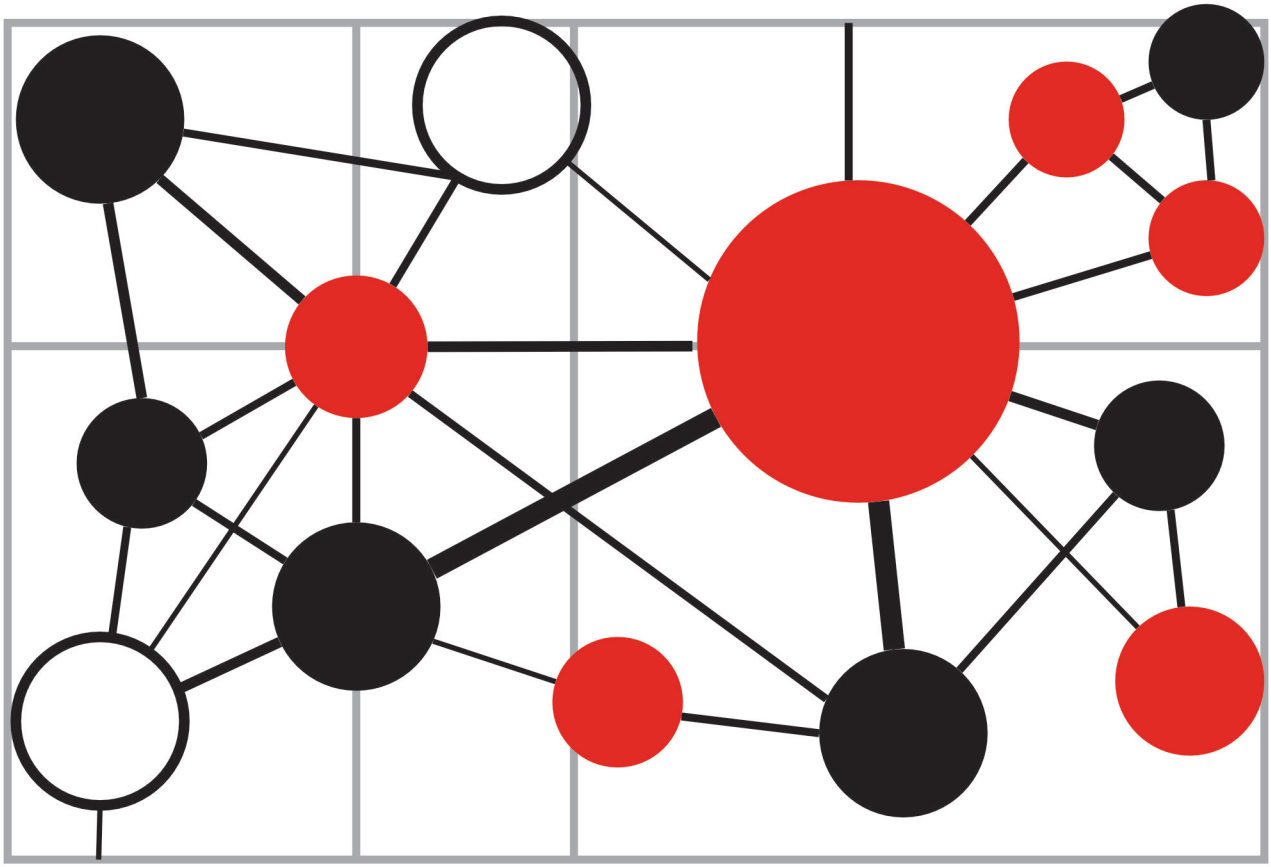
СИСТЕМНЫЙ АНАЛИЗ

А. П. Клишин, Н. Л. Ерёмина



Лабораторный практикум

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Томский государственный педагогический университет»
(ТГПУ)

А. П. Клишин, Н. Л. Ерёмина

СИСТЕМНЫЙ АНАЛИЗ

Лабораторный практикум

Электронное издание локального распространения

Томск
Издательство ТГПУ
2026

ISBN 978-5-907791-72-5

УДК 519.7: 378.147.88
ББК 74.480.46
К49

Рекомендовано к изданию
редакционно-издательским советом
Томского государственного
педагогического университета

Рецензент:
кандидат физико-математических наук, доцент кафедры системного анализа
и математического моделирования
Национального исследовательского Томского государственного университета
Ж. Н. Зенкова

Клишин, А. П.

К49 Системный анализ : лабораторный практикум [Электронный ресурс] / А. П. Клишин, Н. Л. Ерёмина. – Томск : Издательство Томского государственного педагогического университета, 2026. – 152 с. – Электрон. текстовые дан. (3,55 Mb). – 1 электрон. опт. диск (CD-ROM). – Загл. с титул. экрана.

ISBN 978-5-907791-72-5

Лабораторный практикум посвящён практическому освоению современного инструментария и основ ключевых технологий системного анализа. Представлены примеры анализа различных систем, рассмотрены методики структурного анализа и моделирования, а также даны соответствующие рекомендации по применению системного подхода.

Предназначен для студентов различных форм обучения по направлению подготовки 01.04.02 «Прикладная математика и информатика» (уровень магистратуры). Представленные материалы могут быть использованы студентами, аспирантами и преподавателями других технических и естественно-научных специальностей, а также специалистами, которые применяют технологии системного анализа в своей работе.

УДК 519.7: 378.147.88
ББК 74.480.46

Системные требования:
ПК не ниже класса Pentium II; RAM 512 Mb; Windows XP/7–10 (32-разрядная или 64-разрядная версии); разрешение экрана 1 024 × 768 (768 × 1 024); CD-ROM-дисковод, мышь; Adobe Acrobat Reader DC (либо другое, открывающее PDF-файлы).

ISBN 978-5-907791-72-5

Содержание

Список сокращений и условных обозначений

ABM (англ. Agent-Based Modeling, «агент-ориентированное моделирование») – метод имитационного моделирования, исследующий поведение сложных систем через моделирование действий и взаимодействий отдельных агентов – автономных сущностей (индивидов, организаций, групп, программ и т.д.).

ЛПР – лицо, принимающее решение, человек или группа лиц, от которого(ых) зависит решение проблемы.

ООС – отрицательная обратная связь.

ПОС – положительная обратная связь.

DES (англ. Discrete-Event Simulation, «дискретно-событийное моделирование») – вид имитационного моделирования, в котором функционирование системы представляется как хронологическая последовательность событий.

CLD (англ. Causal Loop Diagram, «причинно-следственные диаграммы») – графические диаграммы, предназначенные для качественного анализа сложных систем.

IDEF (агл. Integrated DEFinition, «интегрированное определение») – семейство стандартизированных методологий для комплексного моделирования различных аспектов сложных систем.

IDEF0 – методология, используемая для создания функциональной модели си-стемы. Позволяет создать графическую модель системы, отображающую её структуру, функции, а также потоки информации и материальных объектов, связывающие эти функции.

PEST-анализ (англ. Political, Economic, Social, Technological, «политические, экономические, социально-культурные и технологические факторы») – метод структурированного изучения макросреды как сложной системы; инструмент стратегического анализа внешней среды организации.

SADT (англ. Structured Analysis and Design Technique, «структурный анализ и проектирование») – методология структурного моделирования. Функциональный вариант SADT был формализован в рамках программы ICAM (англ. Integrated Computer Aided Manufacturing, «интегрированная компьютеризация производства») как стандарт IDEF0.

SD (англ. System Dynamics, «системная динамика») – направление в изучении сложных систем, фокусирующееся на исследовании их поведения во времени.

SFD (англ. Stock and Flow Diagram, диаграмма «запасы и потоки») – потоковая диаграмма, предназначенная для количественного анализа сложных систем на основе имитационного моделирования.

SRS (англ. Software Requirements Specification, «спецификация требований к программному обеспечению») – спецификация требований, формализованный документ, в котором подробно изложены требования к информационной системе.

SWOT-анализ (англ. Strengths, Weaknesses, Opportunities, Threats, «сильные, слабые стороны – внутренние факторы, возможности и угрозы – внешние факторы») – метод стратегического планирования, позволяющий оценить внутренние и внешние факторы, влияющие на систему.

Предисловие

Появление и становление системного анализа как направления современной науки управления связано с возникновением и накоплением большого массива сложных социальных, экономических, технических и политических проблем, обусловивших потребность в поиске и обосновании новых решений в различных сферах деятельности. Применение системного подхода в рамках решения научных, управленческих, проектных и технических задач способствует оптимизации процессов принятия решений за счёт повышения их оперативности и одновременного снижения вероятности возникновения ошибок. Наибольшая практическая значимость данного подхода наблюдается в таких сферах деятельности, как управление производственными процессами, проектирование, разработка информационных систем, сложных технических систем, а также техническая эксплуатация машин, оборудования и сооружений.

Лабораторный практикум включает цикл из восьми лабораторных работ, предусмотренных учебной программой курса «Системный анализ». Материал предназначен для обучающихся по основным профессиональным образовательным программам высшего образования (уровень магистратура) по направлению подготовки 01.04.02 «Прикладная математика и информатика». Для успешного выполнения лабораторных работ требуется предварительное освоение теоретического материала в рамках тематических разделов учебной дисциплины.

Лабораторные занятия по курсу «Системный анализ» призваны не только расширить теоретические знания обучающихся в области системного подхода, но и обеспечить освоение прикладных методов исследования систем. Ключевая задача – сформировать у студентов профессиональные компетенции, необходимые для успешного моделирования, прикладного анализа, проектирования и модернизации систем различной природы в своей образовательной и профессиональной деятельности.

Студенты выполняют лабораторные работы на занятиях либо индивидуально, либо небольшими группами по 2–3 человека. Каждому занятию предшествует самостоятельная работа, выполняемая студентом перед аудиторным занятием. Лабораторные работы завершаются оформлением отчётов, подлежащих защите до зачётной недели. Процесс контроля освоения материала включает текущий этап (доклад и ответы на вопросы) и промежуточный (устная проверка).

Лабораторная работа № 1
Проблемы и способы их решения

Тема занятия: проблемы и способы их решения.

Цель занятия: научится описывать проблемы и применять методы анализа проблемной ситуации, а также овладеть методиками SWOT- / PEST-анализа.

Материалы и программное обеспечение:

1. ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам.
2. Программное обеспечение: MS PowerPoint или аналог, графический редактор.
3. Текстовый редактор: MS Word / Writer LibreOffice, Google Docs.

Краткие теоретические сведения

Решение проблем проводится в том случае, если существуют некие обстоятельства в реальном мире, а также есть некий субъект, недовольный этими обстоятельствами и имеющий намерение их изменить.

Чтобы разделить субъектную и объектную составляющие одного явления, вводятся два понятия. *Проблемная ситуация* – объективно существующее стечение обстоятельств в реальном мире, вызывающее желание у кого-либо эту ситуацию изменить. *Проблема* – отрицательное отношение субъекта к рассматриваемой ситуации.

Таким образом, *проблема*, требующая решения, возникает у субъекта под действием объективных обстоятельств. В ней можно выделить три составляющие – *неблагоприятная ситуация, недовольный этой ситуацией субъект и желание субъекта изменить ситуацию*. При отсутствии любой из этих составляющих решение проблемы невозможно или неактуально. Так, при отсутствии желания изменений у субъекта дело заканчивается констатацией факта существования негативных для него обстоятельств.

Решением проблемы можно считать любые действия, которые снимают недовольство субъекта. Они могут быть направлены на изменение как самой ситуации, так и отношения к ней. Это означает, что, формулируя проблему, необходимо не только описать объективно сложившиеся обстоятельства, но и определить, кто является субъектом, в интересах которого будут совершены действия.

Рассмотрим сначала способы решения проблемы, связанные с *изменением отношения субъекта к ситуации*. Такой путь может быть выбран, если изменение реальности почему-либо невозможно или нежелательно.

Первым из таких способов является *информирование субъекта* о том, что ранее ему было неизвестно. Часто причиной негативного отношения к тем или иным обстоятельствам является недостаток информации у субъекта о ситуации. Дополнительное информирование может дать аргументы в пользу того, что казалось нежелательным, а

также ведет к появлению новых стратегий поведения. Кроме того, человеку вообще свойственно избегать недостаточной информации, поэтому чем больше информации он получит о ситуации, тем больше вероятность того, что его негатив уменьшится. Вместе с тем не в каждом случае человек желает иметь полную информацию – иногда информация может быть травматичной для него, так что данный способ не является универсальным.

Другой возможный способ – *изменить восприятие ситуации субъектом.* Изменение может быть осознанным, например, многие люди обращаются за помощью к коучам, психологам с целью выработать новые образцы (паттерны) мышления и поведения. Либо изменение может не осознаваться субъектом и даже осуществлено против его воли, например, под действием пропаганды, манипулятивных техник, физических и химических факторов и т.д. Очевидно, что при выборе способов изменения восприятия необходимо сначала определить, какие из них являются приемлемыми в каждом конкретном случае.

Третьим способом влияния на восприятие реальности субъектом является исключение *субъекта* из *ситуации*, вызывающей его негативное отношение. Так, достаточно часто производственный конфликт разрешается путем перемещения одного из его участников в другое подразделение или на другую позицию. Деловые качества человека, создающие препятствие в одном бизнес-процессе, могут быть полезными в другом. Такой способ также не всегда будет полезен, поэтому пути решения проблем в каждом случае зависят от множества обстоятельств.

Пути решения проблемы связаны с изменением *проблемной ситуации*, т.е. объективно существующих обстоятельств. Заметим, прежде всего, что человек по своей природе социален, т.е. существует и действует не сам по себе, а в обществе, поэтому в каждой ситуации, кроме интересов субъекта, проблема которого решается, необходимо учитывать интересы других участников.

Если рассматривать только интересы субъекта проблемы и изменять ситуацию, не принимая во внимание других, то велика вероятность того, что люди, недовольные изменением ситуации, могут противодействовать изменениям, и это может затруднять достижение результата в виде решения проблемы. Такой подход (*приоритет меньшинства*) также возможен и встречается на практике в разных вариантах – от принятия решения по принципу единоначалия до диктатуры в государстве. Он может быть результативен в некоторых ситуациях, например в военных действиях. Но, как правило, для его реализации необходима сила и готовность применить эту силу. С развитием человечества принятие решений, основанное на принуждении, становится неэффективным в среднесрочной и долгосрочной перспективе.

Другим подходом может быть *приоритет группы* – учет интересов не только субъекта, но и некой группы, к которой принадлежит субъект. Все участники группы рассматриваются как равноценные и важные, интересы других людей игнорируются. На практике этот подход также отличается вариабельностью – от националистической идеологии до спортивных сообществ. Его сильной стороной является возможность использовать в поддержку изменений ресурсов всей группы, слабой же стороной –

изначально присущие ему двойные стандарты, разделяющие общество на «своих» и «чужих» с разным, порой диаметрально противоположным отношением к этим двум категориям.

Прикладной системный анализ использует принцип *приоритета каждого*, в основе которого лежит мысль о том, что все субъекты различны, однако равноценны и равноправны. Поэтому решение проблем одних за счет других недопустимо. С точки зрения прикладного системного анализа единственно приемлемый путь изменения ситуации – осуществление *улучшающего вмешательства*.

Улучшающее вмешательство – изменение проблемной ситуации, которое положительно оценивается хотя бы одним из ее участников и неотрицательно – всеми остальными. Этот термин введен Расселом Линкольном Акоффом в конце 70-х гг. XX в.

Если субъект проблемы расценивает результат *улучшающего вмешательства* как положительный, а для остальных участников ситуация если не улучшилась, то не ухудшилась, можно констатировать, что решение исходной проблемы как минимум не создало новых проблем.

Улучшающее вмешательство принципиально возможно почти в любой ситуации («Улучшающее вмешательство часто трудно найти, но редко невозможно» – Р. Акофф), однако практическая реализация часто показывает, что несистемный подход к принятию решений и нехватка ресурсов приводят к негативному опыту. Поэтому *улучшающее вмешательство* можно рассматривать как идеал, к которому необходимо стремиться, в каждом конкретном случае добиваясь максимально возможного приближения.

Р. Акофф предложил классификацию типов вмешательств в ситуацию, осуществляемых в целях решения проблемы.

1. *Невмешательство* (англ. *absolution*). Англоязычное название этого типа вмешательства соответствует процедуре отпущения грехов католического священника: священник выслушивает прихожан, более ничего не предпринимая. Невмешательство в проблему может быть оправдано, если любой другой вариант приводит к худшему результату. В том случае, если естественный ход событий ведет к решению проблемы, невмешательство становится улучшающим вмешательством.

2. *Частичное решение* (англ. *resolution*). При недостатке ресурсов часто нет возможности полностью решить проблему, но возможно сгладить ее остроту.

3. *Оптимальное решение* (англ. *solution*). Оптимальным называется решение наилучшее в данных условиях. Для определения наилучшего решения необходимо сначала установить критерии сравнения, а также ограничения, которые определят возможные для реализации решения. Наилучшим считается решение, отвечающее всем ограничениям и максимально соответствующее критерию.

4. *Растворение проблемы* (англ. *dissolution*). При изменении угла зрения на проблему заданный формат условий может быть изменен. Такой новый взгляд на ситуацию может привести к расширению числа возможных альтернатив и проявлению новых эффективных решений.

Методика «Пять почему» (5 Why)

Примером «растворения проблемы» может служить методика «Пять почему», разработанная основателем компании Toyota японским инженером Сакити Тоёда. Позднее она вошла в состав наиболее востребованных инструментов, применимых в концепции бережливого производства. В соответствии с этой методикой необходимо сформулировать проблему и записать ее. Запись соответствует корневой вершине графа с древовидной структурой (рис. 1.1). Затем задают вопрос «Почему эта проблема образовалась?». Чаще всего причин будет несколько, и их записывают на следующем уровне дерева. После этого каждая причина анализируется как новая проблема, для нее повторяется процедура поиска причин. В результате образуются новые ветки дерева. Принято считать, что истинная причина проблемы проявляется примерно на пятой итерации. Замена исходной проблемы на истинную причину позволяет предложить эффективные решения, которые не могли быть найдены сразу, так как не соответствовали исходным условиям.

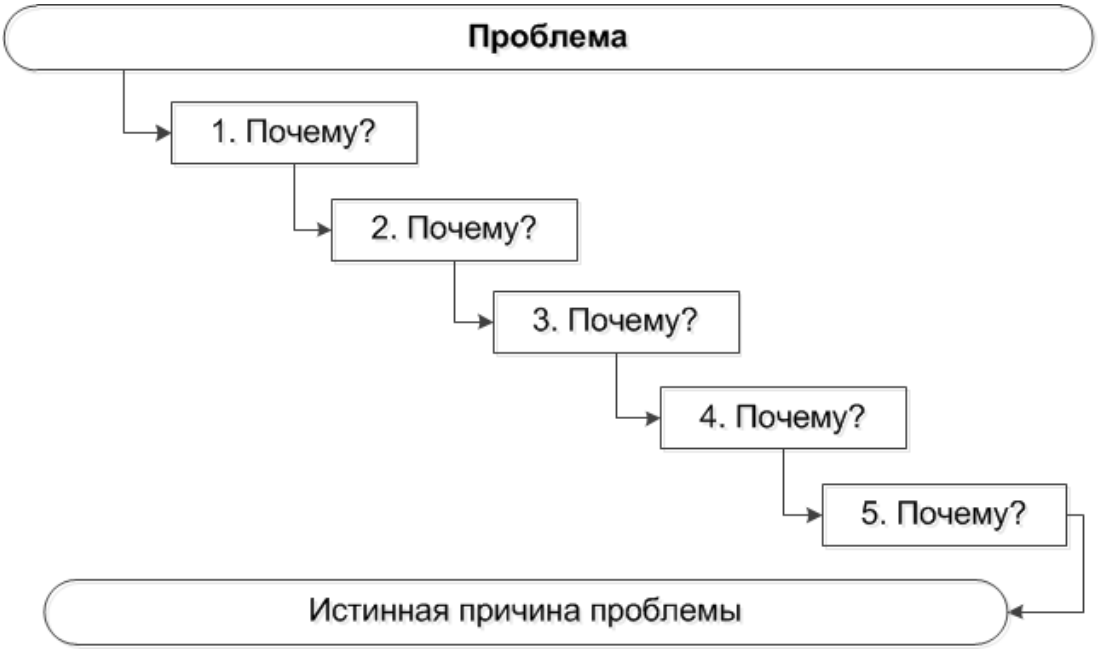


Рис. 1.1. Схема методики «Пять почему»

Пример 1. Представьте, что у вас сломался автомобиль. И вы начинаете задавать вопросы «Почему» и отвечать на них.

1. Автомобиль остановился. Почему?
2. Потому что была перегрузка и полетел предохранитель. Почему?
3. Потому что подшипник был плохо смазан. Почему?
4. Потому что насос, подающий смазку, плохо работал. Почему?
5. Потому что поршень износился и разболтался. Почему?
6. Потому что не поставили фильтр, и в поршень попала металлическая стружка.

SWOT-анализ

SWOT-анализ – метод стратегического планирования, который предполагает анализ четырёх групп факторов, влияющих на деятельность организации. Анализ помогает организациям определять их сильные и слабые стороны, а также выявлять возможности и угрозы, которые связаны с внутренней/внешней средой.

Аббревиатура SWOT состоит из четырёх английских слов: **s**trengths, **w**eaknesses, **o**pportunities, **t**hreats, которые обозначают следующее:

− S (Strengths) – сильные стороны, внутренние характеристики, которые дают компании преимущество перед конкурентами.

− W (Weaknesses) – слабые стороны, внутренние факторы, которые ставят фирму в невыгодное положение.

− O (Opportunities) – возможности, внешние условия, которые можно использовать для роста и развития.

− T (Threats) – угрозы, внешние обстоятельства, которые могут нанести вред бизнесу.

Компоненты SWOT-анализа. Сильные (S) и слабые (W) стороны – факторы внутренней среды, на которые компания может повлиять (например, команда, ресурсы, бизнес-процессы, репутация). Возможности (O) и угрозы (T) – факторы внешней среды, которые могут повлиять на объект извне и при этом не контролируются компанией (например, действия конкурентов, экономическая ситуация, законодательство).

Цель SWOT-анализа заключается в определении (поиске) преимуществ и недостатков в сложившейся ситуации. Это поможет, например, выявить конкурентные преимущества принятого решения (сильные и слабые стороны). Выявить риски означает определить потенциальные угрозы для бизнеса (принятого решения) до того, как они станут проблемой. Найти возможности означает обнаружить перспективные направления развития.

Методика SWOT-анализа. Сформулировать цель анализа – определить, что именно нужно проанализировать: организацию/компанию, конкретный продукт, программное решение или еще что-то.

Основные этапы SWOT-анализа:

1) Привлечь заинтересованных людей (стейкхолдеров) для получения разносторонних взглядов.

2) Подготовить информацию. Собрать данные о внешней среде, конкурентах, внутренних процессах.

3) Определить сильные стороны (S) – выявить внутренние преимущества, которые выделяют решение среди конкурентов.

4) Определить слабые стороны (W).

5) Найти возможности (О) – определить внешние факторы, которые положительно влияют на ситуацию и могут помочь улучшить решение.

6) Обозначить угрозы (Т) – выявить внешние факторы, которые негативно влияют на ситуацию и могут негативно повлиять на решение.

7) Заполнить матрицу SWOT-анализа. С использованием всех выявленных факторов заполнить соответствующие ячейки таблицы (табл. 1.1).

Таблица 1.1

Таблица для заполнения результатов SWOT-анализа

Тип факторов	Положительные стороны	Отрицательные стороны
Внешние факторы	***S*** (Strengths) Сильные стороны 1. 2. 3.	***W*** (Weaknesses) Слабые стороны 1. 2. 3.
Внутренние факторы	***O*** (Opportunities) Возможности 1. 2. 3.	***T*** (Threats) Угрозы 1. 2. 3.

Пример 2. SWOT-анализ для программного приложения киберспортивных клубов (табл. 1.2).

Таблица 1.2

Результат SWOT- анализа для программного приложения киберспортивных клубов

Тип факторов	Положительные стороны	Отрицательные стороны
Внешние факторы	1. Интеграция с игровыми платформами: Battle.net, Steam. 2. Модуль управления ареной, местами, тарифами и временем. 3. Встроенная система учета лояльности и баллов. 4. Поддержка сетевой синхронизации. 5. Возможность авторизации через аккаунт игрока	1. Сложность установки и настройки для небольших клубов. 2. Отсутствие облачной версии. 3. Недостаточный Ux/Uix-дизайн на мобильных приложениях. 4. Ограниченная локализация интерфейса
Внутренние факторы	1. Разработка мобильного приложения для игроков (рейтинги, оплата). 2. Выход на зарубежные рынки. 3. Партнерство с спонсорами. 4. Интеграция с популярными интернет-платформами	1. Увеличение конкуренции со стороны Saas-платформ. 2. Рост затрат на поддержку серверов и лицензий. 3. Снижение интереса к офлайн-киберспорту. 4. Угроза кибербезопасности

PEST-анализ

PEST-анализ – инструмент стратегического планирования, который помогает оценить влияние внешних факторов на бизнес. Аббревиатура расшифровывается как P (Political) – политические факторы, E (Economic) – экономические факторы, S (Social) – социально-культурные факторы, T (Technological) – технологические факторы.

Основные факторы, которые исследуют в PEST-анализе:

- *Политические:* государственная политика, законодательство, регулирование отрасли, налоговая система, торговые пошлины.
- *Экономические:* уровень инфляции, валютные курсы, процентные ставки, налоговая нагрузка, уровень безработицы, покупательская способность населения.
- *Социально-культурные:* демографические изменения, уровень образования, культурные ценности, отношение к труду, мода и стиль жизни, социальная мобильность.
- *Технологические:* новые технологии производства, научные разработки, инновации, уровень внедрения IT-технологий.

Бывает, что один и тот же тренд относится к нескольким факторам. Например, запрет на экспорт определенных товаров можно рассмотреть и с политической, и с экономической точки зрения.

Цели PEST-анализа:

- *Раннее выявление рисков и угроз* – поскольку оперативно выявленные изменения в макросреде позволят своевременно адаптироваться: изменить стратегию, диверсифицировать деятельность.
- *Поиск новых возможностей* – например, изменения в законодательстве могут открыть рынок, рост доходов населения – расширить спрос, а новые технологии – создать шанс для инноваций.
- *Обоснование стратегических решений* – PEST-анализ помогает аргументировать, почему стоит выходить на новый рынок, инвестировать в технологию или менять продуктовую линейку.

Методика и основные этапы PEST-анализа:

1. *Выявление факторов* – обычно проводят мозговой штурм, где эксперты в группах или индивидуально определяют значимые элементы.
2. *Сбор актуальных данных* о состоянии факторов – используют отчеты, исследования и публикации в специализированных изданиях.
3. *Оценка влияния каждого фактора* на компанию – используют баллы по шкале от 1 до 3: 1 – незначительное, 2 – умеренное, 3 – критическое влияние.
4. *Оценка вероятности изменений факторов* в будущем – эксперты оценивают вероятность по пятибалльной шкале, вычисляют среднее арифметическое оценок, что позволяет выявить наиболее рискованные факторы.

5. *Сопоставление влияния факторов и вероятности их изменений* – используют различные формулы, но одна из простейших – вычисление среднего арифметического оценок вероятности и добавление к нему балла из колонки «Влияние».

6. *Создание матрицы*, в которой факторы располагаются по степени важности и формулируются решения (табл. 1.3).

Таблица 1.3

Матрица, используемая для заполнения результатов PEST-анализа

P Политические факторы 1. 2. 3.	***E*** (Economic factors) Экономические факторы 1. 2. 3.
S (Social factors) Социально-культурные факторы 1. 2. 3.	***T*** **(**Technological factors**)** Технологические факторы 1. 2. 3.

Ключевое отличие от SWOT-анализа заключается в том, что SWOT-анализ изучает и внешние, и внутренние факторы, воздействующие на компанию, тогда как PEST-анализ только внешние факторы, т.е. влияние внешней среды. В современных исследованиях, как правило, PEST-анализ является первым этапом SWOT-анализа.

Пример 3. Заполненные таблицы в результате выполнения PEST-анализа в развернутой форме (табл. 1.4).

Таблица 1.4

Результаты PEST-анализа

Фактор	Влияние фактора	Эксперты (степень влияния)				Средняя оценка	Весовой коэфф.
		$Э_1$	$Э_2$	$Э_3$	$Э_4$		
1. Политические факторы							
Фактор 1.1	3	4	3	4	5	4	0,05
Фактор 1.2	1	1	2	1	1	1,25	0,16
Фактор 1.3	2	1	4	2	1	2	0,12
2. Экономические факторы							
Фактор 2.1	2	4	2	1	3	2,5	0,08
Фактор 2.2	2	3	2	4	3	3	0,04
Фактор 2.3	3	3	4	4	4	3,5	0,05
Фактор 2.4	1	5	4	5	4	4,5	0,01
3. Социокультурные факторы							
Фактор 3.1	3	5	5	5	4	4	0,04
Фактор 3.2	2	4	3	5	4	4,75	0,03
Фактор 3.3	2	4	5	3	5	4,25	0,03
Фактор 3.4	1	2	1	4	5	3	0,03

Фактор	Влияние фактора	Эксперты (степень влияния)				Средняя оценка	Весовой коэфф.
		$Э_1$	$Э_2$	$Э_3$	$Э_4$		
4. Технологические факторы							
Фактор 4.1	2	4	3	4	4	3,75	0,04
Фактор 4.2	2	3	3	3	4	3,25	0,05
Фактор 4.3	3	5	4	5	5	4,75	0,03
Фактор 4.4	1	4	3	3	3	3,25	0,02
Итого						51,75	

Пример 4. PEST-анализ для IT-компании (табл. 1.5).

Определяем факторы:

1. Политические:

1.1. Налоговые льготы.

1.2. Обстановка на международной арене.

2. Экономические:

2.1. Нестабильный курс валют.

2.2. Сложность с получением международных переводов.

3. Социокультурные:

3.1. Всплеск на рынке IT-образования.

3.2. Популярность программирования.

4. Технологические:

4.1. Уход с российского рынка некоторых зарубежных провайдеров и сервисов.

4.2. Проблемы с параллельным импортом оборудования.

Таблица 1.5

Результаты PEST-анализа для IT-компании

Фактор	Влияние фактора	Эксперты (степень влияния)					Средняя оценка	Весовой коэфф.
		$Э_1$	$Э_2$	$Э_3$	$Э_4$	$Э_5$		
1. Политические факторы								
1.1. Налоговые льготы	1	2	3	3	4	5	3,4	0,23
1.2. Обстановка на международной арене	3	4	4	4	5	5	4,4	0,88
2. Экономические факторы								
2.1. Нестабильный курс валют	2	1	3	2	4	1	2,2	0,29
2.2. Сложность с получением международных переводов	3	3	2	1	5	5	3,2	0,64
3. Социокультурные факторы								
3.1. Всплеск на рынке IT-образования	2	3	4	2	4	5	3,6	0,48
3.2. Популярность программирования	1	2	3	3	3	2	2,6	0,17

Фактор	Влияние фактора	Эксперты (степень влияния)					Средняя оценка	Весовой коэфф.
		$Э_1$	$Э_2$	$Э_3$	$Э_4$	$Э_5$		
4. Технологические факторы								
4.1. Уход с российского рынка некоторых зарубежных провайдеров и сервисов	3	3	1	4	2	1	2,2	0,44
4.2. Проблемы с параллельным импортом оборудования	2	2	3	4	4	1	2,8	0,37
Итого	17						51,75	1

Оценка влияния фактора (весовой коэффициент) рассчитывается по формуле, приведенной ниже:

$$\text{Оценка влияния фактора} = \textit{влияние фактора} \Big/ \left(\sum_{i=1}^{n} Э_i \right) \times (\textit{среднее значение}).$$

Рассчитаем *оценку влияния фактора* 1.1.

Влияние фактора = 1.

Сумма влияний = 2 + 3 + 3 + 4 + 5 = 17.

Среднее значение = (2 + 3 + 3 + 4 + 5) / 5 = 3,4.

Оценка влияния фактора 1.1 = 1/(17 × 3,4) = 0,23.

Задания к работе

1. Сформулировать проблему (выбор можно осуществить из выпускной квалификационной работы) либо «получить» у преподавателя. Проблема должна соответствовать выбранной вами специальности и профилю обучения.

2. Провести SWOT/PEST-анализы по согласованию с преподавателем.

3. Применить методику «Пять почему».

4. Привести пример проблем(ы), которую, с вашей точки зрения, невозможно решить.

Методика выполнения работы

Перед выполнением задания необходимо ознакомиться со стандартом ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам, а также изучить рекомендуемую литературу. Создать и заполнить табл. 1.6 с описанием выбранной проблемы и сформулировать возможные пути ее решения, заполнить столбец 2 (табл. 1.6). В малых группах (2–3 человека) обсудить выбранную проблему. По результатам обсуждения заполнить столбец 3 (табл. 1.6). Заполнить матрицу с данными SWOT/PEST-анализов. Провести необходимые расчеты.

Рекомендации по обработке и оформлению полученных результатов

Сформировать текстовый документ в электронной форме (.docx) и представить преподавателю. Текст необходимо отформатировать в соответствии с ГОСТ Р 2.105-2019. Единая система конструкторской документации. Общие требования к текстовым документам. Подготовить презентацию по проблеме выпускной квалификационной работы или проблеме, которую вам предоставил преподаватель.

Таблица 1.6

Анализ проблемной ситуации

Вопрос	Описание проблемы и возможных путей решения	Дополнения, сделанные в ходе группового обсуждения
1	2	3
1. Краткое наименование проблемы		
2. Характеристика проблемной ситуации (объективно существующих обстоятельств)		
3. Субъект проблемы		
4. Характеристика отношения субъекта к проблемной ситуации		
5. Какую дополнительную информацию можно предоставить субъекту, чтобы изменить его отношение к ситуации?		
6. Какие приемлемые способы воздействия на позицию субъекта можно предложить?		
7. Возможно ли исключить взаимодействие субъекта с проблемной ситуацией? Если да, то каким образом?		
8. Необходимо ли учитывать мнение других участников ситуации? Почему?		
9. В каком случае проблему можно решить путем невмешательства в нее?		
10. Какие частичные методы решения проблемы могут быть предложены?		
11. Возможно ли предложить оптимальное решение проблемы?		
12. Возможно ли «растворить» проблему?		

Вопросы для самоконтроля

1. Дать определения проблемной ситуации и проблемы. В чем различие между ними?

2. Перечислить три способа решения проблемы без изменения объективной ситуации. В каких случаях их можно использовать? Привести примеры из реальной жизни или профессиональной сферы.

3. Возможно ли решение проблемы для отдельно взятого субъекта в отрыве от других участников ситуации? Перечислить возможные подходы к рассмотрению интересов других людей, охарактеризовать их положительные и отрицательные стороны.

4. Что такое улучшающее вмешательство? Что может помешать осуществлению улучшающего вмешательства в реальности?

5. Охарактеризовать четыре типа вмешательств в ситуацию. Для каждого типа привести примеры ситуаций, в которых такое вмешательство будет наиболее эффективным.

Варианты заданий

1. Повышение качества обучения (успеваемости). Построение информационной системы средней общеобразовательной школы.
2. Формирование гибкого учебного расписания в школе.
3. Переподготовка учительского состава в средней школе.
4. Тестирование остаточных знаний обучающихся.
5. Формирование электронного портфолио обучающегося.
6. Формирование электронных ведомостей.
7. Проблемы школьной библиотеки. Малый фонд литературы, устаревшая литература, средний возраст библиотекаря, поиск новых форм привлечения к чтению и т.д.
8. Привлечение обучающихся (школа/вуз) к научно-исследовательской деятельности.
9. Привлечение обучающихся (школа/вуз) к проектной деятельности.
10. Поиск гостиницы в городе.
11. Заказ такси.
12. Выбор туристического агентства.
13. Выбор страховой компании.
14. Ремонт автомобиля.
15. Проектирование информационной системы «Склад».
16. Покупка лекарств в аптеке.
17. Проектирование информационной системы компьютерного магазина.
18. Проектирование информационной системы строительной компании.
19. Проектирование информационной системы транспортной компании.
20. Покупка/продажа билетов в аэропорту.

Рекомендуемая литература

1. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

2. Акофф, Р. Искусство решения проблем / Р. Акофф. Москва : Мир, 1982. 224 с.

3. Петренко, Е. С. «Пять почему?» – метод анализа проблем качества / Е. С. Петренко // Стандарты и качество. 2012. № 7. С. 60–63.

4. Майсак, О. С. SWOT-анализ: объект, факторы, стратегии. Проблема поиска связей между факторами / О. С. Майсак // Прикаспийский журнал: управление и высокие технологии. 2013. № 1 (21). С. 151–157.

5. Кузьменко, О. В. PEST-анализ в системе стратегического маркетингового анализа / О. В. Кузьменко, В. Н. Чекарь, С. В. Мостипан // Экономика и бизнес. Т. 2, № 96. С. 217–223.

Лабораторная работа № 2
Постановка целей и целеполагание

Тема занятия: постановка целей и целеполагание.

Цель занятия: формулирование и постановка целей, выявление проблематики и построение дерева целей и дерева проблем.

Материалы и программное обеспечение:

1. Программное обеспечение для компьютерной презентации: PowerPoint или аналог, графический редактор.

2. Программное обеспечение для моделирования потоков работ: MS Visio, Draw.io, Visual Paradigm, ConceptDraw PRO.

Краткие теоретические сведения

При решении реальных проблем и задач часто приходится сталкиваться с трудностями и ограничениями, которые можно разделить на два класса:

1. Субъективные, которые следуют из формы понимания действительности субъектом и системы ценностей субъекта.

2. Объективные, в свою очередь, представляют собой воздействие законов природы и ресурсные ограничения.

При формулировании задач (целеполагание) системного анализа необходимо учитывать наличие внутренних и внешних факторов (ресурсов), влияние окружающей среды, а также времени. В некоторых случаях достижение цели при рассматриваемых ограничениях становится невозможным. Тогда аналитик должен поставить перед лицом, принимающим решения, вопрос об ослаблении ограничений или вообще их снятии.

Как правило, при выделении проблемы в исследуемом объекте становится понятно, что проблема может быть связана с другими проблемами. При решении, таким образом, поставленной проблемы может возникнуть несколько проблем в соседних объектах, которые связаны с нашим объектом исследования.

Проблематика – это совокупность взаимосвязанных проблем в соседних объектах, которые неразрывно связаны с проблемой, подлежащей решению, и существующих в рамках системы и её взаимодействия с окружающей средой.

При нахождении решения различных проблем в системном анализе используется подход, в котором проблемы обычно расширяют до границ проблематики. В этом случае проблема(ы) рассматривается(ются) не изолированно, а наоборот, обращается внимание на существенные связи других проблем с исследуемой проблемой, без учета которых она не может быть решена.

При рассмотрении проблемы учитываются мнения, прежде всего, группы заинтересованных лиц (англ. *stakeholders,* «стейкхолдеров»).

К *стейкхолдерам* можно отнести следующие группы:

1. *Заказчик(и)* – лицо(а), которое(ые) ставит(ят) проблему и оплачивает(ют) проведение системного анализа и поиска решения.

2. *Лицо, принимающее решение* (ЛПР), – человек или группа лиц, от которого(ых) зависит решение проблемы.

3. *Участники* – группа лиц, от активных действий которых зависит решение проблемы или на которых повлияет решение проблемы (пассивные участники).

4. *Системный аналитик* – лицо, которое непосредственно осуществляет системный анализ (поиск решения).

Среди стейкхолдеров могут встречаться группы, которые поддерживают выбранный способ решения проблемы, а также те, которые не поддерживают его. Таким образом, у выделенных групп (стейкхолдеров) со временем складывается определенное мнение, виденье проблемы, формируется отношение к ней, которое зависит, прежде всего, от того, как решение этой проблемы повлияет на решение их собственных проблем.

Требования к цели

Правильно сформулированные цели должны обладать следующими основными свойствами:

1. *Конкретность* – при определении цели необходимо, чтобы она точно отражала содержание, объем и время. Достижение цели может принести только конкретный результат, полученный с помощью конкретных средств и в конкретных условиях.

2. *Измеримость* – цель необходимо описать количественными параметрами (показателями) для оценки степени ее достижимости.

3. *Согласованность* – цели необходимо рассматривать не изолированно, а учитывать взаимную связь целей друг с другом, их взаимную подчинённость по иерархии и взаимное влияние.

4. *Достижимость* – реалистичность цели, поставленной с учётом возможностей, ограничений и имеющихся ресурсов.

5. *Гибкость* – возможность менять приоритеты, адаптироваться к изменениям, обновлять и пересматривать планы в соответствии с текущей ситуацией и происходящих в среде изменений.

Анализ постановок целей (целеполагание)

Задача формулирования общей цели в сложных системах сводится к структуризации цели. Чтобы облегчить целеполагание, используют декомпозицию цели в виде упорядоченного или неупорядоченного набора подцелей, которые представляют цель более понятной для всех участвующих в процессе целеполагания. Для наименований подцелей в различных приложениях используются следующие термины: направления, задачи, подзадачи и на нижних уровнях функции.

Любая цель обладает свойством двойственности: является собственно целью и средством для достижения вышестоящей цели, поэтому описание отношений между целями и средствами можно представить в форме схемы (графа), которую называют *дерево целей*. Термин введен У. Черчменом в 1957 г., который предложил и использовал метод «дерево целей» в связи с проблемами принятия решений в промышленности.

В случае применения метода *«дерево целей»* в качестве средства принятия решений используют термин *дерево решений*. Если метод используется для выявления и уточнения функций (информационной) системы, то говорят о *дереве целей и функций*. При систематизации проблематики и тематики исследований используют термин *дерево проблем*, а при создании прогнозов – *дерево прогнозирования развития* или *прогнозный граф*.

При системном анализе организаций или сложных систем с целью их реорганизации или автоматизации используются следующие виды деревьев:

1) *дерево целей* объекта (*дерево желаний*);

2) *дерево проблем* субъекта (диаграмма Исикавы);

3) *дерево целей* субъекта;

4) *дерево решений* (*дерево стратегий*).

Целеполагание, таким образом, сводится к следующим этапам:

1. Выполняется построение *дерева целей* с точки зрения объекта, в котором представлены желания объекта с точки зрения его существования и развития. Далее они декомпозируются на более детальные цели, удовлетворение которых приводит к достижению исходных.

2. Проводится анализ удовлетворения потребностей и желаний объекта (оценивается степень проблемности) и выделяется(ются) ключевая(ые) проблема(ы), которая(ые) декомпозируется(ются) в форме *дерева проблем*.

3. На основе результатов этапа 2 строится *дерево целей* с позиции субъекта как отражение *дерева проблем*.

4. Строится *дерево решений* (*дерево стратегий*), в котором *дерево целей* дополняется вариантами решений выявленных проблем (стратегии).

Для формулировки целей далее необходимо решить два вопроса:

1) рассмотреть *проблематику* – множество проблем, которое порождается решением наших проблем. Это может вызвать проведение коррекции наших целей (например, в случае если окажется, что решение наших проблем приводит к еще более серьезным проблемам);

2) провести аналитический или экспертный анализ для выбора альтернатив решения проблем.

После выбора альтернатив (стратегий) формулируются задачи (функции) и назначаются исполнители.

Построение *дерева целей* (вариант 1)

На начальном этапе целеполагания формулируется основная цель (желание). Формулировка цели строится, как правило, из глагола – «действие и пояснение», под целью

понимается некоторый объект (состояние). Построение *дерева цели* начинается с структуризации цели, расчленения основной цели на составные элементы (подцели), каждая из которых является средством, направлением или этапом достижения основной цели. Затем каждая подцель рассматривается как цель и подвергается разделению на компоненты. Если графически изобразить все полученные элементы (подцели), то получим *дерево целей*. Деление прекращается тогда, когда подцель становится неделимой и объективно измеримой.

Построение *дерева целей* происходит по следующим правилам:

1. Если очередная подцель является средством для предыдущей, то она опускается вниз на один уровень (относительно предыдущей).

2. Если она является целью для предыдущей, то поднимается вверх на один уровень (относительно предыдущей).

3. Если она является ни целью, ни средством достижения, то остается на том же уровне иерархии.

Проверку полноты, непротиворечивости и правильности построения *дерева целей* можно выполнить с использованием четырех правил:

1. При чтении сверху вниз подцель должна отвечать на вопрос: что нужно сделать, чтобы реализовать подцель нижнего уровня?

2. При чтении снизу вверх цель более высокого уровня должна отвечать на вопрос: для чего необходима цель, лежащая непосредственно под ней?

3. При анализе подцелей, необходимых для достижения одной цели, следует уточнить, все ли подцели этого уровня необходимы для ее достижения?

4. При анализе подцелей, необходимых для достижения одной цели, следует уточнить, какие еще подцели этого уровня необходимы для ее достижения?

Пример 1. Дерево целей для коммерческого учебного центра представлено на рис. 2.1.

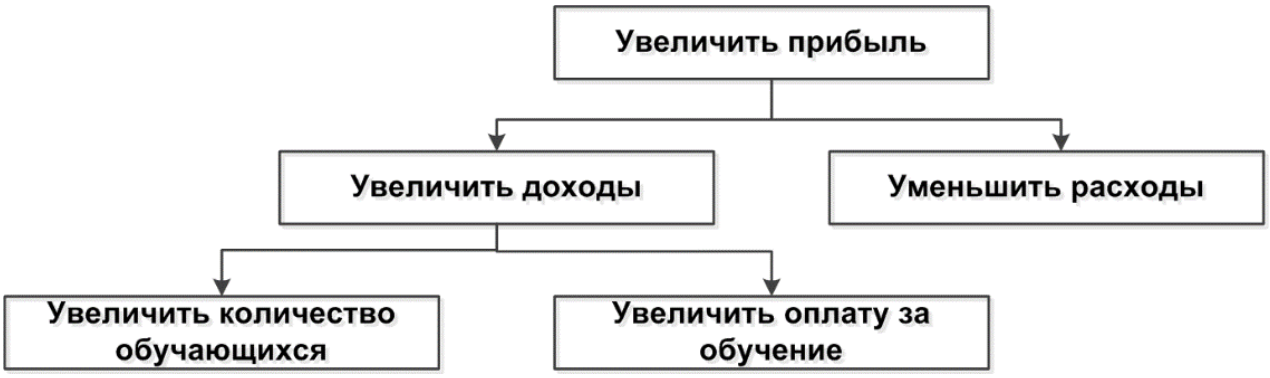


Рис. 2.1. *Дерево целей* для коммерческого учебного центра

Пример 2. Дерево целей для программного проекта представлено на рис. 2.2.

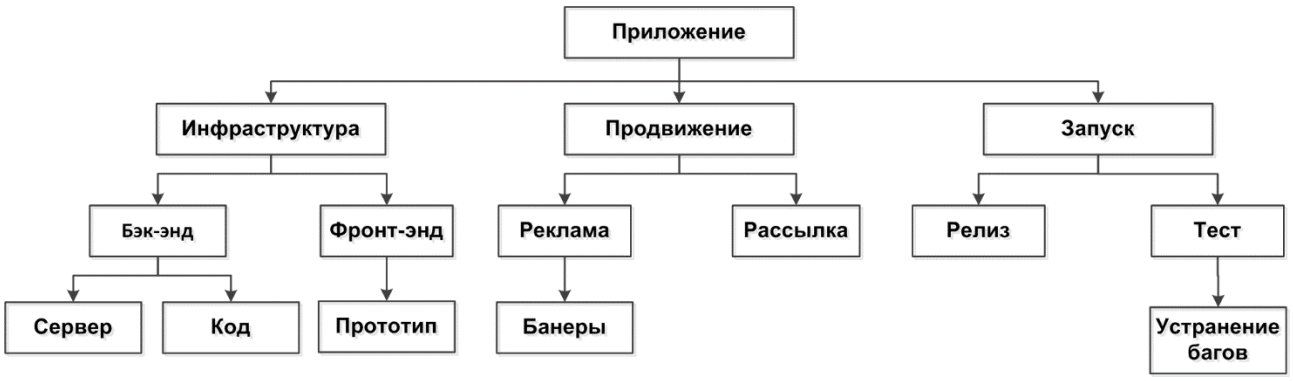


Рис. 2.2. *Дерево целей* для программного проекта

Далее полученную цель (множество целей) будем обозначать (вариант 1), она представляет цель с точки зрения объекта.

Построение *дерева проблем*

На начальной стадии изучения проблематики, как правило, ставится одна ключевая проблема, которая препятствует достижению желаемой цели и находящейся на вершине иерархии *дерева целей*. После этого выстраиваются другие проблемы в *дерево проблем*, при этом руководствуются следующими правилами:

– если очередная проблема является причиной для предыдущей, то она опускается на уровень ниже первой;

– если проблема является следствием для предыдущей, то она поднимается на один уровень вверх;

– если она не является ни причиной, ни следствием, то остается на том же уровне.

Пример *дерева проблем* для коммерческого образовательного центра представлен на рис. 2.3.

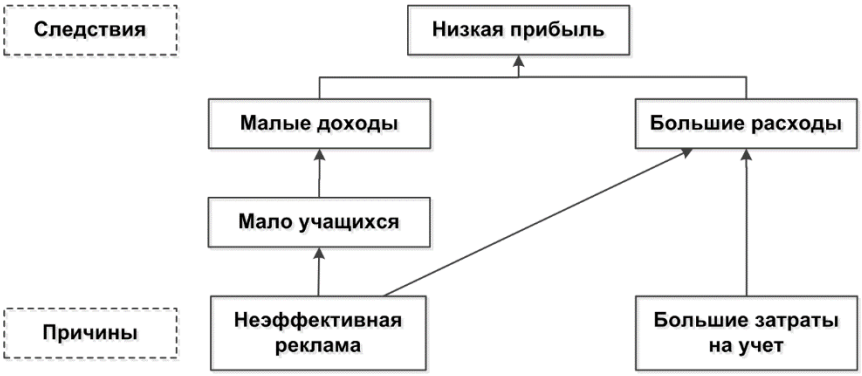


Рис. 2.3. *Дерево проблем* для коммерческого образовательного центра

Диаграмма Исикавы

Для описания *дерева проблем* используется математический аппарат теории графов. Один из вариантов изображения графа, отражающего *дерево проблем*, носит

название *диаграмма Исикавы*, или *рыбий скелет* (стандартное *дерево проблем* представлено на рис. 2.4).

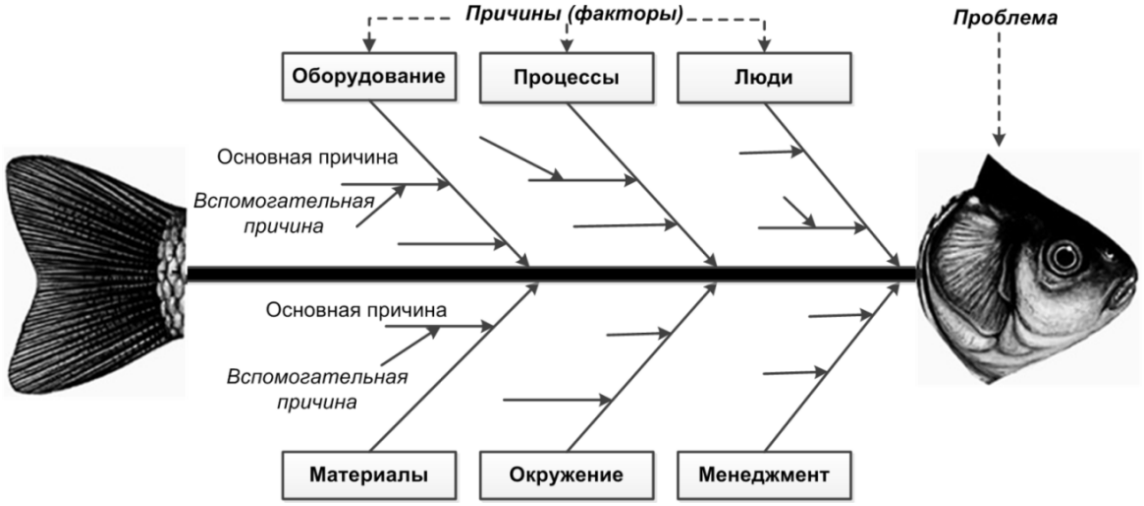


Рис. 2.4. Диаграмма Исикавы

Основная стрелка, проходящая слева направо («рыбий хребет»), соответствует исследуемой проблеме («голове рыбы»). Причины возникновения проблемы объединяются в группы («ребра»), при необходимости углубленного анализа можно добавлять ребра второго и последующего уровней. По сути, диаграмма представляет собой классификатор причин возникновения проблемы в графической форме.

Построение диаграммы Исикавы

Пример 3. Построение диаграммы Исикавы для проблемы повышения качества подготовки специалистов в вузе.

На рис. 2.5 представлена реализация диаграммы, построенная в программе MS Visio.

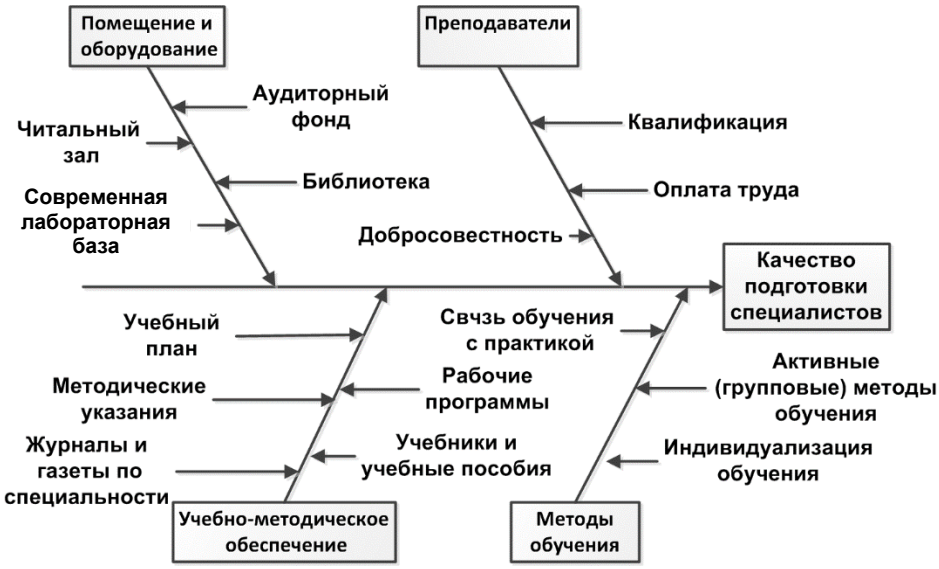


Рис. 2.5. Диаграмма Исикавы

Пример 4. Построение диаграммы Исикавы для проблемы повышения экономической эффективности предприятия.

На рис. 2.6 представлена реализация диаграммы, построенная в программе MS Visio.

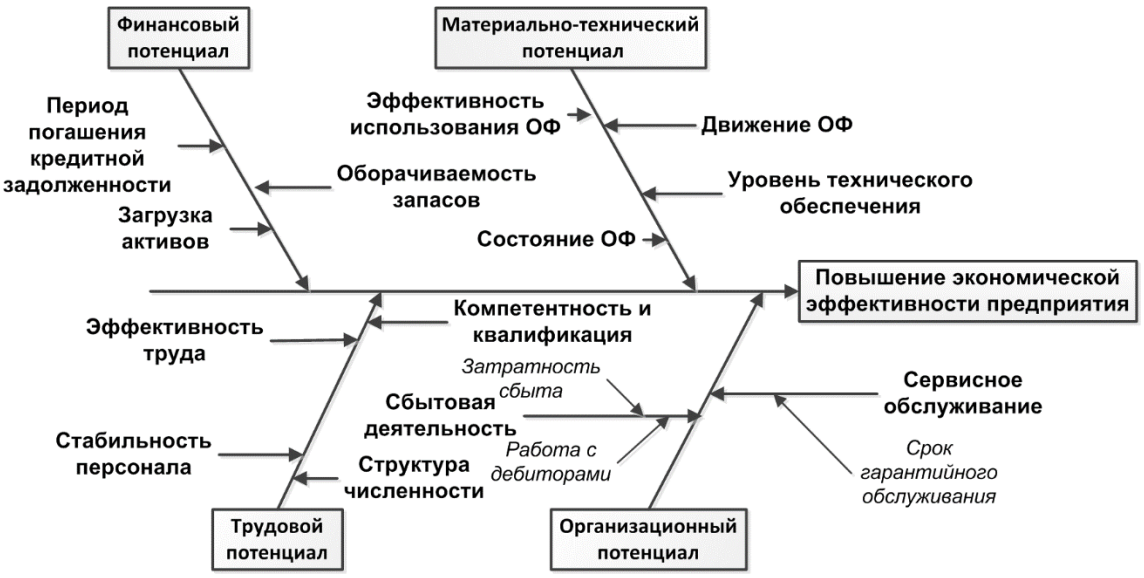


Рис. 2.6. Диаграмма Исикавы

Построение *дерева целей* (вариант 2)

После проведенного анализа *дерева проблем* и изучения проблематики, можно построить *дерево целей* с точки зрения субъекта, при этом обратив симметрично *дерево проблем* в позитивном ключе.

Далее полученную цель (множество целей) будем обозначать (вариант 2), она представляет цель с точки зрения субъекта.

Пример 5. Дерево целей (вариант 2) для коммерческого образовательного центра (рис. 2.7).

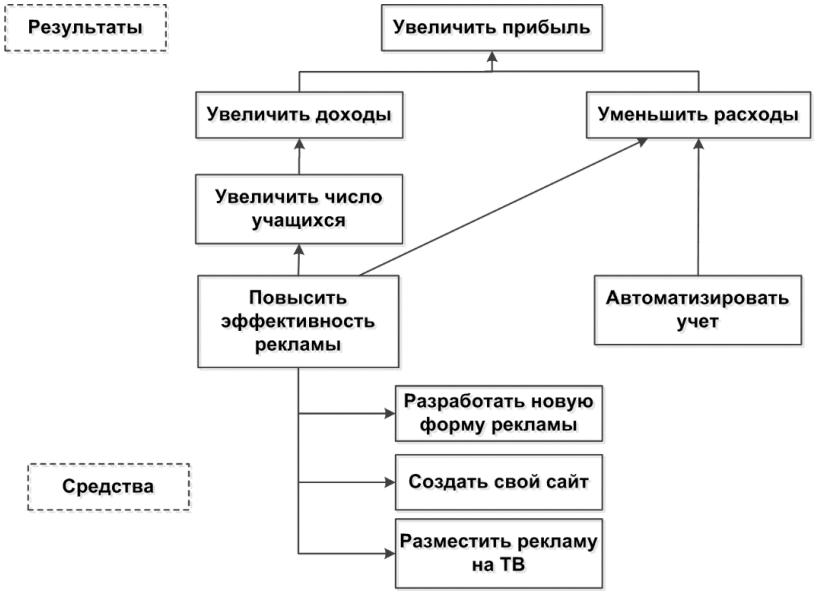


Рис. 2.7. *Дерево целей* (вариант 2) для образовательного центра с вариантами стратегий

Задания к работе

1. Представить описание проблематики исследуемой темы.
2. Найти и описать стейкхолдеров и возможные их группы.
3. Построить *дерево целей* (вариант 1).
4. Построить *дерево проблем* (диаграмму Исикавы).
5. Предложить стратегии для решения проблем(ы).

Методика выполнения работы

В малых группах (2–3 человека) обсудить выбранную проблематику. По результатам обсуждения провести необходимые построения.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет по выбранной проблеме. Сформировать текстовый документ в электронной форме (.docx) и представить преподавателю.

Вопросы для самоконтроля

1. Перечислить основные требования к целеполаганию и постановке цели.
2. Как называется набор лиц, заинтересованных в решении проблемы?
3. Что понимается под точкой зрения в целеполагании?
4. Указать причины построения *дерева целей* и *дерева проблем*.
5. Чем отличается построение *дерева проблем* от диаграммы Исикавы?

Рекомендуемая литература

1. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

2. Качала, В. В. Основы теории систем и системного анализа / В. В. Качала. Москва : Изд-во Горячая линия – Телеком, 2012. 210 с.

3. Как искать причины проблем с помощью «рыбьих костей» Исикавы: разбираем на примере [Электронный ресурс]. URL: https://skillbox.ru/media/management/kak-iskat-prichiny-problem-s-pomoshchyu-rybikh-kostey-isikavy-razbiraem-na-primere/ (дата обращения: 02.12.2025).

Лабораторная работа № 3
Модели и моделирование

Тема занятия: моделирование как деятельность.

Цель занятия: сформировать понятие моделирования как неотъемлемой части познавательной и исследовательской деятельности.

Материалы и программное обеспечение:

1. ГОСТ Р 57700.22-2020. Компьютерные модели и моделирование. Классификация.
2. ГОСТ Р 57412-2017. Компьютерные модели в процессах производства и эксплуатации изделий. Общие положения.
3. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
4. Программное обеспечение RAMUS.
5. Программное обеспечение для моделирования: MS Visio, Draw.io.

Краткие теоретические сведения

Моделирование (построение *модели*) является неотъемлемой частью любой деятельности человека по изучению и (или) преобразованию окружающего мира. Понятие *модели* может быть истолковано узко либо широко. Так, в науке и технике под *моделью* некоего объекта понимают другой объект, воспроизводящий принципиально важные свойства и характеристики исходного объекта, но более удобный для исследования, часто за счет его упрощения. В философии абстрактную *модель* определяют как отображение одной абстрактной структуры в другую либо как результат интерпретации одной структуры в терминах другой.

Модель – упрощенное подобие объекта, которое воспроизводит интересующие нас свойства и характеристики объекта оригинала или объекта проектирования.

Моделирование – деятельность, связанная с построением модели для какого-либо объекта или явления. Можно представить моделирование как процесс исследования или воспроизведения (имитация) свойств реального или создаваемого объекта с помощью другого объекта, процесса или явления.

Моделирование – построение, изучение, исследование и применение *моделей* реально существующих или проектируемых объектов.

Процесс построения модели всегда подчинен основной цели деятельности. В зависимости от конкретной цели модель наделяется теми свойствами исходного объекта или процесса, которые являются существенными для достижения этой цели. Поэтому количество возможных моделей, соответствующих каждому объекту или процессу, может быть достаточно большим или бесконечным. Основной задачей *моделирования* является выбор из этого множества той модели, которая наиболее точно позволяет достичь поставленной цели.

Причины проведения *моделирования*:

1. Высокая сложность реальных объектов. Реальные объекты имеют, как правило, большое (бесконечное) число свойств, которое описывается множеством параметров. Для упрощения и сокращения параметров используют конечные множества, которые описывают упрощенные модели объектов.

2. Необходимость проведения экспериментов. Экспериментальные исследования объектов бывают очень дорогими (иногда невозможными), поэтому использование моделирования является альтернативой для нахождения необходимой информации об объекте.

3. Прогнозирование. Если требуется проследить изменения объекта во времени, то моделирование помогает ответить на вопросы, как будут изменяться параметры объекта со временем.

Моделирование

Модели можно разделить на классы по разным основаниям. Рассмотрим классификацию моделей по целевому назначению, а именно с точки зрения их теоретической или практической направленности.

В современном мире в своей обширной деятельности, прежде всего в точных науках, большая часть времени человека связана с решением экспертных (познавательных) и конструктивных (прагматических) задач.

Описательное моделирование

Описательное (дескриптивное, познавательное) моделирование – построение моделей, предназначенных для описания свойств или поведения реально существующих объектов, процессов или явлений.

В отличие от нормативных («конструктивных») моделей, которые задают желаемое состояние (образец, стандарт), описательные модели отражают существующее положение дел. Их цель – максимально точно зафиксировать и представить знания о действительности, поэтому говорят, что они являются формой организации и представления знаний. В системном анализе такие модели называют «Как есть» (AS IS). Поскольку в процессе моделирования новые знания соединяются с ранее полученными, может возникнуть расхождение между моделью и реальностью. В этом случае модель необходимо изменить так, чтобы она соответствовала реальности. Примером познавательной модели может служить любая научная теория, например, волновая или корпускулярная теории света, таблица химических элементов Д. И. Менделеева и др.

Цели описательного (познавательного) моделирования:

- *изучение объекта* (научные исследования) — максимально полное и точное отражение свойств;
- *управление* – фиксация свойств в рабочем диапазоне изменений параметров;

➢ *прогнозирование* – построение модели, способной предсказывать поведение объекта в будущем;

➢ *обучение* – отражение в модели ключевых свойств, значимых для освоения.

Схема построения *познавательной модели*:

1. *Наблюдение* – эмпирическое изучение объекта, сбор данных, выделение характеристик и их значений.

2. *Кодирование информации* – перевод наблюдений в удобную для обработки форму. С использованием слов, либо символов, в частности математических, либо графических образов или в форме физических предметов, процессов или явлений.

3. *Фиксация* – сохранение закодированных данных в виде модели.

Из-за объективных (ограниченная измеримость, проницаемость среды) и субъективных (целевая и психологическая избирательность) факторов модель всегда проще оригинала и может не включать некоторые важные свойства.

Нормативное моделирование

Нормативное моделирование (прескриптивное, прагматическое) – процесс, при котором указываются цели деятельности и определенный порядок (алгоритм) действий для их достижения. Модель играет роль стандарта или образца, под который «подгоняются» как сама деятельность, так и её результаты.

В отличие от описательных («дескриптивных») моделей, фиксирующих реальность, нормативные модели задают образ будущего – то, как должно быть. В системном анализе их называют моделями «Как должно быть» (AS TO BE).

Цель – образ желаемого будущего, т.е. модель состояния, на реализацию которой направлена деятельность (моделирование).

Алгоритм – модель деятельности (набор действий) для достижения цели.

Основные черты *нормативного моделирования.*

– *Целеполагание*. Модель формулирует желаемый результат (цель) и путь его достижения.

– *Предписывающий характер*. Модель не просто отражает, а предписывает порядок действий, правила, стандарты.

– *Ориентация на реализацию*. При построении модели учитывают возможности исполнителей и условия внедрения.

– *Идеализация*. Модель – упрощённый образ желаемого состояния; реальность должна быть приближена к ней в процессе исполнения.

Схема построения *нормативной модели*:

1. *Определение цели* – формулировка желаемого состояния или результата.

2. *Анализ условий реализации* – учёт возможностей исполнителей, ресурсов, внешней среды.

3. *Разработка алгоритма* – построение последовательности действий, правил, стандартов.

4. *Формализация* – запись модели в удобной для применения форме (текст, схема, программа).

5. *Фиксация* – оформление модели как стандарта, регламента или проекта.

Формы *нормативных (прагматических) моделей:* планы, проекты, технические задания, программы, уставы, законы, административные регламенты, алгоритмы, рабочие чертежи и т.д.

При обнаружении расхождений прагматической модели и реальности изменяют не модель, а реальность, приводя ее в соответствие модели.

Классификации моделей

Познавательные и прагматические модели можно классифицировать по характеру выполняемых функций, по форме, зависимости объекта моделирования от времени.

По *функциональному назначению* можно выделить функции, выполняемые моделью:

− исследовательская – применяется в научном познании;

− практическая – применяется в практической деятельности (проектировании, управлении и т.д.);

− тренинговая – используется для тренировки практических умений и навыков специалистов в различных областях;

− обучения – для формирования у обучаемых знаний, умений и навыков.

По *форме* модели:

− физические – материальные объекты, имеющие сходство с оригиналом (модель автомашины, модель дома и т.д.);

− словесные (вербальные) – используют словесное описание (описание поведения самолета на разных высотах, принципа работа устройства, структуры предприятия);

− графические – используют описание в виде графических изображений (карт, схем, диаграмм, графиков и т.д.);

− знаковые – используют описания в виде символов и знаков (условные обозначения, дорожные знаки, математические соотношения и т.д.).

Одной из разновидностей знаковых моделей является математическая модель.

Математическая модель

Математическая модель – описание какого-либо объекта, явления или процесса с использованием математических понятий, символов, формул и уравнений. Она выражает основные свойства и связи объекта или процесса на языке математики.

Формально пусть дана реальная система S с набором характеристик $X = \{x_1, x_2, \ldots, x_n\}$ и взаимосвязями R. Тогда *математическая модель* M этой системы – математический объект (уравнения, функции, графы, матрицы и т.п.), такой что:

− M отражает существенные свойства S;

− *M* позволяет исследовать поведение *S* без прямого эксперимента на реальной системе;

− *M* допускает вычисления и прогнозирование состояний *S* при разных входных данных.

Этапы построения математической модели:

1. Содержательное описание системы:

− выделение ключевых элементов, подсистем, связей;

− определение целей моделирования и границ системы;

− фиксация известных параметров, констант, диапазонов изменений.

2. Формализация:

− введение символов и переменных для объектов и процессов;

− выбор математического аппарата (дифференциальные уравнения, графы, вероятностные распределения и т.п.);

− запись соотношений, законов, ограничений в виде формул.

3. Построение модели:

− составление системы уравнений, неравенств, логических правил;

− учёт внешних воздействий и обратной связи;

− проверка согласованности и полноты описания.

4. Верификация и валидация:

− проверка внутренней логической корректности модели (верификация);

− сопоставление результатов модели с реальными данными (валидация);

− оценка погрешности и области применимости.

5. Исследование модели

− аналитическое решение (если возможно);

− численное моделирование (методы Рунге–Кутты, Монте-Карло и др.);

− имитационное моделирование, сценарный анализ.

6. Интерпретация и применение:

− перевод математических результатов на язык предметной области;

− принятие решений, оптимизация, прогнозирование.

Типы математических моделей, используемых в системном анализе:

− *Детерминированные* – все параметры и связи заданы точно (например, системы дифференциальных уравнений).

− *Стохастические* – учитывают случайность (вероятностные распределения, марковские цепи).

− *Дискретные* – переменные изменяются скачками (конечные автоматы, сетевые модели).

− *Непрерывные* – переменные меняются плавно (дифференциальные уравнения).

− *Статические* – описывают состояние системы в фиксированный момент.

− *Динамические* – отражают эволюцию системы во времени.

− *Линейные* – связи между переменными линейны.

– *Нелинейные* – содержат нелинейные зависимости (чаще отражают реальную сложность).

Пример обобщенной математической модели.

Пусть система описывается вектором состояний $x(t) \in \mathbb{R}^n$, управлением $u(t) \in \mathbb{R}^m$, внешними возмущениями $d(t) \in \mathbb{R}^k$. Тогда модель может иметь вид:

$$\begin{cases} x'(t) = f(x(t), u(t), d(t), t), \\ u(t) = g(x(t), t), \\ x(t_0) = x_0. \end{cases}$$

где $x'(t)$ – производная по времени (динамика); f – вектор-функция переходов; g – функция выхода (наблюдаемые переменные), x_0 – начальное состояние.

С точки зрения изменчивости объекта во времени модели можно разделить на статические и динамические.

Статическая модель отражает состояние объекта в отдельный момент времени. Ее строят в тех случаях, когда объект на некотором временном промежутке существенно не меняется. К статическим относят, например, модели различных структур.

Динамическая модель отражает процесс изменений состояния объекта, например, модели различных бизнес-процессов.

Модели также можно разделить на *идеальные* (абстрактные) и *реальные* (материальные). Идеальные модели существуют в сознании человека. В силу малой изученности мыслительных процессов современная наука обычно имеет дело в основном с выражением таких моделей средствами естественного, профессионального либо формализованного языка. Яркий пример идеальных моделей – математические модели процессов и явлений, позволяющие прогнозировать поведение реальных объектов с помощью идеального представления.

Простейшей абстрактной моделью является *классификация* – разделение множества объектов на подмножества по их сходству и различию. Классификация дает возможность установить свойства объектов и связи между ними по их принадлежности к той или иной классификационной группировки.

Примеры классификаций – таблица Д. И. Менделеева, система частей речи в русском языке, учебный план образовательной программы, штатное расписание организации.

Существует два основных принципа классификации – иерархический и фасетный.

Процесс построения *иерархической классификации* можно представить следующим образом.

1. Выделить наиболее существенное основание (признак) и зафиксировать сходства и различия между объектами в соответствии с рассматриваемым основанием.

2. Объединить слабо различающиеся между собой по данному основанию объекты в группы.

3. Внутри каждой группы повторять процедуру, пока различие между объектами остается существенным.

Как правило, в соответствии с иерархическим принципом классификации кодируются номера студенческих групп – первая группа знаков обычно соответствует факультету, вторая – году поступления, третья – направлению или специальности обучения. В случае необходимости могут быть добавлены дополнительные уровни классификации, соответствующие делению множества объектов по иным основаниям.

Иерархические классификаторы просты, интуитивно понятны и могут быть использованы в тех случаях, когда признаки объектов (основания группировок) можно ранжировать по важности, если признаки объектов достаточно устойчивы и классификация создается на достаточно долгий период времени.

Фасетный принцип классификации не предполагает создания статичных группировок. Данный принцип эффективен, если признаки объектов нельзя ранжировать или если не для всех объектов актуально разделение по тому или иному основанию, а также для нестабильных ситуаций, когда значимость признаков может меняться.

Основания классификации (свойства объектов) фиксируются в виде *фасетной формулы*. Каждая из частей этой формулы – *фасет*, который является основанием для деления множества объектов на подмножества. Например, для товаров в супермаркете может быть составлена следующая фасетная формула: Наименование товара – Производитель – Размер – Материал – Цвет – Единица измерения – Срок годности. В соответствии с этой формулой товар получит маркировку: Кроссовки – Китай – 44 – Искусственная кожа – Белый – Штука – *, либо Мандарины – Марокко – * – * – * – * – Килограмм – 20.05.2026 (звездочкой обозначено отсутствие соответствующего признака у объекта).

Реальные модели представляют собой материальный объект, способный заместить оригинал в контексте поставленной цели. В зависимости от способа, которым было установлено подобие модели оригиналу, можно выделить три типа моделей.

Прямое подобие устанавливается в результате физического взаимодействия (или цепочки взаимодействий). Модели прямого подобия – фотографии, макеты зданий, модели самолетов для аэродинамических испытаний, выкройки швейных изделий, актеры в сериалах и т.п.

Косвенное подобие (*аналогия*) объективно существует в природе и может быть установлено, если абстрактные модели таких объектов достаточно близки. Примеры реальных моделей с косвенным подобием – часы как аналог времени, следственный эксперимент в криминалистике, подопытные животные в медицине, цифровые двойники в производстве.

Условное подобие устанавливается в результате соглашения, например, деньги как модель стоимости, сигналы светофора как модель проезда через перекресток, чертеж как модель продукции, карта как модель местности.

Рассмотрим свойства моделей, позволяющие использовать их для замещения исходных объектов.

Основным свойством модели является ее упрощенность относительно моделируемого объекта. Именно упрощенность делает целесообразным использование модели вместо оригинала. Из упрощенности следует другое свойство модели – конечность.

Реальный мир бесконечен – он продолжается во времени, в пространстве, и каждый его объект находится в бесконечном числе отношений с другими объектами. В силу сложности работы с бесконечными объектами модель всегда конечна.

Абстрактные модели конечны в силу того, что их создают так, чтобы эти модели имели ограниченное количество свойств и вступали в ограниченное количество отношений с другими объектами. Материальные объекты также можно считать конечными, поскольку из неограниченного количества свойств, которыми они обладают, используют только ограниченное их количество.

Упрощенность и *конечность* модели наделяют ее свойством приближенности, отражающим существование различия между моделью и моделируемым объектом. Степень приемлемости различия зависит от цели моделирования, и одно и то же различие между моделью и оригиналом может быть как вполне допустимым, так и делающим модель непригодной для использования.

Вследствие различий между оригиналом и моделью в модели можно условно выделить совпадающую с оригиналом (истинную) часть и собственную, заведомо отличающуюся от оригинала (ложную) часть.

Модель, которую можно успешно применить для достижения цели моделирования, называется *адекватной* этой цели. Для познавательных моделей адекватность тождественна истинности, для прагматических возможны модели, не являющиеся истинными, но, тем не менее, успешно выполняющие свои функции. Если удается ввести меры истинности и адекватности, особенно измеряемые в количественных шкалах, то модель можно пошагово совершенствовать в заданном направлении.

Еще одним важным свойством модели является *ингерентность* – согласованность модели с культурной средой, в которой она функционирует. В случае низкой *ингерентности* модели она не может быть использована по назначению.

Таким образом, можно определить *модель* как отображение оригинала: целевое, абстрактное либо реальное, упрощенное, приближенное, имеющее как истинное, так и ложное содержание, адекватное цели и *ингерентное* культуре пользователя.

Виды моделирования

В настоящее время моделирование применяется в очень широком спектре человеческой деятельности – от научных дисциплин до практических технологий. Поэтому используемые типы и способы моделирования очень разнообразны и зависят от выбранной области. В практикуме ограничимся физико-математическими и естественными науками, где различают следующие типы моделирования:

1. *Концептуальное моделирование* – метод, в котором с использованием специальных знаков, символов, операций над ними с помощью естественного или искусственных языков истолковывается основная мысль (концепция) об исследуемом объекте.

2. *Интуитивное моделирование* – метод, основанный на неформализованном, неосознанно-образном представлении об объекте исследования. Такое моделирование опирается на интуицию, жизненный опыт и «чувство ситуации», а не на строгую логическую или математическую формализацию. Сводится к мысленному эксперименту на основе практического опыта работников (экспертные решения, творчество и дизайн, применяется в экономике, при построении гипотез).

3. *Физическое моделирование* – метод экспериментального изучения физических объектов и явлений, при котором исследования проводятся не на реальном объекте, а на его физической модели, имеющей ту же природу, что и оригинал. Моделирование в естественных науках (химии, биологии, минералогии и т.п.) проводится по схожим схемам, как и при физическом моделировании с использованием математических и других моделей.

4. *Структурно-функциональное моделирование* – метод, при котором объект исследуется одновременно с двух позиций: структурной (из каких элементов состоит, как они связаны) и функциональной (какие действия/процессы выполняют элементы, как система достигает цели). Моделями в данном случае являются схемы (блок-схемы), графики, чертежи, диаграмм, таблицы, рисунки, дополненные специальными правилами их объединения и преобразования.

5. *Математическое моделирование* – метод научного исследования, при котором реальный объект, явление или процесс описывается и изучается с помощью математического аппарата: формул, уравнений, функций, неравенств, статистических зависимостей и иных формальных конструкций. Вместо прямого эксперимента с реальным объектом строится его математическая модель – упрощённое, но строгое описание ключевых свойств и взаимосвязей на языке математики. Затем модель исследуют аналитически или численно, чтобы понять закономерности поведения систем, спрогнозировать ее состояние, оптимизировать параметры и принять обоснованное решение по ее управлению.

6. *Иммитационне моделирование* – метод исследования сложных систем и процессов, при котором реальный объект или процесс заменяется *компьютерной моделью*, воспроизводящей его поведение во времени с учётом ключевых взаимосвязей и случайных факторов. С этой моделью проводят виртуальные эксперименты, чтобы изучить свойства оригинала, спрогнозировать его поведение и принять решения без вмешательства в реальную систему.

Все вышеперечисленные виды моделирования не являются взаимоисключающими и могут применяться как отдельно, так и в различных комбинациях исходя из целей моделирования.

Компьютерное моделирование

Компьютерное моделирование – метод исследования объектов, систем и процессов путём создания и изучения их компьютерных моделей: формализованных (численных, логических, графических) представлений, реализованных в виде программ и вычислительных экспериментов.

При *компьютерном моделировании* вместо экспериментов с реальным объектом строится абстрактная модель – упрощённое, но математически или алгоритмически точное описание ключевых свойств и взаимосвязей. Модель реализуется на компьютере как программный код, работающий с данными и уравнениями. Проводятся вычислительные эксперименты, где модель «проигрывает» поведение системы при разных входных данных и условиях. Результаты анализируются, интерпретируются и переносятся на реальный объект.

Основные типы компьютерных моделей:

– *Численные (математические).* Решают уравнения (дифференциальные, алгебраические и др.) численными методами.

Пример: расчёт напряжений в конструкции методом конечных элементов.

– *Имитационные*. Воспроизводят логику и последовательность событий в системе (см. имитационное моделирование).

Пример: моделирование очередей в аэропорту.

– *Агентные*. Описывают поведение множества автономных «агентов» с локальными правилами; глобальное поведение возникает из взаимодействий.

Пример: моделирование распространения эпидемии.

– *Графические (2D/3D)*. Визуализируют формы, движения, сцены.

Пример: архитектурные визуализации, анимация персонажей.

– *Симуляционные.* Имитируют управление реальным устройством (тренажёры, виртуальные лаборатории).

Пример: авиационный тренажёр.

– *Сетевые и графовые*. Описывают связи и потоки в сетях (транспортных, социальных, информационных).

Пример: анализ устойчивости энергосистемы.

Компьютерное моделирование – универсальный инструмент для исследования, прогнозирования и оптимизации сложных систем. Оно объединяет математику, физику и другие науки, а также алгоритмику и визуализацию, позволяя получать новые знания и принимать обоснованные решения без прямого вмешательства в реальный объект.

Задания к работе

1. Информационное моделирование. Представить схемы: информационной модели, структурно-функциональной модели, потоков данных и др. (по согласованию с научным руководителем). Используйте UML.

2. Физическое моделирование. Представить схемы: физических моделей (конструкций), сетей и т.п. по согласованию с научным руководителем. Эти схемы приводятся опционально. Например, если тема связана с разработкой информационной системы для поддержки физического эксперимента, то необходимо представить физическую модель.

3. Описать или изобразить структурную модель средней школы, в которой вы учились. Какие средства были использованы для описания модели? Описать или изобра-

зить модель средней школы с точки зрения другого субъекта (например, учителя, родителей, повара школьной столовой). Что общего между этими моделями и в чем их различие? В чем отличие созданной модели от реальности? Что можно изменить, чтобы сблизить модель и реальность?

4. Построить познавательную модель университета, в котором вы учитесь. Какие средства были использованы для описания модели? Построить прагматическую модель университета.

5. Обсудить построенные модели и сформулировать общий результат. Проанализировать различия между познавательной и прагматической моделями университета.

Методика выполнения работы

Перед выполнением задания необходимо изучить следующие стандарты: ГОСТ Р 57700.22-2020, ГОСТ Р 57412-2017, ГОСТ Р 50.1.028-2001. Установить программное обеспечение RAMUS на свой компьютер. Ознакомиться с методологией функционального моделирования. В малых группах (2–3 человека) обсудить выбранную модель. Подготовить описание модели. Провести выбранный тип моделирования.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет по выбранной модели. Отчет по работе должен содержать графическое описание модели (диаграммы, графики, схемы и т.д.) и пояснительный текст, описывающий основные этапы моделирования, а также описание полученных результатов. Сформировать текстовый документ в электронной форме (.docx) и представить преподавателю.

Вопросы для самоконтроля

1. С какой целью и при каких обстоятельствах может потребоваться моделирование? Как моделирование связано с решением проблем?

2. Описать основные виды моделей. Для каждого вида привести примеры ситуаций, в которых создание модели описанного вида приводит к наилучшему результату.

3. Охарактеризовать классификацию как простейшую абстрактную модель. Какие методы классификаций вы знаете?

4. Перечислить основные виды моделирования в естественных науках. В чем специфика математического моделирования и его связь с системным анализом?

5. Привести примеры реальных моделей, с которыми вы ежедневно имеете дело. Какие виды подобия использованы в этих моделях?

6. Перечислить свойства моделей. Рассмотреть какую-либо модель и проанализировать, обладает ли она перечисленными свойствами. Удалось ли во всех случаях использовать количественные шкалы или некоторые свойства данной модели невозможно измерить количественно?

Рекомендуемая литература

1. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

2. Качала, В. В. Основы теории систем и системного анализа / В. В. Качала. Москва : Изд-во Горячая линия – Телеком, 2012. 210 с.

3. Громова, Е. Методы классификации данных: фасетный и иерархический / Е. Громова [Электронный ресурс]. URL: https://sky.pro/wiki/profession/metody-klassifikacii-dannyh-fasetnyj-i-ierarhicheskij (дата обращения: 01.04.2026).

Лабораторная работа № 4
Система, ее свойства и анализ систем

Тема занятия: Анализ состояния системы и среды.

Цель занятия: Определения системы, основные свойства и анализ систем.

Материалы и программное обеспечение:

1. ГОСТ Р 57193-2016. Системная и программная инженерия. Процессы жизненного цикла систем.

2. ГОСТ Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.

3. Программное обеспечение RAMUS.

4. Программное обеспечение для моделирования: MS Visio, Draw.io.

5. Текстовый редактор: MS Word / Writer LibreOffice, Google Docs.

Краткие теоретические сведения

Под *системой* понимают группу взаимосвязанных *элементов*, действующих совместно с целью выполнения заранее поставленной задачи.

В системном анализе *система* определяется как целостный комплекс взаимосвязанных элементов, объединённых общностью цели и образующих особое единство с окружающей средой.

В настоящее время существует большое количество (более 40) определений понятия *система*. Приведем далее несколько дескриптивных определений системы.

Система есть совокупность взаимосвязанных элементов, обособленная от среды и взаимодействующая с ней как целое (Ф. П. Тарасенко).

Система – множество объектов вместе с соотношениями между объектами и их атрибутами (А. Холл, Р. Фейджин).

Система – организованное или сложное целое, сочетание частей, образующих единое целое (Ф. Каст).

Основные свойства системы.

1. *Целостность*. Система – не механическая сумма частей, а единое образование, обладающее общими свойствами и поведением. Её свойства (*эмерджентные*) не сводятся к сумме свойств отдельных элементов.

2. *Структурность*. Элементы системы связаны устойчивыми отношениями, формирующими упорядоченную организацию.

3. *Целеполагание*. Система существует для достижения определённой цели (или набора целей). Цель задаёт критерии эффективности и определяет границы системы.

4. *Взаимодействие со средой*. Система обменивается с окружением:

– *входами* (ресурсы, данные, сигналы, поступающие в систему);

– *выходами* (результаты, продукты, сигналы, выдаваемые системой).

5. *Иерархичность*. Система может состоять из подсистем, каждая из которых также является системой со своей целью и структурой.

6. *Динамичность*. Система способна:

– *функционировать* (стабильно реализовывать цель);

– *развиваться* (менять структуру или цель при необходимости).

Границы системы условны и определяются задачей исследования.

Окружающая среда – совокупность всех внешних по отношению к рассматриваемой системе объектов, условий и факторов, которые влияют на её функционирование и с которыми система взаимодействует.

Система формируется (создается) наблюдателем посредством выбора цели и элементов, поэтому можно говорить об ее субъективном характере.

Существует несколько различных моделей системы, с помощью которых можно проводить исследование свойств систем: модель «черного ящика», «серого ящика», «белого ящика», структурная модель, модель состава, функциональная модель и др.

При выделении *системы* из окружающей среды ключевым критерием является определение *цели*.

Для выделения системы необходимы:

– объект исследования;

– цель, для реализации которой формируется система;

– субъект наблюдения (наблюдатель), формирующий систему;

– входные и выходные переменные, отражающие взаимосвязь системы с окружающей средой.

Таким образом, можно говорить о том, что границы системы являются условными и определяются конкретной задачей исследования. Для одного объекта, в зависимости от цели исследования, может быть сформировано несколько систем различного рода (математическая модель, структурная модель, экономическая система (модель), физическая система (модель) и пр.).

Свойство – сторона объекта, обусловливающая его различие или сходство с другими объектами, проявляющиеся в взаимосвязи с ним.

Всякое *свойство* относительно. Каждый объект обладает бесконечным количеством *свойств*, в совокупности которых определяют его *качество*. Субъект (исследователь) всегда работает только с ограниченным набором свойств, которые могут изменяться в зависимости от целей исследования. Свойства также дают возможность описывать объекты системы количественно, выражая их в единицах, имеющих определенную размерность.

Свойство объекта, от которого зависят все его другие свойства, называется *сущностью*. Форма обнаружения (выражения) *сущности*, отражающая внешние свойства и отношения предмета, называется *явлением.*

Меру количественного описания свойства называют *параметром*.

Когда говорят о входных и выходных параметрах системы, имеют в виду значения ее входных и выходных переменных, а говоря о внутренних параметрах системы – значения ее внутренних свойств.

Интегративные свойства – свойства, которые имеются у системы в целом, но отсутствуют у ее элементов.

Элементы и связи системы

Элементы системы – объекты системы, выполняющие определенные функции, которые рассматриваются как неделимые, в рамках поставленной цели (задачи). Примеры элементов: атомы, электроны (атомные системы), уравнения (система уравнений), правила (система правил) и т.д.

Систему можно описать в виде множества ее элементов:

$$A = \{a_1, a_2 \ldots a_n\}, \tag{4.1}$$

где a_i – *i*-й элемент системы *A*.

В *иерархических структурах* на разных уровнях *элемент* может рассматриваться как система. В связи с этим вводятся понятия *подсистемы* и *надсистемы*.

Система, являющаяся элементом данной системы, называется *подсистемой* данной системы. Подсистемами называются части системы, состоящие более чем из одного элемента.

Система, элементом которой является данная система, называется *надсистемой* данной системы. *Модель состава системы* описывает, из каких подсистем и элементов состоит система (рис. 4.1).

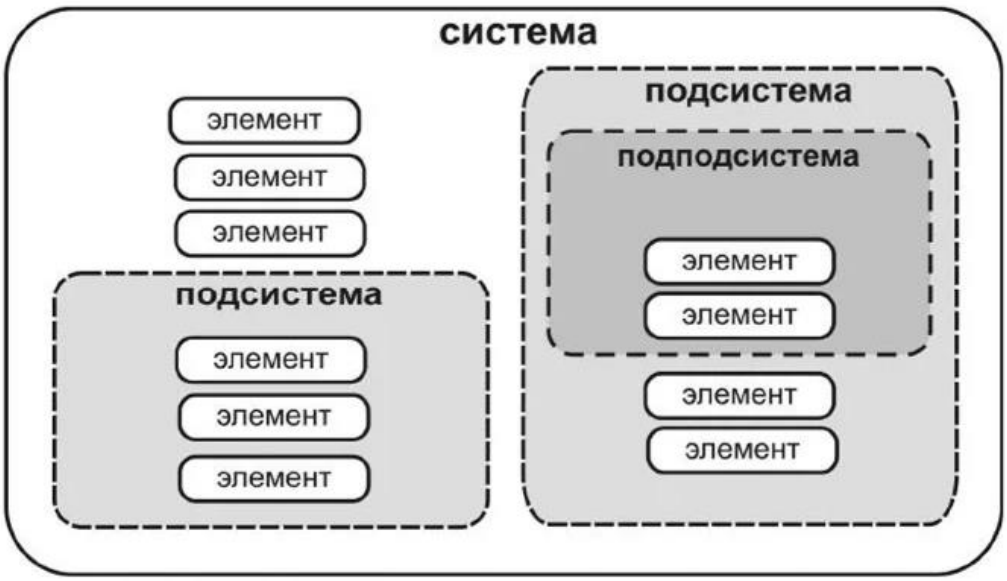


Рис. 4.1. Модель состава системы

Систему, подсистемы и надсистемы можно представить в иерархической форме, как это показано на рис. 4.2.

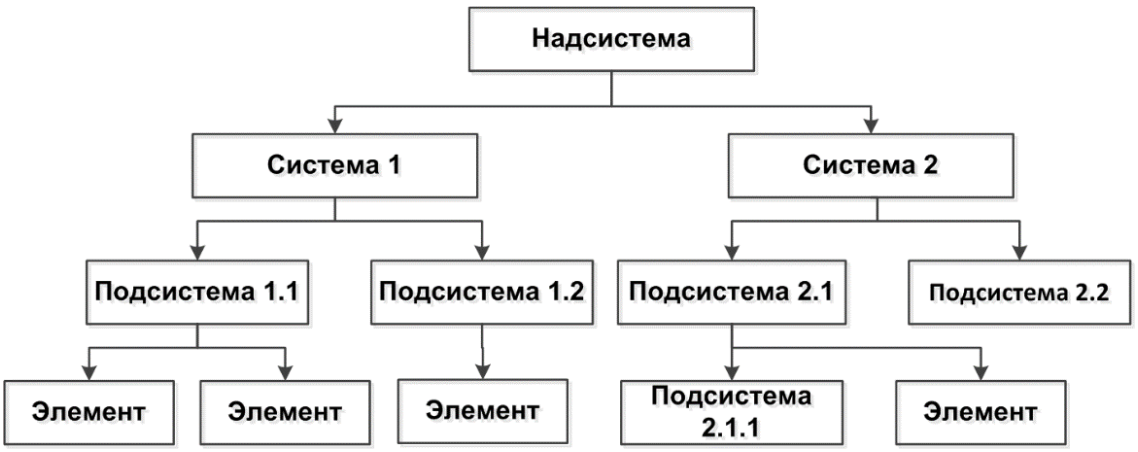


Рис. 4.2. Модель (иерархическая) состава системы

Состав системы в зависимости от знаний исследователя (наблюдателя) может быть различным. С одной точки зрения состав может иметь один набор элементов и подсистем, с другой точки зрения – совершенно иной. Это связано с субъективными факторами построения модели состава наблюдателем, разными уровнями рассмотрения и вследствие того, что всякое разделение системы на подсистемы является относительным (условным).

Совокупность всех элементов системы, из которых состоит система, образует ее состав. Состав системы сводится к полному перечню ее элементов. Системы, имеющие одинаковый состав, могут иметь различные свойства, поскольку определяющим для системы является организация (способ упорядочивания) ее элементов, или *структура*. Пример модели состава представлен на рис. 4.3.

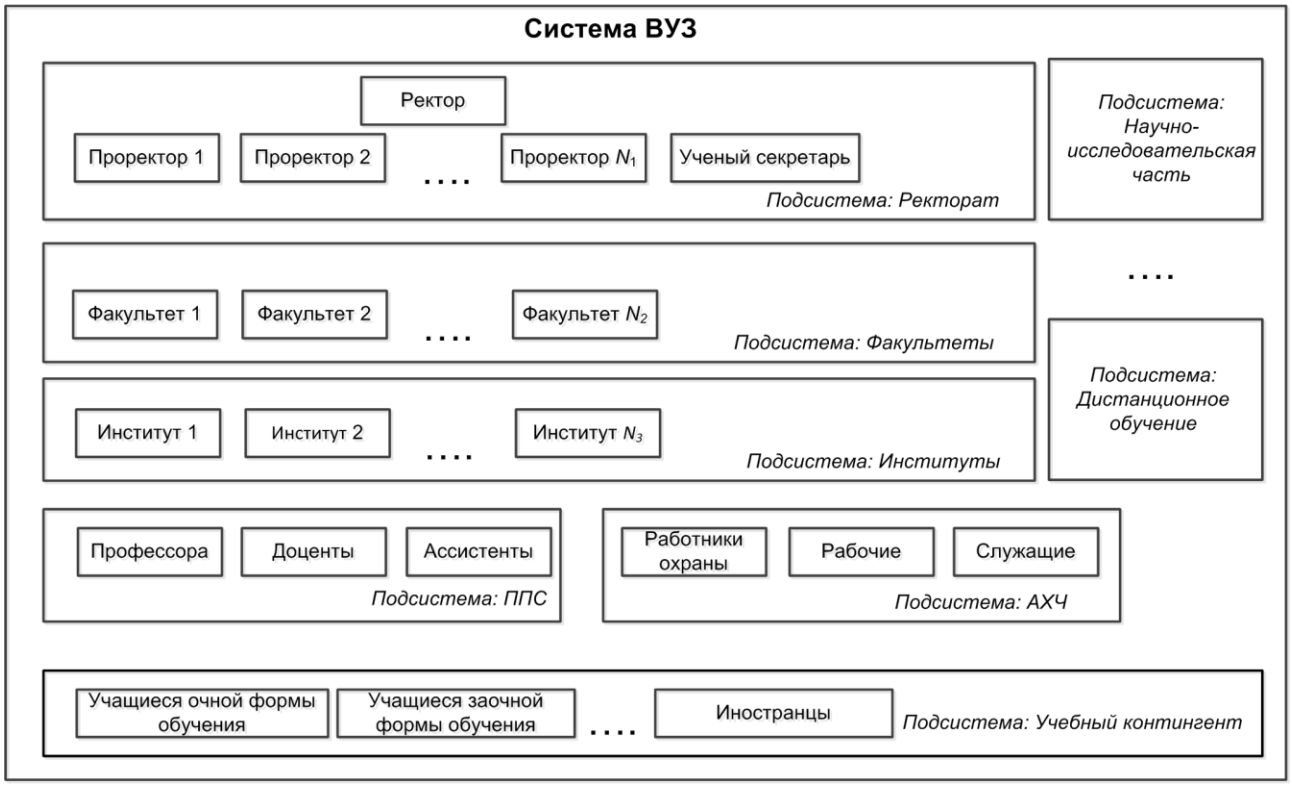


Рис. 4.3. Пример модели состава для системы «ВУЗ»

Связи – компоненты системы, которые осуществляют взаимодействие между ее элементами, а также между системой в целом и ее средой. Можно сказать, что связи объединяют систему в единое целое.

Множество R – связей между элементами системы $A = \{a_1, a_2 \ldots a_n\}$, можно представить в форме матрицы

$$R = \{r_{ij}\}, \tag{4.2}$$

где r_{ij} – связь между элементами a_i и a_j, $i = 1,\ldots, n$; $j = 1,\ldots, n$.

Связи можно подразделять на следующие группы:

– материально-вещественные (физические, химические) – процессы, связанные с передачей вещества (свойств) между элементами;

– энергетические (физические, физико-химические) – процессы, связанные с передачей энергии между элементами;

– информационные – представляют собой информационные потоки.

Существуют различные способы классификации связей: направленная и ненаправленная (по направлению), непрерывная, прерывистая, дискретная (по структуре), односторонняя и двусторонняя, равноправная и неравноправная (по способу влияния).

В системном анализе и информационных технологиях, а также в теории управления большое значение имеет классификация связей на прямую и обратную (рис. 4.4).

Рис. 4.4. Пример прямой (*а*) и обратной (*б*) связи между элементами

Прямая связь – непосредственно прямое воздействие одного элемента на другой (связь между выходом одного элемента и входом другого).

Обратная связь – воздействие результатов функционирования элемента на характер этого функционирования (связь между выходом и входом одного и того же элемента).

Различают *положительную* (усиливающую) *обратную связь* и *отрицательную* (уравновешивающую).

Обратная связь, уменьшающая влияние входного воздействия на выходную величину, называется *отрицательной*. Связь, увеличивающая это влияние, называется *положительной*.

Положительная (усиливающая) обратная связь (ПОС) усиливает тенденцию изменения выхода системы, а отрицательная (уравновешивающая), наоборот, ее уменьшает.

Отрицательная обратная связь (ООС) способствует восстановлению равновесия в системе после осуществления входных воздействий, а положительная, в свою очередь, усиливает изменения и отклонения от равновесного состояния.

Пример 1. Положительная обратная связь. Поведение вирусного контента в сети Интернет, банковский кризис (клиенты забирают деньги), эпидемия гриппа, ядерная реакция.

Пример 2. Отрицательная обратная связь. Термостат в помещении (удерживающий температуру), ПИД-регулятор в промышленности (стабилизирующий температуру в рабочей зоне), регуляция уровня глюкозы.

Обратная связь является основой *саморегуляции*, *самоорганизации* и *развития* системы, приспосабливая её тем самым к изменениям как внутренней, так и внешней среды. Можно сказать, что жизненный цикл живых существ состоит из циклов обратной связи.

Структура системы (модель структуры системы)

Структура (от лат. *structure* – строение, расположение, порядок) отражает определенные взаимосвязи элементов системы, ее строение.

Структура системы – устойчивая упорядоченность в пространстве и во времени ее элементов и связей между ними. Представим структуру в виде множества *St*:

$$St = \{A, R\}, \tag{4.3}$$

где *A*– множество элементов; *R* – множество связей.

Совокупность необходимых и достаточных для достижения цели отношений между элементами называется *структурой системы* (Ф. П. Тарасенко).

Структурные связи обладают относительной независимостью от элементов и при переходе от одной системы к другой могут переносить закономерности, выявленные и отраженные в структуре одной из них, на другие. При этом системы могут уметь различную физическую природу.

Одна и та же система может быть представлена разными структурами в зависимости от имеющихся в данный момент знаний об объектах, уровнях рассмотрения и поставленных целей. В ходе дальнейших исследований знания о системе может изменяться, и поэтому структура также может претерпевать изменения.

Структура является важнейшей характеристикой системы, так как при одном и том же составе элементов, но при различном взаимодействии между ними меняется и назначение системы и ее возможности.

Основные типы структур

Понятие *структура* связывается с графическим отображением элементов и их связей. Структура может быть представлена не только в графической форме, но и в матричной, в теоретико-множественном описании, а также с помощью топологии, алгебры и других средств моделирования систем.

Рассмотрим основные типы структур.

1. По топологии (пространственной организации связей):

– *Линейная* – элементы выстроены в цепочку, каждый связан только с предыдущим и последующим.

– *Иерархическая (древовидная)* – вертикальная соподчинённость: есть главный элемент (корень), от него отходят уровни подчинённых элементов.

– *Сетевая* – произвольные связи между элементами без жёсткой иерархии; возможны множественные пути между узлами (звезда, многосвязная сеть, шина, кольцо).

– *Матричная* – перекрёстные связи по двум осям (например, функциональные и проектные подразделения в организациях).

2. По характеру связей:

– *С прямыми связями* – воздействие идёт от одного элемента к другому без обратной передачи.

– *С обратными связями* – есть каналы возврата информации/воздействия (положительные усиливают процесс, отрицательные стабилизируют).

– *Со смешанными связями* – сочетание прямых и обратных связей.

3. По устойчивости структуры:

– *Детерминированная* – связи жёстко заданы, поведение предсказуемо.

– *Вероятностная* – связи и поведение определяются с некоторой вероятностью.

– *Хаотическая (диссипативная)* – структура неустойчива, чувствительна к малым изменениям.

4. По степени централизации:

– *Централизованная* – есть доминирующий элемент, управляющий остальными.

– *Децентрализованная* – нет главного элемента, управление распределено.

5. По однородности элементов:

– *Гомогенная* – элементы схожи по свойствам, взаимозаменяемы.

– *Гетерогенная* – элементы разнородны, замена нарушает работу системы.

– *Смешанная* – сочетает однородные и разнородные элементы.

Каждая классификация подчёркивает разные аспекты организации системы. На практике часто используют комбинированные описания, чтобы учесть множественность связей и функций**.**

Стратификация и страты

В теории систем М. Месаровича страты – уровни абстрагирования при описании сложной системы. При анализе и описании сложной системы возникает проблема поиска баланса между простотой описания (чтобы сохранить целостное представление об объекте) и детальностью (чтобы отразить многочисленные особенности конкретного объекта). Один из путей решения этой проблемы – *стратификация*.

Решение: задать систему семейством моделей, каждая из которых описывает поведение системы с точки зрения соответствующего уровня абстрагирования. Эти уровни и называются *стратами,* а представление системы – *стратифицированным.*

Стратификация – метод разбиения сложной системы или массива данных на однородные подмножества (уровни или *страты*), каждый из которых описывается своими терминами, концепциями, принципами и типами данных.

Свойства стратифицированного описания системы:

1. *Иерархичность описания*. Система представляется как набор вложенных уровней абстракции. То, что на вышестоящей страте выступает как *элемент*, на нижележащей раскрывается как *подсистема*.

2. *Автономность страт.* На каждом уровне действуют свои законы и модели; принципы одной страты обычно не выводятся из принципов другой.

3. *Асимметричная зависимость.* Требования и ограничения вышестоящей страты становятся обязательными условиями для нижележащих уровней.

4. *Постепенная детализация.* При движении «вниз» по иерархии описание становится всё более конкретным; при движении «вверх» выявляется общий смысл и целевое назначение системы.

Стратификация включает две основные задачи:

1. *Выделение подсистем* – разбиение исходной системы на однородные части по выбранному критерию (например, по функциональной обработке данных).

2. *Установление иерархических связей* – определение порядка и направленности взаимодействий между подсистемами (нисходящие и восходящие информационные потоки).

Выбор *страт* зависит от цели исследования – для одной и той же системы можно построить разные *стратифицированные* модели. Не все *стратифицированные* системы строго иерархичны – иногда слои могут быть относительно независимыми или допускать перекрёстные связи. Полная стратификация достигается, когда отклик каждой страты на внешние воздействия не выходит за её границы (т.е. не влияет на соседние страты помимо установленных интерфейсов).

Примерами простейшей двух уровневой *стратификации* (две *страты*) являются *макроскопический* и *микроскопический* анализы.

Макроскопический анализ заключается в игнорировании деталей структуры системы и наблюдении только общего поведения системы как целого. Цель *макроскопического* анализа заключается в создании модели системы и описании ее взаимодействия с окружением (модель «черного ящика»). В состав модели входят: тип структуры, границы системы, характер взаимодействия вход-выход, особенности функционирования и т.д.

Микроскопический анализ детально описывает каждый из компонентов системы, где центральным понятием является *элемент*, изучаются связи и функции элементов, структура системы и т.д. Задачи *микроанализа* следующие: выделение элементов в системе, изучение каждого из элементов, установление структуры системы, выявление связей между элементами. Однако для описания сложных систем двух *страт* недостаточно.

Примеры трехуровневого анализа систем (три *страты*).

– ЭВМ / компьютерная система:

1) нижняя *страта*: физические процессы (электромагнитные импульсы, тепловыделение);

2) средняя *страта*: логические и арифметические операции (процессор, память);

3) верхняя *страта*: программное обеспечение и пользовательский интерфейс.

– Предприятие:

1) технологическая страта (оборудование, процессы);

2) управленческая страта (планирование, контроль);

3) экономическая страта (финансы, прибыль).

– Информационная система (трёхслойная архитектура):

1) *страта* 1: слой данных (массив / база данных);

2) *страта* 2: слой бизнес-логики (правила обработки);

3) *страта* 3: слой представления (пользовательский интерфейс).

Преимущества *стратификации*:

– *Снижение сложности*. Каждый уровень описывается относительно простыми моделями.

– *Модульность*. Изменения в одной страте минимально затрагивают другие.

– *Ясность интерфейсов*. Чёткие правила взаимодействия между стратами.

– *Масштабируемость.* Возможность добавлять новые страты или детализировать существующие.

– *Удобство анализа и проектирования.* Разделение труда между специалистами разных уровней.

Принципы *стратификации* реализованы в методологиях структурного анализа SADT, DFD и серии стандартов IDEF.

Оценка состояния системы

Для системы $A=\{a_1, a_2 \ldots a_n\}$ можно рассмотреть элемент a_i, который характеризуется конкретными *свойствами* $\{s_{i1}, s_{i2} \ldots s_{ik}\}$, определяющими его в данный момент времени однозначно. Совокупность всех свойств *i*-го элемента определяет *состояние i*-го элемента s_i:

$$s_i = \{s_{i1}, s_{i2} \ldots s_{ik}\}. \quad (4.4)$$

Состояние системы

Значения выходов системы зависят от следующих факторов:

– значений (состояний) входных переменных;

– начального состояния системы;

– функции системы.

Общее *состояние системы* определяется состоянием *элементов* и состоянием (наличием) связей *R*:

$$S = \{s_1, s_2 \ldots s_n\} \cup R. \quad (4.5)$$

Наиболее важной *задачей системного анализа* является установление причинно-следственных связей выходов системы с ее входами и внутренним состоянием.

Состояние системы характеризует мгновенную «фотографию» системы, ее временной «срез».

Состояние системы в определенный момент времени – множество существенных свойств в данный момент времени. В таком случае говорят о состоянии входов, выходов и внутреннем состоянии системы.

Состояние входов системы описывается вектором значений входных параметров: $X = (x_1, ..., x_n)$, которые характеризуют (отражают) состояние и влияние *окружающей среды*.

Внутреннее состояние системы описывается вектором значений ее внутренних состояний (параметров состояния) $S = (s_1, ..., s_k)$ и зависит от вектора *X* и начального состояния S_0:

$$S = f(X, S_0). \tag{4.6}$$

Внутреннее состояние среды очень сложно наблюдать или практически невозможно, поэтому его оценивают по состоянию выходов (значений выходных переменных) системы $Y = (y_1, ..., y_m)$ благодаря зависимости

$$Y = \phi(S). \tag{4.7}$$

Замечание. Выходные переменные, как правило, не полностью и неоднозначно отражают состояния системы. Поэтому для формирования более точных моделей систем требуются дополнительные исследования.

Процесс

О системе, которая может переходить из одного состояния в другое ($S_1 \to S_2 \to ... \to S_N$) с течением времени *t*, говорят, что она обладает *поведением* и ее изменения можно описать таким понятием, как *процесс*.

Процесс – последовательная смена состояний системы.

Для случая, если процесс *P* непрерывно осуществляет смену состояний, его можно описать непрерывной функцией от времени:

$$P = V(t). \tag{4.8}$$

В дискретном случае – множеством:

$$P = \{V(t_1), V(t_2), ...\}. \tag{4.9}$$

По отношению к системе можно рассматривать два вида процессов:

1) *внешний процесс* – последовательная смена воздействий (смена состояний окружающей среды) на систему;

2) *внутренний процесс* – последовательная смена состояний системы, которая наблюдается как процесс на выходе системы.

Статические и динамические системы

Статическая система – это система, которая рассматривается в фиксированный момент времени; её состояние не меняется (или изменения пренебрежимо малы в рамках задачи).

Основные характеристики.

1. Состояние неизменно на интервале наблюдения.

2. Нет явной зависимости от времени: параметры системы не эволюционируют.

3. Описание системы через соотношения «вход → выход» в установившемся режиме.

Статическая система моделируются алгебраическими уравнениями (не дифференциальными).

Примеры статических систем:

1) электрическая цепь в режиме постоянного тока (токи и напряжения не меняются);

2) структурная схема организации (штатное расписание, иерархия подразделений);

3) карта связей в социальной сети на конкретную дату.

Динамическая система – система, которая изменяется во времени; её состояние зависит от предшествующих значений и внешних воздействий.

Основные характеристики.

1. Состояние эволюционирует под действием внутренних процессов и внешних факторов.

2. Есть явная зависимость от времени: параметры – функции времени $x(t), y(t)$.

3. Присутствуют переходные процессы, задержки, инерционность.

Динамическая система моделируется дифференциальными или разностными уравнениями, автоматами, имитационными моделями.

Типы динамики:

– *Функционирование* – стабильные процессы, реализующие фиксированную цель (часы, конвейер, сервер).

– *Развитие* – изменение целей и (или) структуры системы (реорганизация предприятия, эволюция вида).

Примеры:

1) движение планеты по орбите;

2) рост популяции с учётом рождаемости и смертности;

3) работа регулятора температуры (переходные процессы при изменении задания).

Функция системы

При анализе и проектировании систем важно знать не только значения входных/выходных переменных системы, но и по возможности функции системы.

Функция системы – способ (правило, алгоритм) преобразования входной информации в выходную. *Функция системы* описывает целенаправленное поведение, проявляющееся во взаимодействии со средой и ведущее к достижению поставленной цели. Иными словами, *функция* описывает, что система делает и какой результат обеспечивает в заданных условиях:

$$Y = F(X). \tag{4.10}$$

Функционирование системы

Функционирование системы – процесс реализации её функций, т.е. целенаправленной деятельности по переработке входных параметров в выходные результаты при взаимодействии со средой и с учётом обратных связей.

Иными словами, это *реальная работа системы во времени*: как она воспринимает входные воздействия, преобразует их внутри себя и выдаёт итоговый результат, одновременно реагируя на обратную связь и внешние условия.

Функционирование системы – процесс переработки входной информации в выходную.

Функционирование можно описать в виде формулы

$$Y(t) = F(X(t)), \tag{4.11}$$

где *F* – оператор (в частности некоторая формула) определяет алгоритм *функционирования*, который описывает, как меняется состояние системы при изменении состояния ее входов. *F* описывает совокупность математических и логических действий, которые нужно выполнить, чтобы по данным входам $X(t)$ найти выход $Y(t)$.

Условно можно считать, что оператор F состоит из структуры – STF и набора параметров – $M = (m_1, m_2, m_3, ...)$, тогда можно записать как

$$F = \{STF, M\}. \tag{4.12}$$

Можно говорить, что функция (оператор) отражает в определенной степени структуру системы и ее внутренние параметры.

Функция системы является ее свойством, поэтому можно говорить о состоянии системы в заданный момент времени, указывая ее функцию, которая справедлива в этот момент времени.

Состояние системы определяется набором значений ее параметров и функцией, которая, в свою очередь, зависит от структуры и значений параметров:

$$S_t = \{M_t, F_t\} = \{M_t, (STF_t, M_t)\},$$

где $M_t = M(t) = \{m_1(t), m_2(t), m_3(t), \ldots\}$.

Зная состояние функции системы, можно прогнозировать значение выходных параметров для стационарных систем.

Систему считают *стационарной*, если ее функция практически не изменяется в течение определенного периода времени.

Для такой системы реакция на одно и то же воздействие не зависит от момента приложения этого воздействия.

Если функция меняется по времени, то такие системы будут уже гораздо сложнее описываться, и они носят названия *нестационарных* систем.

Анализ систем

Анализ систем представляет собой попытку определить наиболее реальные способы выполнения поставленной задачи, обеспечивающие максимальное удовлетворение сформулированных требований.

Пример 1. Система прогнозирования успеваемости студентов (табл. 4.1).

Выделение системы из окружающей среды.

Таблица 4.1

Выделение системы из окружающей среды

Типы систем	Описание	Назначение
Целевая система	Прогнозирование успеваемости студентов	Мониторинг, анализ и предсказание академической успеваемости, построение траекторий
Надсистема	Цифровой профиль обучающегося	Блок прогноза успеваемости является частью платформы «Цифровой профиль студента»
Системы обеспечения	Django, серверное ПО, NLP-модели, базы данных, API	Инфраструктура и технологии, обеспечивающие функционирование целевой системы
Системы в окружении	Пользователи (студенты, преподаватели), микросервисы электронного портфолио	Внешние акторы и сервисы, взаимодействующие с системой
Подсистемы	Подсистема прогнозирования (нейросеть, модуль анализа данных, веб-интерфейс)	Функциональные блоки, обеспечивающие работу: модель, интерфейс, обработка данных

Целевая система представляет собой интеллектуальный модуль, предназначенный для анализа учебных данных студентов и предсказания их будущей академической успеваемости с целью формирования индивидуальных образовательных траекторий. Система является составной частью цифрового профиля обучающегося и служит инструментом поддержки принятия решений.

Назначение системы.

1. Автоматизированный анализ академической истории студентов.

2. Прогнозирование результатов обучения на основании данных предыдущих семестров.

3. Формирование персонализированных рекомендаций по обучению.

4. Выявление студентов из групп риска (имеющих низкий прогноз по успеваемости).

5. Повышение эффективности мониторинга и планирования образовательного процесса.

Основные функции системы.

– Интеграция с внешними источниками данных (электронное портфолио, цифровой профиль).

– Предобработка и очистка входных данных (оценки, посещаемость, дисциплины).

– Обработка и анализ данных с помощью нейросетевой модели.

– Визуализация результатов прогноза и траектории обучения.

– Обеспечение доступа к информации через веб-интерфейс.

Описание подсистем и элементов системы представлено в табл. 4.2.

Таблица 4.2

Описание системы

№	Система	Подсистема	Элементы
1	Прогнозирования успеваемости студентов	1.1. Нейронная сеть	Входной слой (входные параметры успеваемости студентов)
			Скрытые слои нейронной сети
			Функции активации и функция потерь
			Функции активации и функция потерь
		1.2. Модуль построения отчетов	Механизм формирования и визуализации отчётных данных
		1.3. Модуль формирования прогноза (анализа данных)	Алгоритм анализа входных данных и формирования прогноза
		1.4. Веб-интерфейс	Текстовые окна
			Кнопки
			Радиокнопки
			Выбор из списка
			Надписи
		1.5. База данных	Таблица

Стейкхолдеры:

1. Студенты – для самостоятельного анализа и планирования обучения.

2. Преподаватели и наставники – для контроля и корректировки учебных планов.

3. Администрация вуза – для принятия управленческих решений на основе аналитики.

Роль в надсистеме. Система функционирует как модуль внутри платформы «Цифровой профиль обучающегося» и взаимодействует с другими сервисами вуза, включая электронное расписание, ведомости, рейтинг и портфолио.

Вспомогательные системы. Описание вспомогательных систем представлено в табл. 4.3.

Таблица 4.3

Описание вспомогательных систем

№	Наименование вспомогательной системы	Описание
1	Django (веб-фреймворк)	Обеспечивает реализацию backend-части веб-приложения (управление логикой системы)
2	БД PostgreSQL	Хранение учебных данных студентов, результатов прогнозов, логов и конфигураций
3	Python + библиотеки машинного обучения TensorFlow (keras)	Поддержка разработки и выполнения нейросетевой модели
4	Сервер приложений / облачная платформа (Docker)	Обеспечение размещения и эксплуатации системы
5	Интерфейс цифрового профиля студента	Связь с пользователями через веб-интерфейс, отображение прогноза
6	API внешних микросервисов	Получение исходных учебных данных (оценки, расписание, дисциплины) из других систем вуза

Описание элементов системы представлено в табл. 4.4.

Таблица 4.4

Описание элементов системы

№	Наименование элемента	Назначение	Тип информационного процесса	Принадлежность подсистеме
1	Входные данные об успеваемости	Представление исходных данных о результатах обучения студентов	Ввод информации	База данных
2	Данные о студенте	Хранение характеристик студента (курс, специальность и др.)	Хранение информации	База данных
3	Обучающая выборка	Формирование набора данных для обучения нейронной сети	Подготовка информации	База данных
4	Модель нейронной сети	Реализация алгоритма прогнозирования успеваемости	Обработка информации	Нейронная сеть
5	Алгоритм формирования прогноза	Анализ входных данных и получение прогнозных значений	Обработка информации	Нейронная сеть
6	Программное обеспечение системы	Обеспечение функционирования нейронной сети и модулей системы	Обработка информации	Модуль формирования прогноза (анализа данных)
7	Прогноз успеваемости	Результат работы системы в виде прогнозных показателей	Вывод информации	База данных

Контекстная диаграмма системы предсказания успеваемости студентов представлена на рис. 4.5.

Рис. 4.5. Система предсказания успеваемости студентов

Описание основных связей между элементами системы представлено в табл. 4.5.

Таблица 4.5

Описание основных связей между элементами системы

№	Наименование связи	Подсистема	Характеристика передаваемых данных
1	Оценки студента	База данных	Числовые данные: оценки студентов за предыдущие периоды обучения
2	Данные о студенте	База данных	Категориальные и числовые характеристики студента (курс, специальность и др.)
3	Обучающая выборка	База данных	Размеченные данные для обучения и настройки параметров модели
4	Нейронная сеть	Нейронная сеть	Выходные значения нейронной сети (промежуточные и итоговые прогнозы)
5	Параметры модели	Нейронная сеть	Метаданные: параметры и настройки архитектуры нейронной сети
6	Методические указания	Модуль построения отчетов	Правила, требования и критерии формирования прогноза
7	Программное обеспечение системы	Модуль построения отчетов	Алгоритмы, библиотеки и программные модули для выполнения вычислений
8	Прогноз успеваемости	Веб-интерфейс	Результаты прогноза: показатели успеваемости студента

Задания к работе

Задание 1. Для выбранной системы из предложенных вариантов построить модель состава системы и заполнить табл. 4.6, 4.7, 4.8*. Постить контекстную диаграмму, определить входы и выходы. Указать элементы и подсистемы. Выявить связи между элементами, построить матрицу *R*. Определить целевой показатель*.

* Задание повышенной сложности, обычно выполняется группой обучающихся.

Таблица 4.6

Выделение системы из окружающей среды

Система	Описание	Назначение
Целевая система		
Надсистема		
Системы обеспечения		
Системы в окружении		
Подсистемы		

Таблица 4.7

Описание системы

№	Система	Подсистема	Элементы

Таблица 4.8*

Описание вспомогательных систем

№	Наименование вспомогательной системы	Описание

Затем для элементов составить расширенную табл. 4.9.

Таблица 4.9

Описание элементов системы

№	Наименование элемента	Назначение	Тип информацион-ного процесса	Принадлежность подсистеме

Задание 2. Для выбранной системы из предложенных вариантов построить структурную модель системы. Представить модель в графической форме с использованием программного обеспечения (RAMUS, MS Visio, Draw.io) и заполнить табл. 4.10.

Таблица 4.10

Описание основных связей между элементами системы

№	Наименование связи	Подсистема	Характеристика передаваемых данных

Задание 3. Описать основные свойства системы и заполнить табл. 4.11.

Таблица 4.11

Основные свойства системы

№	Наименование свойства	Характеристика

Задание 4. Указать цели и назначение системы и подсистем.

Необходимо сформулировать для системы и ее подсистем общую цель и составные задачи, которые раскрывают их назначение.

Задание 5. Описать функционирование системы.

Требуется описать основные способы применения системы и режимы работы.

Задание 6. Осуществить начальную формализацию параметров системы при системном исследовании. Определить по возможности векторы: X – входов системы, Y – выходов, S – вектор параметров внутреннего состояния.

Задание 7*. Составить список факторов, влияющих на целевой показатель. Необходимо выявить зависимости выходных переменных от входных и определить степень их влияния на целевой показатель.

Задание 8*. Привести формализованное описание системы. Формализованное описание системы можно представить в виде доступной формы (системы линейных уравнений, функций, операторов, графов, матриц и т.д.).

Методика выполнения работы

Перед выполнением задания необходимо изучить следующие стандарты: ГОСТ Р 57193-2016, Р 50.1.028-2001. В малых группах (2–3 человека) обсудить выбранную систему из представленных вариантов заданий или систему, согласованную с преподавателем. Подготовить описание системы. Провести графическое построение моделей систем с использованием RAMUS, MS Visio, Draw.io.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет по выбранной системе. Отчет по работе должен содержать графическое описание системы (диаграммы, графики, схемы и т.д.) и пояснительный текст, характеризующий основные этапы построения, а также изложение полученных результатов. Сформировать текстовый документ в электронной форме (.docx) и представить преподавателю.

Вопросы для самоконтроля

1. Дать определение понятию *система*. Привести примеры систем, использующихся в информационных технологиях.
2. Что необходимо сделать для выделения системы из окружающей среды?
3. Что означают свойства: устойчивость, развитие системы?

* Задание повышенной сложности, обычно выполняется группой обучающихся.

4. Что такое информационный подход к анализу систем?

5. Дать определения понятиям: *отношение*, *связь*, *структура системы*.

6. Пояснить следующие понятия: *поведение*, *состояние*, *событие*. Каким образом они отображаются в пространстве состояний?

7. Описать жизненный цикл системы. Какие государственные стандарты используются для описания жизненного цикла?

Варианты заданий (примеры систем)

1. Средняя общеобразовательная школа.
2. Расписания учебных занятий в школе.
3. Учебный проект (по выбору).
4. Пакет программ 1С.Бухгалтерия.
5. Электронное тестирование.
6. Пакет для математического моделирования MathLab.
7. Электронное портфолио обучающего.
8. Электронная ведомость.
9. Библиотека.
10. Электронный журнал успеваемости.
11. Образовательный веб-сайт средней школы.
12. Платформа для онлайн-образования.
13. Мониторинг экологии города.
14. Туристическое агентство.
15. Страховая компания.
16. Умный кампус.
17. Управления умным складом.
18. Аптека.
19. Компьютер.
20. Компьютерный магазин.
21. Строительная компания.
22. Транспортная компания.
23. Продажи билетов в аэропорту.

Рекомендуемая литература

1. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

2. Качала, В. В. Основы теории систем и системного анализа / В. В. Качала. Москва : Изд-во Горячая линия – Телеком, 2012. 210 с.

3. Силич, М. П. Теория систем и системный анализ / М. П. Силич, В. А. Силич. Томск : Изд-во ТУСУР, 2013. 340 с.

Лабораторная работа № 5
Измерение и оценивание систем

Тема занятия: измерение и оценивание систем.

Цель занятия: изучить особенности и характеристики основных типов измерительных шкал: наименований (номинальная), порядка (ранговая), интервалов, отношений и абсолютную шкалу. Выбор шкалы для измерения показателей, дающих представление о состоянии системы.

Материалы и программное обеспечение:

1. РМ Г 83-2007. Шкалы измерений. Термины и определения.
2. Табличный процессор: Excel/Base из MS Office / LibreOffice, Google Docs.
3. Текстовый редактор: MS Word / Writer LibreOffice, Google Docs.

Краткие теоретические сведения

Моделирование тесно связано с *экспериментом*. С одной стороны, по результатам проведения эксперимента можно уточнить модель, чтобы приблизить ее к реальности, с другой – модель задает параметры проводимого эксперимента. Эксперимент называется *пассивным*, если экспериментатор не вмешивается в ход событий, а ограничивается наблюдением. В том случае, когда экспериментатор задает некоторые из поступающих на входы системы значений, то эксперимент называется *активным*, или *управляемым*. Если интересующая экспериментатора характеристика может быть измерена напрямую, то эксперимент называется *прямым*, в противном случае – *косвенным*. В *косвенном* эксперименте наблюдают не саму нужную характеристику, а некую связанную с ней величину, по которой можно судить об этой характеристике, например, о тяжести состояния больного при остром респираторном вирусном заболевании косвенно свидетельствует температура тела.

Измерение можно рассматривать как отображение множества реальных объектов, ситуаций, событий или процессов на множество элементов значений знаковой системы, называемой измерительной шкалой. Наиболее известна типология измерительных шкал С.С. Стивенса, предложенная им в 1946 г. Согласно этой типологии, выделяют четыре вида измерительных шкал.

В настоящее время более распространена типология, включающая в себя пять видов базовых шкал измерений.

1. *Шкала наименований (номинальная или классификационная)* используется для измерения значений качественных признаков и основана на классификации. Результат измерений состоит в том, что объект относится к одному из заранее определенных классов на основании наблюдений некоторых классификационных признаков.

Для построения номинальной шкалы на множестве объектов задается разбиение на классы эквивалентности $A_1, A_2, \ldots, A_M$.

Классы эквивалентности удовлетворяют следующим аксиомам эквивалентности:

$A = A$ (рефлексивность);

$A = B \Leftrightarrow B = A$ (симметричность);

$A = B,\ B = C \Rightarrow A = C$ (транзитивность).

Пусть поочередно наблюдаются объекты $x_1, x_2, \ldots, x_N$. Их совокупность называется выборкой объема N. Для каждого из объектов устанавливается класс эквивалентности, к которому он относится ($x_i \in A_k$). Над полученными данными можно установить операцию проверки на совпадение $\delta(x_i, x_j)$:

$$\delta(x_i, x_j) = \begin{cases} 1, & x_i \sim x_j; \\ 0, & x_i \nsim x_j. \end{cases} \tag{5.1}$$

Проверка на совпадение – единственный вид первичной обработки данных, которые были получены с использованием *номинальной* шкалы. Над результатом первичной обработки можно выполнить вторичную обработку данных, например, находить моды или частоты. Широко используют относительную частоту W_i наблюдения i-события, которое заключается в том, что объект отнесен к классу, включающему x_i:

$$W_i = \frac{1}{N} \sum_{j=1}^{N} \delta(x_i, x_j). \tag{5.2}$$

При использовании *относительной* частоты строится множество методов математической статистики, например, критерий согласия Пирсона, критерий Мак-Немара, критерий Фишера.

Номинальные шкалы часто используются в социально-экономической сфере, где в качестве объекта изучения выступает человек и (или) общество.

Пример 1. Группа крови человека, национальность, профессия, имена, виды образовательных учреждений, страны мира, наименования товаров из ассортимента торгового предприятия.

В некоторых случаях для названия классов эквивалентности могут быть использованы числа. Например, направления и специальности высшего образования в России имеют числовые коды. Однако необходимо помнить, что с ними нельзя обращаться как с числами, сравнивать их друг с другом или устанавливать иные соответствия между классами.

2. *Порядковая (ранговая) шкала* предполагает, что можно не только выделить различные классы объектов, но и сравнивать эти классы между собой по некоторому признаку. Это происходит, если кроме аксиом эквивалентности для классов объектов установлены также аксиомы упорядоченности:

$$A \neq B \Rightarrow A < B \text{ или } B < A;$$

$$A < B,\ B < C \Rightarrow A < C.$$

Иначе говоря, каждую пару классов можно упорядочить по некоторому признаку. Такая шкала называется шкалой *совершенного* порядка.

Пример 2. Классы учеников в школе. Распределение мест участников соревнований.

Если выясняется, что некоторые классы по рассматриваемому признаку равны друг другу, то это означает, что введена шкала квазипорядка, и для классов объектов действуют аксиомы частичной упорядоченности:

$$A \neq B \Rightarrow A \leq B \text{ или } B \leq A;$$

$$A \leq B,\ B \leq C \Rightarrow A \leq C.$$

Примером шкалы квазипорядка является степень родства между людьми: мать/отец и сын/дочь образуют упорядоченную пару, но пары «мать и отец», «сын и дочь» считаются равными.

Кроме того, порядковые шкалы не всегда имеют заранее заданные эталонные классы, к которым во время эксперимента относят каждый из наблюдаемых объектов. Иногда для упорядочения объектов проводится их попарное сравнение, например, при составлении рейтинга миллиардеров в газете Forbes.

Возможна также ситуация, когда некоторые пары объектов не сравнимы между собой, т.е. не выполняется ни $A \leq B$, ни $B \leq A$. В этом случае говорят о шкале частичного порядка. Так, при изучении потребительского спроса может выясниться, что покупатель может упорядочить по предпочтению часть товаров, например, чай и кофе, но не может выбрать предпочтительный товар из пары чай и кефир.

Для данных, измеренных с помощью *порядковых* шкал, возможны все операции и способы обработки, допустимые для *номинальных* шкал. Кроме того, в дополнение к операции проверки на совпадение, можно ввести операцию проверки на предпочтение, $C(x_i, x_j)$:

$$C(x_i, x_j) = \begin{cases} 1, & x_i \geq x_j; \\ 0, & x_i < x_j. \end{cases} \tag{5.3}$$

В результате вторичной обработки данных можно вычислить ранг R_i – номер каждого наблюдения в упорядоченном ряду данных:

$$R_i = \sum_{j=1}^{N} C(x_i, x_j). \tag{5.4}$$

На вычислении ранга основаны такие методы математической статистики, как определение коэффициентов ранговой корреляции Спирмена, Кендалла и др.

Ранги, как правило, имеют числовые обозначения, однако необходимо помнить, что результаты измерений в порядковой шкале не являются числами. Установление факта, что ранги двух наблюдений отличаются между собой на величины m или в n раз, не позволяют сделать вывод, что предпочтительность этих наблюдений отличается

на величины, пропорциональные m или в kn раз. На основании ранга можно определить лишь то, какой из результатов наблюдений является более предпочтительным.

Часто исследователи стремятся ввести дискретную балльную шкалу для оценки непрерывных величин, например, шкалы силы ветра по Бофорту, магнитуд землетрясений по Рихтеру. Следует заметить, что при переходе к *порядковой* шкале возможность работать с баллами как с числами утрачивается. Так, теряется смысл вычисления средних значений, в том числе среднего арифметического. Это искусственно ограничивает возможности по обработке данных, поэтому введение дискретных шкал для непрерывных величин – не всегда хороший выбор.

3. *Интервальная шкала (шкала разностей)* может быть получена, если упорядочивание объектов проводится так, что можно определить расстояние между любыми двумя из них. При этом объективно равные интервалы измеряются одинаковыми отрезками шкалы, а начало отсчета и единицу длины можно выбрать произвольно, вследствие чего они могут не совпадать в разных шкалах.

Пример 3. Температура воздуха измеряется в *интервальных* шкалах Цельсия, Фаренгейта, Кельвина; летоисчисление ведется от Рождества Христова, Сотворения мира или переезда Мухаммеда в Медину. Точки отчета в различных шкалах могут не совпадать, а единицы измерения при этом могут быть одинаковые.

Кроме операций, введенных для порядковых шкал, допустимо определение интервала между двумя значениями. Интервал при этом является числом и допускает вторичную обработку данных в виде любых арифметических операций, статистической обработки и т.д.

Если замкнуть интервальную шкалу так, чтобы после прохождения определенного периода ее значения повторялись, получим *циклическую* шкалу, с помощью которой измеряют угловые направления, фазы колебаний, время суток, день года и т.д.

4. *Шкала отношений* может быть получена из интервальной путем фиксации нулевой отметки, что достигается за счет привязки ее к некоторому особому значению. Такая шкала удобна, например, для измерения длин, площадей, объемов, веса.

Из двух степеней свободы интервальной шкалы – свободы выбора нулевой отметки и единицы измерения – остается только одна. Этим обусловлено параллельным существованием нескольких шкал для одних и тех же величин.

Пример 4. Масса может быть измерена в метрической системе единиц (грамм, килограмм, центнер, тонна), английской системе мер (стоун, фунт, унция), британской аптечной системе (аптечный фунт, тройская унция, драхма), русской системе мер (золотник, фунт, пуд) и других системах, распространенных в разных географических областях в определенные периоды истории.

Шкалы отношений являются числовыми и допускают первичную обработку в виде любых арифметических операций, а также любую вторичную обработку.

5. *Абсолютная шкала* лишена степеней свободы в том смысле, что в ней невозможен выбор ни точки отсчета, ни единицы измерения, поскольку в ней заданы абсолютный нуль и абсолютная единица. Значения данных в этой шкале не имеют размерности и наименований.

Пример 5. Абсолютные шкалы для различных коэффициентов.

Выбор шкалы измерения не ограничивается перечисленными базовыми пятью вариантами. Существуют также другие, принципиально отличные от них классы шкал.

В реальной практике также могут быть использованы *нелинейные шкалы*. В них единицы измерения в разных частях шкалы не равны друг другу. Например, стоимость некоторого товара в середине прошлого и начале нынешнего века может быть выражена в рублях и копейках, однако, очевидно, что эти рубли и копейки не равны друг другу. Чтобы сравнить стоимость товара в разных временных периодах, ее необходимо сопоставить со стоимостью других товаров или значениями иных экономических величин.

Следует учитывать, что измерение непрерывных величин объективно проводится с некоторой точностью, заданной используемой измерительной аппаратурой. Это привело к распространению так называемых *дискретизированных шкал*. При измерениях в дискретизированных шкалах не фиксируют дробные значения показателей, а усреднение показателей может как повысить, так и понизить точность результата.

В некоторых случаях классификация строится не на основе отношения эквивалентности, вследствие чего элемент может принадлежать сразу к нескольким классам. Для определения степени уверенности в том, что элемент принадлежит некоторому классу, вводится функция *принадлежности*, значения которой ложатся в основу так называемой *нечеткой измерительной шкалы*.

Таким образом, при углублении в проблематику измерения и оценивания систем возникает множество вопросов, связанных с целесообразностью выбора шкалы, соответствием этой шкалы природе измеряемых величин, допустимости тех или иных способов обработки полученных данных. Возможные ошибки в выборе величин для измерений, способов их измерений и обработки полученных данных приводят к неверной интерпретации результатов эксперимента.

Статистические оценки, использующиеся для обработки экспериментальных данных

Выборка случайной величины X характеризуется множеством *n* значений переменной x_i:

$$X : \{x_i,\ i = 1,...,n\}.$$

Предполагается, что выборка *X* берется из генеральной совокупности ***X***.

Среднее значение (выборочное среднее значение) – термин, который характеризует случайную величину *X*, а именно то значение переменной, которое чаще всего встречается в выборке. Среднее значение величины *X* обозначается $\overline{X}$ и вычисляется по формуле

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (5.5)$$

и называется *выборочным средним* в математической статистике.

Выборочное среднее является оценкой *математического ожидания* случайной величины *X*. Математическое ожидание случайной величины *X* обозначается *E*[*X*].

Дисперсия – мера изменчивости, вариации данных, обозначается $D[X]$.

Дисперсия определяется как математическое ожидание квадрата разности $D[X] = E[(X - E[X])^2]$. Оценка дисперсии обозначается S^2 и вычисляется как сумма квадратов отклонений значений $(x_i - \overline{X})^2$ переменной x_i от выборочного среднего $\overline{X}$:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \frac{1}{n}\sum_{i=1}^{n}x_i\right)^2. \qquad (5.6)$$

Стандартное отклонение (среднеквадратичное отклонение), σ – мера изменчивости, разброса (вариации) данных. Вычисляется, как квадратный корень из оценки дисперсии:

$$\sigma = \sqrt{D[X]} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2}. \qquad (5.7)$$

Любое распределение случайной величины характеризуется *стандартным отклонением*. Точное значение *стандартного отклонения* для генеральной совокупности *X* не известно. Поэтому для оценок рассматривают стандартную ошибку среднего.

Стандартная ошибка среднего оценивает выборочную изменчивость выборочного среднего $\overline{X}$, приближенно показывая, насколько *выборочное среднее* отличается от *среднего генеральной совокупности*, обозначается μ и вычисляется по формуле

$$\mu = \frac{\sigma}{\sqrt{n}} = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2}. \qquad (5.8)$$

Результат эксперимента записывается в виде *среднего значения* с соответствующей *стандартной ошибкой среднего*: $\overline{X} \pm \mu$.

Пример 6. Оценить экспериментальную технологию обучения на основе оценок школьников (табл. 5.1) из экспериментальной группы, полученных после выполнения набора заданий.

Таблица 5.1

Результаты среза знаний экспериментальной группы, полученные на подготовительном этапе

№	Фамилия, имя учащегося	Количественные показатели по каждому из выполненных заданий						Контроль	
		1	**2**	**3**	**4**	**5**	**6**	Балл	Оценка
1	Илеев Михаил	9	10	10	12	7	12	60	5
2	Кинова София	10	8	10	15	8	12	61	5
3	Киселёв Антон	7	7	6	12	7	9	48	3
4	Игнатов Виктор	6	7	8	9	7	12	49	3
5	Мур Александр	10	10	10	12	6	15	63	5

№	Фамилия, имя учащегося	Количественные показатели по каждому из выполненных заданий						Контроль	
		1	**2**	**3**	**4**	**5**	**6**	Балл	Оценка
6	Павиченко Александра	7	8	9	12	7	9	52	4
7	Пищук Алина	7	7	8	12	6	12	52	4
8	Соловьёва Марина	9	10	7	15	7	12	60	5
9	Сокин Михаил	9	10	8	9	7	12	55	4
10	Петров Максим	8	9	8	12	8	12	57	4
11	Лаврентьев Артём	10	10	9	15	8	15	67	5

Используя формулы (5.5)–(5.8), проведем расчеты описательных статистик в табл. 5.2 и построим сравнительную диаграмму для средних баллов (рис. 5.1).

Таблица 5.2

Описательные статистики начального среза знаний экспериментальной группы, полученные на подготовительном этапе

№	Описательные статистики	Количественные показатели по каждому из выполненных заданий						Контроль	
		1	2	3	4	5	6	Балл	Оценка
1	Среднее значение	8,36	8,72	8,45	12,27	7,09	12,00	56,72	4,27
2	Минимальное значение	6	7	6	9	6	9	48	3
3	Максимальное значение	10	10	10	15	8	15	67	5
4	Ст. отклонение	0,50	1,35	1,29	2,10	0,87	1,90	6,06	0,79
5	Ст. ошибка среднего	0,15	0,41	0,39	0,63	0,26	0,57	1,83	0,24

Рис. 5.1. Сравнение средних оценок за выполненные задания на подготовительном этапе в экспериментальной группе

Задания к работе

1. Охарактеризовать сессию как эксперимент по определению качества образовательного процесса. Является он активным или пассивным? Какие величины наблюда-

ются? Как проводятся измерения? В каких шкалах оцениваются наблюдаемые величины? Как обрабатываются полученные данные? Насколько сессия решает задачу по определению качества образовательного процесса?

2. Обсудить достоинства и недостатки сессии как эксперимента с целью определения качества обучения (на занятии в малых группах по 3–5 человек).

3. Разработать систему оценки успеваемости студентов по одному из предметов образовательной программы так, чтобы она максимально адекватно отражала качество обучения, но при этом процесс оценки не был слишком трудоемким. В какой шкале измеряется эта оценка?

4. Использовать процедуру обработки экспериментальных данных для этих оценок, дающую возможность оценить качество обучения студентов и качество работы преподавателей. Подготовить таблицы с оценками, найти минимальные/максимальные значения, средние, среднеквадратичное отклонение, среднеквадратичную ошибку.

Методика выполнения работы

Перед выполнением задания необходимо изучить государственный стандарт РМ Г 83-2007. В малых группах (2–3 человека) обсудить выбранную систему оценок и используемую шкалу, согласованную с преподавателем. Подготовить описание шкалы измерений, охарактеризовать ее свойства. При статистических расчетах использовать табличный процессор MS Office Excel / LibreOffice.org Base, а для формирования текста – текстовый редактор MS Word / LibreOffice.org Writer, Google Docs.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет по выбранной шкале. Отчет по работе должен содержать: таблицы с экспериментальными данными, таблицы с результатами статистической обработки, сравнительные графики и пояснительный текст, описывающий основные этапы работы, а также полученные результаты. Сформировать текстовый документ в электронной форме (.docx) и представить преподавателю.

Вопросы для самоконтроля

1. В чем разница между активным и пассивным экспериментом?

2. Описать возможности номинальной шкалы измерения. Какие величины целесообразно измерять в этих шкалах?

3. Какие действия разрешены с величинами, измеряемыми в порядковых шкалах?

4. Что общего и в чем различие между интервальной шкалой, шкалой отношений и абсолютной шкалой?

Рекомендуемая литература

1. Силич, М. П. Теория систем и системный анализ / М. П. Силич, В. А. Силич. Томск : Изд-во ТУСУР, 2013. 340 с.

2. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

3. РМГ 83-2007. Шкалы измерений. Основные положения. Термины и определения. Москва : Стандартинформ, 2008. 24 с.

Лабораторная работа № 6
Формализация задачи и структурное моделирование

Тема занятия: формализация, задачи и структурное моделирование.

Цель занятия: преобразовывание неформального, текстового описания проблемной ситуации в набор взаимосвязанных формальных моделей, однозначно определяющих границы, элементы, структуру и цели системы.

Материалы и программное обеспечение:

1. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования.
2. Программное обеспечение: RAMUS, табличный процессор MS Excel / Base LibreOffice, Google Docs.
3. Язык программирования: Python, система программирования: VstudioCode.

Краткие теоретические сведения

Формализация с точки зрения системного подхода – это процесс представления знаний, структур, процессов и взаимосвязей системы в строго определённом виде с использованием формальных (логически и математически обоснованных) языков и моделей. *Формализация* в системном подходе представляет собой своеобразный методологический мост между качественным пониманием системы и её количественным, алгоритмическим исследованием. Она превращает размытые идеи в точные конструкции, пригодные для анализа, моделирования и управления.

Формализация позволяет перейти от интуитивных представлений к строгому, воспроизводимому описанию системы, выявлять скрытые закономерности и противоречия в описании, проводить количественный анализ (*устойчивость, чувствительность, оптимальность*), создавать компьютерные инструменты поддержки принятия решений.

Основные цели формализации:

– устранить неоднозначность и субъективность в описании системы;

– обеспечить однозначное понимание элементов, связей и правил функционирования;

– создать базу для анализа, моделирования, прогнозирования и оптимизации системы;

– позволить автоматизированную обработку знаний (в том числе с помощью ЭВМ);

– облегчить сравнение, верификацию и трансляцию знаний о системе.

В рамках системного подхода при анализе предметной области *формализуют*:

– *структуру системы* – состав элементов и типы связей между ними (графы, матрицы смежности, онтологии);

– *функции и процессы* – алгоритмы, сценарии, потоки работ (блок-схемы, сети Петри, UML-диаграммы);

– *динамику системы* – изменения состояний во времени (дифференциальные уравнения, дискретные модели, имитационные модели);

– *правила и ограничения* – логические условия, инварианты, целевые функции (логические формулы, оптимизационные постановки);

– *знания о системе* – понятия, отношения, закономерности (дескрипционные логики, семантические сети, базы знаний).

Инструменты и языки формализации:

– *математические модели* (уравнения, неравенства, функции, графы);

– *логические системы* (исчисления высказываний и предикатов, дескрипционные логики);

– *языки моделирования* (UML, BPMN, SysML, IDEF);

– *формальные грамматики и синтаксисы* (для спецификаций, языков предметных областей);

– *онтологии и семантические модели* (RDF, OWL, концептуальные схемы);

– *компьютерные модели* (симуляции, агентные модели, системная динамика, дискретно-событийное моделирование).

Алгоритм первичной формализации:

– Шаг 1. *Выделение ключевых существительных и глаголов* из текста описания – потенциальные *элементы и функции/процессы*.

– Шаг 2. *Определение миссии и цели анализа*. Ответ на вопросы: «Зачем анализируем эту систему? Какой результат необходимо получить?» (Не цель системы, а цель *собственного анализа*).

– Шаг 3. *Определение границ*. Составление списков:

3.1. Что находится *«внутри»* системы и будет детально изучаться?

3.2. Что находится *«вне»* системы, но активно с ней взаимодействует (внешняя среда)?

3.3. Что находится *«за пределами»* рассмотрения (excluded)?

– Шаг 4. *Идентификация стейкхолдеров* и их интересов.

Обзор инструментов структурного моделирования

– *Контекстная диаграмма (IDEF0 A0).* Единственный блок. Для фиксации *границ* и основных *взаимодействий* (Входы, Выходы, Управление, Механизмы) *системы* с внешним миром.

– *Диаграмма потоков данных (DFD).* Для моделирования *информационных потоков* (данные, документы, сигналы) показывает, как информация перемещается между *процессами, хранилищами данных и внешними сущностями*.

– *Матрица смежности/инцидентности.* Для *математического анализа структуры* связей между элементами.

Анализ структурно-топологических характеристик информационных систем

Структурные характеристики системы позволяют уже на ранней стадии создания оценить качество ее структуры и элементов с позиции общего системного подхода.

Если структурная схема системы представлена в виде графа (ориентированного или неориентированного), то существует ряд количественных оценок для сравнения различных вариантов построения систем.

Степень вершины в графе – количество рёбер, *инцидентных* (примыкающих) к данной вершине.

1. Для *неориентированного графа* просто считаются все рёбра, выходящие из вершины.

Пример. Из вершины *A* выходят три ребра, то степень вершины deg(*A*) = 3.

2. Для *ориентированного графа* различают:

– *полустепень исхода* $\deg^+(v)$ – число рёбер, *выходящих* из вершины *v*;

– *полустепень захода* $\deg^-(v)$ – число рёбер, *входящих* в вершину *v*.

3. *Ребро-петля* (ребро из вершины и в неё же) учитывается дважды при подсчёте степени в неориентированном графе.

Связность структуры R характеризует силу (мощность) связей в системе.

Для ориентированного графа

$$R = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \geq n - 1 \tag{6.1}$$

и неориентированного графа

$$R = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \geq n - 1, \tag{6.2}$$

где n – число элементов в системе, a_{ij} – элемент матрицы смежности A.

Структурная избыточность α – параметр, оценивающий превышение числа связей системы над минимально необходимым:

$$\alpha = \frac{R - R_{\min}}{R_{\min}} = \frac{R}{n-1} - 1, \tag{6.3}$$

где $\alpha = 0$ – минимальная избыточность, $\alpha > 0$ – максимальная избыточность, $\alpha < 0$ – несвязная система.

Структурная компактность Q характеризует инерционность информационных процессов в системе:

$$Q = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}, (i \neq j), \tag{6.4}$$

где d_{ij} – элемент матрицы расстояний *D*, характеризующий меру близости элементов *i* и *j*.

Степень централизации структуры (графа). Степень централизации характеризуется индексом δ.

Для структуры типа неориентированный граф

$$\delta = (n-1)(2Z_{\max} - n)\frac{1}{Z_{\max} - 2}, \tag{6.5}$$

где $Z_{\max}$ – максимальное значение величины:

$$Z_i = \frac{Q}{2}\left(\sum_{j=1}^{n} d_{ij}\right)^{-1}, \; i = 1..., n. \quad (i \neq j). \tag{6.6}$$

Для структуры типа ориентированный граф

$$\delta = \frac{1}{(n-1)(V(k)-1)}\sum_{j=1}^{n} V(k) - V(i)), \tag{6.7}$$

где $V(i)$ – суммарное число входящих и исходящих ребер i-й вершины; $V(k) = \max\limits_{i} V(i)$.

Степень централизации вершины графа. Степень посредничества (англ. Betweenness centrality, «центральность по посредничеству») – мера важности вершины в графе как *«посредника»* на кратчайших путях между другими вершинами.

В любом связном графе между парой вершин существует хотя бы один *кратчайший путь* (по числу рёбер – для невзвешенных графов; по сумме весов – для взвешенных).

Степень посредничества вершины v показывает, как часто v лежит на кратчайших путях между всеми возможными парами других вершин графа.

Для вершины v величина $g(v)$ вычисляется по формуле

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{6.8}$$

где s и t – все возможные пары вершин, не совпадающие с v; σ_{st} – общее число кратчайших путей из s в t; $\sigma_{st}(v)$ – число кратчайших путей из s в t, проходящих через v.

Высокое значение $g(v)$: вершина v – ключевой «перекрёсток». Через неё проходит много кратчайших путей; она связывает разные части графа.

Пример. В социальной сети это человек, который соединяет разные группы; в транспортной сети – узел (станция, перекрёсток), без которого значительно растёт длина маршрутов.

Низкое значение $g(v)$: вершина v не играет роли посредника, большинство кратчайших путей её не используют.

Степень близости (англ. closeness centrality, «центральность по близости») – мера важности вершины в графе, отражающая, насколько она «близка» ко всем остальным вершинам через кратчайшие пути.

Вершина с высокой *степенью близости*:

− находится «в центре» графа;

− может быстро «добраться» до любой другой вершины (по числу шагов или суммарному весу рёбер);

− эффективна для распространения информации: сигнал от неё достигает всех узлов за минимальное время/расстояние.

Для вершины v величина *степень близости* $C(v)$ вычисляется как обратная величина суммы длин кратчайших путей от v до всех остальных вершин:

$$C(v_i) = \left(\sum_{j=1}^{n} d_{ij} \right)^{-1}, \; i = 1,...,n. \quad (i \neq j), \tag{6.9}$$

где n – количество всех вершин графа; d_{ij} – элемент матрицы расстояний D, характеризующий меру близости элементов i и j (в невзвешенном графе – число рёбер; во взвешенном – сумма весов рёбер).

Внутренние связи в структуре, с топологической точки зрения, могут иметь следующие типы структур, изображенные на рис. 6.1.

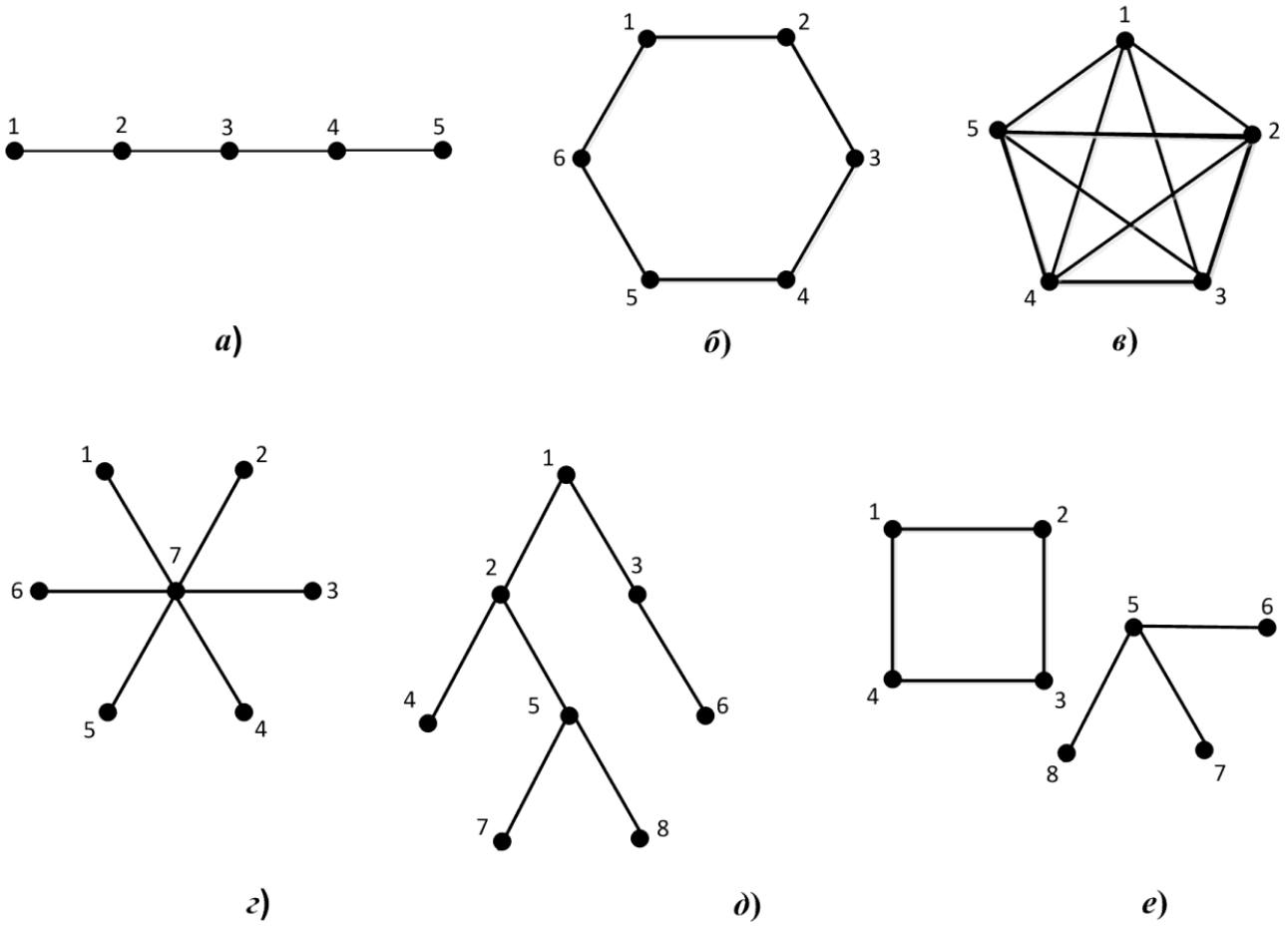


Рис. 6.1. Типы структур: *а* – последовательная; *б* – кольцевая; *в* – «полный граф»; *г* – радиальная; *д* – древовидная; *е* – несвязная

Расчет топологических характеристик системы

Пример. На рис. 6.2 изображена структура с $n = 7$. Определить: R – связность, α – структурную избыточность, Q – структурную компактность и δ – степень централизации структуры (графа).

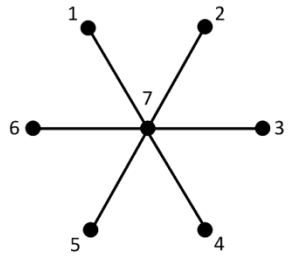


Рис. 6.2. Пример структуры с $n = 7$

1. Определим связность структуры с использованием формулы (6.1). Построим матрицу смежности.

$$Q = \begin{pmatrix} 0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,1 \\ 1,1,1,1,1,1,0 \end{pmatrix}$$

Для неориентированного графа рассчитаем по формуле связность R:

$$R = \frac{1}{2}\sum_{i=1}^{7}\sum_{j=1}^{7} a_{ij} = \frac{1}{2}(1+1+1+1+1+1+1+6) = 6 \geq n-1.$$

$R = n - 1 = 6$, где $n = 7$, поэтому система связна.

2. Структурная избыточность. Используем формулу (6.3):

$$\alpha = \frac{R}{n-1} - 1 = \frac{6}{6-1} - 1 = 0,2\,.$$

Поскольку $\alpha > 0$, то граф избыточный, т.е. число связей превышает минимально необходимое количество для обеспечения связности ее элементов.

3. Структурная компактность. Используем формулу (6.4).

$$Q = \begin{pmatrix} 0,2,2,2,2,2,1 \\ 2,0,2,2,2,2,1 \\ 2,2,0,2,2,2,1 \\ 2,2,2,0,2,2,1 \\ 2,2,2,2,0,2,1 \\ 2,2,2,2,2,0,1 \\ 1,1,1,1,1,1,0 \end{pmatrix}$$

Рассчитаем структурную компактность на основе расстояний в матрице Q:

$$Q = \sum_{i=1}^{7}\sum_{j=1}^{7} d_{ij} = 6(2+2+2+2+2+1) + 6 = 72.$$

4. Степень централизации.

Для определения индекса централизации δ воспользуемся формулами (6.5) и (6.6).

$$Z_i = \frac{Q}{2}\left(\sum_{j=1}^{n} d_{ij}\right)^{-1}, \; i = 1..., n. \quad (i \neq j),$$

$$Z_i = \frac{72}{2}\left(\sum_{j=1}^{n} d_{ij}\right)^{-1} = 36\left(\frac{1}{11}\right) = 3,27, \text{для } i = 1..., 6,$$

$$Z_7 = \frac{72}{2}\left(\sum_{j=1}^{n} d_{7j}\right)^{-1} = 36\left(\frac{1}{6}\right) = 6, \text{ для } i = 7,$$

тогда $Z_{\max} = 6$.

Для структуры типа неориентированный граф

$$\delta = (n-1)(2Z_{\max} - n)\frac{1}{Z_{\max} - 2} = 6(12-6)\frac{1}{6-2} = 9.$$

Структура абсолютно централизована.

Система рекомендаций для стримингового сервиса

Пример. Описание сервиса. Сервис «К-Поток» предоставляет подписчикам доступ к фильмам и сериалам. Для увеличения времени просмотра и удержания клиентов используется система рекомендаций. Она анализирует историю просмотров, явные оценки (лайки/дизлайки), время суток просмотра, популярный контент среди друзей в социальных сетях (при наличии интеграции) и метаданные контента (жанр, режиссер, актеры). Алгоритм формирует персональную ленту «Рекомендуем вам». Пользователи иногда жалуются на однообразие рекомендаций или на неуместные предложения. Команде необходимо понять, как улучшить систему, не нарушая приватность пользователей и не увеличивая вычислительные затраты.

Система рекомендаций для стримингового сервиса – комплекс алгоритмов и технологий, который анализирует поведение пользователя и предлагает ему наиболее подходящий контент. Цель – помочь пользователям находить контент, соответствующий их вкусам и предпочтениям.

Принцип работы системы

Работа системы рекомендаций основана на трёх ключевых принципах:

1. Сбор данных. Система непрерывно отслеживает все действия пользователя: историю просмотров, оценки и лайки, время, проведённое за просмотром контента, промежуточные остановки, поисковые запросы и социальное взаимодействие, если оно доступно на платформе.

2. Анализ данных. Алгоритмы выявляют паттерны поведения, определяют предпочтения пользователя, группируют похожих пользователей между собой и классифицируют контент по различным параметрам.

3. Формирование персонализированного списка рекомендаций. Предложения ранжируются по степени релевантности, и система постоянно обновляет их в реальном времени, учитывая новые действия пользователя.

Существует несколько основных подходов к формированию рекомендаций:

1. *Коллаборативная фильтрация* – анализирует поведение групп пользователей с похожими предпочтениями. Например, если несколько человек высоко оценили один и тот же фильм, система предполагает, что им понравятся и другие похожие фильмы.

2. *Контентная фильтрация* – фокусируется на характеристиках самого контента. Например, если пользователю нравится фильм в жанре научной фантастики с высоким рейтингом, система будет искать похожие фильмы с аналогичными характеристиками.

3. *Гибридные системы* – сочетают различные подходы для достижения максимальной точности рекомендаций. Такие системы учитывают контекст просмотра, сезонные тренды, новинки рынка и социальное влияние.

В рекомендательных системах часто используют нейросети. Например, рекуррентные нейросети (RNN) хорошо справляются с последовательными данными, такими как история просмотров, а автокодировщики помогают выявлять скрытые паттерны в пользовательских предпочтениях.

Используемые технологии

В основе современных рекомендательных систем лежат алгоритмы машинного обучения. Ключевой компонент – глубокое обучение (Deep Learning), которое реализуется через нейронные сети, способные выявлять сложные нелинейные зависимости в данных.

В рекомендательных системах используют технологию обработки естественного языка (Natural Language Processing, NLP), которая анализирует не только прямые оценки и отзывы пользователей, но и более сложные языковые конструкции, включая эмоциональные оттенки и сленговые выражения.

Оценка эффективности рекомендательных систем – многогранная задача, требующая комплексного подхода. Необходимо учитывать бизнес-метрики, удовлетворённость пользователей и технические аспекты работы системы.

Оценку рекомендательных систем разделяют на три основные категории:

1. *Офлайн-оценка* – использует исторические данные для измерения точности предсказаний.

2. *Онлайн-оценка* – измеряет реальное взаимодействие пользователей с рекомендациями.

3. *Пользовательские исследования* – собирают явные отзывы о качестве рекомендаций.

Некоторые метрики оценивают разнообразие элементов в рекомендациях; новизну – способность рекомендовать неизвестные пользователю элементы; охват – долю элементов, которые система способна рекомендовать.

Примеры стриминговых сервисов, где используются системы рекомендаций: Netflix, Spotify, YouTube.

1. Первичный анализ. Из текста выделяются пять ключевых *сущностей* (объектов) и пять ключевых *процессов/функций*.

1.1. Сущности: «Пользовательский профиль», «Лог просмотров», «Метаданные контента», «Рекомендательная лента», «Модель ML».

1.2. Процессы: «Сбор данных о поведении», «Обучение / актуализация модели», «Формирование ранжированного списка рекомендаций», «Отображение ленты пользователя», «Сбор обратной связи».

2. Формулирование цели и определение границ системы. Формулируется *цель системы* с точки зрения владельца (разработчика): оптимизировать алгоритм формирования ленты рекомендаций для увеличения метрики «процент кликов по рекомендациям» (CTR) на 15% при прежних ограничениях на вычислительные ресурсы и соблюдении, соответственно, регуляторных норм (GDPR).

Границы системы. Пример выделения границ системы представлен на рис. 6.3.

Рис. 6.3. Ядро системы (алгоритм ранжирования, механизм сбора и обработки), внешняя среда (каталог контента, сервис аутентификации пользователей, биллинг, внешние социальные сети) и то, что за пределами рассмотрения (CDN, маркетинговые рассылки)

3. Построение контекстной диаграммы (IDEF0 A0). Построим диаграмму A0 для системы «Формирование персональной ленты рекомендаций» на рис. 6.4.

Рис. 6.4. Контекстная диаграмма A0 в нотации IDEF0

4. Построение диаграммы DFD (уровня 1). С использованием программы RAMUS построим диаграмму DFD (рис. 6.5).

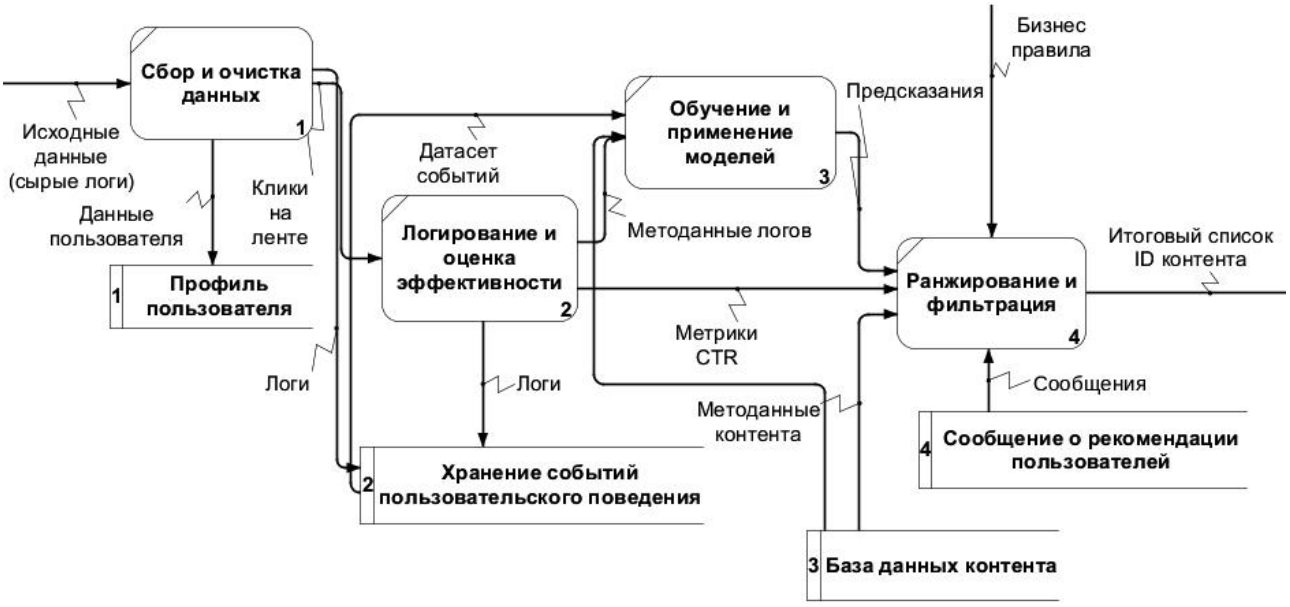


Рис. 6.5. Диаграмма модели в нотации DFD

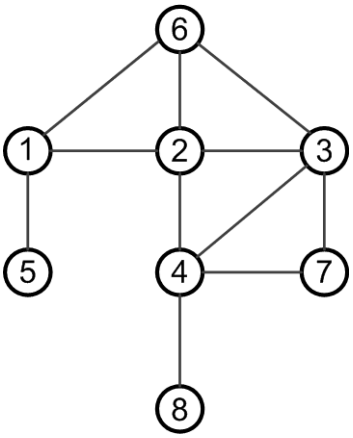
Основные процессы разрабатываемой системы представлены в табл. 6.1.

Таблица 6.1

Основные процессы системы «Формирование персональной ленты рекомендаций»

№	Наименование процесса	Вход	Выход
1	Сбор и очистка данных	Сырые логи	Подготовленные датасеты
2	Обучение и применение моделей	Датасеты	Предсказания релевантности
3	Ранжирование и фильтрация	Предсказания	Итоговый список ID контента
4	Логирование и оценка эффективности	Клики по ленте	Метрики CTR

5. Формальный структурный анализ (графы и матрицы). Представим диаграммы DFD в виде направленного графа G(V,E) на рис. 6.6, где V – количество вершин, а E – количество ребер.



1. Сбор и очистка данных.
2. Обучение и применение моделей.
3. Ранжирование и фильтрация.
4. Логирование и оценка эффективности.
5. Профиль пользователя.
6. Хранение событий пользовательского поведения.
7. База данных контента.
8. Сообщения о рекомендациях пользователей.

Рис. 6.6. DFD в виде ненаправленного графа G(8,10)

Построим матрицу смежности:

$$Q = \begin{pmatrix} 0,1,0,0,1,1,0,0 \\ 1,0,1,1,0,1,0,0 \\ 0,1,0,1,0,1,1,0 \\ 0,1,1,0,0,0,1,1 \\ 1,0,0,0,0,0,0,0 \\ 1,1,1,0,0,0,0,0 \\ 0,0,1,1,0,0,0,0 \\ 0,0,0,1,0,0,0,0 \end{pmatrix}$$

Задание. Используя библиотеку Python NetworkX, рассчитать:

1. Степень вершины «Обучение и применение моделей».

2. Центральность по посредничеству (betweenness centrality).

Вершина «Обучение и применение модели» будет иметь высокую центральность. Это указывает на её критическую роль – она является «мостом» между данными и финальным продуктом.

6. Анализ стейкхолдеров и требований.

Карта стейкхолдеров (Power-Interest Grid). Выявить всех стейкхолдеров системы: прямых пользователей, косвенных пользователей, внутренние команды, регуляторов и т.д.

Чтобы ранжировать влияние и заинтересованность групп для приоритизации требований, строится *карта стейкхолдеров*. Построение карты сводится к следующему: разместить на координатной плоскости (ось абцисс *x*: уровень интереса, ось ординат *y*: уровень власти/влияния) ключевых стейкхолдеров.

Пример *карты стейкхолдеров* для системы представлен на рис. 6.7.

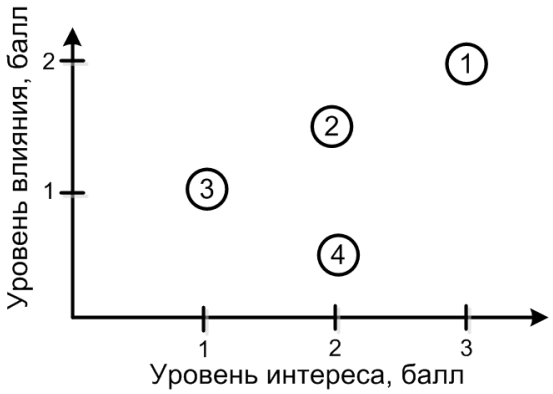


Рис. 6.7. Карта стейкхолдеров: 1 – владелец продукта; 2 – контент-менеджер; 3 – разработчик; 4 – пользователь контента

Формирование таблицы требований к системе.

Типы требований, которые должны быть представлены:

1. Функциональные (что система должна делать).

2. Нефункциональные: производительность, надежность, безопасность, удобство использования.

3. Ограничения (бюджет, сроки, технологии).

Критерии качества требований: каждое требование должно быть измеримо, у каждого требования должен быть источник (стейкхолдер), приоритеты: High/Medium/Low, нет противоречий между требованиями.

С кем нужно согласовывать решения в первую очередь? Чьи требования могут быть противоречивы?

Пример требований к системе представлен в табл. 6.2.

Таблица 6.2

Пример требований к системе

№	Требование	Тип	Стейкхолдер	Критерий приемки
R1	Рекомендации должны обновляться не реже 1 раза в сутки для каждого пользователя	Нефункциональный (Время)	Владелец продукта	95-й процентиль Задержки обновления < 24 ч
R2	Система не должна использовать для рекомендаций данные, полученные более 90 дней назад, без явного согласия пользователя	Функциональный	Контент-менеджер	Мониторинг A/B-тестов
R3	В топ-10 рекомендаций должно попадать не менее 20% нового контента (выпущенного за последний месяц)	Функциональный	Контент-менеджер	Мониторинг A/B-тестов
R4	Среднее время формирования ленты на запрос пользователя – не более 100 мс	Нефункциональный (Производит.)	Разработчик Backend	Нагрузочное тестирование

Задания к работе

1. Система «Умный кампус» (Интернет вещей и управление).

1.1. Исходное описание. Вуз внедряет систему «Умный кампус» для оптимизации расходов и повышения комфорта. Система включает: датчики движения и освещенности в аудиториях, умные термостаты, систему контроля доступа в лаборатории по картам студентов/сотрудников, агрегатор данных с потреблением энергии и воды, мобильное приложение для студентов (просмотр занятости аудиторий, отчет о проблемах). Данные стекаются в единый центр управления. Цель – автоматически регулировать освещение/отопление в пустых помещениях, прогнозировать пиковые нагрузки на инфраструктуру, обеспечивать безопасность.

1.2. Первичный анализ и определение границ.

Система «Умный кампус» – это надсистема. Выберите для детального анализа одну из её критических подсистем.

− Подсистема управления энергопотреблением.

− Подсистема контроля доступа и безопасности.

− Подсистема информационного обеспечения студентов.

Сформулировать цель анализа для выбранной подсистемы и провести её границы.

1.3. Построить диаграмму DFD уровня 0 для выбранной подсистемы.

Выделить процессы, связанные со сбором данных, анализом и принятием решений. Внешние сущности («Городская энергосеть», «Администрация корпуса»).

1.4. Построить матрицу смежности для графа подсистемы.

1.5. Выявить минимум двух стейкхолдеров, чьи интересы в рамках системы «Умный кампус» могут конфликтовать. Построить для них Power-Interest Grid.

1.6. Рассчитать базовые метрики графа (задание на Python). Визуализировать граф с помощью nx.draw. Пример создания направленного графа на основе матрицы смежности представлен на листинге 6.1.

Листинг 6.1

Создание направленного графа на основе матрицы смежности

```
import networkx as nx
import matplotlib.pyplot as plt

# Создание направленного графа на основе матрицы смежности
G = nx.DiGraph()
edges = [("Сбор метрик", "Хранилище метрик"),
         ("Хранилище метрик", "Анализ метрик"),
         ("Анализ метрик", "Очередь тикетов"),…]
G.add_edges_from(edges)
# Расчёт метрик
print("Степени захода (in-degree):", dict(G.in_degree())) # Что влияет на
элемент
print("Степени исхода (out-degree):", dict(G.out_degree())) # На что вли-
яет элемент
print("Центральность по посредничеству (betweenness):", nx.between-
ness_centrality(G))
print("Является ли граф связным?", nx.is_weakly_connected(G))
```

2. Платформа для краудсорсинговых исследований (Социально-техническая система).

2.1. Исходное описание. Ученые разработали информационную платформу, где волонтеры помогают обрабатывать данные научных проектов (классификация галактик, расшифровка древних рукописей). Ученый (исследователь) загружает датасет и инструкцию. Платформа разбивает данные на микрозадачи. Волонтер регистрируется, выбирает проект, проходит калибровочный тест, выполняет задачи. Система агрегирует ответы многих волонтеров, вычисляет согласованность и доверительный рейтинг для каждого. Исследователь получает обработанные данные. Платформа должна мотивировать волонтеров (очки, рейтинги), обеспечивать качество данных и защищать исходные данные исследований.

2.2. Выявление разнородных элементов и связей.

Составить таблицу с тремя колонками: технические элементы, организационные/социальные элементы, информационные потоки между ними.

2.3. Определить границы в социально-технической системе.

Обсудить, где проходит граница системы: включать ли в неё личную мотивацию волонтера (внутренний психологический процесс) или считать её внешней средой? Аргументировать с точки зрения возможностей воздействия на систему.

2.4. Построить карты стейкхолдеров с нетривиальными интересами.

Выявить стейкхолдеров и их интересы. Среди них будут:

− *исследователь (заказчик данных):* качество, скорость, низкая стоимость.

− *волонтер:* интересный опыт, чувство значимости.

− *этический комитет вуза:* соблюдение этики исследований, информированное согласие, защита персональных данных волонтеров.

− *IT-администратор:* надежность платформы, масштабируемость, безопасность.

2.5. Визуализировать их на Power-Interest Grid. Чьи требования, вероятно, будут наиболее конфликтными?

3. Система управления заказами в ресторане быстрого питания.

3.1. Исходное описание. Ресторан имеет следующие процессы: «Прием заказов на кассе и через мобильное приложение», «Передача заказов на кухню», «Приготовление пищи», «Сборка заказов», «Выдача заказов», «Оплата», «Управление запасами ингредиентов».

В ресторане возникают проблемы: длинные очереди в часы пик, ошибки в заказах, нехватка ингредиентов в пиковое время.

3.2. Определить границы системы для анализа. Что включить в систему, а что считать внешней средой?

3.3. Выделить ключевые подсистемы. Предложить 3–4 варианта декомпозиции и обосновать лучший.

3.4. Построить контекстную диаграмму для системы «Управление заказами в ресторане».

3.5. Определить основных стейкхолдеров и их требования. Какие требования могут конфликтовать между собой?

3.6. Предложить три структурных изменения в системе, которые могут решить указанные проблемы. Обосновать каждое предложение.

4. Оптимизация бизнес-процесса в отделе разработки программного обеспечения.

4.1. Исходное описание. В отделе разработки программного обеспечения рабочий бизнес-процесс выглядит следующим образом: продукт-менеджер (англ. Product Manager, PM) создает задачу в Jira. Технический руководитель отдела (англ. Tech Lead) оценивает её и назначает разработчика. Разработчик делает код, создает запрос на слияние изменений (англ. Merge Request, MR) в GitLab. MR должен пройти ревью от другого разработчика и автоматические CI/CD-проверки (сборка, тесты). После успеха MR вливается в основную ветку и размещается на тестовый стенд. QA-инженер тестирует

функциональность. Если находят ошибку (баг), то создается новая задача. Процесс занимает много времени, часто возникают задержки из-за нехватки ревьюеров, «падения» системы автоматизированной сборки кода CI или размытых требований в задаче.

4.2. Построить модель «AS IS». Постройте диаграмму деятельности (UML) или подробную блок-схему существующего (AS IS) процесса. Обязательно отметить на ней точки, где возможны задержки и ошибки (использовать символы решений и исключений).

4.3. Формализация проблемы через графы. Представить бизнес-процесс как простой граф состояний задачи. Статусы: Открытый (Open), In Progress, Code Review, CI, Testing, Done. Описать, какие переходы являются конфликтными (например, задача не может одновременно быть «In Progress» у двух разработчиков) или недетерминированными (например, после «CI» задача может перейти в «Testing» или «Failed»).

4.4. Качественный анализ узких мест. На основе модели «AS IS» провести мозговой штурм и составить список из пяти возможных корневых причин задержек. Использовать технику «Пять почему» для одной из них.

Методика выполнения работы

Выбрать задание из предложенных проектов. Создать проект в программе RAMUS в нотации IDEF0. Добавить элементы управления, механизмы, входы и выходы. Проверить диаграмму на соответствие правилам IDEF0. Создать декомпозицию 1-го уровня модели для каждой функции. Задания можно выполнять группой по 2–4 студента, каждая группа берет один из примеров. Создать проект в нотации DFD. Заранее подготовить шаблоны для диаграмм (например, в Draw.io) и Jupyter Notebook с заготовками кода для анализа графов (NetworkX, вывод метрик, визуализация).

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет в PDF (8–12 страниц), содержащий: титульный лист с Ф.И.О. и номером варианта, ответы на все задания, диаграммы (IDEF0, DFD), граф системы с визуализацией, таблицы (стейкхолдеры, требования), код Python (можно в виде скриншотов или в приложении).

Исходные файлы: .py файл с кодом анализа графа, файлы диаграмм (.drawio, .xml или др.), Jupyter Notebook (если использовались).

Презентация (5–7 слайдов) для краткой защиты: проблема и границы системы, ключевая диаграмма (DFD), провести анализ топологии графа, определить основные требования и конфликты.

Вопросы для самоконтроля

1. Каковы основные ограничения структурного подхода к системному анализу? В каких ситуациях его применение может быть неэффективным?

2. Сравнить подходы IDEF0 и DFD. Для каких типов систем каждый подход более подходит и почему?

3. Почему при построении графовых моделей важно учитывать не только структуру, но и динамику системы? Привести пример, когда статическая структурная модель дает неполную картину.

4. Как изменяется подход к системному анализу при переходе от технических систем к социально-техническим? Какие дополнительные аспекты необходимо учитывать?

5. Каковы этические аспекты системного анализа? Привести пример этической дилеммы, которая может возникнуть при анализе системы.

6. Как результаты структурного анализа используются при динамическом моделировании? Привести конкретные примеры связей.

7. Системный анализ часто сравнивают с медицинской диагностикой. Провести аналогию между этапами системного анализа и этапами медицинской диагностики.

8. Как цифровизация и появление больших данных изменяют подходы к системному анализу? Какие новые методы и инструменты становятся доступными?

Рекомендуемая литература

1. Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

2. Клишин, А. П. Методы и средства проектирования информационных систем и технологий : лабораторный практикум / А. П. Клишин, Ф. Д. Пираков. Томск : Издательство Томского государственного педагогического университета, 2025. 115 с.

Лабораторная работа № 7
Имитационное моделирование

Тема занятия: методика и программные инструменты имитационного моделирования.

Цель занятия: изучение методов и программных инструментов имитационного моделирования (системная динамика, дискретно-событийное моделирование), анализ результатов моделирования.

Материалы и программное обеспечение:

1. ГОСТ Р 57700.3. Численное моделирование динамических рабочих процессов в социотехнических системах. Термины и определения.
2. Программное обеспечение RAMUS. Табличный процессор: MS Excel/Base LibreOffice, Google Docs.
3. Язык программирования Python (с библиотеками: SimPy, SciPy, Matplotlib, Pandas), система программирования Visual Studio Code (VS Code).
4. Программное обеспечение: PySD (библиотека Python), Vensim PLE, Stella, AnyLogic, Visual Paradigm Online, Edraw.AI (онлайн-инструменты с готовыми шаблонами для создания диаграмм запасов и потоков).

Краткие теоретические сведения

Имитационная модель – упрощенное подобие реальной системы, либо существующей, либо той, которую предполагается создать в будущем. *Имитационная модель* обычно представляется компьютерной программой, выполнение которой можно считать имитацией поведения исходной системы во времени.

Имитационное моделирование – разработка и выполнение на компьютере программной системы, отражающей структуру и функционирование моделируемого объекта или явления во времени. Такую программную систему называют *имитационной моделью* объекта или явления. Объекты и сущности имитационной модели представляют объекты и сущности реального мира, а связи структурных единиц объекта моделирования отражаются в интерфейсных связях соответствующих объектов модели.

Термины *имитационное моделирование* и *вычислительный (компьютерный) эксперимент* соответствуют англоязычному термину *simulation.* Данные термины подразумевают разработку модели именно как компьютерной программы и исполнение этой программы на компьютере.

Исходя из вышеизложенного, термин *имитационное моделирование* описывает деятельность по разработке программных моделей сложных систем, выполняемых на компьютере, а также анализ результатов *компьютерных экспериментов* при исследовании поведения моделей.

Имитационное моделирование имеет преимущества перед аналитическим моделированием в тех случаях, когда отношения между переменными в модели нелинейны или аналитические модели сложно или невозможно построить. *Имитационная* (компьютерная) модель может содержать стохастические компоненты или набор параллельно функционирующих сложным образом компонент, поэтому для понимания поведения системы требуется *визуализация* динамики происходящих в ней процессов.

Этапы имитационного моделирования

Имитационное моделирование состоит из двух основных этапов: создания модели и анализа построенной модели с целью принятия решения.

Сначала для новой модели необходимо определить, какие задачи будут решаться с ее помощью, т.е. моделированию должна предшествовать *формулировка цели моделирования*. От цели зависит то, какие процессы в реальной системе следует выделить и отразить в модели, а от каких абстрагироваться, какие характеристики этих процессов учитывать, а какие – нет, какие соотношения между переменными и параметрами должны быть отражены в *модели* (табл. 7.1).

Следующий этап можно охарактеризовать как создание *концептуальной* модели. На этапе происходит разработка *структуры* модели, т.е. выделение отдельных подсистем, определение элементарных компонентов и их связей на каждом уровне иерархии. В *имитационном моделировании* структура модели отражает структуру реального объекта моделирования на некотором уровне абстракции, а связи между компонентами – реальные связи.

Элементы системы, их связи, параметры и переменные, а также их соотношения и законы их изменения должны быть выражены средствами *среды моделирования*, т.е. в этой среде должны быть определены переменные и параметры *модели*, построены процедуры вычисления изменения переменных и характеристик *модели* во времени.

Для лучшего понимания процессов, протекающих в модели, должно быть разработано их *наглядно-графическое* представление.

Затем построенная модель должна быть *проверена* с точки зрения *корректности* ее реализации.

Следующий этап – *идентификация* модели, т.е. сбор данных и проведение измерений тех характеристик реальной системы, которые должны быть введены в модель в виде значений параметров и распределений случайных величин. В этом случае используется также термин *калибровка* модели.

На следующем этапе необходимо выполнить *проверку правильности* модели (*валидацию*), которая состоит в том, что выход модели проверяется при нескольких тестовых режимах, в которых характеристики поведения реальной системы известны либо очевидны.

На заключительном этапе работы с моделью проводится *компьютерный эксперимент*. При этом в простейших случаях осуществляется запуск на выполнение программы, реализующей модель на ЭВМ при различных значениях ее существенных

параметров (факторов), и наблюдение ее поведения с регистрацией характеристик поведения.

Такой способ использования модели называется *экспериментом* типа *«что будет, если...»*. Компьютерное моделирование позволяет не только получить *прогноз*, но и определить, какие управляющие воздействия на систему приведут к благоприятному развитию событий.

Таблица 7.1

Этапы имитационного моделирования

№	Наименование этапа	Основные результаты
1	Анализ системы	Анализ того, что происходит в системе, идентификация ее структуры, определение процессов, происходящих в системе
2	Постановка цели и задач моделирования системы	Формирование списка задач, которые предполагается решить. Определяется список входных и выходных параметров модели, список исходных данных, критерии завершенности исследования
3	Разработка концептуальной структуры модели	Структура модели, состав существенных процессов, подлежащих отображению в модели, зафиксированный уровень абстракции для каждой подсистемы модели (список допущений), описание управляющей логики для подсистем
4	Реализация модели в среде моделирования	Построение подсистем, выбор параметров, описание их поведения, реализация логики и связи
5	Реализация динамического представления модели	Динамическое представление модели, интерфейс пользователя
6	Проверка корректности реализации модели	Проверка (тестирование) того, что модель корректно отражает те процессы реальной системы, которые требуется анализировать
7	Идентификация модели	Фиксация значений параметров, коэффициентов уравнений и распределений случайных величин, отражающих те ситуации, для анализа которых модель будет использоваться
8	Планирование и проведение компьютерного эксперимента	План экспериментальных работ. Результаты моделирования: графики, таблицы и т.д., дающие ответы на поставленные вопросы

Дальнейшие исследования с имитационными моделями приводят к более сложным *компьютерным экспериментам*, в которых выполняется анализ *чувствительности модели* и проводится *оценка рисков* различных вариантов управляющих решений. Вместе с этими процедурами, как правило, проводится оптимизация параметров и условий функционирования модели.

Интерпретация результатов моделирования осуществляется на заключительном этапе. Для этих целей могут привлекаться дополнительные инструменты (пакеты): обработка статистических расчетов для наглядного (визуализации) представления данных, программное обеспечение презентаций и т.д.

Имитационная модель может использоваться в качестве модуля системы поддержки принятия решения (СППР), которая получает данные мониторинга состояния управляемой системы. С помощью модели производится оценка последствий, к которым может привести текущая ситуация. На основании полученных результатов проведенного моделирования предлагается оптимальное управляющее решение для минимизации отрицательных последствий.

Обзор парадигм моделирования

В настоящее время иммитационное моделирование развивается в рамках следующих основных концепций: системной динамики, дискретно-событийного моделирования, агентного моделирования, динамических систем и объединенного подхода.

1. Системная динамика.

Подход был разработан и представлен Дж. Форрестером в 1961 г. в монографии «Индустриальная динамика». В работе представлено исследование сложных систем с помощью информационных обратных связей с целью демонстрации того, как структура системы изменяется в процессах управления и влияет на успешность предприятия. Используется для моделирования непрерывных процессов с обратными связями, где акцент ставится на агрегированных показателях (уровень запасов, поток заявок).

Примеры: эпидемия, динамика популяции, износ оборудования.

Системная динамика (англ. System Dynamics, SD) – методология и набор инструментов для *анализа и моделирования сложных динамических систем*, которые изменяются во времени. С использованием моделей системной динамики удается показать, как *структура системы* (элементы, связи, петли обратной связи) порождают её *поведение* (траектории состояний, циклы, кризисы, рост/упадок).

Основные свойства:

1. Нелинейность и сложность. Системы часто ведут себя сложно из-за: петель обратной связи (положительных – усиливающих, отрицательных – стабилизирующих); временных задержек между причиной и следствием; взаимозависимости компонентов.

2. Структура определяет поведение системы и описывает, как соединены элементы (потоки ресурсов, информации, правила принятия решений). Изменяя структуру, можно менять поведение.

Структурная модель отвечает на вопрос *«Из чего состоит?»*. Динамическая модель отвечает на вопросы *«Как будет меняться со временем?»*, *«Что будет, если...?»*.

Переход от структуры к поведению – ключевая идея системного анализа: структура системы (как она устроена) определяет её поведение (как она функционирует и эволюционирует во времени). Разберём механизм этого перехода поэтапно.

Структура – совокупность элементов системы, их связей и иерархии. Это *«скелет»* системы: элементы (подсистемы, компоненты); отношения и связи между ними (материальные, информационные, управляющие); архитектура и иерархия (кто кому подчинён, какие потоки данных/ресурсов).

Поведение – динамические проявления системы во времени: траектории состояний (как меняются переменные системы); реакции на внешние воздействия (вход → выход); циклы, обратные связи, устойчивые режимы, кризисы, адаптация.

Переход от *структуры* к *поведению* осуществляется через *механизмы взаимодействия* элементов:

2.1. Связи и потоки. Структура задаёт, *кто с кем взаимодействует* и *по каким каналам* (материальные, энергетические, информационные). Эти потоки во времени формируют поведение.

2.2. Обратные связи (англ. feedback loops). Структура определяет, есть ли петли обратной связи (положительные – усиливающие, отрицательные – стабилизирующие). Они порождают типичные поведенческие паттерны: рост, колебания, равновесие.

2.3. Иерархия и делегирование. Уровни управления в структуре задают, *кто принимает решения* и *как быстро* система реагирует.

3. Эмерджентность. Поведение системы – не сумма поведения её частей. Возникают новые качества, которые невозможно предсказать при изолированном анализе компонентов.

Основные компоненты моделей системной динамики (SD):

− *Уровни* (англ. Stock, запасы) – накопители чего-либо (деньги, запасы, численность, знания). Описываются дифференциальными уравнениями.

Пример: объём воды в резервуаре, количество клиентов, капитал компании.

− *Потоки* (англ. Flow, потоки) – скорости изменения уровней (приток/отток).

Пример: продажи (отток клиентов), инвестиции (приток капитала).

− *Петли обратной связи* (англ. Feedback loops) – замкнутые цепочки причинно-следственных связей.

− *Пример:* чем больше клиентов, тем больше рекомендаций → ещё больше клиентов (положительная петля).

− *Временные задержки* (англ. Delays) – запаздывание реакции системы. *Пример.* Эффект от рекламы проявляется через месяцы; обучение персонала даёт результат через квартал.

− *Вспомогательные переменные* (англ. Auxiliaries) – промежуточные расчёты, правила, коэффициенты.

Пример: коэффициент конверсии, ставка дисконтирования.

Этапы построения модели системной динамики представлены в табл. 7.2.

Программные инструменты:

1. PySD (библиотека Python), Vensim, Stella, AnyLogic.
2. Графики и анимации – визуализация динамики уровней и потоков.

Таблица 7.2

Этапы имитационного моделирования

№	Наименование этапа	Основные результаты
1	Концептуальная фаза	Выявление проблемы и границ системы. Построение *каузальных диаграмм* (CLD) – графиков причинно-следственных связей и петель обратной связи
2	Формализация	Перевод CLD в *диаграммы потоков и уровней SFD* (англ. Stock & Flow). Задание уравнений для потоков, уровней и задержек
3	Моделирование	Численное решение уравнений во времени (например, методом Эйлера). Анализ траекторий: рост, колебания, равновесие, коллапс
4	Сценарный анализ	«Что, если?..» – тестирование политик (изменение параметров, структурные реформы)
5	Оценка устойчивости и чувствительности модели	Интерпретация результатов, построение таблиц и графиков исследуемых параметров, анализ поведения

Области применения системной динамики.

В области экономики и бизнеса методы *системной динамики* позволяют проводить моделирование рынков, формировать цепочки поставок, находить стратегии роста и эффективного поведения.

В сфере экологии методы позволяют моделировать динамику популяций, углеродный след, управление ресурсами, а в социальной сфере – распространение инноваций, эпидемий, миграцию и т.д.

В государственном управлении при моделировании может осуществляться поиск оптимальных сценариев и политик налогообложения, поведения хозяйственных субъектов и т.д., кроме того, модели системной динамики применяются в здравоохранении, транспорте, инженерии и т.д.

Системная динамика позволяет увидеть долгосрочные последствия краткосрочных решений; протестировать политики до их внедрения; объяснять, почему «очевидные» меры иногда дают обратный эффект (из-за скрытых петель и задержек). Поэтому говорят, что *системная динамика* – «лаборатория для экспериментов со временем», где можно проигрывать сценарии и находить структурные источники проблем.

2. Дискретно-событийное моделирование.

Дискретно-событийное моделирование (англ. Discrete-Event Simulation, DES) – вид имитационного моделирования, в котором функционирование системы представляется как хронологическая последовательность *событий*. Каждое событие происходит в определенный момент времени и демонстрирует собой *изменение состояния системы*.

Дискретно-событийное моделирование представляет собой мощный инструмент системного анализа для изучения и оптимизации систем с дискретными событиями и очередями. Оно позволяет прогнозировать поведение процессов, выявлять «узкие места (бутылочные горлышки)» и тестировать управленческие решения в виртуальной среде до их внедрения.

Пример: call-центр, работа склада, логистическая цепочка.

В отличие от системной динамики (где время течёт непрерывно), в DES модель «перескакивает» от одного события к другому, пропуская интервалы, когда ничего не меняется.

Основные понятия модели:

1. *Событие* – мгновенное изменение состояния системы (например, поступление заказа, завершение обслуживания клиента, поломка станка).

2. *Состояние системы* – набор переменных, описывающих систему в любой момент времени (например, количество клиентов в очереди, занятость ресурсов, запасы на складе).

3. *Время моделирования* – виртуальные «часы», синхронизирующие наступление событий.

4. *Очередь* – накопитель заявок, ожидающих обработки (например, клиенты у кассы, детали перед станком).

5. *Ресурс* – ограниченный актив, необходимый для выполнения операций (например, сотрудники, станки, транспорт).

Основные компоненты системы DES:

1. *Часы моделирования* – отслеживают текущее виртуальное время и синхронизируют события.

2. *Список событий* – упорядоченная по времени очередь предстоящих событий.

3. *Генераторы случайных чисел* – нужны для стохастических моделей (например, чтобы смоделировать случайное время обслуживания).

4. *Статистика* – сбор данных о работе системы: средняя занятость ресурсов, среднее количество заявок в очереди, среднее время ожидания, пропускная способность.

5. *Условие завершения* – критерий остановки моделирования: достижение заданного времени, накопление определённого числа событий, выполнение целевого показателя (например, время ожидания превысило лимит).

Типы DES-моделей:

− *Детерминированные* – все параметры фиксированы (например, время обслуживания, интервалы поступления заявок).

− *Стохастические* – ключевые параметры задаются вероятностными распределениями (например, время обслуживания – экспоненциальное распределение). Отличаются от методов Монте-Карло наличием механизма времени (часов).

Этапы работы дискретно-событийной модели представлены в табл. 7.3.

Примеры событий в разных системах:

− Логистика: прибытие грузовика на склад, разгрузка, отправка на маршрут.

− Здравоохранение: поступление пациента, начало приема, выписка.

− Производство: запуск детали на станок, завершение обработки, передача на следующий участок.

− IT-системы: запрос к серверу, обработка, возврат ответа.

Таблица 7.3

Этапы работы дискретно-событийной модели (DES)

№	Наименование этапа	Основные результаты
1	Инициализация	Установка начального состояния и времени $t = 0$
2	Выбор ближайшего события из списка событий	Событие из списка
3	Продвижение часов до времени этого события	$t \rightarrow t^*$ события
4	Выполнение действий, связанных с событием	Обновление состояния, генерация новых событий
5	Сбор статистики	Наборы данных, статистические показатели
6	Повторение	Повторение шагов 2–5 до выполнения условия завершения

Область применения. Анализ и оптимизация бизнес-процессов, проектирование цепочек поставок и складов, планирование ресурсов (персонал, оборудование), моделирование транспортных потоков и пассажиропотоков, оценка производительности компьютерных сетей и серверов, управление очередями в банках, call-центрах, аэропортах.

Инструменты реализации. Специализированные среды и библиотеки: SimPy (библиотека Python), Arena, AnyLogic, SIMSCRIPT, SLAM, GPSS, AweSim.

3. Агентное моделирование.

Основная идея моделирования заключается в том, что модель представляется в виде множества отдельных активных объектов – *агентов*, каждый из которых взаимодействует с другими агентами, образующими для него внешнюю среду. *Агентное моделирование* появилось в 1990-х гг. и используется для исследования децентрализованных систем, динамика функционирования которых определяется не глобальными правилами и законами, а когда эти глобальные правила и законы являются результатом индивидуальной активности членов группы. Цель агентных моделей – получить представление об этих глобальных правилах, общем поведении системы, исходя из предположений об индивидуальном, частном поведении ее отдельных активных объектов и взаимодействии этих объектов в системе. *Агент* – некая сущность, обладающая активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, взаимодействовать с окружением, а также самостоятельно изменяться.

Агентное моделирование (англ. Agent-Based Modeling, ABM) – метод имитационного моделирования, исследующий поведение *децентрализованных агентов* и то, как их действия определяют поведение системы в целом. Ключевая особенность *ABM*: моделирование «снизу вверх», в отличие от системной динамики, где анализируется глобальное поведение:

− исследователь задает правила поведения агентов на индивидуальном уровне;

− макроскопические закономерности возникают как *эмерджентный результат* взаимодействия множества агентов.

Основные принципы агентного моделирования:

1. *Объектная ориентированность* – система представляется как совокупность взаимодействующих объектов – агентов.

2. *Обучаемость / эволюция агентов* – агенты могут адаптироваться во времени (к внешней среде), менять поведение на основе опыта.

3. *Сложность вычислений* – простые правила поведения агентов могут приводить к сложному системному поведению.

Каждый агент в модели обладает следующим набором характеристик:

− *автономность* (англ. Autonomy) – самостоятельно управляет своими действиями;

− *социальность* (англ. Sociality) – взаимодействует с другими агентами и средой;

− *реактивность* (англ. Reactivity) – отвечает на изменения локальной среды;

− *условность* (англ. Conditionality) – имеет состояния, определяющие поведение;

− *модульность* (англ. Modularity) – имеет четкие границы;

− *проактивность* (англ. Pro-Activity) – может быть целенаправленным, адаптироваться, самообучаться.

Архитектура модели:

1. *Агенты* – моделируют сущности: людей, транспортные средства, организации, проекты и т.д.

2. *Среда* – определяет условия существования и взаимодействия агентов, может быть:

− дискретной (двумерный массив);

− непрерывной (евклидово пространство);

− графовой (статической или динамической);

− географической (ГИС);

− «супом» (soup), где положение не критично.

3. *Правила взаимодействия* – определяют, как агенты реагируют на состояния среды и других агентов.

Этапы разработки модели представлены в табл. 7.4.

Таблица 7.4

Этапы разработки модели (ABM)

№	Наименование этапа	Основные результаты
1	Постановка задачи	Определение целей, вопросов, критериев адекватности
2	Анализ системы	Выделение ключевых элементов, границ, связей
3	Концептуальная модель	Качественное описание системы, выбор существенных факторов
4	Формализация	Перевод на формальный язык (математический, алгоритмический
5	Реализация	Программирование модели в специализированном ПО
6	Валидация	Проверка соответствия модели реальной системе
7	Экспериментирование	Анализ поведения системы при разных условиях

Преимущества и ограничения ABM. Позволяет моделировать системы, естественно состоящие из взаимодействующих агентов, возможна разработка модели без знания глобального поведения системы, легко вносить локальные уточнения без глобальных изменений, помогает изучать эмерджентные свойства системы, поддерживает адаптацию и самообучение агентов.

Отсутствие единого определения агента (кроме автономности), сложность сбора статистики по индивидуальным объектам, трудности валидации модели на реальных данных, высокие вычислительные затраты при большом числе агентов.

Применения ABM. ABM используется для исследования социальных систем (распространение инноваций, поведение толпы); экономических процессов (потребительское поведение, рынки); логистических сетей (оптимизация поставок); экологических систем (взаимодействия видов); организационных структур (менеджмент ресурсов); транспортных систем (управление потоками).

Инструменты реализации. Популярные платформы для ABM: StarLogo (ориентирован на образование), Python, Swarm – поддержка Objective C и Java, MASON – Java-фреймворк для масштабных симуляций, Repast (Java-библиотека для сложных моделей), AnyLogic (универсальная среда с графическим интерфейсом).

Компьютерный эксперимент в имитационном моделировании

Исследование сложных систем посредством имитационного моделирования предполагает организацию и проведение *компьютерного эксперимента* на основе специально разработанной имитационной модели. Данный подход занимает промежуточное положение между натурным экспериментом с реальным объектом и умозрительным (виртуальным) экспериментом, сочетая достоинства обоих методов и минимизируя их ограничения.

Натурный эксперимент, несмотря на потенциальную высокую точность получаемых результатов, нередко характеризуется значительной ресурсоёмкостью, финансовыми затратами и т.д. В ряде случаев его проведение принципиально невозможно, так например, при исследовании проектируемых систем, ещё не реализованных в материальной форме, или в случае наличия технологических, физических или других барьеров.

Имитационная модель, напротив, позволяет воспроизводить сценарии, которые трудно или нецелесообразно реализовать в рамках натурного эксперимента. Многократное выполнение модели с постепенным усложнением её структуры даёт возможность детально анализировать тонкие эффекты, которые могут оставаться незамеченными при наблюдении реального процесса. Кроме того, варьирование масштаба времени в ходе компьютерного эксперимента существенно расширяет объем получаемой информации и углубляет понимание динамики системы.

В свою очередь, умозрительный эксперимент, основанный на интуитивных предположениях и апелляции к «здравому смыслу», представляет собой субъективный подход к решению проблем, не учитывающий комплексную природу сложных систем.

В условиях высокой степени взаимозависимости элементов и нелинейности процессов интуитивно очевидные решения нередко приводят к ошибочным выводам и неэффективным управленческим решениям.

Традиционные методы прогнозирования, опирающиеся на субъективную оценку и волюнтаристские принципы принятия решений, демонстрируют ограниченную применимость в отношении сложных динамических систем. Их недостаточность обусловлена: игнорированием скрытых взаимосвязей между компонентами системы; неспособностью адекватно учитывать обратные связи и эффекты запаздывания; отсутствием количественной оценки рисков и альтернативных сценариев.

Применение методов имитационного моделирования и проведение компьютерных экспериментов позволяют преодолеть вышеуказанные ограничения. Формализованное представление системы, возможность варьирования параметров и многократного воспроизведения сценариев обеспечивают более глубокий и объективный анализ. В результате данный методологический подход демонстрирует высокую эффективность и получает всё более широкое признание в современной науке и практике управления сложными системами.

Прямая и обратная задачи имитационного моделирования. В имитационном моделировании различают прямые и обратные задачи. Прямая задача предполагает выявление влияния выделенных факторов на выходные переменные модели в ходе компьютерного эксперимента.

Факторы – параметры модели, вариация которых анализируется; их совокупность задаётся вектором $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ длины n, где $\boldsymbol{x} \in X$, а X – множество всех возможных наборов факторов. В число факторов могут включаться и структурные характеристики модели, измеренные в различных шкалах.

Компьютерный эксперимент состоит в многократном запуске модели при различных значениях $\boldsymbol{x}$. Каждый прогон даёт вектор исходов $\boldsymbol{y}$ (например, прогноз плотности населения через 20 лет или коэффициенты загрузки оборудования). Таким образом, прямая задача отвечает на вопрос: *«Каковы последствия реализации решения* $\boldsymbol{x}_k \in X$ *в заданных условиях?»* или *«Что будет, если принять решение* $\boldsymbol{x}_k$*?»*

Прямая задача имитационного моделирования (задача типа *what-if,* или «что-если») заключается в анализе влияния заданных параметров системы на выходные показатели.

Формально она определяется следующим образом:

1) вектор входных параметров: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in X$, где X – множество возможных наборов факторов;

2) вектор выходных показателей: $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in Y$, где Y – множество возможных результатов.

В *детерминированном случае* модель задает однозначное функциональное отображение $\boldsymbol{x} \rightarrow \boldsymbol{y}$. Каждый прогон модели для конкретного $\boldsymbol{x}_k \in X$ даёт соответствующий вектор $\boldsymbol{y}_k \in Y$, который может зависеть от времени.

Инструмент имитационного моделирования должен обеспечивать удобный интерфейс задания входных параметров $\boldsymbol{x}$, регистрацию выходных показателей $\boldsymbol{y}$ и их динамики во времени.

Анализ чувствительности – методологический инструмент имитационного моделирования, предназначенный для исследования зависимости выходных показателей $\boldsymbol{y} \in Y$ от входных факторов $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in X$ и базовых гипотез модели.

Алгоритм проведения.

1. Для каждого фактора x_i выполняется его варьирование в заданном диапазоне при фиксированных значениях остальных параметров.

2. Регистрируется реакция модели – изменение вектора выходных показателей $\Delta\boldsymbol{y}$.

3. На основе полученных данных рассчитывается мера чувствительности (например, коэффициент эластичности или градиент функции отклика).

Итоговые результаты позволяют:

– количественно оценить устойчивость прогноза к изменениям исходных предположений;

– ранжировать факторы x_i по силе влияния на $\boldsymbol{y}$;

– идентифицировать критические параметры, требующие повышенного внимания при калибровке модели.

Обратная задача имитационного моделирования заключается в поиске вектора входных оптимальных параметров $\boldsymbol{x}_{\text{опт}}$, *максимизирующего* показатель эффективности системы.

Формальная постановка:

1) задан вектор входных факторов $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, где X – множество допустимых решений;

2) вектор выходных показателей $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in Y$ определяется имитационной моделью;

3) показатель эффективности $w \in W$ вычисляется с помощью целевой функции (функции полезности) Φ: $Y \rightarrow W$, т.е. $w = \Phi(\boldsymbol{y})$.

Цель *обратной задачи*: найти вектор $\boldsymbol{x}_{\text{опт}} \in X$ такой, что:

$$\Phi(\boldsymbol{y}(\boldsymbol{x}_{опт})) = \max_{\boldsymbol{x} \in X} \Phi(\boldsymbol{y}(\boldsymbol{x})).$$

Решение обратной задачи обычно сводится к многократному решению прямой задачи. При небольшом числе вариантов применяется полный перебор вариантов, в сложных случаях – методы направленного перебора с использованием эвристик, обеспечивающих приближение к оптимальному решению на последовательных итерациях.

Система обработки заказов в стартапе по доставке еды

Шаг 1. Определение ключевых переменных.

– Запасы: Что накапливается в системе? (Количество необработанных заказов, количество свободных курьеров, текущий рейтинг службы).

− Потоки: Что изменяет запасы? (Интенсивность поступления заказов, скорость обработки, скорость доставки).

− Параметры и константы: Среднее время обработки, вероятность ошибки, количество операторов.

Шаг 2. Построение диаграммы причинно-следственных связей (англ. Causal Loop Diagram, CLD).

Цель: Выявить *обратные связи*, определяющие поведение системы.

Для системы «Обработка заказов» построить CLD.

На рис. 7.1, 7.2 показаны фрагменты причинно-следственных связей в форме CLD.

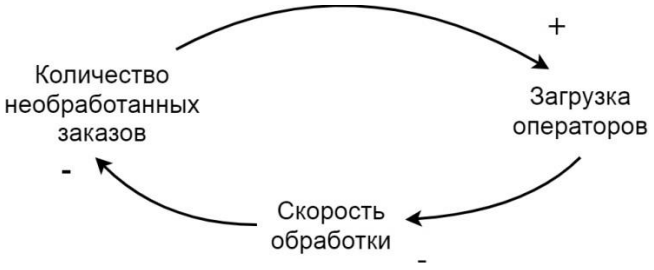


Рис. 7.1. Отрицательная обратная связь (стабилизирующая)

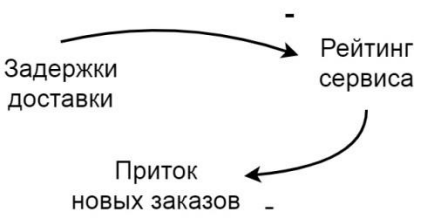


Рис. 7.2. Фрагмент связи

Шаг 3. Переход к диаграмме потоков и запасов (SFD).

Цель: Превратить качественную CLD в количественную модель, готовую для симуляции.

На основе CLD построить SFD. Четко обозначить запасы (прямоугольники), потоки (стрелки с «клапаном»), конвертеры (круги) и связи.

Инструмент: Vensim PLE (бесплатный), Stella, AnyLogic.

Имитационное моделирование

Дискретно-событийная модель процесса (DES).

Цель: Оценить операционную эффективность.

Сценарий: Смоделировать процесс от поступления заказа до передачи курьеру.

− Сущности: Заказы.

− Стадии/Процессы: Прием → Проверка → Сборка → Передача курьеру.

− Ресурсы: Операторы (ограниченное количество), сборщики.

− Стохастика: Время на каждой стадии задано распределением (например, экспоненциальное со средним 2 мин).

Вопросы для анализа

1. При каком среднем интервале между заказами (например, 1 заказ/мин) начинает расти очередь?

2. Какая стадия является «узким местом»?

3. Как добавление одного оператора на стадию «Проверка» повлияет на общее время обработки и использование ресурсов?

Инструмент для моделирования: SimPy (Python).
Пример кода модели обработки заказов приведен на листинге 7.1.

Листинг 7.1

Модель обработки заказов

```
import simpy
import random
import statistics
import pandas as pd
import matplotlib.pyplot as plt

RANDOM_SEED = 42
NUM_OPERATORS = 2
SIM_TIME = 200
MEAN_INTERARRIVAL = 1.0

random.seed(RANDOM_SEED)


class OrderProcessing:
    def __init__(self, env, num_operators):
        self.env = env
        self.operator = simpy.Resource(env, num_operators)

        self.processing_times = []
        self.wait_times = []
        self.resource_usage = []

    def process_order(self, order_id):
        arrival_time = self.env.now

        # Захват ресурса "Оператор"
        with self.operator.request() as request:
            yield request

            wait_time = self.env.now - arrival_time
            self.wait_times.append(wait_time)

            # Стохастическое время обработки
            service_time = random.expovariate(1.0 / 2.0)

            # фиксируем загрузку ресурса
            self.resource_usage.append((self.env.now, self.operator.count))

            yield self.env.timeout(service_time)

        total_time = self.env.now - arrival_time
        self.processing_times.append(total_time)


# генератор заказов
def order_generator(env, system):
    order_id = 0

    while True:
        yield env.timeout(random.expovariate(1.0 / MEAN_INTERARRIVAL))
        order_id += 1
        env.process(system.process_order(order_id))
```

```
# запуск симуляции
env = simpy.Environment()
system = OrderProcessing(env, NUM_OPERATORS)

env.process(order_generator(env, system))
env.run(until=SIM_TIME)

# статистика
print("Количество заказов:", len(system.processing_times))
print("Среднее время обработки:", statistics.mean(system.pro-
cessing_times))
print("Среднее время ожидания:", statistics.mean(system.wait_times))
```

Графики загрузки ресурсов, гистограмма времени обработки, выводы по эффективности.

Верификация и валидация.

Верификация (Правильно ли мы построили модель?). Проверка единиц измерения, проверка на здравый смысл (при нулевом притоке клиентская база стремится к нулю?).

Валидация (Та ли это модель?). Сравнение поведения модели с известными паттернами (логистический рост, затухающие колебания).

Задания к работе

1. Моделирование нагрузки на веб-сервер. Дискретно-событийное моделирование.

Компания разрабатывает новый веб-сервис. Необходимо оценить, как будет вести себя сервер при различной интенсивности запросов, чтобы определить необходимое количество ресурсов (процессоров, памяти) и настроить балансировщик нагрузки.

Исходные данные:

1) Запросы приходят случайным образом. Время между запросами распределено экспоненциально со средним значением λ (запросов в секунду).

2) Сервер обрабатывает каждый запрос. Время обработки зависит от типа запроса и текущей нагрузки. Для простоты предположим, что время обработки распределено по нормальному закону со средним μ и стандартным отклонением σ.

3) У сервера есть очередь неограниченной длины (или ограниченной, что тоже можно варьировать).

4) Можно добавить несколько серверов (мультипроцессинг) или несколько ядер.

1. Разработка концептуальной модели:

1.1. Определить элементы системы: источник запросов, очередь, процессор(ы).

1.2. Определить параметры: интенсивность входного потока, распределение времени обслуживания, количество процессоров.

1.3. Определить метрики, которые будут измерять: среднее время отклика, загрузку процессора, среднюю длину очереди, вероятность отказа (если очередь ограничена).

2. Построение имитационной модели на SimPy:

2.1. Реализовать модель одноканальной системы массового обслуживания СМО с очередью (*M*/*M*/*1*) и многоканальной (*M*/*M*/*c*).

2.2. Провести серию экспериментов, изменяя параметры λ и μ. Построить графики зависимости метрик от коэффициента загрузки $\rho = \frac{\lambda}{c\mu}$, где *c* – количество каналов.

2.3. Исследовать, как ведет себя система при ρ > 1.

3. Анализ результатов и поиск «узких мест»:

3.1. Определить, при какой интенсивности входного потока система перестает справляться (например, время отклика превышает допустимое значение).

3.2. Предложить варианты модернизации системы (увеличить количество процессоров, оптимизировать код для уменьшения μ, ввести приоритеты запросов). Оценить эффект от этих изменений с помощью модели.

Пример кода модели M/M/1 на SimPy приведен на листинге 7.2.

Листинг 7.2

Модель M/M/1 на SimPy

```
import simpy
import random
import statistics
import matplotlib.pyplot as plt

class WebServer:
    def __init__(self, env, service_rate):
        self.env = env
        self.server = simpy.Resource(env, capacity=1)
        self.service_rate = service_rate
        self.response_times = []
        self.queue_lengths = []
        self.utilization = 0

    def process_request(self, request_id):
        arrival_time = self.env.now
        with self.server.request() as req:
            yield req
            wait_time = self.env.now - arrival_time
            service_time = random.expovariate(self.service_rate)
            yield self.env.timeout(service_time)
            total_time = self.env.now - arrival_time
            self.response_times.append(total_time)

def request_generator(env, server, arrival_rate):
    request_id = 0
    while True:
        yield env.timeout(random.expovariate(arrival_rate))
        env.process(server.process_request(request_id))
        request_id += 1

# Запуск симуляции
env = simpy.Environment()
server = WebServer(env, service_rate=1.0)  # среднее время обслуживания
1 сек
```

```
env.process(request_generator(env, server, arrival_rate=0.8))  # интенсив-
ность 0.8 запроса/сек
env.run(until=1000)

# Анализ результатов
print(f"Среднее время отклика: {statistics.mean(server.re-
sponse_times):.2f}")
print(f"Максимальное время отклика: {max(server.response_times):.2f}")
```

2. Динамика популяции с учетом ограничений ресурсов. Системная динамика.

Эколог изучает популяцию животных на изолированном острове. Популяция растет, но ограничена доступными ресурсами (пищей, территорией), также возможно влияние хищников или болезней. Необходимо смоделировать изменение численности популяции во времени и оценить влияние различных факторов.

Описание модели:

1) Базовая модель: логистическое уравнение роста

$$\frac{dN}{dt} = rN\left(1-\frac{N}{K}\right),$$

где N – численность; r – скорость роста; K – ёмкость среды.

Можно усложнить модель, добавив влияние хищников: уравнение Лотки–Вольтерры; сезонные колебания ресурсов, случайные события (эпидемии, засухи).

Задание.

1. Разработка концептуальной модели:

1.1. Построить диаграмму причинно-следственных связей (CLD) для системы. Определить основные переменные (запасы, потоки, параметры) и обратные связи.

1.2. Перейти к диаграмме потоков и запасов (SFD).

2. Построение модели системной динамики:

2.1. Реализовать базовую логистическую модель с помощью дифференциальных уравнений (можно использовать метод Эйлера или готовые решатели в Python).

2.2. Провести эксперименты, изменяя параметры r и K. Покажите, как от них зависит динамика.

2.3. Добавить в модель хищников (система из двух дифференциальных уравнений). Исследовать стабильность системы, возникновение колебаний.

3. Анализ чувствительности и сценариев:

3.1. Исследовать, как влияет начальная численность популяции на долгосрочное поведение.

3.2. Добавить случайные события (например, раз в несколько лет случается засуха, уменьшающая K). Провести множество прогонов (Монте-Карло) и построить доверительные интервалы для траекторий.

3.3. Предложить меры по сохранению популяции (например, искусственное увеличение ресурсов) и смоделировать их эффект.

Пример кода для логистической модели приведен на листинге 7.3.

Листинг 7.3

Логистическая модель

```
import numpy as np
import matplotlib.pyplot as plt

def logistic_growth(N0, r, K, dt, steps):
    N = np.zeros(steps)
    N[0] = N0
    for t in range(1, steps):
        dN = r * N[t-1] * (1 - N[t-1]/K) * dt
        N[t] = N[t-1] + dN
    return N

# Параметры
N0 = 10   # начальная численность
r = 0.1   # скорость роста
K = 1000  # ёмкость среды
dt = 0.1  # шаг по времени
steps = 1000

N = logistic_growth(N0, r, K, dt, steps)

# Визуализация
time = np.arange(0, steps*dt, dt)
plt.plot(time, N)
plt.xlabel('Время')
plt.ylabel('Численность')
plt.title('Логистический рост популяции')
plt.grid()
plt.show()
```

На рис. 7.3. приводится кривая логистического роста популяции.

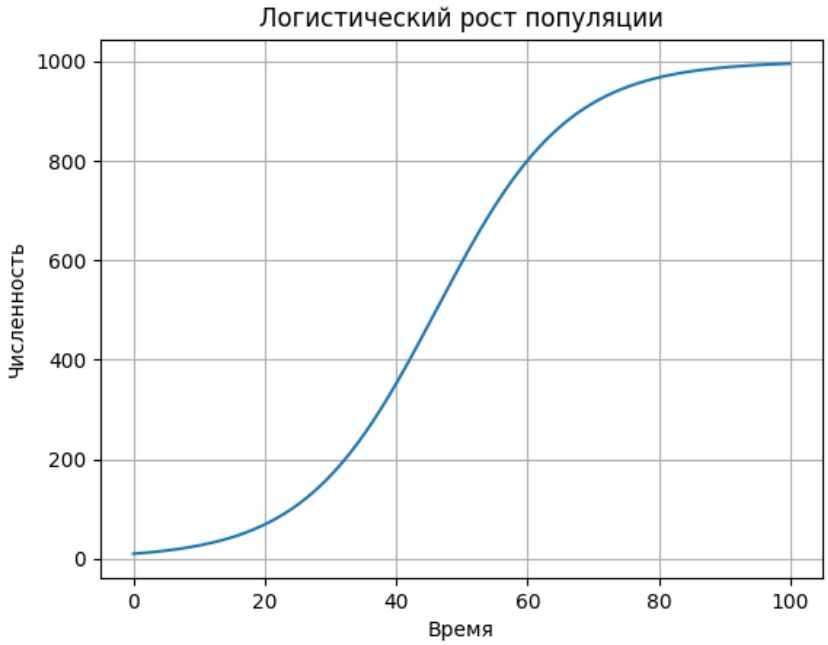


Рис. 7.3. График логистического роста популяции

3. Агентное моделирование.

Пример 3. Распространение информации в социальной сети.

Задание. Исследователи изучают, как распространяется информация (или слух) в социальной сети. Каждый пользователь может быть в одном из трех состояний: не знает информацию (англ. Susceptible), знает и распространяет (англ. Infected), знает, но не распространяет (англ. Recovered). Процесс похож на модель эпидемии SIR, но с учетом структуры социальной сети.

Исходные данные:

1) Социальная сеть задана графом, где вершины – пользователи, ребра – связи.

2) Каждый агент (пользователь) имеет состояние (*S*/*I*/*R*).

3) В каждый момент времени агент, находящийся в состоянии *I*, с вероятностью β передает информацию каждому своему соседу в состоянии *S*. Агент в состоянии *I* с вероятностью γ переходит в состояние *R* (устал распространять).

SIR модель (англ. Susceptible – Infected – Recovered, «восприимчивые» – «инфицированные» – «выздоровевшие») – математическая модель, используемая для описания динамики распространения инфекционных заболеваний в популяции. Модель была предложена в 1920-х гг. шотландскими учёными У.О. Кермаком и А.Г. Маккендриком и стала одной из первых наиболее известных моделей в эпидемиологии.

1. Разработка концептуальной модели:

1.1. Определить состояния агентов, правила перехода между состояниями.

1.2. Определить структуру сети: можно начать с регулярной решетки, случайного графа Эрдёша–Реньи или графа с предпочтительным присоединением (Barabasi–Albert).

1.3. Определить метрики: доля агентов в каждом состоянии во времени, скорость распространения, конечная доля узнавших.

2. Построение агентной модели:

2.1. Реализовать модель SIR на графе с помощью библиотеки Mesa (специализирована для агентного моделирования) или NetworkX.

2.2. Провести эксперименты для разных типов графов и параметров β и γ.

2.3. Исследовать влияние структуры сети на распространение информации. Например, сравнить скорость распространения в случайном графе и в scale-free сети.

3. Анализ и оптимизация:

3.1. Определить, какие агенты (например, с наибольшей степенью центральности) наиболее важны для быстрого распространения информации. Провести эксперименты по «вакцинации» (перевод в состояние *R*) наиболее влиятельных агентов и оценить эффект «вакцинации».

3.2. Предложить стратегию запуска информации (выбор начальных «зараженных» агентов) для максимального охвата.

Пример кода для агентной модели SIR с использованием NetworkX приведен на листинге 7.4.

Листинг 7.4

Агентная модель SIR

```
import networkx as nx
import random
import matplotlib.pyplot as plt
# Создание графа (случайный граф Эрдеша-Реньи)
n = 100  # количество агентов
p = 0.1  # вероятность связи
G = nx.erdos_renyi_graph(n, p)
# Инициализация состояний агентов: 0 - S, 1 - I, 2 - R
states = {node: 0 for node in G.nodes()}
# Выбираем случайно 5 агентов в состоянии I
initial_infected = random.sample(list(G.nodes()), 5)
for node in initial_infected:
    states[node] = 1
# Параметры модели
beta = 0.3  # вероятность передачи
gamma = 0.1  # вероятность выздоровления
steps = 50
# Сбор статистики
S_count = []
I_count = []
R_count = []
# Симуляция
for step in range(steps):
    new_states = states.copy()
    # Обрабатываем всех агентов
    for node in G.nodes():
        if states[node] == 1:  # если агент заражен
            # Передача информации соседям
            for neighbor in G.neighbors(node):
                if states[neighbor] == 0 and random.random() < beta:
                    new_states[neighbor] = 1
            # Выздоровление
            if random.random() < gamma:
                new_states[node] = 2
    states = new_states
    # Сбор статистики
    S_count.append(list(states.values()).count(0))
    I_count.append(list(states.values()).count(1))
    R_count.append(list(states.values()).count(2))
# Визуализация динамики
plt.plot(S_count, label='Восприимчивые')
plt.plot(I_count, label='Зараженные')
plt.plot(R_count, label='Выздоровившие')
plt.xlabel('Шаг')
plt.ylabel('Количество агентов')
plt.legend()
plt.show()
```

На рис. 7.4. приводятся результаты визуализации динамики для параметра «количество агентов».

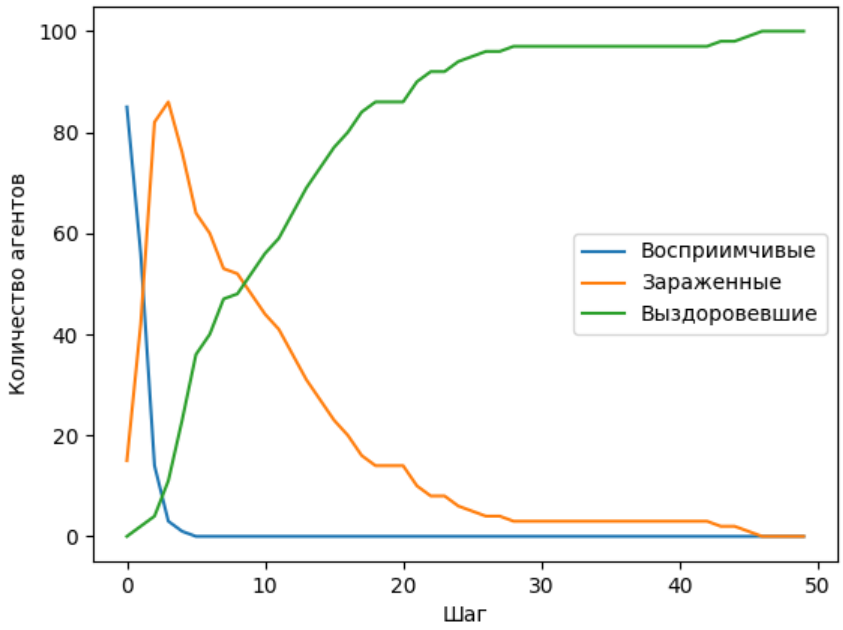


Рис. 7.4. Кривые системной динамики для параметра «количество агентов»

4. Моделирование центра обработки вызовов (call-center).

Пример 1. Дискретно-событийное моделирование центра обработки вызовов (call-центр).

Компания «Телеком-XR» имеет контакт-центр с тремя типами операторов: новички (обрабатывают простые запросы), опытные (запросы средней сложности), эксперты (сложные технические вопросы). Клиенты звонят с разными типами проблем. Необходимо найти оптимальное распределение операторов, чтобы минимизировать среднее время ожидания и процент отказов (когда клиент ждет более 5 мин и бросает трубку).

Исходные данные:

1) Время между звонками: Exp(λ = 2 мин) (экспоненциальное распределение).

2) Распределение типов проблем: простые (40%), средние (40%), сложные (20%).

3) Время обработки:

– Простые: Triangular(2, 5, 10) мин (мин, мода, макс).

– Средние: Triangular(5, 10, 20) мин.

– Сложные: Triangular(15, 25, 40) мин.

4) Количество операторов каждого типа можно менять (в сумме не более 15 человек).

5) Очередь общая, максимальная длина – 20 звонков.

Задание.

1. Построение концептуальной динамической модели.

1.1. Нарисовать причинно-следственные диаграммы (CLD), выявив ключевые обратные связи.

[Длина очереди] → (+) → [Время ожидания] → (+) → [Процент отказов] → (-) → [Эффективная нагрузка].

[Количество операторов] → (-) → [Время ожидания] → (+) → [Затраты на персонал].

1.2. Определить *запасы*: Звонки в очереди, Операторы в разных состояниях (свободны/заняты).

1.3. Определить *потоки*: Приход звонков, Начало обработки, Завершение обработки, Отказы.

2. Реализация дискретно-событийной модели на SimPy.

3. Анализ узких мест и оптимизация.

3.1. Визуализировать загрузку каждого типа операторов.

3.2. Найти оптимальное соотношение операторов методом полного перебора (или градиентного спуска) для минимизации функции:

$$F = t_{avg} + 10\,t_{ab},$$

где t_{avg} – среднее время ожидания, t_{ab} – время отказа.

3.3. Добавить возможность переадресации звонков (если новичок свободен, а звонок средний – может ли он его взять с увеличением времени обработки на 20%?).

Пример кода дискретно-событийной модели с использованием SimPy приведен на листинге 7.5.

Листинг 7.5

Дискретно-событийная модель центра обработки вызовов

```
import simpy
import random
import numpy as np
import pandas as pd
from collections import defaultdict

class CallCenter:
    def __init__(self, env, num_novice, num_experienced, num_expert):
        self.env = env
        # Ресурсы - операторы разных типов
        self.novice = simpy.Resource(env, num_novice)
        self.experienced = simpy.Resource(env, num_experienced)
        self.expert = simpy.Resource(env, num_expert)

        # Статистика
        self.wait_times = []
        self.call_types = {'simple': 0, 'medium': 0, 'hard': 0}
        self.abandoned_calls = 0
        self.total_calls = 0
        self.queue_history = []  # Для отслеживания длины очереди во времени

    def process_call(self, call_id, call_type):
        arrival_time = self.env.now
        self.total_calls += 1
        self.call_types[call_type] += 1

        # Выбор ресурса в зависимости от типа звонка
        if call_type == 'simple':
            resource = self.novice
```

```
            service_time = random.triangular(2, 5, 10)
        elif call_type == 'medium':
            resource = self.experienced
            service_time = random.triangular(5, 10, 20)
        else:  # hard
            resource = self.expert
            service_time = random.triangular(15, 25, 40)

        # Обработка с возможностью отказа
        with resource.request() as request:
            # Клиент ждет максимум 5 минут
            results = yield request | self.env.timeout(5)

            if request in results:
                # Получил оператора
                wait_time = self.env.now - arrival_time
                self.wait_times.append(wait_time)

                # Обслуживание
                yield self.env.timeout(service_time)
            else:
                # Отказ (timeout сработал)
                self.abandoned_calls += 1
                wait_time = 5  # Максимальное время ожидания

def call_generator(env, call_center, mean_interarrival):
    call_id = 0
    while True:
        yield env.timeout(random.expovariate(1.0/mean_interarrival))

        # Определение типа звонка
        r = random.random()
        if r < 0.4:
            call_type = 'simple'
        elif r < 0.8:
            call_type = 'medium'
        else:
            call_type = 'hard'

        env.process(call_center.process_call(call_id, call_type))
        call_id += 1

        # Запись длины очереди (приблизительно)
        queue_len = (len(call_center.novice.queue) +
                     len(call_center.experienced.queue) +
                     len(call_center.expert.queue))
        call_center.queue_history.append((env.now, queue_len))

# Запуск симуляции
def run_simulation(num_novice=5, num_experienced=4, num_expert=3,
                   sim_time=480):  # 8-часовой рабочий день
    env = simpy.Environment()
    cc = CallCenter(env, num_novice, num_experienced, num_expert)
    env.process(call_generator(env, cc, mean_interarrival=2))
    env.run(until=sim_time)

    # Анализ результатов
    avg_wait = np.mean(cc.wait_times) if cc.wait_times else 0
    abandonment_rate = cc.abandoned_calls / cc.total_calls if cc.total_calls else 0

    return {
        'avg_wait': avg_wait,
```

```
        'abandonment_rate': abandonment_rate,
        'total_calls': cc.total_calls,
        'served_calls': len(cc.wait_times),
        'queue_history': cc.queue_history
    }

# Пример запуска
results = run_simulation(5, 4, 3)
print(f"Среднее время ожидания: {results['avg_wait']:.2f} мин")
print(f"Процент отказов: {results['abandonment_rate']*100:.1f}%")
```

5. Динамика эпидемии с вакцинацией. Системная динамика и агентное моделирование.

Моделирование распространения заболевания в городе с разными возрастными группами и стратегиями вакцинации. Особенность – сочетание системной динамики (для агрегированных показателей) и агентного подхода (для индивидуальных контактов).

Исходные данные:

1) Население: 10 000 человек, разделены на три возрастные группы: дети (0–18), взрослые (19–64), пожилые (65+).

2) Параметры заболевания (напоминает COVID-19).

– R_0 = 2,5 (базовое репродуктивное число).

– Инкубационный период: 5 дней.

– Инфекционный период: 10 дней.

3) Вакцинация: доступно несколько стратегий (сначала пожилые, сначала взрослые, случайная).

Задание

1. Построение гибридной модели.

2. Анализ различных стратегий.

2.1. Сравнить три стратегии вакцинации по пиковому количеству больных, общему числу заболевших и времени окончания эпидемии.

2.2. Добавить госпитализацию с разной вероятностью для разных возрастных групп (дети – 1%, взрослые – 5%, пожилые – 20%) и ограниченным количеством коек (50 на 10 000 населения).

2.3. Реализовать адаптивную стратегию: когда свободных коек менее 10%, вводить локдаун (уменьшать R_0 на 50%).

Пример кода гибридной модели приведен на листинге 7.6.

Листинг 7.6

Гибридная модель

```
import numpy as np
import matplotlib.pyplot as plt
from enum import Enum

class HealthState(Enum):
    SUSCEPTIBLE = "S"
    EXPOSED = "E"
```

```
    INFECTIOUS = "I"
    RECOVERED = "R"
    VACCINATED = "V"

class Person:
    def __init__(self, age_group, vaccination_priority):
        self.age_group = age_group  # 'child', 'adult', 'elderly'
        self.state = HealthState.SUSCEPTIBLE
        self.days_in_state = 0
        self.vaccination_priority = vaccination_priority
        self.will_be_vaccinated = False
        self.vaccine_efficacy = 0.0

    def update(self, infection_probability):
        self.days_in_state += 1

        # Логика переходов между состояниями
        if self.state == HealthState.SUSCEPTIBLE:
            if self.will_be_vaccinated and random.random() < 0.01:  # 1%
шанс вакцинации в день
                self.state = HealthState.VACCINATED
                self.vaccine_efficacy = 0.95  # Эффективность вакцины 95%
            elif random.random() < infection_probability:
                self.state = HealthState.EXPOSED
                self.days_in_state = 0

        elif self.state == HealthState.EXPOSED:
            if self.days_in_state >= 5:  # Инкубационный период
                self.state = HealthState.INFECTIOUS
                self.days_in_state = 0

        elif self.state == HealthState.INFECTIOUS:
            if self.days_in_state >= 10:  # Период инфекционности
                self.state = HealthState.RECOVERED
                self.days_in_state = 0

class City:
    def __init__(self, population_size, vaccination_strategy='el-
derly_first'):
        self.population = []
        self.vaccination_strategy = vaccination_strategy
        self.vaccination_rate = 0  # Доза в день
        self.day = 0

        # Инициализация населения
        age_distribution = {'child': 0.2, 'adult': 0.6, 'elderly': 0.2}
        for i in range(population_size):
            age_group = np.random.choice(
                list(age_distribution.keys()),
                p=list(age_distribution.values())
            )

            # Приоритет вакцинации в зависимости от стратегии
            if vaccination_strategy == 'elderly_first':
                priority = 1 if age_group == 'elderly' else (2 if age_group
== 'adult' else 3)
            elif vaccination_strategy == 'adults_first':
                priority = 1 if age_group == 'adult' else (2 if age_group
== 'elderly' else 3)
            else:  # random
                priority = np.random.randint(1, 4)
```

```
            person = Person(age_group, priority)
            self.population.append(person)

        # Заражение первых 10 человек
        for i in range(min(10, population_size)):
            self.population[i].state = HealthState.EXPOSED

    def calculate_infection_probability(self):
        # Динамическое R₀ в зависимости от принятых мер
        base_R0 = 2.5
        current_infectious = sum(1 for p in self.population
                                 if p.state == HealthState.INFECTIOUS)

        # Эффект социального дистанцирования (пример)
        distancing_factor = 1.0
        if current_infectious > 100:
            distancing_factor = 0.7
        # Усиление мер при росте заболеваемости
        daily_prob = (base_R0 * distancing_factor * current_infectious) /
len(self.population) / 10
        return min(daily_prob, 0.3)
    def vaccinate(self, doses_available):
        # Распределение вакцин по стратегии
        candidates = [p for p in self.population
                      if p.state == HealthState.SUSCEPTIBLE and not
p.will_be_vaccinated]
        candidates.sort(key=lambda x: x.vaccination_priority)

        for i in range(min(doses_available, len(candidates))):
            candidates[i].will_be_vaccinated = True
    def simulate_day(self, doses_available=0):
        infection_prob = self.calculate_infection_probability()
                # Вакцинация
        if doses_available > 0:
            self.vaccinate(doses_available)
                # Обновление состояния каждого человека
        for person in self.population:
            person.update(infection_prob)
                    self.day += 1
        # Сбор статистики
        stats = {state.value: 0 for state in HealthState}
        for person in self.population:
            stats[person.state.value] += 1
        return stats

# Симуляция на 180 дней
city = City(10000, vaccination_strategy='elderly_first')
history = []

for day in range(180):
    # Увеличиваем темп вакцинации со временем
    doses = min(100, day) # Начинаем с 0, доходим до 100 доз в день
        stats = city.simulate_day(doses)
    history.append(stats)

    if day % 30 == 0:
        print(f"День {day}: S={stats['S']}, E={stats['E']}, I={stats['I']},
R={stats['R']}, V={stats['V']}")
# Визуализация
history_df = pd.DataFrame(history)
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
```

```
axes[0, 0].plot(history_df['S'], label='Восприимчивые')
axes[0, 0].plot(history_df['V'], label='Вакцинированные')
axes[0, 0].set_title('Восприимчивые и вакцинированные')
axes[0, 0].legend()

axes[0, 1].plot(history_df['I'], 'r-', label='Заразные')
axes[0, 1].set_title('Активные случаи')
axes[0, 1].legend()

axes[1, 0].plot(history_df['E'], 'y-', label='Инкубационные')
axes[1, 0].set_title('Инкубационный период')
axes[1, 0].legend()

axes[1, 1].plot(history_df['R'], 'g-', label='Выздоровевшие')
axes[1, 1].set_title('Иммунная прослойка')
axes[1, 1].legend()

plt.tight_layout()
plt.show()
```

На рис. 7.5. приводятся результаты моделирования распространения заболевания в городе с разными возрастными группами и стратегиями вакцинации.

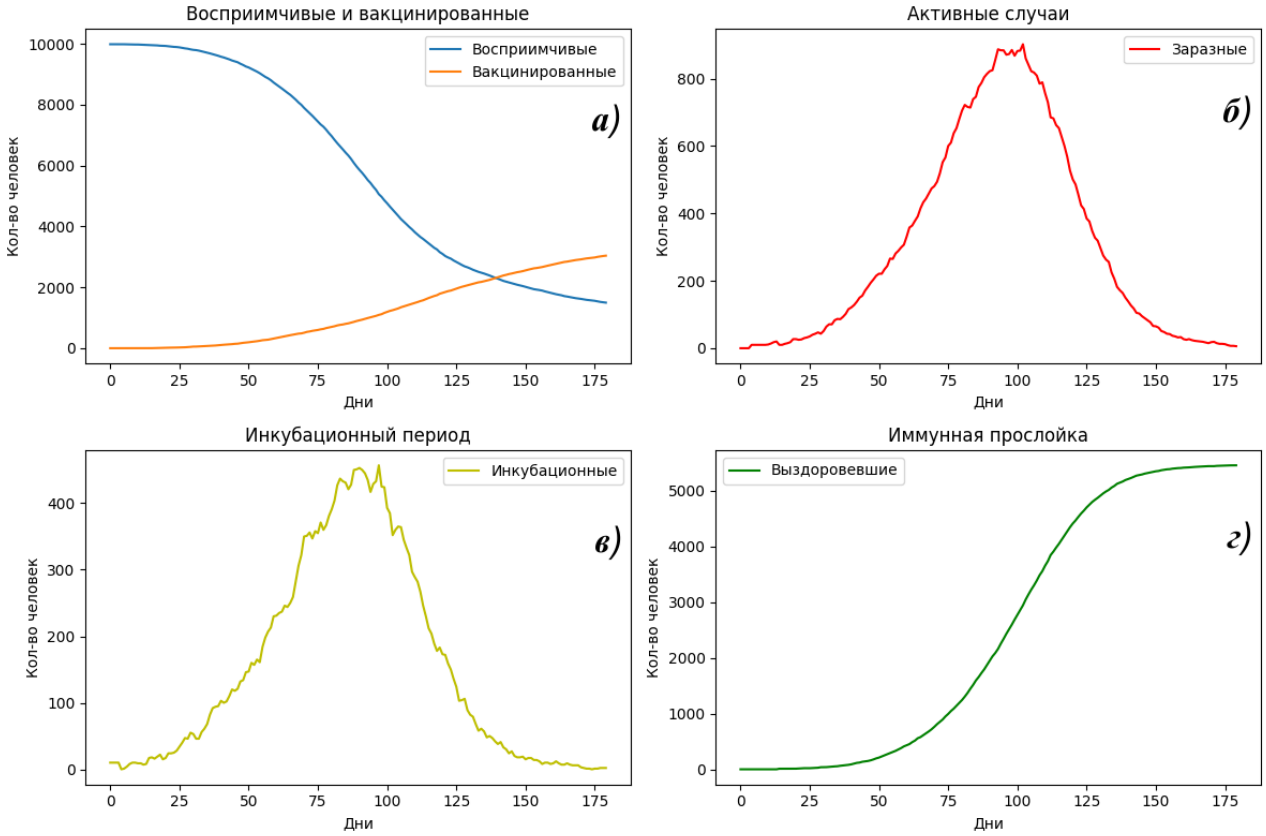


Рис. 7.5. Кривые системной динамики развития заболевания для разных групп населения

Методика выполнения работы

Выбрать задание из предложенных проектов или согласовать новый проект с преподавателем. Подробно изучить программный код и протестировать его на компьютере. Задание можно выполнять группой по 2–4 студента, каждая группа берет один из

примеров. При статистических расчетах использовать табличный процессор MS Office Excel / LibreOffice.org Base, а для формирования текста – текстовый редактор MS Word / Writer LibreOffice.org, Google Docs.

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет в PDF (8–12 страниц), содержащий: титульный лист с Ф.И.О. и номером варианта, ответы на все задания, диаграммы, код Python (можно в виде скриншотов или в приложении).

Исходные файлы: .py файл с кодом анализа графа, файлы диаграмм (.drawio, .xml или др.), Jupyter Notebook (если использовались).

Презентация (5–7 слайдов) для краткой защиты: описание выбранной системы для моделирования, параметры, основные переменные модели, основные требования и особенности принимаемого решения.

Вопросы для самоконтроля

1. Чем отличается динамическое моделирование от структурного анализа? Почему недостаточно только структурных моделей для понимания поведения сложных систем?

2. Объяснить разницу между дискретно-событийным моделированием (DES) и системной динамикой (SD). Для каких типов систем каждый подход наиболее приемлем? Привести примеры.

3. Что такое агентное моделирование и в чем его ключевые особенности? Какие преимущества оно дает по сравнению с DES и SD?

4. Описать концепцию обратных связей в динамических системах. Привести примеры положительной и отрицательной обратной связи в социально-экономических системах.

5. Что такое запаздывания (delays) в системной динамике? Как они влияют на поведение системы? Привести пример.

6. Объяснить концепцию стохастичности в имитационных моделях. Почему важно учитывать случайность и как это реализуется на практике?

Рекомендуемая литература

1. Павловский, Ю. Н. Имитационное моделирование / Ю. Н. Павловский, Н. В. Белотелов, Ю. И. Бродский. Москва : Издательский центр Академия, 2008. 236 с.

2. Вандер, П. Д. Python для сложных задач: наука о данных и машинное обучение / П. Д. Вандер. Санкт-Петербург : Питер, 2018. 576 с.

3. Лычкина, Н. Н. Имитационное моделирование экономических процессов : учебное пособие / Н. Н. Лычкина. Москва : ИНФРА-М, 2014. 254 с.

4. Коровин, А. М. Моделирование систем : учебное пособие к лабораторным работам / А. М. Коровин. Челябинск : Издательский центр ЮУрГУ, 2010. 47 с.

Лабораторная работа № 8
Поддержка принятия решений, многокритериальный выбор и оптимизация

Тема занятия: поддержка принятия решений, многокритериальный выбор и оптимизация.

Цель занятия: поддержка принятия решений. Использование многокритериального анализа и интерпретация результатов для принятия решений.

Материалы и программное обеспечение:

1. Табличный процессор: MS Excel/Base LibreOffice, Google Docs.

2. Язык программирования: Python (с библиотеками: SimPy, SciPy, Matplotlib, Pandas), система программирования: Visual Studio Code (VS Code).

3. Библиотеки оптимизации Python: DEAP (эволюционные алгоритмы), PyGMO, Optuna (фреймворк для оптимизации), для линейного / целочисленного программирования: PuLP, ortools.

Краткие теоретические сведения

Теория принятия решений представляет собой комплексную научную дисциплину, исследующую закономерности выбора оптимальных стратегий в условиях неполной информации, конфликта целей или стохастической среды. Целью *теории принятия решений* является создание методологической базы и инструментария, позволяющих индивидуальным лицам и группам лиц эффективно решать проблемные задачи посредством формирования исчерпывающего множества альтернативных вариантов действий, их комплексной оценки относительно заданных критериев и ограничений, а твкже выявления наилучших решений.

В процессе принятия решений участвуют различные заинтересованные в разной степени лица (стейкхолдеры), играющие определенные роли:

1) владелец проблемы;

2) *лицо, принимающее решение* – человек, фактически осуществляющий выбор рационального варианта действий и несущий ответственность за этот выбор;

3) *эксперты/консультанты* – высококвалифицированные специалисты, имеющие опыт и знания в определенной области;

4) аналитики – специалисты, занимающиеся подготовкой и обоснованием решений.

Возможные варианты решений проблемы называют *альтернативами*, от числа которых зависит сложность задач принятия решений. *Альтернативы* могут быть независимыми и зависимыми. Независимые – *альтернативы*, любые действия с которыми

не влияют на качество других. При зависимых альтернативах оценки одних действий оказывают влияние на качество других.

В теории принятия решений альтернативы характеризуются различными показателями их привлекательности для участников процесса. *Критерии* оценки *альтернатив* – показатели их привлекательности (или непривлекательности) для участников процесса принятия решений. Как и альтернативы, критерии могут быть зависимыми и независимыми. На сложность принятия решений влияет количество используемых критериев. Применение критериев для оценки альтернатив требует определения градаций их качества, т.е. использования определенных шкал оценок по критериям.

Методами принятия решений называют методы анализа альтернатив, которые условно можно разделить на качественные и количественные.

В разработке управленческих решений происходит сочетание формального и неформального подходов. Любые проблемные ситуации, требующие принятия решений, содержат достаточное число неопределенных факторов, которые оказывают влияние как на формальную постановку задачи, так и на средства ее решения: чем больше количественной определенности в характеристике изучаемой проблемы, тем больше доля формальной стороны и наоборот.

Технология разработки управленческих решений представляет собой совокупность последовательно повторяющихся действий, состоящих из отдельных этапов, процедур, операций. Специалистами по управлению предлагаются различные схемы процесса разработки решений (детализированные и агрегированные), различающиеся степенью детализации отдельных процедур и операций.

Процесс принятия решений – один из этапов управленческой деятельности, на котором происходит выбор наиболее предпочтительного решения из допустимого множества решений, или упорядочение множества решений по их важности.

Решение – набор управленческих воздействий (действий со стороны лица, принимающего решение) на объект (систему, комплекс и т.д.) управления, позволяющий привести данный объект в желаемое состояние или достичь поставленной перед ним цели. Это результат анализа, прогнозирования, оптимизации и выбора альтернативы из множества вариантов достижения конкретной цели.

Типовая схема процесса принятия решений устанавливает набор и последовательность этапов на основе жизненного цикла решения проблемы, состоящего из нескольких основных стадий, и представляет собой многоэтапную итеративную процедуру:

– возникновение проблемной ситуации;

– выявление проблемы:

1) содержательное описание проблемы;

2) задание желательного результата решения;

3) определение ограничений;

– постановка задачи:

1) сбор и анализ информации;

2) определение возможных альтернатив;

3) разработка модели проблемной ситуации;

4) формулировка задачи принятия решения;

– поиск решения:

1) разработка (выбор) метода решения задачи;

2) оценка и сравнительный анализ альтернатив;

3) выбор наиболее предпочтительного варианта решения проблемы;

4) изменение формулировки проблемы;

– исполнение решения:

1) реализация и контроль принятого решения;

2) оценка результата решения проблемы.

Многокритериальный выбор и оптимизация

В любой задаче поддержки принятий решений можно выделить три базовых элемента:

1. *Субъект* – лицо, принимающее решение.

2. *Объект* – множество альтернатив (вариантов действий, объектов, стратегий).

3. *Система оценок* – критерии, по которым сравниваются альтернативы.

Многокритериальная задача выбора – задача нахождения такой альтернативы (или множества альтернатив) из допустимого множества, которая наилучшим образом удовлетворяет системе критериев, отражающих различные, часто противоречивые цели ЛПР.

Постановка задачи многокритериального выбора

Пусть:

1. $A = \{A_1, A_2, ..., A_m\}$ – множество альтернатив, $m \geq 2$.

2. $C = \{C_1, C_2, ..., C_n\}$ – множество критериев, $n \geq 2$.

3. $f_j : A \to \mathbb{R}$ (или в шкалу оценок) – функция, ставящая в соответствие каждой альтернативе A_i оценку по критерию C_j. Значение $f_j(A_i)$ – это *критериальная оценка*.

Тогда каждая альтернатива описывается векторной оценкой

$$F(A_i) = (f_1(A_i), f_2(A_i), ... f_n(A_i)).$$

Задача:

Найти альтернативу $A^* \in A$ такую, что $F(A^*)$ является *наилучшим* вектором согласно предпочтениям ЛПР.

Замечание. В отличие от однокритериальной задачи, здесь нет естественного полного порядка. Векторы можно сравнивать лишь частично (по Парето) или с помощью привлечения дополнительной информации (веса, пороги, бинарные отношения).

1) Типы критериев:

– Количественные (цена, время, производительность), можно измерить.

– Качественные (удобство, дизайн, репутация). Требуют экспертных оценок или вербализации.

– Направление оптимизации**:**

а) максимизация («чем больше, тем лучше») – прибыль, скорость;

б) минимизация («чем меньше, тем лучше») – затраты, время отклика.

Критерии могут различаться по нескольким свойствам:

1.1. По типу шкалы. Существуют следующие типы шкал: номинальная, порядковая, интервальная, абсолютная (см. лабораторную работу № 4 «Система, ее свойства и анализ систем»).

– Для номинальных критериев нельзя применять взвешенную сумму.

– Для порядковых надо использовать специальные методы (например, вербализацию, шкалу Саати).

– Для интервальных и абсолютных – нормализация и свертка корректны.

1.2. По направлению оптимизации:

– *Критерий максимизации:* чем больше значение, тем лучше. Примеры: производительность, надежность, прибыль.

– *Критерий минимизации:* чем меньше, тем лучше. Примеры: стоимость, время отклика, риск.

В задачах часто приводят все критерии к единому направлению (например, к максимизации) путём преобразования: $f_j' = -f_j$ или $f_j' = \frac{1}{f_j}$ (если $f_j > 0$).

1.3. По способу получения оценок:

– *Объективные* – измеряемые физически или вычисляемые по формулам (цена, время выполнения).

– *Субъективные* – экспертные оценки (удобство интерфейса, дизайн). Для них применяют методы опроса, парные сравнения.

2) Парето-оптимальность.

3) Свертка критериев.

В случае наличия нескольких критериев для проведения сравнения альтернатив строится *агрегированный показатель*:

$$F(A_i) = (f_1(A_i), f_2(A_i), \ldots f_n(A_i)).$$

Примеры основных методов свертки критериев приведены в табл. 8.1.

Таблица 8.1

Основные методы свертки

Метод	Идея	Формула (упрощ.)
Взвешенная сумма	Линейная комбинация	$\sum_{j=1}^{N} w_j f_j'$
Взвешенное произведение	Учитывает баланс	$\prod_{j=1}^{N} (f_j')^{w_j}$
Идеальная точка	Близость к эталону	$\min \sqrt{\sum_{j=1}^{N} (f_j^* - f_j')^2}$
Лексикографический	Критерии в порядке важности	Сравниваем по первому критерию, при равенстве – по второму критерию и т.д.

4) Нормализация. Приведение всех критериев к безразмерному виду, обычно к интервалу [0,1] или [0,100].

4.1. Линейная (максиминная):

– для максимизации: $f_{\text{norm}} = \dfrac{f - f_{\min}}{f_{\max} - f_{\min}}$;

– для минимизации: $f_{\text{norm}} = \dfrac{f_{\max} - f}{f_{\max} - f_{\min}}$.

4.2. По максимуму: $f_{\text{norm}} = \dfrac{f}{f_{\max}}$ (если все положительны, но теряет ноль).

4.3. Статистическая (Z-оценка): $f_{\text{norm}} = \dfrac{f - \mu}{\delta}$ для работы с выбросами.

5) Выбор весов для критериев:

5.1. Выражают важность критериев относительно друг друга.

5.2. Сумма весов равна 1 (или 100%).

5.3. Способы назначения:

– экспертные оценки (метод парных сравнений Саати);

– статистические (регрессия, энтропийный метод);

– равные веса (если нет информации).

Веса не надо сравнивать с «процентами важности» в обычном смысле, поскольку прежде всего это коэффициенты компромисса. Смена весов кардинально меняет результат.

6) Классификация методов решений:

6.1. *Методы, основанные на свертке* (взвешенная сумма, произведение, идеальная точка).

6.2. *Методы пороговых значений* (ELECTRE, PROMETHEE) – учитывают несравнимость.

6.3. *Методы, основанные на теории полезности* (многокритериальная полезность – MAUT).

6.4. *Методы парных сравнений* (анализ иерархий Саати).

6.5. *Интерактивные методы* (постепенное уточнение предпочтений ЛПР).

7) Этапы решения многокритериальной задачи. На основании вышеприведенных шагов решения многокритериальной задачи можно построить общую схему ее решения, которая приведена на рис. 8.1.

Трудности многокритериального выбора:

1. *Противоречивость критериев*. Улучшение по одному критерию почти всегда ведёт к ухудшению по-другому.

2. *Несоизмеримость*. Критерии могут измеряются в разных единицах (рубли, секунды, баллы).

3. *Субъективность*. Предпочтения ЛПР могут быть нечеткими или противоречивыми.

4. *Размерность*. При количестве альтернатив $n > 5$ человек уже не может интуитивно сравнивать альтернативы.

5. *Качественные критерии.* Имеются значительные трудности при численной формализации критерия.

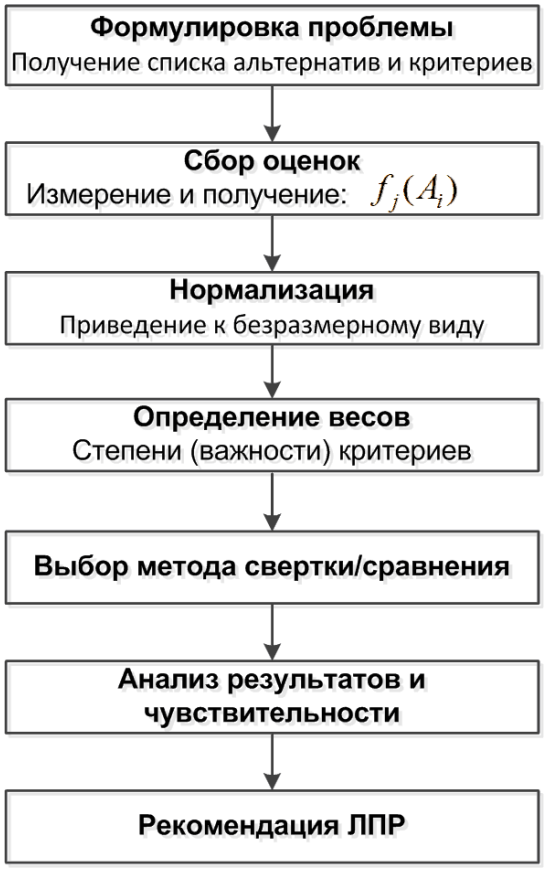


Рис. 8.1. Общие этапы решения многокритериальной задачи

Множество Парето

Итальянский инженер, экономист Вильфредо Парето (1848–1923) изучал распределение богатства и заметил, что в обществе часто невозможно улучшить положение одного человека, не ухудшив положение другого. Его наблюдение и было положено в основу понятия *Парето-эффективности* (или Парето-оптимальности) в экономике, которое со временем стало использоваться и в других науках.

Множество Парето – множество, в котором значение любого из частных критериев оптимальности можно улучшить только за счёт ухудшения других частных критериев.

Отношение доминирования по Парето. Пусть имеется n критериев $C_1, C_2, ..., C_n$. Для простоты будем считать, что все критерии максимизируются.

Альтернатива A_p *доминирует* альтернативу A_q (обозначение: $A_p \succ A_q$ или $A_p >_{Pareto} A_q$), если:

1. $\forall j \in \{1,...,n\}: f_j(A_p) \geq f_j(A_q)$ – по всем критериям A_p *не хуже*, чем A_q.

2. $\forall k \in \{1,...,n\}: f_k(A_p) \geq f_k(A_q)$ – хотя бы по одному критерию A_p *строго лучше*.

Если выполнено только первое условие (везде равенство), то альтернативы *эквивалентны*. Доминирование – это *частичный порядок*. Не все пары альтернатив сравнимы.

Парето-оптимальность (недоминируемость). Альтернатива A^* называется *Парето-оптимальной* (*эффективной*, или *недоминируемой*), если не существует другой альтернативы $A_i \in A$, которая доминирует A^*. Нельзя улучшить A^* ни по одному критерию, не ухудшив её хотя бы по одному другому критерию.

Парето-фронт (множество Парето, эффективное множество) – множество всех Парето-оптимальных альтернатив в данном множестве A.

$$P(A) = \{A_i \in A \mid \nexists A_j \in A : A_j \succ A_i\}.$$

Парето-фронт (множество Парето) – объективная граница достижимого компромисса. Всё, что лежит за ней, – плохо. Всё, что внутри, – недостижимо или неэффективно.

Множество Парето (множество альтернатив, оптимальных в смысле Парето) – набор всех компромиссных решений в задачах многокритериальной оптимизации.

Пример *множества Парето* для случая двухкритериальной задачи представлен на рис. 8.2.

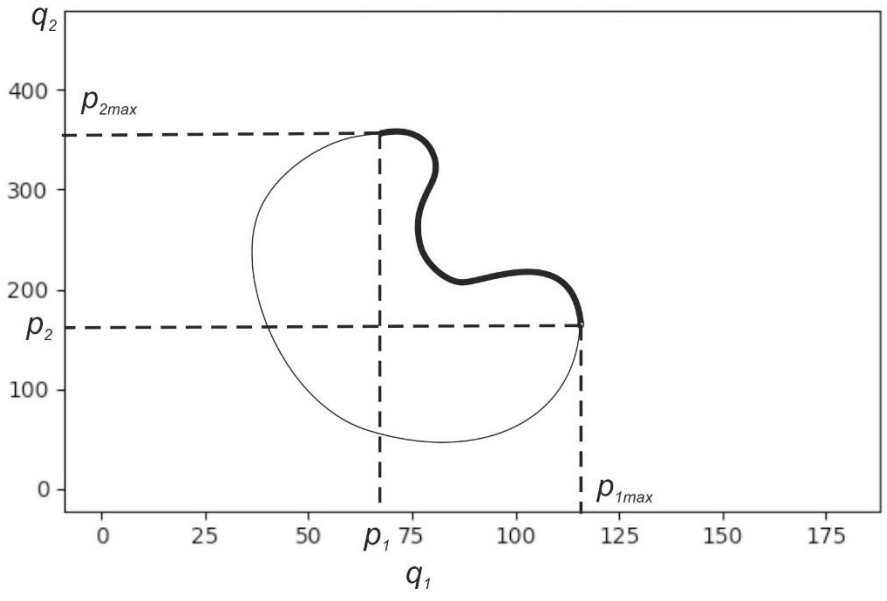


Рис. 8.2. Пример множества Парето для двухкритериальной задачи

Пример. Анализ ИТ-архитектуры программного приложения.

Рассмотрим три альтернативы архитектуры приложения (табл. 8.2).

Таблица 8.2

Альтернативные архитектуры приложения

Архитектура	Производительность (TPS) max	Затраты на разработку (чел-мес) min	Масштабируемость (балл 1–10) max
Монолит	1000	6	3
Микросервисы	800	12	9
Серверлес	900	8	8

Приведём все критерии к максимизации, затраты возьмём со знаком «минус» или преобразуем (табл. 8.3).

Таблица 8.3

Альтернативные архитектуры приложения

Архитектура	Производительность (TPS) max	Затраты на разработку (чел-мес) min	Масштабируемость (балл 1–10) max
Монолит	1000	–6	3
Микросервисы	800	–12	9
Серверлес	900	–8	8

Проведем парные сравнения.

1. Микросервисы и Монолит: 800 < 1000 – не доминирует.
2. Серверлес и Монолит: 900 < 1000 – не доминирует.
3. Микросервисы и Серверлес: 800 < 900, –12 < –8, 9 > 8 – нет доминирования.
4. Монолит и Серверлес: 1000 > 900, –6 > –8, 3 < 8 – нет доминирования.

Таким образом, все три альтернативы находятся на Парето-фронте. Выбор зависит от выбора весов.

Многокритериальный выбор – поиск наилучшей альтернативы на основе векторной оценки. Основная проблема такого подхода заключается в том, что критерии противоречивы и несоизмеримы. Решение обычно требует нормализации, задания весов (важности), применения свертки или другого метода сравнения. Результат зависит от предпочтений ЛПР, причем если будут выбраны разные веса, то получаются, соответственно, и разные решения.

Необходимо отметить разницу между *многокритериальной оптимизацией* (цель – найти множество Парето) и *многокритериальным принятием решений* (цель – выбрать одну альтернативу). В приведенных ниже программах используется *многокритериальное принятие решений*, но с предварительным построением *Парето-фронта*.

Моделирование как метод принятия управленческих решений

Методы принятия управленческих решений представляют собой совокупность методологических подходов, охватывающих последовательность процедур: поиск и анализ альтернатив, их многокритериальную оценку, выбор оптимального решения, а

также организацию его реализации в целях эффективного разрешения управленческих проблем.

Ключевое место в данной системе методов занимает *моделирование* – научно-методический подход к познанию действительности, основанный на создании идеализированных абстрактных репрезентаций (моделей) объектов, систем, процессов и явлений с последующим их исследованием посредством анализа построенных моделей.

Выбор метода моделирования для решения поставленной задачи, проблемы или исследования системы является актуальной задачей, с которой системный аналитик должен уметь справляться.

В практике управленческой деятельности наибольшую эффективность демонстрирует комплексное применение разнородных методов *моделирования*, включая аналитическое, физическое, эвристическое, математическое и иные их виды. В контексте современной профессиональной деятельности *системного аналитика* особое распространение получило *имитационное* моделирование, которое наилучшим образом соответствует специфике задач управления сложными системами, характеризующимися высокой степенью неопределённости, нелинейностью связей и динамичностью процессов.

Традиционное применение электронно-вычислительных машин в управленческой деятельности характеризуется ограниченной эффективностью. Помимо доступа к данным из информационных баз и выполнения стандартных экономических и технологических расчётов, руководитель сталкивается с широким спектром задач по управлению системой, которые не поддаются решению в рамках традиционных информационных технологий ввиду их высокой сложности, многокритериальности и наличия неструктурированных компонентов.

В ответ на данную потребность были разработаны компьютерные системы нового поколения – *системы поддержки принятия решений*, предназначенные для комплексного анализа сложных управленческих ситуаций и содействия лицам, принимающим решения в выборе оптимальных стратегий.

Имитационное моделирование выступает ключевым системообразующим компонентом СППР, поскольку обеспечивает возможность комплексного исследования значительного числа альтернативных вариантов управленческих решений и имитации различных сценариев развития событий при произвольных входных параметрах.

Основным преимуществом данного метода является его способность предоставлять исследователю ответы на вопросы типа «Что будет, если…?» в процессе проверки новых стратегий и анализа потенциальных ситуаций. Это позволяет оценивать последствия управленческих воздействий в условиях виртуальной среды без реализации их в реальной системе, что существенно снижает риски и повышает обоснованность принимаемых решений.

Задания к работе

1. Оптимизация маршрутов доставки. Задача коммивояжера с ограничениями.

Служба доставки еды имеет один ресторан и 10 точек доставки. Необходимо найти оптимальный маршрут для курьера, чтобы минимизировать время доставки (или пройденное расстояние) с учетом временных окон, когда клиенты готовы принять заказ.

Исходные данные:

1) Координаты ресторана и 10 точек доставки (случайно сгенерированы в пределах города 10 × 10 км).

2) Время перемещения между точками пропорционально расстоянию (скорость 30 км/ч).

3) Каждый заказ имеет временное окно [ready_time, due_time] и время обслуживания (разгрузки) 5 мин.

4) Курьер начинает работу в 10:00 и должен вернуться в ресторан.

1. Формализация как задачи VRP (Vehicle Routing Problem):

1.1. Управляющие переменные: бинарные $x_{ij} = 1$, если курьер едет из *i* в *j*.

1.2. Целевая функция: минимизация общего времени маршрута.

1.3. Ограничения:

– каждый заказ должен быть обслужен ровно один раз;

– маршрут начинается и заканчивается в ресторане (депо);

– учет временных окон;

– исключение подциклов.

2. Решение с помощью генетического алгоритма.

Из-за сложности задачи (NP-трудная) используется эволюционный алгоритм.

3. Анализ результатов.

3.1. Построить график сходимости генетического алгоритма.

3.2. Визуализировать лучший маршрут на карте.

Проблема маршрутизации транспортных средств (англ. Vehicle Routing Problem, VRP) – задача маршрутизации транспорта (задача комбинаторной оптимизации). Заключается в нахождении оптимальных маршрутов для группы транспортных средств, которые должны обслуживать множество клиентов с учётом различных ограничений (грузоподъёмность, временные окна и т.д.).

Пример кода упрощенного генетического алгоритма приведен на листинге 8.1.

Листинг 8.1

Код упрощенного генетического алгоритма

```
import numpy as np
import random
from deap import base, creator, tools, algorithms

# Генерация данных
num_customers = 10
restaurant = (0, 0)
customers = [(random.uniform(-5, 5), random.uniform(-5, 5)) for _ in
range(num_customers)]
points = [restaurant] + customers
```

```
# Матрица расстояний
dist_matrix = np.zeros((num_customers+1, num_customers+1))
for i in range(num_customers+1):
    for j in range(num_customers+1):
        dist_matrix[i][j] = np.linalg.norm(np.array(points[i]) - np.ar-
ray(points[j]))

# Временные окна (готовность и крайний срок)
ready_time = [0] + [random.randint(0, 30) for _ in range(num_customers)]  #
в минутах от начала работы
due_time = [1440] + [ready_time[i] + random.randint(30, 60) for i in
range(1, num_customers+1)]
service_time = [0] + [5] * num_customers  # время обслуживания

# Параметры генетического алгоритма
POP_SIZE = 100
NGEN = 50

# Создаем классы для DEAP
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
toolbox.register("indices", random.sample, range(1, num_customers+1),
num_customers)
toolbox.register("individual", tools.initIterate, creator.Individual,
toolbox.indices)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evaluate(individual):
    """Оценка маршрута: общее время с учетом временных окон и штраф за
опоздание."""
    total_time = 0
    current_time = 0
    current_point = 0  # начинаем с ресторана
    penalty = 0

    for next_point in individual:
        travel_time = dist_matrix[current_point][next_point] * 60 / 30  #
переводим в минуты (скорость 30 км/ч)
        arrival_time = current_time + travel_time
        # Если прибыли раньше готовности, ждем
        start_time = max(arrival_time, ready_time[next_point])
        # Если опоздали, штраф
        if arrival_time > due_time[next_point]:
            penalty += (arrival_time - due_time[next_point]) * 10  # штраф
10 у.е. за минуту опоздания
        # Обновляем текущее время
        current_time = start_time + service_time[next_point]
        current_point = next_point
        # Возвращаемся в ресторан
    travel_time_back = dist_matrix[current_point][0] * 60 / 30
    total_time = current_time + travel_time_back
        # Целевая функция: общее время + штраф
    return (total_time + penalty,)

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxPartialyMatched)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)
```

```
# Запуск генетического алгоритма
population = toolbox.population(n=POP_SIZE)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean)
stats.register("min", np.min)

population, logbook = algorithms.eaSimple(population, toolbox, cxpb=0.7,
mutpb=0.2, ngen=NGEN,
stats=stats, halloffame=hof, verbose=True)

# Лучшее решение
best_ind = hof[0]
print("Лучший маршрут:", best_ind)
print("Значение целевой функции:", evaluate(best_ind)[0])
```

Оптимизация портфеля проектов (многокритериальная оптимизация).

Компания имеет 15 потенциальных проектов, но ограниченный бюджет и человеческие ресурсы. Каждый проект имеет ожидаемую прибыль, требуемые инвестиции, необходимое количество разработчиков и степень риска. Необходимо выбрать набор проектов, максимизирующий суммарную прибыль и минимизирующий суммарный риск, с учетом ограничений.

Исходные данные:

1) Для каждого проекта i.

– Прибыль: p_i (выбирается случайно от 50 до 200 ед.).

– Инвестиции: i_i (выбираются случайно от 10 до 100 ед.).

– Разработчики: d_i (выбираются случайно от 1 до 5).

– Риск: r_i (шкала от 1 до 5).

2) Ограничения:

– Бюджет: 300 у.е.

– Количество разработчиков: 20 человек.

3) Цель: максимизировать суммарную прибыль и минимизировать суммарный риск.

Задание.

1. Формализация как многокритериальная задача.

1.1. Управляющие переменные: $x_i = 1$ (выбран проект), $x_i = 0$ (не выбран).

1.2. Целевые функции:

– Максимизировать суммарную прибыль:

$$\sum p_i x_i \to \max .$$

– Минимизировать суммарный риск:

$$\sum r_i x_i \to \min .$$

1.3. Ограничения:

– $\sum i_i x_i \le 300.$

– $\sum d_i x_i \le 20.$

2. Решение с помощью метода взвешивания и поиска Парето-фронта:

2.1. Используется эволюционный алгоритм (NSGA-II) для многокритериальной оптимизации.

3. Анализ множества Парето (фронта):

3.1. Выбрать 3–5 решений с Парето-фронта и обсудить их применимость в зависимости от склонности компании к риску.

3.2. Построить график Парето-фронта.

Обозначения переменных модели и соответствующих им переменных в программе представлены в табл. 8.4.

Таблица 8.4

Обозначение переменных в программе

№	Переменная модели	Наименование переменной	Наименование переменной в программе
1	Прибыль	p_i	profit[*i*]
2	Инвестиции	i_i	investment[*i*]
3	Разработчики	d_i	developers[i]
4	Риск	r_i	risk[i]
5	Проект (выбран/нет)	x_i	x[i]

Пример реализации оптимизации портфеля проектов приведен на листинге 8.2.

Листинг 8.2

Оптимизация портфеля проектов

```
import random
from deap import base, creator, tools, algorithms

# Генерация данных
num_projects = 15
profits = [random.randint(50, 200) for _ in range(num_projects)]
investments = [random.randint(10, 100) for _ in range(num_projects)]
developers = [random.randint(1, 5) for _ in range(num_projects)]
risks = [random.randint(1, 5) for _ in range(num_projects)]

budget = 300
max_developers = 20

# Создаем классы для многокритериальной оптимизации
creator.create("FitnessMulti", base.Fitness, weights=(1.0, -1.0))  # при-
быль максимизировать, риск минимизировать
creator.create("Individual", list, fitness=creator.FitnessMulti)

toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_bool, n=num_projects)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evaluate(individual):
    total_profit = sum(profits[i] * individual[i] for i in range(num_pro-
jects))
    total_risk = sum(risks[i] * individual[i] for i in range(num_projects))
```

```
    total_investment = sum(investments[i] * individual[i] for i in
range(num_projects))
    total_developers = sum(developers[i] * individual[i] for i in
range(num_projects))

    # Штрафы за нарушение ограничений
    penalty = 0
    if total_investment > budget:
        penalty += (total_investment - budget) * 10
    if total_developers > max_developers:
        penalty += (total_developers - max_developers) * 10

    # Возвращаем значения с учетом штрафа (прибыль уменьшается на штраф,
риск увеличивается)
    return (total_profit - penalty, total_risk + penalty/10)

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.1)
toolbox.register("select", tools.selNSGA2)

# Запуск NSGA-II
population = toolbox.population(n=100)
hof = tools.ParetoFront()
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean, axis=0)
stats.register("std", np.std, axis=0)
stats.register("min", np.min, axis=0)
stats.register("max", np.max, axis=0)

population, logbook = algorithms.eaMuPlusLambda(population, toolbox,
mu=100, lambda_=200,
cxpb=0.7, mutpb=0.2, ngen=40,
stats=stats, halloffame=hof, verbose=True)

# Получение Парето-фронта
pareto_front = list(hof)
# Визуализация Парето-фронта
profits_front = [ind.fitness.values[0] for ind in pareto_front]
risks_front = [ind.fitness.values[1] for ind in pareto_front]

import matplotlib.pyplot as plt
plt.scatter(profits_front, risks_front, c='red')
plt.xlabel('Прибыль')
plt.ylabel('Риск')
plt.title('Парето-фронт')
plt.grid(True)
plt.show()
# Вывод нескольких решений с Парето-фронта
for i, ind in enumerate(pareto_front[:5]):
    print(f"Решение {i+1}: прибыль = {ind.fitness.values[0]:.0f}, риск =
{ind.fitness.values[1]:.0f}")
    print("Выбранные проекты:", [idx for idx, val in enumerate(ind) if
val])
    print()
```

На рис. 8.3 приведены результаты моделирования Парето-фронта для оптимизации портфеля проектов.

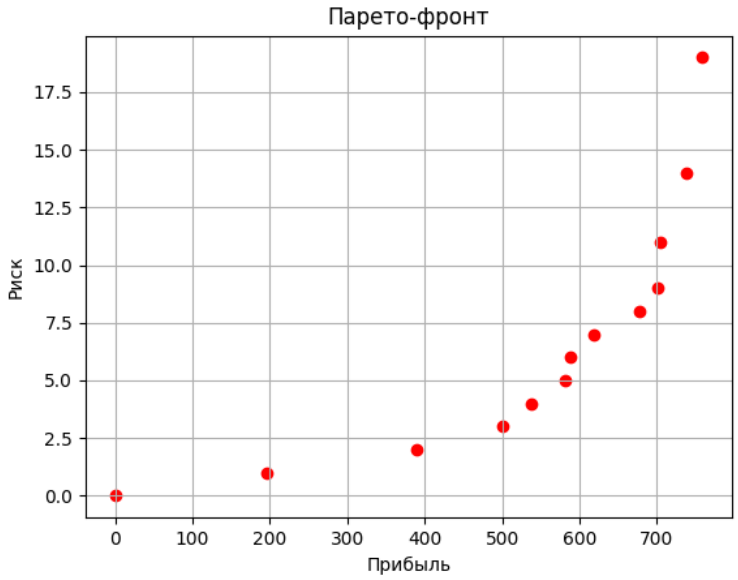


Рис. 8.3. Кривая Парето-фронта для оптимизации портфеля проектов

2. Оптимизация размещения микросервисов в облаке.

Компания развертывает 50 микросервисов на 10 виртуальных машинах. Каждый сервис имеет требования к CPU, RAM, сети. Необходимо найти размещение, минимизирующее стоимость инфраструктуры при соблюдении SLA (задержка между связанными сервисами < 20 мс).

Исходные данные:

1. Сервисы: S_i = {cpu: 0,1–2,0 ядер, ram: 0,5–8 ГБ, network_bw: 10–100 Мбит/с}.

2. Виртуальная машина: V_i = {cpu: 8 ядер, ram: 32 ГБ, cost: 0,1долл/час}.

3. Матрица связности между сервисами (интенсивность обмена данными). Заполняется случайными числами.

4. Ограничения:

– Суммарные требования сервисов на ВМ ≤ ресурсам ВМ.

– Связанные сервисы должны быть размещены близко (в одном датацентре или зоне доступности).

– Максимум 80% загрузки любой ВМ для резерва.

Обозначения переменных модели и соответствующих им переменных в программе представлены в табл. 8.5.

Таблица 8.5

Обозначение переменных в программе

№	Переменная модели	Наименование переменной	Наименование переменной в программе
1	Сервисы	S_i	S[*i*]
2	Тип виртуальной машины	V_i	V[*i*]
3	Зоны доступности	L_i	L[i]
4	Переменная *x*, если сервис *s* на *v*	x_{ij}	x[s,v]
5	Переменная *y*, если *v* используется	y_i	y[i]
6	Переменная *z*, если *v* на зоне *l*	z_{ij}	x[v,l]

Пример реализации задачи с использованием алгоритма целочисленного программирования приведен на листинге 8.3.

Листинг 8.3

Алгоритм целочисленного программирования

```
def optimize_placement(services, vms, connectivity_matrix):
    """Оптимизация размещения сервисов"""

    # Индексы
    S = range(len(services))  # Сервисы
    V = range(len(vms))       # ВМ
    L = ['zone1', 'zone2', 'zone3']  # Зоны доступности

    # Создаем задачу
    prob = pulp.LpProblem('ServicePlacement', pulp.LpMinimize)

    # Переменные решения
    # x[s][v] = 1 если сервис s на ВМ v
    x = pulp.LpVariable.dicts('x',
        [(s, v) for s in S for v in V],
        lowBound=0, upBound=1, cat='Binary')

    # y[v] = 1 если ВМ v используется
    y = pulp.LpVariable.dicts('y', V, lowBound=0, upBound=1, cat='Binary')

    # z[v][l] = 1 если ВМ v в зоне l
    z = pulp.LpVariable.dicts('z',
        [(v, l) for v in V for l in L],
        lowBound=0, upBound=1, cat='Binary')

    # Целевая функция: минимизация стоимости
    prob += pulp.lpSum(vms[v]['cost'] * y[v] for v in V)

    # Ограничения
    # 1. Каждый сервис размещен ровно на одной ВМ
    for s in S:
        prob += pulp.lpSum(x[s, v] for v in V) == 1

    # 2. Ресурсные ограничения ВМ
    for v in V:
        # CPU
        prob += pulp.lpSum(services[s]['cpu'] * x[s, v] for s in S) <=
vms[v]['cpu'] * 0.8 * y[v]
        # RAM
        prob += pulp.lpSum(services[s]['ram'] * x[s, v] for s in S) <=
vms[v]['ram'] * y[v]

    # 3. Связанные сервисы должны быть в одной зоне
    for s1 in S:
        for s2 in S:
            if connectivity_matrix[s1][s2] > 100:  # Интенсивный обмен
                for v1 in V:
                    for v2 in V:
                        prob += x[s1, v1] + x[s2, v2] <= 1 +
pulp.lpSum(z[v1, l] * z[v2, l] for l in L)

    # 4. Каждая ВМ только в одной зоне
    for v in V:
        prob += pulp.lpSum(z[v, l] for l in L) == y[v]
```

```
    # Решение
    prob.solve(pulp.PULP_CBC_CMD(msg=False))

    return prob, x, y, z

# Генерация тестовых данных
services = [{'cpu': np.random.uniform(0.1, 2.0),
             'ram': np.random.uniform(0.5, 8.0)}
            for _ in range(50)]

vms = [{'cpu': 8, 'ram': 32, 'cost': 0.1} for _ in range(10)]

connectivity = np.random.exponential(scale=50, size=(50, 50))
np.fill_diagonal(connectivity, 0)

# Запуск оптимизации
prob, x, y, z = optimize_placement(services, vms, connectivity)
```

Пример реализации эволюционного алгоритма для крупномасштабной задачи представлен на листинге 8.4.

Листинг 8.4

Эволюционный алгоритм для крупномасштабной задачи

```
from deap import base, creator, tools, algorithms
import random

def placement_fitness(individual, services, vms, connectivity):
    """Оценка размещения"""
    # Индивидуум: список [vm_index для каждого сервиса]

    # Счетчики ресурсов ВМ
    vm_cpu = [0] * len(vms)
    vm_ram = [0] * len(vms)

    # Распределение сервисов по ВМ
    for s_idx, vm_idx in enumerate(individual):
        vm_cpu[vm_idx] += services[s_idx]['cpu']
        vm_ram[vm_idx] += services[s_idx]['ram']

    # Штрафы за превышение ресурсов
    penalty = 0
    for v in range(len(vms)):
        if vm_cpu[v] > vms[v]['cpu'] * 0.8:
            penalty += 1000 * (vm_cpu[v] - vms[v]['cpu'] * 0.8)
        if vm_ram[v] > vms[v]['ram']:
            penalty += 1000 * (vm_ram[v] - vms[v]['ram'])

    # Штраф за сетевые задержки
    network_penalty = 0
    for s1 in range(len(services)):
        for s2 in range(len(services)):
            if connectivity[s1][s2] > 50:
                if individual[s1] != individual[s2]:
                    # Сервисы в разных ВМ - дополнительная задержка
                    network_penalty += connectivity[s1][s2] * 0.1

    # Стоимость (количество использованных ВМ)
    used_vms = len(set(individual))
    cost = used_vms * vms[0]['cost']
```

```
    return (cost + penalty + network_penalty,)

# Настройка генетического алгоритма
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
toolbox.register("attr_vm", random.randint, 0, len(vms)-1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
                 toolbox.attr_vm, n=len(services))
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", placement_fitness, services=services,
                 vms=vms, connectivity=connectivity)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutUniformInt, low=0, up=len(vms)-1,
indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)

# Запуск
population = toolbox.population(n=100)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", np.mean)
stats.register("min", np.min)

population, logbook = algorithms.eaSimple(
    population, toolbox, cxpb=0.7, mutpb=0.2, ngen=100,
    stats=stats, halloffame=hof, verbose=True
)
```

Пример реализации анализа компромиссов стоимость/производительность приведен на листинге 8.5.

Листинг 8.5

Анализ компромиссов стоимость/производительность

```
import numpy as np
import matplotlib.pyplot as plt
import random

NUM_SERVICES = 50
NUM_VMS = 10

# Генерация данных
services = [{'cpu': np.random.uniform(0.1, 2.0),
             'ram': np.random.uniform(0.5, 8.0)}
            for _ in range(NUM_SERVICES)]

vms = [{'cpu': 8, 'ram': 32, 'cost': 0.1} for _ in range(NUM_VMS)]

connectivity = np.random.exponential(scale=50, size=(NUM_SERVICES, NUM_SER-
VICES))
np.fill_diagonal(connectivity, 0)

# Стратегии размещения
def optimize_for_cost(services, vms):
    """Минимизируем количество ВМ"""
    placement = []
    current_vm = 0
```

```
    for _ in services:
        placement.append(current_vm)
        if random.random() < 0.1:
            current_vm = min(current_vm + 1, len(vms) - 1)

    return placement
def optimize_for_performance(services, vms, connectivity):
    """Распределяем сервисы по разным ВМ"""
    placement = []

    for i in range(len(services)):
        placement.append(i % len(vms))

    return placement
def optimize_balanced(services, vms, connectivity):
    """Баланс между стоимостью и производительностью"""
    placement = []
    for _ in services:
        placement.append(random.randint(0, len(vms) - 1))
    return placement
# Метрики
def calculate_cost(placement, vms):
    used_vms = len(set(placement))
    return used_vms * vms[0]['cost']

def calculate_performance(placement, connectivity):
    performance = 0
    for s1 in range(len(placement)):
        for s2 in range(len(placement)):
            if placement[s1] == placement[s2]:
                performance += connectivity[s1][s2]
    return performance
def check_sla_violations(placement, connectivity):
    violations = 0
    for s1 in range(len(placement)):
        for s2 in range(len(placement)):
            if connectivity[s1][s2] > 50:
                if placement[s1] != placement[s2]:
                    violations += 1

    return violations

# Симуляция стратегий
strategies = {
    'cost_optimized': lambda: optimize_for_cost(services, vms),
    'performance_optimized': lambda: optimize_for_performance(services,
vms, connectivity),
    'balanced': lambda: optimize_balanced(services, vms, connectivity)
}

results = []

for name, strategy in strategies.items():
    placement = strategy()
    cost = calculate_cost(placement, vms)
    performance = calculate_performance(placement, connectivity)
    sla_violations = check_sla_violations(placement, connectivity)
    results.append({
        'strategy': name,
        'cost': cost,
        'performance': performance,
        'sla_violations': sla_violations
    })
```

```
# Результаты
for r in results:
    print(r)
# Построение графика

costs = [r['cost'] for r in results]
perfs = [r['performance'] for r in results]

plt.scatter(costs, perfs)

for i, r in enumerate(results):
    plt.annotate(r['strategy'], (costs[i], perfs[i]))

plt.xlabel('Стоимость ($/час)')
plt.ylabel('Производительность (операций/сек)')
plt.title('Компромисс стоимость/производительность')
plt.grid(True)

plt.show()
```

На рис. 8.4 приводятся рассчитанные точки для анализа компромиссов стоимость/производительность.

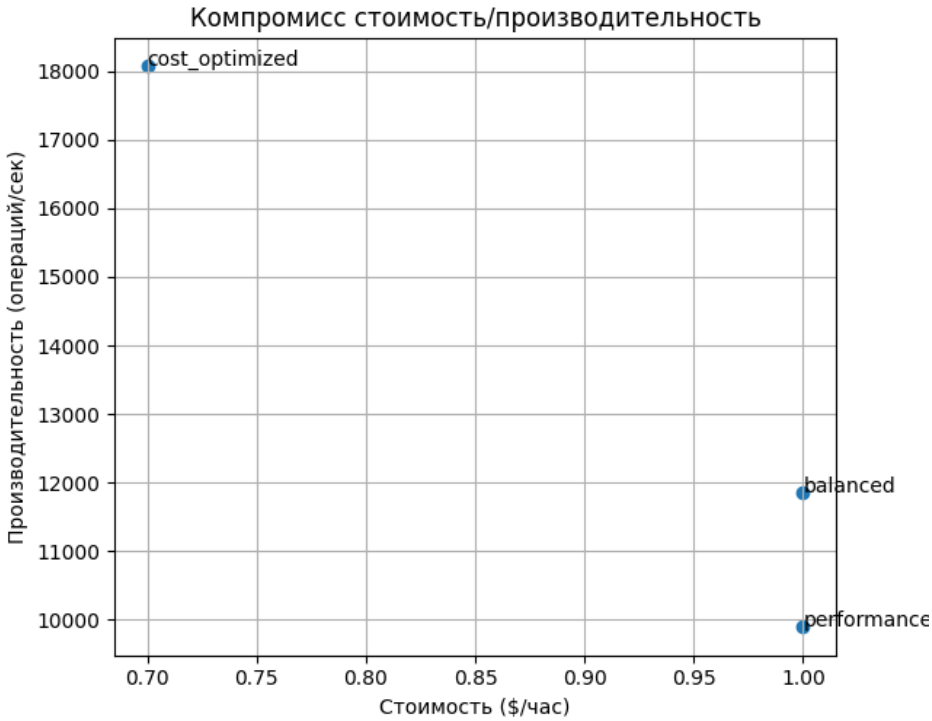


Рис. 8.4. Точки множества Парето для двухкритериальной задачи

3. Система поддержки принятия решений.

Разработать систему поддержки принятия решений. Использовать пример из лабораторной работы № 7 «Имитационное моделирование» для случая дискретно-событийного моделирования центра обработки вызовов. Разработать интерактивную панель для сравнения решений. Предусмотреть следующие функциональные возможности: интерактивная панель для сравнения решений, анализ чувствительности оптимального решения, оценка устойчивости к неопределенности.

На листинге 8.6 приводится пример реализации системы поддержки принятий решений, на листингах 8.8–8.10 – различные дополнения СППР, в которых реализованы: интерактивный выбор решений, анализ чувствительности к изменению нагрузки и анализ устойчивости к неточности оценок.

Листинг 8.6

Система поддержки принятия решений

```
class DecisionSupportSystem:
    """
Система поддержки принятия решений для колл-центра
    """

def __init__(self, pareto_front, simulation_wrapper):
    self.pareto_front = pareto_front
    self.simulation_wrapper = simulation_wrapper
    self.decision_matrix = self.create_decision_matrix()

def create_decision_matrix(self):
   """
Создание матрицы решений с полной информацией
    """
matrix = []

for solution in self.pareto_front:
        _, metrics, cost = self.simulation_wrapper(solution)

    matrix.append({
    'solution': solution,
    'cost': cost,
    'avg_wait': metrics['avg_wait'],
    'abandon_rate': metrics['abandonment_rate'],
    'max_utilization': metrics['max_utilization'],
    'calls_served': metrics['calls_served'],
    'score': self.calculate_decision_score(solution, metrics, cost)
})

return pd.DataFrame(matrix)

def calculate_decision_score(self, solution, metrics, cost):
    """
Расчет интегрального показателя для сравнения решений
    """
   # Нормализация показателей
    norm_cost = cost / 1000000  # к миллионам
    norm_wait = metrics['avg_wait'] / 300  # к 5 минутам
    norm_abandon = metrics['abandonment_rate'] / 0.1  # к 10%

    # Весовые коэффициенты (можно настраивать)
    weights = {'cost': 0.4, 'wait': 0.3, 'abandon': 0.3}
    score = (weights['cost'] * norm_cost +
            weights['wait'] * norm_wait +
            weights['abandon'] * norm_abandon)

return score

def recommend_solution(self, preferences):
   """
    Рекомендация решения на основе предпочтений пользователя
```

```
    preferences: словарь с приоритетами {'cost': 0.5, 'wait': 0.3,
'abandon': 0.2}
   """
   # Пересчет scores с учетом предпочтений пользователя
    self.decision_matrix['user_score'] = self.decision_matrix.apply(
    lambda row: self.calculate_user_score(row, preferences), axis=1
)
        # Рекомендация лучшего решения
    best_idx = self.decision_matrix['user_score'].idxmin()
    best_solution = self.decision_matrix.loc[best_idx]

    return best_solution

def create_interactive_dashboard(self):
        """
   Создание интерактивной панели с использованием Plotly
    """
import plotly.express as px
import plotly.graph_objects as go
fig = go.Figure()
# 3D scatter plot
fig.add_trace(go.Scatter3d(
        x=self.decision_matrix['cost'],
        y=self.decision_matrix['avg_wait'],
        z=self.decision_matrix['abandon_rate'],
        mode='markers',
        marker=dict(
           size=8,
           color=self.decision_matrix['score'],
           colorscale='Viridis',
           showscale=True,
           colorbar=dict(title="Интегральный показатель")
            ),
           text=[f"Решение {i}" for i in range(len(self.decision_matrix))],
           hovertemplate=
            '<b>Решение %{text}</b><br>' +
            'Стоимость: %{x:.0f} руб<br>' +
            'Время ожидания: %{y:.1f} сек<br>' +
            '% отказов: %{z:.2%}<br>' +
            '<extra></extra>'
        ))
        fig.update_layout(
            title='Пространство решений колл-центра',
            scene=dict(
                xaxis_title='Стоимость (руб/мес)',
                yaxis_title='Среднее время ожидания (сек)',
                zaxis_title='Процент отказов'
            ),
            width=1000,
            height=800
        )

return fig
```

В табл. 8.6 приводятся рассчитанные данные для СППР.

Наилучшее решение для пользователя приведено на листинге 8.7.

Таблица 8.6

Выходные данные, полученные для СППР

№	Cost	avg_wait	max_utilization	calls_served	score
1	860064	148.846646	0.835968	55	0.651886
2	366431	76.578013	0.583704	196	0.257187
3	944398	54.971422	0.948341	99	0.632263
4	575181	168.119634	0.547908	58	0.477580
5	281260	277.805388	0.544501	169	0.623622

Листинг 8.7

Наилучшее решения для пользователя

```
solution               2.000000
cost              366431.000000
avg_wait              76.578013
abandon_rate           0.011346
max_utilization        0.583704
calls_served         196.000000
score                  0.257187
user_score             0.282485
```

Задание. С использованием листинга 8.8 разработать графический интерфейс программного прототипа для интерактивного взаимодействия с программой.

Листинг 8.8

Интерактивный выбор решения

```
def create_interactive_sliders():
    """
    Создание интерактивных слайдеров для настройки предпочтений
    """
from ipywidgets import interact, FloatSlider

dss = DecisionSupportSystem(pareto_front, simulation_wrapper)

@interact(
    cost_priority=FloatSlider(value=0.4, min=0, max=1, step=0.1,
      description='Важность стоимости:'),
      wait_priority=FloatSlider(value=0.3, min=0, max=1, step=0.1,
      description='Важность времени ожидания:'),
      abandon_priority=FloatSlider(value=0.3, min=0, max=1, step=0.1,
      description='Важность % отказов:')
    )
def recommend_with_preferences(cost_priority, wait_priority, abandon_prior-
ity):
        # Нормализация весов
    total = cost_priority + wait_priority + abandon_priority
    preferences = {
     'cost': cost_priority / total,
       'wait': wait_priority / total,
       'abandon': abandon_priority / total
}

        # Получение рекомендации
        recommendation = dss.recommend_solution(preferences)
```

```
        # Вывод результатов
print("=== РЕКОМЕНДОВАННОЕ РЕШЕНИЕ ===")
print(f"Конфигурация: {recommendation['solution']}")
print(f"Стоимость: {recommendation['cost']:.0f} руб/мес")
print(f"Среднее время ожидания: {recommendation['avg_wait']:.1f} сек")
print(f"Процент отказов: {recommendation['abandon_rate']:.2%}")
print(f"Утилизация: {recommendation['max_utilization']:.1%}")
print(f"Обслужено звонков: {recommendation['calls_served']:.0f}")

return recommendation
```

Задание. Исследовать, насколько оптимальное решение чувствительно к изменению внешних условий. Используйте программный код, представленный на листингах 8.9 и 8.10.

Листинг 8.9

Анализ чувствительности к изменению нагрузки

```
def sensitivity_analysis(optimal_solution, load_factors):
    """
    Анализ чувствительности к изменению нагрузки
    load_factors: множители нагрузки [0.5, 0.75, 1.0, 1.25, 1.5]
    """
    results = []

    for factor in load_factors:
        # Модификация модели для другого уровня нагрузки
    def modified_simulation_wrapper(x):
        # Копируем оригинальную функцию, но изменяем интенсивность звонков
            original_value, metrics, cost = simulation_wrapper(x)
        # Корректируем метрики в зависимости от нагрузки
        # (упрощенная модель - в реальности нужно перезапускать симуляцию)
            adjusted_metrics = {
'avg_wait': metrics['avg_wait'] * factor**2,  # Квадратичный рост
'abandonment_rate': min(1.0, metrics['abandonment_rate'] * factor),
          'max_utilization': min(1.0, metrics['max_utilization'] * factor)
       }

return original_value, adjusted_metrics, cost

        # Оценка оптимального решения при измененной нагрузке
     value, metrics, cost = modified_simulation_wrapper(optimal_solution)

        results.append({
            'load_factor': factor,
            'avg_wait': metrics['avg_wait'],
            'abandon_rate': metrics['abandonment_rate'],
            'utilization': metrics['max_utilization'],
            'sla_violation': metrics['avg_wait'] > 180 or metrics['abandon-
ment_rate'] > 0.05
        })

return pd.DataFrame(results)
```

Листинг 8.10

Анализ устойчивости к неточности оценок

```
def robustness_analysis(optimal_solution, parameter_uncertainty):
    """
Анализ устойчивости решения к неточности оценок параметров
parameter_uncertainty: стандартное отклонение для каждого параметра
    """
n_simulations = 100
robustness_results = []

for _ in range(n_simulations):
        # Добавление шума к оптимальному решению
    noisy_solution = []
for param, std in zip(optimal_solution, parameter_uncertainty):
       if isinstance(param, int):
           # Для целых параметров - дискретный шум
       noisy_param = int(np.random.normal(param, std))
       noisy_param = max(1, min(10, noisy_param))  # Ограничение диапазона
            else:
                # Для вещественных параметров
                noisy_param = np.random.normal(param, std)
                # Ограничение диапазона
                noisy_param = max(50, min(200, noisy_param))

            noisy_solution.append(noisy_param)

        # Оценка зашумленного решения
    value, metrics, cost = simulation_wrapper(noisy_solution)

    robustness_results.append({
       'solution': noisy_solution,
       'value': value,
       'avg_wait': metrics['avg_wait'],
       'abandon_rate': metrics['abandonment_rate'],
       'cost': cost
})

    # Расчет метрик устойчивости
df = pd.DataFrame(robustness_results)

robustness_metrics = {
    'mean_performance_degradation': df['value'].mean() - simulation_wrap-
per(optimal_solution)[0],
    'std_performance': df['value'].std(),
    'sla_violation_rate': (df['avg_wait'] > 180).mean(),
    'cost_variability': df['cost'].std() / df['cost'].mean()
}

return robustness_metrics, df
```

Методика выполнения работы

Выбрать задание из предложенных проектов или согласовать новый проект с преподавателем. Подробно изучить программный код и протестировать его на компьютере. Задание можно выполнять группой по 2–4 студента, каждая группа берет один из примеров. При статистических расчетах использовать табличный процессор MS Office Excel/ LibreOffice.org Base, а для формирования текста – текстовый редактор: MS Word/

Writer LibreOffice.org, Google Docs. Заранее подготовить шаблоны для диаграмм (например, в Draw.io) и Jupyter Notebook с заготовками кода для анализа графов (NetworkX, вывод метрик, визуализация).

Рекомендации по обработке и оформлению полученных результатов

Подготовить отчет в PDF (8–12 страниц), содержащий титульный лист с Ф.И.О. и номером варианта, ответы на все задания, диаграммы, код Python (можно в виде скриншотов или в приложении). Исходные файлы: .py файл с кодом анализа графа, файлы диаграмм (.drawio, .xml или др.).

Презентация (5–7 слайдов) для краткой защиты, описание выбранной системы для моделирования, параметры, основные переменные модели, определить основные требования и оптимальные решения.

Вопросы для самоконтроля

1. Описать постановку задачи многокритериальной оптимизации. Указать, какие элементы обязательно входят в её формальное описание (множество альтернатив, критерии оценки, целевые направления критериев).

2. Объяснить концепцию Парето-оптимальности. Почему в многокритериальных задачах не существует единственного «лучшего» решения? Привести пример Парето-оптимального выбора?

3. Перечислить и кратко охарактеризовать основные методы свёртки критериев для сведения многокритериальной задачи к однокритериальной.

4. Как учитываются неопределённость и риск в задачах многокритериальной оптимизации?

5. Что такое функция полезности в контексте многокритериального выбора? Объяснить, как она позволяет формализовать предпочтения лица, принимающего решения.

6. Что такое система поддержки принятия решений, дать определение, указать основные функции и назначение. В каких случаях можно применять СППР? Указать в чем сходство и различие СППР и экспертных систем?

7. Роль систем поддержки принятия решений в многокритериальном выборе. Кратко охарактеризовать, какие функции выполняет СППР на этапе формирование набора альтернатив.

Рекомендуемая литература

1. Вдовин, В. М. Теория систем и системный анализ / В. М. Вдовин, Л. Е. Суркова, В. А. Валентинов. Москва : Дашков и К°, 2016. 644 с.

2. Вандер, П. Д. Python для сложных задач: наука о данных и машинное обучение / П. Д. Вандер. Санкт-Петербург : Питер, 2018. 576 с.

3. Хватов, А. А. Современные методы оптимизации с примерами на Python / А. А. Хватов, Н. О. Никитин, А. В. Калюжная. Санкт-Петербург : Университет ИТМО, 2023. 48 с.

4. Медведева, М. А. Системы поддержки принятия управленческих решений / М. А. Медведева, А. О. Коломыцева, А. Ю. Вишнякова, Е. А. Искра. Екатеринбург: Издательство УрФУ, 2019. 221 с.

Подготовка отчета по лабораторной работе

Отчет по лабораторной работе требуется представить в конце занятия или к следующему занятию по согласованию с преподавателем. Отчет принимается только в электронном виде. При подготовке отчета необходимо обратить внимание на применяемые государственные стандарты.

Требования к оформлению отчета

Отчет по лабораторной работе оформляется по стандартным требованиям, которые предусмотрены для реферата или курсовой работы.

Текст отчета должен быть набран в редакторе Microsoft Word / Writer LibreOffice при соблюдении следующих требований: шрифт Times New Roman, размер шрифта – 14 пт, межстрочный интервал – 1,15; размеры полей: левое – 30 мм, нижнее и верхнее – 20 мм, правое – 10 мм, абзацный отступ – 1,25 см, выравнивание текста по ширине.

Объем отчета по лабораторной работе 5–15 страниц.

Правила оформление листинга.

Оформление кода листинга. Шрифт Courier New (Consolas, Monaco, Liberation Mono), размер шрифта 10–12 пт (может быть меньше основного текста для компактности), межстрочный интервал одинарный.

Форматирование кода. Сохранять отступы и структуру исходного кода (иерархия циклов, условий). Использовать подсветку синтаксиса при печати. Разбивать длинные строки (если строка не помещается в ширину страницы).

Комментарии в коде. Оставлять комментарии на русском языке для ключевых блоков. Убирать отладочные комментарии и закомментированные участки кода.

Пример оформления листинга.

Листинг 1.1

Основной модуль программы

```
import numpy as np
from scipy.integrate import odeint

def model(y, t, params):
    # Система дифференциальных уравнений
    dydt = [y[1], -params['k'] * y[0]]
    return dydt
```

Структура отчета по лабораторной работе:

1. Титульный лист.
2. Цель и основные задачи лабораторной работы.
3. Программы и аппаратное обеспечение, используемые в работе.
4. Задание по лабораторной работе.
5. Ход выполнения лабораторной работы.
6. Формирование и описание результатов. Формулирование выводов.

Примерные вопросы к экзамену

1. Предпосылки развития системных представлений. Потребности научного познания. Системность живой и неживой природы. Назвать научно-методологические предпосылки развития системного анализа.

2. Охарактеризовать четыре типа вмешательств в ситуацию. Для каждого типа привести примеры ситуаций, в которых такое вмешательство будет наиболее эффективным.

3. Понятие модели. Моделирование. Классификации моделей. Компьютерное моделирование.

4. Отличие познавательной и прагматической моделей. Что необходимо сделать, чтобы перейти от словесных к математическим моделям? Охарактеризовать свойства модели: ингерентность, конечность, адекватность, истинность.

5. Указать причины построения дерева целей и дерева проблем. Чем отличается построение дерева проблем от диаграммы Исикавы?

6. Понятие системы. Основные свойства: иерархичность, целостность, эмерджентность, коммуникативность. Стратификация и страты. Статические и динамические системы.

7. Определение двух типов понятий системы: конструктивное и дескриптивное. Свойства систем: устойчивость, развитие, целенаправленность, управляемость.

8. Информационный подход к анализу систем. Энтропия, количество информации. Классификация систем.

9. Перечислить основные виды моделирования в естественных науках. Особенность математического моделирования и его связь с системным анализом.

10. Эмерджентность. Условия появления эффекта эмерджентности. Что называется отношением, связью, структурой? Соотношение мощности внутренних и внешних связей.

11. Пояснить следующие понятия: поведение, состояние, событие. Каким образом они отображаются в пространстве состояний? Что называется жизненным циклом?

12. Проблемы и способы их решения. Что такое проблемная ситуация? Назовите и охарактеризуйте четыре типа улучшающих вмешательств.

13. Классификация моделей. Языки описания моделей. Базовые модели систем. Модели «черного ящика» и «серого ящика». Модель состава, модель структуры.

14. Описать жизненный цикл системы. Дать краткую характеристику государственным стандартам, которые используются для описания жизненного цикла.

15. Моделирование как деятельность. Анализ и синтез как методы построения моделей. Аналитический подход.

16. Понятие функции системы. Основные свойства систем. Классификация функций сложной системы. Модели состава и структуры. Комбинирование базовых моделей систем. Взаимосвязь функций и структуры сложной системы.

17. Понятие модели. Моделирование. Познавательные и прагматические модели. Статические и динамические модели. Материальные модели и виды подобия. Соответствие между моделью и действительностью.

18. Измерение/оценивание систем. Дать характеристику основным типам измерительных шкал. Что общего и в чем различие между интервальной шкалой, шкалой отношений и абсолютной шкалой?

19. Анализ структурно-топологических характеристик информационных систем. Перечислить основные топологические характеристики для неориентированного графа.

20. Методы измерений/оценки в условиях определенности. Методы выявления предпочтений экспертов. Ранжирование, парное сравнение.

21. Измерение/оценивание систем. Методы интеграции измерений (свертки). Метод идеальной точки.

22. Методы измерений/оценки в условиях неопределенности. Критерий среднего выигрыша. Критерий Лапласа. Критерий максимина (Вальда).

23. Методы измерений/оценки в условиях неопределенности. Критерий максимакса. Критерий пессимизма-оптимизма (Гурвица). Нечеткие измерения.

24. Управление. Пять компонент управления. Семь типов управления.

25. Элементы теории принятия решений. Описание выбора: критериальный язык, язык бинарных отношений, функции выбора, групповой выбор.

26. Объяснить разницу между дискретно-событийным моделированием и системной динамикой. Для каких типов систем каждый подход более подходит? Привести примеры.

27. Выбор в условиях неопределенности. Оптимальность.

28. Декомпозиция/композиция систем. Принципы формирования и применения стандартных оснований декомпозиции. Методы композиции. Метод морфологического анализа.

29. Что такое имитационное моделирование? Основные этапы имитационного моделирования.

30. Декомпозиция/композиция систем. Метод формирования структуры целей и функций. Метод последовательного синтеза информационных технологий управления.

31. Базовая методология системного анализа. Этапы системного анализа. Анализ ситуации. Постановка целей. Выработка решений.

32. Базовая методология системного анализа. Методы организации экспертиз. Мозговая атака (мозговой штурм). Метод Дельфи.

33. Описать постановку задачи многокритериальной оптимизации. Указать, какие элементы обязательно входят в её формальное описание (множество альтернатив, критерии оценки, целевые направления критериев).

34. Методологии системного анализа. Эвристические приемы. Сущность структурного анализа. Методология IDEF0.

35. Функция полезности в контексте многокритериального выбора. Объяснить, как она позволяет формализовать предпочтения лица, принимающего решения.

36. Роль систем поддержки принятия решений в многокритериальном выборе. Охарактеризовать, какие функции выполняет СППР на каждом этапе решения задачи: формирование набора альтернатив, определение и взвешивание критериев, оценка альтернатив и построение модели предпочтений, анализ результатов (визуализация множества Парето, сценарный анализ).

37. Методологии системного анализа. Методологии построения дерева целей. Методология анализа иерархий.

38. Доминирование альтернатив и множество Парето. Объяснить, почему поиск решений Парето важен в многокритериальных задачах.

Глоссарий

Абстрактная модель – модель, полученная на основе теоретического анализа в виде дескриптивных соотношений. Упрощённое представление реальной системы, которое отражает её ключевые свойства и взаимосвязи между компонентами, но не воспроизводит все детали оригинала. Абстрактная модель строится на основе выделения существенных характеристик и отвлечения от несущественных.

Автоматизация управления – применение человеком или группой лиц технических средств при управлении техническими объектами, технологическими процессами или для обработки информации, используемой при принятии управленческих решений.

Адекватность – свойство модели: 1) модель считается адекватной, если с её помощью достигается цель. Правомерность применения модели для исследования задачи. 2) Отображение проблемной ситуации. В познавательных моделях адекватность – степень соответствия результатов, полученных с помощью модели, реальным свойствам и поведению исследуемой системы. Проверка адекватности заключается в доказательстве того, что точность результатов моделирования не хуже точности расчётов на основе экспериментальных данных.

Алгоритм – строго определённая последовательность действий (операций), выполнение которых по завершению приводит к достижению цели и позволяет преобразовать входные данные в требуемый результат при решении задачи анализа, проектирования или управления системой.

Альтернатива – представляет собой один из допустимых способов достижения поставленной цели, удовлетворяющий объективным законам природы и соответствующим предпочтениям лица, принимающего решения; вариант, одна из нескольких возможностей. На множестве альтернатив осуществляется выбор.

Анализ (англ. *analysis*, от греч. «разложение», «расчленение», «разбор») – 1) метод исследования: мысленное или реальное разделение целого на части; 2) процесс детального изучения какого-либо объекта, явления или ситуации через разбор его компонентов; 3) метод познания основанный на методе п. 1.

Аппроксимация (лат. *approximo*, «приближаюсь») замена одних математических объектов другими, в том или ином смысле близкими к исходным. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых и удобных объектов. Приближенное вычисление или выражение каких-либо величин через другие, более простые величины.

Аттрактор (англ. *attract*, «притягивать») – устойчивое состояние или траектория поведения системы, к которой она стремится со временем из разных начальных усло-

вий. В системном анализе аттрактор описывает «конечную точку» эволюции динамической системы – то, куда сходятся её возможные траектории в фазовом пространстве.

Вход (системы) – 1) связь системы с окружающей средой, которая представлена воздействием среды на систему и направлена от среды к системе; 2) то, что преобразуется системой в выход. По содержанию: входы могут быть физическими (материальными) или информационными (знания, информация, данные).

Выход (системы) – 1) связь системы с окружающей средой, которая представлена воздействием системы на среду и направлена от системы к среде; 2) продукт системы; то, во что преобразуются входы в систему. По содержанию: выходы могут быть физическими (материальными) или информационными (знания, информация, данные).

Валидация, аттестация (англ. *validation*) – подтверждение (на основе представления объективных подтверждений) того, что требования, предназначенные для конкретного использования или применения, выполнены (ГОСТ Р 7193-2016).

Верификация (англ. *verification*) – подтверждение (на основе предоставления объективных подтверждений) того, что заданные требования полностью выполнены (ГОСТ Р 7193-2016).

Выбор – 1) операция, относящаяся к целенаправленной деятельности и представляющая операцию по сужению множества альтернатив; 2) принятие решения.

Гипотеза – обоснованное предположение, утверждение о структуре, поведении, взаимосвязях или закономерностях системы, выдвигаемое для объяснения наблюдаемых явлений и требующее доказательства или проверки.

Голосование – способ выражения коллективного мнения, вынесения коллективного решения, осуществление коллективного выбора с помощью одного из мажоритарных правил (простого или абсолютного большинства, консенсуса и т.п.).

Граница системы – концептуально определённые пределы, отделяющие систему от её внешней среды. Она определяет, какие элементы и процессы считаются частью системы, а какие – внешними по отношению к ней.

Декомпозиция (от лат. *de* «удаление» и *compositio* «составление») – процесс (операция) разделения сложной системы на более простые, понятные и управляемые части (подсистемы, элементы, модули) для их детального изучения и последующего синтеза целостной модели.

Жизненный цикл системы – период времени от возникновения потребности системы, ее становления и далее до снижения эффективности, ликвидации системы.

Знак – 1) сигнал, имеющий конкретное значение, воспринимаемое человеком; 2) реальная модель абстрактного понятия.

Идентификация – построение математических моделей систем по результатам наблюдения входных и выходных переменных (сигналов) исследуемого объекта. Основная цель идентификации заключается в получении формального описания поведения системы, которое можно использовать для прогнозирования, управления и оптимизации.

Иерархия – 1) принцип организации, заключающийся в том, что целое рассматривается как состоящее из частей, каждая из которых сама является целым, состоящим из своих частей, и т.д.; 2) многоуровневая древовидная структура с отношением подчиненности сверху вниз.

Имитационная модель – динамическое представление системы путем продвижения ее от одного состояния к другому по характерным для нее операционным правилам. Имитационная модель по форме представляет собой компьютерную программу или алгоритм, которые воспроизводят поведение реальной системы во времени с учётом её структуры, взаимосвязей и случайных факторов.

Измерение – действие, связанное с сопоставлением определенного состояния объекта (предмета) или явления с некоторой выбранной шкалой. В результате измерения будет выбран символ на шкале, соответствующий и фиксирующий наблюдаемое состояние.

Исследование операций – дисциплина, охватывающая совокупность близких методов, в которых за основу исследования принимаются работы, выполняемые системой в процессе ее функционирования, а состояния системы и значения ее параметров рассматриваются как результаты выполняющихся работ.

Ингерентность – согласованность модели с окружающей культурной средой; принадлежность модели этой среде; 2) условия, необходимые для отражения и представления модельных свойств.

Искусственный интеллект (ИИ, англ. *artificial intelligence*, AI), в самом широком смысле интеллект, демонстрируемый машинами, в частности компьютерными системами. Это область исследований в компьютерных науках, в которой разрабатываются и изучаются методы, алгоритмы и программное обеспечение, позволяющие машинам воспринимать окружающую среду, а также использовать обучение и интеллект для выполнения действий, которые максимизируют их шансы на достижение определенных целей.

Информация – 1) сведения, знания, используемые данные в различных системах произвольной природы; 2) в научно-технических приложениях то, что несет на себе сигнал; 3) философская категория – всеобщее свойство материи, являющееся аспектом отражения, допускающим количественное описание.

Каузальное представление системы – описание системы в терминах влияния одних переменных на другие.

Кибернетика (от греч. *cybernetics*, «управлять») – наука об управлении в системах произвольной природы.

Классификация – разделение совокупности объектов на классы (группы) по некоторым наиболее существенным признакам (критериям). В системном анализе классификация служит инструментом упорядочения знаний о системах.

Конвергенция (от лат. *convergentio*, «сближаю») – процесс сближения, взаимопроникновения и объединения различных систем, подсистем или их элементов, приводящий к формированию общих свойств, структур или функций.

Конфигуратор – агрегат (совокупность) качественно различных языков описания системы, минимально необходимый для её полного и целостного представления в рамках заданной цели. Конфигуратор объединяет разные точки зрения на систему, позволяя описать её многогранность. Без конфигуратора анализ и проектирование сложных систем становятся неполными или ошибочными.

Критерий – мера, стандарт или правило, которые используются для оценки степени достижения цели, сравнения альтернатив и принятия решений в рамках системы. Критерий связывает цель с реальными результатами, позволяя количественно или качественно измерить эффективность системы.

Линейная система – система, которая описывается линейными уравнениями (алгебраическими, дифференциальными, интегральными и др.), в противном случае – нелинейная.

Модель – упрощенное подобие объекта, которое воспроизводит интересующие свойства и характеристики объекта оригинала или объекта проектирования.

Модель абстрактная – идеализированное представление системы, созданное средствами мышления и описания, которое отражает только существенные для конкретной цели свойства и взаимосвязи объекта-оригинала.

Модель динамическая – представление системы, отражающее изменение её состояния во времени под воздействием внутренних процессов и внешних факторов. Динамические модели показывают эволюцию системы – как она переходит из одного состояния в другое.

Модель знаковая – абстрактное представление системы, в котором объекты, их свойства и отношения между ними обозначаются с помощью знаков (символов) по заранее установленным правилам. Подобие между моделью и оригиналом устанавливается по соглашению (условно), а не через физическое сходство.

Модель классификационная – структурированное представление системы, в котором объекты (элементы системы) распределяются по классам (группам) на основе выбранных критериев (признаков). Цель классификации – упорядочить знания о системе, выявить закономерности и упростить работу с большим объёмом данных.

Модель концептуальная – абстрактное представление системы, фиксирующее её ключевые понятия (сущности, объекты), элементы и взаимосвязи между ними. Данная модель отражает смысловую структуру предметной области или конкретного объекта и служит основой для дальнейшего анализа и проектирования.

Модель математическая – формализованное представление системы с помощью математических понятий, символов и отношений (уравнений, функций, матриц, неравенств и т.д.), отражающее основные свойства и взаимосвязи элементов системы. Математическая модель позволяет количественно анализировать поведение системы, прогнозировать её состояние и оптимизировать параметры без прямого вмешательства в реальный объект.

Модель статическая – представление системы, фиксирующее её состояние в определённый момент времени без учёта изменений во времени. Она даёт «срез» или «фотографию» системы, отображая структуру, состав и взаимосвязи элементов на

конкретный момент времени. Среди параметров статической модели нет временного параметра – она не описывает процессы развития или функционирования системы.

Модель состава системы – представление системы, описывающее, из каких подсистем и элементов состоит система.

Модель структуры системы – представление системы, описывающее совокупность устойчивых связей между элементами системы, обеспечивающих её целостность и функционирование как единого целого.

Модель функциональная – представление системы, описывающее её функции (процессы, действия, преобразования), их взаимосвязи, входные/выходные потоки, управляющие воздействия и механизмы выполнения. Она отвечает на вопрос «Что делает система?» и фокусируется на процессах, а не на структуре или временных изменениях.

Модель «черного ящика» – представление системы, при котором её внутренняя структура и механизмы функционирования остаются неизвестными или намеренно не рассматриваются. Анализ и описание системы ведутся исключительно на основе наблюдения за входами (входными воздействиями) и выходами (выходными результатами), а также установлением зависимостей между ними. Модель абстрагируется от внутреннего устройства системы и фокусируется на поведении её входов и выходов.

Модель языковая – система знаков и правил их комбинирования, используемая для описания, анализа и коммуникации знаний о других системах. В системном анализе данная модель служит инструментом формализации представлений о структуре, функциях и поведении исследуемых объектов. Языковая модель оперирует символами (словами, терминами, обозначениями) и правилами их сочетания и используется для описания систем, передачи информации, для формирования логических выводов и прогнозирования поведения систем.

Метамодель – модель, описывающая структуру, правила и ограничения для построения других моделей (базовых моделей). Проще говоря, это «модель моделей»: она задаёт язык и правила, по которым создаются модели конкретной предметной области.

Моделирование – процесс создания упрощённых представлений (моделей) сложных систем для их изучения, анализа, прогнозирования поведения и оптимизации. В системном анализе моделирование позволяет исследовать систему без прямого вмешательства в её работу. Модель не является точной копией оригинала, а отражает только те свойства и связи, которые важны для решения конкретной задачи.

Морфологический анализ – метод системного исследования, позволяющий упорядочить процесс выдвижения и рассмотрения вариантов (альтернатив) решения сложной задачи. Он основан на систематическом переборе возможных комбинаций параметров альтернатив системы.

Надсистема – более широкая система, частью которой является рассматриваемая система (объект анализа). Она включает все внешние элементы и подсистемы, которые влияют на функционирование и существование анализируемой системы.

Неопределенность – неполнота, недостоверность, несвоевременность или неточность исходной информации о системе, её элементах, связях, внешней среде и условиях функционирования.

Неопределенность интервальная – частичное знание о величине параметра с указанием интервала возможных значений.

Неопределенность параметрическая – неизвестны или неточны постоянные параметры системы, но структура и связи известны.

Неопределенность структурная – недостаточная информация о структуре системы, её элементах и связях.

Неопределенность нечёткая – использование нечётких множеств и лингвистических переменных.

Неопределенность стохастическая – параметры описываются вероятностными распределениями.

Неопределенность цели – нечёткость или изменчивость целей системы.

Обратная связь – подача выходного параметра какого-либо элемента в неизменном или преобразованном виде на вход того же элемента.

Обучающая и проверяющая последовательности – последовательности фактически наблюдаемых точек, в которых определяется или проверяется адекватность параметров модели.

Обучение – процесс, способствующий самоприспособлению системы к внешней среде, соответствующий возможностям ее генетической структуры.

Окружающая среда (англ. *environment*) – совокупность всех внешних условий и факторов, которые воздействуют на рассматриваемую систему и с которыми она взаимодействует. Это надсистема по отношению к анализируемому объекту.

Открытая система – система, способная обмениваться с окружающей средой массой, энергией и информацией; система, способная к расширению за счет включения новых элементов.

Оптимальный – означает наилучший вариант среди допустимых при заданном критерии оптимальности (правиле выбора). Качество оценивается с помощью критерия оптимальности, а условия задаются в виде ограничений на дополнительные критерии. Оптимизация – центральная идея кибернетики.

Подсистема – часть системы (также является системой), выделенная по определённому признаку, обладающая некоторой самостоятельностью и допускающая разложение на элементы в рамках данного разложения. Различают подсистемы разных уровней.

Принятие решений – процесс выбора наилучшего варианта действий из множества возможных на основе анализа информации, ценностей и предпочтений лица, принимающего решение. Цель принятия решений – достичь поставленных целей в условиях ограниченных ресурсов и множества противоречивых факторов.

Проблема – субъективно воспринимаемое несоответствие между текущим состоянием системы (или среды) и желаемым состоянием, которое требует целенаправленного преобразования.

Проблематика – совокупность взаимосвязанных проблем в соседних объектах, которые неразрывно связаны с проблемой, подлежащей решению, и существующих в рамках системы и её взаимодействия с окружающей средой. Необходимость рассмотрения проблематики вытекает из того, что проблемосодержащая система сама состоит из подсистем и входит в надсистему, а устранение поставленной проблемы требует учета последствий для всех них.

Проблемная ситуация – ситуация, которая вызывает неудовлетворённость существующего положения у субъекта (лица, принимающего решения, или группы лиц), но не ясно, что следует изменить. Проблема еще не осознана и не поставлена.

Регулирование – процесс коррекции управляющих параметров системы на основе информации об её текущем состоянии и отклонении от желаемой траектории поведения. Цель регулирования – вернуть систему в устойчивое состояние или обеспечить движение к заданной цели.

Решение – набор управленческих воздействий (действий со стороны лица, принимающего решения) на объект (систему, комплекс и т.д.) управления, позволяющий привести данный объект в желаемое состояние или достичь поставленной перед ним цели. Это результат анализа, прогнозирования, оптимизации и выбора альтернативы из множества вариантов достижения конкретной цели.

Развитие – изменения процессов в системе во времени, выраженные в количественных, качественных и структурных преобразованиях от низшего (простого) к высшему (сложному).

Свойство – 1) качество (атрибут), которое постоянно присущее объекту; 2) абстрактное отношение данного объекта с другими объектами, «свернутое отношение», модель отношения.

Семантика – раздел семиотики, изучающий отношения между знаками и тем, что они обозначают.

Семиотика – наука, исследующая знаки и знаковые системы.

Сигнал – материальный носитель информации.

Синергетика – наука о самоорганизации систем. Новое междисциплинарное направление научных исследований, в рамках которого изучаются процессы перехода от хаоса к порядку и обратно (процессы самоорганизации) в открытых нелинейных средах самой различной природы.

Синтез (от греч. *synthesis* «соединение», «сочетание») – 1) процесс мысленного или реального соединение частей в единое целое с заданными свойствами и функциями; 2) метод познания, основанный на п. 1. Синтез противоположен анализу, если анализ разбивает систему на части, то синтез собирает целое из компонентов.

Система – средство достижения цели; совокупность взаимосвязанных элементов, образующих целостное образование с новыми (эмерджентными) свойствами, не сво-

димыми к свойствам отдельных компонентов. Основные особенности систем: целостность, обособленность от окружающей среды, наличие связи со средой, наличие частей и связей между ними.

Система большая – управляемая система, рассматриваемая как совокупность взаимосвязанных подсистем, объединённых общей целью функционирования. Термин введён в рамках кибернетики и системного подхода для описания сложных объектов большого масштаба. Отличие от обычной системы состоит в масштабе, сложности структуры и необходимости учёта множества разнородных факторов.

Система искусственная – система, созданная человеком как средство достижения поставленной цели.

Система поддержки принятия решений (СППР, англ. *Decision Support System, DSS*) – человеко-машинный информационно-вычислительный комплекс, помогающий лицам, принимающим решения (ЛПР), анализировать сложные ситуации и выбирать оптимальные варианты действий.

Система проблеморазрешающая – система, обладающая возможностями (ресурсами, компетенциями и т.д.), необходимыми для ликвидации проблемы.

Система проблемосодержащая – система, в которой возникла проблема, подлежащая решению; обычно эта система является инициатором и заказчиком системного анализа.

Система сложная – система, модель которой, используемая для управления системой, неадекватна заданной цели.

Система социотехническая (СТС) – система, объединяющая социальные (люди, их отношения, ценности, навыки) и технические (оборудование, технологии, инструменты) компоненты в едином процессе для достижения организационных целей. Эффективность системы достигается не за счёт максимизации отдельно технической или социальной подсистемы, а через гармонизацию их взаимодействия. Термин предложен в 1960 г. английским ученым Э. Тристом и австралийским психологом и социологом Ф. Эмери.

Системность – 1) обладание всеми признаками системы; 2) методологический подход к исследованию объектов, при котором они рассматриваются как целостные образования (системы), состоящие из взаимосвязанных элементов и взаимодействующие с внешней средой; 3) всеобщее свойство материи, форма её существования.

Системный анализ – научно-методологическая дисциплина, изучающая принципы, методы и средства исследования сложных объектов путём представления их в виде систем и последующего анализа структуры, функций, взаимосвязей и поведения этих систем. Системный анализ занимается проблемами принятия решений в условиях анализа большого количества информации различной природы и применяется при решении сложных слабоформализуемых проблем.

Системный подход – 1) направление методологии научного познания, в основе которого лежит рассмотрение объекта как целостного комплекса взаимосвязанных элементов (системы); 2) в прикладном аспекте это комплексный (междисциплинарный) подход, обобщающий методы, методики различных дисциплин, связанных с исследованием и проектированием сложных систем.

Сложность системная – характеристика системы, отражающая многообразие её элементов, связей, функций, состояний и динамики, а также трудности в описании, анализе, прогнозировании и управлении системой. Сложность проявляется как на структурном, так и на функционально-поведенческом уровне и зависит от способа восприятия и целей анализа.

Сложность структурная – свойство системы, которое отражает многоуровневый характер системы, многообразие компонентов и связей, сложность поведения и эффективного управления.

Сложность динамическая – сложность описания (дескриптивная), сложность предсказания поведения системы.

Сложность функциональная – определяется разнообразием выполняемых функций и их взаимосвязью.

Сложность информационная – связана с объёмом информации, необходимой для описания и управления системой.

Структура (от лат. *structura,* «строение, порядок связи») – совокупность устойчивых связей между элементами системы, которые обеспечивают её целостность, поддерживают тождественность системы самой себе при внешних и внутренних изменениях, определяют организацию элементов и характер их взаимодействия.

Сценарий – формализованное описание возможного пути развития системы или ситуации во времени, включающее: последовательность событий; причинно-следственные связи; условия перехода между состояниями; возможные исходы.

Управление – процесс целенаправленного воздействия на систему для обеспечения требуемого поведения или достижения заданных целей. С точки зрения системного анализа управление рассматривается как взаимодействие двух подсистем: управляющей (субъект управления) и управляемой (объект управления).

Устойчивость системы – свойство системы приходить к новому установившемуся состоянию после окончания переходного процесса, вызванного каким-либо возмущающим или управляющим воздействием.

Факторы – признаки, которые оказывают наиболее существенное влияние на поведение моделируемого объекта.

Функция выбора – операция над множеством альтернатив, результатом которой является подмножество предпочтительных (допустимых) вариантов.

Функция потерь – математическая функция, отображающая состояние системы или результат процесса в числовое значение, интерпретируемое как «стоимость», «штраф» или «ущерб» из-за отклонения от целевого состояния.

Цель – абстрактная модель желаемого (целевого) состояния системы, к которому она должна стремиться в процессе функционирования или развития.

Целевая функция – математическое выражение, формализующее критерий качества или эффективности системы. Она подлежит оптимизации (минимизации или максимизации) при заданных ограничениях для достижения целей системы.

Эксперт – квалифицированный специалист в исследуемой области.

Экспертные оценки – количественные и качественные оценки процессов и явлений, выполняемые экспертами на основе суждений.

Эвристика – 1) совокупность логических приемов и методических правил теоретического исследования и отыскания истины, выведенных эмпирически, полезность которых обоснована лишь тем, что они во многих случаях приводят к успеху; 2) эмпирическое правило, упрощающее или ограничивающее поиск решений в сложной предметной области.

Эвристические методы – методы решения задач, основанные на эвристике или эвристических рассуждениях, т.е. на использовании правил и приемов, обобщающих прошлый опыт и интуицию решающего.

Эквифинальность системы – способность системы достигать определенного состояния, которое не зависит ни от времени, ни от ее начальных условий, а определяется исключительно ее параметрами.

Экспертные методы – совокупность организационных, логических и математических процедур, направленных на получение информации от специалистов (экспертов), её анализ и обобщение для подготовки и выбора рациональных решений в условиях неопределённости, неполноты данных или невозможности полной формализации задачи.

Эмерджентность (от англ. *emerge*, «возникать», «появляться») – свойство системы, при котором у неё появляются качества или свойства, отсутствующие у отдельных элементов и не являющиеся простой суммой их свойств.

Библиографический список

Антонов, А. В. Системный анализ : учебник для вузов / А. В. Антонов. Москва : Высшая школа, 2004. 454 с.

Вандер, П. Д. Python для сложных задач: наука о данных и машинное обучение / П. Д. Вандер. Санкт-Петербург : Питер, 2018. 576 с.

Вдовин, В. М. Теория систем и системный анализ / В. М. Вдовин, Л. Е. Суркова, В. А. Валентинов Москва : Дашков и К°, 2016. 644 с.

Качала, В. В. Основы теории систем и системного анализа / В. В. Качала. Москва : Горячая линия Телеком, 2012. 210 с.

Клишин, А. П. Методы и средства проектирования информационных систем и технологий : лабораторный практикум / А. П. Клишин, Ф. Д. Пираков. Томск : Издательство Томского государственного педагогического университета, 2025. 115 с.

Коровин, А. М. Моделирование систем : учебное пособие к лабораторным работам / А. М. Коровин. Челябинск : Издательский центр ЮУрГУ, 2010. 47 с.

Лычкина, Н. Н. Имитационное моделирование экономических процессов : учебное пособие / Н. Н. Лычкина. Москва : ИНФРА-М, 2014. 254 с.

Медведева, М. А. Системы поддержки принятия управленческих решений / М. А. Медведева, А. О. Коломыцева, А. Ю. Вишнякова, Е. А. Искра. Екатеринбург : Издательство: УрФУ, 2019. 221 с.

Павловский, Ю. Н. Имитационное моделирование / Ю. Н. Павловский, Н. В. Белотелов, Ю. И. Бродский. Москва : Издательский центр «Академия», 2008. 236 с.

Перегудов, Ф. И. Основы системного анализа / Ф. И. Перегудов, Ф. П. Тарасенко. Томск : Изд-во НТЛ, 2001. 396 с.

Силич, М. П. Теория систем и системный анализ / М. П. Силич, В. А. Силич. Томск : Изд-во ТУСУР, 2013. 340 с.

Системный анализ и принятие решений : словарь-справочник : учебное пособие для вузов / под ред. В. Н. Волковой, В. Н. Козлова. Москва : Высшая школа, 2004. 616 с.

Сурмин, Ю. П. Теория систем и системный анализ / Ю. П. Сурмин. Киев : МАУП, 2003. С. 7–31.

Тарасенко, Ф. П. Прикладной системный анализ / Ф. П. Тарасенко. Москва : Изд-во Кнорус, 2010. 224 с.

Федотова, Д. Э. Case-технологии: Практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. Москва : Горячая линия-Телеком, 2005. 160 с.

Хватов, А.А. Современные методы оптимизации с примерами на Python / А. А. Хватов, Н. О. Никитин, А. В. Калюжная. Санкт-Петербург : Университет ИТМО, 2023. 48 с.

Приложение
Пример оформления титульного листа

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Томский государственный педагогический университет»
(ТГПУ)

Физико-математический факультет

Кафедра информатики

ОТЧЕТ
по лабораторной работе № ___
Наименование работы

Выполнил:
студент(ка) 423 гр., ФМФ
Иванов А.И.

Проверил:
к.ф.-м. н., доцент кафедры информатики
Петров М.С.

Томск 20____

Учебное издание

Андрей Петрович КЛИШИН, Наталия Леонидовна ЕРЁМИНА

СИСТЕМНЫЙ АНАЛИЗ

Лабораторный практикум

Электронное издание локального распространения

Ответственный за выпуск: Е. В. Гребенникова
Корректор: Ю. П. Готфрид
Технический редактор: А. И. Лелоюр
Дизайн обложки Н. В. Удлер

Подписано к использованию: 13.07.2026.
Гарнитура Times. Объем издания: 3,55 Mb.
Комплектация издания – 1 CD.
Тираж 100 CD. Заказ № 101/ЭУ.

Издательство
Томского государственного педагогического университета
634061, г. Томск, ул. Киевская, 60
тел. 8(3822)311-484
E-mail: izdatel@tspu.ru

ISBN 978-5-907791-72-5