Curriculum for

Certified Professional for Software Architecture (CPSA)® *Advanced Level*

Module ARCEVAL

Architecture Evaluation

2022.1-EN-20230413

International Software Architecture Qualification Board

Table of Contents

The abbreviation "e. V." is part of the iSAQB's official name and stands for "eingetragener Verein" (registered association), which describes its status as a legal entity according to German law. For the purpose of simplicity, iSAQB e. V. shall hereafter be referred to as iSAQB without the use of said abbreviation.

List of Learning Goals

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

CPSA-A®
Certified Professional for Software Architecture
Advanced Level

Essentials

Curriculum Structure and Recommended Durations

Content	Recommended minimum duration (minutes)
1. Basic terminology of architecture evaluation	60
2. Requirements in architecture evaluation	120
3. Scenario-based workshops	150
4. Evaluation of existing systems (or their parts)	60
5. Other evaluation methods	90
6. examples	90
Total	570 (9.5h)

Duration, Teaching Method and Further Details

The times stated below are recommendations. The duration of a training course on the ARCEVAL 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 ARCEVAL 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 prerequisite knowledge:

- Fundamentals of documenting software architectures using different views, cross-cutting concepts, design decisions, constraints, etc., as taught in CPSA Foundation Level.
- Desirable is own experience in the development and maintenance of software, especially the architecture of software or software-related systems.

Knowledge in the following areas may be **helpful** for understanding some concepts:

- Knowledge of typical challenges in the design of software architectures:
  - Selection of appropriate documentation structures, notations, result types (stakeholder orientation).
  - Joint work on fundamental concepts of the software to be created
  - Alignment of software architecture with requirements and constraints
  - Measuring and evaluating the quality of designs and decisions

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.

Supplementary Information, Terms, Translations

To the extent necessary for understanding the curriculum, we have added definitions of technical terms to the iSAQB glossary and complemented them by references to (translated) literature.

1. Basic terminology of architecture evaluation

Lesson duration: 30 min	Exercises: approx. 30 min

1.1. Terms and concepts

Software architecture, architecture evaluation, crosscutting concepts/concerns, architecture goals, non-functional and functional requirements for systems, influences, evaluation methods.

1.2. Learning Goals

LG 1-1: Understand the basics of software architecture

- Reason why architectural work is fundamental, how it has broad implications in the project, and that architecture decisions are difficult to revoke
- Reason why architecture work should be iterative and needs feedback loops
- Reason that the focus of software architecture is on quality attributes and not just pure functionality.
- Understand the necessity of group processes for the further development of software architectures

LG 1-2: Explain the benefits of architecture evaluation

- (Early) identification of key risks and problems with regard to the required quality attributes
- Methodical safeguarding of architecture decisions - ensuring the long-term traceability of the architecture.
- Qualified feedback on architecture decisions
- General promotion of communication and transparency
  - Lost trust of stakeholders can be regained
  - Contradictions in requirements are uncovered
  - Classification and weighting of technical and conceptual strengths and weaknesses
- Promotion of methodical architecture development and clean architecture documentation

LG 1-3: Name motives for for architecture evaluation

- Methodological reasons (as a usual part of the architecture process, …)
- Organizational reasons (strategic decisions, investments, …)
- Technical reasons (changes of technology, replacements, …)
- Uncertainty (lack of overview, operational blindness, …)

LG 1-4: Explain prerequisites for architecture evaluation

- Necessary artifacts of software architecture and their appropriate level of detail in concrete project situations.
- Constraints
- Concrete quality requirements

- Documented architecture decisions
- Architecture overview
- Stakeholder willingness to cooperate
- Willingness of the organization to allocate budget and time to the evaluation process

LG 1-5: Differentiate various methods and approaches to architecture evaluation

- Scenario-based evaluation methods (ATAM, Lightweight ATAM, CBAM, …).
- Decision-based methods (DCAR, PBAR, …)
- Expert reviews (TARA, informal, …)
- Quantitative, measuring techniques
- Conformance analyses (architecture, code)

LG 1-6: Understand that different methods complement each other and can be combined

1.3. References

[Clements+2002], [Kazman+2000], [Bachmann+2000], [Bass+2003], [Clements+2003], [Kruchten 1995], [Starke 2011]

2. Requirements in architecture evaluation

Lesson duration: 60 min	Exercises: approx. 60 min

2.1. Terms and concepts

Quality, quality attributes, ISO 9126, ISO 25010, scenarios, utility tree, trade-offs, tactics.

2.2. Learning Goals

LG 2-1: Understand the relationship between software quality and software architecture

- Explain the importance of quality requirements for architecture.
- Clarify the terms "quality" and "quality attribute".
- Interpret different quality models (e.g. DIN/ISO 25010).
- Explain characteristics of important quality attributes:
  - Basic approach to dealing with them.
  - Interrelationship and interactions of quality attributes.

LG 2-2: Elicitate quality scenarios

- Distinguish types of scenarios from each other:
  - Use case scenarios
  - Growth scenarios
  - Exploratory scenarios
- Apply elicitation techniques to scenarios:
  - Brainstorming / Quality Storming
  - Interview
  - Derivation from NFR specification
- Explain possible components of scenarios for refinement and concretion.

LG 2-3: Document quality scenarios

- Describe placement of scenarios in architecture documentation.
- Work with utility trees:
  - Explain their usefulness.
  - Create utility trees.
  - Use them to prioritize quality requirements.

LG 2-4: Be able to select appropriate architectural views for evaluation

- Understand familiar views of software architectures:
  - context view

  - building block view
  - runtime view
  - deployment view
- Explain importance of views in the course of different evaluation methods.
- Be able to assess usability of existing views for evaluation.

LG 2-5: Explain different effects of the terms "architectural approach" and "architectural decision" on the attitudes of participants and evaluators in evaluation workshops.

LG 2-6: Select appropriate forms of decision documentation for the evaluation

- ADRs
- Key points
- Concepts
- Source code fragments
- Measurements and metrics

LG 2-7: Understand constraints

- Know common categories of constraints:
  - Technical constraints
  - Organizational constraints
  - Conventions
- Explain differences between constraints and quality requirements:
  - Constraints are met or not met (binary).
  - Quality requirements can be partially fulfilled or even overfulfilled

2.3. References

[Bass+2003], [Clements+2002], [Kazman+2005], [Barbacci+2003], [Starke 2011]

3. Scenario-based workshops

Lesson duration: 60 min	Exercises: approx. 90 min

3.1. Terms and concepts

ATAM, trade-offs, risks, non-risks, sensitive points, decisions, framework, phases of ATAM, stakeholder participation, scenario-based walk-through, assessment workshop, discovery review, interacting with development, risk clusters, mitigation measures, communicating results, reports, presentations, strengths and weaknesses, traffic light scores.

3.2. Learning Goals

In this part, participants get to know the core of the methodological approach for the qualitative evaluation of software architectures.

LG 3-1: Understand the ATAM (Architecture Tradeoff Analysis Method)

- Be able to prepare evaluation workshops according to the ATAM in a suitable way.
- Explain the phases of the ATAM
- Explain the steps of the core phases of the ATAM (Phase 1 and 2) and adapt them to the needs of your own project.
- Identify participants in each phase
- Know inputs for ATAM workshops and assign them to the steps

LG 3-2: Perform qualitative evaluation of software architectures according to the ATAM

- Explain and perform procedures for qualitative assessment
- Explain and elicitate quality scenarios and utility trees
- Analyze approaches with regard to quality scenarios
- Be able to apply questioning techniques to guide analysis in breadth or in depth and maintain focus
- Identify risks, non-risks, sensitivity points and trade-off points and be able to record them in an understandable way
- Justify the importance of facilitation

LG 3-3: Derive direct and condensed result types

- Risk clusters and risk groups
- Strengths and weaknesses of an architecture
- Gaps to the central quality objectives

LG 3-4: Describe communication tools for communicating results

- Structured reports and their important parts
- Structured presentation of results

- Recommendations and their classification into gaps and potentials

LG 3-5: Understand influences of evaluation results on iterative processes

- actively address risks
- plan open decisions
- reduce uncertainties

LG 3-6: Adapt the evaluation process to your own context

- Understand the supplemental effect of quantitative techniques.
- Know methods to streamline the process (e.g. light weight ATAM).

3.3. References

[Clements+2002], [Bass+2003], [Bass+2006], [Kazman+2000], [Nord+2003], [Kazman+2005], [Starke 2011]

4. Evaluation of existing systems (or their parts)

Lesson duration: 40 min	Exercises: approx. 20 min

4.1. Terms and concepts

Metrics, measurements, mathematical models, implementation check, current architecture, target architecture, pain points, tools.

4.2. Learning Goals

LG 4-1: Be able to evaluate existing system (or parts thereof)

- Name differences from the evaluation of concepts and ideas.
- Know current software tools and categorize them by purpose.
- Emed measurements and dynamic checks in the development process (for example, as part of the check-in process, in the build, etc.).

LG 4-2: Understand how to evaluate the implementation of architectural approaches

- Evaluate software architectures with regard to their implementation, i.e. answer the question to what extent an architecture has also been implemented in the code.
- Name tools for the comparison of implementation and architecture ideas (e.g. Sonargraph, Structure101, ArchUnit).
- Assess the suitability of the tools for collecting bottom-up as-is architectures and performing comparisons with target architectures.

LG 4-3: Use metrics for evaluation

- Explain metrics for reviewing design best practices. (e.g., distance, fan-in/fan-out, cyclomatic complexity).
- Plan use of metrics (to fulfill requirements, not for the sake of using a tool).
- Interpret measurements (question default settings, look for trends).
- Understand that measurements are appropriate starting points for deeper analysis.

LG 4-4: Plan the use of dynamic analysis methods

- Name the potentials of dynamic software analysis.
- Know quality attributes suitable for dynamic analysis.
- Recreate approaches to static analysis in highly distributed environments (microservices) using dynamic means.
- Explain incorporation of these techniques into larger evaluation projects.

4.3. References

[DeMarco+2006], [Martin 2000], [Sonarqube], [Hello2morrow]

5. Other evaluation methods

Lesson duration: 45 min	Exercises: approx. 45 min

5.1. Terms and concepts

Cost-benefit assessments (CBAM), decision-based methods (DCAR), expert-based methods (TARA), and other evualuation options (PBAR, ARID, pre-mortems, …).

5.2. Learning Goals

LG 5-1: Be able to classify cost-benefit analysis as a targeted measure

- Know the Cost-Benefit Analysis Method (CBAM).
- Understand the different use of quality scenarios compared to ATAM.
- Be able to assess and explain the applicability and accuracy of CBAM.

LG 5-2: Understand the broad outlines of DCAR (Decision Centric Architecture Reviews)

- Know the use of ADRs (Architecture Decision Records) in DCAR.
- Understand basic concepts of DCAR (Decision Force, Dependencies Relation Diagram, …)
- Understand the applicability of DCAR in agile and iterative processes
- Understand the difference in applicability compared to other methods (like ATAM)

LG 5-3: Understand the broad outlines of the TARA (Tiny Architecture Review Approach)

- Be able to name the (dis-)advantages of an evaluation by experts.
- Classify ATAM elements in TARA and understand the context.

LG 5-4: Understand the broad outlines of PBAR (Pattern-based Architecture Reviews)

- Understand the applicability of PBAR
- Be able to name strengths/weaknesses of Pattern Based Analysis.

LG 5-5: Understand the broad outlines of ARID (Active Review for Intermediate Designs)

- Be able to explain the Active Review approach
- Be able to classify applicability, effort and benefit of ARID

LG 5-6: Be able to structure larger evaluations

- Be able to identify factors that lead to more elaborate evaluations.
- Be able to combine basic elements of qualitative and quantitative evaluation methods to support larger, structured, and targeted evaluations.
- Be able to estimate staffing requirements on both the reviewer and participant sides.

LG 5-7: Be able to explain ideas for quick software architecture reviews

- Understand ways to streamline quality-focused reviews.
- Be able to name risk collections and brainstorming approaches
- Understand the scaling of small evalutations into more informed reviews and be able to explain the transition

5.3. References

[DeMarco+2006], [Martin 2000], [Sonarqube], [Hello2morrow]

6. Examples

Lesson duration: 90 min	Exercises: none

This section is not examinable.

6.1. Terms and concepts

In every licensed training session, at least one example for ARCEVAL must be presented.

Type and structure of the examples presented may depend on the training and participants' interests. The are not prescribed by iSAQB.

6.2. Learning Goals

Presentation, possibly elaboration and evaluation of at least one example of a software architecture.

LG 98-1: What do participants need to understand?

The process of evaluation workshops and the evaluation possibilities that arise with realistic architectures.

LG 98-2: What do participants need to know?

Some examples of (as close as possible) realistic evaluation inputs and evaluation results:

- Scenarios
- Documented constraints
- Documented decisions
- Architecture overview
- Pain points and sensitive points
- Documented trade-offs, risks and non-risks

6.3. References

None. Training providers are responsible for selecting and describing examples.

References

This section contains references that are cited in the curriculum.

B

- [Bachmann+2000] Bachmann, F., L. Bass, et al.: Software Architecture Documentation in Practice. Software Engineering Institute, CMU/SEI-2000-SR-004.
- [Barbacci+2003] Barbacci, M.R., Ellison, R., et al.: Quality Attribute Workshops (QAWs), Third Edition. Software Engineering Institute, CMU/SEI-2003-TR-016.
- [Bass+2003] Bass, L., Clements, P. und Kazman, R. (2003): Software Architecture in Practice. Addison-Wesley, Reading, Mass.
- [Bass+2006] Bass, L., Nord, R., et al.: Risk Themes Discovered Through Architecture Evaluations. Software Engineering Institute, CMU/SEI-2006-TR-012.

C

- [Clements+2002] Clements, P., R. Kazman, M. Klein: Evaluating Software Architectures – Methods and Case Studies.Addison Wesley, 2002.
- [Clements+2003] Clements, P., F. Bachmann, L. Bass, D. Garlan, J. Ivers et al: Documenting Software Architectures – Views and Beyond. Addison Wesley, 2003.

D

- [DeMarco+2006] DeMarco, T.: Controlling Software Projects: Management, Measurement and Estimation. Prentice Hall, 1986.

H

- [Hello2morrow] Hello2Morrow, online: http://www.hello2morrow.com/

K

- [Kazman+2000] Kazman, R., Klein, M., Clements, P.: ATAM: Method for Architecture Evaluation. Software Engineering Institute, CMU/SEI-2000-TR-004.
- [Kazman+2005] Kazman, R., Bass, L.: Categorizing Business Goals for Software Architectures. Software Engineering Institute, CMU/SEI-2005-TR-021.
- [Kruchten 1995] Kruchten, P.: Architectural Blueprints – The 4-1 View Model of Architecture. IEEE Software November 1995; 12(6), p. 42-50.

M

- [Martin 2000] Martin, R.C.: Design Principles and Design Patterns. White-Paper, 2000.

N

- [Nord+2003] Nord, R.L., Barbacci, M.R., et al.: Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM). Software Engineering Institute, CMU/SEI-2003-TN-038.

S

- [Sonarqube] Sonarqube, online: https://www.sonarqube.org/
- [Starke 2011] Starke, G. (2011): Effektive Softwarearchitekturen - Ein praktischer Leitfaden. 5. Auflage 2011, Carl Hanser Verlag, München.