МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

Утверждаю:
Проректор по учебной работе
_____ О.Г. Локтионова
«11» 06 2024г.

ОСНОВЫ И ПРИНЦИПЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Методические указания для проведения лабораторных занятий и по дисциплине «Моделирование» для студентов направления подготовки 09.04.04 ОПОП ВО Программная инженерия, направленность (профиль) «Предпринимательство, инновации и технологии будущего в программной инженерии»

Курск 2024

УДК 001.891.573
Составитель: Р.А. Томакова

Рецензент
Кандидат технических наук, доцент А. В. Малышев

Основы и принципы имитационного моделирования: методические указания для проведения лабораторных занятий по дисциплине «Моделирование» для студентов направления подготовки 09.04.04 ОПОП ВО Программная инженерия, направленность (профиль) «Предпринимательство, инновации и технологии будущего в программной инженерии» / Юго-Зап. гос. ун-т; сост. Р.А. Томакова. Курск, 2024. -30с.

Рассмотрены основные понятиями и принципы имитационного моделирования, произведена классификация научно-исследовательских работ, выделены особенности и преимущества.

Методические указания составлены в соответствии с ФГОС ВО – магистратура по направлению подготовки 09.04.04 ОПОП ВО Программная инженерия, направленность (профиль) «Предпринимательство, инновации и технологии будущего в программной инженерии».

Предназначены для студентов, обучающихся по направлению подготовки 09.04.04 Программная инженерия, направленность (профиль) «Предпринимательство, инновации и технологии будущего в программной инженерии» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать 11.06.2024. Формат 60×84 1/16.
Усл. печ. л. 1,6 . Уч.- изд. л. 1,5. Тираж 100. Заказ 492 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

ОСНОВЫ И ПРИНЦИПЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Цель и задачи лабораторного занятия (лабораторной работы):

Цель работы – изучение методов имитационного моделирования сложных систем, применяемых для проведения научных исследований и приобретение практических навыков их использования.

Задачи работы.

- Познакомиться с основными понятиями методологии имитационного моделирования, применяемыми в научных исследованиях;
- Изучить классификацию методов имитационного моделирования, этапы построения моделей;
- Изучить функции и значение имитационного моделирования в процессе познания;
- Выделить способы классификации имитационных моделей и развертывания теорий;
- Изучить особенности теоретического уровня проведения исследований;
- Выделить структурные компоненты имитационного моделирования;
- Познакомиться с алгоритмами процесса построения и назначения имитационных моделей;
- *Проанализировать и обосновать* преимущества применения имитационного моделирования в процессе разработки.

Планируемые результаты обучения (формируемые знания, умения, навыки и компетенции):

Код и наименование индикатора достижения компетенции, закрепленного за дисциплиной:

ОПК-4.1 Использует новые научные принципы и методы исследований;

ОПК-4.2 Применяет на практике новые научные принципы и методы исследований;

ОПК-4.3 Решает профессиональные задачи с применением новых научных принципов и методов исследования.

Необходимые материально-техническое оборудование и материалы:

1. Класс ПЭВМ - Athlon 64 X2-2.4; Cel 2.4, Cel 2.6, Cel 800.
2. Мультимедиа центр: ноутбук ASUS X50VL PMD T2330/14"/1024Mb/ 160Gb/ сум-ка/проектор inFocus IN24+ .
3. Экран мобильный Draper Diplomat 60x60
4. Доступ в сеть Интернет.

Шкала оценивания и критерии оценивания выполненной практической работы:

Форма контроля	Минимальный балл		Максимальный балл	
	балл	примечание	балл	примечание
1	2	3	4	5
Лабораторное занятие №3 **Основы и принципы имитационного моделирования**	6	Выполнил, но «не защитил»	12	Выполнил и «защитил»

План проведения лабораторного занятия (лабораторной работы)

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Составить структурную схему процесса имитационного моделирования. Обосновать выбор входящих модулей.
2. Составить структурные компоненты классификации имитационного моделирования. Аргументировать содержание входящих модулей и их назначение.
3. Обосновать особенности алгоритма процесса имитационного моделирования при проведения исследований.
4. Проанализировать проблемную ситуацию как систему, выявляя ее составляющие и связи между ними.
4. Определить пробелы в информации, необходимой для решения проблемной ситуации, обосновать причины возникновения и развития проблемных ситуаций. Обосновать представление модели в распределенном моделировании.
5. Разработать и содержательно аргументировать стратегию решения проблемной ситуации на основе системного и междисциплинарных подходов.
6. Проанализировать основные языки, применяемые в имитационном моделировании. Выполнить структуризацию предметной области и построение модели.
7. Проанализировать и обосновать преимущества и недостатки языково-ориентированной методологии.
8. Постройте алгоритм проверки адекватности с целью воспроизведения моделью с необходимой полнотой всех характеристик объекта, существенных для цели моделирования при всем различии модели и оригинала.
9. Произвести сравнительный анализ результатов.
10. Сделать выводы по работе.
11. Представить отчет.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Имитационное моделирование (**ИМ**) представляет собой метод исследования, при котором изучаемая система заменяется моделью, описывающей реальную систему с целью проведения экспериментов и получения информации о ней. При этом имитационная модель воспроизводит (*имитирует*) поведение моделируемой системы во времени или при различных условиях с достаточной точностью.

Имитационное моделирование представляет частный случай математического моделирования, являясь мощным, а иногда единственным методом исследования динамического поведения сложных систем.

Имитационная модель представляет собой логико-математическое описание объекта, которое используется для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

Следует заметить, что существует класс объектов, для которых по различным причинам не разработаны аналитические модели, либо не разработаны методы решения. В этом случае целесообразна замена математической модели имитатором или имитационной моделью.

Имитационное моделирование целесообразно применять в случаях:

- если дорого или невозможно проводить эксперименты на реальном объекте;
- невозможно построить аналитическую модель: в системе есть время, причинные связи, последствие, нелинейности, стохастические (случайные) переменные;
- необходимо имитировать поведение системы во времени.

1.1 Цель и задачи имитационного моделирования

Цель имитационного моделирования заключается в воспроизведении поведения исследуемой системы на основе результатов анализа наиболее существенных взаимосвязей между

ее элементами или разработке симулятора (simulationmodeling) исследуемой предметной области для проведения различных экспериментов.

Обозначенная цель будет достигнута в ходе решения следующих ***задач исследования:***

1. Определить понятия имитационной модели и имитационного моделирования.

2. Выделить этапы и структуру процесса имитационного моделирования.

3.Изучить историю становления имитационного моделирования.

4. Проанализировать динамику развития и применения метода имитационного моделирования.

5. Определить, как и какие основные системы имитационного моделирования используются в современных условиях для исследования и управления.

Задачи исследования, решаемые с помощью имитационного моделирования, представлены на рисунке 1.

Рисунок 1- Задачи, решаемые с помощью имитационного моделирования

1. *Прямые задачи анализа*, при решении которых исследуемая система задаётся параметрами своих элементов и параметрами исходного режима, структурой или уравнениями и требуется определить реакцию системы на действующие силы.

2. *Обратные задачи анализа*: задачи, в которых по известной реакции системы требуется найти возмущения, заставившие

рассматриваемую систему прийти к данному состоянию и данной реакции.

3. *Задачи синтеза:* при решении определяются такие значения параметров, при которых происходящие процессы в системе имеют желательный по каким-либо соображениям характер.

4. *Индуктивные задачи*, решение которых имеет целью проверку гипотез, уточнение уравнений, описывающих процессы, происходящие в системе, выяснение свойств этих элементов, отладка программ (алгоритмов) для расчётов на компьютере.

Методика построения имитационных моделей состоит из двух этапов:

1. Методология имитации – *постановка задачи, подготовка данных, построение модели, оценка адекватности;*

2. Организация имитационного эксперимента – *планирование эксперимента, экспериментирование, обработка результатов, эксперимента*.

Структура имитационного моделирования представима в виде пяти этапов, выполнение которых осуществляется последовательно. Цикличность исследований проявляется в необходимости возвращения к предыдущим этапам и повторении уже однажды пройденного пути с измененными в силу необходимости данными и параметрами модели.

На первом этапе выполняется оценка проблемы, возможность и способы решения задачи, ожидаемые результаты. Этот этап характерен для практического применения моделирования, к нему возвращаются после окончания исследования модели и обработки результатов для изменения постановки задачи, а иногда и самой цели моделирования.

На втором этапе осуществляют формализацию моделируемого объекта: дается описание элементов исследуемого объекта, взаимодействия между элементами объекта и объекта с внешней средой. Этот этап включает создание имитационной модели, разработку программы для ЭВМ на основе выбранного языка моделирования. Также осуществляется проверка полученной моделирующей программы на соответствие ее той теоретической схеме, которая была положена в основу формального описания объекта моделирования. Этот процесс часто называют *верификацией модели*. Завершением второго этапа является оценка соответствия имитационной модели свойствам реальной системы.

Если условия не выполняются, то следует вернуться к моменту формализации модели, чтобы провести коррекцию в определении теоретической базы модели.

Третий этап *имитационного моделирования* заключается в проведении исследования на разработанной модели путем анализа ее на ЭВМ. Определяется необходимое количество «прогонов» модели, позволяющее получить необходимый объем информации при заданном составе и достоверности исходных данных. Далее на основе разработанного плана эксперимента осуществляют «прогоны» имитационной модели на ЭВМ. В конце этапа выполняется обработка результатов эксперимента.

На четвертом этапе осуществляется анализ результатов исследования, формируются выводы по проведенному моделированию. На этом этапе определяются те свойства реальной системы, которые наиболее важны для исследователя.

Пятый этап является заключительным. На этом этапе формулируются окончательные выводы и разрабатываются рекомендации по использованию результатов моделирования для достижения поставленных целей. На основе этих выводов либо возвращаются к началу процесса моделирования для внесения необходимых изменений в модели и проведении повторных исследований с измененной моделью; либо заканчивают исследования. В результате нескольких подобных циклов получают имитационную модель, наилучшим образом удовлетворяющую поставленным задачам.

Таким образом, метод имитационного моделирования при исследовании сложной проблемной ситуации предполагает выполнение пяти этапов.

1.2 Виды имитационного моделирования

Имитационные модели позволяют проверить верность понимания процессов в исследуемом объекте, и выявить в различных конкретных случаях параметры порядка. Знание последних и дает возможность строить простые модели сложных явлений. Имитационное (компьютерное) моделирование подразделяется на несколько видов, которые представлены на рисунке 2.

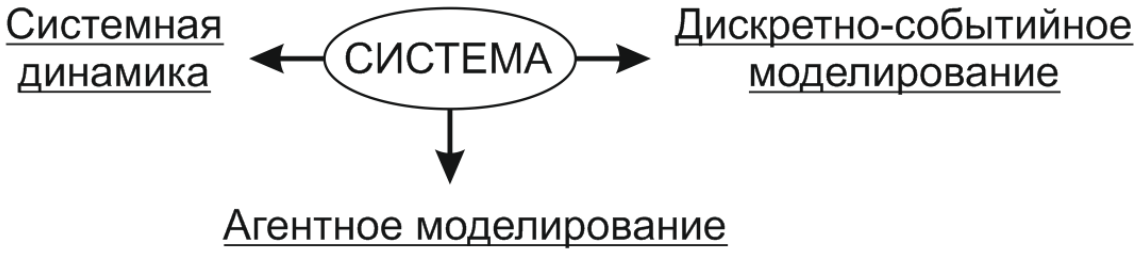

Рисунок 2 - Виды имитационного моделирования

Агентное моделирование представляет собой относительно новое направление в имитационном моделировании, которое разработано в 1990- – 2000-е гг. В настоящее время агентное моделирование применяется для исследования децентрализованных систем, динамика функционирования, которых определяется индивидуальной активностью членов группы.

Цель агентных моделей заключается в получении представлений об общем поведении системы, исходя из возможных предположений об индивидуальном, частном поведении отдельных активных объектов и взаимодействии этих объектов в системе.

Агент в системе рассматривается как некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться.

Дискретно-событийное моделирование представляет собой подход к моделированию, предлагающий рассматривать только основные события моделируемой системы, такие как: «*ожидание*», «*обработка заказа*», «*движение с грузом*», «*разгрузка*» и другие. Дискретно-событийное моделирование наиболее развито и имеет огромную сферу приложений — от логистики и систем массового обслуживания до транспортных и производственных систем. Этот вид моделирования наиболее подходит для моделирования производственных процессов. Он был основан Джеффри Гордоном в 1960-х годах.

Статистическое имитационное моделирование представляет вид имитационного моделирования, позволяющий воспроизводить на ЭВМ функционирование сложных случайных процессов.

При исследовании сложных систем, подверженных случайным возмущениям, используются вероятностные аналитические модели и вероятностные имитационные модели. В вероятностном имитационном моделировании оперируют не с характеристиками случайных процессов, а с конкретными случайными числовыми значениями параметров процесса или системы. При этом результаты, полученные при воспроизведении на имитационной модели рассматриваемого процесса, являются случайными реализациями. Поэтому для нахождения объективных и устойчивых характеристик процесса требуется его многократное воспроизведение, с последующей статистической обработкой полученных данных. Именно поэтому исследование сложных процессов и систем, подверженных случайным возмущениям, с помощью имитационного моделирования принято называть ***статистическим моделированием***.

При реализации на ЭВМ *статистического имитационного моделирования* возникает задача получения на ЭВМ случайных числовых последовательностей с заданными вероятностными характеристиками. При этом чаще всего применяют метод «Монте-Карло», реализующий задачу генерирования последовательности случайных чисел с заданными законами распределения.

1.3 Представление модели в распределенном моделировании

Последовательная имитационная модель может быть выполнена с использованием параллельной вычислительной техники. При этом выигрыш по времени может быть достигнут за счет параллельного выполнения событий, запланированных на один и тот же момент модельного времени.

Распределенное моделирование использует другую общую форму параллелизма, а именно параллельное выполнение событий, запланированных в различных отрезках модельного времени.

В распределенном моделировании в отличие от последовательного моделирования первичной единицей является не объект, а так называемый *логический процесс*. Каждый логический процесс имеет собственный набор объектов и собственную управляющую программу. При этом каждый логический процесс характеризуется локальным списком

событий и часами локального модельного времени. Логические процессы взаимодействуют исключительно с помощью передачи сообщений.

Распределенную модель, состоящую из N логических процессов, можно определить следующим образом:

$$DM = \bigcup_{k=1}^{N} lp^k,$$

$$lp^k = \{sm^k, T^k, S^k\},$$

$$S^k = \{(Q_1^k, t_1^k), (Q_2^k, t_2^k), ..., (Q_m^k, t_m^k), \left| t_1^k \le t_2^k \le ... \le t_m^k \right.\},$$

$$T = \sum_{k=1}^{N} T^k.$$

где DM — распределенная модель; lp — логический процесс; sm — подмодель логического процесса; T — значение локальных часов модельного времени логического процесса; S — локальный список событий логического процесса; Q — событие в списке событий; t — модельное время наступления события Q (временная метка события Q).

Глобальных (общих для всей распределенной модели) часов модельного времени и глобального списка событий явно не существует, так как наличие общей управляющей программы, работающей с этими глобальными структурами, было бы тяжелым в исполнении для параллельного исполнения. Текущее модельное время всей модели TIME в каждый момент равно:

$$TIME = \min_{k=1,\,N} T^k.$$

Каждый логический процесс выполняется в собственном модельном времен и как автономная последовательная модель. Для этого он снабжается экземпляром управляющей программы.

При этом логический процесс общается с другими процессами, передавая им сообщения, которые могут быть разных видов. Будем рассматривать простейший вид сообщения — *асинхронное сообщение*, никак не влияющее на выполнение логического процесса — отправителя.

Сообщение M_{ij}, передаваемое от логического процесса lp^i логическому процессу lp^j, в простейшем случае имеет следующую структуру:

$$M_{ij} = (Q^j, T^i).$$

Логический процесс lp^j, получив сообщение, которое имеет форму события, вставляет это событие в упорядоченный список своих локальных событий в соответствии со значением T^j.

Управляющая программа начинает выполнять модифицированный список локальных событий. Таким образом, полученное сообщение может изменить логику выполнения логического процесса — получателя.

Передача сообщений может осуществляться логическими процессами непосредственно с помощью средств операционной системы, возможная схема которой приведенная на рисунке 3.

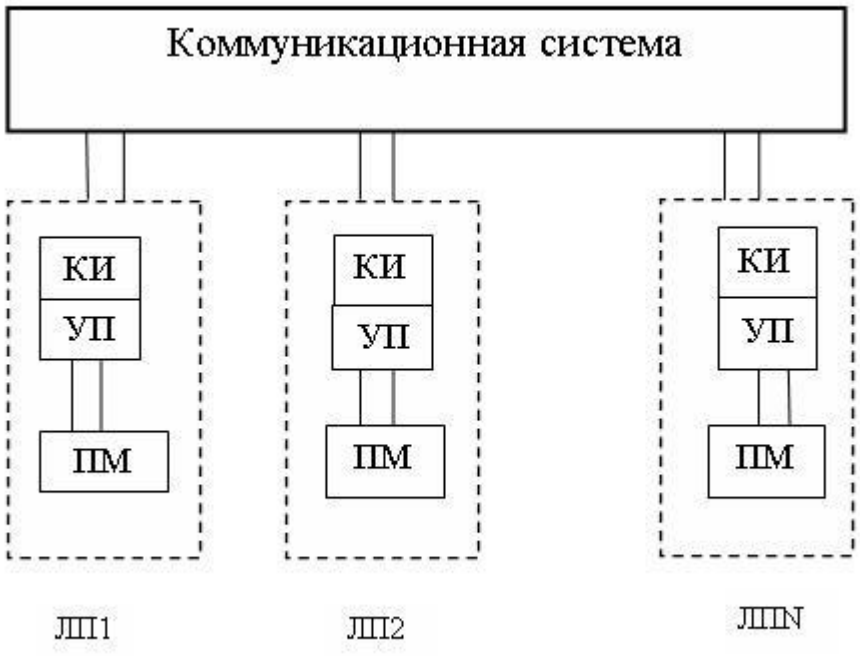

Рисунок 3 - Схема выполнения распределенной модели:
ЛП — логический процесс, ПМ — подмодель, УП — управляющая программа, КИ — коммуникационный интерфейс.

Она включает коммуникационную подсистему в качестве отдельного компонента. Логические процессы

взаимодействуют с коммуникационной подсистемой с помощью определенного интерфейса.

Вся системная часть реализации передачи сообщений сосредоточена в одной программе (коммуникационной подсистеме).

Логические процессы могут быть реализованы на языке **C++.** Для переноса моделей и системы моделирования на другую вычислительную систему достаточно перенести коммуникационную подсистему. Перенос коммуникационной системы упрощается, если при ее реализации использованы стандартные средства, например передача сообщений, реализована с помощью протокола **MPI** (Message passing interface) , который имеет реализации как для Windows, так и для параллельных компьютеров и гетерогенных кластеров.

Коммуникационная подсистема может синхронизировать выполнение логических процессов в модельном времени.

2. ЯЗЫКИ, ПРИМЕНЯЕМЫЕ В ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ

2.1 Предметно-ориентированные языки

Предметно-ориентированный язык представляет собой язык программирования, специализированный для конкретной области применения. Построение такого языка и/или его структура данных отражают специфику решаемых с его помощью задач.

Следует отметить, что деление языков программирования на языки общего назначения и предметно-ориентированные является условным. Существует большое количество языков общего назначения, применяемых в качестве предметно-ориентированных для решения определённых задач, а также предметно-ориентированных языков, применяемых в качестве языков общего назначения.

Ярким примером является **язык СИ**, разработанный в качестве кроссплатформенного ассемблера, но на практике применяемый гораздо шире.

Язык **ML**, породивший целое семейство языков общего назначения, включая **Haskell**, ныне наиболее предпочитаемый

разработчиками предметно-ориентированных языков в качестве базового. Изначально разрабатывался в качестве **DSL** для системы автоматического доказательства теорем LFC.

Примером, показывающим условность классификации, служит язык **БНФ** (и компилятор с него Lex/Yacc): с одной стороны, это яркий пример метаязыка, с другой—предназначен для одной конкретной задачи.

Простейшие предметно-ориентированные языки, используемые в одном конкретном приложении, часто называют ***«мини-языками».***

Часто используют термины «*problemoriented*» и «*domainoriented*», но в англоязычном научном сообществе прижился термин «*domain-specific*», причём именно «*domain-specificlanguage*», а не «*domain-specificprogramminglanguage*». В русскоязычной литературе по программированию встречаются варианты «*доменно-специфичный*», «*проблемно-ориентированный*», «*предметно-ориентированный*».

2.2 Языки имитационного моделирования

Основой любой моделирующей системы является *язык имитационного моделирования (***ЯИМ***)*. Языки имитационного моделирования относятся к классу проблемно-ориентированных языков, которые разрабатывались для программного обеспечения определенных классов ИМ.

Языки имитационного моделирования реализуются в программно-методических комплексах моделирования СМО, имеющих ту или иную степень специализации.

Языки имитационного моделирования отражают определенные специфические и структурные особенности моделируемых явлений при использовании определенных описательных и динамических понятий, например, элементов структуризации, событий и состояний.

Особенность *языков имитационного моделирования дискретных систем* состоит в том, что последовательность выполнения операций программы, написанных на этих языках, определяется в ходе выполнения программы в зависимости от временных интервалов, характеризующих длительности обработок на различных этапах моделируемого процесса.

Во-первых, *языки имитационного моделирования* представляют программисту средства для описания функционирования сложных систем в виде широкого набора специальных операторов, а также освобождают от необходимости программирования служебных операций моделирования, которые автоматически реализуются.

Во-вторых, *языки имитационного моделирования* представляют методологическую основу для описания и исследования систем различной природы с единых позиций Конструкции языков являются абстракциями, применимые к широкому классу явлений. Использование языка заключается в том, чтобы отождествить компоненты конкретной системы с соответствующими единицами языка, установить соотношения между ними и описать эти соотношения средствами языка.

Развитие *языков имитационного моделирования* началось в конце 50 - х годов. Сначала в моделировании использовали либо языки, ориентированные на конкретную ЭВМ, либо универсальные языки.

Одним из популярных *языков имитационного моделирования*, применяемых для решения задач системного анализа, является язык **DYNAMO,** разработанный в США группой Форрестора. Этот язык ориентирован на моделирование простых динамических систем, описываемых обыкновенными дифференциальными уравнениями с возможностью задания случайных возмущений. Все это создает возможность использования языка DYNAMO специалистами, не знающими программирования, но представляющими содержательную сторону своей задачи. Однако существуют и определенные ограничения, связанные с отсутствием векторного представления процессов, а также слабо развитая логика временных взаимодействий.

Модели, построенные с использованием этого языка для отображения динамики системы, состоят из таких специфических логических структур, как уровни, потоки, преобразователи и соединители.

Методологическую основу *языков имитационного моделирования* составляют представления об объекте исследования как о системе с дискретными событиями. При этом модель строится таким образом, что поведение системы воспроизводится в виде последовательности ее состояний во времени. Всякое

фиксируемое изменение состояния системы называется *событием.* Событие, вызываемое в реальной системе изменением параметров в течение некоторого временного интервала, в модели считается происходящим мгновенно. Дискретность моделей такого типа заключается в особом подходе к наблюдению поведения исследуемых систем: фиксируются только те события, которые существенны в плане проводимого исследования. При этом считается, что система не наблюдаема в течение промежутков времени между фиксируемыми событиями. В силу этого языки системного моделирования часто называют языками моделирования систем с дискретными событиями.

Для некоторых *языков машинного имитационного моделирования* (например, **GPSS**) достаточно лишь конкретизировать способ дискретной аппроксимации, и соответствующие случайные события будут генерироваться автоматически.

Следует отметить, что *языки имитационного моделирования* позволяют за сравнительно короткий срок составлять программные модели довольно сложных систем. Для ответа на подобные вопросы часто приходится непосредственно изменять программный код модели, что повышает затраты времени на анализ системы. При использовании языков имитационного моделирования возникает также другая проблема: затраты на изучение и освоение языка, тем более, что эти языки оперируют абстрактными понятиями, в то время как экспериментатор часто является специалистом в той области, которой принадлежит моделируемая система, и применяет специфическую терминологию, что может значительно осложнить освоение языка исследователем.

Существующие методы и *языки имитационного моделирования* часто оказываются неэффективными по причине своей низкой гибкости и сложности моделирования систем принятия решений и управления, особенно если система управления включает в себя человека-оператора, принимающего решения. Использование появившихся на рынке программных продуктов интеллектуальных систем имитационного моделирования снимает часть этих трудностей и предоставляет новые возможности при использовании имитации в гибридных системах для решения прикладных системных задач.

При построении имитационных моделей важное значение имеет выбор *соответствующего языка имитационного моделирования*, использование которого существенно упрощает и ускоряет процесс моделирования.

Поскольку требования к языкам программирования вообще и к *языкам имитационного моделирования,* в частности, четко не определены, поэтом нет и единого мнения относительно того, что же следует считать собственно языком.

Помимо самих языков, существуют диалекты, ответвления (или языки **L-типа**) и расширенные варианты языков. *Диалектом* принято называть сокращенный вариант основного языка. Язык L-типа есть язык, по духу и букве сходный с основным языком L, но обладающий несколько иными свойствами, что не позволяет считать его диалектом.

Важно отметить ***два важных преимущества*** *языков имитационного моделирования* по сравнению с универсальными заключающиеся в удобстве программирования и концептуальной выразительности. Последнее достоинство - возможность четко и ясно описывать различные понятия - важно на стадии моделирования и выбора общего подхода к изучению системы.

Удобство программирования особенно сильно проявляется при написании самой программы. Еще одно преимущество языков имитационного моделирования заключается в том, что их можно использовать как средства коммуникации и документирования.

Языки имитационного моделирования отражают определенные специфические и структурные особенности моделируемых явлений при использовании определенных описательных и динамических понятий, например, элементов структуризации, событий и состояний.

Специализированные языки цифрового имитационного моделирования делятся на две самостоятельные группы, соответствующие двум видам имитации: непрерывных и дискретных процессов, которые проиллюстрированы на рисунке 4.

Группа *языков непрерывного имитационного моделирования* в свою очередь делится на три типа: языки аналогового моделирования; языки, применяемые для решения систем дифференциальных уравнений, описывающих детерминированные замкнутые непрерывные системы; языки уравнений. Первые два

типа рассчитаны на блочное построение моделей, языки третьего типа оперируют с уравнениями.

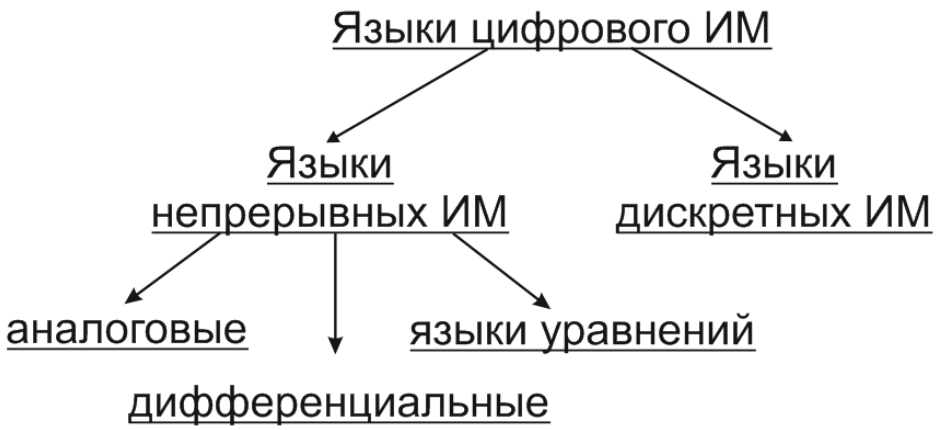

Рисунок 4 - Языки цифрового имитационного моделирования

Таким образом, языки имитационного моделирования позволяют за сравнительно короткий срок составлять программные модели довольно сложных систем.

Однако имитационному моделированию присущи специфические характеристики. Поэтому желательно, чтобы в *языке имитационного моделирования* содержались специальные средства для записи алгоритмов имитации сложных объектов. Кроме того, в каждый момент времени в моделируемой системе одновременно протекают различные процессы, поэтому в моделирующей системе должны быть средства для синхронизации этих процессов.

Таким образом, функционирование модели должно протекать в искусственном (не в реальном и не в машинном) времени, обеспечивая появление событий в требуемом логикой работы исследуемой системы порядке и с надлежащими временными интервалами между ними. При этом надо учитывать, что элементы реальной системы функционируют одновременно (параллельно), а компоненты программной модели действуют последовательно, так как реализуются с помощью ЭВМ последовательного действия.

Для сохранения адекватности причинно-следственных временных связей необходимо в создать механизм задания времени для синхронизации действий элементов модели системы.

В конечном счете, перед разработчиком модели возникает проблема ее описания на языке, приемлемом для используемой

ЭВМ. Имитационные модели обычно имеют очень сложную логическую структуру, характеризующуюся множеством взаимосвязей между элементами системы, причем многие из этих взаимосвязей претерпевают в ходе выполнения программы динамические изменения. Поэтому *типа GPSS, Симскрипт, Симула, Динамо* и им подобные являются языками более высокого уровня, чем универсальные языки типа Фортран, Алгол и Бэйсик.

Ведущие исследователи языково-ориентированного программирования приводят следующие примеры предметно-специфичных языков в качестве классических:

- **TeX**для подготовки (компьютерной вёрстки) текстовых документов;
- **Perl** применяется для манипулирования текстами;
- **SQL** применяется для разработки СУБД;
- **Tcl**для графического интерфейса пользователя;
- **HTML и SGML** для разметки документов;
- **Verilog и VHDL** для описания аппаратного обеспечения;
- **Mathematica и Maple** для символьных вычислений;
- **AutoLisp** для компьютерного моделирования (САПР);
- **Prolog** для задач, сформулированных в терминах исчисления предикатов;
- **ML и Haskell** для задач, сформулированных в терминах функций.

Существуют языки программирования, встроенные в систему управления ресурсами предприятия (язык ABAP в SAP/R3, языки систем Галактика, Парус, 1С) и применяемые для их дополнения специфичными для конкретной организации модулями. Использование встроенного языка упрощает программирование специфичных задач, поскольку в языке изначально присутствуют понятия предметной области. Некоторые геоинформационные системы и САПР также имеют встроенные языки программирования.

Ещё одним примером является G-код язык программирования устройств с числовым программным управлением (ЧПУ).

2.3 Встраиваемые языки

Компьютерные языки реализуются зависимым образом, т.е. «внутри» транслируемого языка, без которого эти языки не только

не способны исполняться, но и зачастую не образуют целостную символьную систему и не обладают Тьюринг-полнотой.

Такие языки принято называть «*встраиваемыми предметно-специфичными языками*» или просто «*встраиваемыми языками*», а также «языками, реализованными *поверх* или *на основе* данного языка».

Встраиваемые языки допускают ещё несколько видов реализации языка:

- чистое встраивание;
- использование макросредств языка (и нередко отождествление их с термином «метапрограммирование»), которое, в свою очередь, подразделяется:
  1. многостадийные вычисления;
  2. квазицитирование (известное из языка **Lisp**);
  3. использование шаблонов.

С другой стороны, можно рассматривать реализацию встраиваемого языка как «реализацию без трансляции», подразумевая, что DSL будет являться синтаксическим и семантическим подмножеством языка, в который он встраивается.

Язык, используемый в качестве базового для реализации другого, часто называют **метаязыком**.

Основных причин для разработки встраиваемых текстовых языков три:

1. Ввод в исходный язык дополнительных возможностей, расширяющих спектр эффективно решаемых задач или синтаксически упрощающих решение часто встречаемых задач.
2. Интенсивное ре-использование компонентов транслятора базового языка: парсера, механизма типизации, реализации тривиальных вещей (таких как арифметика чисел), оптимизатора и др. Это обеспечивает кратное снижение трудоёмкости реализации придуманного языка, а также высокий уровень качества реализации при использовании безопасного языка в качестве базового.
3. Получение возможности эффективно смешивать свойства разных самостоятельных языков в единых фрагментах кода, формируя мультипарадигменный язык широкого профиля, исключая необходимость межъязыкового взаимодействияи расширяя возможности оптимизации.

Наиболее частыми примерами языков первой группы могут служить реализации объектно-ориентированных возможностей в

функциональных или процедурных языках, и классическим примером служит язык **CLOS**. Следует отметить, что термин «язык» здесь используется не всегда —иногда говорят просто о «*реализации в языке новых возможностей*» или о «*расширении языка подсистемой, нацеленной на решение определённых задач*», и нет строго деления на «библиотеки» и «встраиваемые языки», т.к. формально любой API, протокол или структура данных может рассматриваться в качестве языка. Так, например, неотъемлемой частью языка ***Lisp*** является встроенный не полный по Тьюрингу язык S-выражений.

Вторая группа встраиваемых языков наиболее полно представлена в сообществе языка ***Haskell***, и потому сам ***Haskell*** временами определяют как «*DSL для денотационной семантики*». Примерами могут служить Elm и другие языки, представляющие функциональную реактивную парадигму, а также язык ***Curry.*** Временами также встречается похожее выражение в отношении Лиспа: «***Lisp*** *— это не язык, а среда для разработки языков*». Примером языка, реализованного поверх **Lisp**, может служить Qi. Масса встраиваемых мини-языков реализована в языке ***OCaml*** посредством модуля ***CamlpX*** компилятора. Язык ***Rebo***l также проектировался для программирования посредством интенсивной реализации встраиваемых мини-языков. В диалекте **Lisp** ***Scheme*** посредством языка S-выражений реализован не полный по Тьюрингу язык ***SXML*** воплощающий протокол ***XM***L встраиваемым образом.

Встраиваемый язык может иметь самодостаточную полную по Тьюрингу семантику, но тем не менее вместо независимой реализации ре-использовать компоненты базового языка (третья группа, смешение первых двух). Ярким примером является язык ***Schelog,*** реализующий семантику Пролога внутри диалекта **Lisp** ***Schem***e посредством продолжений, и превращающий Пролог из «самостоятельного» языка во встраиваемый.

Учебной задачей для многих функциональных языков служит реализация поверх рассматриваемого языка какого-либо другого, чаще всего языка логики предикатов первого порядка.

В контексте метаязыков самостоятельные языки временами называют «языками первого класса» (по аналогии с сущностями первого класса в языках), а встраиваемые — «объектными языками».

В подавляющем большинстве случаев встраиваемые языки имеют лишь одну поддерживаемую реализацию, и различия в результирующем машинном представлении кода на них зависят лишь от используемого транслятора базового языка. Однако, бывают и исключения — например, язык ***Concurrent ML*** (***CML***), расширяющий Standard ML конструкциями для явного параллелизма, имеет две принципиально различные реализации.

2.4 Визуальные языки

Один из языков (базовый или встраиваемый) может быть визуальным, что нередко применяется в пользовательском программировании .

Тичными примерами таких пар могут служить **AutoLisp—AutoCAD** и **VBA—MicrosoftExcel.** Эти пары образуют целостную интерактивную систему, и с точки зрения пользователя невозможно определить, являются ли визуальные инструменты надстройкой, имитирующей команды встроенного текстового языка, или же текстовый язык управляет визуальными инструментами. Действительные взаимоотношения в этих парах определяются разработчиком.

В паре **Emacs—EmacsLisp** отношения более определённые. **Lisp** традиционно относится к метаязыкам, и в данном случае текстовый редактор надстраивается над ним как визуальный DSL, что и делает последний изменяемым и расширяемым.

В случае, когда оба языка являются визуальными, встраиваемые языки обычно называют иными терминами — *плагинами, фильтрами* и др., и не используют терминологию языково-ориентированного программирования. Формально же можно говорить, например, что для визуального мета-языка обработки графики **AdobePhotoshop** есть множество встраиваемых визуальных мини-языков

2.5 Преимущества и недостатки языково-ориентированной методологии

Преимущества и недостатки использования в конкретной задаче конкретного **DSL** вместо языка общего назначения определяются гораздо явственнее, чем преимущества и недостатки

использования одного языка общего назначения вместо другого. В в большинстве случаев уже разработанный ***DSL*** оказывается концептуально неприменим к одним задачам и даёт бесспорный выигрыш по большинству показателей качества в других, а некоторые подзадачи вообще остаются нерешёнными до разработки **DSL.**

Таким образом, вопрос о преимуществах и недостатках корректнее ставить в свете применения ***языково-ориентированной методологии*** вместо какой-либо другой, сопоставляя потенциальный выигрыш от его использования с затратами на его разработку и сопровождение.

AnyLogic—представляет программное обеспечение, применяемое для имитационного моделирования, разработанное российской компанией TheAnyLogicCompany (бывшая «Экс ДжейТекнолоджис», англ. *XJ Technologies*).

Инструмент обладает современным графическим интерфейсом и позволяет использовать язык ***Java*** для разработки моделей. Версия **AnyLogic PLE** доступна бесплатно для образовательных целей и самообучения и поддерживает все три известных метода моделирования:

- системная динамика;
- дискретно-событийное (процессное) моделирование;
- агентное моделирование.

С помощью **AnyLogic** возможно разрабатывать модели в следующих областях:

- производство;
- логистика и цепочки поставок;
- рынок и конкуренция;
- бизнес-процессы и сфера обслуживания;
- здравоохранение и фармацевтика;
- управление активами и проектами;
- телекоммуникации и информационные системы;
- социальные и экологические системы;
- пешеходная динамика;
- оборона.

AnyLogic и Java

AnyLogic включает в себя графический язык моделирования, а также позволяет пользователю расширять созданные модели с помощью языка Java.

Интеграция компилятора **Java** в **AnyLogic** предоставляет более широкие возможности при создании моделей, а также создание ***Java*** апплетов, которые могут быть открыты любым браузером. Эти апплеты позволяют легко размещать модели ***AnyLog***ic на веб-сайтах. В дополнение к Java-апплетам, ***AnyLogicProfessiona***l поддерживает создание Java-приложений, в этом случае пользователь может запустить модель без инсталляции AnyLogic.

Модели AnyLogic могут быть основаны на любой из основных парадигм имитационного моделирования: дискретно-событийном моделировании, системной динамике и агентном моделировании (рис. 5).

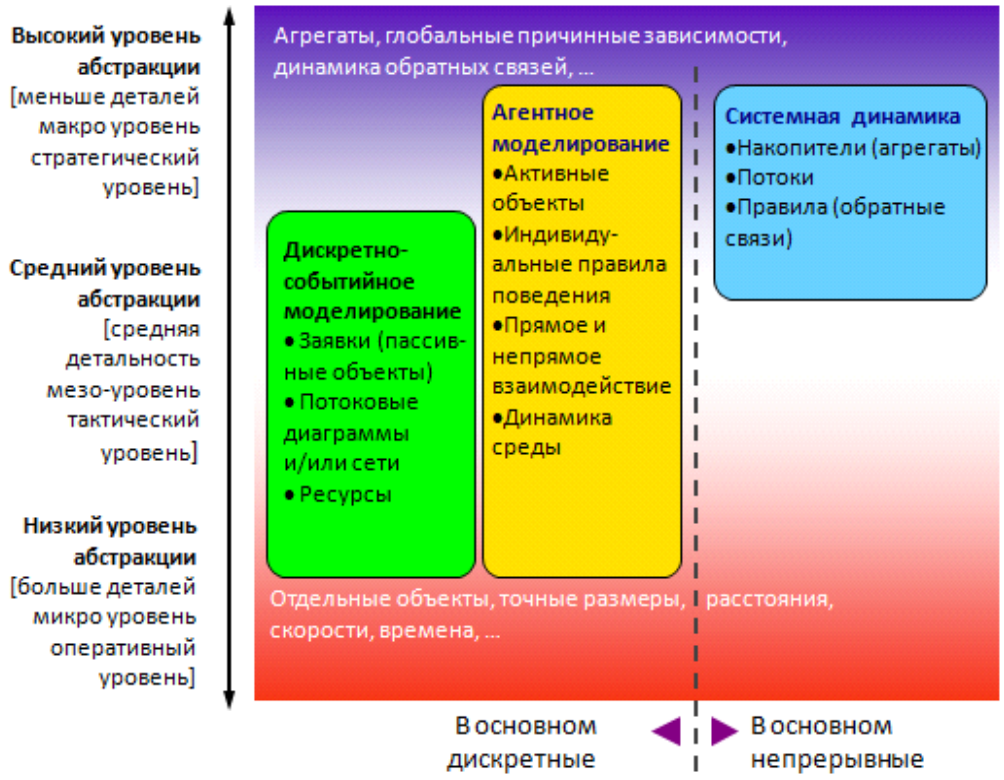

Рисунок 5- Возможные подходы для реализации моделей имитационного моделирования

Системная динамика и дискретно-событийное (процессное) моделирование, под которым мы понимаем любое развитие идей **GPSS** — это традиционные устоявшиеся подходы, агентное моделирование — относительно новый. Системная динамика

оперирует в основном с непрерывными во времени процессами, тогда как дискретно-событийное и агентное моделирование — с дискретными.

Системная динамика и дискретно-событийное моделирование исторически преподаются совершенно разным группам студентов. СД чаще преподается студентам из области менеджмента, ДС — инженерам по организации производства и инженерам-разработчикам систем управления. В результате возникли два практически не пересекающихся сообщества, которые почти никак не общаются друг с другом.

Агентное моделирование до недавнего времени было строго академическим направлением. Однако растущий спрос на глобальную оптимизацию со стороны бизнеса заставил ведущих аналитиков обратить внимание именно на агентное моделирование и его объединение с традиционными подходами с целью получения более полной картины взаимодействия сложных процессов различной природы.

Теперь рассмотрим подходы имитационного моделирования на шкале уровня абстракции. Системная динамика, заменяя индивидуальные объекты их агрегатами, предполагает наивысший уровень абстракции. Дискретно-событийное моделирование работает в низком и среднем диапазоне. Важно отметить, что агентное моделирование может применяться практически на любом уровне и в любых масштабах. Агенты могут представлять пешеходов, автомобили или роботов в физическом пространстве, клиента или продавца на среднем уровне, или же конкурирующие компании на высоком.

При разработке моделей в **AnyLogic** можно использовать концепции и средства из нескольких методов моделирования. Например, в агентной модели можно использовать методы системной динамики для представления изменений состояния среды; в непрерывной модели динамической системы можно учесть дискретные события. При этом производство описывается в рамках дискретно-событийного (процессного) моделирования, где продукт или его части — это заявки, а автомобили, поезда, штабелёры — ресурсы. Сами поставки представляются дискретными событиями, но при этом спрос на товары может описываться непрерывной системно-динамической диаграммой. Возможность смешивать

подходы позволяет описывать процессы реальной жизни, а не подгонять процесс под доступный математический аппарат.

Элементы графической среды:

- **Stock&FlowDiagrams** (диаграмма потоков и накопителей) применяется при разработке моделей, используя метод системной динамики.
- **Statecharts** (карты состояний) в основном используется в агентных моделях для определения поведения агентов. Но также часто используется в дискретно-событийном моделировании, например для симуляции машинных сбоев.
- **Actioncharts** (блок-схемы) используется для построения алгоритмов. Применяется в дискретно-событийном моделировании (маршрутизация звонков) и агентном моделировании (для логики решений агента).
- **Processflowcharts** (процессные диаграммы) основная конструкция, используемая для определения процессов в дискретно-событийном моделировании.

Среда моделирования также включает в себя: низкоуровневые конструкции моделирования (переменные, уравнения, параметры, события и т.п), формы представления (линии, квадраты, овалы и т.п), элементы анализа (базы данных, гистограммы, графики), стандартные картинки и формы экспериментов.

Среда моделирования **AnyLogic** поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов с моделью, включая различные виды анализа: анализ чувствительности модели, оптимизацию параметров модели относительно некоторого критерия.

Интеграция модели с IT-инфраструктурой

Модели **AnyLogic** экспортируются в виде Java-приложений, которые можно запускать отдельно от среды разработки или интегрировать с другими программами. В частности, экспортированная модель может быть встроена в другое ПО и работать в качестве дополнительного модуля для систем типа **ERP**, **MRP** или **TMS**. Модели также интегрируются с файлами TXT, MS Excel, MS Access и любыми базами данных (MS SQL, My SQL, Oracle, и др.). Кроме того, у каждой модели есть своя встроенная база данных на HSQLDB.

2.6 Основные преимущества и недостатки имитационного моделирования

Применение имитационных моделей дает преимущества по сравнению с выполнением экспериментов над реальной системой и использованием других методов.

Стоимость. Например, компания уволила часть сотрудников, что в дальнейшем привело к снижению качества обслуживания и потери части клиентов. Принять обоснованное решение помогла бы имитационная модель, затраты на применение которой состоят лишь из цены программного обеспечения и стоимости консалтинговых услуг.

Время. В реальности оценить эффективность новой сети распространения продукции или измененной структуры склада можно лишь через месяцы или даже годы. Имитационная модель позволяет определить оптимальность таких изменений за считанные минуты, необходимые для проведения эксперимента.

Повторяемость. Современная жизнь требует от организаций быстрой реакции на изменение ситуации на рынке. Например, прогноз объемов спроса продукции должен быть составлен в срок, и его изменения критичны. С помощью имитационной модели можно провести неограниченное количество экспериментов с разными параметрами, чтобы определить наилучший вариант.

Точность. Традиционные расчетные математические методы требуют применения высокой степени абстракции и не учитывают важные детали. Имитационное моделирование позволяет описать структуру системы и её процессы в естественном виде, не прибегая к использованию формул и строгих математических зависимостей.

Наглядность. Имитационная модель обладает возможностями визуализации процесса работы системы во времени, схематичного задания её структуры и выдачи результатов в графическом виде. Это позволяет наглядно представить полученное решение и донести заложенные в него идеи до клиента и коллег.

Универсальность. Имитационное моделирование позволяет решать задачи из любых областей: производства, логистики, финансов, здравоохранения и многих других. В каждом случае модель имитирует, воспроизводит, реальную жизнь и позволяет проводить широкий набор экспериментов без влияния на реальные объекты.

Однако имитационное моделирование наряду с достоинствами имеет и недостатки: - разработка хорошей имитационной модели часто обходится дороже создания аналитической модели и требует больших временных затрат;

- может оказаться, что имитационная модель неточна (что бывает часто), и мы не в состоянии измерить степень этой неточности;

- зачастую исследователи обращаются к имитационному моделированию, не представляя тех трудностей, с которыми они встретятся и совершают при этом ряд ошибок методологического характера.

И, тем не менее, имитационное моделирование является одним из наиболее широко используемых методов при решении задач синтеза и анализа сложных процессов и систем.

Контрольные задания и вопросы

1. Сформулируйте цель имитационного моделирования.
2. Какие виды задач, решаются с помощью имитационного моделирования?
3. Из каких этапов состоит методика построения моделей?
4. Перечислите основные виды имитационного моделирования.
5. Из каких этапов состоит структура имитационного моделирования?
6. В чем заключается цель агентных моделей?
7. В чем заключаются особенности дискретно-событийного моделирования?
8. Сформулируйте особенности статистического имитационного моделирования.
9. Как осуществляется представление модели в распределенном моделировании?
10. Какие требования предъявляются к построению распределенной модели?
11. Какие предметно-ориентированные языки, используемые в имитационном моделировании?
12. В чем состоит особенность языков имитационного моделирования дискретных систем?
13. Методологическую основу *языков* имитационного моделирования составляют?

14. Какие предъявляются требования к языкам имитационного моделирования?

15. Сформулируйте два важных преимущества языков имитационного моделирования по сравнению с универсальными.

16. Какие виды реализации допускают встраиваемые языки?

17. Сформулируйте особенности визуальных языков, применяемых в имитационном моделировании?

18. Какие преимущества дает применение имитационных моделей?

19. Сформулируйте основные недостатки имитационного моделирования.

20. Что является основой построения любой теории?

21. Какие способы построения научных теорий существуют?

22. Перечислите элементы, составляющие основу теоретической модели.

23. Сформулируйте определение научного исследования.

24. Как можно классифицировать научные исследования в зависимости от применяемых методов?

25. Сформулируйте, какую роль эксперимент имеет в формировании научного знания?

26. Сформулируйте особенности эмпирического исследования.

27. Какая связь существует между научным познанием и научным исследованием?

28. Какое значение имеет теория в процессе научного познания?

29. Как осуществляется классификация научных исследований в зависимости от места проведения?

30. Сформулируйте этапы проведения НИР.

Библиографический список

1. Вентцель Е.С. Исследование операций. Задачи, принципы, методология [Текст]/ Е.С. Вентцель. - М.:Высшая школа, 2001.

2. Дворецкий, С.И., Муромцев Ю.А. и др. Моделирование систем [Текст]/С.И. Дворецкий, Ю.А. Муромцев. - М.: Издательский центр «Академия», 2009.

3. Козин Р.Г. Математическое моделирование [Текст]: учеб. Пособие / Р.Г. Козин. – М.: МИФИ, 2008. – 89 с.

4. Советов Б.Я., Яковлев С.А. Моделирование систем [Текст]/ Б.Я. Советов, С.А. Яковлев. – М.: Высшая школа, 2007.

5. Хемди, А. Имитационное моделирование [текст]/ А. Хемди. – М.: «Вильямс» 2009 г. - 737с.

6. Фоменков, С.А. Математическое моделирование системных объектов: учебное пособие [Текст] / С.А.Фоменков, В.А.Камаев, Ю.А.Орлова; ВолгГТУ, Волгоград, 2014.-340 с.

7. Карпов Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. [Текст] /Ю. Карпов. СПб 2008г. – 400с.

8. Строгалев, В. П. Имитационное моделирование [Текст]/ В.П. Строгалев, И.О. Толкачева. – МГТУ им. Баумана, 2008г .– 697с.

9. Куприяшкин, А.Г. Основы моделирования систем [Текст]/ А.Г. Куприяшкин.– Норильск, 2015 г. – 135с.

10. Кудрявцев, Е. М. GPSS World Основы имитационного моделирования различных систем [Текст]/ Е.М. Кудрявцев. – М.: ДМК Пресс, 2004. – 320 с.