Лисенков Сергей Васильевич

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

ГИБРИДНАЯ МОДЕЛЬ АВТОМАТИЗИРОВАННОГО COMPLIANCE-АНАЛИЗА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ НА ОСНОВЕ RULE-BASED И СЛОВАРНЫХ МЕТОДОВ

Аннотация. В статье рассматривается гибридная модель автоматизированного compliance-анализа технических заданий на соответствие требованиям ГОСТ. Модель объединяет rule-based проверки структуры и содержания документа, регулярные выражения, словарь расплывчатых формулировок и глоссарий канонических терминов. Реализовано 15 типов проверок, объединённых в 7 категорий. Входными данными служат документы DOCX и PDF, результатом анализа является структурированный список замечаний с указанием правила, критичности, проблемного фрагмента и рекомендации. Прототип реализован на Python и FastAPI. В тестировании использовано 22 модульных, граничных, функциональных и интеграционных сценария, все сценарии завершены с ожидаемым результатом. На компьютере с процессором AMD Ryzen 5 5500U и 16 ГБ оперативной памяти полный анализ тестовых документов выполнялся менее чем за 60 секунд. Извлечение текста из PDF занимало в среднем на 40 % больше времени по сравнению с DOCX сопоставимого объёма.

Ключевые слова: техническое задание, compliance-анализ, ГОСТ, rule-based, обработка естественного языка, словарный анализ, техническая документация, автоматизированная проверка.

Annotation. The article presents a hybrid model for automated compliance analysis of technical specifications against GOST requirements. The model combines rule-based checks of document structure and content, regular expressions, a dictionary of vague expressions, and a glossary of canonical terms. The implemented prototype contains 15 check types grouped into 7 categories. DOCX and PDF documents are used as input, while

the analysis result is a structured list of issues containing the applied rule, severity level, source fragment, and recommendation. The prototype is implemented in Python using FastAPI. Testing included 22 unit, boundary, functional, and integration scenarios; all scenarios produced the expected results. On a computer with an AMD Ryzen 5 5500U processor and 16 GB of RAM, complete analysis of the test documents took less than 60 seconds. Text extraction from PDF files took approximately 40% longer on average than extraction from DOCX files of comparable size.

Keywords: technical specification, compliance analysis, GOST, rule-based, natural language processing, dictionary analysis, technical documentation, automated verification.

Введение

ГОСТ 34.602-89 задаёт состав технического задания на создание автоматизированной системы и определяет обязательные разделы документа [1]. ГОСТ 19.201-78 устанавливает требования к техническому заданию на программу или программное изделие [2]. Дополнительные требования к терминологии и оформлению задаются ГОСТ 34.003-90 и ГОСТ 2.105-95 [3, 4].

Проверка документа включает контроль структуры, формулировок требований, числовых характеристик, терминологии и ссылок. При ручной обработке технического задания объёмом 30–50 страниц проверка занимает несколько часов [6]. Один документ может одновременно содержать пропущенный обязательный раздел, расплывчатое требование, числовой показатель без единицы измерения и ссылку на отсутствующее приложение. Эти нарушения требуют разных методов обнаружения.

Средства проверки текста Microsoft Word работают с орфографией, грамматикой и редакторскими рекомендациями, но не сопоставляют структуру технического задания с ГОСТ. Системы управления требованиями решают другую задачу: требования создаются и связываются внутри специализированной среды.

Исследовательские NLP-модели позволяют классифицировать и анализировать требования, но для их обучения требуется набор данных с заранее заданной разметкой [10].

В разработанном прототипе обучаемая модель не используется. Анализ выполняется детерминированными правилами, регулярными выражениями, словарями и алгоритмами сравнения текста.

Цель исследования - разработать и проверить гибридную модель автоматизированного compliance-анализа технических заданий, которая объединяет rule-based и словарные методы в одном последовательном конвейере обработки.

Основная часть

Разрабатываемая модель ориентирована на технические задания в форматах DOCX и PDF. В качестве нормативной основы используются ГОСТ 34.602-89, ГОСТ 19.201-78, ГОСТ 34.003-90, ГОСТ 2.105-95 и ГОСТ 8.417-2002.

Обработка документа состоит из пяти последовательных этапов:

1. извлечение текста;
2. нормализация;
3. определение структуры документа;
4. выполнение compliance-проверок;
5. формирование отчёта.

Для DOCX текст извлекается средствами библиотеки python-docx. Для PDF используется pypdf. Внутри системы исходный файл преобразуется в унифицированную модель, содержащую текстовые блоки, полный текст и сведения о найденных разделах. После этого дальнейшие проверки не зависят от исходного формата файла.

Нормализация уменьшает влияние различий в оформлении текста. Выполняется приведение регистра, обработка повторяющихся пробелов и

табуляций, нормализация кавычек и подготовка строк к сопоставлению с шаблонами.

Структура документа определяется до содержательных проверок. Детектор анализирует нумерацию заголовков, написание строк в верхнем регистре и совпадение текста с ожидаемыми названиями разделов. После определения уровней строится иерархическое дерево. Стековый алгоритм сохраняет связь между разделами и подразделами и обрабатывает ситуацию, когда в документе после заголовка первого уровня сразу расположен заголовок третьего уровня.

Rule-based часть модели отвечает за те нарушения, которые можно формализовать через однозначное условие.

Например, наличие обязательных разделов проверяется сопоставлением нормализованных заголовков документа с набором допустимых названий. Для каждого обязательного раздела хранится каноническое название и допустимые варианты. Если совпадение не найдено, система формирует отдельное замечание.

Для проверки последовательности разделов применяется алгоритм поиска наибольшей возрастающей подпоследовательности. Каждому найденному разделу соответствует его позиция в эталонной структуре. Элементы, нарушающие ожидаемую последовательность, определяются как потенциально расположенные в неправильном порядке.

Содержательные проверки используют наборы лексических маркеров. Для определения блока с целью создания системы выполняется поиск конструкций «назначение», «цель создания», «цель разработки», «предназначен для», «создаётся для». Отдельные наборы маркеров применяются для области применения, функциональных требований, требований к надёжности, производительности, безопасности, совместимости и критериев приёмки.

Такой механизм не определяет смысл текста на уровне языковой модели. Он отвечает на более узкий вопрос: присутствуют ли в анализируемом фрагменте признаки обязательного смыслового блока.

Словарная часть используется при анализе формулировок и терминологии. Словарь нежелательных конструкций содержит слова и выражения, которые делают требование труднопроверяемым без дополнительной метрики. В него входят конструкции типа «быстро», «удобно», «понятный интерфейс» и другие качественные оценки. Поиск выполняется регулярными выражениями с учётом предусмотренных в базе вариантов слов.

Формулировка «система должна быстро обрабатывать запросы» вызывает замечание. Формулировка «время обработки запроса не должно превышать 2 с» не вызывает это замечание, поскольку содержит измеримый показатель.

Терминологическая проверка использует отдельный глоссарий. Для каждого понятия задаётся каноническое обозначение и варианты написания. Найденный нестандартный вариант связывается с рекомендуемым термином. Источником терминов выступает нормативная база, включая ГОСТ 34.003-90 [3].

Числовые проверки построены на регулярных выражениях. Первый тип определяет наличие числовых характеристик в требованиях. Второй анализирует числа на наличие сопровождающей единицы измерения. Единицы сопоставляются со списком, ориентированным на обозначения системы СИ.

Ссылочная целостность проверяется отдельными модулями для рисунков, таблиц и приложений. Алгоритм ищет упоминания соответствующих объектов и сопоставляет их с объявлениями или подписями в документе. Несогласованность формирует замечание низкой критичности.

Для поиска дублирующих формулировок используется другой механизм. Текст двух блоков нормализуется по регистру, пробелам и пунктуации. После этого рассчитывается сходство множеств токенов по коэффициенту Жаккара и строковое сходство через SequenceMatcher. Пара фрагментов помечается как потенциальный дубликат при превышении заданного порога. Модель получила семь категорий compliance-проверок.

Таблица 1 - Состав гибридной модели compliance-анализа

Категория	**Количество типов**	**Основной метод**	**Критичность**
Структурные проверки	3	Шаблоны разделов, сравнение порядка	высокая / средняя
Содержательные проверки	5	Rule-based поиск лексических маркеров	высокая / средняя
Качество формулировок	2	Словарь + регулярные выражения	средняя
Числовые характеристики	2	Регулярные выражения	средняя / низкая
Ссылочная целостность	1	Сопоставление упоминаний и объектов	низкая
Терминология	1	Словарь канонических терминов	средняя
Дублирующие формулировки	1	Жаккар + SequenceMatcher	низкая
Итого	15	-	6 высоких, 6 средних, 3 низких

Гибридность модели состоит в использовании нескольких детерминированных способов анализа одного документа. Структурные нарушения обрабатываются правилами и шаблонами, формулировки и терминология - словарями, числовые параметры - регулярными выражениями, дубликаты - алгоритмами текстового сходства. Эти методы выполняются внутри единого конвейера и формируют одинаковый тип результата.

Каждое замечание содержит идентификатор правила, категорию, уровень критичности, сообщение, исходный фрагмент и рекомендацию. Модуль проверки не формирует отдельный формат отчёта. Все результаты приводятся к одной модели Issue, после чего агрегируются.

Такое разделение позволило реализовать проверки независимыми программными модулями. Добавление нового правила не требует изменения интерфейса существующих проверок.

База знаний также отделена от основного конвейера. В ней размещены шаблоны обязательных разделов, наборы лексических маркеров, словарь нежелательных формулировок, терминологический глоссарий и метаданные правил. Rule-based алгоритм отвечает за способ проверки, а база знаний хранит значения, с которыми он работает.

Прототип реализован на Python 3.10+. Серверная часть построена на FastAPI. Для DOCX используется python-docx, для PDF - pypdf. Модульные и интеграционные тесты реализованы с помощью pytest, TestClient и httpx.

Клиентская часть выполнена на HTML5, CSS3 и JavaScript. Основной маршрут POST /api/upload принимает DOCX или PDF и возвращает JSON с результатами анализа. Неподдерживаемый формат получает HTTP 415. Ошибка обработки содержимого возвращает HTTP 422. Экспорт отчёта поддерживает JSON, CSV и DOCX.

Пользователь получает не только число найденных проблем. Для каждого нарушения выводится фрагмент исходного документа и рекомендация. Замечания можно фильтровать по категории и критичности.

Проверка работоспособности модели

Тестирование проводилось на четырёх уровнях: модульном, граничном, функциональном и интеграционном [11, 12].

Синтетические документы создавались программно с помощью python-docx и reportlab. Такой набор позволял заранее задавать нарушение и ожидаемый ответ конкретного модуля. Для части проверок использовались пары документов: первый содержал нарушение, второй - корректную формулировку или структуру.

Например, тест проверки формулировок использовал требование «система должна быстро…». Ожидаемым результатом было формирование замечания с проблемным фрагментом и рекомендацией. Второй сценарий использовал

требование «время отклика ≤ 2 с». Для него список замечаний должен был остаться пустым. Оба сценария прошли.

Структурный модуль проверялся на документе с отсутствующим разделом «Назначение» и на документе с полным набором обязательных разделов. В первом случае замечание было сформировано, во втором ложное срабатывание отсутствовало.

Отдельно проверялись пустой DOCX, PDF без текстового слоя, пропуск уровня в иерархии заголовков и документ без заголовков.

Таблица 2 - Результаты тестовых сценариев

Тип тестирования	**Число сценариев**	**Пройдено**	**Не пройдено**
Модульное	7	7	0
Граничное	4	4	0
Функциональное	7	7	0
Интеграционное	4	4	0
Итого	22	22	0

Интеграционные сценарии проверяли загрузку и анализ DOCX, обработку неподдерживаемого файла, формирование DOCX-отчёта и маршрут контроля состояния сервиса. Для корректного DOCX сервер вернул HTTP 200 и JSON с полями checks, issues и report. Файл TXT был отклонён со статусом HTTP 415. DOCX-отчёт был сформирован и корректно открыт после получения от сервера. Маршрут /api/health вернул HTTP 200 и статус ok.

Результат 22 из 22 показывает прохождение заранее заданных программных сценариев. Он не является значением Accuracy модели на выборке реальных технических заданий.

Для расчёта Precision, Recall и F1 требуется отдельный размеченный корпус документов, где каждое реальное нарушение заранее отмечено экспертом. Такой корпус в проведённое тестирование не входил, поэтому статистические показатели качества обнаружения нарушений не рассчитывались.

Функциональная проверка проводилась через веб-интерфейс. Загружались DOCX и PDF разного объёма и структуры: документы с пропущенными разделами, расплывчатыми формулировками, ошибками ссылок и корректные документы. Контролировались загрузка, получение результата, отображение замечаний, фильтрация и экспорт.

Тестирование времени обработки выполнялось на компьютере с AMD Ryzen 5 5500U и 16 ГБ оперативной памяти. Полный конвейер укладывался в ограничение менее 60 секунд. Извлечение текста из PDF-файлов занимало в среднем примерно на 40 % больше времени, чем обработка DOCX аналогичного объёма. Разница связана с этапом извлечения текста. DOCX предоставляет структурированную объектную модель параграфов, тогда как PDF обрабатывается постранично и не хранит структуру документа в том же виде.

Повторная обработка одного файла при неизменной базе знаний формирует одинаковый результат. Это определяется отсутствием случайных операций и обучаемых компонентов в конвейере.

Ограничения модели

Текущая версия не выполняет OCR. PDF без текстового слоя возвращает пустой результат извлечения и не может быть полноценно проанализирован.

Терминологическая проверка ограничена содержимым глоссария. Термин, которого нет в словаре, не может быть классифицирован как канонический или нестандартный только на основании текущего алгоритма.

Детектор структуры обрабатывает стандартную нумерацию, верхний регистр и ожидаемые названия разделов. Произвольное визуальное оформление заголовка без соответствующих текстовых признаков может не распознаться. Алгоритм поиска дубликатов зависит от заданных порогов коэффициента Жаккара и строкового сходства. Их изменение влияет на количество фрагментов, которые система помечает как потенциально дублирующиеся.

Модель проверяет наличие заданных признаков и формальных условий. Она не определяет фактическую техническую реализуемость требования и не проверяет корректность предметного содержания проекта. Эти задачи требуют экспертной оценки.

Заключение

Разработана гибридная модель автоматизированного compliance-анализа технических заданий. Она объединяет rule-based проверки, регулярные выражения, словарь расплывчатых формулировок, терминологический глоссарий и алгоритмы сравнения текста.

В модели реализовано 15 типов проверок в 7 категориях. Шесть проверок имеют высокий уровень критичности, шесть - средний, три - низкий. Проверяются структура документа, наличие обязательных смысловых блоков, качество формулировок, числовые характеристики, единицы измерения, ссылки, терминология и дубликаты.

Прототип принимает документы DOCX и PDF. Результат анализа содержит идентификатор правила, категорию нарушения, критичность, фрагмент текста и рекомендацию. Результаты экспортируются в JSON, CSV и DOCX.

Выполнено 22 тестовых сценария: 7 модульных, 4 граничных, 7 функциональных и 4 интеграционных. Все 22 сценария завершились с ожидаемым результатом. На компьютере с AMD Ryzen 5 5500U и 16 ГБ оперативной памяти обработка тестового документа выполнялась менее чем за 60 секунд. Извлечение текста из PDF занимало в среднем на 40 % больше времени по сравнению с DOCX сопоставимого объёма.

Проведённые тесты подтверждают работоспособность программной реализации для заданных сценариев, но не определяют Precision, Recall и F1 на реальных технических заданиях. Для расчёта этих метрик требуется размеченный корпус документов и отдельный эксперимент с экспертной разметкой.

Текущие ограничения прототипа - отсутствие OCR для сканированных PDF, зависимость терминологического анализа от полноты словаря и чувствительность поиска дубликатов к заданным порогам сходства. Расширение корпуса тестовых документов позволит отдельно измерить качество каждого типа compliance-проверки.

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 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения. - М.: Издательство стандартов, 1990. - 26 с.
4. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. - М.: Издательство стандартов, 1995. - 25 с.
5. Бакулин И. А., Бычкова Н. А., Червяков Л. М. Моделирование технических требований к программным продуктам // Современные наукоемкие технологии. - 2021. - № 4.
6. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд. - М.: Русская редакция, 2014. - 737 с.
7. Bird S., Klein E., Loper E. Natural Language Processing with Python. - Sebastopol: O’Reilly Media, 2009.
8. Маннинг К., Рагхаван П., Шютце Г. Введение в информационный поиск. - М.: Вильямс, 2011. - 520 с.

9. Мэннинг К., Шютце Г. Основы статистической обработки естественного языка. - М.: Мир, 2002. - 986 с.
10. Япарова Н. М., Низамов Е. Р. Создание нейросетевой модели структуризации и формализации технического задания // Информационные технологии интеллектуальной поддержки принятия решений. - 2022.
11. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. - Hoboken: Wiley, 2011.
12. Позин Б. А. Тестирование в жизненном цикле автоматизированных систем // Программная инженерия. - 2010. - № 1.

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.
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.
3. GOST 34.003-90. Informatsionnaya tekhnologiya. Kompleks standartov na avtomatizirovannye sistemy. Avtomatizirovannye sistemy. Terminy i opredeleniya [Information technology. Automated systems. Terms and definitions]. Moscow: Izdatelstvo standartov; 1990. 26 p.
4. 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.

5. Bakulin I. A., Bychkova N. A., Chervyakov L. M. Modelirovanie tekhnicheskikh trebovaniy k programmnym produktam [Modeling technical requirements for software products]. Sovremennye naukoemkie tekhnologii. 2021;
6. Wiegers K., Beatty J. Software Requirements. 3rd ed. Redmond: Microsoft Press; 2013.
7. Bird S., Klein E., Loper E. Natural Language Processing with Python. Sebastopol: O'Reilly Media; 2009.
8. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press; 2008.
9. Manning C. D., Schütze H. Foundations of Statistical Natural Language Processing. Cambridge: MIT Press; 1999.
10. Yaparova N. M., Nizamov E. R. Sozdanie neyrosetevoy modeli strukturizatsii i formalizatsii tekhnicheskogo zadaniya [Development of a neural network model for structuring and formalizing a technical specification]. Informatsionnye tekhnologii intellektualnoy podderzhki prinyatiya resheniy. 2022.
11. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken: Wiley; 2011.
12. Pozin B. A. Testirovanie v zhiznennom tsikle avtomatizirovannykh sistem [Testing in the life cycle of automated systems]. Programmnaya inzheneriya. 2010;