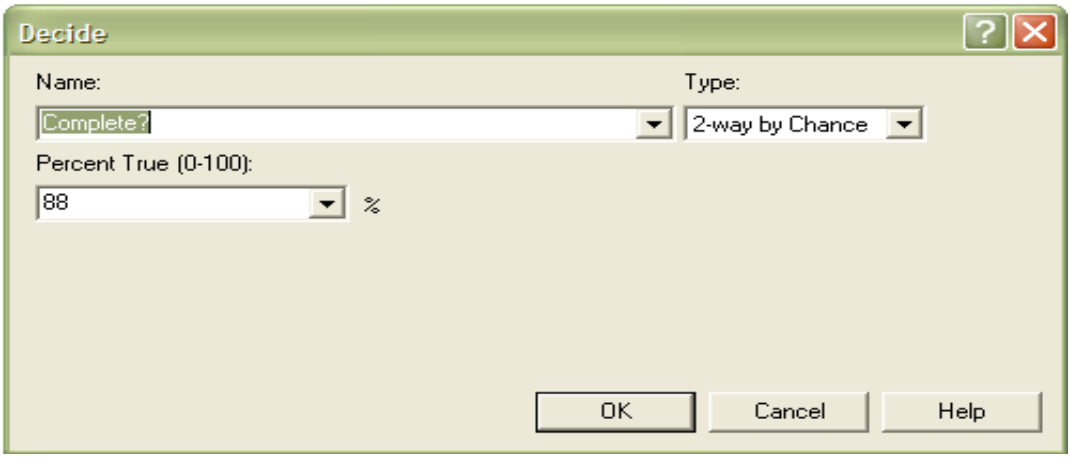


Рис. 4.30. Настройки модуля **Complete?**

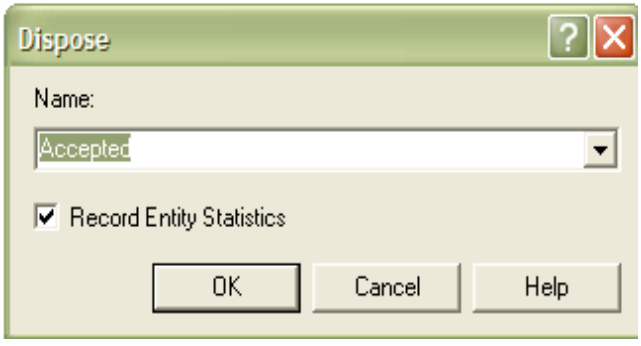


Рис.4.31. Настройки модуля **Accepted**

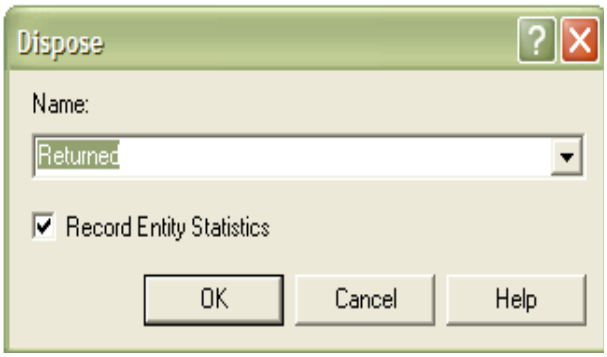


Рис. 4.32. Настройки модуля **Returned**

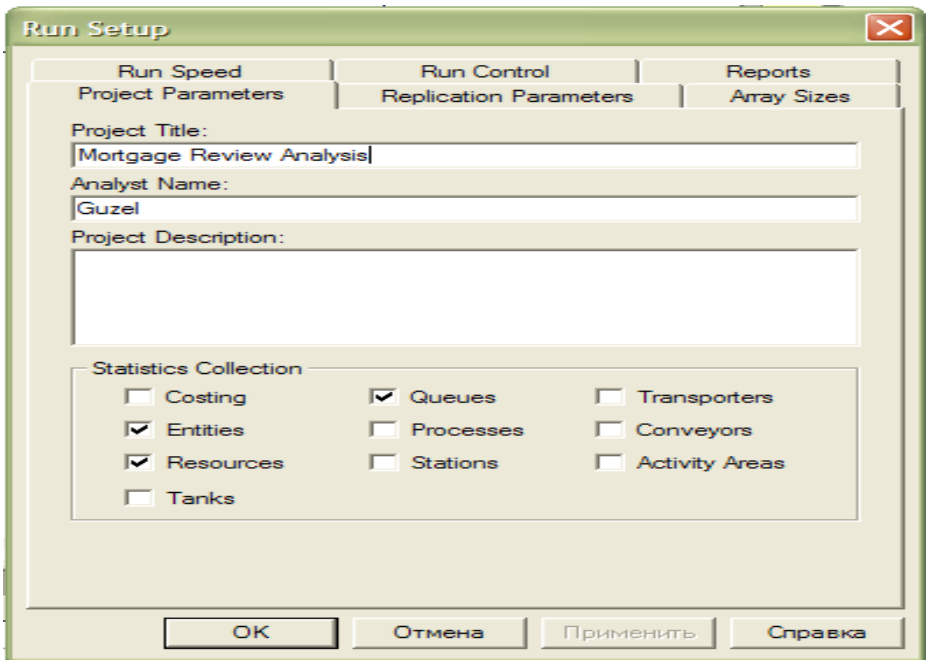


Рис. 4.33. Диалоговое окно **Run-Setup**

Задания длительности моделирования осуществляется в меню **Run-Setup** (рис. 4.33). В поле **Replication Length** установите длительность 20 дней, а в поле **Time Units** единицу измерения времени **Days**.

Шаг 3. После проделанных настроек системы необходимо запустить симуляцию, по нажатию кнопки **F5** на клавиатуре или кнопку на панели инструментов  .

Результаты. Система после проигрывания будет выглядеть следующим образом (рис. 4.34).

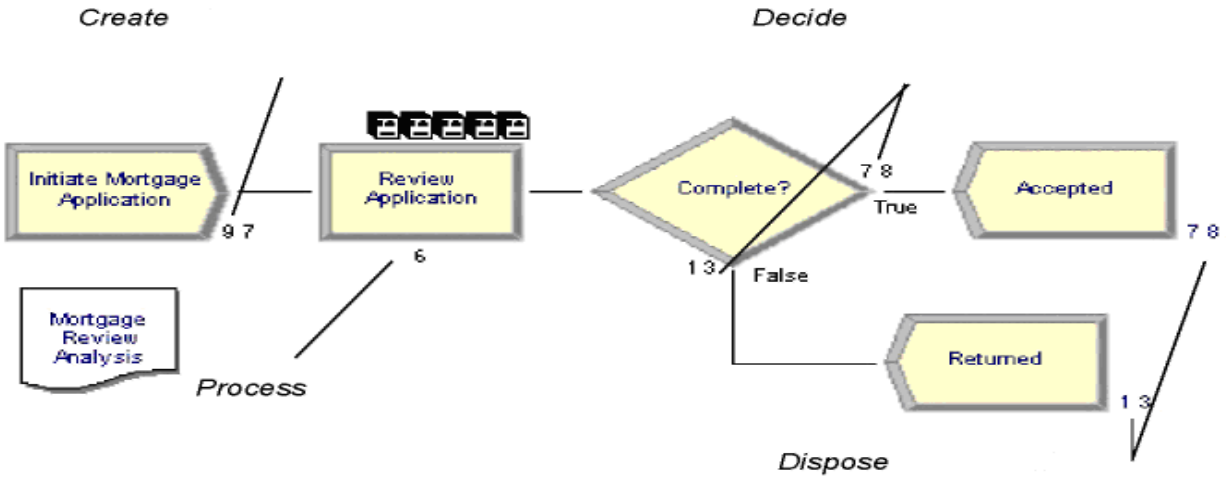


Рис. 4.34 Система после завершения симуляции

Анализ модели. На выходе каждого модуля указывается число заявок. На выходе модуля **Initiate Mortgage Application** стоит 97 штук, это значит, что за рассматриваемый период поступило 97 заявлений.

Под модулем **Review Application** указанно количество не обработанных заявок по окончанию симуляции системы, 6 штук. Сверху модуля отображается очередь из заявок, которые ожидают освобождение клерка.

Модуль **Complete?** показывает количество отказных заявок 13 штук, а также количество обработанных заявок 78 штук.

Модуль **Accepted** показывает общее количество обработанных заявок в системе, а модуль **Returned** – количество отказных заявок.

Просмотреть все отчеты по результатам проигрывания данной модели можно в панеле отчетов **Reports** (рис. 4.35).

Задание на моделирование:

1. Построить дискретно-событийную модель обработки заявок фирмы (рис.4.26 - 4.35) в среде имитационного моделирования ARENA.
2. Провести компьютерные эксперименты в зависимости от характеристик СМО и начальных условий.
3. Провести обработку и анализ результатов компьютерных экспериментов.

Рис. 4.35. Панель отчетов **Reports**

4.5. Система имитационного моделирования AnyLogic

Пакет AnyLogic – профессиональный инструмент нового поколения, который предназначен для разработки и исследования имитационных моделей сложных систем. Программный продукт разработан компанией «Экс Джей Текнолоджис» (XJ Technologies), г. Санкт-Петербург. AnyLogic был разработан на основе новых идей в области информационных технологий, теории параллельных взаимодействующих процессов и теории гибридных систем. Благодаря этим идеям чрезвычайно упрощается построение сложных имитационных моделей, имеется возможность использования одного инструмента при изучении различных стилей моделирования. Продукт поддерживает основные методы имитационного моделирования: системная динамика; дискретно-событийное моделирование; агентное моделирование [2, 3, 16, 19, 20, 31].

Многие крупные компании, как отечественные Билайн, Газпром, Сбербанк России, Русский алюминий, Северсталь и т.д., так и зарубежные General Motors, Mitsubishi, McDonalds, IBM и др., являются клиентами компании и используют AnyLogic для своих потребностей и исследований (рис.4.36).

Рис. 4.36. Клиенты компании AnyLogic

Сайт компании AnyLogic (www.xjtek.ru) позволяет продемонстрировать возможности данного продукта. На сайте представлены различные демо-версии имитационных моделей по различным отраслям (рис. 4.37).

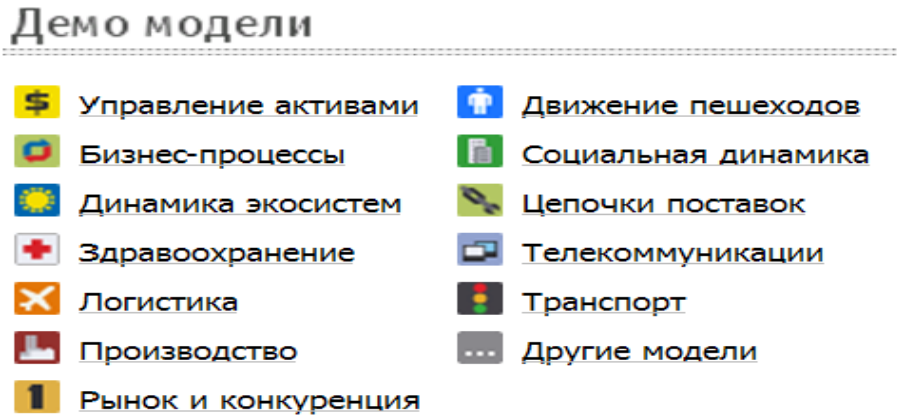
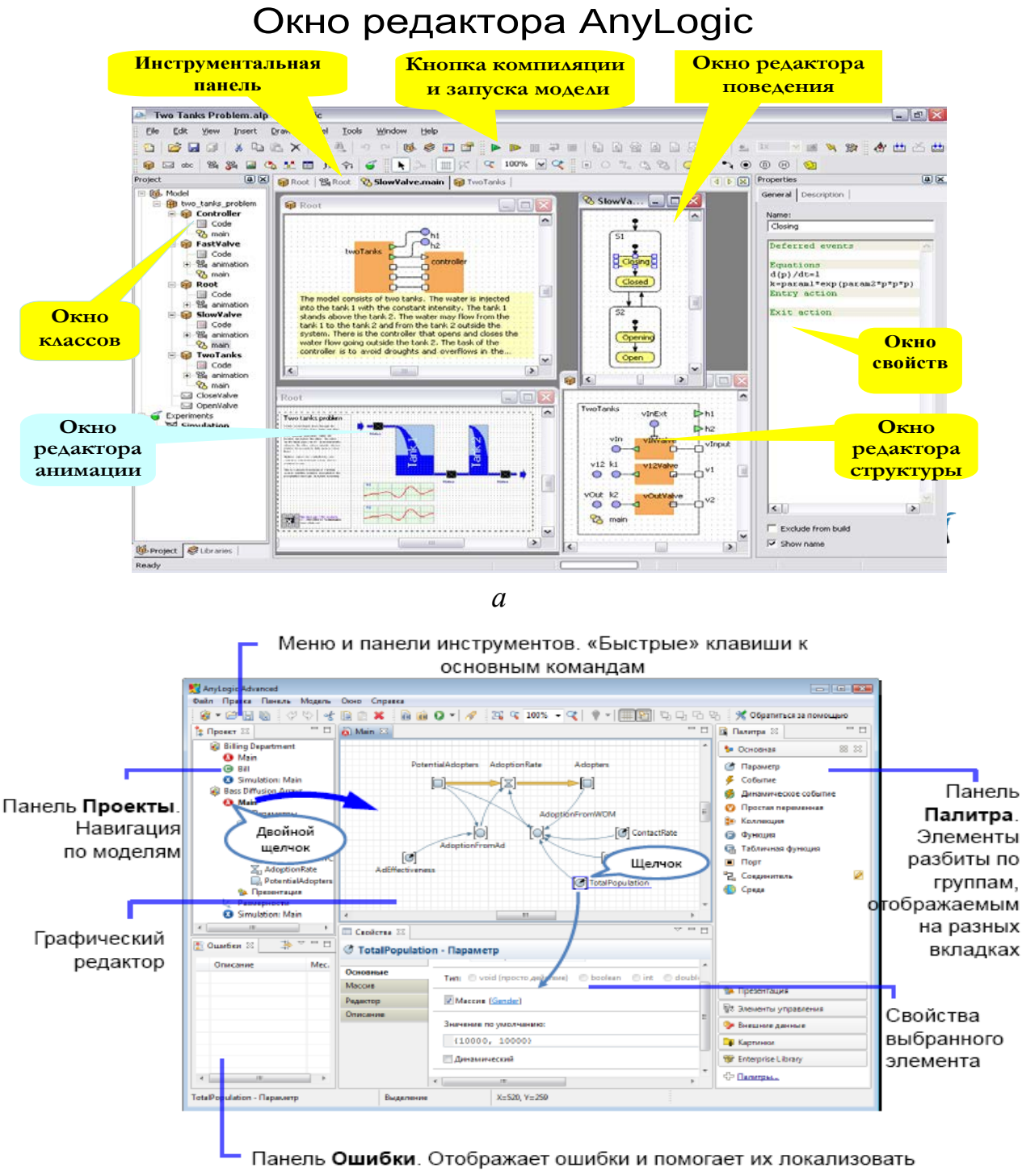


Рис.4.37. Отрасли и сферы использования программного продукта

Для каждого из отраслей на сайте приведены различные примеры демо-моделей (рис.4.38), которые в реальном времени можно прогнать. Описания разработок некоторых из них изложены в работах и книгах по имитационному моделированию [3, 14, 16, 19-20].

Рис.4.38

Графическая среда AnyLogic построена по тому же принципу, что и в Rockwell Arena. Моделирующие конструкции располагаются в палитрах (аналог шаблонов в Arena). Для создания модели, как и в Arena, моделирующие конструкции перетаскивают в область модели и соединяют. Детализировать моделирующие конструкции можно, выделив их и изменив параметры, используя панель свойств.



а

б

Рис. 4.39. Окно пользовательского интерфейса среды AnyLogic: *а* – 5 версии; *б* – 6 версии

AnyLogic поддерживает подход агентного моделирования, в качестве агентов могут быть: люди: потребители, жители, работники, пациенты, доктора, клиенты, солдаты и др.; транспорт, оборудование: автомобили, краны, самолёты, вагоны, станки и др.; нематериальные вещи: проекты, продукты,

инновации, идеи, инвестиции и др.; организации: компании, политические партии, страны и др.

Эти объекты в системе AnyLogic могут создаваться и уничтожаться динамически, перемещаться, общаться друг с другом, иметь поведение, знания, цели, стратегию — то есть обладают всеми свойствами агентов.

Агентный подход используют для моделирования социальных систем, в частности, моделируют рынки (агент - потенциальный покупатель), конкуренцию и цепочки поставок (агент - компания), население (агент - семья, житель города или избиратель) и много другое. Только агентный подход позволяет получить представление об общем поведении системы, исходя из предположений о поведении её элементов при отсутствии знания о глобальных законах — то есть в наиболее общем случае.

AnyLogic основан на Java и базируется на платформе Eclipse - современном стандарте для бизнес-приложений. Благодаря Eclipse AnyLogic работает на всех распространенных операционных системах (Windows, Mac, Linux и т.д.).

В редакторе AnyLogic возможно разработать анимацию и интерактивный графический интерфейс модели. Анимация может быть иерархической и поддерживать несколько перспектив. Например, есть возможность определить глобальный взгляд на процесс производства, а также детальные анимации конкретных операций и переключаться между ними. Имеется возможность использовать различные виды технологических решений для реализации анимации (рис.4.40).

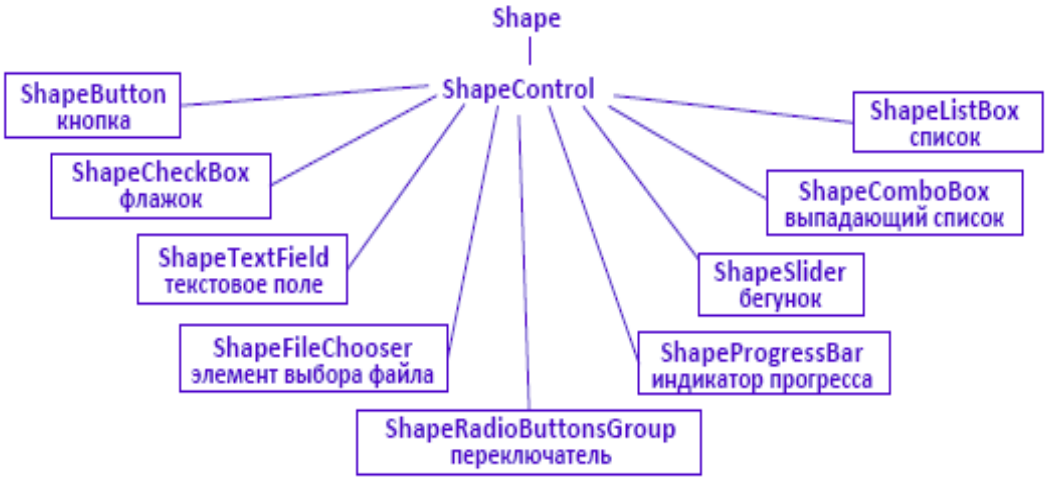


Рис.4.40

В AnyLogic пользователю доступно 35 стандартных теоретических распределений, также можно определить свои. AnyLogic позволяет строить как стохастические, так и детерминированные модели и проводить анализ ре-

зультатов моделирования. С моделью могут быть проведены различные эксперименты:

- моделирование (simulation)
- оптимизация (optimization)
- эксперименты Монте-Карло
- анализ чувствительность
- эксперименты по сценарию пользователя

В AnyLogic встроен оптимизатор OptQuest. Комбинируя эвристики, нейронные сети и математическую оптимизацию, OptQuest позволяет находить значения дискретных и непрерывных параметров модели, соответствующие максимуму или минимуму целевой функции, в условиях неопределённости и при наличии ограничений. Для создания отчетов в AnyLogic отведена специальная палитра «Статистика», в которой содержатся конструкции для сбора данные по ходу работы модели. В этой палитре также находятся различные диаграммы, графики и гистограммы.

AnyLogic имеет исключительно развитый базовый язык дискретного и смешанного дискретно-непрерывного моделирования, на основе которого разработаны стандартные библиотеки:

- Enterprise Library – конструкции для построения дискретно-событийных моделей
- Dynamic Systems Library (системная динамика)
- Material Flow Library (потоки материалов)

Модель и анимация быстро строятся в стиле drag-and-drop и очень гибко параметризуются. Реализация стандартных объектов открыта для пользователя, их функциональность может быть как угодно расширена, вплоть до создания собственных библиотек. Используя иерархию и регулярные структуры объектов, можно создавать масштабируемые модели.

С помощью библиотеки Enterprise Library пакета AnyLogic можно быстро создавать сложные дискретно-событийные модели, такие как:

- модели производственных процессов;
- модели систем обслуживания (банки, аэропорты и т.д.);
- модели бизнес-процессов с оценкой затрат операций;
- модели логистики и цепочек доставки.

Библиотека объектов Enterprise Library позволяет создавать гибкие модели с наглядной визуализацией моделируемого процесса и возможностью сбора необходимой статистики.

Система AnyLogic может быть использована для моделирования непрерывных динамических процессов и систем.

Пример 1. Рассмотрим использование пакета AnyLogic для разработки имитационной и анимационной модели маятника Фуко. В данном примере воспользуемся возможностью программного продукта анализировать системы динамического типа.

Над центром вращающейся горизонтальной платформы подвешен маятник на длинном подвесе. Маятник Фуко является математическим маятником, такой маятник, отклонённый от равновесного положения, совершает колебания в плоскости, неподвижной в инерциальной системе отсчёта. Необходимо найти траекторию движения конца колеблющегося маятника на платформе.

Уравнения движения:

$$\frac{dv_x}{dt} = 2v_y w + w^2 x - g\frac{x}{L}$$

$$\frac{dv_y}{dt} = -2v_x w + w^2 y - g\frac{y}{L}$$

W – относительная частота, L – длина подвеса. Рассмотрим построение математической модели маятника Фуко средствами системы AnyLogic.

Шаг 1. Создадим новый проект.

Щелкните мышью по кнопке панели инструментов “***Создать***”. Появится диалоговое окно “Новый Проект”. Щелкните мышью по кнопке “Выбрать…” и выберите директорию, в которой будем хранить файлы проекта. Укажем имя нового проекта. Например, Foucault's pendulum, в поле редактирования “Имя проекта”. Щелкните мышью по кнопке “OK”. Новый проект создан.

Шаг 3. Зададим данные внутри модели.

Нужно решить, что использовать – параметры или переменные.

Параметр обычно используется для задания статических характеристик модели и обычно хранит одно и то же значение в течение всего "прогона" модели; это значение изменяется пользователем только в какие-то определенные моменты времени при желании изменить характеристики модели. Параметры задаются на странице “Общие” панели “Свойства”.

Зададим величину W (относительная частота). Для этого присвоим параметру имя w, значение по умолчанию, например, 0.04. (рис.4.41).

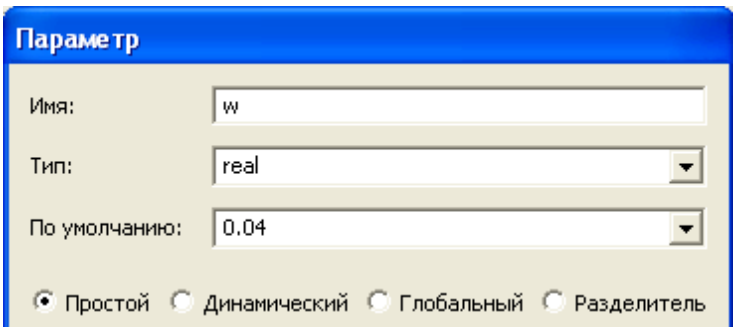


Рис.4.41

Зададим величину L (длина подвеса). Для этого присвоим параметру имя L, значение по умолчанию - 50.

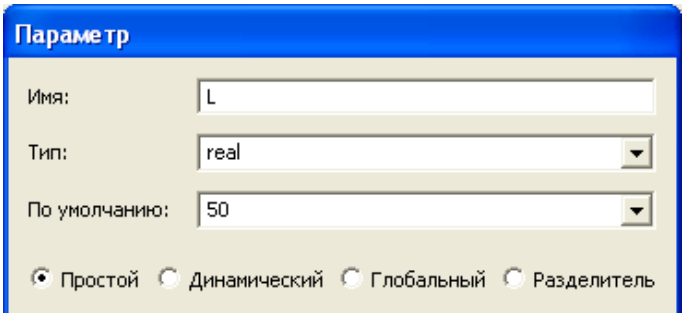


Рис.4.42

Зададим величину g (гравитационная постоянная). Для этого присвоим параметру имя g, значение по умолчанию – 9.8.

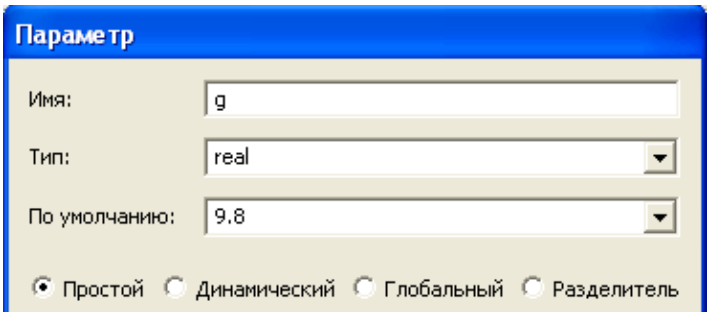


Рис.4.43

Все заданные параметры можно увидеть в панели “Свойства”.

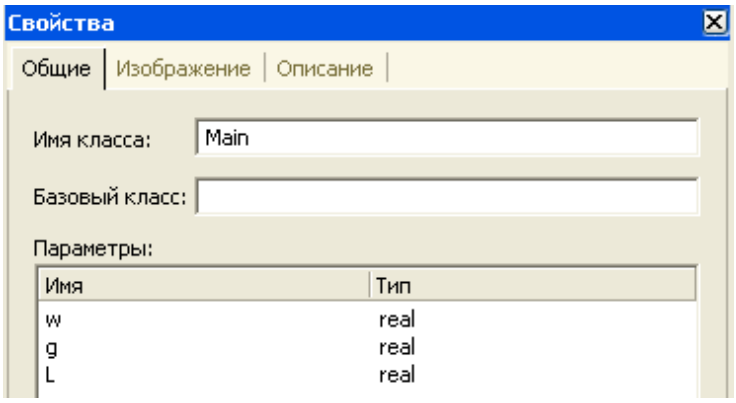


Рис.4.44

Переменные обычно используются для моделирования изменяющихся характеристик объекта или для хранения результатов работы модели.

Зададим 4 переменных – x, y, vx и vy.

Чтобы задать переменным дифференциальные уравнения, необходимо выбрать на странице “Общие” панели “Свойства” в поле “вид” – “интеграл или накопитель”.

В AnyLogic уравнения записываются в строку.

Для переменной x: d(x)/dt = vx, начальное значение – 1.

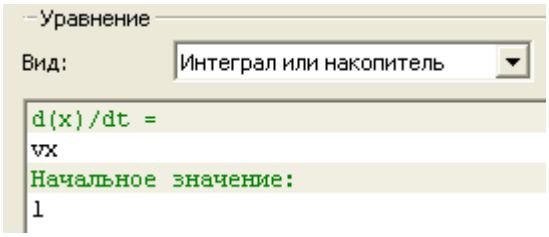


Рис.4.45

Для переменной y: d(y)/dt = vy, начальное значение – 1.

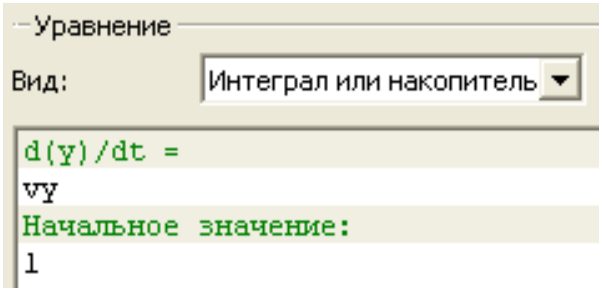


Рис.4.46

Для переменной vx: d(vx)/dt = 2*vy*w+w^2*x-g*x/L, начальное значение – 0 (по умолчанию в AnyLogic, если оставить начальное значение пустым, это также будет 0).

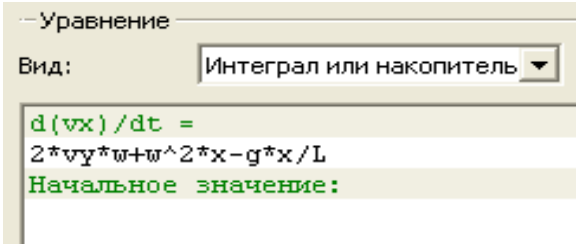


Рис.4.47

Для переменной vy: d(vy)/dt = -2*vx*w+w^2*y-g*y/L, начальное значение – 0.

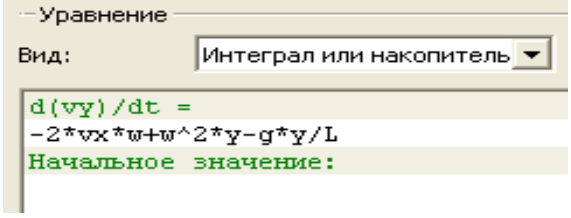
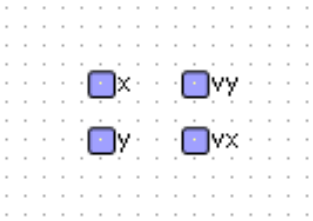


Рис.4.45

Все параметры и переменные видны на презентации модели, и можно изменять их значения во время работы модели либо программно из кода модели, либо с помощью элементов управления.



Параметры:

Имя	Тип
w	real
g	real
L	real

Рис.4.48

Также добавим в модель график – xYChart из библиотеки Business Graphic Library, который понадобится для анимации движения конца маятника Фуко.

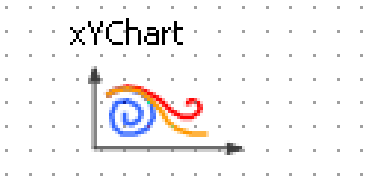


Рис.4.49

Шаг. 3. Создание анимации.

Для создания анимации прежде всего в меню “Вставка” нужно выбрать пункт “Новая анимация”. Укажем размеры анимационной области, например, 1110x600.

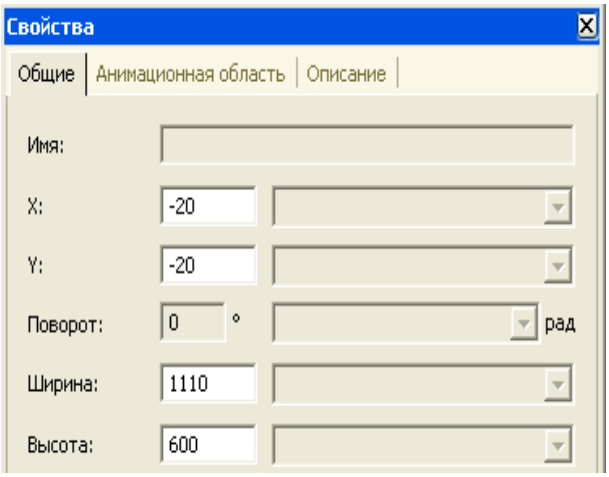


Рис.4.50

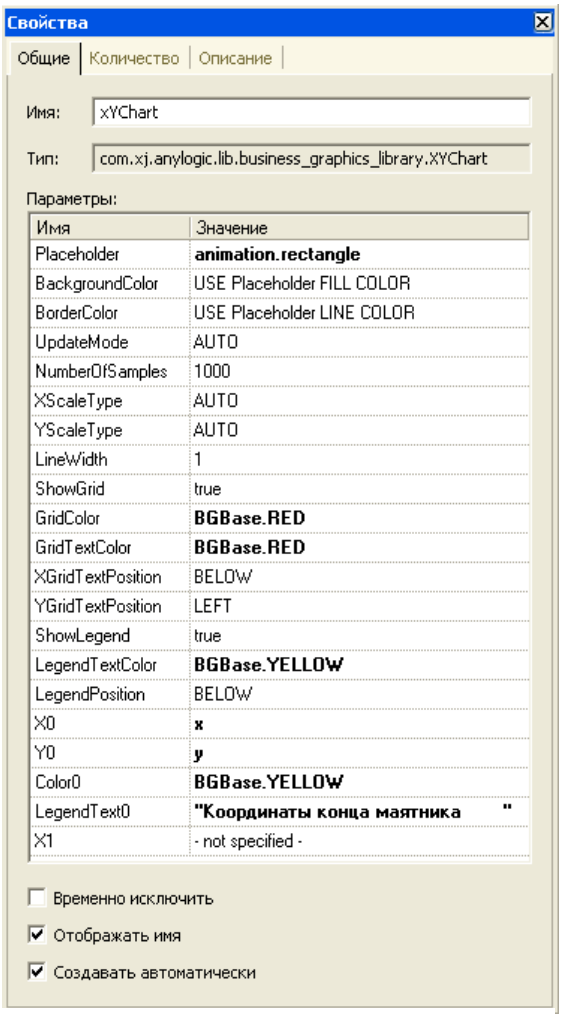


Рис.4.51

Это можно сделать в окне “Свойства” в полях “Ширина” и “Высота”.

Построим прямоугольник (rectangle) нужного размера, в котором будет показываться траектория движения конца маятника. В свойствах графика xYChart укажем следующее: Placeholder – animation.rectangle (это позволит

увидеть в прямоугольнике строящуюся траекторию), GridColor и GridTextColor – Red (оси x и y, а также значения координат на осях будут подписаны красным цветом), LegendTextColor – Yellow (траектория будет показана желтым), XO и YO – x и y соответственно, Color0 – Yellow, LegendText0 – “Координаты конца маятника ” (ниже графика будут показаны текущие координаты).

Модель готова к запуску, но нужно сделать удобное изменение параметров во время ее выполнения.

Шаг. 4. Изменение параметров во время выполнения модели.

В анимации создадим 2 бегунка (sliders). Они будут отвечать за изменение параметров w и L во время выполнения модели.

Бегунок (slider) – элемент управления, позволяющий пользователю графически выбирать число из заданного диапазона значений путем перетаскивания рукоятки (рис.4.52).

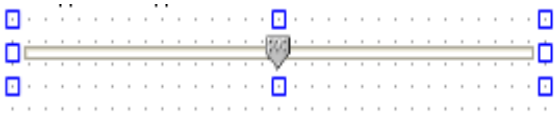


Рис.4.52

Бегунки обычно используются для изменения значений численных переменных и параметров во время выполнения модели. Чтобы добавить бегунок, необходимо перетащить элемент “Бегунок” в нужное место графического редактора.

На странице “Бегунок” панели “Свойства” в поле “Переменная” введем численную переменную или параметр. В результате – присваиваем этой переменной текущее значение этого бегунка.

Создадим бегунок для параметра L. В полях “минимум” и “максимум” введем минимальное и максимальное значение, которое может принять переменная при изменении бегунка. Например, 1-100.

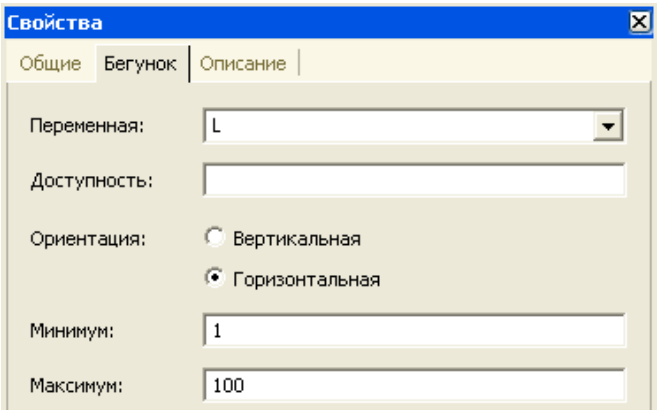


Рис.4.53

Создадим бегунок для параметра w. В полях “минимум” и “максимум” введем минимальное и максимальное значение, которое может принять переменная при изменении бегунка. Например, 0-1 (включая сотые).

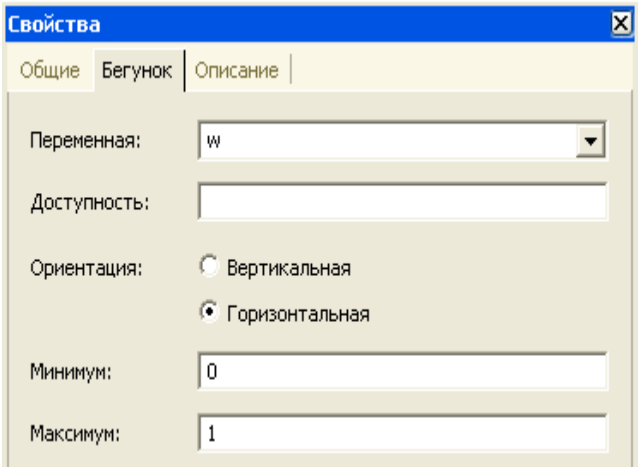


Рис.4.54

Теперь, чтобы не путаться и знать, какое значение задано бегунком, создадим несколько текстовых полей.

Текстовое поле (иди поле ввода) является простейшим текстовым элементом управления, позволяющим пользователю вводить небольшие объемы текста. Чтобы добавить текстовое поле, необходимо перетащить элемент “Поле ввода” в то место графического редактора, где вы хотите его нарисовать.

На странице “Текст” панели “Свойства” в верхнем поле ввода “Текст” можно ввести любой текст, в нашем случае – подпишем бегунки (L и w).

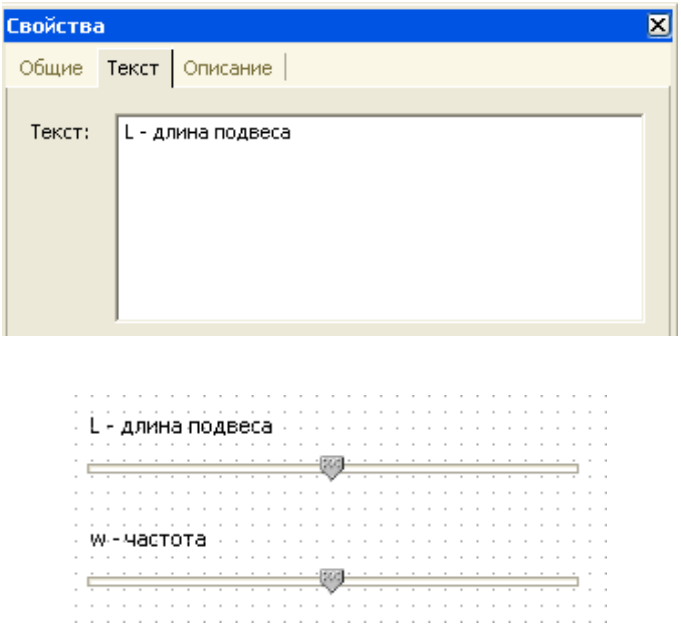


Рис.4.55

В нижнем поле (“Динамический текст”) можно выбрать параметр или переменную, значение которой и будет показывать поле ввода. При измене-

нии этого параметра (например, бегунком), значение текстового поля тоже изменится. Для одного бегунка выберем параметр L, для другого – w.

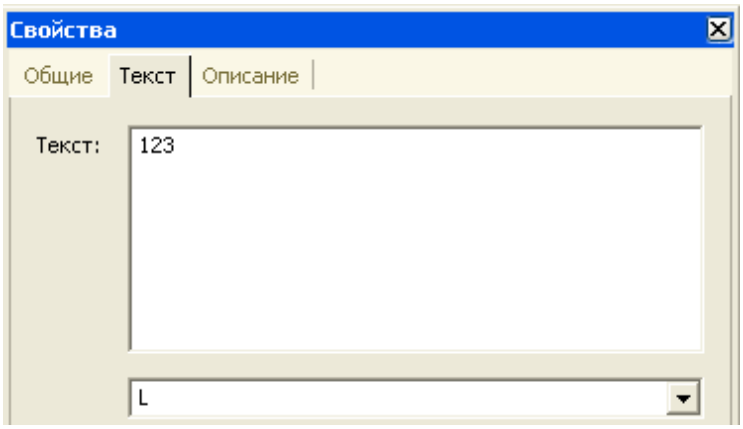


Рис.4.56

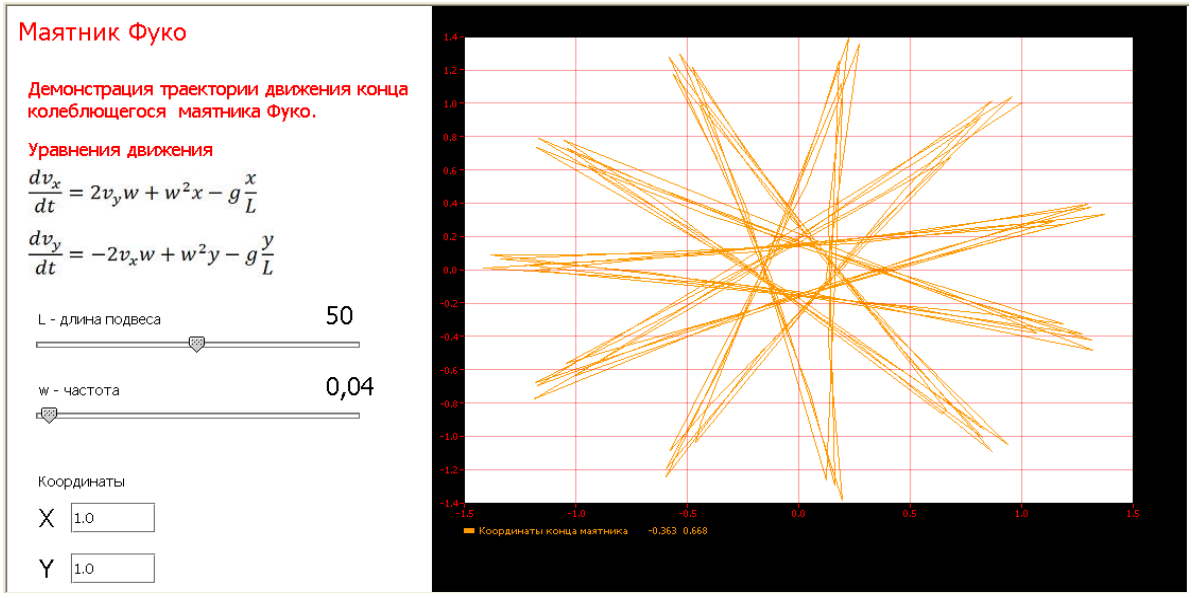


Рис.4.57

Можно также с помощью текстовых полей подписать модель (“Маятник Фуко”), указать используемые уравнения и т.д. В данной модели можно динамически изменять такие параметры, как длину подвеса и частоту, при этом график сразу же изменится. Также можно указать нужные координаты X и Y. Текущие координаты конца маятника указываются под графиком траектории движения.

Задание на моделирование:

4. Построить аналитическую динамическую модель маятника Фуко (рис.4.44 - 4.57) в среде имитационного моделирования Anylogic.
5. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.

6. Провести обработку и анализ результатов компьютерного моделирования.

Пример 2. Модель дорожного перекрестка.

Рассмотрим определенную транспортную развязку, например, перекресток. Через перекресток движутся потоки автомобилей. Для упорядочивания движения служит автоматический светофор или регулировщик.

Создадим модель *Дорога* на основе шаблона Дискретно-событийное моделирование. AnyLogic создаст потоковую диаграмму, состоящая из 4-х элементов: source, queue, delay и sink.

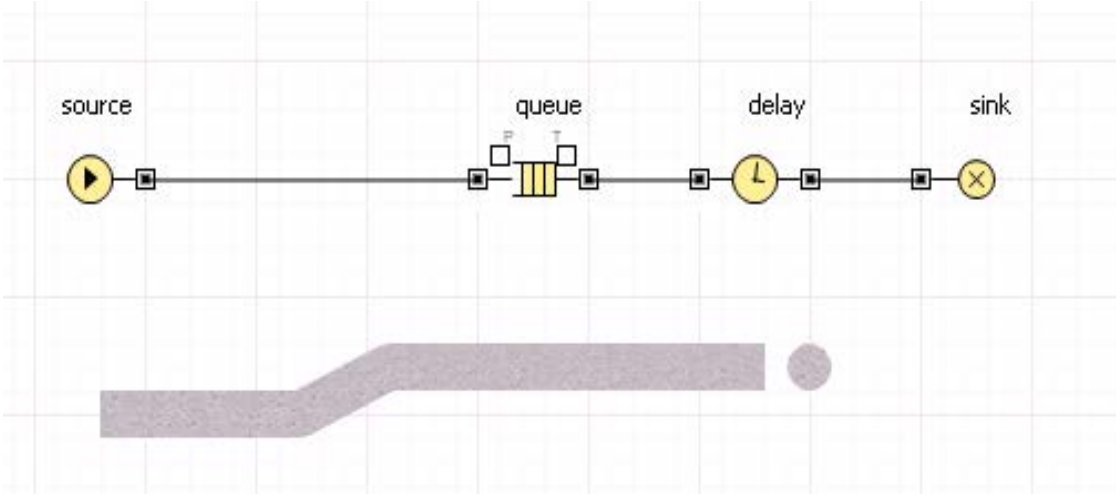


Рис. 4.58. Потоковая диаграмма

Source – создает заявки, в настраиваемые моменты времени. Объект Source не разрешает заявкам храниться в буфере выходного порта, если они не могут покинуть объект. Поэтому в случае, когда за объектом source расположен объект, который по той или иной причине не может принять новую заявку, нужно между ними поставить специальный объект буферизации, например, queue.

Queue – хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме.

Delay – задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным или зависеть от каких-то других условий. Это время может вычисляться как длина фигуры анимации этого объекта, поделенной на "скорость" заявки.

Sink – уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок.

Автомобили подъезжают к перекрестку в произвольные моменты времени, поэтому объект Source также должен создавать заявки в случайные

моменты времени. Для этого в строке свойств объекта Source заявки прибывают согласно времени между прибытиями. В появившемся поле ввода времени между прибытиями запишите exponential (0.1). Функция exponential генерирует реализацию случайной величины с экспоненциальным законом распределения.

В свойствах объекта queue следует удалить связь с анимацией: в поле Фигура анимации панели свойств, удалите ссылку на polyline. Заявки, находящиеся в очереди, анимироваться не будут. Этот объект будет выполнять буферную функцию для предотвращения ошибки в момент, когда source создал заявку, а delay еще не готов ее принять.

В свойствах объекта delay следует сделать следующие изменения:

• задержка задается – укажите: Как длина пути/скорость,

• время задержки – исправьте на triangular(5., 10., 15.),

• вместимость – исправьте на 10,

• фигура анимации – исправьте на polyline,

• тип анимации – исправьте на Путь.

Двигаясь в 3 потока, автомобили должны останавливаться перед стоплинией на красный сигнал светофора. Для моделирования движения с остановкой подходит объект Conveyor, который перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними. Когда заявка достигает конца конвейера, но не может его покинуть, то она там и останется. Если конвейер – накапливающий, то он продолжит двигать заявки, которые имеют достаточно свободного места перед собой.

Перенесите из палитры Enterprise Library на диаграмму класса Main объект conveyor и поместите его между queue и delay.

В свойствах объектов conveyor следует сделать следующие изменения:

• длина задается – укажите: Согласно пути,

• расстояние между заявками – укажите: 55,

• фигура анимации – укажите на соответствующую ломаную.

Для моделирования остановки автомобилей на красный свет необходимо в диаграмму установить элемент, останавливающий движение заявок – hold. Этот объект блокирует/разблокирует поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт и будут ждать, пока объект не будет разблокирован. Поставьте объект hold после conveyor.

Для определенного порядка передвижения заявок после светофора нужно поставить еще один conveyor после объекта hold. Тогда наша диаграмма примет следующий вид:

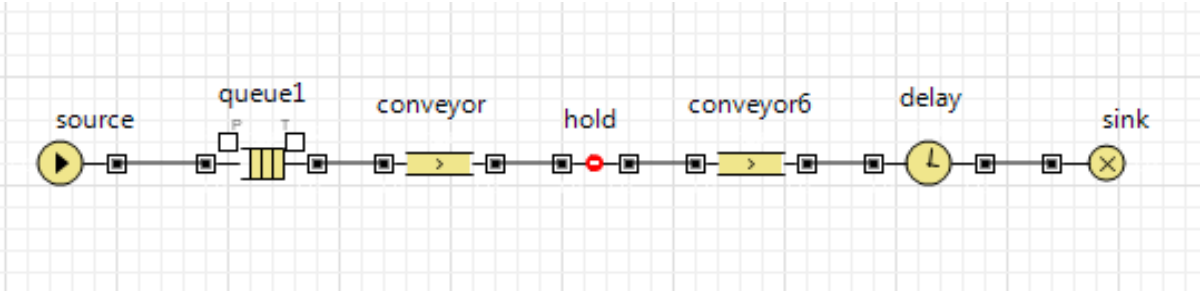


Рис. 4.59. Потоковая диаграмма.

Для анимации движения заявок (автомобилей) по конвейеру (зоне остановки перед светофором) нужно нарисовать ломаные линии и связать их с соответствующим объектом conveyor.

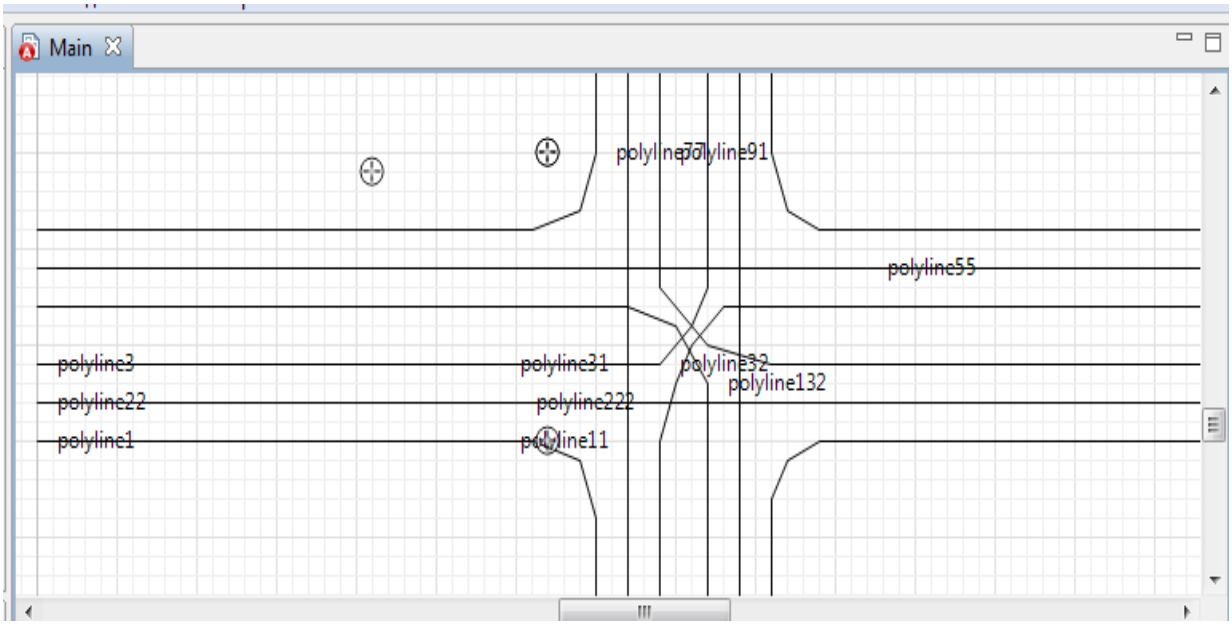


Рис. 4.60. Перекресток

Каждая линия движения у нас будет состоять из двух ломаных линий. Точка соединения ломанных будет находиться в месте остановки перед светофором. Для каждой линии движения у нас будет своя потоковая диаграмма. Тогда в первой диаграмме в conveyor перед объектом hold нужно указать фигуру анимации polyline1, а для conveyor после объекта hold polyline11.

Решим задачу построения модели регулируемого дорожного движения со светофором, разрешающим или запрещающим движение транспорта. Светофор, регулирующий движение автомобилей на пешеходном переходе, может находиться в следующих состояниях: движение транспорта разрешено (зеленый), приготовиться к запрещающему сигналу (мигающий зеленый), приготовиться к остановке (желтый), движение запрещено (красный) и приготовиться к движению (красный и желтый).

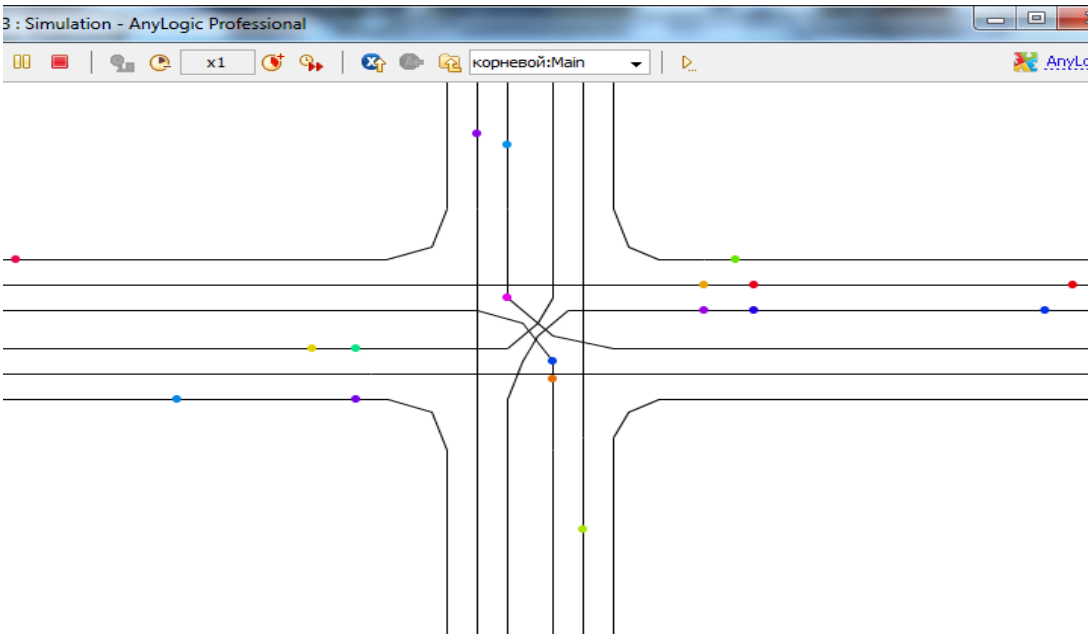


Рис. 4.61. Запуск модели

Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

Для построение модели на диаграмму класса активного объекта Model поместим Начало диаграммы состояний из панели Диаграмма состояний и назовем ее Для_автомобилей. Для того чтобы построить стейтчарт, следует использовать элементы из палитры Диаграмма состояний (рис.4.62).

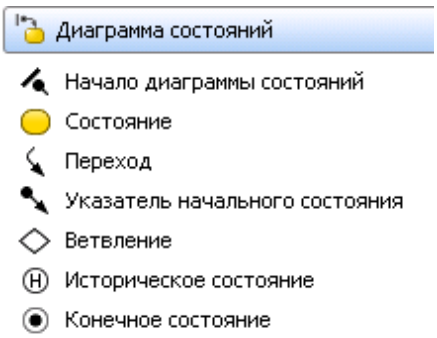


Рис. 4.62. Диаграмма состояний

В соответствии с алгоритмом работы светофора помимо начального состояния в модель нужно ввести дополнительные состояния (рис. 4.62). Начальное состояние назовите движение – движение автомобилям разрешено (зеленый свет), затем светофор переходит в состояние внимание – внимание (мигающий зеленый), медленно – приготовиться к остановке (желтый свет), остановка транспорта stop – запрет движения (красный свет) и приготовиться – приготовиться к движению (красный и желтый свет горят одновременно).

Состояние внимание представим гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (состояние A), в другом – нет (состояние B). Постройте эти состояния и соедините их переходами, как показано на рис. 4.63.

Зададим условия срабатывания переходов. Переходы в нашем автоматическом светофоре выполняются по таймауту, т. е. по истечении интервала времени, который прошел с момента прихода системы в данное состояние.

Рис. 4.63. Модель светофора

1. В состоянии движение светофор находится 10 секунд, 2. затем 7 секунд зеленый сигнал мигает, 3. в состоянии медленно 4 секунды горит желтый, 4. в течение 10 секунд движение запрещено и 5. 4 секунды светофор находится в состоянии приготовиться. В каждом состоянии светофора должен гореть определенный сигнал: в состоянии движение должен гореть зеленый, в состоянии приготовиться должны гореть красный и желтый одновременно и т. п. Создайте три параметра логического типа: красный, желтый и зеленый, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый (рис. 4.64).

Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны false. Мы создали стейтчарт именно для управления значениями этих параметров, каждое состояние отвечает за зажигание своего света или их комбинации. Например, в состоянии медленно должен гореть желтый, в состоянии stop должен загореться красный свет, а в состоянии приготовиться должны гореть красный и желтый одновременно.

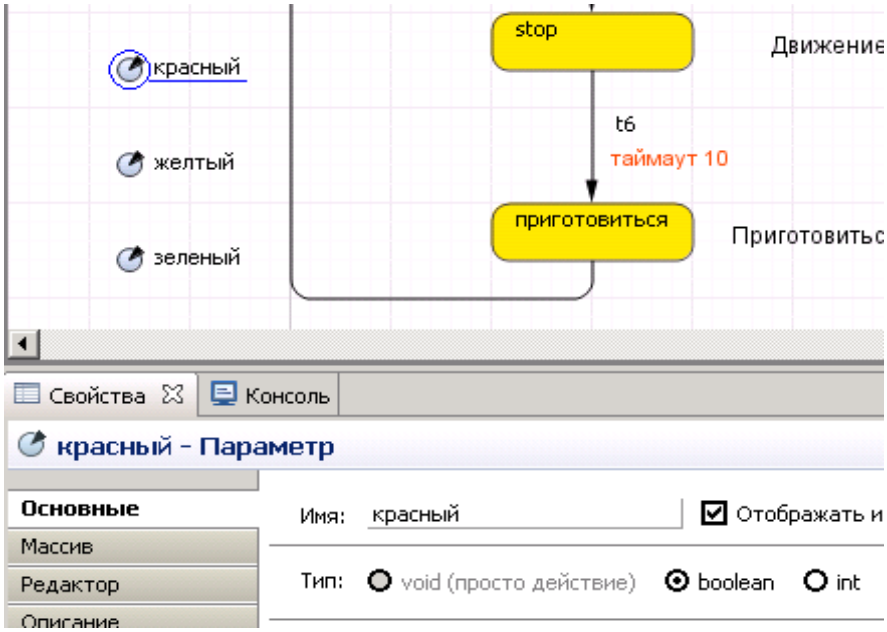


Рис. 4.64. Параметры

1. В свойствах состояния движение в поле Действие при входе запишите зеленый=true;, а в поле Действие при выходе запишите зеленый = false; (рис.4.65).

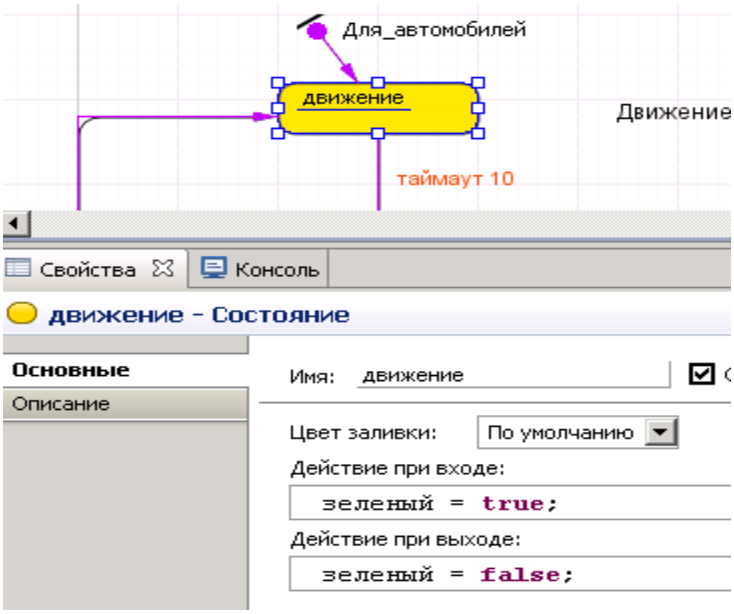


Рис. 4.65. Свойства состояния движения

2. То же самое запишите для состояния В гиперсостояния внимание, а у состояния А эти поля нужно оставить пустыми – когда светофор находится в этом состоянии, он не горит.

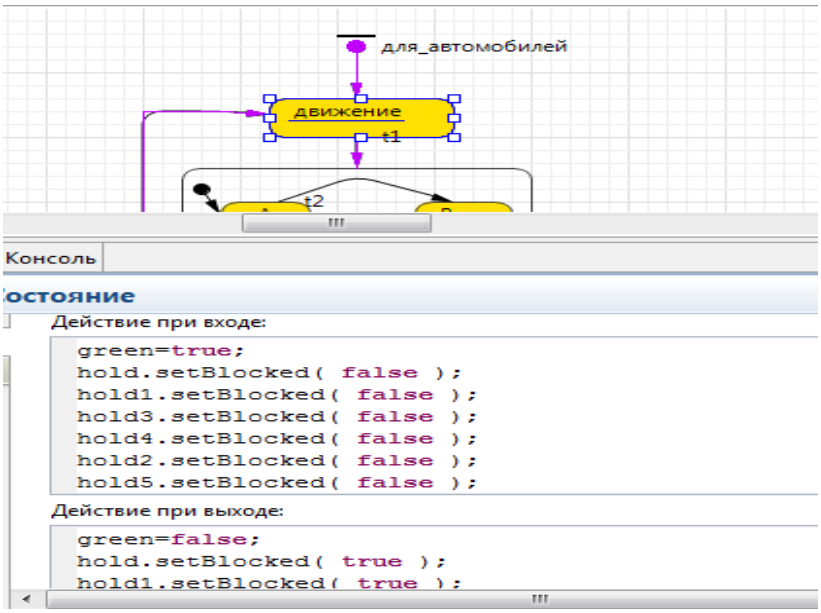


Рис. 4.66. Состояние Движение

3. Аналогично, в состоянии медленно нужно включить желтый сигнал, т. е. при входе в это состояние установить параметр желтый в true, а при выходе из этого состояния установить его в false.

4. Для состояния stop аналогично опишите состояния параметра красный, а для состояния приготовиться оба параметра красный и желтый нужно установить в true при входе, и в false при выходе.

Презентация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса. Графические объекты цвета сигналов светофора в презентации имеют динамические параметры, все остальные – статические. Светофор строится из трех овалов. Установим динамическое значение цвета верхнего сигнала светофора: если переменная красный истинна, то цвет должен быть red (красный), в противном случае его цвет нужно установить gray (серый). Это записывается следующим условным выражением на языке Java:

красный? red: gray

Цвет среднего и нижнего овалов следует установить в поле их динамических значений соответственно так:

желтый? yellow: gray

зеленый? green: gray

red, yellow, green и gray – предопределенные константы, обозначающие стандартные цвета.

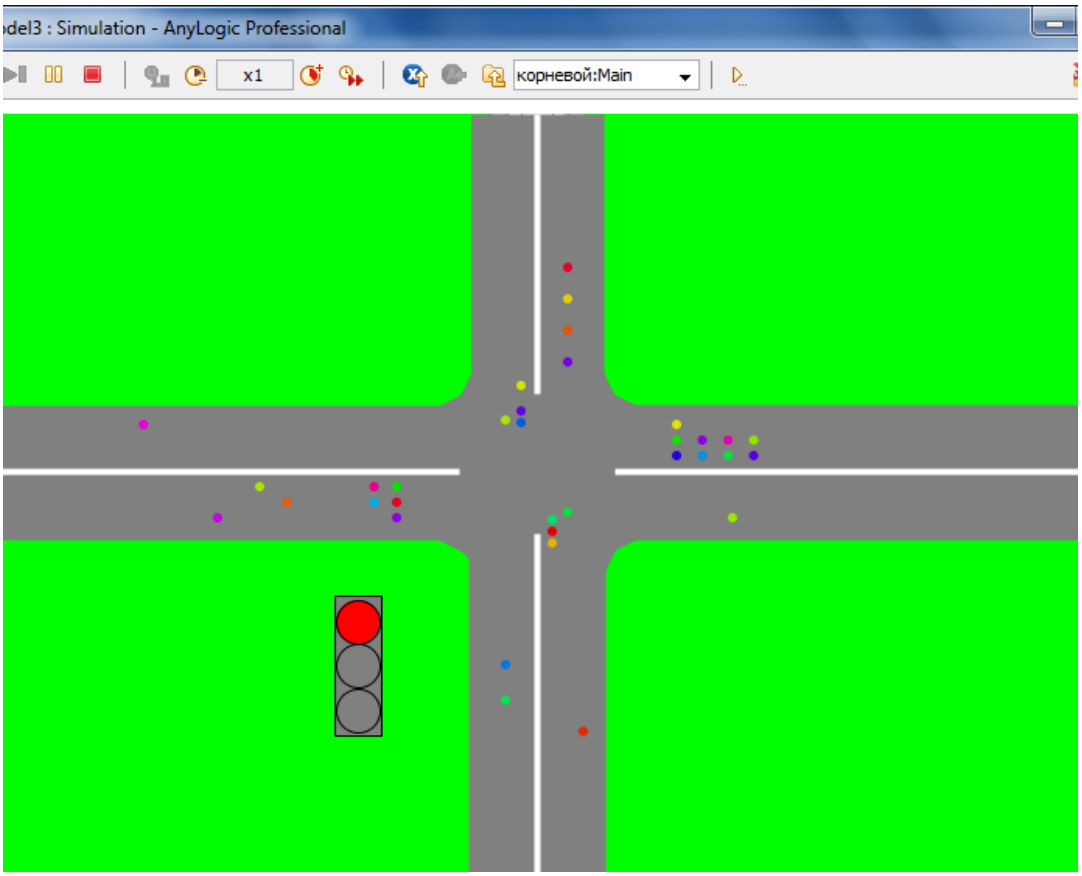


Рис. 4.67 Перекресток, регулируемый светофором.

Для состояний Движение и stop пропишем команды для активации объектов hold.

Также модернизируем внешний вид нашего перекрестка. С помощью среды AnyLogic можно смоделировать различные перекрестки. Можно построить разное количество линий движений, разное количество дорог, которые пересекаются. Можно также построить целую часть дороги. А с помощью стейтчартов можно задавать различное время для работы определенного цвета светофора. Тем самым можно добиться оптимальной работы светофора, что позволит устранить проблему пробок и облегчит жизнь всем автомобилистам.

Задание на моделирование:

1. Построить имитационную модель (рис.4.58 - 4.67) в среде моделирования Anylogic.
2. Провести компьютерные эксперименты в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

4.6. Системы моделирования сетей Петри

Сети Петри — математический аппарат для моделирования динамических дискретных систем. Впервые описаны Карлом Петри в 1962 году. В докторской диссертации "Связь автоматов" он сформулировал основные понятия теории связи асинхронных компонент вычислительной системы. Изучить теорию Петри можно по книгам [29-30,32], а также по многочисленным работам и статьям[2] приведенных в базе поисковых систем Google, Yandex и др. Рассмотрим основные понятия по сетям Петри согласно Википедии [3].

Сети Петри разрабатывались для моделирования систем с параллельными взаимодействующими компонентами.

Сеть Петри представляет собой двудольный ориентированный граф, состоящий из вершин двух типов: позициями P_i и переходами t_j . Вершины соединены дугами, причем вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться метки (маркеры), способные перемещаться по сети, так что поведение системы представляется в виде движения маркеров через переходы от начальной к конечной позициям.

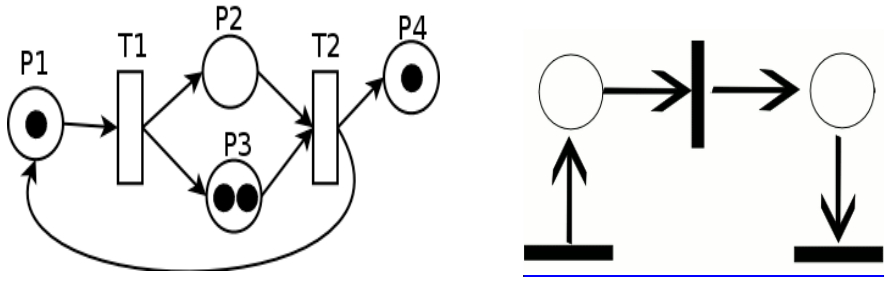


Рис. 4.68

Событием называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно, либо разновременно, при выполнении некоторых условий. Если дуга имеет кратность k , то по ней одновременно могут проходить k маркеров. Маркеры задерживаются в позиции на заданное время Z выполнения операции. Состояние сети Петри в определенный момент времени представляет собой распределение маркеров по позициям и называется маркировкой. Если в позиции P находится n, то в нее помещают n точек.

Основными свойствами сети Петри являются:

- ограниченность — число меток в любой позиции сети не может превысить некоторого значения K;

[2] http://bigor.bmstu.ru/?cnt/?doc=110_Simul/3018.mod/?cou=110_Simul/base.cou (16 марта 2012)
[3] http://ru.wikipedia.org/wiki/Сети_Петри

- безопасность — частный случай ограниченности, K=1;
- сохраняемость — постоянство загрузки ресурсов, $\sum A_i N_i = const$. Здесь N_i — число маркеров в i-той позиции, A_i — весовой коэффициент;
- достижимость — возможность перехода сети из одного заданного состояния (характеризуемого распределением меток) в другое;
- живость — возможность срабатывания любого перехода при функционировании моделируемого объекта.

В основе исследования перечисленных свойств лежит анализ достижимости.

Сети Петри — это аппарат для моделирования динамических дискретных систем (преимущественно асинхронных параллельных процессов). Сеть Петри определяется как четверка $\langle P, T, I, O \rangle$, где P и T— конечные множества позиций и переходов, I и O – множества входных и выходных функций. Другими словами, сеть Петри представляет собой двудольный ориентированный граф, в котором позициям соответствуют вершины, изображаемые кружками, а переходам — вершины, изображаемые утолщенными черточками; функциям I соответствуют дуги, направленные от позиций к переходам, а функциям O – от переходов к позициям.

Как и в системах массового обслуживания, в сетях Петри вводятся объекты двух типов: динамические – изображаются метками (маркерами) внутри позиций и статические — им соответствуют вершины сети Петри.

Распределение маркеров по позициям называют *маркировкой*. Маркеры могут перемещаться в сети. Каждое изменение маркировки называют событием, причем каждое событие связано с определенным переходом. Считается, что события происходят мгновенно и разновременно при выполнении некоторых условий.

Каждому условию в сети Петри соответствует определенная позиция. Совершению события соответствует срабатывание (возбуждение или запуск) перехода, при котором маркеры из входных позиций этого перехода перемещаются в выходные позиции. Последовательность событий образует моделируемый процесс.

Правила срабатывания переходов (рис. 4.69) конкретизируют следующим образом: переход срабатывает, если для каждой из его входных позиций выполняется условие $N_i \geq K_i$, где N_i — число маркеров в i-й входной пози-

ции, K_i — число дуг, идущих от i-й позиции к переходу; при срабатывании перехода число маркеров в i-й входной позиции уменьшается на K_i, а в j-й выходной позиции увеличивается на M_j, где M_j — число дуг, связывающих переход с j-й позицией.

На рис. 4.69 показан пример распределения маркеров по позициям перед срабатыванием, эту маркировку записывают в виде (2,2,3,1). После срабатывания перехода маркировка становится иной: (1,0,1,4).

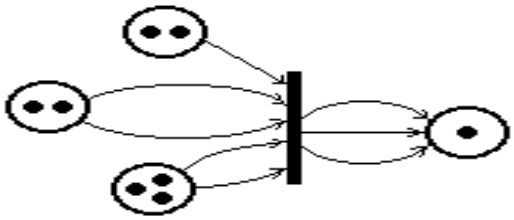


Рис. 4.69. Фрагмент сети Петри

Классификация сетей Петри

- Временная сеть Петри — переходы обладают весом, определяющим продолжительность срабатывания (задержку).
- Стохастическая сеть Петри — задержки являются случайными величинами.
- Функциональная сеть Петри — задержки определяются как функции некоторых аргументов, например, количества меток в каких-либо позициях, состояния некоторых переходов.
- Цветная сеть Петри — метки могут быть различных типов, обозначаемых цветами, тип метки может быть использован как аргумент в функциональных сетях.
- Ингибиторная сеть Петри — возможны ингибиторные дуги, запрещающие срабатывания перехода, если во входной позиции, связанной с переходом ингибиторной дугой, находится метка.
- Иерархическая сеть — содержит не мгновенные переходы, в которые вложены другие, возможно, также иерархические, сети. Срабатывание такого перехода характеризует выполнение полного жизненного цикла вложенной сети.
- WF-сети.

Можно вводить ряд дополнительных правил и условий в алгоритмы моделирования, получая ту или иную разновидность сетей Петри. Так, прежде

всего полезно ввести модельное время, чтобы моделировать не только последовательность событий, но и их привязку ко времени. Это осуществляется приданием переходам веса — продолжительности (задержки) срабатывания, которую можно определять, используя задаваемый при этом алгоритм. Полученную модель называют *временной сетью Петри*.

Если задержки являются случайными величинами, то сеть называют *стохастической сетью Петри*. В стохастических сетях возможно введение вероятностей срабатывания возбужденных переходов. Так, на рис. 4.50 представлен фрагмент сети Петри, иллюстрирующий конфликтную ситуацию — маркер в позиции p может запустить либо переход t_1, либо переход t_2. В стохастической сети предусматривается вероятностный выбор срабатывающего перехода в таких ситуациях.

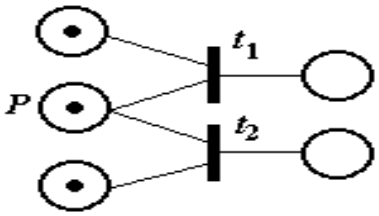


Рис. 4.70. Конфликтная ситуация

Если задержки определяются как функции некоторых аргументов, которыми могут быть количества маркеров в каких-либо позициях, состояния некоторых переходов и т.п., то имеем *функциональную сеть Петри*.

Во многих задачах динамические объекты могут быть нескольких типов, и для каждого типа нужно вводить свои алгоритмы поведения в сети. В этом случае каждый маркер должен иметь хотя бы один параметр, обозначающий тип маркера. Такой параметр обычно называют цветом; цвет можно использовать как аргумент в функциональных сетях. Сеть при этом называют *цветной сетью Петри*.

Среди других разновидностей сетей Петри следует упомянуть *ингибиторные сети Петри*, характеризующиеся тем, что в них возможны запрещающие (ингибиторные) дуги. Наличие маркера во входной позиции, связанной с переходом ингибиторной дугой, означает запрещение срабатывания перехода.

Для моделирования систем сетями Петри можно использовать программные продукты: проблемно-ориентированные имитаторы NetStar или HPSim, приложение Stateflow системы MATLAB и другие системы.

Таблица 4.2

	Программные продукты моделирующие сети Петри	Разработчики
1.	Эмулятор сетей Петри NetStar	Кемеровский научный центр СО РАН, Россия
2.	Эмулятор сетей HP Sim	разработка HP Software
3.	Система Tina	разработка лаборатории LAAS во Франции.
4.	Design/CPN	разработка университета Орху-са (Дания).
5.	Информационная система CPN Tools	разработка университета Орху-са (Дания).
6.	Приложение Stateflow си-стемы MATLAB.	компания **MathWorks**

На рис. 4.71 и 4.72 приведены рабочие окна этих эмуляторов, содержащие контекстное меню, панель управления, панель инструментов, область построения модели и строку состояния.

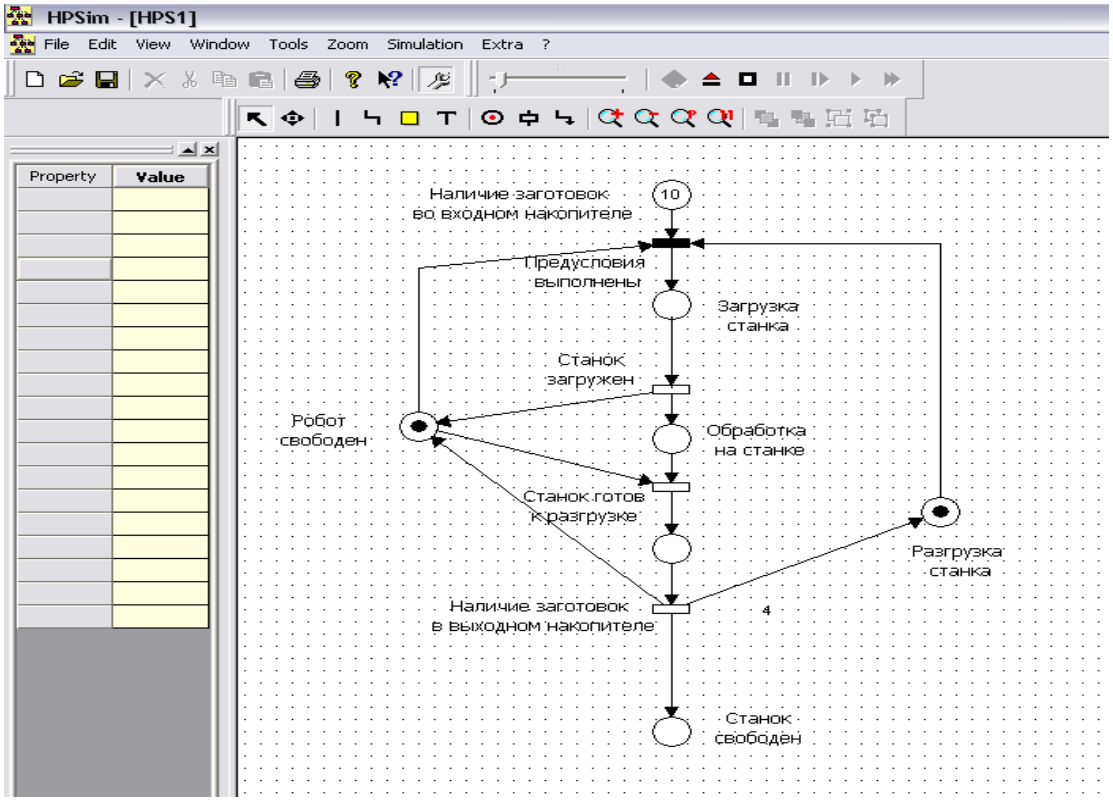


Рис. 4.71. Внешний интерфейс ПО HPSim

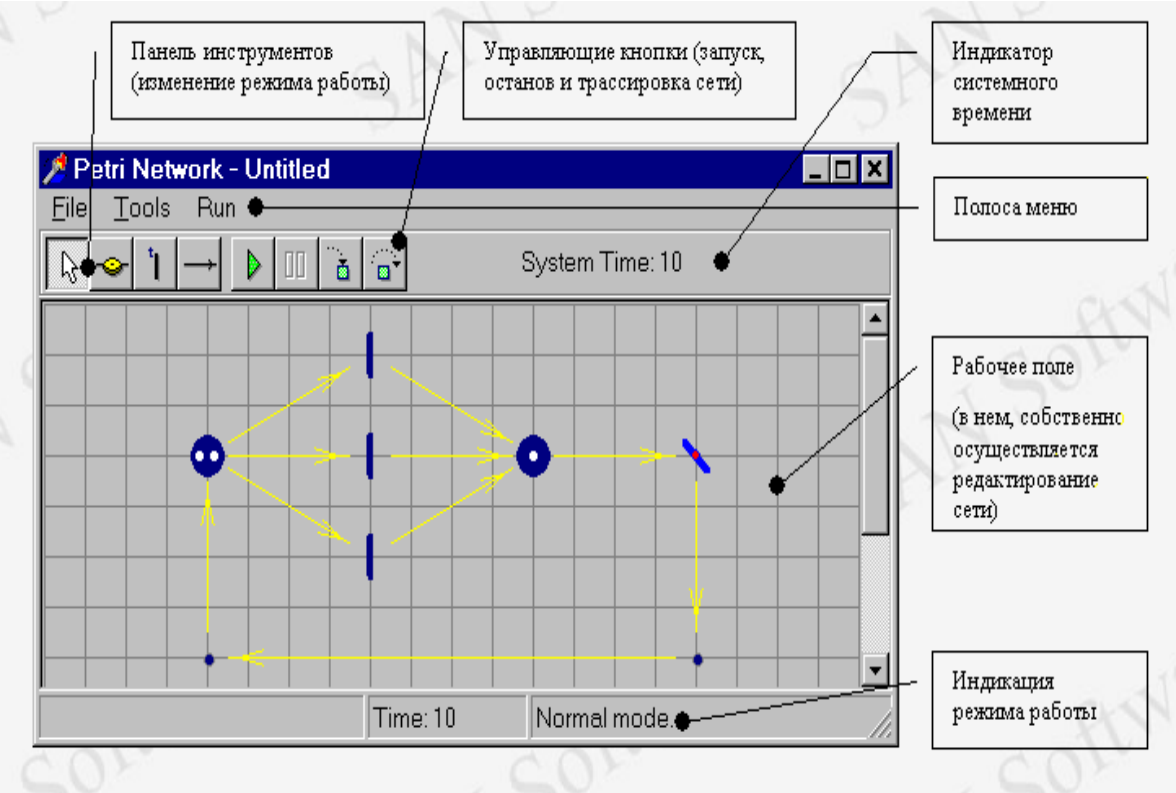


Рис. 4.72. Внешний интерфейс эмулятора сетей Петри (NetStar)

Задание на моделирование:

1. Построить имитационную модель работы сети Петри в (рис.4.71 - 4.72) в среде моделирования HPSim.
2. Провести компьютерное моделирование в зависимости от характеристик среды и начальных условий.
3. Провести обработку и анализ результатов компьютерного моделирования.

Заключение

Технологии моделирования настолько быстро развиваются, что нет возможности охватить все созданные пакеты моделирования. Любой солидный исследовательский институт, занимающийся моделированием, считает необходимым создать свой пакет моделирования. Однако не все получают распространение на рынке пользователей. Это связано в основном с маркетинговой политикой фирмы разработчика. Изучая рынок инструментов моделирования в зависимости от типа решаемой задачи и области применения можно выделить ряд основных подходов к созданию программ для моделирования:

1) разработка программ моделирования для аналитических и численных моделей типа D-схем с помощью языков и сред программирования (Pascal, Visual Basic, Fortran, C++ и др.);
2) разработка программ моделирования для моделей типа D-схем с помощью технологий пакетов компьютерной математики (Derive, Maple, MathCAD, Mathematica и др.);
3) разработка имитационных компьютерных моделей типа D – схем и F-схем с помощью аналоговых схемотехнических пакетов моделирования (MATLAB и его приложения SIMULINK, Vissim, Multisim, LabVIEW, VenSim, MyStudium и др.);
4) разработка имитационных компьютерных моделей типа Q – схем с помощью систем моделирования, реализующих объектно-ориентированный и дискретно-событийный подходы к программированию (GPSS, AnyLogic, Arena, Swarm (Objective C), Simulink, SimBioSys, EXTEND фирмы Imagine That Inc и др.);
5) разработка имитационных компьютерных моделей на основе методологии системной динамики (AnyLogic, Arena, SimBioSys, eM-Plant, Tecnomatix Plant Simulation, SimuLab, Vensim, Powersim, Dynamo, Stella, Ithink и др.);
6) разработка имитационных компьютерных моделей с помощью языков и систем компьютерного моделирования, реализующие агентный подход к моделированию (AnyLogic, Swarm и его расширение MAML - Мультиагентный язык моделирования, SimAgent, AgentSpeak, TeleScript, Oz и др.);
7) разработка моделей имитации и планирования вычислительных сетей в инструментальных средствах, которые обычно включают в себя модули обработки, эмулирующие сетевые устройства: мосты, концентраторы и пр. Для моделирования сетей применяются различные программные продукты: се-

мейство продуктов COMNET, семейство CANE, семейство OPNET, BONeS, NetMaker, Optimal Performance, NanoCAD, PlanNet, Stressmagic, Prophesy, система NETWORK II.5 и др.;

8) разработка моделей графических образов реальных объектов (AutoCAD, ArchiCAD, Compass, 3DStudioMax, Aartform Curvy3D, Google SketchUp, VirtualGrid VRMesh, REALVIZ Image Modeler, и др.).

http://www. 3dmax.com,
http://www.curvy3d.com
http://www.sketchup.com
http://www.vrmesh.com
http://imagemodeler.realviz.com

9) разработка статистических и эконометрических моделей типа линейной и нелинейной регрессии (OLS), авторегрессионной модели, модели линейной вероятности (LPM), логит моделей (Logit), пробит моделей (Probit)и др. (Excel, SPSS, Statistica, и др.).

Знание и применение систем компьютерной математики, технического и имитационного моделирования позволяют модельщикам оперативно выбрать систему моделирования, построить адекватные модели, способы их решения, перейти к полномасштабному исследованию реального явления или процесса на модели, оценить решения моделей и представить поведение и закономерности изучаемого явления.

Здесь мы представили обзор различных подходов и направлений разработок компьютерных комплексов для моделирования реальных процессов и объектов. Разработкой компьютерных моделей и программных комплексов для компьютерного моделирования в настоящее время занимаются не только в классических, технических и педагогических университетах в рамках учебного процесса и научного интереса, но и консалтинговые и компьютерные фирмы. Отметим, что одним из важных направлений работ коллективов и авторов по разработкам компьютерных моделей и комплексов является своевременное представление информации о программном продукте научной общественности на различных научных конференциях и форумах.

О важности развития направления разработок имитационных компьютерных технологий, средств и моделей говорит тот факт, что во всех странах создаются сообщества по имитационному моделированию, целью которых является объединение и координация интересов учёных и специалистов-практиков в области имитационного моделирования (см. приложение 3).

Рекомендуемая литература

К главе 1

1. Андон, Ф.И., Коваль, Г.И., Коротун, Т.М., Суслов, В.Ю. Основы инженерии качества программных систем. – К: Академпериодика, 2002.– 502 с.
2. Андон, Ф.И., Лаврищева, Е.М. Методы инженерии распределенных компьютерных систем. – Киев: Изд-во «Наукова думка», 1997г. – 228 с.
3. Арчибальд, Р. Управление высокотехнологичными программами и проектами / Рассел Д. Арчибальд; ред. А. Д. Баженов, А. О. Арефьева; пер. с. англ. Е. В. Мамонтова. – 3-е изд., перераб. и доп. – М.: ДМКПресс; Компания АйТи, 2006. – 472 с.
4. Бабенко, Л.П., Лаврищева, Е.М. Основы программной инженерии. – М.: Изд-во «Знание», 2001. – 269с.
5. Басс, Л., Клементс, П., Кацман, Р. Архитектура программного обеспечения на практике. – 2-е изд. – СПб.: Питер, 2006. – 575 с.
6. Брауде, Э. Технология разработки программного обеспечения – СПб.: Питер, 2004. – 655 с.
7. Буч, Г. Объектно-ориентированное проектирование с примерами применения: пер. с англ. – М.: Конкорд, 1992.- 519 с., ил.
8. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с.
9. Венчковский, Л.Б. Разработка сложных программных изделий: учеб. пособие для вузов. - М.:ЗАО “Финстатинформ”, 1999. – 109 с.
10. Вигерс, Карл. Разработка требований к программному обеспечению. – М.: Издательско-торговый дом «Русская редакция», 2004. – 576 с.
11. Гвоздев, В.Е., Колоденкова, А.Е. Системные вопросы проектирования программных продуктов. Учебное пособие. –Уфа: АН РБ Изд-во «Гилем», 2010. - 188с.
12. Гецци, К., Джазайери, М., Мандриоли, Д. Основы инженерии программного обеспечения. – СПб. : БХВ-Петербург, 2005.
13. Гейн, К., Сарсон, Т. Системный структурный анализ: средства и методы. – М.: Эйтекс, 1992.
14. ГОСТ 19.001–77. Единая система программной документации. Общие положения.
15. ГОСТ 19.502–78. Единая система программной документации. Общее описание. Требования к содержанию и оформлению.
16. ГОСТ 19.504–79. Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению.
17. ГОСТ 28159-89. Оценка качества программных средств. М.: Изд-во стандартов, 1990 - 38 с.
18. ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
19. ГОСТ Р ИСО/МЭК 8631–94. Информационная технология. Программные конструктивы и условные обозначения для их представления.
20. ГОСТ Р ИСО/МЭК 12119–2000. Информационная технология. Пакеты программ. Требования к качеству и тестирование.

21. ГОСТ Р ИСО/МЭК 12207–99. Информационная технология. Процессы жизненного цикла программных средств.
22. ГОСТ Р ИСО/МЭК 15910–2002. Информационная технология. Процесс создания документации пользователя программного средства.
23. ГОСТ Р ИСО/МЭК 9126–93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.
24. ГОСТ Р ИСО/МЭКТО 15271–2002. Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств).
25. ГОСТ Р ИСО/МЭКТО 16326–2002. Программная инженерия. Руководство по применению ГОСТ Р ИСО/МЭК 12207 при управлении проектом.
26. ГОСТ Р ИСО/МЭКТО 9294–93. Информационная технология. Руководство по управлению документированием программного обеспечения.
27. Гради Буч. Объектно-ориентированное проектирование.- 3-е издание. – М.:"Бином", 1998.-560 с.
28. Гудов, А.М., Завозкин, С.Ю., Трофимов, С. Н. Учебно-методический комплекс. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. http://unesco.kemsu.ru/study_work/ method/po/UMK/lab_pract/lab01.html, вход свободный (2012-08-18)
29. Гультяев, А.К. MS PROJECT 2002. Управление проектами. Русская версия; Практическое пособие. – Спб.: КОРОНА, 2003. –592 с.
30. ДСТУ 2844–1994. Программные средства ЭВМ. Обеспечение качества. Термины и определения.
31. ДСТУ 2850–1994. Программные средства ЭВМ. Обеспечение качества. Показатели и методы оценки качества программного обеспечения.
32. Единая система программной документации (сборник стандартов). - М.: Изд-во стандартов, 1988. – 143с.
33. Канер, С., Фолк, Д., Нгуен, Е.К. Тестирование программного обеспечения: Пер с англ. – К.: DiaSoft. – 2000. – 544 с.
34. Константайн, Л., Локвуд, Л. Разработка программного обеспечения. – СПб.: Питер, 2004. – 592 с.
35. Коробицын, В.В., Фролова, Ю.В. Информатика: учебное пособие. – ОМСК: Изд-во Наследие. Диалог-Сибирь, 2004. – 140с.
36. Лаврищева, Е.М., Петрухин, В.А. Методы и средства инженерии программного обеспечения: учебник. – М.: Московский физико-технический институт (государственный университет), 2006. – 304с.
37. Липаев, В.В. Методы обеспечения качества крупномасштабных программных средств. – М.: СИНТЕГ.– 2003.–520 с.
38. Липаев, В.В. Отладка сложных программ. –М.: Энергоатомиздат, 1993.–296с.
39. Липаев, В.В. Проектирование программных средств: учеб. пособие для вузов по спец. "Автом. сист.обр. информ. и упр." - М.: Высш. шк., 1990. – 303 с.
40. Липаев, В.В. Тестирование программ. – М: Радио и связь, 1986. –295с.
41. Маклаков, С.В. BPWin и ERWin – CASE – средства разработки информационных систем. – М.: Диалог-МИФИ, 1999.
42. Мацяшек, Л.А., Лионг, Б.Л. Практическая программная инженерия на основе учебного примера. – М.: БИНОМ. Лаборатория знаний, 2010. – 956с.

43. Мелкумян, Б.В. Разработка и стандартизация программных средств и информационных технологий: учебно-методическое пособие. - М.: МИЭМП, 2006. – 231 с.
44. Орлов, С.А. Технологии разработки программного обеспечения. Учебник для вузов. СПб: Питер, 2002. – 463с.
45. РД IDEF 0 – 2000. Методология функционального моделирования IDEF0. Руководящий документ. Госстандарт России. – М.: ИПК Изд-во стандартов, 2000. – 80с.
46. Трофимов, С.А. CASE-технологии: практическая работа в Rational Rose.– 2–е изд.– М.: Бином–Пресс, 2002.–288с.
47. Уилсон, С.Ф., Мєйлс, Б., Ленгрейв, Т. Принципы проектирования и разработки программного обеспечения: учебный курс MCSD, пер. с англ. – М.: Из–во торговый дом «Русская редакция», 2000. – 608с.
48. Федотова, Д.Э., Семенов, Ю.Д., Чижик, К.Н. CASE-технологии: практикум. – М.: Горячая линия-Телеком, 2003. – 160 с.
49. Якунин, Ю. Ю. Технологии разработки программного обеспечения. Версия 1.0 [Электронный ресурс]: электрон. учеб. пособие / Ю. Ю. Якунин. – Красноярск: ИПКСФУ, 2008. – (Технологии разработки программного обеспечения): УМКД № 183-2007. Номер гос. регистрации в ФГУП НТЦ «Информрегистр» 0320802414 от 21.11.2008 г.

К главе 2

1. Алексеев, Д.В. Компьютерное моделирование физических задач в Microsoft Visual Basic. – М.: СОЛОН –Пресс, 2004. – 528с.
2. Алексеев, Е.Р., Чеснокова О.В. MATLAB 7: самоучитель. – М.: НТ Пресс, 2006. – 464 с.
3. Бартоломью, Д. Стохастические модели социальных процессов. - М.: Финансы и статистика, 1985.
4. Батоврин, В.К., Бессонов, А.С., Мошкин, В.В. "LabVIEW: практикум по электронике и измерительной технике".– М.: Приборкомплект, 2006.
5. Булыгин, В.С., Гришанин, Ю.С. и др. Основы теории автоматического управления / под ред. Н.В.Судзиловского. – М.: Машиностроение, 1985. – 512с.
6. Бусленко, Н.П. Моделирование сложных систем. - М.: Изд-во «Наука», 1968. – 356 с.
7. Бутырин, П.А., Васьковская, Т.А., Каратаев, В.В., Матеркин, С.В. Автоматизация физических исследований и эксперимента: компьютерные измерения и Виртуальные Приборы на основе LabVIEW 7 Express (30 лекций). – М.: Приборкомплект, 2006.
8. Введение в математическое моделирование: учеб. пособие / под ред. П.В. Трусова . – М.: Университетская книга, Логос, 2007. – 440с.
9. Гвоздев, В.Е., Колоденкова, А.Е. Системные вопросы проектирования программных продуктов. Учебное пособие. –Уфа: АН РБ Изд-во «Гилем», 2010. – 1 88с.
10. Горбацевич, Е.Д., Левинзон, Ф.Ф. Аналоговое моделирование систем управления / под ред. Е.П.Попова и И.М. Тетельбаума. – М.: Наука, 1984. – 336с.
11. Гультяев, А.К. Имитационное моделирование в среде Windows. MatLAB 5.3. Практическое пособие. – СПб.: КОРОНА принт, 2001. – 400с.

12. Гуц, А. К., Паутова, Л. А. Глобальная этносоциология / А. К. Гуц, Л. А. Паутова. - 2-е изд., доп. - М. : Либроком, 2009. – 247 с.
13. Гуц, А.К. Коробицын, В.В., Лаптев, А.А., Паутова, Л.А., Фролова, Ю.В. Социальные системы. Формализация и компьютерное моделирование: учеб. пособие. – Омск: Омск. гос. ун-т, 2000. – 160 с.
14. Гуц, А.К. Коробицын, В.В., Лаптев, А.А., Паутова, Л.А., Фролова, Ю.В. Компьютерное моделирование. Инструменты для исследования социальных систем. – Омск: Омск. гос. ун-т, 2000. – 92 с.
15. Гуц, А.К. Коробицын, В.В., Лаптев, А.А., Паутова, Л.А., Фролова, Ю.В. Математические модели социальных систем. – Омск: Омск. гос. ун-т, 2000. – 256с.
16. Джеффри Тревис. "LabVIEW для всех". – М.: Приборкомплект, 2006.
17. Дьяконов, В.П. VisSim+MathCAD+MATLAB. Визуальное математическое моделирование. – М: СОЛОН-Пресс, 2004. – 384с.
18. Дьяконов, В.П. MATLAB 6/6.1/6.1. Simulink 6.5 в математике и моделировании. Полное руководство пользователя. М.: Солон-Пресс, 2003.
19. Дьяконов, В.П. Maple 7: Учебный курс. СПб.: Питер, 2002.
20. Дьяконов, В.П. Mathematica 4: Учебный курс. СПб.: Питер, 2001.
21. Дьяконов, В.П. MATLAB 6: Учебный курс. СПб.: Питер, 2001.
22. Дьяконов, В.П. Системы компьютерной математики Derive: самоучитель. М.: Солон-Р, 2002.
23. Дьяконов, В.П., Круглое, В.В. Математические пакеты расширения MATLAB: спец. справочник. – СПб.: Питер, 2001.
24. Дьяконов, В.П. MathCAD 2001: Специальный справочник. – СПб.: Питер, 2002.
25. Дьяконов, В.П. Simulink 6. Специальный справочник. – СПб.: Питер, 2002.
26. Загидуллин, Р.Ш. LabView в исследованиях и разработках. – М:. Горячая линия – Телеком, 2005. -352с.
27. Карлащук, В.И. Электронная лаборатория на IBM PC. Программа Electronics Workbench.– М.: Солон-Р, 2000.- 504с.
28. Краснощёков, П.С., Петров, А.А. Принципы построения моделей. – М.: Фазис, 2000. – 400 с.
29. Маликов, Р.Ф. Практикум по компьютерному моделированию физических явлений и объектов: учеб. пособие. – Уфа : Изд-во БГПУ, 2004. -236с.
30. Маликов, Р.Ф. Основы математического моделирования. – М.: Горячая линия – Телеком, 2010. – 368 с.
31. Модели систем автоматического управления и их элементов: учеб. пособие / С.Т.Кусимов, Б.Г.Ильясов, В.И.Васильев и др. – М.: Машиностроение, 2003. – 214с.
32. Пейч, Л.И., Точилин, Д.А., Поллак, Б.П. "LabVIEW для новичков и специалистов". – М:. Горячая линия – Телеком, 2004. -321с.
33. Плотников, А.М., Рыжиков, Ю.И., Соколов, Б.В. Современное состояние и тенденции развития имитационного моделирования в Российской Федерации, 19-21 октября 2011. – Санкт-Петербург, ИММОД-2011. – с.1-47.
34. Самарский, А.А., Михайлов, А.П. Математическое моделирование: Идеи. Методы. Примеры. М: Наука, 1999. - 320 с.

35. Советов, Б.Я., Яковлев, С.Я. Моделирование систем: учебник для бакалавров. – М.: Изд-во Юрайт, 2012. – 343с.
36. Советов, Б.Я., Яковлев, С.А. Моделирование систем: практикум: учеб. пособие. – М.: Высш. шк., 2005. – 295с.
37. Суранов, А.Я.. "LabVIEW 7 Express: справочник по функциям". – М.: Приборкомлект, 2006.
38. Тарасевич, Ю. Ю. Математическое и компьютерное моделирование. Вводный курс: учеб. пособие. 4-е изд., испр. – М.: Едиториал УРСС, 2004. – 152 с.
39. Колесов, Ю.Б. Сениченков, Ю.Б. Моделирование систем: практикум по компьютерному моделированию. – СПб.:БХВ-Петербург, 2007. – 352с.
40. Колесов, Ю.Б. Сениченков, Ю.Б. Моделирование систем: динамические и гибридные системы. – СПб.:БХВ-Петербург, 2006. – 224с.
41. Колесов, Ю.Б. Сениченков, Ю.Б. Моделирование систем: объектно-ориентированный поход. – СПб.: БХВ-Петербург, 2006. – 192с.
42. Морозов, В.К., Рогачев, Г.Н. Моделирование информационных и динамических систем. - М: Академия, 2011. – 384с.
43. Шелухин, О.И. Моделирование информационных систем. – М.: Горячая линия-Телеком, 2011. – 536с.

К главе 3

1. Антонов, А.В. Системный анализ. – М.: Высшая школа, 2004. – 453с.
2. Бережная, Е.В., Бережной, В.И. Математические методы моделирования экономических систем: учеб. пособие. – М.: Финансы и статистика, 2006. – 432с.
3. Бусленко, Н.П. Моделирование сложных систем. –М.: Главная редакция физико-математической литературы изд-ва «Наука», 1968. – 356с.
4. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ // пер. с англ. под ред. Э.А.Трахтенгерца. – Мир, 1981. -319 с.
5. Гвоздев, В.Е., Колоденкова, А.Е. Системные вопросы проектирования программных продуктов. Учебное пособие. –Уфа: АН РБ Изд-во «Гилем», 2010. – 188с.
6. Дружинин, В.В., Конторов, Д.С. Проблемы системологии (проблемы теории сложных систем). – М.: Сов. Радио, 1976. – 296с.
7. Егоров, В.А., Каллистов, Ю.Н., Митрофанов, В.Б., Пионтковский, А.А. Математические модели глобального развития. – Л.: Гидрометеоиздат, 1980.
8. Емельянов, А.А., Власова, Е.А., Дума, Е.А. Имитационное моделирование экономических процессов. — М.: Финансы и статистика, 2002. – 364 с.
9. Имитационное и статистическое моделирование: практикум для студентов мат. и экон. спец. / В. И. Лобач, В. П. Кирлица, В. И. Малюгин, С. Н.Сталевская. – Мн.: БГУ, 2004. – 189с.
10. Калянов, Г.Н. CASE – структурный системный анализ. – М.: Лорн, 1996. – 242 с.
11. Качала, В.В. Основы теории систем и системного анализа. – М.: Горячая линия. – Телеком, 2007. – 216 с.
12. Кондукова Е.А. Моделирование систем. Инструментальные средства: учеб. пособие. – СПб.: Изд-во «Питер», 2004. – 165 с.
13. Коробицын, В.В., Фролова, Ю.В. Информатика: учебное пособие. – ОМСК: Изд-во «Наследие. Диалог-Сибирь», 2004. – 140с.

14. Кузнецов, Ю.А., Перова, В.И. Применение пакетов имитационного моделирования для анализа математических моделей экономических систем: Учебно-методический материал по программе повышения квалификации «Применение программных средств в научных исследованиях и в преподавании математики и механики». – Нижний Новгород, 2007. – 98 с.
15. Левашов, В.К. Устойчивое развитие общества: парадигма, модели, стратегия. – М.: Academia, 2001.
16. Лоу, А.М., Кельтон, В.Д. Имитационное моделирование. Классика CS. 3-е изд. – С-Пб.: Питер, 2004.
17. Месарович, М., Такахара, Я. Общая теория систем: математические основы. – М.: Мир, 1978.
18. Перегудов Ф.И., Тарасенко Ф.П. Введение в системный анализ: Учебник для вузов. – М.: Высш.шк., 1985. – 367с.
19. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-еизд. – СПб.: Питер, 2007. – 544 с.
20. Системный анализ и принятие решений: Словарь-справочник: учеб. пособие для вузов / под ред. В.Н.Волковой, В.Н.Козлова. – М.: Высшая школа, 2004. – 616с.
21. Словари и энциклопедии на Академике [Электронный ресурс] // AnyLogic. – Режим доступа: http://dic.academic.ru/dic.nsf/ruwiki/602198/, свободный.
22. Советов, Б.Я., Яковлев, С.А. Моделирование систем: учеб. пособие для бакалавров. – М.: Изд-во Юрайт, 2012. –343с.
23. Советов, Б.Я., Яковлев, С.А. Моделирование систем. Практикум: учеб. пособие. – М.: Высш. шк., 2005. – 295с.
24. Спицнадель, В.Н. Основы системного анализа. Учебное пособие. – СПб.: Изд. Дом «Бизнес-пресса», 2000. – 326с.
25. Сурмин, Ю.П. Теория систем и системный анализ: учеб. пособие. – Киев: МУАП, 2003. – 368с.
26. Сухомлин, В. А. Введение в анализ информационных технологий: учебник для вузов. – М.: Горячая линия-Телеком, 2003. – 427 с.
27. Фаулер, М., Скотт, К. UML в кратком изложении. Применение стандартного языка объектного моделирования : пер. с англ. – М: Мир.: 1999. –191 с.
28. Федотова, Д.Э., Семенов, Ю.Д., Чижик, К.Н.. CASE-технологии: практикум. – М.: Горячая линия-Телеком, 2003. – 160 с.
29. Форрестер, Дж. Мировая динамика. М., Наука, 1978.
30. Чеппел, Д. Технологии ActiveX и OLE: пер. с англ. – М.: Издательский отдел "Русская редакция" ТОО "Channel Trading Ltd.", 1997. – 320 с.
31. Чернецки, К., Айзенекер, У. Порождающее программирование. Методы, инструменты, применение.– Издательский дом «Питер».– Москва– Санкт-Петербург… Харьков, Минск. – 2005. –730с.
32. Электронный учебник по дисциплине «Математические модели системного анализа» Электронный ресурс] // Имитационное моделирование. – Режим доступа: http://ermak.cs.nstu.ru/mmsa /glava3. htm, свободный.
33. Электронный журнал «Дифференциальные уравнения и процессы управления №1, 1997. Парийская Е.Ю. Сравнительный анализ математических моделей и подходов к моделированию и анализу непрерывно-дискретных систем. http://www.neva.ru/journal/pdf/1997/j004.pdf.

34. Калман Р., Фалб П., Арбиб М.: Очерки по математической теории систем. – М: Мир, 1971.
35. Программное обеспечение моделирования непрерывно-дискретных систем / под ред. В.Глушкова. – М: Наука, 1975.
36. Maler O. Hybrid Systems and Real-World Computations. In Workshop on Theory of Hybrid Systems, Lyndby, Denmark, June 1992, Springer-Verlag.
37. Nicollin X., Olivero A., Sifalis Y., Yovine S.: An Approach to the Description and Analysis of Hybrid Systems. Hybrid Systems, Lecture Notes in Comp. Sci 736, p.149-178. Springer-Verlag, 1993.
38. Henzinger T., Nicollin X., Sifalis J., Yovine S.: Symbolic Model-Checking for Real-Time Systems. In Proc. 7th LICS. IEEE Comp. Soc. Press, 1992.

К главе 4

1. Алексеев, Е.Р., Чеснокова, О.В. MATLAB 7: самоучитель. – М.: НТ Пресс, 2006. – 464 с.
2. Боев, В.Д., Кирик, Д.И., Сыпченко, Р.П. Компьютерное моделирование: пособие для курсового и дипломного проектирования. — СПб.: ВАС, 2011. — 348 с.
3. Боев, В.Д., Сыпченко, Р.П. Компьютерное моделирование. – ИНТУИТ.РУ, 2010. – 349 с.
4. Дьяконов В.П. VisSim+MathCAD+MATLAB. Визуальное математическое моделирование. – М: СОЛОН-Пресс, 2004. – 384с.
5. Дьяконов, В.П. Maple 7: учебный курс. – СПб.: Питер, 2002.
6. Дьяконов, В.П. Mathematica 4: Учебный курс. – СПб.: Питер, 2001.
7. Дьяконов, В.П. MATLAB 6/6.1/6.1. Simulink 6.5 в математике и моделировании. Полное руководство пользователя. – М.: Солон-Пресс, 2003.
8. Дьяконов, В.П. MATLAB 6: учебный курс. – СПб.: Питер, 2001.
9. Дьяконов, В.П., Круглое В. В. Математические пакеты расширения MATLAB: специальный справочник. – СПб.: Питер, 2001.
10. Дьяконов, В.П. MathCAD 2001: специальный справочник. СПб.: Питер, 2002.
11. Дьяконов, В.П. Simulink 6: специальный справочник. СПб.: Питер, 2002.
12. Загидуллин, Р.Ш. LabView в исследованиях и разработках. – М:. Горячая линия – Телеком, 2005. -352с.
13. Карлащук, В.И. Электронная лаборатория на IBM PC. Программа Electronics Workbench.– М.: Солон-Р, 2000.- 504с.
14. Карпов, Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. – СПб.: БХВ-Петербург, 2005. – 400 с.
15. Кирьянов, Д. MathCAD 12. – СПб.: БХВ-Петербург, 2005. – 576с.
16. Киселева, М. В. Имитационное моделирование систем в среде AnyLogic : учебно-методическое пособие / М. В. Киселёва. Екатеринбург : УГТУ - УПИ, 2009. – 88 с.
17. Лабораторный практикум. Имитационное моделирование информационных систем в пакете Arena 9.0. Сост.- Г.Г. Ахметшина. -Уфа, БГАУ, 2010. – 52с.
18. Маликов, Р.Ф. Основы систем компьютерного моделирования. Учебное пособие. – Уфа: Изд-во БГПУ, 2008. – 280с.

19. Мезенцев, К.Н. Моделирование систем в среде AnyLogic 6.4.1. Учебное пособие. Ч. 1,2 / под ред. заслуженного деятеля науки РФ, д.т.н., профессора А.Б.Николаева. МАДИ. — М:2011. -103 с.
20. Осоргин, А.Е. AnyLogic 6: лабораторный практикум. – Самара: ПГК, 2011. – 100 с.
21. Плохотников, К.Э. Вычислительные методы. Теория и практика в среде MATLAB: курс лекций: учеб. пособие для вузов. – М.: Горячая линия – Телеком, 2009. – 496 с.
22. Поршнев, С.В. Компьютерное моделирование физических процессов в пакете MATLAB.+CD: учеб. пособие. – М.: Изд-во «Лань», 2011. -736с.
23. Поршнев, С.В. Компьютерное моделирование физических процессов с использованием пакета Mathcad: учеб. пособие. – М: Горячая линия – Телеком, 2002. –252с.
24. Поршнев, С.В. Компьютерное моделирование физических систем с использованием пакета MathCAD: учеб.пособие. – М.: Горячая линия –Телеком, 2006. – 319с.
25. Семененко, М.Г. Введение в математическое моделирование. – М: Солон-Р, 2002. – 112с.
26. Семененко, М.Г.Математическое моделирование в MathCAD. – М: Альтекс-А, 2003. – 206с.
27. Советов, Б.Я., Яковлев, С.А. Моделирование систем: практикум: учеб. пособие. – М.: Высш. шк., 2005. – 295с.
28. Зайцев, Д. А. Математические модели дискретных систем: учеб. пособие. – Одесса: ОНАС им. А.С. Попова, 2004. – 40 с.
29. Питерсон, Дж. Теория сетей Петри и моделирование систем. – М. Мир, 1984. – 264 с.
30. Котов, В.Э. Сети Петри. – М.: Наука, 1984. – 160 с.
31. Ачасова , С.М., Бандман , О.Л. Корректность параллельных вычислительных процессов. – М.: Наука, 1990. – 253 с.
32. Электронный сайт лаборатории "Суперкомпьютерных и распределенных вычислительных технологий" ИАПУ ДВО РАН «Сети Петри». http://www.iacp.dvo.ru/ lab_11/otchet/ ot2000/ pn3doc. html

Список сокращений

ERD – Entry-Relationship Diagrams – диаграммы «сущность-связь»
CAD – Computer Aided Design
CASE – Computer Aided Software Engineering – программная инженерия с компьютерной поддержкой
DFD – Data Flow Diagrams
GPSS – General Purpose Simulation System – общецелевая система моделирования
FRE – Forward and reverse engineering
IEC – International Electro technical Commission – Международная комиссия по электротехнике
ISO – International Organization of Standardization – Международная организация по стандартизации
MATLAB – Matrix Laboratory
RAD – Rapid Application Development
UML - Unified Modeling Language – универсальный язык моделирования
SADT – Structured Analysis and Design Technique
SWEBOK – Software Engineering of Body Knowledge
STD – State Transition Diagrams - диаграмма переходов состояний
SCM – Software Configuration Management – управление конфигурацией ПО
VRML – Virtual Reality Modeling Language
XP – eXtreme Programming
АС - автоматизированная система
АСУ – автоматизированные системы управления
ГОСТ – государственный стандарт
ДИМ – дискретные имитационные модели
ЕСПД – единая система программной документации
ЖЦ – жизненный цикл
ИС – информационные системы
ИССАМ – инструментальные средства и среды автоматизации моделирования
МИКС – моделирование имитационное комбинированных систем
НДИМ – непрерывно-дискретные имитационные модели
ПК – персональный компьютер
ПО – программное обеспечение
РД – рабочий документ
САПР – системы автоматического проектирования
СИМ – системы имитационного моделирования
СКМ – системы компьютерной математики
СМО – системы массового обслуживания
ТЗ – техническое задание
ЯИМ – языки имитационного моделирования
ЯОМ – языки общего назначения

Глоссарий

DERIVE – система компьютерной математики начального уровня.
GPSS — это интерпретирующая языковая система, применяющаяся, в основном, для имитации пространственно-временного движения объектов различной природы при фиксированной структуре блочной схемы.
АВТОМАТ (от греч. "самодействующий") — устройство, самостоятельно выполняющее некий процесс по заложенной в него программе. Программа может фиксироваться либо непосредственно в устройстве автомата, либо на вводимом в автомат носителе.
АВТОМАТИЗАЦИЯ — внедрение автоматов в практическую деятельность (например, автоматизация управления, нефтедобычи, медицинской диагностики, погрузочных работ и т.п.).
АВТОМАТИЗИРОВАННАЯ ОБУЧАЮЩАЯ СИСТЕМА (АОС) - взаимосвязанный комплекс технического, учебно-методического, лингвистического, программного и организационного обеспечения на базе ЭВМ, предназначенный для индивидуализации обучения.
АГРЕГАТ(от лат. "присоединять") — любая выделенная совокупность, от неструктурированной (множество, конгломерат) до высокоорганизованной системы.
АГРЕГИРОВАНИЕ — 1) операция образования агрегата; 2) преобразование многомерной модели в модель меньшей размерности.
АЛГОРИТМ — (от имени узбекского математика IX в. Аль Хорезми) — 1) полное описание последовательности действий, выполнение которых в конце последовательности приводит к достижению цели; 2) конечный текст, записанный на алгоритмическом языке. Первоначально сугубо математическое понятие алгоритма в настоящее время расширено: допускается включение в алгоритм и указаний на неформализуемые действия, лишь бы они правильно понимать, и выполнялись людьми (например, алгоритм изобретения). Алгоритм, воспринимаемый и исполняемый автоматом, называется программой.
АЛГОРИТМИЗАЦИЯ — 1) составление алгоритма для проектируемого процесса (например, алгоритмизация решения задачи); 2) выявление алгоритма, формализация существующего процесса (обычно в целях его изучения, совершенствования или автоматизации).
Алгоритмическая модель — модель, которая описана некоторым алгоритмом или комплексом алгоритмов, определяющим ее функционирование, развитие. Введение такого, на первый взгляд, непривычного типа моделей (действительно, кажется, что любая модель может быть представлена алгоритмом её исследования), на наш взгляд, вполне обосновано, так как не все модели могут быть исследованы или реализованы алгоритмически.
АЛЬТЕРНАТИВА — вариант, одна из двух или более возможностей; то, что можно иметь, использовать и т.д. вместо чего-то еще. На множестве альтернатив осуществляется выбор.

АНАЛИЗ – (от греч. "расчленение") — 1) мысленное или реальное разделение целого на части (например, химический анализ вещества, декомпозиция глобальной цели и т.д.); 2) до недавнего времени — синоним научного исследования вообще ("подвергнуть анализу" означало "изучить") 3) метод познания, основанный на 1). Познание не сводится к анализу; только в сочетании, переплетении, единстве с синтезом становится возможным познание реальности.

Анализ требований – отображения функций системы и ее ограничений в модели предметной области.

АНИМАЦИЯ – представление изображения в виде отдельных кадров, у которых изображения несколько разнятся. Достаточно быстрая смена кадров ведет к появлению эффектов динамики.

АНИМАЦИЯ КОМПЬЮТЕРНАЯ - динамичная графика, основанная на применении различных динамических визуальных эффектов (движущиеся картинки, выделение цветом, шрифтом отдельных элементов схем/таблиц ит.п.); синтез динамических изображений, создающий иллюзию движения на экране дисплея.

АРТЕФАКТ – любой продукт деятельности специалистов по разработке ПО.

АРХИТЕКТУРА ПРОГРАММНОЙ СИСТЕМЫ – определение системы в терминах вычислительных подсистем и интерфейсов между ними, отображающая правила декомпозиции проблемы.

БАЗА ДАННЫХ — 1)достаточно большие наборы структурированных данных некоторой предметной области, представленные на машинных носителях и имеющие общую и удобную структуру, единые организационно - методические, программно-технические и языковые средства обеспечения использования данных различными программами пользователей 2) накопление, структурирование и хранение с помощью ЭВМ знаний, сведений из различных областей таким организованным способом, что можно иметь доступ к этим знаниям, расширять их, получать, выводить новые знания и т.д. 3) совокупность правил и фактов, описывающих предметную область и вместе с механизмом вывода позволяющая отвечать на вопросы об этой предметной области, ответ на которые в явном виде не присутствует в базе.

БАЗА ДАННЫХ – упорядоченная совокупность данных, предназначенных для хранения, накопления и обработки с помощью ЭВМ. Для создания и ведения базы данных (обновления, обеспечения доступа к ним по запросам и выдачи их пользователю) используется набор языковых и программных средств, называемый системой управления базы данных (СУБД). База данных в сочетании с СУБД представляет собой банк данных.

БАНК ДАННЫХ – совокупность базы данных, а также программных, языковых и других средств, предназначенных для централизованного накопления данных и их использования с помощью ЭВМ.

БЛОК-ДИАГРАММА — графическое представление операций, происходящих внутри системы.

БУЛЕВСКИЕ ПЕРЕМЕННЫЕ – переменные, которые позволяют пользователю проверять в одном блоке GPSS/PC одновременно несколько условий,

исходя из состояния или значения объектов и их атрибутов, т. е. в данном блоке проводится обращение к булевской переменной, выражение которой содержит в себе проверку нескольких условий. Булевская переменная имеет значение 1, если выражение переменной истинно, и 0, если выражение переменной ложно.

ВАЛИДАЦИЯ – проверка соответствия разработки программной системы требованиям заказчика.

ВЕРИФИКАЦИЯ – проверка истинности теоретических положений путем сопоставления с чувственными данными.

ВЕРИФИКАЦИЯ – проверка правильности реализации системы заданным требованиям на каждом этапе жизненного цикла.

ВИЗУАЛЬНАЯ МОДЕЛЬ — модель, которая позволяет визуализировать отношения и связи моделируемой системы, особенно в динамике.

ВИРТУАЛИЗАЦИЯ – замещение реальности ее имитацией/образами.

ВХОД (СИСТЕМЫ) — 1) связь системы с окружающей средой, направленная от среды в систему, т.е. выражающая воздействия из среды на систему; 2) то, что преобразуется системой в выход.

ВЫБОР — 1) операция, входящая во всякую целенаправленную деятельность и состоящая в целевом сужении множества альтернатив (обычно, если позволяют условия,— до одной альтернативы); 2) принятие решения.

ВЬ1ХОД (СИСТЕМЫ) — 1) связь системы с окружающей средой, выражающая воздействие системы на среду и направленная от системы к среде; 2) продукт системы; то, во что преобразуются входы; может иметь как реальный характер (например, материальная продукция), так и абстрактный (например, удовлетворение потребности).

ВЬЮВЕР — средство задания и просмотра диаграмм моделей и результатов моделирования.

ГИПЕРМЕДИА – вид учебных материалов, представленных в электронной форме как гипертекст с мультимедиа дополнениями.

ГИПЕРССЫЛКА – ссылка из одного электронного информационного объекта к другому (например, из текста к примечанию или элементу списка литературы, из одной энциклопедической статьи к другой).

ГИПЕРТЕКСТ – способ представления информации с помощью связей между документами (гиперссылок); вид учебных материалов, представленных в электронной форме как интерактивные тексты, связанные гиперссылками.

ГИПОТЕЗА — 1) предположение; утверждение, требующее доказательства или проверки; 2) форма развития науки.

ГРАФ(от греч. "записывать") — 1) символическая диаграмма, состоящая из множества вершин и ребер (дуг), соединяющих некоторые из них (или все); 2) графическая модель структуры.

ДАННЫЕ — это факты и (или) понятия, описанные в формализованном виде. В ИВС различают пользовательские (информационные) и управляющие (служебные) данные.

ДЕКОМПОЗИЦИЯ — 1) операция разделения целого на части с сохранением признака подчиненности, принадлежности; 2) повторное или многократ-

ное такое разделение, в результате чего получаются древовидные иерархические структуры.
ДЕЛОВАЯ ИГРА – метод принятия управленческих решений в различных имитируемых производственных ситуациях путем игры группы студентов (или одного студента) с ЭВМ по заданным правилам в диалогом режиме. Это активный метод обучения, направленный на формирование у обучаемых самостоятельного мышления.
ДЕРЕВО БЛОК-СХЕМЫ – иерархическое представление модели.
ДЕТЕРМИНИРОВАННАЯ МОДЕЛЬ — модель, которая каждому входному набору параметров соответствует вполне определенный и однозначно определяемый набор выходных параметров; в противном случае - модель недетерминированная, стохастическая (вероятностная).
ДИАГРАММА – графическое представление моделирования системы с помощью классов, сценариев, состояний и т.п.
ДИАЛОГ – форма двусторонней связи, когда стороны строго чередуют свои сообщения, являясь попеременно источниками и приемниками информации.
ДИНАМИЧЕСКАЯ МОДЕЛЬ — модель, в которой, среди ее параметров есть временной параметр, т.е. она отображает систему (процессы в системе) во времени.
ДИНАМИЧЕСКОЕ ТЕСТИРОВАНИЕ – выполнение программ для обнаружения ошибок, установления их причины и устранения.
ДИСКРЕТНАЯ МОДЕЛЬ — описывает поведение системы только в дискретные моменты времени.
ЖИЗНЕННЫЙ ЦИКЛ системы (ЖЦ) – непрерывный процесс, который начинается с момента принятия решения о необходимости ее создания и заканчивается в момент ее полного изъятия из эксплуатации.
ЗАГОЛОВОК – параметры, перемещаемые в начало программы или данных, которые важны для их использования.
ЗНАК - 1) сигнал, имеющий конкретное (обычно одиночное, элементарное) значение, воспринимаемое человеком (например, дорожный знак, знаки отличия, математические знаки, условные жесты и т.д.); 2) реальная модель абстрактного понятия.
ИГРОВАЯ МОДЕЛЬ — описывает, реализует некоторую игровую ситуацию между участниками игры (лицами, коалициями).
ИЕРАРХИЧЕСКАЯ МОДЕЛЬ (древовидная) — модель, которая представима некоторой иерархической структурой (деревом).
ИМИТАЦИОННАЯ МОДЕЛЬ — модель, которая предназначена для испытания или изучения возможных путей развития и поведения объекта путем варьирования некоторых или всех параметров модели.
Имитационное моделирование – моделирование поведения системы в различных аспектах и в разных внешних и внутренних условиях с анализом динамических характеристики анализом распределения ресурсов
ИНСТРУМЕНТАЛЬНАЯ МОДЕЛЬ — средство построения, исследования и/или использования прагматических и/или познавательных моделей.

ИНТЕРАКТИВНАЯ ПРОГРАММА – интерактивный программный модуль (тестирование, моделирование, имитация).
ИНТЕРАКТИВНОСТЬ – реакция со стороны программы в ответ на какие-либо действия пользователя, обеспечивающая режим диалога с ЭВМ.
ИНТЕРПРЕТАТОР GPSS/PC — автоматически собирает стандартную статистику по каждому типу объектов, занятых в модели. В зависимости от того, какие объекты и как используются в модели, пользователь имеет возможность получать дополнительную статистику как в процессе счета модели, так и по окончании счета.
ИНТЕРФЕЙС – набор средств и сервисных услуг для управления функционированием ЭВМ, освобождающих пользователя от выполнения рутинных операций.
ИНФОРМАТИКА — 1) наука о научной и технической информации и ее циркуляции в обществе; 2) информатикой называют научное направление, акцентирующее внимание на использовании ЭВМ в самых разнообразных областях человеческой деятельности.
ИНФОРМАЦИОННАЯ СИСТЕМА – система передачи и приема информации. Состоит из источника информации, передатчика, канала связи, приемника информации и источника помех.

ИНФОРМАЦИОННАЯ СИСТЕМА – система, которая проводит сбор, обработку, сохранение и производство информации с использованием автоматических процессоров и людей.

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ (ИТ) – система методов, производственных процессов и программно-технических средств, интегрированных с целью сбора, обработки, хранения, распространения, отображения и использования информации пользователей этой информации. В состав ИТ входят аппаратные, программные и информационные компоненты.
ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ (ИКТ) — обобщающее понятие, описывающее различные устройства, механизмы, способы, алгоритмы обработки информации. Важнейшим современным устройствами ИКТ являются компьютер, снабженный соответствующим программным обеспечением и средства телекоммуникаций вместе с размещенной на них информацией.
ИНФОРМАЦИЯ — 1) в обыденной речи — любые сведения, известия, сообщения, новости и т.п.; 2) в научно-технических приложениях — то, что несет на себе сигнал; 3) как философская категория — всеобщее свойство материи, являющееся аспектом свойства отражения, допускающим количественное описание.
ИНФОРМАЦИЯ (от латинского informatio -разъяснение, изложение) - совокупность сведений, данных, передаваемых людьми устно (в форме речи), письменно (в виде текста, таблицы, рисунка, чертежа, условных знаков, обозначений) либо другим способом (например, с помощью звуковых или световых сигналов, электрических или нервных импульсов). С середины XX ве-

ка -общенаучное понятие, включающее обмен сведений между людьми, человеком и автоматом, автоматом и автоматом, обмен сигналами в животном и растительном мире.

КИБЕРНЕТИКА (от греч. "управлять") — в широком смысле наука об управлении в системах произвольной природы; наиболее полными и общими определениями кибернетики в современном понимании признаются определения А.И. Берга и А.Н. Колмогорова.

КЛАССИФИКАЦИЯ — 1) операция отнесения заданного объекта к одному из классов, внутри которых объекты считаются неразличимыми; результат этой операции; 2) простейший вид моделирования; в частности, самый слабый вид измерения.

КЛЕТОЧНО-АВТОМАТНАЯ МОДЕЛЬ — представляет систему с помощью клеточного автомата или системы клеточных автоматов.

КЛЕТОЧНЫЙ АВТОМАТ — дискретная динамическая система, аналог физического (непрерывного) поля.

КОД — совокупность условий и правил образования сигнала, использование которых на передающем и на приемном концах позволяет передавать и получать информацию с помощью сигнала.

КОЛИЧЕСТВО ИНФОРМАЦИИ — числовая мера информации, содержащейся в одном случайном объекте о другом случайном объекте. Определяется как некий функционал от соответствующих распределений вероятностей (например, по Шэннону, по Фишеру, по Кульбаку-Лейблеру и др.), либо как объем вычислений, необходимых для алгоритмического определения состояния объекта (по Колмогорову).

КОМАНДА – информация, вводимая пользователем в компьютер и служащая сигналом к выполнению определенных действий.

КОМПЬЮТЕРНАЯ МАТЕМАТИКА – это совокупность теоретических, алгоритмических, аппаратных и программных средств, предназначенных для эффективного решения на компьютерах всех видов математических задач с высокой степенью визуализации всех этапов вычислений.

КОМПЬЮТЕРНАЯ МОДЕЛЬ – учебное издание, основанное на математических моделях; может быть использована не только для демонстрации трудно воспроизводимых в учебной обстановке явлений, но и для выяснения (в диалоговом режиме) влияния тех или иных параметров на изучаемые процессы и явления.

КОМПЬЮТЕРНАЯ ТЕСТИРУЮЩАЯ СИСТЕМА – учебное издание, которое обеспечивает, с одной стороны, возможность самоконтроля для обучаемого, а с другой - принимает на себя рутинную часть текущего или итогового контроля. Компьютерная тестирующая система может представлять собой как отдельную программу, не допускающую модификации, так и универсальную программную оболочку, наполнение которой возлагается на преподавателя.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ – это метод решения задачи анализа или синтеза сложной системы на основе использования ее компьютерной модели.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ (Computer Simulation) – типичная область знания, обладающая специфической "постмодернистской чувственностью" - особым, плюралистичным отношением к миру, избегающ им излишнего обобщения и диктата тоталитаризирующих истин.
КРИТЕРИЙ — 1) средство для вынесения суждения; стандарт для сравнения; правило для оценки; 2) мера степени близости к цели; в атом смысле — модель цели.
ЛОГИЧЕСКАЯ МОДЕЛЬ — модель, которая представима предикатами, логическими функциями.
МАКРОМОДЕЛИРОВАНИЕ – моделирование тех или иных устройств и систем с помощью макромодели.
МАКРОМОДЕЛЬ – модель, сведенная к простой структурной схеме из ряда достаточно сложных моделей.
МАТЕМАТИЧЕСКАЯ МОДЕЛЬ — это упрощенное описание реальности с помощью математических понятий. Существует два основных класса задач, связанных с математическими моделями: прямые и обратные. В первом случае все параметры модели считаются известными, и нам остается только исследовать её поведение. А во втором какие-то параметры модели неизвестны (например, не могут быть измерены явно), и требуется их найти, сопоставляя поведение реальной системы с её моделью.
МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ — процесс построения и изучения математических моделей реальных процессов и явлений.
МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ (в социологии) словосочетание, обозначающее использование математического языка и аппарата для описания и последующего анализа основных свойств социальных явлений и процессов.
МЕНЮ – в интерактивных системах список команд или вариантов ответа на экране дисплея, из которых пользователь выбирает необходимый ему вариант, вводя соответствующий номер или букву, либо указывая на пункт меню курсором.
МОДЕЛИРОВАНИЕ – метод исследования объектов познания на их моделях; построение моделей реально существующих предметов и явлений и конструируемых объектов для определения либо улучшения их характеристик, рационализации способов их построения, управления ими и т.п.
МОДЕЛИРОВАНИЕ — универсальный метод получения, описания и использования знаний. Он используется в любой профессиональной деятельности. В современной науке и технологии роль и значение моделирования усиливается, актуализируется проблемами, успехами других наук.
МОДЕЛИРОВАНИЕ – это изучение объектов познания с помощью их моделей.
МОДЕЛЬ — объект или описание объекта, системы для замещения (при определенных условиях предложениях, гипотезах) одной системы (т.е. оригинала) другой системой для лучшего изучения оригинала или воспроизведения каких-либо его свойств.

МОДЕЛЬ (от лат. "образец") — отображение: целевое; абстрактное или реальное, статическое или динамическое; ингерентное; конечное, упрощенное, приближенное; имеющее наряду с безусловно истинным условно истинное, предположительно истинное и ложное содержание; реализующееся и развивающееся в процессе его практического использования.
МОДЕЛЬ (от лат. modulus – мера, образец) – упрощенное представление явлений или объектов действительности, относящихся к природе и обществу, в виде схем, изображений, описаний, математических формул, какого-либо реального предмета (явления или процесса), изучаемое как их аналог.
МОДЕЛЬ АБСТРАКТНАЯ — идеальная конструкция; модель, построенная средствами мышления, сознания (в частности — языковая модель).
МОДЕЛЬ ДИНАМИЧЕСКАЯ — модель, отображающая процессы, происходящие в системе со временем; в частности, модели функционирования и развития.
МОДЕЛЬ ЗНАКОВАЯ — реальная модель, имеющая абстрактное содержание; модель, условно подобная оригиналу и предназначенная для непосредственного использования человеком.
МОДЕЛЬ ИНГЕРЕНТНАЯ — модель, согласованная с окружающей культурной средой, входящая в нее не как чуждый ей элемент, а как ее естественная часть.
МОДЕЛЬ КЛАССИФИКАЦИОННАЯ — простейший вид модели, в которой фиксируются только отношения тождественности или различия».
МОДЕЛЬ МАТЕМАТИЧЕСКАЯ — абстрактная или знаковая модель, построенная средствами математики (например, в виде системы уравнений, графа, логической формулы и т.п.)
МОДЕЛЬ МОДЕЛЕЙ - иерархия моделей; многоуровневая абстракция; число уровней в иерархии моделей предположительно связывается с развитостью интеллекта.
МОДЕЛЬ ПОЗНАВАТЕЛЬНАЯ – форма организации и представления знаний; средство соединения новых знаний с имеющимися.
МОДЕЛЬ ПРАГМАТИЧЕСКАЯ – средство управления, организации практических действий; образец, эталон правильных действий (например, алгоритм) или их результата (например, модель цели).
МОДЕЛЬ РЕАЛЬНАЯ (ВЕЩЕСТВЕННАЯ, ФИЗИЧЕСКАЯ, ПРЕДМЕТНАЯ) – модель, построенная из реальных объектов; подобие реальной модели и оригинала может быть прямым, косвенным и условным.
МОДЕЛЬ СОСТАВА СИСТЕМЫ – модель, описывающая, из каких подсистем и элементов состоит система.
МОДЕЛЬ СТАТИЧЕСКАЯ – модель, в которой отсутствует временной параметр.
МОДЕЛЬ СТРУКТУРЫ СИСТЕМЫ— модель, описывающая все отношения (связи) между элементами модели состава системы.
МОДЕЛЬ ФУНКЦИОНАЛЬНАЯ – модель, описывающая процессы, которые характеризуют систему как часть более общей, охватывающей ее системы, т.е. связаны с назначением данной системы.

МОДЕЛЬ "ЧЕРНОГО ЯЩИКА" – модель, описывающая только входы и выходы системы, но не внутреннее устройство системы. Например, математическая модель "черной, ящика" — это просто совокупность множеств X и Y (X соответствует входам, Y — выходам); если оператор F, связывающий их (Y = F(X)), и предполагается существующим, то он считается неизвестным.
МОДЕЛЬ ЯЗЫКОВАЯ – любая конструкция на естественном языке, рассматриваемая как описание чего-либо (например, определение как модель определяемого; имя, название как обозначение называемого и т.д.).
МОЗГОВОЙ ШТУРМ – метод, предназначенный для неформального коллективного генерирования возможно большего числа альтернатив; основные идеи этого метода: а) полное запрещение критики на стадии генерирования; б) поощрение и провоцирование ассоциативного мышления на всех стадиях; в) на стадии оценки цель состоит не в отбрасывании "плохой" альтернативы, а в поиске рационального зерна в ней.
МОРФОЛОГИЧЕСКИЙ АНАЛИЗ – формальный метод генерирования альтернатив с помощью перечисления всех возможных сочетаний значений заданных параметров альтернативы.
НЕОПРЕДЕЛЕННОСТЬ СТОХАСТИЧЕСКАЯ – неопределенность, описываемая распределением вероятностей на множестве возможных состояний рассматриваемого объекта; случайность.
НЕТРАНЗИТИВНОСТЬ ГОЛОСОВАНИЯ – одно из свойств коллективного выбора, состоящее в существовании набора альтернатив, на котором выбор единственной альтернативы методом голосования сделан быть не может; в частности, результат выбора может зависеть от последовательности предъявления альтернатив.
ОПТИМАЛЬНЫЙ – наилучший в заданных условиях. Качество оценивается с помощью критерия оптимальности, а условия задаются в виде ограничений на дополнительные критерии. Оптимизация — центральная идея кибернетики.
ПРИНЯТИЕ РЕШЕНИЯ – целевой выбор на множестве альтернатив. Методы принятия решений разнообразны в зависимости от типа неопределенности и других условий выбора.
ПРОБЛЕМА(от греч. "задача") — 1) проблема развития — неудовлетворительное состояние системы, изменение которого к лучшему является непростым делом; 2) проблема функционирования — удовлетворительное состояние системы, сохранение которого требует постоянных и непростых усилий (например, проблема выживания)
ПРОБЛЕМАТИКА – сплетение, клубок проблем, которые неразрывно связаны с проблемой, подлежащей разрешению. Необходимость рассмотрения проблематики вместо отдельной проблемы вытекает из того, что проблемосодержащая система сама состоит из подсистем и входит в надсистему, а устранение поставленной проблемы требует учета: последствий для всех них.

СВОЙСТВО – 1) качество, постоянно присущее объекту; 2) абстракция отношения данного объекта с другими, "свернутое отношение", модель отношения.
СЕМАНТИКА (от греч. "обозначение") — раздел семиотики, изучающий отношения между знаками и тем, что они обозначают.
СЕМИОТИКА(от греч. "знак") — наука, исследующая знаки и знаковые системы.
СИГНАЛ (от лат. "знаки') – материальный носитель информации.
МОДЕЛЬ ПРОЦЕССОВ – определенная последовательность действий, сопровождающая изменение состояния программного объектов.
МОДЕЛЬ СОСТОЯНИЙ – отображения динамики изменения состояния объекта класса, которое изменяют его поведение.
МУЛЬТИМЕДИА (мультимедиа средства) – компьютерные средства создания, хранения, обработки и воспроизведения в оцифрованном виде информации разных типов: текста, рисунков, схем, таблиц, диаграмм, фотографий, видео- и аудио- фрагментов и т.п.
МУЛЬТИМЕДИА КУРС – это комплекс логически связанных структурированных дидактических единиц, представленных в цифровой и аналоговой форме, содержащий все компоненты учебного процесса. Мультимедиа курс является средством комплексного воздействия на обучающегося путем сочетания концептуальной, иллюстративной, справочной, тренажерной и контролирующей частей.
НАТУРНАЯ МОДЕЛЬ — есть материальная копия объекта моделирования.
НЕПРЕРЫВНАЯ МОДЕЛЬ — описывает поведение системы для всех моментов времени из некоторого промежутка времени.
ОБЪЕКТНО–ОРИЕНТИРОВАННАЯ МОДЕЛЬ – структура из совокупности объектов, которые взаимодействуют между собою, обладают свойствами и поведением.
ОНТОЛОГИЯ – совокупность элементарных понятий, терминологии и парадигмы их интерпретации в среде проблемы, которую требуется разработать
ОТЛАДКА – проверка программы на наличие в ней ошибок и их устранение без внесения новых.
ПЕРЕХОДНАЯ ХАРАКТЕРИСТИКА ЗВЕНЬЕВ – позволяет и качественно, и количественно характеризовать быстродействие звеньев и систем. ПЕРЕХОДНЫЙ ПРОЦЕСС может быть как монотонным, так и колебательным и его длительность и является количественной характеристикой быстроты реакции звена на прикладываемые к нему воздействия.
ПЕЧАТНЫЕ МАТЕРИАЛЫ – вид учебных материалов, представленных в полиграфическом исполнении (книги, учебники, методические пособия).
ПОЗНАВАТЕЛЬНАЯ МОДЕЛЬ — форма организации и представления знаний, средство соединения новых и старых знаний. Познавательная модель, как правило, подгоняется под реальность и является теоретической моделью.
ПОСТРОЕНИЕ МОДЕЛИ — системная задача, требующая анализа и синтеза исходных данных, гипотез, теорий, знаний специалистов. Системный под-

ход позволяет не только построить модель реальной системы, но и использовать эту модель для оценки (например, эффективности управления, функционирования) системы.
ПРАГМАТИЧЕСКАЯ МОДЕЛЬ — средство организации практических действий, рабочего представления целей системы для ее управления. Реальность в них подгоняется под некоторую прагматическую модель. Это, как правило, прикладные модели.
ПРОГРАММНЫЙ КОД – это компьютерная программа, написанная на определенном языке программирования по алгоритму, заданному педагогическим и технологическим сценарием.
СЕТЕВАЯ МОДЕЛЬ — модель, которая представима некоторой сетевой структурой.
СЕТЕВЫЕ УЧЕБНЫЕ МАТЕРИАЛЫ – это учебные материалы, которые включают сетевые версии мультимедиа курсов, материалы, имеющиеся в Интернет, эксперименты с удаленным доступом и т.п.
СЕТЬ — совокупность средств передачи и распределения данных.
СИНЕКТИКА – метод генерирования альтернатив, основанный на догадках по ассоциации, возникающих в группе экспертов, специально подготовленных для поиска аналогий, в особенности аналогий двигательным ощущениям.
СИНТЕЗ(от греч. "соединение") — 1) мысленное или реальное соединение частей в единое целое; 2) метод познания, основанный на 1) Познание является единением, сочетанием анализа и синтеза.
СИСТЕМА – средство достижения цели; основные особенности систем: целостность, относительная обособленность от окружающей среды, наличие связей со средой, наличие частей и связей между ними (структурированность), подчиненность всей организации системы некоторой цели.
СИСТЕМА - это совокупность элементов, находящихся в определенных отношениях друг с другом и со средой.
СИСТЕМА БОЛЬШАЯ — система, для актуализации модели которой в целях управления недостает материальных ресурсов (времени, емкости памяти, других материальных средств моделирования).
СИСТЕМА ЕСТЕСТВЕННАЯ — система (т.е. многокомпонентный объект, обладающий всеми признаками системы), возникшая в природе в результате естественных процессов.
СИСТЕМА ИСКУССТВЕННАЯ — система, созданная человеком как средство достижения поставленной цели.
СИСТЕМА ПРОБЛЕМОРАЗРЕШАЮЩАЯ - система, обладающая возможностями (ресурсами, компетенцией и пр.), необходимыми для ликвидации проблемы; обязательный участник системного анализа; проблеморазрешающая и проблемосодержащая системы могут не совпадать.
СИСТЕМА ПРОБЛЕМОСОДЕРЖАЩАЯ – система, в которой возникла проблема, подлежащая решению; обычно эта система является инициатором и заказчиком проведения системного анализа.

СИСТЕМА СЛОЖНАЯ — система, модель которой, используемая для управления системой, не адекватна заданной цели.
СИСТЕМА СОЦИОТЕХНИЧЕСКАЯ — система, в составе которой имеются люди и коллективы, интересы которых существенно связаны с функционированием системы.
СИСТЕМА ЦЕННОСТЕЙ — идеологическая основа для постановки целей социотехнических систем; объект системного анализа на этапе выявления действительных целей лиц, причастных к решаемой проблеме.
СИСТЕМНОСТЬ — 1) обладание всеми признаками системы; 2) всеобщее свойство материи, форма ее существования, а, следовательно, неотъемлемое свойство человеческой практики, включая мышление.
СИСТЕМНЫЙ АНАЛИЗ – 1) с практической стороны системный анализ есть система методов исследования или проектирования сложных систем, поиска, планирования и реализации изменений, предназначенных для ликвидации проблем; 2) с методологической стороны системный анализ является прикладной диалектикой, так как реализует идеи материалистической диалектики применительно к конкретным практическим задачам, особенность которых состоит в необходимости выяснения причин их сложности и устранения этих причин; 3) с методической стороны системный анализ отличается междисциплинарным и наддисциплинарным характером и вовлечением в работу как неформальных, эвристических, экспертных методов, так и эмпирических, экспериментальных методов, а также при возможности и необходимости — строгих формальных математических методов.
СИСТЕМНЫЙ ПОДХОД – в настоящее время рассматривается либо как одна из ранних форм системного анализа, либо как начальная фаза современного системного анализа, этап первоначального качественного анализа проблемы и постановки задач.
СИСТЕМЫ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ – новые средства, автоматизирующие выполнение как численных, так и аналитических вычислений.
СЛОВАРЬ - справочное издание, содержащее упорядоченный перечень языковых единиц (слов, словосочетаний, фраз, терминов, имен, знаков), снабженных относящимися к ним справочными данными.
СЛОЖНОСТЬ – свойство некоторого явления (объекта, процесса, системы), выражающееся в неожиданности, непредсказуемости; необъяснимости, случайности, "антиинтуитивности" его поведения.
СОЦИАЛЬНАЯ СИСТЕМА – это способ организации жизни коллектива людей, который возникает в результате взаимодействия (социальных) действий индивидов на базе диктуемых социальных ролей.
СОЦИАЛЬНЫЙ ИНСТИТУТ – организованная система связей и социальных норм, которая объединяет значимые общественные ценности и процедуры, удовлетворяющие основным потребностям общества.
СТАНДАРТНЫЕ ЧИСЛОВЫЕ АТРИБУТЫ (СЧА) — специальные числовые характеристики, используемые в GPSS/PC

СТАТИЧЕСКАЯ МОДЕЛЬ — модель, которая в описании параметров, нет временного параметра. Статическая модель в каждый момент времени дает лишь "фотографию" системы, ее срез.
СТРУКТУРА(от лат. "строение") — совокупность связей между частями системы.
СТРУКТУРНАЯ МОДЕЛЬ — модель, которая представима структурой данных или структурами данных и отношениями между ними.
СТРУКТУРНАЯ СХЕМА СИСТЕМЫ – конструкция системы; объединение моделей "черного ящика", состава и структуры.
СУБД (DBMS - DataBase Management System) — программная система, обеспечивающая общение (интерфейс) программ пользователя и данных из БД.
СЦЕНАРИЙ – воображаемая, но правдоподобная последовательность действий и вытекающих из них событий, которые могут произойти в будущем с исследуемой системой; модель будущего после принятия решения, представленная до его принятия.
ТИПОВЫЕ ЗВЕНЬЯ – простые модели элементов сложных линейных систем и даже систем в целом.
ТРАНЗАКТ — это некоторое сообщение (заявка, требование на обслуживание), которое поступает извне на вход системы и подлежит обработке.
ТРЕНАЖЕРНЫЙ КОМПЛЕКС – электронное учебное издание, которое дает обучающемуся возможность самостоятельно отработать навыки, заданные теоретическим материалом, обнаружить слабые места в усвоении курса. Тренажерный комплекс, как правило, представляет собой серию вопросов, задач, практических заданий, предполагающих типовые ответы.
УЧЕБНИК – учебное издание, содержащее систематическое изложение учебной дисциплины (ее раздела, части), соответствующее учебной программе и официально утвержденное в качестве данного вида издания.
ФАЗОВОЕ ПРОСТРАНСТВО – это система координат, каждая из которых соответствует разным величинам интенсивности одной "характеристики".
ФАКТОРИЗАЦИЯ – это разложение на множители выделенных выражений или подвыражений относительно одной или ряда переменных.
ФЛУКТУАЦИЯ – случайное отклонение величины, характеризующей систему из большого числа частиц, от среднего значения.
ФРАКТАЛЬНАЯ МОДЕЛЬ — модель, которая описывает эволюцию моделируемой системы эволюцией фрактальных объектов. Если физический объект однородный (сплошной), т.е. в нем нет полостей, можно считать, что плотность не зависит от размера.
ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ — если она представима в виде системы каких- либо функциональных соотношений.
ФУНКЦИЯ ВЫБОРА – наиболее общая математическая модель выбора; отображение совокупности множеств в совокупность их подмножеств без поэлементного отображения одного множества на другое и без отображения множеств на числовую ось.

ФУНКЦИЯ КРИТЕРИАЛЬНАЯ – функция от обозначения альтернатив, значения которой упорядочены в том же порядке, что и предпочтения альтернатив.
ФУНКЦИЯ ПОТЕРЬ – функция, выражающая потери, которые вынужден нести пользователь статистического решения, отличающегося от истинного суждения.
ФУНКЦИЯ ПРИНАДЛЕЖНОСТИ – функция, характеризующая расплывчатое множество и принимающая для каждой альтернативы значение из интервала [0,1], выражающее степень принадлежности данного элемента этому расплывчатому множеству.
ФУНКЦИЯ РЕШАЮЩАЯ – функция, отображающая каждую выборку в пространство статистических решений.
ШКАЛЫ ИЗМЕРИТЕЛЬНЫЕ — множество обозначений, используемых для регистрации состояния наблюдаемого объекта; в зависимости от введенных отношений на этом множестве, шкалы различаются по их силе; сила измерительной шкалы должна согласовываться с природой наблюдаемого явления.
ЭКСПЕРТНЫЕ МЕТОДЫ — методы системного анализа, в которых для выполнения тех или иных неформализуемых операций используются знания, опыт, интуиция, изобретательность, интеллект экспертов, специалистов в нужной области.
ЭМЕРДЖЕНТНОСТЬ(от англ. "внезапное появление") — 1) особенность систем, состоящая в том, что свойства системы не сводятся к совокупности свойств частей, из которых она состоит, и не выводятся из них; 2) внутренняя целостность систем.
ЭНТРОПИЯ (от греч. "превращение") — мера неопределенности случайного объекта.
ЭТНИЧЕСКОЕ ПОЛЕ – это поле, обеспечивающее взаимодействие членов этноса и регулирующее их совместную целенаправленную деятельность.
Этнометодология – изучение методов, которыми люди создают социальный порядок.
ЯДРО СИСТЕМЫ – коды множества заранее откомпилированных функций и процедур, обеспечивающих достаточно представительный набор встроенных функций и операторов системы.
ЯЗЫК ПРОГРАММИРОВАНИЯ – язык, предназначенный для записи алгоритмов программ и для их последующего выполнения на компьютере.
ЯЗЫКОВАЯ МОДЕЛЬ — модель, которая представлена некоторым лингвистическим объектом, формализованной языковой системой или структурой. Иногда такие модели называют вербальными, синтаксическими и т.п.
ЯЧЕЙКИ – элементы для сохранения некоторой числовой информации в системе в системе. Бывают ячейки сохраняемых величин и матрицы ячеек сохраняемых величин.

Приложение 1

ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы

ГОСТ 34.602-89

Information technology. Set of standards for automated systems. Technical directions for developing of automated system

Г О С У Д А Р С Т В Е Н Н Ы Й С Т А Н Д А Р Т С О Ю З А С С Р

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы

ГОСТ 34.602-89

Information technology. Set of standards for automated systems. Technical directions for developing of automated system

---

Дата введения 01.01.90

Настоящий стандарт распространяется на автоматизированные системы (АС) для автоматизации различных видов деятельности (управление, проектирование, исследование и т. п.), включая их сочетания, и устанавливает состав, содержание, правила оформления документа «Техническое задание на создание (развитие или модернизацию) системы» (далее - ТЗ на АС).

Рекомендуемый порядок разработки, согласования и утверждения ТЗ на АС приведен в приложении 1.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. ТЗ на АС является основным документом, определяющим требования и порядок создания (развития или модернизации - далее создания) автоматизированной системы, в соответствии с которым проводится разработка АС и ее приемка при вводе в действие.

1.2. ТЗ на АС разрабатывают на систему в целом, предназначенную для работы самостоятельно или в составе другой системы.

Дополнительно могут быть разработаны ТЗ на части АС; на подсистемы АС, комплексы задач АС и т. п. в соответствии с требованиями настоящего стандарта; на комплектующие средства технического обеспечения и программно-технические комплексы в соответствии со стандартами ЕСКД и СРПП; на программные средства в соответствии со стандартами ЕСПД; на информационные изделия в соответствии с ГОСТ 19.201 и НТД, действующей в ведомстве заказчика АС.

Примечание. В ТЗ на АСУ для группы взаимосвязанных объектов следует включать только общие для группы объектов требования. Специфические требования отдельного объекта управления следует отражать в ТЗ на АСУ этого объекта.

1.3. Требования к АС в объеме, установленном настоящим стандартом, могут быть включены в задание на проектирование вновь создаваемого объекта автоматизации. В этом случае ТЗ на АС не разрабатывают.

1.4. Включаемые в ТЗ на АС требования должны соответствовать современному уровню развития науки и техники и не уступать аналогичным требованиям, предъявляемым к лучшим современным отечественным и зарубежным аналогам.

Задаваемые в ТЗ на АС требования не должны ограничивать разработчика системы в поиске и реализации наиболее эффективных технических, технико-экономических и других решений.

1.5. ТЗ на АС разрабатывают на основании исходных данных, в том числе содержащихся в итоговой документации стадии «Исследование и обоснование создания АС», установленной ГОСТ 24.601.

1.6. В ТЗ на АС включают только те требования, которые дополняют требования к системам данного вида (АСУ, САПР, АСНИ и т. д.), содержащиеся в действующих НТД, и определяются спецификой конкретного объекта, для которого создается система.

1.7. Изменения к ТЗ на АС оформляют дополнением или подписанным заказчиком и разработчиком протоколом. Дополнение или указанный протокол являются неотъемлемой частью ТЗ на АС. На титульном листе ТЗ на АС должна быть запись «Действует с … ».

2. СОСТАВ И СОДЕРЖАНИЕ

2.1. ТЗ на АС содержит следующие разделы, которые могут быть разделены на подразделы:

1) общие сведения;

2) .назначение и цели создания (развития) системы;

3) характеристика объектов автоматизации;

4) требования к системе;

5) состав и содержание работ по созданию системы;

6) порядок контроля и приемки системы;

7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;

9) источники разработки.

В ТЗ на АС могут включаться приложения.

2.2. В зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ.

В ТЗ на части системы не включают разделы, дублирующие содержание разделов ТЗ на АС в целом.

2.3. В разделе «Общие сведения» указывают:

1) полное наименование системы и ее условное обозначение;

2) шифр темы или шифр (номер) договора;

3) наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты;

4) перечень документов, на основании которых создается система, кем и когда утверждены эти документы;

5) плановые сроки начала, и окончания работы по созданию системы;

6) сведения об источниках и порядке финансирования работ;

7) порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы.

2.4. Раздел «Назначение и цели создания (развития) системы» состоит из подразделов:

1) назначение системы;

2) цели создания системы.

2.4.1. В подразделе «Назначение системы» указывают вид автоматизируемой деятельности (управление, проектирование и т.п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать.

Для АСУ дополнительно указывают перечень автоматизируемых органов (пунктов) управления и управляемых объектов.

2.4.2. В подразделе «Цели создания системы» приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, которые должны быть достигнуты в результате создания АС, и указывают критерии оценки достижения целей создания системы.

2.5. В разделе «Характеристики объекта автоматизации» приводят:

1) краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию;

2) сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

Примечание. Для САПР в разделе дополнительно приводят основные параметры и характеристики объектов проектирования.

2.6. Раздел «Требования к системе» состоит из следующих подразделов:

1) требования к системе в целом;

2) требования к функциям (задачам), выполняемым системой;

3) требования к видам обеспечения.

Состав требований к системе, включаемых в данный раздел ТЗ на АС, устанавливают в зависимости от вида, назначения, специфических особенностей и условий функционирования конкретной системы. В каждом под-

разделе приводят ссылки на действующие НТД, определяющие требования к системам соответствующего вида.

2.6.1. В подразделе «Требования к системе в целом» указывают:

требования к структуре и функционированию системы; требования к численности и квалификации персонала системы и режиму его работы;

показатели назначения;

требования к надежности;

требования безопасности;

требования к эргономике и технической эстетике;

требования к транспортабельности для подвижных АС;

требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;

требования к защите информации от несанкционированного доступа;

требования по сохранности информации при авариях требования к защите от влияния внешних воздействии;

требования к патентной чистоте;

требования по стандартизации и унификации;

дополнительные требования.

2.6.1.1. В требованиях к структуре и функционированию системы приводят:

1) перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы;

2) требования к способам и средствам связи для информационного обмена между компонентами системы;

3) требования к характеристикам взаимосвязей создаваемой системы со смежными системами, требования к ее совместимости, в том числе указания о способах обмена информацией (автоматически, пересылкой документов, по телефону и т. п.);

4) требования к режимам функционирования системы;

5) требования по диагностированию системы;

6) перспективы развития, модернизации системы.

2.6.1.2. В требованиях к численности и квалификации персонала АС приводят:

требования к численности персонала (пользователей) АС;

требования к квалификации персонала, порядку его подготовки и контроля знаний и навыков;

требуемый режим работы персонала АС.

2.6.1.3. В требованиях к показателям назначения АС приводят значения параметров, характеризующие степень соответствия системы по назначению.

Для АСУ указывают:

степень приспособляемости системы к изменению процессов и методов управления, к отклонениям параметров объекта управления;.

допустимые пределы модернизации и развития системы;

вероятностно-временные характеристики, при которых сохраняется целевое назначение системы.

2.6.1.4. В требования к надежности включают:

1) состав и количественные значения показателей надежности для системы в целом или ее подсистем;

2) перечень аварийных ситуаций, по которым должны быть регламентированы требования к надежности, и значения соответствующих показателей;

3) требования к надежности технических средств и программного обеспечения;

4) требования к методам оценки и контроля показателей надежности на разных стадиях создания системы в соответствии с действующими нормативно-техническими документами.

2.6.1.5. В требования по безопасности включают требования по обеспечению безопасности при монтаже, наладке, эксплуатации, обслуживании и ремонте технических средств системы (защита от воздействий электрического тока, электромагнитных полей, афотических шумов и т. п.), по допустимым уровням освещенности, вибрационных и шумовых нагрузок.

2.6.1.6. В требования по эргономике и технической эстетике включают показатели АС, задающие необходимое качество взаимодействия человека с машиной и комфортность условий работы персонала.

2.6.1.7. Для подвижных АС в требования к транспортабельности включают конструктивные требования, обеспечивающие транспортабельность технических средств системы, а также требования к транспортным средствам.

2.6.1.8. В требования к эксплуатации, техническому обслуживанию, ремонту и хранению включают:

1) условия и регламент (режим) эксплуатации, которые должны обеспечивать использование технических средств (ТС) системы с заданными техническими показателями, в том числе виды и периодичность обслуживания ТС системы или допустимость работы без обслуживания;

2) предварительные требования к допустимым площадям для размещения персонала и ТС системы, к параметрам сетей энергоснабжения и т. п.;

3) требования по количеству, квалификации обслуживающего персонала и режимам его работы;

4) требования к составу, размещению и условиям хранения комплекта запасных изделий и приборов;

5) требования к регламенту обслуживания.

2.6.1.9. В требования к защите информации от несанкционированного доступа включают требования, установленные в НТД, действующей в отрасли (ведомстве) заказчика.

2.6.1.10. В требованиях по сохранности информации приводят перечень событий: аварий, отказов технических средств (в том числе - потеря питания) и т. п., при которых должна быть обеспечена сохранность информации в системе.

2.6.1.11. В требованиях к средствам защиты от внешних воздействии приводят:

1) требования к радиоэлектронной защите средств АС;

2) требования по стойкости, устойчивости и прочности к внешним воздействиям (среде применения).

2.6.1.12. В требованиях по патентной чистоте указывают перечень стран, в отношении которых должна быть обеспечена патентная чистота системы и ее частей.

2.6.1.13. В требования к стандартизации и унификации включают:

показатели, устанавливающие требуемую степень использования стандартных, унифицированных методов реализации функций (задач) системы, поставляемых программных средств, типовых математических методов и моделей, типовых проектных решений, унифицированных форм управленческих документов, установленных ГОСТ 6.10.1, общесоюзных классификаторов технико-экономической информации и классификаторов других категорий в соответствии с областью их применения, требования к использованию типовых автоматизированных рабочих мест, компонентов и комплексов.

2.6.1.14. В дополнительные требования включают:

1) требования к оснащению системы устройствами для обучения персонала (тренажерами, другими устройствами аналогичного назначения) и документацией на них;

2) требования к сервисной аппаратуре, стендам для проверки элементов системы;

3) требования к системе, связанные с особыми условиями эксплуатации;

4) специальные требования по усмотрению разработчика или заказчика системы.

2.6.2. В подразделе «Требования к функциям (задачам)», выполняемым системой, приводят:

1) по каждой подсистеме перечень функций, задач или их комплексов (в том числе обеспечивающих взаимодействие частей системы), подлежащих автоматизации;

при создании системы в две или более очереди - перечень функциональных подсистем, отдельных функций или задач, вводимых в действие в 1-й и последующих очередях;

2) временной регламент реализации каждой функции, задачи (или комплекса задач);

3) требования к качеству реализации каждой функции (задачи или комплекса задач), к форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов;

4) перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

2.6.3. В подразделе «Требования к видам обеспечения» в зависимости от вида системы приводят требования к математическому, информационному,

лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечения системы.

2.6.3.1. Для математического обеспечения системы приводят требования к составу, области применения (ограничения) и способам использования в системе математических методов и моделей, типовых алгоритмов и алгоритмов, подлежащих разработке.

2.6.3.2. Для информационного обеспечения системы приветят требования:

1) к составу, структуре и способам организации данных в системе;

2) к информационному обмену между компонентами системы;

3) к информационной совместимости со смежными системами;

4) по использованию общесоюзных и зарегистрированных республиканских, отраслевых классификаторов, унифицированных документов и классификаторов, действующих на данном предприятии;

5) по применению систем управления базами данных;

6) к структуре процесса сбора, обработки, передачи данных в системе и. представлению данных;

7) к защите данных от разрушений при авариях и сбоях в электропитании системы;

8) к контролю, хранению, обновлению и восстановлению данных;

9) к процедуре придания юридической силы документам, продуцируемым техническими средствами АС (в соответствии с ГОСТ 6.10.4).

2.6.3.3. Для лингвистического обеспечения системы приводят требования к применению в системе языков программирования высокого уровня, языков взаимодействия пользователей и технических средств системы, а также требования к кодированию и декодированию данных, к языкам ввода-вывода данных, языкам манипулирования данными, средствам описания предметной области (объекта автоматизации), к способам организации диалога.

2.6.3.4. Для программного обеспечения системы приводят перечень покупных программных средств, а также требования:

1) к независимости программных средств от используемых СВТ и операционной среды;

2) к качеству программных средств, а также к способам его обеспечения и контроля;

3) по необходимости согласования вновь разрабатываемых программных средств с фондом алгоритмов и программ.

2.6.3.5. Для технического обеспечения системы приводят требования:

1) к видам технических средств, в том числе к видам комплексов технических средств, программно-технических комплексов и других комплектующих изделий, допустимых к использованию в системе;

2) к функциональным, конструктивным и эксплуатационным характеристикам средств технического обеспечения системы.

2.6.3.6. В требованиях к метрологическому обеспечению приводят:

1) предварительный перечень измерительных каналов;

2) требования к точности измерений параметров и (или) к метрологическим характеристикам измерительных каналов;

3) требования к метрологической совместимости технических средств системы;

4) перечень управляющих и вычислительных каналов системы, для которых необходимо оценивать точностные характеристики;

5) требования к метрологическому обеспечению технических и программных средств, входящих в состав измерительных каналов системы, средств встроенного контроля, метрологической пригодности измерительных каналов и средств измерений, используемых при наладке и испытаниях системы;

6) вид метрологической аттестации (государственная или ведомственная) с указанием порядка ее выполнения и организаций, проводящих аттестацию.

2.6.3.7. Для организационного обеспечения приводят требования:

1) к структуре и функциям подразделений, участвующих в функционировании системы или обеспечивающих эксплуатацию;

2) к организации функционирования системы и порядку взаимодействия персонала АС и персонала объекта автоматизации;

3) к защите от ошибочных действий персонала системы.

2.6.3.8. Для методического обеспечения САПР приводят требования к составу нормативно-технической документации системы (перечень применяемых при ее функционировании стандартов, нормативов, методик и т. п.).

2.7. Раздел «Состав и содержание работ по созданию (развитию) системы» должен содержать перечень стадий и этапов работ по созданию системы в соответствии с ГОСТ 24.601, сроки их выполнения, перечень организаций-исполнителей работ, ссылки на документы, подтверждающие согласие этих организаций на участие в создании системы, или запись, определяющую ответственного (заказчик или разработчик) за проведение этих работ.

В данном разделе также приводят:

1) перечень документов, по ГОСТ 34.201, предъявляемых по окончании соответствующих стадий и этапов работ;

2) вид и порядок проведения экспертизы технической документации, (стадия, этап, объем проверяемой документации, организация-эксперт);

3) программу работ, направленных на обеспечение требуемого уровня надежности разрабатываемое системы (при необходимости);

4) перечень работ по метрологическому обеспечению на всех стадиях создания системы с указанием их сроков выполнения и организации-исполнителей (при необходимости).

2.8. В разделе «Порядок контроля и приемки системы» указывают:

1) виды, состав, объем и методы испытаний системы и ее составных частей (виды испытаний в соответствии с действующими нормами, распространяющимися на разрабатываемую систему);

2) общие требования к приемке работ по стадиям (перечень участвующих предприятий и организаций, место и сроки проведения), порядок согласования и утверждения приемочной документации;

3) статус приемочной комиссии (государственная, межведомственная, ведомственная).

2.9. В разделе «Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие» необходимо привести перечень основных мероприятий и их исполнителей, которые следует выполнить при подготовке объекта автоматизации к вводу АС в действие.

В перечень основных мероприятий включают:

1) приведение поступающей в систему информации (в соответствии с требованиями к информационному и лингвистическому обеспечению) к виду, пригодному для обработки с помощью ЭВМ;

2) изменения, которые необходимо осуществить в объекте автоматизации;

3) создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ;

4) создание необходимых для функционирования системы подразделений и служб;

5) сроки и порядок комплектования штатов и обучения персонала.

Например, для АСУ приводят:

изменения применяемых методов управления;

создание условий для работы компонентов АСУ, при которых гарантируется соответствие системы требованиям, содержащимся в ТЗ.

2.10. В разделе «Требования к документированию» приводят:

1) согласованный разработчиком и заказчиком системы перечень подлежащих разработке комплектов и видов документов, соответствующих требованиям ГОСТ 34.201 и НТД отрасли заказчика; перечень документов, выпускаемых на машинных носителях; требования к микрофильмированию документации;

2) требования по документированию комплектующих элементов межотраслевого применения в соответствии с требованиями ЕСКД и ЕСПД;

3) при отсутствии государственных стандартов, определяющих требования к документированию элементов системы, дополнительно включают требования к составу и содержанию таких документов.

2.11. В разделе «Источники разработки» должны быть перечислены документы и информационные материалы (технико-экономическое обоснование, отчеты о законченных научно-исследовательских работах, информационные материалы на отечественные, зарубежные системы-аналоги и др.), на основании которых разрабатывалось ТЗ и которые должны быть использованы при создании системы.

2.12. В состав ТЗ на АС при наличии утвержденных методик включают приложения, содержащие:

1) расчет ожидаемой эффективности системы;

2) оценку научно-технического уровня системы.

Приложения включают в состав ТЗ на АС по согласованию между разработчиком и заказчиком системы.

3. ПРАВИЛА ОФОРМЛЕНИЯ

3.1. Разделы н подразделы ТЗ на АС должны быть размещены в порядке, установленном в разд. 2 настоящего стандарта.

3.2. ТЗ на АС оформляют в соответствии с требованиями ГОСТ 2.105 на листах формата А4 по ГОСТ 2.301 без рамки, основной надписи и дополнительных граф к ней.

Номера листов (страниц) проставляют, начиная с первого листа, следующего за титульным листом, в верхней части листа (над текстом, посередине) после обозначения кода ТЗ на АС.

3.3. Значения показателей, норм и требований указывают, как правило, с предельными отклонениями или максимальными минимальным значениями. Если эти показатели, нормы, требования однозначно регламентирована НТД, в ТЗ на АС следует приводить ссылку на эти документы или их разделы, а также дополнительные требования, учитывающие особенности создаваемой системы. Если конкретные значения показателей, норм и требований не могут быть установлены в процессе разработки ТЗ на АС, в нем следует сделать запись о порядке установления и согласования этих показателей, норм и требований:

«Окончательное требование (значение) уточняется в процессе ... и согласовывается протоколом с ... на стадии ...». При этом в текст ТЗ на АС изменений не вносят.

3.4. На титульном листе помещают подписи заказчика, разработчика и согласующих организаций, которые скрепляют гербовой печатью. При необходимости титульный лист оформляют на нескольких страницах. Подписи разработчиков ТЗ на АС и должностных лиц, участвующих в согласовании и рассмотрении проекта ТЗ на АС, помещают на последнем листе.

Форма титульного листа ТЗ на АС приведена в приложении 2. Форма последнего листа ТЗ на АС приведена в приложении 3.

3.5. При необходимости на титульном листе ТЗ на АС допускается помещать установленные в отрасли коды, например: гриф секретности, код работы, регистрационный номер ТЗ и др.

3.6. Титульный лист дополнения к ТЗ на АС оформляют аналогично титульному листу технического задания. Вместо наименования «Техническое задание» пишут «Дополнение № ... к ТЗ на АС … ».

3.7. На последующих листах дополнения к ТЗ на АС помещают основание для изменения, содержание изменения и ссылки на документы, в соответствии с которыми вносятся эти изменения.

3.8. При изложении текста дополнения к ТЗ следует указывать номера соответствующих пунктов, подпунктов, таблиц основного ТЗ на АС и т.п. и применять слова: «заменить», «дополнить», «исключить», «изложить в новой редакции».

ПОРЯДОК РАЗРАБОТКИ, СОГЛАСОВАНИЯ И УТВЕРЖДЕНИЯ ТЗ НА АС

1. Проект ТЗ на АС разрабатывает организация---разработчик системы с участием заказчика на основании технических требований (заявки, тактико-технического задания и т. п.).

При конкурсной организации работ варианты проекта ТЗ на АС рассматриваются заказчиком, который либо выбирает предпочтительный вариант, либо на основании сопоставительного анализа подготавливает с участием будущего разработчика АС окончательный вариант ТЗ на АС.

2. Необходимость согласования проекта ТЗ на АС с органами государственного надзора и другими заинтересованными организациями определяют совместно заказчик системы и разработчик проекта ТЗ на АС.

Работу по согласованию проекта ТЗ на АС осуществляют совместно разработчик ТЗ на АС и заказчик системы, каждый в организациях своего министерства (ведомства).

3. Срок согласования проекта ТЗ на АС в каждой организации не должен превышать 15 дней со дня его получения. Рекомендуется рассылать на согласование экземпляры проекта ТЗ на АС (копии) одновременно во все организации (подразделения).

4. Замечания по проекту ТЗ на АС должны быть представлены с техническим обоснованием. Решения по замечаниям должны быть приняты разработчиком проекта ТЗ на АС и заказчиком системы до утверждения ТЗ на АС.

5. Если при согласовании проекта ТЗ на АС возникли разногласия между разработчиком и заказчиком (или другими заинтересованными организациями), то составляется протокол разногласий (форма произвольная) и конкретное решение принимается в установленном порядке.

6. Согласование проекта ТЗ на АС разрешается оформлять отдельным документом (письмом). В этом случае под грифом Согласованы» делают ссылку на этот документ.

7. Утверждение ТЗ на АС осуществляют руководители предприятий (организаций) разработчика и заказчика системы.

8. ТЗ на АС (дополнение к ТЗ) до передачи его на утверждение должно быть проверено службой нормоконтроля организации - разработчика ТЗ и при необходимости, подвергнуто метрологической экспертизе.

9. Копии утвержденного ТЗ на АС в 10-дневный срок после утверждения высылаются разработчиком ТЗ на АС участникам создания системы.

10. Согласование и утверждение дополнений к ТЗ на АС проводят в порядке, установленном для ТЗ на АС.

11. Изменения к ТЗ на АС не допускается утверждать после представления системы для ее очереди на приемо-сдаточные испытания.

12. Регистрация, учет и хранение ТЗ на АС и дополнений к нему проводят в соответствии с требованиями ГОСТ 2.501.

ФОРМА ТИТУЛЬНОГО ЛИСТА ТЗ НА АС

наименование организации - разработчика ТЗ на АС

УТВЕРЖДАЮ	УТВЕРЖДАЮ
Руководитель (должность, наименование предприятия - заказчика АС)	Руководитель (должность, наименование предприятия - разработчика АС)
Личная **Расшифровка**	**Личная** **Расшифровка**
подпись подписи	подпись подписи
Печать	Печать
Дата	Дата

_____

наименование вида АС

_____

наименование объекта автоматизации

_____

сокращенное наименование АС

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На_____ листах

Действует с

СОГЛАСОВАНО

Руководитель (должность, наименование согласующей организации)

Личная подпись Расшифровка подписи

Печать

Дата

ФОРМА ПОСЛЕДНЕГО ЛИСТА ТЗ НА АС

_____

(код ТЗ)

СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

СОГЛАСОВАНО

Наименование организации, предприятия	Должность	Фамилия, имя, отчество	Подпись	Дата

ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. РАЗРАБОТАН И ВНЕСЕН Государственным комитетом СССР по стандартам Министерством приборостроения, средств автоматизации и систем управления СССР

ИСПОЛНИТЕЛИ

Л. М. Зайденберг, канд. техн. наук; **Ю. Б. Ирз**, канд. техн. наук; **В. И. Передков; И. С. Митяев; Б. А. Дюков** (руководители темы); **Л. О. Хвилевицкий**, канд. техн. наук; **С. В. Гаршина; П. А. Шалаев**, канд. техн. наук; **Е. С. Кранков**, канд. техн. наук; **А. М. Мустафина; Г. И. Шалатова; Р. С. Седегов**, д-р эконом. наук; **Я. Г. Виленчик; В. И. Махнач; В. Н. Семенов; И. П. Вахлаков; Е. Г. Савина**

2. УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Государственного комитета СССР по стандартам от 24.03.89 № 661

3. ВЗАМЕН ГОСТ 24.201-85

4. ССЫЛОЧНЫЕ НОРМАТИВНО-ТЕХНИЧЕСКИЕ ДОКУМЕНТЫ

Обозначение НТД. на который дана ссылка	Номер пункта, приложения
ГОСТ 2.105-79	3.2
ГОСТ 2.301-68	3.2
ГОСТ 2.501-68	Приложение 1
ГОСТ 6.10.1-80	2.6.1.13
ГОСТ 6.10.4-84	2.6.3.2
ГОСТ 19.201-78	1.2
ГОСТ 24.601-86	2.7
ГОСТ 34.201-89	2.10.1, 2.7

Приложение 2

Пример оформления технического задания на разработку компьютерной модели

В этом примере к требованиям разработки приведены некоторые рекомендации.

1. Общие сведения

1.1	Полное наименование системы:	Программа моделирования прохождения нейтронов через пластинку
1.2	Условное обозначение системы:	Программа нейтрон
1.3	Шифр темы или шифр (номер) договора:	ПМ/0001
1.4	Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты:	Башкирский государственный педагогический университет
1.5	Перечень документов, на основании которых создается система:	1) Приказ об утверждении дипломных работ 2) Заключение о теме дипломной работы Дата согласования и утверждения: 11-10-2004
1.6	Плановые сроки начала и окончания работы по созданию системы:	Начало: 11-01-2005 Окончание: 29-05-2005
1.7	Сведения об источниках и порядке финансирования работ: источник финансирования; объем финансирования.	отсутствуют

2 Назначение и цели создания системы

2.1.	Назначение системы	Программная установка для изучения студентами явления прохождения движущихся нейтронов через пластинку.
2.2	Вид автоматизируемой деятельности	Учет сорта ядра и типа взаимодействия нейтронов с атомами вещества при прохождении их через пластинку
2.3	Объект разработки автоматизации:	Взаимодействие нейтронов с веществом
2.4	Цели создания про-	Функциональное назначение:

	граммной установки:	Программная установка для моделирования взаимодействия нейтронов с веществом Эксплуатационное назначение: Для студентов изучающих дисциплину «Компьютерное моделирование»

3 Характеристики объекта: Отсутствуют.

4 Требования к программной установке

4.1	Требования к системе в целом	ПМ Нейтрон. Уровень 0.
4.2	Перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы	Модуль интерфейса пользователя. Модуль помощи и справки Модуль ввода данных и вывода результатов моделирования
4.3	Требования к способам и средствам связи для информационного обмена между компонентами системы	Реестр активов Источник - АИС управления входными и выходными данными
4.4	Требования к характеристикам взаимосвязей создаваемой системы со смежными системами, требования к ее совместимости, способы обмена информации	Совместимость с ОС Windows
4.5	Требования к численности и квалификации персонала системы и режиму его работы	Не более двух: пользователя обладающего навыкам оператора ЭВМ и системного администратора с профильным образованием.
4.6	Требования к показателям назначения	Отсутствуют
4.7	Требования к надежности	Программный продукт будет надежно функционировать, если обеспечить: • организацию бесперебойного питания технических средств; • использования лицензионного программного обеспечения; • регулярного выполнения рекомендаций

		Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»; • регулярного выполнения требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов.
4.8	Требования к безопасности	отсутствуют
4.9	Требования к эргономике и технической эстетике	отсутствуют
4.10	Требования к транспортабельности для подвижных АС	Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта (в том числе в отапливаемых герметизированных отсеках самолетов без ограничения расстояний). При перевозке в железнодорожных вагонах вид отправки - мелкий малотоннажный. При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков. Не допускается кантование программного изделия. Климатические условия транспортирование приведены ниже: • температура окружающего воздуха, °С - от плюс 5 до плюс 50; • атмосферное давление, кПа - такое-то; • относительная влажность воздуха при 25 °С - такая-то.
4.11	Требования к эксплуатации, техническому обслуживанию, ремонту и хранению комплектов системы:	Программа будет работать на компьютере при температуре от плюс 5 до плюс 35 °С при относительной влажности 90% и атмосферном давлении 462 мм.рт. ст.
4.12	Требования к защите информации от несанкционированного доступа:	Требования к защите информации и программ не предъявляются. Подобных требований следует избегать. Обеспечить некоторый уровень защиты информации и программ

		возможно, обеспечить безопасность невозможно. Заказчик, скорее всего, это осознает и проявлять настойчивость не станет.
4.13	Требования к сохранности информации:	отсутствуют
4.14	Требования к средствам защиты от внешних воздействий:	Отсутствуют
4.15	Требования к патентной чистоте:	Отсутствуют
4.16	Требования к стандартизации и унификации:	Отсутствуют
4.17	Требования к составу и параметрам технических средств	В состав технических средств должен входить IBM- совместимый персональный компьютер (ПЭВМ), включающий в себя: • процессор Pentium-1000 с тактовой частотой, ГГц - 10, не менее; • материнскую плату с FSB, ГГц - 5, не менее; • оперативную память объемом, Тб - 10, не менее; и так далее…
4.18	Требования к видам обслуживания	Программа не требует проведения каких-либо видов обслуживания. Если Заказчик в ходе согласования технического задания сошлется на отсутствие ресурсов или желания проводить все виды обслуживания собственными силами, имеет смысл предложить разработку технического задания на сопровождение программного средства за отдельные деньги отдельным договором. Откажется – следует считать программу необслуживаемой.
4.19	Дополнительные требования:	Программа должна обеспечивать взаимодействие с пользователем (оператором) посредством графического пользовательского интерфейса, разработанного согласно рекомендациям компании-производителя операционной системы.

5 Требования к программе или программному изделию

5.1	Требования к составу выполняемых функций	Программа должна обеспечивать возможность выполнения следующих функций: 1. функции открытия (загрузки) существующего файла. 2. функции редактирования открытого (далее - текущего) файла путем ввода, замены, удаления содержимого файла с применением стандартных устройств ввода. 3. функции сохранения файла с исходным именем. . 4. функции сохранения файла с именем, отличным от исходного. 5. функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы. 6. функции вывода оперативных справок в строковом формате (подсказок). 7. функции интерактивной справочной системы. 8. функции отображения названия программы, версии программы, копирайта и комментариев разработчика
5.2	Требования к организации входных данных	Входные данные задаются через окно ввода с использованием пользовательского интерфейса
5.3	Требования к организации выходных данных	Возможность просмотра графиков через экранную форму Возможность перенесения результатов моделирования в виде графиков в Текстовый файл. Возможность распечатки результата моделирования в виде графиков

6 Требования к программной документации

Состав программной документации предусмотрен ГОСТ 19.101-77.

Предварительный состав программной документации оформляется по ГОСТ 19.106-78.

Состав программной документации должен включать в себя:

1. Техническое задание;
2. Программу и методики испытаний;
3. Руководство системного программиста;
4. Руководство оператора;
5. Ведомость эксплуатационных документов.

Программа и методики испытаний потребуются, чтобы показать

Заказчику, что разработанная Исполнителем программа соответствует требованиям согласованного и утвержденного технического задания. После проведения совместных (приемо-сдаточных) испытаний Заказчик и Исполнитель подпишут Акт приемки (сдачи) работы. И, тем самым, работа будет закрыта, условия Договора выполнены.

Согласно п. 2.6. ГОСТ 19.101-77 «Допускается объединять отдельные виды эксплуатационных документов (за исключением ведомости эксплуатационных документов и формуляра). Необходимость объединения этих документов указывается в техническом задании. Объединенному документу присваивают наименование и обозначение одного из объединяемых документов».

7	**Технико-экономические показатели**	Предполагаемое число использования программы в год – 365 сеансов работы на одном рабочем месте. В разделе должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами. Экономические преимущества разработки в сравнении с лучшими отечественными и зарубежными аналогами составляет

8. Стадии и этапы разработки

Стадии разработки и этапы регламентированы ГОСТ 19.102-77.

ГОСТ 19.102-77 не препятствует исключению отдельных стадий работ, а также объединению отдельных этапов работ.

Разработка должна быть проведена в три стадии:

1. Разработка технического задания;
2. Рабочее проектирование;
3. Внедрение.

Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

- разработка программы;
- разработка программной документации;
- испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- постановка задачи;
- определение и уточнение требований к техническим средствам;
- определение требований к программе;
- определение стадий, этапов и сроков разработки программы и документации на неё;
- выбор языков программирования;
- согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованием п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- разработка, согласование и утверждение программы (в ГОСТ, похоже, опечатка – «порядка») и методики испытаний;
- проведение приемо-сдаточных испытаний;
- корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

9	**Порядок контроля и приемки.**	В разделе должны быть указаны виды испытаний и общие требования к приемке работы.
9.1	Виды испытаний.	Приемо-сдаточные испытания должны проводиться на объекте Заказчика в сроки определенные в техническом задании. Приемо-сдаточные испытания программы должны проводиться согласно разработанной (не позднее такого-то срока) Исполнителем и согласованной Заказчиком Программы и методик испытаний.
9.2	Ход проведения приемо-сдаточных испытаний	Заказчик и Исполнитель документируют в Протоколе проведения испытаний.
9.3	Общие требования к приемке работы	На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком

		подписывают Акт приемки-сдачи программы в эксплуатацию
10	Приложения	В приложениях к техническому заданию, при необходимости, приводят: • перечень научно- исследовательских и других работ, обосновывающих разработку; • схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке; • другие источники разработки. Выложить перечень ГОСТ, на основании которых должна проводиться разработка. Например: ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению; и так далее...

Приложение 3

Ресурсы Интернет по математическому моделированию

1. http://www.exponenta.ru/ – образовательный математический сайт.
2. http://mathmod.aspu.ru/?id=2 – сайт лаборатории «Математическое моделирование и информационные технологии в науке и образовании».
3. http://www.physicon.ru/ – сайт компании «Физикон» – разработчика электронных средств обучения и курсов для школ, колледжей и вузов.
4. http://obr.1c.ru/product.jsp?id=831 – сайт компании 1- С разработчика электронных средств обучения и курсов для школ, колледжей и вузов.
5. http://www.labfor.ru – сайт лаборатории электронных средств обучения СибГУТИ (ЛЭСО).
6. http://www.anylogic.ru/ – сайт компании AnyLogic Company разработчика инструментов и бизнес-приложений имитационного моделирования.
7. http://www.scs.org/ – Международное общество имитационного моделирования (SCS).
8. http://www.systemdynamics.org/ - Международное общество системной динамики (SDS).
9. http://www.eurosim.info/ - Европейское общество имитационного моделирования (EUROSIM).
10. http://www.asim-gi.org/ - Общество имитационного моделирования немецкоязычных стран (Австрия, Германия, Швейцария) (ASIM).
11. http://www.eurosim.info/ - Голландское общество имитационного моделирования Бенилюкса (DBSS) было основано в июле 1986. Действует в пределах голландской языковой области. DBSS активно продвинул создание подобных организаций в других языковых областях. DBSS - член EUROSIM.DBSS член EUROSIM.
12. http://www.eurosim.info/ - Венгерское Общество Имитационного моделирования (HSS). HSS было основано в 1981 году. HSS член EUROSIM.
13. http://www.itl.rtu.lv/imb/index.php?lang=en&id=11 - Латвийское общество имитационного моделирования (LSS). LSS было основано в 1990 году. LSS член EUROSIM.
14. http://www.uksim.org.uk/ - Общество имитационного моделирования Великобритании (UKSIM).
15. http://www.china-simulation.com/ - Китайская ассоциация систем имитационного моделирования (CASS). Более 5000 членов в Китае и за границей.

Проводит ежегодно более 20 конференций по имитационному моделированию, моделированию и компьютерам.

16. http://www.i-m-cs.org/ - Международное общество ИМ стран Средиземноморья и Латинской Америки (IM & LACS).
17. http://www.scs-europe.net/ - Европейский Совет по Моделированию и Симуляции (ECMS).
18. http://www.iasted.org/ - Технический комитет по моделированию и имитации международной ассоциации по науке и технологиям (IASTED).
19. http://www.sisostds.org/ - Организация стандартов имитации (SISO).
20. http://www.simulationinformation.com/ - Национальный центр США по моделированию (NCS).
21. http://www.ptsk.man.bialystok.pl/ - Общество имитационного моделирования Польши (PSCS).
22. http://romsim.ici.ro/ - Общество имитационного моделирования Румынии (ROMSIM). Общество основано в 1990 году. В настоящее время насчитывает примерно 100 членов из Румынии и Республики Молдавия.
23. http://www.scansims.org/ - Общество имитационного моделирования скандинавских стран (Дания, Финляндия, Норвегия, Швеция и Исландия) (SIMS). История SIMS ведет начало с 1959 года. Управление деятельностью SIMS осуществляет правление общества, куда входят по два представителя от каждой скандинавской страны. Исландия представлена одним членом правления.
24. http://www.fit.vutbr.cz/CSSS/ - Общество имитационного моделирования Чехии и Словакии (CSSS).
25. http://www.simulationalliance.com/ - Мировой альянс имитационного моделирования скандинавских стран (Дания, Финляндия, Норвегия) (WSA).
26. http://www.liophant.org/ - Некоммерческая ассоциация имитационного моделирования Liophant.
27. http://www.jsst.jp/ - Японское общество имитационного моделирования (JSST).
28. http://www.simulation.or.kr/ - Общество имитационного моделирования Кореи (KSS).
29. http://www.eurosim.info/index.php?id=10 - Общество имитационного моделирования Хорватии. CROSSIM основано 28 марта 1992 году. Общество – аффилировано с SCS с 1994 года и является членом EUROSIM с 1997 года.

30. http://www.eurosim.info/index.php?id=13 - Общество имитационного моделирования франкоговорящих стран (Бельгия, Франция) (FRANCOSIM). FRANCOSIM было основано в 1991 году.
31. http://www.eurosim.info/index.php?id=15 - Итальянское общество имитационного моделирования (ISCS). ISCS было основано в 1984 году, является членом EUROSIM.
32. http://www.slosim.si/ - Общество имитационного моделирования Словении (SLOSIM). SLOSIM основано в 1994 году и стало членом EUROSIM в 1996 году. В настоящее время насчитывает 76 членов (из университетов, институтов и промышленности).
33. http://www.ifac-control.org/
34. http://www.ceautomatica.es/en/portal
35. http://www.ceautomatica.es/en/og/modelado-y-simulacion/presentacion - Испанская группа моделирования и имитационного моделирования (CEA SMSG). IFAC – Международная Федерация Автоматизации управления, многонациональная федерация организаций, представляющих разработку и научные общества, заинтересованные в автоматизации управлении. CEA – испанское Общество на Автоматизации и Контроля, национальный член IFAC в Испании. Одно из них называется «Моделирование и имитационное моделирование», составляя CEA-SMSG (Испанская группа моделирования и имитационного моделирования).
36. http://www.msco.mil/ - Координационный офис «Имитационное моделирование» департамента обороны США (DoD M&S CO).
37. http://www.alabamamscouncil.org/ - Совет по имитационному моделированию Алабамы (AMSC).
38. http://sysdynamics.ru/ - Российское общество системной динамики.
39. http://www.eskovostok.ru/solutions/simulation/ - Сайт компании Эко Восток, занимающейся имитационным моделированием.

Программное обеспечение

40. http://www.gpss.ru/ - Система имитационного моделирования GPSS.
41. http://www.gpc.de/e_poses.html - Имитационное моделирование с помощью сетей Петри – система POSES++.
42. http://www.iseesystems.com/ - Программное обеспечение ithink 7.0 и Stella 7.0 для моделирования непрерывно-дискретных процессов. По сравнению с ithink в Stella имеются возможности по построению моделей большой размерности и их свертки.

43. http://imaginethatinc.com/pages/demo.html - Extend – один из наиболее мощных инструментов для имитационного моделирования. Можно получить бесплатно демо-версию на CD.
44. http://modelsmo.narod.ru/ - Визуальная среда для моделирования СМО. Позволяет создавать GPSS-модели.
45. http://objectgpss.narod.ru/ - Сайт разработки Object GPSS.
46. http://www.arenasimulation.com/ - Официальный сайт системы дискретного моделирования Arena.
47. http://www.systemsnavigator.com/sn_website/?q=node/38 - Arena Simulation Software. Software demonstration.
48. http://www.dualis-it.de/ - Официальный сайт фирмы DUALIS Gmbh - разработчика программгного продукта ISSOP/Arena.
49. http://www.iteam.ru/soft/modelling/990/ - Описание и характеристики системы GPSS/H.
50. http://www.gpss-forum.narod.ru/ - Сайт GPSS форум.

Общие ресурсы по моделированию

51. http://www.simulation.org.ua - Имитационное моделирование систем, НТУУ "КПИ", Украина.
52. http://www.gpss.ru/ - Общероссийский портал по вопросам имитационного моделирования GPSS.
53. http://www.efg2.com/Lab/Library/SimulationAndModeling.htm - Самые общие ресурсы по моделированию.
54. http://www.simulationinformation.com/ - Национальный центр США по моделированию (National Center for Simulation). Сайт содержит много полезной информации по моделированию в различных областях деятельности человека. Основная направленность - моделирование в военной области. Много ссылок на ресурсы по моделированию в США.
55. http://web.mit.edu/jsterman/www/DID.html - Обзор литературы по системной динамике.

Личные страницы специалистов

56. http://manta.cs.vt.edu/balci/ - Осман Балси, профессор факультета информатики технический колледж Виржиния, Моделирование и имитация, проверка достоверности, проверка правильности, тестирование.
57. http://science.kennesaw.edu/~jgarrido - Jose M. Garrido, факультет информатики и информационных систем, дисциплина "Моделирование и имитация" Государственный университет Кеннесав. Разработчик Psim.

Учебное издание

Рамиль Фарукович Маликов

Основы разработки компьютерных моделей сложных систем

Редактор Т.В. Подкопаева
Технический редактор И.В. Пономарев

Лиц. на издат. деят. Б848421 от 03.11.2000 г. Подписано в печать 05.12.2012.
Формат 60Х84/16. Компьютерный набор. Гарнитура Times New Roman.
Отпечатано на ризографе. Усл. печ. л. – 14,6. Уч.-изд. л. – 14,4.
Тираж 100 экз. Заказ №

ИПК БГПУ 450000, г.Уфа, ул. Октябрьской революции, 3а