УДК 004.89 **DOI: 10.14529/cmse260206**

ФОРМАЛИЗАЦИЯ ЗАДАЧИ ПРОЕКТНОГО ПЛАНИРОВАНИЯ ПРИ РАБОТЕ С НЕСТРУКТУРИРОВАННЫМ ОПИСАНИЕМ ЦЕЛИ

© 2026 Т.Р. Рафиков, А.В. Попцов, В.Д. Олисеенко, М.В. Абрамов

Санкт-Петербургский Федеральный исследовательский центр Российской академии наук
(199178 Санкт-Петербург, 14-я линия В.О., д. 39)
E-mail: trr@dscs.pro, avp@dscs.pro, vdo@dscs.pro, mva@dscs.pro

Поступила в редакцию: 08.05.2026

Большие языковые модели (LLM) демонстрируют высокую эффективность в задачах обработки естественного языка. Однако при применении LLM в планировании качество получаемых решений может снижаться, если задача содержит сложные зависимости между подзадачами, ограничения по ресурсам и срокам. Это ограничивает применимость LLM-планировщиков, где корректность плана критична, — в частности, в управлении проектами. Одним из перспективных направлений преодоления данного недостатка является использование LLM как средства построения формальной модели планирования, решение которой затем находится детерминированными алгоритмами. В данной статье рассматривается задача проектного планирования в условиях, когда исходная информация о проекте задана в неструктурированном текстовом виде. Вводится формализация задачи, в которой большая языковая модель используется для извлечения информации о проекте и построение формальной модели на языке PDDL. Такой подход позволяет связать текстовые описания задач, зависимостей, сроков и ресурсов с формальными средствами автоматического планирования. В результате задача проектного планирования переводится в форму, пригодную для проверки, валидации и последующего решения алгоритмическими методами. Кроме того, данное исследование расширяет математический аппарат классической постановки ресурсно-ограниченного проектного планирования. Помимо общего времени завершения проекта, добавляется учет штрафов за нарушение крайних сроков и качества назначения исполнителей. Тем самым классическая постановка дополняется критериями, которые отражают более сложные и практически значимые требования управления проектами. Данные изменения позволяют перейти от упрощенной однокритериальной модели к более реалистичной постановке. Проведенные вычислительные эксперименты показали, что предложенный подход увеличивает долю корректных планов с 0.1–0.9 до 0.95–1.0 для всех четырех протестированных моделей по сравнению с прямым использованием LLM в качестве планировщика.

Ключевые слова: планирование, большие языковые модели, формализация, LLM-агент.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Рафиков Т.Р., Попцов А.В., Олисеенко В.Д., Абрамов М.В. Формализация задачи проектного планирования при работе с неструктурированным описанием цели // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2026. Т. 15, № 2. С. 104–126. DOI: 10.14529/cmse260206.

Введение

Умение составлять планы и достигать поставленной цели — одна из важных способностей больших языковых моделей (англ. Large Language Models, LLM) и агентов на их основе (LLM-агентов). Данное умение требует глубокого понимания сложных зависимостей, способностей к рассуждению и принятию решений [1]. Несмотря на текущее возможности, LLM не демонстрируют высокого качества, особенно при ограничениях во времени или ресурсах [2].

Для описания окружения и задач планирования зачастую используется язык Planning Domain Definition Language (PDDL) [3]. Он позволяет строить формальные модели, ис-

пользуемые автоматическими планировщиками для решения поставленной задачи планирования [4, 5]. Кроме того, PDDL позволяет описывать дискретными терминами ключевые зависимости, предусловия и ограничения окружения, в которых работает LLM-агент [6]. В рамках данной работы PDDL рассматривается как язык формального описания задачи, который может использоваться планировщиками и автоматическими средствами валидации [7].

Особенно актуальным применение данного подхода представляется в предметных областях со сложной внутренней структурой задач и большим количеством ограничений. К их числу относится управление проектами, для которого характерны зависимости между подзадачами, многочисленные условия выполнения, ограничения по срокам, распределение по исполнителям и ресурсы [8]. В классической постановке такие задачи относятся к классу Resource Constrained Project Scheduling Problem (RCPSP) [9]. Однако реальные проекты значительно сложнее базовых академических моделей, поскольку включают неоднородность источников или неполноту данных, неточности в формулировках, мягкие и жесткие крайние сроки [10, 11].

Таким образом, целью данной работы является формализация задачи проектного планирования в условиях, когда исходная информация о проекте задана в неструктурированном текстовом виде. Достижение этой цели обеспечивается за счет извлечения информации о проекте из текстовых данных с помощью большой языковой модели и ее последующего представления на языке PDDL. Научная новизна работы состоит в объединении в рамках одной постановки неструктурированных проектных данных, LLM как средства формализации и PDDL как языка описания задачи планирования, а также в применении подхода LLM-as-formalizer к задаче проектного планирования. Теоретическая значимость работы заключается в расширении классической модели RCPSP за счет учета просрочек задач и качества назначения исполнителей. Практическая значимость работы состоит в создании формальной основы для применения существующих планировщиков и валидаторов к задачам управления проектами, исходно заданным в неструктурированном текстовом виде. Статья организована следующим образом. В разделе 1 проведен обзор существующих подходов использования LLM в задачах планирования. Раздел 2 посвящен языку PDDL, на котором строится дальнейшая формализация. В разделе 3 предлагается формализация задачи проектного планирования и ее перевод в PDDL. Раздел 4 описывает программную реализацию предложенного подхода. В разделе 5 предложенный подход сравнивается с LLM-as-planner на вычислительных экспериментах. В разделе 6 представлено обсуждение результатов работы, а также ограничения формализации. Заключение подводит итоги и обозначает направления дальнейших исследований.

1. Обзор литературы

В работах по LLM-агентам [1, 12, 13] планирование рассматривается как отдельный модуль агентной архитектуры. Декомпозиция задач, рефлексия, выбор действий, использование внешних модулей и памяти в обзоре [14] выделяют как неотъемлемые возможности планирования LLM. Другие работы посвящены использованию LLM в автоматическом планировании и составлении расписания [15]. Обзор [16] показывает, что LLM используются не только для генерации планов, но и для перевода в другой формат, интеграции инструментов, интерактивного планирования и выбора эвристик. При этом именно построение формальных моделей планирования и интеграция с внешними планировщиками рассмат-

риваются как наиболее перспективные направления [6]. В обзоре [6] LLM детально рассматривается как формализатор, то есть как средство перевода с естественного языка в формальное представление, позволяющее решать задачу планирования детерминированными алгоритмами.

При этом исследования подчеркивают и ограничения такого подхода. Работа по LLM-as-formalizer [17] показывает, что качество построения формальной модели зависит от сложности домена и структуры входного текста. Однако данный подход остается более надежным по сравнению с LLM-планировщиками [16]. Именно это лежит в основе дальнейшей формализации, где LLM отвечает за извлечение и структурирование знаний, а план ищет внешний планировщик. Сравнительная характеристика различных подходов по применению LLM к задаче планирования приведена в табл. 1.

Таблица 1. Сравнение подходов применения LLM в планировании

Источник	**Ключевые идеи, выделяемые категории**	**LLM как планировщик**	**LLM как формализатор**	**Формат**
A brain-inspired agentic architecture to improve planning with LLMs [18]	Модульная архитектура планирования с использованием декомпозиции задач и оркестратором	Да	Нет	Естественный язык
The rise and potential of large language model based agents: a survey [14]	Декомпозиция, поиск плана, внешний планировщик, рефлексия и память.	Да	Нет	Естественный и формальный языки
On the Prospects of Incorporating LLMs in APS [15]	Перевод в формальные языки, генерация плана, построение моделей, мультиагентное планирование, интерактивное планирование, оптимизация эвристик, вызов инструментов и когнитивное планирование	Да	Да	Естественный и формальный языки
LLM-as-Formalizer Survey [6]	Обзорная работа о процессе формализации окружения планирования с помощью LLM, рассматривая их генерацию, редактирование и оценку	Нет	Да	Естественный и формальный языки
Classical Planning with LLM-Generated Heuristics [19]	Использование LLM для генерации доменно-зависимых эвристик	Да	Нет	Формальный язык

В строках табл. 1 представлены рассматриваемые работы. В столбцах отражены источник, ключевые идеи, роль LLM и формат представления задачи. Столбцы «LLM как планировщик» и «LLM как формализатор» показывают, используется ли модель для прямого построения плана или для перевода задачи в формальное описание. Столбец «Формат» указывает, в каком виде задается задача: в естественном, формальном или в обоих языках. Анализ данных работ показывает, что существующие подходы к применению LLM в планировании можно условно разделить на два направления. В первом случае LLM используется непосредственно как планировщик. Во втором случае LLM выступает как формализатор для преобразования текстового описания задачи в структуру, поддерживаемую внешними планировщиками. При этом часть работ сочетает оба подхода, однако для задач со строгой формальностью подход LLM-формализатор является более надежным.

Одним из основных языков для формального описания задачи планирования выступает PDDL и его расширения. Следующая версия языка [20] вводит числовые выражения, временные действия и метрики плана, что позволяет использовать язык для временного планирования. Дальнейшее развитие PDDL получил в работе [21], где были добавлены предпочтения и мягкие ограничения, позволяя описывать желаемые свойства плана, что особенно важно в прикладных задачах.

Проектное планирование и задача его оптимизации получили свое развитие в работах [9, 10]. Исследование [11] показывает, что в реальных сценариях существенную роль играют гибкость ресурсов, различия в навыках исполнителей и их распределение. Поэтому минимизации времени завершения проекта как критерия качества в прикладных задачах недостаточно. В работе [22] показано, что соответствие работника требованиям конкретной задачи положительно связано с эффективностью выполнения этой задачи. Кроме того, само распределение исполнителей по задачам существенно влияет на общую эффективность, что подтверждается исследованием [23]. Это дает основание рассматривать качество назначения исполнителей как отдельный фактор для проектного планирования.

2. Формальный язык планирования PDDL

Для того чтобы формально описать перечисленные зависимости, ресурсы и качество назначений в едином представлении, обратимся к Planning Domain Definition Language (PDDL), на котором строится дальнейшая формализация [20]. Данный язык является одним из основных языков описания задач автоматического планирования. Его назначение состоит в том, чтобы построить формальную модель предметной области и конкретного экземпляра задачи. Данная модель далее используется автоматическими планировщиками [4, 5]. Исторически PDDL был введен как единый язык описания задач для International Planning Competition [3]. В дальнейшем он стал фактическим стандартом в исследованиях по классическому планированию.

Описание задачи в PDDL делится на две части: domain (домен) и problem (проблема). В файле домена задаются типы объектов, предикаты, числовые функции и действия. В файле проблемы задаются конкретные объекты, начальное состояние, целевые условия и, при необходимости, метрика качества плана. Такое разделение позволяет многократно использовать одно и то же формальное описание окружения для разных задач.

Синтаксис PDDL основан на префиксной записи в стиле языка программирования Lisp [24]. Основными конструкциями языка являются `:types`, `:predicates`, `:functions`, `:action`, `:preconditions`, `:effects` и `:goal`. Раздел `:types` задает типы объектов предмет-

ной области. Предикаты `:predicates` используются для описания логических свойств объектов и состояний. Функции `:functions` используются для задания числовых параметров, например, длительности или стоимости. Действие `:action` описывается через параметры, предусловия и эффекты. Предусловия `:preconditions` задают, при каких условиях действие может быть выполнено. Эффекты `:effects` задают, как изменяется состояние после его выполнения. Раздел `:goal` задает целевое состояние, которое должно быть достигнуто в результате построения плана. Ниже эти конструкции приведены на минимальном примере домена и связанной с ним проблемы.

В листинге 1 домен объявляется конструкцией `(define (domain mini) ...)`. В блоке `:requirements` указано, что используются типизированные объекты и числовые функции. Блок `:types` вводит типы `task` и `employee`, блок `:predicates` задает состояния задачи `ready`, `done` и отношение квалификации `qualified`. В разделе `:functions` вводится числовая функция `cost`, а действие `do-task` связывает задачу и исполнителя через параметры, проверяет готовность задачи и квалификацию исполнителя в `:precondition`, после чего переводит задачу в состояние `done` в `:effect`.

Листинг 1. Пример домена в PDDL

```
(define (domain mini)
  (:requirements :typing :fluents)      ; типизация и функции
  (:types task employee)                ; типы объектов
  (:predicates                          ; задаваемые отношения (предикаты)
    (ready ?t - task)                   ; задача готова
    (done ?t - task)                    ; задача выполнена
    (qualified ?e - employee ?t - task) ; квалификация
  )
  (:functions
    (cost ?t - task)                    ; функция стоимости задачи
  )
  (:action do-task                      ; задаваемое действие
    :parameters (?t - task ?e - employee)               ; параметры
    :precondition (and (ready ?t) (qualified ?e ?t))    ; предусловия
    :effect (done ?t)                                   ; итог
  )
)
```

В листинге 2 конструкция `(define (problem mini-problem) ...)` объявляет конкретный экземпляр задачи, а строка `(:domain mini)` связывает его с доменом из листинга 1. В блоке `:objects` создаются объект задачи `t1` и объект исполнителя `e1`. В разделе `:init` задается начальное состояние: задача готова к выполнению, исполнитель обладает нужной квалификацией, а значение функции `cost` для задачи равно единице. Блок `:goal` формулирует цель планирования как достижение состояния `(done t1)`.

В классическом варианте действия в PDDL рассматриваются без привязки ко времени. Однако для прикладных задач этого часто недостаточно. По этой причине в языке PDDL 2.1 были введены временные действия, числовые переменные и метрики плана [20]. Временное действие задается конструкцией `:durative-action`. Для него можно определить длительность, условия начала и окончания, а также условия, которые должны сохраняться

Листинг 2. Пример проблемы в PDDL

```
(define (problem mini-problem)
  (:domain mini)                          ; связь с доменом
  (:objects                               ; задаваемые объекты
    t1 - task                             ; задача
    e1 - employee                         ; сотрудник
  )
  (:init                                  ; изначальное состояние
    (ready t1)                            ; готовность задачи t1
    (qualified e1 t1)                     ; сотрудник e1 может исполнить t1
    (= (cost t1) 1)                       ; стоимость задачи 1
  )
  (:goal                                  ; задаваемая цель
    (done t1)                             ; выполнить задачу t1
  )
)
```

на протяжении всего выполнения действия. Дальнейшее развитие язык получил в работе PDDL 3 [21], где появилась возможность использовать мягкие ограничения и предпочтения.

3. Формализация задачи проектного планирования с представлением в PDDL

Как упоминалось выше, на практике задача оптимизации проектного планирования обычно выходит за рамки базовой формализации RCPSP. Реальные проекты часто включают дополнительные крайние сроки, многопроектную структуру, приоритеты задач, неоднородные ресурсы и иные ограничения, которые не всегда удобно выразить в классической постановке [25, 26]. По этой причине дальнейшая формализация в работе опирается на язык PDDL и его расширения. Такой выбор позволяет явно задавать действия, ограничения, ресурсы и целевые условия планирования. Однако перед описанием перевода в PDDL требуется формализовать поступающие на вход неструктурированные данные и задать критерий качества плана.

3.1. Формализация

В данной работе предполагается, что исходное описание проекта и задач задано в неструктурированном текстовом виде с описанием всех зависимостей, ограничений, крайних сроков и т.д. Обозначим множество неструктурированных данных о проекте как

$$D = \{v^{(1)}, v^{(2)}, \ldots, v^{(K)}\}, \tag{1}$$

где элементы $v^{(k)}$ представляют собой элементы текстовой информации: описания и очередь задач, проектную документацию, требования и так далее. Исходя из предоставленной информации в D впоследствии строится формальная модель для планирования.

Из множества D необходимо выделить следующие атрибуты предметной области:

- $T = \{t_1, t_2, \ldots, t_n\}$ — множество задач проекта;
- $E \subseteq T \times T$ — зависимости между задачами;
- $U = \{g_1, g_2, \ldots, g_p\}$ — множество исполнителей;

– $R = \{r_1, r_2, \ldots, r_m\}$ — множество типов ресурсов.

При этом, $(t_1, t_2) \in E \Leftrightarrow$ «t_2 зависит от t_1» и не существует зацикливающихся зависимостей. Кроме того, каждой задаче $t_i \in T$ задаются параметры:

– $d_i \in \mathbb{N}$ — длительность выполнения;
– $R_i \subseteq R$ — требуемые ресурсы;
– $U_i \subseteq U$ — исполнители для этой задачи;
– $\delta_i \in \mathbb{N}$ — крайний срок задачи;
– $w_i \in \mathbb{R}$ — штраф за просрочку задачи.

Для ресурсов введем функцию доступности $c_r(t) : \mathbb{N} \to \mathbb{R}$ в момент времени t, задающую объем ресурса r, доступный в момент t. Стоит отметить, что функция $c_r(t)$ задает ограничение допустимости плана и не входит непосредственно в оптимизируемую функцию, которая будет введена ниже. Кроме того, для каждой задачи t_i введем момент ее завершения $f_i \in \mathbb{N}$. Для описания качества распределения исполнителей по задачам введем параметр

$$p_{iu} \in R, \tag{2}$$

который задает стоимость или штраф назначения исполнителя u на задачу t_i. Также введем переменную

$$x_{iu} \in \{0, 1\}, \tag{3}$$

которая показывает, был ли назначен исполнитель u на задачу t_i. Кроме того, ни один исполнитель не может быть занят на двух задачах одновременно:

$$\sum_{i \in T:\ f_i - d_i \le t < f_i} x_{iu} \le 1, \quad \forall u \in U \text{ — исполнитель},\ \forall t \in \mathbb{N} \text{ — момент времени}. \tag{4}$$

Далее рассмотрим большую языковую модель как отображение

$$\Phi_\theta : D \to M = \langle Dom, Prob \rangle, \tag{5}$$

где Φ_θ — LLM с параметрами θ, а M — представление задачи в PDDL, состоящее из описания предметной области Dom и описания задачи $Prob$. LLM в предлагаемой постановке не выступает непосредственным планировщиком, а выполняет функцию извлечения и структурирования всей необходимой информации.

После построения модели M задача сводится к нахождению плана

$$\pi^* = \arg \min_{\pi \in \Pi(M)} J(\pi), \tag{6}$$

где $\Pi(M)$ — множество всех допустимых планов для модели M формата PDDL, а $J(\pi)$ — целевая функция качества плана. В изначальном варианте RCPSP в качестве целевой функции рассматривается минимизация времени [9]:

$$\min J(\pi) = C_{\max}, \tag{7}$$

где

$$C_{\max} = \max_{i \in T} f_i \tag{8}$$

— момент завершения последней задачи.

Для более реалистичной постановки в данной работе предлагается взять за целевую функцию

$$J(\pi) = \alpha C_{\max} + \beta \sum_{i \in T} w_i L_i + \gamma \sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}, \tag{9}$$

где $L_i = \max(0, f_i - \delta_i)$ — просрочка задачи t_i, а α, β, $\gamma \in R_{\geq 0}$ обозначают весовые коэффициенты критериев качества плана. Данный выбор обусловлен тем, что в RCPSP базовым критерием является минимизация $C_{\max}$, однако современные расширения проектного планирования рассматривают также задержки относительно крайних сроков и прочие различные характеристики описания как самостоятельные цели [10, 25, 26].

Первый член $\alpha C_{\max}$ был взят из классической постановки задачи оптимизации проектного планирования как стандартный критерий. Кроме времени завершения проекта также нужно учитывать и поставленные промежуточные крайние сроки по задачам. Это подтверждается работами [10, 27, 28], где рассматриваются цели со сроками и штраф за несоблюдение крайнего срока. Поэтому вторым членом $\beta \sum_{i \in T} w_i L_i$ учитывается нарушение сроков отдельных задач. Кроме того, исследования [22, 23, 29—31] показывают, что не менее важным фактором является назначение квалифицированных исполнителей на подходящую им задачу. Чтобы учесть этот критерий, вводится третье слагаемое $\gamma \sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}$, отвечающее за качество распределения исполнителей по задачам.

3.2. Перевод в PDDL

Кроме формализации необходимо также определить, каким образом выделенные данные отображаются в PDDL. В предлагаемой постановке модель M представляется в виде пары PDDL-представлений (файлов с расширением «.pddl»). Файл *Dom* задает описание предметной области: типы объектов, предикаты, числовые функции и действия. Файл *Prob*, в свою очередь, определяет поставленную задачу: множество задач проекта, исполнителей, ресурсов, начальное состояние и целевые условия.

В рамках данной работы множество задач в T будет представлено в объектах типа `task`, множество исполнителей U — в объектах типа `employee`, а множество ресурсов R — в объектах типа `resource`. Зависимости между задачами могут быть отображены явно через предикаты вида `(depends ?t1 ?t2)`. Состояния выполнения задач описываются предикатами `(ready ?t)` и `(finished ?t)`. Числовые функции в разделе `:functions` PDDL 3 могут быть использованы для длительности, доступности ресурсов и совокупной стоимости плана.

Одним из ключевых элементом в рассматриваемой задаче является продолжительность выполнения действия. Задачи с данным свойством в PDDL 3 задаются с помощью durative actions. В общем виде действие должно учитывать условия начала выполнения задачи, поддерживаемые условия на всем интервале ее выполнения и вклад выполненной задачи. Данная структура поддерживается во временных действиях PDDL 3. Пример такого временного действия приведен в листинге 3.

В листинге 3 конструкция `:durative-action` задает действие, выполнение которого занимает ненулевое время. Блок `:parameters` связывает действие с конкретной задачей `?t` и исполнителем `?u`, а выражение `:duration` определяет длительность через значение числовой функции `(task-duration ?t)`. Условия в разделе `:condition` помечены как `at start`: задача должна быть готова, а исполнитель должен обладать требуемой квалификацией в момент начала действия. В разделе `:effect` эффект `assigned` фиксирует назначение испол-

Листинг 3. Пример временного действия выполнения задачи

```
(:durative-action execute-task                    ; временное действие
  :parameters (?t - task ?u - employee)           ; задача, исполнитель
  :duration (= ?duration (task-duration ?t))      ; длительность задачи
  :condition (and
    (at start (ready ?t))                         ; задача готова
    (at start (qualified ?u ?t)))                 ; нужна квалификация
  :effect (and
    (at start (assigned ?t ?u))                   ; назначение в начале
    (at end (finished ?t))))                      ; завершение в конце
```

нителя в начале выполнения, а эффект `finished` применяется в конце действия. Благодаря этому завершение задачи может использоваться как условие для запуска зависимых работ.

Соответствующий файл *Prob* содержит имеющиеся объекты проекта, изначальные условия и поставленные цели. Например, исполнители, параметры задач и ресурсные ограничения задаются в изначальных условиях. Цели могут быть заданы как завершение всех задач проекта. Критерий качества можно задать как минимизацию целевой функции (9) с помощью `(:metric minimize <numeric_expression>)`.

Если в проекте присутствуют мягкие ограничения, например, нежелательность использования некоторых исполнителей для определенных задач, то их можно задать как `preference` в PDDL 3 [21]. В этом случае допустимость плана определяется ограничениями, а предпочтения влияют на его качество и ранжирование относительно других планов. Данный подход позволяет разделять задачи с безусловными условиями и задачи, допускающие компромиссы в условиях.

Таким образом, PDDL позволяет перейти от неструктурированных данных о проекте к формальной модели, решаемой детерминированными планировщиками. При этом LLM играет ключевую роль в формировании домена Dom и проблемы $Prob$ на основе данных D. Представленная схема соответствует подходу LLM-as-formalizer, демонстрирующему более надежную работу в сравнении с LLM-as-planner.

С помощью введенной формализации задача проектного планирования задается формулой (6) как поиск оптимального плана $\pi^* \in \Pi(M)$. Полученная формальная модель делает возможным применение существующих автоматических планировщиков: LAMA [5], A*-планировщиком [32], жадный поиском [4, 33], а также алгоритмы для решения комбинаторных задач оптимизаций, например, CP-SAT [34]. Для построения формальной модели $M = \langle Dom, Prob \rangle$ из множества неструктурированных проектных данных D при помощи LLM выделяются множества задач T, зависимостей E, исполнителей U и ресурсов R, а также параметры задач d_i, δ_i, w_i и функция доступности ресурсов $c_r(t)$. Тем самым осуществляется переход от неструктурированного описания проекта к математической модели, на которой определяется минимизируемая функция качества $J(\pi)$. Предложенная целевая функция (9) позволяет одновременно учитывать сроки проекта, просроченные задачи и качество распределения исполнителей. За счет этого классическая модель проектного планирования расширяется до более реалистичной постановки.

3.3. Примеры для формализации

Для наглядной иллюстрации применимости предложенной формализации ниже рассмотрены несколько типовых примеров.

Пример 1. Рассмотрим текстовое описание проекта вида: «Необходимо разработать внутренний веб-сервис. Сначала аналитик подготавливает требования, затем архитектор проектирует систему, после этого разработчик реализует модуль, а тестировщик проводит тестирование. Архитектура не может начинаться до завершения требований, разработка — до завершения архитектуры, тестирование — до завершения разработки. Для каждой задачи требуется один исполнитель соответствующей роли». В этом случае D содержит четыре текстовых артефакта или один общий артефакт с четырьмя задачами; множество задач задается как $T = \{t_1, t_2, t_3, t_4\}$, где t_1 — сбор требований, t_2 — проектирование, t_3 — разработка, t_4 — тестирование; зависимости задаются как $E = \{(t_1, t_2), (t_2, t_3), (t_3, t_4)\}$; исполнители $U = \{u_a, u_b, u_c, u_d\}$, ресурсы R могут быть сведены к ролям или рабочему времени; длительности, например, $d_1 = 2$, $d_2 = 3$, $d_3 = 5$, $d_4 = 2$. В PDDL это представляется объектами `req design code test` — `task`, предикатами `(depends req design)`, `(depends design code)`, `(depends code test)`, `(qualified analyst req)`, `(qualified architect design)`, а выполнение задачи задается через `:durative-action execute-task` с `:duration (= ?duration (task-duration ?t))`.

Пример 2. Рассмотрим текстовое описание проекта вида: «Команда готовит релиз мобильного приложения. Прототип интерфейса должен быть готов не позднее 5-го дня, серверная часть — не позднее 10-го дня, итоговая интеграция должна завершиться до 14-го дня. Нарушение промежуточных сроков допустимо, но нежелательно». Здесь кроме стандартных множеств T, E, U, R вводятся крайние сроки δ_i, например $\delta_{ui} = 5$, $\delta_{backend} = 10$, $\delta_{integration} = 14$, а функция качества уже задается не только через $C_{\max}$, но и через просрочки $L_i = \max(0, f_i - \delta_i)$. Если два плана имеют одинаковый $C_{\max}$, но один нарушает крайний срок прототипа, а другой нет, то они различаются именно вторым слагаемым $\sum_{i \in T} w_i L_i$. В PDDL такая постановка может быть задана через стандартные временные действия и метрику `(:metric minimize ...)`, а при необходимости мягкие крайние сроки могут быть выражены через preferences в PDDL 3, где нарушение срока не запрещает план, но ухудшает его качество.

Пример 3. Рассмотрим текстовое описание проекта вида: «В проекте разработки продукта задача «frontend» может быть выполнена либо senior-разработчиком, либо junior-разработчиком, но senior выполняет ее быстрее и с меньшим риском доработок; задача «архитектура» может быть назначена только senior-разработчику; задача «тестирование» может быть назначена двум разным тестировщикам с разной стоимостью времени». В этом случае недостаточно только множеств T и E; дополнительно требуется определить назначения x_{iu}, допустимость исполнителей и стоимость или предпочтительность назначения p_{iu}. Например, для задачи frontend можно задать $x_{front,u1}$, $x_{front,u2} \in \{0, 1\}$, где $u1$ — senior, $u2$ — junior, а значения $p_{front,u1} < p_{front,u2}$ или наоборот определяются выбранной интерпретацией качества назначения. В PDDL это выражается объектами `senior junior` — `employee`, предикатами `(qualified senior architecture)`, `(qualified senior frontend)`, `(qualified junior frontend)`, а также числовыми параметрами, связанными с длительностью или стоимостью выполнения.

4. Реализация предложенного подхода

Основные этапы формирования модели M и плана π^* приведены в Алгоритме 1. На шаге 1 отображение Φ_θ (большая языковая модель) по текстовому описанию задачи D строит формальную модель $M = \langle Dom, Prob \rangle$. Это пара PDDL-файлов: Dom задает словарь типов, предикатов и функций, а $Prob$ кодирует извлеченные из D работы T, зависимости E, исполнителей U, ресурсы R, параметры $\{d_i\}$, $\{\delta_i\}$, $\{w_i\}$, $\{p_{iu}\}$ и веса α, β, γ. Шаг 2 проверяет внутреннюю согласованность M функцией CheckConsistency; если модель некорректна, шаги 3–4 завершают обработку без построения плана. На шаге 6 по модели M с помощью BuildModel строится множество допустимых планов $\Pi(M)$. Шаг 7 задает целевую функцию $J(\pi)$, введенную в разделе 3. На шаге 8 среди допустимых планов $\Pi(M)$ находится план π^*, минимизирующий $J(\pi)$. Шаг 9 возвращает модель M и найденный план π^*.

Алгоритм 1. Формализация задачи проектного планирования при помощи LLM

Вход: D

Выход: $M = \langle Dom, Prob \rangle$, π^*

1: $M \leftarrow \Phi_\theta(D)$, $M = \langle Dom, Prob \rangle$ ▷ Извлечение данных с помощью LLM

2: $valid \leftarrow$ CheckConsistency(M) ▷ Проверка согласованности

3: **if not** $valid$ **then**

4: **return** $\varnothing$ ▷ Ошибка формализации

5: **end if**

6: $\Pi(M) \leftarrow$ BuildModel$(T, E, U, R, d, \delta, w, p)$ ▷ Построение расписания

7: $J(\pi) \leftarrow \alpha C_{\max}(\pi) + \beta \sum\limits_{i \in T} w_i L_i(\pi) + \gamma \sum\limits_{i \in T} \sum\limits_{u \in U} p_{iu} x_{iu}$ ▷ Целевая функция

8: $\pi^* \leftarrow \arg \min\limits_{\pi \in \Pi(M)} J(\pi)$ ▷ Оптимизация плана

9: **return** M, π^* ▷ Результат

Проверка CheckConsistency выполняется на двух уровнях. Первый уровень — внутренняя согласованность M: наличие всех необходимых сущностей T, U, R, корректность ссылок между ними, отсутствие циклов в зависимостях E. Второй уровень — соответствие извлеченных T, E, U, R и параметров d, δ, w, p эталонному ответу задачи. Результаты этого сравнения используются для оценки точности извлечения информации из текстового описания D, приведенной в разделе 5.

Функция BuildModel по извлеченным параметрам T, E, U, R, d, δ, w, p строит множество допустимых планов $\Pi(M)$. Каждой работе из T сопоставляются переменные начала и завершения, зависимости E задаются условием $\mathrm{end}(before) \leq \mathrm{start}(after)$, каждой работе назначается требуемое число исполнителей из допустимых по U с учетом того, что один исполнитель не может выполнять две работы одновременно. Потребление ресурсов из R на любом интервале времени не превышает их вместимости, а превышение крайнего срока δ_i учитывается как неотрицательная переменная просрочки, входящая в целевую функцию с весом w_i. Таким образом, $\Pi(M)$ включает все планы, удовлетворяющие перечисленным ограничениям, среди которых на следующем шаге отыскивается план π^*, минимизирующий $J(\pi)$.

Программная реализация выполнена на языке Python. Большинство классических PDDL-планировщиков не поддерживают модификации в оптимизируемую функцию [4, 5]. Поэтому для шагов 6–8 (построение множества допустимых планов $\Pi(M)$, задание целе-

вой функции и поиск оптимального плана π^*) используется алгоритм для решения задач комбинаторной оптимизации CP-SAT из библиотеки Google OR-Tools [34].

5. Вычислительные эксперименты

Дизайн эксперимента направлен на сравнение предложенного подхода (раздел 4) с базовым способом, при котором LLM планирует самостоятельно (LLM-as-planner). На вход в обоих случаях подается одно и то же текстовое описание D с одинаковой постановкой задачи: минимизировать функцию $J(\pi)$ с заданными параметрами и условиями. Код эксперимента и данные доступны в репозитории [35].

В случае LLM-as-planner текстовое описание D передается большой языковой модели с целью построения плана π в структурированном виде без формализации. Данный план проверяется на соответствие следующим условиям: каждая работа из T выполняется ровно один раз; момент начала каждой работы неотрицателен; длительность и число назначенных исполнителей соответствуют условию задачи; ни один исполнитель не занят одновременно на двух несовместимых по времени работах; зависимости E между работами соблюдены; потребление ресурсов R не превышает их вместимости на любом интервале времени. Для предложенного подхода, подразумевающего формализацию задачи, допустимость плана π^* обеспечивается уже самим построением множества $\Pi(M)$, описанным в разделе 4, поэтому отдельная проверка для него не требуется.

В качестве данных для эксперимента подготовлено 20 примеров, построенных на основе задач из PSPLIB [36]: исходные экземпляры незначительно упрощены и дополнены параметрами из раздела 3 — исполнителями U, крайними сроками δ_i и штрафами w_i, p_{iu}. Текстовые описания D формировались путем подстановки параметров каждого экземпляра PSPLIB — длительностей, зависимостей, требований к ресурсам, а также добавленных исполнителей U, крайних сроков δ_i и штрафов w_i, p_{iu} — в единый текстовый шаблон. Это обеспечивает единообразие и полноту описаний для всех 20 примеров. Оба способа тестировались на моделях GPT-OSS-120B[1], Qwen3-32B[2], Qwen3-235B-A22B-Thinking-2507[2] (ниже сокращенно Qwen3-235B) и DeepSeek V4 Flash[3]

Для сопоставления различных подходов использовались следующие метрики: 1) доля валидных планов; 2) значение целевой функции $J(\pi)$ и ее компоненты — 3) время завершения проекта, 4) суммарная взвешенная просрочка и 5) штраф качества назначения исполнителей; а также метрики качества формализации — F_1 по основным сущностям задачи и точность извлечения числовых параметров. Опишем, как вычисляется каждая из них.

Компоненты целевой функции $J(\pi)$, введенной в разделе 3 формулой (9), вычисляются следующим образом. Время завершения проекта:

$$\max_{i \in T} f_i.$$

Суммарная взвешенная просрочка:

$$\sum_{i \in T_\delta} w_i L_i = \sum_{i \in T_\delta} w_i \max(0, f_i - \delta_i),$$

[1]Agarwal S., Ahmad L., Ai J. и др. gpt-oss-120b & gpt-oss-20b Model Card. 2025. URL: `https://arxiv.org/abs/2508.10925` (дата обращения: 14.07.2026)

[2]Qwen Team. Qwen3 Technical Report. 2025. URL: `https://arxiv.org/abs/2505.09388`. (дата обращения: 14.07.2026)

[3]DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. 2026. URL: `https://arxiv.org/abs/2606.19348` (дата обращения: 14.07.2026)

где $T_\delta \subseteq T$ — подмножество работ с заданным крайним сроком; для работ без крайнего срока вклад в просрочку равен нулю. Штраф качества назначения исполнителей:

$$\sum_{i \in T} \sum_{u \in U} p_{iu} x_{iu}.$$

В проведенных экспериментах веса α, β, γ были независимыми параметрами для каждого из сэмплов. Данные параметры оставались одинаковыми для LLM-as-planner и LLM-as-formalizer в пределах одного примера. Меньшее значение каждой из приведенных метрик является более лучшим показателем.

Таблица 2. Усредненные метрики целевой функции

Модель	**Подход**	**Валидность** (доля)	$J(\pi)\downarrow$ (усл. ед.)	**Время**$\downarrow$ (дни)	**Просрочки**$\downarrow$ (усл. ед.)	**Делегирование**$\downarrow$ (усл. ед.)
GPT-OSS-120B	Планировщик	0.9	91.35	40.00	22.33	26.72
	Формализатор	0.95	53.14	36.89	4.68	27.53
Qwen3-32B	Планировщик	0.1	50.50	44.00	0.00	26.00
	Формализатор	1.0	49.14	35.65	3.40	26.75
Qwen3-235B	Планировщик	0.3	75.50	41.67	13.00	31.33
	Формализатор	1.0	52.00	36.30	4.45	27.20
DeepSeek-Flash	Планировщик	0.7	72.79	37.64	14.29	26.29
	Формализатор	1.0	52.00	36.30	4.45	27.20

Результаты, полученные по описанным метрикам, приведены в табл. 2. Значения $J(\pi)$, просрочки и делегирования выражены в условных единицах, поскольку целевая функция объединяет разнородные по своей природе слагаемые с весовыми коэффициентами α, β, γ. Значения метрик рассчитывались только по корректным планам, поскольку для планов, нарушающих ограничения задачи, величины метрик не имеют содержательной интерпретации.

6. Обсуждение результатов

Доля валидных планов у предложенного подхода составляет от 0.95 до 1.0 для всех четырех моделей, тогда как у способа LLM-as-planner она заметно ниже и колеблется от 0.1 (Qwen3-32B) до 0.9 (GPT-OSS-120B). Значение целевой функции $J(\pi)$ у предложенного подхода ниже для всех моделей: 53.14 против 91.35 у GPT-OSS-120B, 49.14 против 50.50 у Qwen3-32B, 52.00 против 75.50 у Qwen3-235B и 52.00 против 72.79 у DeepSeek V4 Flash. Время выполнения проекта у предложенного подхода в среднем быстрее (35.65–36.89 против 37.64–44.00), штраф за нарушение крайних сроков также в среднем ниже (3.40–4.68 против 13.00–22.33). Исключение — Qwen3-32B в способе планировщика, где просрочка равна 0.00; это значение получено всего по 10% построенных планов, оставшихся допустимыми, и поэтому не может считаться показательным. По качеству назначения исполнителей значения обоих подходов близки и находятся в диапазоне 26.00–31.33.

Помимо качества самого плана, отдельно оценивалось качество формализации по метрикам, описанным выше. Показатели F_1 по работам, зависимостям, исполнителям и ресурсам, а также точность извлечения длительностей и крайних сроков близки к 1.00 для всех четырех моделей. Отклонения от 1.00 наблюдаются только у F_1 по зависимостям для Qwen3-32B (0.95) и у F_1 по длительностям задач для отдельных сотрудников — для Qwen3-32B (0.97) и DeepSeek-Flash (0.94). Кроме того, можно заметить, что метрики у моделей

Qwen3-235B и DeepSeek V4 Flash при подходе «Формализатор» получились идентичными. Это объясняется тем, что обе модели дают на большинстве сэмплов практически безошибочное извлечение. В результате чего построенные по их PDDL-описаниям модели CP-SAT оказываются эквивалентны и приводят к одному и тому же оптимальному расписанию. Полученные результаты подтверждают возможность построения формальной модели задачи планирования при помощи LLM.

Предложенная формализация задачи планирования имеет несколько сильных сторон. Прежде всего, разделяются этап извлечения данных и этап планирования. Это важно для задач, где исходные данные представлены в естественном языке без заданной структуры. В таком случае LLM используется как средство структурирования описания проекта, а не как планировщик. Именно такой подход рассматривается в современных исследованиях [6, 17] как наиболее перспективный и эффективный для LLM-агентов.

Следующее преимущество состоит в том, что предложенная формализация учитывает более широкий набор ограничений, чем это обычно предполагается в базовых постановках проектного планирования. В модели одновременно присутствуют зависимости между задачами, ресурсы, крайние сроки и назначения исполнителей. При этом сохраняется баланс между классическими постановками RCPSP и прикладными задачами, возникающими в реальных проектах. Тем самым предложенный подход не заменяет классическое проектное планирование, а расширяет его за счет новых ограничений, целевой функции качества и использования LLM как формализатора.

Важной особенностью предложенного подхода является использование PDDL как формального средства представления задачи. Это дает возможность задать зависимости, временные параметры, числовые характеристики и критерии качества плана в единой модели, а затем применять к ней автоматические планировщики и средства проверки. Также благодаря данному языку появляется возможность автоматически проверять результат. Если вся информация переведена в PDDL, то корректность плана, домена и проблемы можно оценить с помощью автоматического валидатора [7]. Это упрощает проверку и дополнительную валидацию, а также минимизирует риск получения логически некорректного плана. Именно такой недостаток отмечается в работах [2, 16], где LLM используется как планировщик.

Вместе с тем, предложенная постановка имеет и ограничения. Одно из них связано с качеством исходных данных D. Если текстовое описание проекта неполно или противоречиво, то на этапе извлечения задач и их параметров может возникнуть ошибка. В этом случае планировщик будет работать корректно, однако на модели, которая уже не вполне соответствует реальному проекту. Работа [17] прямо показывает, что качество формализации снижается с ростом сложности и расплывчатости описания. Отметим также, что в проведенных экспериментах текстовые описания D генерировались по единому шаблону на основе PSPLIB, поэтому оценка устойчивости подхода к стилистически более разнородным формулировкам, характерным для реальных проектных текстов, выходит за рамки данной работы.

Следующее ограничение касается целевой функции. Ее выбор мотивирован с прикладной точки зрения, но он остается моделирующим допущением. Минимизация общего времени выполнения соответствует классической постановке RCPSP, а учет просрочек и назначений исполнителей отражает реальные требования управления проектами. Однако веса α, β и γ требуют калибровки в зависимости от приоритетов. Без такой калибровки трудно

гарантировать, что компромисс между общим временем выполнения, крайними сроками и распределением исполнителей соответствует приоритетам команды. Отметим также, что в проведенных экспериментах веса задавались случайным образом независимо для каждого примера. Попарное сравнение между подходами по отдельным метрикам остается корректным, однако усредненное по выборке значение $J(\pi)$ отражает эффект только по случайно распределенным весам.

Отдельного обсуждения заслуживает третье слагаемое в целевой функции. Его достоинство состоит в том, что оно позволяет учитывать различия между исполнителями. Соответствие навыков исполнителя требованиям задачи, а также качественное распределение задач между работниками повышает эффективность и результативность работы, что подтверждается литературой [22, 23, 29]. Однако в модели величина p_{iu} задается как стоимость или штраф назначения. Это удобно с точки зрения вычислений, но данная величина неизбежно агрегирует в одном параметре разные эффекты: квалификацию, опыт, производительность, предпочтительность назначения и, возможно, организационные ограничения.

Описанные недостатки не связаны с внутренней несостоятельностью постановки задачи. При этом выбранная схема опирается на формальную основу RCPSP и использует возможности PDDL 2.1 для задания временных действий, числовых параметров и метрик плана, а при необходимости — возможности PDDL 3 для задания предпочтений и мягких ограничений. Вместе с этим формализация соответствует современному направлению LLM-as-formalizer, которое рассматривается как более надежное, чем LLM-планировщик. В совокупности это позволяет считать предложенную формализацию достаточно обоснованной и масштабируемой для дальнейшего развития.

Заключение

В рамках данной статьи была предложена формализация поставленной задачи проектного планирования. На основе неструктурированных текстовых данных D выделяются задачи проекта, зависимости между ними, исполнители, ресурсы, крайние сроки и другие параметры. Большая языковая модель в данной формализации определяется как отображение $\Phi_\theta : D \to M$, где $M = \langle Dom, Prob \rangle$ — представление задачи на языке PDDL. Далее задача сводится к построению плана $\pi \in \Pi(M)$, минимизирующего заданную целевую функцию качества. Данная постановка позволяет связать неструктурированную информацию о проекте с автоматическими планировщиками посредством перевода в PDDL с помощью LLM.

Предложенная целевая функция учитывает не только время завершения проекта, но и соблюдение крайних сроков задач, а также качество распределения исполнителей. Тем самым постановка RCPSP сохраняется, но расширяется в сторону мультикритериальной постановки. Использование PDDL обосновано тем, что этот язык поддерживает все необходимое для задачи: временные действия, числовые переменные и метрики плана.

Проведенное обсуждение показывает, что введенная формализация имеет как достоинства, так и ограничения. Ее сильные стороны состоят в разделении этапов формализации и планирования, в поддержке сложных ограничений, в возможности формальной проверки результата. Ограничения связаны прежде всего с качеством исходных текстовых данных и трудностью точного восстановления структуры проекта. Однако эти ограничения не свидетельствуют о слабости самой постановки, напротив, они указывают на естественные направления ее дальнейшего развития.

Проведенные эксперименты показали, что в среднем доля корректных планов у предложенного подхода составляет от 0.95 до 1.0 для всех четырех моделей, тогда как у LLM-as-planner эта доля колеблется от 0.1 до 0.9. Время завершения проекта при этом снижается незначительно, тогда как суммарная просрочка относительно крайних сроков уменьшается заметно сильнее. На более сложных и объемных задачах разница в качестве может стать более заметной.

Таким образом, предложенная формализация может рассматриваться как основа для дальнейших исследований применения LLM к задаче оптимизации проектного планирования. Она сохраняет связь с классической постановкой RCPSP, использует возможности PDDL и в то же время применяет LLM как средство формализации неструктурированных данных. Перспективными направлениями дальнейшей работы являются расширение постановки в сторону неопределенности и динамического планирования.

Статья выполнена в рамках научно-исследовательской работы по государственному заданию СПб ФИЦ РАН Mol_Lab (молодежная_лаб) No FFZF-2024-0003.

Литература

1. Wang L., Ma C., Feng X., *et al.* A survey on large language model based autonomous agents // Frontiers of Computer Science. 2024. Vol. 18, no. 6. P. 186345. DOI: 10.1007/s11704-024-40231-1.
2. Valmeekam K., Marquez M., Olmo A., *et al.* PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change // Advances in Neural Information Processing Systems. Vol. 36. 2023. P. 38975–38987. DOI: 10.52202/075280-1693.
3. McDermott D., Ghallab M., Howe A., *et al.* PDDL – The Planning Domain Definition Language: Technical Report / Yale Center for Computational Vision; Control. 1998.. URL: https://ipc08.icaps-conference.org/deterministic/data/mcdermott-et-al-tr-1998.pdf.
4. Helmert M. The Fast Downward Planning System // Journal of Artificial Intelligence Research. 2006. Vol. 26. P. 191–246. DOI: 10.1613/jair.1705.
5. Richter S., Westphal M. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks // Journal of Artificial Intelligence Research. 2010. Vol. 39. P. 127–177. DOI: 10.1613/jair.2972.
6. Tantakoun M., Muise C., Zhu X. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models // Findings of the Association for Computational Linguistics: ACL 2025 / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 25167–25188. DOI: 10.18653/v1/2025.findings-acl.1291.
7. Howey R., Long D., Fox M. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL // 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15–17 November 2004, Boca Raton, FL, USA. IEEE Computer Society, 2004. P. 294–301. DOI: 10.1109/ICTAI.2004.120.

8. Atkinson R. Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, Its Time to Accept Other Success Criteria // International Journal of Project Management. 1999. Vol. 17, no. 6. P. 337–342. DOI: 10.1016/S0263-7863(98)00069-6.

9. Brucker P., Drexl A., Moehring R., *et al.* Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods // European Journal of Operational Research. 1999. Vol. 112, no. 1. P. 3–41. DOI: 10.1016/S0377-2217(98)00204-5.

10. Herroelen W., Leus R. Project Scheduling Under Uncertainty: Survey and Research Potentials // European Journal of Operational Research. 2005. Vol. 165, no. 2. P. 289–306. DOI: 10.1016/j.ejor.2004.04.002.

11. Bahroun Z., As'ad R., Tanash M., Athamneh R. The Multi-Skilled Resource-Constrained Project Scheduling Problem: A Systematic Review and an Exploration of Future Landscapes // Management Systems in Production Engineering. 2024. Vol. 32, no. 1. P. 108–132. DOI: 10.2478/mspe-2024-0012.

12. Yehudai A., Eden L., Li A., *et al.* A Survey on Evaluation of LLM-based Agents // Findings of the Association for Computational Linguistics: ACL 2026 / ed. by M. Liakata, V.P. Moreira, J. Zhang, D. Jurgens. San Diego, California, United States: Association for Computational Linguistics, 2026. P. 26690–26714. DOI: 10.18653/v1/2026.findings-acl.1330.

13. Xie J., Zhang K., Chen J., *et al.* TravelPlanner: A Benchmark for Real-World Planning with Language Agents // Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 21–27 Jul 2024. P. 54590–54613. Proceedings of Machine Learning Research.

14. Xi Z., Chen W., Guo X., *et al.* The rise and potential of large language model based agents: a survey // Science China Information Sciences. 2025. Vol. 68, no. 2. P. 121101. DOI: 10.1007/s11432-024-4222-0.

15. Pallagani V., Roy K., Muppasani B., *et al.* On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS) // Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 34. 2024. P. 432–444. DOI: 10.1609/icaps.v34i1.31503.

16. Kambhampati S., Valmeekam K., Guan L., *et al.* Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks // Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 2024. P. 22895–22907. Proceedings of Machine Learning Research. DOI: 10.48550/arXiv.2402.01817.

17. Huang C., Zhang L. On the Limit of Language Models as Planning Formalizers // Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 4880–4904. DOI: 10.18653/v1/2025.acl-long.242.

18. Webb T., Mondal S.S., Momennejad I. A Brain-Inspired Agentic Architecture to Improve Planning with LLMs // Nature Communications. 2025. Vol. 16, no. 1. P. 8633. DOI: 10.1038/s41467-025-63804-5.

19. Corrêa A.B., Pereira A.G., Seipp J. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code // Advances in Neural Information Processing Systems 38 (NeurIPS 2025). 2025. DOI: 10.48550/arXiv.2503.18809.

20. Fox M., Long D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains // Journal of Artificial Intelligence Research. 2003. Vol. 20. P. 61–124. DOI: 10.1613/jair.1129.

21. Gerevini A., Long D. Preferences and Soft Constraints in PDDL3 // ICAPS Workshop on Planning with Preferences and Soft Constraints. 2006. P. 46–53. URL: https://artificial-intelligence.unibs.it/gerevini/papers/WS-PREF-ICAPS06.pdf.

22. Kristof-Brown A.L., Zimmerman R.D., Johnson E.C. Consequences of Individuals' Fit at Work: A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit // Personnel Psychology. 2005. Vol. 58, no. 2. P. 281–342. DOI: 10.1111/j.1744-6570.2005.00672.x.

23. Ruiz-Torres A.J., Ablanedo-Rosas J.H., Mukhopadhyay S., Paletta G. Scheduling Workers: A Multi-Criteria Model Considering Their Satisfaction // Computers & Industrial Engineering. 2019. Vol. 128. P. 747–754. DOI: 10.1016/j.cie.2018.12.070.

24. McCarthy J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I // Communications of the ACM. 1960. Vol. 3, no. 4. P. 184–195. DOI: 10.1145/367177.367199.

25. Browning T.R., Yassine A.A. Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited // International Journal of Production Economics. 2010. Vol. 126, no. 2. P. 212–228. DOI: 10.1016/j.ijpe.2010.03.009.

26. Bold M., Goerigk M. A Compact Reformulation of the Two-Stage Robust Resource-Constrained Project Scheduling Problem // Computers & Operations Research. 2021. Vol. 130. P. 105232. DOI: 10.1016/j.cor.2021.105232.

27. Ranjbar M., Khalilzadeh M., Kianfar F., Etminani K. An Optimal Procedure for Minimizing Total Weighted Resource Tardiness Penalty Costs in the Resource-Constrained Project Scheduling Problem // Computers & Industrial Engineering. 2012. Vol. 62, no. 1. P. 264–270. DOI: 10.1016/j.cie.2011.09.013.

28. Khoshjahan Y., Najafi A.A., Afshar-Nadjafi B. Resource Constrained Project Scheduling Problem with Discounted Earliness-Tardiness Penalties: Mathematical Modeling and Solving Procedure // Computers & Industrial Engineering. 2013. Vol. 66, no. 2. P. 293–300. DOI: 10.1016/j.cie.2013.06.017.

29. Goetz N., Wald A. Employee Performance in Temporary Organizations: The Effects of Person-Environment Fit and Temporariness on Task Performance and Innovative Performance // European Management Review. 2020. Vol. 18, no. 2. P. 25–41. DOI: 10.1111/emre.12438.

30. Sackett P.R. Reflections on a Career Studying Individual Differences in the Workplace // Annual Review of Organizational Psychology and Organizational Behavior. 2021. Vol. 8, no. 1. P. 1–18. DOI: 10.1146/annurev-orgpsych-012420-061939.

31. Chowdhury S., Schulz E., Milner M., Voort D.V.D. Core Employee Based Human Capital and Revenue Productivity in Small Firms: An Empirical Investigation // Journal of Business Research. 2014. Vol. 67, no. 11. P. 2473–2479. DOI: 10.1016/j.jbusres.2014.03.007.

32. Hart P., Nilsson N., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, no. 2. P. 100–107. DOI: 10.1109/TSSC.1968.300136.

33. Richter S., Helmert M. Preferred Operators and Deferred Evaluation in Satisficing Planning // Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19–23, 2009. AAAI Press, 2009. P. 273–280. DOI: 10.1609/icaps.v19i1.13345.

34. Perron L., Didier F., Gay S. The CP-SAT-LP Solver // 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. P. 3:1–3:2. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.CP.2023.3.

35. Рафиков Т.Р., Попцов А.В., Олисеенко В.Д., Абрамов М.В. formalizplanning: репозиторий с исходным кодом и данными вычислительных экспериментов. 2026. URL: https://github.com/dscspro/formalizplanning (дата обращения: 14.07.2026).

36. Kolisch R., Sprecher A. PSPLIB – A Project Scheduling Problem Library: OR Software – ORSEP Operations Research Software Exchange Program // European Journal of Operational Research. 1997. Vol. 96, no. 1. P. 205–216. DOI: 10.1016/S0377-2217(96)00170-1.

Рафиков Тимур Ришатович, стажер-исследователь, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Попцов Александр Владимирович, стажер-исследователь, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Олисеенко Валерий Дмитриевич, научный сотрудник, лаборатория прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Абрамов Максим Викторович, к.т.н., доцент, руководитель лаборатории прикладного искусственного интеллекта, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

DOI: 10.14529/cmse260206

FORMALIZATION OF THE PROJECT PLANNING PROBLEM WHEN WORKING WITH AN UNSTRUCTURED DESCRIPTION OF THE GOAL

© 2026 T.R. Rafikov, A.V. Poptsov, V.D. Oliseenko, M.V. Abramov

St. Petersburg Federal Research Center of the Russian Academy of Sciences

(Line 14 V.O. 39, St. Petersburg, 199178 Russia)

E-mail: trr@dscs.pro, avp@dscs.pro, vdo@dscs.pro, mva@dscs.pro

Received: 08.05.2026

Large language models demonstrate high efficiency in natural language processing tasks. However, their application to planning as planners under complex dependencies remains limited. One promising direction is to use large language models as a tool for constructing formal planning models. This paper considers the project planning problem in the case where the initial project information is given in an unstructured textual form. A formalization is proposed in which a large language model is used to extract this information and construct a formal model in PDDL. This approach makes it possible to connect textual descriptions of tasks, dependencies, deadlines, and resources with formal tools for automated planning. As a result, the project planning problem is transformed into a form suitable for verification, validation, and subsequent solution by algorithmic methods. In addition, this study extends the mathematical apparatus of the classical resource-constrained project scheduling problem. In addition to the total project completion time, penalties for missed deadlines and the quality of performer assignment are taken into account. Thus, the classical formulation is supplemented with criteria that reflect more complex and practically significant project management requirements. These changes make it possible to move from a simplified single-criterion model to a more realistic formulation. Translation into PDDL makes it possible to use automated tools for validation and planning of the stated problem. The computational experiments demonstrated that the proposed approach increased the proportion of valid plans from 0.1–0.9 to 0.95–1.0 for all four tested models compared with the direct use of an LLM as a planner.

Keywords: planning, large language models, formalization, LLM agent.

FOR CITATION

Rafikov T.R., Poptsov A.V., Oliseenko V.D., Abramov M.V. Formalization of the Project Planning Problem When Working with an Unstructured Description of the Goal. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2026. Vol. 15, no. 2. P. 104–126. (in Russian) DOI: 10.14529/cmse260206.

References

1. Wang L., Ma C., Feng X., *et al.* A survey on large language model based autonomous agents. Frontiers of Computer Science. 2024. Vol. 18, no. 6. P. 186345. DOI: 10.1007/s11704-024-40231-1.
2. Valmeekam K., Marquez M., Olmo A., *et al.* PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. Advances in Neural Information Processing Systems. Vol. 36. 2023. P. 38975–38987. DOI: 10.52202/075280-1693.

3. McDermott D., Ghallab M., Howe A., *et al.* PDDL – The Planning Domain Definition Language: Technical Report / Yale Center for Computational Vision; Control. 1998.. URL: https://ipc08.icaps-conference.org/deterministic/data/mcdermott-et-al-tr-1998.pdf.

4. Helmert M. The Fast Downward Planning System. Journal of Artificial Intelligence Research. 2006. Vol. 26. P. 191–246. DOI: 10.1613/jair.1705.

5. Richter S., Westphal M. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. Journal of Artificial Intelligence Research. 2010. Vol. 39. P. 127–177. DOI: 10.1613/jair.2972.

6. Tantakoun M., Muise C., Zhu X. LLMs as Planning Formalizers: A Survey for Leveraging Large Language Models to Construct Automated Planning Models. Findings of the Association for Computational Linguistics: ACL 2025 / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 25167–25188. DOI: 10.18653/v1/2025.findings-acl.1291.

7. Howey R., Long D., Fox M. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 15–17 November 2004, Boca Raton, FL, USA. IEEE Computer Society, 2004. P. 294–301. DOI: 10.1109/ICTAI.2004.120.

8. Atkinson R. Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, Its Time to Accept Other Success Criteria. International Journal of Project Management. 1999. Vol. 17, no. 6. P. 337–342. DOI: 10.1016/S0263-7863(98)00069-6.

9. Brucker P., Drexl A., Moehring R., *et al.* Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. European Journal of Operational Research. 1999. Vol. 112, no. 1. P. 3–41. DOI: 10.1016/S0377-2217(98)00204-5.

10. Herroelen W., Leus R. Project Scheduling Under Uncertainty: Survey and Research Potentials. European Journal of Operational Research. 2005. Vol. 165, no. 2. P. 289–306. DOI: 10.1016/j.ejor.2004.04.002.

11. Bahroun Z., As'ad R., Tanash M., Athamneh R. The Multi-Skilled Resource-Constrained Project Scheduling Problem: A Systematic Review and an Exploration of Future Landscapes. Management Systems in Production Engineering. 2024. Vol. 32, no. 1. P. 108–132. DOI: 10.2478/mspe-2024-0012.

12. Yehudai A., Eden L., Li A., *et al.* A Survey on Evaluation of LLM-based Agents. Findings of the Association for Computational Linguistics: ACL 2026 / ed. by M. Liakata, V.P. Moreira, J. Zhang, D. Jurgens. San Diego, California, United States: Association for Computational Linguistics, 2026. P. 26690–26714. DOI: 10.18653/v1/2026.findings-acl.1330.

13. Xie J., Zhang K., Chen J., *et al.* TravelPlanner: A Benchmark for Real-World Planning with Language Agents. Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 21–27 Jul 2024. P. 54590–54613. Proceedings of Machine Learning Research.

14. Xi Z., Chen W., Guo X., *et al.* The rise and potential of large language model based agents: a survey. Science China Information Sciences. 2025. Vol. 68, no. 2. P. 121101. DOI: 10.1007/s11432-024-4222-0.

15. Pallagani V., Roy K., Muppasani B., *et al.* On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 34. 2024. P. 432–444. DOI: 10.1609/icaps.v34i1.31503.

16. Kambhampati S., Valmeekam K., Guan L., *et al.* Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. Proceedings of the 41st International Conference on Machine Learning. Vol. 235. PMLR, 2024. P. 22895–22907. Proceedings of Machine Learning Research. DOI: 10.48550/arXiv.2402.01817.

17. Huang C., Zhang L. On the Limit of Language Models as Planning Formalizers. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) / ed. by W. Che, J. Nabende, E. Shutova, M.T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, 2025. P. 4880–4904. DOI: 10.18653/v1/2025.acl-long.242.

18. Webb T., Mondal S.S., Momennejad I. A Brain-Inspired Agentic Architecture to Improve Planning with LLMs. Nature Communications. 2025. Vol. 16, no. 1. P. 8633. DOI: 10.1038/s41467-025-63804-5.

19. Corrêa A.B., Pereira A.G., Seipp J. Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code. Advances in Neural Information Processing Systems 38 (NeurIPS 2025). 2025. DOI: 10.48550/arXiv.2503.18809.

20. Fox M., Long D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research. 2003. Vol. 20. P. 61–124. DOI: 10.1613/jair.1129.

21. Gerevini A., Long D. Preferences and Soft Constraints in PDDL3. ICAPS Workshop on Planning with Preferences and Soft Constraints. 2006. P. 46–53. URL: https://artificial-intelligence.unibs.it/gerevini/papers/WS-PREF-ICAPS06.pdf.

22. Kristof-Brown A.L., Zimmerman R.D., Johnson E.C. Consequences of Individuals' Fit at Work: A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit. Personnel Psychology. 2005. Vol. 58, no. 2. P. 281–342. DOI: 10.1111/j.1744-6570.2005.00672.x.

23. Ruiz-Torres A.J., Ablanedo-Rosas J.H., Mukhopadhyay S., Paletta G. Scheduling Workers: A Multi-Criteria Model Considering Their Satisfaction. Computers & Industrial Engineering. 2019. Vol. 128. P. 747–754. DOI: 10.1016/j.cie.2018.12.070.

24. McCarthy J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. Communications of the ACM. 1960. Vol. 3, no. 4. P. 184–195. DOI: 10.1145/367177.367199.

25. Browning T.R., Yassine A.A. Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited. International Journal of Production Economics. 2010. Vol. 126, no. 2. P. 212–228. DOI: 10.1016/j.ijpe.2010.03.009.

26. Bold M., Goerigk M. A Compact Reformulation of the Two-Stage Robust Resource-Constrained Project Scheduling Problem. Computers & Operations Research. 2021. Vol. 130. P. 105232. DOI: 10.1016/j.cor.2021.105232.

27. Ranjbar M., Khalilzadeh M., Kianfar F., Etminani K. An Optimal Procedure for Minimizing Total Weighted Resource Tardiness Penalty Costs in the Resource-Constrained Project Scheduling Problem. Computers & Industrial Engineering. 2012. Vol. 62, no. 1. P. 264–270. DOI: 10.1016/j.cie.2011.09.013.

28. Khoshjahan Y., Najafi A.A., Afshar-Nadjafi B. Resource Constrained Project Scheduling Problem with Discounted Earliness-Tardiness Penalties: Mathematical Modeling and Solving Procedure. Computers & Industrial Engineering. 2013. Vol. 66, no. 2. P. 293–300. DOI: 10.1016/j.cie.2013.06.017.

29. Goetz N., Wald A. Employee Performance in Temporary Organizations: The Effects of Person-Environment Fit and Temporariness on Task Performance and Innovative Performance. European Management Review. 2020. Vol. 18, no. 2. P. 25–41. DOI: 10.1111/emre.12438.

30. Sackett P.R. Reflections on a Career Studying Individual Differences in the Workplace. Annual Review of Organizational Psychology and Organizational Behavior. 2021. Vol. 8, no. 1. P. 1–18. DOI: 10.1146/annurev-orgpsych-012420-061939.

31. Chowdhury S., Schulz E., Milner M., Voort D.V.D. Core Employee Based Human Capital and Revenue Productivity in Small Firms: An Empirical Investigation. Journal of Business Research. 2014. Vol. 67, no. 11. P. 2473–2479. DOI: 10.1016/j.jbusres.2014.03.007.

32. Hart P., Nilsson N., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4, no. 2. P. 100–107. DOI: 10.1109/TSSC.1968.300136.

33. Richter S., Helmert M. Preferred Operators and Deferred Evaluation in Satisficing Planning. Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009), Thessaloniki, Greece, September 19–23, 2009. AAAI Press, 2009. P. 273–280. DOI: 10.1609/icaps.v19i1.13345.

34. Perron L., Didier F., Gay S. The CP-SAT-LP Solver. 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. P. 3:1–3:2. Leibniz International Proceedings in Informatics (LIPIcs). DOI: 10.4230/LIPIcs.CP.2023.3.

35. Rafikov T.R., Poptsov A.V., Oliseenko V.D., Abramov M.V. formalizplanning: repository containing source code and computational experiment data. 2026. URL: https://github.com/dscspro/formalizplanning (accessed: 14.07.2026).

36. Kolisch R., Sprecher A. PSPLIB – A Project Scheduling Problem Library: OR Software – ORSEP Operations Research Software Exchange Program. European Journal of Operational Research. 1997. Vol. 96, no. 1. P. 205–216. DOI: 10.1016/S0377-2217(96)00170-1.