Федеральное государственное автономное образовательное учреждение
высшего образования

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БЕЛГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Р.Г. АСАДУЛЛАЕВ

Моделирование систем

Учебное пособие
по направлению подготовки 38.03.05 «Бизнес-информатика»

Белгород 2016

УДК 004.942
ББК 22.18

Составитель: Кандидат технических наук, доцент кафедры прикладной информатики и информационных технологий Института инженерных технологий и естественных наук

Р.Г. Асадуллаев

Рецензенты: Кандидат технических наук, доцент кафедры технической кибернетики БГТУ им. В.Г. Шухова

А.Г. Бажанов;

Кандидат технических наук, доцент, заведующий кафедрой прикладной информатики и информационных технологий

В.В. Ломакин

Моделирование систем: учебное пособие / сост. Р.Г. Асадуллаев. – Белгород, 2016. – 236 с.

Учебное пособие предназначено для проведения лекционных и практических занятий по дисциплине «моделирование систем». Пособие содержит теоретические положения процесса моделирования в зависимости от типа объекта исследования, а также лабораторный практикум в пакете MatLab.

Оглавление

Тема 1. Современное состояние проблемы моделирования систем

Цель изучения темы: изучить современного состояния проблемы моделирования систем.

Задачи изучения темы:

- рассмотреть процесс моделирования как метод научного познания;
- рассмотреть процесс моделирования при исследовании и проектировании сложных систем;
- ознакомиться с перспективами развития методов и средств моделирования систем в свете новых информационных технологий.

1.1. Моделирование как метод научного познания

Моделирование (в широком смысле) является основным методом исследований во всех областях знаний и научно обоснованным методом оценок характеристик сложных систем, используемым для принятия решений в различных сферах инженерной деятельности. Существующие и проектируемые системы можно эффективно исследовать с помощью математических моделей (аналитических и имитационных), реализуемых на современных ЭВМ, которые в этом случае выступают в качестве инструмента экспериментатора с моделью системы.

В настоящее время нельзя назвать область человеческой деятельности, в которой в той или иной степени не использовались бы методы моделирования. Особенно это относится к сфере управления различными системами, где основными являются процессы принятия решений на основе получаемой информации.

Методологическая основа моделирования. Все то, на что направлена человеческая деятельность, называется **объектом** (лат. objection — предмет). Выработка методологии направлена на упорядочение получения и обработки информации об объектах, которые существуют вне нашего сознания и взаимодействуют между собой и внешней средой.

В научных исследованиях большую роль играют **гипотезы**, т. е. определенные предсказания, основывающиеся на небольшом количестве опытных данных, наблюдений, догадок. Быстрая и полная проверка выдвигаемых гипотез может быть проведена в ходе специально поставленного эксперимента. При формулировании и проверке правильности гипотез большое значение в качестве метода суждения имеет аналогия.

Аналогией называют суждение о каком-либо частном сходстве двух объектов, причем такое сходство может быть существенным и несущественным. Необходимо отметить, что понятия существенности и несущественности сходства или различия объектов условны и относительны. **Существенность сходства** (различия) зависит от уровня абстрагирования и в

общем случае определяется конечной целью проводимого исследования. Современная научная гипотеза создается, как правило, по аналогии с проверенными на практике научными положениями. Таким образом, аналогия связывает гипотезу с экспериментом.

Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам; такие логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие природу явлений, называются **моделями**. Другими словами, **модель** (лат. modulus — мера) — это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

Определение моделирования. Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется **моделированием**. Таким образом, моделирование может быть определено как представление объекта моделью для получения информации об этом объекте путем проведения экспериментов с его моделью. Теория замещения одних объектов (оригиналов) другими объектами (моделями) и исследования свойств объектов на их моделях называется **теорией моделирования.**

Определяя гносеологическую роль теории моделирования, т. е. ее значение в процессе познания, необходимо, прежде всего, отвлечься от имеющегося в науке и технике многообразия моделей и выделить то общее, что присуще моделям различных по своей природе объектов реального мира. Это общее заключается в наличии некоторой структуры (статической или динамической, материальной или мысленной), которая подобна структуре данного объекта. В процессе изучения модель выступает в роли относительного самостоятельного квазиобъекта, позволяющего получить при исследовании некоторые знания о самом объекте.

Если результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов, протекающих в исследуемых объектах, то говорят, что модель адекватна объекту. При этом адекватность модели зависит от цели моделирования и принятых критериев.

Обобщенно моделирование можно определить как метод опосредованного познания, при котором изучаемый объект-оригинал находится в некотором соответствии с другим объектом-моделью, причем модель способна в том или ином отношении замещать оригинал на некоторых стадиях познавательного процесса. Стадии познания, на которых происходит такая замена, а также формы соответствия модели и оригинала могут быть различными:

1) моделирование как познавательный процесс, содержащий переработку информации, поступающей из внешней среды, о происходящих

в ней явлениях, в результате чего в сознании появляются образы, соответствующие объектам;

2) моделирование, заключающееся в построении некоторой системы-модели (второй системы), связанной определенными соотношениями подобия с системой-оригиналом (первой системой), причем в этом случае отображение одной системы в другую является средством выявления зависимостей между двумя системами, отраженными в соотношениях подобия, а не результатом непосредственного изучения поступающей информации.

С точки зрения философии моделирование — эффективное средство познания природы. Процесс моделирования предполагает наличие объекта исследования; исследователя, перед которым поставлена конкретная задача; модели, создаваемой для получения информации об объекте и необходимой для решения поставленной задачи. Причем по отношению к модели исследователь является, по сути дела, экспериментатором, только в данном случае эксперимент проводится не с реальным объектом, а с его моделью. Такой эксперимент для инженера есть инструмент непосредственного решения организационно-технических задач.

Любой эксперимент может иметь существенное значение в конкретной области науки только при специальной его обработке и обобщении. Единичный эксперимент никогда не может быть решающим для подтверждения гипотезы, проверки теории. Поэтому инженеры (исследователи и практики) должны быть знакомы с элементами современной методологии теории познания и, в частности, не должны забывать основного положения материалистической философии, что именно экспериментальное исследование, опыт, практика являются критерием истины.

1.2. Использование моделирования при исследовании и проектировании сложных систем

Одна из проблем современной науки и техники — разработка и внедрение в практику проектирования новейших методов исследования характеристик сложных информационно-управляющих и информационно-вычислительных систем различных уровней (например, автоматизированных систем научных исследований и комплексных испытаний, систем автоматизации проектирования, АСУ технологическими процессами, а также интегрированных АСУ, вычислительных систем, комплексов и сетей, информационное систем, цифровых сетей интегрального обслуживания и т. д). При проектировании сложных систем и их подсистем возникают многочисленные задачи, требующие оценки количественных и качественных закономерностей процессов функционирования таких систем, проведения структурного алгоритмического и параметрического их синтеза.

Особенности разработки систем. Системы информатики и вычислительной техники, автоматизированные системы обработки информации и управления, информационные системы относятся к классу больших систем, этапы проектирования, внедрения, эксплуатации и эволюции которых в настоящее время невозможны без использования различных видов моделирования. На всех перечисленных этапах для сложных видов различных уровней необходимо учитывать следующие особенности:

- сложность структуры и стохастичность связей между элементами;
- неоднозначность алгоритмов поведения при различных условиях;
- большое количество параметров и переменных;
- неполноту и недетерминированность исходной информации;
- разнообразие и вероятностный характер воздействий внешней среды;
- и т. д.

Ограниченность возможностей экспериментального исследования больших систем делает актуальной разработку методики их моделирования, которая позволила бы в соответствующей форме представить процессы функционирования систем, описание протекания этих процессов с помощью математических моделей, получение результатов экспериментов с моделями по оценке характеристики исследуемых объектов. Причем на разных этапах создания и использования перечисленных систем для всего многообразия входящих, в них подсистем применив методы моделирования преследует конкретные цели, а эффективность метода зависит от того, насколько грамотно разработчик использует возможности моделирования.

Независимо от разбиения конкретной сложной системы на подсистемы при проектировании каждой из них необходимо выполнить внешнее проектирование (макропроектирование) и внутреннее проектирование (микропроектирование). Так как на этих стадиях разработчик преследует различные цели, то и используемые при этом методы и средства моделирования могут существенно отличаться.

На стадии макропроектирования должна быть разработана обобщенная модель процесса функционирования сложной системы, позволяющая разработчику получить ответы на вопросы об эффективности различных стратегий управления объектом при его взаимодействии с внешней средой. Стадию внешнего проектирования можно разбить на анализ и синтез. **При анализе** изучают объект управления, строят модель воздействий внешней среды, определяют критерии оценки эффективности, имеющиеся ресурсы, необходимые ограничения. Конечная цель стадии анализа — построение модели объекта управления для оценки его характеристик. **При синтезе** на этапе внешнего проектирования решаются задачи выбора стратегии управления на основе модели объекта моделирования, т. е. сложной системы.

На стадии микропроектирования разрабатывают модели с целью создания эффективных подсистем. Причем используемые методы и средства

моделирования зависят от того, какие конкретно обеспечивающие подсистемы разрабатываются: информационные, математические, технические, программные и т. д.

Особенности использования моделей. Выбор метода моделирования и необходимая детализация моделей существенно зависят от этапа разработки сложной системы. На этапах обследования объекта управления, *например* промышленного предприятия, разработки технического задания и проектирования автоматизированной системы управления модели в основном носят описательный характер и преследуют цель наиболее полно представить в компактной форме информацию об объекте, необходимую разработчику системы.

На этапах разработки технического и рабочего проектов систем, модели отдельных подсистем детализируются, и моделирование служит для решения конкретных задач проектирования, т. е. выбора оптимального по определенному критерию при заданных ограничениях варианта из множества допустимых. Поэтому в основном на этих этапах проектирования сложных систем используются модели для целей синтеза.

Целевое назначение моделирования на этапе внедрения и эксплуатации сложных систем — это проигрывание возможных ситуаций для принятия обоснованных и перспективных решений по управлению объектом. Моделирование (имитацию) также широко применяют при обучении и тренировке персонала автоматизированных систем управления, вычислительных комплексов и сетей, информационных систем в различных сферах. В этом случае моделирование носит характер деловых игр. Модель, реализуемая обычно на ЭВМ, воспроизводит поведение управляемого объекта и внешней среды, а люди в определенные моменты времени принимают решения по управлению объектом.

АСОИУ являются системами, которые развиваются по мере эволюции объекта управления, появления новых средств управления и т.-д. Поэтому при прогнозировании развития сложных систем роль моделирования очень высока, так как это единственная возможность ответить на многочисленные вопросы о путях дальнейшего эффективного развития системы и выбора из них наиболее оптимального.

1.3. Перспективы развития методов и средств моделирования систем в свете новых информационных технологий

В последние годы основные достижения в различных областях науки и техники неразрывно связаны с процессом совершенствования ЭВМ. Сфера эксплуатации ЭВМ — бурно развивающаяся отрасль человеческой практики, стимулирующая развитие новых теоретических и прикладных направлений.

Аналитические и имитационные методы. Исторически первым сложился аналитический подход к исследованию систем, когда ЭВМ

использовалась в качестве вычислителя по аналитическим зависимостям. Анализ характеристик процессов функционирования больших систем с помощью только аналитических методов исследования наталкивается обычно на значительные трудности, приводящие к необходимости существенного упрощения моделей либо на этапе их построения, либо в процессе работы с моделью, что может привести к получению недостоверных результатов.

Поэтому в настоящее время наряду с построением аналитических моделей большое внимание уделяется задачам оценки характеристик больших систем на основе имитационных моделей, реализованных на современных ЭВМ с высоким быстродействием и большим объемом оперативной памяти. Причем перспективность имитационного моделирования как метода исследования характеристик процесса функционирования больших систем возрастает с повышением быстродействия и оперативной памяти ЭВМ, с развитием математического обеспечения, совершенствованием банков данных и периферийных устройств для организации диалоговых систем моделирования. Это, в свою очередь, способствует появлению новых «чисто машинных» методов решения задач исследования больших систем на основе организации имитационных экспериментов с их моделями. Причем ориентация на автоматизированные рабочие места на базе персональных ЭВМ для реализации экспериментов с имитационными моделями больших систем позволяет проводить не только анализ их характеристик, но и решать задачи структурного, алгоритмического и параметрического синтеза таких систем при заданных критериях оценки эффективности и ограничениях.

Достигнутые успехи в использовании средств вычислительной техники для целей моделирования часто создают иллюзию, что применение современной ЭВМ гарантирует возможность исследования системы любой сложности. При этом игнорируется тот факт, что в основу любой модели положено трудоемкое по затратам времени и материальных ресурсов предварительное изучение явлений, имеющих место в объекте-оригинале. И от того, насколько детально изучены реальные явления, насколько правильно проведена их формализация и алгоритмизация, зависит в конечном итоге успех моделирования конкретного объекта.

Средства моделирования систем. Расширение возможностей моделирования различных классов больших систем неразрывно связано с совершенствованием средств вычислительной техники и техники связи. Перспективным направлением для моделирования является создание иерархических многомашинных вычислительных систем и сетей.

При создании больших систем их компоненты разрабатываются различными коллективами, которые используют средства моделирования при анализе и синтезе отдельных подсистем. При этом разработчикам необходимы оперативный доступ к программно-техническим средствам

моделирования, а также оперативный обмен результатами моделирования отдельных взаимодействующих подсистем. Таким образом, появляется необходимость в создании диалоговых систем моделирования, для которых характерны следующие особенности:

- возможность одновременной работы многих пользователей, занятых разработкой одной или нескольких систем;
- доступ пользователей к программно-техническим ресурсам системы моделирования, включая, базы данных и знаний, пакеты прикладных программ моделирования;
- обеспечение диалогового режима работы с различными вычислительными машинами и устройствами, включая цифровые и аналоговые вычислительные машины;
- установки натурного и физического моделирования;
- элементы реальных систем и т. п.;
- диспетчирование работ в системе моделирования и оказание различных услуг пользователям, включая обучение работе с диалоговой системой моделирования при обеспечении дружественного интерфейса.

В зависимости от специфики исследуемых объектов в ряде случаев эффективным оказывается моделирование на аналоговых вычислительных машинах (АВМ). При этом надо иметь в виду, что АВМ значительно уступают ЭВМ по точности и логическим возможностям, но по быстродействию, схемной простоте реализации, сопрягаемости с датчиками внешней информации АВМ превосходят ЭВМ или, по крайней мере, не уступают им.

Для сложных динамических объектов перспективным является моделирование на базе **гибридных** (аналого-цифровых) вычислительных комплексов. Такие комплексы реализуют преимущества цифрового и аналогового моделирования и позволяют наиболее эффективно использовать ресурсы ЭВМ и АВМ в составе единого комплекса. При использовании гибридных моделирующих комплексов упрощаются вопросы взаимодействия с датчиками, установленными на реальных объектах, что позволяет, в свою очередь, проводить комбинированное моделирование с использованием аналого-цифровой части модели и натурной части объекта. Такие гибридные моделирующие комплексы могут входить в состав многомашинного вычислительного комплекса, что еще больше расширяет его возможности с точки зрения моделируемых классов больших систем.

Вопросы для повторения и закрепления материала

1. Дайте определение понятию аналогия.
2. Чем занимается теория моделирования?
3. Укажите особенности разработки систем.
4. В чем заключаются особенности использования моделей?
5. В чем заключается отличие аналитических и имитационных методов моделирования?

6. Для диалоговых систем моделирования характерны?

7. Что такое АВМ?

8. Каким образом строится гибридный вычислительный комплекс?

Задания для самостоятельной работы

1. Рассмотреть современные средства и инструментальные среды моделирования систем.

Тема 2. Основные понятия теории моделирования систем

Цель изучения темы: изучить основные понятия теории моделирования систем.

Задачи изучения темы:

- рассмотреть принципы системного подхода в моделировании систем;
- выявить общую характеристику проблемы моделирования систем.

2.1. Принципы системного подхода в моделировании систем

Моделирование начинается с формирования предмета исследований — системы понятий, отражающей существенные для моделирования характеристики объекта. Эта задача является достаточно сложной, что подтверждается различной интерпретацией в научно-технической литературе таких фундаментальных понятий, как система, модель, моделирование. Подобная неоднозначность не говорит об ошибочности одних и правильности других терминов, а отражает зависимость предмета исследований (моделирования) как от рассматриваемого объекта, так и от целей исследователя. Отличительной особенностью моделирования сложных систем является его многофункциональность и многообразие способов использования; оно становится неотъемлемой частью всего жизненного цикла системы. Объясняется это в первую очередь технологичностью моделей, реализованных на базе средств вычислительной техники: достаточно высокой скоростью получения результатов моделирования и их сравнительно невысокой себестоимостью.

В настоящее время при анализе и синтезе сложных (больших) систем получил развитие системный подход, который отличается от классического (или индуктивного) подхода. Последний рассматривает систему путем перехода от частного к общему и синтезирует (конструирует) систему путем слияния ее компонент, разрабатываемых раздельно. В отличие от этого системный подход предполагает последовательный переход от общего к частному, когда в основе рассмотрения лежит цель, причем исследуемый объект выделяется из окружающей среды.

Объект моделирования. Специалисты по проектированию и эксплуатации сложных систем имеют дело с системами управления различных уровней, обладающими общим свойством — стремлением достичь некоторой цели. Эту особенность учтем в следующих определениях системы. **Система** S — целенаправленное множество взаимосвязанных элементов любой природы. **Внешняя среда** E — множество существующих вне системы элементов любой природы, оказывающих влияние на систему или находящихся под ее воздействием.

В зависимости от цели исследования могут рассматриваться разные соотношения между самим объектом S и внешней средой E.

Таким образом, в зависимости от уровня, на котором находится наблюдатель объект исследования может выделяться по-разному и могут иметь место различные взаимодействия этого объекта с внешней средой. **Системный подход** — это элемент учения об общих законах развития природы и одно из выражений диалектического учения. Важны выделение самой системы S и внешней среды E из объективно существующей реальности и описание системы исходя из общесистемных позиций.

При системном подходе к моделированию систем необходимо прежде всего четко определить цель моделирования. Поскольку невозможно полностью смоделировать реально функционирующую систему (систему-оригинал, или первую систему), создается модель (система-модель, или вторая система) под поставленную проблему. Таким образом, применительно к вопросам моделирования цель возникает из требуемых задач моделирования, что позволяет подойти к выбору критерия и оценить, какие элементы войдут в создаваемую модель. Поэтому необходимо иметь критерий отбора отдельных элементов в создаваемую модель.

Подходы к исследованию систем. Важным для системного подхода является определение структуры системы — совокупности связей между элементами системы, отражающих их взаимодействие. Структура системы может изучаться извне с точки зрения состава отдельных подсистем и отношений между ними, а также изнутри, когда анализируются отдельные свойства, позволяющие системе достигать заданной цели, т. е. когда изучаются функции системы. В соответствии с этим наметился ряд подходов к исследованию структуры системы с ее свойствами, к которым следует прежде всего отнести структурный и функциональный.

При структурном подходе выявляются состав выделенных элементов системы S и связи между ними. Совокупность элементов связей между ними позволяет судить о структуре системы. Последняя в зависимости от цели исследования может быть описана на разных уровнях рассмотрения. Наиболее общее описание структуры — это топологическое описание, позволяющее определить в самых общих понятиях составные части системы и хорошо формализуемое на базе теории графов.

Менее общим является функциональное описание, когда рассматриваются отдельные функции, т. е. алгоритмы поведения системы, и реализуется функциональный подход, оценивающий функции которые выполняет система, причем под функцией понимается свойство, приводящее к достижению цели. Поскольку функция отображает свойство, а свойство отображает взаимодействие системы S с внешней средой E, то свойства могут быть выражены в виде либо некоторых характеристик элементов S;W и подсистем St системы, либо системы S в целом.

При наличии некоторого эталона сравнения можно ввести:

- количественные характеристики систем, для которых вводятся числа, выражающие отношения между данной характеристикой и эталоном;
- качественные характеристики систем, которые находятся, например, с помощью метода экспертных оценок.

Проявление функций системы во времени S(t), т. е. функционирование системы, означает переход системы из одного состояния в другое, т. е. движение в пространстве состояний Z. При эксплуатации системы S весьма важно качество ее функционирования, определяемое показателем эффективности и являющееся значением критерия оценки эффективности. Существуют различные подходы к выбору критериев оценки эффективности. Система S может оцениваться либо совокупностью частных критериев, либо некоторым общим интегральным критерием.

Следует отметить, что создаваемая модель M с точки зрения системного подхода также является системой, т. е. S' = S'(M), и может рассматриваться по отношению к внешней среде E. Наиболее просты методы по представлению модели, в которых сохраняется прямая аналогия явления. Применяют также модели, в которых нет прямой аналогии, а сохраняются лишь законы и общие закономерности поведения элементов системы S. Правильное понимание взаимосвязей как внутри самой модели M, так и взаимодействия ее с внешней средой E в значительной степени определяется тем, на каком уровне находится наблюдатель.

Простой подход к изучению взаимосвязей между отдельными частями модели предусматривает рассмотрение их как отражение связей между отдельными подсистемами объекта. Такой классический подход может быть использован при создании достаточно простых моделей. Процесс синтеза модели M на основе классического (индуктивного) подхода представлен на рисунке 2.1. Реальный объект, подлежащий моделированию, разбивается на отдельные подсистемы, т. е. выбираются исходные данные Д для моделирования и ставятся цели Ц, отображающие отдельные стороны процесса моделирования. По отдельной совокупности исходных данных Д ставится цель моделирования отдельной стороны функционирования системы, на базе этой цели формируется некоторая компонента К будущей модели. Совокупность компонент объединяется в модель M.

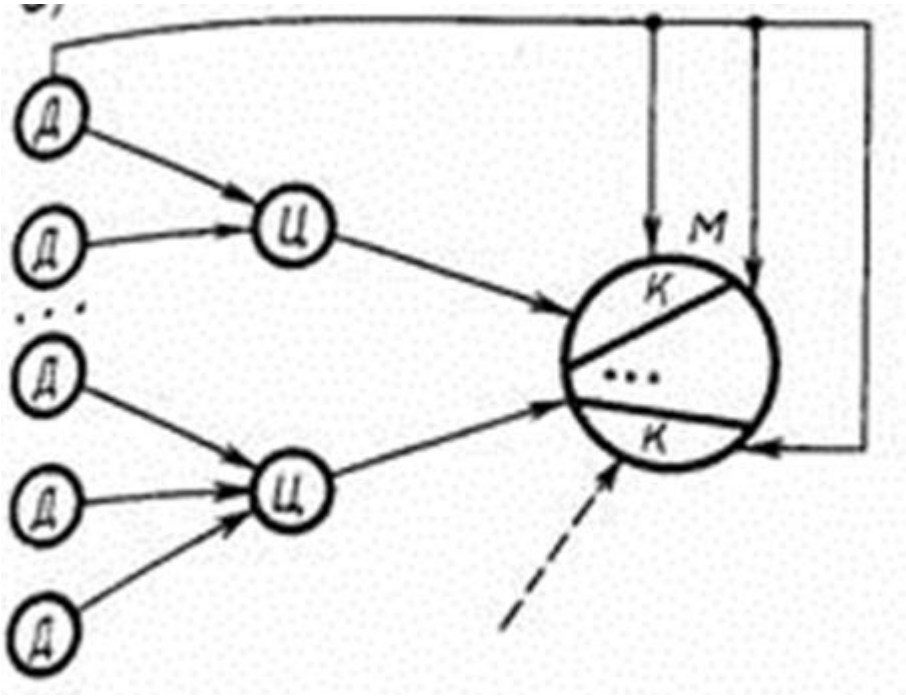


Рисунок 2.1 – Процесс синтеза модели на основе классического подхода

Таким образом, разработка модели М на базе классического подхода означает суммирование отдельных компонент в единую модель, причем каждая из компонент решает свои собственные задачи и изолирована от других частей модели. Поэтому классический подход может быть использован для реализации сравнительно простых моделей, в которых возможно разделение и взаимно независимое рассмотрение отдельных сторон функционирования реального объекта. Для модели сложного объекта такая разобщенность решаемых задач недопустима, так как приводит к значительным затратам ресурсов при реализации модели на базе конкретных программно-технических средств. Можно отметить две отличительные стороны классического подхода наблюдается движение от частного к общему, создаваемая модель (система) образуется путем суммирования отдельных ее компонент и не учитывается возникновение нового системного эффекта.

С усложнением объектов моделирования возникла необходимость наблюдения их с более высокого уровня. В этом случае наблюдатель (разработчик) рассматривает данную систему как некоторую подсистему какой-то метасистемы, т. е. системы более высокого ранга, и вынужден перейти на позиции нового системного подхода, который позволит ему построить не только исследуемую систему, решающую совокупность задачи создавать систему, являющуюся составной частью метасистемы. **Например,** если ставится задача проектирования АСУ предприятием, то с позиции

системного подхода нельзя забывать о том, что эта система является составной частью АСУ объедением.

Системный подход позволяет решить проблему построения сложной системы с учетом всех факторов и возможностей, пропорциональных их значимости, на всех этапах исследования системы S и построения модели М. Системный подход означает, что каждая система S является интегрированным целым даже тогда, когда она состоит из отдельных разобщенных подсистем. Таким образом, в основе системного подхода лежит рассмотрение системы как интегрированного целого, причем это рассмотрение при разработке начинается с главного — формулировки цели функционирования. Процесс синтеза модели М на базе системного подхода условно представлен на рисунке 2.2. На основе исходных данных Д, которые известны из анализа внешней системы, тех ограничений, которые накладываются на систему сверху либо, исходя из возможностей ее реализации, и на основе цели функционирования формулируются исходные требования Т к модели системы S. На базе этих требований формируются ориентировочно некоторые подсистемы П, элементы Э и осуществляется наиболее сложный этап синтеза — выбор В составляющих системы, для чего используются специальные критерии выбора КВ.

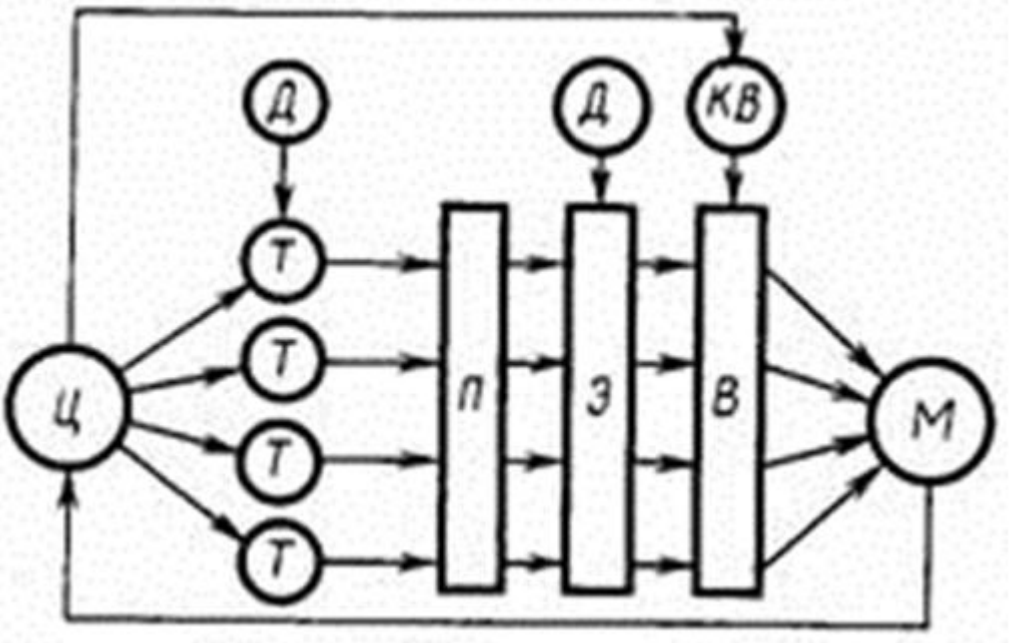


Рисунок 2.2 – Процесс синтеза модели на основе системного подхода

Стадии разработки моделей. На базе системного подхода может быть предложена и некоторая последовательность разработки моделей, когда выделяют две основные стадии проектирования: макропроектирование и микропроектирование.

На стадии макропроектирования на основе данных о реальной системе S и внешней среде Е строится модель внешней среды, выявляются ресурсы и ограничения для построения модели системы, выбирается модель системы и

критерии, позволяют оценить адекватность модели M реальной системы S. Построив модель системы и модель внешней среды, на основе критерия эффективности функционирования системы в процессе моделирования выбирают оптимальную стратегию управления, что позволяет реализовать возможности модели по воспроизведению отдельных сторон функционирования реальной системы S.

Стадия микропроектирования в значительной степени зависит от конкретного типа выбранной модели. В случае имитационной модели необходимо обеспечить создание информационного, математического, технического и программного обеспечений системного моделирования. На этой стадии можно установить основные характеристики созданной модели, оценить время работы с ней и затраты ресурсов для получения заданного качества соответствия модели процессу функционирования системы S.

Независимо от типа используемой модели M при ее построении необходимо руководствоваться рядом принципов системного подхода:

1) пропорционально-последовательное продвижение по этапам и направлениям создания модели;

2) согласование информационных, ресурсных, надежностных и других характеристик;

3) правильное соотношение отдельных уровней иерархии в системе моделирования;

4) целостность отдельных обособленных стадий построения модели.

Модель M должна отвечать заданной цели ее создания, поэтому отдельные части должны компоноваться взаимно, исходя из единой системной задачи. Цель может быть сформулирована качественно, тогда она будет обладать большей содержательностью и длительное время может отображать объективные возможности данной системы моделирования. При количественной формулировке цели возникает целевая функция, которая точно отображает наиболее существенные факторы, влияющие на достижение цели.

Построение модели относится к числу системных задач, при решении которых синтезируют решения на базе огромного числа исходных данных, на основе предложений больших коллективов специалистов. Использование системного подхода в этих условиях позволяет не только построить модель реального объекта, но и на базе этой модели выбрать необходимое количество управляющей информации в реальной системе, оценить показатели ее функционирования и тем самым на базе моделирования найти наиболее эффективный вариант построения и выгодный режим функционирования реальной системы S.

2.2. Общая характеристика проблемы моделирования систем

С развитием системных исследований, с расширением экспериментальных методов изучения реальных явлений все большее значение приобретают абстрактные методы, появляются новые научные дисциплины, автоматизируются элементы умственного труда. Важное значение при создании реальных систем S имеют математические методы анализа и синтеза, целый ряд открытий базируется на чисто теоретических изысканиях.

Экспериментальные исследования систем. Одновременно с развитием теоретических методов анализа и синтеза совершенствуются и методы экспериментального изучения реальных объектов, появляются новые средства исследования. **Однако эксперимент был и остается одним из основных и существенных инструментов познания**. Подобие и моделирование позволяют по-новому описать реальный процесс и упростить экспериментальное его изучение. Совершенствуется и само понятие моделирования. Если раньше моделирование означало реальный физический эксперимент либо построение макета, имитирующего реальный процесс, то в настоящее время появились новые виды моделирования, в основе которых лежит постановка не только физических, но также и математических экспериментов.

Познание реальной действительности является длительным и сложным процессом. Определение качества функционирования большой системы, выбор оптимальной структуры и алгоритмов поведения, построение системы S в соответствии с поставленной перед нею целью — основная проблема при проектировании современных систем, поэтому моделирование можно рассматривать как один из методов, используемых при проектировании и исследовании больших систем.

Моделирование базируется на некоторой аналогии реального и мысленного эксперимента. **Аналогия** — основа для объяснения изучаемого явления, однако критерием истины может служить только практика, только опыт. Хотя современные научные гипотезы могут создаться чисто теоретическим путем, но, по сути, базируются на широких практических знаниях. Для, объяснения реальных процессов выдвигаются гипотезы, для подтверждения которых ставится эксперимент либо проводятся такие теоретические рассуждения, которые логически подтверждают их правильность. В широком смысле под экспериментом можно понимать некоторую процедуру организации и наблюдения каких-то явлений, которые осуществляют в условиях, близких к естественным, либо имитируют их.

Различают **пассивный эксперимент**, когда исследователь наблюдает протекающий процесс, и **активный**, когда наблюдатель вмешивается и организует протекание процесса. В последнее время распространен активный эксперимент, поскольку именно на его основе удается выявить критические

ситуации, получить наиболее интересные закономерности, обеспечить возможность повторения эксперимента в различных точках и т. д.

В основе любого вида моделирования лежит некоторая модель, имеющая соответствие, базирующееся на некотором общем качестве, характеризующем реальный объект. Объективно реальный объект обладает некоторой формальной структурой, поэтому для любой модели характерно наличие некоторой структуры, соответствующей формальной структуре реального объекта, либо изучаемой стороне этого объекта.

В основе моделирования лежат информационные процессы, поскольку само создание модели М базируется на информации о реальном объекте. В процессе реализации модели получается информация о данном объекте, одновременно в процессе эксперимента с моделью вводится управляющая информация, существенное место занимает обработка полученных результатов, т. е. информация лежит в основе всего процесса моделирования.

Характеристики моделей систем. В качестве объекта моделирования выступают сложные организационно-технические системы, которые можно отнести к классу больших систем. Более того, по своему содержанию и созданная модель М также становится системой S{M) и тоже может быть отнесена к классу больших систем, для которых характерно следующее:

1. Цель функционирования, которая определяет степень целенаправленности поведения модели М. В этом случае модели могут быть разделены на одноцелевые, предназначенные для решения одной задачи, и многоцелевые, позволяющие разрешить или рассмотреть ряд сторон функционирования реального объекта.

2. Сложность, учитывая, что модель М является совокупностью отдельных элементов и связей между ними, можно оценить по общему числу элементов в системе и связей между ними. По разнообразию элементов можно выделить ряд уровней иерархии, отдельные функциональные подсистемы в модели М, ряд входов и выходов и т. д., т. е. понятие сложности может быть идентифицировано по целому ряду признаков.

3. Целостность, указывающая на то, что создаваемая модель М является одной целостной системой S(M), включает в себя большое количество составных частей (элементов), находящихся в сложной взаимосвязи друг с другом.

4. Неопределенность, которая проявляется в системе: по состоянию системы, возможности достижения поставленной цели, методам решения задач, достоверности исходной информации и т. д. Основной характеристикой неопределенности служит такая мера информации, как энтропия, позволяющая в ряде случаев оценить количество управляющей информации, необходимой для достижения заданного состояния системы.

5. Поведенческая страта, которая позволяет оценить эффективность достижения системой поставленной цели. В зависимости от наличия случайных воздействий можно различать детерминированные и

стохастические системы, по своему поведению — непрерывные и дискретные и т. д. Поведенческая страта рассмотрения системы S позволяет применительно к модели М оценить эффективность построенной модели, а также точность и достоверность полученных при этом результатов. Очевидно, что поведение модели М не обязательно совпадает с поведением реального объекта, причем часто моделирование может быть реализовано на базе иного материального носителя.

6. Адаптивность, которая является свойством высокоорганизованной системы. Благодаря адаптивности удается приспособиться к различным внешним возмущающим факторам в широком диапазоне изменения воздействий внешней среды. Применительно в модели существенна возможность ее адаптации в широком спектре возмущающих воздействий, а также изучение поведения модели в изменяющихся условиях, близких к реальным. Надо отметить, что существенным может оказаться вопрос устойчивости модели к различным возмущающим воздействиям. Поскольку модель М—сложная система, весьма важны вопросы, связанные с ее существованием, т. е. вопросы живучести, надежности и т. д.

7. Организационная структура системы моделирования, которая во многом зависит от сложности модели и степени совершенства средств моделирования. Одним из последних достижений в области моделирования можно считать возможность использования имитационных моделей для проведения машинных экспериментов. Необходимы оптимальная организационная структура комплекса технических средств, информационного, математического и программного обеспечений системы моделирования S'(M), оптимальная организация процесса моделирования, поскольку следует обращать особое внимание на время моделирования и точность получаемых результатов.

8. Управляемость модели, вытекающая из необходимости обеспечивать управление со стороны экспериментаторов для получения возможности рассмотрения протекания процесса в различных условиях, имитирующих реальные. В этом смысле наличие многих управляемых параметров и переменных модели в реализованной системе моделирования дает возможность поставить широкий эксперимент и получить обширный спектр результатов. Управляемость системы тесно связана и со степенью автоматизации моделирования.

9.Возможность развития модели, которая исходя из современного состояния науки и техники, позволяет создавать мощные системы моделирования S(M) для исследования многих сторон функционирования реального объекта. Однако нельзя при создании системы моделирования ограничиваться только задачами сегодняшнего дня. Необходимо предусматривать возможность развития системы моделирования как по горизонтали в смысле расширения спектра изучаемых функций, так и по вертикали в смысле расширения числа подсистем, т. е. созданная система

моделирования должна позволять применять новые современные методы и средства. Естественно, что интеллектуальная система моделирования может функционировать только совместно с коллективом людей, поэтому к ней предъявляют эргономические требования.

Цели моделирования систем. Любую модель строят в зависимости от цели, которую ставит перед ней исследователь, поэтому одна из основных проблем при моделировании — это проблема целевого назначения. Подобие процесса, протекающего в модели М, реальному процессу является не целью, а условием правильного функционирования модели, и поэтому в качестве цели должна быть поставлена задача изучения какой-либо стороны функционирования объекта.

Для упрощения модели М цели делят на подцели и создают более эффективные виды моделей в зависимости от полученных подцелей моделирования. Можно указать целый ряд примеров целей моделирования в области сложных систем.

Если цель моделирования ясна, то возникает следующая проблема, а именно проблема построения модели М. Построение модели оказывается возможным, если имеется информация или выдвинуты гипотезы относительно структуры, алгоритмов и параметров исследуемого объекта. На основании их изучения осуществляется идентификация объекта. В настоящее время широко применяют различные способы оценки параметров: по методу наименьших квадратов, по методу максимального правдоподобия, байесовские, Марковские оценки.

Если модель М построена, то следующей проблемой можно считать проблему работы с ней, т. е. реализацию модели, основные задачи которой — минимизация времени получения конечных результатов и обеспечение их достоверности.

Для правильно построенной модели М характерным является то, что она выявляет лишь те закономерности, которые нужны исследователю, и не рассматривает свойства системы S, не существенные для данного исследования. Следует отметить, что оригинал и модель должны быть одновременно сходны по одним признакам и различны по другим, что позволяет выделить наиболее важные изучаемые свойства. В этом смысле модель выступает как некоторый «заместитель» оригинала, обеспечивающий фиксацию и изучение лишь некоторых свойств реального объекта.

Таким образом, характеризуя проблему моделирования в целом, необходимо учитывать, что от постановки задачи моделирования до интерпретации полученных результатов существует большая группа сложных научно-технических проблем, к основным из которых можно отнести следующие: идентификацию реальных объектов, выбор вида моделей, построение моделей и их машинную реализацию, взаимодействие исследователя с моделью в ходе машинного эксперимента, проверку правильности полученных в ходе моделирования результатов, выявление

основных закономерностей, исследованных в процессе моделирования. В зависимости от объекта моделирования и вида используемой модели эти проблемы могут иметь разную значимость.

В одних случаях наиболее сложной оказывается идентификация, в других — проблема построения формальной структуры объекта. Возможны трудности и при реализации модели, особенно в случае имитационного моделирования больших систем. При этом следует подчеркнуть роль исследователя в процессе моделирования. Постановка задачи, построение содержательной модели реального объекта во многом представляют собой творческий процесс и базируются на эвристике. И в этом смысле нет формальных путей выбора оптимального вида модели. Часто отсутствуют формальные методы, позволяющие достаточно точно описать реальный процесс. Поэтому выбор той или иной аналогии, выбор того или иного математического аппарата моделирования полностью основывается на имеющемся опыте исследователя и ошибка исследователя может привести к ошибочным результатам моделирования.

Средства вычислительной техники, которые в настоящее время широко используются либо для вычислений при аналитическом моделировании, либо для реализации имитационной модели системы, могут лишь помочь с точки зрения эффективности реализации сложной модели, но не позволяют подтвердить правильность той или иной модели. Только на основе обработанных данных, опыта исследователя можно с достоверностью оценить адекватность модели по отношению к реальному процессу.

Если в ходе моделирования существенное место занимает реальный физический эксперимент, то здесь весьма важна и надежность используемых инструментальных средств, поскольку сбои и отказы программно-технических средств могут приводить к искаженным значениям выходных данных, отображающих протекание процесса. И в этом смысле при проведении физических экспериментов необходимы специальная аппаратура, специально разработанное математическое и информационное обеспечение, которые позволяют реализовать диагностику средств моделирования, чтобы отсеять те ошибки в выходной информации, которые вызваны неисправностями функционирующей аппаратуры. В ходе машинного эксперимента могут иметь место и ошибочные действия человека-оператора. В этих условиях серьезные задачи стоят в области эргономического обеспечения процесса моделирования.

Вопросы для повторения и закрепления материала

1. В чем заключается принцип системного подхода?
2. Что является объектом моделирования?
3. Что относится к внешней среде по отношению к модели?
4. Каким образом работает структурный подход при исследовании систем?

5. Каким образом работает функциональный подход при исследовании систем?

6. Как выглядит процесс синтеза модели при классическом подходе?

7. Как выглядит процесс синтеза модели при системном подходе?

8. В чем заключается отличие пассивного и активного экспериментов?

9. Перечислите и разъясните характеристики моделей систем.

10. Как определяют цели моделирования?

Задания для самостоятельной работы

1. Рассмотреть методы системного анализа, используемые для выявления исходных данных для построения модели.

Тема 3. Классификация видов моделирования и возможности имитационного моделирования

Цель изучения темы: изучить классификацию видов моделирования, а также рассмотреть возможности имитационного моделирования.

Задачи изучения темы:

- рассмотреть классификацию видов моделирования систем;
- рассмотреть возможности и эффективность моделирования систем на вычислительных машинах.

3.1. Классификация видов моделирования систем

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие может иметь место лишь при замене одного объекта другим точно таким же. При моделировании абсолютное подобие не имеет места и стремятся к тому, чтобы модель достаточно хорошо отображала исследуемую сторону функционирования объекта.

Классификационные признаки. В качестве одного из первых признаков классификации видов моделирования можно выбрать степень полноты модели и разделить модели в соответствии с этим признаком на:

- полные, в основе лежит полное подобие, которое проявляется как во времени, так и в пространстве.
- неполные, характерно неполное подобие модели изучаемому объекту.
- приближенные, в основе лежит приближенное подобие, при котором некоторые стороны функционирования реального объекта не моделируются совсем.

Классификация видов моделирования систем S приведена на рисунке 3.1.

Рисунок 3.1 – Классификация видов моделирования систем

В зависимости от характера изучаемых процессов в системе S все виды моделирования могут быть разделены на:

1. **Детерминированные**. Детерминированное моделирование отображает детерминированные процессы, т. е. процессы, в которых предполагается отсутствие всяких случайных воздействий.

2. **Стохастические**. Стохастическое моделирование отображает вероятностные процессы и события. В этом случае анализируется ряд реализаций случайного процесса, и оцениваются средние характеристики, т. е. набор однородных реализаций.

3. **Статические**. Статическое моделирование служит для описания поведения объекта в какой-либо момент времени.

4. **Динамические**. Динамическое моделирование отражает поведение объекта во времени.

5. **Дискретные.** Дискретное моделирование служит для описания процессов, которые предполагаются дискретными.

6. **Непрерывные**. Непрерывное моделирование позволяет отразить непрерывные процессы в системах.

7. **Дискретно-непрерывные**. Дискретно-непрерывное моделирование используется для случаев, когда хотят выделить наличие как дискретных, так и непрерывных процессов.

В зависимости от формы представления объекта (системы S) можно выделить: мысленное и реальное моделирование.

8. **Мысленное** моделирование часто является единственным способом моделирования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания. **Например**, на базе мысленного моделирования могут быть проанализированы многие ситуации микромира, которые не поддаются физическому эксперименту. Мысленное моделирование может быть реализовано в виде:

8.1. **Наглядное моделирование**. При наглядном моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте. Наглядное моделирование подразделяется на:

8.1.1. **Гипотетическое**. В основу гипотетического моделирования исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта. Гипотетическое моделирование используется, когда знаний об объекте недостаточно для построения формальных моделей.

8.1.2. **Аналоговое.** Аналоговое моделирование основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов. С усложнением объекта используют аналогии последующих уровней, когда аналоговая модель отображает несколько либо только одну сторону функционирования объекта.

8.1.3. **Макетирование.** Существенное место при мысленном наглядном моделировании занимает макетирование. Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования. В основе построения мысленных макетов также лежат аналогии, однако обычно базирующиеся на причинно-следственных связях между явлениями и процессами в объекте. Если ввести условное обозначение отдельных понятий, т. е. знаки, а также определенные операции между этими знаками, то можно реализовать знаковое моделирование и с помощью знаков отображать набор понятий — составлять отдельные цепочки из слов и предложений. Используя операции объединения, пересечения и дополнения теории множеств, можно в отдельных символах дать описание какого-то реального объекта.

8.2. **Символическое моделирование.** Символическое моделирование представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков или символов:

8.2.1. **Языковое.** В основе языкового моделирования лежит некоторый тезаурус. Последний образуется из набора входящих понятий, причем этот

набор должен быть фиксированным. Следует отметить, что между тезаурусом и обычным словарем имеются принципиальные различия. Тезаурус — словарь, который очищен от неоднозначности, т. е. в нем каждому слову может соответствовать лишь единственное понятие, хотя в обычном словаре одному слову могут соответствовать несколько понятий.

8.3. **Математическое моделирование**. Для исследования характеристик процесса функционирования любой системы S математическими методами, включая и машинные, должна быть проведена формализация этого процесса, т. е. построена математическая модель.

Под математическим моделированием будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта. Вид математической модели зависит как от природы реального объекта, так и задач исследования объекта и требуемой достоверности и точности решения этой задачи. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности. Математическое моделирование для исследования характеристик процесса функционирования систем можно разделить на:

8.3.1. **Аналитическое.** Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегродифференциальных, конечно-разностных и т. п.) или логических условий. Аналитическая модель может быть исследована следующими методами: а) аналитическим, когда стремятся получить в общем виде явные зависимости для искомых характеристик; б) численным, когда, не умея решать уравнений в общем виде, стремятся получить числовые результаты при конкретных начальных данных; в) качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решения).

Наиболее полное исследование процесса функционирования системы можно провести, если известны явные зависимости, связывающие искомые характеристики с начальными условиями, параметрами и переменными системы S. Однако такие зависимости удается получить только для сравнительно простых систем. При усложнении систем исследование их аналитическим методом наталкивается на значительные трудности, которые часто бывают непреодолимыми. Поэтому, желая использовать аналитический метод, в этом случае идут на существенное упрощение первоначальной модели, чтобы иметь возможность изучить хотя бы общие свойства системы. Такое исследование на упрощенной модели аналитическим методом помогает получить ориентировочные результаты для определения более точных оценок другими методами. Численный метод позволяет исследовать

по сравнению с аналитическим методом более широкий класс систем, но при этом полученные решения носят частный характер. Численный метод особенно эффективен при использовании ЭВМ.

В отдельных случаях исследования системы могут удовлетворить и те выводы, которые можно сделать при использовании качественного метода анализа математической модели. Такие качественные методы широко используются, например, в теории автоматического управления для оценки эффективности различных вариантов систем управления.

8.3.2. **Имитационное**. При имитационном моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы S во времени, причем имитируются элементарные явления, составляющие процесс, с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состояниях процесса в определенные моменты времени, дающие возможность оценить характеристики системы S.

Основным преимуществом имитационного моделирования по сравнению с аналитическим является возможность решения более сложных задач. Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики элементов системы, многочисленные случайные воздействия и др., которые часто создают трудности при аналитических исследованиях. В настоящее время имитационное моделирование — наиболее эффективный метод исследования больших систем, а часто и единственный практически доступный метод получения информации о поведении системы, особенно на этапе ее проектирования.

Когда результаты, полученные при воспроизведении на имитационной модели процесса функционирования системы S, являются реализациями случайных величин и функций, тогда для нахождения характеристик процесса требуется его многократное воспроизведение с последующей статистической обработкой информации и целесообразно в качестве метода машинной реализации имитационной модели использовать метод статистического моделирования. Первоначально был разработан метод статистических испытаний, представляющий собой численный метод, который применялся для моделирования случайных величин и функций, вероятностные характеристики которых совпадали с решениями аналитических задач (такая процедура получила название метода Монте-Карло). Затем этот прием стали применять и для машинной имитации с целью исследования характеристик процессов функционирования систем, подверженных случайным воздействиям, т. е. появился метод статистического моделирования. Таким образом, методом статистического моделирования будем в дальнейшем называть метод машинной реализации имитационной модели, а методом статистических испытаний (Монте-Карло) — численный, метод решения аналитической задачи.

Метод имитационного моделирования позволяет решать задачи анализа больших систем S, включая задачи оценки: вариантов структуры системы, эффективности различных алгоритмов управления системой, влияния изменения различных параметров системы. Имитационное моделирование может быть положено также в основу структурного, алгоритмического и параметрического синтеза больших систем, когда требуется создать систему, с заданными характеристиками при определенных ограничениях, которая является оптимальной по некоторым критериям оценки эффективности.

При решении задач машинного синтеза систем на основе их имитационных моделей помимо разработки моделирующих алгоритмов для анализа фиксированной системы необходимо также разработать алгоритмы поиска оптимального варианта системы. Далее в методологии машинного моделирования будем различать два основных раздела: статику и динамику,— основным содержанием которых являются соответственно вопросы анализа и синтеза систем, заданных моделирующими алгоритмами.

8.3.3. **Комбинированное**. Комбинированное (аналитико-имитационное) моделирование при анализе и синтезе систем позволяет объединить достоинства аналитического и имитационного моделирования. При построении комбинированных моделей проводится предварительная декомпозиция процесса функционирования объекта на составляющие под процессы и для тех из них, где это, возможно, используются аналитические модели, а для остальных подпроцессов строятся имитационные модели. Такой комбинированный подход позволяет охватить качественно новые классы систем, которые не могут быть исследованы с использованием только аналитического и имитационного моделирования в отдельности.

9. **Реальное моделирование**. При реальном моделировании используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части. Такие исследования могут проводиться как на объектах, работающих в нормальных режимах, так и при организации специальных режимов для оценки интересующих исследователя характеристик (при других значениях переменных и параметров, в другом масштабе времени и т. д.). Реальное моделирование является наиболее адекватным, но при этом его возможности с учетом особенностей реальных объектов ограничены. **Например**, проведение реального моделирования АСУ предприятием потребует, во-первых, создания такой АСУ, а во-вторых, проведения экспериментов с управляемым объектом, т. е. предприятием, что в большинстве случаев невозможно. Рассмотрим разновидности реального моделирования:

9.1. **Натурное моделирование**. Натурным моделированием называют проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия. При функционировании объекта в соответствии с поставленной целью удается

выявить закономерности протекания реального процесса. Надо отметить, что такие разновидности натурного эксперимента, как производственный эксперимент и комплексные испытания, обладают высокой степенью достоверности. Натурное моделирование подразделяется на:

9.1.1. **Натурный эксперимент.** С развитием техники и проникновением в глубь процессов, протекающих в реальных системах, возрастает техническая оснащенность современного научного эксперимента. Он характеризуется широким использованием средств автоматизации проведения, применением весьма разнообразных средств обработки информации, возможностью вмешательства человека в процесс проведения эксперимента, и в соответствии с этим появилось новое научное направление — автоматизация научных экспериментов.

Отличие эксперимента от реального протекания процесса заключается в том, что в нем могут появиться отдельные критические ситуации и определяться границы устойчивости процесса. В ходе эксперимента вводятся новые факторы и возмущающие воздействия в процессе функционирования объекта.

9.1.2. **Комплексные испытания.** Комплексные испытания можно отнести к натурному моделированию, когда вследствие повторения испытаний изделий выявляются общие закономерности о надежности этих изделий, о характеристиках качества и т. д. В этом случае моделирование осуществляется путем обработки и обобщения сведений, проходящих в группе однородных явлений.

9.1.3. **Производственный эксперимент**. Наряду со специально организованными испытаниями возможна реализация натурного моделирования путем обобщения опыта, накопленного в ходе производственного процесса, т. е. можно говорить о производственном эксперименте. Здесь на базе теории подобия обрабатывают статистический материал по производственному процессу и получают его обобщенные характеристики.

9.2. **Физическое моделирование**. Отличается от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием. В процессе физического моделирования задаются некоторые характеристики внешней среды и исследуется поведение либо реального объекта, либо его модели при заданных или создаваемых искусственно воздействиях внешней среды. Физическое моделирование может протекать в масштабе:

9.2.1. **Реальное время.** Наибольшие сложность и интерес с точки зрения верности получаемых результатов представляет физическое моделирование в реальном масштабе времени.

9.2.2. **Нереальное время** (псевдореальное).

9.2.3. **Без учета времени**. В данном случае изучению подлежат так называемые «замороженные» процессы, которые фиксируются в некоторый момент времени.

С точки зрения математического описания объекта и в зависимости от его характера модели можно разделить на модели:

- **Аналоговые** (непрерывные). Под аналоговой моделью понимается модель, которая описывается уравнениями, связывающими непрерывные величины.

- **Цифровые** (дискретные). Под цифровой понимают модель, которая описывается уравнениями, связывающими дискретные величины, представленные в цифровом виде.

- **Аналого-цифровые** (комбинированные). Под аналого-цифровой понимается модель, которая может быть описана уравнениями, связывающими непрерывные и дискретные величины.

Особое место в моделировании занимает кибернетическое моделирование, в котором отсутствует непосредственное подобие физических процессов, происходящих в моделях, реальным процессам. В этом случае стремятся отобразить лишь некоторую функцию и рассматривают реальный объект как «черный ящик», имеющий ряд входов и выходов, и моделируют некоторые связи между выходами и входами. Чаще всего при использовании кибернетических моделей проводят анализ поведенческой стороны объекта при различных воздействиях внешней среды. Таким образом, в основе кибернетических моделей лежит отражение некоторых информационных процессов управления, что позволяет оценить поведение реального объекта. Для построения имитационной модели в этом случае необходимо выделить исследуемую функцию реального объекта, допытаться формализовать эту функцию в виде некоторых операторов связи между входом и выходом и воспроизвести на имитационной модели данную функцию, причем на базе совершенно иных математических соотношений и, естественно, иной физической реализации процесса.

3.2. Возможности и эффективность моделирования систем на вычислительных машинах

Обеспечение требуемых показателей качества функционирования больших систем, связанное с необходимостью изучения протекания стохастических процессов в исследуемых и проектируемых системах S, позволяет проводить комплекс теоретических и экспериментальных исследований, взаимно дополняющих друг друга. Эффективность экспериментальных исследований сложных систем оказывается крайне низкой, поскольку проведение натурных экспериментов с реальной системой либо требует больших материальных затрат и значительного времени, либо вообще практически невозможно (**например**, на этапе проектирования, когда реальная система отсутствует). Эффективность теоретических исследований

с практической точки зрения в полной мере проявляется лишь тогда, когда их результаты с требуемой степенью точности и достоверности могут быть представлены в виде аналитических соотношений или моделирующих алгоритмов, пригодных для получения соответствующих характеристик процесса функционирования исследуемых систем.

Средства моделирования систем. Появление современных ЭВМ было решающим условием широкого внедрения аналитических методов в исследование сложных систем. Стало казаться, что модели и методы, **например**, математического программирования, станут практическим инструментом решения задач управления в больших системах. Действительно, были достигнуты значительные успехи в создании новых математических методов решения этих задач, однако математическое программирование так и не стало практическим инструментом исследования процесса функционирования сложных систем, так как модели математического программирования оказались слишком грубыми и несовершенными для их эффективного использования. Необходимость учета стохастических свойств системы, недетерминированности исходной информации, наличия корреляционных связей между большим числом переменных и параметров, характеризующих процессы в системах, приводят к построению сложных математических моделей, которые не могут быть применены в инженерной практике при исследовании таких систем аналитическим методом. Пригодные для практических расчетов аналитические соотношения удается получить лишь при упрощающих предположениях, обычно существенно искажающих фактическую картину исследуемого процесса. Поэтому в последнее время все ощутимее потребность в разработке методов, которые дали бы возможность уже на этапе проектирования систем исследовать более адекватные модели. Указанные обстоятельства приводят к тому, что при исследовании больших систем все шире применяют методы имитационного моделирования.

Наиболее конструктивным средством решения инженерных задач на базе моделирования в настоящее время стали ЭВМ. Современные ЭВМ можно разделить на две группы:

- **универсальные**, прежде всего предназначенные для выполнения расчетных работ;

- **управляющие**, позволяющие проводить не только расчетные работы, но прежде всего приспособленные для управления объектами в реальном масштабе времени.

Управляющие ЭВМ могут быть использованы как для управления технологическим процессом, экспериментом, так и для реализации различных имитационных моделей. В зависимости от того, удается ли построить достаточно точную математическую модель реального процесса, или вследствие сложности объекта не удается проникнуть в глубь функциональных связей реального объекта и описать их какими-то

аналитическими соотношениями, можно рассматривать два основных пути использования ЭВМ:

- как средства расчета по полученным аналитическим моделям;
- как средства имитационного моделирования.

Для известной аналитической модели, полагая, что она достаточно точно отображает исследуемую сторону функционирования реального физического объекта, перед вычислительной машиной стоит задача расчета характеристик системы по каким-либо математическим соотношениям при подстановке числовых значений. В этом направлении вычислительные машины обладают возможностями, практически зависящими от порядка решаемого уравнения и от требований к скорости решения, причем могут быть использованы как ЭВМ, так и АВМ.

При использовании ЭВМ разрабатывается алгоритм расчета характеристик, в соответствии с которым составляются программы (либо генерируются с помощью пакета прикладных программ) дающие возможность осуществлять расчеты по требуемым аналитическим соотношениям. Основная задача исследователя заключается в том, чтобы попытаться описать поведение реального объекта одной из известных математических моделей.

Использование АВМ, с одной стороны, ускоряет для достаточно простых случаев процесс решения задачи, с другой стороны, могут возникать погрешности, обусловленные наличием дрейфа параметров отдельных блоков, входящих в АВМ, ограниченной точностью с которой могут быть заданы параметры, вводимые в машину, а также неисправностями технических средств и т. д.

Перспективно сочетание ЭВМ и АВМ, т. е. использование гибридных средств вычислительной техники — гибридных вычислительных комплексов (ГВК), что в ряде случаев значительно ускоряет процесс исследования.

В ГВК удается сочетать высокую скорость функционирования аналоговых средств и высокую точность расчетов на базе цифровых средств вычислительной техники. Одновременно удается за счет наличия цифровых устройств обеспечить контроль проведения операций. Опыт использования вычислительной техники в задачах моделирования показывает, что с усложнением объекта большую эффективность по скорости решения и по стоимости выполнения операций дает использование гибридной техники.

Конкретным техническим средством воплощения имитационной модели могут быть ЭВМ, АВМ и ГВК. Если использование аналоговой техники ускоряет получение конечных результатов, сохраняя некоторую наглядность протекания реального процесса, то применение средств цифровой техники позволяет осуществить контроль за реализацией модели, создать программы по обработке и хранению результатов моделирования, обеспечить эффективный диалог исследователя с моделью.

Обычно модель строится по иерархическому принципу, когда последовательно анализируются отдельные стороны функционирования объекта и при перемещении центра внимания исследователя, рассмотренные ранее подсистемы переходят во внешнюю среду. Иерархическая структура моделей может раскрывать и ту последовательность, в которой изучается реальный объект, а именно последовательность перехода от структурного (топологического) уровня к функциональному (алгоритмическому) и от функционального к параметрическому.

Результат моделирования в значительной степени зависит от адекватности исходной концептуальной (описательной) модели, от полученной степени подобия описания реального объекта, числа реализаций модели и многих других факторов. В ряде случаев сложность объекта не позволяет не только построить математическую модель объекта, но и дать достаточно близкое кибернетическое описание, и перспективным здесь является выделение наиболее трудно поддающейся математическому описанию части объекта и включение этой реальной части физического объекта в имитационную модель. Тогда модель реализуется, с одной стороны, на базе средств вычислительной техники, а с другой — имеется реальная часть объекта. Это значительно расширяет возможности и повышает достоверность результатов моделирования.

Имитационная система реализуется на ЭВМ и позволяет исследовать имитационную модель М, задаваемую в виде определенной совокупности отдельных блочных моделей и связей между ними в их взаимодействии в пространстве и времени при реализации какого-либо процесса. Можно выделить три основные группы блоков:

- блоки, характеризующие моделируемый процесс функционирования системы S;

- блоки, отображающие внешнюю среду E и ее воздействие на реализуемый процесс;

- блоки, играющие служебную вспомогательную роль, обеспечивая взаимодействие первых двух, а также выполняющие дополнительные функции по получению и обработке результатов моделирования.

Кроме того, имитационная система характеризуется набором переменных, с помощью которых удается управлять изучаемым процессом, и набором начальных условий, когда можно изменять условия проведения машинного эксперимента.

Таким образом, имитационная система есть средство проведения машинного эксперимента, причем эксперимент может ставиться многократно, заранее планироваться, могут определяться условия его проведения. Необходимо при этом выбрать методику оценки адекватности получаемых результатов и автоматизировать как процессы получения, так и процессы обработки результатов в ходе машинного эксперимента.

Обеспечение моделирования. Эксперимент с имитационной моделью требует серьезной подготовки, поэтому имитационная система характеризуется наличием:

- **Математического обеспечения**. Математическое обеспечение имитационной системы включает в себя совокупность математических соотношений, описывающих поведение реального объекта, совокупность алгоритмов, обеспечивающих как подготовку, так и работу с моделью. Сюда могут быть отнесены алгоритмы ввода исходных данных, имитации, вывода, обработки.

- **Программного обеспечения**. Программное обеспечение по своему содержанию включает в себя совокупность программ: планирования эксперимента, имитационной модели, проведения эксперимента, обработки и интерпретации результатов. Кроме того, программное обеспечение имитационной системы должно обеспечивать синхронизацию процессов в модели, т. е. необходим блок, организующий псевдопараллельное выполнение процессов в модели. Машинные эксперименты с имитационными моделями не могут проходить без хорошо разработанного и реализованного информационного обеспечения.

- **Информационного обеспечения**. Информационное обеспечение включает в себя средства и технологию организации и реорганизации базы данных моделирования, методы логической и физической организации массивов, формы документов, описывающих процесс моделирования и его результаты. Информационное обеспечение имитационной системы является наименее разработанной частью, поскольку только в настоящее время наблюдается переход к созданию сложных имитационных моделей и разрабатывается методология их использования при анализе и синтезе сложных систем с использованием концепции базы данных и знаний.

- **Технического обеспечения**. Техническое обеспечение имитационной системы включает в себя прежде всего средства вычислительной техники, связи и обмена между оператором и сетью ЭВМ, ввода и вывода информации, управления проведением эксперимента. К техническому обеспечению предъявляются весьма серьезные требования по надежности функционирования, так как сбои и отказы технических средств, ошибки оператора ЭВМ могут резко увеличить время работы с имитационной моделью и даже привести к неверным конечным результатам.

- **Эргономического обеспечения**. Эргономическое обеспечение имитационной системы представляет собой совокупность научных и прикладных методик и методов, а также нормативно-технических и организационно-методических документов, используемых на всех этапах взаимодействия человека-экспериментатора с инструментальными средствами (ЭВМ, гибридными комплексами и т. д.). Эти документы, используемые на всех стадиях разработки и эксплуатации имитационных систем и их элементов, предназначены для формирования и поддержания

эргономического качества путем обоснования и выбора организационно-проектных решений, которые создают оптимальные условия для высокоэффективной деятельности человека во взаимодействии с моделирующим комплексом.

- **Других видов обеспечения.**

Таким образом, имитационная система может рассматриваться как машинный аналог сложного реального процесса. Позволяет заменить эксперимент с реальным процессом функционирования системы экспериментом с математической моделью этого процесса в ЭВМ. В настоящее время имитационные эксперименты широко используют в практике проектирования сложных систем, когда реальный эксперимент невозможен.

Возможности машинного моделирования. Несмотря на то что имитационное моделирование на ЭВМ является мощным инструментом исследования систем, его применение рационально не во всех случаях. Известно множество задач, решаемых более эффективно другими методами. Вместе с тем для большого класса задач исследования и проектирования систем метод имитационного моделирования наиболее приемлем. Правильное его употребление возможно лишь в случае четкого понимания сущности метода имитационного моделирования и условий его использования в практике исследования реальных систем при учете особенностей конкретных систем и возможностей их исследования различными методами.

В качестве основных критериев целесообразности применения метода имитационного моделирования на ЭВМ можно указать следующие:

- отсутствие или неприемлемость аналитических, численных и качественных методов решения поставленной задачи;
- наличие достаточного количества исходной информации о моделируемой системе S для обеспечения возможности построения адекватной имитационной модели;
- необходимость проведения на базе других возможных методов решения очень большого количества вычислений, трудно реализуемых даже с использованием ЭВМ;
- возможность поиска оптимального варианта системы при ее моделировании на ЭВМ.

Имитационное моделирование на ЭВМ, как и любой метод исследований, имеет достоинства и недостатки, проявляющиеся в конкретных приложениях. К числу основных достоинств метода имитационного моделирования при исследовании сложных систем можно отнести следующие:

- машинный эксперимент с имитационной моделью дает возможность исследовать особенности процесса функционирования системы S в любых условиях;

- применение ЭВМ в имитационном эксперименте существенно сокращает продолжительность испытаний по сравнению с натурным экспериментом;

- имитационная модель позволяет включать результаты натурных испытаний реальной системы или ее частей для проведения дальнейших исследований;

- имитационная модель обладает известной гибкостью варьирования структуры, алгоритмов и параметров моделируемой системы, что важно с точки зрения поиска оптимального варианта системы;

- имитационное моделирование сложных систем часто является единственным практически реализуемым методом исследования процесса функционирования таких систем на этапе их проектирования.

Основным недостатком, проявляющимся при машинной реализации метода имитационного моделирования, является то, что решение, полученное при анализе имитационной модели М, всегда носит частный характер, так как оно соответствует фиксированным элементам структуры, алгоритмам поведения и значениям параметров системы S, начальных условий и воздействий внешней среды Е. Поэтому для полного анализа характеристик процесса функционирования систем, а не получения только отдельной точки приходится многократно воспроизводить имитационный эксперимент, варьируя исходные данные задачи. При этом, как следствие, возникает увеличение затрат машинного времени на проведение эксперимента с имитационной моделью процесса функционирования исследуемой системы S.

Эффективность машинного моделирования. При имитационном моделировании, так же как и при любом другом методе анализа и синтеза системы S, весьма существен вопрос его эффективности. Эффективность имитационного моделирования может оцениваться рядом критериев, в том числе точностью и достоверностью результатов моделирования, временем построения и работы с моделью М, затратами машинных ресурсов (времени и памяти), стоимостью разработки и эксплуатации модели. Очевидно, наилучшей оценкой эффективности является сравнение получаемых результатов с реальным исследованием, т. е. с моделированием на реальном объекте при проведении натурного эксперимента. Поскольку это не всегда удается сделать, статистический подход позволяет с определенной степенью точности при повторяемости машинного эксперимента получить какие-то усредненные характеристики поведения системы. Существенное влияние на точность моделирования оказывает число реализаций, и в зависимости от требуемой достоверности можно оценить необходимое число реализаций воспроизводимого случайного процесса.

Существенным показателем эффективности являются затраты машинного времени. В связи с использованием ЭВМ различного типа суммарные затраты складываются из времени по вводу и выводу данных по

каждому алгоритму моделирования, времени на проведение вычислительных операций, с учетом обращения к оперативной памяти и внешним устройствам, а также сложности каждого моделирующего алгоритма. Расчеты затрат машинного времени являются приближенными и могут уточняться по мере отладки программ и накопления опыта у исследователя при работе с имитационной моделью. Большое влияние на затраты машинного времени при проведении имитационных экспериментов оказывает рациональное планирование таких экспериментов. Определенное влияние на затраты машинного времени могут оказать процедуры обработки результатов моделирования, а также форма их представления.

Построение имитационных моделей больших систем и проведение машинных экспериментов с этими моделями представляют собой достаточно трудоемкий процесс, в котором в настоящее время много неизученного. Однако специалисты в области проектирования, исследования и эксплуатации больших систем должны в совершенстве знать методологию машинного моделирования, сложившуюся к настоящему времени, чтобы быть готовыми к появлению ЭВМ следующих поколений, которые позволят сделать еще один существенный шаг в автоматизации построения моделей и использования имитационного моделирования систем.

Вопросы для повторения и закрепления материала

1. Укажите особенности детерминированных систем.
2. Укажите особенности стохастических систем.
3. Укажите особенности статических систем.
4. Укажите особенности динамических систем.
5. Укажите особенности дискретных систем.
6. Укажите особенности непрерывных систем.
7. В чем заключается суть имитационного моделирования?
8. Каким обеспечением должна обладать имитационная система?

Задания для самостоятельной работы

1. Рассмотреть классификации видов моделирования отличные от указанной в курсе.

Тема 4. Математические схемы моделирования систем. Непрерывно-детерминированные модели (D-схемы)

Цель изучения темы: рассмотреть основные виды математических схем моделирования.

Задачи изучения темы:

- рассмотреть основные подходы к построению математических моделей систем;
- рассмотреть непрерывно-детерминированные модели;
- рассмотреть возможные приложения D-схем.

4.1. Основные подходы к построению математических моделей систем

Наибольшие затруднения и наиболее серьезные ошибки при моделировании возникают при переходе от содержательного к формальному описанию объектов исследования, что объясняется участием в этом творческом процессе коллективов разных специальностей: специалистов в области систем и специалистов в области машинного моделирования. Эффективным средством для нахождения взаимопонимания между этими группами специалистов является язык математических схем, позволяющий во главу угла поставить вопрос об адекватности перехода от содержательного описания системы к ее математической схеме, а лишь затем решать вопрос о конкретном методе получения результатов с использованием ЭВМ: аналитическом или имитационном, а возможно, и комбинированном, т. е. аналитико-имитационном. Применительно к конкретному объекту моделирования, т. е. к сложной системе, разработчику модели должны помочь конкретные, уже прошедшие апробацию для данного класса систем математические схемы, показавшие свою эффективность в прикладных исследованиях на ЭВМ и получившие название типовых математических схем.

Исходной информацией при построении математических моделей процессов функционирования систем служат данные о назначении и условиях работы, исследуемой (проектируемой) системы. Эта информация определяет основную цель моделирования системы S и позволяет сформулировать требования к разрабатываемой математической модели М. Причем уровень абстрагирования зависит от круга тех вопросов, на которые исследователь системы хочет получить ответ с помощью модели, и в какой-то степени определяет выбор математической, схемы.

Математические схемы. Введение понятия «математическая схема» позволяет рассматривать математику не как метод расчета, а как метод мышления, как средство формулирования понятий, что является наиболее важным при переходе от словесного описания системы к формальному представлению процесса ее функционирования в виде некоторой математической модели (аналитической или имитационной). При пользовании математической схемы исследователя системы S в первую очередь должен интересовать вопрос об адекватности отображения в виде конкретных схем реальных процессов в исследуемой системе, а не возможность получения ответа (результата решения) на конкретный вопрос исследования. **Например**, представление процесса функционирования информационно-вычислительной системы коллективного пользования в виде сети схем массового обслуживания дает возможность хорошо описать процессы, происходящие в системе, но при сложных законах распределения входящих потоков и потоков обслуживания не дает возможности получения результатов в явном виде.

Математическую схему можно определить, как звено при переходе от содержательного к формальному описанию процесса функционирования системы с учетом воздействия внешней среды, т. е. имеет место цепочка «описательная модель — математическая схема — математическая [аналитическая или (и) имитационная] модель».

Каждая конкретная система характеризуется набором свойств, под которыми понимаются величины, отражающие поведение моделируемого объекта (реальной системы) и учитывающие условия ее функционирования во взаимодействии с внешней средой (системой) Е. При построении математической модели системы необходимо решить вопрос об ее полноте. Полнота модели регулируется в основном выбором границы «система S—среда Е». Также должна быть решена задача упрощения модели, которая помогает выделить основные свойства системы, отбросив второстепенные. Причем отнесение свойств системы к основным или второстепенным существенно зависит от цели моделирования системы (**например**, анализ вероятностно-временных характеристик процесса функционирования системы, синтез структуры системы и т. д.).

Формальная модель объекта. Модель объекта моделирования, т. е. системы S, можно представить в виде множества величин, описывающих процесс функционирования реальной системы и образующих в общем случае следующие подмножества:

- совокупность входных воздействий на систему - $x_i \in X,\ i=\overline{1,\ n_X};$

- совокупность воздействий внешней среды $v_l \in V,\ l=\overline{1,\ n_V};$

- совокупность внутренних (собственных) параметров системы $h_k \in H,\ k=\overline{1,\ n_H};$

- совокупность выходных характеристик системы $y_j \in Y,\ j=\overline{1,\ n_Y}.$

При этом в перечисленных подмножествах можно выделить управляемые и неуправляемые переменные. В общем случае x_i v_l h_k, y_j являются элементами непересекающихся подмножеств и содержат как детерминированные, так и стохастические составляющие.

При моделировании системы S входные воздействия, воздействия внешней среды Е и внутренние параметры системы являются **независимыми (экзогенными) переменными**, которые в векторной форме имеют соответственно вид $\vec{x}(t)=(x_1(t),x_2(t),\ldots,x_{nX}(t))$, $\vec{v}(t)=(v_1(t),v_2(t),\ldots,v_{nV}(t))$, $\vec{h}(t)=(h_1(t),h_2(t),\ldots,h_{nH}(t))$, а выходные характеристики системы являются

зависимыми (эндогенными) переменными и в векторной форме имеют вид $\vec{y}(t) = (y_1(t), y_2(t), \ldots, y_{nY}(t))$.

Процесс функционирования системы S описывается во времени оператором Fs, который в общем случае преобразует экзогенные переменные в эндогенные в соответствии с соотношениями вида:

$$\vec{y}(t) = F_S(\vec{x}, \vec{v}, \vec{h}, t) \qquad (4.1)$$

Совокупность зависимостей выходных характеристик системы от времени $y_j(t)$ для всех видов $j = \overline{1, n_Y}$ называется **выходной траекторией** $\vec{y}(t)$. Зависимость (4.1) называется **законом функционирования системы** S и обозначается Fs. В общем случае закон функционирования системы Fs может быть задан в виде функции, функционала, логических условий, в алгоритмической и табличной формах или в виде словесного правила соответствия.

Весьма важным для описания и исследования системы S является понятие **алгоритма функционирования** As, под которым понимается метод получения выходных характеристик с учетом входных воздействий $\vec{x}(t)$, внешней среды $\vec{v}(t)$ и собственных параметров системы $\vec{h}(t)$. Очевидно, что один и тот же закон функционирования Fs системы S может быть реализован различными способами, т. е. с помощью множества различных алгоритмов функционирования As.

Соотношение (4.1) является математическим описанием поведения объекта (системы) моделирования во времени t, т. е. отражает его динамические свойства. Поэтому математические модели такого вида принято называть **динамическими моделями** (системами).

Для статических моделей математическая модель (4.1) представляет собой отображение между двумя подмножествами свойств моделируемого объекта Y и {X, V, H}, что в векторной форме может быть записано как:

$$\vec{y}(t) = f(\vec{x}, \vec{v}, \vec{h}) \qquad (4.2)$$

Соотношения (4.1) и (4.2) могут быть заданы различными способами:

- аналитически (с помощью формул);
- графически;
- таблично;
- и т. д.

Такие соотношения в ряде случаев могут быть получены через свойства системы S в конкретные моменты времени, называемые состояниями. Состояние системы S характеризуется векторами:

$$\vec{z}' = (z'_1, z'_2, \ldots, z'_k) \text{ и } z'' = (\vec{z}''_1, \vec{z}''_2, \ldots, z''_k),$$

где $z'_1 = z_1(t')$, $z'_2 = z_2(t')$, ..., $z'_k = z_k(t')$ в момент $t'' \in (t_0, T)$; $z''_1 = z_1(t'')$, $z''_2 = z_2(t'')$, ..., $z''_k = z_k(t'')$ в момент $t'' \in (t_0, T)$ и т. д., $k = \overline{1, n_Z}$.

Если рассматривать процесс функционирования системы S как последовательную смену состояний $z_1(t)$, $z_2(t)$, ..., $z_k(t)$, то они могут быть интерпретированы как координаты точки в k -мерном фазовом пространстве, причем каждой реализации процесса будет соответствовать некоторая фазовая траектория. Совокупность всех возможных значений состояний { z } называется пространством состояний объекта моделирования Z, причем $z_K \in Z$.

Состояния системы S в момент времени $t_0 < t^* < T$ полностью определяются начальными условиями $z = (z_1, z_2, ..., z_k)$ [где $z_1=z_l(t_0)$, $z_2=z_2(t_0)$, ..., $z_k=z_k(t_0)$], входными воздействиями x(t), внутренними параметрами h(t) и воздействиями внешней среды v(t), которые имели место за промежуток времени t* —10, с помощью двух векторных уравнений

$$z(t) = \Phi(\vec{z}^0, \vec{x}, \vec{v}, \vec{h}, t) \qquad (4.3)$$

$$\vec{y}(t) = F(\vec{z}, t) \qquad (4.4)$$

Первое уравнение по начальному состоянию z° и экзогенным переменным x,v,h определяет вектор-функцию z(t), а второе по полученному значению состояний z (t) — эндогенные переменные на выходе системы y (t). Таким образом, цепочка уравнений объекта «вход — состояния — выход» позволяет определить характеристики системы

$$\vec{y}(t) = F[\Phi(\vec{z}^0, \vec{x}, \vec{v}, \vec{h}, t)]. \qquad (4.5)$$

В общем случае время в модели системы S может рассматриваться на интервале моделирования (O, T) как непрерывное, так и дискретное, т. е. квантованное на отрезки длиной Δt временных единиц каждый, когда $T=m\Delta t$, где m=l, mT — число интервалов дискретизации.

Таким образом, под математической моделью объекта (реальной системы) понимают конечное подмножество переменных {x (t), v (t), h (t)} вместе с математическими связями между ними и характеристиками y (t).

Если математическое описание объекта моделирования не содержит элементов случайности или они не учитываются, т. е. если можно считать, что в этом случае стохастические воздействия внешней среды v (t) и стохастические внутренние параметры h (t) отсутствуют, то модель называется **детерминированной** в том смысле, что характеристики однозначно определяются детерминированными входными воздействиями

$$\vec{y}(t) = f(\vec{x}, t). \qquad (4.6)$$

Очевидно, что детерминированная модель является частным случаем стохастической модели.

Типовые схемы. Приведенные математические соотношения представляют собой математические схемы общего вида и позволяют описать широкий класс систем. Однако в практике моделирования объектов в области системотехники и системного анализа на первоначальных этапах исследования системы рациональнее использовать типовые математические схемы:

- дифференциальные уравнения;
- конечные и вероятностные автоматы;
- системы массового обслуживания;
- сети Петри
- и т. д.

Не обладая такой степенью общности, как рассмотренные модели, типовые математические схемы имеют преимущества простоты и наглядности, но при существенном сужении возможностей применения. В качестве детерминированных моделей, когда при исследовании случайные факторы не учитываются, для представления систем, функционирующих в непрерывном времени, используются дифференциальные, интегральные, интегро-дифференциальные и другие уравнения, а для представления систем, функционирующих в дискретном времени,— конечные автоматы и конечно-разностные схемы. В качестве стохастических моделей (при учете случайных факторов) для представления систем с дискретным временем используются вероятностные автоматы, а для представления системы с непрерывным временем — системы массового обслуживания и т. д.

Перечисленные типовые математические схемы, естественно, не могут претендовать на возможность описания на их базе всех процессов, происходящих в больших информационно-управляющих системах. Для таких систем в ряде случаев более перспективным является применение агрегативных моделей. **Агрегативные модели** (системы) позволяют описать широкий круг объектов исследования с отображением системного характера этих объектов. Именно при агрегативном описании сложный объект (система) расчленяется на конечное число частей (подсистем), сохраняя при этом связи, обеспечивающие взаимодействие частей.

Таким образом, при построении математических моделей процессов функционирования систем можно выделить следующие основные подходы:

- непрерывно-детерминированный (например, дифференциальные уравнения);
- дискретно-детерминированный (конечные автоматы);
- дискретно-стохастический (вероятностные автоматы);
- непрерывно-стохастический (системы массового обслуживания);
- обобщенный, или универсальный (агрегативные системы).

4.2. Непрерывно-детерминированные модели (D-схемы)

Рассмотрим особенности непрерывно-детерминированного подхода на примере использования в качестве математических моделей дифференциальных уравнений. Дифференциальными уравнениями называются такие уравнения, в которых неизвестными будут функции одной или нескольких переменных, причем в уравнение входят не только функции, но и их производные различных порядков. Если неизвестные — функции

многих переменных, то уравнения называются **уравнениями в частных производных**, в противном случае при рассмотрении функции только одной независимой переменной уравнения называются **обыкновенными дифференциальными уравнениями**.

Основные соотношения. Обычно в таких математических моделях в качестве независимой переменной, от которой зависят неизвестные искомые функции, служит время t. Тогда математическое соотношение для детерминированных систем (4.6) в общем виде будет

$$\vec{y}'(t) = \vec{f}(\vec{y}, t); \quad \vec{y}(t_0) = \vec{y}_0, \qquad (4.7)$$

где $\vec{y}' = d\vec{y}/dt$, $\vec{y} = (y_1, y_2, \ldots, y_n)$ и $\vec{f} = (f_1, f_2, \ldots, f_n)$ — n-мерные

векторы; f(y, t) — вектор-функция, которая определена на некотором (n+ 1)-мерном (y, t) множестве и является непрерывной.

Так как математические схемы такого вида отражают динамику изучаемой системы, т. е. ее поведение во времени, то они называются **D-схемами** (англ. dynamic).

В простейшем случае обыкновенное дифференциальное уравнение имеет вид

$$y'(t) = f(y, t). \qquad (4.8)$$

Наиболее важно применение D-схем в качестве математического аппарата в теории автоматического управления. Для иллюстрации особенностей построения и применения D-схем рассмотрим **пример** формализации процесса функционирования механической системы SM колебания маятника (рисунок 4.1).

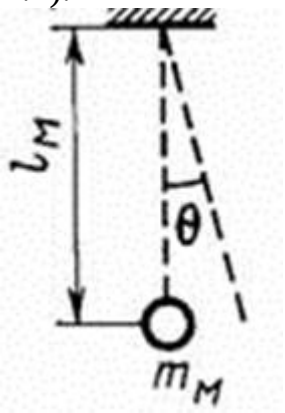


Рисунок 4.1 – Колебания маятника

Процесс малых колебаний маятника описывается обыкновенным дифференциальным уравнением:

$$m_м l_м^2 [d^2\theta(t)/dt^2] + m_м g l_м \theta(t) = 0,$$

где m_M l_M — масса и длина подвеса маятника;

g — ускорение свободного падения;

θ(t) — угол отклонения маятника в момент времени t.

Из этого уравнения свободного колебания маятника можно найти оценки интересующих характеристик. **Например**, период колебания маятника:

$$T_м = 2\pi\sqrt{l_м/g}.$$

4.3. Возможные приложения D-схем.

При решении задач системотехники важное значение имеют проблемы управления большими системами. Следует обратить внимание на системы автоматического управления — частный случай динамических систем, описываемых D-схемами и выделенных в отдельный класс моделей в силу их практической специфики.

Описывая процессы автоматического управления, придерживаются обычно представления реального объекта в виде двух систем: управляющей и управляемой (объекта управления). Структура многомерной системы автоматического управления общего вида представлена на рисунке 4.2, где обозначены эндогенные переменные:

- x(t) — вектор входных (задающих) воздействий;
- v(t) — вектор возмущающих воздействий;
- h'(t) — вектор сигналов ошибки;
- h''(t) — вектор управляющих воздействий;

А также экзогенные переменные:

- z — вектор состояний системы S;
- y(t) — вектор выходных переменных, обычно y (t)=z (t).

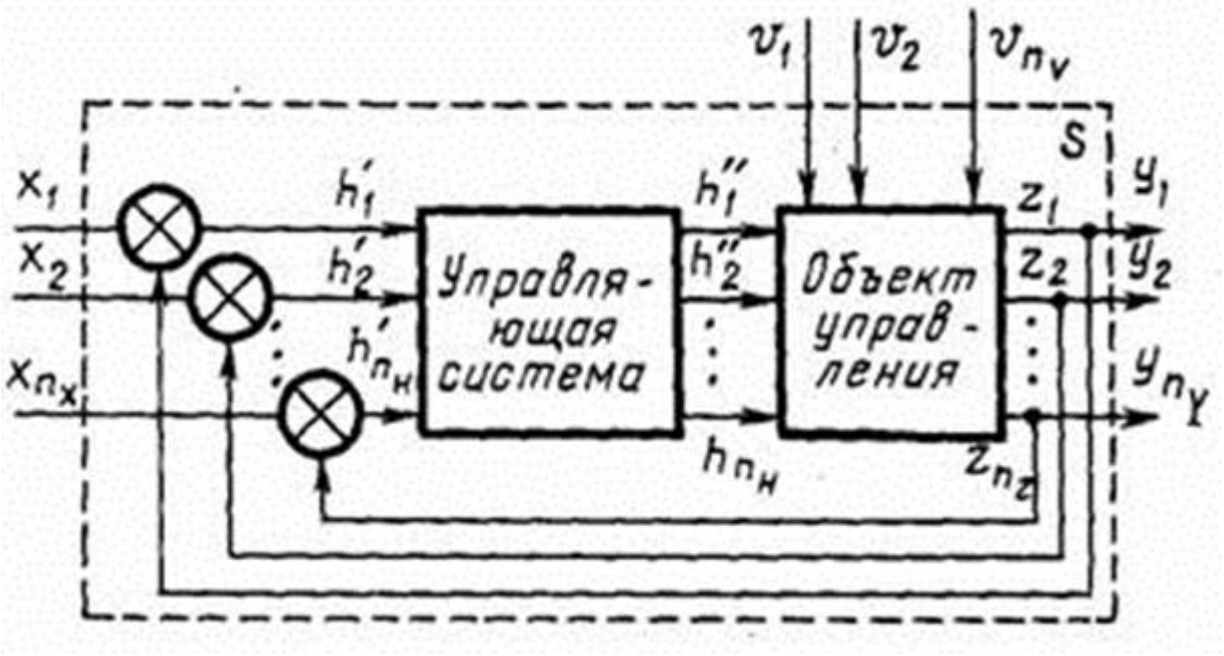


Рисунок 4.2 – Структура системы автоматического управления

Современная управляющая система — это совокупность программно-технических средств, обеспечивающих достижение объектом управления

определенной цели. Насколько точно объект управления достигает заданной цели, можно судить для одномерной системы по координате состояния y(t). Разность между заданным $y_{зад}(t)$ и действительным y(t) законами изменения управляемой величины есть ошибка управления $h'(t)=y_{зад}(t)—y(t)$. Если предписанный закон изменения управляемой величины соответствует закону изменения входного (задающего) воздействия, т. е. $x(t)=y_{зад}(t)$, то $h'(t)=x(t)-y(t)$.

Системы, для которых ошибки управления h'(t) = O во все моменты времени, называются **идеальными**. На практике реализация идеальных систем невозможна. Таким образом, ошибка h’ (t) — необходимый субстрат автоматического управления, основанного на принципе отрицательной обратной связи, так как для приведения в соответствие выходной переменной y(t) ее заданному значению используется информация об отклонении между ними. Задачей системы автоматического управления является изменение переменной y(t) согласно заданному закону с определенной точностью (с допустимой ошибкой). При проектировании и эксплуатации систем автоматического управления необходимо выбрать такие параметры системы S, которые обеспечили бы требуемую точность управления, а также устойчивость системы в переходном процессе.

Если система устойчива, то представляют практический интерес поведение системы во времени, максимальное отклонение регулируемой переменной y(t) в переходном процессе, время переходного процесса и т. п. Выводы о свойствах систем автоматического управления различных классов можно сделать по виду дифференциальных уравнений, приближенно описывающих процессы в системах. Порядок дифференциального уравнения и значения его коэффициентов полностью определяются статическими и динамическими параметрами системы S.

Таким образом, использование D-схем позволяет формализовать процесс функционирования непрерывно-детерминированных систем S и оценить их основные характеристики, применяя аналитический или имитационный подход, реализованный в виде соответствующего языка для моделирования непрерывных систем или использующий аналоговые и гибридные средства вычислительной техники.

Вопросы для повторения и закрепления материала

1. Что представляет собой совокупность входных воздействий на систему?
2. Что представляет собой совокупность выходных характеристик системы?
3. Что представляет собой совокупность возмущающих воздействий на систему?
4. Что представляет собой совокупность внутренних состояний системы?
5. Какие характеристики относятся к эндогенным?

6. Какие характеристики относятся к экзогенным?

7. Что такое выходная траектория?

8. Что подразумевается под законом функционирования системы?

9. Какими особенностями обладают непрерывно-детерминированные модели?

10. Назовите возможные приложения D-схем.

Задания для самостоятельной работы

1. Рассмотреть области применения D-схем с учетом специфики предмета исследования.

Тема 5. Дискретно-детерминированные модели (F-схемы)

Цель изучения темы: изучить дискретно-детерминированные модели.

Задачи изучения темы:

- рассмотреть основные соотношения дискретно-детерминированных моделей;
- рассмотреть возможные приложения F-схем.

5.1. Основные соотношения дискретно-детерминированных моделей (F-схемы)

Особенности дискретно-детерминированного подхода на этапе формализации процесса функционирования систем рассмотрим на примере использования в качестве математического аппарата теории автоматов. **Теория автоматов** — это раздел теоретической кибернетики, в котором изучаются математические модели — автоматы. На основе этой теории система представляется в виде автомата, перерабатывающего дискретную информацию и меняющего свои внутренние состояния лишь в допустимые моменты времени. Понятие «автомат» варьируется в зависимости от характера конкретно изучаемых систем, от принятого уровня абстракции и целесообразной степени общности.

Основные соотношения. Автомат можно представить, как некоторое устройство (черный ящик), на которое подаются входные сигналы и снимаются выходные и которое может иметь некоторые внутренние состояния. Конечным автоматом называется автомат, у которого множество внутренних состояний и входных сигналов (а следовательно, и множество выходных сигналов) являются конечными множествами.

Абстрактно конечный автомат (англ. finite automata) можно представить, как математическую схему (F-схему), характеризующуюся шестью элементами:

- конечным множеством X входных сигналов (входным алфавитом);
- конечным множеством Y выходных сигналов (выходным алфавитом);

- конечным множеством Z внутренних состояний (внутренним алфавитом или алфавитом состояний);

- начальным состоянием z_0, где $z_0 \in Z$;

- функцией переходов φ(z, x);

- функцией выходов ψ(z, x).

Автомат, задаваемый F-схемой: F=(Z, X, Y, φ, ψ z_o) — функционирует в дискретном автоматном времени, моментами которого являются такты, т. е, примыкающие друг к другу равные интервалы времени, каждому из которых соответствуют постоянные значения входного и выходного сигналов и внутренние состояния. Обозначим состояние, а также входной и выходной сигналы, соответствующие t-му такту при t=0, 1, 2, ..., через z(t) x(t), y(t). При этом по условию, $z(0)=z_o$, а $z(t) \in Z$, $x(t) \in X$.

Абстрактный конечный автомат имеет один входной и один выходной каналы. В каждый момент t = 0, 1, 2, ... дискретного времени F-автомат находится в определенном состоянии z(t) из множества Z состояний автомата, причем в начальный момент времени t=0 он всегда находится в начальном состоянии $z(0)=z_o$. В момент t, будучи в состоянии z(t), автомат способен воспринять на входном канале сигнал $x(t) \in X$ и выдать на выходном канале сигнал y(t) = ψ[z (t), x (t)], переходя в состояние z (t+1) = φ[z (t), x (t)], $z(t) \in Z$, $y(t) \in Y$. Абстрактный конечный автомат реализует некоторое отображение множества слов входного алфавита X на множество слов выходного алфавита Y. Другими словами, если на вход конечного автомата, установленного в начальное состояние z_0, подавать в некоторой последовательности буквы входного алфавита y(0), y(1), x(2),..., т. е. входное слово, то на выходе автомата будут последовательно появляться буквы выходного алфавита y(0), y(1), y{2),..., образуя выходное слово.

Таким образом, работа конечного автомата происходит по следующей схеме: в каждом t-м такте на вход автомата, находящегося в состоянии z(t), подается некоторый сигнал x(t), на который он реагирует переходом в (t+1)-м такте в новое состояние z(t+1) с выдачей некоторого выходного сигнала. Сказанное выше можно описать следующими уравнениями:

- для F-автомата первого рода, называемого также автоматом Мили

$$z(t+1)=\varphi[z(t), x(t)], \; t=0, 1, 2, \ldots; \tag{5.1}$$

$$y(t)=\psi[z(t), x(t)], \; t=0, 1, 2, \ldots; \tag{5.2}$$

- для F-автомата второго рода

$$z(t+1)=\varphi[z(t), x(t)], \; t=0, 1, 2, \ldots; \tag{5.3}$$

$$y(t)=\psi[z(t), x(t-1)], \; t=1, 2, 3, \ldots \tag{5.4}$$

Автомат второго рода, для которого

$$y(t)=\psi[z(t)],\ t=0,\ 1,\ 2,\ \ldots, \tag{5.5}$$

т. е. функция выходов не зависит от входной переменной x(t), называется автоматом Мура.

По числу состояний различают конечные автоматы с памятью и без памяти. **Автоматы с памятью** имеют более одного состояния, а **автоматы без памяти** (комбинационные или логические схемы) обладают лишь одним состоянием. При этом, согласно (5.2), работа комбинационной схемы заключается в том, что она ставит в соответствие каждому входному сигналу x(t) определенный выходной сигнал y(t), т. е. реализует логическую функцию вида

$$y(t)=\psi[x(t)],\ t=0,\ 1,\ 2,\ \ldots\ .$$

Эта функция называется булевой, если алфавиты X и Y, которым принадлежат значения сигналов хну, состоят из двух букв.

По характеру отсчета дискретного времени конечные автоматы делятся на:

- **Синхронные**. В синхронных F-автоматах моменты времени, в которые автомат «считывает» входные сигналы, определяются принудительно синхронизирующими сигналами. После очередного синхронизирующего сигнала с учетом «считанного» и в соответствии с уравнениями (2.13) — (2.17) происходит переход в новое состояние и выдача сигнала на выходе, после чего автомат может воспринимать следующее значение входного сигнала. Таким образом, реакция автомата на каждое значение входного сигнала заканчивается за один такт, длительность которого определяется интервалом между соседними синхронизирующими сигналами.

- **Асинхронные**. Асинхронный F-автомат считывает входной сигнал непрерывно, и поэтому, реагируя на достаточно длинный входной сигнал постоянной величины x, он может, как следует из (5.1) — (5.5), несколько раз изменять состояние, выдавая соответствующее число выходных сигналов, пока не перейдет в устойчивое, которое уже не может быть изменено данным входным сигналом.

5.2. Возможные приложения F-схем

Чтобы задать конечный F-автомат, необходимо описать все элементы множества $F=\langle Z, X, Y, \varphi, \psi, z_0\rangle$, т. е. входной, внутренний и выходной алфавиты, а также функции переходов и выходов, причем среди множества состояний необходимо выделить состояние z_0, в котором автомат находился в момент времени t=0. Существует несколько способов задания работы F-автоматов, но наиболее часто используются табличный, графический и матричный.

Простейший табличный способ задания конечного автомата основан на использовании таблиц переходов и выходов, строки которых соответствуют входным сигналам автомата, а столбцы — его состояниям. При этом обычно первый слева столбец соответствует начальному состоянию z_0. На пересечений i- й строги и k-го столбца таблицы переходов помещается соответствующее значение $\psi(z_k, x_i)$ функции переходов, а в таблице выходов — соответствующее значение $\psi(z_к,x_i,)$ функции выходов. Для F-автомата Мура обе таблицы можно совместить, получив так называемую отмеченную таблицу переходов, в которой над каждым состоянием z_k автомата, обозначающим столбец таблицы, стоит соответствующий этому состоянию, согласно (5.5), выходной сигнал $\psi(z_i)$.

Описание работы F-автомата Мили таблицами переходов φ и выходов ψ иллюстрируется таблицей 5.1.

Таблица 5.1 – Описание автомата Мили

x_i	z_k			
	z_0	z_1	...	z_k
	Переходы			
x_1	$\varphi(z_0, x_1)$	$\varphi(z_1, x_1)$	...	$\varphi(z_k, x_1)$
x_2	$\varphi(z_0, x_2)$	$\varphi(z_1, x_2)$	...	...
...	...	...	...	$\varphi(z_k, x_2)$
x_l	$\varphi(z_0, x_l)$	$\varphi(z_1, x_l)$	...	$\varphi(z_k, x_l)$
	Выходы			
x_1	$\psi(z_0, x_1)$	$\psi(z_1, x_1)$	...	$\psi(z_k, x_1)$
x_2	$\psi(z_0, x_2)$	$\psi(z_1, x_2)$	...	$\psi(z_k, x_2)$
...	...	...	...	...
x_l	$\psi(z_0, x_l)$	$\psi(z_1, x_l)$	...	$\psi(z_k, x_l)$

Описание F-автомата Мура — таблицей переходов (таблица 5.2).

Таблица 5.2 – Описание автомата Мура

x_i	$\psi(z_k)$			
	$\psi(z_0)$	$\psi(z_1)$	...	$\psi(z_k)$
	z_0	z_1	...	z_k
x_1	$\varphi(z_0, x_1)$	$\varphi(z_1, x_1)$	...	$\varphi(z_k, x_1)$
x_2	$\varphi(z_0, x_2)$	$\varphi(z_1, x_2)$	...	$\varphi(z_k, x_2)$
...	...	...	...	...
x_l	$\varphi(z_0, x_l)$	$\varphi(z_1, x_l)$	...	$\varphi(z_k, x_l)$

При другом способе задания конечного автомата используется понятие направленного графа. Граф автомата представляет собой набор вершин, соответствующих различным состояниям автомата и соединяющих вершины дуг графа, соответствующих тем или иным переходам автомата. Если входной сигнал $x_к$ вызывает переход из состояния z_i, в состояние z_j, то на графе автомата дуга, соединяющая вершину z_i с вершиной z_j обозначается $x_к$. Для того чтобы задать функцию выходов, дуги графа необходимо отметить соответствующими выходными сигналами. Для автоматов Мили эта разметка производится так: если входной сигнал $x_к$ действует на состояние z_i, то,

согласно сказанному, получается дуга, исходящая из z_i и помеченная $x_к$; эту дугу дополнительно отмечают выходной сигналом $y=\psi(z_i, x_к)$. Для автомата Мура аналогичная разметка графа такова: если входной сигнал $x_к$, действуя на некоторое состояние автомата, вызывает переход в состояние z_i, то дугу, направленную в z_i и помеченную $x_к$. дополнительно отмечают выходным сигналом $y=\psi(z_j, x_k)$.

На рисунке 5.3, а, б приведены заданные ранее таблицами F-автоматы Мили F1 и Мура F2 соответственно.

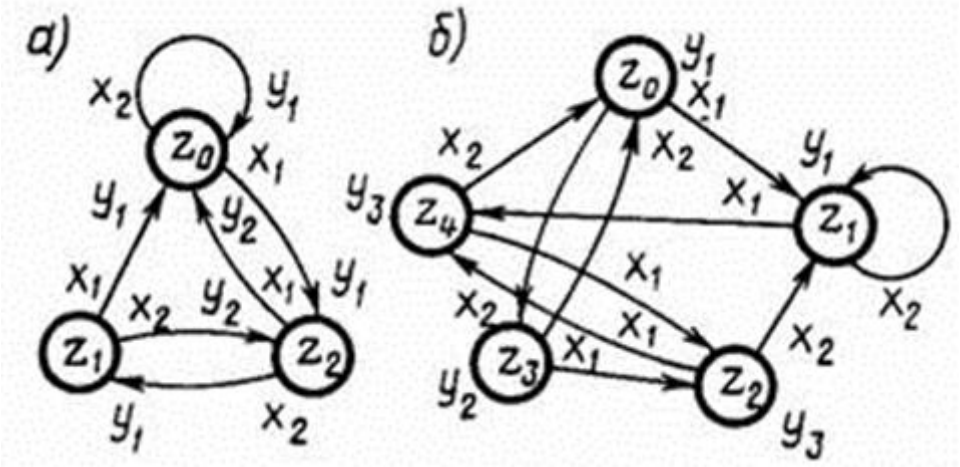


Рисунок 5.3 – Графы автоматов Мили (а) и Мура (б)

При решении задач моделирования систем часто более удобной формой является матричное задание конечного автомата. При этом матрица соединений автомата есть квадратная матрица $\mathbf{C}=\|c_{ij}\|$, строки которой соответствуют исходным состояниям, а столбцы - состояниям перехода. Элемент $c_{ij}= x_k/y_s$, стоящий на пересечении i-й строки и j-го столбца, в случае автомата Мили соответствует входному сигналу x_k, вызывающему переход из состояния z_j, в состояние z_j, и выходному сигналу y_s, выдаваемому при этом переходе. Для автомата Мили F1, рассмотренного выше, матрица соединений имеет вид

$$\mathbf{C}_1=\begin{Vmatrix} x_2y_1 & — & x_1y_1 \\ x_1y_1 & — & x_2/y_2 \\ x_1/y_2 & x_2/y_1 & — \end{Vmatrix}.$$

Если переход из состояния z_i в состояние z_j происходит под действием нескольких сигналов, элемент матрицы c_{ij} представляет собой множество пар «вход-выход» для этого перехода, соединенных знаком дизъюнкции.

Для F-автомата Мура элемент c_{ij} равен множеству входных сигналов на переходе (z_i, z_j), а выход описывается вектором выходов

$$\vec{y} = \begin{Vmatrix} \psi(z_0) \\ \psi(z_1) \\ \cdots \\ \psi(z_k) \\ \cdots \\ \psi(z_K) \end{Vmatrix},$$

Где i-я компонента которого — выходной сигнал, отмечающий состояние Z.

Пример. Для рассмотренного выше F-автомата Мура F2 запишем матрицу соединений и вектор выходов:

$$C_2 = \begin{Vmatrix} - & x_1 & - & x_2 & - \\ - & x_2 & - & - & x_1 \\ - & x_2 & - & - & x_1 \\ x_2 & - & x_1 & - & - \\ x_2 & - & x_1 & - & - \end{Vmatrix}; \quad \vec{y} = \begin{Vmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{Vmatrix}.$$

Для детерминированных автоматов выполняется условие однозначности переходов: автомат, находящийся в некотором состоянии, под действием любого входного сигнала не может перейти более чем в одно состояние. Применительно к графическому способу задания F-автомата это означает, что в графе автомата из любой вершины не могут выходить два ребра и более, отмеченные одним и тем же входным сигналом. Аналогично этому в матрице соединений автомата С в каждой строке любой входной сигнал не должен встречаться более одного раза.

Рассмотрим вид таблицы переходов (таблица 5.3) и графа (рисунок 5.4) асинхронного конечного автомата. Для F-автомата состояние z_k называется устойчивым, если для любого входа $x_i \in X$, для которого $\varphi\{z_k, x_i)=z_k$, имеет место $\psi(z_к, x_i)=y_к$. Таким образом, .F-автомат называется асинхронным, если каждое его состояние $z_k \in Z$ устойчиво.

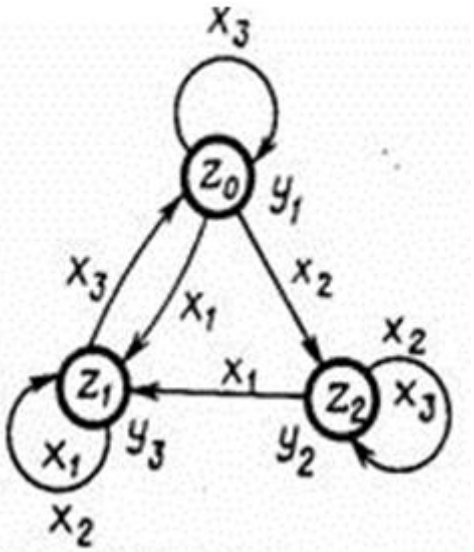


Рисунок 5.4 – Граф асинхронного автомата Мура

Необходимо отметить, что вообще на практике автоматы всегда являются асинхронными, а устойчивость их состояний обеспечивается тем или иным способом, например введением сигналов синхронизации. Однако на уровне абстрактной теории, когда конечный автомат выступает в виде математической схемы для формализации конкретных объектов без учета ряда второстепенных особенностей, часто удобно оказывается оперировать с синхронными конечными автоматами.

Таблица 5.3 – Переходы асинхронного конечного автомата

x_i	y		
	y_1	y_2	y_3
	z_0	z_1	z_2
x_1	z_1	z_1	z_1
x_2	z_2	z_1	z_2
x_3	z_0	z_0	z_2

Таким образом, понятие F-автомата в дискретно-детерминированном подходе к исследованию на моделях свойств объектов является математической абстракцией, удобной для описания широкого класса процессов функционирования реальных объектов в автоматизированных системах обработки информации и управления. В качестве таких объектов в первую очередь следует назвать элементы и узлы ЭВМ, устройства контроля, регулирования и управления, системы временной и пространственной коммутации в технике обмена информацией и т. д. Для всех перечисленных объектов характерно наличие дискретных состояний и дискретный характер

работы во времени, т. е. их описание с помощью F-схем является эффективным. Но широта их применения не означает универсальности этих математических схем. **Например,** этот подход непригоден для описания процессов принятия решений, процессов в динамических системах с наличием переходных процессов и стохастических элементов.

Вопросы для повторения и закрепления материала

1. Чему посвящена теория автоматов?
2. Какими элементами задается абстрактный автомат?
3. Что такое функция переходов и как она описывается?
4. Что такое функция выходов и как она задается?
5. В чем заключается отличие автоматов с памятью и без памяти?
6. Каким образом задается модель в форме автомата Мура?
7. Каким образом задается модель в форме автомата Мили?
8. В чем заключается отличие синхронных автоматов от асинхронных?
9. Укажите способы задания конечных автоматов?
10. Опишите возможные приложения F-схем.

Задания для самостоятельной работы

1. Определите области применения F-схем.

Тема 6. Дискретно-стохастические модели (P-схемы)

Цель изучения темы: изучить дискретно-стохастические.

Задачи изучения темы:

- рассмотреть основные соотношения дискретно-стохастических моделей;
- рассмотреть возможные приложения P-схем.

6.1. Основные соотношения дискретно-стохастических моделей (P-схемы)

Рассмотрим особенности построения математических схем при дискретно-стохастическом подходе к формализации процесса функционирования исследуемой системы S. Так как сущность дискретизации времени при этом подходе остается аналогичной конечным автоматам, то влияние фактора стохастичности проследим также на разновидности таких автоматов, а именно на вероятностных (стохастических) автоматах.

Основные соотношения. В общем виде **вероятностный автомат** (англ. probabilistic automat) можно определить, как дискретный потактный преобразователь информации с памятью, функционирование которого в каждом такте зависит только от состояния памяти в нем и может быть описано статистически.

Применение схем вероятностных автоматов (Р-схем) имеет важное значение для разработки методов проектирования дискретных систем, проявляющих статистически закономерное случайное поведение, для выяснения алгоритмических возможностей таких систем и обоснования границ целесообразности их использования, а также для решения задач синтеза по выбранному критерию дискретных стохастических систем, удовлетворяющих заданным ограничениям.

Введем математическое понятие Р-автомата, используя понятия, введенные для F-автомата. Рассмотрим множество G, элементами которого являются всевозможные пары (x_i z_s), где x_i и z_s, — элементы входного подмножества X и подмножества состояний Z соответственно. Если существуют две такие функции φ и ψ и с их помощью осуществляются отображения G→Z и G→Y, то говорят, что F = <Z, X, Y, φ, ψ> определяет автомат детерминированного типа.

Введем в рассмотрение более общую математическую схему. Пусть Ф — множество всевозможных пар вида (z_k, y_j), где y_j — элемент выходного подмножества Y. Потребуем, чтобы любой элемент множества G индуцировал на множестве Ф некоторый закон распределения следующего вида:

Элементы из Φ	…	(z_1, y_1)	…	(z_1, y_2)	…	(z_K, y_{J-1})	(z_K, y_J)
(x_i, z_s)	…	b_{11}	…	b_{12}	…	$b_{K(J-1)}$	b_{KJ}

При этом $\sum_{k=1}^{K}\sum_{j=1}^{J} b_{kj} = 1$, где b_{kj} вероятности перехода автомата в состояние z_k и появления на выходе сигнала y_j если он был в состоянии z_j и на его вход в этот момент времени поступил сигнал x_i. Число таких распределений, представленных в виде таблиц, равно числу элементов множества G. Обозначим множество этих таблиц через B. Тогда четверка элементов P=(Z, X, Y, B) называется **вероятностным автоматом** (Р-автоматом).

Вероятностный автомат Мили

Пусть элементы множества G индуцируют некоторые законы распределения на подмножествах Y и Z, что можно представить соответственно в виде:

Элементы из Y	…	y_1	y_2	…	y_{J-1}	y_J
$(x_i z_s)$	…	q_1	q_2	…	q_{J-1}	q_J
Элементы из Z	…	z_1	z_2	…	z_{K-1}	z_K
$(x_i z_s)$	…	v_1	v_2	…	v_{K-1}	v_K

При этом $\sum_{j=1}^{J} q_j = 1$ и $\sum_{k=1}^{K} \nu_k = 1$, где q_j и v_k вероятности перехода Р-автомата в состояние z_k и появления выходного сигнала $y_к$ при условии, что Р-автомат находился в состоянии z_s и на его вход поступил входной сигнал x_i.

Если для всех *к* и *j* имеет место соотношение $q_k z_i = b_{kj}$, то такой Р-автомат называется **вероятностным автоматом Мили**. Это требование означает выполнение условия независимости распределений для нового состояния Р-автомата и его выходного сигнала.

Вероятностный автомат Мура

Пусть теперь определение выходного сигнала Р-автомата зависит лишь от того состояния, в котором находится автомат в данном такте работы. Другими словами, пусть каждый элемент выходного подмножества Y индуцирует распределение вероятностей выходов, имеющее следующий вид:

Элементы из Y	…	y_1	y_2	…	y_{J-1}	y_J
z_s	…	s_1	s_2	…	s_{I-1}	s_I

$\sum_{i=1}^{I} s_i = 1$

Здесь , где s_i — вероятность появления выходного сигнала y_i при условии, что Р-автомат находился в состоянии z_s.

Если для всех k и i имеет место соотношение $z_k s_i = b_{ki}$, то такой Р-автомат называется **вероятностным автоматом Мура**. Понятие Р-автоматов Мили и Мура введено по аналогии с детерминированным F-автоматом, задаваемым F= ‹Z, X, Y, φ, ψ›. Частным случаем Р-автомата, задаваемого как P=‹Z, X, Y, B›, являются автоматы, у которых либо переход в новое состояние, либо выходной сигнал определяются как детерминированные. Если выходной сигнал Р-автомата определяется детерминировано, то такой автомат называется **Y-детерминированным вероятностным автоматом**. Аналогично, **Z-детерминированным вероятностным автоматом** называется Р-автомат, у которого выбор нового состояния является детерминированным.

6.2. Возможные приложения Р-схем

Схемы вероятностных автоматов (Р-схем) применяются:

- в проектировании дискретных систем, проявляющих статистически закономерное случайное поведение;
- в определении алгоритмических возможностей систем;
- в обосновании границ целесообразности их использования;
- в решении задач синтеза по выбранному критерию дискретных стохастических систем, удовлетворяющих заданным ограничениям.

Пример. Рассмотрим У-детерминированный Р-автомат, который задан таблицей переходов (таблица 6.1) и таблицей выходов:

Z	z_1	z_2	$\dots$	z_{k-1}	z_k
Y	y_{i1}	y_{i2}	$\dots$	y_{ik-1}	y_{ik}

Таблица 6.1 – У-детерминированный Р-автомат

z_k	z_k				
	z_1	z_2	…	$z_{K\text{-}1}$	z_K
z_1	p_{11}	p_{12}	…	$p_{1(K\text{-}1)}$	p_{1K}
z_2	p_{21}	p_{22}	…	$p_{2(K\text{-}1)}$	p_{2K}
…	…	…	…	…	…
z_K	p_{K1}	p_{K1}	…	$p_{K(K\text{-}1)}$	p_{KK}

В этих таблицах p_{ij} — вероятность перехода Р-автомата из состояния z_i, в состояние z_j. При этом, как и ранее, $\sum_{j=1}^{K} p_{ij} = 1$

Первую из этих таблиц можно представить в виде квадратной матрицы размерности КхК, которую будем называть **матрицей переходных вероятностей** или просто **матрицей переходов** Р-автомата. В общем случае такая матрица переходов имеет вид:

$$P = \begin{Vmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{Vmatrix}$$

Для описания Y-детерминированного Р-автомата необходимо задать начальное распределение вероятностей вида:

Z	z_1	z_2	…	$z_{K\text{-}1}$	z_K
D	d_1	d_2	…	$d_{K\text{-}1}$	d_K

Здесь d_K — вероятность того, что в начале работы Р-автомат находится в состоянии z_k. При этом $\sum_{k=1}^{K} d_k = 1.$

Будем считать, что до начала работы (до нулевого такта времени) Р-автомат всегда находится в состоянии z_0 и в нулевой такт времени меняет состояние в соответствии с распределением D. Дальнейшая смена состояний Р-автомата определяется матрицей переходов P_p. Информацию о начальном состоянии Р-автомата удобно внести в матрицу P_p увеличив ее размерность до (К+1)х(К+1). При этом первая строка такой матрицы, сопоставляемая

состоянию z_0, будет иметь вид (0, d_l, d_2, ..., d_K), а первый столбец будет нулевым.

Описанный У-детерминированный Р-автомат можно задать в виде ориентированного графа, вершины которого сопоставляются состояниям автомата, а дуги — возможным переходам из одного состояния в другое. Дуги имеют веса, соответствующие вероятностям перехода p_{ij}, а около вершин графа пишутся значения выходных сигналов, индуцируемых этими состояниями.

Пример. Пусть задан Y-детерминированный Р-автомат

$$P_P = \begin{Vmatrix} 0 & 0{,}50 & 0 & 0 & 0{,}50 \\ 0 & 0 & 0 & 1{,}00 & 0 \\ 0 & 0 & 0{,}75 & 0 & 0{,}25 \\ 0 & 0 & 0{,}40 & 0 & 0{,}60 \\ 0 & 1{,}00 & 0 & 0 & 0 \end{Vmatrix};$$

Z	z_0	z_1	z_2	z_3	z_4
Y	0	0	1	1	0

.

На рисунке 6.1 показан граф переходов этого автомата. Требуется оценить суммарные финальные вероятности пребывания этого Р-автомата в состояниях z_2 и z_3.

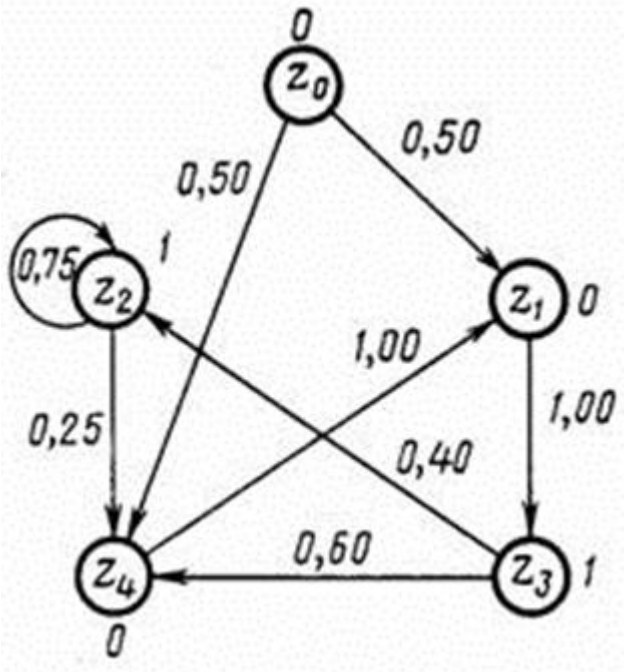


Рисунок 6.1 – Граф вероятности автомата

При использовании аналитического подхода можно записать известные соотношения из теории Марковских цепей и получить систему уравнений для определения финальных вероятностей. При этом начальное состояние z_0 можно не учитывать, так как начальное распределение не оказывает влияния на значения финальных вероятностей. Тогда имеем

$$C=\vec{C}\left\|\begin{array}{cccc} 0 & 0 & 1{,}00 & 0 \\ 0 & 0{,}75 & 0 & 0{,}25 \\ 0 & 0{,}40 & 0 & 0{,}60 \\ 1{,}00 & 0 & 0 & 0 \end{array}\right\|; \qquad \vec{C}=(c_R)=(c_1, c_2, c_3, c_4),$$

где c_k — финальная вероятность пребывания Р-автомата в состоянии z_k. Получаем систему уравнений

$$\begin{cases} c_1 = c_4, \\ c_2 = 0{,}75c_2 + 0{,}40c_3, \\ c_3 = c_1, \\ c_4 - = 0{,}25c_2 + 0{,}60c_3. \end{cases}$$

Добавим к этим уравнениям условие нормировки $c_1+c_2 + c_3 + c_4 = 1$. Тогда, решая систему уравнений, получим $c_1 = 5/23$, $c_2=8/23$, $c_3 = 5/23$, $c_4 = 5/23$. Таким образом, $c_2+c_3 = 13/23=0{,}5652$. Другими словами, при бесконечной работе заданного в этом примере У-детерминированного Р-автомата на его выходе формируется двоичная последовательность с вероятностью появления единицы, равной 0,5652.

Подобные Р-автоматы могут использоваться как генераторы марковских последовательностей, которые необходимы при построении и реализации процессов функционирования систем S или воздействий внешней среды Е.

Для оценки различных характеристик исследуемых систем, представляемых в виде Р-схем, кроме рассмотренного случая аналитических моделей можно применять и имитационные модели, реализуемые, например, методом статистического моделирования.

Вопросы для повторения и закрепления материала

1. Что такое вероятностный автомат?
2. Каким образом вероятностный автомат учитывает случайную составляющую модели?
3. Что такое Y-детерминированный вероятностный автомат?
4. Что такое Z-детерминированный вероятностный автомат?
5. Как задается вероятностный автомат Мура?
6. Как задается вероятностный автомат Мили?

Задания для самостоятельной работы

1. Изучить и систематизировать информацию по цепям Маркова.
2. Определить области применения вероятностных автоматов.

Тема 7. Непрерывно-стохастические модели (Q-схемы)

Цель изучения темы: изучить непрерывно-стохастические модели.

Задачи изучения темы:

- рассмотреть основные соотношения непрерывно-стохастических моделей;
- рассмотреть возможные приложения Q-схем.

7.1. Основные соотношения непрерывно-стохастических моделей (Q-схемы)

Особенности непрерывно-стохастического подхода рассмотрим на примере использования в качестве типовых математических схем систем массового обслуживания (англ. queuing system), которые будем называть Q-схемами. Системы массового обслуживания представляют собой класс математических схем, разработанных в теории массового обслуживания и различных приложениях для формализации процессов функционирования систем, которые по своей сути являются процессами обслуживания.

Основные соотношения. В качестве процесса обслуживания могут быть представлены различные по своей физической природе процессы функционирования экономических, производственных, технических и других систем, **например** потоки поставок продукции некоторому предприятию, потоки деталей и комплектующих изделий на сборочном конвейере цеха, заявки на обработку информации ЭВМ от удаленных терминалов и т. д. При этом характерным для работы таких объектов является случайное появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени, т. е. стохастический характер процесса их функционирования. Остановимся на основных понятиях массового обслуживания, необходимых для использования Q-схем, как при аналитическом, так и при имитационном.

В любом элементарном акте обслуживания можно выделить две основные составляющие: ожидание обслуживания заявкой и собственно обслуживание заявки. Это можно изобразить в виде некоторого i-го прибора обслуживания $П_i$ (рисунок 7.1), состоящего из накопителя заявок $Н_i$ в котором может одновременно находиться $l_i = \overline{0, L_i^H}$ заявок, где L_i^H — емкость i-го накопителя, и канала обслуживания заявок (или просто канала) $К_i$. На каждый элемент прибора обслуживания $П_i$ поступают потоки событий: в накопитель $Н_i$ — поток заявок w_i, на канал $К_i$ — поток обслуживаний u_i.

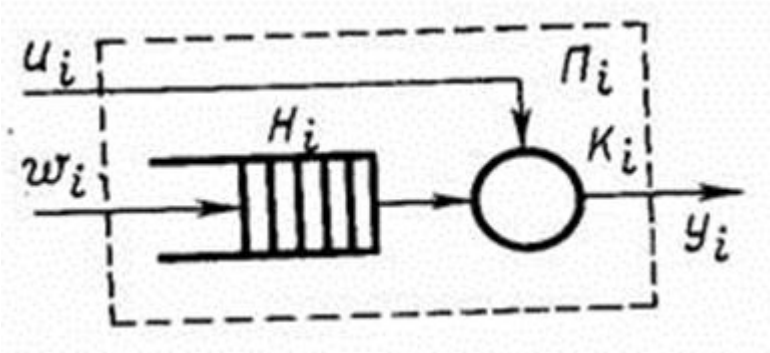


Рисунок 7.1 – Прибор обслуживания заявок

Потоком событий называется последовательность событий, происходящих одно за другим в какие-то случайные моменты времени. Различают:

- **Потоком однородных событий** называется такой поток, если все заявки равноправны и поток характеризуется только моментами поступления этих событий (вызывающими моментами) и задается последовательностью $\{t_n\} = \{0 \le t_1 \le t_2 ... \le t_n \le ...\}$, где t_n — момент наступления n-го события — неотрицательное вещественное число. Однородный поток событий также может быть задан в виде последовательности промежутков времени между n-м и (n—1)-м событиями $\{\tau_n\}$, которая однозначно связана с последовательностью вызывающих моментов $\{t_n\}$, где $\tau_n = t_n - t_{n-1}$, $n \ge 1$, $t_0 = 0$ т.е. $\tau_1 = t_1$.

- **Потоком неоднородных событий** называется последовательность $\{t_n, f_n\}$, где t_n -вызывающие моменты; f_n — набор признаков события. Например, применительно к процессу обслуживания для неоднородного потока заявок могут быть заданы принадлежность к тому или иному источнику заявок, наличие приоритета, возможность обслуживания тем или иным типом канала и т. п.

Рассмотрим поток, в котором события разделены интервалами времени τ_1, τ_2,... ..., которые вообще являются случайными величинами. Пусть интервалы τ_1; τ_2,... независимы между собой. Тогда поток событий называется **потоком с ограниченным последействием**.

Пример потока событий приведен на рисунке 7.2, где обозначено T_j — интервал между событиями (случайная величина); $T_н$ — время наблюдения, T_c — момент совершения события.

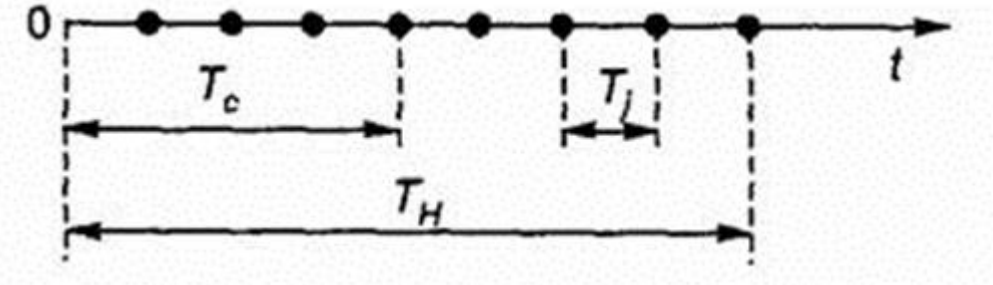


Рисунок 7.2 – Графическое изображение Q-схемы

Интенсивность потока можно рассчитать экспериментально по формуле

$$\lambda = \frac{N}{T_н}$$

где N— число событий, произошедших за время наблюдения $T_н$. Если T_j =const или определено какой-либо формулой $T_j = f(T_{j-1})$, то поток называется **детерминированным потоком**. Иначе поток называется **случайным потоком.**

Случайные потоки бывают:

1. **Ординарный случайный поток**, когда вероятность одновременного появления 2-х и более событий равна нулю.

Поток событий называется **ординарным**, если вероятность того, что на малый интервал времени Δt, примыкающий к моменту времени t, попадает больше одного события $P_{>1}(t, \Delta t)$, пренебрежительно мала по сравнению с вероятностью того, что на этот же интервал времени Δt попадает ровно одно событие $P_1(t, \Delta t)$, т. е. $P_1(t, \Delta t) \gg P_{>1}(t, \Delta t)$. Если для любого интервала Δt событие

$$P_0(t, \Delta t) + P_1(t, \Delta t) + P_{>1}(t, \Delta t) = 1$$

как сумма вероятностей событий, образующих полную группу и несовместных, то дня ординарного потока событий

$$P_0(t, \Delta t) + P_1(t, \Delta t) \approx 1,\ P_{>1}(t, \Delta t) = 0(\Delta t),$$

где $0(\Delta t)$ — величина, порядок малости которой выше, чем Δt, т. е.

$$\lim_{\Delta t \to 0} [0(\Delta t)/\Delta t] = 0.$$

2. **Стационарный случайный поток**, когда частота появления событий постоянная;

Стационарным потоком событий называется поток, для которого вероятность появления того или иного числа событий на интервале времени τ зависит лишь от длины этого участка и не зависит от того, где на оси времени 0t взят этот участок.

Рассмотрим на оси времени 0t ординарный поток событий и найдем среднее число событий, наступающих на интервале времени Δt, примыкающем к моменту времени t. Получим

$$0 \cdot P_0(t, \Delta t) + 1 \cdot P_1(t, \Delta t) = P_1(t, \Delta t).$$

Тогда среднее число событий, наступающих на участке времени Δt в единицу времени, составит $[P_1(t, \Delta t)]/\Delta t$. Рассмотрим предел этого выражения при $\Delta t \to 0$. Если этот предел существует, то она называется **интенсивностью (плотностью) ординарного потока событий** $\lim_{\Delta t \to 0} [P_1(t, \Delta t)/\Delta t] = \lambda(t).$

Интенсивность потока может быть любой неотрицательной функцией времени, имеющей размерность, обратную размерности времени. Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянное значение, равное среднему числу событий, наступающих в единицу времени

3. **Случайный поток без последействия**, когда вероятность не зависит от момента совершения предыдущих событий.

7.2. Возможные приложения Q-схем

Обычно в приложениях при моделировании различных систем применительно к элементарному каналу обслуживания $К_i$ можно считать, что поток заявок $w_i \in W$, т. е. интервалы времени между моментами появления заявок (вызывающие моменты) на входе $К_i$ образует подмножество неуправляемых переменных, а поток обслуживания, $u_i \in U$ т. е. интервалы времени между началом и окончанием обслуживания заявки, образует подмножество управляемых переменных.

Заявки, обслуженные каналом $К_i$ и заявки, покинувшие прибор $П_i$ по различным причинам необслуженными (например, из-за переполнения накопителя $Н_i$), образуют выходной поток $y_i \in Y$, т. е. интервалы времени между моментами выхода заявок образуют подмножество выходных переменных.

Процесс функционирования прибора обслуживания $П_i$ можно представить как процесс изменения состояний его элементов во времени $z_i(t)$. Переход в новое состояние для $П_i$ означает изменение количества заявок, которые в нем находятся (в канале $К_i$ и в накопителе $Н_i$). Таким образом, вектор состояний для $П_i$ имеет вид $\vec{z}_i = (z_i^H, z_i^K)$, где z_i^H — состояние накопителя $Н_i$ (z_i^H =0 — накопитель пуст, z_i^H =1 — в накопителе имеется одна заявка, ..., $z_i^H = L_i^H$ — емкость накопитель полностью заполнен); L_i^H — емкость накопителя $Н_i$ измеряемая числом заявок, которые в нем могут поместиться; z_i^K — состояние канала Ki (z_i^K =0— канал свободен, z_i^K =1— канал занят и т. д.).

В практике моделирования систем, имеющих более сложные структурные связи и алгоритмы поведения, для формализации используются не отдельные приборы обслуживания, а Q-схемы, образуемые композицией многих элементарных приборов обслуживания $П_i$ (сети массового обслуживания). Если каналы $К_i$ различных приборов обслуживания соединены параллельно, то имеет место **многоканальное обслуживание** (многоканальная Q-схема), а если приборы $П_i$ и их параллельные композиции соединены последовательно, то имеет место **многофазное обслуживание** (многофазная Q-схема). Таким образом, для задания Q-схемы необходимо

использовать оператор сопряжения R, отражающий взаимосвязь элементов структуры (каналов и накопителей) между собой.

Связи между элементами Q-схемы изображают в виде стрелок (линий потока, отражающих направление движения заявок). Различают:

- **разомкнутые Q-схемы**, в которых выходной поток обслуженных заявок не может снова поступить на какой-либо элемент, т. е. обратная связь отсутствует;

- **замкнутые Q-схемы**, в которых имеются обратные связи, по которым заявки двигаются в направлении, обратном движению вход-выход.

Собственными (внутренними) параметрами Q-схемы будут являться количество фаз $L_{ф}$, количество каналов в каждой фазе L_{kj}, $\overline{j=1,\ L^{Ф}}$, количество накопителей каждой фазы $\overline{L_{Нк},\ k=1,\ L_{ф}}$, емкость i-го накопителя L_i^H. Следует отметить, что в теории массового обслуживания в зависимости от емкости накопителя применяют следующую терминологию для систем массового обслуживания:

- **системы с потерями** ($L_i^H=0$, т. е. накопитель в приборе $П_i$ отсутствует, а имеется только канал обслуживания K_i);

- **системы с ожиданием** ($L_i^H \to \infty$, т. е. накопитель $П_i$, имеет бесконечную емкость и очередь заявок не ограничивается);

- **системы смешанного типа** (с ограниченной емкостью накопителя H_i).

Всю совокупность собственных параметров Q-схемы обозначим как подмножество H.

Для задания Q-схемы также необходимо описать алгоритмы ее функционирования, которые определяют набор правил поведения заявок в системе в различных неоднозначных ситуациях. В зависимости от места возникновения таких ситуаций различают алгоритмы (дисциплины) ожидания заявок в накопителе H_i и обслуживания заявок каналом K_i каждого элементарного обслуживающего прибора $П_i$ Q-схемы. Неоднородность заявок, отражающая процесс в той или иной реальной системе, учитывается с помощью введения классов приоритетов.

В зависимости от динамики приоритетов в Q-схемах различают:

- **Статические приоритеты** назначаются заранее и не зависят от состояний Q-схемы, т. е. они являются фиксированными в пределах решения конкретной задачи моделирования.

- **Динамические приоритеты** возникают при моделировании в зависимости от возникающих ситуаций.

Исходя из правил выбора заявок из накопителя H_i на обслуживание каналом K_i можно выделить

- **Относительный приоритет** означает, что заявка с более высоким приоритетом, поступившая в накопитель H_i ожидает окончания

обслуживания предшествующей заявки каналом K_i и только после этого занимает канал.

- **Абсолютный приоритет** означает, что заявка с более высоким приоритетом, поступившая в накопитель H_i, прерывает обслуживание каналом K_i, заявки с более низким приоритетом и сама занимает канал (при этом вытесненная из K_i заявка может либо покинуть систему, либо может быть снова записана на какое-то место в H_i.

При рассмотрении алгоритмов функционирования приборов обслуживания $П_i$ (каналов K_i и накопителей H_i) необходимо также задать набор правил, по которым заявки покидают H_i и K_i. Для H_i — либо **правила переполнения**, по которым заявки в зависимости от заполнения H_i покидают систему, либо **правила ухода**, связанные с истечением времени ожидания заявки в H_i. Для K_i — **правила выбора маршрутов** или **направлений ухода.** Кроме того, для заявок необходимо задать правила, по которым они остаются в канале $К_i$ или не допускаются до обслуживания каналом K_i т. е. **правила блокировок канала**. При этом различают **блокировки $К_i$ по выходу** и **по входу**. Такие блокировки отражают наличие управляющих связей в Q-схеме, регулирующих поток заявок в зависимости от состояний Q-схемы. Весь набор возможных алгоритмов поведения заявок в Q-схеме можно представить в виде некоторого **оператора алгоритмов поведения заявок A.**

Таким образом, Q-схема, описывающая процесс функционирования системы массового обслуживания любой сложности, однозначно задается в виде **Q=<W, U, H, Z, R, A>.**

При ряде упрощающих предположений относительно подмножеств входящих потоков W и потоков обслуживания U (выполнение условий стационарности, ординарности и ограниченного последействия) оператора сопряжения элементов структуры R (однофазное одноканальное обслуживание в разомкнутой системе), подмножества собственных параметров H (обслуживание с бесконечной емкостью накопителя), оператора алгоритмов обслуживания заявок A (бесприоритетное обслуживание без прерываний и блокировок) для оценки вероятностно-временных характеристик можно использовать аналитический аппарат, разработанный в теории массового обслуживания. При принятых предположениях в обозначениях Д. Кендалла будет иметь место классическая система обслуживания типа M/M/1 (одноканальная система с марковским входящим потоком заявок и марковским потоком обслуживания). Рассмотрим на примере основные аналитические соотношения для такой элементарной Q-схемы.

Пример. Допустим, что процесс обслуживания начинается при отсутствии заявок в накопителе. Тогда состояния системы массового обслуживания описываются следующей системой уравнений:

$$\begin{cases} P_n(t+\Delta t) = P_n(t)\,[1-(\lambda+\mu)\Delta t] + P_{n-1}(t)\lambda\Delta t + P_{n+1}(t)\mu\Delta t, \; n \geqslant 1, \\ P_0(t+\Delta t) = P_0(t)\,(1-\lambda\Delta t) + P_1(t)\mu\Delta t, \end{cases}$$

где $P_n(t)$ — вероятность нахождения системы в состоянии $z_n(t) \in Z$ в момент времени t, т. е. когда в ней имеется n заявок.

Эти уравнения следуют из того, что вероятность нахождения в системе n заявок в момент времени (t+Δt) равна вероятности нахождения в системе n заявок в момент t, умноженной на вероятность того, что за время Δt в систему не поступит ни одной заявки и ни одна заявка не будет обслужена, плюс вероятность нахождения в системе (n—1) заявок в момент t, умноженная на вероятность того, что за время Δt поступит одна заявка и ни одна заявка не будет обслужена, плюс вероятность нахождения в системе (n+1) заявок в момент t, умноженная на вероятность того, что за время Δt одна заявка покинет систему и не поступит ни одной заявки. Вероятность того, что за время Δt не поступит ни одной заявки и ни одна заявка не покинет систему, равна (1—λΔt)(1—μΔt). Член, содержащий $(\Delta t)^2$, при составлении дифференциального уравнения опускается. Следовательно, можно записать 1— (λ+μ)Δt. Относительно остальных двух членов первого уравнения заметим, что

$$\lambda\Delta t\,(1-\mu\Delta t) \approx \lambda\Delta t, \; \mu\Delta t\,(1-\lambda\Delta t) \approx \mu\Delta t.$$

Перенеся $P_n(t)$ влево и устремив Δt к нулю, получим систему дифференциальных уравнений

$$\begin{cases} dP_n(t)/dt = -(\lambda+\mu)P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t), \; n \geqslant 1, \\ dP_0(t)/dt = -\lambda P_0(t) + \mu P_1(t). \end{cases}$$

Найдем выражение для математического ожидания числа заявок, находящихся в накопителе, и среднего времени ожидания заявок в накопителе для стационарного состояния $p=\lambda/\mu<1$. Приравняв нулю производные по времени и исключив, таким образом, время t из уравнений, получим систему алгебраических уравнений

$$\begin{cases} (\lambda+\mu)p_n = \lambda p_{n-1} + \mu p_{n-1}, \; n \geqslant 1, \\ \lambda p_0 = \mu p_1, \end{cases} \qquad \begin{cases} (1+\rho)p_n = p_{n+1} + \rho p_{n-1}, \; n \geqslant 1, \\ p_1 = \rho p_0. \end{cases}$$

Пусть в первом уравнении n=1. Тогда $(1+\rho)p_1 = p_2 + \rho p_0$. Подставив сюда значение p_1 из второго уравнения, находим $p_2 = \rho^2 p_0$. Повторяя эти операции, получаем $p_n = \rho^2 p_0$, причем $\sum_{n=0}^{\infty} p_n = 1$, так как это сумма вероятностей того, что в системе нет ни одной заявки, имеется одна заявка, две заявки и т. д. Сумма этих вероятностей должна быть равна единице, так как рассматриваются все возможные состояния системы. Поэтому $\sum_{n=0}^{\infty} \rho^n p_0 = 1$,

или $\sum_{n=0}^{\infty} p_0\rho^n = p_0 \sum_{n=0}^{\infty} \rho^n = p_0/(1-\rho) = 1$, откуда $p_0 = 1-\rho$. Следовательно, $p_n = \rho^n (1-\rho)$.

Полученное выражение представляет собой геометрическое распределение.

Математическое ожидание числа заявок, находящихся в системе (приборе),

$$\bar{l}_n = \sum_{n=0}^{\infty} np_n = (1-\rho) \sum_{n=0}^{\infty} n\rho^n = \rho\,(1-\rho).$$

Отметим, что l_n — среднее значение и возможны колебания числа заявок, ожидающих обслуживания, что можно оценить с помощью дисперсии:

$$D\,[l_n] = \sum_{n=0}^{\infty} (n-\bar{l}_n)^2 p_n = \sum_{n=0}^{\infty} n^2 p_n - [\rho/(1-\rho)]^2.$$

При этом

$$\sum_{n=0}^{\infty} n^2 p_n = (1-\rho) \sum_{n=0}^{\infty} n^2 \rho^n = \rho/(1-\rho) + 2\rho^2/(1-\rho)^2.$$

Следовательно,

$$D\,[l_n] = \rho/(1-\rho) + 2\rho^2/(1-\rho)^2.$$

Математическое ожидание числа заявок, находящихся в накопителе,

$$\bar{l}_н = \sum_{n=1}^{\infty} (n-1)p_n = \bar{l}_n - \rho = \rho/(1-\rho) - \rho = \rho^2/(1-\rho).$$

Среднее время ожидания заявок в накопителе

$$\bar{t}_н = \bar{l}_н/\lambda = \rho^2/[\lambda\,(1-\rho)].$$

Возможности оценки характеристик с использованием аналитических моделей теории массового обслуживания являются весьма ограниченными по сравнению с требованиями практики исследования и проектирования систем, формализуемых в виде Q-схем. Несравненно большими возможностями обладают имитационные модели, позволяющие исследовать Q-схему, задаваемую Q=<W, U,H, Z, Y, R, A>, без ограничений.

Вопросы для повторения и закрепления материала

1. Что такое система массового обслуживания?
2. Как выглядит прибор обслуживания заявок в общем виде?
3. За что отвечает накопитель заявок?
4. Что такое поток событий?
5. В каком случае поток событий называется однородным?
6. В каком случае поток событий называется неоднородным?
7. Какими бывают случайные потоки?
8. Какими могут быть возможные приложения Q-схем?

Задания для самостоятельной работы

1. Определить области применения систем массового обслуживания.

Тема 8. Моделирование процессов функционирования систем на базе Q-схем

Цель изучения темы: изучить основы моделирования процессов функционирования на базе Q-схем.

Задачи изучения темы:

- рассмотреть процесс формализации на базе Q-схем;
- рассмотреть способы построения моделирующих алгоритмов на базе Q-схем;
- рассмотреть особенности моделирования на базе Q-схем.

8.1. Формализация на базе Q-схем

Рассмотрим возможности использования Q-схем для формального описания процесса функционирования некоторой системы S. Характерная ситуация в работе таких систем — появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени, т. е. стохастический характер процесса их функционирования. В общем случае моменты поступления заявок в систему S из внешней среды E образуют входящий поток, а моменты окончания обслуживания образуют выходящий поток обслуженных заявок.

Формализация на базе Q-схем. Формализуя какую-либо реальную систему с помощью Q-схемы, необходимо построить структуру такой системы. В качестве элементов структуры Q-схем будем рассматривать элементы трех типов: И — источники; Н — накопители; К — каналы обслуживания заявок.

Пример структуры системы S, представленной в виде Q-схемы, приведен на рисунке 8.1. Кроме связей, отражающих движение заявок в Q-схеме (сплошные линии), можно говорить о различных управляющих связях. Примером таких связей являются различные блокировки обслуживающих каналов (по входу и по выходу «клапаны» изображены в виде треугольников, а управляющие связи — пунктирными линиями. Блокировка канала по входу означает, что этот канал отключается от входящего потока заявок, а блокировка канала по выходу указывает, что заявка, уже обслуженная блокированным каналом, остается в этом канале до момента снятия блокировки (открытия «клапана»). В этом случае, если перед накопителем нет «клапана», при его переполнении будут иметь место потери заявок. Помимо выходящего потока обслуженных заявок можно говорить о потоке потерянных заявок.

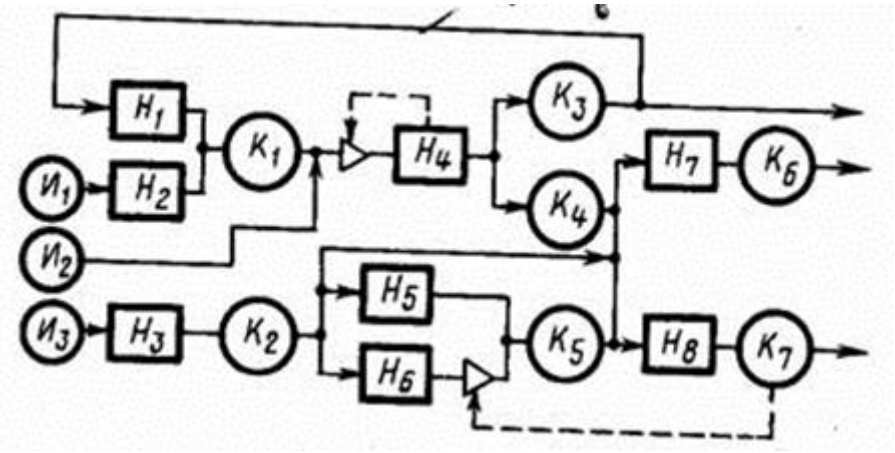


Рисунок 8.1 – Структура системы, представленной в виде Q-схемы

Как отмечалось выше, **Q-схему** можно считать **заданной**, если определены:

1. потоки событий (входящие потоки заявок и потоки обслуживании для каждого Н и К);

2. структура системы S (число фаз $L^{ф}$, число каналов обслуживания $L^{К}$, число накопителей $L^{Н}$ каждой из $L^{ф}$ фаз обслуживания заявок и связи И, Н и К);

3. алгоритмы функционирования системы (дисциплины ожидания заявок в Н и выбора на обслуживание К, правила ухода заявок из Н и К).

Рассмотрим возможности формализации воздействий внешней среды Е, представляемых в Q-схемах в виде источников (И). Формирование однородных потоков событий, заданных в общем виде многомерным интегральным законом или плотностью распределения вероятностей, т.е.

$$F(y_1, y_2, \ldots, y_k) = P\{\tau_1 < y_1, \tau_2 < y_2, \ldots, \tau_k < y_k\},$$

$$f(y_1, y_2, \ldots, y_k) = F^k(y_1, y_2, \ldots, y_k)/(\partial y_1 \partial y_2 \ldots \partial y_k),$$

сводится к рассмотренным ранее методам машинной имитации k-мерных векторных величин, требующих больших затрат машинных ресурсов. При моделировании систем, формализуемых в виде Q-схем, часто возникают задачи имитации потоков заявок с некоторыми ограничениями, позволяющими упростить как математическое описание, так и программную реализацию генераторов потоков заявок.

Так, для ординарных потоков с ограниченным последействием интервалы между моментами поступления заявок являются независимыми и совместная плотность распределения может быть представлена в виде произведения частных законов распределения: $f(y_1, y_2, ..., y_k) = f_1(y_1), f_2(y_2), ... f_k(y_k)$, где $f_i(y_i)$, $i = \overline{1,k}$, при $i > 1$ являются условными функциями плотности величин y_i при условии, что в момент начала i-го интервала поступит заявка.

Относительно начального момента времени t_0 никаких предположений не делается, поэтому функция $f_1(y_1)$ - безусловная.

Если поток с ограниченным последействием удовлетворяет условию стационарности, т.е. вероятность появления к событий на интервале $(t_0, t_0+\Delta t)$ зависит только от длины интервала Δt, то при $i>0$ интервалы τ_i, распределены одинаково, т. е.

$$f_2(y_2)=f_3(y_3)=\ldots=f_k(y_k)$$

Плотность распределения первого интервала $f_1(y_1)$ может быть найдена с использованием соотношения Пальма

$$f_1(y_1)=\lambda\left(1-\int_0^{y_1} f(y)\,dy\right), \qquad (8.1)$$

где λ— интенсивность потока событий.

Порядок моделирования моментов появления заявок в стационарном потоке с ограниченным последействием следующий. Из последовательности случайных чисел, равномерно распределенных на интервале (0, 1), выбирается случайная величина и формируется первый интервал y_1 в соответствии с (8.1) любым из рассмотренных выше способов формирования случайной величины. Момент наступления первого события $t_1 = t_0+y_1$ следующие моменты появления событий определяются как

$$t_2=t_1+y_2,\ \ldots,\ t_k=t_{k-1}+y_k, \qquad (8.2)$$

где y_k - случайная величина с плотностью f(y).

Пример. Пусть при моделировании некоторой системы необходимо сформировать на ЭВМ простейший поток заявок. Распределение длин интервалов между заявками является экспоненциальным, т. е. $f(y) = \lambda e^{-\lambda y}$, $y>0$.

Используем формулу Пальма для определения первого интервала y, т.е.

$$f_1(y_1)=\lambda\left(1-\int_0^{y_1}\lambda e^{-\lambda y}\,dy\right)=\lambda e^{-\lambda y_1}.$$

Из этого выражения следует, что $f_1(y_1)=f(y)$, т. е. первый интервал распределен так же, как и остальные. Этого и следовало ожидать ввиду отсутствия последействия в простейшем потоке. Формируя на ЭВМ равновероятностные случайные числа x_i на интервале (0, 1), будем преобразовывать их в соответствии с выражением

$$\int_0^{y_i} f(y)\,dy = x_i.$$

Тогда длина интервала между (i—1)-м и i-m событиями $y_i= -(1/\lambda)\ln x_i$, а моменты появления заявок в потоке определяются согласно (8.2).

8.2. Способы построения моделирующих алгоритмов Q-схем

Моделирующий алгоритм должен адекватно отражать процесс функционирования системы S и в то же время не создавать трудностей при машинной реализации модели $M_м$. При этом моделирующий алгоритм должен отвечать следующим основным требованиям:

- обладать универсальностью относительно структуры, алгоритмов функционирования и параметров системы S;

- обеспечивать одновременную (в один и тот же момент системного времени) и независимую работу необходимого числа элементов системы S;

- укладываться в приемлемые затраты ресурсов ЭВМ (машинного времени и памяти) для реализации машинного эксперимента;

- проводить разбиение на достаточно автономные логические части, т. е. возможность построения блочной структуры алгоритма;

- гарантировать выполнение рекуррентного правила — событие, происходящее в момент времени t_k, может моделироваться только после того, как промоделированы все события, произошедшие в момент времени $t_{k-1}<t_k$.

При этом необходимо иметь в виду, что появление одной заявки входящего потока в некоторый момент времени t_i может вызвать изменение состояния не более чем одного из элементов Q-схемы, а окончание обслуживания заявки в момент t_i в некотором канале (К) может привести в этот момент времени к последовательному изменению состояний нескольких элементов (Н и К), т. е. будет иметь место процесс распространения смены состояний в направлении, противоположном движению заявок в системе S.

Известно, что существует два основных принципа построения моделирующих алгоритмов: - «**принцип Δt»** и «**принцип δz**». При построении моделирующего алгоритма Q-схемы по «принципу Δt», т. е. алгоритма с детерминированным шагом, необходимо для построения адекватной модели $M_м$ определить минимальный интервал времени между соседними событиями $\Delta t'=\min\{u_i\}$ (во входящих потоках и потоках обслуживании) и принять, что шаг моделирования равен Δt'. В моделирующих алгоритмах, построенных по «принципу δz», т. е. в алгоритмах со случайным шагом, элементы Q-схемы просматриваются при моделировании только в моменты особых состояний (в моменты появления заявок из И или изменения состояний К). При этом длительность шага Δt =var зависит как от особенностей самой системы S, так и от воздействий внешней среды Е.

Моделирующие алгоритмы со случайным шагом могут быть реализованы синхронным и асинхронным способами. При синхронном способе один из элементов Q-схемы (И, Н или К) выбирается в качестве ведущего и по нему «синхронизируется» весь процесс моделирования. При асинхронном способе построения моделирующего алгоритма ведущий (синхронизирующий) элемент не используется, а очередному шагу

моделирования (просмотру элементов Q-схемы) может соответствовать любое особое состояние всего множества элементов И, Н и К. При этом просмотр элементов Q-схемы организован так, что при каждом особом состоянии либо циклически просматриваются все элементы, либо спорадически — только те, которые могут в этом случае изменить свое состояние (просмотр с прогнозированием).

Классификация возможных способов построения моделирующих алгоритмов Q-схем приведена на рисунке 8.2.

Рисунок 8.2 – Классификация способов построения моделирующих алгоритмов Q- -схем

8.3. Особенности моделирования на базе Q-схем

Математическое обеспечение и ресурсные возможности современных ЭВМ позволяют достаточно эффективно провести моделирование различных систем, формализуемых в виде Q-схем, используя либо пакеты прикладных программ, созданные на базе алгоритмических языков общего назначения, либо специализированные языки имитационного моделирования. Но прежде чем применять эти средства автоматизации моделирования, необходимо глубже вникнуть в суть процесса построения и реализации моделирующих алгоритмов.

Пример. Для более детального ознакомления с технологией машинной имитации остановимся на рассмотрении Q-схемы достаточно общего вида, показанной на рисунке 8.3.

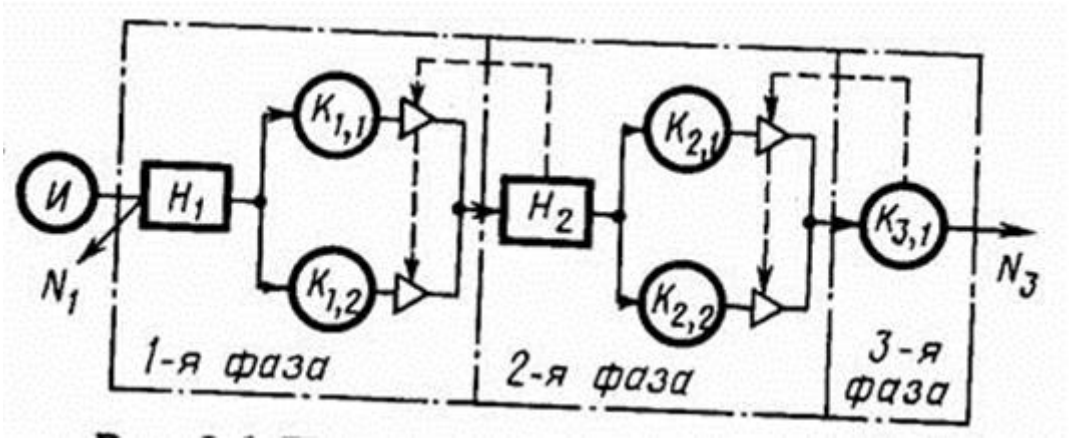


Рисунок 8.3 – Пример Q-схемы общего вида

В частности, разберем на данном примере, какое влияние оказывает на особенности построения схемы моделирующего алгоритма принцип, положенный в основу его машинной реализации.

На рисунке представлена трехфазная Q-схема ($L^{Ф}$=3) с блокировкой каналов по выходу в 1-й и 2-й фазах обслуживания (пунктирные линии на рисунке). В качестве выходящих потоков такой Q-схемы могут быть рассмотрены поток потерянных заявок из H_1 (N_l) и поток обслуженных заявок из $К_{3,1}$ (N_3).

Для имитационной модели рассматриваемой Q-схемы можно записать следующие переменные и уравнения:

- эндогенная переменная Р — вероятность потери заявок;
- экзогенные переменные: t_m—время появления очередной заявки из И;
- $t_{k,j}$ — время окончания обслуживания каналом $К_{k,j}$ очередной заявки, к=1, 2, 3; j= 1, 2;
- вспомогательные переменные: z_i и $z_{k,j}$ — состояния H_i и $К_{к,j}$, i=l, 2; к=1, 2, 3; j =1, 2;
- параметры: L_i — емкость i-го H_i; L_k^K — число каналов в к-й фазе; L_1^K =2, L_2^K=2, L_3^K =1;
- переменные состояния: N_1 — число потерянных заявок в H_1, N_3 — число обслуженных заявок, т. е. вышедших из 3-й фазы;
- уравнение модели: $P=N_1/(N_1+N_3)=N_1/N$.

При имитации процесса функционирования Q-схемы на ЭВМ требуется организовать массив состояний. В этом массиве должны быть выделены:

- подмассив К для запоминания текущих значений $z_{k,j}$ соответствующих каналов $К_{k,j}$ и времени окончания обслуживания очередной заявки $t_{k,j}$, j=(1, L_k^K);
- подмассив Н для записи текущего значения z_i, соответствующих накопителей H_i, i=l, 2;

- подмассив И, в который записывается время поступления очередной заявки t_m из источника (И).

Процедура моделирования процесса обслуживания каждым элементарным каналом $К_{k,j}$ сводится к следующему:

- путем обращения к генератору случайных чисел с законом распределения, соответствующим обслуживанию данных $К_{k,j}$, получается длительность времени обслуживания и вычисляется время окончания обслуживания $t_{k,j}$, а затем фиксируется состояние $z_{k,j}$=l;

- при освобождении $z_{k,j}$=0; в случае блокировки $K_{k,j}$ записывается $z_{k,j}$=2.

- при поступлении заявки в H_i к его содержимому добавляется единица, т. е. $z_i = z_i$ +l, а при уходе заявки из H_i, на обслуживание вычитается единица, т. е. $z_i = z_i$-1, i=l, 2.

Вопросы для повторения и закрепления материала

1. Что представляет собой элемент системы источник?
2. Что представляет собой элемент системы накопитель?
3. Что представляет собой элемент системы канал?
4. Каким требованиям должен отвечать моделирующий алгоритм в форме Q-схемы?
5. В чем заключается сущность принципа Δ?
6. В чем заключается сущность принципа δz?
7. Каким образом классифицируются способы построения моделирующих алгоритмов Q- -схем?
8. Укажите особенности моделирования на базе Q-схем.

Задания для самостоятельной работы

1. Разработать собственные примеры моделей, основанных на базе Q-схем.

Тема 9. Сетевые модели (N-схемы)

Цель изучения темы: изучить сетевые модели.

Задачи изучения темы:

- рассмотреть основные соотношения сетевых моделей;
- рассмотреть возможные приложения N-схем.

9.1. Основные соотношения сетевых моделей (N-схемы)

В практике моделирования объектов часто приходится решать задачи, связанные с формализованным описанием и анализом причинно-следственных связей в сложных системах, где одновременно параллельно протекает несколько процессов. Самым распространенным в настоящее время формализмом, описывающим структуру и взаимодействие параллельных систем и процессов, являются сети Петри (англ. Petri Nets), предложенные К. Петри.

Основные соотношения. Теория сетей Петри развивается в нескольких направлениях: разработка математических основ, структурная теория сетей, различные приложения (параллельное программирование, дискретные динамические системы и т. д.).

Формально сеть Петри (N-схема) задается четверкой вида

$$N=\langle B, D, I, O\rangle,$$

где B — конечное множество символов, называемых позициями, B ≠Ø;

- D — конечное множество символов, называемых переходами, D≠Ø, B∩D≠Ø;

- I—входная функция (прямая функция инцидентности), I: BxD→{0,1};

- O — выходная функция (обратная функция инцидентности), O: DxB→{0,1}.

Таким образом, входная функция отображает переход d_j в множество входных позиций $b_i \in I(d_j)$, а выходная функция O отображает переход d_j в множество выходных позиций $b_i \in D(d_j)$. Для каждого перехода $d_j \in D$ можно определить множество входных позиций перехода $I(d_j)$ и выходных позиций перехода $O(d_i)$ как

$$I(d_j)=\{b_i \in B \mid I(b_i,\ d_j)=1\},$$
$$O(d_j)=\{b_i \in B \mid O(d_j,\ b_i)=1\},$$
$$i=\overline{1,\ n};\ j=\overline{1,\ m},\ n=|B|,\ m=|D|.$$

Аналогично, для каждого перехода $b_i \in B$ вводятся определения множества входных переходов позиции $I(b_i)$ и множества выходных переходов позиции $I(b_i)$:

$$I(b_i)=\{d_j \in D \mid I(d_j,\ b_i)=1\},$$
$$O(b_i)=\{d_j \in D \mid 0(b_i,\ d_j)=1\}.$$

Графически N-схема изображается в виде двудольного ориентированного мультиграфа, представляющего собой совокупность позиций и переходов N-схемы (рисунок 9.1). Как видно из этого рисунка, граф N-схемы имеет два типа узлов: позиции и переходы, изображаемые 0 и 1 соответственно. Ориентировочные дуги соединяют позиции и переходы, причем каждая дуга направлена от элемента одного множества (позиции или перехода) к элементу другого множества (переходу или позиции). Граф N-схемы является мультиграфом, так как он допускает существование кратных дуг от одной вершины к другой.

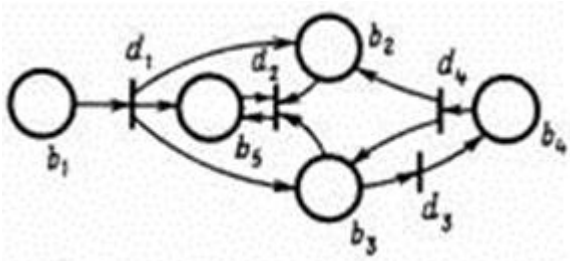


Рисунок 9.1 – Графическое изображение N-схемы

Пример. Представим формально N-схему, показанную в виде графа на рисунке 9.1:

$$N=\langle B,\ D,\ I,\ O\rangle,$$
$$B=\langle b_1,\ b_1,\ b_3,\ b_4,\ b_5\rangle,$$
$$D=\langle d_1,\ d_2,\ d_3,\ d_4\rangle,$$
$$I(d_1)=\{b_1\}, \qquad O(d_1)=\{b_2,\ b_3,\ b_5\},$$
$$I(d_2)=\{b_2,\ b_3,\ b_5\}, \qquad O(d_2)=\{b_5\},$$
$$I(d_3)=\{b_3\}, \qquad O(d_3)=\{b_4\},$$
$$I(d_4)=\{b_4\}, \qquad O(d_4)=\{b_2,\ b_3\}.$$

9.2. Возможные приложения N-схем

Приведенное представление N-схемы может использоваться только для отражения статики моделируемой системы (взаимосвязи событий и условий), но не позволяет отразить в модели динамику функционирования моделируемой системы. Для представления динамических свойств объекта вводится функция маркировки (разметки) M: B→{0, 1, 2, ...}. Маркировка M есть присвоение неких абстрактных объектов, называемых метками (фишками), позициям N-схемы, причем количество меток, соответствующее каждой позиции, может меняться. При графическом задании N-схемы разметка отображается помещением внутри вершин-позиций соответствующего числа точек (когда количество точек велико, ставят цифры).

Маркированная (размеченная) N-схема может быть описана в виде пятерки NM = ‹B, D, I, O, M› и является совокупностью сети Петри и маркировки M.

Функционирование N-схемы отражается путем перехода от разметки к разметке. Начальная разметка обозначается как M_o: B→ {0, 1, 2, ...}. Смена разметок происходит в результате срабатывания одного из переходов $d_j \in D$ сети. Необходимым условием срабатывания перехода d_j является $b_i \in I(d_j)\{M(b_i) \geq 1\}$, где M{$b_i$} — разметка позиции b_i. Переход b_j, для которого выполняется указанное условие, определяется как находящийся в состоянии готовности к срабатыванию или как возбужденный переход.

Срабатывание перехода d_j изменяет разметку сети $M(b) = (M(b_1), M(b_2), ..., M(b_n))^2$ на разметку M'(b) по следующему правилу:

$$M'(b) = M(b) - I(d_j) + O(d_j),$$

т. е. переход d_j изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Для изображения смены разметки M на M' применяют обозначение $M |^{dj}\_ M'$.

Пример. Рассмотрим размеченную N-схему с начальной разметкой Mo={1, 0, 0, 0, 1, 0, 1}, которая приведена на рисунке 9.2, а. При такой начальной разметке N-схемы единственным готовым к срабатыванию является переход d_2, срабатывание которого ведет к смене разметки Mo |-2 M_1, где M_1 = {0, 1, 1, 0, 1, 0, 1} (рисунок 9.2, б). При разметке M_1 возможно срабатывание переходов d1, d3 и d5. В зависимости от того, какой переход сработал первым, получается одна из трех возможных новых маркировок (рисунок 9.2, в, г. д). Функционирование N-схемы продолжается до тех пор, пока существует хотя бы один возможный переход.

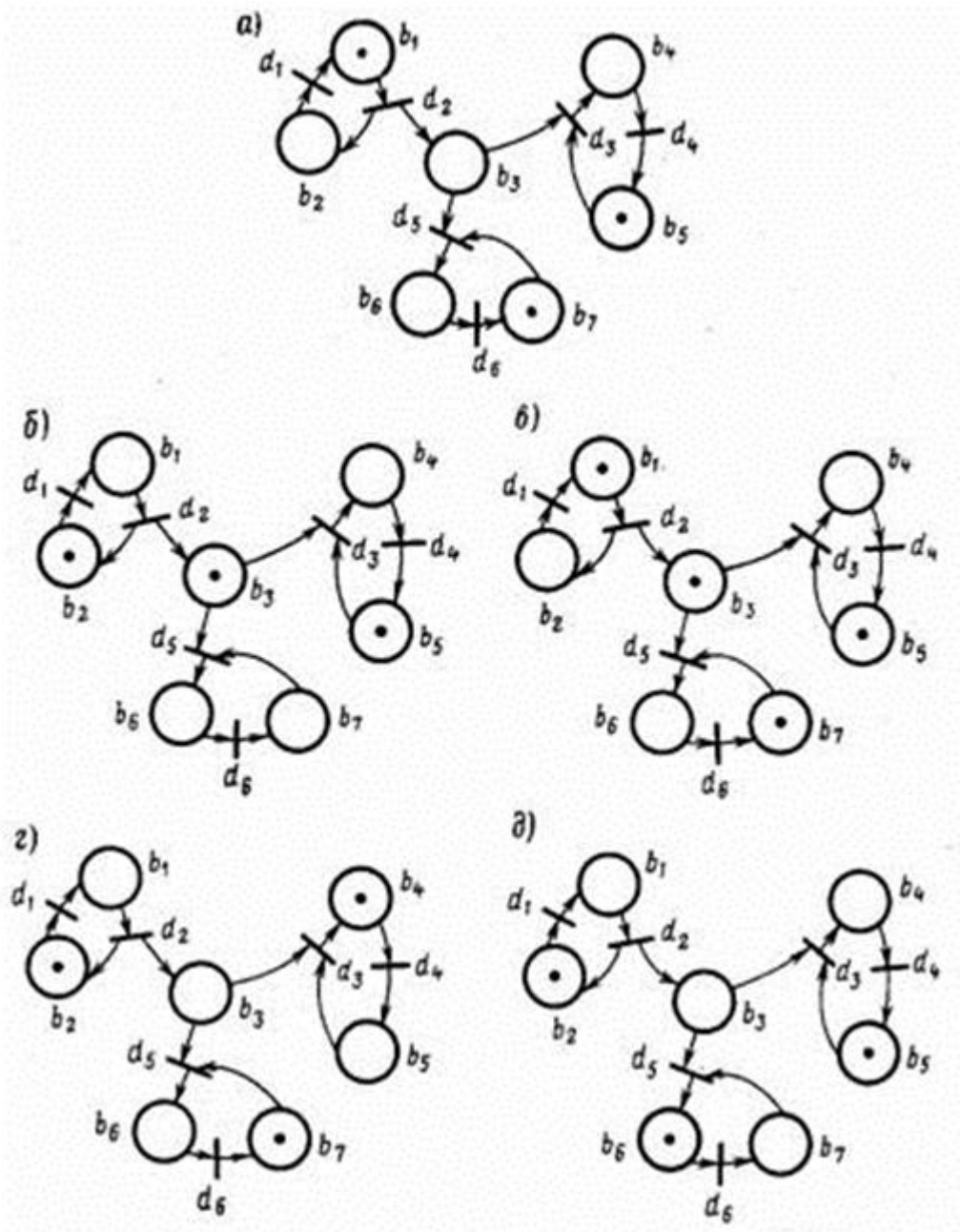


Рисунок 9.2 – Пример функционирования размеченной N-схемы

Таким образом, N-схема выполняется путем запусков переходов под управлением количества меток и их распределения в сети. Переход запускается удалением меток из его входных позиций и образованием новых меток, помещаемых в выходные позиции. Переход может запускаться только тогда, когда он разрешен. Переход называется разрешенным, если каждая из его входных позиций имеет число меток, по крайней мере равное числу дуг из позиции в переход.

Пример. Для некоторой заданной размеченной N-схемы (рисунок 9.3) с начальной маркировкой Mo = {1, 2, O, 0, 1} (рисунок 9.3, а) разрешенным является только переход d_1, а остальные переходы d_2, d_3 и d_4 — запрещенные. В результате выполнения этого перехода получим новую размеченную N-

схему (рисунок 9.3, б). Теперь разрешены переходы d_2 и d_3; в результате их запуска получим новую размеченную N-схему. Переходы d_2 и d_3 находятся в конфликте, так как запущен, может быть только один из них. Например, при запуске d_3 получим сеть, показанную на рисунке 9.3, в. Теперь разрешен только переход d_4. и получим новую размеченную сеть (рисунок 9.3, г). Теперь разрешено два перехода: d_2 и d_3 (в конфликте). Запустим переход d_2 (рисунок 9.3, д). Теперь ни один переход не может быть запущен и выполнение сети прекращается.

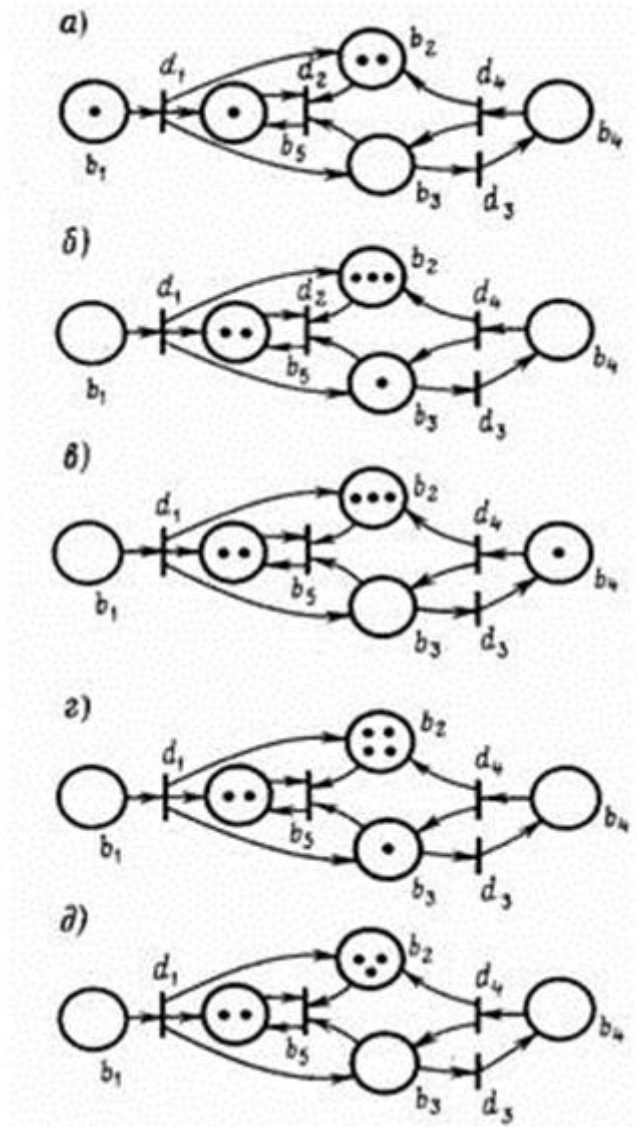


Рисунок 9.3 – Пример функционирования размеченной заданной N-схема

Важной особенностью моделей процесса функционирования систем с использованием типовых N-схем является простота построения иерархических конструкций модели. С одной стороны, каждая N-схема может рассматриваться как макропереход или макропозиция модели более высокого уровня. С другой стороны, переход, или позиция N-схемы, может детализироваться в форме отдельной подсети для более углубленного

исследования процессов в моделируемой системе. Отсюда вытекает возможность эффективного использования N-схем для моделирования параллельных и конкурирующих процессов в различных системах.

Типовые N-схемы на основе обычных размеченных сетей Петри пригодны для описания в моделируемой системе S событий произвольной длительности. В этом случае модель, построенная с использованием таких N-схем, отражает только порядок наступления событий в исследуемой системе S. Для отражения временных параметров процесса функционирования моделируемой системы S на базе N-схем используется расширение аппарата сетей Петри: временные сети, Е-сети, сети Мерлина и т. д. Детально вопросы, связанные с имитационным моделированием с использованием N-схем, будут рассмотрены далее.

Вопросы для повторения и закрепления материала

1. Каким элементами задается сеть Петри?
2. Что такое позиция?
3. Что такое переходы?
4. Каким образом задается выходная функция?
5. Каким образом задается входная функция?
6. Что такое маркированная N-схема?

Задания для самостоятельной работы

1. Определите области применений сетей Петри.

Тема 10. Моделирование процессов функционирования систем на базе N-схем

Цель изучения темы: изучить процесс моделирования систем на базе N-схем.

Задачи изучения темы:

- рассмотреть структурный подход на базе N-схем;
- рассмотреть синхронизацию событий в N-схеме;
- рассмотреть особенности моделирования параллельных процессов.

10.1. Структурный подход на базе N-схем

Рассмотрим возможности применения N-схем для формального описания процесса функционирования некоторой моделируемой системы S. Характерной особенностью N-схем является то, что с их помощью можно моделировать процессы в системах S, в которых происходит последовательная смена дискретных состояний, в том числе если эта смена происходит при выполнении разнообразных условий. Таким образом, с использованием N-схем могут быть описаны системы S, относящиеся к разным классам: аппаратные, физические, программные, экономические и т.д.

Структурный подход на базе N-схем. Применение аппарата N-схем позволяет осуществить структурный подход к построению имитационной модели системы S, при котором обеспечиваются наглядность модели, модульный принцип ее разработки (сборки), возможность перехода к автоматизированной интерактивной процедуре проектирования. Рассмотрим особенности такого подхода, используя для общности и простоты понятия, введенные ранее, на следующих примерах.

Пример. Пусть процесс функционирования некоторой реальной системы S (процессор ЭВМ, мультиплексный канал, станок в технологической цепочке н т. п.), являющийся по своей природе процессом обслуживания, представлен в виде двухфазной одноканальной Q-схемы (рисунок 10.1).

Рисунок 10.1 – Моделируемый процесс обслуживания реальной системы

Тогда этот процесс можно представить N-схемой, структура, которой показана на рисунке 10.2.

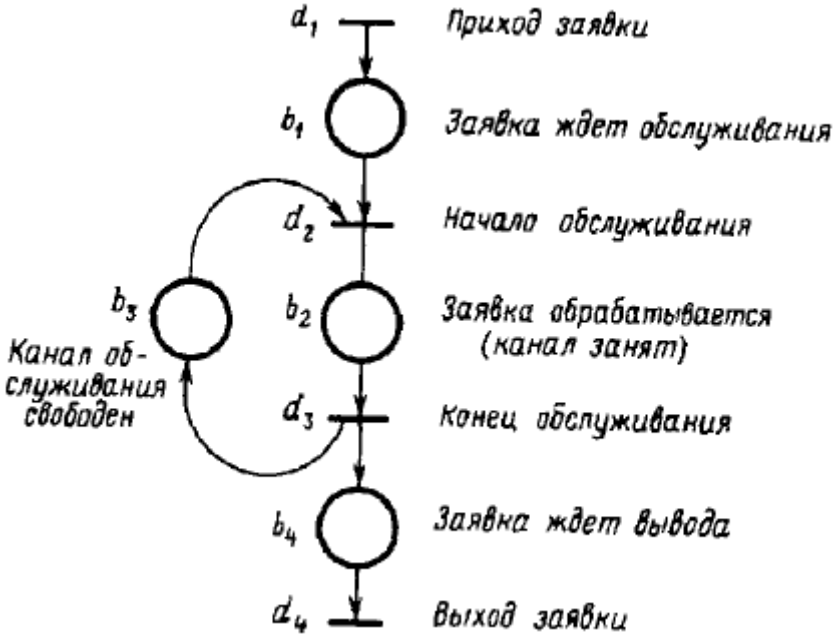


Рисунок 10.2 – Структура N-схемы

Чтобы маркировать эту структуру, нужно задать состояние системы. Пусть в накопителе H_1 находятся две заявки, в H_2 — заявок нет, канал обслуживания K_1 свободен. Такому состоянию соответствует маркировка, показанная на рисунке 10.3.

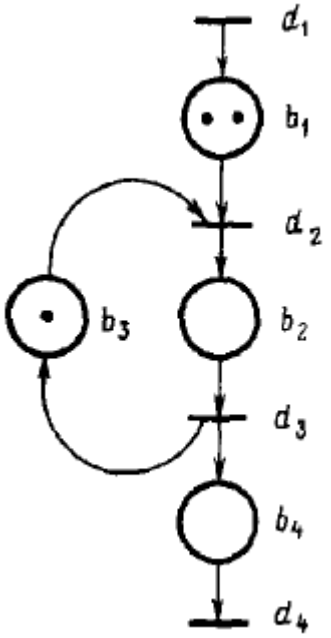


Рисунок 10.3 – Маркировка структуры

Процесс выполнения этой N-схемы моделирует процесс функционирования системы S, представленной в виде Q-схемы.

Через переход d_1 эта N-схема может быть сведена с другой N-схемой, моделирующей процесс порождения заявок на обслуживание, аналогично — по переходу d_4 — с системой потребления заявок.

Пример. Пусть имеется некоторая система S, например производственно-технологическая, процесс функционирования которой представлен в виде Q-схемы (рисунок 10.4).

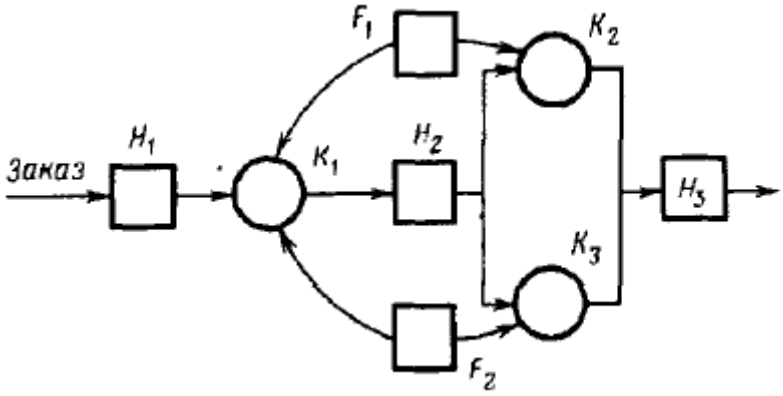


Рисунок 10.4 – Моделируемый процесс обслуживания

По технологическому циклу для выполнения заказа необходимо выполнить две фазы обслуживания: сначала обслуживание в канале $К_1$; затем либо в $К_2$, либо в $К_3$. Операторы F_1 и F_2 обслуживают (поддерживают в работоспособном состоянии) каналы, причем F_1 обслуживает $К_1$ и $К_2$, а F_2 — $К_1$ и $К_3$.

Тогда в этой системе могут быть следующие состояния:

а — заказ пришел и ждет в накопителе $Н_1$;

б — заказ обработан $К_1$ и ждет в накопителе $Н_2$;

в — заказ выполнен и находится в накопителе $Н_3$;

г — канал $К_1$ не занят;

д — канал $К_2$ не занят;

е — канал $К_3$ не занят;

ж — оператор F_1 не занят;

з — оператор F_2 не занят;

и — канал $К_1$ выполняет заказ под управлением F_1;

к — канал $К_2$ выполняет заказ под управлением F_2;

л — канал $К_2$ выполняет заказ под управлением F_1

м — канал $К_3$ выполняет заказ под управлением F_2 и могут происходить следующие события-переходы:

1 — поступление заказа;

2 — F_1 начинает выполнение заказа на $К_1$;

3 — F_1 закончил выполнение заказа на $К_1$;

4 —F_1 начинает выполнение заказа на $К_1$;

5 — F_2 закончил выполнение заказа на $К_1$;

6 — F_1 начинает выполнение заказа на $К_2$;

7 — F_1 закончил выполнение заказа на $К_2$;

8 — F_2 начинает выполнение заказа на $К_3$;

9 — F_2 закончил выполнение заказа на $К_3$;

10 — заказ отправляется на доставку.

После этого построение N-схемы происходит формально: состояниям системы соответствуют позиции N-схемы, событиям — переходы. Нанесем маркировку, соответствующую такому состоянию системы, при котором каналы свободны, операторы не заняты, в системе нет заказов (рисунок 10.5).

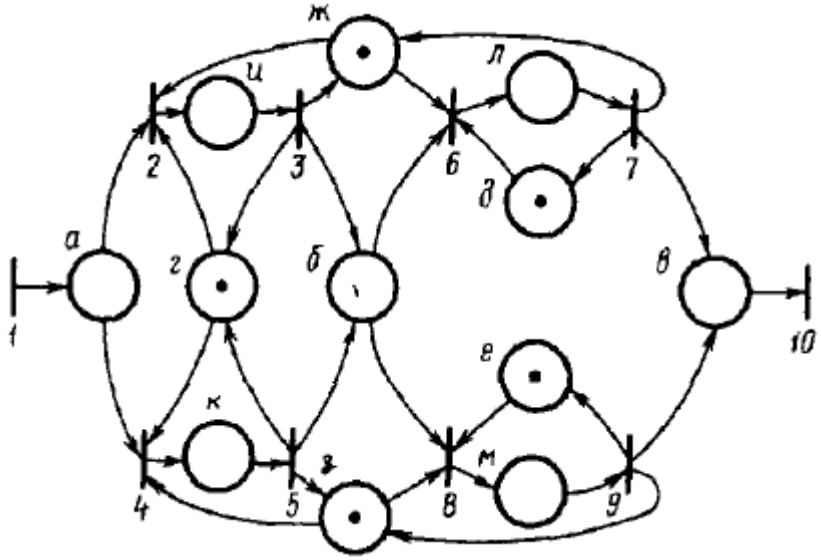


Рисунок 10.5 – Структура N-схемы

10.2. Синхронизация событий в N-схемах

Из приведенных примеров видно, что для выполнения каждого события (перехода) необходимо выполнение определенных условий. Эти условия в N-схемах (сетях Петри) называются предусловиями. Выполнение события может вызвать нарушение предусловий и привести к выполнению условий для совершения других событий — постусловий.

Для примера, рассмотренного ранее, построена таблица предусловий и постусловий (таблица 10.1). Эта таблица является описанием структуры N-схемы, удобным для ввода в ЭВМ.

Таблица 10.1 – Предусловия и постусловия

События	Предусловия	Постусловия	События	Предусловия	Постусловия
1	Нет	а	6	б, ж, д	л
2	а, ж, г	и	7	л	в, д, ж
3	и	б, ж, г	8	б, е, з	м
4	а, г, з	к	9	м	е, в, з
5	к	б, г, з	10	в	Нет

Кроме таблицы для выполнения процесса моделирования должна быть задана начальная маркировка в виде n-мерного вектора.

$$M=(0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0).$$

Процесс моделирования заключается в последовательном вычислении маркировок, получающихся в результате выполнения событий (переходов). События, по которым нет предусловий, являются входами N-схемы. Каждый вход должен быть присоединен к модели, генерирующей запуск события в соответствии с условиями, определяемыми моделируемой реальностью. В частности, это может быть другая N-схема, моделирующая процесс появления этих событий.

В N-схемах два или несколько разрушенных невзаимодействующих событий могут происходить независимо друг от друга, т. е. N-схемам и их моделям свойствен параллелизм, или одновременность. Синхронизировать события, пока этого не требует моделируемая система, нет нужды. Таким образом, N-схемы удобны для моделирования системы с распределенным управлением, в которых несколько процессов выполняются одновременно.

Другая важная особенность N-схем — это их асинхронная природа. Внутри N-схемы отсутствует измерение времени. Для простоты обычно вводят следующее ограничение. Запуск перехода (и соответствующего события) рассматривается как мгновенное событие, занимающее нулевое время, а возникновение двух событий одновременно невозможно. Моделируемое таким образом событие называется **примитивным** (примитивные события мгновенны и неодновременны).

Непримитивными называются такие события, длительность которых отлична от нуля. Любое непримитивное событие может быть представлено в

виде двух примитивных событий: «начало непримитивного события», «конец непримитивного события» — и состояния (условия) «непримитивное событие происходит».

Пример. Рассмотрим особенности использования понятий примитивных и непримитивных событий на примере обработки заданий процессором ЭВМ. На рисунке 10.6 представлены N-схемы (эквивалентные сети Петри) для моделирования обработки задания в процессоре с применением перехода, соответствующего непримитивному событию (рисунок 10.6, а), и с применением только переходов — примитивных событий (рисунок 10.6, б).

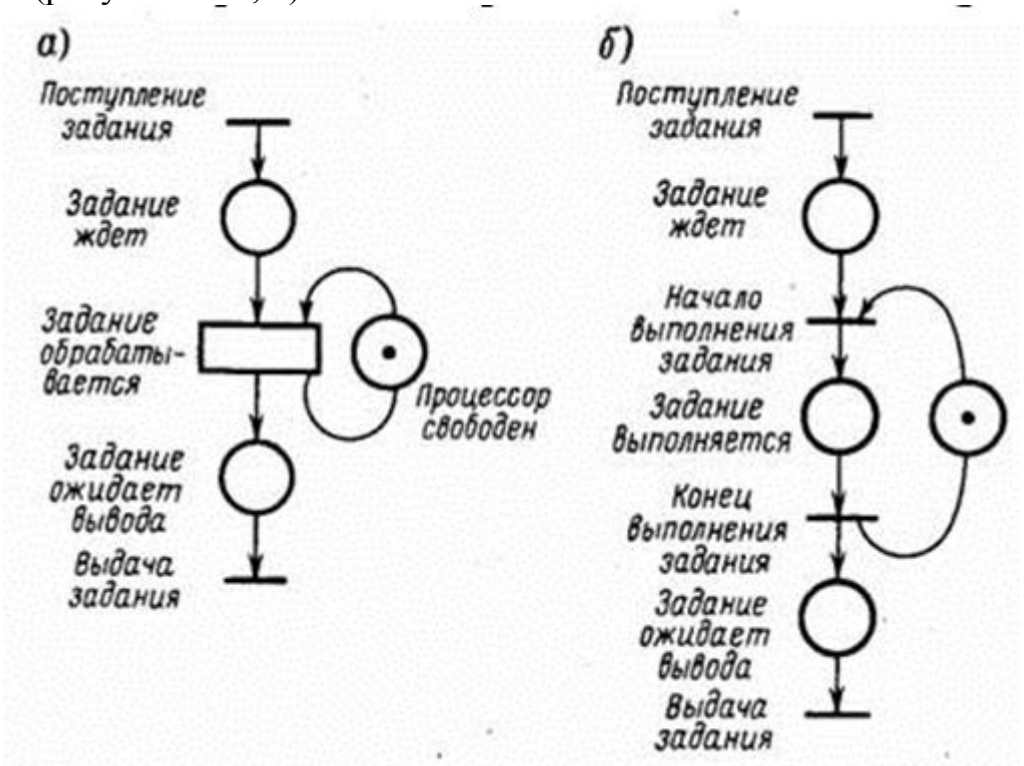


Рисунок 10.6 – Структура N-схемы с непримитивными (а) и примитивными (б) событиями

Ранее упоминалось, что в N-схемах все разрешенные переходы срабатывают одновременно и независимо. Однако с помощью N-схем можно моделировать и такие системы S, в которых порядок запуска в разрешенных переходах имеет существенное значение. Ситуация, в которой невозможно одновременное выполнение двух разрешенных переходов, изображена на рисунке 10.7, где два разрешенных перехода d_j и d_k находятся в конфликте. Может быть запущен только один из них, так как при запуске он удаляет метку из общего входа и запрещает другой переход.

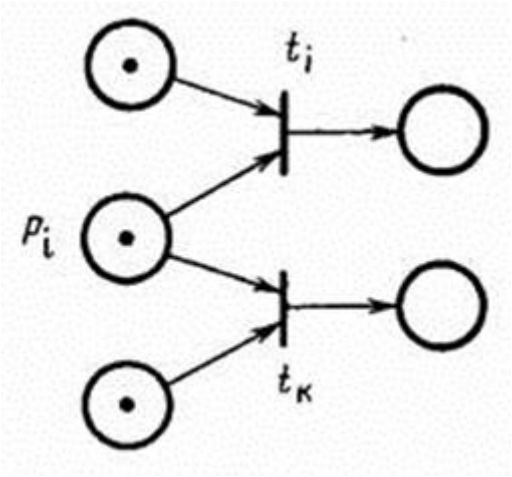


Рисунок 10.7 – Конфликт двух переходов в N-схеме

10.3. Моделирование параллельных процессов.

Возможность моделирования параллелизма и довольно простые процедуры объединения подсистем, представленных N-схемами, делают их весьма полезным инструментом моделирования сложных аппаратно-программных информационно-вычислительных комплексов и сетей, состоящих из большого количества одинаковых компонент.

Пример. Рассмотрим процесс функционирования ЭВМ с конвейерной обработкой. При построении высокопроизводительных асинхронных ЭВМ широко применяют метод конвейерной обработки чисел. Этот метод обработки подобен функционированию сборочного конвейера и особенно удобен для работы с векторами и массивами. Конвейер состоит из набора операций, которые могут выполняться одновременно в разных блоках ЭВМ. Когда операция в k- м блоке завершается, ее результат передается в (к+ 1)-й блок, а k-й блок принимает результат операции(k— 1)-го блока. Если каждая операция запускается по завершении предыдущей, то имеем дело с асинхронным способом управления конвейером. Для управления k-м блоком такого конвейера необходима информация о выполнении следующих условий:

— входной регистр заполнен;
— входной регистр пуст;
— выходной регистр заполнен;
— выходной регистр пуст;
— блок к занят;
— блок к свободен;
— пересылка осуществляется.

На рисунке 10.8,а показано, как строится N-схема для моделирования асинхронного конвейера такого типа, причем эта модель позволяет анализировать взаимодействия между блоками, игнорируя конкретные детали процессов, которые происходят внутри блоков. Эти процессы в свою

очередь могут быть промоделированы N-схемами и соединены между собой в соответствии со схемой, показанной на рисунке 10.8,б. Такая возможность построения иерархических моделей может быть весьма полезной при моделировании сложных систем S.

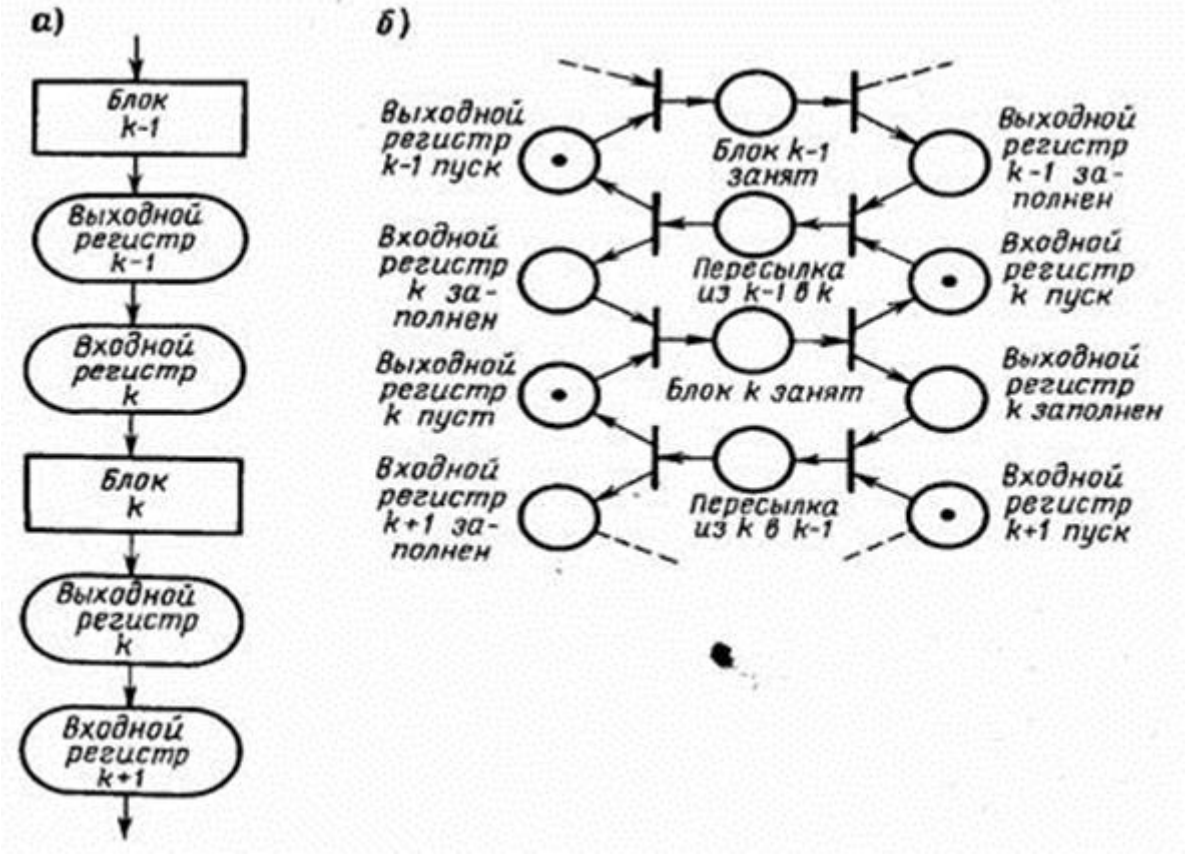


Рисунок 10.8 – Блок-схема (а) и модель (б) устройства управления асинхронной ЭВМ с конвейерной обработкой

Вопросы для повторения и закрепления материала

1. В чем заключается суть структурного подхода?
2. Каким образом производится моделирование процесса управления заявками с использованием N-схем?
3. Что такое синхронизация событий?
4. Как задается начальная маркировка сети Петри?
5. Что такое непримитивные события?
6. За счет чего сети Петри позволяют моделировать параллельные процессы?

Задания для самостоятельной работы

1. Провести обзор систем ИКТ, которые можно смоделировать при помощи N-схем.

Тема 11. Комбинированные модели (А-схемы)

Цель изучения темы: изучить комбинированные модели.

Задачи изучения темы:

- рассмотреть основные соотношения комбинированных моделей;

- рассмотреть возможные приложения А-схем.

Результат освоения темы:

Индекс компетенции	Индекс образовательного результата	Образовательный результат
ОК-16	З-1	Знать способы работы с информацией из различных источников
ПК-20	З-1	Знать соответствующий математический аппарат и инструментальные средства для обработки, анализа и систематизации информации по теме исследования
ОК-16	У-1	Уметь работать с информацией из различных источников
ПК-20	У-1	Уметь использовать соответствующий математический аппарат и инструментальные средства для обработки, анализа и систематизации информации по теме исследования
ОК-16	В-1	Владеть процессом получения и обработки информации, полученной из различных источников
ПК-20	В-1	Владеть соответствующим математическим аппаратом и инструментальными средствами для обработки, анализа и систематизации информации по теме исследования

11.1. Основные соотношения комбинированных моделей (А-схемы)

Наиболее известным общим подходом к формальному описанию процессов функционирования систем является подход, предложенный Н. П. Бусленко. Этот подход позволяет описывать поведение непрерывных и дискретных, детерминированных и стохастических систем, т. е. по сравнению с рассмотренными является обобщенным (универсальным) и базируется на понятии агрегативной системы (от англ. aggregate system), представляющей собой формальную схему общего вида, которую будем называть А-схемой.

Основные соотношения. Анализ существующих средств моделирования систем и задач, решаемых с помощью метода моделирования на ЭВМ, неизбежно приводит к выводу, что комплексное решение проблем, возникающих в процессе создания и машинной реализации модели, возможно лишь в случае, если моделирующие системы имеют в своей основе единую формальную математическую схему, т. е. А-схему. Такая схема должна одновременно выполнять несколько функций: являться адекватным

математическим описанием объекта моделирования, т. е. системы S, служить основой для построения алгоритмов и программ при машинной реализации модели M, позволять в упрощенном варианте (для частных случаев) проводить аналитические исследования.

Приведенные требования в определенной степени противоречивы. Тем не менее в рамках обобщенного подхода на основе А-схем удается найти между ними некоторый компромисс.

По традиции, установившейся в математике вообще и в прикладной математике в частности, при агрегативном подходе сначала дается формальное определение объекта моделирования — агрегативной системы, которая является математической схемой, отображающей системный характер изучаемых объектов. При агрегативном описании сложный объект (система) разбивается на конечное число частей (подсистем), сохраняя при этом связи, обеспечивающие их взаимодействие. Если некоторые из полученных подсистем оказываются в свою очередь еще достаточно сложными, то процесс их разбиения продолжается до тех пор, пока не образуются подсистемы, которые в условиях рассматриваемой задачи моделирования могут считаться удобными для математического описания. В результате такой декомпозиции сложная система представляется в виде многоуровневой конструкции из взаимосвязанных элементов, объединенных в подсистемы различных уровней.

В качестве элемента А-схемы выступает агрегат, а связь между агрегатами (внутри системы S и с внешней средой E) осуществляется с помощью оператора сопряжения R. Очевидно, что агрегат сам может рассматриваться как А-схема, т. е. может разбиваться на элементы (агрегаты) следующего уровня.

Любой агрегат характеризуется следующими множествами:

- моментов времени T;
- входных X сигналов;
- выходных Y сигналов;
- состояний Z в каждый момент времени t.

Состояние агрегата в момент времени $t \in T$ обозначается как $z(t) \in Z$, а входные и выходные сигналы — как $x(t) \in X$ и $y(t) \in Y$ соответственно.

Будем полагать, что переход агрегата из состояния $z(t_j)$ в состояние $z(t_2) \neq z(t_1)$ происходит за малый интервал времени, т. е. имеет место скачок Sz. Переходы агрегата из состояния z(t1) в z(t2) определяются собственными (внутренними) параметрами самого агрегата $h(t) \in H$ и входными сигналами $x(t) \in X$.

В начальный момент времени t_0 состояния z имеют значения, равные z°, т. е. $z^\circ = z(t_0)$, задаваемые законом распределения процесса z(t) в момент времени t_0, а именно $L[z(t_0)]$. Предположим, что процесс функционирования агрегата в случае воздействия входного сигнала x_n описывается случайным

оператором V. Тогда в момент поступления в агрегат $t_n \in T$ входного сигнала x_n можно определить состояние

$$z(t_n+0)=V[t_n, z(t_n), x_n].$$

Обозначим полуинтервал времени $t_1<t\leq t_2$ как (t_1,t_2) а полуинтервал $t_l\leq t<t_2$ — как (t_l, t_2). Если интервал времени (t_n, t_{n+l}) не содержит ни одного момента поступления сигналов, то для $t \in (t_n, t_{n+i})$ состояние агрегата определяется случайным оператором U в соответствии с соотношением

$$z(t)=U[t, t_n, z(t_n+0)].$$

Совокупность случайных операторов V и U рассматривается как оператор переходов агрегата в новые состояния. При этом процесс функционирования агрегата состоит из скачков состояний z в моменты поступления входных сигналов x (оператор V) и изменений состояний между этими моментами t_n и t_{n+1} (оператор U). На оператор U не накладывается никаких ограничений, поэтому допустимы скачки состояний δz в моменты времени, не являющиеся моментами поступления входных сигналов x. В дальнейшем моменты скачков δz будем называть особыми моментами времени t_δ, а состояния z (t_s) — особыми состояниями А-схемы. Для описания скачков состояний δz в особые моменты времени U будем использовать случайный оператор W, представляющий собой частный случай оператора U, т. е.

$$z(t_\delta+0)=W[t_\delta, z(t_\delta)].$$

В множестве состояний Z выделяется такое подмножество $Z^{(y)}$, что если $z(t_\delta)$ достигает $Z^{(Y)}$, то это состояние является моментом выдачи выходного сигнала, определяемого оператором выходов

$$y=G[t_\delta, z(t_\delta)].$$

Таким образом, под **агрегатом** будем понимать любой объект, определяемый упорядоченной совокупностью рассмотренных множеств T, X, Y, Z, $Z^{(y)}$, H и случайных операторов V, U, W, G.

Последовательность входных сигналов, расположенных в порядке их поступления в А-схему, будем называть входным сообщением или x-сообщением. Последовательность выходных сигналов, упорядоченную относительно времени выдачи, назовем выходным сообщением или y-сообщением.

11.2 Возможные приложения А-схем

Существует класс больших систем, которые ввиду их сложности не могут быть формализованы в виде математических схем одиночных

агрегатов, поэтому их формализуют некоторой конструкцией из отдельных агрегатов A_n, $n=\overline{1,N_A}$, которую назовем агрегативный системой или А-схемой. Для описания некоторой реальной системы S в виде А-схемы необходимо иметь описание как отдельных агрегатов A_n так и связей между ними.

Пример. Рассмотрим А-схему, структура которой приведена на рисунке 11.1. Функционирование А-схемы связано с переработкой информации, передача последней на схеме показана стрелками. Вся информация, циркулирующая в А-схеме, делится на внешнюю и внутреннюю. Внешняя информация поступает от внешних объектов, не являющихся элементами рассматриваемой схемы, а внутренняя информация вырабатывается агрегатами самой А-схемы. Обмен информацией между А-схемой и внешней средой Е происходит через агрегаты, которые называются полюсами А-схемы. При этом различают **входные полюсы** А-схемы, представляющие собой агрегаты, на которые поступают х-сообщения (агрегаты A_1, A_2, A_6), и **выходные полюсы** А-схемы, выходная информация которых является у- сообщениями (агрегаты A_1, A_3, A_4, A_5, A_6). Агрегаты, не являющиеся полюсами, называются **внутренними.**

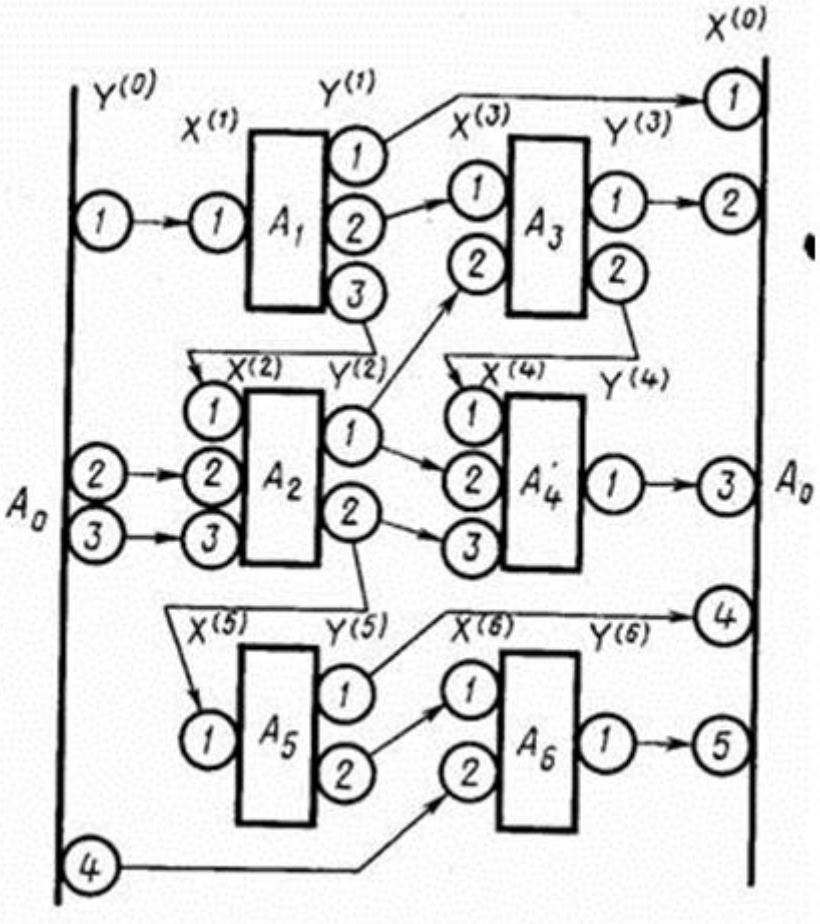


Рисунок 11.1 – Структура агрегативной системы

Каждый n-й агрегат А-схемы A_n имеет входные контакты, на которые поступает совокупность элементарных сигналов $x_i(t)$, $i=\overline{1,I_n}$ одновременно возникающих на входе элемента, и выходные контакты, с которых снимается совокупность элементарных сигналов $y_j(t)$, $j=\overline{1,J_n}$. Таким образом, каждый агрегат А-схемы A_n имеет I_n входных и J_n выходных контактов.

Описание отдельного агрегата уже рассмотрено, поэтому для построения формального понятия А-схемы остается выбрать достаточно удобные способы математического описания взаимодействия между агрегатами. Для этого введем ряд предположений о закономерностях функционирования А-схем, хорошо согласующихся с опытом исследования реальных сложных систем:

1) взаимодействие между А-схемой и внешней средой Е, а также между отдельными агрегатами внутри системы S осуществляется при передаче сигналов, причем взаимные влияния, имеющие место вне механизма обмена сигналами, не учитываются;

2) для описания сигнала достаточно некоторого конечного набора характеристик;

3) элементарные сигналы мгновенно передаются в А-схеме независимо друг от друга по элементарным каналам;

4) к входному контакту любого элемента А-схемы подключается не более чем один элементарный канал, к выходному контакту — любое конечное число элементарных каналов при условии, что ко входу одного и того же элемента А-схемы направляется не более чем один из упомянутых элементарных каналов.

Взаимодействие А-схемы с внешней средой Е рассматривается как обмен сигналами между внешней средой Е и элементами А-схемы. В соответствии с этим внешнюю среду Е можно представить в виде фиктивного элемента системы A_o, вход которого содержит I_o входных контактов $X_i^{(0)}, i=\overline{1,I_0}$, а выход —$J_0$ выходных контактов $Y_i^{(0)}, i=\overline{1,J_0}$. Сигнал, выдаваемый А-схемой во внешнюю среду Е, принимается элементом A_o как входной сигнал, состоящий из элементарных сигналов $x_1^{(0)}(t), x_2^{(0)}(t), ..., x_{I0}^{(0)}(t)$. Сигнал, поступающий в А-схему из внешней среды Е, является выходным сигналом элемента Ао и состоит из элементарных сигналов $y_1^{(0)}(t), y_2^{(0)}(t), ..., y_{J0}^{(0)}(t)$.

Таким образом, каждый A_n, (в том числе и A_o) как элемент А-схемы в рамках принятых предположений о механизме обмена сигналами достаточно охарактеризовать множеством входных контактов $X_1^{(n)}, X_2^{(n)}, ..., X_I^{(n)}$, которое обозначим $\{X_I^{(n)}\}$, и множеством выходных контактов $Y_1^{(n)}, Y_2^{(n)}, ..., Y_J^{(n)}$ которое обозначим $\{Y_J^{(n)}\}$, где $n=\overline{0,N_A}$. Полученная пара множеств $\{X_I^{(n)}\}$, $\{Y_J^{(n)}\}$ является математической моделью элемента A_n, используемого для

формального описания сопряжения его с прочими элементами А-схемы и внешней средой Е. В силу предположения о независимости передачи сигналов каждому входному контакту

$$X_I^{(n)} \in \bigcup_{n=0}^{N_A} \{X_I^{(n)}\}$$

соответствует не более чем один выходной контакт

$$Y_i^k \in \bigcup_{n=0}^{N_A} \{Y_j^{(n)}\},$$

где $\bigcup_{n=0}^{N_A} \{X_j^{(n)}\}$ - — множество входных контактов всех элементов А--

$$Y_i^k \in \bigcup_{n=0}^{N_A} \{Y_j^{(n)}\},$$

схемы и внешней среды — множество выходных контактов всех элементов А-схемы и внешней среды Е, с которыми она связана элементарным каналом; k, n=0, NA.

Поэтому можно ввести однозначный оператор $Y_i^k \in \bigcup_{n=0}^{N_A} \{Y_j^{(n)}\},$ с областью

$$Y_i^k \in \bigcup_{n=0}^{N_A} \{Y_j^{(n)}\},$$

определения в множестве и областью значений в множестве сопоставляющий входному контакту $X_I^{(n)}$ выходной контакт $Y_J^{(n)}$ связанный с ним элементарным каналом. Если в А-схеме к контакту $X_I^{(n)}$ не подключен никакой элементарный канал, то оператор R не определен на этом контакте $X_I^{(n)}$. Оператор R называется оператором сопряжения элементов (агрегатов) в А-схему. Совокупность множеств $\{X_I^{(n)}\}$, $\{Y_J^{(k)}\}$ и оператор R образуют схему сопряжения элементов в систему S.

Рассмотрим, оператор сопряжения для А-схемы, структура которой показана на рисунке 11.1. Оператор сопряжения R можно задать в виде таблицы, в которой на пересечении строк с номерами элементов (агрегатов) n и столбцов с номерами контактов i располагаются пары чисел к,l, указывающие номер элемента k и номер контакта l, с которым соединен контакт $X_I^{(n)}$ (таблица 11.1).

Таблица 11.1 – Оператор сопряжения для А-схемы (рисунок 9.1)

n	i				
	1	2	3	4	5
0	1,1	3,1	4,1	5,1	6,1
1	0,1				
2	1,3	0,2	0,3		
3	1,2	2,1			
4	3,2	2,1	2,2		
5	2,2				
6	5,2	0,4			

Если столбцы и строки такой таблицы пронумеровать двойными индексами n,i и k,l соответственно и на пересечении помещать для контактов n,i и k,l, соединенных элементарным каналом и 0 в противном случае, то получим матрицу смежности ориентированного графа, вершинами которого являются контакты агрегатов, а дугами — элементарные каналы А-схемы.

Рассмотренная схема сопряжения агрегатов в А-схему, заданная совокупностью множеств $\{X_I^{(n)}\}$, $\{Y_J^{(n)}\}$ и оператором R, является одноуровневой схемой сопряжения. В более сложных случаях могут быть использованы многоуровневые иерархические схемы сопряжения. Схема сопряжения агрегата, определяемая оператором R, может быть использована для описания весьма широкого класса объектов. Однако взаимодействие элементов реальных систем даже в рамках механизма обмена сигналами не сводится к одному лишь сопряжению. Помимо сопряжения контактов серьезную роль играют также согласование совокупности элементарных сигналов, поступающих в элементарный канал от выходных контактов и воспринимаемых входными, а также влияние реальных средств передачи сигналов на их содержание. Кроме того, оказываются полезными некоторые дополнительные ограничения на структуру сопряжения агрегатов системы S с внешней средой E. Поэтому с практической точки зрения представляет интерес понятие А-схемы как типовой математической, отражающей наши представления о взаимодействии реальных объектов в рамках механизмов обмена сигналами.

Упорядоченную совокупность конечного числа агрегатов A_n $n=\overline{1,N_A}$ системы S, агрегата A_o, характеризующего внешнюю среду E, и оператора R, реализующего отображение $\bigcup_{n=0}^{N_A}\{X_I^{(n)}\}\to\bigcup_{n=0}^{N_A}\{Y_I^{(k)}\}$, будем называть А-схемой при следующих условиях:

1) для любых $X_i^{(0)}\in\{X_i^{(0)}\}$ и $Y_l^{(0)}\in\{Y_l^{(0)}\}$ в данной А-схеме $Y_l^{(0)}=R(X_i^{(n)})$, то

2) если $\dot{Y}_l^k\in\dot{X}_l^{(n)}$, (11.1)

где $Y_i^{(k)}$ — соответствующие множества элементарных сигналов; для любого момента t' выдачи непустого элементарного сигнала

$$Y_i^k(t') \in \dot{Y}_i^k \qquad (11.2)$$

Имеет место

$$t' \in (T^k \bigcap T^{(n)}) \qquad (11.3)$$

$$Y_i^{(k)}(t') = x_i^{(n)}(t') \qquad (11.4)$$

Где $x_i^{(n)}(t') \in X_i^{(n)}$.

Ограничение (11.1) относится к структуре сопряжения агрегатов А-схемы системы S с внешней средой E и требует, чтобы каждый элементарный канал, передающий сигналы во внешнюю среду, начинался в одном из выходных контактов одного из агрегатов системы, каждый элементарный канал, передающий сигналы из внешней среды, заканчивался на одном из входных контактов А-схемы. Ограничение (11.2) предусматривает, что сигналы в А-схеме передаются непосредственно от одного агрегата к другому без устройств, способных отсеивать сигналы по каким-либо признакам. Ограничение (11.3) относится к согласованию функционирования агрегатов А-схемы во времени. Ограничение (11.4) предусматривает, что сигналы между агрегатами А-схемы передаются мгновенно, без искажений и перекодирования, изменяющего структуру сигнала. Для многих реальных систем ограничения (11.2) и (11.4) оказываются несправедливыми. Для того чтобы А-схема была адекватной моделью реального объекта, достаточно описать селектирующие устройства, реальные средства передачи сигналов и всевозможные вспомогательные устройства как самостоятельные агрегаты, связи между которыми удовлетворяют перечисленным ограничениям.

Пример. Рассмотрим представление некоторой системы в виде отдельного агрегата. Для того чтобы упростить описание объекта моделирования и проследить связи с уже рассмотренными схемами, воспользуемся в качестве объекта такого моделирования схемой массового обслуживания (Q-схемой) и представим ее в виде агрегата (А-схемы). Для определенности полагаем, что имеется однофазная одноканальная система SQ, показанная на рисунке 9.1. В моменты времени t_j, образующие однородный поток случайных событий, в прибор (П) поступают заявки, каждая из которых характеризуется случайным параметром e_j. Если обслуживающий канал (К) занят, то заявка поступает в накопитель (Н) и может ждать там не более чем $\gamma_j = \varphi(e_j, h)$, где h — параметр, характеризующий производительность системы обслуживания. Если к

моменту $(t_j + \gamma_j)$ заявка не будет принята к обслуживанию, то она теряется. Время обслуживания заявки $\tau_j = \psi(e_j, h)$.

При представлении этой Q-схемы в виде А-схемы опишем ее состояния вектором $\vec{Z}(t) \in Z$ со следующими компонентами: $z_1(t)$ — время, оставшееся до окончания обслуживания заявки, которая находится в канале (К); $z_2(t)$ — количество заявок в приборе (П); $z_m(t)=e_k$, где e_k — параметр к-й заявки в накопителе (Н); $z_e(t)$ — оставшееся время ожидания к-й заявки в накопителе (Н) до момента, когда она получит отказ, m=1+2к, l=2+2k, $k = \overline{1, z_2(t)-1}$.

Входные сигналы (заявки) поступают в А-схему в моменты t_j и принимают значения $x_j=e_j$. Рассмотрим случайные операторы V, U и G, описывающие такой агрегат. Пусть в момент t_j поступает новая заявка. Тогда оператор V можно записать следующим образом:

$$
\left.\begin{array}{l}
z_1(t_j+0)=z_1(t_1),\\
z_2(t_j+0)=z_2(t_j)+1,\\
\left.\begin{array}{l} z_m(t_j+0)=z_m(t_j),\\ z_l(t_j+0)=z_l(t_j), \end{array}\right\} k \leqslant z_2(t_j),\\
z_{1+2k}(t_j+0)=e_j,\\
z_{2+2k}(t_j+0)=\varphi(e_j,\ h),
\end{array}\right\} z_2(t_j)>0;
$$

$$
\left.\begin{array}{l}
z_1(t_j+0)=\psi(e_j,\ h)\\
z_2(t_j+0)=1,
\end{array}\right\} z_2(t_j)=0.
$$

Пусть $t = t_{\delta 1}$, т. е. обслуживание очередной заявки окончено. Этот момент является особым, так как в этот момент z(t) достигает $Z^{(Y)}$, т. е. $z_1(t_{\delta 1}) = 0$. Поэтому скачок состояний $z(t_{\delta 1})$j определяется оператором W вида

$$
\left.\begin{array}{l}
z_1(t_{\delta_1}+0)=\varphi(e'_{k+1},\ h),\\
z_2(t_{\delta_1}+0)=z_2(t_{\delta_1})-1,\\
\left.\begin{array}{l} z_m(t_{\delta_1}+0)=z_{m+2}(t_{\delta_1}),\\ z_l(t_{\delta_1}+0)=z_{l+2}(t_{\delta_1}), \end{array}\right\} 1 \leqslant k \leqslant z_3(t_{\delta_1}),\\
\left.\begin{array}{l} z_1(t_{\delta_1}+0)=z_1(t_{\delta_1}),\\ z_2(t_{\delta_1}+0)=0, \end{array}\right\} z_2(t_{\delta_1})=0.
\end{array}\right\} z_2(t_{\delta_1})>0;
$$

Рассмотрим еще один особый момент времени $t_{\delta 2}$, не являющийся моментом поступления входного сигнала. В момент $t_{\delta 2}$, когда истекает время

ожидания одной из заявок, например i-й, число заявок в системе уменьшается на 1. Состояние А-схемы z_2 ($t_{\delta 2}+0$) определяется оператором W" вида

$$z_1(t_{\delta_2}+0)=z_1(t_{\delta_2}),$$
$$z_2(t_{\delta_2}+0)=z_2(t_{\delta_2})-1,$$
$$\left.\begin{aligned} z_m(t_{\delta_2}+0)&=z_m(t_{\delta_2}),\\ z_l(t_{\delta_2}+0)&=z_l(t_{\delta_2}),\end{aligned}\right\} k<i,$$
$$\left.\begin{aligned} z_m(t_{\delta_2}+0)&=z_{m+2}(t_{\delta_2}),\\ z_l(t_{\delta_2}+0)&=z_{l+2}(t_{\delta_2}),\end{aligned}\right\} i\leqslant k\leqslant z_2(t_{\delta_2}).$$

В полуинтервалах (t_n t_{n+1}) между особыми моментами времени t_n и t_{n+1} к которым относятся моменты поступления в А-схему входных сигналов и выдачи выходных сигналов, состояния А-схемы изменяются по закону, задаваемому оператором U, который можно записать так:

$$z_1(t)=z_1(t_n+0)-(t-t_n),$$
$$z_2(t)=z_2(t_n+0),$$
$$z_m(t)=z_m(t_n+0),$$
$$z_l(t)=z_l(t_n+0)-(t-t_n).$$

Выходными сигналами А-схемы будем считать сведения о заявках, покидающих прибор (П). Пусть y (y^1, y^2), где y^1 — признак (y^1 = 1, если заявки обслужены; y^1 =0, если заявки не обслужены); y^2 — совокупность сведений о заявке, например y^2 = (e_j, h, t_δ), т. е. заявки поступили в систему обслуживания с параметром e_j, обслуживались при значении параметра системы h, покинули систему в момент t_g. Таким образом, действия оператора G сводятся к выбору признака y^1 и формированию сведений о заявке y^2. Для моментов $t_{\delta 1}$ и $t_{\delta 2}$ выходной сигнал y определяется параметром G и может быть записан в следующем виде:

$$y=(1,\ e_j,\ h,\ t_{\delta_1}),\ y=(0,\ e_j,\ h,\ t_{\delta_2}),$$

где $t_{\delta 1}$ находят из z_1 ($t_{\delta 1}$)=0, а t_δ — из z_1($t_{\delta 2}$)=0.

На основании состояний системы S_Q можно оценить ее вероятностно-временные характеристики, например вероятность нахождения в

обслуживающем приборе (Н) заданного числа заявок, среднее время ожидания заявок в накопителе (Н) и т. д.

Таким образом, дальнейшее использование обобщенной типовой математической схемы моделирования, т. е. А-схемы, в принципе не отличается от рассмотренных ранее D-, F-, P-, N-, Q-схем. Для частного случая, а именно для кусочно-линейных агрегатов, результаты могут быть получены аналитическим методом. В более сложных случаях, когда применение аналитических методов неэффективно или невозможно, прибегают к имитационному методу, причем представление объекта моделирования в виде А-схемы может являться тем фундаментом, на котором базируется построение имитационной системы и ее внешнего и внутреннего математического обеспечения. Стандартная форма представления исследуемого объекта в виде А-схемы приводит к унификации не только алгоритмов имитации, но и к возможности применять стандартные методы обработки и анализа результатов моделирования системы S.

Рассмотренные примеры использования типовых математических схем (D-, F-, P-, Q-, N-, A-схем) позволяют формализовать достаточно широкий класс больших систем, с которыми приходится иметь дело в практике исследования и проектирования сложных систем.

Вопросы для повторения и закрепления материала

1. Что такое агрегативная система?
2. Чем характеризуется агрегат?
3. Приведите пример структуры А-схемы.

Задания для самостоятельной работы

1. Найти примеры использования А-схем.

Тема 12. Иерархические модели процессов функционирования систем

Цель изучения темы: изучить принцип иерархического моделирования процессов функционирования систем.

Задачи изучения темы:

- рассмотреть блочную конструкцию модели;
- рассмотреть моделирующий алгоритм.

12.1. Блочная конструкция модели

При машинной реализации любой из рассмотренных типовых математических схем (D, F, P, Q, N, A-схем) необходимо решить вопрос о взаимодействии блоков модели Мм при использовании аналитического, имитационного или комбинированного (аналитико-имитационного) подходов.

Блочная конструкция модели. Рассмотрим машинную модель Мм системы S как совокупность блоков $\{m_i\}$, i=(1, n). Каждый блок модели можно охарактеризовать конечным набором возможных состояний $\{z_0\}$, в которых он может находиться. Пусть в течение рассматриваемого интервала времени (0, T), т. е. времени прогона модели, блок изменяет состояния в моменты времени $t_i^{(j)} \le T$, где j — номер момента времени. Вообще моменты времени смены состояний блока m_i, можно условно разделить на три группы:

1) случайные моменты, связанные с внутренними свойствами части системы S, соответствующей данному блоку;

2) случайные моменты, связанные с изменением состояний других блоков (включая блоки, имитирующие воздействия внешней среды E);

3) детерминированные моменты, связанные с заданным расписанием функционирования блоков модели.

Моментами смены состояний модели Мм в целом $t^{(k)} \le T$ будем считать все моменты изменения состояний блоков $\{m_i\}$, т. е. $\{t_i^{(j)}\} \in \{t^k\}$, i = (1, n). Пример для модели с тремя блоками m_1, m_2 и m_3 показан на рисунке 12.1.

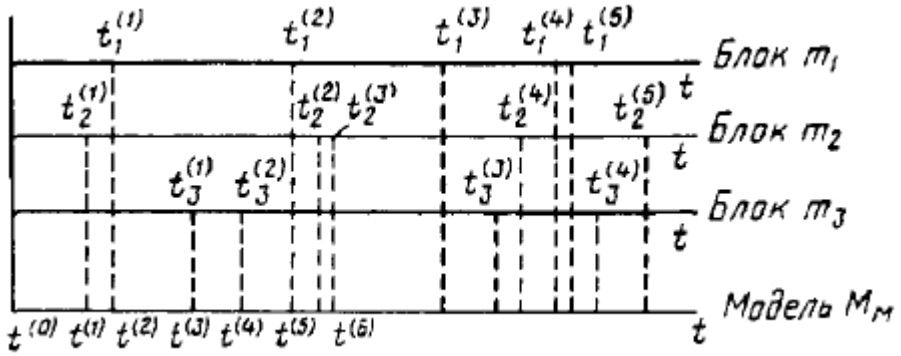


Рисунок 12.1 – Смена состояний модели для случая трех блоков

При этом моменты $t_i^{(j)}$ и t^k являются моментами системного времени, т.е. времени, в котором функционирует моделируемая система S, но не моментами машинного времени. Мгновенные изменения состояний модели во время дискретного события (особого состояния) возможны только при моделировании в системном времени.

При моделировании для каждого блока модели m_i, i = (1, n), необходимо фиксировать момент очередного перехода блока в новое состояние $t_i^{(j)}$ и номер этого состояния s_i, образуя при этом массив состояний. Этот массив отражает динамику функционирования модели системы, так как в нем фиксируются все изменения в процессе функционирования моделируемой системы S по времени. В начале моделирования в массив состояний должны быть занесены исходные состояния, заданные начальными условиями.

При машинной реализации модели Мм ее блоки, имеющие аналогичные функции, могут быть представлены в виде отдельных программных модулей. Работа каждого такого модуля имитирует работу всех

однотипных блоков. В общем случае при числе блоков модели n можно получить набор машинных модулей $l \leq n$. Таким образом, каждому блоку или элементу модели будет соответствовать некоторый модуль или "стандартная подпрограмма", число которых не будет превосходить числа блоков модели.

12.2. Моделирующий алгоритм

Типовая укрупненная схема моделирующего алгоритма, построенного по блочному принципу, для систем с дискретными событиями приведена на рисунке 12.2.

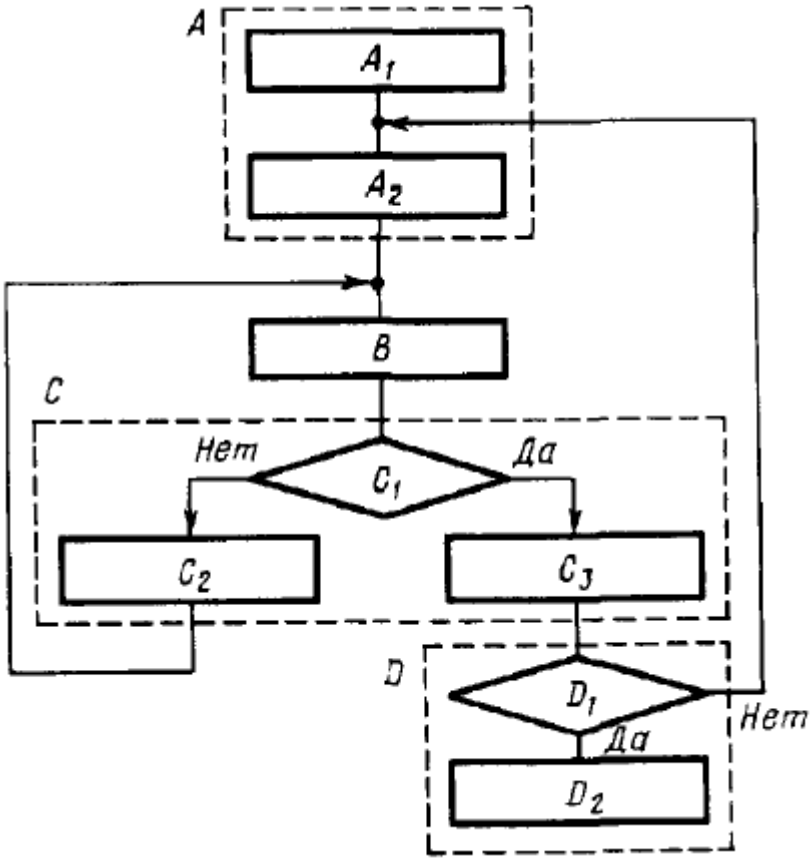


Рисунок 12.2 - Типовая укрупненная схема моделирующего алгоритма

Эта схема содержит следующие укрупненные модули:

- A - модуль задания начальных значений состояний, содержащий два подмодуля (A1 - для задания начальных состояний моделируемого варианта и A2 - для задания начальных состояний для одного прогона модели);

- B - модуль определения очередного момента смены состояния, осуществляющий просмотр массива состояний и выбирающий блок модели m_i, i = (1, n), с минимальным временем смены состояния $\min t_i^{(j)}$;

- C - модуль логического переключения, содержащий три подмодуля (C1 - для логического перехода по номеру блока модели i или по времени T, т. е. для решения вопроса о завершении прогона; C2 - для фиксации информации о состояниях, меняющихся при просмотре блока, а также для определения момента следующей смены состояния блока m_i и номера

следующего особого состояния s_0; С3 - для завершения прогона в случае, когда $t_i^{(j)} \geq T$, фиксации и предварительной обработки результатов моделирования);

- D - модуль управления и обработки, содержащий два подмодуля (D1 - для проверки окончания исследования варианта модели Мм по заданному числу прогонов или по точности результатов моделирования; D2 - для окончательной обработки информации, полученной на модели Мм и выдачи результатов моделирования).

Данная укрупненная схема моделирующего алгоритма соответствует статике моделирования. При необходимости организации моделирования последовательностей вариантов модели Мм и проведении оптимизации моделируемой системы S, например, на этапе ее проектирования, т. е. для решения вопросов, относящихся к динамике моделирования, следует добавить внешний цикл для варьирования структуры, алгоритмов и параметров модели Мм.

Пример. Рассмотрим модульный принцип реализации модели S, формализованной в виде Q-схемы. Пусть имеется $L^{Ф}$-фазная многоканальна Q-схема без потерь с $L^{И}$-входными потоками заявок. В каждой фазе имеется L_j^{K}, j = (1, $L^{Ф}$), каналов обслуживания. Определить распределения времени ожидания заявок в каждой фазе и времени простоя каждого обслуживающего канала.

В качестве блоков модели Мм будем рассматривать:

- $m^{И}$ - блоки источников заявок, имитирующие $L^{И}$ входных потоков;
- m^{K} - блоки каналов обслуживания, имитирующие функционирование каналов;
- m^{B} - блок взаимодействия, отражающий взаимосвязь всех блоков машинной модели Мм.

При этом в массиве состояний будем фиксировать моменты поступления заявок, освобождения каналов и окончания моделирования, т. е. количество элементов этого массива будет равно

$$L^{И} + \sum_{j=1}^{L^{Ф}} L_j^{K} + 1.$$

Схема моделирующего алгоритма для данного примера приведена на рисунке 12.3. Как видно из схемы, в подмодуле C_2 предусмотрены три вида процедур: C_2', C_2'' и C_2'''. Первая процедура C_2' работает при поступлении заявки из любого входного потока, вторая процедура C_2' работает в момент освобождения канала любой фазы обслуживания, кроме последней, третья процедура C_2''' работает при освобождении канала последней фазы, т. е. при окончании обслуживания заявки Q-схемой.

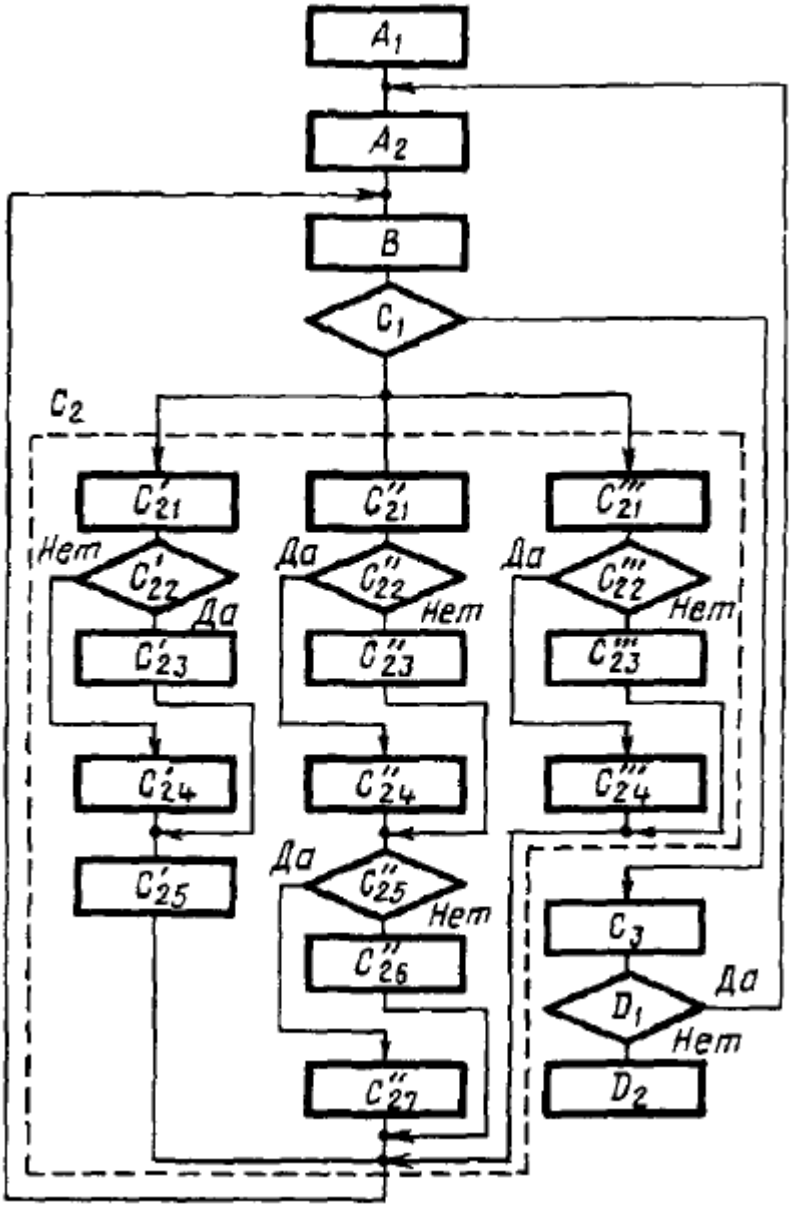


Рисунок 12.3 – Схема моделирующего алгоритма многофазной многоканальной Q-схемы

Рассмотрим более подробно операторы процедур C_2', C_2'' и C_2'''. Оператор C'_{21} определяет принадлежность заявки к одному из $L^{И}$ входных потоков, генерируемых модулем В. Оператор C'_{22} проверяет, есть ли на первой фазе очередь свободных каналов обслуживания. Если очередь есть, то управление передается оператору C'_{23}, в противном случае - оператору C'_{24}. Оператор C'_{23} фиксирует момент поступления заявки в массиве очереди заявок первой фазы. Оператор C'_{24} выбирает номер канала из массива очереди канала первой фазы, уменьшая ее длину на единицу, вычисляет и фиксирует длительность простоя канала, определяет длительность обслуживания и засылает новый момент освобождения канала в массив

состояний. Оператор C'_{25} определяет новый момент поступления заявки и засылает его в соответствующую ячейку массива состояний.

Оператор C_{21}'' служит для определения j-й фазы и k-го канала, $\overline{j=2,L^{\Phi}-1}, \quad \overline{k=1,L_j^K}$. Оператор C_{22}'' проверяет наличие очереди заявок на выбранной j-й фазе. При отсутствии очереди управление передается оператору C_{23}'', а при ее наличии - оператору C''_{24}. Оператор C''_{23} засылает момент освобождения канала в массив очереди каналов j-й фазы, уменьшает длину очереди на единицу и фиксирует время ожидания выбранной заявкой начала ее обслуживания. Далее определяется длительность обслуживания этой заявки освободившимся каналом, вычисляется и засылается в массив состояний новый момент освобождения канала. Операторы C''_{25}, C''_{26} и C''_{27} выполняют те же действия с заявкой, обслуживаемой на j-й фазе, что и операторы C'_{22}, C'_{23} и C'_{24} с заявкой, которая поступила в первую фазу Q-схемы.

Оператор C'''_{24} настраивает операторы этой процедуры C'''_{22}, C'''_{23} и C'''_{24} на выбранный канал обслуживания последней, L^{Φ}-й, фазы. Работа операторов C'''_{22}, C'''_{23} и C'''_{24} аналогична работе операторов C''_{22}, C''_{23} и C''_{24}.

Назначение остальных подмодулей алгоритма не отличается от рассмотренного ранее для моделирующего алгоритма, приведенного на рисунке 12.2.

Построение моделирующего алгоритма по блочному принципу позволяет за счет организации программных модулей уменьшить затраты времени на моделирование системы S, так как машинное время в этом случае не тратится на просмотр повторяющихся ситуаций. Кроме того, данная схема моделирующего алгоритма получается проще, чем в случае, когда модули не выделяются.

Автономность процедур подмодуля C_2 позволяет проводить их параллельное программирование и отладку, причем описанные процедуры могут быть стандартизованы, положены в основу разработки соответствующего математического обеспечения моделирования и использованы для автоматизации процесса моделирования систем. Если говорить о перспективах, то блочный подход создает хорошую основу для автоматизации имитационных экспериментов с моделями систем, которая может полностью или частично охватывать этапы формализации процесса функционирования системы S, подготовки исходных данных для моделирования, анализа свойств машинной модели Мм системы, планирования и проведения машинных экспериментов, обработки и интерпретации результатов моделирования системы. Такие машинные эксперименты должны носить научный, а не эмпирический характер, т. е. в результате должны предлагаться не только методы решения конкретной поставленной задачи, но и указываться границы эффективного использования этих методов, оцениваться их возможности. Лишь только автоматизация процесса моделирования создаст перспективы использования

моделирования в качестве инструмента повседневной работы системного специалиста.

Вопросы для повторения и закрепления материала

1. Что такое блочная конструкция модели?
2. Приведите пример типовой укрупненной схемы моделирующего алгоритма.
3. Опишите модульный принцип построения модели.

Задания для самостоятельной работы

1. Опишите системы, в которых необходимо применение блочных конструкций.

Тема 13. Методика разработки и машинной реализации моделей систем

Цель изучения темы: изучить методику разработки и машинной реализации моделей систем.

Задачи изучения темы:

- рассмотреть методологические аспекты моделирования;
- рассмотреть этапы моделирования систем;
- изучить основы построения концептуальных моделей систем и их формализации.

13.1. Методологические аспекты моделирования

С развитием вычислительной техники наиболее эффективным методом исследования больших систем стало машинное моделирование, без которого невозможно решение многих крупных народнохозяйственных проблем. Поэтому одной из актуальных задач подготовки специалистов является освоение теории и методов математического моделирования с учетом требований системности, позволяющих не только строить модели изучаемых объектов, анализировать их динамику и возможность управления машинным экспериментом с моделью, но и судить в известной мере об адекватности создаваемых моделей исследуемым системам, о границах применимости и правильно организовать моделирование систем на современных средствах вычислительной техники.

Методологические аспекты моделирования. Прежде чем рассматривать математические, алгоритмические, программные и прикладные аспекты машинного моделирования, необходимо изучить общие методологические аспекты для широкого класса математических моделей объектов, реализуемых на средствах вычислительной техники. Моделирование с использованием средств вычислительной техники (ЭВМ, АВМ, ГВК) позволяет исследовать механизм явлений, протекающих в реальном объекте с большими или малыми скоростями, когда в натурных

экспериментах с объектом трудно (или невозможно) проследить за изменениями, происходящими в течение короткого времени, или когда получение достоверных результатов сопряжено с длительным экспериментом. При необходимости машинная модель дает возможность как бы «растягивать» или «сжимать» реальное время, так как машинное моделирование связано с понятием системного времени, отличного от реального. Кроме того, с помощью машинного моделирования в диалоговой системе можно обучать персонал принятию решений в управлении объектом, например при организации деловой игры, что позволяет выработать необходимые практические навыки реализации процесса управления.

Сущность машинного моделирования системы состоит в проведении на вычислительной машине эксперимента с моделью, которая представляет собой некоторый программный комплекс, описывающий формально и (или) алгоритмически поведение элементов системы S в процессе ее функционирования, т. е. в их взаимодействии друг с другом и внешней средой Е. Машинное моделирование с успехом применяют в тех случаях, когда трудно четко сформулировать критерий оценки качества функционирования системы и цель ее не поддается полной формализации, поскольку позволяет сочетать программно-технические возможности ЭВМ со способностями человека мыслить неформальными категориями.

Требования пользователя к модели. Сформулируем основные требования, предъявляемые к модели М процесса функционирования системы S.

1. Полнота модели должна предоставлять пользователю возможность получения необходимого набора оценок характеристик системы с требуемой точностью и достоверностью.

2. Гибкость модели должна давать возможность воспроизведения различных ситуаций при варьировании структуры, алгоритмов и параметров системы.

3. Длительность разработки и реализации модели большой системы должна быть по возможности минимальной при учете ограничений на имеющиеся ресурсы.

4. Структура модели должна быть блочной, т. е. допускать возможность замены, добавления и исключения некоторых частей без переделки всей модели.

5. Информационное обеспечение должно предоставлять возможность эффективной работы модели с базой данных систем определенного класса.

6. Программные и технические средства должны обеспечивать эффективную (по быстродействию и памяти) машинную реализацию модели и удобное общение с ней пользователя.

7. Должно быть реализовано проведение целенаправленных (планируемых) машинных экспериментов с моделью системы с

использованием аналитико-имитационного подхода при наличии ограниченных вычислительных ресурсов.

С учетом этих требований рассмотрим основные положения, которые справедливы при моделировании на ЭВМ систем S, а также их подсистем и элементов. При машинном моделировании системы S характеристики процесса ее функционирования определяются на основе модели M, построенной исходя из имеющейся исходной информации об объекте моделирования. При получении новой информации об объекте его модель пересматривается и уточняется с учетом новой информации, т. е. процесс моделирования, включая разработку и машинную реализацию модели, является итерационным. Этот итерационный процесс продолжается до тех пор, пока не будет получена модель M, которую можно считать адекватной в рамках решения поставленной задачи исследования и проектирования системы S.

Моделирование систем с помощью ЭВМ можно использовать в следующих случаях:

1. для исследования системы S до того, как она спроектирована, с целью определения чувствительности характеристики к изменениям структуры, алгоритмов и параметров объекта моделирования и внешней среды;

2. на этапе проектирования системы S для анализа и синтеза различных вариантов системы и выбора среди конкурирующих такого варианта, который удовлетворял бы заданному критерию оценки эффективности системы при принятых ограничениях;

3. после завершения проектирования и внедрения системы, т. е. при ее эксплуатации, для получения информации, дополняющей результаты натурных испытаний (эксплуатации) реальной системы, и для получения прогнозов эволюции (развития) системы во времени.

Существуют общие положения, применяемые ко всем перечисленным случаям машинного моделирования. Даже в тех случаях, когда конкретные способы моделирования отличаются друг от друга и имеются различные модификации моделей, например в области машинной реализации моделирующих алгоритмов с использованием конкретных программно-технических средств, в практике моделирования систем можно сформулировать общие принципы, которые могут быть положены в основу методологии машинного моделирования.

13.2. Этапы моделирования систем

Рассмотрим основные этапы моделирования системы S, к числу которых относятся:

- построение концептуальной модели системы и ее формализация;
- алгоритмизация модели системы и ее машинная реализация;
- получение и интерпретация результатов моделирования системы.

Взаимосвязь перечисленных этапов моделирования систем и их составляющих (подэтапов) может быть представлена в виде сетевого графика, показанного на рисунке 13.1.

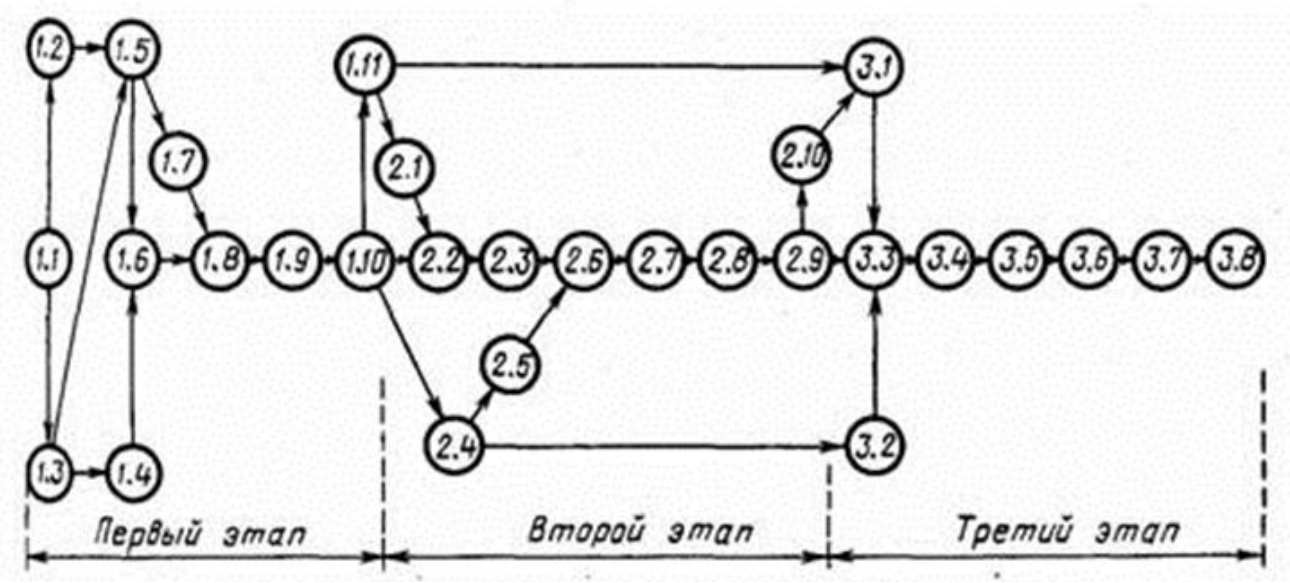


Рисунок 13.1 – Взаимосвязь этапов моделирования систем

Перечислим эти подэтапы:

1.1 — постановка задачи машинного моделирования системы;

1.2 — анализ задачи моделирования системы;

1.3 —определение требований к исходной информации об объекте моделирования и организация ее сбора;

1.4 — выдвижение гипотез и принятие предположений;

1.5 — определение параметров и переменных модели;

1.6 — установление основного содержания модели;

1.7 — обоснование критериев оценки эффективности системы;

1.8 — определение процедур аппроксимации;

1.9 — описание концептуальной модели системы;

1.10 — проверка достоверности концептуальной модели;

1.11 — составление технической документации по первому этапу;

2.1 — построение логической схемы модели;

2.2 — получение математических соотношений;

2.3 — проверка достоверности модели системы;

2.4 — выбор инструментальных средств для моделирования;

2.5 — составление плана выполнения работ по программированию;

2.6 —спецификация и построение схемы программы;

2.7 — верификация и проверка достоверности схемы программы;

2.8 — проведение программирования модели;

2.9 — проверка достоверности программы;

2.10 — составление технической документации по второму этапу;

3.1 — планирование машинного эксперимента с моделью системы;

3.2 — определение требований к вычислительным средствам;
3.3 — проведение рабочих расчетов;
3.4 — анализ результатов моделирования системы;
3.5 — представление результатов моделирования;
3.6 — интерпретация результатов моделирования;
3.7 — подведение итогов моделирования и выдача рекомендаций;
3.8 — составление технической документации по третьему этапу.

Таким образом, процесс моделирования системы S сводится к выполнению перечисленных подэтапов, сгруппированных в виде трех этапов. На этапе построения концептуальной модели Мм и ее формализации проводится исследование моделируемого объекта с точки зрения выделения основных составляющих процесса его-функционирования, определяются необходимые аппроксимации и получается обобщенная схема модели системы S, которая преобразуется в машинную модель Мм на втором этапе моделирования путем последовательной алгоритмизации и программирования модели. Последний третий этап моделирования системы сводится к проведению согласно полученному плану рабочих расчетов на ЭВМ с использованием выбранных программно-технических средств, получению и интерпретации результатов моделирования системы S с учетом воздействия внешней среды E. Очевидно, что при построении модели и ее машинной реализации при получении новой информации возможен пересмотр ранее принятых решений, т. е. процесс моделирования является итерационным. Рассмотрим содержание каждого из этапов более подробно.

13.3. Построение концептуальных моделей систем и их формализация

На первом этапе машинного моделирования — построения концептуальной модели Мм системы S и ее формализации — формулируется модель и строится ее формальная схема, т. е. основным назначением этого этапа является переход от содержательного описания объекта к его математической модели, другими словами, процесс формализации. Моделирование систем на ЭВМ в настоящее время — наиболее универсальный и эффективный метод оценки характеристик больших систем. Наиболее ответственными и наименее формализованными моментами в этой работе являются проведение границы между системой S и внешней средой E, упрощение описания системы и построение сначала концептуальной, а затем формальной модели системы. Модель должна быть адекватной, иначе невозможно получить положительные результаты моделирования, т. е. исследование процесса функционирования системы на неадекватной модели вообще теряет смысл. Под **адекватной моделью** будем понимать модель, которая с определенной степенью приближения на уровне понимания моделируемой системы S разработчиком модели отражает процесс ее функционирования во внешней среде E.

Переход от описания к блочной модели. Наиболее рационально строить модель функционирования системы по блочному принципу. При этом могут быть выделены три автономные группы блоков такой модели:

- блоки первой группы представляют собой имитатор воздействий внешней среды E на систему S;
- блоки второй группы являются собственно моделью процесса функционирования исследуемой системы S;
- блоки третьей группы — вспомогательными и служат для машинной реализации блоков двух первых групп, а также для фиксации и обработки результатов моделирования.

Рассмотрим механизм перехода от описания процесса функционирования некоторой гипотетической системы к модели этого процесса. Для наглядности введем представление об описании свойств процесса функционирования системы S, т. е. об ее концептуальной модели Мм как совокупности некоторых элементов, условно изображенных квадратами (рисунок 13.2, а). Эти квадраты представляют собой описание некоторых подпроцессов исследуемого процесса функционирования системы S, воздействия внешней среды E и т. д. Переход от описания системы к ее модели в этой интерпретации сводится к исключению из рассмотрения некоторых второстепенных элементов описания (элементы 5 — 8,39 — 41,43 — 47).

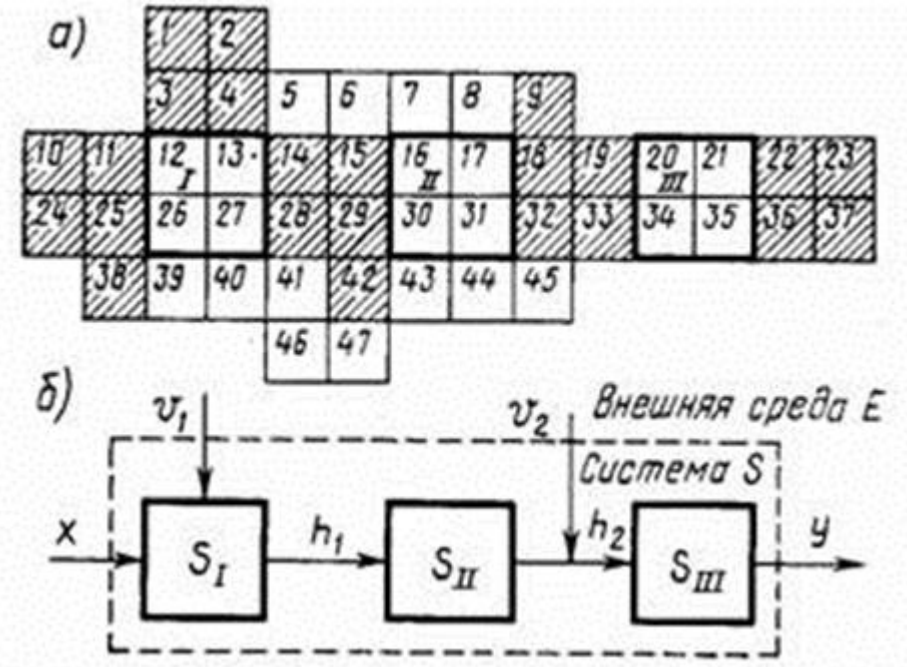


Рисунок 13.2 – Модель системы: а – концептуальная; б - блочная

Предполагается, что они не оказывают существенного влияния на ход процессов, исследуемых с помощью модели. Часть элементов (14, 15, 28, 29, 42) заменяется пассивными связями h_l отражающими внутренние свойства системы (рисунок 3.2, б). Некоторая часть элементов (1 — 4, 10, 11, 24, 25)

заменяется входными факторами x и воздействиями внешней среды v1. Возможны и комбинированные замены: элементы 9, 18, 19, 32, 33 заменены пассивной связью h_2 и воздействием внешней среды E. Элементы 22, 23,36,37 отражают воздействие системы на внешнюю среду y.

Оставшиеся элементы системы S группируются в блоки S_I, S_{II}, S_{III}, отражающие процесс функционирования исследуемой системы. Каждый из этих блоков достаточно автономен, что выражается в минимальном количестве связей между ними. Поведение этих блоков должно быть хорошо изучено и для каждого из них построена математическая модель, которая в свою очередь может содержать ряд подблоков. Построенная блочная модель процесса функционирования исследуемой системы S предназначена для анализа характеристик этого процесса, который может быть проведен при машинной реализации полученной модели.

Математические модели процессов. После перехода от описания моделируемой системы S к ее модели M, построенной по блочному принципу, необходимо построить математические модели процессов, происходящих в различных блоках. Математическая модель представляет собой совокупность соотношений (например, уравнений, логических условий, операторов), определяющих характеристики процесса функционирования системы S в зависимости от структуры системы, алгоритмов поведения, параметров системы, воздействий внешней среды E, начальных условий и времени. Математическая модель является результатом формализации процесса функционирования исследуемой системы, т. е. построения формального (математического) описания процесса с необходимой в рамках проводимого исследования степенью приближения к действительности.

Для иллюстрации возможностей формализации рассмотрим процесс функционирования некоторой гипотетической системы S, которую можно разбить на n подсистем с характеристиками $y_1(t)$, $y_2(t)$… $y_{nY}(t)$ с параметрами h_1, h_2, ...,h_{nH} при наличии входных воздействии x_1, x_2,..., x_{nX} и воздействий внешней среды v_1, v_2,..., v_{nV}. Тогда математической моделью процесса может служить система соотношений вида

$$\left.\begin{aligned}
y_1(t) &= f_1(x_1, x_2, \ldots, x_{nX}; v_1, v_2, \ldots, v_{nV}; h_1, h_2, \ldots, h_{nH}; t);\\
y_2(t) &= f_2(x_1, x_2, \ldots, x_{nX}; v_1, v_2, \ldots, v_{nV}; h_1, h_2, \ldots, h_{nH}; t);\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
y_{nY}(t) &= f_m(x_1, x_2, \ldots, x_{nX}; v_1, v_2, \ldots, v_{nV}; h_1, h_2, \ldots, h_{nH}; t).
\end{aligned}\right\} \quad (13.1)$$

Если бы функции f_1, f_2, …, f_m были известны, то соотношения (13.1) оказались бы идеальной математической моделью процесса функционирования системы S. Однако на практике получение модели достаточно простого вида для больших систем чаще всего невозможно, поэтому обычно процесс функционирования системы S разбивают на ряд

элементарных подпроцессов. При этом необходимо так проводить разбиение на подпроцессы, чтобы построение моделей отдельных подпроцессов было элементарно и не вызывало трудностей при формализации. Таким образом, на этой стадии сущность формализации подпроцессов будет состоять в подборе типовых математических схем. **Например**, для стохастических процессов это могут быть схемы вероятностных автоматов (P-схемы), схемы массового обслуживания (Q-схемы) и т. д., которые достаточно точно описывают основные особенности реальных явлений, составляющих подпроцессы, с точки зрения решаемых прикладных задач.

Таким образом, формализации процесса функционирования любой системы S должно предшествовать изучение составляющих его явлений. В результате появляется содержательное описание процесса, которое представляет собой первую попытку четко изложить закономерности, характерные для исследуемого процесса, и постановку прикладной задачи. Содержательное описание является исходным материалом для последующих этапов формализации: построения формализованной схемы процесса функционирования системы и математической модели этого процесса. Для моделирования процесса функционирования системы на ЭВМ необходимо преобразовать математическую модель процесса в соответствующий моделирующий алгоритм и машинную программу.

Подэтапы первого этапа моделирования. Рассмотрим более подробно основные подэтапы построения концептуальной модели M, системы и ее формализации (рисунок 13.1).

1.1. Постановка задачи машинного моделирования системы. Дается четкая формулировка задачи исследования конкретной системы S и основное внимание уделяется таким вопросам, как:

а) признание существования задачи и необходимости машинного моделирования;

б) выбор методики решения задачи с учетом имеющихся ресурсов;

в) определение масштаба задачи и возможности разбиения ее на подзадачи.

Необходимо также ответить на вопрос о приоритетности решения различных подзадач, оценить эффективность возможных математических методов и программно-технических средств их решения. Тщательная проработка этих вопросов позволяет сформулировать задачу исследования и приступить к ее реализации. При этом возможен пересмотр начальной постановки задачи в процессе моделирования.

1.2. Анализ задачи моделирования системы. Проведение анализа задачи способствует преодолению возникающих в дальнейшем трудностей при ее решении методом моделирования. На рассматриваемом втором этапе основная работа сводится именно к проведению анализа, включая:

а) выбор критериев оценки эффективности процесса функционирования системы S;

б) определение эндогенных и экзогенных переменных модели М;

в) выбор возможных методов идентификации;

г) выполнение предварительного анализа содержания второго этапа алгоритмизации модели системы и ее машинной реализации;

д) выполнение предварительного анализа содержания третьего этапа получения и интерпретации результатов моделирования системы.

1.3. Определение требований к исходной информации об объекте моделирования и организация ее сбора. После постановки задачи моделирования системы S определяются требования к информации, из которой получают качественные и количественные исходные данные, необходимые для решения этой задачи. Эти данные помогают глубоко разобраться в сущности задачи, методах ее решения. Таким образом, на этом подэтапе проводится:

а) выбор необходимой информации о системе S и внешней среде E;

б) подготовка априорных данных;

в) анализ имеющихся экспериментальных данных;

г) выбор методов и средств предварительной обработки информации о системе.

При этом необходимо помнить, что именно от качества исходной информации об объекте моделирования существенно зависят как адекватность модели, так и достоверность результатов моделирования.

1.4. Выдвижение гипотез и принятие предположений. Гипотезы при построении модели системы S служат для заполнения «пробелов» в понимании задачи исследователем. Выдвигаются также гипотезы относительно возможных результатов моделирования системы S справедливость, которых проверяется при проведении машинного эксперимента. Предположения предусматривают, что некоторые данные неизвестны или их нельзя получить. Предположения могут выдвигаться относительно известных данных, которые не отвечают требованиям решения поставленной задачи. Предположения дают возможность провести упрощения модели в соответствии с выбранным уровнем моделирования. При выдвижении гипотез и принятия предположений учитываются следующие факторы:

а) объем имеющейся информации для решения задач;

б) подзадачи, для которых информация недостаточна;

в) ограничения на ресурсы времени для решения задачи;

г) ожидаемые результаты моделирования.

Таким образом, в процессе работы с моделью системы S возможно многократное возвращение к этому подэтапу в зависимости от полученных результатов моделирования и новой информации об объекте.

1.5. Определение параметров и переменных модели. Прежде чем перейти к описанию математической модели, необходимо определить параметры системы h_k, $k=\overline{1,n_H}$, входные и выходные переменные

x_i, $i=\overline{1,n_X}$, y_j, $j=\overline{1,n_Y}$, воздействия внешней среды v_l, $l=\overline{1,n_V}$. Конечной целью этого подэтапа является подготовка к построению математической модели системы S, функционирующей во внешней среде E, для чего необходимо рассмотрение всех параметров и переменных модели и оценка степени их влияния на процесс функционирования системы в целом. Описание каждого параметра и переменной должно даваться в следующей форме:

а) определение и краткая характеристика;

б) символ обозначения и единица измерения;

в) диапазон изменения;

г) место применения в модели.

1.6. Установление основного содержания модели. На этом подэтапе определяется основное содержание модели и выбирается метод построения модели системы, которые разрабатываются на основе принятых гипотез и предположений. При этом учитываются следующие особенности:

а) формулировка задачи моделирования системы;

б) структура системы S и алгоритмы ее поведения, воздействия внешней среды E;

в) возможные методы и средства решения задачи моделирования.

1.7. Обоснование критериев оценки эффективности системы. Для оценки качества процесса функционирования моделируемой системы S необходимо выбрать некоторую совокупность критериев оценки эффективности, т. е. в математической постановке задача сводится к получению соотношения для оценки эффективности как функции параметров и переменных системы. Эта функция представляет собой поверхность отклика в исследуемой области изменения параметров и переменных и позволяет определить реакцию системы. Эффективность системы S можно оценить с помощью интегральных или частных критериев, выбор которых зависит от рассматриваемой задачи.

1.8. Определение процедур аппроксимации. Для аппроксимации реальных процессов, протекающих в системе S, обычно используются три вида процедур:

а) детерминированную. При детерминированной процедуре результаты моделирования однозначно определяются по данной совокупности входных воздействий, параметров и переменных системы S. В этом случае отсутствуют случайные элементы, влияющие на результаты моделирования.

б) вероятностную. Вероятностная (рандомизированная) процедура применяется в том случае, когда случайные элементы, включая воздействия внешней среды E, влияют на характеристики процесса функционирования системы S и когда необходимо получить информацию о законах распределения выходных переменных.

в) определения средних значений. Процедура определения средних значений используется тогда, когда при моделировании системы интерес представляют средние значения выходных переменных при наличии случайных элементов.

1.9. Описание концептуальной модели системы. На этом подэтапе построения модели системы:

а) описывается концептуальная модель Мх в абстрактных терминах и понятиях;

б) дается описание модели с использованием типовых математических схем;

в) принимаются окончательно гипотезы и предположения;

г) обосновывается выбор процедуры аппроксимации реальных процессов при построении модели.

Таким образом, на этом подэтапе проводится подробный анализ задачи, рассматриваются возможные методы ее решения и дается детальное описание концептуальной модели Мk, которая затем используется на втором этапе моделирования.

1.10. Проверка достоверности концептуальной модели. После того как концептуальная модель Мх описана, необходимо проверить достоверность некоторых концепций модели перед тем, как перейти к следующему этапу моделирования системы S. Проверять достоверность концептуальной модели достаточно сложно, так как процесс ее построения является эвристическим и такая модель описывается в абстрактных терминах и понятиях. Один из методов проверки модели Мх — применение операций обратного перехода, позволяющий проанализировать модель, вернуться к принятым аппроксимациям и, наконец, рассмотреть снова реальные процессы, протекающие в моделируемой системе S. Проверка достоверности концептуальной модели Мт должна включать:

а) проверку замысла модели;

б) оценку достоверности исходной информации;

в) рассмотрение постановки задачи моделирования;

г) анализ принятых аппроксимаций;

д) исследование гипотез и предположений.

Только после тщательной проверки концептуальной модели М, следует переходить к этапу машинной реализации модели, так как ошибки в модели Мk не позволяют получить достоверные результаты моделирования реальных или частных критериев, выбор которых зависит от рассматриваемой задачи.

1.11. Составление технической документации по первому этапу. В конце этапа построения концептуальной модели Мм и ее формализации составляется технический отчет по этапу, который включает в себя:

а) подробную постановку задачи моделирования системы S;

б) анализ задачи моделирования системы;

в) критерии оценки эффективности системы;

г) параметры и переменные модели системы;

д) гипотезы и предположения, принятые при построении модели;

е) описание модели в абстрактных терминах и понятиях;

ж) описание ожидаемых результатов моделирования системы S.

Составление технической документации — обязательное условие успешного проведения моделирования системы S, так как в процессе разработки модели большой системы и ее машинной реализации принимают участие на различных этапах коллективы специалистов разных профилей (начиная от постановщиков задач и кончая программистами) и документация является средством обеспечения их эффективного взаимодействия при решении поставленной задачи методом моделирования.

Вопросы для повторения и закрепления материала

1. Что такое ГВК?

2. Перечислите требования пользователя к модели.

3. В каких случаях используется моделирование систем при помощи ЭВМ?

4. Перечислите основные этапы моделирования систем?

5. Каким образом осуществляется переход от описания к блочной модели?

6. Что такое адекватная модель?

7. Как выглядит концептуальная модель?

8. Обоснуйте важность постановки задачи моделирования системы.

9. Почему необходимо выдвижение гипотезы моделирования?

10. Нужно ли обосновывать критерии оценки эффективности моделирования.

Задания для самостоятельной работы

1. Найдите аналогичные методики разработки и машинной реализации моделей систем.

Тема 14. Алгоритмизация моделей систем, получение и интерпретация результатов моделирования

Цель изучения темы: изучить процесс алгоритмизации моделей систем.

Задачи изучения темы:

- рассмотреть алгоритмизацию моделей систем и их машинную реализацию;

- рассмотреть процесс получения и интерпретации результатов моделирования систем.

14.1. Алгоритмизация моделей систем и их машинная реализация

На втором этапе моделирования — этапе алгоритмизации модели и ее машинной реализации — математическая модель, сформированная на первом этапе, воплощается в конкретную машинную модель. Этот этап представляет собой этап практической деятельности, направленной на реализацию идей и математических схем в виде машинной модели Мм процесса функционирования системы S. Прежде чем рассматривать подэтапы алгоритмизации и машинной реализации модели, остановимся на основных принципах построения моделирующих алгоритмов и формах их представления.

Принципы построения моделирующих алгоритмов. Процесс функционирования системы S можно рассматривать как последовательную смену ее состояний $\vec{z}$ =z(z_1(t), z_2(t), ..., z_k(t)) в k-мерном пространстве. Очевидно, что задачей моделирования процесса функционирования исследуемой системы S является построение функций z, на основе которых можно провести вычисление интересующих характеристик процесса функционирования системы. Для этого должны иметься соотношения, связывающие функции z с переменными, параметрами и временем, а также начальные условия $\vec{z}^0$ =z(z_1(t_o), z_2(t_0), ..., z_k(t_0)) в момент времени t = t_0.

Рассмотрим процесс функционирования некоторой детерминированной системы S_D, в которой отсутствуют случайные факторы, т. е. вектор состояний такой системы можно определить из как $\vec{z} = \Phi(\vec{z^0}, \vec{x}, t$. Тогда состояние процесса в момент времени t_0+jΔt может быть однозначно определено из соотношений математической модели по известным начальным условиям. Это позволяет строить моделирующий алгоритм процесса функционирования системы. Для этого преобразуем соотношения модели Z к такому виду, чтобы сделать удобным вычисление z_1(t+Δt), z_2(t+Δt), ..., z_k(t+Δt) по значениям $z_i(\tau)$, $i = \overline{1,k}$, где $\tau \le t$. Организуем, счетчик системного времени, который в начальный момент показывает время t_0. Для этого момента $z_i(t_0)$=z°. Прибавим интервал времени Δt, тогда счетчик будет показывать $t_1 = t_0$+Δt. Вычислим значения $z_1 = t_0$+Δt. Затем перейдем к моменту времени $t_2 = t_1$+Δt и т. д. Если шаг Δt достаточно мал, то таким путем можно получить приближенные значения z.

Рассмотренный принцип построения моделирующих алгоритмов называется **принципом Δt**. Это наиболее универсальный принцип, позволяющий определить последовательные состояния процесса функционирования системы S через заданные интервалы времени Δt. Но с точки зрения затрат машинного времени он иногда оказывается неэкономичным.

При рассмотрении процессов функционирования некоторых систем можно обнаружить, что для них характерны два типа состояний:

1) особые, присущие процессу функционирования системы только в некоторые моменты времени (моменты поступления входных или управляющих воздействий, возмущений внешней среды и т. п.);

2) неособые, в которых процесс находится все остальное время.

Особые состояния характерны еще и тем обстоятельством, что функции состояний $z_i(t)$ в эти моменты времени изменяются скачком, а между особыми состояниями изменение координат z(i) происходит плавно и непрерывно или не происходит совсем. Таким образом, следя при моделировании системы S только за ее особыми состояниями в те моменты времени, когда эти состояния имеют место, можно получить информацию, необходимую для построения функций zi(t). Очевидно, для описанного типа систем могут быть построены моделирующие алгоритмы по «**принципу особых состояний**». Обозначим скачкообразное (релейное) изменение состояния z как δz, а «принцип особых состояний» — как принцип δz.

Например, для системы массового обслуживания (Q-схемы) в качестве особых состояний могут быть выбраны состояния в моменты поступления заявок на обслуживание в прибор П и в моменты окончания обслуживания заявок каналами К, когда состояние системы, оцениваемое числом находящихся в ней заявок, меняется скачком.

Отметим, что характеристики процесса функционирования таких систем с особыми состояниями оцениваются по информации об особых состояниях, а неособые состояния при моделировании не рассматриваются. «Принцип δz» дает возможность для ряда систем существенно уменьшить затраты машинного времени на реализацию моделирующих алгоритмов по сравнению с «принципом Δt». Логика построения моделирующего алгоритма, реализующего «принцип δz», отличается от рассмотренной для «принципа Δt» только тем, что включает в себя процедуру определения момента времени ts, соответствующего следующему особому состоянию системы S. Для исследования процесса функционирования больших систем рационально использование комбинированного принципа построения моделирующих алгоритмов, сочетающего в себе преимущества каждого из рассмотренных принципов.

Формы представления моделирующих алгоритмов. Удобной формой представления логической структуры моделей процессов функционирования систем и машинных программ является схема. На различных этапах моделирования составляются обобщенные и детальные логические схемы моделирующих алгоритмов, а также схемы программ.

Обобщенная (укрупненная) схема моделирующего алго**ритма** задает общий порядок действий при моделировании системы без каких-либо уточняющих деталей. Обобщенная схема показывает, что необходимо

выполнить на очередном шаге моделирования, например обратиться к датчику случайных чисел.

Детальная схема моделирующего алгоритм**а** содержит уточнения, отсутствующие в обобщенной схеме. Детальная схема показывает не только, что следует выполнить на очередном шаге моделирования системы, но и как это выполнить.

Логическая схема моделирующего алгоритма представляет собой логическую структуру модели процесса функционирования системы S. Логическая схема указывает упорядоченную во времени последовательность логических операций, связанных с решением задачи моделирования.

Схема программы отображает порядок программной реализации моделирующего алгоритма с использованием конкретного математического обеспечения. Схема программы представляет собой интерпретацию логической схемы моделирующего алгоритма разработчиком программы на базе конкретного алгоритмического языка. Различие между этими схемами заключается в том, что логическая схема отражает логическую структуру модели процесса функционирования системы, а схема программы — логику машинной реализации модели с использованием конкретных программно-технических средств моделирования.

Логическая схема алгоритма и схема программы могут быть выполнены как в укрупненной, так и в детальной форме.

Подэтапы второго этапа моделирования. Рассмотрим подэтапы, выполненные при алгоритмизации модели системы и ее машинной реализации, обращая основное внимание на задачи каждого подэтапе и методы их решения.

2.1. Построение логической схемы модели. Рекомендуется строить модель по блочному принципу, т. е. в виде некоторой совокупности стандартных блоков. Построение модели систем S из таких блоков обеспечивает необходимую гибкость в процессе ее эксплуатации, особенно на стадии машинной отладки. При построении блочной модели проводится разбиение процесса функционирования системы на отдельные достаточно автономные подпроцессы. Таким образом, модель функционально подразделяется на подмодели, каждая из которых в свою очередь может быть разбита на еще более мелкие элементы. Блоки такой модели бывают двух типов: основные и вспомогательные. Каждый основной блок соответствует некоторому реальному подпроцессу, имеющему место в моделируемой системе S, а вспомогательные блоки представляют собой лишь составную часть машинной модели, они не отражают функции моделируемой системы и необходимы лишь для машинной реализации, фиксации и обработки результатов моделирования.

2.2. Получение математических соотношений. Одновременно с выполнением подэтапа построения логической схемы модели необходимо получить, если это возможно, математические соотношения в виде явных

функций, т. е. построить аналитические модели. Этот подэтап соответствует неявному заданию возможных математических соотношений на этапе построения концептуальной модели. При выполнении первого этапа еще не может иметься информации о конкретном виде таких математических соотношений, а на втором этапе уже необходимо получить эти соотношения. Схема машинной модели Мм должна представлять собой полное отражение заложенной в модели концепции и иметь:

а) описание всех блоков модели с их наименованиями;

б) единую систему обозначений и нумерацию блоков;

в) отражение логики модели процесса функционирования системы;

г) задание математических соотношений в явном виде.

Таким образом, в общем случае построенная машинная модель Мм системы будет иметь комбинированный характер, т. е. отражать аналитико-имитационный подход, когда часть процесса в системе описана аналитически, а другая часть имитируется соответствующими алгоритмами.

2.3. Проверка достоверности модели системы. Эта проверка является первой из проверок, выполняемых на этапе реализации модели. Так как модель представляет собой приближенное описание процесса функционирования реальной системы S, то до тех пор, пока не доказана достоверность модели Мм, нельзя утверждать, что с ее помощью будут получены результаты, совпадающие с теми, которые могли бы быть получены при проведении натурного эксперимента с реальной системой S. Поэтому определение достоверности модели можно считать наиболее важной проблемой при моделировании систем. От решения этой проблемы зависит степень доверия к результатам, полученным методом моделирования. Проверка модели на рассматриваемом подэтапе должна дать ответ на вопрос, насколько логическая схема модели системы и используемые математические соотношения отражают замысел модели, сформированный на первом этапе. При этом проверяются:

а) возможность т решения поставленной задачи;

б) точность отражения замысла в логической схеме;

в) полнота логической схемы модели;

г) правильность используемых математических соотношений.

Только после того, как разработчик убеждается путем соответствующей проверки в правильности всех этих положений, можно считать, что имеется логическая схема модели системы S, пригодная для дальнейшей работы по реализации модели на ЭВМ.

2.4. Выбор инструментальных средств для моделирования. На этом подэтапе необходимо окончательно решить вопрос о том, какую вычислительную машину (ЭВМ, АВМ, ГВК) и какое программное обеспечение целесообразно использовать для реализации модели системы S. Вообще, выбор вычислительных средств может быть проведен и на предыдущих подэтапах, но рассматриваемый подэтап является последним,

когда этот выбор должен быть сделан окончательно, так как в противном случае возникнут трудности в проведении дальнейших работ по реализации модели. Вопрос о выборе ЭВМ сводится к обеспечению следующих требований:

а) наличие необходимых программных и технических средств;

б) доступность выбранной ЭВМ для разработчика модели;

в) обеспечение всех этапов реализации модели;

г) возможность своевременного получения результатов.

2.5. Составление плана выполнения работ по программированию. Такой план должен помочь при программировании модели, учитывая оценки объема программы и трудозатрат на ее составление, План при использовании универсальной ЭВМ должен включать в себя:

а) выбор языка (системы) программирования модели;

б) указание типа ЭВМ и необходимых для моделирования устройств;

в) оценку примерного объема необходимой оперативной и внешней памяти;

г) ориентировочные затраты машинного времени на моделирование;

д) предполагаемые затраты времени на программирование и отладку программы на ЭВМ.

2.6. Спецификация и построение схемы программы. Спецификация программы — формализованное представление требований, предъявляемых к программе, которые должны быть удовлетворены при ее разработке, а также описание задачи, условия и эффекта действия без указания способа его достижения. Наличие логической блок-схемы модели позволяет построить схему программы, которая должна отражать:

а) разбиение модели на блоки, подблоки и т. д.;

б) особенности программирования модели;

в) проведение необходимых изменений;

г) возможности тестирования программы;

д) оценку затрат машинного времени;

е) форму представления входных и выходных данных.

Построение схемы программы представляет собой одну из основных задач на этапе машинной реализации модели. При этом особое внимание должно быть уделено особенностям выбранного для реализации модели языка: алгоритмического языка общего назначения или языка моделирования (например, MatLab).

2.7. Верификация и проверка достоверности схемы программы.

Верификация программы — доказательство того, что поведение программы соответствует спецификации на программу. Эта проверка является второй на этапе машинной реализации модели системы. Очевидно, что нет смысла продолжать работу по реализации модели, если нет уверенности в том, что в схеме программы, по которой будет вестись дальнейшее программирование, допущены ошибки, которые делают ее

неадекватной логической схеме модели, а следовательно, и неадекватной самому объекту моделирования. При этом проводится проверка соответствия каждой операции, представленной в схеме программы, аналогичной ей операции в логической схеме модели.

2.8. Проведение программирования модели. При достаточно подробной схеме программы, которая отражает все операции логической схемы модели, можно приступить к программированию модели. Если имеется адекватная схема программы, то программирование представляет собой работу только для программиста без участия и помощи со стороны разработчика модели. При использовании пакетов прикладных программ моделирования проводится непосредственная генерация рабочих программ для моделирования конкретного объекта, т. е. программирование модели реализуется в автоматизированном режиме.

2.9. Проверка достоверности программы. Эта последняя проверка на этапе машинной реализации модели, которую необходимо проводить:

а) обратным переводом программы в исходную схему;

б) проверкой отдельных частей программы при решении различных тестовых задач;

в) объединением всех частей программы и проверкой ее в целом на контрольном примере моделирования варианта системы S.

На этом подэтапе необходимо также проверить оценки затрат машинного времени на моделирование. Полезно также получить достаточно простую аналитическую аппроксимацию зависимости затрат машинного времени от количества реализаций, что позволит разработчику модели (заказчику) правильно сформулировать требования к точности и достоверности результатов моделирования.

2.10. Составление технической документации по второму этапу. Для завершения этапа машинной реализации модели Мм необходимо составить техническую документацию, содержащую:

а) логическую схему модели и ее описание;

б) адекватную схему программы и принятые обозначения;

в) полный текст программы;

г) перечень входных и выходных величин с пояснениями;

д) инструкцию по работе с программой;

е) оценку затрат машинного времени на моделирование с указанием требуемых ресурсов ЭВМ.

Таким образом, на этом этапе разрабатывается схема модели системы S, проводится ее алгоритмизация и программирование с использованием конкретных программно-технических средств, т. е. строится машинная модель Мм, с которой предстоит работать для получения необходимых результатов моделирования по оценке характеристик процесса функционирования системы S (задача анализа) или для поиска оптимальных структур, алгоритмов и параметров системы S (задача синтеза).

14.2. Получение и интерпретация результатов моделирования систем

На третьем этапе моделирования — этапе получения и интерпретации результатов моделирования — ЭВМ используется для проведения рабочих расчетов по составленной и отлаженной программе. Результаты этих расчетов позволяют проанализировать и сформулировать выводы о характеристиках процесса функционирования моделируемой системы S.

Особенности получения результатов моделирования. При реализации моделирующих алгоритмов на ЭВМ вырабатывается информация о состояниях процесса функционирования исследуемых систем $z(t) \in Z$. Эта информация является исходным материалом для определения приближенных оценок искомых характеристик, получаемых в результате машинного эксперимента, т. е. критериев оценки. Критерием оценки будем называть любой количественный показатель, по которому можно судить о результатах моделирования системы. Критериями оценки могут служить показатели, получаемые на основе процессов, действительно протекающих в системе, или получаемых на основе специально сформированных функций этих процессов.

В ходе машинного эксперимента изучается поведение исследуемой модели М процесса функционирования системы S на заданном интервале времени [0, Т]. Поэтому критерий оценки является в общем случае векторной случайной функцией, заданной на этом же интервале:

$$\vec{q}(t) = (q_1(t),\ q_2(t),\ \ldots,\ q_n(t)).$$

Часто используют более простые критерии оценки, например вероятность определенного состояния системы в заданный момент времени $t \in [0, T]$, отсутствие отказов и сбоев в системе на интервале [0, Т] и т. д. При интерпретации результатов моделирования вычисляются различные статистические характеристики закона распределения критерия оценки.

Рассмотрим общую схему фиксации и обработки результатов моделирования системы, которая приведена на рисунке 14.1.

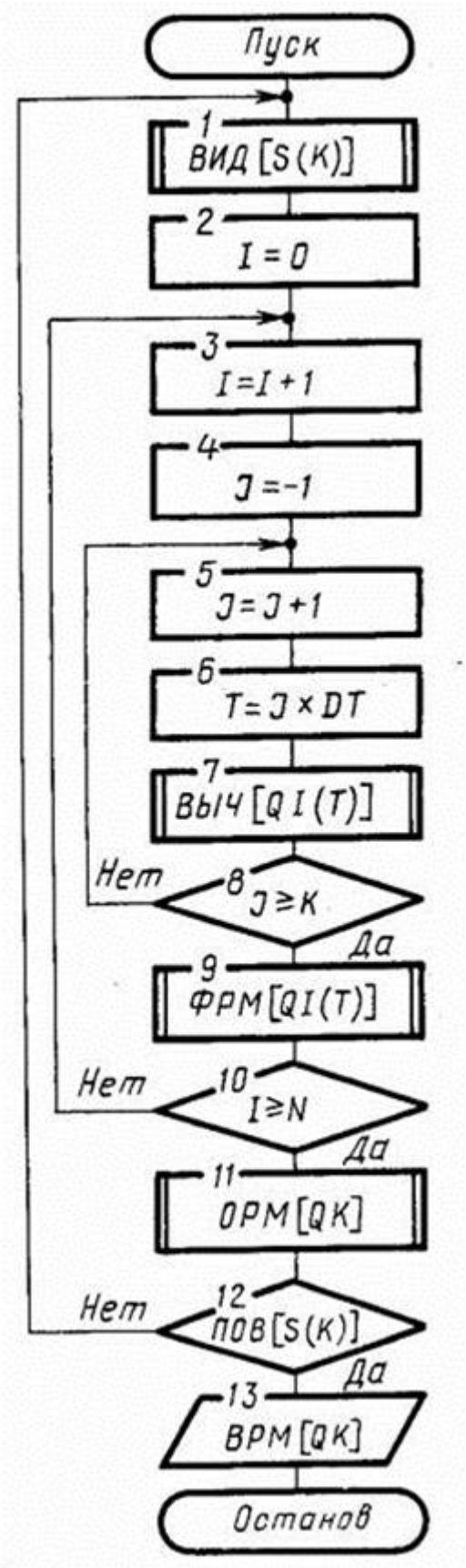


Рисунок 14.1 – Алгоритм фиксации и обработки результатов моделирования системы

Будем рассматривать гипотетическую модель М, предназначенную для исследования поведения системы S на интервале времени [0, Т]. В общем случае критерием интерпретации результатов моделирования является нестационарный случайный n-мерный процесс q(t), 0≤t≤T. Полагаем для определенности, что состояние моделируемой системы S проверяется каждые Δ* временных единиц, т. е, используется «принцип Δt». При этом вычисляют значения q(jΔt), $j=\overline{0,k}$, критерия q(j). Таким образом, о свойствах случайного процесса q(i) судят по свойствам случайной последовательности q(j'Δi), $j=\overline{0,k}$.

$$\vec{q}=(\vec{q}\,(0),\ \vec{q}\,(\Delta t),\ \ldots,\ \vec{q}\,[(k-1)\Delta t],\ \vec{q}\,(T)),\ m=n(k+1),\ T=k\Delta t.$$

Процесс функционирования системы S на интервале [0, Т] моделируется N-кратно. Работа модели на интервале [0, Т] называется **прогоном модели.**

На схеме, изображенной на рисунке 3.4, обозначено: I≡i; J≡j; К≡к; N≡N; T≡t; DT≡Δt.

В общем случае алгоритмы фиксации и статистической обработки данных моделирования содержат три цикла. Полагаем, что имеется машинная модель Мм системы S:

1. Внутренний цикл (блоки 5 — 8) позволяет получить последовательность $q_i(t) = q\ (j\Delta t)$, $j=\overline{0,k}$ в моменты времени t=0, Δt, 2Δt, kΔt = Т. Основной блок 7 реализует процедуру вычисления последовательности $q_t(t)$: ВЫЧ [QI(T)]. Именно в этом блоке имитируется процесс функционирования моделируемой системы S на интервале времени [0, Т].

2. Промежуточный цикл (блоки 3 — 10), в котором организуется N-кратное повторение прогона модели, позволяющее после соответствующей статистической обработки результатов судить об оценках характеристик моделируемого варианта системы. Окончание моделирования варианта системы S может определяться не только заданным числом реализаций (блок 10), как это показано на схеме, но и заданной точностью результатов моделирования. В этом цикле содержится блок 9, реализующий процедуру фиксации результатов моделирования по i-му прогону модели qt(t): ФРМ

3. Внешний цикл (блоки 1 —12) охватывает оба предшествующих цикла и дополнительно включает блоки 1, 2,11, 12, управляющие последовательностью моделирования вариантов системы S. Здесь организуется поиск оптимальных структур, алгоритмов и параметров системы S, т. е. блок 11 обрабатывает результаты моделирования исследуемого к-го варианта системы ОРМ [QK], блок 12 проверяет удовлетворительность полученных оценок характеристик процесса

функционирования системы q(t) требуемым (ведет поиск оптимального варианта системы ПОВ [S(K)], блок 1 изменяет структуру, алгоритмы и параметры системы S на уровне ввода исходных данных для очередного к-го варианта системы ВИД [S(K)]. Блок 13 реализует функцию выдачи результатов моделирования по каждому k-му варианту модели системы Sk т. е. ВРМ ГОТ

Рассмотренная схема позволяет вести статистическую обработку результатов моделирования в наиболее общем случае при нестационарном критерии q(t). В частных случаях можно ограничиться более простыми схемами.

Если свойства моделируемой системы S определяются значением критерия q(t) в некоторый заданный момент времени, например в конце периода функционирования модели $t=k\Delta t=T$, то обработка сводится к оценке распределения n -мерного вектора q(t) по независимым реализациям $t_{ji}(t)$, $i=\overline{0,N}$, полученным в результате N прогонов модели.

Если в моделируемой системе S по истечению некоторого времени с начала работы $t_o = k_o\Delta t$ установится стационарный режим, то о нем можно судить по одной, достаточно длинной реализации $q_t(t)$ критерия q(t), стационарного и эргодического на интервале $[t_0, T]$. Для рассмотренной схемы это означает, что исключается средний цикл (n= 1) и добавляется оператор, позволяющий начать обработку значений $q_1(j\Delta t)$ при $j\geq k_0$.

Другая особенность применяемых на практике методов статистической обработки результатов моделирования связана с исследованием процесса функционирования систем с помощью моделей блочной конструкции. В этом случае часто приходится применять раздельное моделирование отдельных блоков модели, когда имитация входных воздействий для одного блока проводится на основе оценок критериев, полученных предварительно на другом блоке модели. При раздельном моделировании может иметь место либо непосредственная запись в накопителе реализаций критериев, либо их аппроксимация, полученная на основе статистической обработки результатов моделирования с последующим использованием генераторов случайных чисел для имитации этих воздействий.

Подэтапы третьего этапа моделирования. Прежде чем приступить к последнему, третьему, этапу моделирования системы, необходимо для его успешного проведения иметь четкий план действий, сводящийся к выполнению следующих основных подэтапов.

3.1. Планирование машинного эксперимента с моделью системы. Перед выполнением рабочих расчетов на ЭВМ должен быть составлен план проведения эксперимента с указанием комбинаций переменных и параметров, для которых должно проводиться моделирование системы S. Планирование машинного эксперимента призвано дать в итоге максимальный объем необходимой информации об объекте моделирования при минимальных затратах машинных ресурсов. При этом различают

стратегическое и тактическое планирование машинного эксперимента. При стратегическом планировании эксперимента ставится задача построения оптимального плана эксперимента для достижения цели, поставленной перед моделированием (например, оптимизация структуры, алгоритмов и параметров системы S, исследуемой методом моделирования на ЭВМ). Тактическое планирование машинного эксперимента преследует частные цели оптимальной реализации каждого конкретного эксперимента из множества необходимых, заданных при стратегическом планировании (например, решение задачи выбора оптимальных правил остановки при статистическом моделировании системы S на ЭВМ). Для получения наиболее эффективного плана машинного эксперимента необходимо использовать статистические методы.

3.2. Определение требований к вычислительным средствам. Необходимо сформулировать требования по времени использования вычислительных средств, т. е. составить график работы на одной или нескольких ЭВМ, а также указать те внешние устройства ЭВМ, которые потребуются при моделировании. При этом также рационально оценить, исходя из требуемых ресурсов, возможность использования для реализации конкретной модели персональной ЭВМ или локальной вычислительной сети.

3.3. Проведение рабочих расчетов. После составления программы модели и плана проведения машинного эксперимента с моделью системы S можно приступить к рабочим расчетам на ЭВМ, которые обычно включают в себя:

а) подготовку наборов исходных данных для ввода в ЭВМ;

б) проверку исходных данных, подготовленных для ввода;

в) проведение расчетов на ЭВМ;

г) получение выходных данных, т. е. результатов моделирования.

Проведение машинного моделирования рационально выполнять в два этапа: контрольные, а затем рабочие расчеты. Причем контрольные расчеты выполняются для проверки машинной модели Мм и определения чувствительности результатов к изменению исходных данных.

3.4. Анализ результатов моделирования системы. Чтобы эффективно проанализировать выходные данные, полученные в результате расчетов на ЭВМ, необходимо знать, что делать с результатами рабочих расчетов и как их интерпретировать. Эти задачи могут быть решены на основании предварительного анализа на двух первых этапах моделирования системы S. Планирование машинного эксперимента с моделью Мм позволяет вывести необходимое количество выходных данных и определить метод их анализа. При этом необходимо, чтобы на печать выдавались только те результаты, которые нужны для дальнейшего анализа. Также необходимо полнее использовать возможности ЭВМ с точки зрения обработки результатов моделирования и представления этих результатов в наиболее наглядном виде. Вычисление статистических характеристик перед выводом

результатов из ЭВМ повышает эффективность применения машины и сводит к минимуму обработку выходной информации после ее вывода из ЭВМ.

3.5. Представление результатов моделирования. Как уже отмечалось, необходимо на третьем этапе моделирования уделить внимание форме представления окончательных результатов моделирования в виде таблиц, графиков, диаграмм, схем и т. п. Целесообразно в каждом конкретном случае выбрать наиболее подходящую форму, так как это существенно влияет на эффективность их дальнейшего употребления заказчиком. В большинстве случаев наиболее простой формой считаются таблицы, хотя графики более наглядно иллюстрируют результаты моделирования системы S. При диалоговых режимах моделирования наиболее рациональными средствами оперативного отображения результатов моделирования являются средства мультимедиа технологии.

3.6. Интерпретация результатов моделирования. Получив и проанализировав результаты моделирования, их нужно интерпретировать по отношению к моделируемому объекту, т. е. системе S. Основное содержание этого подэтапа — переход от информации, полученной в результате машинного эксперимента с моделью Ми, к информации применительно к объекту моделирования, на основании которой и будут делаться выводы относительно характеристик процесса функционирования исследуемой системы S.

3.7. Подведение итогов моделирования и выдача рекомендаций. Проведение этого подэтапа тесно связано с предыдущим вторым этапом (см. п. 3.3). При подведении итогов моделирования должны быть отмечены главные особенности, полученные в соответствии с планом эксперимента над моделью Мм результатов, проведена проверка гипотез и предположений и сделаны выводы на основании этих результатов. Все это позволяет сформулировать рекомендации по практическому использованию результатов моделирования, например на этапе проектирования системы S.

3.8. Составление технической документации по третьему этапу. Эта документация должна включать в себя:

а) план проведения машинного эксперимента;

б) наборы исходных данных для моделирования;

в) результаты моделирования системы;

г) анализ и оценку результатов моделирования;

д) выводы по полученным результатам моделирования;

е) указание путей дальнейшего совершенствования машинной модели и возможных областей ее приложения.

Полный комплект документации по моделированию конкретной системы S на ЭВМ должен содержать техническую документацию по каждому из трех рассмотренных этапов.

Таким образом, процесс моделирования системы S сводится к выполнению перечисленных этапов моделирования. На этапе построения

концептуальной модели Мк проводится исследование моделируемого объекта, определяются необходимые аппроксимации и строится обобщенная схема модели, которая преобразуется в машинную модель Мм на втором этапе моделирования путем последовательного построения логической схемы модели и схемы программы. На последнем этапе моделирования проводят рабочие расчеты на ЭВМ, получают и интерпретируют результаты моделирования системы S.

Рассмотренная последовательность этапов и подэтапов отражает наиболее общий подход к построению и реализации модели системы S. В дальнейшем остановимся на наиболее важных составляющих процесса моделирования.

Вопросы для повторения и закрепления материала

1. В чем заключаются принципы построения моделирующих алгоритмов?
2. В чем заключается суть формы представления моделирующего алгоритма в виде обобщенной схемы?
3. В чем заключается суть формы представления моделирующего алгоритма в виде детальной схемы?
4. В чем заключается суть формы представления моделирующего алгоритма в виде логической схемы?
5. В чем заключается суть формы представления моделирующего алгоритма в виде схемы программы?
6. Охарактеризуйте особенности получения результатов моделирования.
7. Необходимо ли планирование машинного эксперимента с моделью?

Задания для самостоятельной работы

1. Изучить подходы получения и интерпретации результатов моделирования, отличные о т представленного в лекции.

Лабораторная работа №1. Построение дискретно-детерминированных моделей посредством конечных автоматов Мили и Мура

Цель работы: овладеть практическими навыками построения моделей дискретно-детерминированного класса на основании конечно-автоматных структур

Теоретическая часть

Перед выполнением лабораторной работы необходимо овладеть теоретическими знаниями в области построения конечных автоматов, которые содержатся в 5 теме теоретического материала.

Построение дискретно-детерминированных моделей будет осуществляться в среде MatLab, а именно при помощи среды разработки моделей Simulink и Stateflow, который является интерактивным инструментом разработки в области моделирования сложных, управляемых событиями систем. Руководство пользователя инструмента Stateflow описано Рогачевым Геннадием Николаевичем - к.т.н., доцент кафедры "Автоматика и управление в технических системах" Самарского государственного технического университета, с которым можно ознакомится по ссылке: http://matlab.exponenta.ru/stateflow/book1/index.php.

Введение в Stateflow

Stateflow - инструмент для численного моделирования систем, характеризующихся сложным поведением. К числу таких систем относятся гибридные системы. Примерами гибридных систем могут служить системы управления, используемые в промышленности (автоматизированные технологические процессы), в быту (сложные бытовые приборы), в военной области (высокотехнологичные виды вооружений), в сфере космонавтики, транспорта и связи. Все эти системы состоят из аналоговых и дискретных компонентов. Поэтому гибридные системы - это системы со сложным взаимодействием дискретной и непрерывной динамики. Они характеризуются не только непрерывным изменением состояния системы, но и скачкообразными вариациями в соответствии с логикой работы управляющей подсистемы, роль которой как правило выполняет то или иное вычислительное устройство (конечный автомат).

В том случае, когда логика работы управляющей подсистемы является жесткой, а внешние условия относительно стабильны, говорят о трансформационных системах (рисунок 1.1). Для таких систем фазы получения информации, её обработки и выдачи выходных сигналов четко разграничены. На момент обращения к системе все входные сигналы определены. Сигналы на выходах образуются после некоторого периода

вычислений. Вычисления производятся по некоторому алгоритму, трансформирующему (преобразующему) входной набор данных в выходной.

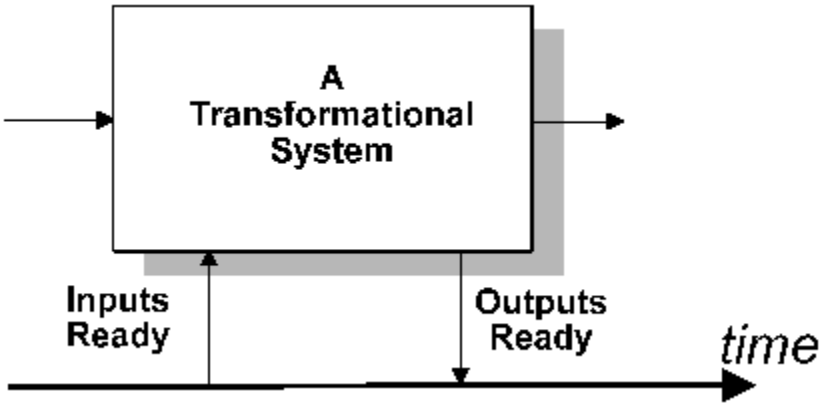


Рисунок 1.1 - Трансформационная система

В противоположном случае систему относят к классу управляемых событиями или реактивных (рисунок 1.2). Реактивная - это такая динамическая система, которая воспринимает внешние дискретные воздействия и отвечает своими реакциями на эти воздействия. Причем реакции системы различны и сами зависят как от воздействий, так и от состояния, в котором система находится. Основное отличие реактивных систем от трансформационных - в принципиальной непредсказуемости моментов поступления тех или иных воздействий. Эта непредсказуемость - следствие изменчивости условий, в которых такие системы работают.

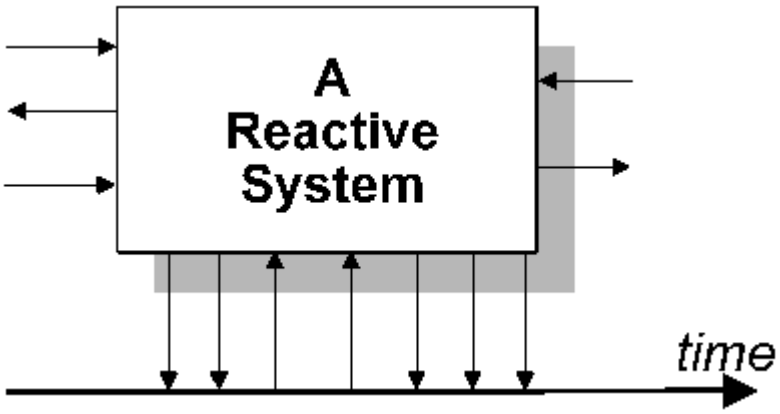


Рисунок 1.2 - Реактивная система

Пример простой реактивной системы - контроллер светофора, управляемого пешеходами. Его входы никогда не приобретут законченного вида - сигналы на них поступают постоянно и в неизвестной заранее последовательности.

В настоящее время для моделирования дискретной динамики реактивных систем широко используется предложенный Д. Харелом визуальный формализм - Statechart (диаграммы состояний и переходов).

Основные неграфические компоненты таких диаграмм - это событие и действие, основные графические компоненты - состояние и переход.

Событие - нечто, происходящее вне рассматриваемой системы, возможно требуя некоторых ответных действий. События могут быть вызваны поступлением некоторых данных или некоторых задающих сигналов со стороны человека или некоторой другой части системы. События считаются мгновенными (для выбранного уровня абстрагирования).

Действия - это реакции моделируемой системы на события. Подобно событиям, действия принято считать мгновенными.

Состояние - условия, в которых моделируемая система пребывает некоторое время, в течение которого она ведет себя одинаковым образом. В диаграмме переходов состояния представлены прямоугольными полями со скругленными углами.

Переход - изменение состояния, обычно вызываемое некоторым значительным событием. Как правило, состояние соответствует промежутку времени между двумя такими событиями. Переходы показываются в диаграммах переходов линиями со стрелками, указывающими направление перехода.

Каждому переходу могут быть сопоставлены условия, при выполнении которых переход осуществляется.

С каждым переходом и каждым состоянием могут быть соотнесены некоторые действия. Действия могут дополнительно обозначаться как действия, выполняемые однократно при входе в состояние; действия, выполняемые многократно внутри некоторого состояния; действия, выполняемые однократно при выходе из состояния.

Рассмотрим в качестве примера диаграмму состояний и переходов цифровых часов, представленную на рисунке 1.3.

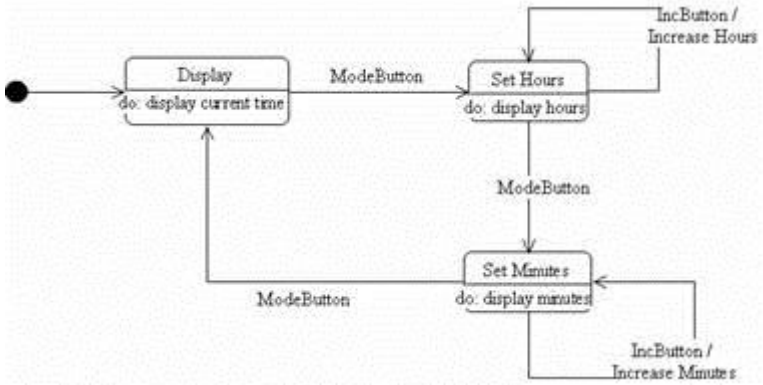


Рисунок 1.3 - Диаграмма состояний и переходов цифровых часов

На часах имеется две кнопки: ModeButton и IncButton (Кнопка Образа Действия и Кнопка Увеличения). Нажатие любой из них генерирует событие, которое может вызывать переход из одного состояния в другое. Имеются три состояния: Display, Set Hours and Set Minutes (Дисплей, Установка Часов и Установка Минут). Состояние Дисплей - начальное состояние (что

обозначается стрелкой, направленной от блока перехода по умолчанию в виде черного круга). В состоянии Set Hours (Установка Часов), событие ModeButton вызывает переход к состоянию Set Minutes (Установка Минут), тогда как событие IncButton увеличивает текущее время (число часов), которое отображается на экране, причем это происходит без изменения состояния. Каждому состоянию часов соответствует действие, записанное ниже горизонтальной линии. Оно начинает выполняться после того, как переход в это состояние произошел.

Для предотвращения эффекта возрастания сложности при моделировании больших систем были предложены дальнейшие усовершенствования. Наряду с состояниями теперь могут использоваться **гиперсостояния** (суперсостояния), объединяющие несколько состояний, имеющих идентичную реакцию на одно и то же событие. При этом вместо изображения таких переходов в некоторое состояние из всех состояний, охватываемых гиперсостоянием, изображается только один переход из гиперсостояния в указанное состояние (обобщение переходов). Гиперсостояния теоретически могут иметь произвольную глубину вложения. Переходы из гиперсостояния связаны со всеми уровнями вложения. Гиперсостояния могут объединять ИЛИ-состояния (последовательные состояния) и И-состояния (параллельные состояния). В первом случае, перейдя в гиперсостояние, система может находиться только в одном из состояний. Во втором случае система, перейдя в гиперсостояние, будет пребывать одновременно в нескольких состояниях.

Определение Stateflow интерфейса

Каждый блок Stateflow соответствует единственной диаграмме Stateflow. Блок Stateflow связывает с моделью Simulink с помощью интерфейса. С помощью интерфейса блок Stateflow подключается к источникам, поступающим от модели Simulink (данные, события, код пользователя).

Stateflow диаграммы управляются событиями. События могут быть локальными для блока Stateflow или могут поступать от модели Simulink и источников кода, внешних к Simulink. Данные также могут быть локальными для блока Stateflow или могут поступать от модели Simulink и источников кода, внешних к Simulink.

Вы должны определить интерфейс для каждого блока Stateflow. Определение интерфейса для блока Stateflow может включать некоторые или все задач из списка:

- Определение метода модификации блока Stateflow.
- Определение **Output to Simulink** (Выходных к Simulink) событий.
- Добавление и определение нелокальных событий и нелокальных данных в пределах диаграммы Stateflow.

- Определение отношений с любыми внешними источниками.

Объекты Stateflow диаграммы

Далее приводится пример Stateflow диаграммы, в которой используются основные графические компоненты. Кроме того, детально описываются эти графические компоненты, а также некоторые неграфические объекты и связи между ними (рисунок 1.4).

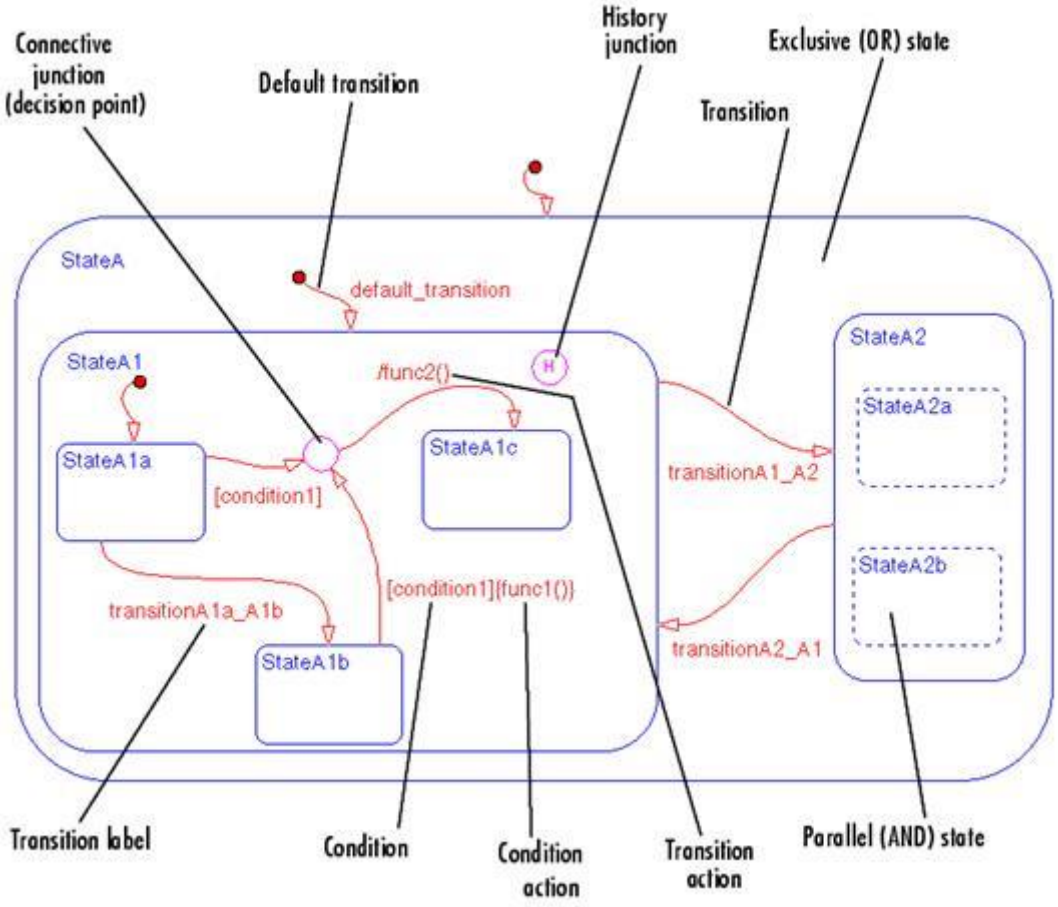


Рисунок 1.4 - Объекты Stateflow диаграммы

Состояние (State)

Состояние описывает режим управляемой событиями системы. Динамические переходы состояний от активности к неактивности базируются на событиях и условиях. Каждое состояние имеет родителя. В диаграмме Stateflow, состоящей из единственного состояния, родитель состояния - непосредственно диаграмма Stateflow (также называемая корнем диаграммы Stateflow). Вы можете размещать состояния в пределах других состояний высшего уровня. На рисунке StateA1 - потомок в иерархии по отношению к StateA.

Состояние также имеет хронологию. Хронология обеспечивает эффективные средства базирования будущего действия на прошлом действии.

Состояния имеют метки, которые могут определить действия, выполненные в последовательности, основанной на типе действия. Типы действия - **entry** (на входе), **during** (в течение), **exit** (на выходе) и **on event_name** (в случае события с именем _).

В примере автоматической передачи, передача может быть в нейтральном положении или включена в работу. Два состояния этой системы - neutral (нейтраль) и engaged (включена) (рисунок 1.5).

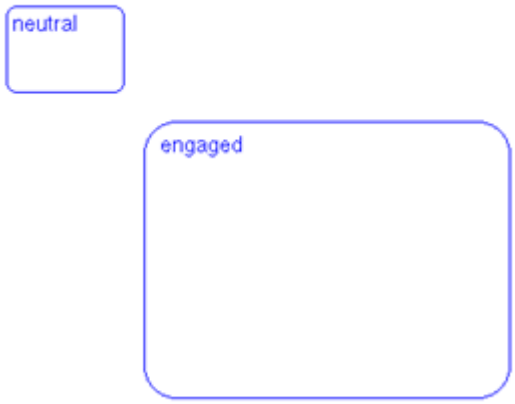


Рисунок 1.5 – Состояния модели автоматической коробки передач

Stateflow обеспечивает два типа состояний: параллельный (И) и исключительный (ИЛИ) тип состояния. Параллелизм представлен И (параллельными) состояниями. Автоматическая передача - пример исключительного (ИЛИ) состояния. Исключительные (ИЛИ) состояния используются, чтобы описать режимы, которые являются взаимоисключающими. Система находится либо в нейтральном состоянии, либо во включенном состоянии в каждый момент времени.

Переходы (Transitions)

Переход - графический объект, который в большинстве случаев связывает один объект с другим. Один конец перехода приложен к объекту-источнику, а другой конец - к объекту-адресату. Источник - то место, где переход начинается, а адресат - то место, где переход заканчивается. Метки переходов описывают обстоятельства, под действием которых система переходит из одного состояния к другому. Эти обстоятельства - наступление некоторых событий, которые заставляют переход совершаться. На рисунке 1.4 переход от StateA1 к StateA2 помечен событием transitionA1_A2, которое заставляет переход произойти.

Рассмотрим снова автоматическую передачу (рисунок 1.6). Clutch_engaged (включение передачи) - событие, которое требуется, чтобы осуществить переход из нейтрального положения в состояние "включено".

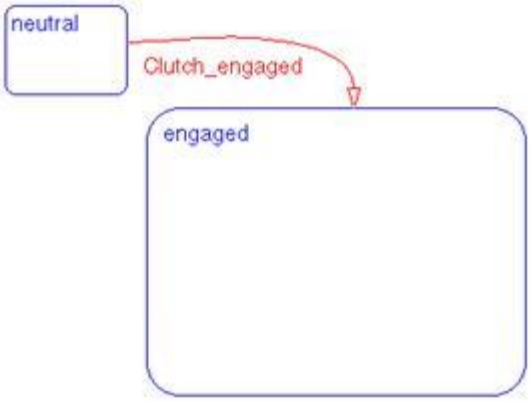


Рисунок 1.6 – Состояния и переход модели автоматической коробки передач

События (Events)

События управляют выполнением диаграммы Stateflow, но являются неграфическими объектами и таким образом не представлены непосредственно в диаграмме Stateflow. Все события, которые имеют отношение к диаграмме Stateflow, должны быть определены. Наступление события заставляет статус состояния (активно - неактивно) в диаграмме Stateflow изменяться. Наступление события может запускать переход, и тогда он происходит, или может запускать действие, и тогда оно выполняется. События наступают по-нисходящей, начиная от родителя события в иерархии.

События создаются и изменяются при помощи Stateflow Explorer (Stateflow проводника). События могут быть созданы на любом уровне иерархии. Событие имеет такое свойство, как видимость. Видимость определяет, является ли событие

- Локальным для диаграммы Stateflow
- Входит в Stateflow диаграмму от модели Simulink
- Выходит из Stateflow диаграммы в модель Simulink
- Экспортируется в код, внешний к Stateflow диаграмме и модели Simulink
- Импортируется из источника кода, внешнего к Stateflow и Simulink

Данные (Data)

Объекты-данные используются, чтобы хранить числовые значения для применения в диаграмме Stateflow. Они являются неграфическими объектами и таким образом не представлены непосредственно в диаграмме Stateflow.

Данные создаются и изменяются в Stateflow Explorer. Они могут быть созданы на любом уровне иерархии. Данные имеют такое свойство, как видимость. Видимость определяет для объектов-данных одну из следующих возможностей:

- Быть локальными для диаграммы Stateflow
- Поступать в Stateflow диаграмму от модели Simulink
- Выходить из Stateflow диаграммы в модель Simulink
- Быть временными данными
- Быть определенными в рабочем пространстве MATLAB
- Быть Константами
- Экспортировается в код, внешний к Stateflow диаграмме и модели Simulin
- Импортировается из источника кода, внешнего к Stateflow и Simulink

Иерархия

Иерархия дает возможность организовать сложные системы, определяя предка и структуру объектов-потомков. Иерархически построенный проект обычно сокращает число переходов и приводит к четким, понятным диаграммам. Stateflow поддерживает иерархическую организацию как для диаграмм, так и для состояний. Диаграммы могут существовать внутри других диаграмм. Диаграмма, которая существует в другой диаграмме, называется поддиаграммой.

Точно так же состояния могут существовать внутри других состояний. Stateflow представляет иерархию состояний с суперсостояниями и подсостояниями. Например, эта диаграмма Stateflow имеет суперсостояние, которое содержит два подсостояния (рисунок 1.7).

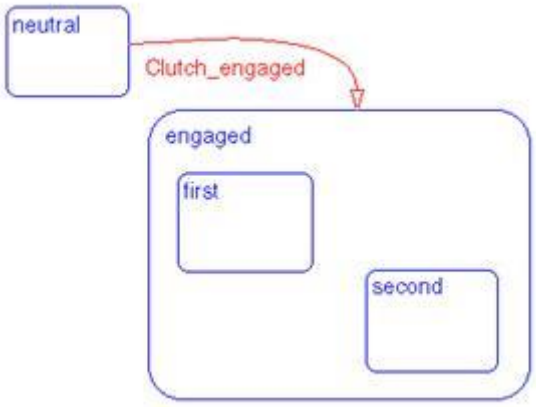


Рисунок 1.7 – Иерархия состояний модели автоматической коробки передач

Суперсостояние engaged (передача включена) содержит подсостояния first (первая передача) и second (вторая передача). Суперсостояние engaged - предок в иерархии по отношению к состояниям first и second. Когда событие clutch_engaged происходит, система переходит из нейтрального состояния к суперсостоянию "включено". Переходы внутри суперсостояния преднамеренно опущены в этом примере для простоты.

Выход из состояния высшего уровня или суперсостояния также подразумевает выход из любых активных подсостояний этого суперсостояния. Переходы могут пересекать границы суперсостояния, чтобы достичь подсостояния-адресата. Если подсостояние активно, его родительское состояние (суперсостояние) также активно.

Условия (Conditions)

Условие - булево выражение, определяющее, что переход происходит, если указанное выражение является истинным. На рисунке 1.8 компонента Stateflow диаграммы [condition1] представляет булево выражение, которое должно быть истинным, чтобы переход произошел.

В автоматической коробке передач переход с первой скорости ко второй происходит, если булево условие [speed > threshold] ([скорость > пороговое_значение]) истинно (рисунок 1.8).

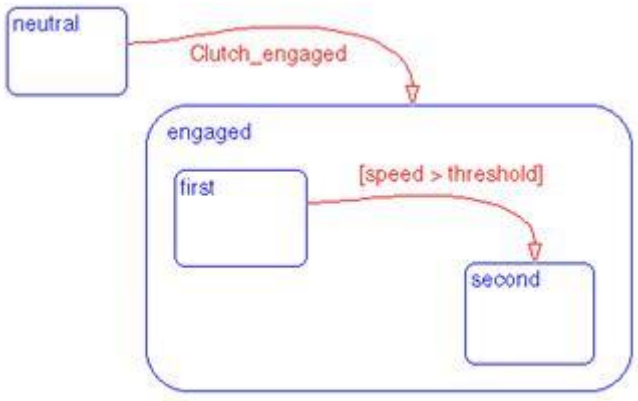


Рисунок 1.8 – Условный переход модели автоматической коробки передач

Хронологические соединения (History Junction)

Хронология - средство для определения подсостояния-адресата перехода по предыстории. Если суперсостояние с исключительной (ИЛИ) декомпозицией имеет хронологическое соединение, подсостоянием-адресатом при переходе будет подсостояние, посещенное до этого последним. Хронологическое соединение применяется к тому уровню иерархии, в котором присутствует. Хронологическое соединение отменяет любые переходы по умолчанию. На рисунке 1.4 хронологическое соединение в StateA1 указывает, что, когда переход к StateA1 происходит, становится активным то из подсостояний (StateA1a, StateA1b или StateA1c), которое было активным в последнюю очередь.

В автоматической передаче хронология указывает, что, когда clutch_engaged вызывает переход от нейтрали до суперсостояния включено, становится активным то подсостояние (первая или вторая скорость), которое было активным в последнюю очередь (рисунок 1.9).

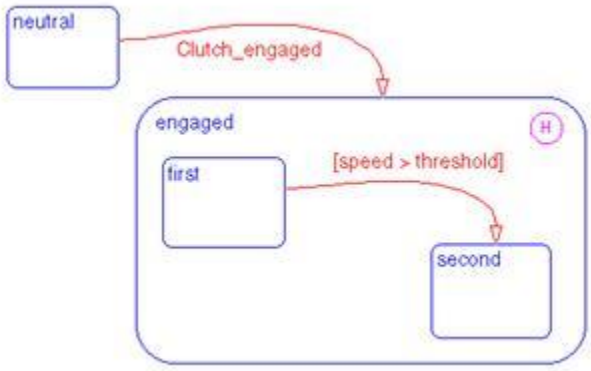


Рисунок 1.9 – Хронологические соединения модели автоматической коробки передач

Действия (Actions)

Действия - это результат выполнения какой-либо части диаграммы Stateflow. Действие может быть выполнено в результате перехода от одного состояния к другому. Действие может быть также реакцией на состояние. На рисунке 1.4 сегмент перехода от StateA1b до соединения помечен действием func1() условия condition 1, а сегмент перехода от соединения до StateA1c помечен действием func2() перехода.

Переход, заканчивающийся в состоянии, может иметь действие условия (condition action) и действие перехода (transition action), как рассмотрено ниже (рисунок 1.10). Однако переходы, которые заканчиваются в соединениях, могут иметь только действия условий (не допускаются действия переходов).

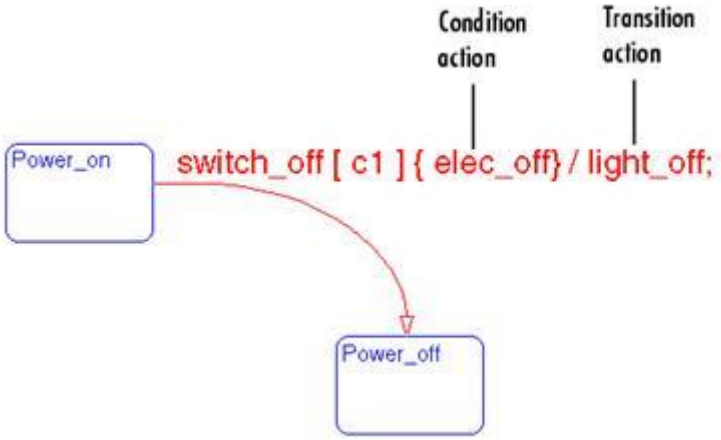


Рисунок 1.10 – Действия модели автоматической коробки передач

Состояния могут иметь действия (рисунок 1.11) **have entry** (на входе), **during** (в течение), **exit** (на выходе) и **on*event_name*** (в случае события с именем).

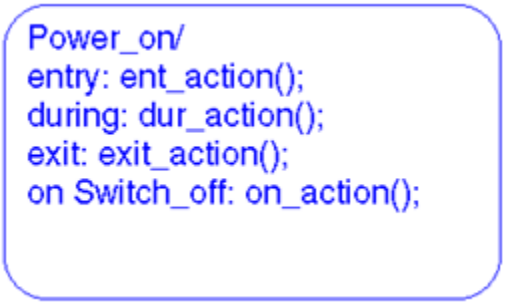


Рисунок 1.11 – Пример действий состояний

Язык действий определяет типы действий, которые Вы можете использовать и связанные с ними системы обозначений. Действием может быть обращение к функции, наступление события, присвоение некоторого значения переменной и т.д.

Stateflow поддерживает парадигмы моделирования конечного автомата Мура и Мили. В Мили модели действия связаны с переходами, в то время как в Мура модели связаны с состояниями. Stateflow поддерживает действия состояний, действия переходов и действия условий.

Переходы по умолчанию (Default Transitions)

Переходы по умолчанию определяют, какое из нескольких исключительных (ИЛИ) состояний должно быть активным, когда имеется неопределенность между двумя или более исключительными (ИЛИ) состояниями на одном уровне в иерархии.

В следующей подсистеме Lights (Осветительные приборы) переход по умолчанию к подсостоянию Lights.Off (Осветительные приборы.выключены) указывает, что, когда суперсостояние Lights становится активным, Lights.Off подсостояние становится активным по умолчанию (рисунок 1.12).

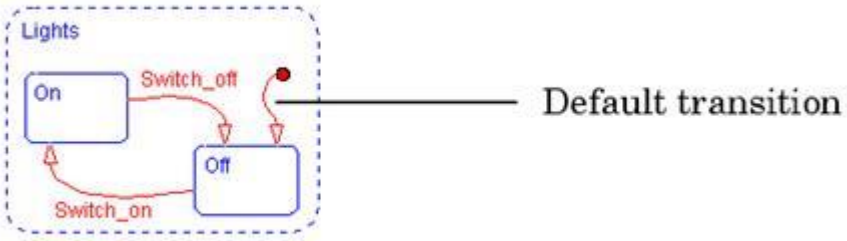


Рисунок 1.12 – Переход по умолчанию модели автоматической коробки переда

Обратите внимание! Хронологические соединения отменяют переходы по умолчанию в суперсостояниях с исключительными (ИЛИ) декомпозициями.

Обратите внимание! В параллельных (И) состояниях переходы по умолчанию всегда должны присутствовать, чтобы указать, какие из его исключительных (ИЛИ) состояний активны, когда параллельное состояние становится активным.

Соединения (Connective Junctions)

Соединения - точки принятия решений в системе. Соединение - графический объект, который упрощает Stateflow схематические представления и облегчает порождение эффективного кода. Соединения обеспечивают альтернативные способы представить желаемое поведение системы. На представленной выше Stateflow диаграмме соединение используется как точка принятия решения для двух сегментов перехода, завершающихся в состоянии StateA1c.

Следующий пример показывает, как соединения (отображаемые в виде окружностей) используются для конструкции if (рисунок 1.13).

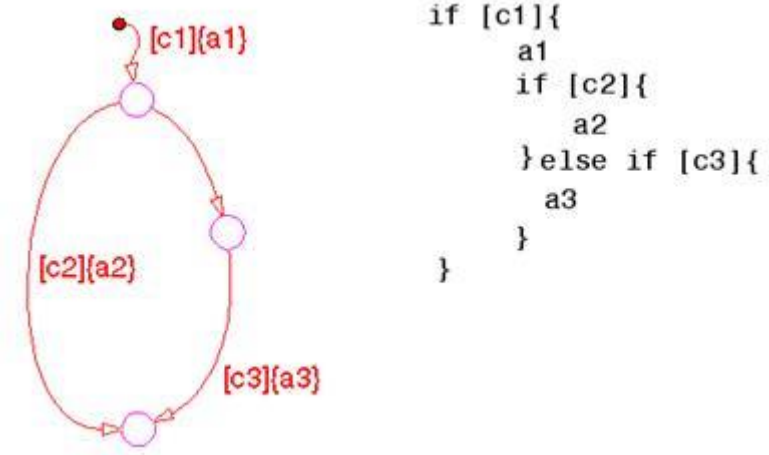


Рисунок 1.13 - Пример соединений используемых для конструкции if

Этот фрагмент выполняется следующим образом:

1. Если условие [c1] истинно, условное действие a1 выполняется и происходит безусловный переход к первому (верхнему) соединению.
2. Stateflow определяет, какой сегмент перехода верхнего соединения выбрать (можно выбрать только один). Соединения с условиями имеют приоритет над соединениями без условий, т. о. переход с условием [c2] рассматривается первым.
3. Если условие [c2] истинно, действие a2 выполняется и происходит переход к нижнему соединению. Так как нет перехода, выходящего из этого соединения, выполнение диаграммы завершено.
4. Если условие [c2] ложно, происходит безусловный перехода по правому из сегментов (он не имеет условия).
5. Если условие [c3] истинно, условное действие a3 выполняется и происходит переход к нижнему соединению. Выполнение диаграммы завершено.

6. Если условие [c3] ложно, выполнение заканчивается на среднем соединении.

Нотации Stateflow.

Приводится описание отдельных объектов Stateflow, из которых строятся Stateflow-диаграммы.

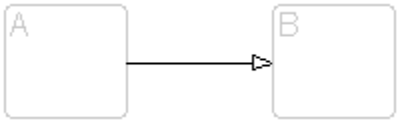
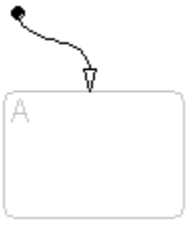
Виды объектов Stateflow

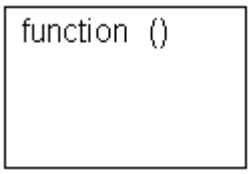
Перечислим различные типы доступных в Stateflow объектов.

- графические объекты - диаграммы, состояния, блоки, функции, переходы и соединения, с которыми вы работаете в редакторе Stateflow.
- неграфические объекты - данные и события, которые представлены в текстовом виде в редакторе диаграммы Stateflow или инструментах Проводника Stateflow.
- словарь данных - объединяет все объекты Stateflow (графические и неграфические) в иерархическую базу данных.
- представление иерархии - отображает иерархию объектов Stateflow в словаре данных для графических и неграфических объектов.

Графические Объекты

В таблице приводится название (Name) и условное обозначение (Notation) каждого графического объекта Stateflow. Объект появляется, когда вы рисуете диаграмму в редакторе диаграмм, для этого используется соответствующий значок (Toolbar icon) на панели инструментов.

Name	**Notation**	**Toolbar Icon**
State (состояние)		
Transition (переход)		-
Default transition (переход по умолчанию)		
Connective junction (подключаемое соединение)		

History junction **(соединение с памятью)**		
Box (ящик)		
Graphical function (графическая функция)		

Неграфические Объекты

Stateflow определяет неграфические объекты, описанные в следующих разделах:

- События
- Данные
- Коды

События, данные и коды не имеют графического представления в редакторе диаграмм Stateflow. Но их можно увидеть в проводнике Stateflow.

События

События - это объекты Stateflow, которые могут переключать всю диаграмму Stateflow или конкретные действия в диаграмме. Так как диаграммы Stateflow выполняются посредством реакции на события, вы определяете и программируете события в ваших диаграммах, чтобы контролировать их выполнение. Вы можете распространять событие на все объекты, либо посылать событие на определенный объект. Вы можете подробно определить события, которые вы сами назначаете. Так же вы можете определить события по умолчанию, имеющие место, например, при входе в состояние.

Данные

Диаграмма Stateflow хранит и восстанавливает данные, которые используются для управления действием диаграммы. Данные Stateflow помещены в свое собственное рабочее пространство, но вы можете иметь доступ к данным, которые располагаются в модели Simulink или приложениях, которые встроены в машину Stateflow. Когда вы создаете диаграмму Stateflow, вы должны определить внутренние и внешние данные, которые вы используете в языке действий диаграммы Stateflow.

Коды

Вы строите коды в Stateflow для выполнения приложений, которые вы программируете в диаграммах Stateflow и моделях Simulink. Код - это программа, которая выполняет диаграммы Stateflow и модели Simulink, включенные в машину Stateflow. Вы строите коды моделирования

(называемые sfun) для выполнения вашей модели. Вы строите коды Мастерской Реального Времени (называемые rtw) для выполнения модели Simulink или поддержки окружения процессора. Вы строите обычные коды (они могут быть названы как угодно, за исключением sfun или rtw), чтобы поместить ваше приложение в определенное окружение.

Словарь данных

Словарь данных - это база данных, включающая в себя всю информацию о графических и неграфических объектах. Словарь данных для графических объектов создается автоматически. Для неграфических объектов необходимо определять данные, используя Проводник. Грамматический отладчик анализирует входы и связи между входами в словаре данных, чтобы проверить правильность нотаций.

Как представлена иерархия

Нотации Stateflow поддерживают представление графических объектов в виде иерархии в диаграммах Stateflow. Рассмотрим следующие примеры представления иерархий в диаграммах Stateflow:

- Пример представления иерархии состояний
- Пример представления иерархии переходов

Иерархия данных и событий, как неграфических объектов, представлена в проводнике. Иерархия данных и событий определяются при их создании назначением им родительского объекта.

Пример представления иерархии состояний

В представленном примере изображено одно состояние, в рамках которого другое состояние показывает, что внутреннее состояние - это подсостояние или потомок внешнего состояния или надсостояния, и внешнее состояние - родительское по отношению к внутреннему (рисунок 1.17).

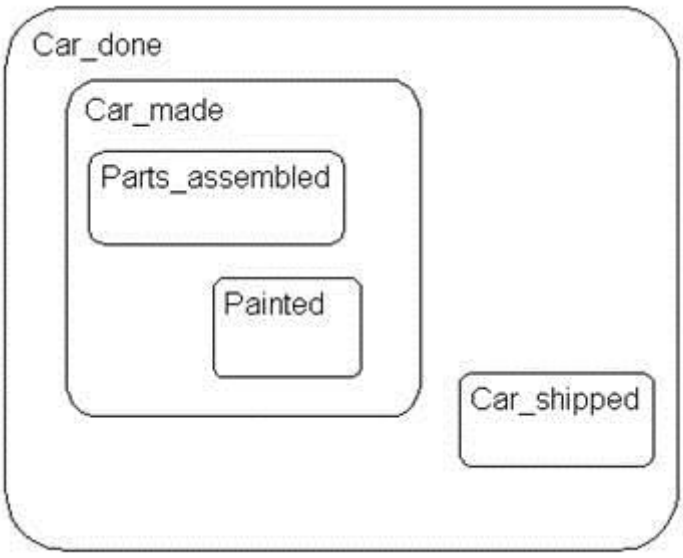


Рисунок 1.17 – Пример представления иерархии состояний

В этом примере диаграмма Stateflow - родитель по отношению к состоянию Car_done. Состояние Car_done - родитель состояний Car_made и Car_shipped. Состояние Car_made - также родитель состояний Parts_assembled и Painted. Вы также можете сказать, что состояния Parts_assembled и Painted - потомки состояния Car_made. Иерархия Stateflow также может быть представлена в текстовом виде, тогда символ слэш (/) представляет диаграмму Stateflow, и каждый уровень иерархии состояний разделяется символом (.). Далее показано текстовое представление иерархии объектов предыдущего примера:

- /Car_done
- /Car_done.Car_made
- /Car_done.Car_shipped
- /Car_done.Car_made.Parts_assembled
- /Car_done.Car_made.Painted

Пример представления иерархии переходов

Это пример того, как представляется иерархия переходов (рисунок 1.18).

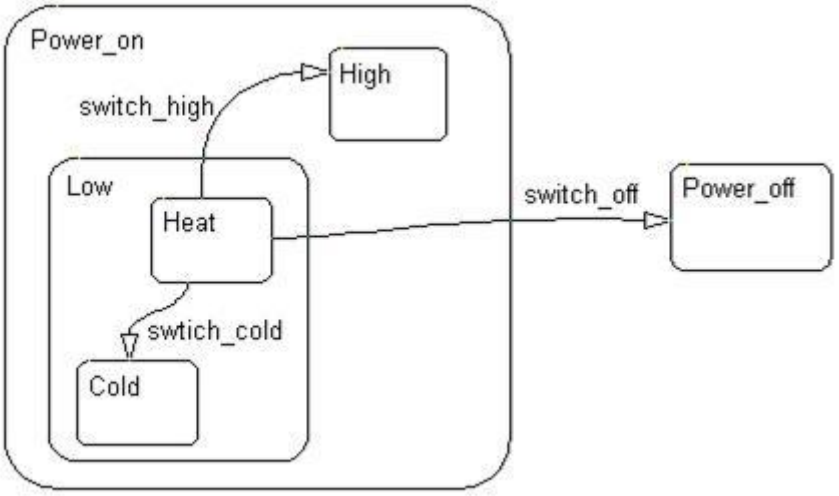


Рисунок 1.18 – Пример представления иерархии переходов

Иерархия переходов описывает родителя перехода, источник и расположение. Родитель перехода - это низший уровень из тех, которые включают в себя источник и расположение перехода. Машина - это корень (основание) иерархии. Диаграмма Stateflow представляется символом /. Каждый уровень иерархии состояний разделяется символом (.). Три перехода, указанных в примере, представлены в следующей таблице.

Метка Перехода	Родитель Перехода	Источник Перехода	Расположение Перехода
switch_off	/	/Power_on.Low.Heat	/Power_off

Switch_high	/Power_on	/Power_on.Low.Heat	/Power_on.High
Switch_cold	/Power_on.Low	/Power_on.Low.Heat	/Power_on.Low.Cold

Состояния

Этот раздел описывает основные объекты Stateflow - состояния. Состояния представляют режимы реактивной системы. Рассмотрим следующие вопросы для получения информации о состояниях и их свойствах:

- Что такое состояние? - описывает состояния как режимы, которые могут быть активны или неактивны.
- Декомпозиция состояний - показывает, как состояния могут быть последовательны или параллельны по отношению друг к другу в процессе выполнения системы.
- Нотации меток состояний - показывает, как определяется имя состояния и его действия посредством меток.

Что такое состояние?

Состояния описывают режимы реактивных структур Stateflow. Состояния в структурах Stateflow представлены этими режимами.

Иерархия состояний

Состояния могут быть надсостояниями, подсостояниями и просто состояниями. Состояние называется надсостоянием, если включает в себя другие состояния, называемые подсостояниями. Состояние называется подсостоянием, если оно находится внутри другого состояния. Если состояние не является ни надсостоянием, ни подсостоянием, то это просто состояние. Каждое состояние - это часть иерархии. В диаграмме Stateflow, состоящей из одного состояния, родителем этого состояния является сама диаграмма Stateflow. Состояние также имеет память, которая применима к его уровню иерархии диаграммы Stateflow. Состояния могут иметь действия, которые выполняются в последовательности, которая оперделяется их типом. Типы действий: действия при входе (entry), действия во время активности (during), действия при выходе (exit) и действия при наступлении события с именем*event_name* (on *event_name*).

Активные и неактивные состояния

Когда состояние активно, диаграмма переходит в этот режим. Когда состояние неактивно, диаграмма не в этом режиме. Активность и неактивность состояний диаграммы динамически изменяется, базируясь на событиях и условиях. Появление событий управляет выполнение диаграммы Stateflow посредством активности и неактивности состояний. На любой стадии выполнения диаграмма Stateflow есть комбинация активных и неактиных состояний.

Декомпозиция состояний

Каждое состояние и диаграмма имеет декомпозицию, которая определяет, какие виды подсостояний оно может включать в себя. Все

подсостояния должны быть такого же типа, как декомпозиция надсостояния. Декомпозиция может быть последовательной (ИЛИ) или параллельной (И). Эти типы декомпозиций описаны в следующих подразделах:

- Последовательная декомпозиция состояний (ИЛИ)
- Параллельная декомпозиция состояний (И)

Последовательная декомпозиция состояний (ИЛИ)

Если подсостояния имеют сплошные границы, декомпозиция состояний является последовательной (ИЛИ). Последовательная (ИЛИ) декомпозиция используется для описания режимов системы, которые взаимно последовательны. Когда состояние имеет последовательную (ИЛИ) декомпозицию, в любой момент времени может быть активно только одно состояние. Потомками родителей с последовательной (ИЛИ) декомпозицией являются состояния ИЛИ. В следующем примере активным может быть или состояние A, или состояние B. Если состояние A активно, то в любой момент времени может быть активным одно из состояний A1 и A2 (рисунок 1.19).

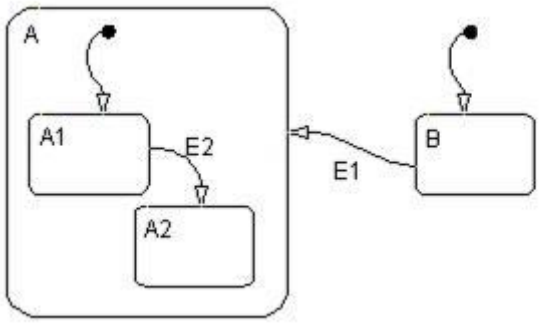


Рисунок 1.19 – Последовательная декомпозиция состояний ИЛИ

Нотации меток состояний

Метки состояний отображаются в верхнем левом углу рамки состояния в следующем основном формате:

name/
entry:entry action
during:during action
exit:exit action
on event_name:on event_name action

Метки состояний начинаются с имени состояния. Кроме того, метка состояния может иметь символ / и одно или несколько ключевых слов. Ключевые слова определяют различные типы действий, связанных с состоянием. Эти ключевые слова приведены в таблице (сокращенная форма представлена полужирным шрифтом):

Приставка	Тип действия
Entry	Действия при входе в состояние.

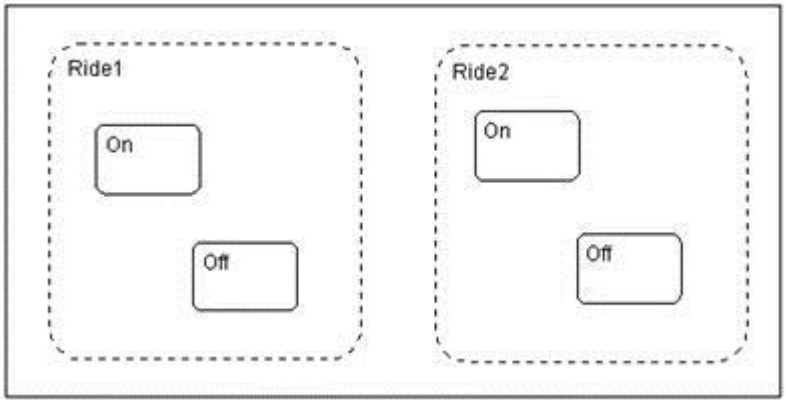
During	Действия во время активности состояния.
Exit	Действия при выходе из состояния.
On event_name	Действия при наступлении события event_name.

После ввода метки, которая определяет тип действия, Вы должны указать, какие именно действия будут выполняться. **Составные действия для каждого типа разделяются одним из символов: возврат на начало строки; точка с запятой; запятая.**

Вы можете определить составные действия типа on event_name для различных событий.

Каждое ключевое слово опционально и независимо. Вы можете не определять их вовсе либо определить одно или несколько. **После каждого ключевого слова ставится двоеточие. За именем состояния может идти / (слэш). Если вы вводите имя и слэш / непосредственно перед действиями, действия интерпретируются как действия при входе в состояние**. Сокращенная форма нужна, когда вы определяете только действия при входе в состояние. Действия определяются в метках состояний, составляющих часть языка действий Stateflow.

Правильное имя состояния может состоять из букв, цифр и может включать знак подчеркивания (_), например, Transmission или Green_on. Использование иерархии придает гибкость наименованию состояний. Имя, которое вы вводите в качестве части метки, должно быть уникальным по отношению к именам предков в его иерархии. Имя сохраняется в словаре данных в виде полного имени. Несколько состояний могут иметь одинаковое имя, если их полные имена в словаре данных уникальны. Иначе при грамматическом разборе выявится ошибка. Следующий пример показывает, как иерархия поддерживает уникальность имен состояний (рисунок 1.22).



иерархия поддерживает уникальность имен состояний состояний

Каждое из этих состояний имеет уникальное имя в иерархии диаграммы Stateflow. Хотя имена, являющееся частью метки, не уникальны, когда иерархия будет добавлена в имя в словаре данных, результат будет

уникальным. Полные имена этих состояний, как они выглядят в словаре данных, показаны ниже:

- Ride1.On
- Ride1.Off
- Ride2.On
- Ride2.Off

Пример метки состояния

Следующий пример демонстрирует компоненты метки состояния (рисунок 1.23).

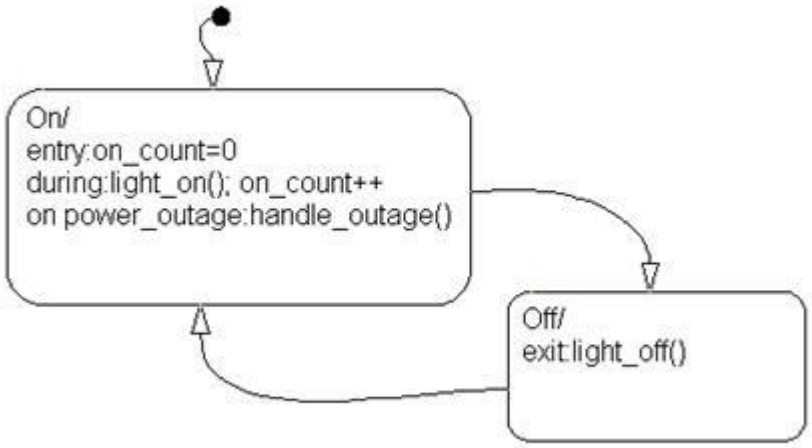


Рисунок 1.23 – Пример метки состояний

В этом примере имена состояний - On и Off. Имя состояния формирует первую часть метки состояния. Далее указаны следующие действия:

Entry Action (Действие при входе в состояние). Состояние On имеет действие при входе on_count=0. Это значит, что значение on_count устанавливается равным 0, когда выполняется вход в состояние On.

During Action (Действие во время активности состояния). Состояние On имеет два During действия: light_on() и on_count++. Эти действия выполняются, когда выполняются текущие действия состояния On.

Exit Action (Действие при выходе из состояния). Состояние Off имеет действие при выходе light_off(). Это действие выполняется, когда выполняется выход из состояния Off.

On Event_Name Action (Действие при наступлении события Event_Name). Состояние On имеет такое действие. Когда событие power_outage произойдет, действие handle_outage() выполнится.

Переходы

Реактивная система меняет одно состояние на другое при помощи объекта, называемого переходом.

Что такое переход?

Переход - это линия со стрелкой, соединяющая один графический объект с другим. В большинстве случаев переход представляет скачок

системы из одного режима (состояния) в другой. Переход соединяет объект-источник с объектом-адресатом. Объект-источник - это место, где переход начинается, объект-адресат - это место, где переход заканчивается. Вот пример перехода из состояния-источника On к состоянию-адресату Off (рисунок 1.24).

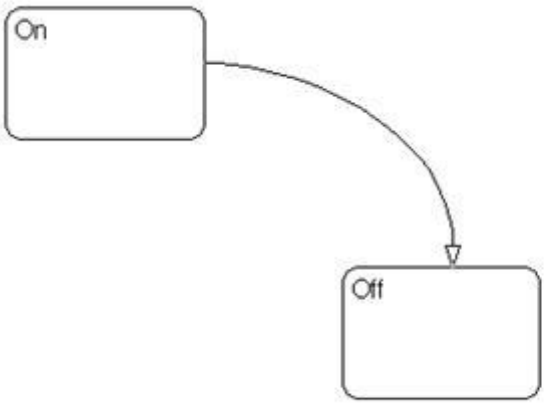


Рисунок 1.24 – Переход между состояниями

Соединения делят переходы на сегменты. В этом случае переход состоит из сегментов перехода и в процессе определения действительности всего перехода каждый сегмент анализируется по отдельности. В следующем примере имеется два сегментированных перехода: один от состояния On к состоянию Off, а другой - от состояния On обратно к состоянию On (рисунок 1.25).

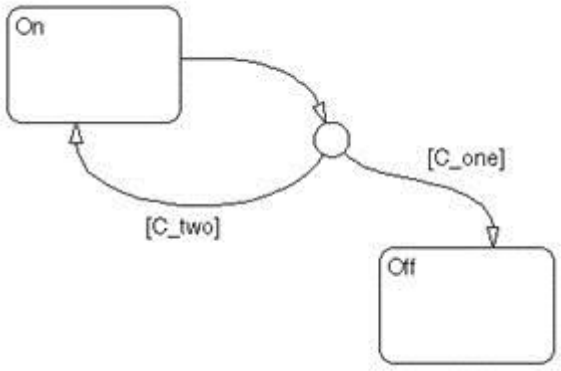


Рисунок 1.25 – Переход с действием

Имеется еще один тип перехода - безусловный переход. Безусловный переход - это специальный тип перехода, у которого нет объекта источника.

Нотации меток переходов

Переходы характеризуются метками. Метка может включать в себя имя события, условие, действие условия и/или действие перехода. Первоначально переходы помечаются символом (?). Метки перехода имеют следующий основной формат: **event[condition]{condition_action}/transition_action.**

Вы заполняете части event, condition, condition_action и transition_action, как показано в примере. Любая часть метки может отсутствовать. Действия, которые вы определяете в метках, составляют часть языка действий Stateflow.

Пример метки перехода

Использование метки перехода иллюстрируется следующим примером (рисунок 1.26).

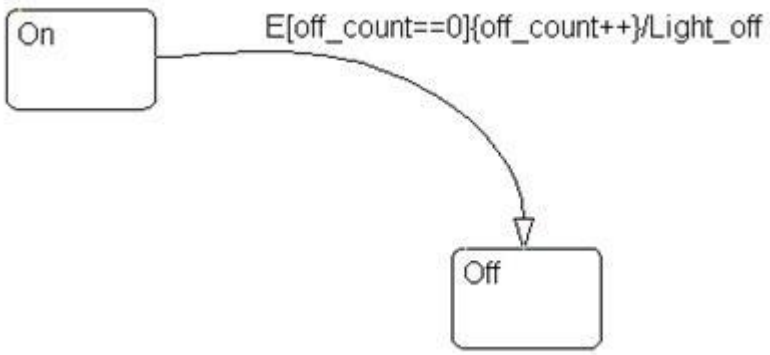


Рисунок 1.26 – Метки перехода

Переход происходит при наступлении события, но с учетом истинности условия, если оно определено. Определение имени события необязательно. Если имя события не указано, то переход произойдет при наступлении любого события. **Составные события определяются посредством использования логического оператора ИЛИ (|).** В этом примере при наступлении события E происходит переход из состояния On в состояние Off, если при этом будет истинно условие [off_count==0].

Условия - это булевы выражения, которые должны быть истинны для осуществления перехода. Условия заключаются в квадратные скобки ([]). В этом примере условие [off_count==0] должно быть истинным для того, чтобы произошло действие условия и переход стал возможен.

Действия условий следуют за условиями и заключаются в фигурные скобки ({}). Они выполняются тогда, когда условие становится истинным, но перед тем, как переход осуществится. Если ни одно условие не определено, подразумеваемое условие принимается за истинное и действие выполняется. В этом примере если условие [off_count==0] истинно, то действие off_count++ немедленно выполняется.

Действия перехода выполняется после того, как переход стал возможен и при истинности условия, если оно определено. Если переход состоит из сегментов, действие перехода выполняется только когда становится возможен полный путь перехода. Действия перехода обозначаются символом (\). В данном примере, если условие [off_count==0] истинно и состояния Off достигнуто, осуществляется действие перехода Light_off.

Когда переход возможен

В большинстве случаев переход возможен, когда состояние-источник активно и метка перехода действительна. Переходы по умолчанию отличаются тем, что не имеют состояния-источника. Переход по умолчанию к подсостоянию возможен, когда есть переход к надсостоянию или оно активно. Следующие критерии применим и к переходам, и к безусловным переходам.

Метка перехода	Переход стал возможен, если…
Событие	Событие произошло
Событие и условие	Событие произошло и условие истинно
Условие	Любое событие произошло и условие истинно
Действие	Любое событие произошло
Не определена	Любое событие произошло

Типы переходов

Нотации Stateflow поддерживают самые различные типы переходов, которые рассмотрены в следующих разделах.

Переходы к последовательным (ИЛИ) состояниям

Рассмотрим простой пример (рисунок 1.27).

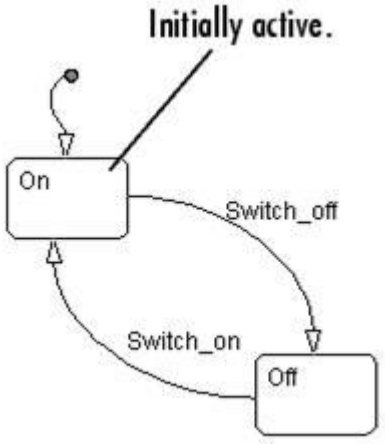


Рисунок 1.27 – Пример переходов к последовательным (ИЛИ) состояниям

Переход от On к Off действителен, когда состояние On активно и событие Switch_off произошло. Переход от Off к On действителен, когда состояние Off активно и событие Switch_on произошло.

Переходы к подключаемым соединениям

Этот пример перехода к подключаемому соединению (к точке принятия решения) и из подключаемого соединения (рисунок 1.28).

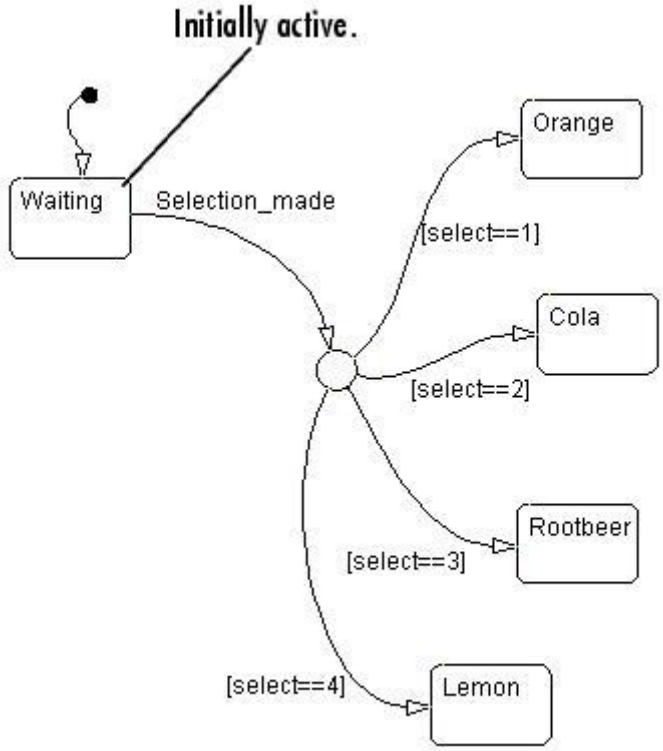


Рисунок 1.28 – Пример переходов к подключенным состояниям

Это Stateflow-диаграмма автомата для продажи газированной воды. Stateflow-диаграмма вызывается, когда внешнее событие Selection_made (Выбор_сделан) произошло. В начальный момент активно состояние Waiting (Ожидание). Состояние Waiting - это общий источник для всех состояний. Когда событие Selection_made происходит, переход из состояния Waiting к одному из остальных четырех состояний базируется на значении переменной select (выбор). Так, если значение переменной select равно 2, диаграмма переходит в состояние Cola. От состояния Waiting к подключаемому соединению ведет один переход, от подключаемого соединения к четырем возможным состояниям ведут четыре различных перехода.

Переходы к последовательным (ИЛИ) надсостояниям

Этот пример показывает переходы к последовательным (ИЛИ) надсостояниям и из них, а также использование безусловного перехода (рисунок 1.29).

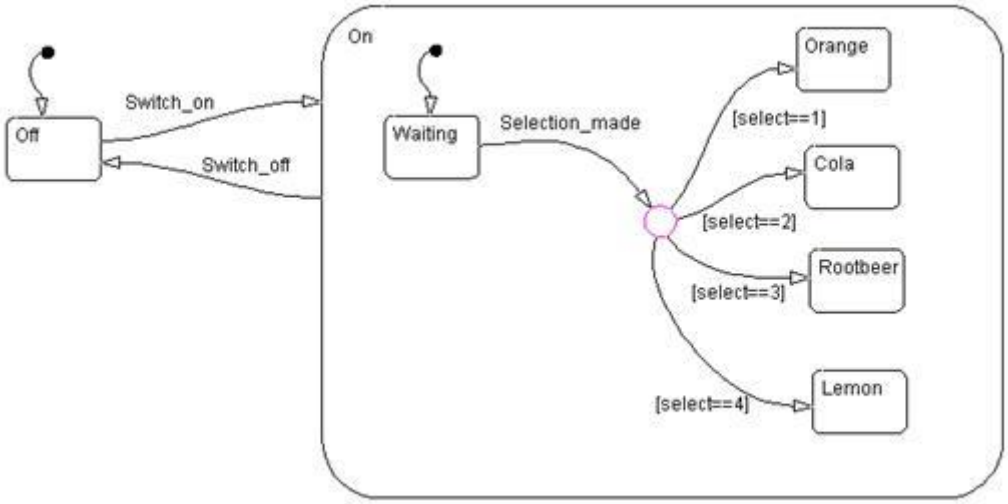


Рисунок 1.29 – Пример переходов к последовательным (ИЛИ) надсостояниям

Это - продолжение двух предыдущих примеров. Сейчас On (автомат включен) - это надсостояние, которое включает состояние Waiting и состояния, соответствующие выбранным напиткам. Переход от Off к On действителен, когда состояние Off активно и событие Switch_on произошло. Однако теперь, когда On является надсостоянием, мы имеем дело с переходом к надсостоянию. Чтобы переход к надсостоянию был действителен, подсостояние-адресат также должно быть определено. Такое подсостояние неявно определено посредством безусловного перехода к Waiting. Приведенная в этом примере нотация определяет, что переход из состояния Off будет сделан обязательно к состоянию On.Waiting. Состояние Off (автомат выключен) не является надсостоянием. Переход от On к Off действителен, когда состояние On активно и событие Switch_off произошло. Другими словами, когда событие Switch_off произошло, переход к состоянию Off выполнится невзирая на то, какое из подсостояний On активно. Это упрощает наблюдения за переходами внутри Stateflow-диаграмм.

Переходы к подсостояниям

Следующий пример показывает переходы к последовательным (ИЛИ) подсостояниям (рисунок 1.30).

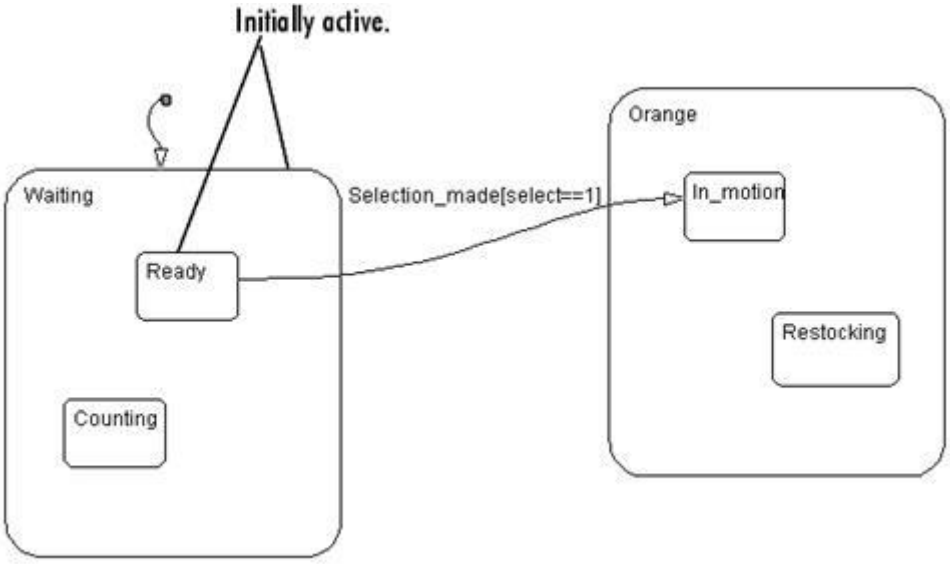


Рисунок 1.30 – Переходы к подсостояниям

Эта диаграмма Stateflow показывает переход от одного ИЛИ-подсостояния к другому ИЛИ-подсостоянию: переход Waiting.Ready к Orange. Переход к состоянию In_motion действителен, когда состояние Waiting.Ready активно, событие Selection_made произошло и переменная select равна 1. Этот переход определяет явный выход из состояния Waiting.Ready и неявный выход из надсостояния Waiting. На стороне адресата этот переход определяет неявный вход в надсостояние Orange и явный вход в подсостояние Orange.In_motion.

Циклические переходы

Сегмент перехода от состояния к подключаемому соединению, а от него назад к тому же самому состоянию, называется циклическим переходом. Вот пример циклического перехода (рисунок 1.31).

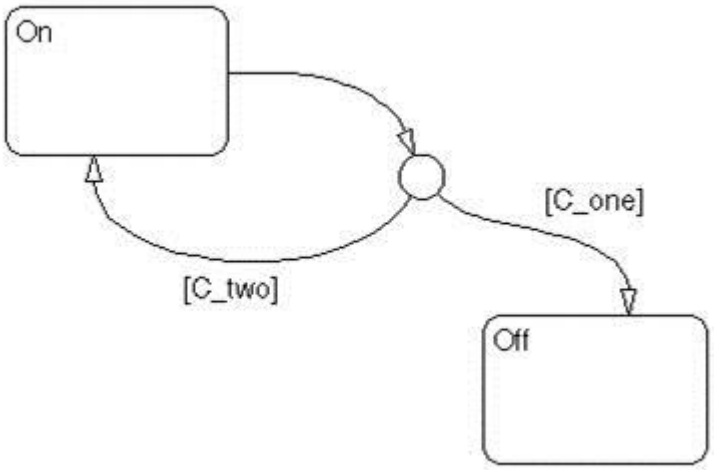


Рисунок 1.31 – Циклические переходы

Внутренние переходы

Внутренние переходы - это переходы, которые не покидают состояния-источника. Использование внутренних переходов значительно упрощает диаграмму Stateflow. В наибольшей степени это проявляется, когда внутренние переходы определены для надсостояний с последовательной декомпозицией.

Рассмотрим Stateflow-диаграмму, которую можно упростить, если применить внутренние переходы (рисунок 1.32).

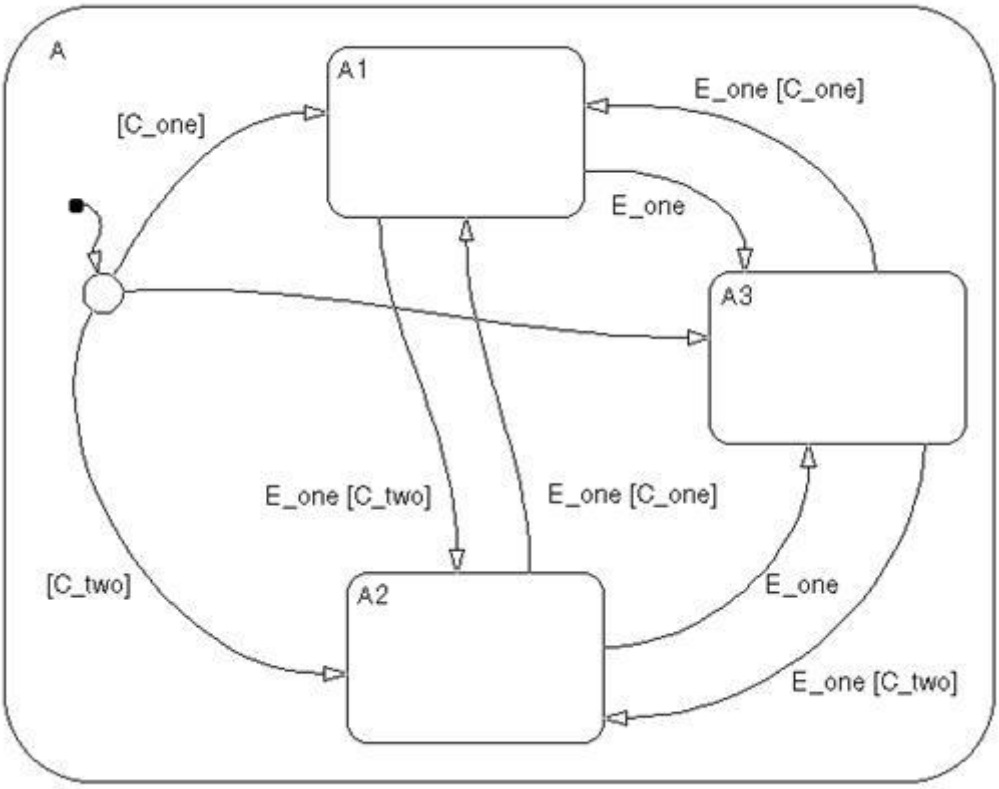


Рисунок 1.32 – Внутренние переходы

Пусть произошло какое-либо событие. При этом обновляется Stateflow-диаграмма. Происходит безусловный переход к подключаемому соединению. Переход-адресат определяется после анализа условий [C_one] и [C_two]. Если [C_one] истинно, осуществится переход к A1. Если [C_two] истинно, осуществится переход к A2. Если ни условие [C_one], ни условие [C_two] не является истинным, осуществится переход к A3. Переходы между A1, A2, и A3 определены метками E_one, [C_one], и [C_two].

А вот как выглядит аналогичная Stateflow-диаграмма после использования внутреннего перехода для подключаемых соединений (рисунок 1.33).

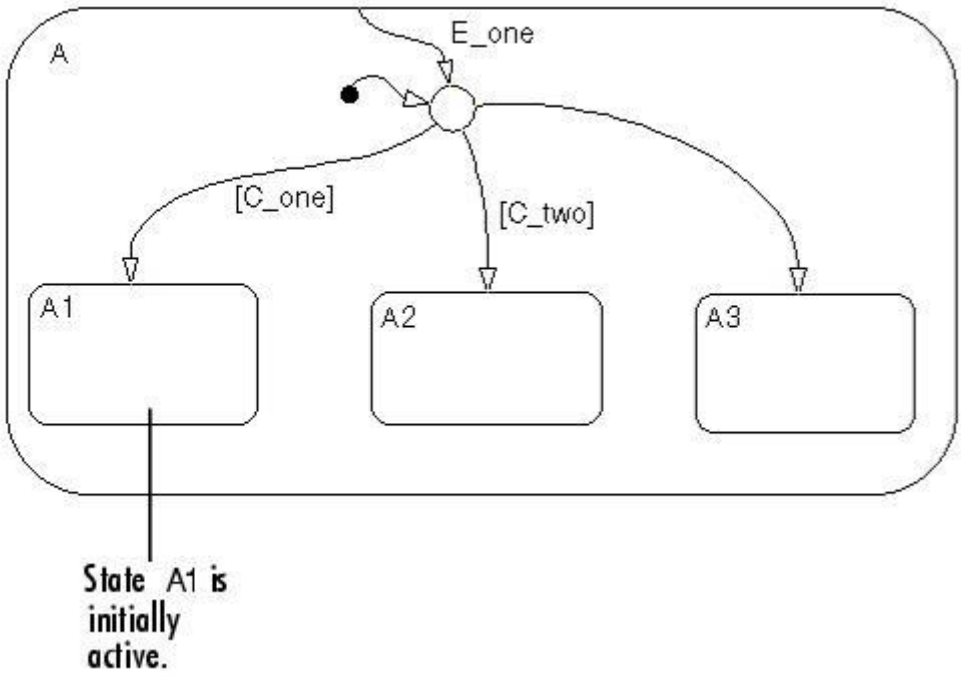


Рисунок 1.33 – Stateflow-диаграмма после использования внутреннего перехода для подключаемых соединений

Пусть произошло какое-либо событие. При этом обновляется Stateflow-диаграмма. Происходит безусловный переход к подключаемому соединению. Переход-адресат определяется анализом условий [C_one] и [C_two]. Диаграмма упрощается использованием одного внутреннего перехода вместо большого числа переходов, как в данном примере. Если состояние A уже активно, внутренний переход используется для определения того, какое из подсостояний A активно. Когда событие E_one произошло, внутренний переход потенциально действителен. Если [C_one] истинно, переход к A1 действителен. Если [C_two] истинно, переход к A2 действителен. Если ни условие [C_one], ни условие [C_two] не является истинным, произойдет переход к A3. Это решение проще, чем предыдущее.

Использование внутреннего перехода для соединения с памятью

Этот пример показывает внутренний переход к соединению с памятью (рисунок 1.34).

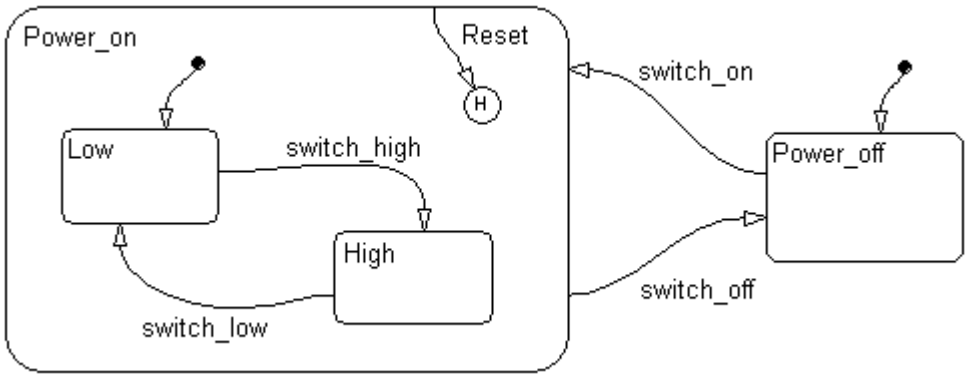


Рисунок 1.34 – Использование внутреннего перехода для соединения с памятью

Пусть состояние Power_on.High изначально активно. Когда событие Reset происходит, внутренний переход к соединению с памятью действителен. Так как внутренний переход действителен, выходим из текущего состояния Power_on.High. Когда происходит внутренний переход к соединению с памятью, предыдущее активное состояние, Power_on.High, вновь становится активным (повторный вход). Аналогично, если состояние Power_on.Low было активным, то в результате произойдет выход из Power_on.Low и повторный вход в него. Внутренний переход в этом примере эквивалентен двум циклическим переходам для Power_on.Low и Power_on.High.

Безусловные переходы

Вы используете безусловные переходы, чтобы сообщить Stateflow, какое именно (одно из нескольких возможных) состояние должно стать активным, когда активизируется состояние или диаграмма, имеющие подсостояния.

Что такое безусловный переход?

Безусловные переходы преимущественно используются для определения, какое последовательное (ИЛИ) состояние должно стать активным, когда есть неоднозначность между двумя или более ИЛИ-подсостояниями. Безусловные переходы имеют объект-адресат, но у них нет объекта-источника. Безусловный переход определяет, какое из подсостояний надсостояния с последовательной (ИЛИ) декомпозицией должно стать активным в отсутствие любой другой информации, такой, как соединения с памятью. Безусловные переходы также используются, чтобы определить, что переход должен быть безусловно выполнен.

Чтобы нарисовать безусловный переход, щелкните по кнопке безусловного перехода на панели инструментов, а затем - по свободной зоне в рабочей области рядом с состоянием или соединением, которое будет адресатом безусловного перехода. Перемещайте мышь к объекту-адресату до присоединения к нему безусловного перехода. В некоторых случаях полезно

создать метку безусловного перехода. Одна из наиболее частых ошибок программирования Stateflow-диаграмм - это создание многочисленных последовательных (ИЛИ) состояний без использования безусловных переходов. В отсутствии безусловного перехода не известно состояние, которое становится активным по умолчанию. Заметим, что эта ошибка выявляется, когда модель запускается с использованием Отладчика (Debugger) с включенной опцией **State Inconsistencies** (Непротиворечивость Состояний).

В таблице показывается кнопка значка и кратко описывает безусловный переход.

Имя	Кнопка значка	Описание
Безусловный переход		Используется для указания, при достижении этого уровня иерархии, какой из объектов становится активным по умолчанию.

Метки безусловных переходов

В некоторых случаях есть необходимость в метке безусловного перехода. Можно пометить безусловный переход аналогично тому, как помечаются другие переходы. Когда Вы помечаете безусловные переходы, позаботьтесь о том, чтобы всегда минимум один безусловный переход был действителен. Иначе диаграмма Stateflow может перейти в неопределенное состояние

Примеры безусловных переходов

Следующие примеры показывают использование безусловных переходов в Stateflow-диаграммах.

Примеры безусловных переходов для состояний

Этот пример показывает, как использовать безусловные переходы (рисунок 1.35).

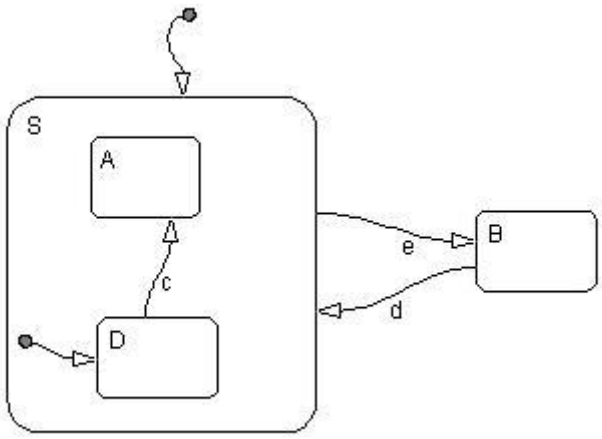


Рисунок 1.35 – Пример безусловных переходов для состояний

Когда Stateflow-диаграмма первый раз обновляется, должен решиться вопрос, какое из состояний - S или B активировать, т. к. они являются последовательными (ИЛИ) состояниями. Ответ дается безусловным переходом к надсостоянию S. Так как у этого безусловного перехода нет условий, он происходит. Состояние S, которое теперь активно, имеет два подсостояния, A и D. Какое подсостояние становится активным? Только одно из них, может быть активным, потому что они - последовательные (ИЛИ) состояния. Это определяется безусловным переходом к подсостоянию D. Так как у безусловного перехода нет условий, он происходит.

Пусть теперь Stateflow-диаграмма обновляется событием d при активном состоянии B. Переход от состояния B к состоянию S осуществляется. Когда система входит в состояние S, она входит в подсостояние D, потому что определен безусловным переходом.

Безусловные переходы требуются для выполнения Stateflow-диаграмм. Без безусловного перехода к состоянию S при обновлении Stateflow-диаграммы, ни одно из состояний не становится активным. Вы можете зафиксировать эту ситуацию в процессе отладки.

Примеры безусловных переходов для соединений

Этот пример показывает безусловный переход к подключаемому соединению (рисунок 1.36).

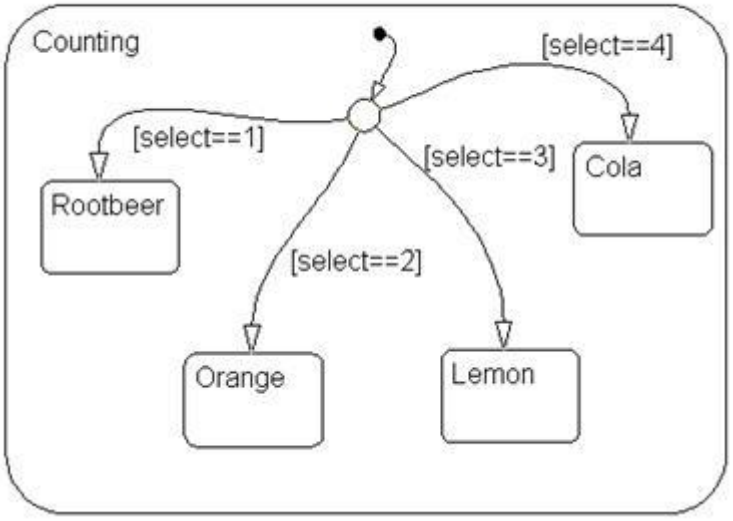


Рисунок 1.36 – Пример безусловных переходов для соединений

В этом примере безусловный переход к подключаемому соединению показывает, что после входа в состояние Counting, подсостояние-адресат определено условиями на сегментах перехода.

Пример безусловного перехода с меткой

Следующий пример показывает, как пометить безусловный переход (рисунок 1.37).

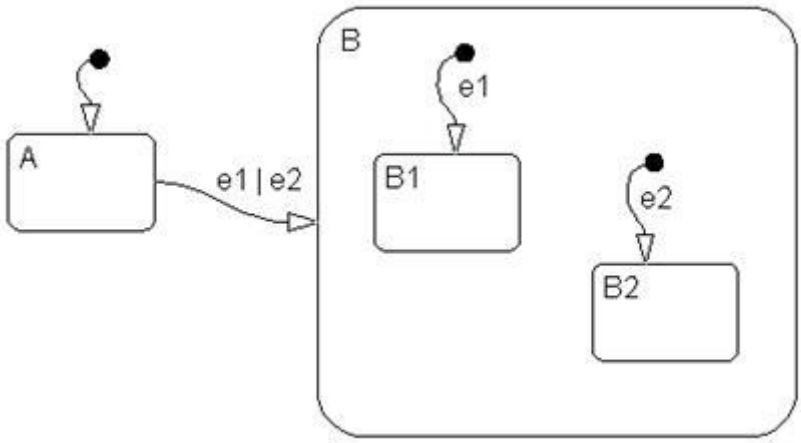


Рисунок 1.37 – Пример безусловного перехода с меткой

Если состояние A изначально активно и произошло какое-то из двух событий e1 или e2, переход от состояния A к надсостоянию B действителен. Надсостояние B1 и B2 имеют безусловные переходы. Безусловные переходы помечаются определением событий, которые переключают переход. Если событие e1 произошло, переход от A к B1 действителен. Если событие e2 произошло, то переход от A к B2 действителен.

Подключаемые соединения

Подключаемые соединения представляют точки, в которых принимаются решения относительно выбора между альтернативными путями одного перехода.

Подключаемые соединения позволяют представить различные возможные пути для одного перехода. Подключаемое соединения используются в следующих случаях:

- Для представления в виде фрагментов Stateflow-диаграмм конструкции if-then-else посредством определения условий для некоторых или всех исходящих из подключаемого соединения переходов.
- В случае цикличных переходов, возвращающихся к состоянию-адресату в том случае, если ни один из остальных исходящих переход не действителен.
- При построении различных вариантов конструкции for посредством возврата перехода из подключаемого соединения в него же.
- Для организации переходов из общего источника к множеству адресатов.
- Для организации переходов от множества источников к общему адресату.
- Для организации переходов от источника к адресату, если они основываются на общих событиях.

Однако есть и ограничения. Так, событие не может инициировать переход из подключаемого соединения в состояние-адресат.

Блок-схемы с подключаемыми соединениями

Подключаемые соединения могут использоваться также для представления структуры алгоритмов (например, конструкций for или if-then-else) в виде блок-схем, т.е. без использования состояний. За счет снижения в Stateflow-диаграмме числа состояний вырабатывается более эффективный код, который оптимизирует использование памяти. При этом применяются следующие комбинации средств:

- Переходы к подключаемым соединениям и из них.
- Циклы для подключаемых соединений.
- Внутренние соединения для подключаемых соединений.

Диаграммы в виде состояний и переходов между ними и диаграммы в виде блок-схем могут сосуществовать в пределах одной Stateflow-диаграммы. Ключ к успешному представлению диаграмм - в использовании меток переходов, как показано в следующих примерах.

Пример подключаемого соединения с заданием условий перехода

На примере (рисунок 1.38) слева состояние A активно, когда произошло событие e, и переход от состояния A к одному из состояний D, E или F имеет место, если выполнено одно из условий [c1], [c2] или [c3]. В эквивалентном представлении справа переход от состояния-источника к подключаемому соединению помечен событием. Переходы от подключаемого соединения к состояням-адресатам помечены условиями. Если состояние A активно, когда происходит событие e, то вначале осуществляется переход от состояния A к подключаемому соединению. Переход от подключаемого соединения к состоянию-адресату происходит, базируясь на истинности условий [c1], [c2], или [c3]. Если ни одно из них не истинно, переход не происходит и состояние A остается активным.

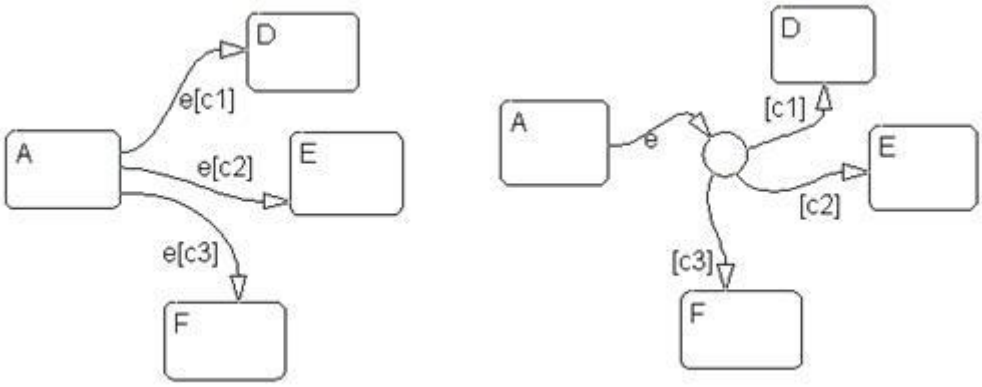


Рисунок 1.38 – Пример подключаемого соединения с заданием условий перехода

Пример подключаемого соединения с одним переходом, не ограниченным условиями

Переход от A к B действителен, когда состояние A активно, событие E_one произошло, и [C_one] истинно (рисунок 1.39). Переход от A к C

действителен, когда A активно, событие E_one произошло, и [C_two] истинно. Иначе, если A активно и событие E_one произошло, то действителен переход от A к D. Если вы не определили четко условие [C_three], то подразумевается, что условие перехода от A к D - ложность [C_one] и [C_two].

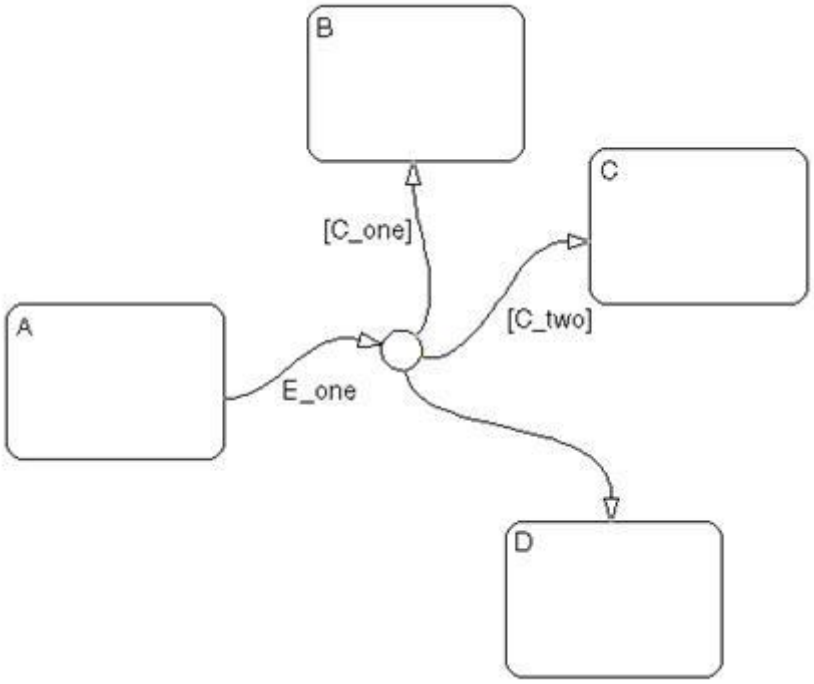


Рисунок 1.39 - Пример подключаемого соединения с одним переходом, не ограниченным условиями

Подключаемое соединение - пример цикла

Пусть событие перехода произошло, но условие перехода ложно (рисунок 1.40). Переход в новое состояние не может произойти, но какое-то действие должно быть произведено. Вы можете представить эту ситуацию с использованием подключаемого соединения или циклического перехода (перехода от состояния к нему же).

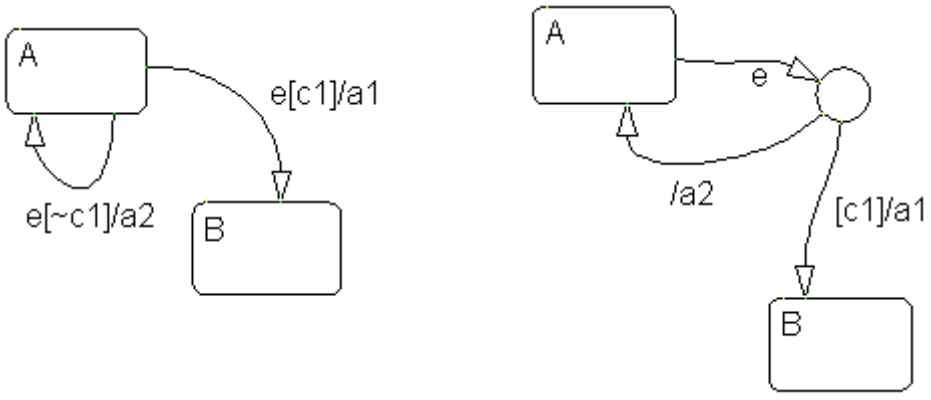


Рисунок 1.40 - Подключаемое соединение - пример цикла

На левом рисунке, если состояние A активно, событие e произошло и условие [c1] истинно, происходит переход от A к B и выполняется действие a1. Переход от состояния A к состоянию A действителен, если произошло событие e и условие [c1] ложно. В этом цикличном переходе осуществляется выход и повторяется вход в A, а также выполняется действие a2. В эквивалентном представлении справа, в силу использования подключаемых соединений отпадает необходимость в явном задании условия [~c1].

Пример подключаемого соединения и цикла for

Этот пример (рисунок 1.41) показывает комбинацию нотации в виде блок-схемы и нотации в форме состояний и переходов. Цикличные переходы могут быть использованы в подключаемых соединениях для представления циклов. Пусть в состоянии A произошло событие E. Переход от состояния к A к состоянию B действителен, если условие на пути перехода истинны. Первый сегмент перехода не имеет условия, но имеет действие условия. Действие условия {i=0} выполняется. Условие циклического перехода [i<10] становится истинным и действия условия {i++;func1()} выполняются. Действия условия выполняются, пока условие [i<10] не станет ложным (такая диаграмма эффективно выполняет цикл for для значений i от 0 до 9, вызывая каждый раз функцию func1()). Затем выполняется выход из цикла и переход в состояние B.

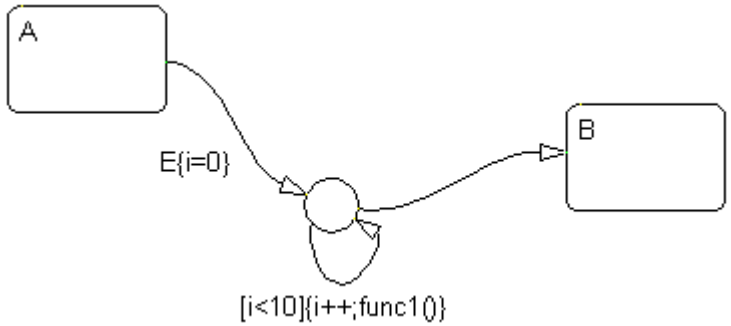


Рисунок 1.41 - Пример подключаемого соединения и цикла for

Пример диаграммы в виде блок-схемы

Это - пример реального использования комбинации нотации в виде блок-схемы и нотации в форме состояний и переходов (рисунок 1.42). На рисунке показана Stateflow-диаграмма модели 8-битного аналого-цифрового преобразователя (АЦП). Рассмотрим случай, когда состояние Sensor.Low активно и событие UPDATE произошло. Внутренний переход от Sensor к подключаемому соединению действителен. Следующий сегмент перехода имеет действие условия {start_adc()}, которое инициирует чтение с АЦП. Цикл на втором подключаемом соединении повторяется, пока истинно условие [adc_busy()]. Этот циклический переход используется для представления задержки, нужной для чтения с АЦП. Задержка может быть представлена и в виде отдельного состояния, но при этом был бы получен менее эффективный код. Действие условия следующего сегмента перехода

{sensorValue=read_adc()} устанавливает новое значение, считываемое с АЦП в переменной sensorValue. Заключительный сегмент перехода определяется значением переменной sensorValue. Если [sensorValue <100] истинно, то адресат - это состояние Sensor.Low. Если [sensorValue >200] истинно, то адресат - это состояние Sensor.High. Иначе адресат - это состояние Sensor.Normal.

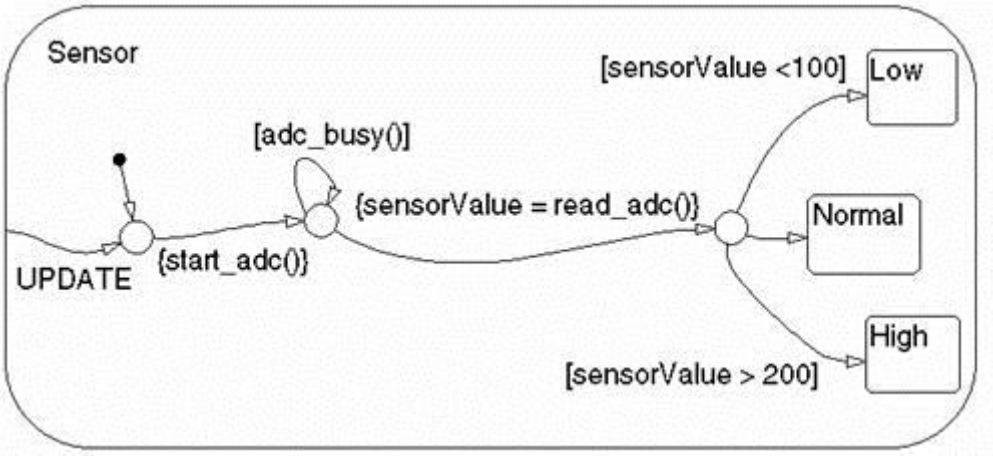


Рисунок 1.42 -Пример диаграммы в виде блок-схемы

Пример подключаемого соединения при одном источнике и нескольких адресатах

В переходах от A к B и от A к C участвует общее состояние-источник A. Альтернативное представление использует одну стрелку от A к подключаемому соединению и множество стрелок, помеченных событиями, от соединения к состояниям назначения B и C (рисунок 1.43).

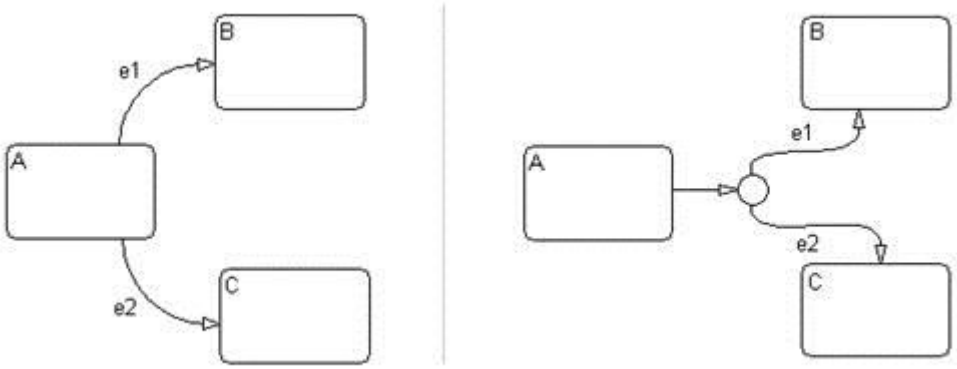


Рисунок 1.43 - Пример подключаемого соединения при одном источнике и нескольких адресатах

Пример подключаемого соединения с общим событием

Предположим, что когда происходит событие e1, система, в независимости от того, в каком из двух состояний (A или B) она находится, переходит в состояние C. Другими словами, переходы от A к C и от B к C инициируются одним и тем же событием e1. И состояние-адресат и событие

являются общими для двух переходов. Есть три способа, чтобы представить это:

- Нарисовать переход от A к B и от A к C, пометить каждый событием e1.
- Поместив A и B в одно надсостояние S, нарисовать один переход от S к C, пометив его событием e1.
- Нарисовав переходы от A и B к подключаемому соединению, и один переход от него к C, пометив его событием e1.

На диаграмме показаны первый и третий варианты (рисунок 1.44).

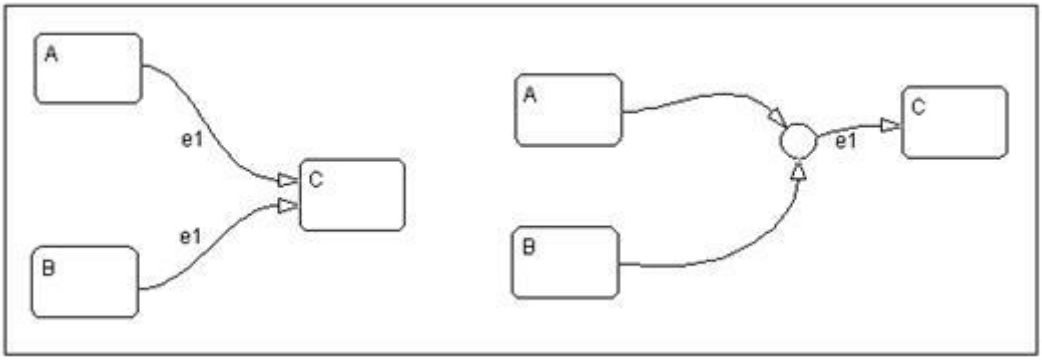


Рисунок 1.44 - Пример подключаемого соединения при одном источнике и нескольких адресатах

Соединение с памятью

Соединение с памятью запоминает предыдущее активное состояние.

Что такое соединения с памятью?

Соединения с памятью используются, чтобы отметить в Stateflow-диаграмме объекты, где решения принимаются не на основе текущей информации, а опираясь на предысторию. Эта предыстория связана с активностью состояния. Размещение соединения с памятью в надсостоянии показывает, что информация об активности состояния в прошлом используется при определении состояния, которое станет активным в настоящий момент. Соединения с памятью применяются только к тому уровню иерархии, на котором находятся.

Пример использования соединения с памятью

В следующем примере используется соединение с памятью (рисунок 1.45).

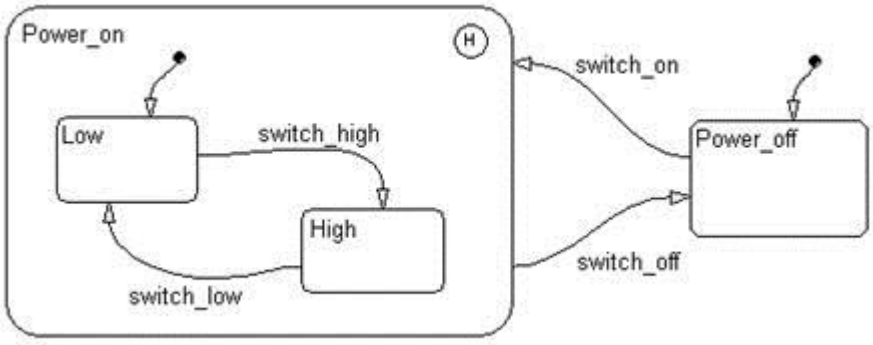


Рисунок 1.45 - Пример использования соединения с памятью

Надсостояние Power_on имеет соединение с памятью и два подсостояния. Если состояние Power_off активно и событие switch_on произошло, система может попасть либо в состояние Power_on.Low, либо в состояние Power_on.High. Во время первого входа в надсостояние Power_on осуществляется вход в состояние Power_on.Low, потому что к нему ведет безусловный переход. В некоторый следующий момент, когда состояние Power_on.High активно и происходит событие switch_off, система покидает надсостояние Power_on и состояние Power_off становится активным. Пусть затем происходит событие switch_on. Так как состояние Power_on.High было активным последним, оно снова становится активным. За исключением первого раза, при активности состояния Power_on выбор между Power_on.Low или Power_on.High определяется предысторией.

Соединения с памятью и внутренние переходы

Создавая внутренний переход для объекта с памятью, вы показываете, что при наступлении события и/или условия должен быть выполнен выход из активного состояния, а затем повторен вход в него.

Блоки

Блоки используются для улучшения графического представления диаграммы. Помимо этого, блоки участвуют в выполнении диаграммы Stateflow. Это пример использования Stateflow объекта "блок" (рисунок 1.46).

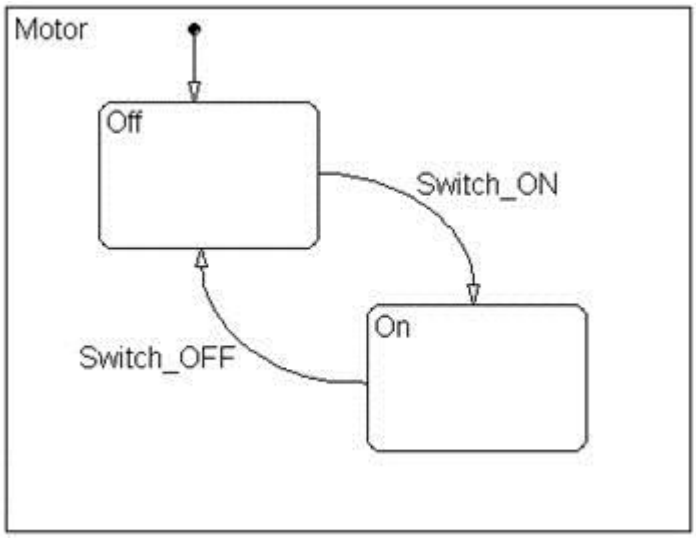


Рисунок 1.46 – Блоки

В этом примере блок, помеченный как Motor, группирует все объекты, которые необходимы для управления простым мотором. На диаграмме наряду с этим блоком могут быть и другие объекты, но теперь все элементы управления мотором выделены в отдельный объект.

Графические функции

Графические функции - это функции, которые определены графом переходов. Наличие графических функций обеспечивает удобство и повышает выразительность языка действий Stateflow. На рисунке приведен пример графической функции и Stateflow-диаграммы с переходом, который ее вызывает (рисунок 1.47).

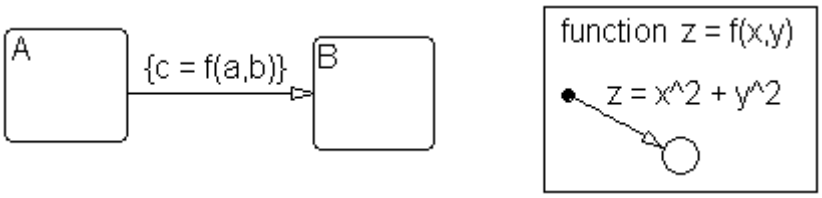


Рисунок 1.47 – графические функции

Здесь функция z = f(x,y) вызывается действием условия при переходе от состояния A к состоянию B. Функция определена с использованием символов, которые действительны только в ней самой (формальных параметров x, y и z). Функция вызывается с использованием объектов (фактических параметров c, a и b), доступных состояниям A и B и их родительским состояниям (если таковые имеются). Графические функции подобны текстовым функциям MATLAB и C-функциям. Подобно функциям C и MATLAB, графические функции могут иметь аргумент и возвращать результат. Вызываются функции в действиях переходов и состояний. В отличии от функций C и MATLAB, графические функции - это полноправные графические объекты Stateflow. Вы используете редактор Stateflow, чтобы создать их, и они изменяются в вашей модели Stateflow

вместе с диаграммами, которые их вызывают. Это делает графические функции более простыми в создании и управлении, чем текстовые функции, создание которых требует внешних инструментов и которые изменяются отдельно от модели.

Язык действий

Этот раздел знакомит с языком действий, используемым в метках переходов и состояний

Что такое язык действий?

Язык действий определяет действия, которые можно задать, если использовать нотации меток состояний и переходов. Stateflow сопоставляет действия состояниям или переходам при наличии соответствующих меток. Действия состояний и переходов делятся на несколько типов. Переходы имеют события переключения, условия, действия условий и действий переходов. Состояния могут иметь действия при входе в состояние, во время активности состояния, действия при выходе из состояния и при наступлении события с именем *event_name*.

Пример использования языка действий

Следующая диаграмма показывает примеры различных категорий действий для состояний и переходов (рисунок 1.48).

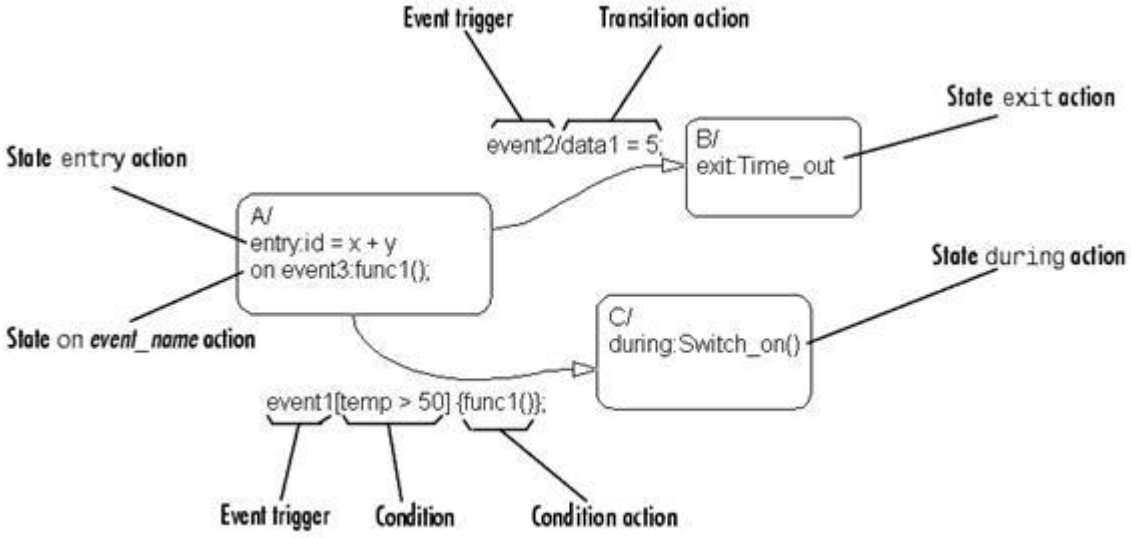


Рисунок 1.48 - Пример использования языка действий

Действия в языке действий могут приводить к наступлению событий, установлению условий, вызову функций, установлению значений переменных и операндов и т. д. Большинство из них аналогично действиям в C или MATLAB.

Нотации языка действий

Язык действий включает метки состояний и переходов, к которым применятся язык действий..

Семантика языка действий

Правила взаимодействия между различными нотациями Stateflow относятся к семантике Stateflow. Этому вопросу будет посвящена глава "Семантика Stateflow"

Определение событий.

Событие - это объект Stateflow, который переключает действия в машине состояний или ее объекты. Stateflow определяет набор событий, которые происходят, когда выполняются состояния машины. Вы можете определить типы событий, которые появляются только в течение выполнения особых состояний машины или ее объектов.

Определение событий:

1. Добавьте описание события к словарю данных Stateflow.
2. Выберите такие значения свойств событий, чтобы они отображались в заданном виде.

Добавление событий к словарю данных

Вы можете использовать другие редакторы Stateflow или Проводник, чтобы добавлять события, видимые в диаграмме. Но чтобы добавлять события, которые видимы в машине состояний или только в особом состоянии, вы должны использовать проводник Stateflow.

Использование редактора Stateflow

Чтобы использовать редактор Stateflow для добавления событий, необходимо:

1. Из меню **Add** редактора Stateflow, выбрать **Event….**
2. В появившемся подменю, выбрать индикатор события. Stateflow добавляет описание нового события к словарю данных и отображает диалоговое окно **Event.**
3. Используйте диалоговое окно **Event**, чтобы определить свойства события.

Использование проводника

Чтобы определить событие, используйте Проводник Stateflow:

1. Выберете **Explore** из меню **Tools** редактора Stateflow.

Проводник Stateflow открывает:

2. Выберите объект (машину состояний, диаграмму или состояние) в той области иерархии проводника объекта, где Вы хотите увидеть новое событие.

3. Выберите **Event** из меню проводника **Add.**

Stateflow добавляет описание нового события в словарь данных и показывает вход для нового события в окне Проводника.

4. Назначьте такие значения свойств событий, чтобы они отображались в заданном виде.

Изменение свойств события

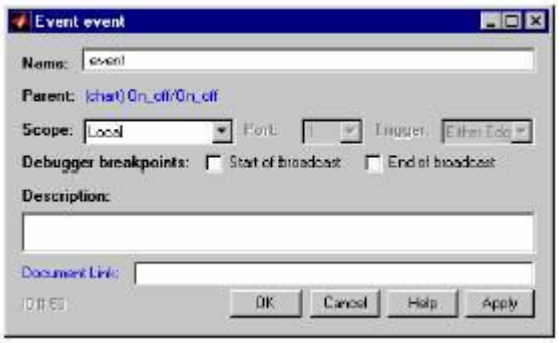
Чтобы изменить свойства события, необходимо:

1. Выбрать **Explorer** из меню **Tools** редактора Stateflow.
2. Выбрать событие в оглавлении проводника.
3. Выбрать **Properties** из меню проводника **Edit** или контекстного меню. Stateflow отобразит диалоговое окно **Event** для выбора события.
4. Редактирование диалогового окна.
5. Выберете **OK,** чтобы применить ваши изменения и закрыть диалоговое окно **Event.**

Примечание Вы также можете установить обзор события с помощью редактора соответствующего раздела в оглавлении проводника. Если вы хотите установить только это свойство, вам не нужно открывать диалоговое окно **Event.**

Диалоговое окно событие

Диалоговое окно Событие позволяет Вам определять свойства события.



Диалоговое окно показывает следующие области и параметры.

Name

Имя этого события. Имя позволяет Вам определять это событие в действиях Stateflow.

Parent

Щелчок по этой области показывает родителя этого события в редакторе Stateflow.

Родитель - объект, в котором располагается событие. Когда событие вызвано, Stateflow передает событие родителю и его потомкам. Родителем события может быть машина состояний, диаграмма или состояние. Вы определяете родителя события, когда добавляете его к словарю данных.

Scope

Индикатор события. Индикатор определяет, где происходит событие относительно его родителя.

Разделы, которые указаны ниже, описывают каждый параметр индикатора.

Local

Это событие происходит в машине состояний и становится родительским с ее помощью.

Input from Simulink

Это событие происходит в одном блоке Simulink и передается в другой. Первый блок может относиться к любому типу блоков. Второй блок должен быть блоком диаграммы.

Output to Simulink

Это событие происходит в одном блоке Simulink и передается в другой. Первый блок - блок диаграммы. Второй блок может относиться к любому типу блоков.

Export. Экспортируемое событие - событие Stateflow, которое может быть передано внешним кодом, включая автономный объект и мастерскую реального времени.

Import. Импортированное событие - это событие, определенное извне, которое может быть передано машиной состояний, вложенной во внешний код.

Примечание Если Вы копируете блок диаграммы Stateflow из одной модели Simulink в другую, все объекты в иерархии диаграммы копируются, кроме тех, которые становятся родителями с помощью машины Stateflow. Это значит, что события, которые становятся родителями с помощью машины Stateflow (локальное, импортированное, экспортированное) не скопированы в новую машину. Однако Вы можете использовать Проводник, чтобы копировать отдельные события, перетаскивая их курсором из машины в машину.

Trigger

Тип сигнала, который переключает вход или выход события.

Index

Индекс входа сигнала, который переключает это событие. Эти параметры применяется только ко входу события и появляются, когда Вы выбираете Input from Simulink как индикатор этого события.

Port

Индекс порта - это выход этого события. Это свойство применяется только ко входу события и появляется, когда Вы выбираете Output to Simulink как индикатор этого события.

Description

Описание этого события. Stateflow сохраняет содержание этой области в словаре данных.

Document Link

Щелчок в этой области показывает документацию для этого события.

Присваивание имени событию

Имя события позволяет действиям ссылаться на определенные события. Вы присваиваете имя событию, изменяя его свойства. Вы можете назначить любое имя, которое начинается с алфавитного знака, не включая пробелов.

Определение локальных событий

Локальное событие это событие, которое может происходить во всей машине состояний, но видимо только в ее родителе (и потомке родителя). Чтобы определить событие, как локальное, установите свойства его индикатора как локальные.

Определение входящих событий

Входящее событие происходит вне диаграммы и видимо только в ней. Этот тип события позволяет блокам Simulink и блокам Stateflow уведомлять особые события, которые происходят вне диаграммы. Чтобы определить событие, как входящее, установите свойства его индикатора как вход в Simulink.

Вы можете определять многократные входящие события для диаграммы. Первый раз, когда Вы определяете входящее событие для диаграммы, Stateflow добавляет переключатель порта к блоку диаграммы. Внешние блоки могут переключать входящие события диаграммы через сигнал или вектор сигналов, связанных с переключателем порта, связывая входящие события с управляющим сигналом. При определении входящих событий для диаграммы Вы должны определить, как управляющие сигналы, связанные с диаграммой, переключают входящие события.

Соединение входных событий с управляющими сигналами

Свойства индекса входящего события связывают событие с определенным элементом вектора управляющего сигнала, соединенного с переключателем порта диаграммы, который является родителем события.

Первый элемент вектора сигнала переключает входящее событие, индекс которого - 1; второй, индекс которого - 2, и так далее. Stateflow назначает 1 как индекс первого входящего события, который Вы определяете

для диаграммы, 2 как индекс второго события, и так далее. По умолчанию Вы можете изменять соединения для события, устанавливая свойства индекса события к индексу сигнала, так, чтобы переключить событие.

Входящие события происходят в порядке возрастания их индексов, когда больше чем одно такое событие происходит в течение обновления. Например, предположим, что при определении входящих событий для диаграммы Вы назначаете индексы 3, 2, и 1 к событиям названным A, B, и C соответственно. Теперь предположим, что в течение создания модели, содержащей диаграмму, события A и C происходят в особом порядке. В этом случае, порядок возникновения событий такой: сначала происходит событие C, сопровождаемое A.

Определение выходящих событий

Выходящее событие - событие, которое происходит в определенной диаграмме и видимо в определенных блоках вне диаграммы. Этот тип события позволяет диаграмме уведомлять другие блоки в модели событий, которые происходят в диаграмме. Чтобы определить событие как входящее, установите свойство его индикатора как выход в Simulink. Вы можете определить многократные выходящие события для данной диаграммы. Stateflow создает диаграмму входящего порта для каждого выходящего события, который Вы определите. Ваша модель может использовать выходящие порты, чтобы переключать выходящие события в других блоках Simulink в той же самой модели.

Соединение выходящего события с выходящим портом

Свойства порта выходящего события связывает событие с выходящим портом на блоке диаграммы, который является его родителем. Свойства определяют положение порта относительно других портов события на блоке диаграммы. Порты события появляются ниже портов данных, справа от блока диаграммы. Stateflow нумерует порты последовательно сверху вниз, начиная с 1. Stateflow назначает порт 1 первому выходящему событию, для которого Вы определяете диаграмму, порт 2 - второму, и так далее. По умолчанию Вы можете изменять порт назначения события, перезагружая свойства порта или выбирая выходящее событие в Проводнике.

Экспорт событий

Stateflow позволяет машине состояний экспортировать события. Экспорт событий запускает внешний код, чтобы переключить события в машине состояний. Чтобы экспортировать событие, сначала добавляют его к

словарю данных как потомка машины состояний. Затем установите новое свойство индикатора события как Экспортируемое.

Примечание Внешние события могут быть родительскими только с помощью машины состояний. Это значит, что Вы должны использовать Проводник, для добавления внешних событий к словарю данных. Это значит, что внешние события видимы во всей машине состояний.

Когда кодируется машина состояний, родители которой экспортировали события, генератор кода Stateflow производит функцию для каждого экспортируемого события. Модель С для экспортируемой функции события имеет следующую форму

void external_broadcast_EVENT ()

Где EVENT - имя экспортируемого события. Внешний код, включающий объект, содержащий машину состояний, может переключать событие, вызывая его функцию. Например, предположим, что Вы определяете экспортируемое событие, которое называется switch_on. Внешний код

может переключать это событие, вызывая произведенную функцию

External_broadcast_trigger_on.

Импорт Событий

Машина состояний может импортировать события, определенные внешним кодом. Импортирование события позволяет машине состояний переключать событие во внешний код. Чтобы импортировать событие, сначала добавьте событие к словарю данных как потомка машины состояний, которая должна вызвать событие. Затем установите новое свойство индикатора события как Импортированное.

Примечание Машина состояний служит как замена родителя для импортированного события. Это значит, что Вы должны использовать Проводник, чтобы добавить импортированные события к словарю данных.

Stateflow предполагает, что внешний код определяет каждое импортированное событие как функцию, чья модель имеет следующую форму

void external_broadcast_EVENT

Где EVENT - имя импортированного события. Предположим,

что машина состояний импортирует внешнее событие, которое называется switch_on. Тогда Stateflow предполагает, что внешний код определяет следующую функцию: External_broadcast_switch_on, которая передает событие внешнему коду. Когда кодируется машина состояний и генератор кода Stateflow кодирует действия, этот сигнал импортирует события как запросы к соответствующему внешнему передающему событию функций, определенных внешним кодом.

Определение типов переключателя

Тип переключателя определяет, как управляющие сигналы переключают входящие и выходящие события, связанные с диаграммой. Типы переключателя делятся на две категории: функцию запроса и границы. Основное различие между этими двумя типами это уведомление о возникновении результирующих блоков. Получившиеся блоки уведомлены о границе переключения событий только в начале следующего такта моделирования, независимо от того, когда произошли события в течение предыдущего такта. Кроме того, получившиеся блоки уведомлены о функции

запроса переключения события в момент, когда события происходят, даже если они происходят в середине такта.

Вы устанавливаете тип переключателя диаграммы, выбирая свойство Trigger любого входящего или выходящего события, определенного для диаграммы. Если Вы хотите, чтобы диаграмма уведомила другие блоки в момент, когда происходит выходящее событие, установите свойство Trigger выходящего события как Function Call.Тип переключателя выходящего события должен быть Either Edge. Если диаграмма соединена с блоком, который является выходящим событием функции запроса, Вы должны определить свойство Trigger получаемого входящего события диаграммы, как Function Call. Stateflow изменяет все другие входящие события диаграммы в Function Call.

Если уведомление блоков о событиях, происходящих в этот момент не вызывает критическую ситуацию, Вы можете определить события как границу переключения. Вы можете задавать любой из следующих типов границы переключения:

Rising Edge. Повышенный уровень управляющего сигнала переключает событие.

Falling Edge. Пониженный уровень управляющего сигнала переключает событие.

Either Edge. Изменение в уровне сигнала переключает событие.

В любом случае, сигнал должен пересечь 0, чтобы совершить правильное переключение. Например, изменение от -1 до 1 составляет правильную повышенную границу, но при переходе 1 до 2 изменение не происходит.

Если Вы создаете тип границы переключения, который отличается от типа границы, определенного для диаграммы, Stateflow изменяет тип Trigger входящего события диаграммы на Either Edge.

Описание событий

Stateflow позволяет Вам хранить краткие описания событий в словаре данных. Чтобы описать особые события, установите свойство Description.

Документация событий

Stateflow позволяет Вам обеспечивать диалоговую документацию для событий, определенных моделью. Чтобы документировать особое событие, установите свойство Documentation, которое показывает документацию в диалоговом формате (например, HTML файл или текст в MATLAB).

Неявные события

Stateflow определяет и переключает следующие события, которые обычно происходят, когда выполняется:

- Активизация диаграммы
- Вход в состояние
- Выход из состояния
- Значение, назначенное внутренним данным объекта

Эти события называются неявными событиями, потому что Вы не должны определять или переключать их явно. Неявные события - потомки диаграммы, в которой они происходят и видимы только в них.

Ссылка на неявные события

Действия могут использовать следующий синтаксис, чтобы ссылаться на неявные события.

event(object)

Где event - имя неявного события, и object - состояние, где произошло событие. Действительные имена неявного события (и их сокращения) - wakeup (также упоминается как tick), enter (en), exit (ex), и change (chg).

Если более чем один объект имеет одно и то же имя, ссылка на событие должна определить имя объекта вместе с его предком. Несколько примеров действительных ссылок неявных событий.

enter(switch_on)

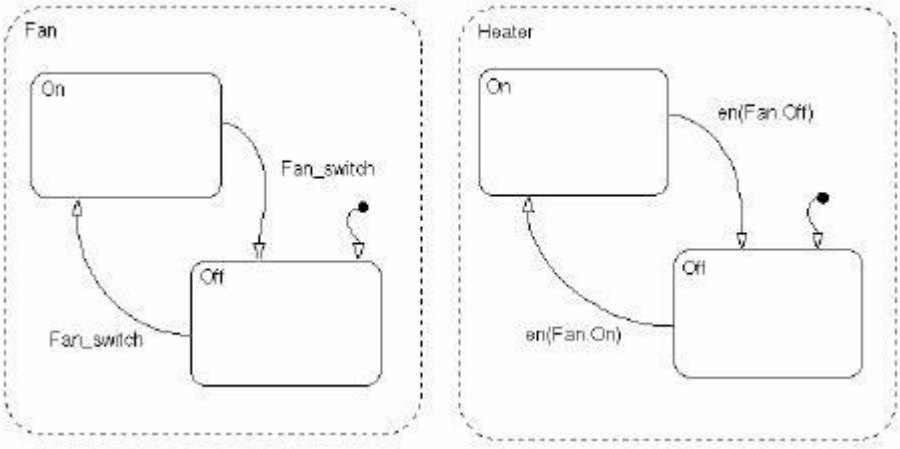
en(switch_on)

change(engine.rpm)

Примечание Когда пробуждение события вычисляется, оно всегда ссылается на диаграмму действия.

Пример

Этот пример иллюстрирует использование неявного enter события.



Fan и Heater параллельные (И) надсостояния. По умолчанию, первый раз диаграмма Stateflow пробуждается событием состояния Fan. Fan.Off и Heater.Off становятся активными. Сначала происходят Fan_switch и переход от Fan.Off к Fan.On. Когда выполняется входное действие Fan.On , передается неявное локальное событие (то есть, en(Fan.On) == 1). Эта передача события переключает переход от Heater.Off к Heater.On (условием en(Fan.On)). Точно так же, когда система переходит от Fan.On к Fan.Off и неявное локальное событие Fan.Off - передается, происходит переход от Heater.On to Heater.Off.

Определение данных

Машина состояний может хранить и восстанавливать данные, которые находятся внутри ее собственного рабочего пространства. Также это могут быть данные доступа, которые находятся в модели Simulink или в приложении, которое включает машину состояний. При создании модели Stateflow, Вы должны определить любые внутренние или внешние данные, упомянутые действиями машины состояний.

Определение объекта данных:

1. Добавьте объект к словарю данных.
2. Установите свойства нового объекта.

Добавление данных к словарю

Вы можете использовать либо редактор Stateflow, либо Проводник, чтобы добавить данные, которые являются доступным только в определенной диаграмме. Вы должны использовать Проводник, чтобы добавить данные, которые являются доступными во всей машине состояний или только в определенном состоянии.

Использование редактора Stateflow

Чтобы использовать редактор Stateflow для добавления данных, необходимо:

1. Из меню **Add** редактора Stateflow, выбрать **Data....**
2. В получившемся подменю выбрать индикатор данных.

По умолчанию Stateflow добавляет определение нового объекта к словарю данных и показывает диалоговое окно **Data**, которое отображает свойства нового объекта. Используйте диалоговое окно **Data**, чтобы задать такие значения свойств объекта, при которых он отображается в нужном виде.

Использование проводника

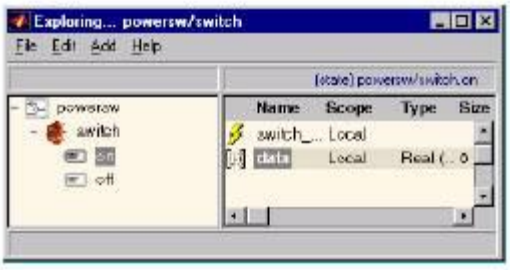
Используйте Проводник, чтобы определить объект данных:

1. Выберете **Explore** из меню редактора Stateflow **Tools.**

Stateflow откроет Проводник.

2. Выберите объект (машину состояний, диаграмму, или состояние) в иерархии объекта Проводника, там, где Вы хотите поместить новый объект.
3. Выберите **Data** из меню Проводника **Add**

По умолчанию Stateflow добавляет определение для нового объекта в словарь данных и показывает вход для объекта в оглавлении Проводника.



4. Выберите такие значения свойств объекта, чтобы они отображались в заданном виде.

Установка свойств данных

Вы определяете данные объекта, устанавливая его свойства.

Чтобы установить свойства данных объекта, выполните следующее:

1. Выберите Проводник из меню **Tools** редактора Stateflow. Выберите объект в содержании Проводника.

2. Выберите **Properties** из **Edit** Проводника или контекстного меню.

3. Stateflow показывает диалоговое окно **Data** для выбора объекта.

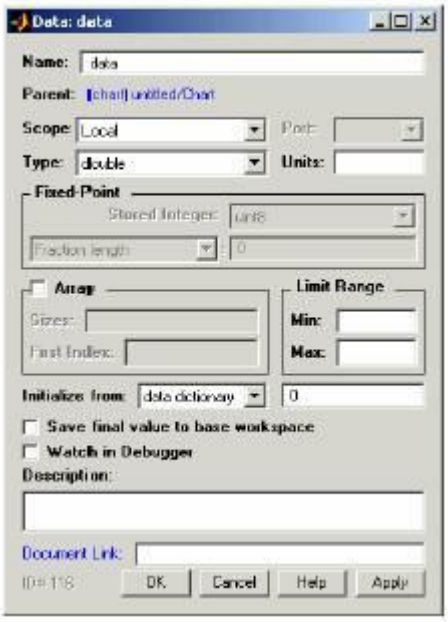
4. Используйте управление диалоговым окном, чтобы установить свойства объекта.

5. Выберите **OK**, чтобы применить ваши изменения и отменить диалоговое окно **Data**.

Примечание Вы может также устанавливать индикатор данных объекта, тип, размеры, начальное значение, минимальное и максимальное значение, в свойствах рабочего пространства редактированием соответствующих областей на входе объекта в содержании Проводника. Если Вы хотите установить только эти свойства, Вам не нужно открывать диалоговое окно **Data** для события.

Диалоговое окно Data

Диалоговое окно **Data** позволяет Вам устанавливать свойства объекта диалога.



Диалоговое окно включает в себя следующие параметры.

Name

Имя данных объекта. Имя данных может иметь любую длину и состоять из любой алфавитно-цифровой комбинации, за исключением вложенного пространства. Имя не может начинаться с числа.

Parent

Родитель этого объекта данных. Родитель определяет объекты, которые могут быть доступны ему. Только родитель объекта и его потомки могут быть доступны объекту. Вы определяете родителя данных объекта, когда добавляете его к словарю данных.

Scope

Индикатор этого объекта данных. Индикатор объекта данных определяет, где он находится в памяти относительно его родителя. Параметры для свойств индикатора:

Lokal. Локальный объект данных находится и доступен только в машине, диаграмме, или состоянии.

Input from Simulink. Это объект данных, который доступен в блоке диаграммы Simulink, но находится в другом блоке, который может быть блоком диаграммы или другим блоком. Получившийся блок диаграммы считывает значение объекта данных из порта входа, связанного с объектом данных.

Output to Simulink. Это объект данных, который находится в блоке диаграммы и доступен в другом блоке, который может быть блоком диаграммы или другим блоком. Блок диаграммы выходного значения данной величины и порт выхода связаны с объектом данных.

Temporary. Временные данные объекта существуют только тогда, когда его родитель выполняется.

Constant. Постоянные данные объекта существуют только для чтения и сохраняют начальный набор значений в диалоговом окне свойств **Data.**

Export. Экспортируемые данные объекта - это данные машины состояний, которые могут быть доступны с помощью внешнего кода, который включает машину состояний.

Import. Импортированные данные - данные, определяемые внешним кодом, который может быть доступен с помощью машины состояний, вложенной в него.

Примечание Если Вы копируете блок диаграммы Stateflow из одной модели Simulink в другую, все объекты в иерархии диаграмм скопированы, кроме тех, которые стали родительскими с помощью машины состояний Stateflow. Это значит, что данные, которые становятся родительскими с помощью машины Stateflow (локальное, импортированное, экспортированное), не скопированы в новую машину. Однако Вы можете использовать Проводник, чтобы копировать индивидуальные данные, перетаскивая их курсором из одной машины в другую.

Type

Тип данных этого объекта, например, целочисленный, двойной, и т.д.

Port

Индекс порта связан с данными этого объекта. Это управление применяется только к входящим и выходящим данным.

Units

Единицы - например, дюймы, сантиметры, и т.д., представленные данными этого объекта. Значение этой области с объектом хранятся в словаре данных машины состояний.

Array

Данные этого объекта - множество. Параметры:

Sizes. Размер этого множества. Значением этой величины может быть скаляр или вектор MATLAB. Если это - скаляр, то определяется размер одномерного множества, (то есть, вектор). Если вектор MATLAB, то указывается размер каждого измерения многомерного множества, номер измерений которого соответствует длине вектора.

First Index. Определяет индекс первого элемента этого множества. Например, первый индекс нулевого множества - 0.

Limit Range

Эта группа управления определяет значения, используемые машиной состояний, чтобы проверить законность данных этого объекта. Она включает следующие два управления:

Min. Минимальное значение, которое данные этого объекта могут иметь в течение выполнения или моделирования машины состояний.

Max. Максимальное значение, которое данные этого объекта могут иметь в течение выполнения или моделирования машины состояний.

Initialize from.Источник начального значения для данных этого объекта - это либо словарь данных Stateflow, либо рабочее пространство MATLAB. Если данные этого объекта - множество, Stateflow устанавливает каждый элемент множества к указанному начальному значению.

Если источник - словарь данных, введите начальное значение в смежную область текста. Stateflow сохраняет значение, которое Вы вводите в словарь данных.

Если источник - рабочее пространство MATLAB, этот объект получает свое начальное значение из подобного названия переменной в рабочем пространстве. Предположим, что имя объекта данных Stateflow - A. Если родительское рабочее пространство имеет переменную по имени A, его значение использует инициализацию данных объекта.

Следующая таблица суммирует время инициализации для каждого индикатора.

Родитель	Индикатор	Инициализация
Машина	Local	Начало моделирования

	Export Import	Начало моделирования Не применяется
Диаграмма	Input from Simulink Output to Simulink Local	Не применяется Начало моделирования или когда диаграмма заново инициализируется как часть подсистемы Simulink
Состояние Без соединения с памятью С соединением с памятью	Local	Состояние активации (состояние введено) Начало моделирования или когда диаграмма заново инициализируется как часть подсистемы Simulink
Графическая Функция	Input from Simulink Output to Simulink Local	Не применяется Когда функция активизирована Начало моделирования или когда диаграмма заново инициализируется как часть подсистемы Simulink

Примечание Вы также может использовать Проводник Stateflow, чтобы установить область **Initialize from**.

Save final value to base workspace

Проверка этого параметра вызывает значение данных объекта, назначенных как название переменной в рабочем пространстве модели в конце моделирования.

Watch in debugger

Этот параметр вызывает отладчика, чтобы остановиться, если данные этого объекта изменены.

Description

Описание данных этого объекта.

Document link

Щелчок по этой области показывает предоставленную пользователем диалоговую документацию для данных этого объекта.

Определение множеств данных

Stateflow позволяет Вам определять множества данных. Чтобы определить множество, необходимо:

1. Добавьте данные объекта к словарю как потомка состояния, диаграммы, или машины, которая нуждается в доступе данных.

2. Откройте диалоговое окно **Data**. Проверьте окно **Array**. Установите свойство объекта **Sizes** к размеру каждого из измерений множества **Setting Data Properties**.

Например, чтобы определить вектор со 100 элементами, установите свойство Sizes в 100.

Чтобы определить множество 2 на 4, установите свойство Sizes в [2 4].

3. Установите свойство индекса объекта Initial в индекс множества первого элемента. Например, чтобы определить нулевое множество, установите свойство индекса Initial в 0.

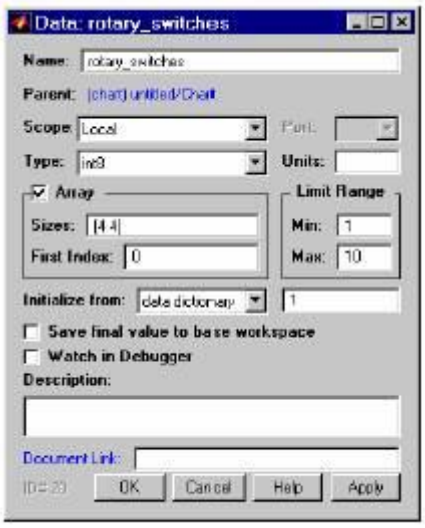
4. Установите источник инициализации объекта и если инициализация происходит из словаря данных, то установите начальное значение.

Например, чтобы определить, что элементы множества инициализированы в 0, установите **Initialized from** из параметра диалогового окна **Data** в словаре данных и введите 0 в смежную область текста.

5. Установите другие параметры в диалоговом окне (например, Name, Type, и так далее), чтобы отразить предназначенное использование данных.

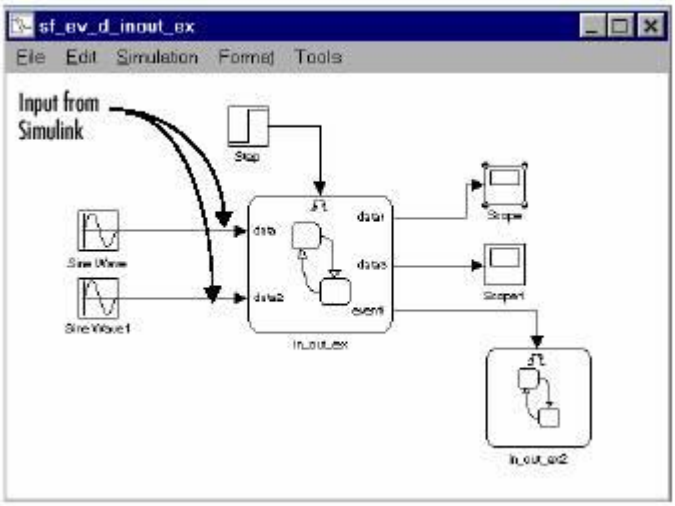
Пример.

Предположим, что Вы хотите определить локальное нулевое множество целого типа 4 на 4, названное rotary_switches. Далее предположим, что каждый элемент множества был первоначально равен 1 и не мог иметь никаких значений меньше чем 1 или больше чем 10. Следующее диалоговое окно **Data** показывает назначения для такого множества.



Определение входящих данных

Stateflow позволяет модели снабжать данными диаграмму через порты входа в блоке диаграммы. Такие данные называются входящими данными. Чтобы определить объект входящих данных, добавьте его к словарю данных Stateflow как потомка диаграммы, которая будет входящими данными. Установите индикатор нового объекта как **Input from Simulink.** Stateflow добавляет порт входа к диаграмме для каждого объекта входящих данных, которые Вы определяете для диаграммы.

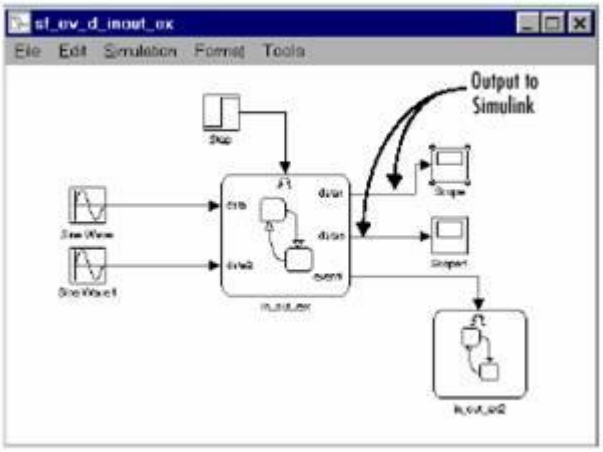


Установите другие свойства объекта (например, Name, Type, и т.д.) соответствующих значений.

Вы можете устанавливать тип данных входящего объекта в любой поддерживающийся тип Stateflow. Если активизирован параметр обязательного заполнения данных, входящие сигналы должны соответствовать указанному типу, иначе появится ошибка несоответствия. Если параметр обязательного заполнения данных не активизирован, входящие сигналы должны быть типа **double**. В этом случае Stateflow преобразует входящие значения к указанному типу. Если входящий объект - вектор, модель должна предоставить данные через сигнальный вектор, связанный с соответствующим портом входа в диаграмме.

Определение выходящих данных

Выходящие данные - данные, которыми диаграмма снабжает другие блоки через порты выхода. Чтобы определить объект выходящих данных, добавьте данные объекта к данных как потомка диаграммы, который снабжает объект. Затем установите индикатор свойств нового объекта в **Output to Simulink.** Stateflow добавляет порт выхода к диаграмме для каждого выходящего объекта.



Вы можете устанавливать тип выходящего объекта в любой поддерживающийся тип данных Stateflow (например, целочисленный). Если

параметр обязательного заполнения данных активизирован, диаграмма выведет сигнал того же типа данных, как тип выходящих данных объекта. Если этот параметр не активизирован, блок диаграммы Stateflow преобразовывает выходящие данные к типу **double.**

Общая постановка задачи

Овладеть практическими навыками построения моделей в форме конечных автоматов на основании средств Stateflow.

Список индивидуальных данных

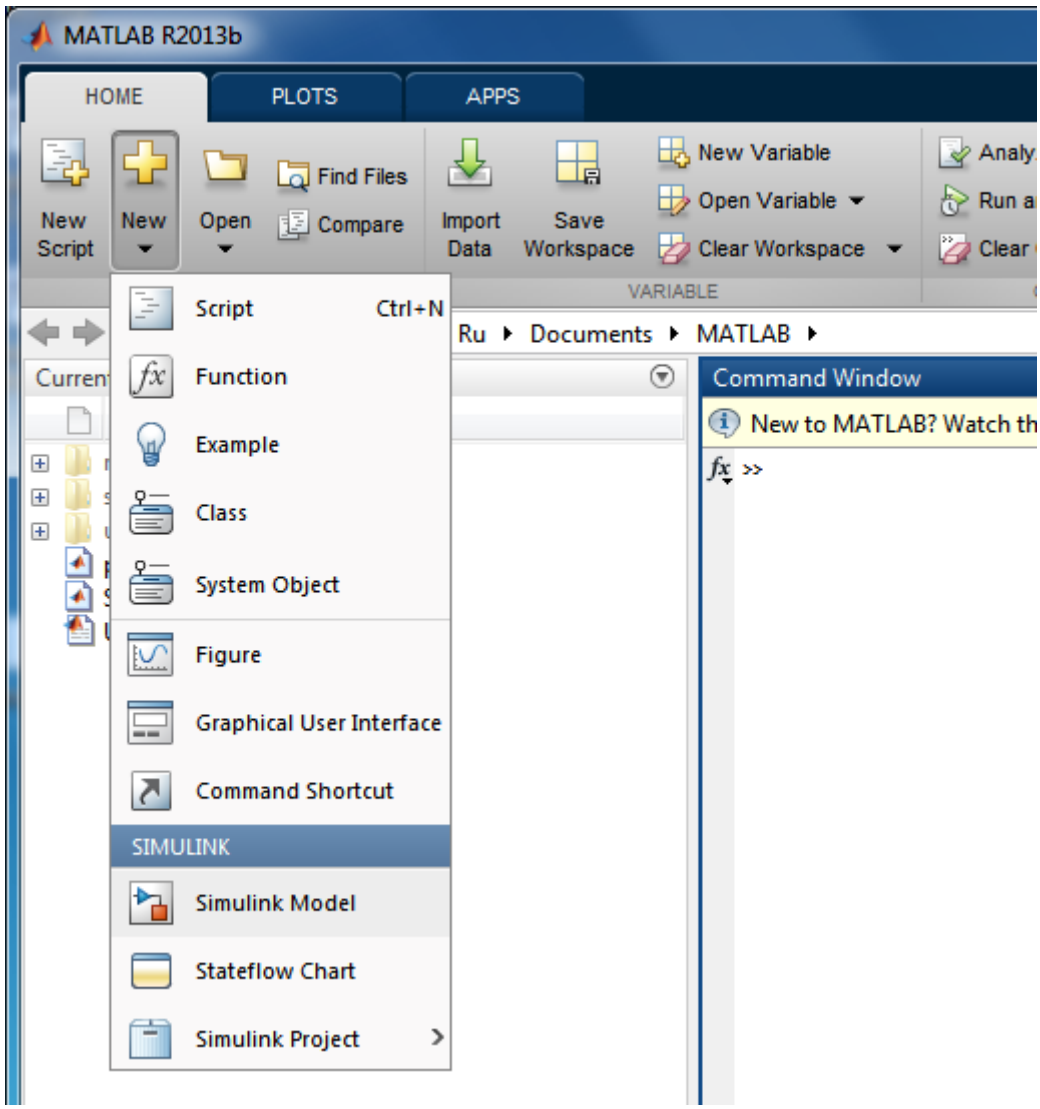
Разрабатываемую модель первоначально необходимо формализовать в виде графа и матриц перехода и выходов.

1. Смоделировать работу банкомата, выдающего наличные денежные средства. При этом у банкомата имеется возможность выдачи средств с разменом.
2. Смоделировать работу банкомата, выдающего наличные денежные средства.
3. Смоделировать работу стиральной машины.
4. Смоделировать работу отопительного котла.
5. Смоделировать работу мультиварки.
6. Смоделировать работу светофора, имеющего дополнительные стрелки влево и вправо.
7. Смоделировать работу двоичного сумматора.
8. Смоделировать работу автомата, выдающего различные напитки.
9. Смоделировать работу лифта.
10. Смоделировать работу АКП, а именно процесс переключения передач в зависимости от скорости авто.
11. Смоделировать работу семафора на переезде.
12. Смоделировать работу осветительного столба.
13. Смоделировать работу микроволновой печи.
14. Смоделировать работу электрочайника.
15. Смоделировать работу утюга.

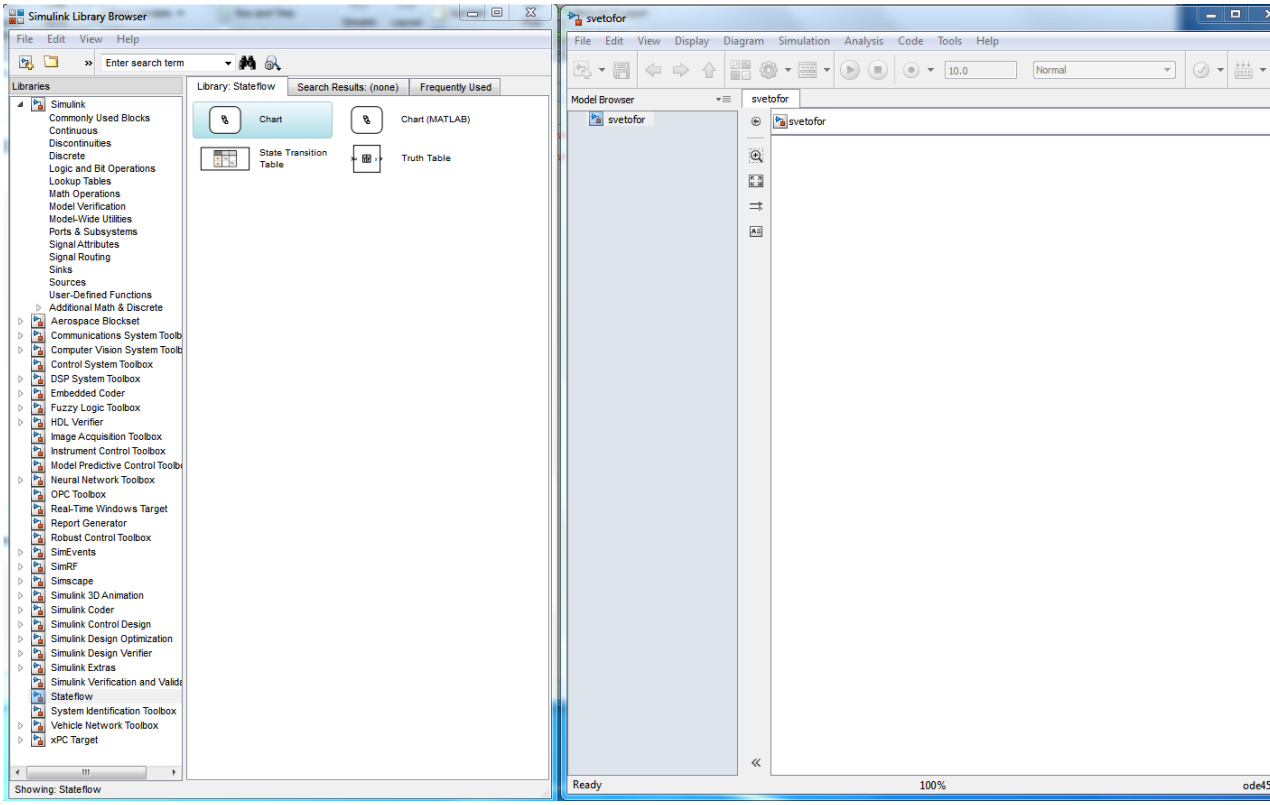
Пример выполнения работы

В качестве примере рассмотрим процесс моделирования работы светофора. Светофор имеет 3 цветовых индикатора (красный, желтый и зеленый). При этом переход от одного цвета к другому привязан ко времени. Время перехода из одного состояния к другому зависит от места установки светофора, которое определяется интенсивностью движения транспортных средств и пешеходов.

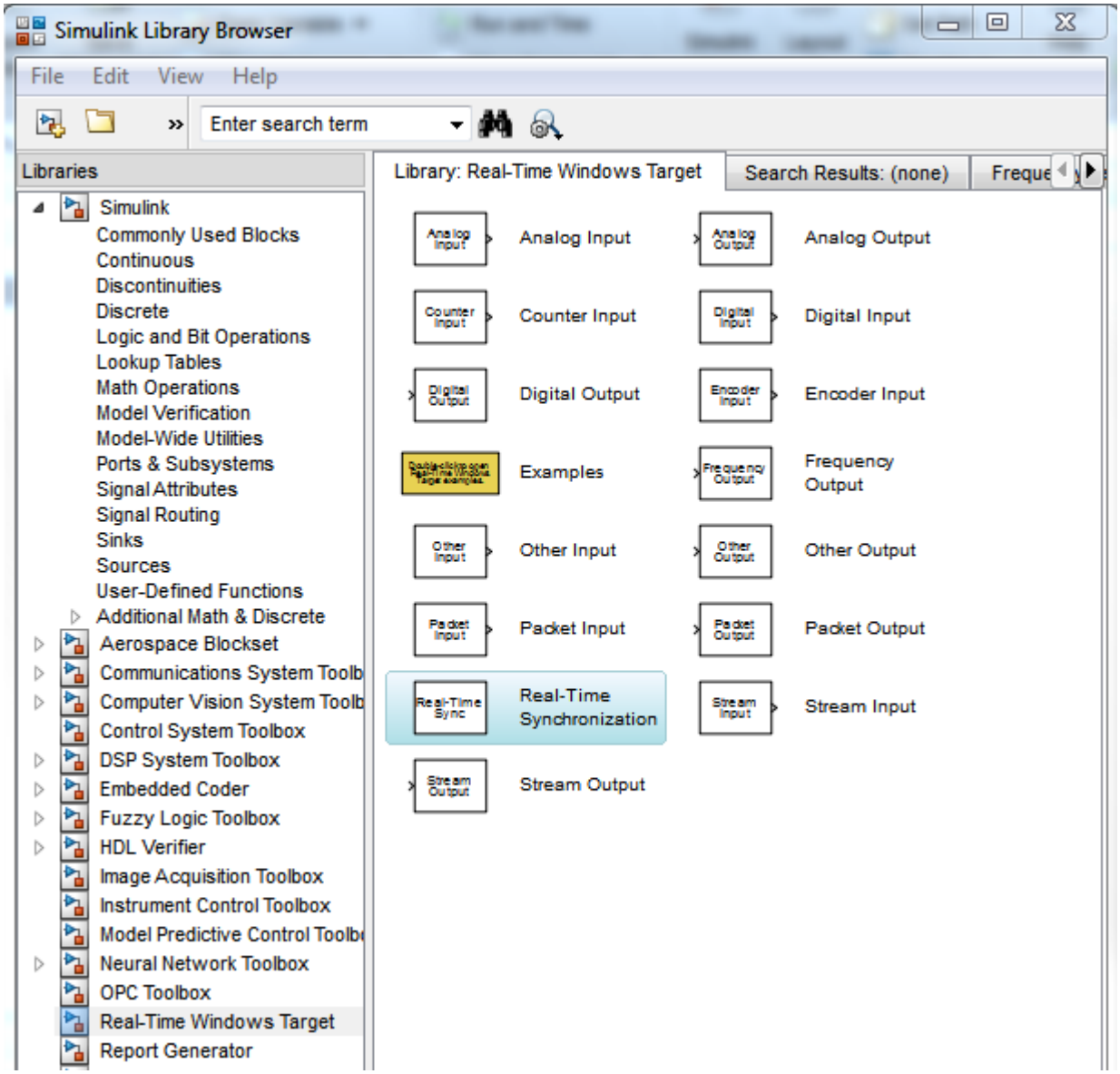
Первое с чего необходимо начать моделирование – это создание в пакете Matlab новой модели Simulink и сохранение ее под заданным Вами именем.



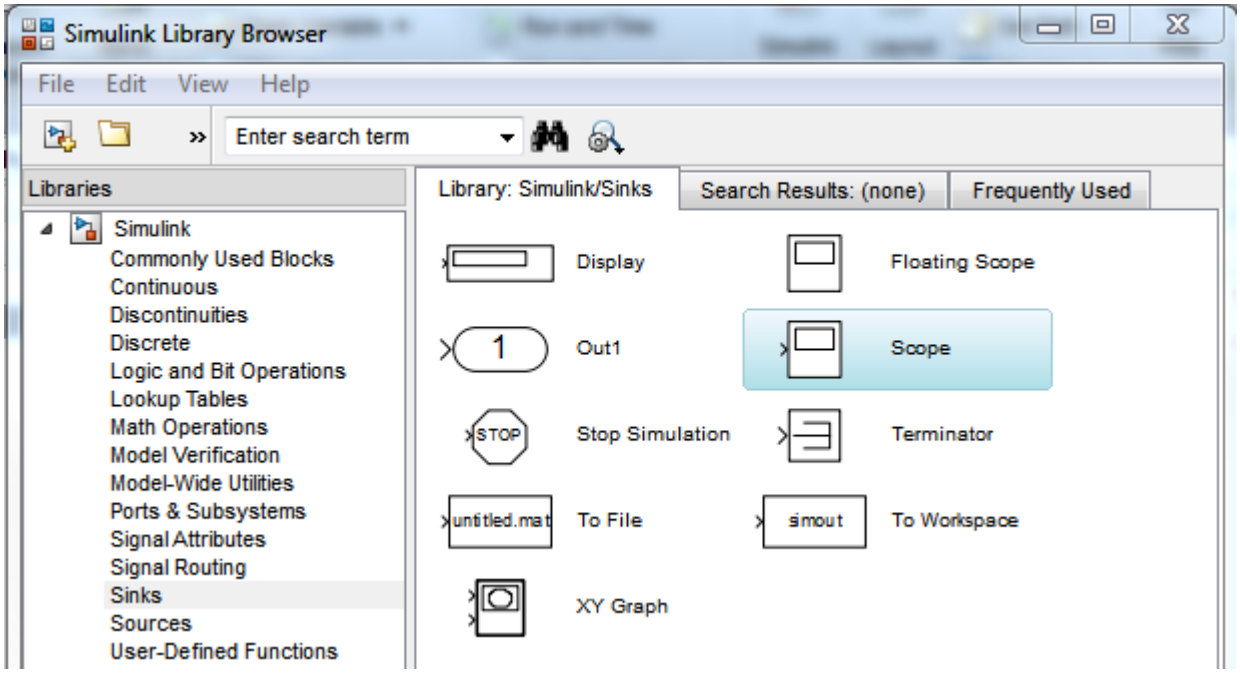
После этого настриваем рабочую область в удобном для работы виде. А именно растягиваем до необходимых размер окно новой модели, открываем библиотеку Simulink и выбираем библиотеку Stateflow.



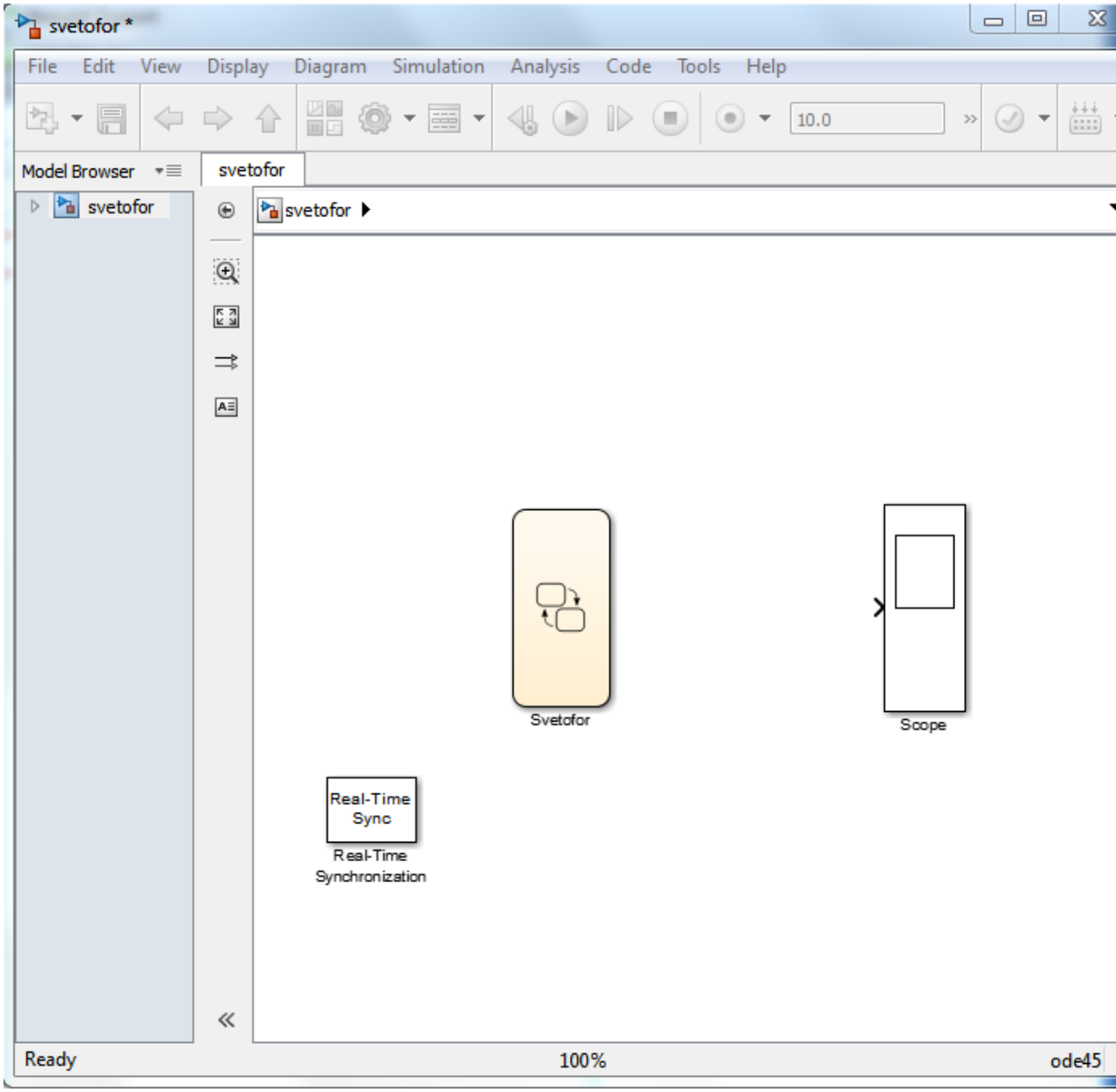
Перетаскиваем элемент Chart в рабочую область нашей модели. Данный элемент и будет производить имитацию работы светофора. Для того, чтобы имитация производитлась в реальном времени необходимо в модель добавить элемент Real-Time-Synchronization, расположенный в библиотеке Real-Time Windows Target.



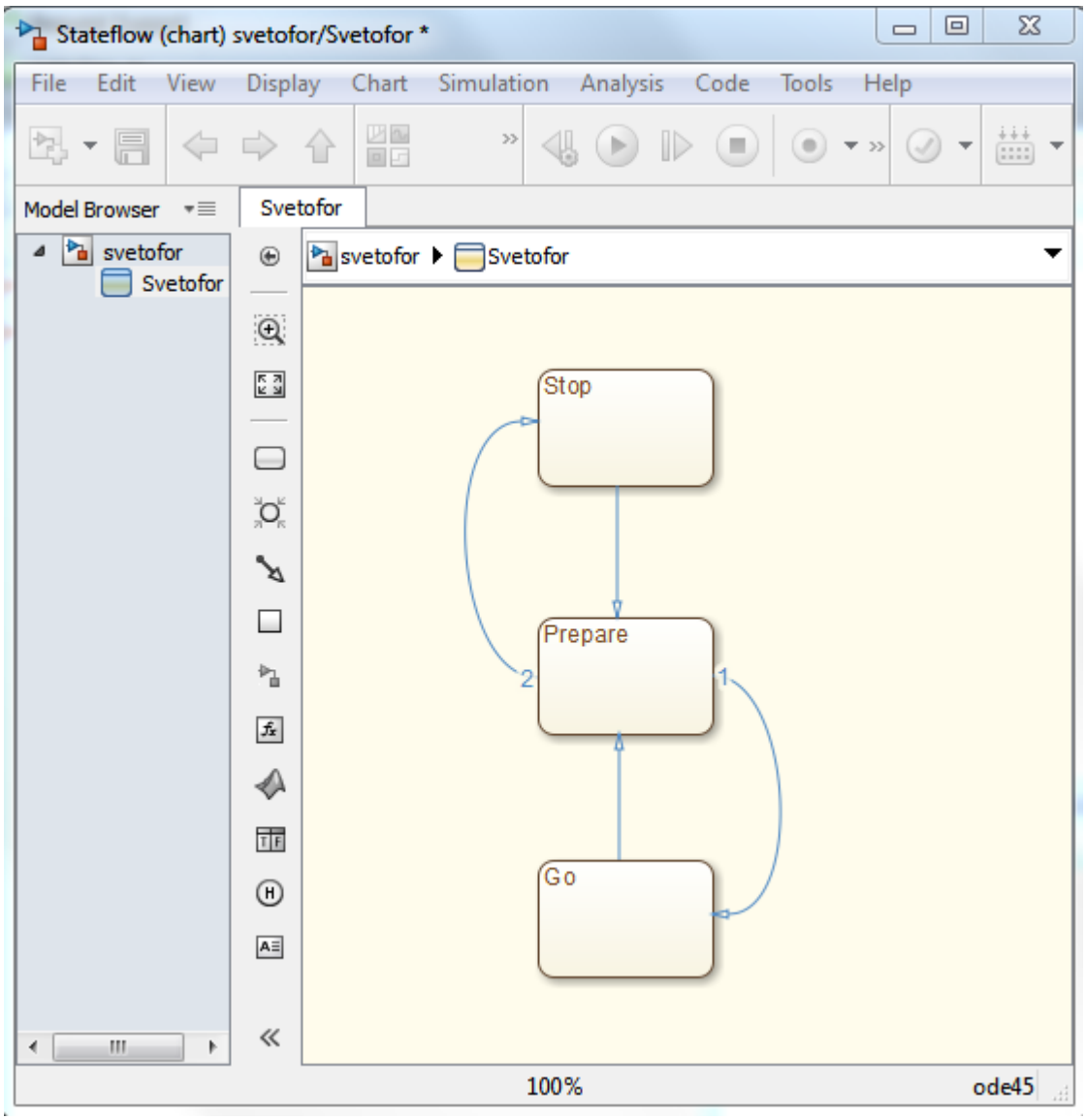
Вставим еще один элемент, который будет отображать ход процесса моделирования во времени. Таким элементом будет осцилограф Scope, расположенный в библиотеке Skins.



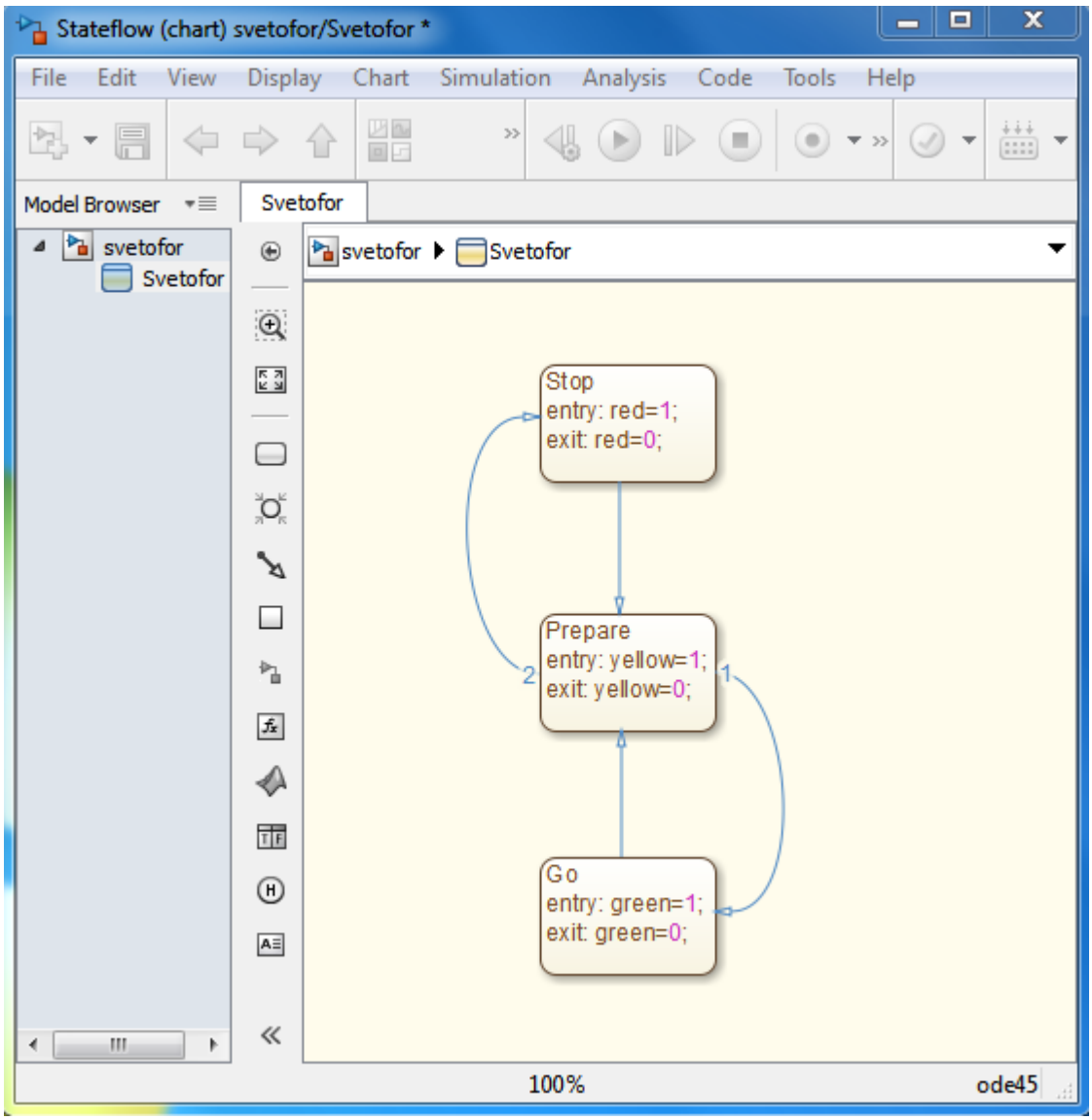
В результате наша модель будет выглядеть следующим образом.



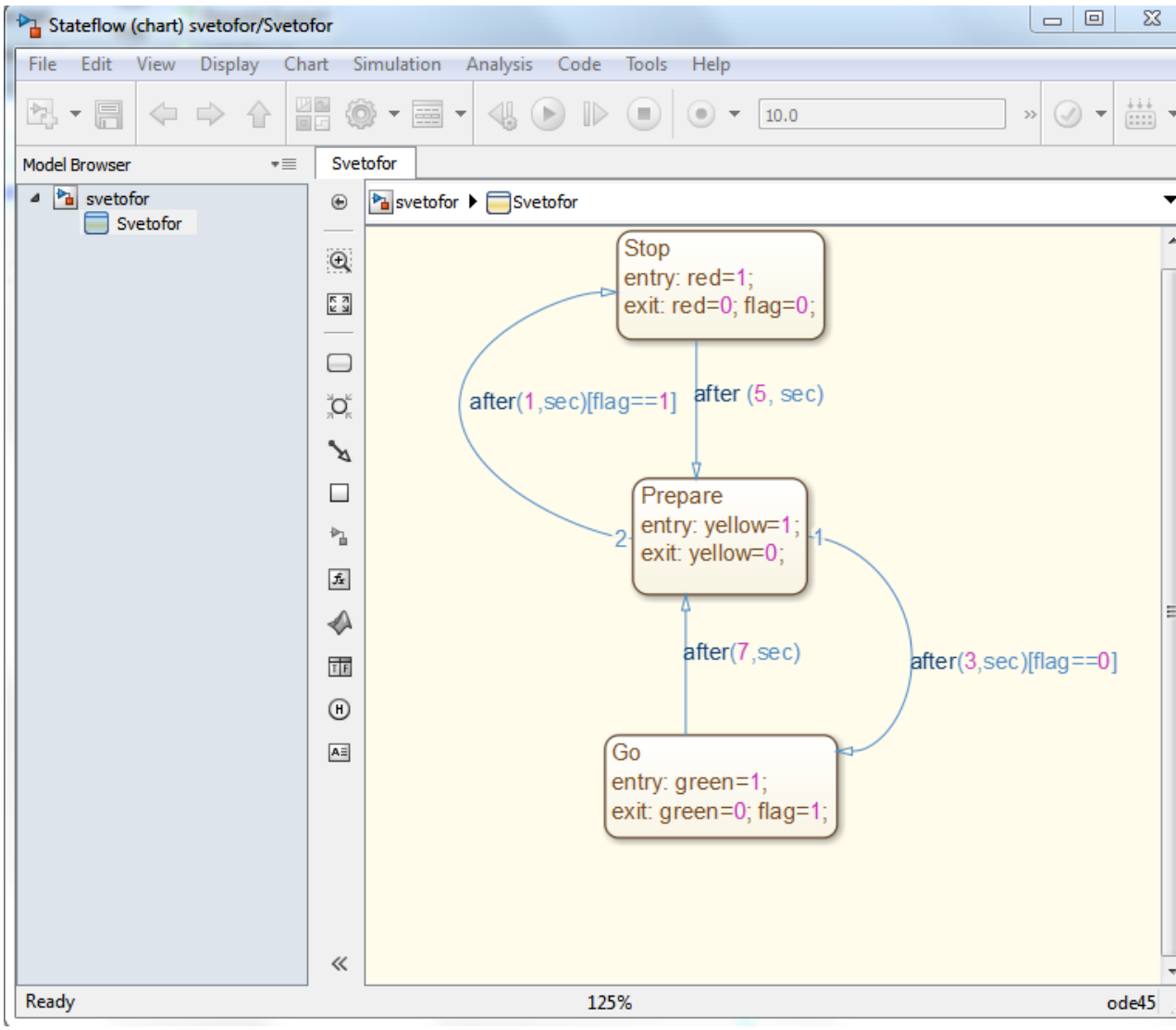
Откроем элемент Svetofor и определим в нем три состояния (Go, Prepare, Stop) и свяжем их между собой, исходя из логики перехода между состояниями. В результате получаем структуру следующего вида.



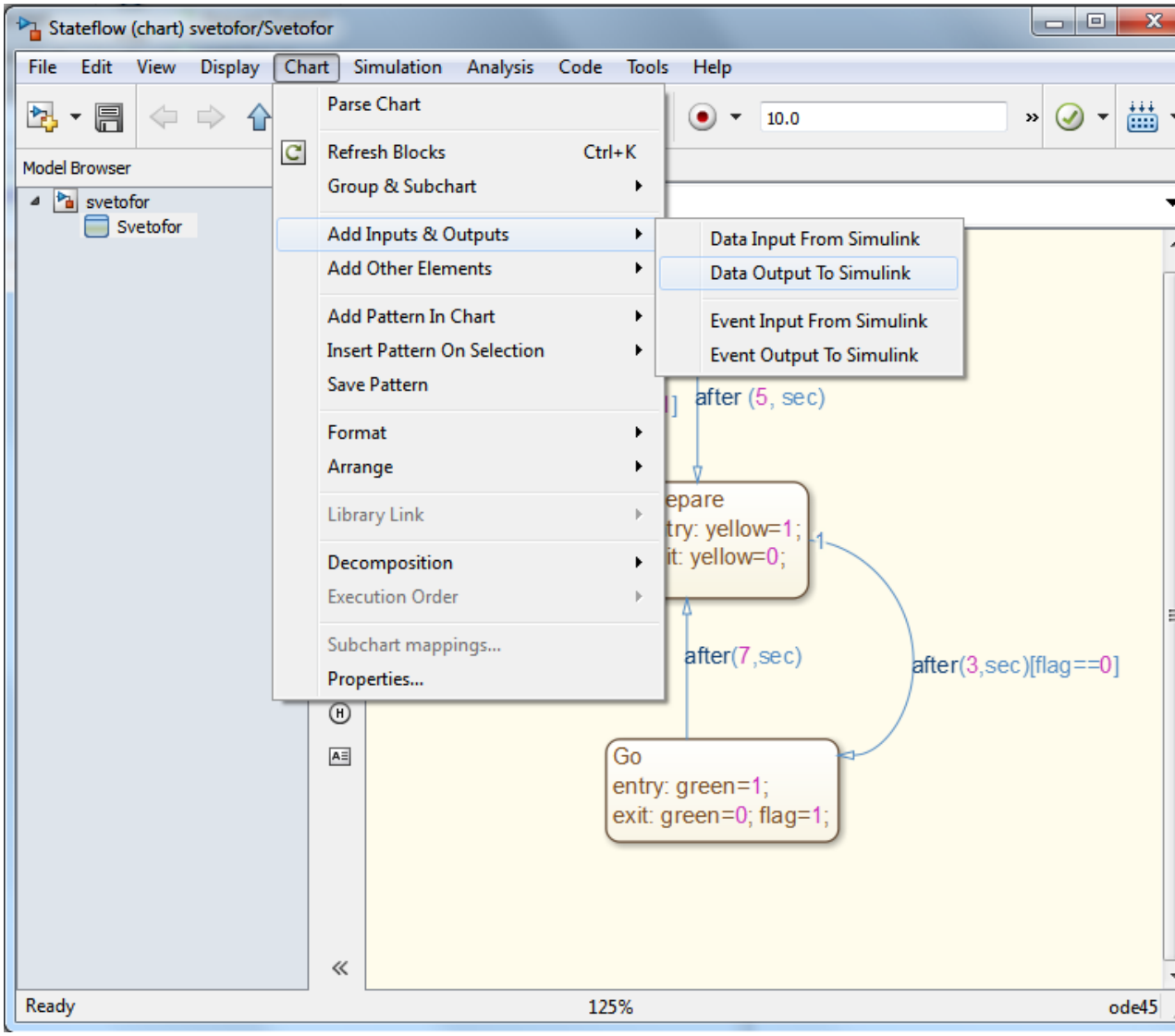
Далее определим, что будет происходить при входе в состояния и выходе из них. С этой целью зададим три переменных (red, green, yellow).



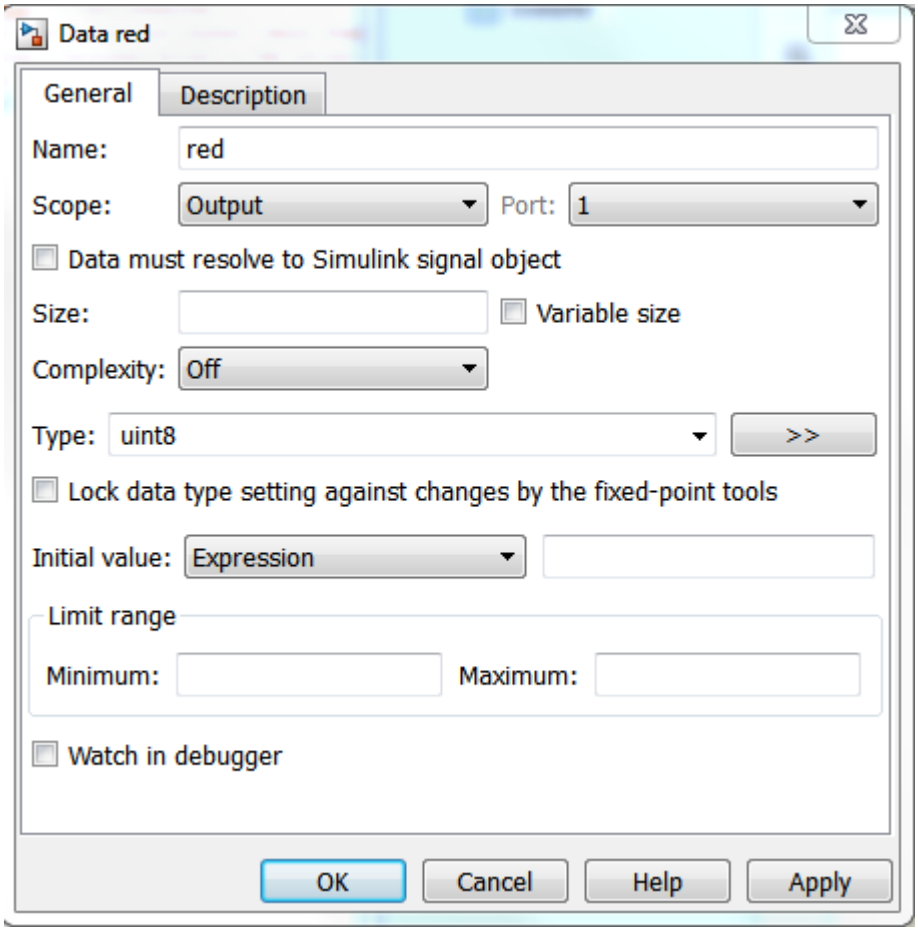
Теперь необходимо определить условия перехода между состояниями. Переходы между состояниями помимо условий будут определять временные интервалы, определяющие время активности состояния. Для определения параметров переходов из состояний Stop и Go в состояние Prepare достаточно лишь задать время при помощи after(время, единица измерения). А вот для перехода из состояния Prepare необходим дополнительный параметр, определяющий по какому именно переходу будет осуществлен переход. С этой целью введем дополнительную переменную flag, которая будет в состоянии Stop принимать значение 0, а в состоянии Go равняться 1.



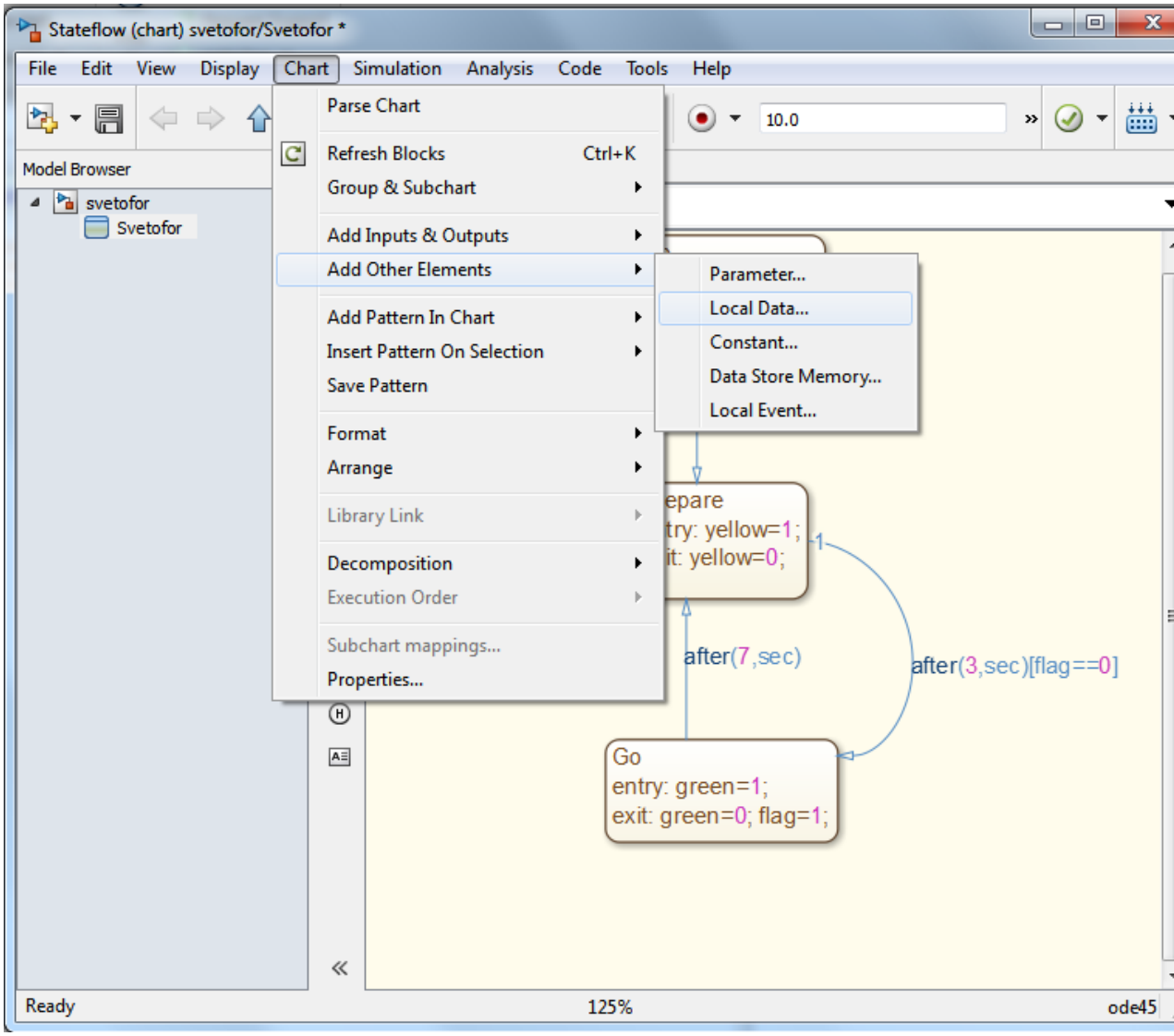
Определим тип переменных. Переменные red, yellow, green будут передаваться в Simulink.



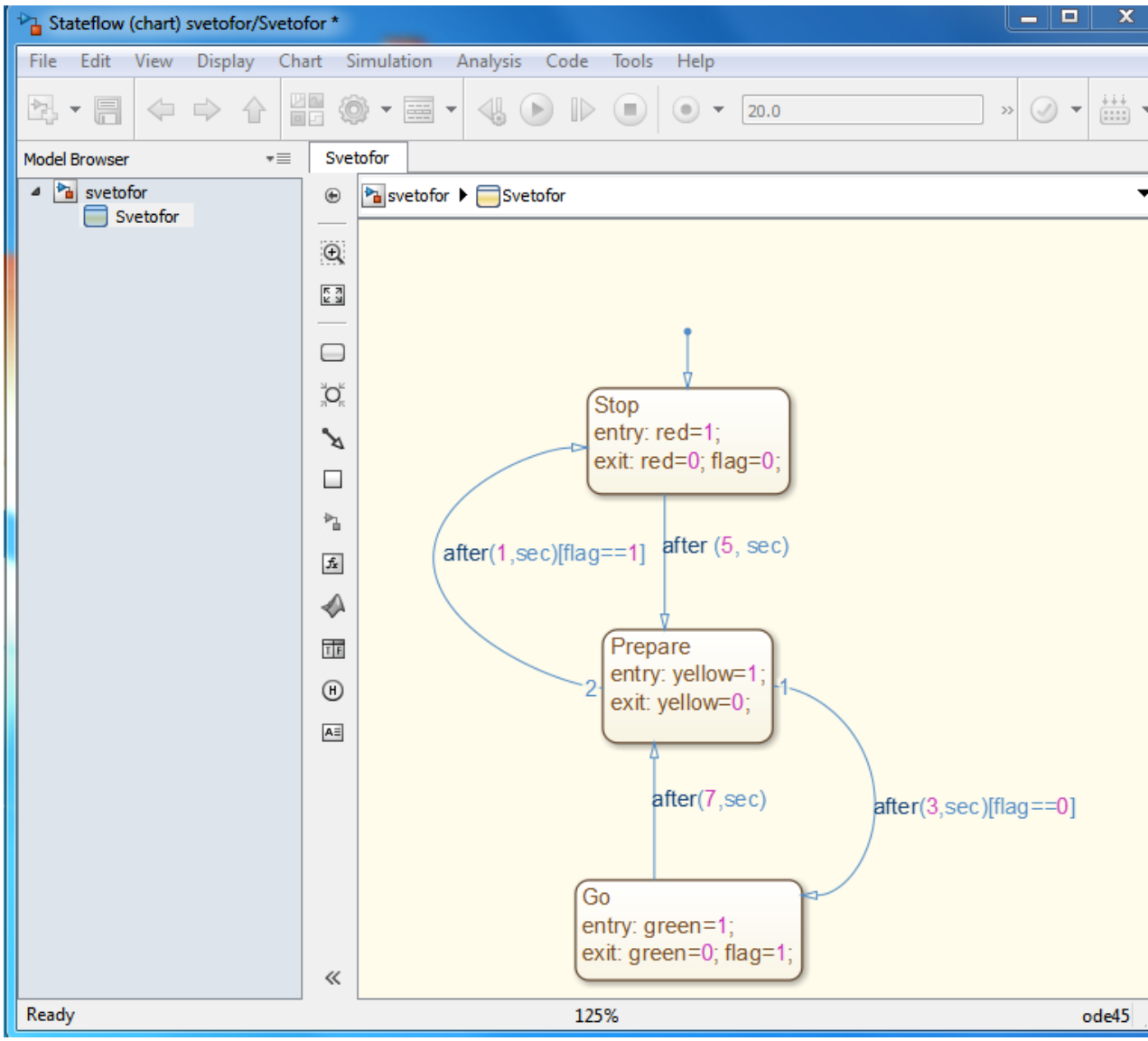
В результате высветится следующее окно, в котором необходимо указать имя переменной и тип.



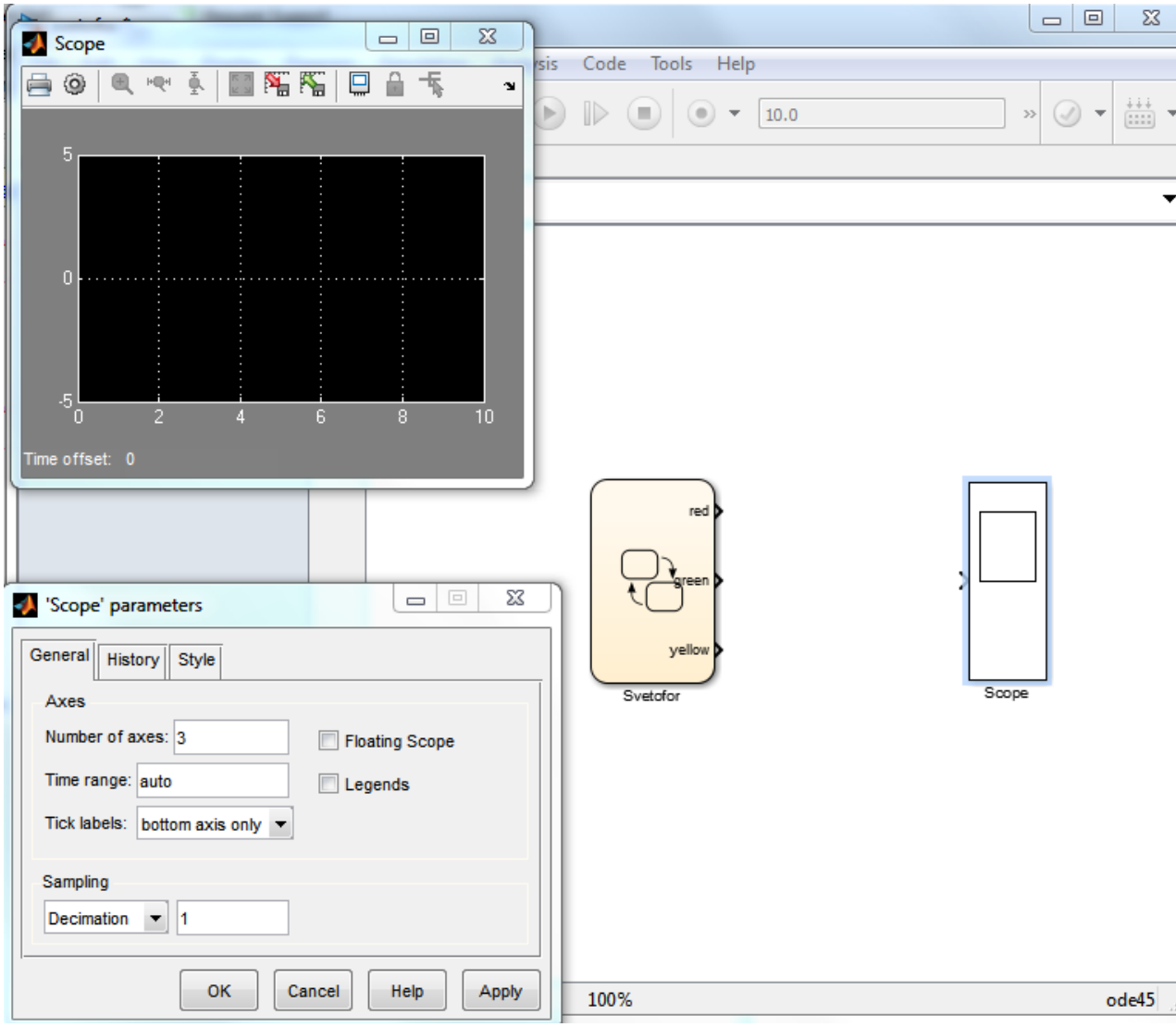
Переменная flag будет использоваться только внутри диаграммы (local). Поэтому необходимо вызвать следующее окно и по аналогии с предыдущими переменными указать имя и тип переменной.



Теперь необходимо определить состояние, которое будет активно при запуске модели.



Далее необходимо подключить выходные порты нашей диаграммы к осциллографу. По умолчанию осциллограф имеет 1 входной порт, поэтому необходимо открыть свойства осциллографа и определить 3 входа.



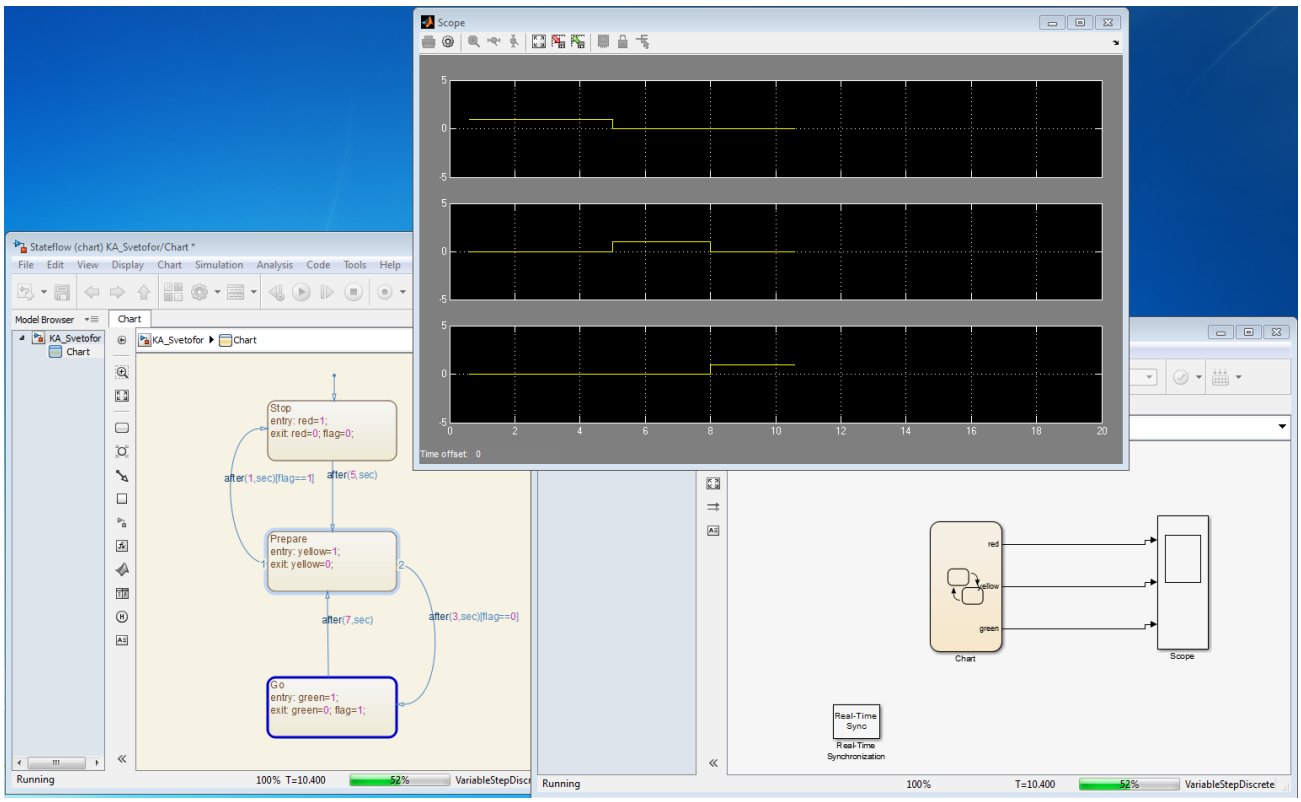
Теперь соединяем выходные порты диаграммы со входами осциллографа. Зададим время моделирования 20 секунд и запустим модель.

При запуске модель может сработать не в режиме реального времени, то есть сразу построит готовый график. При этом в диалоговом режиме matlab высветится предупреждение следующего вида:

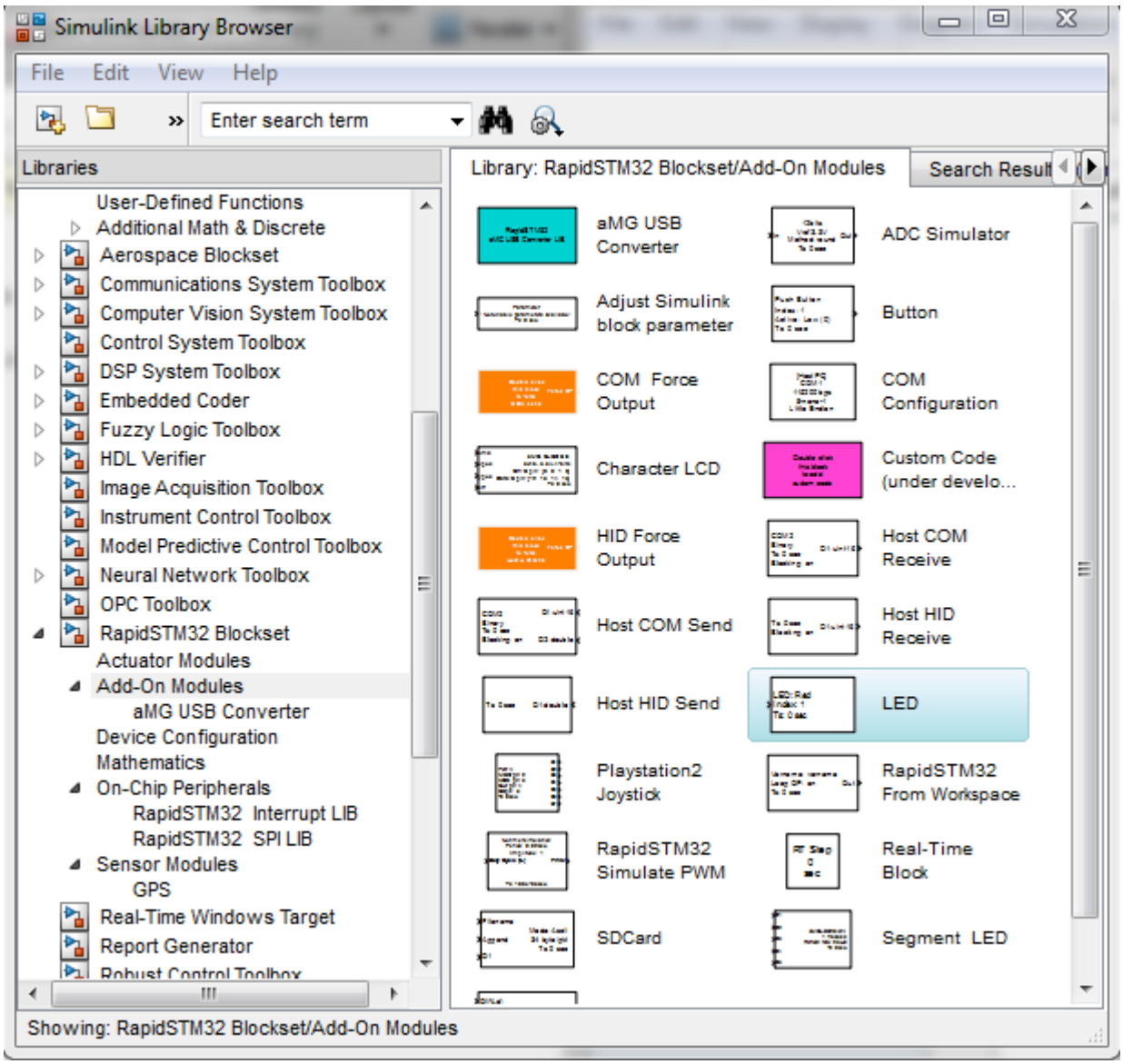
Warning: Real-Time Windows Target kernel is not installed. The "Real-Time Synchronization" block will not perform I/O operations and will not be synchronized to real time.

В нем сообщается, что не установлено ядро блока real-time windows target, поэтому не может быть запущен блок real-time synchronization.

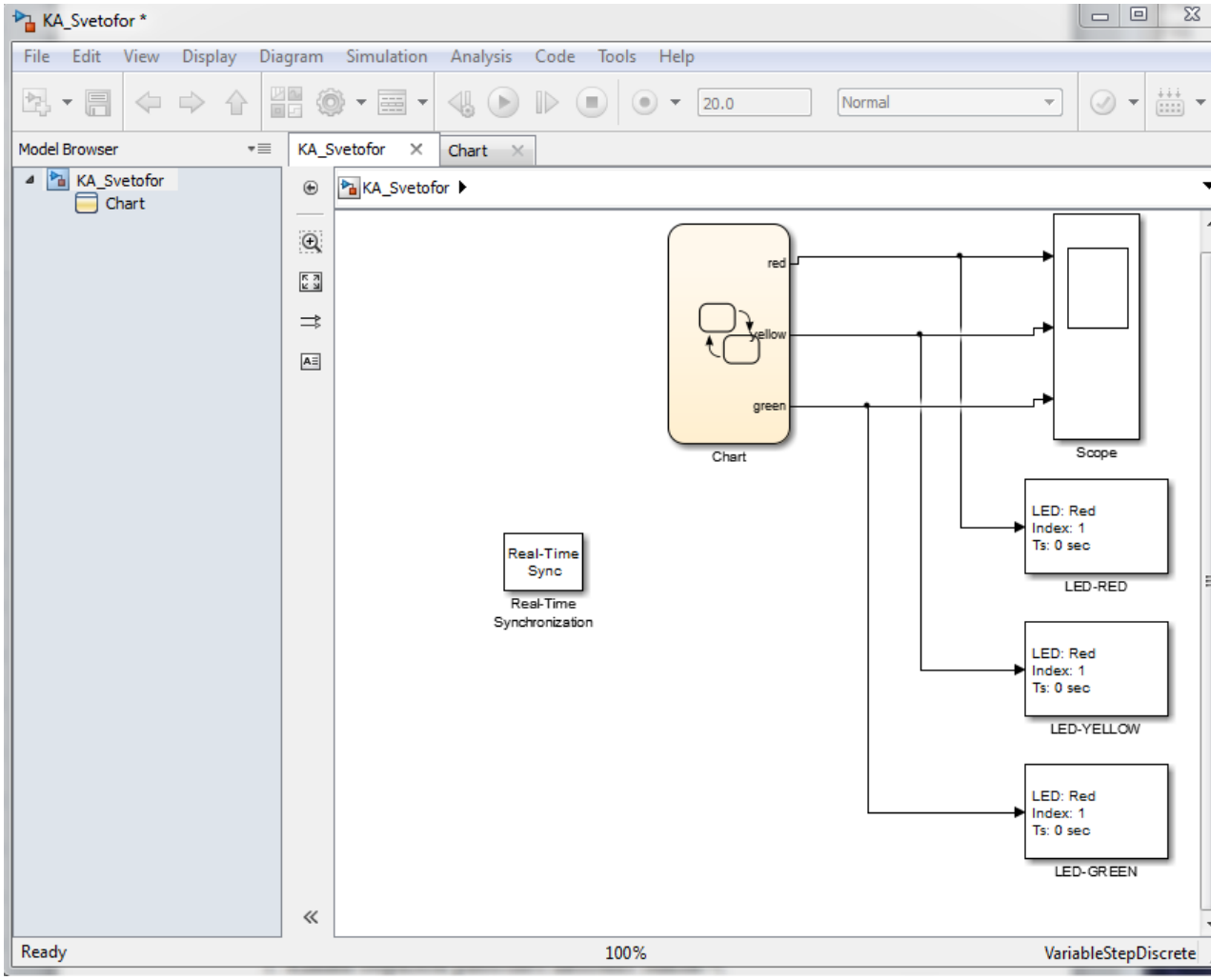
Для устранения возникшей проблемы необходимо в строке команд matlab запустить следующую команду: rtwintgt -setup и подтвердить ее выполнение введя Y. Теперь наша модель будет выполняться в режиме реального времени.



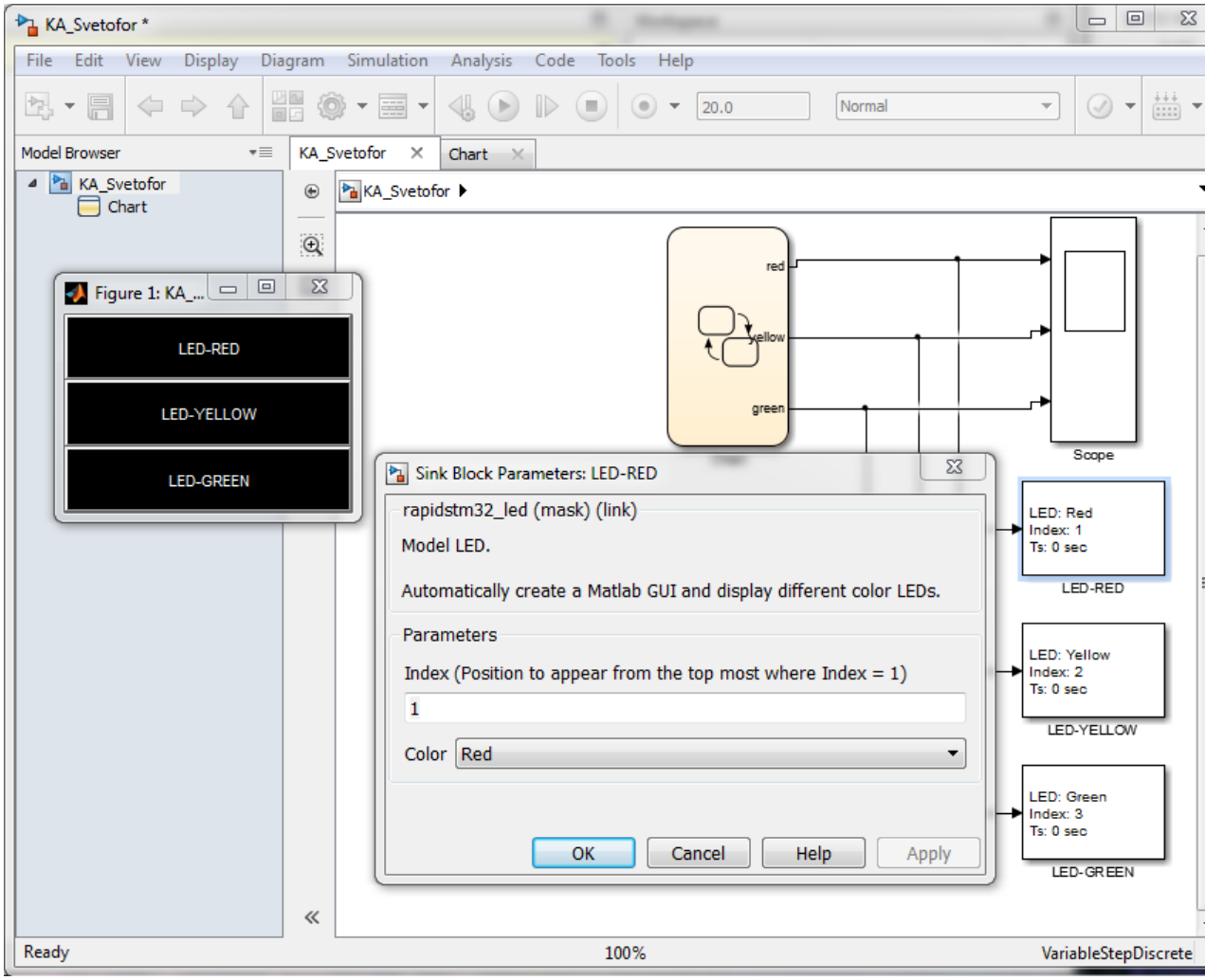
Для более реальной имитации работы светофора добавим к созданной модели светодиодную индикацию. Для этого необходимо установить дополнительную библиотеку RapidSTM32. Добавим к модели три модуля LED, которые будут отвечать за цветовую индикацию 3 цветов светофора.



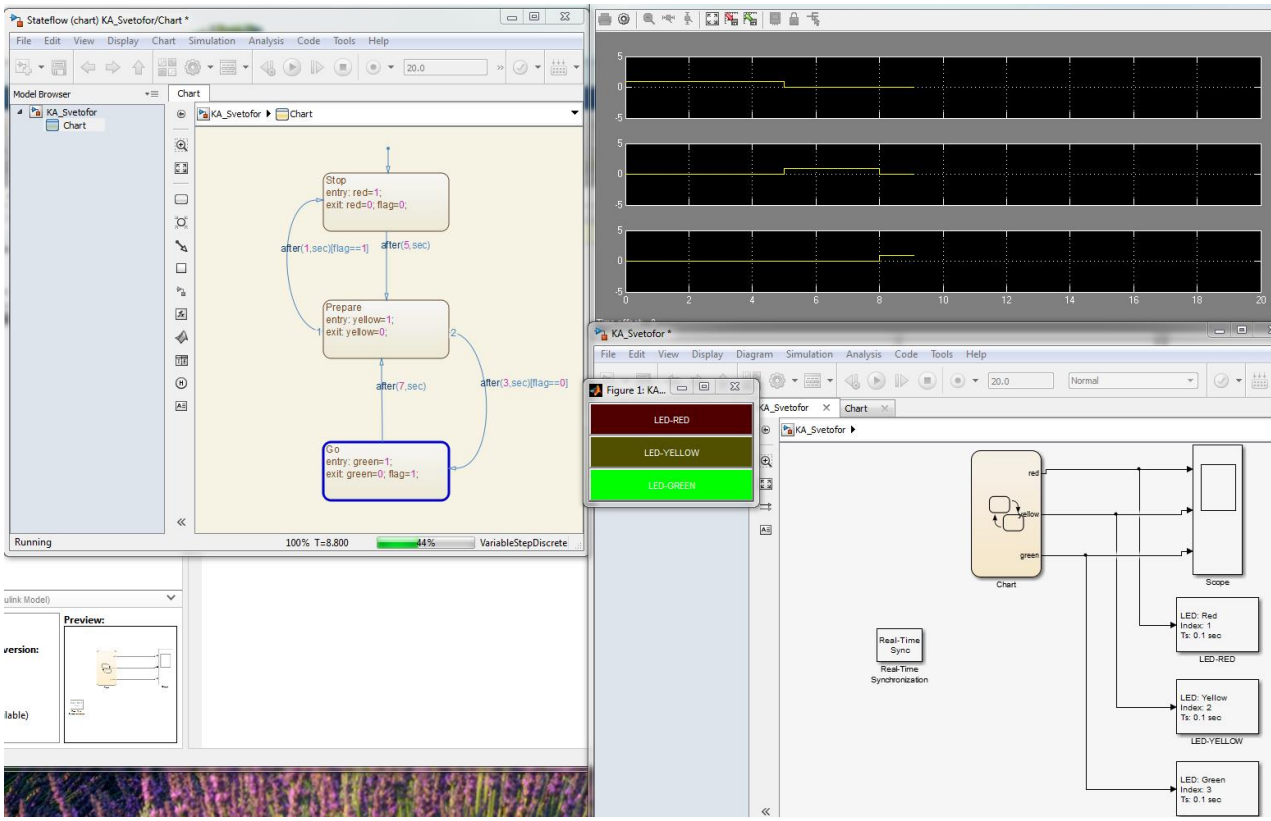
После того как мы подписали каждый новый модуль и соединили его с соответствующим выходом конечного автомата, мы получим следующую модель.



Теперь необходимо задать цвет для каждого индикатора LED и определить позицию в общем цветовом индикаторе, задав соответствующий индекс.



Запустив модель, мы можем наблюдать как за графической (диаграмма stateflow и графики осциллографа) и цветовой индикацией.



Контрольные вопросы к защите

1. Каким образом работает автомат Мили?
2. В чем заключается принципиальное отличие автомата Мура от Мили?
3. Что такое Simulink?
4. Для чего предназначен Stateflow?
5. Что такое состояние в нотации Stateflow?
6. Что такое событие в нотации Stateflow?
7. Что такое переход в нотации Stateflow?
8. Где используются условия при построении диаграмм Stateflow?
9. Каким образом работает подключаемое соединение с заданным условием на переход?
10. Перечислите и охарактеризуйте основные компоненты Stateflow?

Лабораторная работа №2. Построение дискретно-стохастических моделей посредством вероятностных автоматов и цепей Маркова

Цель работы: получение практических навыков построения дискретно-стохастических моделей.

Теоретическая часть

Перед выполнением лабораторной работы необходимо овладеть теоретическими знаниями в области построения конечных автоматов, которые содержатся в 6 теме теоретического материала.

1. Цепи Маркова

Маркова цепь *(Markov Chain) - марковский процесс с дискретным временем, заданный в измеримом пространстве.*

Марковские случайные процессы названы по имени выдающегося русского математика А.А. Маркова (1856-1922), впервые начавшего изучение вероятностной связи случайных величин и создавшего теорию, которую можно назвать "динамикой вероятностей".

В дальнейшем основы этой теории явились исходной базой общей теории случайных процессов, а также таких важных прикладных наук, как теория диффузионных процессов, теория надежности, теория массового обслуживания и т.д. В настоящее время теория марковских процессов и ее приложения широко применяются в самых различных областях.

Благодаря сравнительной простоте и наглядности математического аппарата, высокой достоверности и точности получаемых решений, особое внимание марковские процессы приобрели у специалистов, занимающихся исследованием операций и теорией принятия оптимальных решений.

Пример. **Бросание монеты.**

Предположим, что речь идет о последовательных бросаниях монеты при игре "в орлянку "; монета бросается в условные моменты времени t = 0, 1, ... и на каждом шаге игрок может выиграть ±1 с одинаковой вероятностью 1/2, таким образом в момент t его суммарный выигрыш есть случайная величина $\xi(t)$ с возможными значениями j = 0, ±1, ...

При условии, что $\xi(t) = k$, на следующем шаге выигрыш будет уже равен $\xi(t+1) = k \pm 1$, принимая указанные значения $j = k \pm 1$ с одинаковой вероятностью 1/2.

Условно можно сказать, что здесь с соответствующей вероятностью происходит переход из состояния $\xi(t) = k$ в состояние $\xi(t+1) = k \pm 1$.

2. Формулы и определения

Обобщая рассмотренный пример, можно представить себе "систему" со счетным числом возможных "фазовых" состояний, которая с течением дискретного времени t = 0, 1, ... случайно переходит из состояния в состояние.

Пусть $\xi(t)$ есть ее положение в момент t в результате цепочки случайных переходов:

$$\xi(0) - \xi(1) - ... - \xi(t) - ... ... \quad (1)$$

Формально обозначим все возможные состояния целыми i = 0, ±1, ... Предположим, что при известном состоянии $\xi(t) = k$ на следующем шаге система переходит в состояние $\xi(t+1) = j$ с условной вероятностью:

$$p_{kj} = P(\xi(t+1) = j|\xi(t) = k) ... \quad (2)$$

независимо от ее поведения в прошлом, точнее, независимо от цепочки переходов (1) до момента t:

$$P(\xi(t+1) = j|\xi(0) = i, ..., \xi(t) = k) = P(\xi(t+1) = j|\xi(t) = k) \text{ при всех } t, k, j \quad ... (3)$$

- марковское свойство.

Такую вероятностную схему называют ***однородной цепью Маркова со счетным числом состояний*** - ее однородность состоит в том, что определенные в (2) *переходные вероятности* p_{kj}, $\sum_j p_{kj} = 1$, $k = 0, \pm1, ...$, не зависят от времени, т.е. $P(\xi(t+1) = j|\xi(t) = k) = P_{ij}$ - *матрица вероятностей перехода* за один шаг не зависит от n.

Ясно, что P_{ij} - квадратная матрица с неотрицательными элементами и единичными суммами по строкам.

Такая матрица (конечная или бесконечная) называется **стохастической матрицей**. Любая стохастическая матрица может служить матрицей переходных вероятностей.

3. Пример «Доставка оборудования по округам»

Предположим, что некая фирма осуществляет доставку оборудования по Москве: в северный округ (обозначим A), южный (B) и центральный (C). Фирма имеет группу курьеров, которая обслуживает эти районы. Понятно, что для осуществления следующей доставки курьер едет в тот район, который на данный момент ему ближе. Статистически было определено следующее:

1) после осуществления доставки в A следующая доставка в 30 случаях осуществляется в A, в 30 случаях - в B и в 40 случаях - в C;

2) после осуществления доставки в B следующая доставка в 40 случаях осуществляется в A, в 40 случаях - в B и в 20 случаях - в C;

3) после осуществления доставки в C следующая доставка в 50 случаях осуществляется в A, в 30 случаях - в B и в 20 случаях - в C.

Таким образом, район следующей доставки определяется только предыдущей доставкой.

Матрица вероятностей перехода будет выглядеть следующим образом:

$$P = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.3 \\ 0.4 & 0.2 & 0.2 \end{pmatrix} \end{array}$$

Например, $p_{12} = 0.4$ - это вероятность того, что после доставки в район B следующая доставка будет производиться в районе A.

Допустим, что каждая доставка с последующим перемещением в следующий район занимает 15 минут. Тогда, в соответствии со статистическими данными, через 15 минут 30% из курьеров, находившихся в A, будут в A, 30% будут в B и 40% - в C.

Так как в следующий момент времени каждый из курьеров обязательно будет в одном из округов, то сумма по столбцам равна 1. И поскольку мы имеем дело с вероятностями, каждый элемент матрицы $0<p_{ij}<1$.

Наиболее важным обстоятельством, которое позволяет интерпретировать данную модель как цепь Маркова, является то, что местонахождение курьера в момент времени t+1 зависит **только** от местонахождения в момент времени t.

Теперь зададим простой вопрос: если курьер стартует из C, какова вероятность того, что осуществив две доставки, он будет в B, т.е. как можно достичь B в 2 шага? Итак, существует несколько путей из C в B за 2 шага:

1) сначала из C в C и потом из C в B;

2) C-->B и B-->B;

3) C-->A и A-->B.

Учитывая правило умножения независимых событий, получим, что искомая вероятность равна:

P = P(CA)*P(AB) + P(CB)*P(BB) + P(CC)*P(CB)

Подставляя числовые значения:

P = 0.5*0.3 + 0.3*0.4 + 0.2*0.3 = 0.33

Полученный результат говорит о том, что если курьер начал работу из C, то в 33 случаях из 100 он будет в B через две доставки.

Ясно, что вычисления просты, но если Вам необходимо определить вероятность через 5 или 15 доставок - это может занять довольно много времени.

Покажем более простой способ вычисления подобных вероятностей. Для того, чтобы получить вероятности перехода из различных состояний за 2 шага, возведем матрицу P в квадрат:

$$P^2 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.41 & 0.38 & 0.37 \\ 0.33 & 0.34 & 0.33 \\ 0.26 & 0.28 & 0.3 \end{pmatrix} \end{array}$$

Тогда элемент (2, 3) - это вероятность перехода из C в B за 2 шага, которая была получена выше другим способом. Заметим, что элементы в матрице P^2 также находятся в пределах от 0 до 1, и сумма по столбцам равна 1.

Т.о. если Вам необходимо определить вероятности перехода из C в B за 3 шага:

1 способ. p(CA)*P(AB) + p(CB)*P(BB) + p(CC)*P(CB) = 0.37*0.3 + 0.33*0.4 + 0.3*0.3 = 0.333, где p(CA) - вероятность перехода из C в A за 2 шага (т.е. это элемент (1, 3) матрицы P^2).

2 способ. Вычислить матрицу P^3:

$$P^3 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.385 & 0.39 & 0.393 \\ 0.333 & 0.334 & 0.333 \\ 0.282 & 0.276 & 0.274 \end{pmatrix} \end{array}$$

Матрица переходных вероятностей в 7 степени будет выглядеть следующим образом:

$$P^7 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 0.3888825 & 0.3888910 & 0.3888953 \\ 0.3333333 & 0.3333334 & 0.3333333 \\ 0.2777842 & 0.2777765 & 0.2777714 \end{pmatrix} \end{array}$$

Легко заметить, что элементы каждой строки стремятся к некоторым числам. Это говорит о том, что после достаточно большого количества доставок уж не имеет значение в каком округе курьер начал работу. Т.о. в конце недели приблизительно 38,9% будут в А, 33,3% будут в В и 27,8% будут в С.

Подобная сходимость гарантировано имеет место, если все элементы матрицы переходных вероятностей принадлежат интервалу (0, 1).

Общая постановка задачи

Построить вероятностный автомат и разработать имитационную модель, позволяющую отследить работу созданного автомата.

Список индивидуальных данных

Для разрабатываемой модели необходимо определить тип вероятностного автомата, формализовать его в виде графа и матриц.

1. Смоделировать работу пешеходного перехода, оснащенного светофором с вызовом.

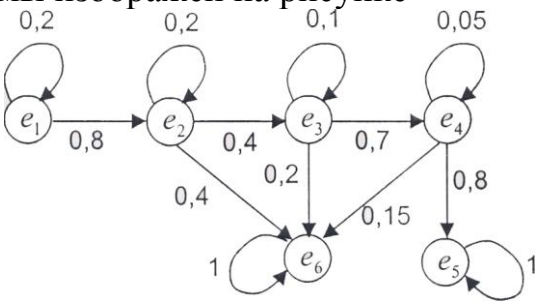
2. Смоделировать работу центрального процессора мультипрограммной компьютерной системы в любой момент времени выполняющую либо приоритетную программу, либо фоновую программу, либо находится в состоянии ожидания. Продолжительность нахождения системы в каждом состоянии кратна длительности шага Δt. Исходное состояние простой процессора. Матрица переходов имеет вид:

$$|\mathbf{p}| = \begin{vmatrix} \mathbf{0,7} & \mathbf{0,2} & \mathbf{0,1} \\ \mathbf{0,8} & \mathbf{0,1} & \mathbf{0,1} \\ \mathbf{0,8} & \mathbf{0,05} & \mathbf{0,15} \end{vmatrix}$$

Пусть 1 – состояние обслуживания основной программы; 2 – состояние обслуживания фоновой программы; 3 – состояние простоя.

3. Посетитель банка с намерением получить кредит проходит ряд проверок (состояний): 1 – оформление документов; 2 – кредитная история; 3

– возвратность; 4 – платежеспособность. По результатам проверки возможны два исхода: отказ в выдаче кредита (6) и получение кредита (5). Граф этой системы изображен на рисунке



Требуется:

а) описать данный процесс как Марковскую цепь и построить переходную матрицу;

б) найти среднее время получения положительного и отрицательного результата.

4. Рассматриваются следующие состояния телефона-автомата: телефон свободен, телефон занят и нет очереди, телефон занят и в очереди один человек, телефон занят и в очереди 2 человека. Предполагается, что третьим в очередь никто не встает, предпочитая искать другой телефон. В каждую минуту с вероятностью $\frac{1}{11}$ может подойти один человек (больше одного подойти не может), а с вероятностью $\frac{1}{5}$ разговор в данную минуту заканчивается. С какой вероятностью через 3 минуты в очереди будет один человек, если в настоящий момент времени телефон свободен?

5. Автомашина может находиться в одном из четырех состояний: - исправна, - неисправна, -осматривается, - ремонтируется, - списана. Если машина исправна, то с вероятностью 0,8 она может сломаться, если машина неисправна, то она с вероятностью 0,7 ремонтируется или с вероятностью 0,3 списывается; если же машина ремонтируется, то она с вероятностью 0,6 становится исправной, либо с вероятностью 0,4 продолжает ремонтироваться. Остальные переходы считать невозможными. Найти вероятность того, что машина будет исправна в субботу, если известно, что она была исправна в среду.

6. Постороить имитационную модель в форме вероятностного автомата, которая моделирует перемещение точки в двухмерной системе координат случайным образом. В момент времени ноль точка находится в начале координат и остается там в течение одной секунды. Возможные перемещение точки: влево, вправо, вверх, вниз. При этом из каждого состояния возможны только три перехода, например, из состояния влево можно перейти вверх, вправо и влево (определяется самостоятельно). Шаг дискретизации выбирается индивидуально, чтобы можно наблюдать процес

имитации. Вероятности перехода из состояния в состояние определить самостоятельно.

7. Производятся поочередные выстрелы по мишени, которая может находиться в четырех состояниях:

- S1 —невредима;
- S2 —незначительно повреждена;
- S3 —получила существенные повреждения;
- S4 —полностью поражена.

Необходимо построить вероятностный автомат. При этом вероятности попадания в мишень определяются самостоятельно, формируя при этом индивидуальную модель стрелка.

8. Устройство S состоит из двух узлов A и B, каждый из которых в процессе работы может отказывать. Возможны следующие состояния системы:

- S1 – оба узла работают;
- S2 – узел A отказал, B работает;
- S3 – узел B отказал, A работает;
- S4 – оба узла отказали.

Вероятности отказа узлов A и B определяются соответственно 0,6 и 0,4.

9. Система S представляет собой устройство, состоящее из двух узлов A и B, каждый из которых может в какой-то момент времени отказать. Отказавший узел немедленно начинает восстанавливаться. Возможны такие состояния системы:

- S1 - оба узла работают;
- S2 - узел A восстанавливается, узел B - работает;
- S3 - узел A работает, узел B восстанавливается;
- S4 - оба узла восстанавливаются.

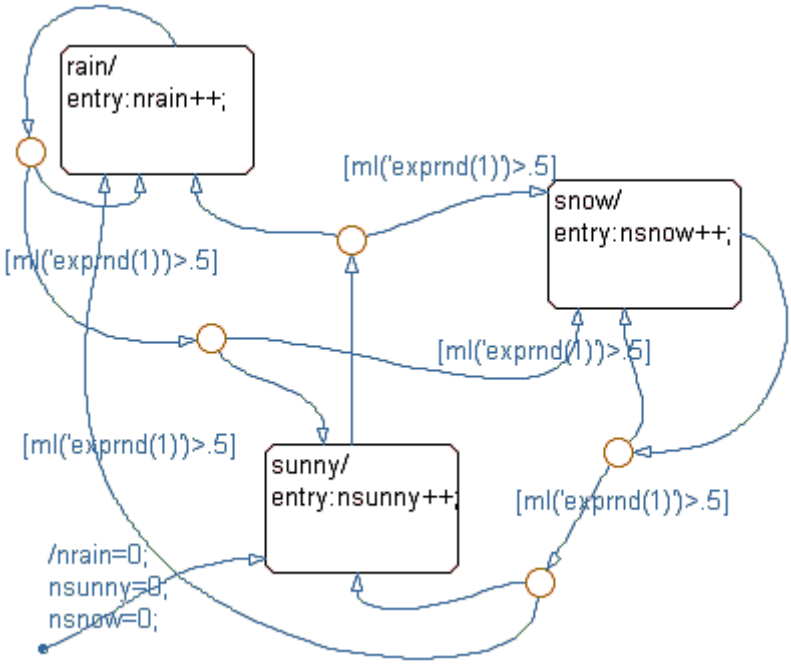
Пример выполнения работы

Модель эргодической цепи Маркова

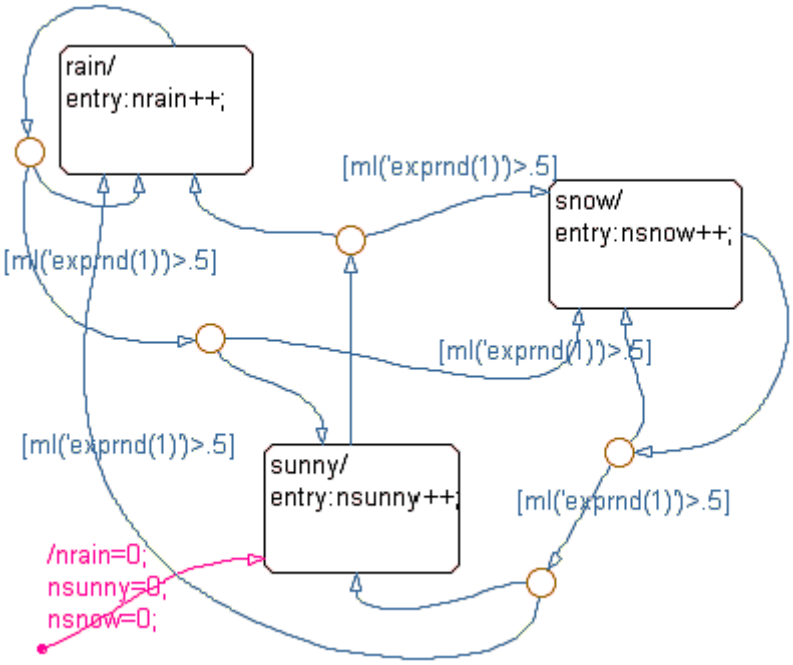
Пример из книги Кемени Дж., Снелл Дж. Кибернетическое моделирование. Некоторые приложения. М.Советское радио, 1972,192 с.

На Земле Оз погода бывает всего трех типов: дождь, солнечно или снег. Если сегодня солнечный день, то завтра будет дождь или снег с одинаковой вероятностью. Если сегодня-дождь (или снег), то половина шансов за то, что такая же погода будет завтра. Если же происходит изменение, то только в половине случаев оно приводит к солнечному дню.

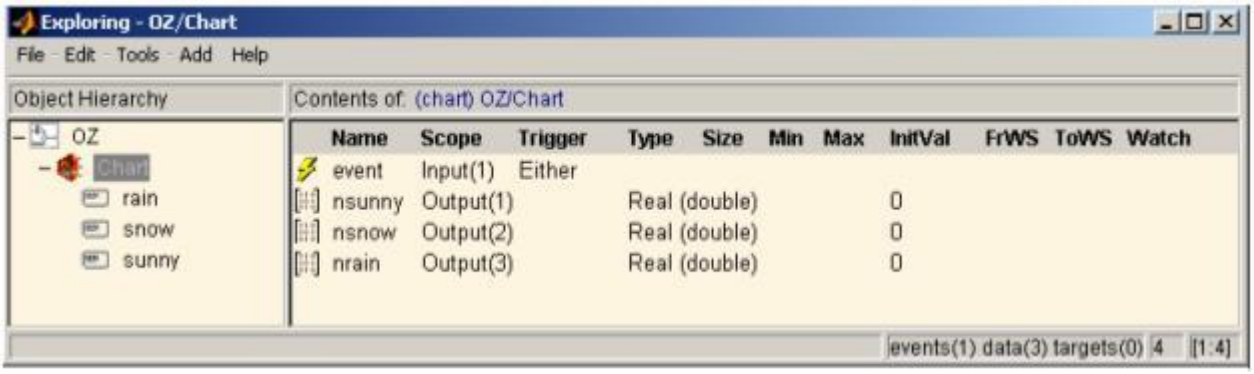
Решение. Образуем эргодическую цепь Маркова с тремя состояниями rain, sunny и snow для дождя, солнца и снега соответственно. Ее моделью будет Stateflow-диаграмма.



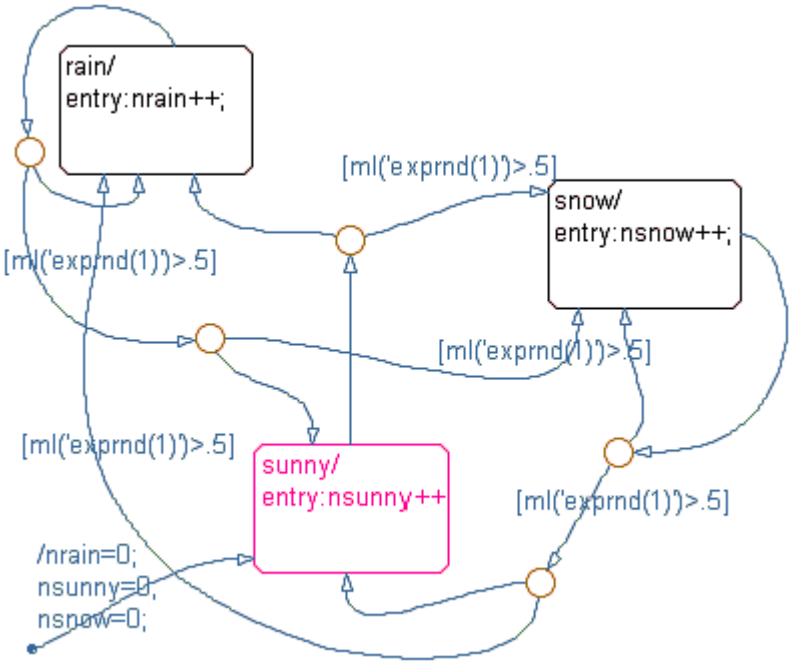
Начальное состояние - sunny, о чем свидетельствует наличие графического объекта переход по умолчанию (Default transition) к состоянию sunny. Этот переход сопровождается действием перехода (Transition action) /nrain=0;nsunny=0;nsnow=0. Это действие устанавливает в ноль счетчики количества дождливых, солнечных дней и дней, когда идет снег.



Заметим, что данном случае это действие является избыточным, так как начальные значения этих переменных равны нулю по умолчанию. В этом нетрудно убедиться, открыв проводник Stateflow Explorer и просмотрев графу InitVal.



При входе в это состояние выполняется действие nsunny++, т.е. количество солнечных дней увеличивается на единицу.



Следующее событие event (смена суток организована в модели при помощи генератора прямоугольных импульсов Pulse Generator) переводит диаграмму в соединяемое подключение Connective Junction, откуда с вероятностью 0.5 диаграмма переходит в состояние snow и с вероятностью 0.5 - в состояние rain. Вероятностный переход основан на использовании условия ml('exprnd(1)')>.5 (вызов MATLAB-функции exprnd(1), т.е. генерация случайного числа из диапазона (0,1) и сравнение этого числа с числом 0.5). Остальные переходы организованы аналогичным образом в соответствие с логикой, описанной в задаче. Результат работы модели на протяжении 10 лет модельного времени, как это следует из рисунка, дал 728 солнечных, 1514 снежных и 1408 дождливых дней. Аналитическое решение дает вероятность для солнечной погоды 1/5, а для снега и дождя - 2/5.

Контрольные вопросы к защите

1. Что такое вероятностный автомат?
2. В чем заключается особенностей цепей Маркова?

3. Каким образом при помощи диаграмм stateflow организуется построение случайных переходов?

4. Какие генераторы из пакета Simulink возможно использовать для получения случайных величин?

5. Приведите собственный пример, процесс работы которого описывается при помощи вероятностного автомата.

Лабораторная работа №3. Построение непрерывно-стохастических моделей. Моделирование систем массового обслуживания в Simulink+SimEvents

Цель работы: овладеть практическими навыками моделирования систем массового обслуживания

Теоретическая часть

Перед выполнением лабораторной работы необходимо овладеть теоретическими знаниями в области построения конечных автоматов, которые содержатся в темах 7 и 8 теоретического материала.

Построение непрерывно-стохастических моделей будет осуществляться в среде MatLab, а именно при помощи среды разработки моделей Simulink и SimEvents, который является интерактивным инструментом разработки в области моделирования систем массового обслуживания.

1. Системы массового обслуживания.

Для описания объектов реального мира, функционирующих в условиях действия случайных факторов на практике, часто используется класс математических моделей, называемых системами массового обслуживания (СМО). Функционирование этих объектов носит характер обслуживания поступающих в систему заявок или клиентов.

Для выполнения совокупности действий, включаемых в понятие “обслуживание”, система располагает наборами оборудования, называемых обслуживающими каналами или линиями. Например, в телефонных сетях заявками являются вызовы, возникающие на АТС в момент снятия абонентом телефонной трубки, а под обслуживанием понимается предоставление абоненту свободной линии для разговора; на бензозаправочной станции в качестве носителей заявок клиентов выступают автомобили, прибывающие на заправку, а обслуживающими каналами являются заправочные колонки. Аналогичные ситуации наблюдаются в системах посадки самолетов, разгрузки судов, в парикмахерских, магазинах и т.д. Любое производство можно представить в виде последовательности систем обслуживания.

Обычно предполагается, что заявки на обслуживание образуют поток – последовательность заявок со специальным чередованием моментов их появления во времени. Общее понятие потока основано на предположении,

что интересующее нас событие будет происходить в заранее неизвестные моменты времени t_1, t_2, … , t_k (обычно их считают случайными).

Математическое моделирование систем массового обслуживания складывается из моделирования потока заявок и моделирования процесса функционирования совокупности обслуживающих каналов.

Для описания СМО необходимо задать:

1) входящий поток требований или заявок, которые поступают на обслуживание;

2) порядок постановки заявки в очередь и выбора из нее;

3) правило, по которому осуществляется обслуживание;

Для описания входящего потока требований необходимо описать моменты времени их поступления в систему и количество требований, которое поступило одновременно. Закон поступления может быть детерминированный (например, одно требование поступает каждые 5 мин) или вероятностный (требование может появиться с равной вероятностью в интервале 5±2 мин). В общем случае входящий поток требований описывается распределением вероятностей интервалов времени между соседними требованиями.

Правила обслуживания характеризуются длительностью обслуживания (распределением времени обслуживания), количеством требований, которые поступили одновременно и дисциплиной обслуживания. Для характеристики свойств линии задается величина $\tau_з$ – длительность обслуживания заявки или время занятости линии, как случайная величина с заданным законом распределения.

В общем случае обслуживающая система может состоять из n линий, способных одновременно и независимо друг от друга обслуживать заявки. В любой момент времени линия находится в одном из двух состояний: свободном или занятом. Системы, обслуживаемые с помощью одной линии, называются одноканальными системами. Системы с несколькими линиями – многоканальными.

Относительно порядка принятия заявок в том случае, когда в системе имеется очередь заявок, используются следующие правила.

1. Заявки принимаются к обслуживанию в порядке очереди. Освободившаяся линия приступает к обслуживанию той заявки, которая ранее других поступила в систему. Такую дисциплину называют “раньше поступил – раньше обслужился” (в англоязычной литературе FIFO – First In - First Out).

2. Заявки принимаются к обслуживанию по минимальному времени получения отказа. Освободившаяся линия приступает к обслуживанию той заявки, которая в кратчайшее время может получить отказ.

3. Заявки принимаются к обслуживанию в случайном порядке в соответствии с заданными вероятностями (RANDOM).

Реальный процесс массового обслуживания состоит из последовательности фаз обслуживания, выполняемой различными устройствами. При этом следующее устройство приступает к обслуживанию заявки тогда, когда работа предыдущего с данной заявкой завершена.

Пример. Многофазное обслуживание покупателей.

1. Выбор товара и оформление товарного чека
2. Оплата в кассе.
3. Получение товара.

Пример. Технологический процесс.

Изделие может поступить на обработку станком (n+1)-й фазы лишь тогда, когда его обработка на станке n-й фазы закончена.

Система массового обслуживания – математический (абстрактный) объект, содержащий один или несколько приборов П (каналов), обслуживающих заявки З, поступающие в систему, и накопитель Н, в котором находятся заявки, образующие очередь О и ожидающие обслуживания (рисунок1).

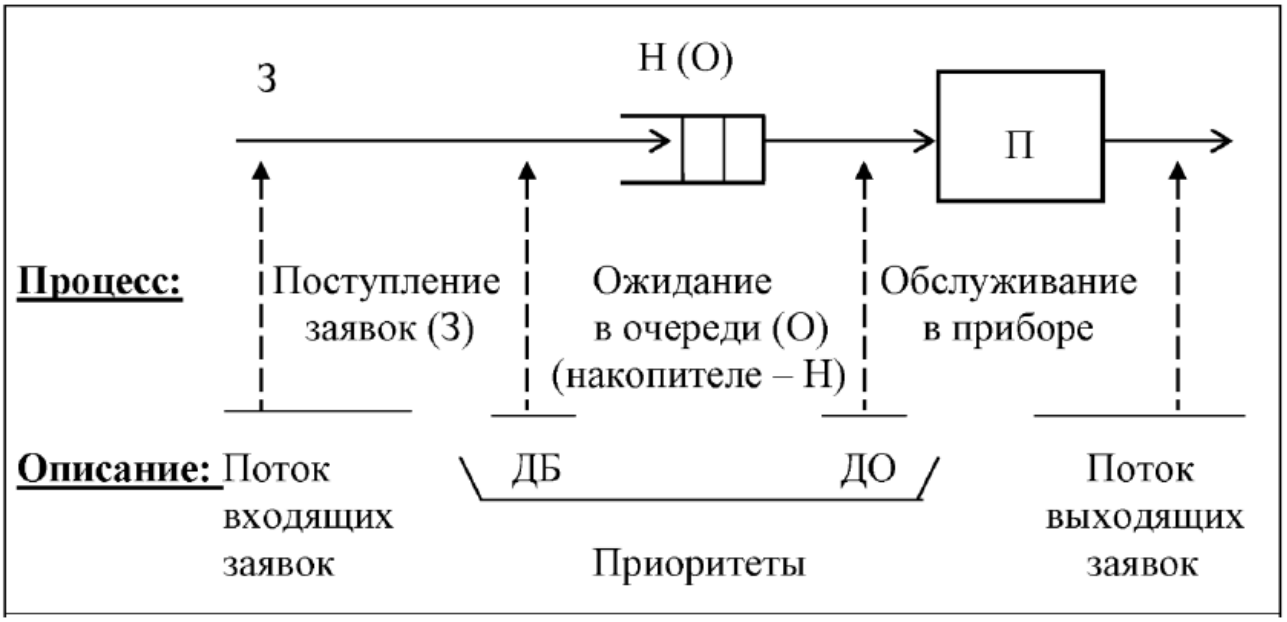


Рисунок 1 – Система массового обслуживания

Имитационная модель СМО – это модель, отражающая поведение системы и изменения ее состояния во времени при заданных потоках заявок, поступающих на входы системы. Выходными параметрами являются величины, характеризующие качество функционирования системы, например, такие как:

- коэффициенты использования каналов обслуживания;
- максимальная и средняя длина очередей в системе;
- время нахождения заявок в очередях и каналах обслуживания.

2. Система дискретного событийного моделирования SimEvents

Для реализации дискретно-событийного моделирования в среде **Simulink** используется компонента **SimEvents**. С помощью **SimEvents** можно моделировать и проектировать распределенные системы управления, аппаратные конфигурации, сети передачи и сбора информации для аэрокосмических, автомобильных, электронных и других приложений. Можно также моделировать событийно-управляемые процессы, такие например, как стадии производственного процесса для определения потребностей в ресурсах и оценки узких мест производства.

SimEvents и **Simulink** создают интегрированную среду для моделирования гибридных динамических систем, содержащих непрерывные компоненты и компоненты с дискретными событиями и дискретным временем.

Что такое моделирование дискретных событий? При моделировании дискретных событий или моделировании на основе событий состояния системы изменяются в результате наступления асинхронных дискретных инцидентов, которые называются событиями. В противоположность этому моделирование, основанное исключительно на дифференциальных или разностных уравнениях, в которых время является независимой переменной, – моделирование на основе времени, потому что состояния системы зависят от времени. Simulink предназначен для моделирования на основе времени, в то время как SimEvents создавался для моделирования дискретных событий. Выбор типа моделирования зависит как от самого изучаемого явления, так и от способа, которым его предполагается изучать.

При дискретно-событийном моделировании используется понятие **сущности** (entity). Сущности могут перемешаться через **сети очередей** (**queues**), **серверов (servers**) и **переключателей (switches)**, управляемых дискретными событиями, в процессе моделирования.

Графические блоки **SimEvents** могут представлять компонент, который обрабатывает сущности, но сами сущности не имеют графического представления. Примеры сущностей приведены в следующей таблице

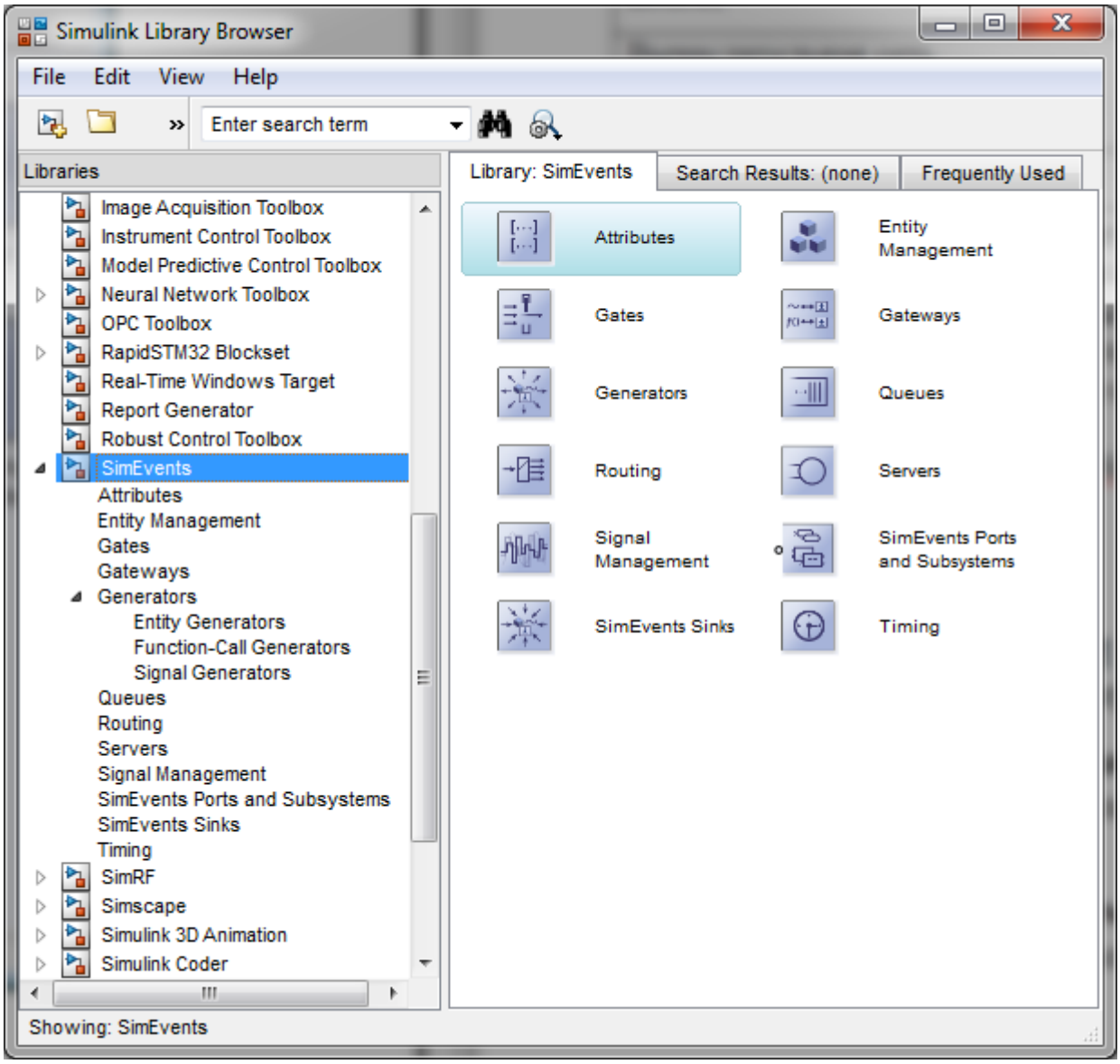
Таблица 1. Примеры сущностей.

Приложение	Сущность
Аэропорт с очередью на взлетной полосе	Самолет, ждущий доступа к взлетной полосе
Вычислительная сеть	Пакеты, фреймы и сообщения для передачи
Конвейер для сборки агрегатов	Сборочные единицы

Событие (event) – это мгновенное дискретное явление, которое изменяет переменную состояния, выход и/или является причиной появления других событий. Примерами событий в модели SimEvents являются:

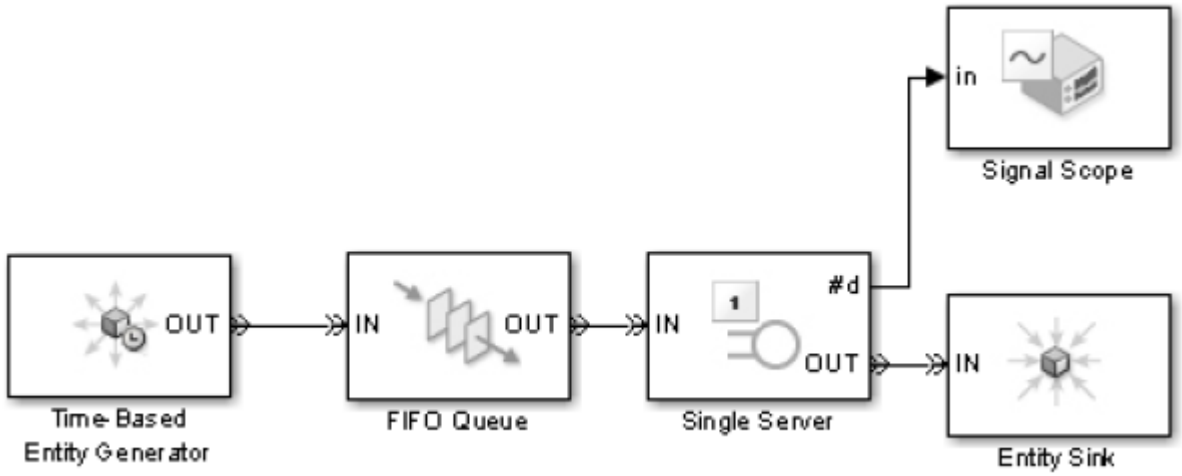
- перемещение сущности от одного блока к другому;
- завершение обслуживания сущности в сервере.

Для открытия библиотек SimEvents наберите в командной строке simeventslib.



В появившемся окне отображены иконки, представляющие все разделы библиотеки блоков simevents.

Выберем из раздела **Generators** (подраздел **Entity Generators** – генераторы сущностей) блок **Time-Based Entity Generator**, из раздела **Queues** блок **FIFO Queue**, из раздела **Servers** – блок **Single Server**, из раздела **SimEvents Sinks** – блок **Entity Sink** и блок **Signal Scope**. В результате в окне модели будет отображаться следующая картина.



Представленная на рисунке модель содержит блоки для всех ключевых процессов моделирования: **генерации сущностей** (блок **Time-Based Entity Generator**), **хранения сущностей в очереди** (блок – **FIFO Queue**), **обслуживания сущностей** (блок – **Single Server**) и **отображения информации о ходе моделирования** (блок – **Signal Scope**).

При дискретно-событийном моделировании очереди (queues) хранят сущности в течение некоторого интервала времени, который заранее неизвестен. Очередь пытается выпустить сущность как можно быстрее, но успех операции зависит от возможности следующего блока принять новую сущность. Повседневным примером очереди является ситуация, когда вы стоите в очереди в кассу в магазине для обслуживания и не можете заранее определить сколько придется ждать обслуживания.

Отличительными свойствами очереди являются:

- **емкость (capacity**), то есть максимальное количество сущностей, которые очередь может хранить одновременно;

- **дисциплина очереди,** определяющая какая из сущностей покинет очередь первой, если она хранит несколько сущностей.

Блоки очередей находятся в разделе **Queues** библиотеки SimEvents.

Сервер (server) в дискретно событийном моделировании хранит сущности в течение некоторого промежутка времени, называемого **временем обслуживания (service time)** и затем пытается выпустить сущность. Во время периода обслуживания говорят, что блок-сервер обслуживает сущность.

Время обслуживания для каждой сущности вычисляется в момент ее прибытия в сервер. В отличие от этого время хранения блока в очереди принципиально заранее никогда неизвестно. Однако, если следующий блок не принимает сущность, которую уже обслужил сервер, то сервер должен хранить сущность дальше.

Отличительными свойствами сервера являются:

- число сущностей, которые можно обслужить одновременно. Это число может быть конечным или бесконечным;

- характеристиками или методами вычисления времен обслуживания поступающих сущностей;

- разрешает ли сервер прибывающим сущностям занимать сервер при наличии других сущностей, хранящихся в сервере. При отсутствии этого свойства сервер с конечной емкостью, если он полон, не принимает к обработке новые прибывающие сущности.

Блоки серверов находятся в разделе **Servers** библиотеки SimEvents.

3. Описание некоторых блоков библиотеки SimEvents

Time-Based Entity Generator (Раздел Generators/Entity Generator.) – блок генерирует сущности в моменты времени, определяемые входным сигналом или статистическим распределением.

Порядок генерирования определяется значением параметра Generate entities upon:

Intergeneration time from dialog – моменты времени генерации определяются в зависимости от параметров в диалоговых полях блока;

Тип распределения моментов генерации определяется полем Distribution. Возможные значения:

- Constant (постоянное) – постоянное время между генерируемыми событиями задается в поле Period.

- Uniform (равномерное) – случайное равномерное распределение задается диапазоном в полях Minimum и Maximum.

- Exponential – экспоненциальное распределение задается параметром Mean (среднее).

При установке параметра распределения в значения Uniform или Exponential в диалоговом окне имеется также параметр Initial seed, определяющий генерируемый набор случайных чисел. Для фиксированного значения этого параметра случайная последовательность при следующем запуске модели повторится. Типично, значение этого параметра устанавливается в большое (например, пятизначное) нечетное число.

Intergeneration time from port t – моменты времени генерации определяются через сигнальный порт t, информация с которого считывается при старте моделирования и в каждый момент генерации новой сущности. Сигнал должен быть событийным (event-based). Если выставлено это значение, то у блока появляется дополнительный сигнальный порт t.

Для сбора статистики блока нужно отметить галочкой нужные сигналы на вкладке Statistics в панели свойств блока. Если отмечен соответствующий пункт, то у блока появляется новый сигнальный выходной порт, на который выводится следующая статистическая информация в соответствии:

- #d – число сущностей покинувших блок после начала моделирования;

- w – среднее время между генерациями сущностей.

FIFO Queue – блок одновременно хранит до N сущностей, где N – значение параметра Capacity (Емкость). Блок пытается выпустить сущность через выходной порт OUT, однако если порт OUT блокирован, то сущность остается в блоке. Если в блоке хранятся несколько сущностей, то сущности покидают блок в соответствии с дисциплиной первый вошел – первый вышел (first in – first out (FIFO)). Если блок уже хранит N сущностей, то входной порт IN блока не доступен.

Для сбора статистики блока нужно отметить галочкой нужные сигналы на вкладке Statistics в панели свойств блока. Если отмечен соответствующий пункт, то у блока появляется новый сигнальный выходной порт, на который выводится следующая статистическая информация:

- #d – число сущностей покинувших блок через порт OUT после начала моделирования;
- #n – число сущностей в очереди;
- w – среднее время ожидания в этом блоке для всех сущностей, покинувших блок через любой порт;
- len – среднее число сущностей в очереди по времени, то есть средний по времени сигнал #n.

Single Server – блок обслуживает одновременно одну сущность за некоторый интервал времени и затем пытается выпустить сущность через выходной порт OUT. Если порт OUT блокирован, то сущность остается в блоке до тех пор пока выходной порт не разблокируется.

Время обслуживания (Service Time) определяется через параметры, атрибуты или сигналы в зависимости от значения параметра Service Time From. Блок определяет время обслуживания сущности при ее поступлении. Времена обслуживания определяются в секундах.

Значения параметра Service Time From:

- Dialog – значения времени обслуживания задаются в поле параметра Service Time;
- Attribute – значения времени обслуживания задаются атрибутом, имя которого указано в поле параметра Attribute name;
- Signal port t – значения времени обслуживания задаются через сигнальный порт t. Сигнал должен быть событийным (event-based). Если выставлено это значение, то у блока появляется дополнительный сигнальный порт t.

Для сбора статистики блока нужно отметить галочкой нужные сигналы на вкладке Statistics в панели свойств блока. Если отмечен соответствующий

пункт, то у блока появляется новый сигнальный выходной порт, на который выводится следующая статистическая информация:

- #d – число сущностей покинувших блок через порт OUT после начала моделирования;

- #n – число сущностей в сервере, 0 или 1;

- w – среднее время ожидания в этом блоке для всех сущностей, покинувших блок;

- util – утилизация сервера, то есть доля времени моделирования, использованная на хранение сущностей. В начале моделирования утилизация равна 0 или 1 в зависимости от того хранит ли сервер сущность.

Для завершения путей сущностей используется блок Entity Sink.

Параметры блока:

Input port available for entity arrivals – если выбран этот параметр, то блок принимает прибывающие сущности, в противном случае – блок сущности не принимает, а при попытке прибытия сущности выдается сообщение об ошибке.

Report number of entities arrived, #a – при выборе параметра у блока появляется сигнальный порт #a, на котором после каждого прибытия сущности выдается информация о количестве принятых блоком сущностей. Начальное значение сигнала после начала моделирования до первого обновления блока равно 0.

Для вывода результатов моделирования в виде графиков используется блок Signal Scope из раздела библиотеки SimEvents Sinks.

Блок создает график, используя данные из событийного сигнала. Данные, откладываемые по вертикальной оси, берутся из сигнала, связанного с входным сигнальным портом in.

Параметры блока:

Plot type – тип графика:

Stair – ступенчатый;

Continuous – непрерывный. Тип графика не меняет дискретный сигнал на непрерывной, а только определяет способ отображения графика.

Данные для горизонтальной оси определяются параметром X value from: Event time – выводится график данных с порта in по времени моделирования; Index – выводится график данных с порта in по их индексу, т.е. первое значение имеет индекс 1, второе – 2 и т.д.

Общая постановка задачи

Произвести имитационное моделирование системы массового обслуживания, согласно Вашего варианта.

Для каждого из вариантов определить:

a) к какому классу относится объект СМО;

b)число каналов n;

c)длину очереди m;

d)интенсивность потока заявок λ;

e)интенсивность обслуживания одним каналом μ;

f) количество всех состояний объекта СМО.

В ответах указать значения по каждому пункту, используя следующие сокращения и размерности:

a) ОО – одноканальная с отказами; МО – многоканальная с отказами; ОЖО – одноканальная с ожиданием с ограниченной очередью; ОЖН - одноканальная с ожиданием с неограниченной очередью; МЖО – многоканальная с ожиданием с ограниченной очередью; МЖН - многоканальная с ожиданием с неограниченной очередью;

b) n=… (единиц);

c) m =… (единиц);

d) λ =xxx/xxx (единиц /мин);

e) μ =xxx/xxx (единиц /мин);

f) (единиц).

Для каждой из указанных в вариантах СМО интенсивность потока заявок равна λ и интенсивность обслуживания одним каналом μ . Требуется: - составить перечень возможных состояний; - построить граф состояний по схеме "гибели и размножения". В ответе указать для каждой задачи:

- количество состояний системы;
- интенсивность перехода из последнего состояния в предпоследнее.

Список индивидуальных данных

1. Дежурный по администрации города имеет пять телефонов. Телефонные звонки поступают с интенсивностью 90 заявок в час, средняя продолжительность разговора составляет 2 мин.

2. На стоянке автомобилей возле магазина имеются 3 места, каждое из которых отводится под один автомобиль. Автомобили прибывают на стоянку с интенсивностью 20 автомобилей в час. Продолжительность пребывания автомобилей на стоянке составляет в среднем 15 мин. Стоянка на проезжей части не разрешается.

3. АТС предприятия обеспечивает не более 5 переговоров одновременно. Средняя продолжительность разговоров составляет 1 мин. На станцию поступает в среднем 10 вызовов в сек.

4. В грузовой речной порт поступает в среднем 6 сухогрузов в сутки. В порту имеются 3 крана, каждый из которых обслуживает 1 сухогруз в

среднем за 8 ч. Краны работают круглосуточно. Ожидающие обслуживания сухогрузы стоят на рейде.

5. В службе «Скорой помощи» поселка круглосуточно дежурят 3 диспетчера, обслуживающие 3 телефонных аппарата. Если заявка на вызов врача к больному поступает, когда диспетчеры заняты, то абонент получает отказ. Поток заявок составляет 4 вызова в минуту. Оформление заявки длится в среднем 1,5 мин.

6. Салон-парикмахерская имеет 4 мастера. Входящий поток посетителей имеет интенсивность 5 человек в час. Среднее время обслуживания одного клиента составляет 40 мин. Длина очереди на обслуживание считается неограниченной.

7. На автозаправочной станции установлены 2 колонки для выдачи бензина. Около станции находится площадка на 2 автомашины для ожидания заправки. На станцию прибывает в среднем одна машина в 3 мин. Среднее время обслуживания одной машины составляет 2 мин.

8. На вокзале в мастерской бытового обслуживания работают три мастера. Если клиент заходит в мастерскую, когда все мастера заняты, то он уходит из мастерской, не ожидая обслуживания. Среднее число клиентов, обращающихся в мастерскую за 1 ч, равно 20. Среднее время, которое затрачивает мастер на обслуживание одного клиента, равно 6 мин.

9. АТС поселка обеспечивает не более 5 переговоров одновременно. Время переговоров в среднем составляет около 3 мин. Вызовы на станцию поступают в среднем через 2 мин.

10. На автозаправочной станции (АЗС) имеются 3 колонки. Площадка при станции, на которой машины ожидают заправку, может вместить не более одной машины, и если она занята, то очередная машина, прибывшая к станции, в очередь не становится, а проезжает на соседнюю станцию. В среднем машины прибывают на станцию каждые 2 мин. Процесс заправки одной машины продолжается в среднем 2,5 мин.

11. В небольшом магазине покупателей обслуживают два продавца. Среднее время обслуживания одного покупателя – 4 мин. Интенсивность потока покупателей – 3 человека в минуту. Вместимость магазина такова, что одновременно в нем в очереди могут находиться не более 5 человек. Покупатель, пришедший в переполненный магазин, когда в очереди уже стоят 5 человек, не ждет снаружи и уходит.

12. Железнодорожную станцию дачного поселка обслуживает касса с двумя окнами. В выходные дни, когда население активно пользуется железной дорогой, интенсивность потока пассажиров составляет 0,9 чел./мин. Кассир затрачивает на обслуживание пассажира в среднем 2 мин.

Пример выполнения работы

Требуется промоделировать работу парикмахерской. Интервалы прихода клиентов в парикмахерскую с одним креслом распределены равномерно на интервале 18±6мин. Время стрижки также распределено

равномерно на интервале 16±4мин. Клиенты приходят в парикмахерскую, стригутся в порядке очереди: «первым пришел – первым обслужился». Промоделируйте работу парикмахерской в течение 8 часов.

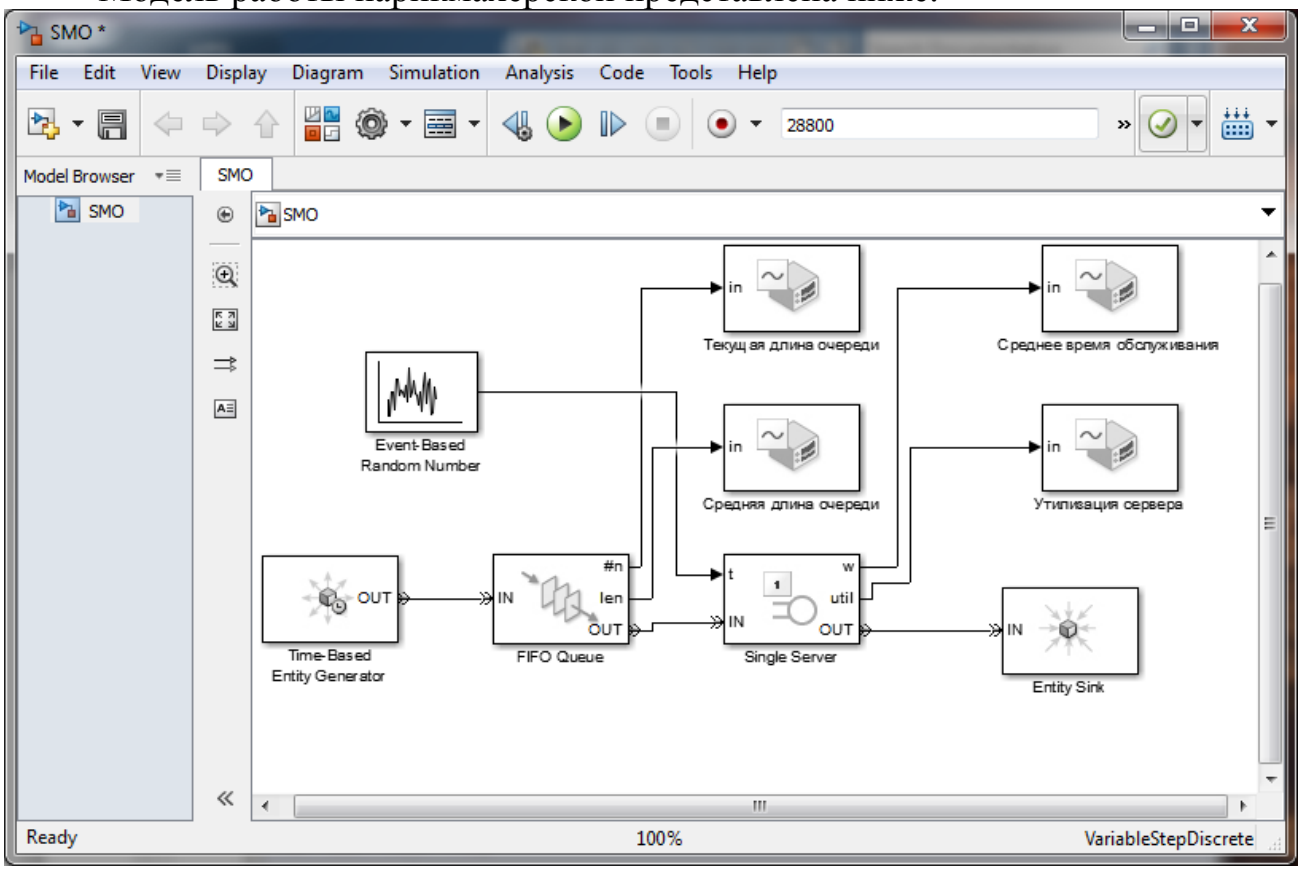
Требуется определить параметры функционирования парикмахерской:

1. коэффициент загрузки парикмахера;
2. максимальное, среднее и текущее число посетителей в очереди;
3. среднее время обслуживания.

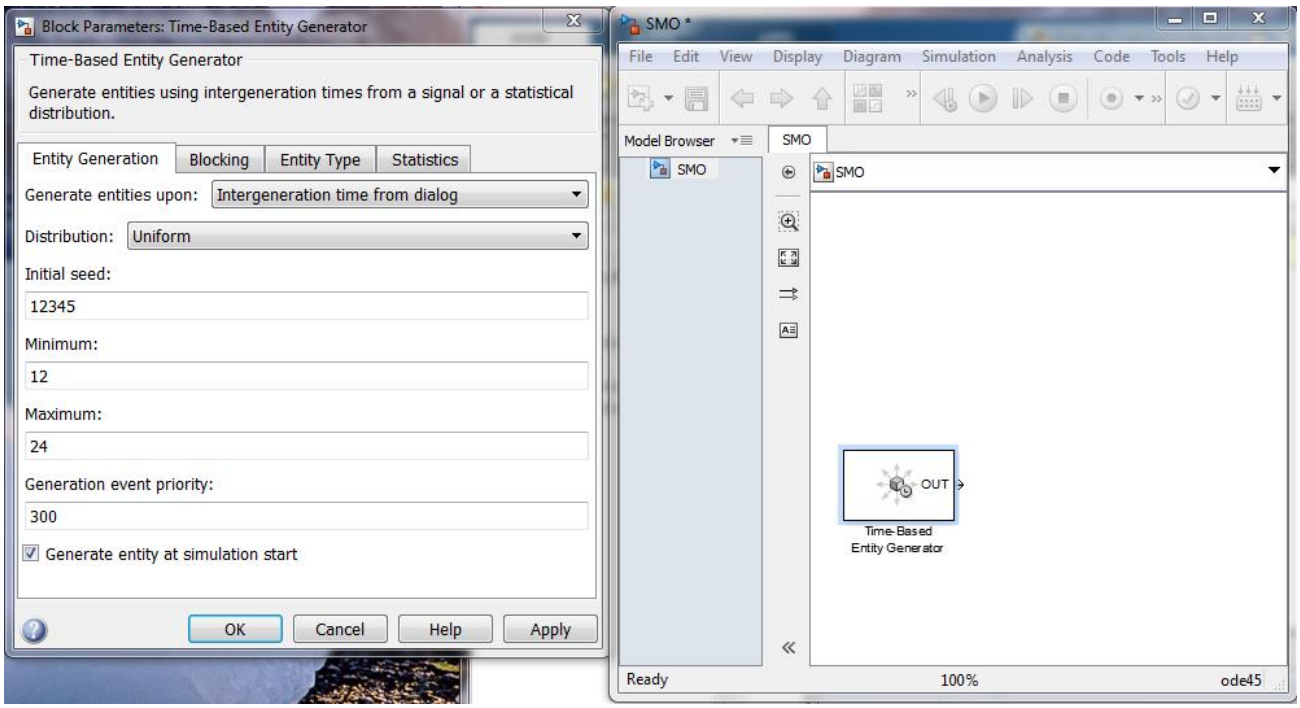
Построение модели. Порядок блоков в модели соответствует порядку фаз, в которых клиент оказывается при движении в реальной системе:

1) клиент приходит;
2) если необходимо, ждет своей очереди;
3) садится в кресло парикмахера;
4) парикмахер обслуживает клиента;
5) клиент уходит из парикмахерской.

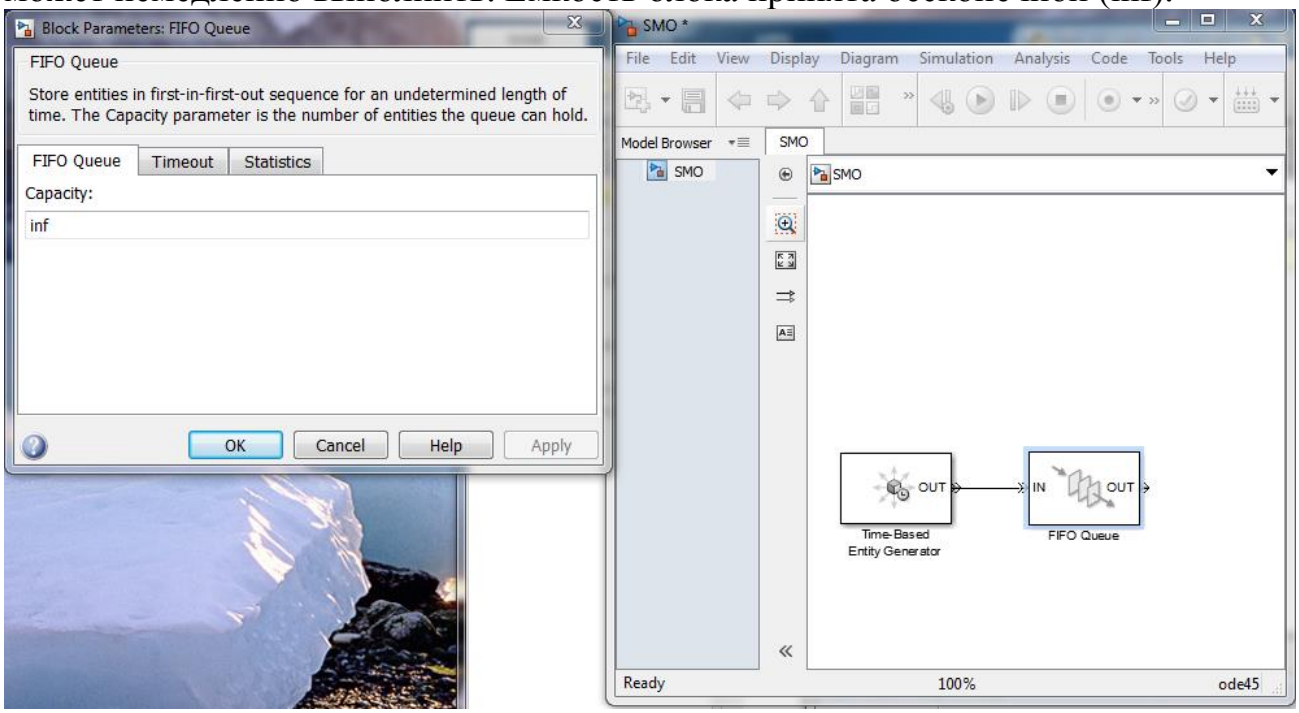
Модель работы парикмахерской представлена ниже.



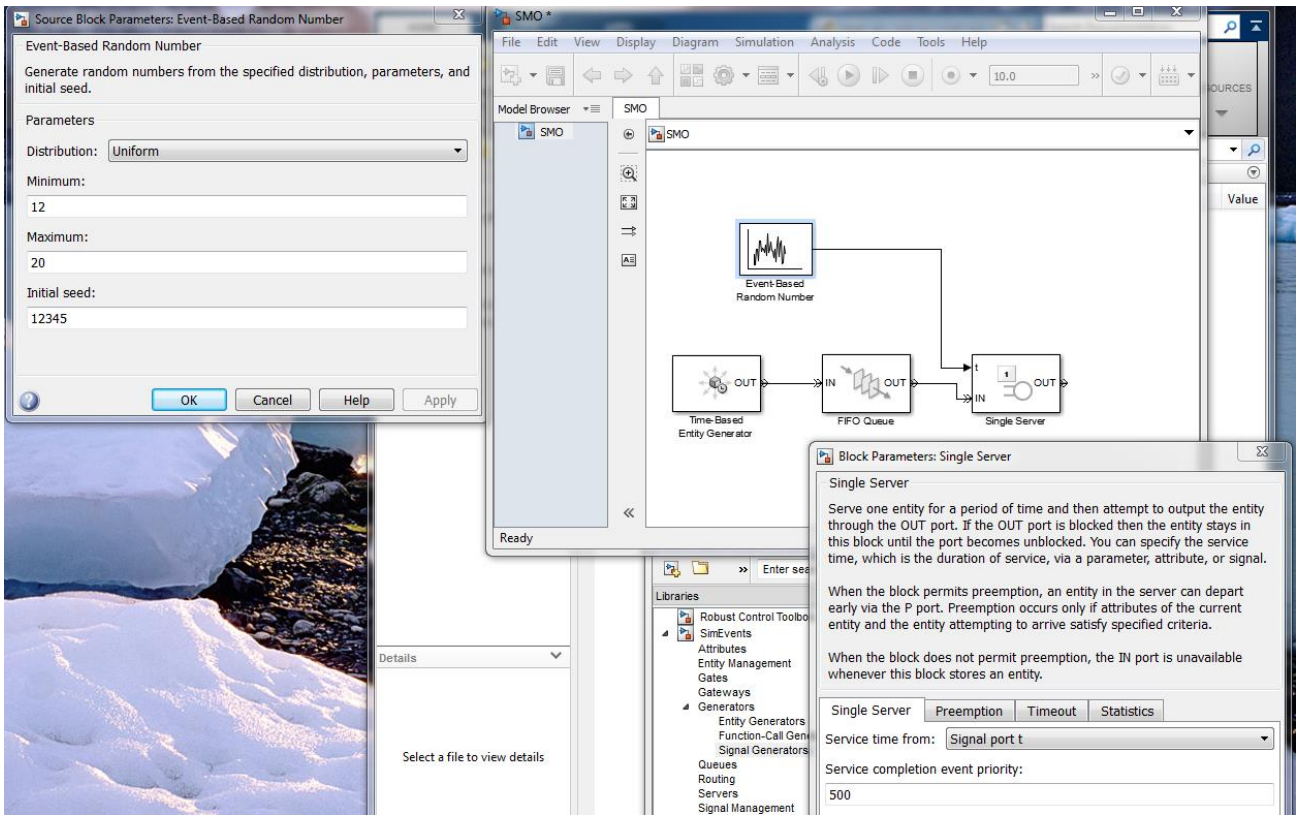
В случайные моменты времени блок **Time-Based Entity Generator** генерирует события, моделирующие приход клиентов в парикмахерскую. В параметрах блока указан тип распределения **Uniform (равномерный)**, параметры: **Minimum – 12, Maximum 24**.



Блок **FIFO Queue** сохраняет заявки (клиентов), которые парикмахер не может немедленно выполнить. Емкость блока принята бесконечной (inf).



Блок **Single Server** моделирует обслуживание клиента парикмахером. Этот блок может выполнить не более одной работы одновременно, тем самым ограничивая обработку новых работ. Времена обслуживания задаются через сигнальный порт t (**Service time from – Signal port t**) блоком **Event Based Random Number**, генерирующим равномерно распределенные случайные числа – тип распределения **Uniform (равномерный**), параметры: **Minimum – 12, Maximum 20**.



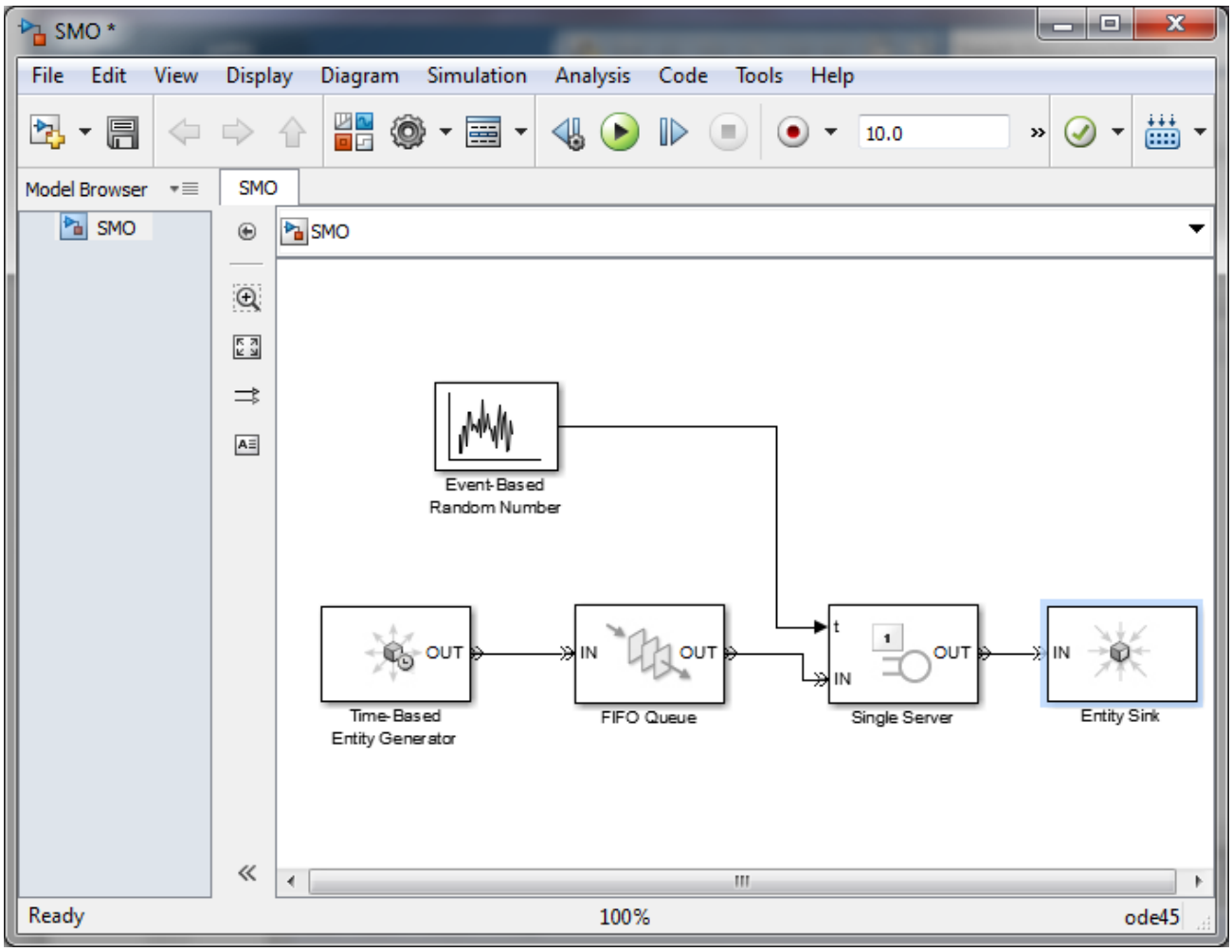
Линии для передачи сущностей (**entity connection line**) показывают связь между двумя блоками (или между их входными/выходными портами для сущностей) путем отображения пути, по которому сущность может:

- покинуть один из блоков;
- одновременно прибыть в следующий блок.

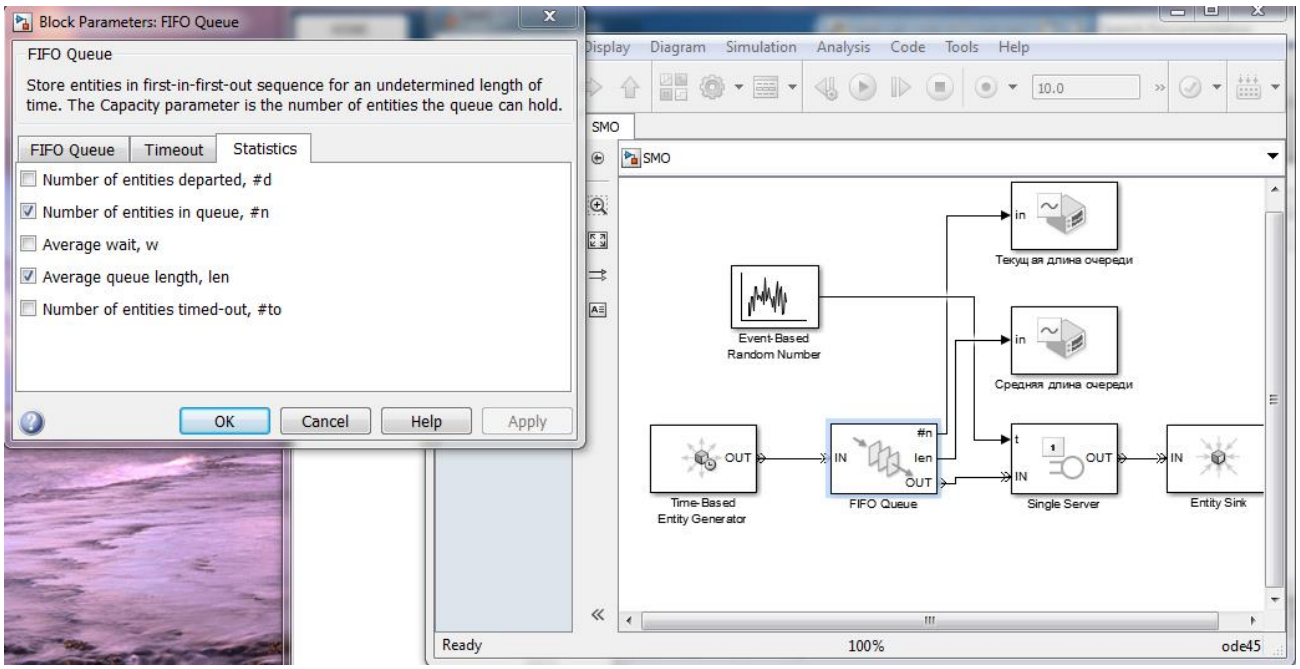
При моделировании сущность, которая покидает порт **OUT** одновременно прибывает в порт **IN** следующего связанного блока.

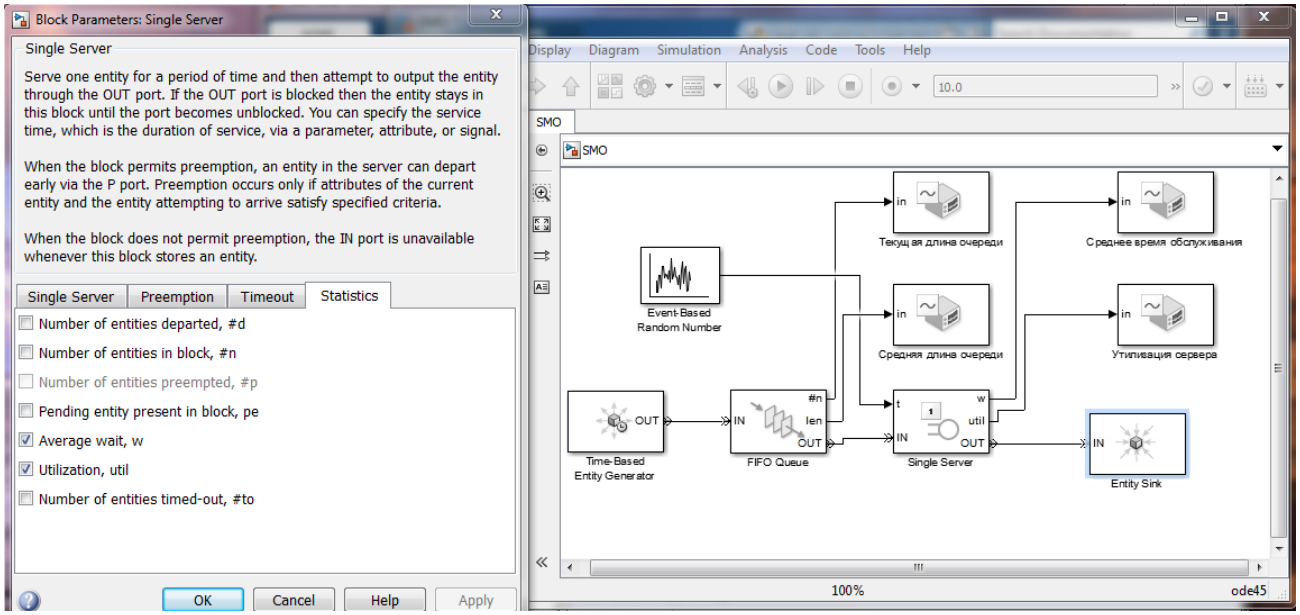
Линии для связи сущностей нельзя разветвлять. Если в приложении требуется, чтобы сущность прибыла в несколько блоков, для создания копий сущностей используется блок **Replicate**.

Блок **Entity Sink** поглощает работы, обработка которых завершена.



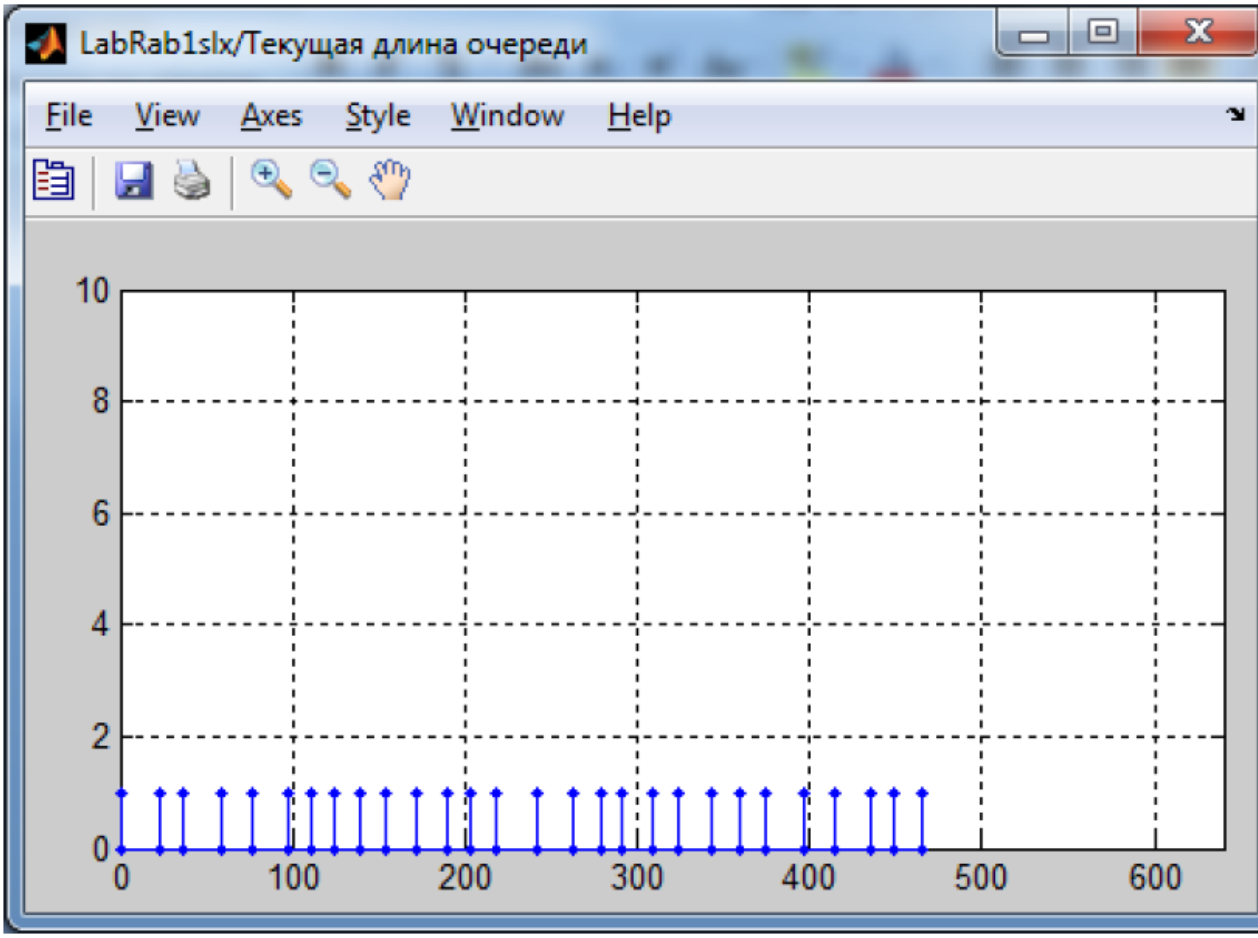
Часть блоков в рассматриваемой модели могут обрабатывать сигналы. Сигналы представляют собой численные величины, определенные в любой момент времени в течение процесса моделирования (не только в дискретные моменты времени). Сигналы появляются на соединительных линиях между сигнальными портами двух блоков. Например, сигнальный выходной порт (**signal output port) #n** блока **FIFO Queue** вязан с сигнальным входным портом (**signal input port) in** блока Текущая длина очереди (**Signal Scope**).



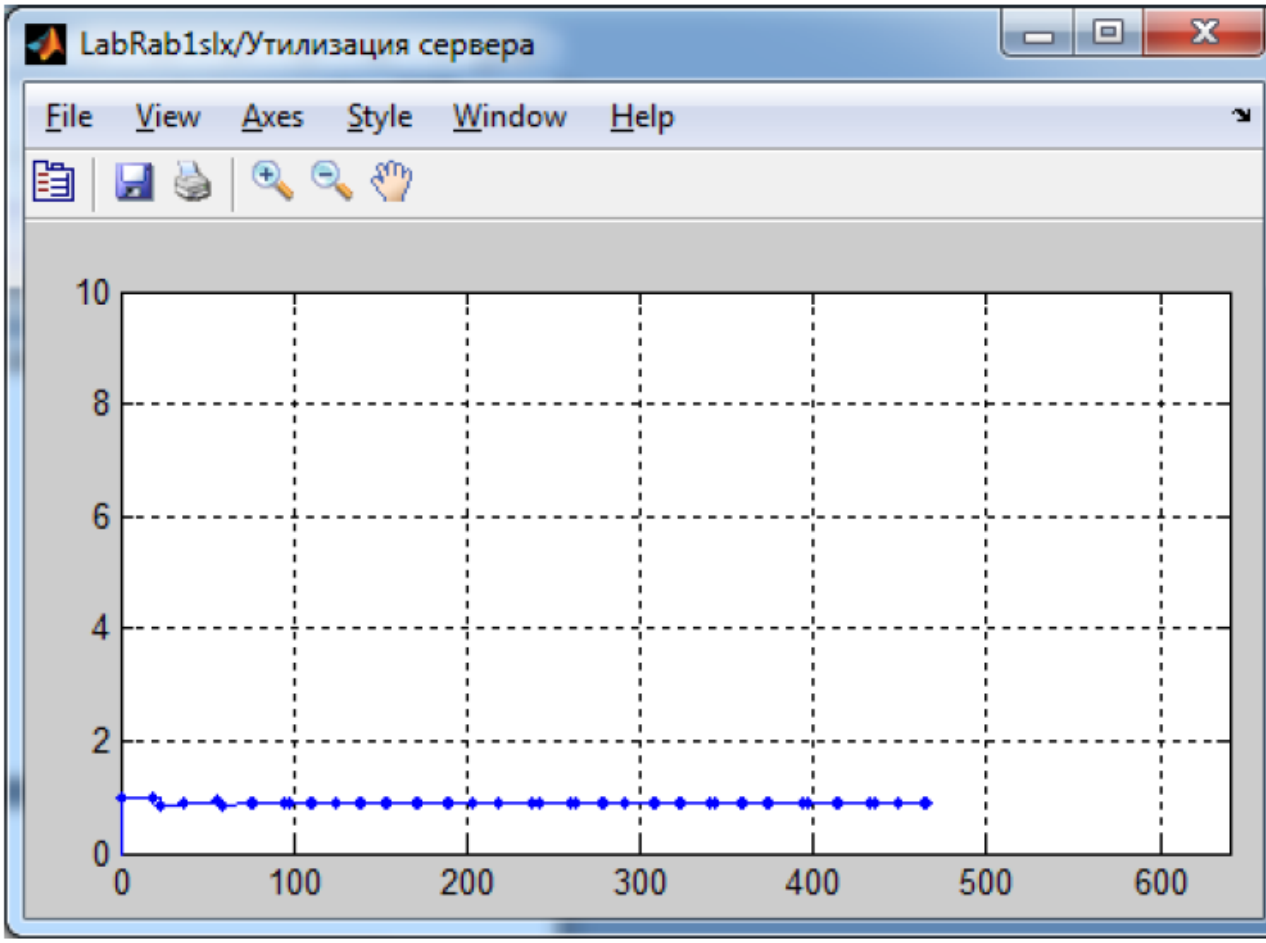


При моделировании событийно-управляемых систем, не содержащих блоков с непрерывными состояниями, необходимо настроить соответствующим образом параметры моделирования. Выберите команду **Simulation->Configuration parameters->Solver**. В разделе **Solver options** в поле **Type** выберите **Variable-step** и в поле **Solver – Discrete**. В поле **Max step size** (максимальный размер шага) введите **inf** (бесконечность).

Время моделирования принято 480 минут (8 часов). На рисунке ниже представлено изменение длины очереди по времени.



На данном рисунке представлено изменение загрузки парикмахера по времени.



Контрольные вопросы к защите

1. Что такое система массового обслуживания?
2. Опишите работу структуры типа LIFO.
3. Опишите работу структуры типа FIFO.
4. В чем заключается сущность накопителя?
5. Для чего предназначена система SimEvents?
6. Что подразумевается под сущностью?
7. Опишите пример моделирования работы парикмахерской.
8. Опишите используемые в данной лабораторной работе блоки системы SimEvents.

Лабораторная работа №4. Построение сетевых моделей посредством сетей Петри

Цель работы: получение практических навыков моделирования параллельных процессов посредством сетей Петри.

Теоретическая часть

Перед выполнением лабораторной работы необходимо овладеть теоретическими знаниями в области построения конечных автоматов, которые содержатся в темах 9 и 10 теоретического материала.

1. Задание сети Петри. Достоинства сети Петри

Сеть Петри – это пятёрка N=<P,T,I,O,S0>.

P – конечное множество вершин, называющиеся позиции. P={p1, p2, … , pn}.

T – конечное множество других вершин, которые называются переходами.

T={t1, t2, … , tm}

$$\left.\begin{array}{l} I \le P \times T \\ Q \le T \times P \end{array}\right\}$$ – бинарные отношения инцидентности, задающие для каждого перехода ti множество входных и выходных позиций.

S0: $P \to E$ – начальная маркировка сети, которая в каждой позиции ставит в однозначное соответствие элемент из множества неотрицательных целых чисел.

Сеть Петри – ориентированный граф с двумя типами вершин (вершина-позиция и вершина-переход), в которых дуги соединяют только разноимённые вершины. Граф является помеченным (маркированным), то есть его вершины-позиции содержат метки (фишки).

Маркировку μ можно определить как n-мерный вектор μ = (μ1, μ2, … , μn), где n = |P|, а μi ∈ E – число фишек в позиции pi, или как комплект, включающий μi раз элемент pi.

Число и расположение фишек могут изменяться при выполнении сети Петри, которое само зависит от числа и распределения фишек по сети. Под выполнением сети Петри понимается последовательность запусков переходов, в результате которых удаляются фишки из входных и появляются в выходных позициях переходов.

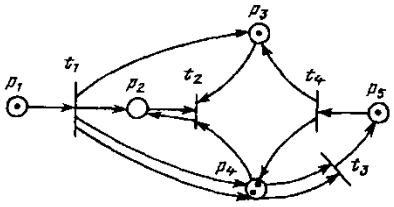


Рис. 1.

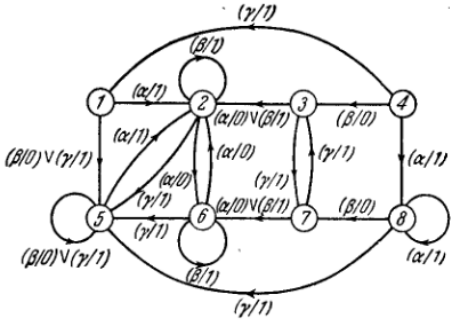


Рис. 2.

Сеть Петри дает преимущество в 2-х случаях:

- при наличии параллельных процессов;
- при наличии конвейерных процессов.

2. Понятие о правильных сетях Петри. Живость и безопасность сетей.

Ограниченность (или К-ограниченность) имеет место (при заданной начальной маркировке S0), если число меток в любой позиции сети не может превысить значения К. При проектировании автоматизированных систем определение К позволяет обоснованно выбирать емкости накопителей. Возможность неограниченного роста числа меток свидетельствует об опасности неограниченного роста длин очередей.

Безопасность - частный случай ограниченности, а именно это ограниченность, если для некоторой позиции установлено, что она безопасна, то ее можно представлять одним триггером.

Живость сети Петри определяется возможностью срабатывания любого перехода при функционировании моделируемого объекта. Сеть Петри называется живой (при заданной начальной маркировке S0), если в ней для любой пары маркировок Si, Sj, принадлежащих множеству R(S0) имеет место Si ├ Sj и для любого перехода t в множестве T существует пара маркировок Sg, Sn в множестве R(S0) таких, что Sg ├t Sn, т.е. $\forall S_i, S_j \in R(S_0) \Rightarrow S_i \mid\!- S_j$.

Условие отсутствия тупиков:

$\forall t \in T, \exists S_g, S_n \in R(S_0) \Rightarrow S_g \mid\!-^t S_n$.

Отсутствие живости либо означает избыточность аппаратуры в проектируемой системе, либо свидетельствует о возможности возникновения зацикливаний, тупиков, блокировок.

Если сеть Петри безопасная и живая, то сеть называется **правильной.**

3. Входные и выходные позиции сети Петри. Условия срабатывания переходов сети Петри

Понятия входной и выходной позиции рассматриваются относительно перехода. Для перехода входные позиции – позиции из которых в данный переход имеются дуги, выходные позиции – позиции в которых из данного перехода идут дуги.

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.

Правила срабатывания переходов (рис. 1), 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 позиции, 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. 3 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).

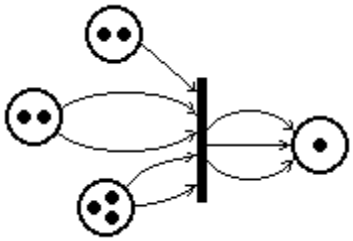


Рис. 3. Фрагмент сети Петри

4. Ординарные и обобщенные сети Петри. Ингибиторные сети.

Ординарная сеть Петри – сеть с единичной кратность дуг между разноименными вершинами.

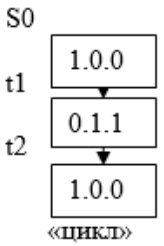
Сеть, в которой допускается применение кратных дуг, называется **обобщенной сетью Петри**.

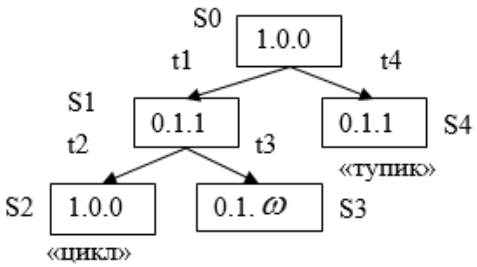
Ингибиторная сеть Петри – это сеть, дополненная специальными ингибиторными дугами, которые связывают только позиции с переходами в отличие от обычных дуг.

Переход может сработать, если каждая его входная позиция, соединенная с ним обычной дугой имеет хотя бы одну метку, а каждая его входная позиция, соединенная с ним ингибиторной дугой не имеет меток.

5. Понятие о дереве достижимых маркировок и цель его построения

Одним из способов проверки правильности сетей Петри является построение дерева достижимых маркировок. Это ориентированный граф дерева, корнем которого является начальная маркировка, а остальные вершины соответствуют возможным маркировкам сети. При этом дуги помечаются переходами, которые срабатывают для достижения данной маркировки. Ветвь дерева заканчивается, если достигается «тупик», «цикл» или имея маркировку с обозначением «омега».





6. Параллелизм в StateFlow

Система с параллелизмом имеет два или больше состояний, которые могут быть активны в одно и то же время. Действия каждого параллельного состояния по существу независимы от других состояний.

Например, эта диаграмма Stateflow имеет параллельную декомпозицию суперсостояния (рисунок 4).

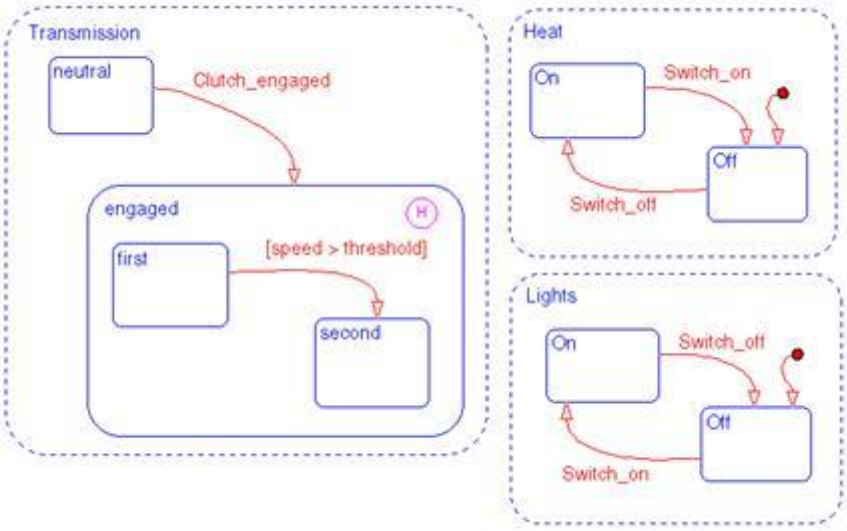


Рисунок 4 – Параллелизм модели автоматической коробки переда

Передача (Transmission), обогрев (Heat) и осветительные приборы (Lights) - параллельные подсистемы в автомобиле. Они существуют параллельно и физически независимы друг от друга. Имеется много других параллельных компонентов в автомобиле, например подсистема торможения и подсистема очистки ветрового стекла.

Вы представляете параллелизм в Stateflow, задавая параллельную (И) декомпозицию. Параллельные (И) состояния отображены обведенными штриховой линией областями.

Параллельная декомпозиция состояний (И)

Потомки родителя с параллельной (И) декомпозицией - это параллельные (И) состояния. Если подсостояния имеют пунктирную границу, декомпозиция состояний является последовательной (ИЛИ). Это представление применяется, если все состояния на этом уровне иерархии всегда активны в одно и тоже время. В следующем примере, когда состояние A активно, и A1 и A2 активны одновременно (рисунок 5).

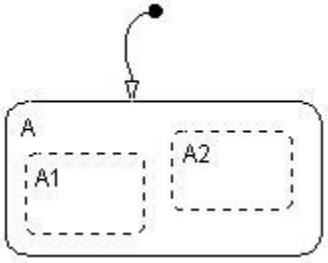


Рисунок 5 – Параллельная декомпозиция состояний И

Активность в параллельных состояниях практически независима, как это показано в следующем примере (рисунок 6). Когда состояние A становится активным, то B и C становятся активными одновременно. Когда C становится активным, активным может стать или C1 или C2.

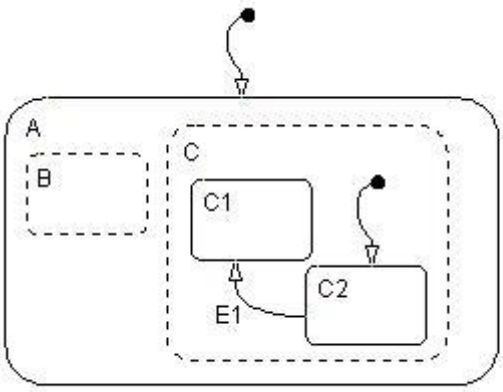


Рисунок 6 – Независимость активности параллельных состояний

Общая постановка задачи

Провести моделирование системы, имеющей в своем составе параллельные процессы.

Список индивидуальных данных

1. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону.
2. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. При этом имеются дополнительные стрелки светофора, разрешающие движение вправо.
3. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. При этом имеются дополнительные стрелки светофора, разрешающие движение влево.
4. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. При этом имеются дополнительные стрелки светофора, разрешающие движение вправо и влево.
5. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется один второстепенный перекресток, в расположении которого по одной полосе в каждом направлении движения.
6. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется один второстепенный перекресток, в расположении которого

по одной полосе в каждом направлении движения. При этом имеются дополнительные стрелки светофора, разрешающие движение вправо.

7. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется один второстепенный перекресток, в расположении которого по одной полосе в каждом направлении движения. При этом имеются дополнительные стрелки светофора, разрешающие движение влево.

8. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется два второстепенных перекрестка, в расположении которых по одной полосе в каждом направлении движения.

9. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется два второстепенных перекрестка, в расположении которых по одной полосе в каждом направлении движения. При этом имеются дополнительные стрелки светофора, разрешающие движение вправо.

10. Смоделировать работу светофоров на перекрестке. Перекресток состоит из 2 взаимно-равнозначных полос, по две полосы в каждую сторону. Также имеется два второстепенных перекрестка, в расположении которых по одной полосе в каждом направлении движения. При этом имеются дополнительные стрелки светофора, разрешающие движение влево.

11. Имеется автомат с напитками (кола стоимостью 30руб, кофе стоимостью 40руб, чай стоимостью 15руб, вода стоимостью 5руб). Смоделировать работу автомата при условии, что автомат принимает монеты достоинством 1, 2, 5 и 10 рублей соответственно. При этом автомат не выдает сдачи.

12. Имеется автомат с напитками (кола стоимостью 30руб, кофе стоимостью 40руб, чай стоимостью 15руб, вода стоимостью 5руб). Смоделировать работу автомата при условии, что автомат принимает монеты достоинством 1, 2, 5 и 10 рублей соответственно. При этом автомат выдает сдачу.

13. Имеется автомат с напитками (кола стоимостью 30руб, кофе стоимостью 40руб, чай стоимостью 15руб, вода стоимостью 5руб). Смоделировать работу автомата при условии, что автомат принимает монеты достоинством 1, 2, 5 и 10 рублей соответственно. При этом автомат не выдает сдачи и имеется возможность отмены заказа.

14. Смоделировать работу железно-дорожного перекрестка с семафором. Необходимо, чтобы в модели была учтена возможность срабатывания вручную (диспетчер) и автоматически (приближающийся поезд).

15.Смоделировать работу железно-дорожного перекрестка с семафором и шлагбаумом. Необходимо, чтобы в модели была учтена возможность

срабатывания вручную (диспетчер) и автоматически (приближающийся поезд).

Пример выполнения работы

Контрольные вопросы к защите

1. Опишите основные множества сети Петри.
2. При моделировании каких процессов, используются сети Петри.
3. Что такое маркировка сети?
4. Что такое ингибиторная сеть?
5. Дайте определение живучести сети.
6. Дайте определение ограниченности сети.
7. Что такое ординарная и обобщенная сеть?

Литература

1. Афонин В. В. Моделирование систем [Текст] / Афонин В. В. , Федосин С. А. М : ИНТУИТ ; БИНОМ. Лаборатория знаний, 2011.
2. Голощапова В.А. Компьютерное моделирование : Учебно-методический комплекс [Электронный ресурс] / Голощапова В.А., Белгород, 2013. Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=5932, свободный.
3. Гультяев А.А. Визуальное моделирование в среде MATLAB [Текст] / Гультяев А.А. СПб: Питер, 2000.
4. Кельтон В. Имитационное моделирование [Текст] / Кельтон В., Лоу А. СПб: Питер; Киев: Изд. группа BHV, 2004.
5. Корнеев Л.Г. Инструментальные средства моделирования сложных систем : Учебно-методический комплекс [Электронный ресурс] / Корнеев Л.Г., Белгород : НИУ БелГУ, 2011. Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=4053, свободный.
6. Королев А.Л. Компьютерное моделирование: Лабораторный практикум [Электронный ресурс] / Королев А.Л. Москва : БИНОМ. Лаборатория знаний, 2013. - 296 с. Режим доступа: http://www.studentlibrary.ru/book/ISBN9785996322558.html, свободный.
7. Ослин Б. Г. Имитационное моделирование систем массового обслуживания [Текст] / Ослин Б. Г. Томск : Изд-во ТПУ, 2003.
8. Сирота А. А. Компьютерное моделирование и оценка эффективности сложных систем [Текст] / Сирота А. А. М.: Техносфера, 2006
9. Советов Б.Я. Моделирование систем [Текст] / Советов Б.Я., Яковлев С.А. М.: Высш.шк., 2005.
10. Центр компетенций MathWorks [Электронный ресурс] - http://matlab.ru/