Best Practice: Using DITA with DITAworks in software documentation projects

Software documentation has specific challenges which must be tackled by an effective documentation process. Software release cycles have shortened due to agile processes. The main challenge in documentation is to keep the documentation up-to-date and synchronized with these shorter release cycles. This challenge requires an effective documentation process that is tightly integrated with the software development process.

In this article, I describe a practical case study of how DITA, in a DITAworks CMS environment, is used at Fujitsu Enabling Software Technology GmbH. I discuss how the software documentation projects are implemented and how a tight integration with an agile software development process is achieved.

Why do we use DITA? Before implementing DITA, our documentation team used a variety of tools for creating and maintaining the documentation. Our documentation landscape was quite heterogeneous, and it became more and more costly and time-consuming to maintain the documentation projects.

Today, we use DITA in all our documentation projects within a homogeneous authoring environment, using DITAworks which is a DITA-based CMS.

There are many good reasons for using DITA. The main reason and starting point for us was a large documentation project in which we needed a great number of manual variants based on different products, different target audiences, and different system requirements. See Figure 1 for an example of output variants.

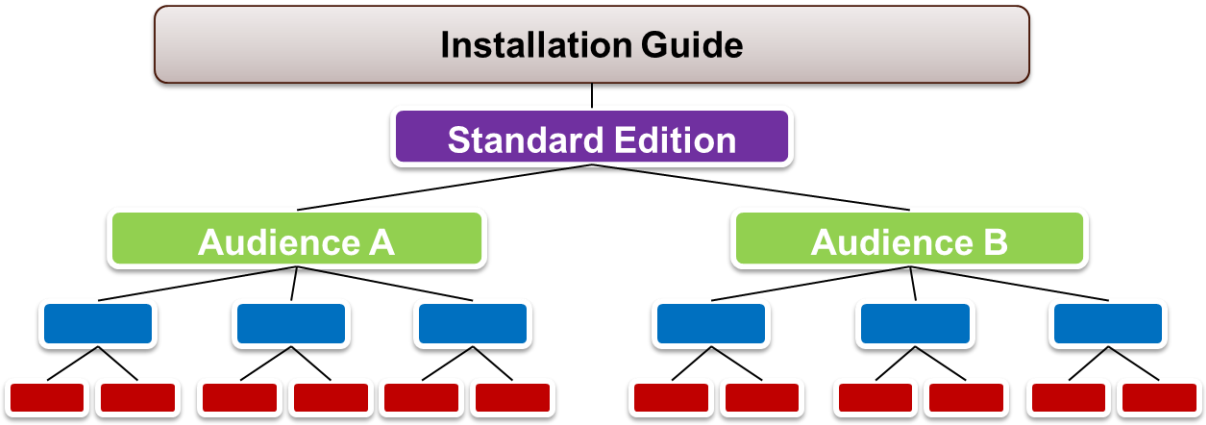

Figure 1. Example of output variants

For this project, we had to produce the installation guide for the standard edition with one set of target audiences and three different combinations of operating systems, databases, and application servers. Each variant had to be published in PDF and HTML. This combination resulted in six different output variants. If you had to do this in a traditional word processing system, it would be very cumbersome. A practical solution was urgently required.

DITA, with its modular content concept and the possibility of conditional processing, was the best solution to meet these requirements. The project was started in 2006 with the DITA Open Toolkit version 1.1 and XMetal as the editor. Great effort was necessary at the beginning, especially for adapting the PDF output to the FUJITSU guidelines. This adaption involved XSL programming, creating a style guide, and so on. That effort has already paid off many times since.

Why did we select DITAworks? After working with DITA Open Toolkit (DITA OT) as our authoring environment for a few years, additional requirements came up for which we needed a new solution.

In 2009, we came across DITAworks, which offered the features we needed. The deciding factors for choosing DITAworks were

- easy administration of large documentation projects and enhanced usage of DITA features, especially for re-using content
- easy integration of our documentation authoring and publishing processes with our Eclipse IDE-based software development processes
- support of the DITA Open Toolkit version 1.2.

Migration Phases

The migration to DITAworks was done in phases. The main reason for phased migration was that we had to do the migration in parallel with the regular documentation tasks within our product development.

- Phase1: Migrating the existing documentation to DITAworks, but still using pure DITA OT for publishing.
- Phase 2: Implementing various content concepts to benefit more strongly from structured information.
- Phase 3: Changing the publishing process from the DITA OT command line interface to DITAworks.
- Phase 4: Integrating the publishing process with our software build process to ensure that the current documentation is automatically included in the regular software builds.

Phase 1: Migrating Existing Projects to DITAworks

In the first phase, the main tasks were to adapt the existing XML content and to connect the XML sources to the DITAworks content model.

During this phase, we had to ensure that publishing still worked with the DITA Open Toolkit based on the DITA OT version 1.1. Last but not least, we had to get used to the new environment, which is of course always the case when implementing a new environment.

During this migration phase, we had to deal with several challenges:

- As pilot customers, we started with a beta version of DITAworks with the usual issues.
- We had to solve issues regarding the use of encoding and locales. For example, using the UTF-8 encoding caused some problems when publishing the German version.
- There were also layout requirements defined in our guidelines that needed extra effort.
- We had to clean up our sources such as making the use of elements consistent.

Phase 2. Implementing Content Concepts

In the second phase, we implemented some content concepts which we had defined to benefit more comprehensively from the main DITA concept, namely from re-using and sharing content. Our content concept comprises the following:

- We use a structural concept to manage topics. For example, we use a “Shared“ folder for commonly used topics and images.
- We use variables for text elements that frequently change. The use of variables has already saved us from wasting our time changing product names, for example, which occurs at least four times for each product we develop.

- We use conditional processing to produce documentation variants for different purposes, as previously mentioned.
- We extensively apply the DITA concepts for re-using content such as conref elements.
- We developed a rule for the use of topic references, using references only in one direction from so-to-say “master topics“ to all other topics.

Phase 3. Implementing the DITAworks Publishing Process

In phase three, we implemented the DITAworks publishing process. Previously, we had used the DITA OT for publishing our documentation. But in the course of time, the DITA OT had become quite uncomfortable and error-prone. Therefore, we wanted to switch to publishing our documentation with DITAworks.

In this phase, we had to perform the following steps:

- The stylesheets had to be adapted.
- The publishing scripts had to be adapted.
- The publishing had to be configured and tested.
-

We expected the implementation effort to be quite small, but the devil is in the details. In fact, we ran into unexpected problems, especially regarding the PDF layout. We needed several cycles for testing and bug fixing to address all issues that came up. Fortunately, we could finally solve the problems within an acceptable time frame.

Phase 4: Integrating the Documentation Process with the Development Process

As already pointed out, the integration with our development process was a very important piece of our decision.

Our software development procedure is an agile process, and we use the Scrum method. Usually, we have six or seven sprints of approximately one month each for developing new software features for a given product. This schedule results in release cycles of six months. The release cycles slightly overlap.

The characteristics of such a development process are as follows:

- The whole process is driven by product requirements.
- The process is based on continuous software builds.
- The sprint result should be a potentially shippable product.

Developing the documentation is rated as highly as developing or testing the software. The documentation process must be tightly integrated with the software process, and it must be ensured that the documentation is always deployed along with the software in each build.

Using the SCM System for Documentation

The development department uses an Eclipse-based SCM system for their development projects. The same SCM system is used to manage the documentation projects which results in the following advantages:

- Documentation and development can use the same SCM system.
- We have the same versioning system for source files (XML) and output results as we have for the software resources.

- We achieve a complete integration with our software configuration and release management.
- We can make use of the change management features, for example, by means of incidents, reports, history, or compare functions.

Since DITAworks is also Eclipse-based, we could smoothly integrate with the SCM system without leaving the authoring environment.

Integrating the Publishing Process with the Software Build Process

The last step was to integrate the publishing process with the software build process. This integration ensures that the documentation process fulfills the requirements of an agile process as well. The main tasks of the last step were

- installing the publishing server
- adapting the publishing configurations
- creating the build scripts for updating the workspace on the server with the current sources, launching the publishing for each document, and storing the output in an output folder
- including the documentation in the build scripts to store the output to the correct delivery packages
- testing the whole integration

Summary

In technical documentation, choosing a structured authoring solution is more effective. DITA-based solutions seem to be a great choice for software documentation. Although the DITA Open Toolkit provides the necessary environment, you will, in most cases, need and end up with a CMS that can meet additional requirements regarding administration and publishing.

In our case, the main advantages of the chosen solution are:

- We have full support of the DITA standard.
- We can profit from the administration and publishing features DITAworks offers.
- We have achieved a tight integration with our development process.