Combining contributions from multiple sources

By Joaquim Baptista.

Joaquim reflects how small technical communication teams can use flexible tools and workflows to combine contributions from other authors into meaningful publications.

When I worked at Altitude Software circa 2000, a small team of communicators was responsible for the technical documentation of a sophisticated suite of software. The team wrote or rewrote all the words seen by customers and partners. The team also took screenshots and created diagrams to illustrate the text.

In recent years, companies have pushed us to work as enablers, coaching a larger number of engineers and other professionals into writing their user documentation. For example, in 2020 Farfetch decided not to try to hire the team of 40 or so people that might sustain a traditional technical communication workflow, because it would be difficult to hire so many skilled technical communicators in Portugal. Instead, five technical communicators were expected to combine technology and workflows to curate the writing efforts of thousands of engineers.

We have thus been forced into different ways of working. Whatever authoring we do is strategic. For instance, we may write if necessary, to provide an example, or to intervene in critical situations, or to build bridges between teams.

Multiple sources and formats

A traditional technical communication team typically maintains a single repository of official documentation, affectionately called the "docset".

When working with hundreds of teams and thousands of engineers, such simplicity is harder to achieve. The sheer variation between teams implies variations in needs, tools, and workflows. Any "one system" would be hard pressed to fulfill the needs of every team in a satisfactory way. Whilst ideally some spurious variation is best avoided:

- Specific needs drive engineers to adopt several tools and documentation formats.
- Convenience, governance, and historic reasons drive engineers to store documents in multiple repositories.

As enablers, it is easier for us to build upon existing practices than to force thousands of engineers to adopt something built from scratch. Ideally, we want to amplify the good writing that is already being done, using the tools and workflows that are either comfortable to engineers, or perhaps demanded by the company.

For example, Farfetch documentation formats included source code in C#, API specifications and infrastructure blueprints in YAML, and documentation topics in Markdown and DITA XML. These source files were scattered across Git repositories.

Continuous evolution

Authoring workflows may be forced to evolve quickly, to adapt to shifts of responsibility within the company. When Farfetch sought to create public APIs to enable developers working for partners to directly use some of the functionality created by Farfetch developers, the evolution of these APIs shows how shifting responsibilities have imposed changes to tools and workflows:

The public APIs started with a small central team that created half a dozen customized API gateways in C#. These gateways invoked functionality implemented by product teams in internal services. We collaborated with engineers to add or improve the descriptions embedded within the C# source files.

Two years later, Farfetch decentralized the evolution of the customized API gateways to product teams. Each product team became responsible for maintaining and evolving as necessary, the part of the public APIs related to their functionality. Furthermore, those APIs gained a specification (in OpenAPI, a text-based YAML format that combines structured data with longer textual descriptions) that supported the negotiation of changes to APIs before implementation. We became part of the negotiation, ensuring that APIs could easily be used by partners, adding descriptions to YAML files as documentation.

Another two years later, Farfetch replaced the customized C# code with a generic solution. Product teams could now promote to a public API any operation in their internal services, as specified in the Open API specification (YAML files) of each internal service. We started to edit descriptions of public API operations across hundreds of YAML files, and we curated those operations into a coherent whole.

Each of these imposed evolutions was an opportunity to seek ways to collaborate effectively with engineers. For example, we argued that the hundreds of API specifications for internal services should be kept in a monorepo (a single repository used by all teams) instead of in hundreds of separate Git repositories, enabling us to easily edit multiple APIs at the same time. In a specific situation, we led the adoption of OpenAPI extensions by adding explicit default values to each operation, which served as a useful reminder to thousands of developers.

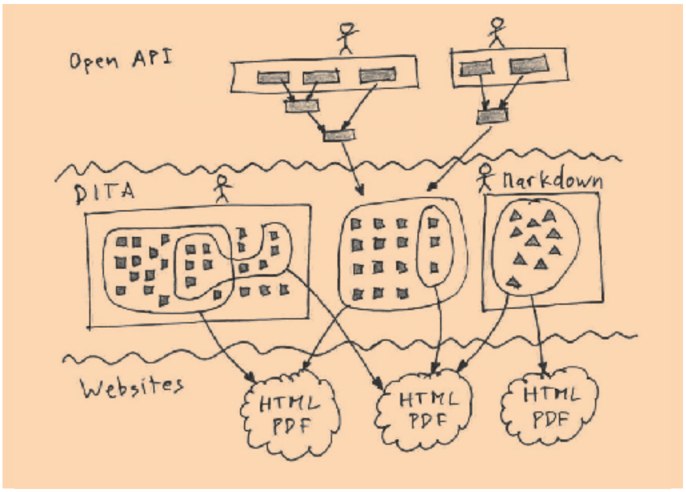
Multiple audiences

While some documentation repositories may focus on a single audience, teams often produce documentation both for themselves and for multiple audiences.

A specific audience may need documentation that is scattered across different repositories. Therefore, we must consolidate that documentation into published documentation. We must also implement a navigation structure that makes sense to that specific audience.

The diagram below shows how multiple sources can contribute to the creation of websites to three different audiences.

- Small filled triangles, squares, and rectangles represent topics and documents.
- Rectangles with persons represent repositories of source files.
- Round bubbles represent subsets of topics and documents.
- Clouds represent websites for audiences.
- Arrows represent transformations.



At Farfetch, each audience had its own top-level navigation structure, and its own way to group API operations. We were responsible for the curation of these navigations and groupings.

The documentation aimed at each audience may be specified through different approaches:

- Having a "table of contents" that enumerates and organizes the documentation for an audience.
- Having conventions that identify topics for an audience. For example, documents in a specific folder or documents with specific filename conventions.
- Having metadata that identifies the audiences of specific documents.

At Farfetch, for example, we promoted OpenAPI extensions to specify the audience (such as team, internal, and external) and maturity (such as draft, limited, production, and obsolete) of each API operation. Public operations in production would be published for external partners, while other operations would be published for different internal audiences.

Integration and flexibility

The typical technical implementation for a multi-source system propagates the structure of the sources to HTML silos, for example as React modules that designers might combine into a single HTML page. This kind of siloed implementation has two bad consequences:

1. The structure of the HTML output reflects the internal structure of the sources, which may be a hindrance to the navigation for some audiences.
2. The siloed HTML limits the ability to link between different silos. For example, we might want bidirectional links between API tutorials and their API references, because references add detail to tutorials, while tutorials serve as inspiration to references.

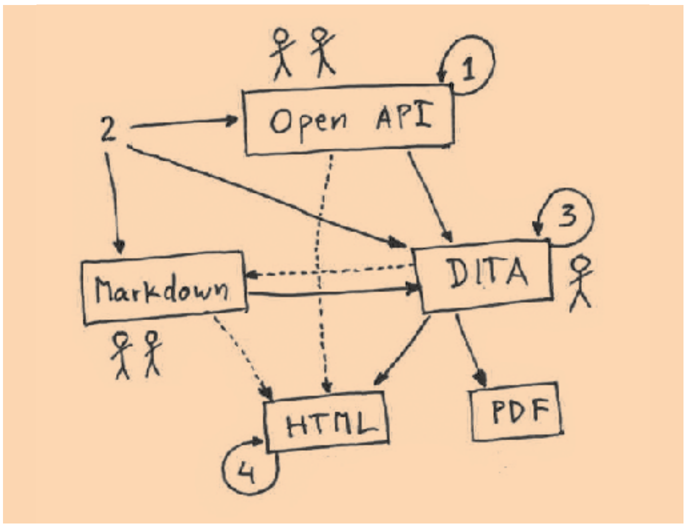
One possible alternative is to convert different input formats to a common intermediate format, such as DITA XML. Having all the documentation available in a common format has the following advantages:

- Multiple output formats, such as PDF and ePub, are useful in specific scenarios.
- Custom one-off outputs, for example to collect the requirements of all tutorials to facilitate editing.
- Manual experiments to explore solutions to new company needs, before automation is put into place. For example, to explore different navigation structures or how to address a new audience.

At Farfetch, the intermediate integration formats were Open API (YAML) and DITA XML. Although the conversion between formats risks losing information, in practice the thousands of engineers at Farfetch contributed mostly reference documentation, often in structured YAML formats (OpenAPI and blueprints), that made the conversions straightforward.

The diagram below represents the ecosystem of documentation formats at Farfetch:

- Arrows represent conversions between formats, although conversions represented by dotted arrows were not part of the publication flow.
- People indicate authors writing Open API specifications, Markdown topics, and DITA topics.

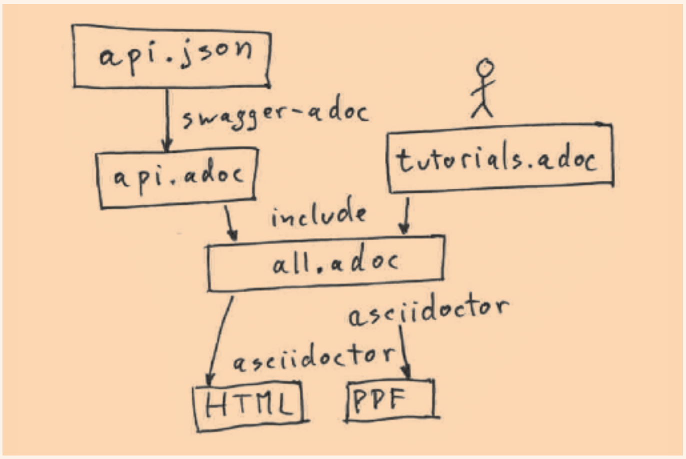


Numbers indicate the following transformations:

1. Filter and consolidate information from services APIs into APIs for audiences.
2. Convert information to Open API extensions, DITA topics, or Markdown topics.
3. Process DITA topics to create topic variants, for example to have better PDF output.
4. Use metadata to link topics together, after adopting modular DITA builds for performance reasons For example, if a tutorial uses an API operation, automatically link the tutorial to the operation and the operation to the tutorial.

The reliability of the transformations depends on well-defined text formats, such as JSON, XML, and HTML. To keep with the spirit of docs-as-code, Farfetch initially tried to use AsciiDoc as an intermediate format. AsciiDoc is a plain-text format that describes the book structure of DocBook, used by O'Reilly authors. In particular, the open source editor AsciiDocFX provides a convenient way to write large documents with integrated diagrams and generate the usual output formats such as HTML, ePub, and PDF.

However, Farfetch ultimately abandoned AsciiDoc as an intermediate format. While AsciiDoc proved to be convenient for authoring, it was too easy to run into edge cases when generating documents with code samples and index entries.



Governance

The publication process may be triggered manually or automatically, for example when sources change. The publication process may also be fully automated, requiring no human interaction, or it may incorporate quality controls and approval steps. For example, product people may need to approve changes to product documentation.

The publication process at Farfetch started as a lengthy manual procedure executed on a monthly cadence, deployed to static websites complemented with authorization and server-side search. Over time, we automated the whole process towards daily automatic publishing, while maintaining the ability to fix or improve documentation when needed. For example, we delayed editing APIs with less business value until we detected large changes or until the API had accumulated a number of minor changes.

To provide feedback, internal audiences need the ability to quickly navigate to the source of published documentation. This enables readers (or the company support team) to interact with engineers and to propose changes to a published document. For example, a button in each HTML page that takes the reader to the Git history of the source document.

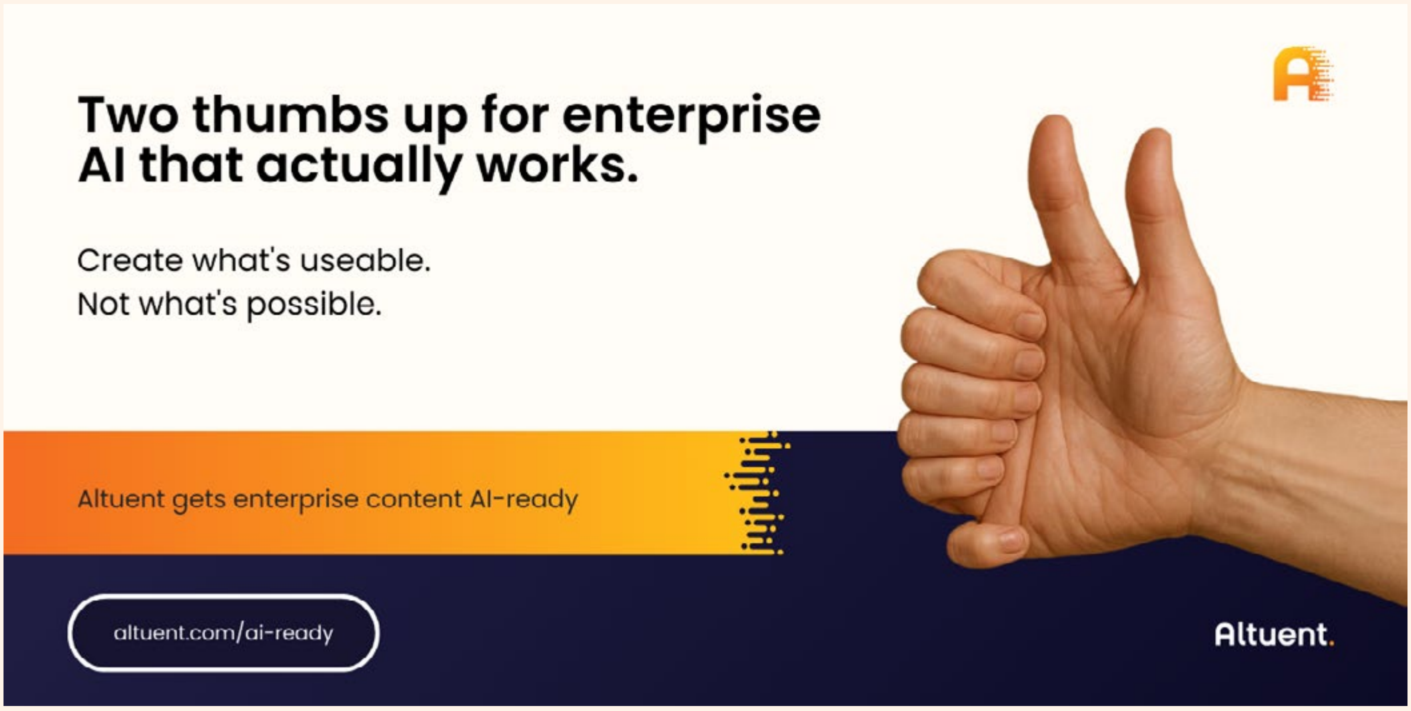
Recommendations for enablers

We know that any technical writing job quickly grows into more than “just writing”, to the point where writing often becomes a minor part of the overall effort. If you find yourself in the role of enabler, you may want to:

- Learn what needs to be communicated and why, to focus on outcomes.
- Learn current practices, to build upon them.
- Write and edit strategic projects, to demonstrate what could be.
- Develop a compelling technical communication vision, to evangelize to stakeholders.
- Distill best practices into guidelines, to coach teams and engineers.
- Negotiate effective collaboration workflows with teams, supported by effective tools and automation, to maximize your impact.
- Curate what is published, especially the navigation structures, to care for the needs of each audience.
- Work as a commissioning editor to identify and elicit missing content.
- Work as a facilitator to drive difficult conversations. ■

Joaquim Baptista

Joaquim is the Principal Technical Writer at Millennium bcp. Joaquim has documented large and evolving software products that require industrial writing instead of just writing craftsmanship since 1997. Facilitation is an occasional need. Joaquim is a member of IEEE PCS and a leader at ISTC, APCOMTEC, and EuroSIGDOC.



Two thumbs up for enterprise AI that actually works.
Create what's useable.
Not what's possible.
Altuent gets enterprise content AI-ready
altuent.com/ai-ready
Altuent.