**Лисенков Сергей Васильевич**

*студент, Государственный университет «Дубна», Россия, город Дубна*

# ЭВРИСТИЧЕСКИЙ МЕТОД ВОССТАНОВЛЕНИЯ ИЕРАРХИЧЕСКОЙ СТРУКТУРЫ ТЕХНИЧЕСКОГО ЗАДАНИЯ ДЛЯ ПОСЛЕДУЮЩЕЙ НОРМАТИВНОЙ ПРОВЕРКИ

**Аннотация.** В статье рассматривается эвристический метод восстановления иерархической структуры технического задания из последовательности текстовых блоков. Метод использует три признака заголовка: цифровую нумерацию, запись короткого блока в верхнем регистре и совпадение с заданным перечнем наименований разделов. Для нумерованных заголовков уровень иерархии определяется количеством частей цифрового префикса. Дерево разделов строится за один последовательный проход с использованием стека. После построения дерева для каждого раздела определяются начальный и конечный блоки и формируется полный текст раздела. Реализованный метод проверен на сценариях с заголовками трёх уровней, пропущенным уровнем иерархии, отсутствием заголовков и эвристическим распознаванием нумерованных строк. Все четыре сценария структурного модуля завершились с ожидаемым результатом. Дополнительные два сценария подтвердили возможность использовать восстановленную структуру для определения отсутствующего обязательного раздела без ложного срабатывания на полном документе.

**Ключевые слова:** техническое задание, структурный анализ, иерархия документа, эвристический алгоритм, ГОСТ, обработка текста, нормативная проверка.

**Annotation.** The article presents a heuristic method for reconstructing the hierarchical structure of a technical specification from a sequence of text blocks. The method uses three heading features: numeric numbering, a short uppercase block, and exact matching against a predefined list of section names. For numbered headings, the hierarchy level is determined by the number of components in the numeric prefix. The section tree is built in a single sequential pass using a stack. After the tree is

constructed, the start and end blocks and the complete text of each section are determined. The implemented method was tested on scenarios involving three heading levels, a skipped hierarchy level, a document without headings, and heuristic recognition of numbered lines. All four structural-analysis scenarios produced the expected results. Two additional scenarios confirmed that the reconstructed structure can be used to detect a missing mandatory section without producing a false issue for a complete document.

**Keywords:** technical specification, structural analysis, document hierarchy, heuristic algorithm, GOST, text processing, compliance checking.

## Введение

ГОСТ 34.602-89 задаёт состав технического задания на создание автоматизированной системы и перечень его основных разделов [1]. ГОСТ 19.201-78 устанавливает структуру технического задания на программу или программное изделие [2]. Для автоматической проверки наличия и порядка разделов сначала требуется восстановить структуру самого документа.

После извлечения текста из DOCX или PDF документ представлен последовательностью текстовых блоков. Полного текста недостаточно для структурной проверки: требуется определить, какой блок является заголовком, к какому уровню он относится и какие последующие блоки образуют содержимое раздела.

Заголовки в техническом задании могут быть оформлены обычными абзацами без специальных стилей. В PDF сведения о стилях абзацев отсутствуют. В разработанном прототипе промежуточная модель не использует параметры форматирования, поэтому структура восстанавливается по текстовым признакам.

**Цель исследования —** разработать и проверить эвристический метод, который преобразует последовательность текстовых блоков технического задания в иерархическое дерево разделов для последующего сравнения со структурой, заданной нормативным стандартом.

## Основная часть

### Представление исходного документа

Модуль структурного анализа получает унифицированный список текстовых блоков $B = \{b_1, b_2, \dots, b_n\}$, где $b_i$ — отдельный блок документа. Для DOCX блоки формируются после извлечения абзацев и нормализации текста. Для PDF используются текстовые блоки, сформированные из извлечённого содержимого страниц.

Дальнейшая логика работает с единым представлением и не зависит от расширения исходного файла. Результат структурного анализа содержит список найденных заголовков, дерево разделов и текстовое содержимое каждого раздела.

Модуль разделён на три этапа: определение заголовков, построение иерархического дерева и добавление границ разделов.

### Эвристическое определение заголовков

Детектор последовательно обрабатывает непустые блоки. Блок считается кандидатом в заголовки, если выполняется хотя бы один из трёх признаков: допустимая цифровая нумерация, короткая запись в верхнем регистре или точное совпадение с ключевым названием раздела.

Логику детектора можно записать как $H(b_i) = N(b_i) \vee U(b_i) \vee K(b_i)$, где $N$ — признак цифровой нумерации, $U$ — признак верхнего регистра, $K$ — совпадение с перечнем названий разделов.

Первая эвристика распознаёт строки с типовой цифровой нумерацией. В реализации предусмотрено несколько регулярных выражений для заголовков разной глубины. После отдельных двухчастных числовых префиксов проверяется первый текстовый токен. Если он входит в список обозначений измерений, строка не считается заголовком. Эта проверка уменьшает число ложных срабатываний на числовых характеристиках.

Вторая эвристика работает с ненумерованными заголовками, записанными прописными буквами. Блок должен быть непустым, иметь ограниченную длину, содержать хотя бы одну букву и совпадать со своей версией в верхнем регистре.

Третья эвристика сравнивает нормализованный текст блока с перечнем ключевых названий разделов технического задания. Совпадение позволяет определить заголовок без номера и без записи в верхнем регистре.

Таблица 1 — Эвристики определения заголовков

| Эвристика | Проверяемый признак | Пример |
|---|---|---|
| Цифровая нумерация | Строка начинается с допустимого числового префикса | 2.3 Требования к системе |
| Верхний регистр | Короткий блок содержит буквы и полностью записан в верхнем регистре | ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ |
| Ключевое наименование | Текст совпадает с элементом перечня названий разделов | Источники разработки |

Проверки выполняются в фиксированном порядке: цифровой признак, верхний регистр, ключевое наименование. Для найденного заголовка сохраняются исходный текст и индекс начального блока.

**Определение уровня иерархии**

Уровень нумерованного заголовка определяется количеством непустых частей цифрового префикса. Для «1 Требования к системе» получается уровень 1, для «1.2 Требования к функциям» — уровень 2, для «1.2.3 Требования к времени выполнения» — уровень 3.

В программной реализации цифровой префикс разделяется по точкам, пустые элементы удаляются, а количество оставшихся частей принимается за уровень. Ненумерованные заголовки получают уровень 1.

Метод использует уже существующую нумерацию документа и не требует отдельного классификатора уровней.

### Построение иерархического дерева

После работы детектора формируется плоская последовательность заголовков $S = \{s_1, s_2, \dots, s_m\}$. Каждый элемент содержит текст, уровень и позицию в исходном документе.

Для преобразования последовательности в дерево используется стек. Для очередного заголовка создаётся узел с полями title, level, start_block и children. Если стек пуст, узел добавляется в корень дерева. Если уровень нового заголовка больше уровня вершины стека, узел становится дочерним для текущей вершины.

Если уровень вершины стека равен или больше уровня нового заголовка, элементы удаляются из стека до появления узла более высокого уровня иерархии. Новый заголовок добавляется к найденному родителю, после чего помещается в стек.

Для последовательности «1 Требования к системе» → «1.1 Требования к системе в целом» → «1.1.1 Требования к персоналу» → «1.2 Требования к функциям» первые три узла образуют вложенную цепочку. При обработке «1.2» из стека удаляются узлы уровней 3 и 2, после чего новый раздел становится дочерним для раздела уровня 1.

Каждый найденный заголовок добавляется в стек один раз и удаляется из него не более одного раза. Для m заголовков построение дерева имеет линейную сложность O(m) по числу операций со стеком.

### Обработка пропущенных уровней

Алгоритм не создаёт искусственные промежуточные узлы. Если после заголовка уровня 1 встречается заголовок уровня 3, новый узел связывается с ближайшим существующим родителем более высокого уровня.

Сценарий с заголовками уровней 1 и 3 без уровня 2 был выделен в отдельный граничный тест. Ожидаемый результат — корректное дерево без исключения. Фактический результат совпал с ожидаемым.

### Определение границ разделов

После построения дерева каждый узел содержит начало раздела, но ещё не содержит его полный текст. Обогатитель структуры определяет для раздела начальный и конечный индексы, список входящих блоков и объединённый текст.

На выходе получается структура «заголовок → уровень → границы → содержимое → дочерние разделы». Она используется структурными и содержательными проверками.

Например, проверка функциональных требований может анализировать текст конкретного раздела, а не выполнять поиск по всему документу. Структурная проверка использует плоский список найденных заголовков для контроля наличия и порядка разделов.

### Использование структуры в нормативной проверке

После восстановления дерева найденные заголовки сопоставляются с эталонной структурой ГОСТ 34.602-89 или ГОСТ 19.201-78 [1, 2]. В разработанном сервисе на этой структуре выполняются проверки наличия обязательных разделов, порядка разделов и полноты структуры.

Проверка наличия сопоставляет нормализованные наименования найденных разделов с шаблонами из базы знаний. Для контроля порядка используется последовательность найденных заголовков. Разделы, нарушающие ожидаемую последовательность, определяются через поиск наибольшей возрастающей подпоследовательности.

Восстановленное дерево также передаётся содержательным проверкам. Это позволяет проверять цель, область применения, функциональные требования и критерии приёмки внутри соответствующих разделов.

### Тестирование метода

Тестирование выполнялось на синтетических документах и минимальных наборах блоков. Такой формат позволяет заранее задать входную структуру и точный ожидаемый результат. Для тестов использовался pytest [8].

Таблица 2 — Результаты тестирования структурного анализа

| № | Сценарий | Входные данные | Ожидаемый результат | Фактический результат | Статус |
|---|---|---|---|---|---|
| 1 | Определение заголовков нескольких уровней | DOCX с нумерованными и ключевыми заголовками | Определены 3 заголовка уровней 1, 2 и 3 | Все 3 заголовка определены | Пройден |
| 2 | Эвристическое определение нумерованного заголовка | Строки вида «1. Назначение» | Нумерованные строки определяются как заголовки | Заголовки определены эвристически | Пройден |
| 3 | Пропуск уровня иерархии | Заголовки уровней 1 и 3 без уровня 2 | Дерево строится без ошибки | Пропуск обработан | Пройден |
| 4 | Документ без заголовков | Список обычных текстовых блоков | Возвращается пустое дерево | Получено пустое дерево | Пройден |

Четыре из четырёх заданных сценариев структурного модуля завершились с ожидаемым результатом. Этот показатель относится только к заранее заданным тестам и не является значением статистической точности распознавания заголовков на произвольных документах.

Ещё два теста проверяли использование восстановленной структуры в нормативной проверке. В первом документе отсутствовал обязательный раздел «Назначение», во втором присутствовали все обязательные разделы.

Таблица 3 — Проверка использования восстановленной структуры

| Сценарий | Ожидаемый результат | Фактический результат | Статус |
|---|---|---|---|
| Отсутствует раздел «Назначение» | Замечание об отсутствии раздела | Замечание сформировано | Пройден |
| Присутствуют все обязательные разделы | Структурных замечаний нет | Ложное замечание не сформировано | Пройден |

Оба сценария завершились с ожидаемым результатом. В первом случае восстановленная структура позволила определить отсутствующий обязательный раздел. Во втором случае полный документ не вызвал ложного замечания.

## Ограничения метода

В проведённом тестировании не использовался размеченный корпус реальных технических заданий. Значения Precision, Recall и F1 для определения заголовков не рассчитывались.

Эвристика верхнего регистра может принять короткий выделенный фрагмент за заголовок, если весь текст записан прописными буквами. Обратный случай возможен для ненумерованного заголовка, который записан обычным регистром и отсутствует в словаре ключевых названий.

Для PDF применяется тот же набор текстовых признаков. Результат зависит от качества извлечения текста, поскольку PDF не предоставляет структурную модель абзацев, эквивалентную DOCX.

Ненумерованные заголовки текущая реализация относит к первому уровню. Вложенность ненумерованных разделов по трём используемым признакам не восстанавливается.

Пропущенные уровни также не восстанавливаются искусственно. Заголовок связывается с ближайшим существующим родителем более высокого уровня.

## Заключение

Разработан эвристический метод восстановления иерархической структуры технического задания из плоского набора текстовых блоков. Для определения заголовков используются три признака: цифровая нумерация, верхний регистр и совпадение с ключевыми названиями разделов.

Уровень нумерованного заголовка определяется количеством частей цифрового префикса. Дерево строится стековым алгоритмом за один последовательный проход по списку найденных заголовков. Пропуск промежуточного уровня не приводит к ошибке построения.

После построения дерева для каждого раздела вычисляются границы и формируется полный текст. Полученная структура используется для проверки

наличия обязательных разделов, их порядка и содержательных признаков внутри разделов.

Четыре тестовых сценария структурного модуля завершились с ожидаемым результатом. Ещё два сценария подтвердили использование восстановленной структуры для нормативной проверки: отсутствующий раздел был найден, а на документе с полной структурой ложное замечание не сформировано.

Текущие результаты не определяют статистическую точность метода на реальных технических заданиях. Для такой оценки требуется размеченная выборка документов с эталонными границами разделов и уровнями каждого заголовка.

## Список литературы


1. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. — М.: Издательство стандартов, 1989. — 16 с.
2. ГОСТ 19.201-78. Техническое задание. Требования к содержанию и оформлению. — М.: Издательство стандартов, 1978. — 4 с.
3. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. — М.: Издательство стандартов, 1995. — 25 с.
4. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд. — М.: Русская редакция, 2014. — 737 с.
5. Маннинг К., Рагхаван П., Шютце Г. Введение в информационный поиск. — М.: Вильямс, 2011. — 520 с.
6. Мэннинг К., Шютце Г. Основы статистической обработки естественного языка. — М.: Мир, 2002. — 986 с.
7. Bird S., Klein E., Loper E. Natural Language Processing with Python. — Sebastopol: O’Reilly Media, 2009.
8. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. — Hoboken: Wiley, 2011.

## References


1. GOST 34.602-89. Informatsionnaya tekhnologiya. Kompleks standartov na avtomatizirovannye sistemy. Tekhnicheskoe zadanie na sozdanie avtomatizirovannoy sistemy [Information technology. Set of standards for automated systems. Technical specification for creation of an automated system]. Moscow: Izdatelstvo standartov; 1989. 16 p. (In Russ.).
2. GOST 19.201-78. Tekhnicheskoe zadanie. Trebovaniya k soderzhaniyu i oformleniyu [Technical specification. Requirements for content and presentation]. Moscow: Izdatelstvo standartov; 1978. 4 p. (In Russ.).
3. GOST 2.105-95. Edinaya sistema konstruktorskoy dokumentatsii. Obshchie trebovaniya k tekstovym dokumentam [Unified system for design documentation. General requirements for textual documents]. Moscow: Izdatelstvo standartov; 1995. 25 p. (In Russ.).
4. Wiegers K., Beatty J. Software Requirements. 3rd ed. Redmond: Microsoft Press; 2013.
5. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press; 2008.
6. Manning C. D., Schütze H. Foundations of Statistical Natural Language Processing. Cambridge: MIT Press; 1999.
7. Bird S., Klein E., Loper E. Natural Language Processing with Python. Sebastopol: O'Reilly Media; 2009.
8. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken: Wiley; 2011.