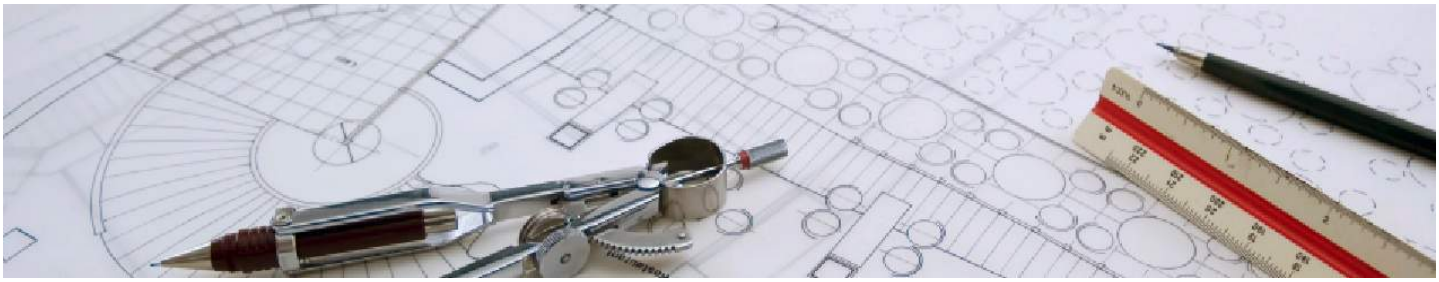
Pragmatic, Lean and Painless Architecture Documentation



SIEMENS MC

Architecture Day 2019

Weisendorf

Peter Hruschka
Atlantic Systems Guild
www.b-agile.de
www.arc42.de
www.req42.de

The Atlantic Systems Guild

Do you suffer from ...

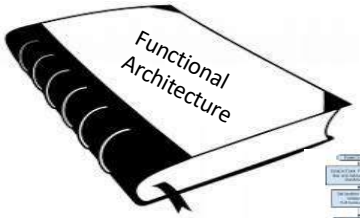
... too much source code – and no (up-to-date, trustworthy) documentation?

Do you suffer from ...

... heaps of documentation that nobody reads?

Do you suffer from...

... too many non-related documents, each of them known to only one person (or a small group) and unknown to all others?



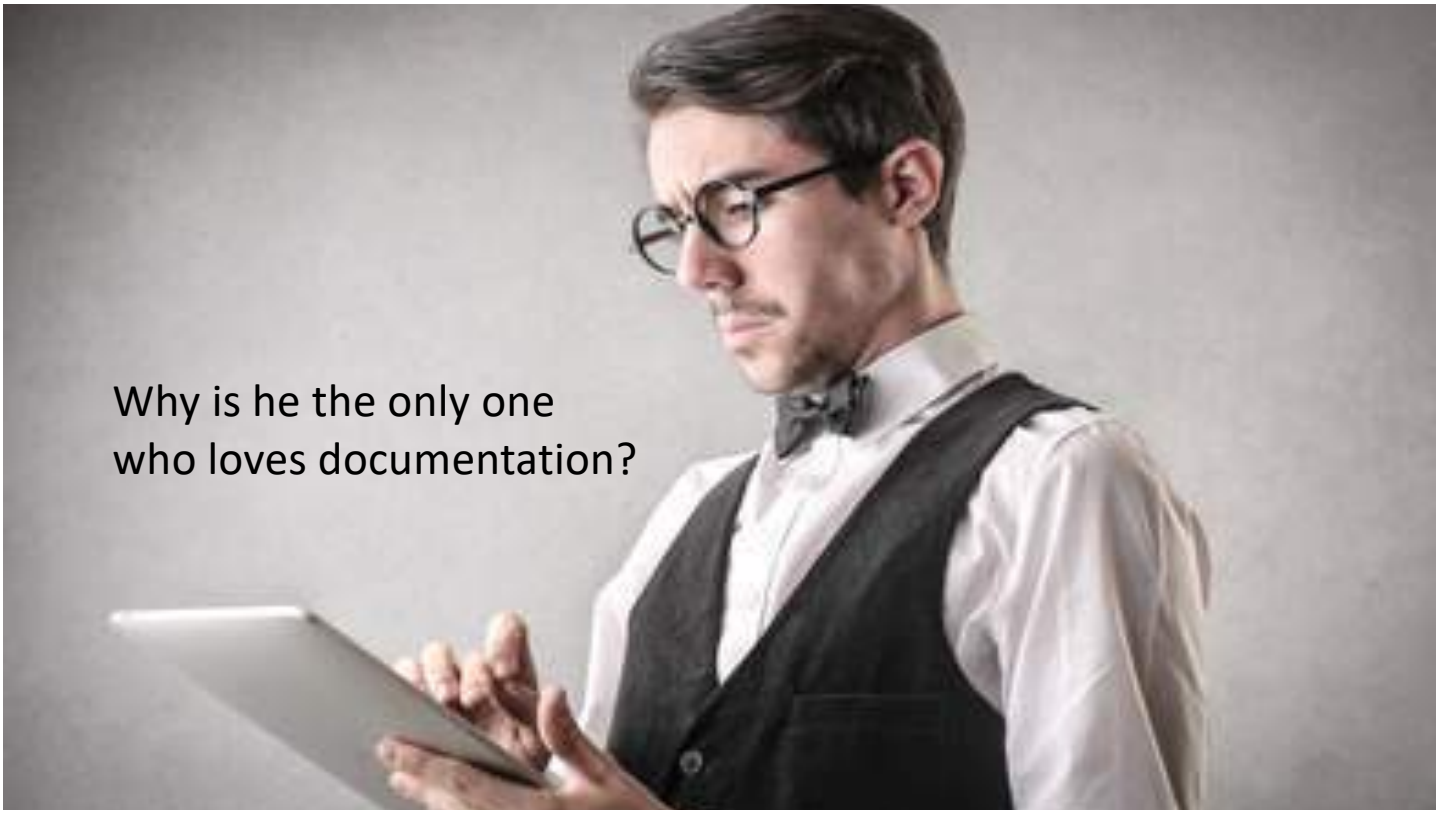
Test Plan

Detailed
Design

Roadmap

Project Plan

UI-Spec

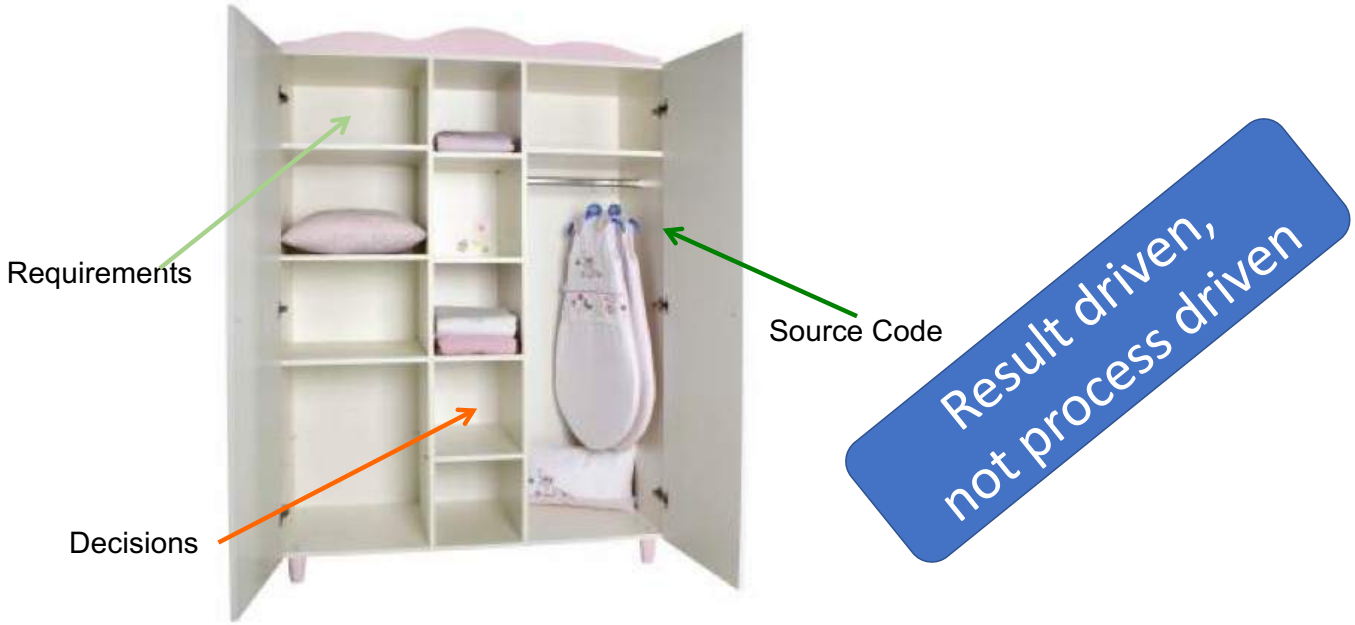


Requirements for Architecture Documentation

- **Correct (with respect to source code)**
  - Preferably less, but definitely correct!
  - current
  - ABNC (**a**ccurate, **b**ut **n**ot necessarily **c**omplete)
- **Maintainable**
  - No redundancy, refrain from too many details
  - Adequate, appropriate: Abstraction instead of completeness
  - compact
- **Understandable**
  - Can only be determined by readers
- **Shall be created in parallel to development**
  - Rarely a priori (exceptions allowed)!

Less is more!

Template-based Documentation & Development



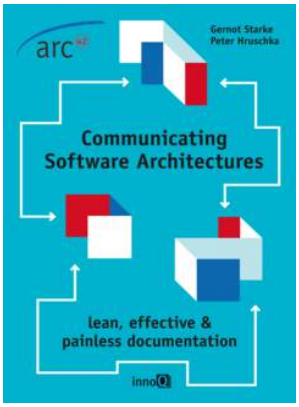
Result driven, not process driven

arc42

1 Intro & Goals
2 Constraints
3 Context
4 Solution Strategy
5 Building Block View
6 Runtime View
7 Deployment View
8 Concepts
9 Design Decisions
10 Quality Scenarios
11 Risks/Technical Debts
12 Glossary

The ARC42 Template

„Support for Structured Laziness“

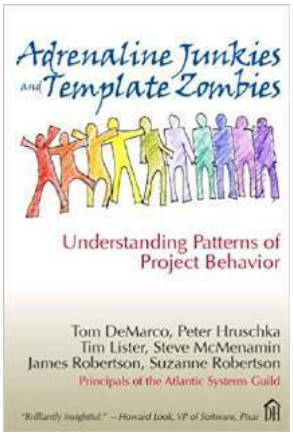


Identify your Stakeholders

1. Introduction and Goals
   - 1.1 Requirements Overview
   - 1.2 Quality Goals
   - 1.3 Stakeholders

Architecture Constraints….
- 2.1 …..

- Ask them what they expect to find in the documentation
- You should never create “orphaned deliverables” *)



*) Anti-Pattern from „Adrenaline Junkies and Template Zombies“:

„People develop project artifacts that no one values enough to pay for.“

arc42

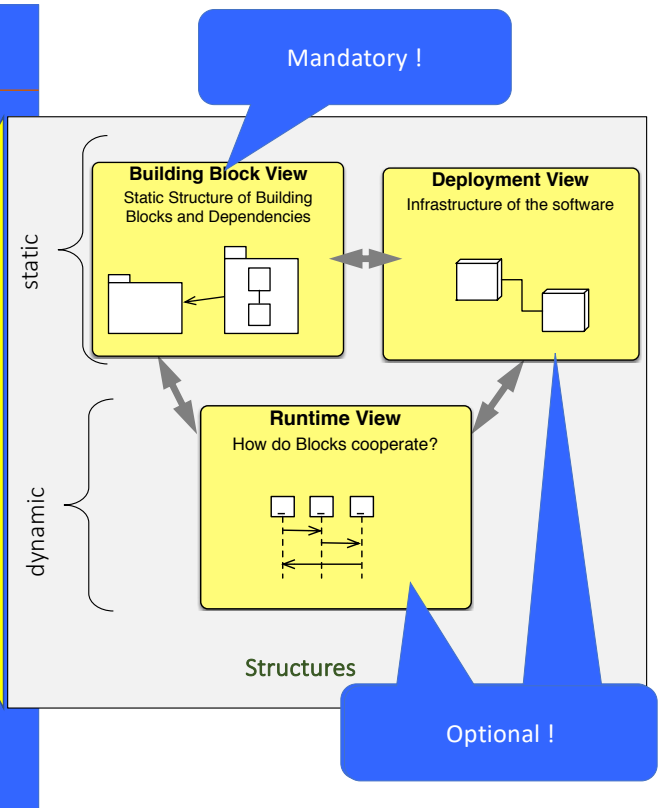
1 Intro & Goals
2 Constraints
3 Context
4 Solution Strategy
5 Building Block View
6 Runtime View
7 Deployment View
8 Concepts
9 Design Decisions
10 Quality Scenarios
11 Risks/Technical Debts
12 Glossary

3 Views (only)

not 15 (like TOGAF),
not 30 (like the Zachmann Framework)

arc42

1 Intro & Goals
2 Constraints
3 Context
4 Solution Strategy
5 Building Block View
6 Runtime View
7 Deployment View
8 Concepts
9 Design Decisions
10 Quality Scenarios
11 Risks/Technical Debts
12 Glossary



Standardize

CONTENT

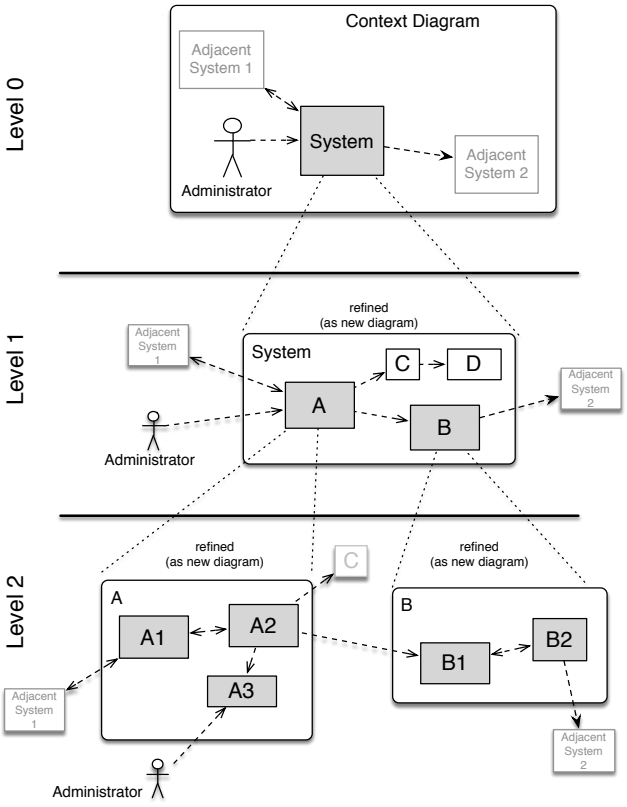
not so much form, notation, tools, *)

…

*) If you can agree, go ahed!
but do it bottom-up, not by management decree

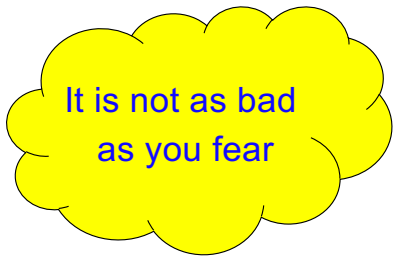
Building Block View:

A hierarchy of black and white boxes

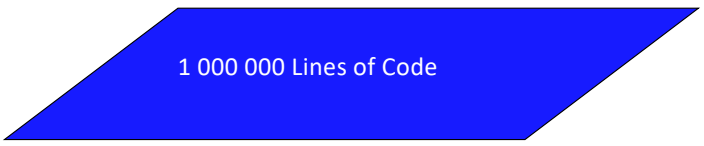


A Game with Numbers

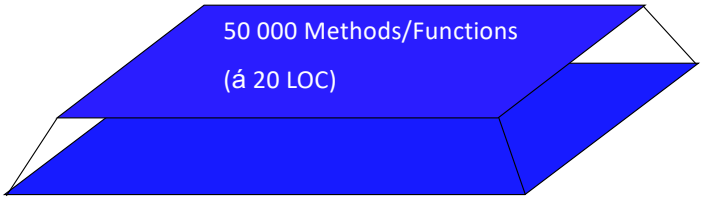
- How many levels?
- How many black boxes?
- How many white box diagrams?



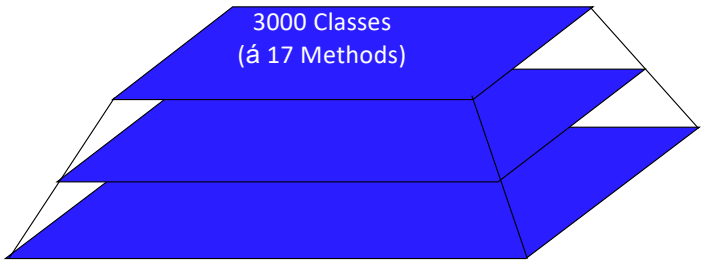
Source Code Level



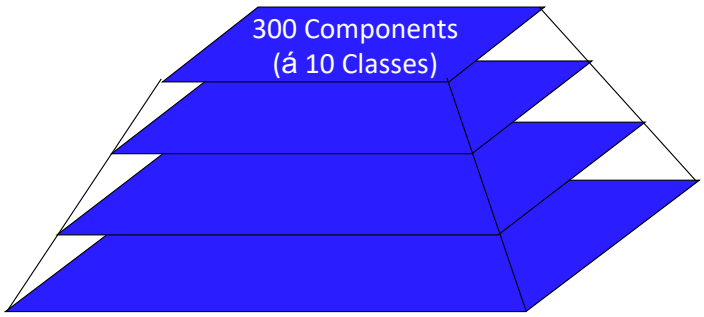
Functions/Methods Level



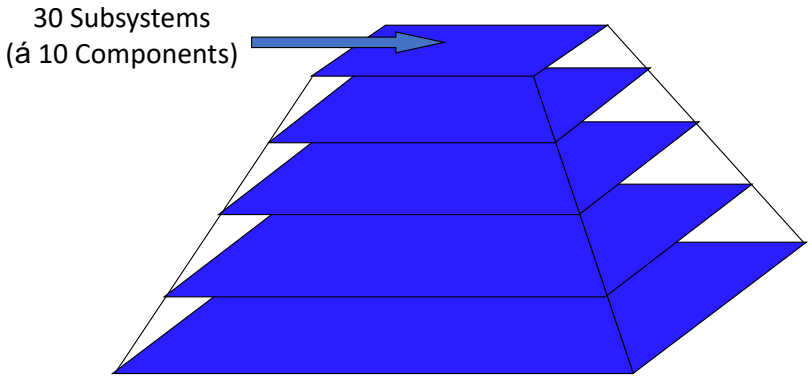
Class Level



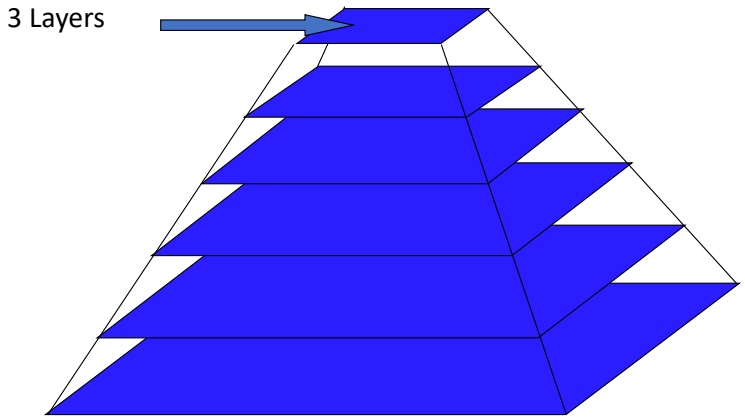
Component Level



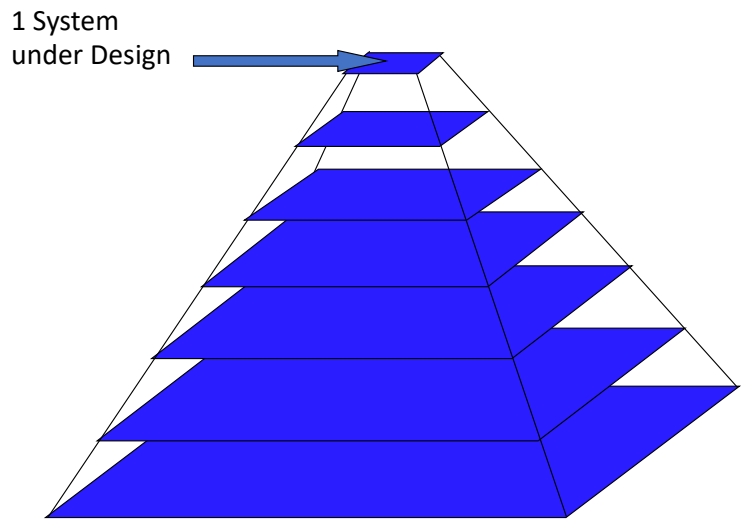
Subsystem Level



System Level

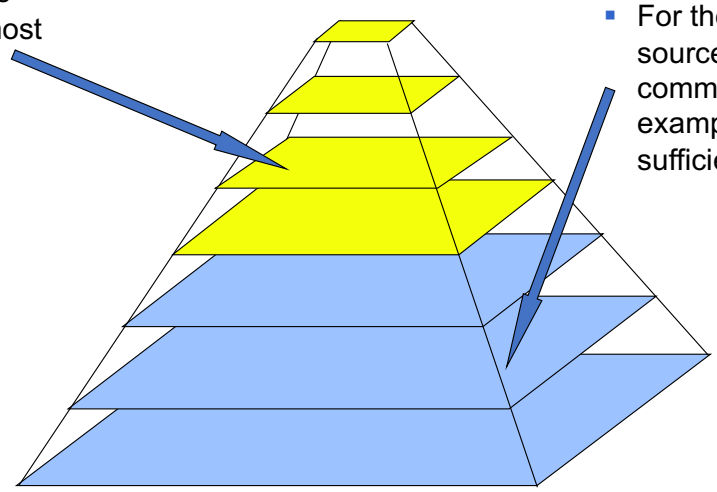


System under Design

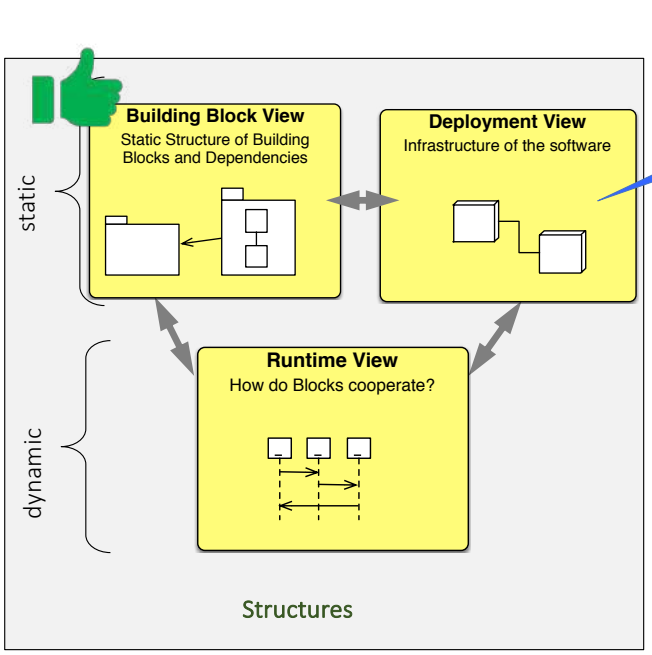


Document your Top-Levels only

- 2 – 3 levels are sufficient for most systems
- For the bottom levels source code with comments and some examples may be sufficient



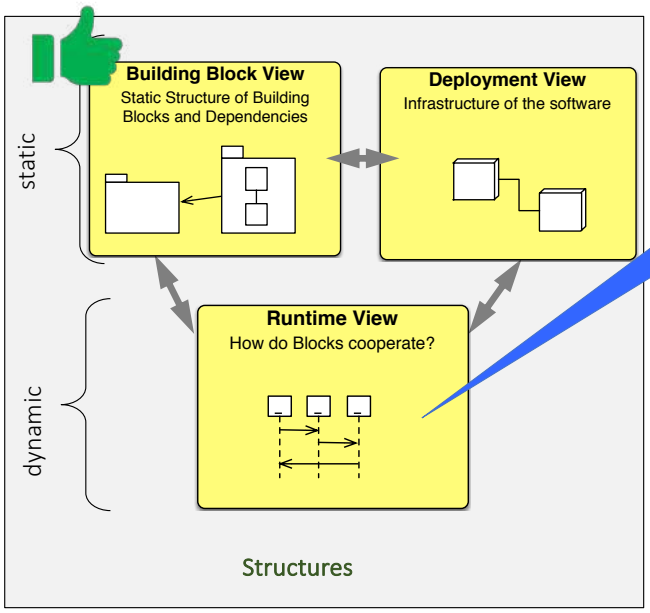
Our goal is NOT to have every line of source code documented in UML



Optional !

Use the deployment view only, if your software is distributed to multiple processors and devices

- Model your infrastructure
- Show the mapping of building blocks to that infrastructure



Optional !

Only document as many scenarios as you can promise to keep up-to-date.

Use scenarios to

1. find missing building blocks
2. add functionality or interfaces to building blocks
3. verify correct cooperation between building blocks

BUT THROW THEM AWAY (OR GIVE THEM TO THE TESTERS TO MAINTAIN) RATHER THAN LET THEM GET OUTDATED

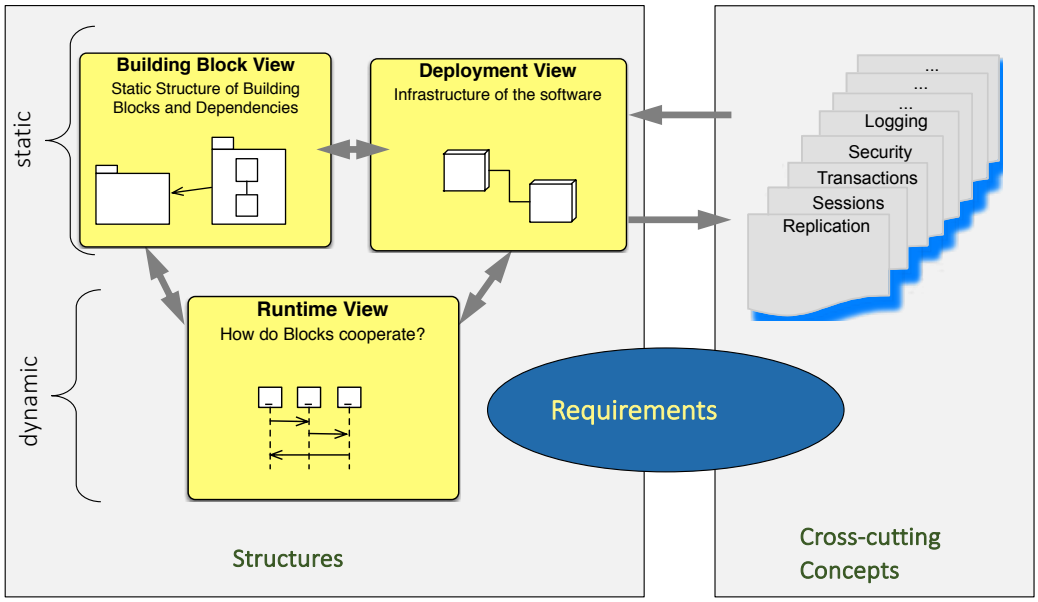
arc42

1 Intro & Goals
2 Constraints
3 Context
4 Solution Strategy
5 Building Block View
6 Runtime View
7 Deployment View
8 Concepts
9 Design Decisions
10 Quality Scenarios
11 Risks/Technical Debts
12 Glossary

The ARC42 Template

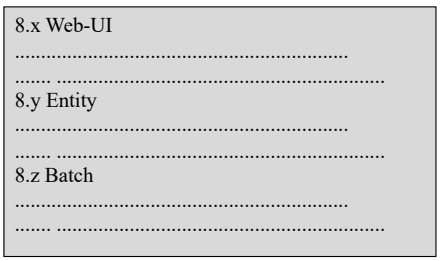
VIEWS
&
CROSS-CUTTING
CONCEPTS

Documenting Architecture...
... with different views ... and crosscutting concepts

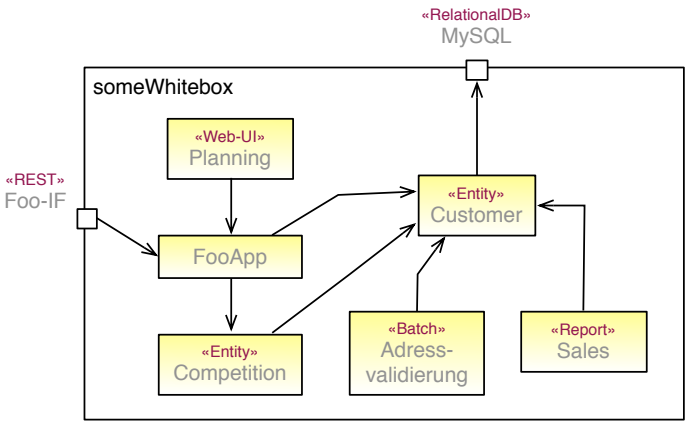


Use concepts instead of repeated documentation...

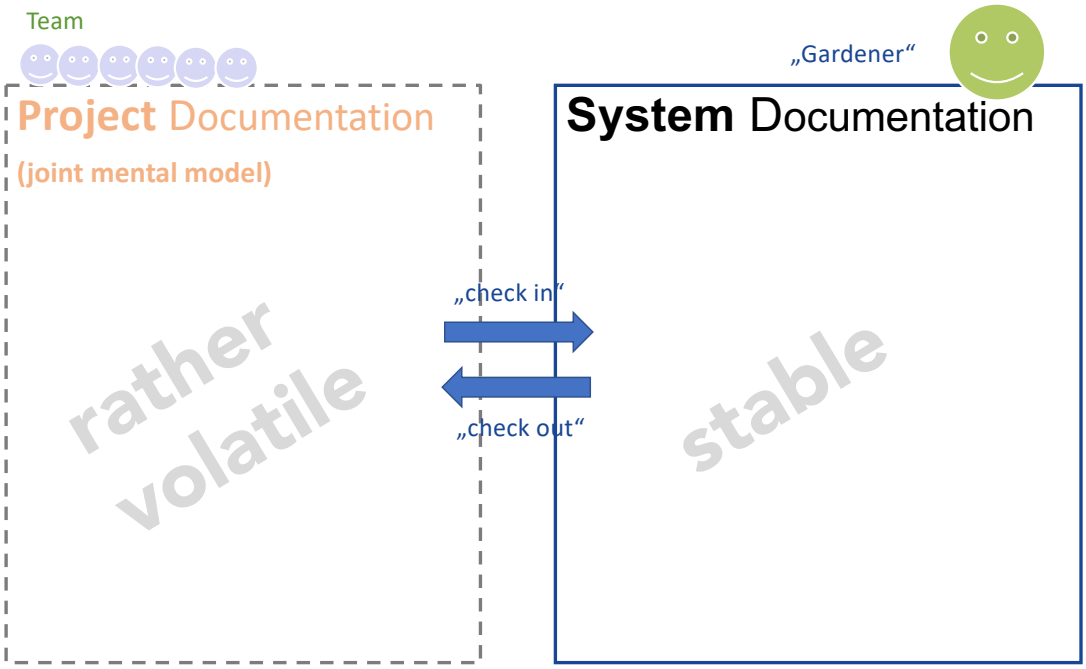
1. Explain cross-cutting concepts once in chapter 8
   - with concrete technology stacks
   - with examples
   - with rules for implementation



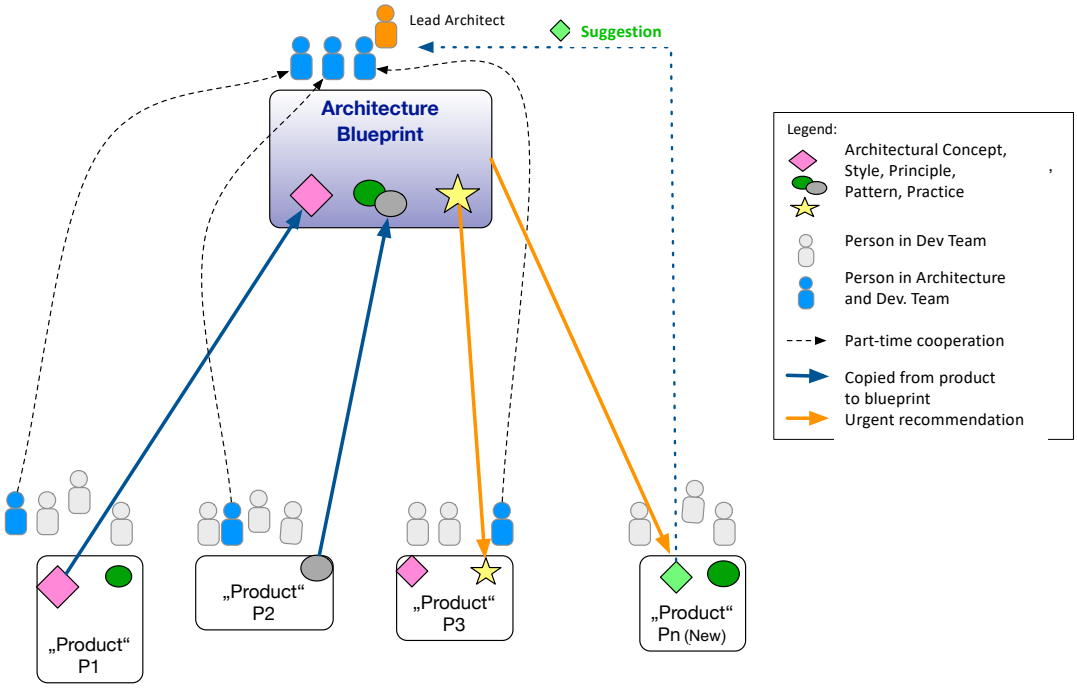
2. Link concepts to building blocks e.g. via stereotypes



Manage Documentation like Source Code



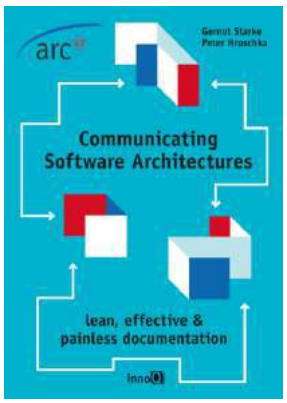
Architecture in larger Teams



Be pragmatic: Less is more

- Enough documentation to quickly understand the architecture of your system instead of source code reengineering
- Ask stakeholders what they will be looking for instead of guessing what to write
- Document the overview, not the details
- Concentrate on communication, not completeness
- Extract cross-cutting concepts instead of redundantly documenting them in multiple parts of your documentation
- Outdated documentation is worse than no documentation: keep it simple!

More Info



- G. Starke, P. Hruschka: Communicating Software Architecture – lean, effective & painless documentation (Leanpub)
- www.arc42.org
  - The home of the arc42 template
- faq.arc42.org
  - 132 frequently asked questions about arc42
- arc42 by Example (Leanpub)
  - 6 real world examples of architecture documentation
- arc42 by Example Volume 2 (Leanpub)
  - 2 real world examples from embedded systems
  - to be published later in 2019