МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

01.04.02 Прикладная математика и информатика

(код и наименование направления подготовки)

Математическое моделирование

(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему: «Сравнительный анализ языков программирования высокого уровня»

Обучающийся М.Ю. Барышной
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд. физ.-мат. наук, доцент Г.А. Тырыгина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Содержание

Введение

Актуальность темы исследования обусловлена тем, что возрастающая сложность программных продуктов и информационных технологий с уменьшением среднего срока жизненного цикла разработки требует адекватного, обоснованного и качественного выбора языка программирования высокого уровня в рамках реализации различных ИТ-проектов. Для повышения эффективности выбора необходим инструментарий поддержки, в том числе и инструментарий, основанный на математических моделях, что обеспечит высокий уровень достоверности результата и его обоснованность. На сегодняшний день представлено множество математических моделей для системы выбора и поддержки принятий управленческих решений в различных областях, все они имеют свои достоинства и недостатки и обеспечивают необходимый уровень достоверности. Следовательно, развитие системы анализа и выбора языков программирования высокого уровня, выстроенной на основе эффективных математических моделей, является актуальным направлением исследований в сфере прикладной информатики и математики.

Объект исследования: математические модели для принятия решения при проведении сравнительного анализа языков программирования высокого уровня.

Предмет исследования: процесс принятия решения о выборе на основе сравнительных характеристик эффективности языков программирования высокого уровня.

Цель исследования: разработка системы поддержки принятия решения о выборе в рамках сравнительного анализа языков программирования высокого уровня, основанной на математических моделях.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучить теоретические и методические аспекты проведения

сравнительного анализа языков программирования высокого уровня;

- изучить преимущества и недостатки языков высокого уровня, их характеристика и отличительные черты;
- исследовать математические модели, применимые для решения поставленной исследовательской задачи;
- разработать математическую модель для проведения сравнительного анализа и оценки языков высокого уровня;
- провести эксперимент с использованием математической модели для решения исследовательской задачи;
- оценить результаты моделирования.

Методы исследования: математическое моделирование, системный анализ, аналитические методы, экспертные методы, сравнение, синтез, анализ и обобщение и другие.

Гипотеза исследования состоит в том, что развитие системы поддержки принятия решения, основанная на математических моделях нечеткой логики позволит повысить эффективность выбора в рамках сравнительного анализа языков программирования высокого уровня, а также обеспечить обоснованность выбора и предотвратить ошибки на этапах проектирования и планирования ИТ-проектов по разработке программного продукта, исправление которых может повлечь финансовые потери до 80% от стоимости всего проекта.

Теоретико-методологическую основу исследования составили исследовательские работы ведущих отечественных и иностранных ученых. Методы и критерии сравнительного анализа языков программирования высокого уровня в своих работах рассматривают такие авторы, как Н. А. Борсук, Д. В. Мастыкаш, В.Ш. Кауфман, М. С. Коротков, Е. С. Заньков, В. Н. Сысоев и М. О. Ильин.

Проведении сравнительного анализа, основанного на построении математических моделей предлагают такие эксперты-исследователи, как Пупыкина А. А., Сатунина А. Е., А. В. Потудинский, А. П. Преображенский, А. В. Шпинев, В. С. Лучников, Т. П. Новикова, А.А. Акинин, Ю.С. Акинина, С.В. Тюрин и другие.

Прорабатывают отельные этапы математического моделирования при решении задач анализа и выбора языков программирования высокого уровня в своих работах, такие ученые, как: В. Н. Волкова, А. Е. Леонова, А.А. Самарский, А.П. Михайлов, Ю. С. Ризен.

Теоретико-методологическая база, использованная для разработки математической модели на основе нечеткой логики, основывается на трудах видных математиков, таких как Р. Каплан, К. Асаи, Д. Ватада, А.Б. Кригер, К.А. Аксенов, Е.М. Сафрыгина, Л.Г. Доросинский, А.И. Титов, А.Д. Хомоненко, Г.Э. Яхъяева и других. Их исследования предоставили прочный фундамент для понимания и дальнейшего развития концепций и методов нечеткой логики. Вклад этих ученых стал основополагающим для создания моделей, способных эффективно решать задачи в условиях неопределенности и нечётко определенных данных. Современные исследования продолжают развивать эту область, что позволяет нечеткой логике эволюционировать и находить новые приложения в различных сферах.

В настоящее время наблюдается активная интеграция математического аппарата нечеткой логики с интеллектуальными парадигмами, что способствует развитию таких технологий, как нечеткие нейронные сети и другие аспекты искусственного интеллекта. Этот процесс представляет собой одно из наиболее перспективных направлений в науке и технике. Совмещение нечеткой логики с нейронными сетями позволяет создавать модели, обладающие высокой адаптивностью и способностью обучаться на основании неточных и частично определенных данных, что существенно расширяет их

применение в различных областях, включая обработку естественного языка, прогнозирование и управление сложными системами.

Научная новизна магистерской диссертации состоит в применении математической модели нечеткой логики к решению задачи сравнительного анализа языков программирования высокого уровня, направленного на развитие систем принятия решений при выборе инструментов реализации ИТ-проектов по разработке программных продуктов.

Теоретическая значимость исследования заключается в расширении теоретических положений в направлении развития математического аппарата при решении задачи сравнительного анализа языков программирования высокого уровня.

Практическая значимость исследования состоит в возможности применения результатов моделирования при выборе инструментов реализации ИТ-проектов по разработке программных продуктов.

Достоверность и обоснованность результатов исследования подтверждается экспериментальным моделированием.

К научным результатам, выносимым на защиту, относятся:

– система параметров построения нечеткой математической модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня;
– математическая модель Мамдани для решения задачи сравнительного анализа языков программирования высокого уровня.

Структура и объем магистерской диссертации. Диссертация состоит из введения, трех разделов, общего заключения, выводов, списка литературы и приложений. Список литературы насчитывает 53 наименования.

1 Теоретические и методические аспекты проведения сравнительного анализа языков программирования высокого уровня

1.1 Обзор языков программирования высокого уровня, их достоинства и недостатки, в рамках постановки задачи анализа и выбора

На сегодняшний день существует множество языков программирования высокого уровня. Ежегодно они продолжают развиваться, появляются новые системы программирования, это связано с активным высокоскоростным развития сквозных информационных технологий цифровой экономики, таких как искусственный интеллект, интернет вещей, большие данные и других [3]. В рамках развития языков программирования необходимо понимать, что меняются как сами языки, так и технологии программирования, которые они реализуют, а также усложняются ИТ-проекты, основой реализации которых языки являются. Поэтому задача анализа и выбора языка программирования усложняется ежегодно. Не всегда выбирается правильный язык, подходящий для решения конкретных задач, что снижает эффективность реализации ИТ-проектов и других задач, решение которых связано с языками программирования высокого уровня [2] [40].

Выбор языка программирования высокого уровня, эффективного при создании программного ИТ-проекта, определяется рядом факторов, включая особенности языка и предметной области автоматизации, а также предпочтения заказчика и разработчика [1].

«Язык программирования - это формальный язык, предназначенный для написания компьютерных программ» [15]. Как отмечают С. В. Чернова и В. А. Ларина: «Язык программирования высокого уровня определяет набор лексических, синтаксических и семантических правил, которые, в свою

очередь, определяют внешний вид программы и действия, выполняемые исполнителем (обычно компьютером)» [13].

При проведении и анализе языков программирования высокого уровня очень важна область применения, как отмечают Н. А. Борсук и Д. В. Мастыкаш. «Областей применения языков программирования в современном мире предостаточно: научные приложения, бизнес-приложения, искусственный интеллект и другие сквозные информационные технологии и технологии Индустрии 4.0, системное программирование, веб-приложения и так далее» [14]. Поэтому первой задачей при выборе языка является перечисление основных требований к проекту, за которым следуют второстепенные требования [10].

«Высокоуровневый язык программирования предоставляет более абстрактные концепции и инструменты, позволяющие программистам сконцентрироваться на решении задач, а не на деталях работы компьютера. Они обычно написаны так, чтобы программный код был более понятен и читаем, что упрощает сопровождение и модификацию программ» [12]. Высокоуровневые языки программирования обычно поставляются с обширными библиотеками функций, что позволяет программистам использовать готовые решения для многих задач [5]. «Код, написанный на высокоуровневом языке программирования, часто можно переносить на различные операционные системы и аппаратные платформы без значительных изменений. Высокоуровневые языки программирования обычно предоставляют более удобные средства для работы с различными типами данных, такими как текст, числа, массивы, структуры и т.д. Многие из них обеспечивают автоматическое управление памятью, что упрощает процесс программирования и уменьшает риск утечек памяти и других ошибок» [24].

Рассмотрим также еще одно определение термина «Язык программирования высокого уровня». «Высокоуровневый язык программирования – это средство записи компьютерных программ,

обеспечивающее высокую скорость и удобство работы. Его отличительной чертой является абстракция. Другими словами, высокоуровневый язык программирования обеспечивает возможность введения смысловых конструкций, способных коротко описать форматы данных и операции с ними в тех случаях, когда описания на низкоуровневом языке (например, на машинном коде) будут сложными для восприятия и очень длинными» [16].

Первые высокоуровневые языки программирования создавались с целью предотвращения зависимости сути алгоритмов от платформы. «В этом случае платформенная независимость обеспечивается перекладыванием связей на инструментальные программы, которые осуществляют перевод текстов с высокоуровневых языков на машинный код. Инструментальные программы выступают своего рода трансляторами. Таким образом, высокоуровневые языки программирования облегчают выполнение сложных задач программирования и упрощают адаптацию программного обеспечения (ПО) [4]. Применение инструментальных трансляторов позволяет связать программы, написанные с использованием высокоуровневых языков и операционной системы (ОС) устройств и оборудования. В идеальном случае при использовании интерпретаторов не требуется модификация первоначального текста на языке программирования высокого уровня для всех видов платформ» [17] [19].

«Основным достоинством машинно-независимых языков программирования являются их простота и универсальность. Как следствие, значительно сокращается продолжительность написания кода и отладки. Одна и та же программа может быть выполнена на компьютерах разной архитектуры» [28]. Преимущества языков программирования высокого уровня приведены на рисунке 1.

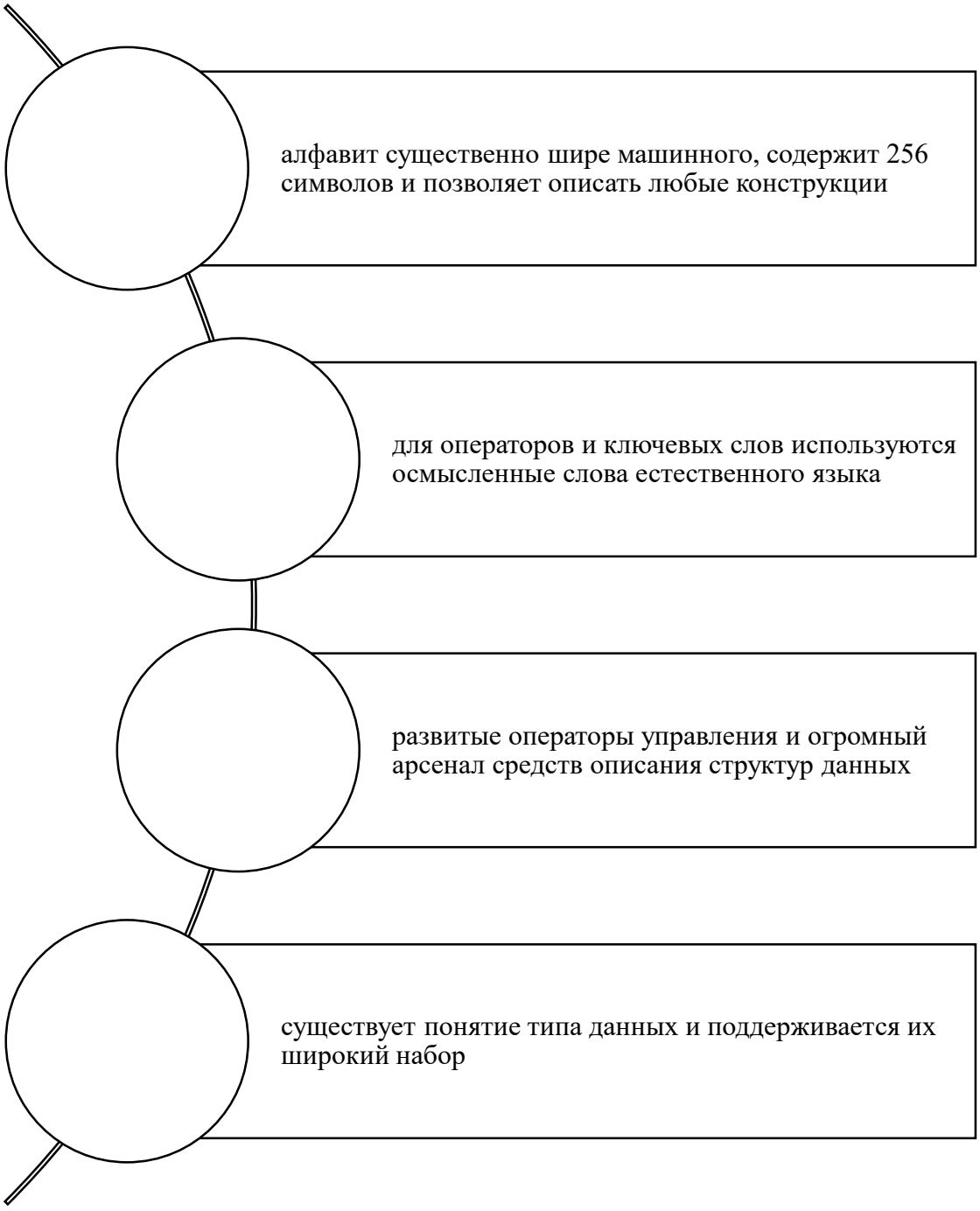


Рисунок 1 – Преимущества языков программирования высокого уровня [34]

Высокоуровневые языки программирования обладают значительными преимуществами, такими как удобство написания и отладки кода, а также высокая абстракция от аппаратных деталей. Однако эти преимущества сопровождаются одним из главных недостатков — большим размером создаваемых программ. Это объясняется тем, что высокоуровневые языки требуют дополнительных ресурсов для интерпретации и выполнения кода, что приводит к увеличению его объема. В связи с этим, несмотря на широкое распространение высокоуровневых языков, существует ряд областей, где использование низкоуровневых языков, таких как ассемблер, остается актуальным и даже необходимым. Например, при разработке компиляторов для высокоуровневых языков, драйверов, системного кода и программ для микроконтроллеров, где требуется максимально эффективное использование памяти и ресурсов процессора [8].

Основная область применения высокоуровневых языков программирования — это разработка программного обеспечения для компьютеров и различных устройств, обладающих значительными объемами памяти. Высокоуровневые языки позволяют разработчикам создавать сложные приложения с минимальными затратами времени и усилий на программирование и отладку. Однако при создании систем, где критичны производительность и компактность кода, низкоуровневые языки сохраняют свое значение. Они позволяют контролировать каждую операцию на уровне процессора и обеспечивать оптимальное управление ресурсами системы. Различия между этими категориями языков программирования становятся особенно заметны при анализе таких параметров, как производительность, размер кода, удобство разработки и уровень абстракции [23] [38]. Параметры, по которым имеют существенные отличия языки высокого уровня от низкоуровневых языков программирования представлены на рисунке 2.

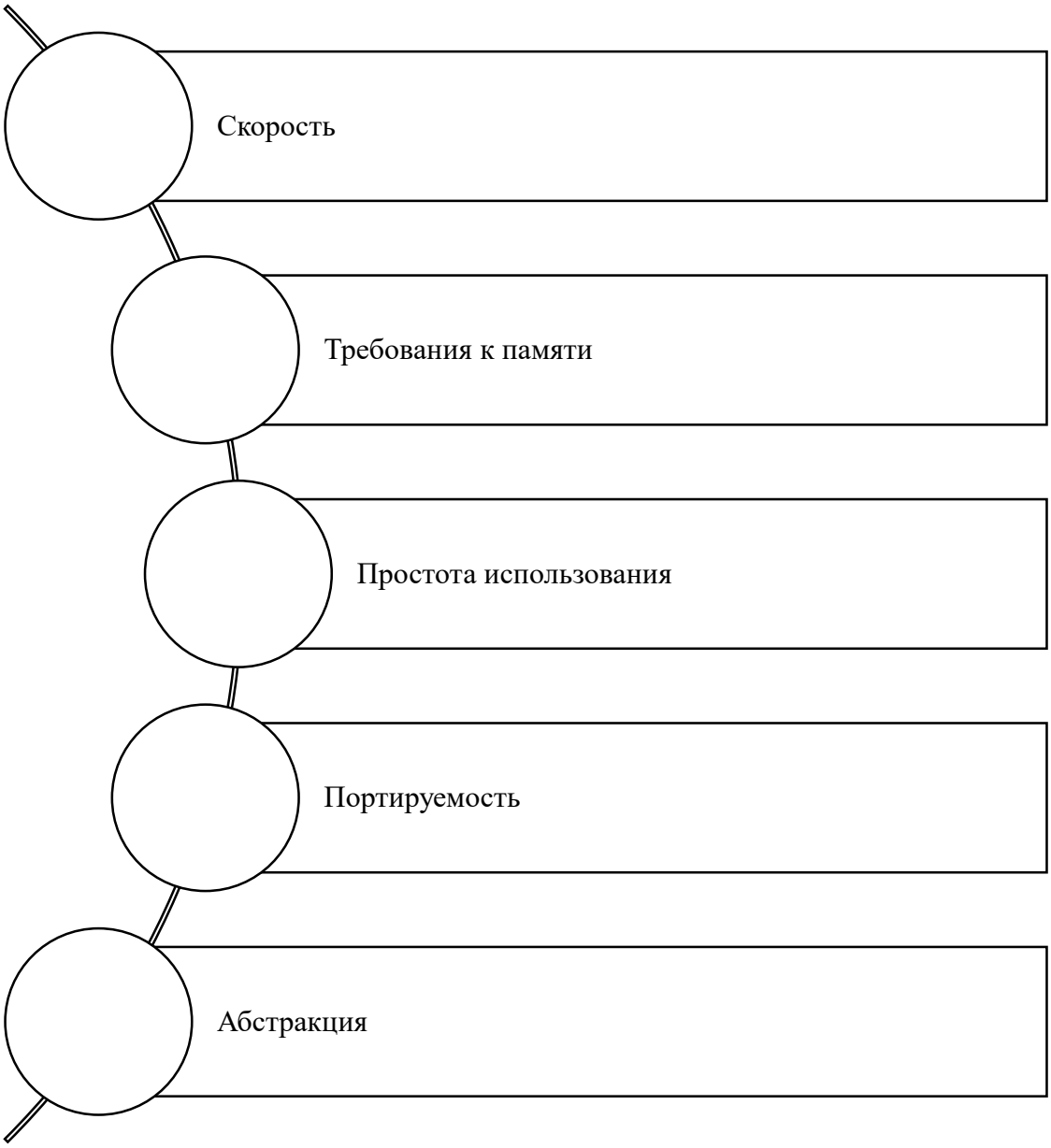


Рисунок 2 – Параметры, по которым имеют существенные отличия языки высокого уровня от низкоуровневых языков программирования [14]

Существует множество классификаций языков программирования высокого уровня по разным параметрам. На рисунке три представлена одна из классификаций языков программирования высокого уровня, которой придерживаются практики в области разработки программного обеспечения (рисунок 3).

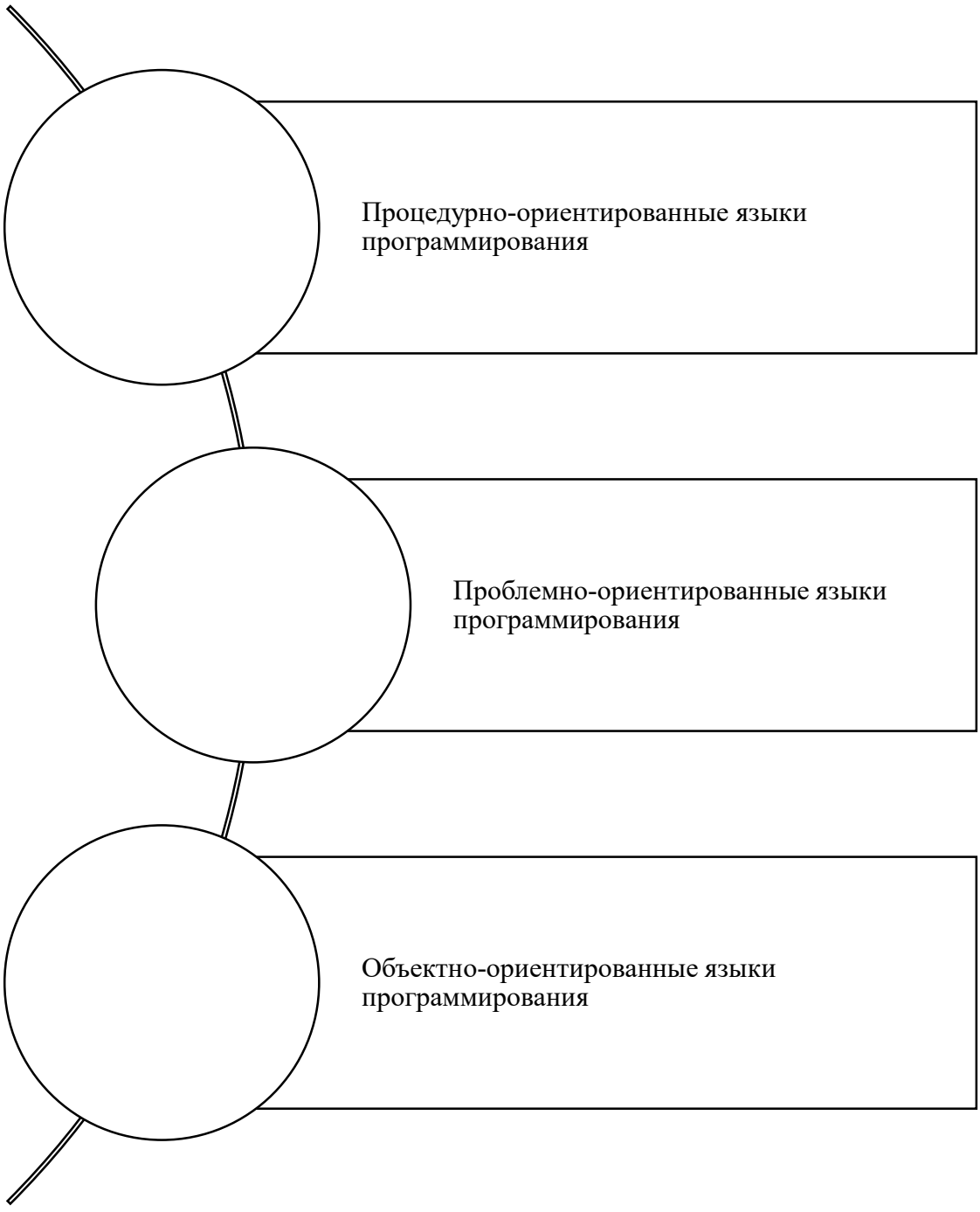


Рисунок 3 – Одна из классификаций языков программирования высокого уровня [14]

Объектно-ориентированные языки программирования высокого уровня появились по итогам развития процедурных языков программирования. «Концепция объектно-ориентированных языков программирования заключается в представлении программы, как совокупности объектов. В структурных языках основную роль играет логика, понимание последовательности действий. В объектно-ориентированных языках важно взаимодействие друг с другом объектов, логика работы каждого объекта не важна. Языки объектно-ориентированного программирования позволяют программисту использовать не только существующие классы, но и создавать пользовательские. Таким образом можно соперничать с проблемно-ориентированными языками программирования в решении прикладных задач. Использование классов лучше структурирует программу и значительно сокращает ее размер. Ни одна большая программа не может быть написана без использования объектно-ориентированного программирования» [26].

Объектно-ориентированное программирование (ООП) продолжает оставаться важнейшей парадигмой в разработке программного обеспечения, обеспечивая структурированный и модульный подход к созданию сложных систем. Языки программирования, такие как C++, C#, и JavaScript, остаются лидерами в этой области, благодаря своей гибкости, мощности и широкому спектру применения [41]. Однако, помимо них, существует множество других высокоуровневых языков, которые также поддерживают ООП и находят своё применение в различных сферах программирования. Эти языки, включая Java, Delphi, Eiffel, и Python, предоставляют разработчикам разнообразные инструменты и возможности для реализации объектно-ориентированных концепций, таких как наследование, полиморфизм и инкапсуляция [44].

Каждый из этих языков обладает уникальными характеристиками и особенностями, что позволяет выбирать наиболее подходящий инструмент для конкретных задач. Например, Delphi и Object Pascal исторически использовались для быстрой разработки настольных приложений, в то время

как Python известен своей простотой и широким применением в научных вычислениях и веб-разработке [6] [11]. Языки, такие как Ruby и Smalltalk, акцентируют внимание на чистоте объектной модели, тогда как ActionScript и JScript .NET обеспечивают возможности ООП в контексте веб-разработки и работы с мультимедийным контентом [20]. Таким образом, разнообразие объектно-ориентированных языков программирования позволяет разработчикам выбирать наиболее подходящий инструмент в зависимости от требований проекта и личных предпочтений, что способствует более эффективной и продуктивной работе в области программирования. Далее представлен перечень языков программирования высокого уровня из категории объектно-ориентированных, в том числе:

- C#;
- C++;
- Java;
- Delphi;
- Object Pascal;
- Visual DataFlex;
- Perl;
- PowerBuilder;
- Python;
- ActionScript (3.0);
- JavaScript;
- JScript .NET;
- PHP и другие.

Для проведения анализа и выбора языка программирования высокого уровня (ЯПВУ) и выделения характеристик (критериев и показателей) рассмотрим некоторые языки программирования высокого уровня, их функционал, достоинства и недостатки (рисунок 4) [42], [43].

Python	Один из востребованных языков программирования среди разработчиков из-за его синтаксической простоты и универсальности. Python используется для машинного обучения разработчиками, поскольку он менее сложен по сравнению с C и Java. Python - переносимый язык, он может использоваться на платформах Linux, Windows, Mac OS и UNIX. Он привлекателен благодаря своим функциям, таким как интерактивность, интерпретируемость, модульность, динамичность, переносимость и высокоуровневость, что делает его более уникальным, чем Java. Python - это многопарадигмальная программа, поддерживающая объектно-ориентированные, процедурные и функциональные стили программирования. Python поддерживает нейронные сети и разработку решений NLP благодаря своей простой библиотеке функций и более удобной структуре. Поддерживает тестирование алгоритмов без необходимости их применения. Python развивается быстрее, чем Java и C. В отличие от C и Java, Python работает с помощью интерпретатора, который замедляет компиляцию и выполнение. Программы на Python работают медленнее, чем программы на Java, но при этом на их разработку уходит гораздо меньше времени. Программы на Python обычно в 3-5 раз короче эквивалентных программ на Java. Подобное различие объясняется встроенными в Python типами данных высокого уровня и его динамической типизацией. Например, разработчик Python не тратит время на объявление типов аргументов или переменных. Мощный полиморфный список Python и тип словаря, для которых богатая синтаксическая поддержка встроена непосредственно в язык, находят применение почти в каждой программе Python. Из-за типизации во время выполнения, среда разработки Python нагружается больше, чем Java. Например, при оценке выражения a + b оно должно исследовать объекты a и b, чтобы найти их тип, который не известен во время компиляции. Затем вызывается соответствующая операция сложения, которую можно перегрузить с помощью пользовательского метода. Java может выполнять сложение целых чисел, он требует объявления переменных для a и b и не допускает перегрузки для оператора + для объектов, определяемых пользователем классов. По этим причинам Python гораздо лучше подходит в качестве «связующего» языка.

Рисунок 4 - Сравнение языков программирования и сопоставление их функционала (начало)

C #	Самый быстрый компьютерный язык. Его скорость важна для разработчиков, которые больше всего ценят время. Он обеспечивает быстрое выполнение и меньшее время загрузки, поэтому применяется в поисковых системах и при разработке компьютерных игр. C # позволяет широко использовать алгоритмы и эффективно использовать статистические методы искусственного интеллекта. C поддерживает повторное использование программ в разработке за счет наследования и сокрытия данных, что эффективно с точки зрения экономии времени и затрат. C подходит для машинного обучения и нейронных сетей. Данный язык программирования хорошо применим для поиска решений сложных проблем. Он богат библиотечными функциями и набором программных инструментов. C - это многопарадигмальная программа, поддерживающая объектно-ориентированные принципы, полезные для получения организованных данных. В то же время C плохо справляется с многозадачностью. Данный язык программирования подходит только для применения ядра или базы данных определенных систем или алгоритмов. Подход снизу вверх чрезвычайно сложен и затрудняет для начинающих разработчиков. Код Python зачастую в 5-10 раз короче эквивалентного кода C. К примеру, разработчик Python может выполнить за два месяца то, что два разработчика C # не смогут выполнить за год.

Рисунок 4 - Сравнение языков программирования и сопоставление их функционала (продолжение)

Java	Язык программирования с несколькими парадигмами, который следует объектно-ориентированным принципам и принципу Once Written Read / Run Anywhere (WORA). Это язык программирования искусственного интеллекта, который может работать на любой поддерживающей его платформе без необходимости перекомпиляции. Большая часть синтаксиса заимствована из C. Java подходит не только для NLP и поисковых алгоритмов, но и для нейронных сетей. В отличие от C , Java прост в использовании. Имеет автоматический менеджер памяти, облегчающий работу разработчиков. Java медленнее C, имеет более низкую скорость выполнения и большее время отклика. Java портативен, но старые платформы потребуют изменений программного и аппаратного обеспечения.

Рисунок 4 - Сравнение языков программирования и сопоставление их функционала (окончание)

В условиях постоянно развивающихся технологий, повсеместного внедрения искусственного интеллекта, инноваций, развития цифровой экономики и необходимости ускорения цифровой трансформации в различных отраслях возрастает сложность программных продуктов, меняются технологии программирования, требуется постоянное расширение функционала и сокращение среднего срока жизненного цикла разработки.

В этих условиях необходим точный, адекватный, быстрый и качественный выбор языка программирования высокого уровня в рамках реализации различных информационных и инновационных проектов цифровизации и цифровой трансформации, проектов, основанных на современных сквозных технологиях [21].

Для повышения эффективности выбора необходим инструментарий поддержки, в том числе и инструментарий, основанный на математических моделях, что обеспечит высокий уровень достоверности результата и его обоснованность. Следовательно, развитие системы анализа языков программирования высокого уровня и выбранная тематика является актуальным практико-научным направлением [9].

В данном параграфе выполнен обзор языков программирования высокого уровня, выделены их достоинства и недостатки, в рамках постановки задачи анализа и выбора языков программирования высокого уровня. Далее перейдем к рассмотрению проблемы выбора критериев показателей эффективности языка программирования высокого уровня в условиях сравнительного анализа на основе математического моделирования.

1.2 Определение проблемы и выбора критериев показателей эффективности языка программирования высокого уровня в условиях сравнительного анализа на основе математического моделирования

На сегодняшний день существует множество языков программирования высокого уровня. Их количество растет, изменяются технологии программирования, интерфейсы, развиваются сами языки программирования. Следовательно, должны совершенствоваться и инструменты поддержки эффективного выбора языка программирования высокого уровня для реализации конкретных задач и проектов.

Эффективность выбора оптимального языка программирования высокого уровня определяется рядом характеристик (показателей), как количественных, так и качественных. При анализе и оценке языков программирования высокого уровня учитываются такие характеристики, как [17]:

- особенность языка;
- предметная область;
- требования к языку заказчиков и тех, кто реализует проекты разработки на данном языке программирования.

Чаще всего, как отмечает Кауфман В.Ш., языки программирования рассматриваются в соответствии с общими критериями. Например, М. С. Коротков, Е. С. Заньков, В. Н. Сысоев и М. О. Ильин рекомендуют для сравнительного анализа выбирать следующие критерии:

- «удобочитаемость (насколько сложно будет прочитать код);
- простота (насколько легко будет работать и использовать библиотеки);
- ортогональность (управляющие операторы и структуры данных языка могут быть выражены с использованием относительно

небольшого числа элементарных конструкций);

- GUI (графический интерфейс пользователя): насколько легко будет работать с графическим приложением;
- кроссплатформенность (легко запускать программу на разных платформах)» [22].

С ними согласны С. В. Дедов, О. Д. Кирсанов, О. Ю. Тимошевская и другие [25], [28].

Для сравнительного анализа очень часто выбирают такие языки программирования высокого уровня, которые наиболее конкурентоспособны на ИТ-рынке, в том числе [29], [32]:

- C++;
- C#;
- Java;
- JavaScript;
- Python.

Это набор наиболее конкурентоспособных языков программирования, который далее будет использоваться для работы с математическими моделями.

С этим согласные многие современные исследователи языков программирования, в том числе: Т. А. Коваленко, А. Ю. Лобачев, И. В. Шишова, М. М. Абдусаламов, Т. В. Аветисян, А. М. Бородай [18], [49].

При проведении сравнительного анализа и построения математических моделей анализа многие исследователи предлагают определять весовые коэффициенты каждого выбранного критерия, чтобы понять, какой из них более важен [12], [35]. Этого мнения придерживаются такие эксперты, как Пупыкина А. А., Сатунина А. Е., А. В. Потудинский, А. П. Преображенский, А. В. Шпинев, В. С. Лучников, Т. П. Новикова, А.А. Акинин, Ю.С. Акинина, С.В. Тюрин и другие.

«Весовые коэффициенты критериев или показателей определяются экспертным методом и основаны на использовании обобщенного опыта и интуиции специалистов-экспертов. В экспертном методе оценка уровня важности конкретного свойства определяется в безразмерных единицах» [1], [3].

В соответствие с рекомендациями, изложенными такими авторами, как: В. Н. Волкова, А. Е. Леонова, Самарский А.А., Михайлов А.П., Ризен, Ю. С., дальше применяется процедура ранжирования. Объекты анализа – языки программирования высокого уровня - расположены в порядке их предпочтения (ранжирования). «Место, занимаемое при таком расположении в ранжированном ряду, называется рангом. Наиболее важному критерию, по мнению эксперта, присваивается наивысший балл, всем остальным в порядке уменьшения их относительной важности – баллы до 1.

Полученные результаты измерений нормализуются (делятся на общее количество баллов). Весовые коэффициенты, полученные таким образом, принимают значения от 0 до 1, и их сумма становится равно 1» [1], [5].

При обработке результатов, полученных с помощью ранжирования, необходимо выполнить следующие операции:

1. Определить сумму баллов, набранных всеми экспертами по определенному критерию (показателю).

2. Определить сумму баллов по всем критериям, установленным всеми экспертами.

3. Определение весового коэффициента j-го критерия.

В соответствии с суммой рангов каждого объекта экспертизы строится ранжированный ряд, который является результатом многократных измерений. Определяется вес серии.

В рамках развития процедуры сравнительного анализа языков программирования высокого уровня необходимо использовать систему

количественных критериев или показателей, которая является основной частью инструментария для сравнения на основе математических моделей.

Система критериев (показателей) должна включать несколько классов показателей, отражающих количественные и качественные характеристики сравнительного анализа языков программирования высокого уровня.

В основу системы количественных показателей могут лечь сравнительные характеристики, анализ по которым приведен в таблице 1. В рамках анализа и выбора языка программирования высокого уровня можно проводить сравнительный анализ следующих языков по выделенным характеристикам (показателям) (таблица 1), как один из вариантов:

Таблица 1 - Анализ языков программирования Java, C# и Python на основе выделенных характеристик

Сравнительные характеристики	Язык программирования высокого уровня		
	Java	C#	Python
Встроенная поддержка объектно-ориентированного подхода	+	+	+
Обеспечение возможности компиляции в машинный код	–	–	+
Встроенная поддержка обработки исключений	+	+	+
Обеспечение статической типизации	+	–	–
Обеспечение кроссплатформенности	+	+	+
Возможность подключения существующих в организации готовых решений	–	–	+

Также при проведении анализа и оценки языков при разработке системы показателей могут быть использована система внешних характеристик (показателей) качества программной разработки, которая отображена графически на рисунке 5.

Рисунок 5 - Система внешних характеристик качества программной разработки

А также система внутренних качеств (показателей), которая включает такие, как:

- «сопровождаемость (Maintainability) - набор атрибутов, влияющих на усилия, необходимые для внесения определённых изменений;
- тестируемость (Testability) - набор атрибутов, влияющих на усилия, необходимые для проверки программного обеспечения после проведения какого-либо видоизменения» [25].

Выделенные семь характеристик являются ключевыми индикаторами, которые определяют качество программного обеспечения. Эти характеристики охватывают такие аспекты, как функциональность, надежность, удобство использования, эффективность, сопровождаемость, переносимость и производительность. Комплексный учет этих аспектов позволяет не только оценить текущее состояние программного продукта, но и прогнозировать его поведение в различных условиях эксплуатации. Это делает данные характеристики универсальным инструментом для анализа и сравнения различных программных решений, обеспечивая объективность и всесторонность оценок.

В контексте сравнительного анализа языков программирования высокого уровня, эти характеристики могут служить основой для разработки системы критериев, которые помогут выявить сильные и слабые стороны каждого языка. Такой подход способствует выявлению оптимальных решений для конкретных задач, что, в свою очередь, повышает эффективность разработки и использования программного обеспечения. В качестве частных показателей, которые также предлагается включить в систему критериев или показателей можно также использовать показатели, представленные на рисунке (рисунок 6).

В свою очередь, каждый из частных показателей зависит от множества разнообразных характеристик.

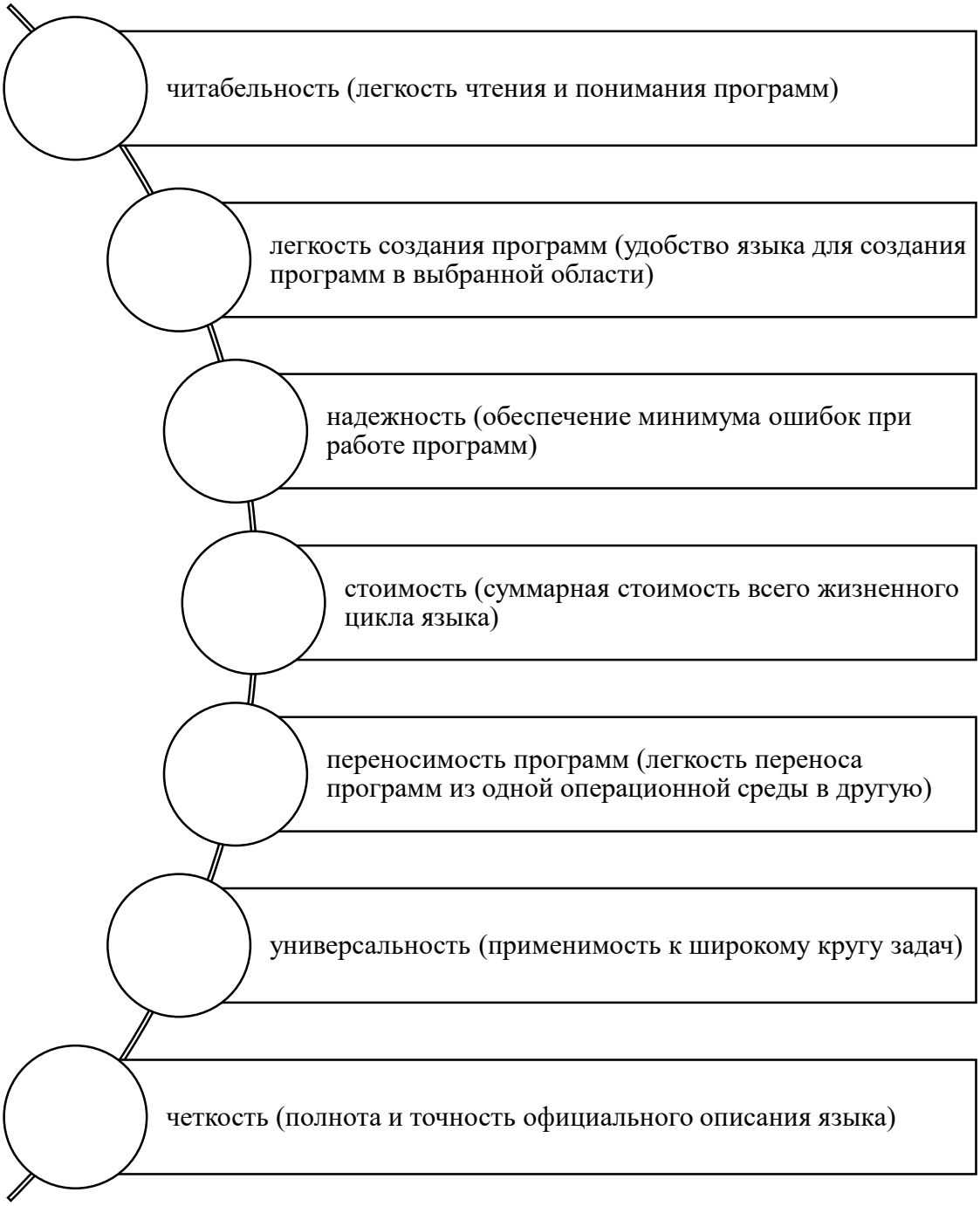


Рисунок 6 - Система частных показателей для включения в систему показателей сравнения языков программирования

Для сравнительного анализа в рамках исследования предлагается взять за основу критерии рисунка 6, а также добавить такие критерии, как:

- читабельность (насколько сложно будет читать код);
- простота (насколько легко будет работать и пользоваться библиотеками);
- ортогональность (управляющие операторы и структуры данных языка могут быть выражены с помощью относительно небольшого числа элементарных конструкций);
- GUI (графический интерфейс пользователя): насколько легко будет работать с графическим приложением;
- кроссплатформенность (простота запуска программы на разных платформах).

При разработке математической модели в рамках развития системы поддержки принятия решения при проведении сравнительного анализа языков программирования высокого уровня будут использоваться следующие критерии (показатели):и читабельность, ортогональность, простота, GUI, надежность, стоимость, переносимость программ, универсальность, кроссплатформенность, четкость [45].

Далее необходимо будет определить весовые коэффициенты каждого критерия, чтобы понять, какой из них более важен. Весовые коэффициенты в данном исследовании определяются экспертным методом и основаны на использовании обобщенного опыта и интуиции специалистов-экспертов. В экспертном методе оценка уровня важности конкретного свойства определяется в безразмерных единицах.

Процедура ранжирования выполняется в соответствие со следующим алгоритмом (рисунок 7).

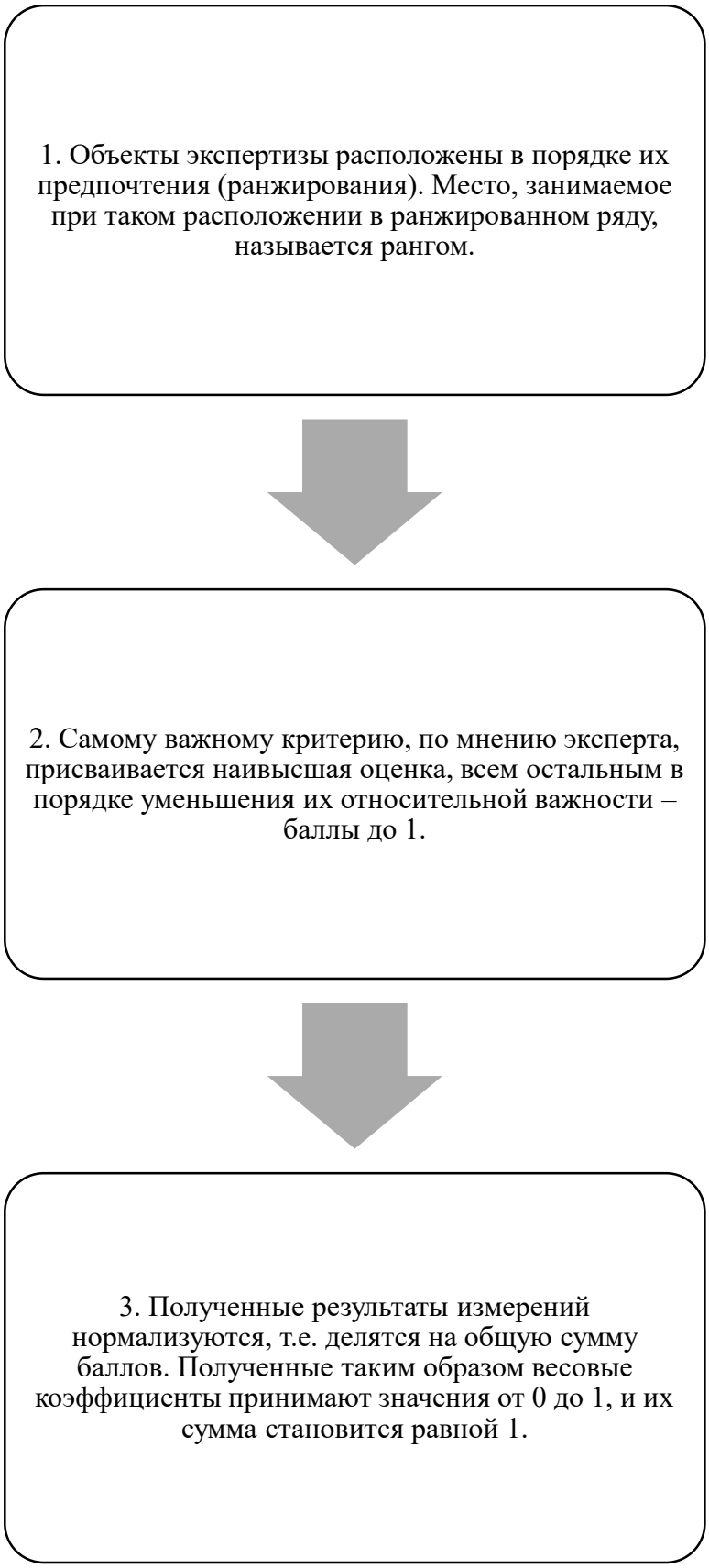


Рисунок 7 - Процедура ранжирования

Теоретическое исследование, проведенное в рамках магистерской диссертации, позволило глубоко изучить методологические аспекты сравнительного анализа языков программирования высокого уровня. В результате были выявлены ключевые преимущества и недостатки различных языков, что обеспечило понимание их характеристик и отличительных черт. Такой подход способствовал всестороннему рассмотрению предметной области и позволил четко определить параметры, влияющие на выбор языка программирования в различных контекстах [27].

Проведенное исследование систематизировало знания о языках программирования, создавая основу для дальнейших эмпирических исследований и практических рекомендаций. Подробный анализ позволил структурировать информацию и предложить методологические подходы к проведению сравнительного анализа, что, в свою очередь, способствует более осознанному и обоснованному выбору языков программирования для решения конкретных задач. Это исследование имеет значительный вклад в теоретическую базу, формируя основу для разработки новых методик и инструментов для анализа языков программирования высокого уровня. Далее перейдем к анализу математических моделей, применимых для решения поставленной исследовательской задачи.

2 Исследование математических моделей, применимых в рамках сравнительного анализа языков программирования высокого уровня

2.1 Балльная математическая модель

Рассмотрим ряд базовых математических моделей, которые обычно используются при сравнении и анализе различных систем. При принятии решения о выборе того или иного языка программирования высокого уровня для реализации того или иного проекта процесс принятия управленческого решения необходимо выражать не только в качественных характеристиках, но и в количественных показателях (критериях). Эти количественные показатели можно охарактеризовать как показатели эффективности работы или применения того или иного языка программирования высокого уровня [7].

Определение количественных критериев (показателей) применимости или эффективности применения того или иного языка основывается на применении математических моделей. Наиболее простой математической моделью, которую можно адаптировать под поставленную исследовательскую задачу является балльная модель. Данную математическую модель можно применить и в рамках решения задачи по проведению сравнительного анализа языков программирования высокого уровня [37].

Сущность этой модели заключается в следующем (рисунок 8).

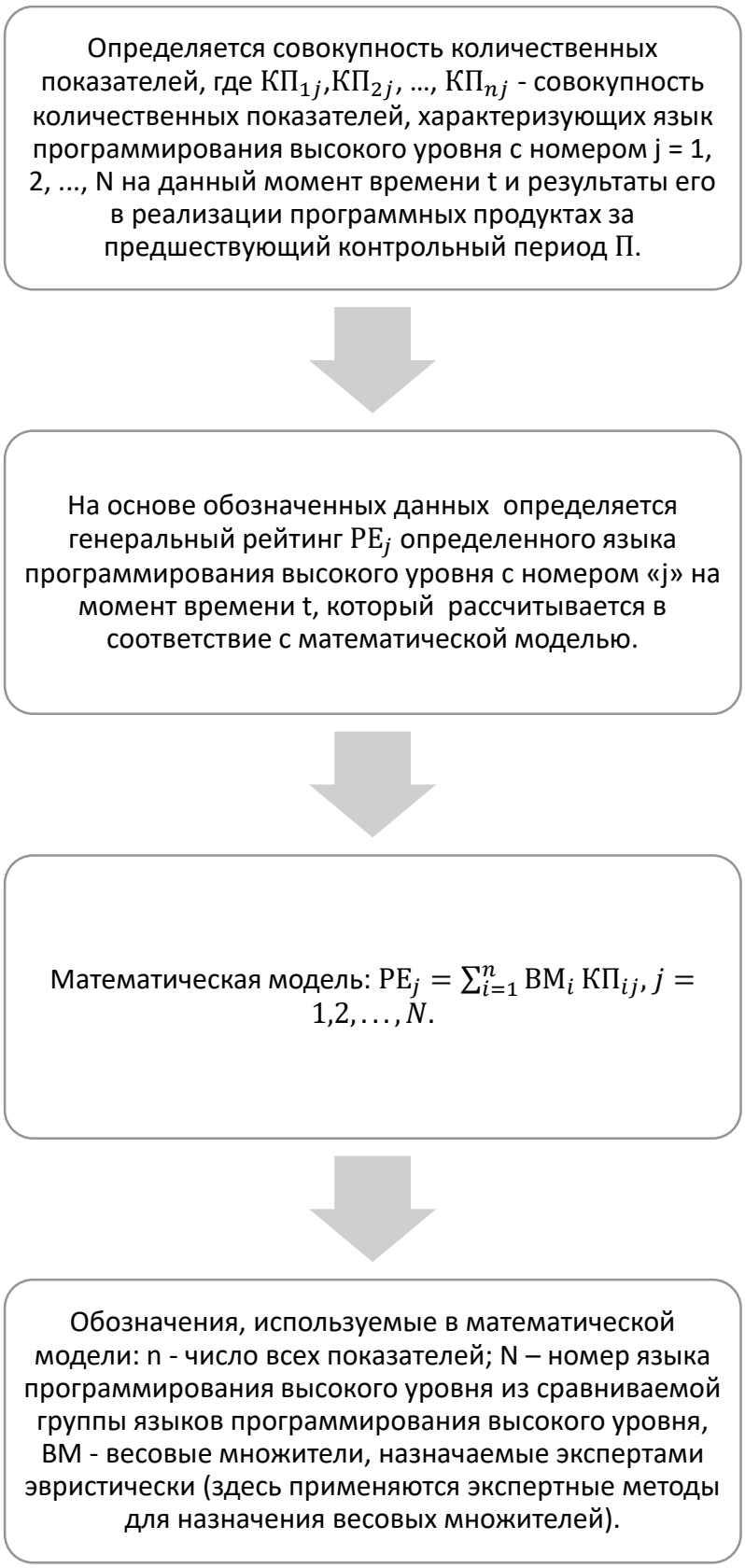


Рисунок 8 – Сущность балльной модели в рамках развития сравнительного анализа языков программирования высокого уровня

У балльной модели есть определенные достоинства и недостатки, которые приведены на рисунке 9 [37].

Рисунок 9 – Достоинства и недостатки балльной модели для сравнительного анализа языков программирования высокого уровня

Как мы видим, балльная математическая модель является достаточно простой и обеспечивает низкую трудоемкость решения исследовательской задачи, однако ряд недостатков, которые приведены на рисунке 9 ставят под вопрос применение данной модели в качестве инструмента развития системы принятия решения при проведении сравнительного анализа языков программирования высокого уровня.

2.2 Нормативно-классификационная и эталонная математические модели

Некоторым развитием балльной математической модели являются нормативно-классификационная и эталонная модели расчета рейтингов (рисунок 10).

Рассмотрим характеристики нормативно-классификационной модели расчета рейтингов [7], «в которой совокупность исходных показателей состояния и активности деятельности заданного объекта разделяется на два множества показателей: во-первых, показатели потенциальных возможностей, характеризующие состояние и потенциальные возможности выполнения различных видов деятельности; во-вторых, показатели активности или результативности, характеризующие результаты функционирования данного объекта за предшествующий период планирования» [37].

«Показатели первого множества разделяются на некоторое количество классов, каждый из которых имеет специфические признаки, а показатели второго множества подразделяются на некоторое количество видов деятельности. Далее показатели потенциальных возможностей и видов деятельности каждой из групп нормируются относительно некоторых величин, имеющих экономический смысл. Для каждой группы потенциальных возможностей и вида деятельности экспертным методом определяются весовые коэффициенты и осуществляется вычисление рейтингов потенциалов

по различным видам потенциальных возможностей и рейтингов активности (результативности) по различным видам деятельности» [34].

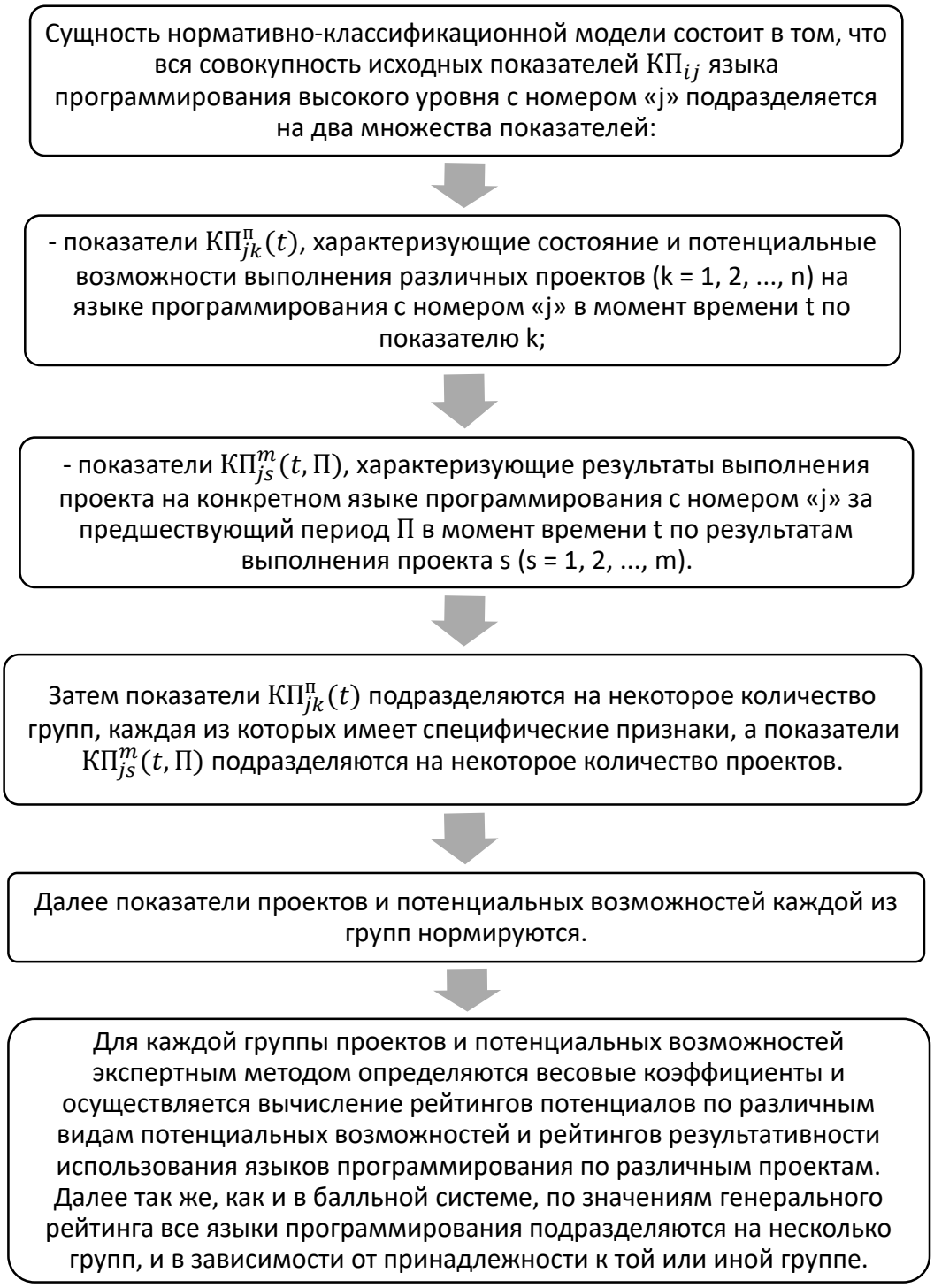


Рисунок 10 – Сущность нормативно-классификационной модели расчета рейтингов в рамках развития сравнительного анализа языков программирования высокого уровня

«Эталонная модель управления большими системами управления была предложена профессором В.И. Чернецким для синтеза самонастраивающихся (адаптирующихся) систем автоматического управления сложными многоконтурными техническими объектами» [37]. Модель такого типа «предполагает выбор некоторого идеального (гипотетического, эталонного) объекта управления, значения параметров которого являются оптимальными в том или ином смысле (это предположение не всегда практически реализуемо). Затем значения контролируемых параметров всех других объектов управления, относящихся к тому же типу, что и эталонный объект, оцениваются по отклонениям (или отношениям) от значений параметров эталонного объекта. На основе результатов сравнения фактических и эталонных значений контролируемых параметров синтезируется дополнительный контур управления (контур адаптации и самонастройки), основной целью которого является последовательное изменение некоторых технологических параметров, определяющих динамику движения объекта в необходимом направлении. При использовании основной идеи принципа эталонных моделей рейтинговые модели управления большими системами, по существу, являются математическими моделями синтеза дополнительных контуров адаптации большой системы управления, обеспечивающих достаточно высокую эффективность их функционирования в сложных динамически изменяющихся или непредсказуемых (неопределенных) условиях при наличии субъективных (человеческих) факторов, определяемых, в частности, множеством неконтролируемых воздействий» [37].

В рамках исследовательской задачи по развитию системы принятия решения при проведении сравнительного анализа языков программирования высокого уровня эталонные модели достаточно сложно использовать, так как исходная информация не является достаточно четкой и отсутствуют эталонные языки программирования, параметры которых могли бы быть использованы при использовании данной математической модели.

2.3 Математические модели, построенные на аппарате нечеткой логики

Далее рассмотрим математические модели, построенные на аппарате нечеткой логики, которые применяют в условиях неопределенности результата применения того или иного языка программирования высокого уровня при реализации ИТ-проектов по разработке программных продуктов.

«Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 г. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов» [43].

Прежде чем нечеткий подход к моделированию сложных систем получил признание во всем мире, прошло не одно десятилетие с момента зарождения теории нечетких множеств. И на этом пути развития нечетких систем принято выделять три периода.

«Первый период (конец 1960-х–начало 1970 годов) характеризуется развитием теоретического аппарата нечетких множеств (Л. Заде, Э. Мамдани, Беллман). Во втором периоде (1970–80-е годы) появляются первые практические результаты в области нечеткого управления сложными техническими системами (парогенератор с нечетким управлением). Одновременно стало уделяться внимание вопросам построения экспертных систем, основанных на нечеткой логике, разработке нечетких контроллеров. Нечеткие экспертные системы для поддержки принятия решений находят широкое применение в медицине и экономике [44]. В третьем периоде, который длится с конца 80-х годов и продолжается в настоящее время, появляются пакеты программ для построения нечетких экспертных систем, а

области применения нечеткой логики заметно расширяются. Она применяется в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других» [39].

«Отдаваться предпочтение нечеткой логики по миру началось после доказательства в конце 1980-х гг. Бартоломеем Коско знаменитой теоремы FAT (Fuzzy Approximation Theorem). В бизнесе и финансах нечеткая логика получила признание после того, как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных фаззи-применений в настоящее время исчисляется тысячами» [47].

Алгоритмы нечеткого вывода различаются, главным образом, видом используемых правил, логических операций и разновидностью метода дефаззификации. Разработаны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото [48], [49].

Рассмотрим подробнее нечеткий вывод на примере механизма Мамдани (Mamdani) [48]. Это наиболее распространенный способ логического вывода в нечетких системах. В нем используется минимаксная композиция нечетких множеств. Данный механизм включает в себя следующую последовательность действий, которая представлена на рисунке (рисунок 11).

Рисунок 11 – Система нечеткого логического вывода

Процедура фаззификации является начальным этапом обработки данных в нечеткой логике, где определяются степени истинности для предпосылок каждого правила. Это достигается путём применения функций принадлежности, которые оценивают, насколько данный элемент соответствует нечётким множествам. В процессе нечёткого вывода эти степени истинности используются для установления уровней отсечения, определяющих значимость каждого правила. Затем вычисляются усеченные функции принадлежности, отражающие откорректированные степени истинности, соответствующие данным уровням отсечения [45].

На этапе композиции происходит объединение усеченных функций принадлежности. Это осуществляется с использованием максимальной композиции нечетких множеств, что позволяет получить обобщённую оценку выходных значений. Заключительный этап - дефаззификация - направлен на преобразование нечётких данных обратно в четкие значения. Существует несколько методов дефаззификации, среди которых метод среднего центра или центроидный метод является наиболее распространённым. Этот метод вычисляет центр тяжести объединённой функции принадлежности, что позволяет получить конечное четкое значение, представляющее наиболее вероятный результат на основании первоначальных нечётких данных [53].

Методы нечеткой логики являются одним из активно развивающихся направлением. Математические модели, построенные на базе аппарата нечеткой логики, подходят для решения задач в неопределенных условиях, в которых реализуются ИТ-проекты, поэтому применения аппарата нечеткой логики для решения исследовательской задачи магистерской диссертации будет достаточно эффективным.

«В результате объединения нескольких технологий искусственного интеллекта появился специальный термин — «мягкие вычисления» (soft computing), который ввел Л. Заде в 1994 году. В настоящее время мягкие вычисления объединяют такие области как: нечеткая логика,

искусственные нейронные сети, вероятностные рассуждения и эволюционные алгоритмы. Они дополняют друг друга и используются в различных комбинациях для создания гибридных интеллектуальных систем.

Влияние нечеткой логики оказалось, пожалуй, самым обширным. Подобно тому, как нечеткие множества расширили рамки классической математическую теорию множеств, нечеткая логика вторглась практически в большинство методов Data Mining, наделив их новой функциональностью. Ниже приводятся наиболее интересные примеры таких объединений.

Нечеткие нейронные сети (fuzzy-neural networks) осуществляют выводы на основе аппарата нечеткой логики, однако параметры функций принадлежности настраиваются с использованием алгоритмов обучения НС. Поэтому для подбора параметров таких сетей применим метод обратного распространения ошибки, изначально предложенный для обучения многослойного персептрона. Для этого модуль нечеткого управления представляется в форме многослойной сети. Нечеткая нейронная сеть, как правило, состоит из четырех слоев: слоя фаззификации входных переменных, слоя агрегирования значений активации условия, слоя агрегирования нечетких правил и выходного слоя. Наибольшее распространение в настоящее время получили архитектуры нечеткой НС вида ANFIS и TSK. Доказано, что такие сети являются универсальными аппроксиматорами» [28].

Быстрые алгоритмы обучения и интерпретируемость накопленных знаний — эти факторы сделали сегодня нечеткие нейронные сети одним из самых перспективных и эффективных инструментов мягких вычислений [46].

Классические нечеткие системы имеют ограничение, заключающееся в необходимости участия экспертов для определения правил и функций принадлежности, что нередко представляет собой значительную сложность. Эксперты должны обладать глубокими знаниями в конкретной предметной области, чтобы сформулировать адекватные правила, описывающие

поведение системы. Однако доступность таких специалистов не всегда гарантирована, и их привлечение может быть затруднено из-за различных факторов, включая ограниченные ресурсы или отсутствие необходимых компетенций. Это обстоятельство существенно снижает эффективность классических нечетких систем, поскольку их настройка и корректировка становятся трудоемкими и затратными процессами.

Адаптивные нечеткие системы предлагают эффективное решение данной проблемы. В таких системах процесс подбора параметров нечеткой системы автоматизирован и осуществляется на основе обучения с использованием экспериментальных данных. Это позволяет существенно уменьшить зависимость от человеческого фактора и обеспечить более гибкую и точную настройку системы. В процессе обучения адаптивная нечеткая система самостоятельно корректирует свои параметры, основываясь на анализе поступающей информации, что значительно повышает ее адаптивность и применимость в различных условиях [50]. Таким образом, адаптивные нечеткие системы представляют собой более прогрессивный и эффективный подход к обработке нечеткой информации, обеспечивая высокую степень автоматизации и минимизируя необходимость привлечения узкопрофильных экспертов. Алгоритмы обучения адаптивных нечетких систем относительно трудоемки и сложны по сравнению с алгоритмами обучения нейронных сетей, и, как правило, состоят из двух стадий:

– Генерация лингвистических правил;
– Корректировка функций принадлежности.

«Первая задача относится к задаче переборного типа, вторая — к оптимизации в непрерывных пространствах. При этом возникает определенное противоречие: для генерации нечетких правил необходимы функции принадлежности, а для проведения нечеткого вывода — правила. Кроме того, при автоматической генерации нечетких правил необходимо обеспечить их полноту и непротиворечивость.

Значительная часть методов обучения нечетких систем использует генетические алгоритмы. В англоязычной литературе этому соответствует специальный термин — Genetic Fuzzy Systems. Значительный вклад в развитие теории и практики нечетких систем с эволюционной адаптацией внесла группа испанских исследователей во главе с Ф. Херрера (F. Herrera).

Нечеткие запросы к базам данных (fuzzy queries) — перспективное направление в современных системах обработки информации. Данный инструмент дает возможность формулировать запросы на естественном языке, например: «Вывести список недорогих предложений о съеме жилья близко к центру города», что невозможно при использовании стандартного механизма запросов. Для этой цели разработана нечеткая реляционная алгебра и специальные расширения языков SQL для нечетких запросов. Большая часть исследований в этой области принадлежит западноевропейским ученым Д. Дюбуа и Г. Праде.

Нечеткие ассоциативные правила (fuzzy associative rules) — инструмент для извлечения из баз данных закономерностей, которые формулируются в виде лингвистических высказываний. Здесь введены специальные понятия нечеткой транзакции, поддержки и достоверности нечеткого ассоциативного правила. Нечеткие когнитивные карты (fuzzy cognitive maps) были предложены Б. Коско в 1986 г. и используются для моделирования причинных взаимосвязей, выявленных между концептами некоторой области. В отличие от простых когнитивных карт, нечеткие когнитивные карты представляют собой нечеткий ориентированный граф, узлы которого являются нечеткими множествами. Направленные ребра графа не только отражают причинно-следственные связи между концептами, но и определяют степень влияния (вес) связываемых концептов» [48].

Активное использование нечетких когнитивных карт (НКК) в качестве инструмента моделирования систем обусловлено рядом значительных

преимуществ. Одним из основных достоинств является возможность наглядного представления исследуемой системы, что способствует лучшему пониманию и анализу сложных взаимосвязей между различными элементами системы. Визуализация причинно-следственных связей между концептами позволяет исследователям и практикам легко интерпретировать и объяснять динамику системы, что делает НКК незаменимыми в различных областях знаний, включая управление, экономику и экологию [52]. Однако, несмотря на очевидные преимущества, процесс построения НКК сталкивается с существенными проблемами. Одной из главных трудностей является отсутствие строгой формализации, что затрудняет систематическое создание карт и делает этот процесс зависимым от субъективных оценок экспертов.

«В связи с этим, одной из ключевых задач становится проверка адекватности построенной когнитивной карты реальной моделируемой системе. Это требует разработки методов, способных обеспечить объективность и надежность карт, чтобы они действительно отражали функционирование системы. Одним из перспективных подходов для решения этой задачи являются алгоритмы автоматического построения НКК на основе данных. Эти алгоритмы используют выборки данных для генерации карт, что позволяет минимизировать влияние субъективных факторов и обеспечить более точное соответствие карт реальности. Таким образом, автоматизация процесса построения НКК не только упрощает создание моделей, но и повышает их точность и достоверность, что открывает новые возможности для их применения в различных областях.

Нечеткие методы кластеризации, в отличие от четких методов (например, нейронные сети Кохонена), позволяют одному и тому же объекту принадлежать одновременно нескольким кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более «естественна», чем четкая, например, для объектов, расположенных на границе кластеров. Наиболее

распространены: алгоритм нечеткой самоорганизации c-means и его обобщение в виде алгоритма Густафсона-Кесселя» [32], [48].

Список можно продолжить и дальше: нечеткие деревья решений, нечеткие сети Петри, нечеткая ассоциативная память, нечеткие самоорганизующиеся карты и другие гибридные методы.

В рамках раздела были рассмотрены математические модели, которые можно было бы использовать для решения задачи сравнительного анализа языков программирования высокого уровня. Определено, что для разработки системы принятия решений будут использованы математические модели, построенные на аппарате нечеткой логики. Нечеткий подход к моделированию сложных систем получил признание во всем мире. В условиях неопределенности реализации ИТ-проектов с использованием того или иного языка программирования высокого уровня, для решения поставленной задачи целесообразно применить инструменты, основанные на нечеткой логике. В том числе необходимо задействовать математические модели нечетких множеств для принятия решения по выбору языка программирования. Это позволит оперировать количественными и качественными критериями (показателями) в условиях отсутствия достоверных статистических данных. Нечетко-множественный метод представит полный спектр возможных вариантов развития событий в отличие от вероятностно-статистических способов с конечными множествами сценариев.

3 Разработка математической модели для решения задачи по проведению сравнительного анализа языков высокого уровня

3.1 Разработка математической модели Мамдани для решения задачи исследования

Разработка математической модели поддержки принятия решения при сравнительном анализе и выборе языка программирования высокого уровня выполнялась на основе алгоритма, показанного на рисунке 12.

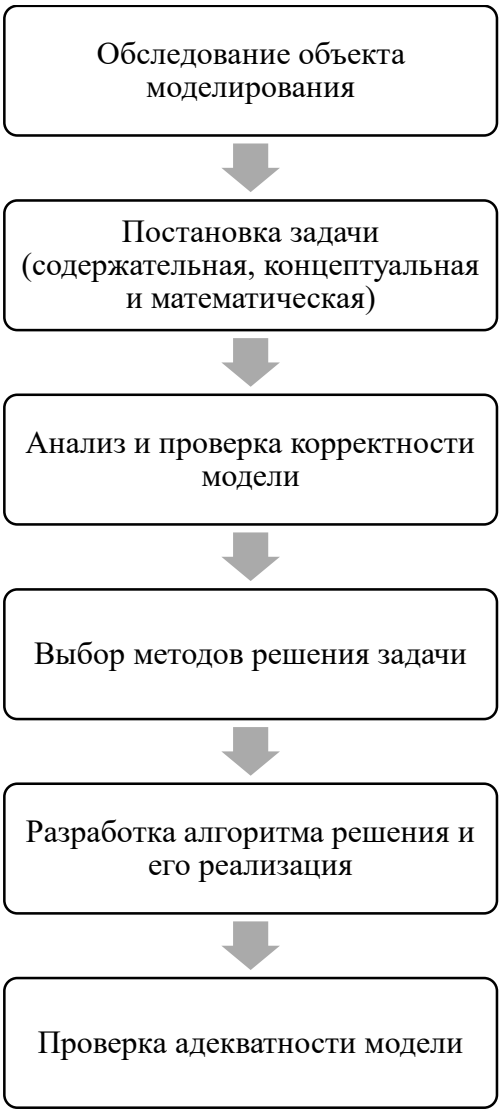


Рисунок 12 – Алгоритм разработки математической модели для решения исследовательской задачи

Это классический подход к разработке математической модели, на котором будет основано дальнейшее моделирование [30]. В разрабатываемой модели будет реализован алгоритм Мамдани, который был рассмотрен в предыдущем параграфе. Реализация самой модели Мамдани для решения задачи сравнительного анализа языков программирования высокого уровня и система параметров построения нечеткой математической модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня будет основываться на системе критериев, рассмотренных в первом разделе магистерской диссертации. Разработка алгоритма решения и его реализация в рамках апробации модели будут выполняться в Microsoft Excel, что позволит доказать эффективность работы математической модели Мамдани для решения задачи исследования.

В рамках моделирования работы нечеткого вывода предлагается использовать алгоритм принятия решения на основе использования методов нечеткой логики (таблица 2).

Согласно таблице 2, алгоритм принятия управленческих решений на основе методов нечеткой логики является гибким инструментом, который позволяет учитывать неопределенность и неполноту информации при принятии сложных стратегических и тактических решений в сфере управления ИТ-проектами по разработке программных продуктов и может быть использован для решения задачи сравнительного анализа языков программирования высокого уровня. Предлагаемый алгоритм позволяет эффективно моделировать и анализировать различные факторы, которые могут влиять на результат принятия решения, и учитывать множество переменных одновременно. Далее перейдем к математическому моделированию задачи сравнительного анализа языков программирования высокого уровня на основе модели Мамдами.

Таблица 2 - Алгоритм принятия управленческих решений на основе методов нечеткой логики (составлено автором)

Этап	Описание
1	2
1. Определение целей и задач	В этом шаге определяются цели и задачи, которые должны быть достигнуты при принятии решения. Цели могут быть как краткосрочными (например, повышение производительности), так и долгосрочными (например, увеличение прибыли)
2. Сбор данных	В этом шаге происходит сбор данных, которые могут быть использованы для принятия решения. Данные могут включать в себя информацию о производственных показателях, финансовых показателях, клиентской базе и т.д.
3. Определение нечетких переменных и функций принадлежности	В этом шаге определяются нечеткие переменные, которые будут использоваться для моделирования нечеткости и неопределенности. Нечеткая переменная представляет собой характеристику или параметр, который может принимать различные значения на основе определенной функции принадлежности
4. Определение правил вывода	В этом шаге формулируются правила вывода на основе знаний и экспертного опыта. Правила вывода являются логическими высказываниями, которые связывают условия (входные данные) с результатами (выходными данными). Каждое правило имеет вид «ЕСЛИ [условие], ТО [результат]»
5. Вычисление степени принадлежности для каждого правила	В этом шаге используются функции принадлежности для вычисления степени принадлежности каждого входного значения к определенной нечеткой переменной
6. Агрегация результатов	В этом шаге происходит агрегация результатов, полученных на предыдущем шаге, с целью получения окончательного вывода. Агрегация может осуществляться различными способами, например, с помощью операции максимума или среднего значения
7. Принятие решения	В этом шаге на основе агрегированных результатов принимается управленческое решение. Решение может быть принято на основе определенной стратегии или критериев оценки
8. Оценка и корректировка решений	В этом шаге происходит оценка принятого решения и необходимость его корректировки на основе новой информации или изменения в условиях окружающей среды

Рассмотрим моделирование работы нечеткого вывода по подтверждению гипотезы магистерской диссертации.

3.2 Моделирование работы нечеткого вывода

В рамках использования метода нечеткой логики разработаем математическую модель для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня, где сначала определим исходные множества M (X, Y) и N (Y, Z), где X – ИТ-проект разработки программного продукта; Y – критерии выбора языков программирования высокого уровня; Z – языки программирования высокого уровня.

Далее определим содержание множеств, где : X = (x1, x2, x3, x4, x5): ИТ-проект 1, ИТ-проект 2, ИТ-проект 3, ИТ-проект 4, ИТ-проект 5. Все ИТ-проекты по разработке программных продуктов отличаются по содержанию и точно не определены характеристики языка программирования для реализации проектов, поэтому надо провести сравнительный анализ и отобрать языки программирования, которые будут использоваться для решения задач конкретного проекта.

Y = (y1, y2, y3, y4, y5, y6, y7, y8, y9, y10): читабельность, ортогональность, простота, GUI, надежность, стоимость, переносимость программ, универсальность, кроссплатформенность, четкость.

Z = (z1, z2, z3, z4, z5): C++, C#, Java, JavaScript, Python.

Основные параметры математической модели и алгоритм ее реализации представлен в таблице, составленной автором (таблица 3).

Таблица 3 – Основные параметры построения нечеткой модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня

Параметр	Значение
1	2
1. Исходные множества	M (X, Y) N (Y, Z) X – ИТ-проект разработки программного продукта; Y – критерии выбора языков программирования высокого уровня; Z – языки программирования высокого уровня
2. Содержание множеств	X = (x_1, x_2, x_3, x_4, x_5): ИТ-проект 1, ИТ-проект 2, ИТ-проект 3, ИТ-проект 4, ИТ-проект 5. Y = (y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}): читабельность, ортогональность, простота, GUI, надежность, стоимость, переносимость программ, универсальность, кроссплатформенность, четкость. Z = (z_1, z_2, z_3, z_4, z_5): C++, C#, Java, JavaScript, Python.
3. Алгоритм решения	1) Получение исходных данных в виде отношений M и N. Определение функций принадлежности (таблицы 4, 5); 2) Определение их матриц: - для M: $\begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.4 & 0.5 & 0.3 & 0.6 & 0.2 & 0.9 & 0.8 \\ 0.8 & 0.5 & 0.9 & 0.3 & 0.1 & 0.2 & 0.2 & 0.2 & 0.5 & 0.5 \\ 0.3 & 0.9 & 0.6 & 0.5 & 0.9 & 0.8 & 0.9 & 0.8 & 0.6 & 0.3 \\ 0.5 & 0.4 & 0.5 & 0.5 & 0.2 & 0.2 & 0.3 & 0.3 & 0.9 & 0.8 \\ 0.7 & 0.8 & 0.8 & 0.2 & 0.6 & 0.2 & 0.2 & 0.3 & 0.3 & 0.2 \end{bmatrix}$ - для N: $\begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.9 & 1 \\ 0.6 & 0.4 & 0.8 & 0.5 & 0.6 \\ 0.5 & 0.2 & 0.3 & 0.8 & 0.7 \\ 0.5 & 0.9 & 0.5 & 0.8 & 0.4 \\ 1 & 0.6 & 0.5 & 0.7 & 0.8 \\ 0.4 & 0.5 & 1 & 0.7 & 0.8 \\ 0.5 & 0.8 & 0.9 & 0.5 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.6 & 0.5 \\ 0.8 & 1 & 0.2 & 0.5 & 0.6 \\ 0.3 & 0.5 & 0.9 & 0.6 & 0.8 \end{bmatrix}$ 3) Определение матрицы нечеткого отношения. Данные определяются как минимальные значения функции принадлежности всех пар построчно: $\begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.9 & 0.9 \\ 0.8 & 0.8 & 0.7 & 0.8 & 0.8 \\ 0.9 & 0.8 & 0.9 & 0.7 & 0.8 \\ 0.8 & 0.9 & 0.8 & 0.6 & 0.8 \\ 0.7 & 0.7 & 0.8 & 0.8 & 0.7 \end{bmatrix}$ 4) Преобразование матрицы в табличную форму (таблица 6).

Продолжение таблицы 3

1	2
4. Выводы	С точки зрения сравнительного выбора языка программирования высокого уровня, наиболее подходят: - для ИТ проекта 1: C++, C#, JavaScript, Python; - для ИТ проекта 2: C++, C#, JavaScript, Python; - для ИТ проекта 3: C++, Java; - для ИТ проекта 4: C#; - для ИТ проекта 5: Java, JavaScript

Определим содержание отношения M (таблицы 4).

Таблица 4 - Содержание нечеткого отношения M

X\ Y	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
x_1	0,9	0,9	0,8	0,4	0,5	0,3	0,6	0,2	0,9	0,8
x_2	0,8	0,5	0,9	0,3	0,1	0,2	0,2	0,2	0,5	0,5
x_3	0,3	0,9	0,6	0,5	0,9	0,8	0,9	0,8	0,6	0,3
x_4	0,5	0,4	0,5	0,5	0,2	0,2	0,3	0,3	0,9	0,8
x_5	0,7	0,8	0,8	0,2	0,6	0,2	0,2	0,3	0,3	0,2

Определим содержание отношения N (таблицы 5).

Таблица 5 - Содержание нечеткого отношения N

Y \ Z	z_1	z_2	z_3	z_4	z_5
1	2	3	4	5	6
y_1	0,9	0,8	0,7	0,9	1,0
y_2	0,6	0,4	0,8	0,5	0,6
y_3	0,5	0,2	0,3	0,8	0,7
y_4	0,5	0,9	0,5	0,8	0,4
y_5	1,0	0,6	0,5	0,7	0,4
y_6	0,4	0,5	1,0	0,7	0,8
y_7	0,5	0,8	0,9	0,5	0,4
y_8	0,5	0,6	0,7	0,6	0,5
y_9	0,8	1,0	0,2	0,5	0,6
y_{10}	0,3	0,5	0,9	0,6	0,8

Преобразование итоговой матрицы к табличной форме представлено в таблице 6. Данные определяются как минимальные значения функции принадлежности всех пар построчно.

Таблица 6 – Итоговая композиция отношений M и N

ИТ-проект\ Язык программирования	C++	C#	Java	JavaScript	Python
ИТ-проект 1	0,9	0,9	0,8	0,9	0,9
ИТ-проект 2	0,8	0,8	0,7	0,8	0,8
ИТ-проект 3	0,9	0,8	0,9	0,7	0,8
ИТ-проект 4	0,8	0,9	0,8	0,6	0,8
ИТ-проект 5	0,7	0,7	0,8	0,8	0,7

Согласно таблице 6, при сравнительном анализе языков программирования высокого уровня менеджеру проекта следует опираться на полученные результаты моделирования, для реализации проектов отобраны следующие языки:

- для ИТ проекта 1: C++, C#, JavaScript, Python;
- для ИТ проекта 2: C++, C#, JavaScript, Python;
- для ИТ проекта 3: C++, Java;
- для ИТ проекта 4: C#;
- для ИТ проекта 5: Java, JavaScript.

Следовательно, применение методов нечеткой логики позволяет анализировать разнородные и качественные показатели и формировать на их основе решения, направленные на максимально полное соответствие требованиям содержанию ИТ-проектов [51].

Таким образом, методы нечеткой логики способствуют развитию математического подхода к принятию управленческих решений, что важно

для эффективной реализации ИТ-проектов в условиях неопределенности. Одной из важнейших проблем интеграции математического аппарата в управление проектами является недостаточный уровень знаний и популярности методов неточной логики у специалистов, осуществляющих подготовку информации для принятия управленческих решений. В качестве актуальных подходов в практике разработки управленческих решений используются программные средства, экспертный подход, либо принятие решений на основе интуиции, которые неэффективны с точки зрения временных и финансовых затрат. В то же время, математические методы повышают точность анализа и выводов, способствуют снижению субъективизма в формировании данных, использование которых влияет на рациональность принимаемых решений. Все эти аргументы говорят в пользу необходимости внедрение математических инструментов анализа, в том числе методов нечеткой логики, в деятельность менеджеров проектов, у которых предполагается наличие потребности в применении такого аналитического аппарата [31].

4 Результаты моделирования

Разработка математической модели для решения исследовательской задачи магистерской диссертации на основе модели Мамдани осуществлялась в третьем разделе, далее рассмотрим эффективность модели Мамдани с использованием программного продукта моделирования Microsoft Excel.

Исследование позволило разработать математическую модель поддержки принятия решений в контексте выбора языка программирования высокого уровня, что было достигнуто через применение алгоритма Мамдани. Данный алгоритм, базирующийся на нечеткой логике, позволил создать модель, учитывающую множественные параметры, влияющие на выбор языка программирования. Система параметров для построения нечеткой математической модели обеспечивает гибкость и адаптивность модели, что особенно важно в условиях быстро меняющихся технологий и требований к языкам программирования [53].

Моделирование выполнялось по следующему алгоритму.

Шаг 1. Построение входных и выходных нечетких множеств.

1. Ввод адреса ячеек, в которых хранятся параметры функций принадлежности.

2. В ячейках A10:A110 ввод значения из диапазона от 20 до 120.

3. Далее в ячейках B10:D110 ввод формулы для расчета треугольных функций принадлежности.

4. После ввода формул для заполнения всего диапазона значений выполняется автозаполнение.

5. Затем строится график первого нечеткого множества (рисунок 13).

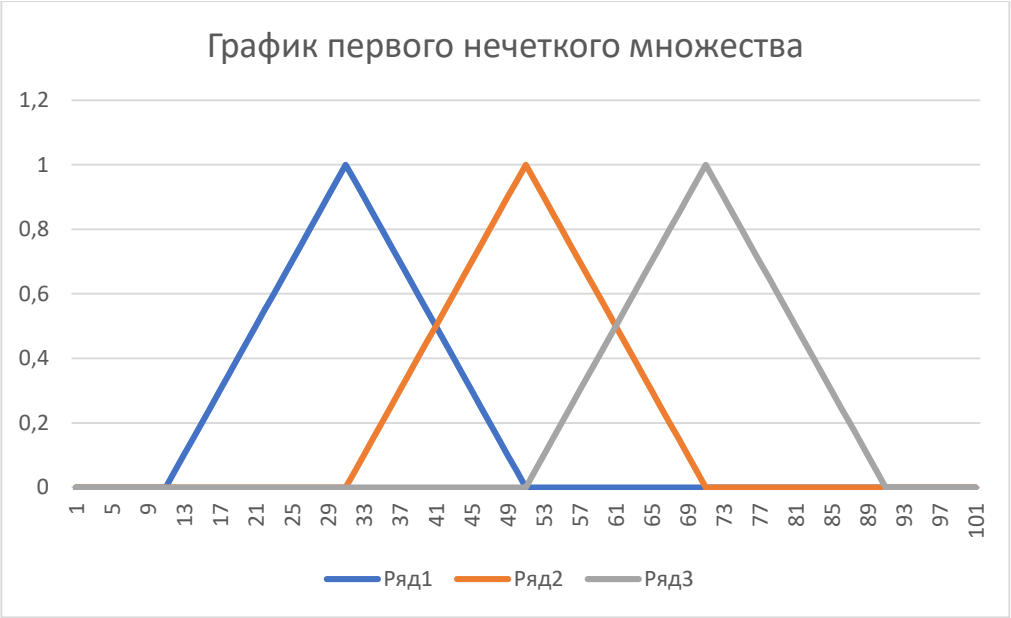


Рисунок 13 - График первого нечеткого множества

6. Для построения второго нечеткого множества в ячейках F10:F110 ввод значения из диапазона от 200 до 300.

7. Далее в ячейках G10:I110 ввод формулы для расчета треугольных функций принадлежности второго нечеткого множества (рисунок 14).

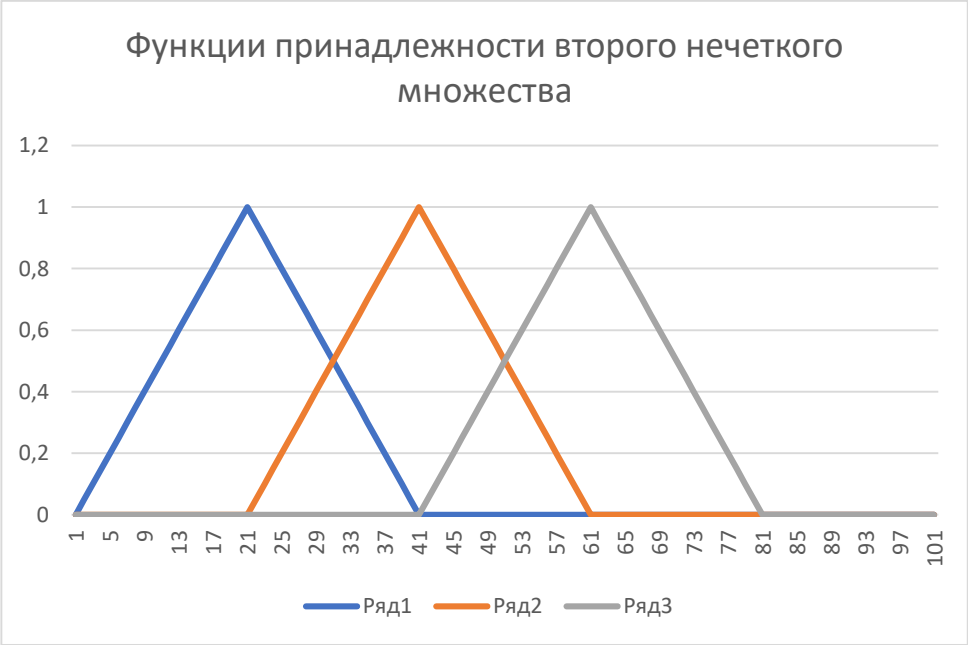


Рисунок 14 - График второго нечеткого множества

8. Для построения выходного нечеткого множества в ячейках K10:K110 ввод значения из диапазона от 500 до 600.

9. Далее в ячейках L10:P10 ввод формулы для расчета треугольных функций принадлежности выходного нечеткого множества (рисунок 15).

Рисунок 15 - Функции принадлежности выходного нечеткого множества

Шаг 2. Фаззификация входных нечетких множеств.

1. Для реализации данного шага, необходимо с помощью ВПР в зависимости от четких значений параметров x1 и x2 найти степени принадлежности входных нечетких множеств.

2. Для определения степеней принадлежности в ячейки AH117:AK119 ввод формулы.

3. В результате данной операции в ячейках AI117:AI119 и AK117: AK119 будут отображены значения степеней принадлежности к входным нечетким множествам.

Данные от датчика		a		b
		75		215
Степени п	x11	0	x21	0,875
	x12	0,75	x22	0,375
	x13	0,25	x23	0

Рисунок 16 - Значения степеней принадлежности к входным нечетким множествам

Шаг 3. Создание базы нечетких правил.

1. Чтобы определить степень возможности каждого правила в программе Excel делаю матрицу нечетких отношений.

Матрица нечетких отношений		
0		
0	0,75	
0	0,375	0,25
0,25	0	
0		

Рисунок 17 - Матрица нечетких отношений

2. После этого определяю степени усечения выходных термов. Для этого в ячейках AL122:AL126 ввод формулы.

y5	0
y4	0,75
y3	0,375
y2	0,25
y1	0

Рисунок 18 - Степени усечения выходных термов

Шаг 4. Аккумуляция

1. В ячейках AB10:AB110 ввод диапазон значений от 500 до 600.

2. В ячейках AC10:AG10 ввод значения. После этого делаю автозаполнение.

3. После формирования заключений нечетко-логического вывода, получается следующий график (рисунок 19).

Рисунок 19- Графическое представление аккумуляции

Шаг 5. Агрегация

1. В ячейках AI10:AI110 ввод значения и после этого делаю автозаполнение.

2. На экране монитора получается следующее изображение.

	AI	AJ
9	Объединение	
10	0,25	125
11	0,25	125,25
12	0,25	125,5
13	0,25	125,75

Рисунок 20 - Таблица для агрегации

3. Графическая интерпретация полученных результатов имеет следующий вид (рисунок 21).

Рисунок 21 – Графическое представление агрегации

Шаг 6. Дефаззификация

1. Чтобы выполнить данный шаг в ячейках AJ10:AJ110 ввод значения и после этого делаю автозаполнение.

2. На экране монитора получился график, аналогичный графику выше.

3. Далее в ячейках AI111:AJ112 ввод следующие данные:

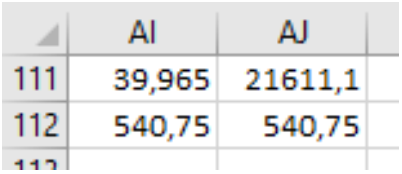
Ячейка / Формула

AI111 =СУММ(AI10:AI110)

AJ111 =СУММ(AJ10:AJ110)

AJ112 =ОКРУГЛ(AJ111/AI111;2)

4. В ячейке AJ112 формируется выходное значение, полученное с помощью модели Мамдани. AI112 =ЕСЛИОШИБКА(AJ112;500)



	AI	AJ
111	39,965	21611,1
112	540,75	540,75

Рисунок 22 - Выходное значение

5. Далее с помощью языка программирования VBA моделирую работу нечеткого вывода.

6. Для этого нужно нажать клавиши Alt + F11 и выбрать команду Insert – Module.

7. В модули вставляю следующий программный код (рисунок 23).

```
Sub hard_Def()
On Error GoTo ErrorHandler
Dim i As Integer, R As Integer, j As Integer, var As Double, columNum As Integer
columNum = 45
For j = 200 To 300 Step 5
 columNum = columNum + 1
 R = 119
 For i = 20 To 120 Step 5
 ThisWorkbook.Worksheets("Defuzzy").Range("ai" & (116)).Value = i
 ThisWorkbook.Worksheets("Defuzzy").Range("ak" & (116)).Value = j
 var = ThisWorkbook.Worksheets("Defuzzy").Range("ai" & (112))
GoTo lab:
ErrorHandler:
 var = 0
Resume Next
lab:
 ThisWorkbook.Worksheets("Defuzzy").Cells(R, columNum).Value = var
 R = R + 1
 Next i
Next
End Sub
```

Рисунок 23 – Листинг

8. В результате выполнения данного программного кода в ячейках AT119:BN139 появляются значения.

	200	205	210	215	220	225	230	235	240	245	250	255	260	265	270	275	280	285	290	295	300
20	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
25	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
30	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
35	500	575	575	575	575	567,5	567,5	567,5	560	552,5	552,5	552,5	545	545	545	545	500	500	500	500	500
40	500	575	575	575	575	569,84	567,5	565,16	560	554,84	552,5	550,16	545	545	545	545	500	500	500	500	500
45	500	575	575	575	575	570,66	567,5	564,34	560	555,66	552,5	549,34	545	545	545	545	500	500	500	500	500
50	500	575	575	575	575	570,66	567,5	564,34	560	555,66	552,5	549,34	545	545	545	545	500	500	500	500	500
55	500	567,5	569,84	570,66	570,66	565,22	562,81	560	555,66	550,22	547,81	545	540,66	540,66	539,84	537,5	500	500	500	500	500
60	500	567,5	567,5	567,5	567,5	562,81	560	557,19	552,5	547,81	545	542,19	537,5	537,5	537,5	537,5	500	500	500	500	500
65	500	567,5	535,16	564,34	564,34	560	557,19	554,78	549,34	545	542,19	539,78	534,34	534,34	535,16	537,5	500	500	500	500	500
70	500	560	560	560	560	555,66	552,5	549,34	545	540,66	537,5	534,34	530	530	530	530	500	500	500	500	500
75	500	552,5	554,84	555,66	555,66	550,22	547,81	545	540,66	535,22	532,81	530	535,66	525,66	524,84	522,5	500	500	500	500	500
80	500	552,5	552,5	552,5	552,5	547,81	545	542,19	537,5	532,81	530	527,19	522,5	522,5	522,5	522,5	500	500	500	500	500
85	500	552,5	550,16	549,34	549,34	545	542,19	539,78	534,34	530	527,19	524,78	519,34	519,34	520,16	522,5	500	500	500	500	500
90	500	545	545	545	545	540,66	537,5	534,34	530	525,66	522,5	519,34	515	515	515	515	500	500	500	500	500
95	500	545	545	545	545	540,66	537,5	534,34	530	525,66	522,5	519,34	515	515	515	515	500	500	500	500	500
100	500	545	545	545	545	539,84	537,5	535,16	530	524,84	522,5	520,16	515	515	515	515	500	500	500	500	500
105	500	545	545	545	545	537,5	537,5	537,5	530	522,5	522,5	522,5	515	515	515	515	500	500	500	500	500
110	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
115	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
120	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500

Рисунок 24 – Значения, полученные в результате работы модели

Апробация модели Мамдани на реальных данных подтвердила её практическую применимость и полезность для принятия обоснованных решений в процессе выбора языка программирования. Это свидетельствует о том, что применение нечеткой логики в данной области может значительно улучшить качество и точность принимаемых решений, обеспечивая более взвешенный и обоснованный подход к выбору инструментов разработки [50].

Графическая интерпретация полученных результатов представлена на рисунке 25.

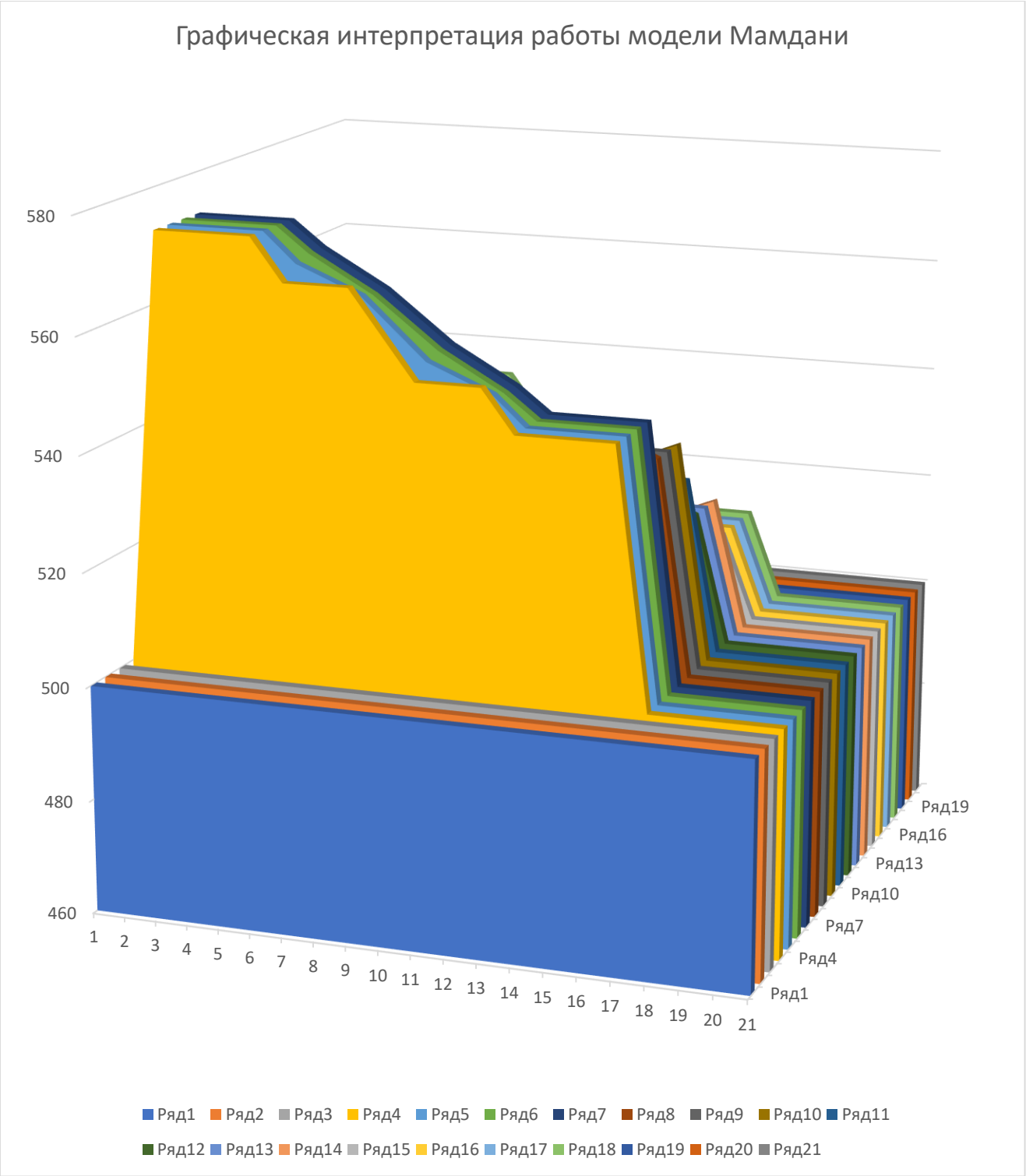


Рисунок 25. Графическая интерпретация работы модели Мамдани

Разработка математической модели поддержки принятия решения при сравнительном анализе и выборе языка программирования высокого уровня

выполнялась на основе выбранного алгоритма Мамдани.

Реализация самой модели Мамдани для решения задачи сравнительного анализа языков программирования высокого уровня и система параметров построения нечеткой математической модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня, а также апробация модели Мамдани, выполненная в Microsoft Excel, показали эффективность ее работы для решения задачи исследования [33].

На основании проведенного моделирования, можно отметить, что достоверность и обоснованность результатов исследования подтверждается экспериментальным моделированием.

В рамках магистерской диссертации получен ряд результатов, обладающих научной новизной, а именно:

- система параметров построения нечеткой математической модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня;
- математическая модель, основанная на алгоритме Мамдани, для решения задачи сравнительного анализа языков программирования высокого уровня.

Заключение

В условиях постоянно развивающихся технологий, повсеместного внедрения искусственного интеллекта, инноваций, развития цифровой экономики и необходимости ускорения цифровой трансформации в различных отраслях возрастает сложность программных продуктов, меняются технологии программирования, требуется постоянное расширение функционала и сокращение среднего срока жизненного цикла разработки.

В этих условиях необходим точный, адекватный, быстрый и качественный выбор языка программирования высокого уровня в рамках реализации различных информационных и инновационных проектов цифровизации и цифровой трансформации, проектов, основанных на современных сквозных технологиях.

Для повышения эффективности выбора необходим инструментарий поддержки, в том числе и инструментарий, основанный на математических моделях, что обеспечит высокий уровень достоверности результата и его обоснованность. Следовательно, развитие системы анализа языков программирования высокого уровня и выбранная тематика является актуальным практико-научным направлением.

Определение эффективности использования различных языков программирования основывается на применении математических моделей, которые позволяют объективно оценить и сравнить их по ряду параметров. Одной из таких моделей является балльная модель, которая, благодаря своей простоте и адаптивности, может быть эффективно использована для анализа и выбора подходящего языка программирования высокого уровня. В данной модели каждому языку присваиваются баллы по определенным критериям, что позволяет структурировано оценить их производительность, удобство использования, функциональные возможности и другие важные характеристики. Этот метод является удобным инструментом для принятия

обоснованных решений в процессе выбора языка программирования, учитывая множество факторов, которые могут повлиять на конечный выбор.

В рамках данного исследования объектом являются математические модели, используемые для принятия решений при сравнительном анализе языков программирования высокого уровня. Предмет исследования фокусируется на процессе принятия решения, основанного на сравнительных характеристиках эффективности этих языков. Подход, основанный на балльной модели, позволяет не только систематизировать и упростить процесс анализа, но и сделать его более прозрачным и объективным. В результате, применение такой модели способствует повышению точности и обоснованности выбора, что особенно важно в условиях быстро меняющихся технологий и требований к программным продуктам.

Цель исследования: разработка системы поддержки принятия решения о выборе в рамках сравнительного анализа языков программирования высокого уровня, основанной на математических моделях.

Для достижения поставленной цели решены следующие задачи:

- изучены теоретические и методические аспекты проведения сравнительного анализа языков программирования высокого уровня;
- изучены преимущества и недостатки языков высокого уровня, их характеристика и отличительные черты;
- исследованы математические модели, применимые для решения поставленной исследовательской задачи;
- разработана математическая модель для проведения сравнительного анализа и оценки языков высокого уровня;
- проведен эксперимент с использованием математической модели для решения исследовательской задачи;
- оценены результаты моделирования.

Научная новизна магистерской диссертации состоит в применении математической модели нечеткой логики к решению задачи сравнительного анализа языков программирования высокого уровня, направленного на развитие систем принятия решений при выборе инструментов реализации ИТ-проектов по разработке программных продуктов.

Теоретическая значимость исследования заключается в расширении теоретических положений в направлении развития математического аппарата при решении задачи сравнительного анализа языков программирования высокого уровня.

Практическая значимость исследования состоит в возможности применения результатов моделирования при выборе инструментов реализации ИТ-проектов по разработке программных продуктов.

Достоверность и обоснованность результатов исследования подтверждается экспериментальным моделированием.

К научным результатам, полученным в рамках исследования, относятся:

– система параметров построения нечеткой математической модели для развития системы поддержки принятия решений при проведении анализа языков программирования высокого уровня;
– математическая модель Мамдани для решения задачи сравнительного анализа языков программирования высокого уровня.

Список используемой литературы и источников

1 Абдрахманов, Б. Т. Языки программирования высокого уровня / Б. Т. Абдрахманов // Актуальные научные исследования : Сборник научных трудов по материалам IV Международной междисциплинарной конференции, Москва, 28 июня 2022 года. – Москва: ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ "НАУЧНО-ИЗДАТЕЛЬСКИЙ ЦЕНТР ТОЛМАЧЕВО", 2022. – С. 14-17. – DOI 10.56545/9785604835760_14.

2 Аветисян, Т. В. Сравнительный анализ двух языков программирования: Java и Scala / Т. В. Аветисян, А. М. Бородай // Вестник Воронежского института высоких технологий. – 2022. – № 2(41). – С. 67-69.

3 Акинин, А. А. Оценка быстродействия программной реализации алгоритмов полиномиального преобразования булевых функций / А.А. Акинин, Ю.С. Акинина, С.В. Тюрин // Моделирование систем и процессов. – 2018. – Т. 11, № 3. – С. 4-9.

4 Бабаева, С. С. Анализ языков программирования для разработки автоматизированных рабочих мест / С. С. Бабаева // Научный электронный журнал Меридиан. – 2020. – № 13(47). – С. 42-44.

5 Борсук, Н. А. Сравнительный анализ алгоритмических языков высокого и низкого уровня / Н. А. Борсук, Д. В. Мастыкаш // Перспективы инновационных научно-практических исследований и разработок : сборник статей международной научной конференции, Санкт-Петербург, 24 февраля 2023 года. – Санкт-Петербург: Частное научно-образовательное учреждение дополнительного профессионального образования Гуманитарный национальный исследовательский институт «НАЦРАЗВИТИЕ», 2023. – С. 24-25.

6 Буковцова, Е. А. Сравнительный анализ и выбор языка программирования для реализации системы анализа факторов ранжирования web-ресурсов // Форум молодых ученых. 2023. № 1 (77). С. 45-50.

7 Волкова, В. Н. Модель формирования портфеля заказов в научно-производственной организации / В. Н. Волкова, А. Е. Леонова // In Situ. – 2015. – № 3(3). – С. 44-46. – EDN UMRMMV.

8 Гарднер Г. Структура разума: теория множественного интеллекта / Г. Гарднер. – М.: ООО «И.Д. Вильямс», 2007. – 512 с.

9 Головнин, О. К. Программирование на языке высокого уровня с элементами системного анализа информации : УЧЕБНОЕ ПОСОБИЕ / О. К. Головнин, А. А. Столбова. – Самара : Самарский национальный исследовательский университет имени академика С.П. Королева, 2019. – 132 с. – ISBN 978-5-7883-1430-3.

10 Дедов, С. В. Анализ преимуществ наиболее востребованных современных языков программирования / С. В. Дедов, О. Д. Кирсанов, О. Ю. Тимошевская // Актуальные вопросы современной науки : сборник статей по материалам XVII международной научно-практической конференции, Томск, 19 декабря 2018 года. Том Часть 1(4). – Томск: Общество с ограниченной ответственностью Дендра, 2018. – С. 63-72.

11 Документация по библиотекам Python с примерами. [Электронный ресурс]. - URL: https://pythonru.com/biblioteki

12 Игнатьев, В. Н. Статический анализ программ для проверки настраиваемых ограничений языков программирования C и C++ : специальность 05.13.11 "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей" : автореферат диссертации на соискание ученой степени кандидата физико-математических наук / Игнатьев Валерий Николаевич. – Москва, 2015. – 22 с.

13 История создания языка программирования Python. [Электронный ресурс]. - URL: https://web.informatics.ru/works/17-18/web_online/barabanov_n_v/language_python.html.

14 Кауфман В.Ш. Языки программирования. Концепции и принципы [Электронный ресурс]: [пособие для студентов вузов]. В.Ш. Кауфман –

Москва: ДМК ПРЕСС, 2020 – 464 с.

15 Коваленко, Т. А. Анализ языков программирования / Т. А. Коваленко, А. Ю. Лобачев // Фундаментальные проблемы основных направлений научно-технических исследований : сборник статей по итогам Международной научно-практической конференции, Волгоград, 17 марта 2018 года. – Волгоград: Общество с ограниченной ответственностью "Агентство международных исследований", 2018. – С. 48-50.

16 Коваленко, Т. А. Анализ языков программирования / Т. А. Коваленко, А. Ю. Лобачев // Фундаментальные проблемы основных направлений научно-технических исследований : сборник статей по итогам Международной научно-практической конференции, Волгоград, 17 марта 2018 года. – Волгоград: Общество с ограниченной ответственностью "Агентство международных исследований", 2018. – С. 48-50.

17 Кононов, В. А. Сравнительный анализ языков программирования Python и C# / В. А. Кононов // Студенческий вестник. – 2020. – № 28-2(126). – С. 50-52.

18 Кононов, В. А. Сравнительный анализ языков программирования Python и C# / В. А. Кононов // Студенческий вестник. – 2020. – № 28-2(126). – С. 50-52.

19 Кравченко, И. В. Обзор современных языков программирования: анализ, область применения, возможности / И. В. Кравченко, А. А. Зевеке, С. В. Кирильчик // Исследования и творческие проекты для развития и освоения проблемных и прибрежно-шельфовых зон юга России : сборник трудов XIV Всероссийской Школы-семинара, молодых ученых, аспирантов, студентов и школьников, Геленджик, 26–28 апреля 2023 года. – Геленджик: Южный федеральный университет, 2023. – С. 187-190.

20 Лукин, И. К. Сравнительный анализ языков программирования для разработки веб-приложений / И. К. Лукин // Теория и практика современной науки. – 2017. – № 2(20). – С. 380-385.

21 Милютин А. Метрики кода программного обеспечения. URL: http://www.viva64.com/ru/a/0045/ (дата обращения: 28.05.2023).

22 Мину М. Математическое программирование. Теория и алгоритмы. М.: Наука (1990).

23 Названы самые популярные языки программирования. C# стремительно рвется в лидеры. [Электронный ресурс]. - URL: https://www.cnews.ru/news/top/2021-08-27_nazvany_samye_populyarnye

24 Потудинский А.В. Модели для определения моментов контроля в многоуровневых организационных системах / А. В. Потудинский, А. П. Преображенский // Моделирование, оптимизация и информационные технологии. – 2020. – Т. 8. – № 2 (29).

25 Пупыкина А. А., Сатунина А. Е. Система оценок моделей Web-приложений // Статистика и Экономика. 2015. №3. С. 255–261.

26 Рейтинг языков программирования для ИИ и машинного обучения. [Электронный ресурс]. - URL: http://digitrode.ru/articles/1643-yazyki-programmirovaniya-dlya-iskusstvennogo-intellekta.html.

27 Рейтинг языков программирования-2021: доля Python падает, а TypeScript обошел C++, в лидерах JavaScript, Java, C#. [Электронный ресурс]. - URL: https://habr.com/ru/post/543346/

28 Ризен, Ю. С. Математическое моделирование образовательного процесса в оценке качества деятельности ВУЗа / Ю. С. Ризен, А. А. Захарова, М. Г. Минин // Информационное общество. – 2014. – № 3. – С. 25-33. – EDN SXZKNF.

29 Руководство по использованию Python-библиотеки NUMPY. [Электронный ресурс]. - URL: https://pythonru.com/biblioteki/rukovodstvopo-ispolzovaniju-python-biblioteki-numpy

30 Самарский А.А., Михайлов А.П. Математическое моделирование. Идеи. Методы. Примеры. - 2-е изд., испр. - М.: Физматлит, 2001. - 320 с.

31 Сисеналиев, Д. Е. Анализ языков промышленного

программирования / Д. Е. Сисеналиев // Студенческий вестник. – 2020. – № 29-2(127). – С. 47-48.

32 Системный анализ и принятие решений : словарь-справ. : учеб. пособие для студентов вузов, обучающихся по направлению подгот. бакалавров и магистров "Систем. анализ и упр." / под общ. ред. В.Н. Волковой и В.Н. Козлова. - М. : Высш. шк., 2004 (ГУП Смол. обл. тип. им. В.И. Смирнова). - 613, [1] с. : ил., табл.; 25 см.; ISBN 5-06-004875-6 (в пер.)

33 Смоленцева Л.В., Сафиуллина Ф.Ф., Малаева А.В. Модели качества информационных систем // Вестник «ТИСБИ». - 2020. - № 4. - С. 82-88.

34 Соложенцева, Р. С. Применение языка программирования PYTHON в анализе данных / Р. С. Соложенцева // Вестник Университета управления "ТИСБИ". – 2021. – № 4. – С. 103-112.

35 Сравнительный анализ инструментальных средств для проведения статического анализа исходных текстов (языки программирования: c/c++) / М. С. Коротков, Е. С. Заньков, В. Н. Сысоев, М. О. Ильин // Математика и математическое моделирование : Сборник материалов XV Всероссийской молодёжной научно-инновационной школы, Саров, 13–15 апреля 2021 года. – Саров: ООО "Интерконтакт", 2021. – С. 282-284.

36 У языков программирования революция. Сменился самыйпопулярный язык в мире. [Электронный ресурс]. - URL: https://www.cnews.ru/news/top/2021-10-11_tsel_dostignutapython_vpervye

37 Чернецкий, В. И. Математическое моделирование динамических систем. Петрозаводск: изд-во Петр.ГУ, 1996.

38 Чернова, С. В. Сравнительный анализ языков программирования высокого уровня Python и C++ / С. В. Чернова, В. А. Ларина // Аспирант и соискатель. – 2019. – № 5(113). – С. 9-11.

39 Чернова, С. В. Сравнительный анализ языков программирования высокого уровня Python и C++ / С. В. Чернова, В. А. Ларина // Аспирант и

соискатель. – 2019. – № 5(113). – С. 9-11.

40 Шишова, И. В. Анализ тенденций развития языков программирования / И. В. Шишова, М. М. Абдусаламов // Неделя науки - 2018 : сборник материалов XXXIX итоговой научно-технической конференции преподавателей, сотрудников, аспирантов и студентов ДГТУ, ДГТУ, 23–28 апреля 2018 года. – ДГТУ: Дагестанский государственный технический университет, 2018. – С. 83-85.

41 Шпинев, А. В. Сравнительный анализ языков программирования Java и C++ / А. В. Шпинев, В. С. Лучников, Т. П. Новикова // Моделирование информационных систем : Материалы Международной научно-практической конференции, Воронеж, 19–20 мая 2021 года. – Воронеж: ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова», 2021. – С. 313-319. – DOI 10.34220/MIS313-319.

42 Analysis of theoretical and methodological bases of teaching object - oriented programming languages in higher school / B. Shayakhmetova, N. Orumbayeva, Sh. Omarova, Yu. Antipov // Bulletin of the Karaganda University. Mathematics Series. – 2017. – No. 4(88). – P. 73-79. – DOI 10.31489/2017M4/73-79.

43 C++ vs Java. – URL: https://www.javatpoint.com/cpp-vs-java (дата обращения: 30.03.2024).

44 C++ vs Java: Basic Comparison, Key Differences, & Similarities. – URL: https://hackr.io/blog/cpp-vs-java (дата обращения: 30.03.2024).

45 Hart, W.E., Laird, C., Watson, J.-P., Woodruff D.L. Pyomo–optimization modeling in python. 2012. - Vol. 67: Springer, 238 p.

46 JavaScript Higher Order Functions and Arrays // oodlesTechnologies URL: https://www.oodlestechnologies.com/blogs/javascript-higher-order-functions-and-arrays. (дата обращения: 30.03.2024).

47 Kosko B. Fuzzy Systems as Universal Approximators // IEEE Trans. on Computers. 1994. Vol. 43.№11. P.1329 1333.

48 Mamdani, E. Application of fuzzy algorithms for the control of a simple dynamic plant. Proc.IEEE 121, 1974. — p.1585 — 1588.

49 Mernik M., Heering J., Sloane A. M. When and how to develop domain-specific languages // ACM computing surveys (CSUR). 2015. Vol. 37. No. 4. P. 316–344.

50 Parr T. Language implementation patterns: create your own domain-specific and general programming languages. Pragmatic Bookshelf, 2019. - P. 175.

51 Rotshtein, A. INTEGRATION OF FUZZY LOGIC AND CHAOS THEORY APPROACHES IN RELIABILITY MODELING AND OPTIMIZATION. Proceedings of the Academy of Sciences. Theory and control systems. 2012. - V.4. – P. 77-87.

52 Smirnov, S., Voloshinov, V., Sukhosroslov, O. “Distributed Optimization on the Base of AMPL Modeling Language and Everest Platform”, in Procedia Computer Science, 2016. - Vol. 101. - P. 313-322.

53 Stepanova, D. S. Comparative analysis of programming languages / D. S. Stepanova // Young people. Society. Modern science, technology and innovation. – 2021. – No. 20. – P. 75-77.