Curriculum for

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Module
ADOC

Architecture Documentation

2025.1-RC1-EN-20251018

Table of Contents

Learning Goals Overview

- LG 1-1: Understand and explain the benefits and goals of architecture documentation
- LG 1-2: Understand the importance of architecture documentation for different stakeholders
- LG 1-3: Differentiate architectural documentation from other types of documentation
- LG 1-4: Choose appropriate notations for architecture documentation
- LG 1-5: Adapt documentation efforts to different development processes
- LG 2-1: Create architecture documentation
- LG 2-2: Identify target audiences and goals of architectural documentation
- LG 2-3: Define a suitable structure for architecture documentation
- LG 2-4: Define a suitable process for architecture documentation
- LG 2-5: Identify application scenarios for GenAI in documentation
- LG 2-6: Know and select suitable documentation templates for a given context
- LG 3-1: Document influences and constraints
- LG 3-2: Document architecture decisions
- LG 3-3: Apply different views to document architectures
- LG 3-4: Document cross-cutting concepts
- LG 3-5: Ensure consistency between parts of the documentation
- LG 4-1: Select appropriate tools for documentation work
- LG 4-2: Define requirements for documentation tools and artifacts
- LG 4-3: Know about Docs-As-Code Approach
- LG 4-4: Apply GenAI tools to support architecture documentation
- LG 5-1: Ensure documentation is fit for its purpose
- LG 5-2: Differentiate different types of assessment
- LG 5-3: Distinguish between a documentation review and an architecture review
- LG 5-4: Identify and explain goals of documentation reviews
- LG 5-5: Create checklists and questionnaires for reviews
- LG 5-6: Perform different roles in documentation reviews
- LG 5-7: Improve the quality of documentation based on review results
- LG 6-1: Explain pros and cons of sample architecture documentation

Introduction: General information about the iSAQB Advanced Level

What is taught in an Advanced Level module?

- The iSAQB Advanced Level offers modular training in three areas of competence with flexibly designable training paths. It takes individual inclinations and priorities into account.
- The certification is done as an assignment. The assessment and oral exam is conducted by experts appointed by the iSAQB.

What can Advanced Level (CPSA-A) graduates do?

CPSA-A graduates can:

- Independently and methodically design medium to large IT systems
- In IT systems of medium to high criticality, assume technical and content-related responsibility
- Conceptualize, design, and document actions to achieve quality requirements and support development teams in the implementation of these actions
- Control and execute architecture-relevant communication in medium to large development teams

Requirements for CPSA-A certification

- Successful training and certification as a Certified Professional for Software Architecture, Foundation Level® (CPSA-F)
- At least three years of full-time professional experience in the IT sector; collaboration on the design and development of at least two different IT systems
  - Exceptions are allowed on application (e.g., collaboration on open source projects)
- Training and further education within the scope of iSAQB Advanced Level training courses with a minimum of 70 credit points from at least three different areas of competence
- Successful completion of the CPSA-A certification exam

Essentials

What does the module “ADOC” convey?

ADOC teaches the skills to:

- independently design and create software architecture documentation.
- evaluate existing documentation for suitability and adequacy.
- improve existing documentation.

Participants learn to systematically describe the essential aspects of software architectures (including decisions, structures, concepts, quality requirements and views). For this purpose, ADOC introduces basic description and notation methods as well as possible tools for practical implementation.

Curriculum Structure and Recommended Durations

Content	Minimum recommended duration (min)	Exercises (min)
1. Fundamentals	70	20
2. Process for architecture documentation	90	60
3. Elements of architecture documentation	90	90
4. Tools	90	30
5. Evaluating documentation	75	45
6. Examples of software architecture documentation	60	0
Total	475	245

Total duration: 720 min (12h)

Duration, Teaching Method and Further Details

The times stated below are recommendations. The duration of a training course on the ADOC module should be at least 2 days, but may be longer. Providers may differ in terms of duration, teaching method, type and structure of the exercises and the detailed course structure. In particular, the curriculum provides no specifications on the nature of the examples and exercises.

Licensed training courses for the ADOC module contribute the following credit points towards admission to the final Advanced Level certification exam:

Methodical Competence:	20 Points
Technical Competence:	0 Points
Communicative Competence:	0 Points

Prerequisites

Participants should have the following knowledge and/or experience:

- Fundamentals of describing architectures using different views, crosscutting concepts, design decisions, constraints, etc., as taught in CPSA- F Foundation Level.
- Own experience in the creation and maintenance of technical documentation of software, especially the architecture of software or software-related systems is desirable.

Helpful for the understanding of some concepts are furthermore:

- Knowledge of typical challenges in creating and maintaining technical documentation:
  - Selection of appropriate documentation structures, notations, deliverable types (stakeholder orientation)
  - Handling of large documentation (especially existing or obsolete documentation)
  - Selection, configuration and implementation of tool chains (by what means can documentation be adequately created and maintained),
  - Versioning of documents (models and texts)
  - Documentation in teams, creation and maintenance based on work distribution
  - Content-based and formal reviews of documentation.

Structure of the Curriculum

The individual sections of the curriculum are described according to the following structure:

- **Terms/principles**: Essential core terms of this topic.
- **Teaching/practice time**: Defines the minimum amount of teaching and practice time that must be spent on this topic or its practice in an accredited training course.
- **Learning goals**: Describes the content to be conveyed including its core terms and principles.

This section therefore also outlines the skills to be acquired in corresponding training courses.

1. Fundamental terms of architecture documentation

Lesson duration: 70 min	Exercises: approx. 20 min

1.1. Terms and concepts

Software architecture, documentation, stakeholder, notation, process model, solution design, UML.

1.2. Learning Goals

LG 1-1: Understand and explain the benefits and goals of architecture documentation

Software architects understand that architecture documentation is not an end in itself.

They can explain the benefits and various goals of architectural documentation, such as:

- Architecture documentation supports solution design, and communication of the solution to the team and others (e.g., clients)
- Support implementation by communicating architectural and development decisions, facts, and rationale
- Written communication of relevant structural and conceptual topics
- Preservation of knowledge about development and architecture

They can advocate these goals and the benefits of documentation to stakeholders in a convincing manner.

LG 1-2: Understand the importance of architecture documentation for different stakeholders

Software architects understand that different stakeholders usually have different requirements for architecture documentation, for example with regard to..:

- Types of information (e.g. about interfaces, building blocks, cross-sectional topics, decisions, technical risks, approaches to solutions, etc.)
- Representation of such information (as (graphical) models, as pure graphics or textual)
- Degree of detail (e.g. overview, abstract or detailed representation)
- technical representation (with/without writing/editing options, PDF, HTML, Wiki, word processing, etc.)
- Degree of timeliness (how *up-to-date* does documentation needs to be, up-to-the latest development/release/test version? minor/major version?)

LG 1-3: Differentiate architectural documentation from other types of documentation

Software architects understand that architecture documentation has points of contact and partial overlap with other types of documentation, for example:

- Requirements documentation
- Implementation documentation
- Management documentation
- Test documentation

- Operations documentation

LG 1-4: Choose appropriate notations for architecture documentation

Software architects are familiar with various methods of notation and representation of architectural information, for example:

- Modeling methods, such as UML, C4 ([Brown (C4)]), ArchiMate, Business-Process Modeling Notation (BPMN), Entity-Relationship (ER) diagrams, Event-driven process chains (EPC)
- Textual explanations of decisions
- Canvas (e.g. Architecture Communication Canvas [ACC] or Bounded Context Canvas [DDD-Canvas])

They know that more formal notations such as UML can be helpful for architecture documentation, but are not mandatory.

LG 1-5: Adapt documentation efforts to different development processes

Software architects know that

- in more formal development projects (e.g. *safety-critical*) high demands on architecture documentation apply.
- in more lightweight development processes (agile processes) teams should choose a type of documentation appropriate to the system and its environment.
- depending on the process and the constraints of the project, very different documentation work has to be carried out.

2. Process for architecture documentation

Lesson duration: 90 min	Exercises: 60 min

2.1. Terms and concepts

Process model, role, artifact, iterative/incremental, life cycle, agile.

2.2. Learning Goals

LG 2-1: Create architecture documentation

Software architects can create appropriate architecture documentation for medium to large IT systems in situations such as the following:

- Building new systems
- Ongoing development of existing systems with existing documentation
- Post-documentation of already existing systems without available documentation
- Documentation in extraordinary situations (little budget, little time, few available sources of information, etc.)

They can identify target groups and goals of the documentation (see LG 2-2).

Details on the necessary methods can be found in the learning goals LG 3-1 to LG 3-5.

LG 2-2: Identify target audiences and goals of architectural documentation

Software architects can identify the target audiences ("stakeholders") for architecture documentation (e.g. management, development team, business stakeholders, operations stakeholders, test/QA).

They can work together with these stakeholders to identify specific goals for the documentation.

LG 2-3: Define a suitable structure for architecture documentation

Software architects are able to select or develop a suitable structure for architecture documentation for given situations and systems.

Target audiences and goals (see LG 2-2) and templates (see LG 2-5) serve as the basis for this.

LG 2-4: Define a suitable process for architecture documentation

Software architects find a suitable documentation procedures for given situations. They can explain the advantages and disadvantages of different approaches to stakeholders and the development team, such as:

- Documentation is created **alongside** development work
- Documentation is created **before** development/implementation
- Documentation is created **after** development ("documentation sprint")
- Documentation is created and maintained by people **from the development team**.
- Documentation is created and maintained by people **outside the development team**.

They know that agile development and architecture documentation are not in conflict and can justify this to stakeholders.

LG 2-5: Identify application scenarios for GenAI in documentation

Software architects know where and how the use of GenAI tools can add value in the creation, maintenance, and use of technical documentation. This includes, for example:

- Generating documentation from code artifacts
- Identifying and securing architecture-relevant requirements
- Comparing existing documentation with source code
- Using AI models as a replacement for static documentation
- Extracting information from unstructured or weakly structured sources (e.g., emails, chat histories, tickets) and linking it to existing content

LG 2-6: Know and select suitable documentation templates for a given context

Software architects know examples of architecture templates, such as:

- arc42 ([arc42],[Starke+2023],[Starke+2016])
- C4 ([Brown (C4)], [Brown (Guidebook)])
- ISO/IEE 42010 ([ISO/IEC 42010:2022] „Recommended Practice for Architecture Description of Software-Intensive Systems“)

3. Elements of architecture documentation

Lesson duration: 90 min	Exercises: 90 min

3.1. Terms and concepts

Views, decisions, cross-cutting concepts and topics, interfaces, structures, constraints, risks, quality goals.

3.2. Learning Goals

LG 3-1: Document influences and constraints

Software architects can identify factors that influence software architecture and document them appropriately. These include:

- Constraints
- Quality requirements and goals
- Technical risks

LG 3-2: Document architecture decisions

Software architects can record architectural decisions in a comprehensible manner. For example, they are familiar with *Architecture Decision Records* (see [Nygard 2011], [ADR-GitHub]) as a fixed structure for such decisions.

LG 3-3: Apply different views to document architectures

Software architects can use and document different architecture views appropriately. They know for which use cases static and dynamic views are best suited. They also understand when which level of detail is appropriate.

They are familiar with (and can apply) at least the following views:

- Building block view (also: component view) as a static structure
- Runtime view for describing interactions, sequences, or time conditions.
- Deployment view for describing technical infrastructure and its mapping to the source code components (see building block view) of the system

See also [arc42], [Starke+2016] und [Zörner 2021].

LG 3-4: Document cross-cutting concepts

Software architects can recognize concepts that go beyond module boundaries or cross-cut through the software, and document them accordingly.

LG 3-5: Ensure consistency between parts of the documentation

Software architects are able to establish and maintain consistency between individual parts of the documentation.

4. Tools

Lesson duration: 90 min	Exercises: 30 min

4.1. Terms and concepts

Analog and digital tools, modelling tools, tool chain.

4.2. Learning Goals

LG 4-1: Select appropriate tools for documentation work

Software architects can select appropriate tools for different activities:

- Creation and maintenance of parts of architectural documentation
- Parts management
- Communication of content with the support of the parts

They are able to:

- use analog and digital documentation tools according to the situation and requirements.
- comprehensibly select a complete tool chain for architecture documentation based on concrete requirements, constraints and other influences.

They are aware of respective strengths, weaknesses and typical challenges when using and integrating common tool categories, for example:

- Wikis
- Modeling tools
- Drawing software
- Word processors
- Version control systems
- Other (such as: blogs, issue trackers, etc.)

LG 4-2: Define requirements for documentation tools and artifacts

Software architects know how tools can support them in various activities related to architecture documentation.

They know that documentation:

- should be managed in a versioned and release-ready manner similar to source code.
- should be producible or generatable for every defined state of the software.

LG 4-3: Know about Docs-As-Code Approach

Software architects are familiar with the Documentation-As-Code approach, where architectural documentation is generated in whole or in part from source code-like artifacts (e.g. Markdown or AsciiDoc).

They are familiar with tools that can also be used to generate graphical representations from textual descriptions (e.g. [PlantUML]).

LG 4-4: Apply GenAI tools to support architecture documentation

Software architects are aware of the opportunities and risks involved in using generative AI (GenAI) tools for documentation purposes. They can apply these tools in a targeted manner for individual sub-areas (e.g., architecture-related requirements, solution description, evaluation of documentation) and are able to validate the results.

5. Evaluating documentation

Lesson duration: 75 min	Exercises: 45 min

5.1. Terms and concepts

Review, checklists, questionnaires.

The learning objectives in this section deal with the examination of architectural documentation for its usability. This is an essential success factor for the practical use of documentation.

5.2. Learning Goals

LG 5-1: Ensure documentation is fit for its purpose

Software architects recognize checking for fitness for purpose as a key success factor of documentation.

See especially LG 1-2: Understand the importance of architecture documentation for different stakeholders.

LG 5-2: Differentiate different types of assessment

Software architects can differentiate between content and formal assessment of architectural documentation.

LG 5-3: Distinguish between a documentation review and an architecture review

Software architects can distinguish reviewing documentation from evaluating the architecture or the system.

LG 5-4: Identify and explain goals of documentation reviews

Software architects can identify possible goals of documentation reviews with the concerned persons ("stakeholders") and explain these goals to other persons.

LG 5-5: Create checklists and questionnaires for reviews

Software architects can create checklists and questionnaires for documentation reviews.

LG 5-6: Perform different roles in documentation reviews

Software architects can take on the following roles in the context of documentation reviews

- Deal with objections appropriately as an author,
- Provide constructive feedback to authors as reviewers,
- Lead a subject matter review as a facilitator.

LG 5-7: Improve the quality of documentation based on review results

Software architects can use review results to further develop and maintain existing architecture documentation and systematically improve the quality of this documentation.

6. Examples of software architecture documentation

Lesson duration: 60 min	Exercises: none.

At least one example of a documented software architecture must be presented within each accredited training course.

The type and characteristics of the examples presented may depend on the specific training or the demands of the participants and are not specified by iSAQB.

6.1. Learning Goals

LG 6-1: Explain pros and cons of sample architecture documentation

Software architects have learned about architecture documentation from at least one practical example.

They can explain advantages and disadvantages of presented examples.

See for example [arc42-by-Example].

References

This section contains references that are cited in the curriculum.

A

- [ACC] arc42 and Contributors: The Architecture Communication Canvas, online: https://canvas.arc42.org
- [ADR-GitHub] Various Authors: ADR GitHub Organization. Online: https://adr.github.io/ .
- [arc42] Gernot Starke, Peter Hruschka and Contributors: arc42 - (open-source) template for architecture documentation and communication. https://arc42.org and https://docs.arc42.org
- [arc42-by-Example] Gernot Starke, Michael Simons, Stefan Zörner, Ralf D. Müller, and Hendrik Lösch: arc42 by Example: Software Architecture Documentation in Practice. E-Book: https://leanpub.com/arc42byexample

B

- [Bass 2021] L. Bass, P. Clements und R. Kazman: Software Architecture in Practice. 4. Edition, Addison-Wesley 2021.
- [DDD-Canvas] DDD-Crew: The Bounded Context Canvas, online https://github.com/ddd-crew/bounded-context-canvas.
- [Brown (C4)] S. Brown: The C4 model for visualising software architecture. Leanpub https://leanpub.com/visualising-software-architecture
- [Brown (Guidebook)] S. Brown: The software guidebook. A guide to documenting your software architecture. Leanpub https://leanpub.com/documenting-software-architecture

C

- [Clements+2010] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers et al.: Documenting Software Architectures – Views and Beyond. 2. Edition 2010, Addison Wesley.

H

- [Hargis+2004] G. Hargis et al.: Quality Technical Information: A Handbook for Writers and Editors. Prentice Hall, IBM Press, 2004.

I

- [ISO/IEC 42010:2022] ISO/IEC: 42010:2022: Software, systems and enterprise — Architecture description. Available on https://www.iso.org/standard/74393.html.

K

- [Kruchten 1995] P. Kruchten: Architectural Blueprints – The 4-1 View Model of Architecture. IEEE Software November 1995; 12(6), p. 42-50.

N

- [Nygard 2011] M. Nygard: Documenting Architecture Decisions. Blog Post 2011

https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions. See also https://adr.github.io/

P

- [PlantUML] PlantUML: Open-Source toolset and language definition to create diagrams from textual descriptions. Online: https://plantuml.com/ **R**
- [Read2023] J. Read: Communication Patterns: A Guide for Developers and Architects. O'Reilly Media, 2023.

S

- [Starke 2024] G. Starke: Effektive Softwarearchitekturen - Ein praktischer Leitfaden. 10. Auflage 2024, Carl Hanser Verlag, München.
- [Starke+2023] G. Starke, P. Hruschka: arc42 in Aktion. Carl-Hanser Verlag, 2. Auflage 2022.
- [Starke+2016] G. Starke, P. Hruschka: Communicating Software Architectures: lean, effective and painless documentation. Leanpub https://leanpub.com/arc42inpractice

U

- [UML] Universal Modeling Language, homepage: https://www.uml.org. Provides both the standard plus a collection of resources for further information.

Z

- [Zörner 2021] S. Zörner: Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten. 3. Auflage 2021, Carl-Hanser Verlag.