# Template

# table of contents

**About arc42**

arc42, the Template for documentation of software and system architecture.

Created and maintained by Dr. Peter Hruschka, Dr. Gernot Starke and contributors.

Template Revision: 8.0 EN (based on asciidoc), February 2022

# 1. Introduction and Goals

This Java application represents an architecture showcase for an application which is implemented with ideas from DDD and Clean Architecture.

It consists of an client app based on JavaFX/Gluon (ready for cross compiling on different platforms, e.g. with GitHub Actions) and a server part based on Quarkus (which exposes its functionality with some RESTful endpoints and a tiny UI created with JSF).

This document is based on the arc42-Template template but only extends some parts of the template to exemplify how the ideas of arc42, DDD, C4-Model and docs-as-code can be connected.

If you are looking for a complete example of a fully filled arc42 template, you may have a look at the DokChess example from the arc42 team.

Except for "Event Storming Workshop result pictures", all diagrams are created with plantuml.

Since plantuml diagrams, like asciidoc documents, are basically plain text files, this has the advantage, among other things, that the VCS used can display diffs correctly and so every change is absolutely traceable.

In the case of C4 models, the appropriate extension for plantuml is set so that these models can also be created with plantuml.

For further Information see Introduction and Goals in the arc42 documentation.

# 2. Architecture Constraints

The showcase combines ideas from the two standard works "Domain Driven Design" and "Clean Architecture".

The package structure is based on the BCE concept (boundary control entity) to enable a simple and standardized structure.

For further Information see Architecture Constraints in the arc42 documentation.

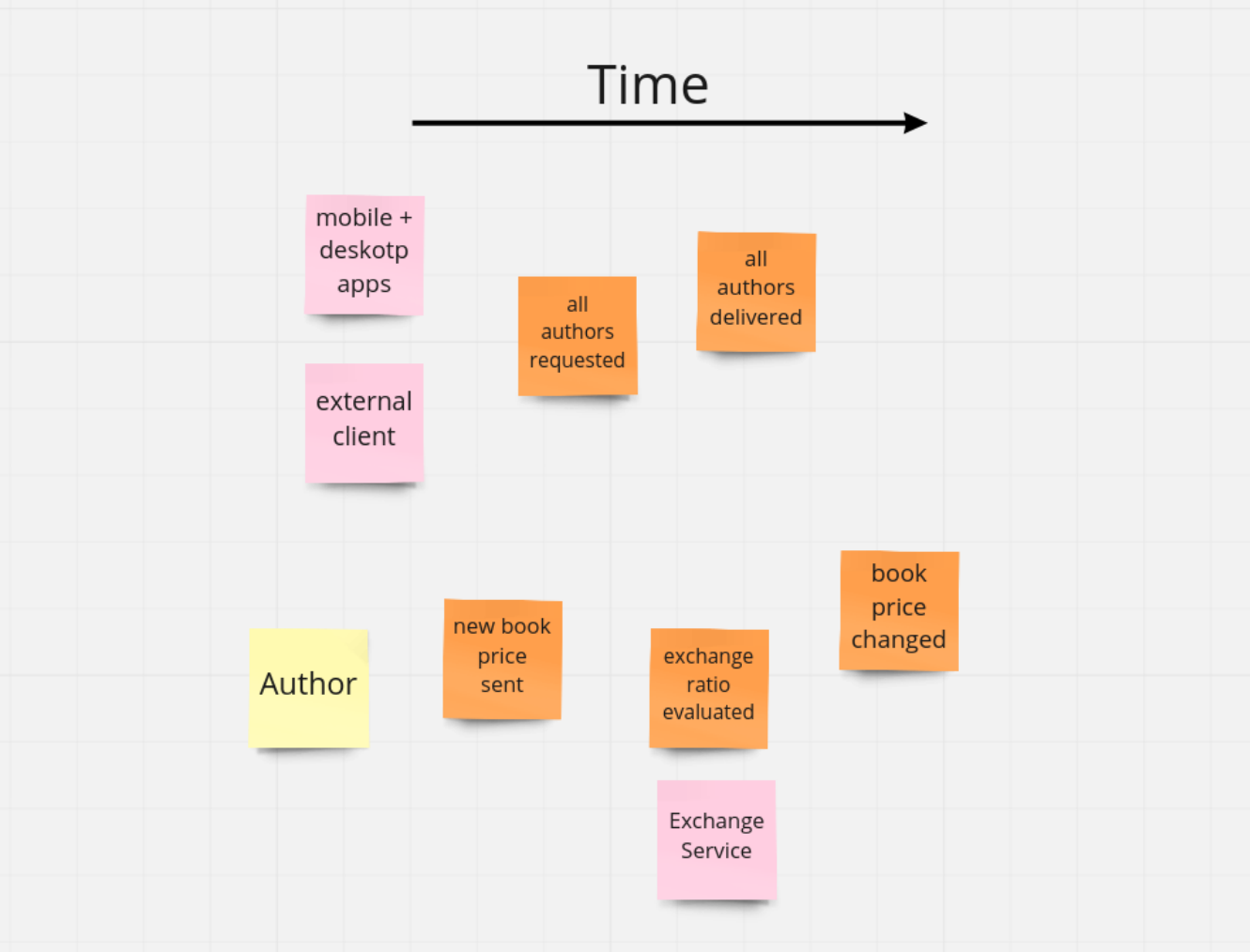
# 3. System Scope and Context

A picture says more than a thousand words, use text to write additions or more in-depth descriptions of the picture.

## 3.1. Business Context

The Business Context could be the result of "Big Picture" Event Storming workshops. It is important that the "problem space" (domain) can be understood on a "high" level (this is exactly the idea behind a Big Picture Event Storming Workshop).

The results of such a workshop are also the basis of technical decisions (e.g. a Bounded Context could be a 1:1 mapping to a Microservice).



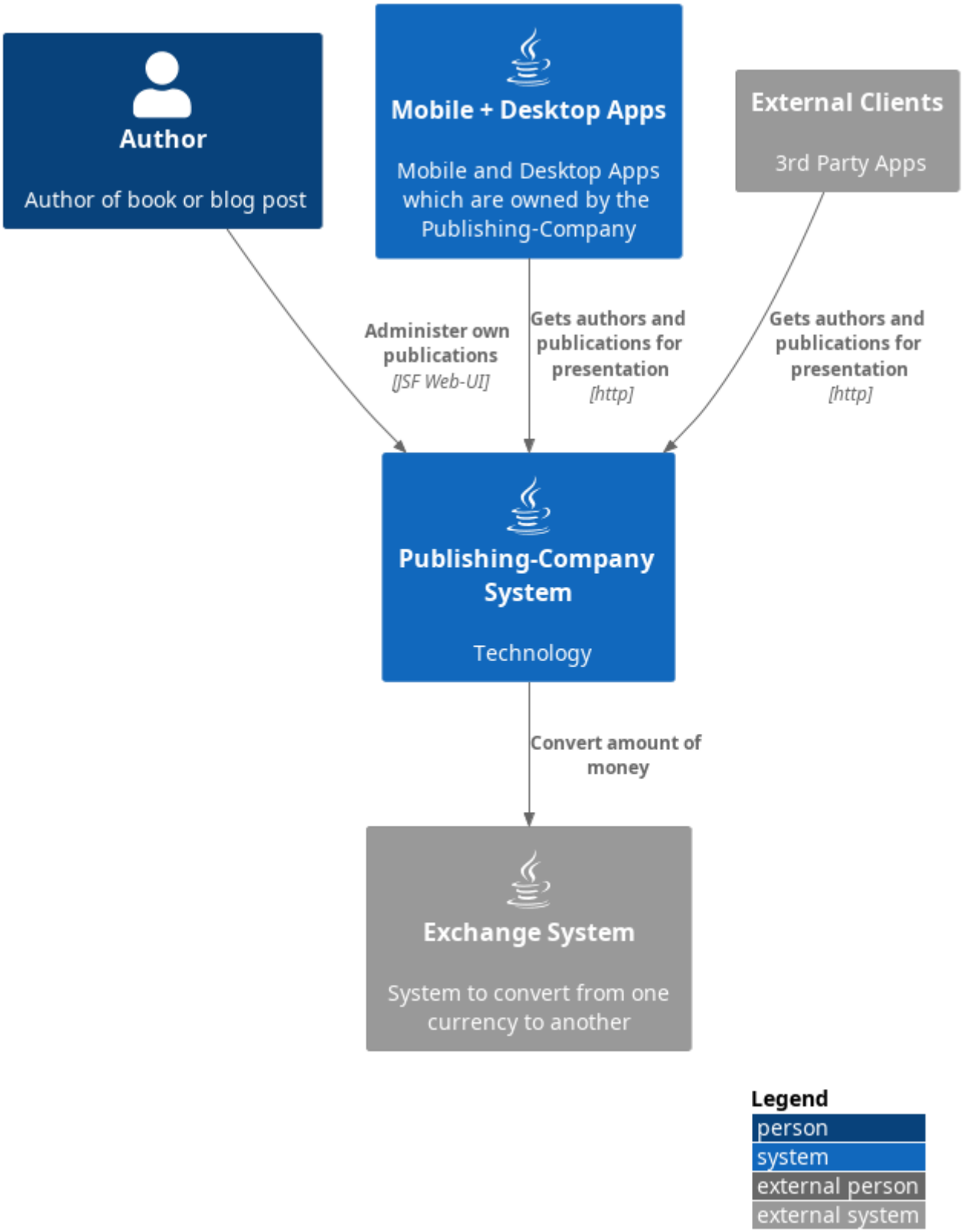

Another part of this chapter could be context map, distilled from the "big picture", if your system consists of more than one Bounded Context (which is normally the case).

## 3.2. Technical Context

The technical context or the C4 System Context Diagram can be derived directly from the results of a Big Picture Workshop (external systems and actors are a fundamental part of such a workshop).

Put simply, the technical context outlines the technical communication (e.g. communication protocols) between the external systems and your own system. The internal architecture of your own system is not of interest at this point (this will be derived and described in the course of the next chapter).

The benefit of using C4-Model is, that you have a standardized template for visual modelling from technical parts of your software.




For further Information see Context and Scope in the arc42 documentation.

# 4. Solution Strategy

See Solution Strategy in the arc42 documentation.

# 5. Building Block View

The truth of how a system is implemented, is in the code. This alone, however, is rarely enough to understand a system as a whole. Therefore it is necessary to offer the possibility to dive deeper and deeper into the system, starting from a relatively high level of flight.

These possiblity is offered by the "Building Block Views". The following table gives an overview of how the building block views, C4 diagrams and DDD are connectable.

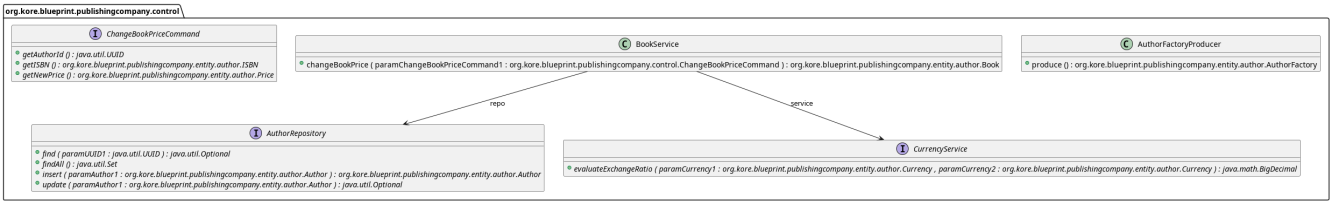
*Table 1. Building Block Views Mapping*

| **arc42 Building Block View** | **C4 Model** | **DDD** |
|---|---|---|
| Level 1 | Container diagram | Technical view of the Context Map<br>(Bounded Context as container) |
| Level 2 | Component diagram | View on the Aggregate(s)<br>of a Bounded Context |
| Level 3 | Class diagram | Always generate class diagrams from code,<br>Otherwise they outdate too quickly.<br>The following example is generated during each build.<br>When using BCE, just the control and entity packages are necessary.<br>The contents of the boundary package are implementation details and not building blocks, because an system should be build upon domain logic and not technical details.<br>The necessary informations about this layer are visible in the container diagram or Deployment View. |

For further information see Building Block View in the arc42 documentation.

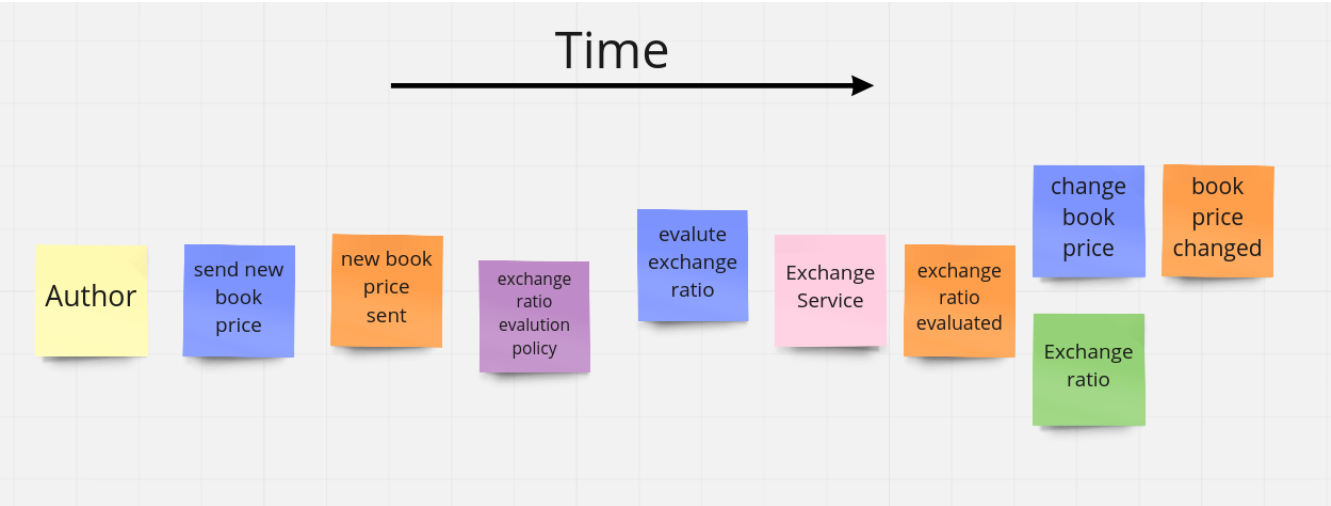
# 5.1. Class diagram

This diagram is optional and should just be used when required to understand parts of a system. In this case, it is just an example of how to get the plantuml generated during build from a maven plugin.



# 6. Runtime View

This chapter is a great place to document the outcomes of Event Storming "Process Modelling" workshops.




For further Information see Runtime View in the arc42 documentation.

# 7. Deployment View

The plantuml-stdlib provides a variety of symbols for different infrastructure components, cloud providers or "plain Kubernetes".

For further Information see Deployment View in the arc42 documentation.

# 8. Cross-cutting Concepts

See Concepts in the arc42 documentation.

# 9. Architecture Decisions

Architecture Decision Record is a great format to document decisions. But they should be organized separately from this document und just be included (use asciidoc also for the records and everything will be hassle free). The benefits are, that you can publish they separately and they are easier to find outside this document.

Don’t forget to add a section "Compliance", so that one can understand how it is checked that the decision will be complied (code snippet to ArchUnit Tests, four-eyes principle threw pull requests, and so on).

For further Information see Architecture Decisions in the arc42 documentation.

There you will find links and examples about ADR.

# 10. Quality Requirements

Quality goals and quality requirements can be derived using Big Picture Event Storming workshops.

These arise either implicitly, e.g. through external systems (keyword interoperability) or can be explicitly represented, e.g. through hotspots.

Results from process modeling workshops can be used as a basis for quality scenarios. In the form of an additional description, the connection to quality requirements can subsequently be established.

See Quality Requirements in the arc42 documentation.

# 11. Risks and Technical Debts

In addition to risks and technical debts, it is also possible to list security threats which are taken into account. But when you do this, be careful with the visibility of the document because it could be a security threat when some people know what defense strategties you are using.

For further Information see Risks and Technical Debt in the arc42 documentation.

# 12. Glossary

This chapter is a great place to put Bounded Context specific meanings of words. So it can be helpful, to a "Bounded Context" column to a glossary table.

For further Information seee Glossary in the arc42 documentation.