Agile Architecture Documentation System

System documentation generator

Version 0.0.7, 2021-12-01T08:56:41Z

Table of Contents

Chapter 1. Context

If you encounter any error, feel free to enter an issue on GitHub.

agile-architecture-documentation-system tries to provide a solution to a rather usual, but not so well solved, problem: how to document architecture of a software system with the minimal effort and the maximal result. As a consequence, this product tries to help the architect in all possible ways. == Context diagram

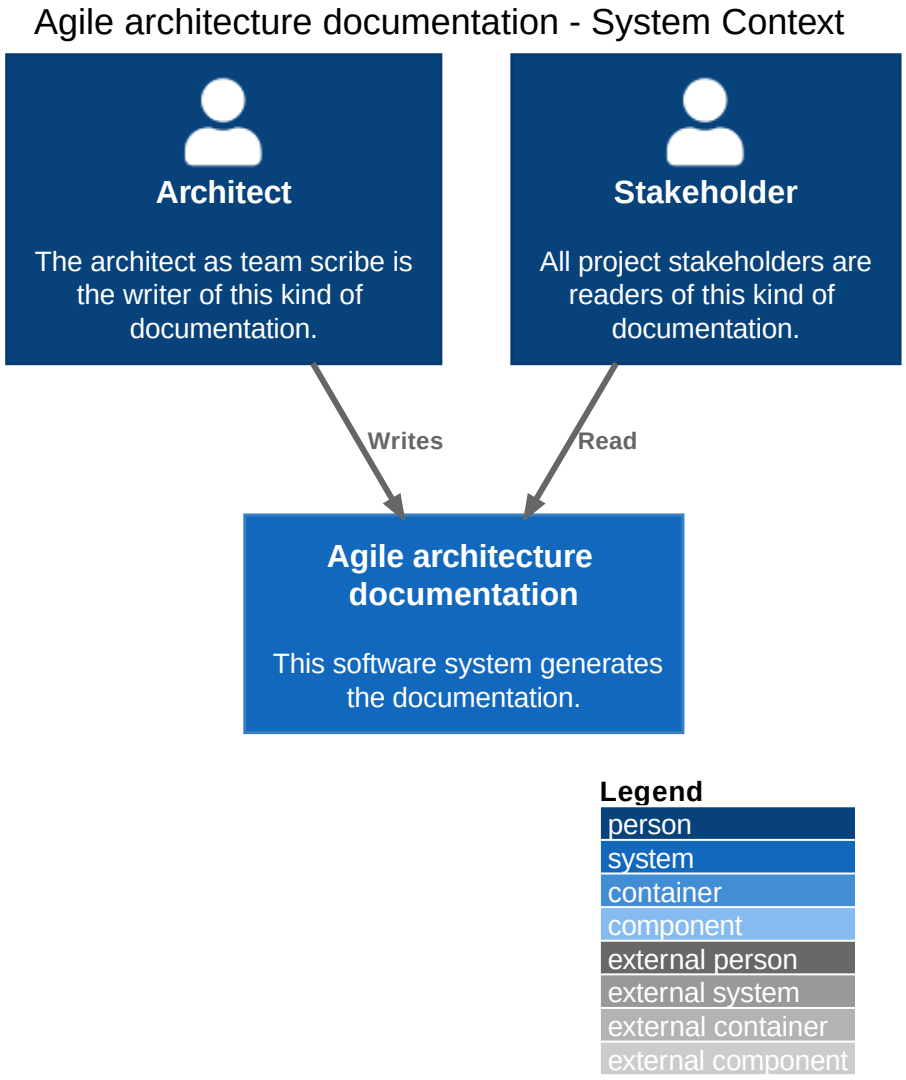


Illustration of agile-architecture-documentation usage

We identify different user profiles:

- the architect as a scribe role, which is responsible for maintaining a correct documentation of the architecture
- the various stakeholders, be them developers, product owners, ops teams, and so on, which will refer to the produced documentation as an entrypoint.

Chapter 2. Functional Overview

If you encounter any error, feel free to enter an issue on GitHub.

This software helps the architect writing a coherent, complete and readable software architecture documentation. Usage of this software is quite simple (for a developer).

The first step is to initiate a documentation (which is documented in README.md).

Then, architect has to write content in all `.adoc` files generated in `src/docs/asciidoc`. Notice writing those files will be eased out by using the `Architecture` java class to generate diagrams, and the enhancers this project provides.

The enhancers allows architect to have content added to this documentation without having to write it himself. Typically, we're able to include README documents, architecture decisions, and more to come later … (well, provided the users ask for additionnal includes)

Chapter 3. Quality Attributes

If you encounter any error, feel free to enter an issue on GitHub.

The documentation build should be fast enough to allow architect to write its documentation and have it reloaded in "human real time". In other words, we expect a simple documentation to generate in less than 10 seconds and a complex documentation (involving reading configuration from remote systems and so on) to generate in less than 1 minute. **TODO**

Chapter 4. Constraints

If you encounter any error, feel free to enter an issue on GitHub.

TODO

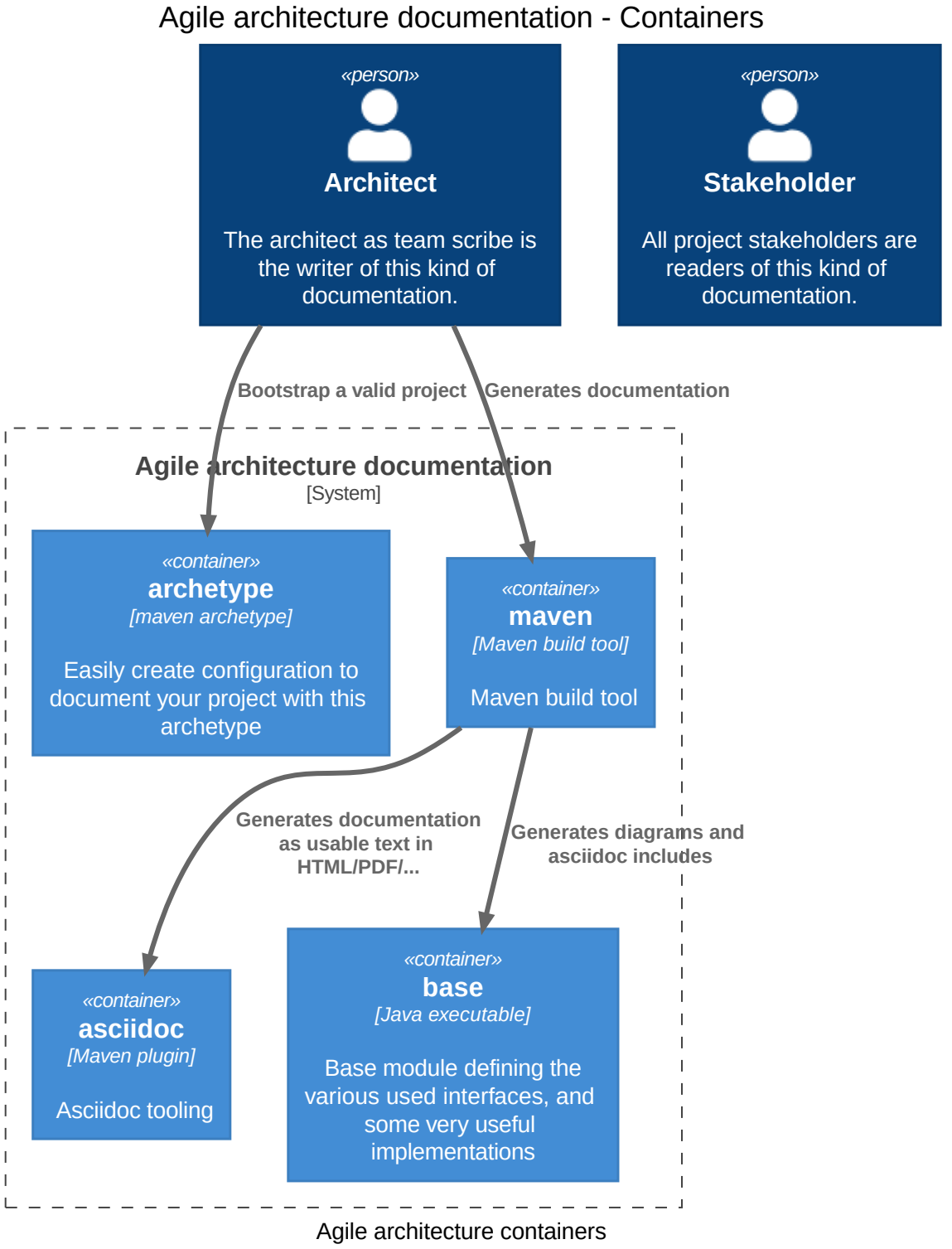
Chapter 5. Principles

If you encounter any error, feel free to enter an issue on GitHub.

- Architecture model must be totally described in the Architecture class.
- Additional architecture information (typically, location of README files for components) has to be set has properties in the various Structurizr `Element` subclasses (`SoftwareSystem`, `Container`, `Component`)

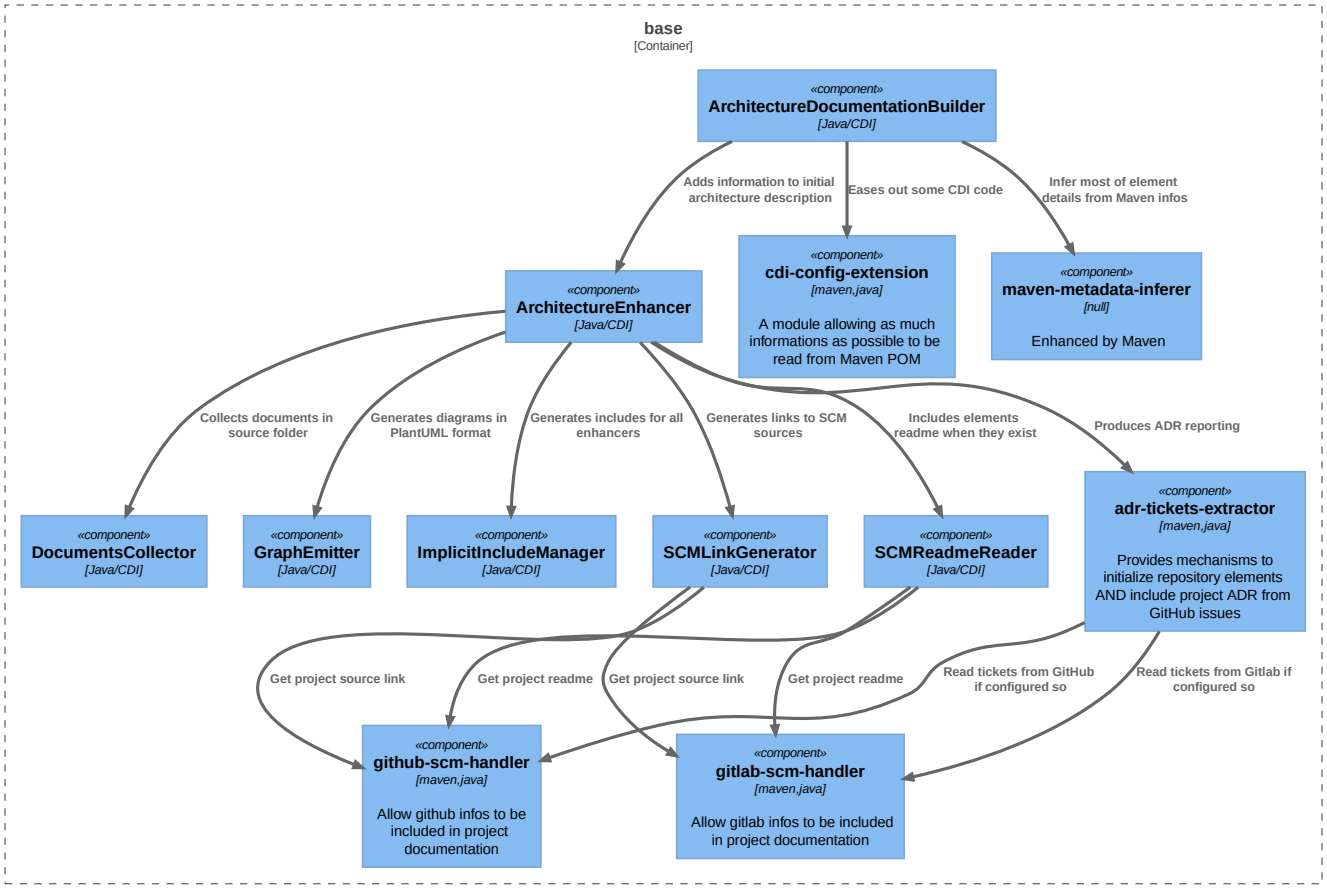
Chapter 6. Software Architecture

If you encounter any error, feel free to enter an issue on GitHub.



Agile architecture containers

Agile architecture documentation - base - Components



Agile architecture base components view

TODO

Chapter 7. Code

If you encounter any error, feel free to enter an issue on GitHub.

Chapter 8. Data

If you encounter any error, feel free to enter an issue on GitHub.

Application hosts no data other, since verything is in the Maven folder.

Chapter 9. Infrastructure Architecture

If you encounter any error, feel free to enter an issue on GitHub.

The main goal of this application is to produce content that can be browsed as HTML or PDF. As a consequence, there is no requirement for any production infrastructure.

Chapter 10. Deployment

If you encounter any error, feel free to enter an issue on GitHub.

Deployment of the agile-architecture-documentation code is done by pushing code to GitHub. On push, there is a GitHub action (defined in main module .github folder) which will compile everything and generate the github pages.

Chapter 11. Development Environment

If you encounter any error, feel free to enter an issue on GitHub.

Development of this documentation solution requires a Java environnment. In other words, you need

- A recent JDK (at least Java 11)
- Maven (at least version 3.6)
- A Java IDE

Optionally, things will be better if you have * an Asciidoc editor * a livereload compatible browser

Chapter 12. Operation and Support

If you encounter any error, feel free to enter an issue on GitHub.

TODO

Chapter 13. Decision Log

If you encounter any error, feel free to enter an issue on GitHub.

13.1. Use ADR to document architecture decisions

Date: 2020-01-14

We will store decisions as asciidoc documents in `src/docs/asciidoc/decisions`.

Accepted

The purpose of this section is to simply record the major decisions that have been made, including both the technology choices (e.g. products, frameworks, etc) and the overall architecture (e.g. the structure of the software, architectural style, decomposition, patterns, etc). For example:

- Why did you choose technology or framework "X" over "Y" and "Z"?
- How did you do this? Product evaluation or proof of concept?
- Were you forced into making a decision about "X" based upon corporate policy or enterprise architecture strategies?
- Why did you choose the selected software architecture? What other options did you consider?
- How do you know that the solution satisfies the major quality attributes?
- ...
    - Decisions will be integrated in generated documentation
    - Rendering will be optimal
    - Decisions will be burried deep inside Asciidoc hiearchy
    - Decisions can be put in a specific folder
    - We can use "tooling" to access those decisions
    - Decisions will be isolate from architecture documentation
    - Decision rendering will use another tooling (typically Markdown)

Integrating decisions in Asciidoc requires changing a section of architecture.