УДК 004.9

ПЕРЕХОД ОТ DOCX К DOCS-AS-CODE: ГЛУБИННАЯ ТРАНСФОРМАЦИЯ ПРОЦЕССОВ СОЗДАНИЯ ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ

О.В.Перфильев, И.Н.Корепин

Общество с ограниченной ответственностью «Открытые решения»,

г.Пенза, Россия

TRANSITION FROM DOCX TO DOCS-AS-CODE: A DEEP TRANSFORMATION OF TECHNICAL DOCUMENTATION PROCESSES

Perfiliev O.V., Korepin I.N.

Open Solutions LLC, Penza, Russia

Аннотация: В статье представлена реализованная архитектура технической документации как инженерного продукта на основе методологии Docs-as-Code. Экспериментально показано, что внедрение подхода сокращает время подготовки документации на 35–45% по сравнению с традиционным DOCX-ориентированным процессом.

Ключевые слова: Docs-as-Code; техническая документация; AsciiDoc; CI/CD; автоматизация; Pandoc; модульная архитектура; сниппеты; метрики эффективности.

Abstract: The article presents an implemented architecture of technical documentation as an engineering product based on the Docs-as-Code methodology. Experimental results demonstrate that the proposed approach reduces document preparation time by 35–45% compared to traditional DOCX-based workflows.

Keywords: Docs-as-Code; technical documentation; AsciiDoc; CI/CD; automation; Pandoc; modular architecture; snippets; efficiency metrics.

В эпоху цифровой трансформации и ускоренного жизненного цикла программного обеспечения традиционные подходы к созданию технической документации, основанные на ручном редактировании файлов в форматах DOCX, становятся серьёзным узким местом. Данные подходы характеризуются высокой трудоёмкостью, подвержены человеческим ошибкам (форматирование, битые ссылки, несогласованность стилей), и слабой интеграцией с современными процессами разработки, что приводит к отставанию документации от кода на недели и более [1, 2]. В условиях, когда документация является неотъемлемой частью продукта и критически важна для пользователей, поддержки и дальнейшей разработки, возникает острая необходимость в переходе к более эффективным, автоматизированным и управляемым методологиям. Одним из перспективных решений является подход Docs-as-Code (DaC), который рассматривает документацию как инженерный артефакт,

подчиняющийся тем же принципам контроля версий, автоматизированной сборки и непрерывной интеграции, что и исходный код [3]. Данная статья представляет собой стадии внедрения методологии DaC в проекте, демонстрируя её практическую применимость и измеримые выгоды.

Цель и задачи статьи

Целью работы является разработка и оценка архитектуры технической документации, основанной на методологии DaC, с количественным сравнением её эффективности по сравнению с традиционным подходом на основе DOCX.

Для достижения цели решались следующие задачи:

1. Миграция существующей базы документов из формата DOCX в текстовую разметку AsciiDoc.
2. Автоматизация процессов сборки, валидации и публикации документации в рамках CI/CD-пайплайна.
3. Проведение сравнительного анализа ключевых метрик (время подготовки) до и после внедрения новой архитектуры.

Исходные инструменты и технологии

Для перехода к языку разметки использовались следующие инструменты:

- Язык разметки и редакторы:
  - Asciidoc (.adoc) — основной язык разметки для документации;
  - VS Code или VS Codium — среда разработки для редактирования и генерации документации из исходных файлов разметки.
- Плагины для VS Code / VS Codium (устанавливаются сразу после запуска среды):
  - Asciidoc (структура, предварительный просмотр, подсветка синтаксиса);
  - Russian Language Pack;
  - Microsoft Edge Tools;
  - Git Extension Pack (donjayamanne);
  - vsCode-pdf (редактирование и генерация PDF).
- Конвертация и системы контроля версий:
  - Pandoc —универсальный конвертер документов для преобразования файлов между различными форматами [4];
  - GitLab —система контроля версий для управления изменениями в исходных файлах документации [5].
- Скрипты и автоматизация:
  - Python — скрипты для автоматизации задач, например: построение иерархической структуры файлов документации; автоматическое создание главного файла;
  - Ruby — язык программирования, поддерживающий отдельные процессы автоматизации документации [6].

Конвертация исходных документов через Pandoc

Исходные файлы в формате DOCX конвертируются в AsciiDoc с помощью утилиты Pandoc:

bash

```
pandoc input.docx -f docx -t asciidoc -o output.adoc
```

После конвертации выполняется создание структуры документации с помощью скриптов на языке Python и её последующая проверка.

Формирование структуры документации:

- документ автоматически разбивается на логические модули или главы и приложения, которые помещаются в определённые папки (например, 1.chapter.adoc, 2.chapter.adoc);
- автоматически с помощью скрипта создаётся главный файл (main.adoc), который собирает все модули воедино с помощью директивы include::.

Ниже приведён фрагмент кода (листинг), демонстрирующий базовый подход к созданию структуры документации с главами, приложениями и главным файлом. Его можно расширять и адаптировать под конкретные проекты.

```
        # Создание структуры документации
        try:
            self.generate_structure(root_dir, product_name, doc_dir, chapters_count, appendix_count, create_appendix)
            self.show_message("Успех", "Структура документации успешно создана!")
        except Exception as e:
            self.show_message("Ошибка", f"Произошла ошибка: {str(e)}")

    def generate_structure(self, root_dir, product_name, doc_dir, chapters_count, appendix_count=0, create_appendix=F
        """
        Генерирует структуру документации.

        :param root_dir: Название главной директории.
        :param product_name: Название продукта.
        :param doc_dir: Директория для документации.
        :param chapters_count: Количество глав.
        :param appendix_count: Количество приложений.
        :param create_appendix: Флаг для создания приложений.
        """
        base_path = os.path.join(doc_dir, root_dir, product_name, "documentation")
        os.makedirs(base_path, exist_ok=True)

        # Создание основных директорий
        main_dirs = [
            os.path.join(base_path, "chapters"),
            os.path.join(base_path, "media")
        ]
        for dir_path in main_dirs:
            os.makedirs(dir_path, exist_ok=True)

        # Создание поддиректорий для медиа контента по номерам глав
        for i in range(1, chapters_count + 1):
            chapter_media_dir = os.path.join(base_path, "media", f"{i}.chapter")
            os.makedirs(chapter_media_dir, exist_ok=True)

            # Создание файла .gitkeep для пустых папок
            gitkeep_path = os.path.join(chapter_media_dir, ".gitkeep")
            with open(gitkeep_path, "w") as gitkeep_file:
                pass  # Файл создается пустым

        # Создание файлов глав
        for i in range(1, chapters_count + 1):
```

```
        # Создание файлов глав
        for i in range(1, chapters_count + 1):
            chapter_file_path = os.path.join(base_path, "chapters", f"{i}.chapter.adoc")
            with open(chapter_file_path, "w", encoding="utf-8") as chapter_file:
                chapter_file.write("")  # Создаем пустые файлы для глав

        # Создание приложений, если выбрано
        if create_appendix and appendix_count > 0:
            appendix_dir = os.path.join(base_path, "appendix")
            os.makedirs(appendix_dir, exist_ok=True)

            # Создание файлов приложений
            for i in range(1, appendix_count + 1):
                appendix_file_path = os.path.join(appendix_dir, f"{i}.appendix.adoc")
                with open(appendix_file_path, "w", encoding="utf-8") as appendix_file:
                    appendix_file.write(f"== Приложение {i}\n\n")

            # Создание карты приложений appendix.adoc
            appendix_map_path = os.path.join(base_path, "appendix.adoc")
            with open(appendix_map_path, "w", encoding="utf-8") as appendix_map:
                appendix_map.write(":appendix-number: @\n:sectnums:\n:toc:\n\n")
                appendix_map.write("== Приложения\n\n")
                for i in range(1, appendix_count + 1):
                    appendix_map.write(f".<<Приложение_{i}>>\n")
                    appendix_map.write(f"include::{i}.appendix.adoc[]\n")

        # Создание основного файла документации main.adoc
        main_adoc_path = os.path.join(base_path, "main.adoc")
        with open(main_adoc_path, "w", encoding="utf-8") as main_adoc:
            main_adoc.write(":sectnums:\n")
            main_adoc.write(":toc:\n\n")
            main_adoc.write("= Руководство пользователя\n\n")
            main_adoc.write("== Перечень включенных в документ глав\n\n")

            # Добавление include для каждой главы
            for i in range(1, chapters_count + 1):
                main_adoc.write(f"include::{{chapter-dir}}/{i}.chapter.adoc[]\n")

            # Если выбрано создание приложений
            if create_appendix and appendix_count > 0:
                main_adoc.write("\n== Приложения\n\n")
                main_adoc.write("include::appendix.adoc[]\n")

    def show_message(self, title, message):
```

Структура проекта

Репозиторий организован по модульному принципу, где каждый файл .adoc представляет собой независимый блок контента. Пример модульной структуры документа представлен на рис. 1

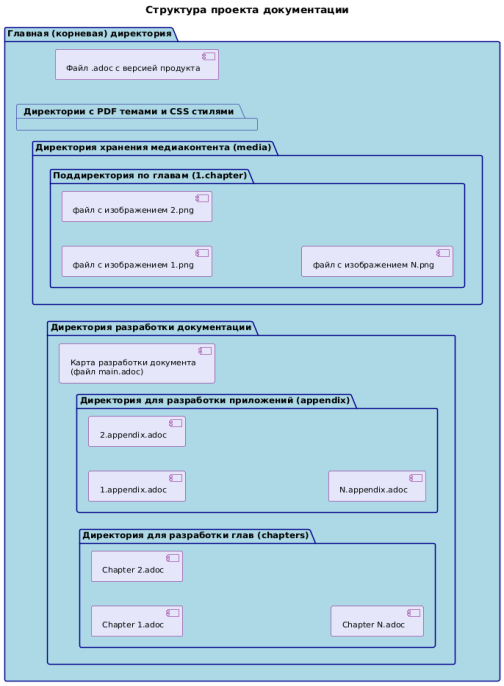


Рисунок 1 — Пример модульной структуры документа

Внедрение и автоматизация процессов подготовки документации с использованием CI/CD

Внедрение CI/CD в методологию DaC обеспечивает автоматизацию этапов подготовки технической документации, включая валидацию синтаксиса и структуры AsciiDoc, сборку итоговых форматов, проверку целостности ссылок. Интеграция с системами контроля версий позволяет организовать управление изменениями и ревью, что снижает вероятность ошибок.

Результаты и метрики эффективности

На рис. 2 показано сокращение временных затрат на подготовку документов разных типов.

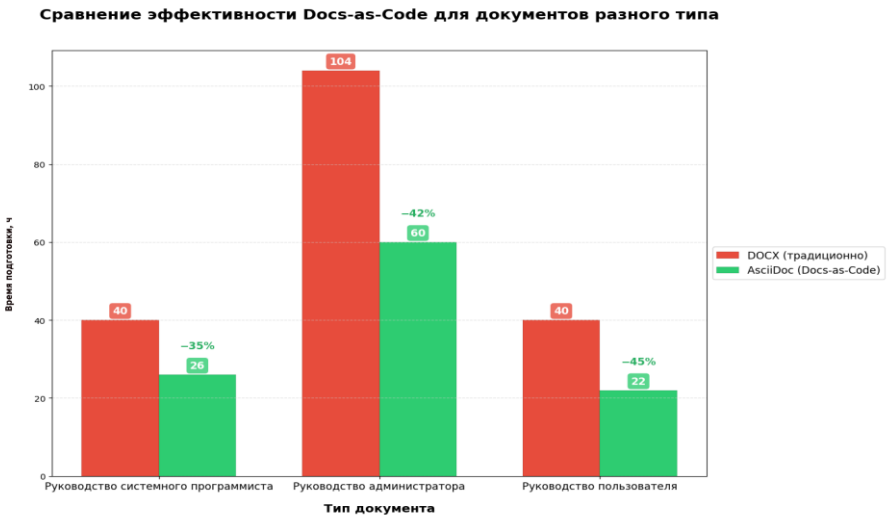


Рисунок 2 — Сравнение эффективности до и после внедрения DaC

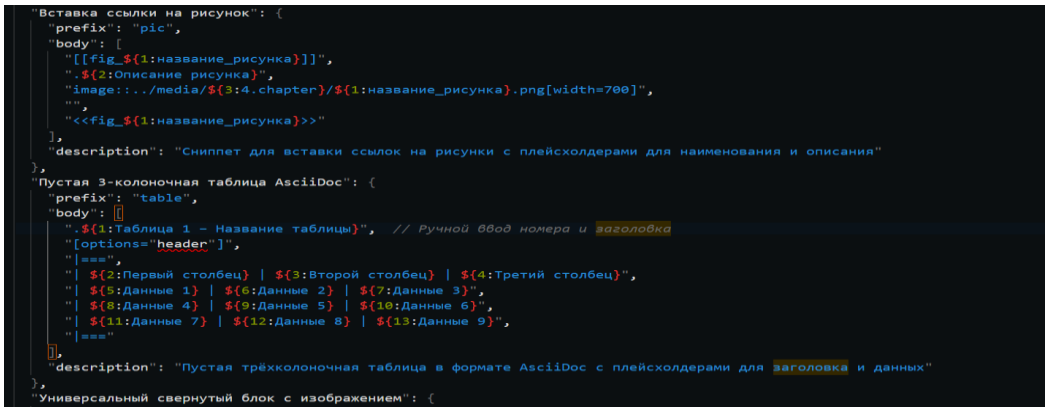
Результаты показывают, что внедрение DaC обеспечивает значительную экономию времени на создание технической документации:

- для руководства администратора объёмом 130 стр. экономия составила 42%;
- для руководства пользователя объёмом 50 стр. экономия составила 45%;
- для системного руководства объёмом 50 стр. экономия составила 35%.

Использование сниппетов

Сниппеты[1] отдельно создаются в файле .json, хранятся в папке /snippets/, вызываются с помощью команды (Ctrl+Shift+P) в текстовом редакторе кода VS Code или просто введя короткий уникальный фрагмент — префикс сниппета. После этого шаблон автоматически развернётся в полный блок текста или кода с возможностью сразу заполнить нужные поля.

Ниже представлен фрагмент сниппета в формате . json.



Преимущества сниппетов:

- скорость;
- удобство редактирования;
- меньшее число ошибок.

Заключение

Переход от DOCX к DaC с использованием стека AsciiDoc и GitLab CI/CD представляет собой не просто смену инструментов, а фундаментальную трансформацию культуры и процессов создания технической документации. Эксперимент, проведённый в рамках проекта, подтвердил значительные преимущества данного подхода. Внедрение подхода позволило сократить время подготовки документов на 35-45% в зависимости от типа и объёма документов.

Методология DaC обеспечивает качественные улучшения: документация становится частью кодовой базы, что гарантирует её актуальность и версионность; модульная архитектура упрощает управление и переиспользование контента; автоматизация освобождает технических

[1] Заранее определённые фрагменты кода (шаблоны), которые могут быть быстро вставлены в редактор кода путём набора нескольких букв

писателей от рутинных задач, позволяя им сосредоточиться на содержании. Опыт, описанный в данной статье, а также лучшие практики, заимствованные у лидеров отрасли, таких как SITRONICS Group [1] и Alfa-Bank [2], показывают, что DaC является зрелой и эффективной методологией, применимой в любых организациях, где качество и своевременность документации являются критически важными факторами успеха. Дальнейшие исследования могут быть направлены на интеграцию DaC с системами управления знаниями и расширение автоматизированной валидации за счёт применения методов искусственного интеллекта для анализа семантической полноты и согласованности документации.

Список литературы

1. SITRONICS Group. Опыт внедрения AsciiDoc [Электронный ресурс] // Хабр. – 2021. – URL: https://habr.com/ru/companies/sitronics_group/articles/654355/ (дата обращения: 15.09.2025).
2. Alfa-Bank. Документация как код [Электронный ресурс] // Хабр. – 2022. – URL: https://habr.com/ru/company/alfa/articles/757872/ (дата обращения: 15.09.2025).
3. Автоматизация процесса создания технической документации на основе подхода Docs-as-Code [Электронный ресурс] // Информационные технологии. – 2023. – № 5. – URL: https://na-journal.ru/5-2023-informacionnye-tekhnologii/5036/ (дата обращения: 15.09.2025).
4. Pandoc. Universal Document Converter [Электронный ресурс]. – URL: https://pandoc.org/ (дата обращения: 15.09.2025).
5. GitLab — система контроля версий для управления изменениями в исходных файлах документации [Электронный ресурс]. – URL: https://about.gitlab.com/ (дата обращения: 01.10.2025).
6. Ruby — язык программирования, поддерживающий отдельные процессы автоматизации документации [Электронный ресурс]. – URL: https://www.ruby-lang.org/ (дата обращения: 01.10.2025).