How to document a Software Development Project

There is no one-size-fits-all for documenting software projects. What we do is giving you an introduction on how to get started with the projectdoc Toolbox and the Software Development Add-on to define your documentation requirements with Confluence.

The development of software products needs cross-functional teams. The team is not limited to the people who actually write and test the code. There are technical writers to create the documentation, marketing specialists, product owners, domain experts and many more. And the list of stakeholders – people taking interest in the project or product – makes this crowd even larger: managers to provide sufficient resources, customers, users and also many more.

While we believe that every member of a team needs to have the freedom to select the tools that are most efficient and effective to use for their tasks, we also believe that there has to be an information system that is capable to provide access to all information for each and every stakeholder. A wiki (or team collaboration platform) – like Atlassian Confluence – is a great tool for this task.

In this howto we explain how to get started to document a software project using the projectdoc Toolbox and Atlassian Confluence. We assume that many aspects are relevant for teams independent of their chosen infrastructure / toolset – that is if your tooling is different, this aspects still apply. The projectdoc Toolbox provides tools, mainly macros and blueprints for different document types (called doctypes in projectdoc lingo), which makes it easier for teams in their task to collaborate and share information with all stakeholders.

After giving a brief introduction of some of the different aspects of documentation, we focus on how to start documenting your software architecture with this set of tools.
Prerequisites

If you want to get hands-on, this introduction assumes that you have access to a Confluence server with at least the following add-ons installed:

1. projectdoc Toolbox for Confluence - available on the Atlassian Marketplace
2. Core Doctypes - available as free add-on for the projectdoc Toolbox on the Atlassian Marketplace
3. Software Development Doctypes - available as free add-on for the projectdoc Toolbox on the Atlassian Marketplace
4. arc42 Template (for software architecture documentation) - available as free add-on for the projectdoc Toolbox on the Atlassian Marketplace

Sources for the free doctype add-ons are available on Bitbucket under the Apache 2 License. This makes it easier for users to adapt the blueprints to their specific needs.

We will not cover what software is or discuss software development methodologies.

Maven Integration for Confluence
Software developers using Maven may have a look at the Maven Extension. This extension is in beta status but already provides some useful integration points to import information from Maven POMs into a Confluence space.

Information Architecture

A documentation for a software project is a huge endeavor. Independent of your tool set you need to answer questions like

- what do you want to document?
  - e.g. do you track project progress, facilitate team communication, or provide user documentation?
- who is your audience?
  - e.g. is a particular document read by team members or customers?
- What tools you want to use?
  - e.g. how do you sketch diagrams, render code snippets, or access information from remote servers?

You may also have to handle the details like

- selecting a style guide for each of the supported languages
  - may be less important for internal documentation
- start creating a glossary

Once you have decided what to document you need to define the document types which includes the properties (mostly metadata) and sections. These doctypes define the document templates that support your team writing documents. Finally you need to organize the workspace by defining which documents are stored at which location. Some documents may be added as children (e.g. a retrospective to a review document) or stored in a metadata independent location (e.g. meeting minutes; see Physical Location).

All this work is done or coordinated by information architects (or documentation architects). It is important to put much thought into this right from the start. But typically you’ll find out during the documentation process that some assumptions were wrong or new requirements are discovered that need to be handled. So keep in mind that documentation, like software development, is a process where teams learn and adapt their collaboration routines. This will lead to constant improvements of the documentation or the process of how people want to work together.

Principles and Practices for Agile Documentation

Agility does not strive for no documentation. Agility emphasises the fact that the investment in a given document – including all the maintenance costs that come with its creation – must be outweighed by the positive effects this written information has.

In Agile Documentation we have compiled a number of principles and practices we consider useful to organize the technical communication for collaborating teams.

If you are looking for some inspiration for some more fundamental tools, check them out! Especially Know your Mission for making sure your documentation effort is worth the investment.

Software Systems or Architecture Documentation

With all this basics settled the answer to “How to document a Software Development Project” is organized by the following topics.

<table>
<thead>
<tr>
<th>Name</th>
<th>Short Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Documentation Basics</td>
<td>Software documentation is a large field to communicate with different stakeholders with different information needs. This topic introduces to the basics of documenting a project with a wiki.</td>
<td></td>
</tr>
<tr>
<td>Create your Software Documentation Space</td>
<td>How to create a space to start documenting your software architecture or software system using the projectdoc Toolbox for Confluence and additional free add-ons.</td>
<td></td>
</tr>
<tr>
<td>Software Project Documentation</td>
<td>How to start your software project documentation? Here are the steps to get started with Confluence and the projectdoc Toolbox.</td>
<td></td>
</tr>
</tbody>
</table>

Teams may either want to transclude content from remote information systems to documents in their wiki or update documents in their wiki via a REST API. The projectdoc Toolbox provides tools for both use cases.

<table>
<thead>
<tr>
<th>Name</th>
<th>Short Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Systems Extension</td>
<td>Add-on to extend the projectdoc Toolbox to integrate remote information systems.</td>
<td>Provides access to external systems, like image repositories, artifact repositories, or source repositories, to link to and transclude information.</td>
</tr>
<tr>
<td>Web API Extension</td>
<td>Add-on to extend projectdoc with an API to access on the web.</td>
<td>Provides access to information on projectdoc documents from systems that have access to the Confluence instance via a REST API.</td>
</tr>
</tbody>
</table>

Documenting other Aspects
To document a software project does not only require the systems or architecture documentation. In *Software Architecture Documentation* we list the four quadrants:

1. Process Documentation  
2. Project Documentation  
3. System Documentation  
4. User Documentation

Here are some more examples on how to use the *projectdoc Toolbox* to provide project relevant information.

Facilitate Team Communication

**Developer Journals**

Developers may keep *journals* to track open questions and interesting findings. These journals may be open for all team members to find relevant information on project topics.

```
This information is part of the project quadrant (Q2)
```

**Team Journals**

Besides the individual journals the team may write a team journal to plan and run iterations or sprints. The *Doctypes for Agile Planning* provide templates for these communication needs.

```
This information is part of the project quadrant (Q2)
```

**Teamwork**

Use *Doctypes for Teamwork* to define checklists, processes, patterns, tools, and rules a team agrees upon. Writing them down makes them accessible for anyone - especially for new team members. Keep these documents short and to the point!

```
This information is part of the process quadrant (Q1)
```

**Project Management**

*Doctypes for Project Management* provides tools to communicate important project information with decisions, risks, open issues, and meeting minutes.

```
This information is part of the project quadrant (Q2)
```

**Domain Knowledge**

**Glossary**
A **glossary** is helpful for most project and product documentations.

**Project Library**

A **project library** collects relevant information for the project that is typically provided by external resources. These resources include books, website, blogs, or articles. The library is especially helpful to provide information about the domain or technology stack for new team members.

**Domain Space**

In order to create a glossary the team may need to do some **domain crunching**. Create a separate space to collect all information relevant for the domain. Add things you learned about the domain that should be accessible beyond the scope of the current iteration or sprint of your team.

**Stakeholder Communication**

Different stakeholders have different information and communication needs.

Some stakeholder need information for a given topic. In this case the team may create a **workspace** to **collaborate to create the requested document**. After the information is delivered the space may be archived.

**User Documentation**

Products, especially technical products with powerful or complex interfaces, need documentation for their users.

**Plugin for Maven**

Teams whose builds depend on **Maven** and that use automation heavily typically end up in writing their own **plugins for Maven**. Although these plugins are often only released for internal use, developers who employ these plugins need to have access to **proper documentation**.

**Library for Java**

Teams using continuous deployment may choose to create libraries to modularize their code base. Each library is a product of its own with its own release cycle. Especially if the library is made public available, a **sound and similar structure of the documentation** helps developers to find information on how to use this library easily.

**Resources**

More information on creating documentation with Confluence.

[Building better documentation](#) Information on building better documentation with Confluence by Atlassian.
**Agile Documentation**
Information on agile documentation: from values to principles and practices. A set of actionable tools for team communication.

**Use Cases**
An overview over the use cases for which the projectdoc Toolbox provides support.

**Show Cases**
A set of Confluence spaces showing projectdoc in action.

**Service Management Documentation**
Communicate the necessary information on maintaining and using a service of your portfolio.

**Doctypes for V-Modell®XT**
Use products (templates) from the V-Modell®XT in your Confluence wiki as blueprints!