Научная статья
УДК 004.89
doi: 10.17586/2713-1874-2024-4-60-68

КОМБИНИРОВАННЫЙ МЕТОД ПЛАНИРОВАНИЯ РЕЛИЗОВ В AGILE-КОМАНДАХ НА ОСНОВАНИИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Максим Игоревич Ардатовский[1]✉, Татьяна Геннадьевна Максимова[2]

[1,2]Университет ИТМО, Санкт-Петербург, Россия
[1]ardatovskym@gmail.com✉, https://orcid.org/0009-0004-3851-2762
[2]tgmaximova@itmo.ru, https://orcid.org/0000-0002-8532-7963
Язык статьи – русский

Аннотация: В статье произведен анализ практик планирования релизов и распределения задач в условиях гибкого управления проектами по методологии agile с применением методов многокритериальной оптимизации и генетических алгоритмов. Разработан комбинированный метод, решающий задачу выбора требований для релиза (Next Release Problem) на основе алгоритма NSGA-II и оптимизации расписания задач с учетом ограничений по ресурсам и последовательности выполнения этапов (Resource Constrained Project Scheduling Problem). Описаны особенности использования библиотеки DEAP на языке Python для реализации эволюционных вычислений, а также применение библиотеки NumPy для ускорения векторных операций и Matplotlib для визуализации результатов. Представлена архитектура программного решения, включающая модульную организацию кода и интеграцию пользовательских эвристик для построения оптимальных расписаний. Приведено описание функций оценки полезности, рисков и штрафов за нарушения зависимостей и ресурсных ограничений, а также применение методов кроссовера и мутации для словарных структур расписаний. Показана итеративная оптимизация, связывающая этапы отбора эпиков и построения расписаний для достижения устойчивого оптимума. Экспериментальная проверка подтвердила эффективность предложенного подхода, так как в ней достигается максимизация полезности набора эпиков, минимизация штрафов и простоя сотрудников при построении оптимального расписания. Разработанная система применима для комплексного планирования релизов в Agile-командах и формирует основу для дальнейших исследований в области гибкого проектного управления и ресурсной оптимизации.

Ключевые слова: многокритериальная оптимизация, генерация расписаний, генетические алгоритмы, планирование релизов agile, управление ресурсами

Ссылка для цитирования: Ардатовский М. И., Максимова Т. Г. Комбинированный метод планирования релизов в Agile-командах на основании генетических алгоритмов // Экономика. Право. Инновации. 2024. № 4. С. 60–68. http://dx.doi.org/10.17586/2713-1874-2024-4-60-68.

COMBINED METHOD OF RELEASE PLANNING IN AGILE TEAMS BASED ON GENETIC ALGORITHMS

Maksim I. Ardatovsky[1]✉, Tatiana G. Maximova[2]

[1,2]ITMO University, Saint Petersburg, Russia
[1]ardatovskym@gmail.com✉, https://orcid.org/0009-0004-3851-2762
[2]tgmaximova@itmo.ru, https://orcid.org/0000-0002-8532-7963
Article in Russian

Abstract: The paper analyzes the practices of release planning and task allocation in agile project management using multi-criteria optimization methods and genetic algorithms. A combined method that solves the problem of selecting requirements for release (Next Release Problem) based on NSGA-II algorithm and optimization of task scheduling considering resource constraints and sequence of stages (Resource Constrained Project Scheduling Problem) is developed. The peculiarities of using DEAP library in Python to implement evolutionary computations are described, as well as the use of NumPy library to accelerate vector operations and Matplotlib to visualize the results. The architecture of the software solution is presented, including modular code organization and integration of custom heuristics for building optimal schedules. The description of utility, risk and penalty functions for dependency violations and resource constraints is given, as well as the application of crossover and mutation methods for dictionary structures of schedules. Iterative optimization linking the stages of epics selection and schedule construction to reach a stable optimum is shown. Experimental validation confirmed the effectiveness of the proposed approach as it achieves utility maximization of the set of epics, minimization of penalties and idle time of employees in constructing optimal schedules. The developed system is

applicable for complex release scheduling in Agile teams and forms the basis for further research in agile project management and resource optimization.

Keywords: agile, genetic algorithms, multi-criteria optimization, release planning, resource management, schedule generation

For citation: Ardatovsky M. I., Maximova T. G. Combined Method of Release Planning in Agile Teams Based on Genetic Algorithms. *Ekonomika. Pravo. Innovacii.* 2024. No. 4. pp. 60–68. (In Russ.). http://dx.doi.org/10.17586/2713-1874-2024-4-60-68.

Введение. Современные подходы к разработке программного обеспечения значительно изменились благодаря внедрению методологии Agile, которая сместила акцент с жесткой структуры процессов на гибкость, сотрудничество и удовлетворение запросов клиентов, что делает её незаменимой в условиях динамично меняющихся требований [1]. Agile завоевал популярность в различных сферах благодаря своей способности преодолевать ограничения традиционных моделей управления проектами, таких как waterfall, которому присуще негибкость и низкая адаптивность к новым задачам [2].

Среди множества аспектов Agile особое внимание следует уделить процессам планирования релизов и управления задачами. Планирование релизов обеспечивает стратегическую основу для регулярных релизов функциональности и включает прогнозирование зависимостей, согласование усилий и адаптацию команды к меняющимся требованиям [3]. Управление задачами, в свою очередь, направлено на эффективное распределение ролей, устранение узких мест и оптимизацию командной работы.

Тем не менее, планирование релизов и управление задачами остаются одними из самых сложных аспектов для многих Agile-команд. Что обусловлено необходимостью баланса между гибкостью и долгосрочным видением проекта. Современные решения включают использование совместных платформ и аналитических инструментов для предиктивного управления, однако их реализация может варьироваться в зависимости от контекста проекта и используемых технологий [4].

Литературный обзор. На данный момент существует множество методов и алгоритмов, направленных на решение задач планирования и оптимизации расписаний. Классические подходы включают линейное и целочисленное программирование, динамическое программирование и различные эвристические методы. В контексте проблемы выбора требований для следующего релиза (Next Release Problem, NRP) широко используются многокритериальные оптимизационные методы, такие как алгоритм NSGA-II (Non-dominated Sorting Genetic Algorithm II) [5]. Для решения проблемы составления расписания проектов с ограниченными ресурсами (Resource Constrained Project Scheduling Problem, RCPSP) применяются генетические алгоритмы, табу-поиск и другие эволюционные методы [6,7]. Обе проблемы являются NP-полными задачами, поэтому в основном для их решения используются эвристические алгоритмы. С другой стороны, некоторые авторы предлагают альтернативный подход к выбору оптимального пути выполнения проекта путем разбиения проекта на партии [8].

Одним из ключевых направлений является исследование планирования релизов на основе пользовательских предпочтений. Например, работа [9] предлагает подход, основанный на анализе зависимостей и предпочтений пользователей, который позволяет улучшить планирование и адаптировать релизы под требования конечных потребителей. Данное исследование подчеркивает значимость учета пользовательских данных при разработке программного обеспечения.

Сложности, связанные с гибким планированием проектов с ограниченными ресурсами, рассматриваются в исследовании [10]. Исследование предлагает гибридный алгоритм дифференциальной эволюции, направленный на решение задач, связанных с проектным планированием, включая динамическую структуру проектов и управление ресурсами, что особенно актуально для крупных и долгосрочных проектов, где требуется высокая адаптивность.

Исследования по изучению NRP [11, 12] исследуют оптимизационные подходы для управления приоритетами задач и выделения

ресурсов в условиях ограничения времени и бюджета. Работы демонстрируют, что применение многокритериального анализа и эмпирических данных позволяет улучшить результаты планирования и удовлетворить ожидания заинтересованных сторон. Работа [13] показывает, что применение алгоритмов роя позволяет находить более эффективные решения для задачи NRP, а в статье [14] оценивается использование нейронных сетей для оценки усилий в рамках Agile-разработки. Такие подходы показывают потенциал машинного обучения в оптимизации процессов планирования релизов. Ряд исследований рассматривают управление заинтересованными сторонами и идентификации их требований. Например, в работе [15] предлагается структурированный подход к идентификации стейкхолдеров в автомобильной промышленности, что способствует более осмысленному и согласованному планированию релизов.

Однако, несмотря на значительный прогресс, остаются нерешенные вопросы. Комплексное исследование [5] представляет собой систематический обзор литературы по задаче NRP, которое показывает, что, несмотря на обширные исследования в этой области, нет единого подхода к решению NRP, который бы охватывал все аспекты, включая учет ограничений, предпочтений стейкхолдеров и доступности ресурсов. Что свидетельствует о необходимости дальнейших эмпирических исследований для разработки более универсальных и адаптивных решений. Кроме того, работа [6] отмечает, что внедрение методов машинного обучения в процесс планирования релизов находится на ранней стадии и требует дальнейших испытаний. Наконец, стоит отметить влияние ограничений ресурсов на процесс планирования расписаний. Исследование [7] рассматривает применение эволюционных алгоритмов для решения задачи RCPSP, показывая, что такие подходы могут существенно повысить эффективность работы в условиях ограниченных ресурсов.

Основная проблема в существующих работах заключается в отсутствии интегрированного подхода к планированию, который одновременно учитывал бы выбор требований для релиза и возможности команды по их реализации в рамках ограничений ресурсов и времени. Отдельные эвристические методы не обеспечивают достаточной гибкости и адаптивности, необходимые для эффективного планирования в условиях неопределенности и динамично меняющихся требований как со стороны планирования задач, так и со стороны планирования расписаний.

Аналогичные выводы относятся и к традиционным точным методам, так как обе рассматриваемые задачи являются NP-полными и предполагают высокую комбинаторную сложность с множеством различных условий и параметров, а также критериев оптимизации. Для применения точных методов в контексте данных задач потребовалось бы выполнение целого ряда допущений, таких как сокращение критерия оптимизации до одного, упрощение структуры эпиков и ресурсов расписаний. Например, в контексте задачи NRP её упрощение до однокритериальной задачи сводилось бы к задаче о рюкзаке. В этом случае основная цель была бы в максимизации полезности выбранных эпиков при ограничениях на ресурсы, что накладывало бы на него дополнительные ограничения. Подобные ограничения снижают возможность практического применения алгоритма, а также приводят к уходу от конвенциональной постановки решаемых задач.

Цель исследования заключается в разработке алгоритма планирования agile релиза, который объединяет решение NRP с использованием многокритериального метода NSGA-II и решение RCPSP с помощью генетических алгоритмов, учитывая при этом возможности команды и ограниченные ресурсы.

Методы и материалы исследования. Формальную постановку задачи первого этапа работы алгоритма можно описать как, необходимо найти и запланировать набор эпиков, максимизирующий полезность и минимизирующий риск. Набор эпиков представляет собой вектор длины N, где N – количество эпиков, и каждый бит (0 или 1) означает, включен ли эпик в набор. Тогда полезность набора эпиков выражается:

$$U(x) = \sum_{i=1}^{N} x_i * u_i, \qquad (1)$$

где $x_i \in \{0,1\}$ – решение о включении эпика i, u_i – полезность эпика i.

Для учета затрат по ролям необходимо предусмотреть, чтобы суммарные часы по

каждой роли не превышали доступные ресурсы. Если для роли r сумма часов превысила лимит, то накладывается штраф.

Тогда риск оценивается как:

$$R(x) = \sum_{i=1}^{N} x_i \left(\sum_r h_{i,r}, d_r\right), \qquad (2)$$

где $h_{i,r}$– количество часов на выполнения эпика i по роли r, а d_r – отклонение по роли r.

Дополнительно на функцию приспособленности накладываются штрафы за невыполненные зависимости, так, если эпик j зависит от эпика k, а k не выбран (x_k=0), то добавляется штраф $P_{dep}(x)$

Итоговая функция приспособленности для многокритериального NSGA-II рассчитывается как:

$$fitness(x) = (U(x) - P_{dep}(x), R(x)). \qquad (3)$$

Алгоритм используются для максимизации первой компоненты (полезность минус штраф) и одновременно минимизации риска, NSGA-II проводит эволюцию, обеспечивая построение финальной популяции особей, которая представляет собой Парето-фронт. В контексте рассматриваемой многокритериальной задачи Парето-фронт позволяет получить множество компромиссных решений с разным соотношением «полезность–риск», из которых затем дополнительно отбирается лучшая особь из популяции на основании критерия максимизации функции приспособленности по первой цели и, при равенстве этого критерия, минимизации по второй цели, так как общая полезность является более приоритетным критерием по сравнению с размером рисков

Далее на втором этапе работы алгоритма на основе набора эпиков составляется расписание. Расписание по сути является списком задач, которые выполняются определенным сотрудником определенной роли в определенном временном промежутке. Для ускорения процесса поиска решения и повышения качества выполнения в NP-трудных задач в нее накладываются дополнительно эвристики планирования.

1) Задачи размещаются в доступные промежутки, стараясь минимизировать разрывы между этапами эпика.

2) Если задача не может быть размещена для эпика, эпик полностью исключается из расписания.

3) Для каждого эпика проверяется, все ли роли выполнены, все ли зависимости удовлетворены, не нарушен ли порядок.

Для оценки заданного расписания дополнительно необходимо обозначить:

X – множество эпиков, вошедших в расписание.

D – общее время завершения последней задачи

P – суммарный штраф за невыполненные зависимости между эпиками, которые возникают при наличии таких связей, как комбинация (два эпика необходимы одновременно), импликация (один эпик необходим для выполнения другого) и исключение (два эпика противоречат друг другу) – P_dep, частично выполненные эпики – P_part, нарушения порядка этапов – P_seq, выход за пределы релиза/занятость сотрудника – P_res, тогда итоговый P будет равен:

$$P = P_{dep} + P_{part} + P_{seq} + P_{res}. \qquad (4)$$

I – суммарный простой сотрудников, который рассчитывается как сумма всех дней, когда сотрудник не занят.

$$I = \sum_{i=1}^{N} \sum_{d=0}^{D} \mathbf{1}(y_{i,d} = 0), \qquad (5)$$

где i – порядковый номер сотрудника, N – количество сотрудников, d – порядковый номер дня, 1(*) – индикаторная функция, значение которой равно единице в случае, если условии внутри нее верно и нулю в противном случае, $y_{i,d}$ – бинарная переменная занятости сотрудника i в день d

Общая полезность запланированных эпиков записывается как:

$$U_{plan} = \sum_{x \in X} u_x. \qquad (6)$$

Теперь необходимо минимизировать функцию приспособленности расписания, которая рассчитывается как:

$$fitness(plan) = D + P + I - \alpha U_{plan}, \qquad (7)$$

где α – положительный коэффициент, определяющий вес полезности. Чем больше

U_{plan}, тем меньше значение функции, а следовательно, тем лучше решение в целом.

Таким образом, разработанный на втором этапе генетический алгоритм по составлению расписания задач минимизирует время, штрафы, простой и при этом стремится максимизировать полезность задачи.

Ключевой составляющей работы алгоритма, предоставляющему ему преимущества над ранее изученными является итеративная оптимизация, которая подразумевает итерации, на каждой из которой:

1) Осуществляется отбор эпиков методом NSGA-II.

2) Осуществляется построение расписания для отобранных эпиков методом GA.

3) Осуществляется оценка решения по формуле, которая учитывает параметр полезности и значение функции принадлежности:

$$score = U_{plan} * fitness(plan) \ . \qquad (8)$$

4) Итерации продолжаются, пока улучшение не станет незначительным.

Таким образом, алгоритм на первой фазе решает задачу многокритериальной оптимизации:

$$\max_x U(x) - \ P_{dep}(x), \ \min_x R(x) \ . \qquad (9)$$

Тогда как вторая фаза решает задачу минимизации:

$$\min_{plan}\left[D(plan) + P(plan) + I(plan) - \alpha U_{plan}(plan)\right] \qquad (10)$$

Вследствие чего обе фазы связываются в итеративный цикл, улучшая решение, пока не достигнет стабильного оптимума по критерию score.

Обеспечение допустимости решения осуществляется посредством удовлетворения всех ограничений, накладываемых на задачу. Для чего на обеих фазах алгоритм контролирует соблюдение ресурсных лимитов, последовательности зависимостей и прочих условий. Так, если при формировании набора эпиков суммарные часы для роли r превышают доступный лимит, то решение получает штраф P_res, достаточный для его отсеивания эволюционным механизмом NSGA-II. Аналогично обрабатываются невыполненные зависимости между эпиками, например если для выполнения эпика i необходимо реализовать эпик k, но x_k = 0, то добавляется штраф P_dep. На этапе построения расписания для отобранных эпиков проверяется, возможно ли полностью разместить все задачи эпика с учётом доступности сотрудников нужной роли и заданных временных промежутков. Для этого вводится переменная w_(i,r,d), обозначающая объем в человеко-часах выполнения эпика i с ролью r в день d и накладывается ограничение вида:

$$\sum_{d=1}^{D} w_{i,r,d} \ \geq \ w_{i,r} * \ x_i \ . \qquad (11)$$

Данное ограничение гарантирует, что при включении эпика i в план, часы работы для каждой роли r будут полностью запланированы, а если не включен в план, то ресурсы для него не выделятся.

Используемый подход позволяет алгоритму понижать решения с нарушениями так, чтобы они не проходили в итоговую популяцию, либо значительно снижали свой ранг при отборе по Парето-фронту. В результате таких ограничивающих требований алгоритм постепенно эволюционирует к решениям, которые не только оптимальны, но и допустимы.

Результаты исследования. Для реализации проекта был выбран язык программирования Python, поскольку он обладает развитой экосистемой для задач по оптимизации, специализированных вычислений и визуализации данных, а также обеспечивает быструю скорость разработки. В качестве основных компонентов были использованы библиотека NumPy для ускорения векторных и матричных операций, фреймворк DEAP для эволюционных вычислений и библиотека Matplotlib для визуализации результатов в виде диаграммы Ганта.

Дополнительно в работе применялись стандартные модули Python, такие как collections, random и datetime, обеспечивающие упрощение работы со структурами данных, генерацией случайных чисел и сохранением меток времени для результатов. Программный код имеет модульную структуру и разделен на несколько логических компонентов, что обеспечивает читаемость и гибкость кода. На этапе генерации данных для

первичных тестов эпики и сотрудники инициализировались с использованием псевдослучайных чисел. Для воспроизводимости экспериментов были зафиксированы семена генераторов случайных чисел с помощью random.seed и np.random.seed.

Оценка эпиков в рамках алгоритма NSGA-II осуществлялась на основе функции, вычисляющей полезность, риск и штрафы в методе evaluate_individual, которая использует двухкомпонентный подход, на первом этапе которого бинарный вектор указывает на включение эпика, тогда как штрафы за невыполненные зависимости считаются отдельно. На этом этапе также были зарегистрированы типы и операторы эволюции через creator и toolbox из библиотеки DEAP.

В процессе планирования расписаний функция приспособленности оценивалась с учетом таких критериев, как длительность выполнения задачи, простой сотрудников, штрафы и суммарная полезность, что описано в методе генерации расписания generate_schedule_individual. Функции кроссовера и мутации были описаны с использованием специальных операторов для словарных структур, которые описывают расписания задач по ролям и сотрудникам. Также были описаны эвристики для формирования логики построения расписаний, в них учитывалось, что для каждого эпика задачи распределяются среди сотрудников в соответствии с доступным временем и этапами выполнения, что учитывалось в методе оценки расписания evaluate_schedule.

Итеративный процесс оптимизации, комбинирующий NSGA-II для отбора эпиков и генетический алгоритм для планирования расписаний был реализован в отдельном методе evaluate_solution, а визуализация результатов, включая построение диаграммы Ганта на основе финального решения реализовано в методе main.

Важной особенностью данной реализации является гибкость настройки параметров алгоритма, поскольку основные структуры данных DEAP (типы индивидуумов, операторы кроссовера и мутации) определяются на этапе инициализации основного кода, что облегчает параметризацию и модификацию алгоритмов. В качестве функций кроссовера и мутации выступали cxTwoPointDict и mutShuffleDict, так как они корректно работают со словарями, описывающими расписание задач.

Применение NumPy позволяет ускорить выполнение векторных операций, а модульная структура кода облегчает профилирование и последующую оптимизацию кода. Выбор Python и фреймворка DEAP обеспечил гибкость реализации, позволив быстро внедрять новые эвристики, изменять параметры алгоритмов и адаптировать решения под требования задачи комплексного планирования эпиков. Исходный код разработанного алгоритма доступен на платформе GitHub по адресу https://github.com/Flyneot/GenPlan.

Экспериментальная проверка. Проверка разработанной системы на возможность эффективно отбирать эпики и составлять для них расписание, учитывая заданные ресурсы и ограничения требует формирования метрик полезности итогового набора эпиков, суммарные временные затраты на их реализацию, отсутствие значительных штрафов за нарушение последовательности задач внутри эпиков и выполнение зависимостей между ними, оптимальность распределения по сотрудникам (сумма простоя)

Так, были сгенерированы 50 эпиков со случайным распределением часов по ролям (backend, frontend, design, analytics и testing) и случайной полезностью. Количество сотрудников и длительность релиза (60 дней) заданы таким образом, чтобы количество доступных трудовых ресурсов было в среднем достаточным для планирования нескольких эпиков. Генетический алгоритм NSGA-II для отбора эпиков запускался на 200 поколений с размером популяции 300, а для составления расписания применялся аналогичный по параметрам генетический алгоритм, но уже с другим типом индивидуумов (расписания).

Анализ штрафов, рассчитанных алгоритмом, показал, что большинство эпиков выполнены без критичных нарушений последовательности. Небольшие задержки между этапами влекут за собой умеренные штрафы, однако общая полезность перевешивает данные издержки. Итоговый набор эпиков, вошедший в расписание, обеспечивает суммарную полезность, близкую к верхним значениям, достигнутым в процессе прохождения

итерационного цикла. В ходе апробации было проверено, что увеличение коэффициента важности полезности в целевой функции действительно стимулирует включение более «выгодных» эпиков, даже при некотором увеличении временных интервалов или простоев.

Для проверки эффективности была проведена серия тестов, в одном из которых не применялись штрафы и не учитывалась полезность. В таком случае алгоритм выдавал решения, включавшие большее количество эпиков, но с низкой совокупной ценностью и значительными простоем. После включения в оценку полезности и штрафов улучшилось качество отобранных эпиков – выбирались более полезные, а общее время и простой стали ниже за счет попыток минимизации итоговой функции приспособленности. Таким образом, текущее решение превосходит «наивный» подход как по полезности, так и по согласованности планирования.

Проведенный эксперимент показал, что разработанный алгоритм достигает следующих целей.

1) Оптимизируется не только количество эпиков, но и их совокупная полезность.

2) Учитываются ограничения по ролям и последовательности этапов, что подтверждается низкими штрафами за некорректный порядок задач.

3) Решение способно распределить задачи по сотрудникам так, чтобы итоговая диаграмма Ганта была достаточно плотной и сбалансированной, при этом максимизируя полезность набора работ.

На рисунке 1 представлено итоговое расписание, полученное после нескольких итераций отбора эпиков и составления расписаний. По оси Y расположены сотрудники, сгруппированные по ролям. По оси X расположена временная шкала (дни релиза). Каждый прямоугольник соответствует отдельной задаче одного из эпиков, располагаясь по временной оси согласно старту и длительности, а по вертикали – согласно назначенному сотруднику. На диаграмме видно, что задачи распределены между сотрудниками разных ролей в соответствии с их специализацией. Хотя между задачами встречаются интервалы бездействия, общая загрузка выглядит сбалансированной.

Благодаря рисунку видно, что система формирует осмысленное распределение задач, в которой сотрудники одинаковой роли нагружены задачами разных эпиков в разные промежутки времени, без катастрофических простоев или параллельных конфликтов. Итоговые метрики позволяют заключить, что при данном наборе параметров и данных система генерирует решения, приближенные к «выгодному» балансу между полезностью, риском, длительностью и эффективностью использования ресурсов.

Таким образом, экспериментальная проверка подтверждает применимость и эффективность предложенного подхода.

Заключение. В рамках исследования был разработан комбинированный метод планирования релизов в Agile-командах на основе генетических алгоритмов и метода многокритериальной оптимизации NSGA-II. Разработанный алгоритм включает два этапа заключенных в цикл. На первом этапе применяется алгоритм NSGA-II для решения задачи многокритериального выбора эпиков с учетом их полезности, рисков и ограничений на зависимости. На втором этапе генетический алгоритм используется для оптимизации расписаний, минимизации простоев сотрудников, штрафов за некорректное распределение задач и максимизации общей полезности релиза. Оба этапа повторяются до тех пор, пока система не получит устойчивый оптимальный результат.

Экспериментальная проверка продемонстрировала, что предложенный метод эффективно решает поставленные задачи, достигая оптимизации количества и полезности отобранных эпиков, минимизации штрафов за несоблюдение зависимостей и порядок выполнения этапов и уменьшения простоев сотрудников, и повышение эффективности использования ресурсов. Результаты экспериментов подтвердили практическую значимость предложенного подхода и его способность формировать решения, приближенные к оптимальным.

Таким образом, предложенная методология решает задачу комплексного планирования релизов, обеспечивая баланс между полезностью, риском, ресурсными ограничениями и эффективностью распределения задач. В дальнейшем развитии темы может быть

исследован учет дополнительных факторов, таких как стоимостные ограничения, приоритеты заказчиков и динамическая изменчивость требований, разработка моделей для учета квалификации сотрудников и их реальной производительности, а также интеграция предложенного алгоритма с алгоритмами машинного обучения для предсказания оптимальных решений на основе исторических данных.

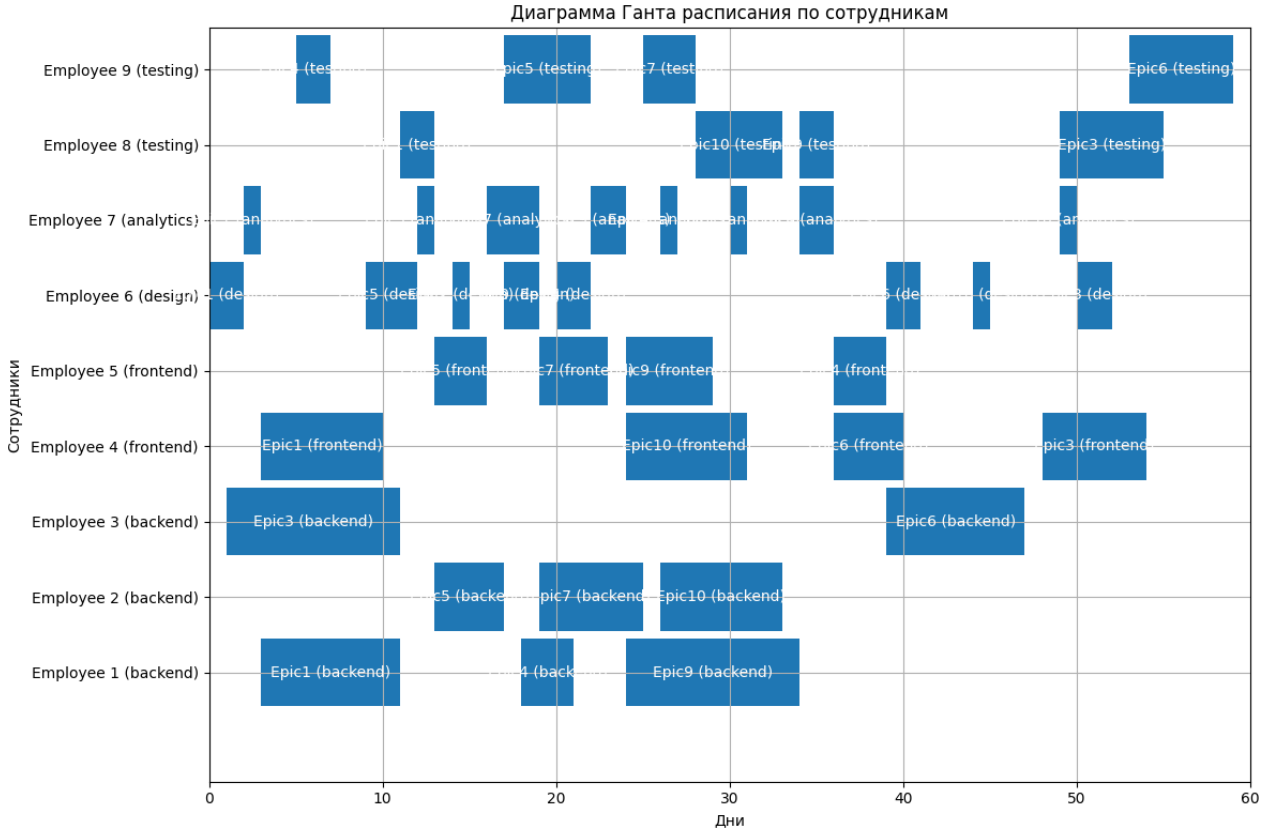

Рисунок 1 – Диаграмма Ганта для тестового релиза

Источник: составлено авторами

Список источников

1. Beck K., Beedle M., van Bennekum, A., et al. Manifesto for Agile Software Development. 2001 [Электронный ресурс]. – Режим доступа: https://agilemanifesto.org/ (In Eng.).
2. Highsmith J. Agile Project Management: Creating Innovative Products. 2009 [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/234809670_Agile_Project_Management_Creating_Innovative_Products (In Eng.).
3. Leffingwell D. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. – Addison-Wesley Professional, 2010. – 560 C. (In Eng.).
4. Ramesh B., Cao L., Baskerville R. Agile requirements engineering practices: An empirical study // IEEE Software. 2012. № 25 (1). C. 60–67. (In Eng.).
5. Umer Iqbal, Khubaib Amjad Alam. Next Release Problem: A Systematic Literature Review // KIET Journal of Computing and Information Sciences. 2020. № 3 (1). C. 16. (In Eng.).

References

1. Beck K., Beedle M., van Bennekum, A., et al. Manifesto for Agile Software Development. 2001. Available at: https://agilemanifesto.org/

2. Highsmith J. Agile Project Management: Creating Innovative Products. 2009. Available at: https://www.researchgate.net/publication/234809670_Agile_Project_Management_Creating_Innovative_Products
3. Leffingwell D. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. *Addison-Wesley Professional*, 2010. 560 p.
4. Ramesh B., Cao L., Baskerville R. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software.* 2012. No. 25 (1). pp. 60–67.
5. Umer Iqbal, Khubaib Amjad Alam. Next Release Problem: A Systematic Literature Review. *KIET Journal of Computing and Information Sciences*. 2020. No. 3 (1). P. 16.

6. Nigar N., Shahzad M., Islam Sh., Oki O., Lukose J. Multi-Objective Dynamic Software Project Scheduling: A Novel Approach to Handle Employee's Addition. IEEE Access. 2023. T. 11. C. 39792–39806. (In Eng.). DOI: 10.1109/ACCESS.2023.3265716.
7. Minku L., Sudholt D., Yao Xi. Evolutionary Algorithms for the Project Scheduling Problem: Runtime Analysis and Improved Design // GECCO'12. Proceedings of the 14th International Conference on Genetic and Evolutionary Computation. 2012. C. 1221–1228. (In Eng.).
DOI: 10.1145/2330163.2330332.
8. Пунтиков А. Н., Шиков А. Н. Алгоритм разбиения проекта на партии при гибких технологиях планирования // Экономика. Право. Инновации. 2023. № 4. С. 81–91.
DOI: 10.17586/2713-1874-2023-4-81-91
9. Mougouei D., Powers D. Dependency-Aware Software Release Planning through Mining User Preferences // Soft Computing. 2020. T. 24. C. 11673–11693. (In Eng.).
DOI: 10.1007/s00500-019-04630-y.
10. Van der Beek T., Souravlias D., van Essen J.T., Pruyn J., Aardal K. Hybrid differential evolution algorithm for the resource constrained project scheduling problem with a flexible project structure and consumption and production of resources // European Journal of Operational Research. 2024. T. 313. № 1. C. 92–111. (In Eng.).
DOI: 10.1016/j.ejor.2023.07.043.
11. Durillo J., at al. A study of the bi-objective next release problem // Empirical Software Engineering. 2011. T. 16. C. 29–60. (In Eng.).
DOI: 10.1007/s10664-010-9147-3.
12. Bagnall A. J., Rayward-Smith V. J., Whittley I. M. The next release problem // Information and Software Technology. 2001. T. 43. № 14. C. 883–890. (In Eng.). DOI: 10.1016/S0950-5849(01)00194-X.
13. Qassem D. Y., Al_saati N. A. A Solution to the Next Release Problem by Swarm Intelligence. Technium // Romanian Journal of Applied Sciences and Technology. 2023. T. 12. C. 58–64. (In Eng.).
DOI: 10.47577/technium.v12i.9439.
14. Panda A., at al. Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points // Procedia Computer Science. 2015. T. 57. C. 772–781. (In Eng.).
DOI: 10.1016/j.procs.2015.07.474.
15. Marner K., Wagner S., Ruhe G. Stakeholder identification for a structured release planning approach in the automotive domain // Requirements Engineering. 2022. T. 27. C. 211–230. (In Eng.).
DOI: 10.1007/s00766-021-00369-x.

6. Nigar N., Shahzad M., Islam Sh., Oki O., Lukose J. Multi-Objective Dynamic Software Project Scheduling: A Novel Approach to Handle Employee's Addition. *IEEE Access*. 2023. Vol. 11. pp.39792–39806. (In Eng.). DOI: 10.1109/ACCESS.2023.3265716.
7. Minku L., Sudholt D., Yao Xi. Evolutionary Algorithms for the Project Scheduling Problem: Runtime Analysis and Improved Design. *GECCO'12. Proceedings of the 14th International Conference on Genetic and Evolutionary Computation.* 2012. pp. 1221–1228.
DOI: 10.1145/2330163.2330332.
8. Puntikov A. N., Shikov A. N. Algorithm for Dividing a Project into Batches with Flexible Planning Technologies. *Ekonomika. Pravo. Innovacii*. 2023. No. 4. pp. 81–91. (In Russ.).
DOI: 10.17586/2713-1874-2023-4-81-91
9. Mougouei D., Powers D. Dependency-Aware Software Release Planning through Mining User Preferences. *Soft Computing.* 2020. Vol. 24. pp. 11673–11693.
DOI: 10.1007/s00500-019-04630-y.
10. Van der Beek T., Souravlias D., van Essen J.T., Pruyn J., Aardal K. Hybrid Differential Evolution Algorithm for the Resource Constrained Project Scheduling Problem with a Flexible Project Structure and Consumption and Production of Resources. *European Journal of Operational Research.* 2024. Vol. 313. No. 1. pp. 92–111. (In Eng.).
DOI: 10.1016/j.ejor.2023.07.043.
11. Durillo J., at al. A Study of the Bi-Objective Next Release Problem. *Empirical Software Engineering*. 2011. Vol. 16. pp. 29–60.
DOI: 10.1007/s10664-010-9147-3.
12. Bagnall A. J., Rayward-Smith V. J., Whittley I. M. The Next Release Problem. *Information and Software Technology.* 2001. Vol. 43. No. 14. pp. 883–890. DOI: 10.1016/S0950-5849(01)00194-X.
13. Qassem D. Y., Al_saati N. A. A Solution to the Next Release Problem by Swarm Intelligence. Technium. *Romanian Journal of Applied Sciences and Technology*. 2023. Vol. 12. pp. 58–64.
DOI: 10.47577/technium.v12i.9439.
14. Panda A., at al. Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points. *Procedia Computer Science.* 2015. Vol. 57. pp. 772–781.
DOI: 10.1016/j.procs.2015.07.474.
15. Marner K., Wagner S., Ruhe G. Stakeholder Identification for a Structured Release Planning Approach in the Automotive Domain. *Requirements Engineering*. 2022. Vol. 27. pp. 211–230.
DOI: 10.1007/s00766-021-00369-x.

Статья поступила в редакцию 05.11.2024; одобрена после рецензирования 24.11.2024; принята к публикации 19.12.2024.
The article was submitted 05.11.2024; approved after reviewing 24.11.2024; accepted for publication 19.12.2024.