МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
/ Б.М. Суховилов /
«_____» _____20____ г.

Реализация библиотеки на языке программирования PHP

по методам теории принятия решений.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.03.2021.1757.ВКР

Руководитель, доцент
_____/ И.А. Прохорова /
«_____» _____2021 г.

Автор
студент группы зЭУ-582
_____/ М.А. Шумаков /
«_____» _____2021 г.

Нормоконтролер, ст. преподаватель
_____/Е.Н. Горных/
«_____» _____2021 г.

Челябинск 2021

АННОТАЦИЯ

Шумаков М. А. Реализация библиотеки на языке программирования PHP по методам теории принятия решений: Выпускная квалификационная работа. – Челябинск: ЮУрГУ, ВШЭУ, зЭУ-582, 2021. – 66 с., 7 ил., 12 табл., библиогр. список – 24 наим., 12 прил.

В выпускной квалификационной работе на основе данных по логистической и складской деятельности компании ООО «ВсеИнстурменты.ру» предложено решение, которое позволит сократить проблемы с размещением товаров разного размера, их консолидации и выбора оптимальных логистических цепочек для перемещения. Решение реализовано в виде библиотеки на языке программирования PHP для принятия оптимальных решений в условиях многокритериальности по ограниченному списку методов теории принятия решений (метод главного критерия и метод аддитивной свертки).

Для разработки библиотеки выполнен анализ предметной области, определены требования к функционалу, спроектирована архитектура библиотеки, проведено кодирование, отладка и тестирование.

Предлагаемая библиотека предназначена для веб-сервиса компании ООО «ВсеИнструменты.ру» по управлению складом.

Для обоснования эффективности решения в выпускной квалификационной работе использована сравнительная оценка результатов, сгенерированных через библиотеку и через математические модели в Excel.

Результаты выпускной квалификационной работы имеют практическую ценность, которая подтверждается сравнительным анализом.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Реализация систем по хранению и обработке данных является перспективным направлением.

Для повышения уровня релевантного взаимодействия с пользователем необходимы механизмы, которые позволяют формировать шаблоны прогнозирования на основе высокого числа вариантов и представлять самые подходящие. Чтобы успешно решать задачи с выбором оптимального варианта, требуются специальные методы теории принятия решений, теории вероятности и числовых рядов, которые позволят находить подходящие, с точки зрения лица принимающего решения, варианты и тем самым более успешно предоставлять данные пользователю. Поэтому реализация систем, библиотек и конечных модулей с применением методов теории принятия решений является нужным направлением.

Теория принятия решений – область исследования, вовлекающая понятия и методы математики, статистики, экономики, менеджмента и психологии с целью изучения закономерностей выбора людьми путей решения проблем и задач, а также способов достижения желаемого результата.

Основным объектом исследования работы является логистика компании ООО «ВсеИнструменты.ру» и прочие складские процессы.

Предметом исследования является экономическая и логистическая эффективность деятельности предприятия.

Цель работы – реализовать механизм для принятия оптимальных решений по методам главного критерия и аддитивной свертки на стороне вычислительной системы с минимальным участием пользователя.

Для достижения поставленной цели необходимо решить следующие задачи.

1. Провести анализ предметной области.
2. Рассмотреть алгоритмы решения многокритериальных задач.

3. Дать общую характеристику выбранных алгоритмов и провести их анализ.

4. Реализовать библиотеку для применения конкретных методов теории принятия решений.

5. Провести анализ результатов реализованной библиотеки с аналогичной математической моделью в Excel.

В рамках работы реализована библиотека на языке программирования PHP для возможности использования методов главного критерия и аддитивной свертки.

Методы исследования: анализ аналогичных решений по проблеме исследования; программная аналитика; проектирование; разработка и отладка; тестирование; количественная и качественная обработка данных.

Для разработки основной функциональности библиотеки используется симплекс-метод для осуществления поиска решений при заданных наборах ограничений и методы главного критерия, и метод аддитивной свертки для реализации функции полезности.

Предметом защиты является разработанная библиотека на языке программирования PHP для принятия оптимальных решений.

Практическая ценность библиотеки заключается в том, что можно на уровне программных решений (проектов), написанных на языке программирования PHP, использовать методы принятия решений и тем самым находить оптимальные варианты. Основная новизна библиотеки в том, что для языка программирования PHP таких решений не предоставлено и в сообществе нет адекватных проектов и модулей, которые бы могли решать поставленные задачи с выбором оптимального варианта из набора вариантов.

Структура выпускной квалификационной работы: введение, четыре главы, заключение, список используемой литературы и приложения.

Во введении работы обоснована актуальность выбранной темы исследования, определена цель и задачи исследования.

В первой главе дана характеристика предметной области; рассмотрены методы главного критерия и аддитивной свертки; приведена общая характеристика выбранных алгоритмов; произведена постановка задачи.

Во второй главе приведены шаги по сбору требований к функционалу; проектированию архитектуры библиотеки; кодированию и отладке; тестированию и внедрению.

В третьей главе приведены математические модели для принятия решений на основе Excel; произведено сравнение эффективности реализованной библиотеки с математической моделью Excel; проведен анализ результатов.

В четвертой главе проведены экономические расчеты по определению затрат на разработку продукта.

В заключении работы сделаны выводы по результатам исследования.

ГЛАВА 1 ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

В общем смысле теория принятия оптимальных решений это комплекс математических и численных методов, направленных на поиск наилучших вариантов из множества альтернатив и позволяющий исключить их полного перебора.

Предпосылки к появлению и развитию дисциплины «Теории принятия решений»:

- повышение «цены ошибки»;
- ускорение научно-технической революции техники и технологии;
- развитие ЭВМ.

Цель теории принятия решений применить научные методы, чтобы помочь лицу, принимающему решение, выбрать оптимальное значение [1].

Принятие решения – это процесс выявления альтернатив и выбора среди вариантов решения наилучшего решения, основанного на ценностях и предпочтениях лица, принимающего решения [2].

Теория принятия решений обеспечивает основу, с помощью которой вся информация о предпочтениях лица, принимающего решения, используется для вывода, какое альтернативное решение кажется «лучшим».

Теория принятия решений включает в себя различные нормативные процедуры, которые помогают лицу, принимающему решения, формализовать свои предпочтения.

Ключевые проблемы теории принятия решений.

1. Потребность учета большого числа несовместимых критериев – проблема оптимальности при векторном критерии.

2. Глубина неопределенности, обусловленная недостающей степенью информации для принятия аргументированных выводов [3].

В таблице 1 приведены элементы процесса принятия решений [4].

Таблица 1 – Элементы процесса принятия решений

Элементы процесса принятия решений	
Цель	Субъективный образ (абстрактная модель) несуществующего, но желаемого состояния среды, которое решило бы возникшую проблему
Лицо, принимающее решение (ЛПР)	Единоличный или коллегиальный орган управления, утверждающий решение и несущий за него ответственность, т. е. ЛПР, может считаться как человек, так и организация, имеющая единый интерес или единую цель, которая служит мотивом ее решения.
Альтернативные (взаимоисключающие) варианты решения	Другие решения смотрятся как средства достижения цели.
Внешние условия	Внешними условиями (внешней средой или совокупностью состояний природы) называется полная система несовместимых явлений, влияющих на исход решения, причем ЛПР известна не вся информация о внешней среде.
Исходы	ЛПР учитывает, что в зависимости от выбранной альтернативы и состояния природы результат будет иметь для него различную ценность.
Правило выбора решения	Правило выбора (критерий) решения дает возможность однозначно выбрать наиболее предпочтительное в каком-либо смысле решение.

Для понимания проблемы принятия решения используются две величины:

- ценность (через использование теории полезности);
- информация (через использование теории вероятностей).

Схема процесса принятия решений.

1. Предварительный анализ проблемы.
2. Постановка задачи.
3. Получение исходных данных.
4. Решение задачи принятия решений с привлечением математических методов и вычислительной техники, экспертов и ЛПР.
5. Анализ и интерпретация полученных результатов.

В таблице 2 приведена классификация задач принятия решений.

Таблица 2 – Классификация задач принятия решений

Решения принимаются отдельными людьми / группами лиц			
в условиях определённости	в условиях риска	в условиях неопределённости	в условиях противодействия
оценка по одному или по многим критериям			

В зависимости от того, на какой этапе хода принятия решений обнаруживаются и применяются предпочтения ЛПР, возможно обозначить три группы многокритериальных методов принятия решений.

1. Базирующиеся на том, что ЛПР может сформулировать свои предпочтения перед началом хода многокритериальной оптимизации.

2. Интерактивные методы.

3. Способы построения множества успешных решений с дальнейшим показом его ЛПР.

В рамках данного исследования рассматриваются следующие методы:

- метод главного критерия;
- метод аддитивной свертки.

Для поиска решений в реализуемой библиотеке используется симплекс-метод, который применяется для нахождения оптимального значения при заданных ограничениях.

Симплекс-метод – метод решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве.

Основа метода: построение базовых решений, на которых однообразно спадает линейный функционал, до ситуации, когда производятся требуемые обстоятельства локальной оптимальности [5].

В работе Л. В. Канторовича «Математические методы организации и планирования производства» (1939) [6] впервые рассказаны взгляды свежей отрасли математики, которая впоследствии приобрела наименование линейного программирования.

Цель линейного программирования в том, чтобы максимизировать либо минимизировать определенный линейный функционал на многомерном пространстве при установленных линейных ограничениях.

Одной из особенно сложных операций в симплекс-методе представляется исследование включаемого в базис столбца. Для лучшей сходимости необходимо подбирать переменную с наилучшей невязкой, но для этого необходим абсолютный просмотр, то есть необходимо перемножить столбец дуальных переменных (которые временами именуются теневыми ценами) на все столбцы матрицы [7].

Подобный ход неплохо функционирует для небольших задач, которые решаются вручную. Более того, жесткое следование правила предпочтения наибольшей по модулю невязки сможет оказаться неоптимальным с точки зрения совместного числа итераций, важных для получения экстремума.

Наибольшая выгода на одной итерации сможет ввергнуть к неторопливому убыванию значения целевой функции на дальнейших шагах и, следовательно, застопорить ход решения задачи.

1.2 Характеристика выбранных алгоритмов

Методы главного критерия и аддитивной свертки базируются на том, что ЛПР может сформулировать свои предпочтения до начала хода многокритериальной оптимизации [8].

Поэтому, как правило, данными способами не получается за один раз заполучить применимое решение.

Многократное использование с корректировкой определяемых величин практически переводит их в разряд интерактивных, однако не приспособленных для дружественного общения с ЛПР, способов.

На выбор методов главного критерия и аддитивной свертки для реализации в библиотеки повлияли следующие причины:

- минимальная сложность для разработки;

- проверка прототипа библиотеки в минимальный срок;
- возможность решать задачи логистических цепочек и размещений по габаритам.

Для достижения рабочего прототипа библиотеки в течение небольшого количества времени, необходимо отказаться от сложных деталей реализации и плохо формализуемой доменной модели, которые сильно влияют на темпы разработки решения.

Таким образом, сравнение метода главного критерия и аддитивной свертки показало, что данные методы просты в формализации на программном уровне и не требуют больших временных ресурсов для их реализации.

Перевести многокритериальную задачу к однокритериальной – это отметить один (ключевой) критерий и направить его в экстремум, а на оставшиеся критерии поставить ограничения, потребовав, чтобы они были не меньше (больше) неких установленных величин.

Основной тезис метода главного критерия содержится в том, что критерии, естественно, неравнозначны между собой (одни из них более важны, чем другие), поэтому возможно выделить ключевой критерий, а другие (критерии) анализировать как дополнительные, сопутствующие.

Для возможности расчета целевых функций по пороговому значению для логистических задач применен метод главного критерия.

К недостаткам метода главного критерия можно отнести:

- оптимизация проводится, по существу, по одному критерию, а оставшиеся могут получать значения на границе возможной области;
- в методе главного критерия ограничения наложенные на сопутствующие критерии могут привести к неразрешимости задачи.

Для построения общего критерия зачастую применяют аддитивные и мультипликативные преобразования. В случае аддитивной свертки обобщенный критерий определяется суммой частных критериев с соответствующими весами.

К недостатку аддитивной свертки можно отнести:

- наличие множества решений за счет взаимной компенсации критериев;
- изменение весов модели аддитивной свертки может не отразиться на результате.

1.3 Постановка задачи

В рамках исследования требуется решить проблему выбора оптимальных значений для корректного выполнения логистических и складских задач.

Сферы применения библиотеки не должны ограничиваться только указанным применениям. Есть еще множество направлений, где применение данного решения имеет место и необходимость. Это, например, оценка контрагентов по их истории взаимодействия с компанией для выстраивания систем лояльности, персональных цен, контрольных групп к функционалу или оценки номенклатур по их реализации для формирования представлений по ликвидации товаров и пересмотра закупочной политики и т. п.

Библиотека может быть использована и в других отраслевых направлениях, где требуется производить выбор и принимать решения.

В связи со всем вышесказанным можно сформулировать следующие требования:

- библиотека должна обладать простым алгоритмом внедрения в проекты;
- в первой стабильной версии должен быть реализован метод главного критерия и метод аддитивной свертки (функции полезности);
- для поиска решений должен быть применен симплекс-метод;
- библиотека должна быть как можно меньше зависеть от других пакетов;

- необходимо, чтобы библиотека была гибкой в конфигурации и позволяла в режиме рабочего потока (runtime) менять выбранную стратегию принятия решения;
- библиотека должна иметь высокое покрытие тестами, а именно не менее 80% кодовой базы, чтобы её дальнейшее развитие и активное изменение не приводило к регрессу и потере обратной совместимости.

Так как есть множество вариантов с разными критериями оптимального значения, то возникает необходимость в реализации автоматического механизма принятия решений. Требуется это из-за того, что по каждому из обрабатываемых процессов проходит большое количество данных, которые анализировать в ручном режиме является экономически и технологически не оптимальным решением.

После рассмотрения предметной области, изучения литературы, технической документации и других материалов по теме выпускной квалификационной работы решено реализовать библиотеку на языке программирования PHP.

Основные задачи, которые требуется решить для разработки библиотеки:

- анализ требований;
- проектирование архитектуры библиотеки;
- кодирование и отладка;
- тестирование.

Выводы по первому разделу

Произведен анализ предметной области по теории принятия решений и рассмотрены основные характеристики группы методов, в которых ЛПР заранее может высказать свои предпочтения.

Выделены конкретные методы позволяющие решать проблемы логистики и складских процессов (метод главного критерия и аддитивная свертка).

Выполнена постановка задач для разработки библиотеки и обозначены требования для кодирования и отладки.

ГЛАВА 2 РАЗРАБОТКА БИБЛИОТЕКИ

2.1 Анализ требований

Анализ требований – это элемент процесса разработки программного обеспечения, который включает добавление условий в программное обеспечение (программное обеспечение), их систематизацию, выявление взаимосвязей, а также документацию.

В процессе сбора требований важно учитывать вероятные противоречия мнений различных заинтересованных сторон, таких как клиенты, разработчики или пользователи.

Достоверность и качество анализа требований имеют первостепенное значение для успеха всего проекта.

Требования к программному обеспечению (ПО) должны быть задокументированы, исполняться, тестироваться с уровнем детализации, необходимым для проектирования системы. Требования могут быть функциональными или нефункциональными.

Разбор условий содержит 3 типа деятельности.

1. Сбор требований. Связь с клиентами и пользователями, чтобы определить их предпочтения.

2. Исследование предметной области. Анализ условий – определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем; выявление взаимосвязи требований.

3. Протоколирование требований. Требования могут быть представлены в разнообразных формах, таких как описания, сценарии использования, пользовательские истории или спецификации процессов.

Новые системы изменяют среду и связи между людьми, поэтому важно охватить все заинтересованные стороны, принять во внимание все их потребности и убедиться, что они понимают значение новых систем.

Ход разбора условий к информативной системе охватывает следующие фазы:

- разработка условий (выявление требований, исследование требований, классификация условий и проверка требований);
- управление требованиями.

Для выявления первостепенных условий по разрабатываемому решению, необходимо осознать проблему, которую планируем решить.

2.2 Проектирование архитектуры библиотеки

Для реализации библиотечной архитектуры рассмотрены основные шаблоны проектирования, позволяющие обеспечить достаточную гибкость и простоту реализуемого решения.

Для реализации требования: «библиотека должна обладать простым алгоритмом внедрения в проекты»: выбран шаблон проектирования – фасад, пример структуры шаблона описан в приложении А.

Для реализации требования: «в первой стабильной версии должен быть реализован метод главного критерия и метод аддитивной свертки (функции полезности)»: был выбран шаблон проектирования – строитель, пример структуры шаблона описан в приложении Б.

UML-диаграмма класса с описанием точки входа в библиотеку представлена на рисунке 1.

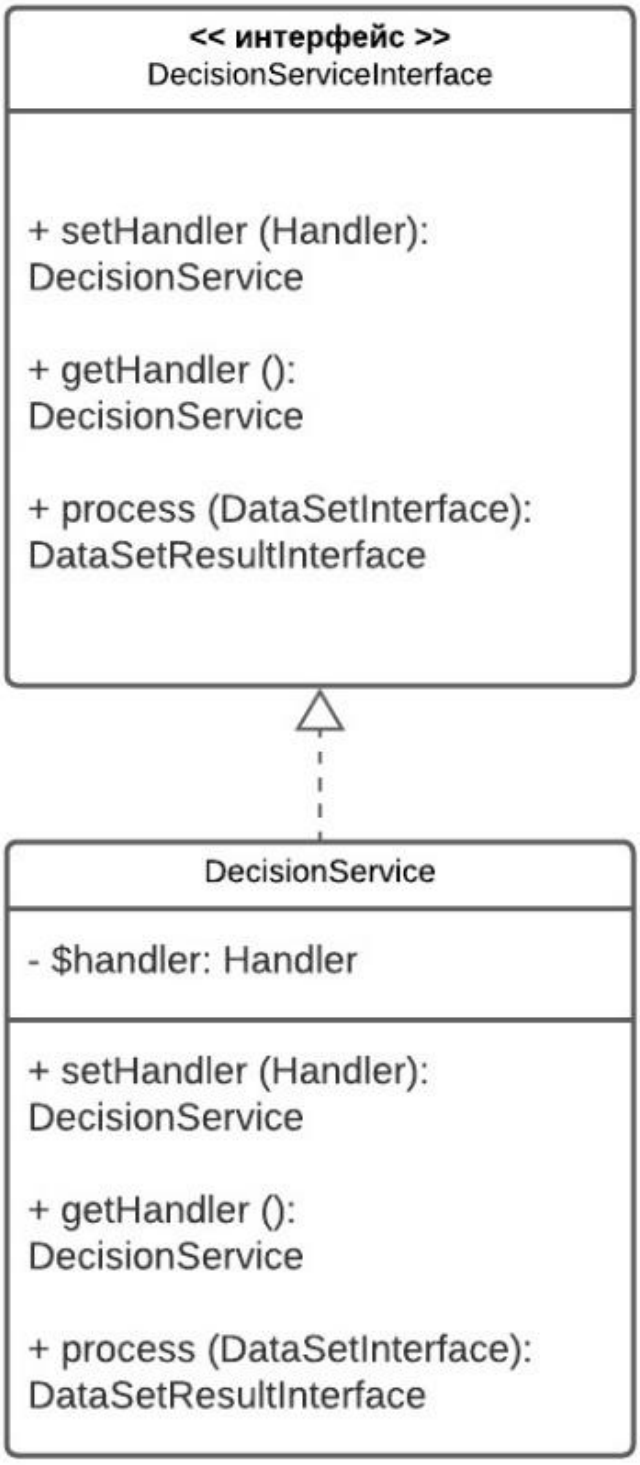


Рисунок 1 – UML-диаграмма класса с описанием точки входа

UML-диаграмма классов для формализации методов по главному критерию и аддитивной свертки представлена на рисунке 2.

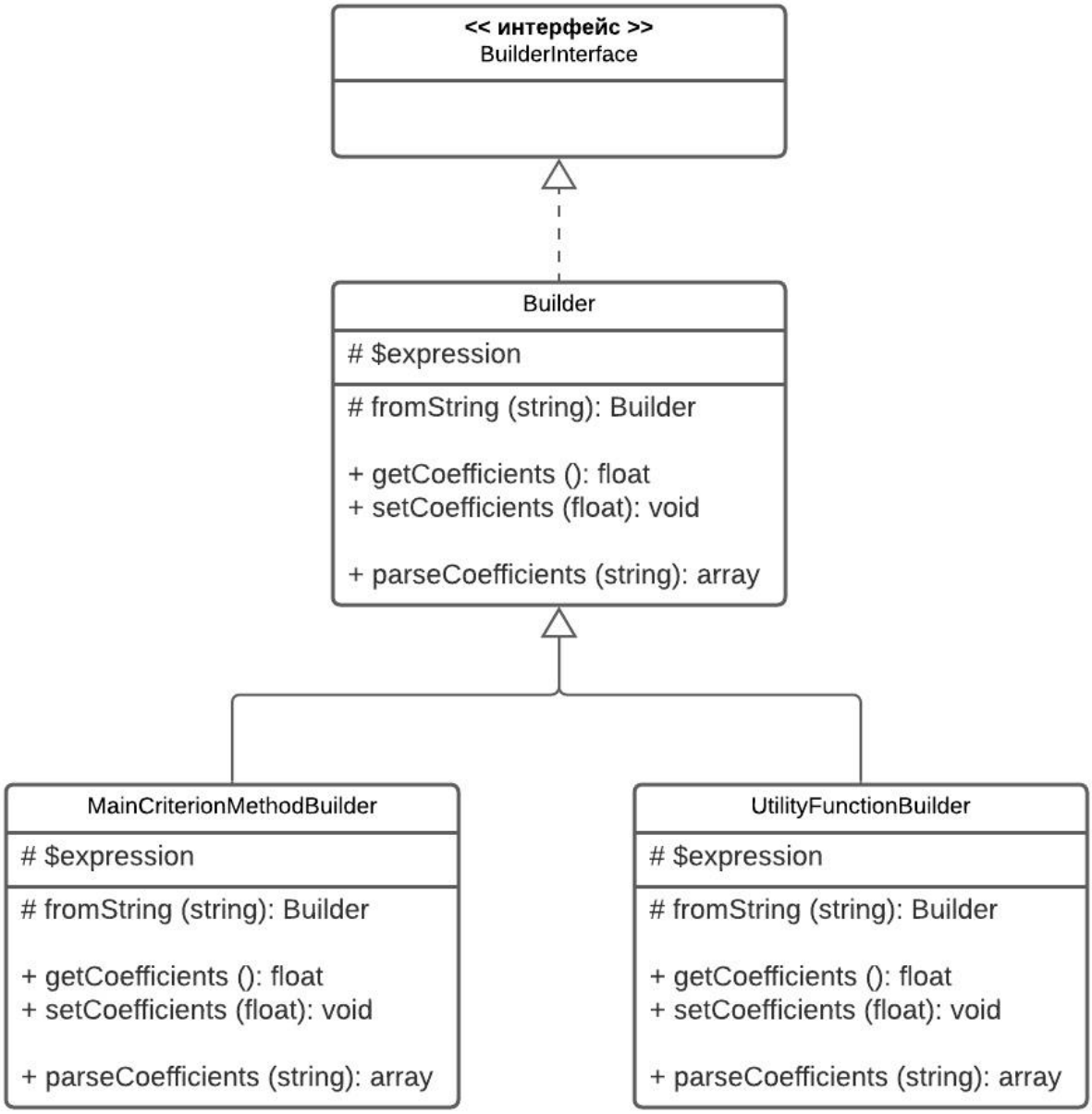


Рисунок 2 – UML-диаграмма классов для формализации методов

Для реализации требования: «для поиска решений должен быть применен симплекс-метод»: решено реализовать защищенное поведение в абстрактном классе Handler, который будет применять логику, описанную в суперклассе, для всех реализаций. Основная структура данного шаблона раскрыта в диаграмме фасада.

Для реализации требования: «необходимо, чтобы библиотека была гибкой в конфигурации и позволяла в режиме рабочего потока (runtime) менять

выбранную стратегию принятия решения»: выбран шаблон проектирования – стратегия, пример структуры шаблона описан в приложении В.

Для реализации требования: «библиотека должна иметь высокое покрытие тестами, а именно не менее 80% кодовой базы, чтобы её дальнейшее развитие и активное изменение не приводило к регрессу и потере обратной совместимости»: принято решение строить процесс разработки через Test Driven Development (TDD) – разработка на основе тестов.

2.3 Кодирование и отладка

Основной задачей после проектирования архитектуры библиотеки является непосредственная реализация программного продукта.

Кодирование представляет лишь часть программирования вместе с анализом, проектированием, компиляцией, тестированием (мануальным, автоматическим), отладкой и сопровождением.

Так как для разработки библиотеки выбран подход через тестирование, то сначала необходимо реализовать требования к разрабатываемому решению в виде тестов, чтобы далее итеративно наращивать кодовую базу под контролем тестовых сценариев.

После определения основного списка используемых шаблонов программирования и создания необходимых тестовых сценариев для тестирования требований начинается этап реализации требований.

При реализации функциональности, определен используемый стандарт кодирования PSR-12 [9], который призван помочь программистам унифицировать решения задач и упростить работу над данным решением при командном взаимодействии с кодовой базой. Основная точка входа в библиотеку приведена в приложение Г.

Проект создан в удобной интегрированной среде разработки Phpstorm [10].

На рисунке 3 приведена структура библиотеки сразу после её создания.

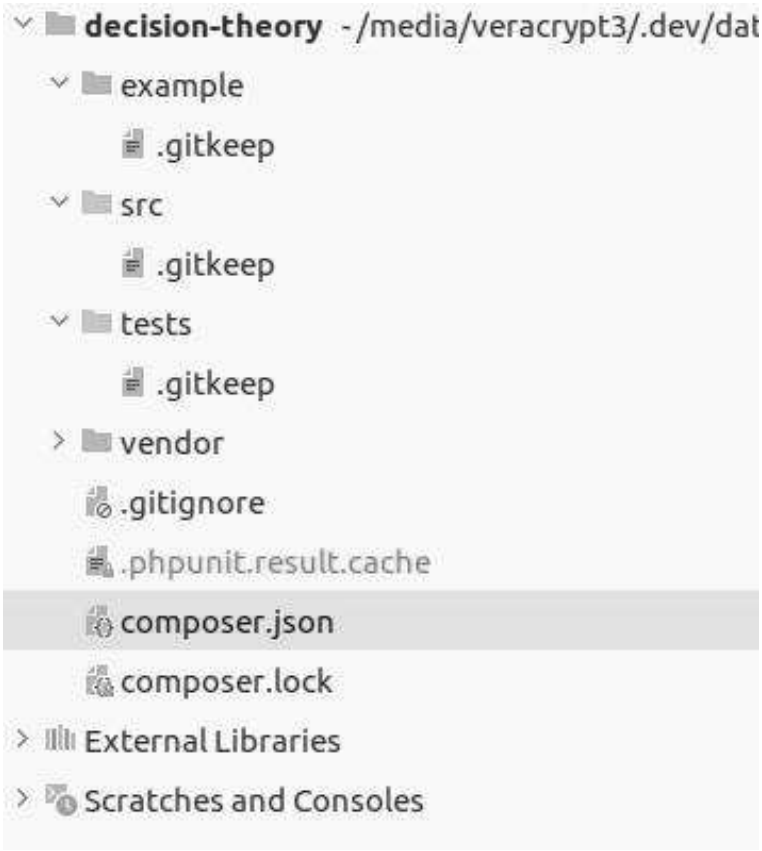


Рисунок 3 – Структура созданной библиотеки

При формировании структуры библиотеки выделены основные директории для хранения непосредственно исполняемой части библиотеки (src), тестовый части (tests) и директории с примера внедрения и использования.

Позднее на основе заложенной структуры возникли дополнительные директории и файлы, которые позволили настроить автоматические запуски тестов в облачном сервисе github для продолжения непрерывной интеграции continuous integration [11].

На рисунке 4 приведена структура библиотеки с добавлением базовых классов.

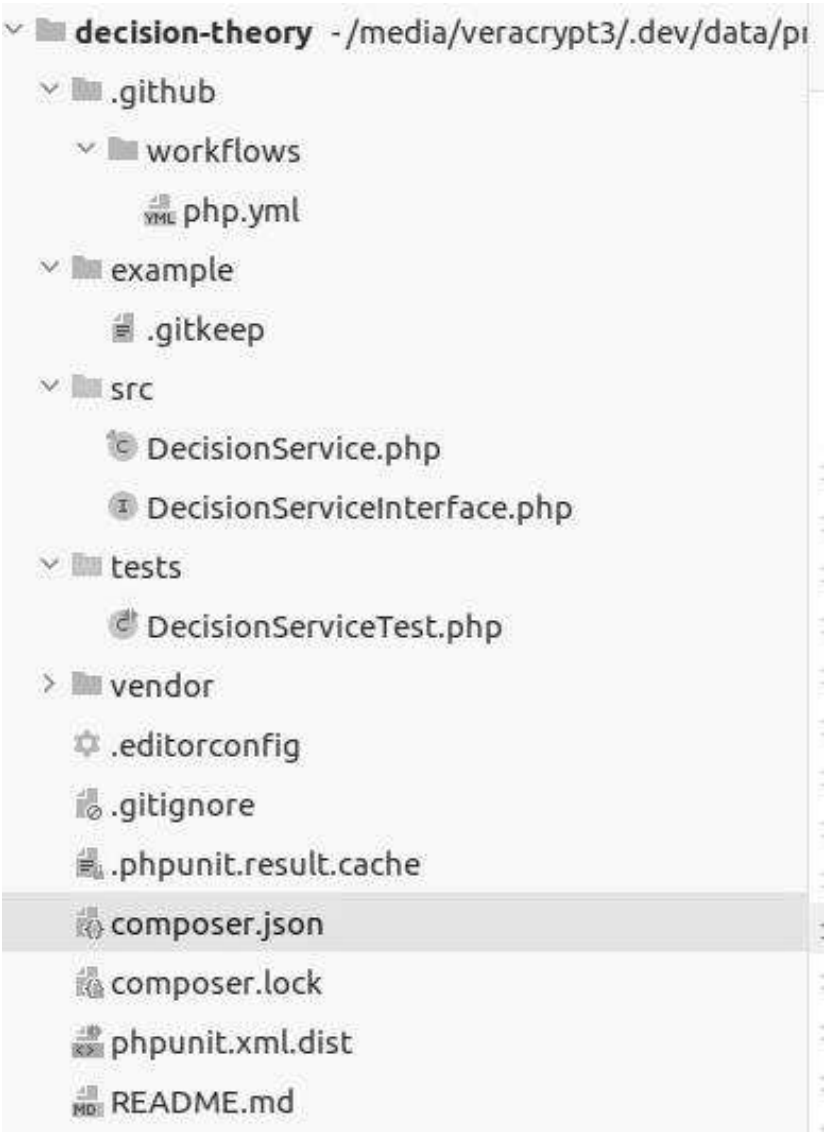


Рисунок 4 – Структура библиотеки с добавлением базовых классов

При итеративном улучшении кодовой базы возникают новые структуры и логика, которая в дальнейшем необходима для решения задач линейного программирования по методам главного критерия и аддитивной свертки.

На рисунке 5 приведена структура эволюции исполняемой части кодовой базы библиотеки.

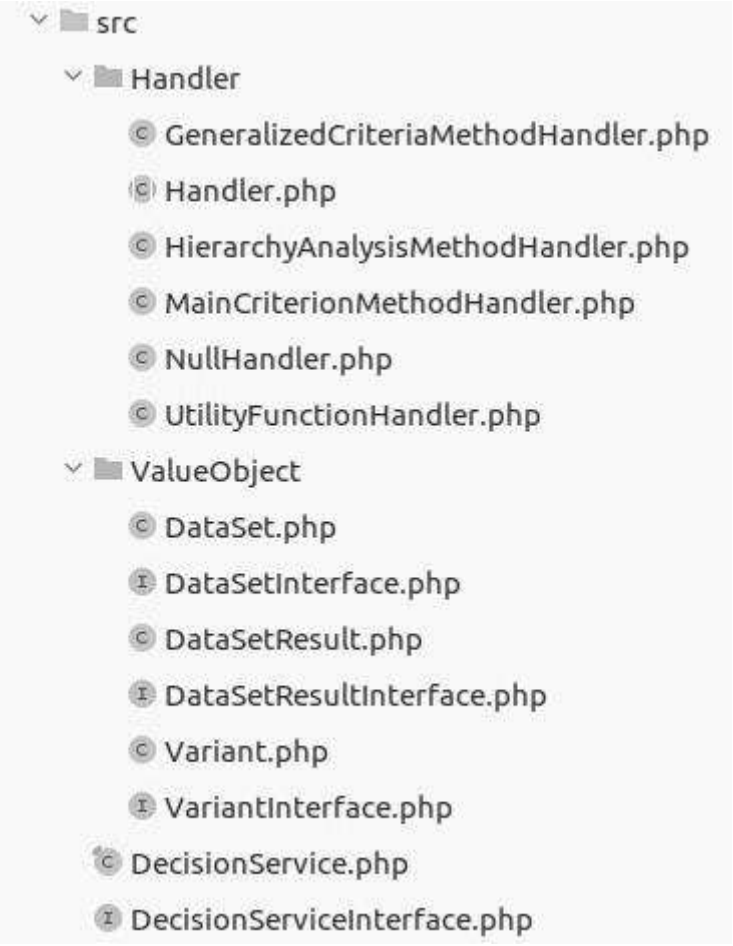


Рисунок 5 – Структура эволюции исполняемой части кодовой базы библиотеки

Как видно из рисунка 5 в библиотеке появились реализации шаблонов программирования, которые изменяют структуру исполняемой части кодовой базы и вводят такие директории как Hander и ValueObject [12].

В директории Handler располагаются классы, описывающие логику обработки по методу главного критерия и функции полезности, которую выбрано реализовать за счет аддитивной свертки.

Value Object (VO) – это иммутабельная структура данных. Как и в случае с data transfer object (DTO), у VO нет строгого правила принадлежности к конкретному слою, однако VO, обычно, располагается на доменном слое. В широком смысле VO это неидентифицируемая модель.

На рисунке 6 приведено изменение структуры тестов из-за изменения исполняемой части библиотеки.

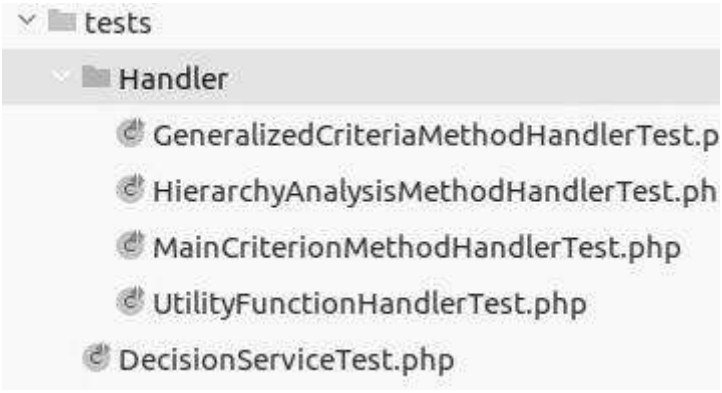


Рисунок 6 – Изменения структуры тестов из-за изменения исполняемой части

Из-за изменений кодовой базы по исполняемой части библиотеки в структуре тестов также требуются изменения. Происходит выделение конкретной логики главного критерия и логики аддитивной свертки в отдельные классы. Появляются свои тесты с вложенными тестовыми сценариями.

Листинг тестового сценария описан в приложении Д.

2.4 Тестирование

При разработке библиотеки использован подход TDD [13].

Основной проблемой, которая часто возникает при работе над разными проектами, является тестирование. А точнее её отсутствие, что в дальнейшем сказывается на сложности разработки и поддержки программного обеспечения.

Именно тесты позволяют до ввода в эксплуатацию кода проверить его готовность, определить, как его использовать и провести документирование. Хорошие тесты самодокументированы и позволяют описать функционально систему, для которой реализованы.

Разработка через тестирование – это навык разработки программного обеспечения, основанный на повторении очень коротких циклов разработки: сначала вы пишете тест, чтобы охватить желаемое изменение, затем вы пишете код, позволяющий пройти тест, а затем заново сгенерированный код подвергается рефакторингу соответствующим стандартам.

Кент Бек, считающийся изобретателем техники TDD, заявлял в 2003 году, что разработка посредством тестирования одобряет несложный дизайн и вносит уверенность [14].

Листинг интерфейса для DecisionService приведен в приложении Е.

Разработка посредством тестирования призывает от разработчика создания автоматизированных модульных тестов, устанавливающих условия к коду непосредственно накануне написания самого кода. Тест включает испытания условий, которые могут либо выполняться, либо нет. Когда они выполняются, говорят, что тест пройден. Проход теста свидетельствует поведение, предполагаемое программистом.

Разработчики зачастую используют библиотеки для автоматизации запуска группы тестов.

На практике модульные тесты покрывают опасные и нетривиальные отделы кода. Это может быть код, который подвержен нередким изменениям, код, от работы которого зависит работоспособность наибольшего числа прочего кода, или код с большим числом связей [15].

На рисунке 7 приведен метод разработки при помощи тестов.

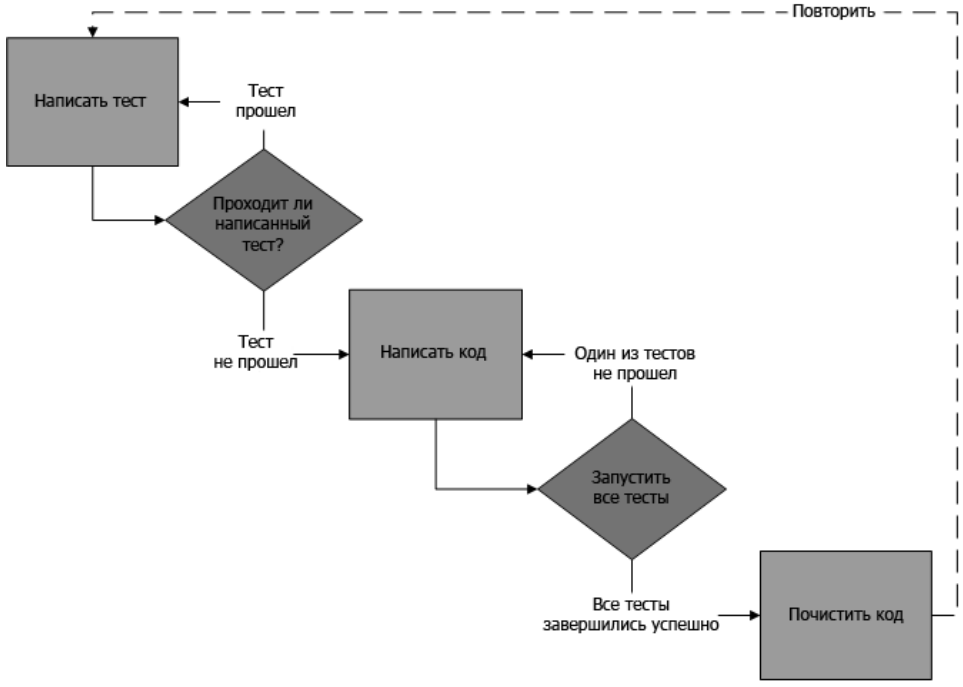


Рисунок 7 – Метод разработки при помощи тестов

Среда разработки обязана быстро реагировать на незначительные изменения кода.

TDD не только подразумевает испытание корректности, но и воздействует на дизайн программы. Опираясь на тесты, разработчики могут скорее представить, какая работоспособность нужна пользователю. Таким образом, подробности интерфейса возникают задолго до окончательной реализации решения. Листинг класса для решения задач линейного программирования приведен в приложении Ж, И.

К тестам применяются те же требования стандартов кодирования, что и к основному коду. Чтобы составить тест, разработчик обязан отчетливо осознавать предъявляемые требования. Для этого рассматриваются вероятные сценарии употребления и пользовательские истории.

Новые условия могут также вызвать модифицирование имеющийхся тестов. Листинг класса для решения задач линейного программирования приведен в приложении Ж, И.

Разработка посредством тестирования непосредственно сопряжена с подобными принципами как «поддерживай это простым и тупым» (keep it simple, stupid, KISS) и «вам это не понадобится» (you ain’t gonna need it, YAGNI).

Листинг интерфейса и его реализации для DataSet приведен в приложении К.

Дизайн может быть чище и яснее, при написании исключительно такого кода, который нужен для прохождения теста.

Тесты обязаны писаться для тестируемой функциональности. Это также способствует тому, что тестами будет восполнена вся функциональность.

Листинг интерфейса и его реализации для DataSet приведен в приложении Л.

Когда функциональность реализуется до тестов, разработчики склонны переходить к реализации последующей функциональности, не протестировав

существующую, что в дальнейшем повышает риск регресса при новых изменениях в кодовой базе.

Идея проверять, что вновь написанный тест не проходит, дает убедиться, что тест действительно что-то проверяет. Только после этого испытания можно начинать к реализации свежей функциональности.

Листинг интерфейса для DataSet приведен в приложении М.

Этот приём, популярный как «красный/зелёный/рефакторинг», именуют «мантрой разработки через тестирование». Под красным здесь разумеют непрошедшие тесты, а под зелёным – прошедшие.

Листинг реализации для DataSet приведен в приложении Н.

Для формализации задач линейного программирования реализован класс Constraint, который позволяет обрабатывать передаваемые ограничения и функция для дальнейшей обработки.

Листинг формализации условий задач линейного программирования приведен в приложении П.

2.5 Внедрение

Для внедрения библиотеки в существующие проекты, требуется с самого начала обозначить ограничения, которые есть для её использования.

Первым ограничением является то, что для осуществления поиска решения требуется дополнительно установить расширение ip_solve к языку PHP реализованную на C++.

Вторым ограничением, является то, что библиотека разработана на языке программирования PHP и, соответственно, внедрение данного решение ограничено только проектами, написанными на данном языке.

Третьим и последним ограничением является то, что версия PHP должна быть 7.3 и выше, что ограничивает использование библиотеки проекта младшей версии 5 и начальной 7.

Если все указанные ограничения не мешают, то далее проведем установку пакета через стандартный механизм менеджера пакетов composer.

Выводы по второму разделу

Сделан анализ требований к функционалу библиотеки и выделены ключевые условия. На основе выделенных требований выполнено проектирование архитектуры.

Произведено кодирование ожидаемого функционала библиотеки и выполнена отладка для корректной работы.

Проведен модульное тестирование функционала через пакет phpunit. По итогу прохождения тестовых сценариев сгенерирован отчет покрытия, который удовлетворяет поставленному требованию и составляет 87%.

ГЛАВА 3 АНАЛИЗ РЕШЕНИЙ

3.1 Реализация метода главного критерия в Excel

Для реализации метода главного критерия требуется выполнить следующий алгоритм действий.

Этот алгоритм позволяет произвести сведение задачи многокритериальной оптимизации к однокритериальной оптимизации.

Задачу в этом случае можно выразить следующим образом. Сначала необходимо установить главный критерий. Для этого выбираем один из рассматриваемых критериев в качестве основного критерия, а остальные преобразовываются в ограничения.

Для каждого критерия необходимо указать, что он не должен превышать или быть меньше определенного заданного значения.

Если же задача обладает более чем одним эффективным решением, то множество данных решений включает эффективные по Парето решения. Но оно также может включать и слабоэффективные решения.

Реализация метода главного критерия в Excel приведена в приложении Р.

3.2 Реализация метода аддитивной свертки в Excel

В методе равномерной оптимизации, который представляется частным случаем аддитивной свертки, весовые коэффициенты берутся одинаковыми друг другу.

Для реализации точной модели в терминах функций полезности предпочтительным является метод аддитивной свертки.

Аддитивную свертку критериев можно рассматривать как реализацию принципа справедливой компенсации абсолютных значений нормированных частных критериев. В этом случае суперкритерий обычно строятся как взвешенная сумма частных критериев.

Весовые коэффициенты выбираются так, чтобы их сумма была равна единице.

В улучшенном методе оптимизации, который кажется частным случаем аддитивной свертки, веса берутся таким же образом.

Для каждого конкретного критерия находят его нормализованное значение и умножают на весовой коэффициент. А затем из всех полученных значений выбирается либо максимальное, либо минимальное значение.

Методы свертывания критериев широко используются при решении задач многокритериальной оптимизации.

Реализация метода аддитивной свертки в Excel приведена в приложении С.

3.3 Сравнение эффективности

После реализации математических моделей по главному критерию и аддитивной свертки накоплен определенный набор данных, по которым возможно провести проверку корректности принятия решений, выполняемых разработанной библиотекой.

По итогам результирующих данных, полученных от моделей реализованных в Excel и с помощью библиотеки принятия решений, возникает возможность провести сравнительный анализ результатов, и определить на сколько эффективно разработанная библиотека выполняет свои расчеты.

Проведено сравнение результатов математической модели сделанной в Excel и через разработанную библиотеку.

В таблице 3 представлено сравнение эффективности библиотеки и Excel по методу главного критерия.

Таблица 3 Сравнение эффективности библиотеки и Excel по методу главного критерия.

Итерация	Порог	Библиотека		Excel (эталон)	
		Z_1	Z_2	Z_1	Z_2
1	5000	4999,98143	761,2351	4999,999958	764,2522
2	4970	4975,98149	765,4511	4980,999946	766,4513
3	4940	4960,98142	767,6546	4961,999928	768,6555
4	4910	4939,98143	770,1477	4942,999911	770,8584
5	4880	4920,98143	775,0564	4923,999754	773,0564
6	4850	4904,98143	777,2532	4904,999913	775,2523
7	4820	4886,98154	779,4566	4885,999952	777,4595
8	4790	4957,98124	780,6512	4866,999956	779,6572
9	4760	4939,98188	783,8542	4847,996957	781,8556
10	4730	4925,98141	786,0511	4828,999954	784,0522

В таблице 4 представлено сравнение эффективности библиотеки и Excel по методу аддитивной свертки.

Таблица 4 Сравнение эффективности библиотеки и Excel по методу аддитивной свертки.

Итерация	Вес 1	Вес 2	Библиотека		Excel (эталон)	
			Z_1	Z_2	Z_1	Z_2
1	0,5	0,5	1191,1122	611,3431	1200,1124	600,2233
2	0,4	0,6	6803,2244	760,2222	6800,1111	760,3242
3	0,2	0,8	4716,2333	778,5223	4722,3411	774,1123
4	0,1	0,9	4716,2333	778,5223	4722,3411	774,1123
5	0,6	0,4	6811,1223	752,3333	6800,2213	760,3242
6	0,7	0,3	6812,2226	752,3555	6800,6677	760,3257
7	0,9	0,1	6241,1123	752,3444	6800,2341	760,3226
8	0,5	0,5	1144,2333	530,8889	1200,4444	600,0931
9	0,4	0,6	6831,1112	759,5664	6800,4641	760,3212

Выводы по третьему разделу

Анализ показал, что при вычислении целевых функций методом главного критерия и аддитивной свертки с использованием библиотеки и Excel отклонения от эталона есть, но не значительные.

Поиск решения позволяет корректно определять значения на границах локального оптимума.

После сравнения эффективности установлено, что библиотека способна выполнять поставленную задачу.

ГЛАВА 4 РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ ВНЕДРЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1 Общая характеристика экономической части

Разработка программного обеспечения сопряжена с вложениями, которые необходимы для осуществления подготовительных работа, разработку, тестирование и внедрение. Поэтому необходимо экономическое обоснование ведения разработок.

Корректно составленный бизнес-план является частью успеха дальнейшей жизни разрабатываемого продукта, поэтому требуется раскрыть основные экономические определения и показатели.

4.2 Определение затрат на разработку продукта

В разделе определены необходимые инвестиции для разработки и внедрения программного продукта.

В таблице 5 приведен расчет единовременных затрат.

Таблица 5 – Расчет единовременных затрат

	Наименование технического средства и ПО	Тип или модель	Стоимость
1	Операционная система	MS Windows XP Professional	6300
2	Системный блок	DEXP Atlas H270	24999
3	Монитор	Asus VA24EHE	8499
4	Принтер	HP LaserJet 1018	6940
5	Клавиатура и мышь	Logitech Wireless Combo MK270	1999
6	Стол компьютерный		5000
7	Стул		4000
Итого			57737

Подсчитаны единовременные затраты:

$З_к = 6300 + 24999 + 8499 + 6940 + 1999 + 5000 + 4000 = 57737$ руб.

Стоимость создания программного продукта состоит из затрат на оплату разработчика и затрат на оплату машинного времени при отладке кода.

В таблице 6 приведен расчет текущих затрат.

Таблица 6 – Расчет текущих затрат

Наименование этапа	Дата начала	Длительность, Недель
1. Подготовительный	12.01.2021	3
2. Теоретическая разработка	02.02.2021	3
3.Проектирование и выполнение технического задания на ЭВМ	23.02.2021	4
4. Консультации с руководителем проекта	23.03.2021	1
5. Машинные расчёты и отчёт в электронном виде	30.03.2021-13.04.2021	2

Затраты на теоретическую часть и работу с литературой вычислены по формуле (4.1).

$$З_т = C_р \cdot T_{т,} \quad (4.1)$$

где $С_р = 30$ руб./час - тарифная ставка инженера II категории;

$Т_т$ - время, затраченное на работу с литературой и теоретический анализ;

$З_т = 30 \cdot 120 = 3600$ руб.

Затраты на теоретические разработки (4.2)

$$З_{рс} = C_р \cdot T_{р,} \quad (4.2)$$

где $Т_р$ - время, затраченное на теоретические разработки.

$З_{рс} = 30 \cdot 120 = 3600$ руб.

Затраты на проектирование и выполнение технического задания на ЭВМ (4.3)

$$З_п = C_р \cdot T_{п,} \quad (4.3)$$

где $Т_п$ - время, затраченное на проектирование и выполнение технического задания на ЭВМ.

$$З_п = 30 \cdot 160 = 4800 \text{ руб.}$$

Затраты на оплату машинного времени (4.4)

$$З_м = С_м \cdot Т_м, \quad (4.4)$$

где $С_м$ =30 руб./час - стоимость одного часа машинного времени;

$Т_м$ =160 + 158 час - время использования машины.

$$З_м = 30 \cdot 318 = 9540 \text{ руб.}$$

Затраты на консультацию с руководителем (4.5)

$$З_{кп} = С_{кр} \cdot Т_{кп}, \quad (4.5)$$

где $С_{кр}$ =35руб./час - тарифная ставка ведущего инженера;

$Т_{кп}$ - время консультаций с руководителем.

$$З_{кп} = 35 \cdot 230 = 1400 \text{ руб.}$$

Полные затраты при использовании ЭВМ для расчетов и составления отчета (4.6).

$$З_{сум} = З_т + З_{тр} + З_п + З_м + З_{кп}, \quad (4.6)$$

$$З_{сум} = 3600 + 3600 + 4800 + 9540 + 1400 = 22940 \text{ руб.}$$

В таблице 7 приведены затраты для проектирования.

Таблица 7 – Составляющие затрат при проектировании

Наименование работ	Обозначение	Сумма	
		в рублях	в %
1. Работа с литературой и теоретическая часть	$З_т$	3600	15,6
2. Теоретические разработки	$З_{тр}$	3600	15,6
3. Проектирование и выполнение технического задания на ЭВМ	$З_п$	4800	20,9
4. Оплата машинного времени	$З_м$	9540	41,6
5. Консультация с руководителем	$З_{кп}$	1400	6,2
ИТОГО	$З_{сум}$	22940	100

На основании значений, полученных по этим формулам, формируется график работы над проектом.

Затраты на накладные расходы, связанные с проектированием и отладкой ПП. Накладные расходы, связанные с проектированием и отладкой ПП, в том

числе стоимость используемых материалов (бумаги, картриджей к принтерам и т. п.).

Для выполнения работ необходимо материалов на сумму 2175,4руб.

В таблице 8 приведены потребности.

Таблица 8 – Потребности

Материалы	Потребность	Стоимость одной ед., руб.	Общая стоимость, руб.	Примечания
Бумага	200 листов	0.5	100	Печать текстов
Расходные материалы для лазерного принтера	1 шт.	997	997	Для печати необходимых данных
Перезаписываемый диск флеш-накопитель	2 шт.	460	920	Резервирование данных и материалов
Использование сети Internet	18 часов	8,8	158,4	Поиск информации и литератур.
Итого			2175,4	

Оплата пользования электричеством составляет:

$$З_э = 0{,}5\text{кВатт} \cdot 318 \text{ часа} \cdot 3{,}36 \text{ руб} = 534{,}24 \text{ руб.}$$

Текущие затраты (себестоимость) (С) включают затраты на постановку задачи, разработку алгоритмов и программ, а также затраты, связанные с содержанием и эксплуатацией ВТ. Рассчитываем их по следующей формуле (4.7):

$$C = З_{пр} + Н_з + З_{маш} + З_н + З_{э,} \quad (4.7)$$

где $З_{пр}$ - затраты на заработную плату проектировщиков и программистов;

$$З_{пр} = 25019{,}2 \text{ руб.}$$

В таблице 9 приведены текущие затраты.

Таблица 9 – Текущие затраты

Наименование статей затрат	Сумма
1. Затраты на заработную плату ($З_{пр}$)	25019,2
2. Отчисления в фонды с заработной платы труда (ФФОМС, ПФР, ФСС) ($З_{ф}$)	7556
3. Затраты, связанные с использованием машинного времени ($З_{маш}$)	9540
5. Накладные расходы ($З_{н}$)	2175,4
6. Расходы на электричество при пользовании вычислительной техникой	534,24
Итого затрат (З)	44824,84

В таблице 10 приведены расчеты затрат на разработку программного средства.

Таблица 10 – Расчет затрат на разработку программного средства

Затраты и ожидаемые доходы от внедрения ИС	Период			
Затраты	1 квартал	2 квартал	3 квартал	4 квартал
1. Капитальные затраты	57737			
2. Текущие затраты:	44824,84	44824,84	44824,84	44824,84
- Заработная плата	25019,2	25019,2	25019,2	25019,2
- Отчисления в фонды (ФФОМС, ПФР, ФСС)	7556	7556	7556	7556
- Затраты, связанные с использованием машинного времени	9540	9540	9540	9540
- Накладные расходы	2175,4	2175,4	2175,4	2175,4
- Расходы на электричество при пользовании вычислительной техникой	534,24	534,24	534,24	534,24
Итого затрат:	102561,84	44824,84	44824,84	44824,84

Отчисления от фонда оплаты труда ($З_{ф}$) (30,2 % от $З_{пр}$);

$$З_{ф} = 30{,}2\% \cdot З_{пр} = 7556 \text{ руб.}$$

$З_{маш}$ - затраты, связанные с использованием машинного времени на разработку и отладку программ. $З_{маш} = 9540$ руб.

Определение доходов от внедрения в работу программного обеспечения.

В таблице 11 приведены доходы.

Таблица 11 – Доходы

Доходы/Месяцы	1 квартал	2 квартал	3 квартал	4 квартал
1 Снижение затрат на ручную обработку больших объемов информации, руб.	12150	12150	12150	12150
2. Привлечение дополнительных клиентов, руб.	30000	40000	50000	60000
Итого доходов тыс. руб.:	42150	52150	52150	72150

$З_н$ - накладные расходы, связанные с проектированием и отладкой ПП, в том числе стоимость используемых материалов. После расчета единовременных и текущих затрат составим общую таблицу потребности в инвестициях (4.6).

4.3 Показатели эффективности

Расчет экономической эффективности проводится по следующим показателям:

- чистый дисконтированный доход (ЧДД) или интегральный эффект;
- индекс доходности (ИД);
- внутренняя ставка доходности (ВСД);
- срок окупаемости;

Чистый дисконтированный доход – это превышение интегральных результатов над интегральными затратами. Он определяется как сумма текущих эффектов за весь расчетный период, приведенная к начальному шагу.

Выбор данного метода был обусловлен следующими причинами:

- необходимость инфляционной корректировки показателей;
- обеспечение наглядности и простоты расчета срока окупаемости;
- стабильность и предсказуемость денежных потоков;
- равномерность денежных потоков.

Этот метод нашел широкое применение в зарубежной практике при оценке вложений в материальные и нематериальные активы, особенно в случаях

нестабильной макроэкономической ситуации (инфляция, возможность кризисных ситуаций и т. д.).

Суть этого метода заключается в отнесении предполагаемых денежных потоков (и положительных, и отрицательных) к временным интервалам, начиная с момента начала инвестирования.

Как правило, при низких показателях рентабельности продукта стандартный «срок службы» продукта или срок окупаемости продукта-заменителя выбирается периодом анализа этого метода. В этом случае для установления срока окупаемости достаточно ограничить срок до 3 лет.

К недостаткам метода можно отнести то, что он не позволяет учитывать случайные риски (как внешние, так и внутренние), а также известную условность расчета срока окупаемости, связанного с выбором ставки дисконтирования.

Чистый дисконтированный доход определяется как сумма текущих эффектов за весь расчетный период.

Предполагаемые доходы от продаж (Д) определяются по формуле (4.8):

$$Д = Ц_{пр} \cdot N, \quad (4.8)$$

где $Ц_{пр}$ - цена продажи ПП, руб;

N - объем продаж по периодам в соответствии с исследованиями рынка, шт.

Расходы (Р) включают, кроме текущих затрат, расходы на тиражирование (4.9) и рекламу (4.10):

$$Р = З_{тир} \cdot N + З_{р}, \quad (4.9)$$

$$Р = З_{тир} \cdot N + З_{р} \cdot n \text{ мес}; \quad (4.10)$$

где n - количество месяцев в рассматриваемом периоде.

В таблице 12 приведены технико-экономические показатели работы.

Таблица 12 – Технико-экономические показатели работы

Периоды (месяц)	Показатели		$1/(1+a)^t$	Дисконтированные	Годовая экономическая эффективность	ЧДД с нарастающим итогом	
	Доходы	Расходы					
	1	2	3	4=1·3	5=2·3	6=4-5	7
1 квартал 2021	42150	102561.84	0.8771	36973.68	89966.53	-52992.84	-52992.84
2 квартал 2021	52150	44824.84	0.7694	40127.73	34491.26	5636.47	-47356.37
3 квартал 2021	62150	44824.84	0.6749	41949.48	30255.49	11693.99	-35662.38
4 квартал 2021	72150	44824.84	0.5920	42718.59	26539.90	16178.69	-19483.69
1 квартал 2022	82150	44824.84	0.4555	37426.43	20421.59	17004.84	-2478.85
2 квартал 2022	92150	44824.84	0.5193	47859.82	23280.62	24579.21	22100.36
Итого:	310750	281861.2	3.3694	247055.74	201674.77		

Выводы по четвертому разделу

Индекс доходности - представляет собой отношение суммы приведенных эффектов к величине капитальных вложений.

Индекс доходности строится из тех же элементов, что и ЧДД. Если ЧДД положителен, то ИД > 1 и наоборот. Индекс доходности: 1,39.

Расчет ЧДД инвестиционного проекта показывает, является ли он эффективным при некоторой заданной норме дисконта.

В разделе выпускной квалификационной работы проведён анализ основных разделов бизнес-плана, осуществлена калькуляция темы с оценкой экономической эффективности реализации программного продукта.

Затраты на разработку составляют 44 824,84 руб. ежеквартально. Экономический эффект от использования данного программного продукта за расчетный период (1 год и 6 месяцев) составит 22 100,36, при этом разработчик окупит свои затраты на создание автоматизированной системы примерно за 1,5 года и затем начнет получать прибыль. Можно сделать вывод о целесообразности и актуальности разработки данного программного продукта.

ЗАКЛЮЧЕНИЕ

В работе рассмотрены метод главного критерия, аддитивная свертка и симплекс-метод для поиска решений. Разработана библиотека для использования методов теории принятия решений. Проведен вычислительный эксперимент и сравнение эффективности разработанного решения.

В первой главе рассмотрены общие характеристики предметной области теории принятия решений. Выделены алгоритмы для решения задач данного исследования и произведена постановка задачи. Рассмотрен общий принцип работы алгоритма симплекс-метода, метода главного критерия и аддитивной свертки.

Во второй главе описаны этапы разработки решения по задачам исследования, такие как: сбор требований, проектирование архитектуры, кодирование и отладка, тестирование и внедрение. Описаны технические подходы и практики по реализации подобных решений, а также проведено обоснование выбранных шаблонов проектирования для применения в библиотеке.

В третьей главе проведен вычислительный эксперимент, связанный с вопросами эффективности разработанной библиотеки для применения методов теории принятия решений. Разработаны математические модели в Excel для проведения сравнительного анализа.

Представленные в главе результаты экспериментальных исследований показали возможность применения данного решения во множестве сфер, где стоит задача выбора оптимального значения из набора вариантов по определённым критериям оценки.

В дальнейшем представляется интересным повышение точности принятия решений на основе уже реализованных методов, а также увеличения списка методов принятия решений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кини Р.Л., Райфа Х. Принятие решений при многих критериях: предпочтения и замещения. – М.: Радио и связь,1981.

2. Ларичев О. И. Теория и методы принятия решений, а также Хроника событий в Волшебных Странах: Учеб. Для вузов. – М.: Логос, 2000, 2006г.

3. Жуковский В. И., Молоствов В.С. Многокритериальное принятие решений в условиях неопределенности. – М.:1988.

4. Черноруцкий И.Г. Методы оптимизации и принятия решений. – СПб.: Лань, 2001, 2005г.

5. Акулич И.Л. Глава 1. Задачи линейного программирования // Математическое программирование в примерах и задачах. — М.: Высшая школа, 1986. — 319 с.

6. Хемди А. Таха. Глава 3. Симплекс-метод // Введение в исследование операций — 7-е изд. — М.: «Вильямс», 2007. — С. 95—141

7. Системы поддержки принятия решений: учебник и практикум для бакалавриата и магистратуры / В. Г. Халин [и др.]; под редакцией В. Г. Халина, Г. В. Черновой. – Москва: Издательство Юрайт, 2019. – 494 с.

8. Леффингуелл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. — М.: Вильямс, 2002

9. "PSR-12: Extended Coding Style Guide - PHP-FIG". www.PHP-fig.org. Retrieved 2020-06-04.

10. Roman Pronskiy. PHPStorm 2021.1.3 is released (англ.) (2 June 2021). Дата обращения: 2 июня 2021. Архивировано 2 июня 2021 года.

11. Поль М. Дюваль, Стивен М. Матиас III, Эндрю Гловер. Непрерывная интеграция: улучшение качества программного обеспечения и снижение риска = Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series). — Вильямс, 2008. — 2000 экз.

12. Fowler, Martin (2003). "Value Object". Catalog of Patterns of Enterprise Application Architecture. Martin Fowler (martinfowler.com). Retrieved 17 July 2011.

13. Beck, K. Test-Driven Development by Example, Addison Wesley, 2003.

14. Lee Copeland. Extreme Programming. Computerworld (December 2001). Дата обращения: 11 января 2011. Архивировано 27 августа 2011 года.

15. Koskela, L. «Test Driven: TDD and Acceptance TDD for Java Developers», Manning Publications, 2007.

ПРИЛОЖЕНИЕ А

Шаблон программирования – фасад

На рисунке А.1 приведен фасад, структурный паттерн проектирования.

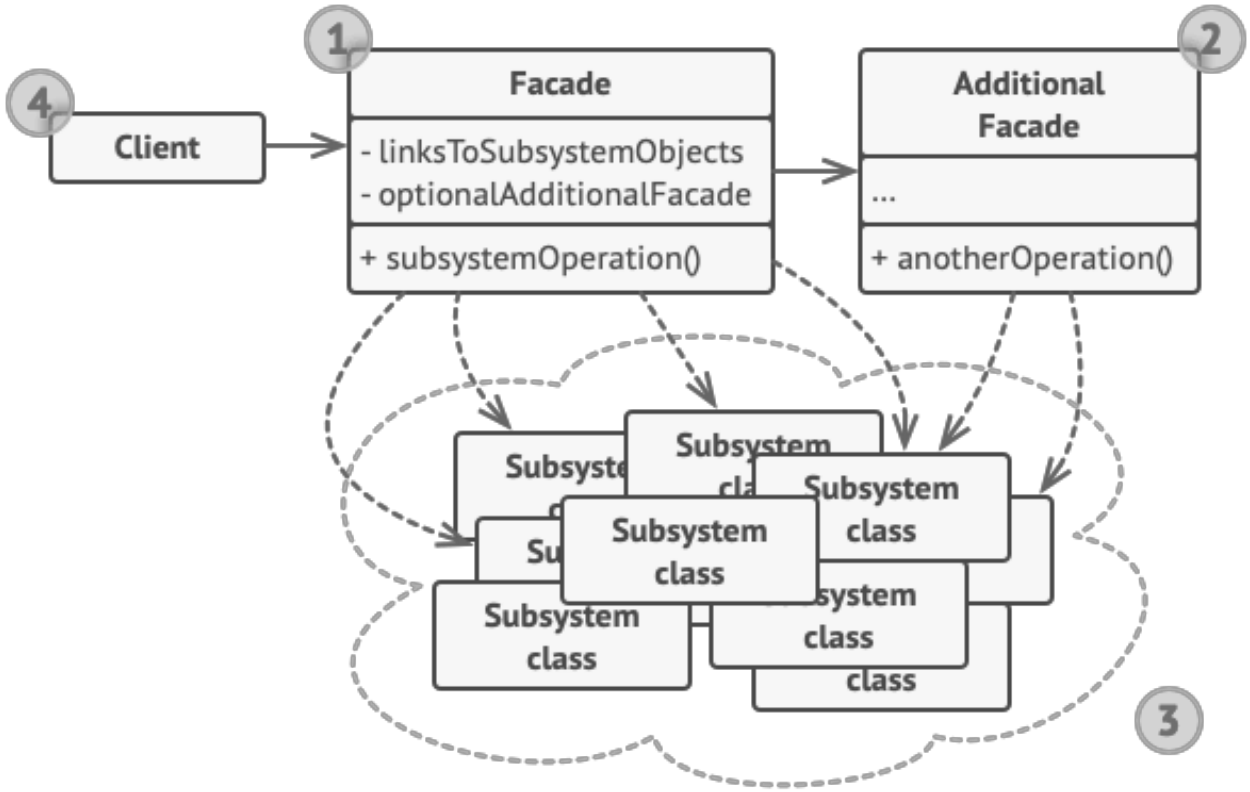


Рисунок А.1 – Фасад, структурный паттерн проектирования

Фасад – это структурный паттерн проектирования, который дает несложный интерфейс к непростой системе классов, библиотеке или фреймворку.

ПРИЛОЖЕНИЕ Б

Шаблон программирования – строитель

На рисунке Б.1 приведен строитель, порождающий паттерн проектирования.

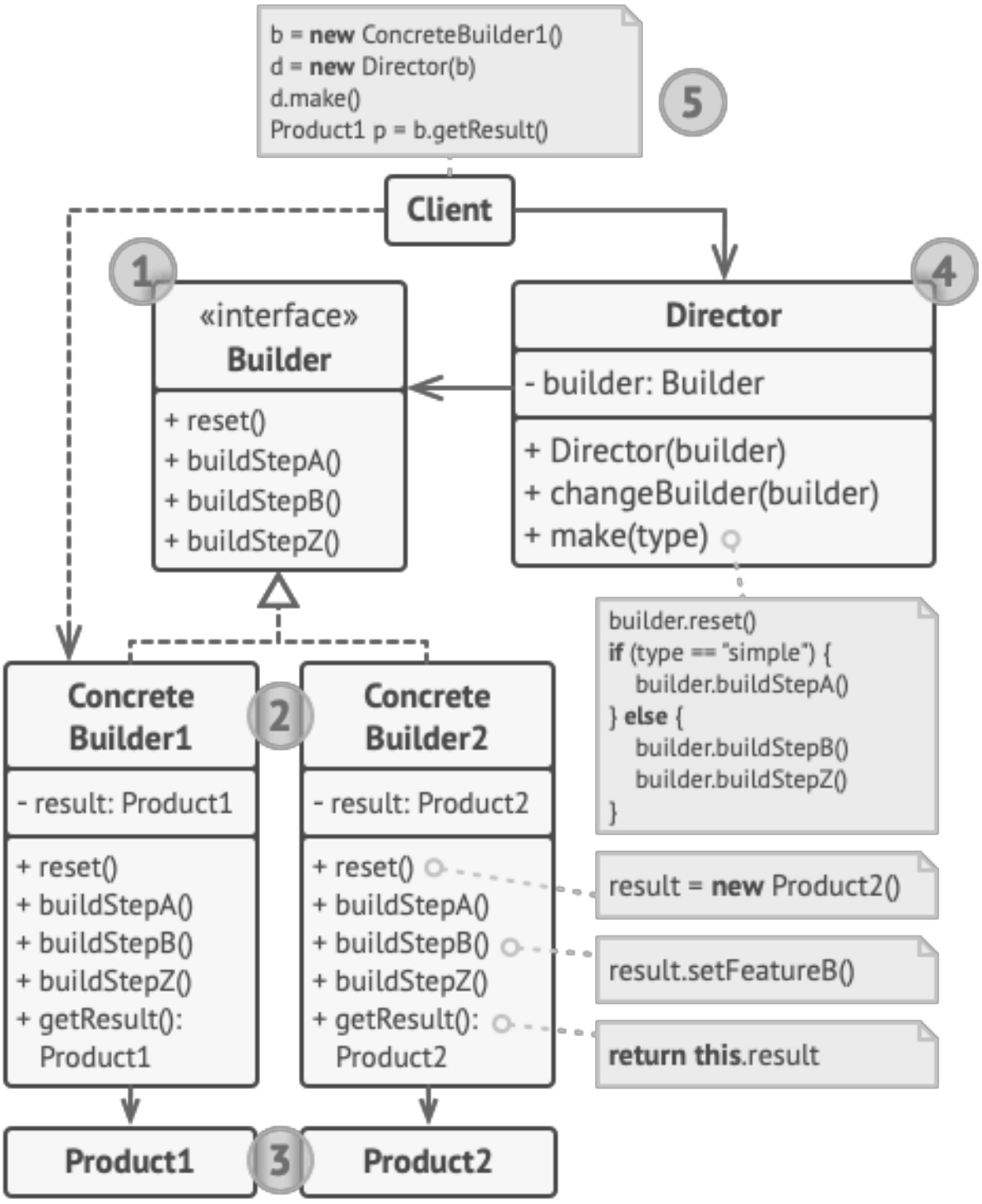


Рисунок Б.1 – Строитель, порождающий паттерн проектирования

Строитель позволяет использовать один и тот же код для получения различных представлений объектов.

1. Интерфейс конструктора объявляет этапы проектирования продукта, общие для всех типов конструкторов.

2. Некоторые строители реализуют этапы построения, каждый по-своему, некоторые строители могут создавать неоднородные объекты, не имеющие общего интерфейса.

Builder – это шаблон порождающего проектирования, который позволяет создавать сложные объекты шаг за шагом.

ПРИЛОЖЕНИЕ В

Шаблон программирования – стратегия

На рисунке В.1 приведен поведенческий паттерн проектирования стратегия.

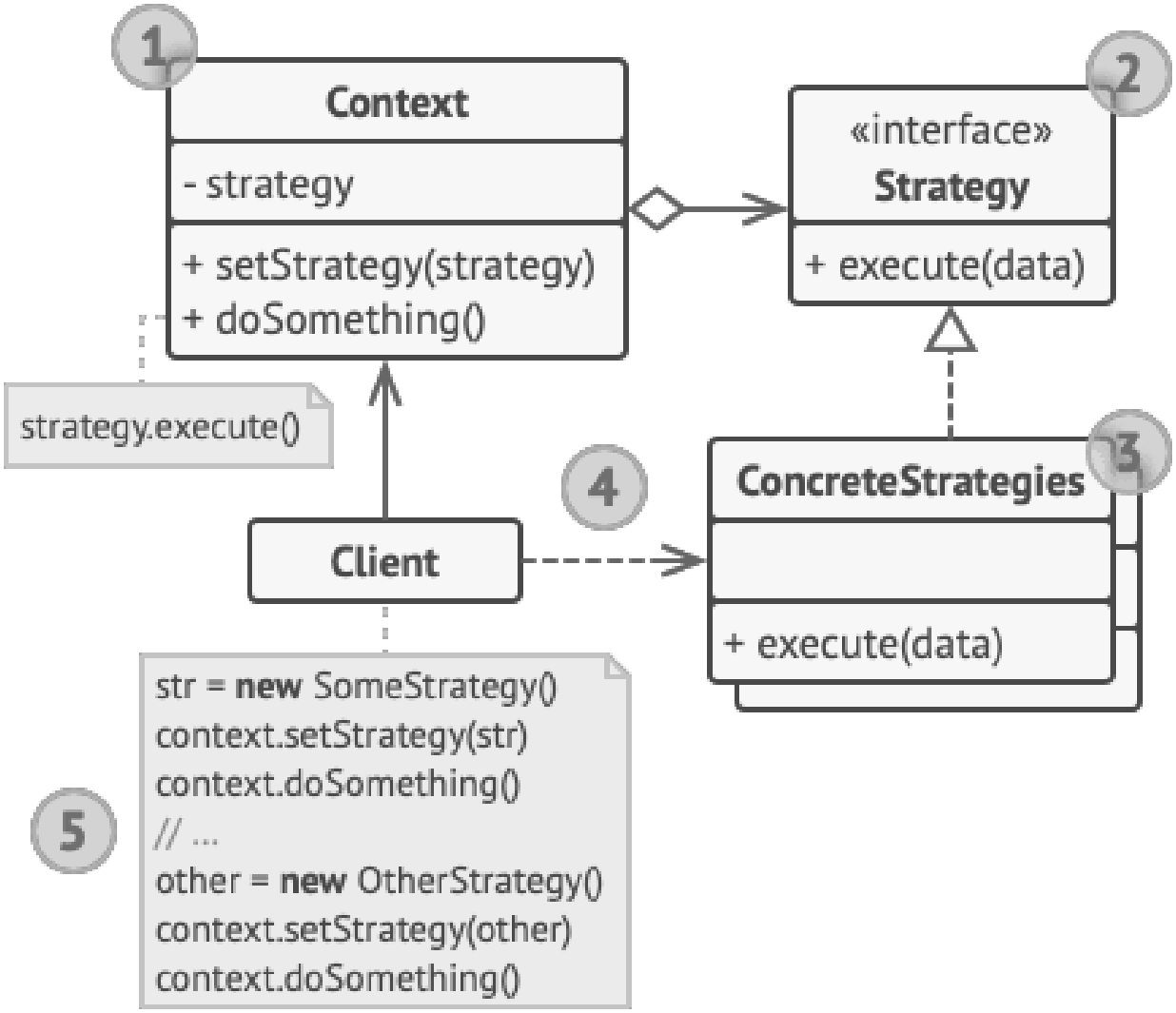


Рисунок В.1 – Стратегия, поведенческий паттерн проектирования

Стратегия – это шаблон поведенческого проектирования, который описывает семейство похожих алгоритмов и помещает каждый из них в свой собственный класс, после чего алгоритмы можно явно менять местами во время выполнения.

1. Контекст сохраняет ссылку на объект определенной стратегии, функционируя с ним через интерфейс общей стратегии.

2. Стратегия описывает интерфейс, общий для всех разновидностей алгоритма. Контекст использует этот интерфейс для вызова алгоритма.

3. Не имеет значения для контекста, какой алгоритм выбран, поскольку все они имеют одинаковый интерфейс.

4. Конкретные стратегии реализуют различные варианты алгоритма.

5. Во время выполнения кода контекст получает вызовы от клиента и направляет их на объект определенной стратегии

.

ПРИЛОЖЕНИЕ Г

Листинг точки входа в библиотеку

Листинг 1 – Код точки входа в библиотеку.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory;

use …

final class DecisionService implements DecisionServiceInterface
{
    /** @var Handler */
    private $handler;

    public function setHandler(Handler $handler): DecisionService
    {
        $this->handler = $handler;
        return $this;
    }

    public function getHandler(): Handler
    {
        return $this->handler;
    }

    public function process(DataSetInterface $dataSet): DataSetResultInterface
   {
        return $this->handler->process($dataSet);
    }
}
```

ПРИЛОЖЕНИЕ Д

Листинг основного тестового сценария

Листинг 2 – Код основного тестового сценария.

```
<?php

declare(strict_types=1);

namespace Test\MSdev\Component\DecisionTheory;

use ...

class DecisionServiceTest extends TestCase
{
    /** @var DecisionService */
    private $decisionService;

    protected function setUp(): void
    {
        parent::setUp();

        $this->decisionService = new DecisionService();
        $this->decisionService->setHandler(new NullHandler());
    }

    public function testProcessingWithEmptyDataSetByNullHandler(): void
    {
        $dataSetResult = $this->decisionService->process(new DataSet());
        self::assertFalse($dataSetResult->isSuccess());
        self::assertEquals(
            new NullHandler(),
                $this->decisionService->getHandler()
            );
        }
    }
```

```
    public function testProcessingWithDataSetByNullHandler(): void
    {
        $dataSet = new DataSet([
            new Variant('UUID_1', false, [
                new Property( 'x1', [10]),
                new Property( 'x2', [15]),
            ]),
            new Variant('UUID_2', true, [
                new Property( 'x1', [17]),
                new Property( 'x2', [8]),
            ])
        ]);

        $dataSetResult = $this->decisionService->process($dataSet);

        self::assertEquals(new DataSetResult(), $dataSetResult);
    }

}
```

ПРИЛОЖЕНИЕ Е

Листинг интерфейса для DecisionService

Листинг 3 – Код интерфейса для DecisionService.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory;

use MSdev\Component\DecisionTheory\Handler\Handler;
use MSdev\Component\DecisionTheory\ValueObject\DataSetInterface;
use MSdev\Component\DecisionTheory\ValueObject\DataSetResultInterface;

interface DecisionServiceInterface
{
    public function setHandler(Handler $handler): DecisionService;

    public function getHandler(): Handler;

    public function process(DataSetInterface $dataSet): DataSetResultInterface;
}
```

ПРИЛОЖЕНИЕ Ж

Листинг класса для решения задач линейного программирования

Листинг 4 – Код класса для решения задач линейного программирования.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\Handler;

use …

abstract class Handler
{
    public function process(DataSetInterface $dataSet): DataSetResultInterface
    {
        if ($dataSet->isEmpty() || !$dataSet->getRestrictions()) {
            return new DataSetResult();
        }

        return $this->calculate($dataSet);
    }

    abstract public function handle(Variant $variant, array $restrictions):
FindSolutionInterface;

…

}
```

ПРИЛОЖЕНИЕ И

Листинг класса с поиском решения по задаче линейного программирования

Листинг 5 – Код класса с поиском решения по задаче линейного программирования.

```
<?php
declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\Handler;

use ...

abstract class Handler
{
    public function calculate(DataSetInterface $dataSet): DataSetResultInterface
    {
        $list = [];
        $restrictions = $dataSet->getRestrictions();
        /** @var VariantInterface $variant */
        foreach ($dataSet->getList() as $variant) {
            if (!$variant->isObjectiveFunction()) {
                continue;
            }
            $findSolution = $this->handle($variant, $restrictions);
            $solutionValue = $findSolution->getValue();
            if ((null !== $solutionValue) && !$variant->isEmpty()) {
                $list[$variant->getKey()] = $solutionValue;
            }
        }
        asort($list, SORT_NUMERIC);
        $list = array_reverse($list);

        return new DataSetResult($list);
    }
}
```

ПРИЛОЖЕНИЕ К

Листинг интерфейса для DataSet

Листинг 6 – Код интерфейса DataSet.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\ValueObject;

interface DataSetInterface
{
    public function getList(): array;

    public function getRestrictions(): array;

    public function isEmpty(): bool;
}
```

ПРИЛОЖЕНИЕ Л

Листинг объекта транспортировки данных DataSet

Листинг 7 – Код реализации объекта транспортировки данных DataSet.

```
<?php
declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\ValueObject;

class DataSet implements DataSetInterface
{
    /** @var array */
    private $list;
    /** @var array */
    private $restrictions;

    public function __construct(array $list = [], array $restrictions = [])
    {
        $this->list = $list;
        $this->restrictions = $restrictions;
    }
    public function isEmpty(): bool
    {
        return empty($this->list);
    }
    public function getList(): array
    {
        return $this->list;
    }
    public function getRestrictions(): array
    {
        return $this->restrictions;
    }
}
```

ПРИЛОЖЕНИЕ М

Листинг интерфейса для DataSetResult

Листинг 8 – Код интерфейса DataSetResult.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\ValueObject;

interface DataSetResultInterface
{
    public function getList(): array;

    public function isSuccess(): bool;
}
```

ПРИЛОЖЕНИЕ Н

Листинг объекта транспортировки данных DataSetResult

Листинг 9 – Код реализации объекта транспортировки данных DataSetResult.

```
<?php

declare(strict_types=1);

namespace MSdev\Component\DecisionTheory\ValueObject;

class DataSetResult implements DataSetResultInterface
{
    /** @var Variant[] */
    private $list;

    public function __construct(array $list = [])
    {
        $this->list = $list;
    }

    public function getList(): array
    {
        return $this->list;
    }

    public function isSuccess(): bool
    {
        return !empty($this->list);
    }
}
```

ПРИЛОЖЕНИЕ П

Листинг формализации условий задач линейного программирования

Листинг 10 – Код формализации условий задач линейного программирования.

```
<?php

namespace MSdev\Component\DecisionTheory\ValueObject;

class Constraint
{
    private $coefficients = [];
    private $comparison;
    private $value;

    /**
    * Constructor
    *
    * @param array $coefficients Constraint left side
    * @param int|float $comparison Comparison sign (LE, GE, EQ, FR)
    * @param int|float $value Constraint right side
    */
    public function __construct(array $coefficients, $comparison, $value)
    {
        $this->coefficients = $coefficients;
        $this->comparison = $comparison ?: FR;
        $this->value = $value;
    }

    /**
    * Create constraint from string
    *
    * @param string $string String constraint
    * @return self
    */
    public static function fromString($string)
    {
```

```
    $split = preg_split('/(<=|=|>=)/', $string, -1, PREG_SPLIT_DELIM_CAPTURE);

    $coefficients = self::parseCoefficients($split[0]);
    $comparison = self::parseComparison($split[1]);
    $value = floatval($split[2]);

    return new Constraint($coefficients, $comparison, $value);
}

public function getCoefficients()
{
    return $this->coefficients;
}

public function setCoefficients($coefficients)
{
    $this->coefficients = $coefficients;

    return $this;
}

/**
* Parse coefficients from string
*
* @param string $expression Left side of equation
* @return array
*/
private static function parseCoefficients($expression)
{
    $coefficients = [];
    $split = preg_split('/[a-zA-Z]/', trim($expression), -1, PREG_SPLIT_NO_EMPTY);

    foreach ($split as $coefficient) {
        $coefficients[] = floatval(preg_replace('/\s+/', '', $coefficient));
    }
```

```
    return $coefficients;
}

public function getComparison()
{
    return $this->comparison;
}

public function setComparison($comparison)
{
    $this->comparison = $comparison;

    return $this;
}

/**
* Parse comparison sign
*
* @param string $comparison Comparison sign
* @return int
*/
private static function parseComparison($comparison)
{
    if ($comparison === '<=') {
        return LE;
    }

    if ($comparison === '>=') {
        return GE;
    }

    return EQ;
}
```

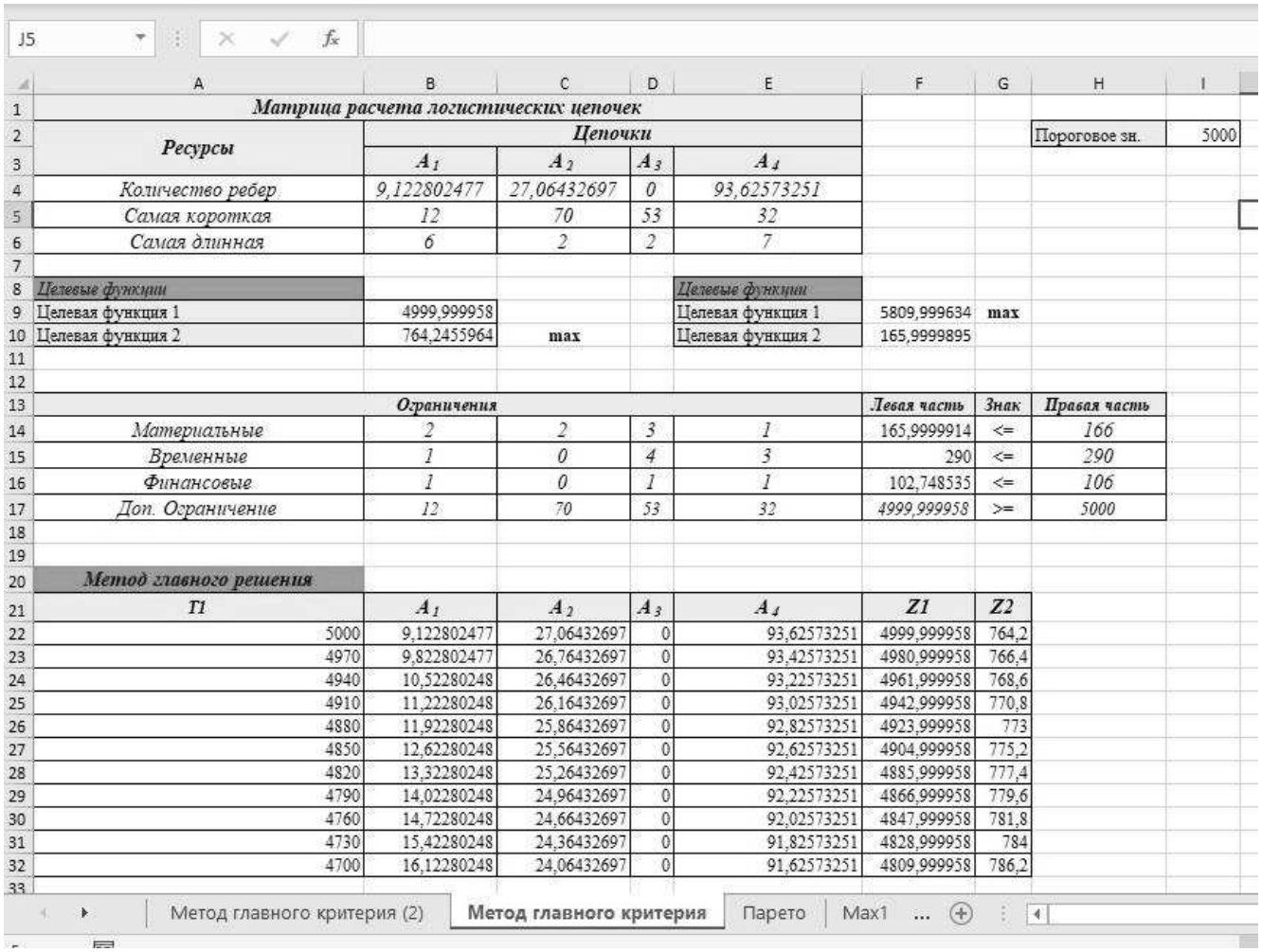
```
    public function getValue()
    {
        return $this->value;
    }

    public function setValue($value)
    {
        $this->value = $value;

        return $this;
    }
}
```

ПРИЛОЖЕНИЕ Р

Реализация главного критерия в Excel

После того, как сведем проблему многокритериальной оптимизации к задаче однокритериальной оптимизации, получаем право использовать обычные программные инструменты, в частности, такие как поиск решения в Microsoft Excel.

На рисунке Р.1 приведен вариант реализации метода главного критерия.

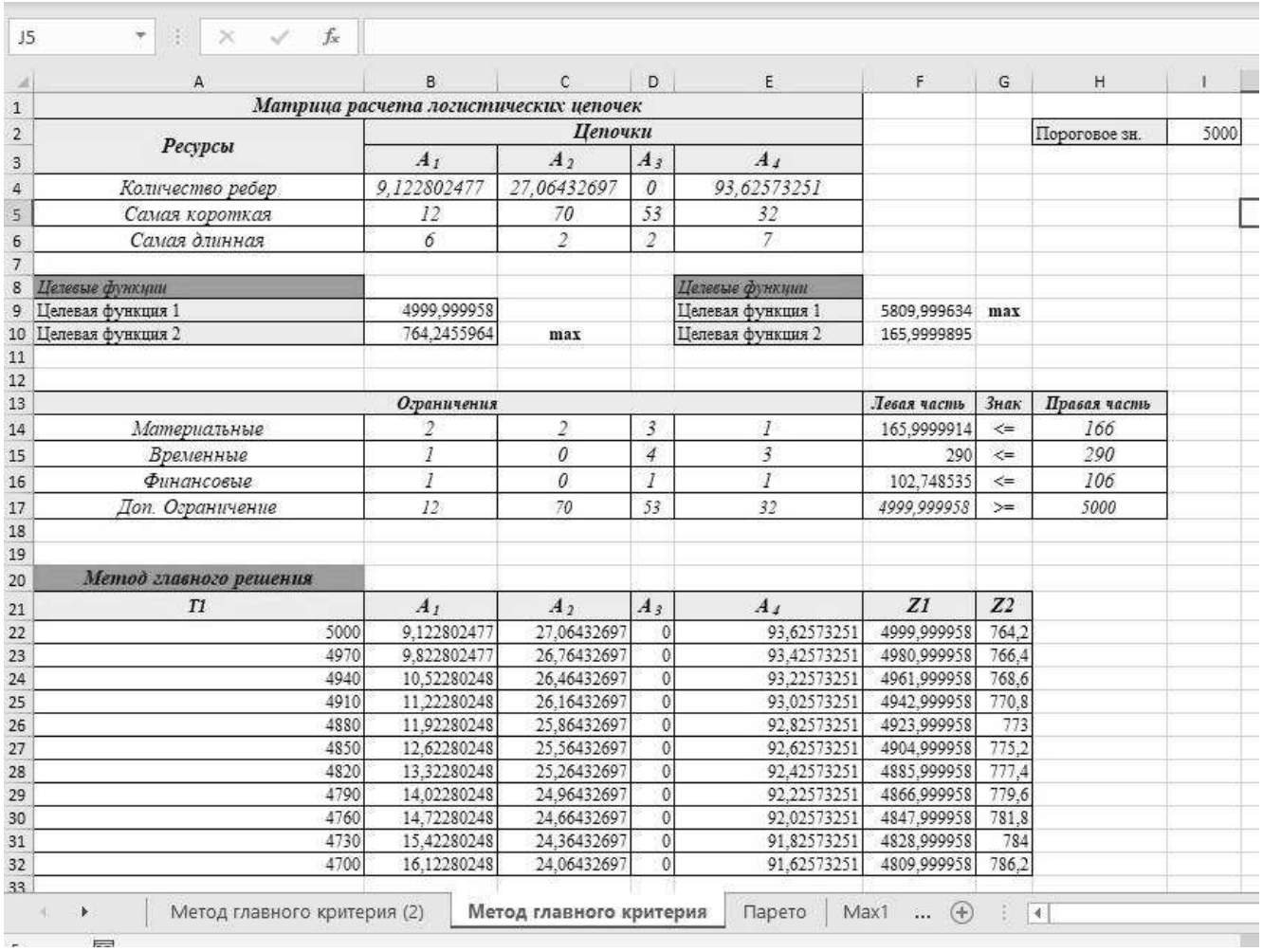


	A	B	C	D	E	F	G	H	I
1	*Матрица расчета логистических цепочек*								
2	*Ресурсы*	*Цепочки*						Пороговое зн.	5000
3		A_1	A_2	A_3	A_4				
4	*Количество ребер*	*9,122802477*	*27,06432697*	*0*	*93,62573251*				
5	*Самая короткая*	*12*	*70*	*53*	*32*				
6	*Самая длинная*	*6*	*2*	*2*	*7*				
7									
8	*Целевые функции*				*Целевые функции*				
9	Целевая функция 1	4999,999958			Целевая функция 1	5809,999634	max		
10	Целевая функция 2	764,2455964	max		Целевая функция 2	165,9999895			
11									
12									
13	*Ограничения*					*Левая часть*	*Знак*	*Правая часть*	
14	*Материальные*	*2*	*2*	*3*	*1*	165,9999914	<=	*166*	
15	*Временные*	*1*	*0*	*4*	*3*	290	<=	*290*	
16	*Финансовые*	*1*	*0*	*1*	*1*	102,748535	<=	*106*	
17	*Доп. Ограничение*	*12*	*70*	*53*	*32*	*4999,999958*	>=	*5000*	
18									
19									
20	*Метод главного решения*								
21	*T1*	A_1	A_2	A_3	A_4	*Z1*	*Z2*		
22	5000	9,122802477	27,06432697	0	93,62573251	4999,999958	764,2		
23	4970	9,822802477	26,76432697	0	93,42573251	4980,999958	766,4		
24	4940	10,52280248	26,46432697	0	93,22573251	4961,999958	768,6		
25	4910	11,22280248	26,16432697	0	93,02573251	4942,999958	770,8		
26	4880	11,92280248	25,86432697	0	92,82573251	4923,999958	773		
27	4850	12,62280248	25,56432697	0	92,62573251	4904,999958	775,2		
28	4820	13,32280248	25,26432697	0	92,42573251	4885,999958	777,4		
29	4790	14,02280248	24,96432697	0	92,22573251	4866,999958	779,6		
30	4760	14,72280248	24,66432697	0	92,02573251	4847,999958	781,8		
31	4730	15,42280248	24,36432697	0	91,82573251	4828,999958	784		
32	4700	16,12280248	24,06432697	0	91,62573251	4809,999958	786,2		
33									

Метод главного критерия (2) | Метод главного критерия | Парето | Max1 | ...

Рисунок Р.1 – Вариант реализации метода главного критерия

На рисунке Р.2 приведен вариант реализации метода главного критерия с детализацией.



	A	B	C	D	E	F	G	H	I
1	Матрица расчета логистических цепочек								
2	Ресурсы	Цепочки						Пороговое зн.	5000
3		A_1	A_2	A_3	A_4				
4	Количество ребер	9,122802477	27,06432697	0	93,62573251				
5	Самая короткая	12	70	53	32				
6	Самая длинная	6	2	2	7				
7									
8	Целевые функции				Целевые функции				
9	Целевая функция 1	4999,999958			Целевая функция 1	5809,999634	max		
10	Целевая функция 2	764,2455964	max		Целевая функция 2	165,9999895			
11									
12									
13	Ограничения					Левая часть	Знак	Правая часть	
14	Материальные	2	2	3	1	165,9999914	<=	166	
15	Временные	1	0	4	3	290	<=	290	
16	Финансовые	1	0	1	1	102,748535	<=	106	
17	Доп. Ограничение	12	70	53	32	4999,999958	>=	5000	
18									
19									
20	Метод главного решения								
21	T1	A_1	A_2	A_3	A_4	Z1	Z2		
22	5000	9,122802477	27,06432697	0	93,62573251	4999,999958	764,2		
23	4970	9,822802477	26,76432697	0	93,42573251	4980,999958	766,4		
24	4940	10,52280248	26,46432697	0	93,22573251	4961,999958	768,6		
25	4910	11,22280248	26,16432697	0	93,02573251	4942,999958	770,8		
26	4880	11,92280248	25,86432697	0	92,82573251	4923,999958	773		
27	4850	12,62280248	25,56432697	0	92,62573251	4904,999958	775,2		
28	4820	13,32280248	25,26432697	0	92,42573251	4885,999958	777,4		
29	4790	14,02280248	24,96432697	0	92,22573251	4866,999958	779,6		
30	4760	14,72280248	24,66432697	0	92,02573251	4847,999958	781,8		
31	4730	15,42280248	24,36432697	0	91,82573251	4828,999958	784		
32	4700	16,12280248	24,06432697	0	91,62573251	4809,999958	786,2		
33									

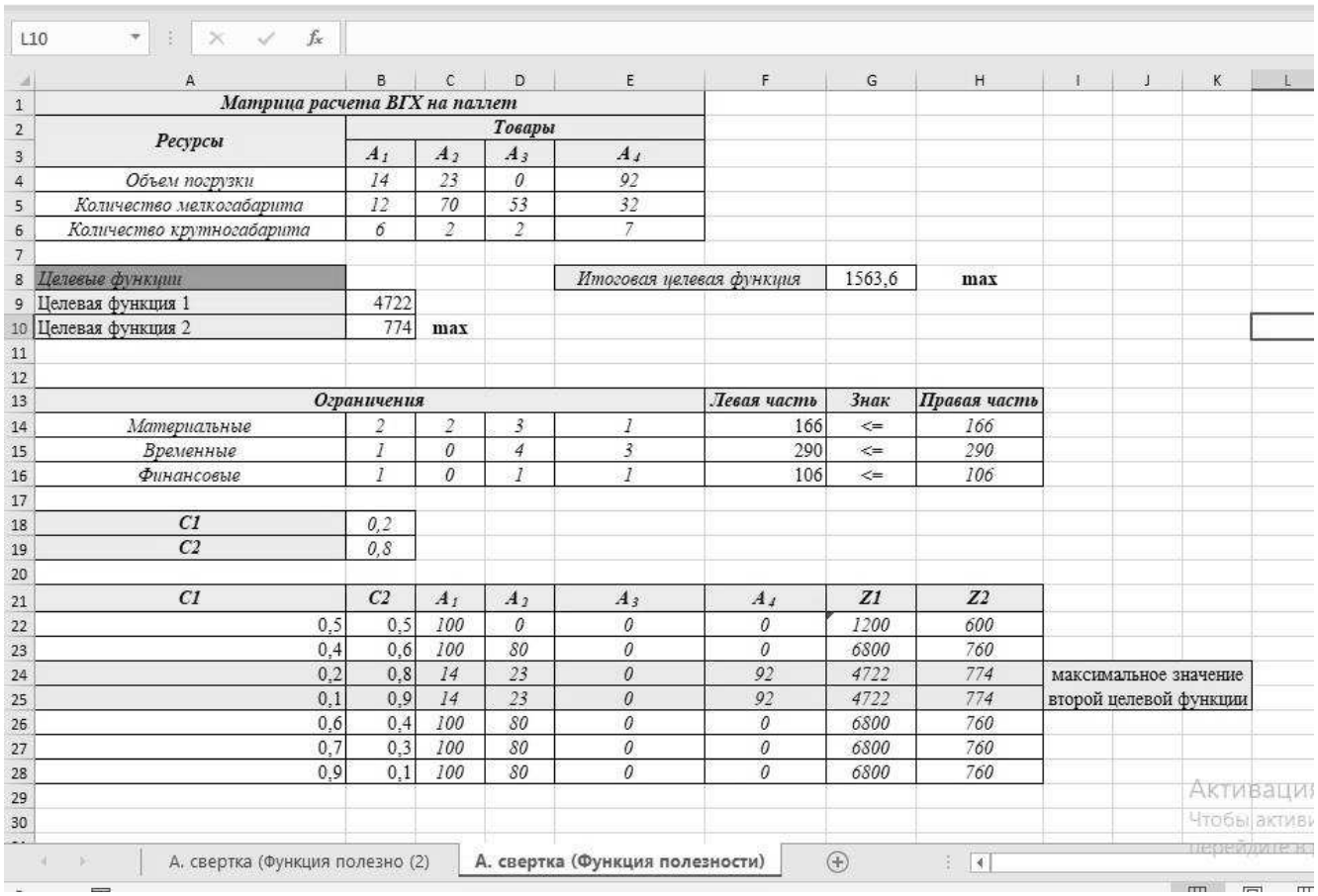
Рисунок Р.2 – Вариант реализации метода главного критерия с детализацией

ПРИЛОЖЕНИЕ С

Реализация аддитивной свертки в Excel

Ниже приводится рисунок с таблицей Excel, показывающий базовую логику для поиска оптимального значения с использованием метода аддитивной свертки с визуальным представлением формул, необходимых для получения результата.

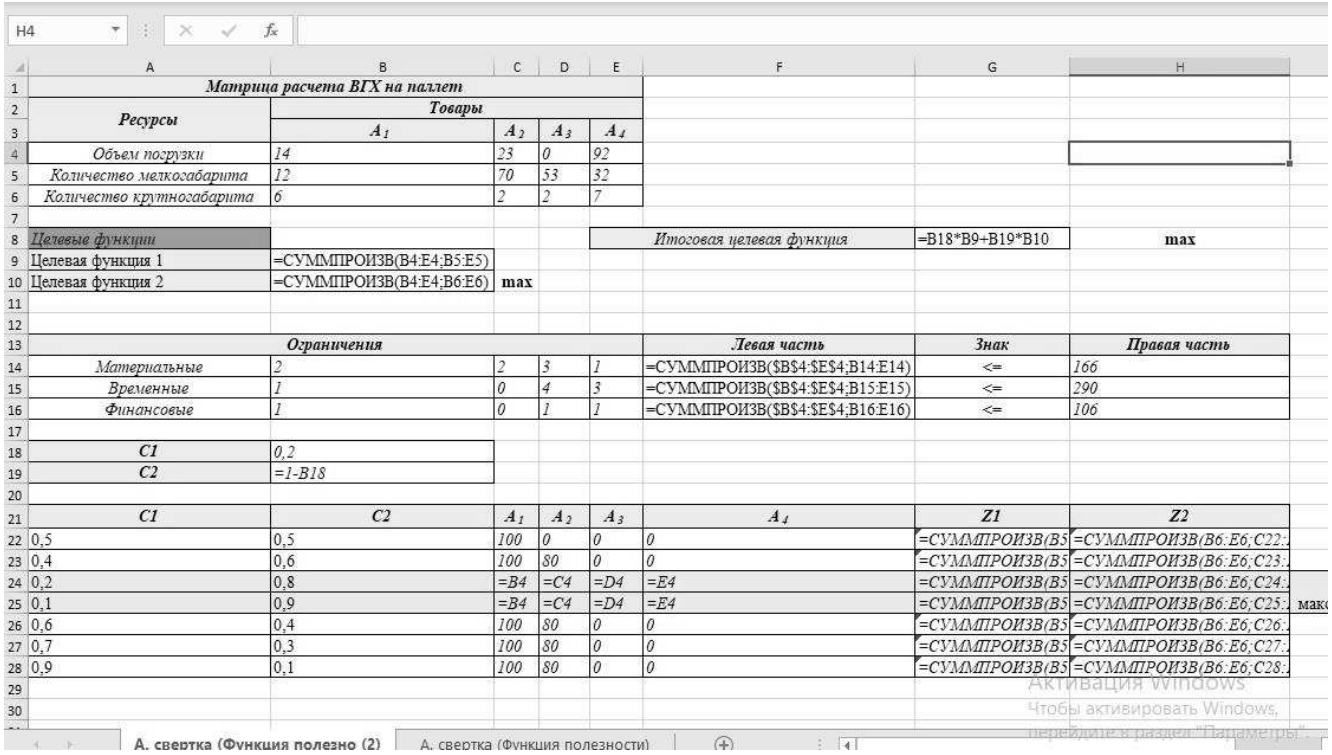
На рисунке С.1 приведен вариант реализации метода аддитивной свертки.



	A	B	C	D	E	F	G	H	I
1	**Матрица расчета ВГХ на паллет**								
2	**Ресурсы**	**Товары**							
3		A_1	A_2	A_3	A_4				
4	Объем погрузки	14	23	0	92				
5	Количество мелкогабарита	12	70	53	32				
6	Количество крупногабарита	6	2	2	7				
8	Целевые функции				Итоговая целевая функция		1563,6	max	
9	Целевая функция 1	4722							
10	Целевая функция 2	774	max						
13	**Ограничения**					**Левая часть**	**Знак**	**Правая часть**	
14	Материальные	2	2	3	1	166	<=	166	
15	Временные	1	0	4	3	290	<=	290	
16	Финансовые	1	0	1	1	106	<=	106	
18	C1	0,2							
19	C2	0,8							
21	C1	C2	A_1	A_2	A_3	A_4	Z1	Z2	
22	0,5	0,5	100	0	0	0	1200	600	
23	0,4	0,6	100	80	0	0	6800	760	
24	0,2	0,8	14	23	0	92	4722	774	максимальное значение
25	0,1	0,9	14	23	0	92	4722	774	второй целевой функции
26	0,6	0,4	100	80	0	0	6800	760	
27	0,7	0,3	100	80	0	0	6800	760	
28	0,9	0,1	100	80	0	0	6800	760	

Рисунок С.1 – Вариант реализации метода аддитивной свертки

На рисунке С.2 приведен вариант реализации метода аддитивной свертки с детализацией.



	A	B	C	D	E	F	G	H
1	**Матрица расчета ВГХ на паллет**							
2	**Ресурсы**	**Товары**						
3		A_1	A_2	A_3	A_4			
4	Объем погрузки	14	23	0	92			
5	Количество мелкогабарита	12	70	53	32			
6	Количество крупногабарита	6	2	2	7			
7								
8	Целевые функции					Итоговая целевая функция	=B18*B9+B19*B10	max
9	Целевая функция 1	=СУММПРОИЗВ(B4:E4;B5:E5)						
10	Целевая функция 2	=СУММПРОИЗВ(B4:E4;B6:E6)	max					
11								
12								
13	**Ограничения**					**Левая часть**	**Знак**	**Правая часть**
14	Материальные	2	2	3	1	=СУММПРОИЗВ($B$4:$E$4;B14:E14)	<=	166
15	Временные	1	0	4	3	=СУММПРОИЗВ($B$4:$E$4;B15:E15)	<=	290
16	Финансовые	1	0	1	1	=СУММПРОИЗВ($B$4:$E$4;B16:E16)	<=	106
17								
18	**C1**	0,2						
19	**C2**	=1-B18						
20								
21	**C1**	**C2**	A_1	A_2	A_3	A_4	**Z1**	**Z2**
22	0,5	0,5	100	0	0	0	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C22:…
23	0,4	0,6	100	80	0	0	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C23:…
24	0,2	0,8	=B4	=C4	=D4	=E4	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C24:…
25	0,1	0,9	=B4	=C4	=D4	=E4	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C25:… мак
26	0,6	0,4	100	80	0	0	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C26:…
27	0,7	0,3	100	80	0	0	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C27:…
28	0,9	0,1	100	80	0	0	=СУММПРОИЗВ(B5	=СУММПРОИЗВ(B6:E6;C28:…
29								
30								

Рисунок С.2 – Вариант реализации метода аддитивной свертки с детализацией