Documenting software architecture

1 “Working software over comprehensive documentation” does not mean “don’t write documentation”.

2 The code doesn’t tell the whole story. Supplementary documentation can be used to describe what you can’t get from the code. Make the scope of the documentation a single software system. Here is a starting point, and there are others including arc42.

Context

A system context diagram, plus some narrative text to “set the scene”.

Functional Overview

An overview of the software system; perhaps including wireframes, UI mockups, screenshots, workflow diagrams, business process diagrams, etc.

Quality Attributes

A list of the quality attributes (non-functional requirements; e.g. performance, scalability, security, etc).

Constraints

A list of the environmental constraints (e.g. timescales, budget, technology, team size/skills, etc).

Principles

A list of the development and architecture principles (e.g. coding conventions, separation of concerns, patterns, etc).

Software Architecture

A description of the software architecture, including static structure (e.g. containers and components) and dynamic/runtime behaviour.

Code

A description of important or complicated component implementation details, patterns, frameworks, etc.

Data

Data models, entity relationship diagrams, security, data volumes, archiving strategies, backup strategies, etc.

Infrastructure Architecture

A description of the infrastructure available to run the software system.

Deployment

The mapping of software (e.g. containers) to infrastructure.

Development Environment

A description of how a new developer gets started.

Operation and Support

An overview of how the software system is operated, supported, monitored, etc.

Decision Log

A log of the major decisions made; e.g. as free format text or a collection of “Architecture Decision Records”.

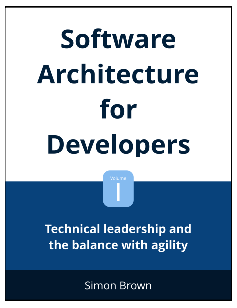
This is a **starting point**; add and remove sections as necessary.

3 Think about supplementary documentation as being a **guidebook** containing maps, points of interest, sights, itineraries, history, culture, practical information, etc. It should be lightweight, readable in 1-2 hours and give software developers enough information to get started, accelerating the process of exploring an unfamiliar codebase.

4 Documentation isn’t a one-time task. Instead, create living documentation that evolves continuously. Keep it up to date automatically with tooling, or by adding an item to your “definition of done”.

5 There are many tooling options; from Microsoft Word and Atlassian Confluence to Markdown and AsciiDoc files versioned alongside the source code. Reduce duplication and increase consistency by generating diagrams and documentation from a single source where possible.

“**Software Architecture for Developers**”
A developer-friendly, practical and pragmatic guide to lightweight software architecture, technical leadership and the balance with agility.



Software Architecture for Developers

1

Technical leadership and the balance with agility

Simon Brown

Software Architecture for Developers

2

Visualise, document and explore your software architecture

Simon Brown

Structurizr
Visualise, document and explore your software architecture

A collection of tooling to help you visualise, document and explore your software architecture.

@simonbrown