**Коробицин Сергей**
*Руководитель команды разработки платформы*
*ARTA Software, Астана, Казахстан*

# О ПОСТРОЕНИИ СИСТЕМ РАЗРАБОТКИ И СОПРОВОЖДЕНИЯ ДОКУМЕНТАЦИИ ПРОЕКТА

**Введение.** Документирование — важнейшая жизненного цикла любого проекта. От того, каким образом будет устроен процесс создания, внесения изменений, хранения и публикации документации, зависит достижение целей проекта, а также затраченные на документирование ресурсы.

В этой статье мы сформулируем основные принципы построения системы разработки и сопровождения документации проекта и его продуктов, а также приведем практический пример построения такой системы.

**Требования к системе разработки и сопровождения документации.**

Любой комплект документации проекта можно разрабатывать различными способами, например, используя «традиционный» подход — с использованием текстовых процессоров (т. е. программных продуктов «Microsoft Office», «LibreOffice» и аналогичных). Однако с ростом объема требуемой документации, её структурной сложности: количества составных частей, перекрёстных ссылок и т. д., а также количества работающих над документацией людей и объема вносимых изменений в единицу времени управление этим процессом и поддержание требуемого уровня качества становится всё сложнее. Кроме этого, некоторые необходимые возможности сложно или невозможно реализовать, используя только вышеуказанные инструменты (например, полнотекстовый по всему комплекту документации), а для уменьшения требуемых на разработку документации человеческих ресурсов следует подвергнуть автоматизации как можно большее количество необходимых для неё действий.

Сформулируем требования к системе разработки и сопровождения документации:

1. Применение «принципа одного источника» [1] при построении системы
2. Отделение логической структуры документации (в том числе разметки) от её физической структуры

3. Возможность задания произвольного (проект- или контекст-специфичного) способа разбиения комплекта документации на модули

4. Возможность применения любого необходимого жизненного цикла разработки документации

5. Возможность рецензирования документации участниками проекта в процессе и по завершении одной или нескольких фаз её разработки

6. Возможность получения различных выходных форматов для публикации — например, печатный формат (PDF), формат для публикации в WWW (HTML), контекстная справка программного продукта (CHM, HTML, другие форматы), «электронные книги» (FB2, ePUB) — и унификация их оформления

7. Возможность построения любой необходимой навигации и поиска по документации

8. Возможность локализации [2] документации на любой наперед неизвестный язык

9. Возможность интеграции с производственной частью проекта (например, в случае проекта по разработке программного продукта — включение автоматически сгенерированной по исходному коду документации)

10. Возможность профилирования выходной документации по различным принципам (например, пользовательская документация по программному продукту для различной аудитории — непосредственно пользователей системы и специалистов по её технической поддержке).

**Пример системы разработки и сопровождения документации в проекте по разработке программного продукта**

Для соответствия принципу «единого источника» (1) положим в основу нашей системы одну из широко применяемых для разработки программного обеспечения систему контроля версий (СКВ) — Subversion [3] или Git [4] (предпочтительно). Если жизненный цикл разработки документации проекта совпадает с жизненным циклом создаваемого в проекте программного продукта, то можно использовать тот же самый репозиторий, в котором лежат его исходные коды (если таковой имеется). В противном случае необходимо выделить под документацию нашего проекта отдельный репозиторий.

Для соответствия требованиям (2), (3) и (5) требуется выбрать удобный формат исходного текста документации. На данный момент имеется две альтернативы:

Семейство простых текстовых форматов разметки:

- Markdown [5], reStructuredText [6], AsciiDoc [7] и другие
- Формат DocBook XML (версии 4.5 или 5.0)

Мы останавливаем выбор на формате Markdown ввиду следующих его преимуществ:

- Высокая распространённость и поддержка многими программными системами
- Легкий способ расширения и аннотирования («подсказок») для выходных форматов (формат допускает включение конструкций TeX и HTML)

Для задания модульной структуры (3) будем использовать универсальный предобработчик gpp [8] — и стоить полный документ, начиная с индексного файла с подключением других файлов, которые, в свою очередь, тоже могут быть разбиты аналогичным образом.

Пример индексного файла при использовании формата Markdown и GPP:

```
---
author:
- Сергей Коробицин
date: 30.11.2016
title: Спецификация
copyright: Для внутреннего использования
...

<#include "reqs.md">
<#include "glossary.md">
<#include "configurator/index.md">
```

Система контроля версий не накладывает никаких ограничений на жизненный цикл (4) разработки её содержимого. В зависимости от потребностей мы можем использовать как широко распространённые git-flow [9], GitHub Flow [10], так и разработать собственный процесс.

Для рецензирования документации ввиду использования формата Markdown возможно использовать любой инструмент рецензирования исходного кода, например Review Boad [11], Atlassian Crucible [12], либо любую другую, подходящую по возможностям и стоимости.

Для получения разнообразных выходных форматов мы будем организовывать цепочки преобразований для получения необходимого нам формата. Первым элементом этой цепочки будет предобработчик gpp (см. выше), а вторым — «универсальный преобразователь документов» Pandoc [13]. Далее, в зависимости от конкретного выходного формата могут быть подключены другие преобразователи (если в Pandoc нет необходимой функциональности).

Примеры цепочек преобразований:

- HTML: gpp index.md → pandoc full.md → xmlto full.xml → HTML
- PDF: gpp index.md → pandoc full.md (→ xelatex full.tex) → PDF

(части, заключенные в скобки, Pandoc выполняет самостоятельно).

Построение навигации (7) по документации зависит от основного места её публикации. Если в качестве последнего использовать web-сайт, то для организации навигации необходимо:

- Задать структуру публикации выходной документации (например site/html и site/pdf)
- Создать соответствующие индексные файлы (в формате исходной документации или напрямую в HTML) с относительными ссылками на выходные артефакты

Для поиска по документации мы предлагаем задействовать поисковые системы Интернет (Google, Яндекс и другие), либо использовать любую удобную «отдельностоящую» систему полнотекстового поиска, если документация не размещается в Интернете в открытом доступе.

Для локализации (8) документации предлагается использовать систему GNU Gettext [14] и инфраструктуру вокруг неё (Translate Toolkit [15], Weblate [16] и другие). Для конвертации исходного текста нашей документации в POT/PO предлагаем использовать инструментарий po4a [17].

Интеграцию (9) с производственной частью необходимо рассматривать, отталкиваясь от выходных форматов генерируемой документации. В общем случае можно публиковать сгенерированную документацию на том же веб-сайте, что и выходную (например site/generated/html) и связывать её с основной документацией при помощи относительных гиперссылок.

И, наконец, профилирование (10) можно осуществить с помощью универсального предобработчика gpp, используя несколько индексных файлов, включающих разный набор документации.

## Литература


1. Разработка технической документации с помощью DocBook/XML. Принцип единого источника. Обзор. Михаил Острогорский, http://www.uml2.ru/forum/index.php?action=dlattach;topic=763.0;attach=1525
2. Локализация по сравнению с интернационализацией. Richard Ishida, W3C, Susan K. Miller, Boeing. https://www.w3.org/International/questions/qa-i18n.ru.php
3. Apache™ Subversion®, https://subversion.apache.org/
4. Git, https://git-scm.com
5. Формат Markdown. Джон Грубер, https://daringfireball.net/projects/markdown/
6. reStructuredText, Markup Syntax and Parser Component of Docutils, http://docutils.sourceforge.net/rst.html
7. AsciiDoc, Text based document generation, http://asciidoc.org/
8. GPP, general-purpose preprocessor, https://logological.org/gpp
9. A successful Git branching model, Vincent Driessen, http://nvie.com/posts/a-successful-git-branching-model/
10. GitHub Flow. Scott Chacon, http://scottchacon.com/2011/08/31/github-flow.html
11. https://www.reviewboard.org/
12. https://ru.atlassian.com/software/crucible
13. Pandoc, a universal document converter, https://pandoc.org/
14. https://www.gnu.org/software/gettext/
15. http://toolkit.translatehouse.org/
16. https://weblate.org/ru/