Documentation

Every software development project requires documentation. Agile Software Development values *working software over comprehensive documentation*. Still, projects should include the key information needed to understand the development and the use of the generated software.

Documentation shouldn't be an afterthought. Different written documents and materials should be created during the whole life cycle of the project, as per the project needs.

Goals

- Facilitate onboarding of new team members.
- Improve communication and collaboration between teams (especially when distributed across time zones).
- Improve the transition of the project to another team.

Challenges

When working in an engineering project, we typically encounter one or more of these challenges related to documentation (including some examples):

- **Non-existent**.
    - No onboarding documentation, so it takes a long time to set up the environment when you join the project.

    - No document in the wiki explaining existing repositories, so you cannot tell which of the 10 available repositories you should clone.
    - No main README, so you don't know where to start when you clone a repository.
    - No "how to contribute" section, so you don't know which is the branch policy, where to add new documents, etc.
    - No code guidelines, so everyone follows different naming conventions, etc.
- **Hidden**.
    - Impossible to find useful documentation as it's scattered all over the place. E.g., no idea how to compile, run and test the code as the README is hidden in a folder within a folder within a folder.
    - Useful processes (e.g., grooming process) explained outside the backlog management tool and not linked anywhere.
    - Decisions taken in different channels other than the backlog management tool and not recorded anywhere else.
- **Incomplete**.
    - No clear branch policy, so everyone names their branches differently.
    - Missing settings in the "how to run this" document that are required to run the application.
- **Inaccurate**.
    - Documents not updated along with the code, so they don't mention the right folders, settings, etc.
- **Obsolete**.
    - Design documents that don't apply anymore, sitting next to valid documents. Which one shows the latest decisions?

- **Out of order (subject / date)**.
  - Documents not organized per subject/workstream so not easy to find relevant information when you change to a new workstream.
  - Design decision logs out of order and without a date that helps to determine which is the final decision on something.
- **Duplicate**.
  - No settings file available in a centralized place as a single source of truth, so developers must keep sharing their own versions, and we end up with many files that might or might not work.
- **Afterthought**.
  - Key documents created several weeks into the project: onboarding, how to run the app, etc.
  - Documents created last minute just before the end of a project, forgetting that they also help the team while working on the project.

What Documentation Should Exist

- Project and Repositories
- Commit Messages
- Pull Requests
- Code
- Work Items
- REST APIs
- Engineering Feedback

Best Practices

- Establishing and managing documentation
- Creating good documentation
- Replacing documentation with automation

Tools

- Wikis
- Languages
  - markdown
  - mermaid
- How to automate simple checks
- Integration with Teams/Slack

Recipes

- How to sync a wiki between repositories
- Using DocFx and Companion Tools to generate a Documentation website
- Deploy the DocFx Documentation website to an Azure Website automatically
- How to create a static website for your documentation based on MkDocs and Material for MkDocs

Resources

- Software Documentation Types and Best Practices

Last update: August 22, 2024