Curriculum for

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Module
AGILA

Agile Software Architecture

2025.1-rev2-EN-20250714

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.

Learning Goals Overview

- LG 1-1: Knowing and being able to explain the significance of agile ideas for architecture work
- LG 1-2: Knowing the tasks involved in architecture development and how they are modified in the agile environment
- LG 1-3: Being able to appropriately align architecture work to the specific problem and project
- LG 2-1: Being able to structure architecture work in an agile way to provide a vision
- LG 2-2: Being able to continuously and iteratively evolve architectural work
- LG 2-3: Knowledge of role models for architects in agile projects
- LG 2-4: Knowledge of ways of involving stakeholders in architecture work
- LG 3-1: Being able to formulate quality requirements appropriately for specific target groups
- LG 3-2: Being able to use agile concepts for architecture requirements
- LG 3-3: Being able to apply iterative approaches for continuous identification of architecture requirements
- LG 3-4: Being able to prepare architecture work for refinements
- LG 3-5: Being familiar with urgency as a driver for architectural work and being able to explain it
- LG 4-1: Being able to use methods for making decisions in groups
- LG 4-2: Being able to support groups and teams in reaching decisions
- LG 4-3: Being able to create the necessary prerequisites for team decisions
- LG 4-4: Being able to understand and make architecture decisions as a process and just-in-time
- LG 4-5: Being familiar with ways of communicating architecture decisions in agile projects
- LG 4-6: Being able to document architecture decisions
- LG 5-1: Being familiar with techniques for joint reflection on architecture decisions
- LG 5-2: Being able to find the causes of architecture problems in a targeted manner
- LG 5-3: Being familiar with feedback opportunities from implementation and relating them to architectural goals
- LG 5-4: Being able to understand fitness functions for continuous architecture validation
- LG 6-1: Being familiar with ways to establish architecture as a cross-cutting concern in agile organizations
- LG 6-2: Being familiar with the forces in architectural work between centralization and decentralization
- LG 6-3: Being able to classify and reflect on central architectural structures in organizations
- LG 6-4: Being familiar with the relationship between architectural styles and agile architectural work
- LG 6-5: Being able to establish architectural concepts that promote local decision-making capabilities
- LG 6-6: Being able to accompany architectural decisions in an advisory capacity
- LG 6-7: Being familiar with socio-technical systems and consider them in architectural work
- LG 7-1: Being familiar with and understanding examples for decision-making procedures in agile projects

- LG 7-2: Being familiar with and understanding examples for agile architecture requirements
- LG 7-3: Being familiar with physical characteristics of agile communication concepts
- LG 7-4: Being able to understand the postponement of architecture decisions
- LG 7-5: Being familiar with and understanding examples of agilely organised architecture groups

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 “AGILA” convey?

In this module, participants learn how to design, evolve, and collaboratively take responsibility for software systems and architectures in line with agile principles. The module covers how to apply agile thinking and working models to architectural work, as well as how to effectively and in a lightweight way integrate architectural practices into agile approaches.

It provides the skills and techniques needed to conduct architectural work in self-organized teams – including technical, methodological, and communicative competencies. Topics include collaborative decision-making, continuous architecture evaluation, stakeholder involvement, and the design of enabling roles and structures in agile organizations. These are addressed both theoretically and through practical exercises.

Curriculum Structure and Recommended Durations

Content	Recommended minimum duration (minutes)
Basics	90
Agile approach to architecture	120
Architecture requirements in agile projects	210
Designing architectures in a team	300
Reflection and feedback	120
Supporting Architecture (spanning multiple teams)	120
Examples of agile architecture work	120
Total	1080min = 18h

Duration, Teaching Method and Further Details

The times stated below are recommendations. The duration of a training course on the AGILA module should be at least 3 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 AGILA module contribute the following credit points towards admission to the final Advanced Level certification exam:

Methodical Competence:	20 Points
Technical Competence:	00 Points
Communicative Competence:	10 Points

Prerequisites

Participants **should** have the following prerequisite knowledge:

- Practical experience in the design and development of small to medium-sized software systems.
- Practical experience with the preparation, documentation and communication of architecture

decisions.

- Initial practical experience in agile software projects.

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

- Knowledge of agile procedure models:
  - The Agile Manifesto [AgileM2001]
  - Scrum [Schwaber2013]
  - Lean/Kanban [Anderson2010], [Kniberg2011].
- Knowledge and initial practical experience in the development and definition of architecture requirements, and interaction with the different stakeholders of development projects.
- Initial practical experience or knowledge in the area of automated integration and delivery of software:
  - Knowledge and initial experience in automated testing at the unit and integration level
  - Basic knowledge of runtime analysis of software, for example profiling, tracing, log analysis, data analysis
  - Basic knowledge of automated Build, Integration and Delivery processes for software.

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. Introduction to agile software architecture

Duration: 90 min

1.1. Terms and Principles

- Software architecture, general
- The Agile Manifesto: Values and principles
- Cynefin-Framework
- 5 Revolutions of Software Engineering
- Cargo Cult Agility
- Agile procedure models incl. Scrum and Lean
- The agile approach to software architecture
- Anchoring software architecture in agile practices/tools

1.2. Learning Goals

LG 1-1: Knowing and being able to explain the significance of agile ideas for architecture work

- Knowing definitions of software architecture:
  - Which definitions are suitable for agile environments?
  - How is architecture distinguished from design?
- Knowing the agile world view and principles:
  - Cynefin framework and complexity as a factor influencing process models
  - Relevant value pairs and principles of the Agile Manifesto (e.g., “Responding to change over following a plan”)
  - Recognizing and critically reflecting on typical misapplications (e.g., Cargo Cult Agility)
- Understanding the interaction between architecture and agility:
  - Synergies between architecture and agility
  - Influences of Scrum, Lean, and agile thinking models on architectural decisions
  - Core mindset of agile architecture work: early feedback, collaborative decision-making, evolutionary development through hypotheses

LG 1-2: Knowing the tasks involved in architecture development and how they are modified in the agile environment

- Clarification of how the following architecture development tasks defined in the CPSA Foundation Curriculum are perceived in the agile environment:
  - Clarification, scrutiny and where necessary refinement of requirements and conditions/constraints.
  - Making structure decisions in respect of system decomposition and module structure, and defining dependencies and interfaces between the modules.
  - Defining and where necessary implementing general technical concepts (for example

persistence, communication, GUI).

- Communicating and documenting software architecture based on views, architectural patterns and technical concepts.
- Supporting the realisation and implementation of the architecture, where necessary incorporation of feedback from the involved stakeholders into the architecture, and checking and ensuring consistency of the source code and the software architecture.
- Reviewing software architectures, in particular in terms of risks in respect of the specified quality criteria.
- Little change to the topics covered by architecture work, but often changes in the methods used when addressing architecture issues (allocation of tasks, decision making, communication of the results etc.).
- Mapping of the most important changes to the contents of this curriculum.

LG 1-3: Being able to appropriately align architecture work to the specific problem and project

- Tension between rapid responsiveness and sufficient planning or manageability:
  - In the extreme case: tension between low responsiveness (rigidity) and poor controllability (chaos)
- Just enough architecture work and supporting concepts – limiting effort to relevant problems:
  - Criteria for evaluating architectural relevance and targeted application of architectural activities
  - Applying Lean principles (e.g., waste reduction) to architectural work
  - Focusing communication

2. The agile architecture approach

Duration: 120 min	

2.1. Terms and Principles

- Iterative, incremental architecture development: Risk-driven architecture work and release planning
- Architecture work with Scrum and Kanban
- Architecture Vision and Architecture Communication Canvas
- Walking Skeleton
- Agile architecture roles
- Stakeholder involvement.

2.2. Learning Goals

LG 2-1: Being able to structure architecture work in an agile way to provide a vision

- Explaining the purpose and boundaries of an architecture vision in an agile context
- Developing a lightweight architectural idea (e.g., Architecture Vision or Architecture Communication Canvas)
- Planning a Walking Skeleton with broad architectural coverage and being able to differentiate it from a Minimal Viable Product
- Making initial risks visible and mitigable using architecture vision and Walking Skeleton

LG 2-2: Being able to continuously and iteratively evolve architectural work

- Risk-driven architecture work: Architectural risk as the determining factor for using architectural practices
- Assessing the urgency of architectural issues and deferring decisions when appropriate
- Integrating architectural topics into backlogs and dynamically prioritizing them
- Using hypothesis-driven approaches to (in)validate architectural options
- Optional: Embedding architectural work into Kanban systems and managing it using flow metrics (e.g., WIP limits, lead time)

LG 2-3: Knowledge of role models for architects in agile projects

- Selecting appropriate architectural role models depending on the context. For example:
  - Architecture Owner / Supporting Architect as enabler, coach, and moderator
  - Architecture champions
  - Self-organized teams without a designated architect role
- Understanding how responsibilities are shared within a Scrum team (PO, SM, Dev Team) in the context of architectural tasks
  - Describing the collaboration between development teams (and possibly architects) and Product Owners for clarifying and prioritizing architecture topics

LG 2-4: Knowledge of ways of involving stakeholders in architecture work

- Creating methodical points of interaction for collaboration
- Motivating stakeholders to contribute to architectural work
- Making architectural aspects and their impact understandable for business stakeholders, e.g., using visualizations, storytelling, or canvas models.

3. Architecture requirements in agile projects

Duration: 210 min	

3.1. Terms and Principles

- Collaboration with customers and other stakeholders: Explanation of communication levels and architecture problems appropriately for specific target groups, translation of requirements and wishes into architecture concepts
- Quality attributes and ISO/IEC 25010
- Quality requirements in agile processes
- Technical debt
- Architectural risks incl. their determination
- Requirements management; refinements, evaluation and prioritisation of architecture requirements
- Urgency of architecture issues and the last responsible moment.

3.2. Learning Goals

LG 3-1: Being able to formulate quality requirements appropriately for specific target groups

- Naming and differentiating quality attributes (e.g., maintainability, performance, security) based on ISO/IEC 25010
- Formulating quality requirements at different levels of abstraction (e.g., as quality goal, quality scenario, acceptance criterion, …)
- Expressing architectural problems and technical trade-offs in qualitative statements

LG 3-2: Being able to use agile concepts for architecture requirements

- Supplementing stories with qualitative aspects (acceptance criteria)
- Anchoring architecture requirements in backlogs (e.g., as quality story, enabler story, or technical task), and understanding advantages and disadvantages of these approaches
- Structuring and slicing architecture topics so they can be addressed in agile iterations
- Using the Definition of Done (DoD) for qualitative objectives

LG 3-3: Being able to apply iterative approaches for continuous identification of architecture requirements

- Ongoing collaboration with customers (or their representatives):
  - Highlighting the value for business stakeholders
  - Techniques for effective and regular involvement of business stakeholders
- Using refinements, reviews, and risk workshops correctly
- Identifying, recording, and processing technical debt on the architectural level
- Using static analyses, tests, and monitoring data as sources for new architectural tasks

LG 3-4: Being able to prepare architecture work for refinements

- Regularly reflecting, refining, and reprioritizing architecture requirements
- Using appropriate techniques for joint estimation (e.g., T-Shirt Sizing, Magic Estimation, Risk-Based Sizing)
- Applying methods for prioritizing architectural topics
- Being able to plan refinement activities and related preparation work

LG 3-5: Being familiar with urgency as a driver for architectural work and being able to explain it

- Interpreting late architectural decisions and open options as value (Real Options Theory)
- Identifying signals for urgently required architectural decisions (e.g., increasing implementation effort, integration issues)
- Reflecting on the balance between “decide early” and “delay deliberately”

4. Designing and developing architectures in a team

Duration: 300 min	

4.1. Terms and Principles

- Group decision-making procedures: Consensus decisions, veto procedure
- Moderation of decision-making
- Design integrity in the case of multiple decision makers
- Vertical architecture styles (in contrast to system-wide layering)
- Architekturentscheidungen as a process
- Just-in-Time-Decisions & Set-Based Design
- Architecture principles
- Architecture wall
- Architecture Decision Records (ADRs)

4.2. Learning Goals

LG 4-1: Being able to use methods for making decisions in groups

- Methodological fundamentals for effectively guiding (equal) group members to decisions:
  - Voting and veto procedures
  - Consensus decision-making
  - Separation of decision preparation and final decision
  - Understanding the dynamics of delegated decision-making authority
  - Use in conjunction with agile methodologies (e.g., Scrum)
  - Group dynamics and role clarity in the architectural context

LG 4-2: Being able to support groups and teams in reaching decisions

To reach effective decisions in groups, methodological and communicative aspects are important: - Methodically supporting and moderating decision-making processes in groups - Constructively dealing with objections and resistance - Structuring discussions in a goal-oriented way (e.g., through structured decision-making formats, facilitation techniques)

LG 4-3: Being able to create the necessary prerequisites for team decisions

- Clearly defining roles, responsibilities, and decision-making scopes
- Establishing architectural principles as guardrails for consistent decisions
- Identifying and designing technical preconditions for local architectural decisions
- Naming risks of distributed decision authority (e.g., redundancy, inconsistencies)

LG 4-4: Being able to understand and make architecture decisions as a process and just-in-time

- Understanding architecture decisions as an iterative decision-making process (open → in progress →

decided → deprecated)

- Correctly applying the concept of the last responsible moment
- Technical options for postponing architectural decisions
- Splitting architectural issues into sub-decisions
- Techniques for addressing urgent and high-risk decisions (e.g., Set-Based Design)

LG 4-5: Being familiar with ways of communicating architecture decisions in agile projects

- Making architecture decisions visible in meetings and informal settings (e.g., architecture wall, team stand-ups, reviews)
- Explaining the benefit of lightweight evaluation formats (see also Chapter 5)
- Recognizing when formalized, documented communication is required

LG 4-6: Being able to document architecture decisions

- Explaining the purpose and structure of Architecture Decision Records (ADRs)
- Using ADRs to support transparency, traceability, and team participation
- Understanding differences to traditional architectural documentation (lightweight, focused, incrementally created)
- Being familiar with options for introducing ADRs within teams (e.g., wiki, Markdown, arc42)

5. Reflection and feedback on architecture work in the agile context

Duration: 120 min	

5.1. Terms and Principles

- Lightweight architecture evaluation and other feedback mechanisms (DCAR, LASR)
- Root cause analysis
- Testing of qualitative system attributes
- Quality indicators and metrics
- Fitness Functions

5.2. Learning Goals

LG 5-1: Being familiar with techniques for joint reflection on architecture decisions

- Selecting and applying architecture evaluation formats that are iterative and lightweight in nature:
  - DCAR – Decision Centric Architecture Reviews – for ADR-based, iterative reviews
  - LASR – Lightweight Approach for Software Reviews – as lightweight risk-focused sessions
  - Reality check for architecture goals to identify architectural gaps
- Moderating and structuring small, iterative reviews
- Formulating constructive and focused feedback on architectural ideas (feedback rules, moderation techniques)

LG 5-2: Being able to find the causes of architecture problems in a targeted manner

- Performing cause analysis in groups (e.g., using root cause analysis, 5-Whys)
- Distinguishing symptoms, causes, and effects of architectural problems
- Establishing the connection between test or measurement results and architecture decisions

LG 5-3: Being familiar with feedback opportunities from implementation and relating them to architectural goals

- Testing of qualitative attributes: techniques, tools, and integration into development activities
- Selecting and using static quality indicators
- Performing trend analyses on quality indicators and deriving insights
- Leveraging the advantages of Continuous Integration and Continuous Deployment in this context

LG 5-4: Being able to understand fitness functions for continuous architecture validation

- Explaining the concept of fitness functions
- Using the Fitness Function Quadrant to select appropriate evaluation strategies (holistic vs. atomic, triggered vs. continuous)
- Identifying tools and techniques for implementation (e.g., ArchUnit, jQAssistant, Chaos Mesh, linters, metric dashboards)

- Integrating fitness functions into CI/CD processes, deployments or runtime environments to receive automated feedback on architectural quality
- Naming the benefits of continuous quality feedback for architectural work (within the team)

6. Supporting Architecture (spanning multiple teams)

Duration: 120 min	

6.1. Terms and Principles

- Enabling Architecture: Architects as enablers, not decision-makers
- Centralization vs. decentralization, governance vs. autonomy
- Architecture Advice Process
- Architecture boards, communities of practice, advice forums
- Architecture radar, architectural principles
- Path of least resistance
- Socio-technical systems, Conway's Law

6.2. Learning Goals

LG 6-1: Being familiar with ways to establish architecture as a cross-cutting concern in agile organizations

Establishing architecture as a base capability of developers. For example through: - Open, cross-project communication of best practices - Building and maintaining architecture-related communities of practice - Using architecture katas as a way to broaden architectural knowledge - Communicating and transferring architectural responsibility

LG 6-2: Being familiar with the forces in architectural work between centralization and decentralization

- Tension between centralized control and decentralized decision-making autonomy
- Connecting governance approaches with agile thinking (principles instead of rules)
- Actively shaping the boundary between macro and micro architecture
- Naming important factors for successful decentralization (broad understanding of goals, shared architectural understanding, timely feedback, transparency and cross-pollination, …)

LG 6-3: Being able to classify and reflect on central architectural structures in organizations

- Describing various central architectural structures (e.g., architecture board, advice forum, virtual architecture groups)
- Assessing their benefits, risks, and suitable application contexts
- Adapting formats for knowledge sharing, consulting, and standardization to the needs of agile organizations
- Differentiating between controlling and enabling structures

LG 6-4: Being familiar with the relationship between architectural styles and agile architectural work

- Explaining the influence of architectural styles on team autonomy and adaptability (e.g., microservices vs. layered architectures, self-contained systems vs. modular monoliths)

- Understanding how architectural style, organizational structure, and viable working models influence each other

LG 6-5: Being able to establish architectural concepts that promote local decision-making capabilities

- Identifying and introducing measures that promote local technical decisions
- Actively shaping a “path of least resistance”
- Assessing the risks of excessive autonomy (e.g., inconsistency, redundant logic, technological fragmentation)
- Developing suitable architectural principles and boundary conditions to create balance

LG 6-6: Being able to accompany architectural decisions in an advisory capacity

- Applying the basic idea and phases of the Architecture Advice Process
- Differentiating between decision preparation, accompaniment, and responsibility
- Selecting suitable formats for advisory architectural work (e.g., pairing, shadowing, reviews, consensus formats, ADR workshops)
- Acting as a coach instead of a decision-maker – focusing on enabling and self-responsibility of teams

LG 6-7: Being familiar with socio-technical systems and consider them in architectural work

- Describing the effect of team structures on architecture (and vice versa)
- Knowing team definitions according to Team Topologies and understanding how to support them architecturally in different ways
- Being able to reflect on system structure and team structure and identify approaches to alignment (e.g., with fracture planes)

7. Examples of agile architecture work

Duration: 120 min	

Note: Examples of real agile architecture work can be individually chosen by trainers and providers of training courses.Due to the confidentiality and non-disclosure stipulations that often apply to real-life IT systems, the iSAQB allows trainers and providers of training courses to use abstracts or excerpts of examples in courses.

7.1. Learning Goals

LG 7-1: Being familiar with and understanding examples for decision-making procedures in agile projects

Participants should be able to understand, based on at least two examples, how agile projects of different sizes make and communicate architecture decisions. Relevant meetings, roles and decision-making procedures should be described in trainings, and the specific embedding into the agile process of the respective project should be made clear.

LG 7-2: Being familiar with and understanding examples for agile architecture requirements

- Participants should be able to understand at least one example for the formulation of architecture requirements and their embedding into agile requirements artefacts (such as stories or backlogs).
- Participants should understand examples that illustrate joint work on architecture requirements in real projects (refinement, planning etc.).

LG 7-3: Being familiar with physical characteristics of agile communication concepts

Participants should be able to see and understand, based on examples, how agile projects communicate architecture information and decisions (photos of architecture walls, layouts of information radiators, examples of architecture radars etc.).

LG 7-4: Being able to understand the postponement of architecture decisions

Participants should, based on a real example, become familiar with and understand the postponement of architecture decisions, and see how the decision-making process can be made visible (e.g., using status labels such as "Open – In Discussion – Decided – Deprecated").

LG 7-5: Being familiar with and understanding examples of agilely organised architecture groups

Participants should be able to become familiar with and understand, based on at least two real examples, the organization of architecture communities or guilds, as well as the activities and techniques that companies use to establish and actively maintain these groups. Formats such as Architecture Advice Forums, peer reviews, radar workshops or architecture reviews should be vividly illustrated.

References

This section contains references that are cited in the curriculum.

A

- [AgileM2001] agilemanifesto.org – The Agile Manifesto. http://agilemanifesto.org/principles.html
- [Anderson2010] David Anderson: "Kanban", Blue Hole Press, 2010, Sequim, WA
- [Appelo2010] Jurgen Appelo: "Management 3.0: Leading Agile Developers, Developing Agile Leaders", Addison Wesley 2010

E - [] Murat Erder, Pierre Pureur, Eoin Woods: “Continuous Architecture in Practice”, Addison-Wesley 2021

F

- [Fairbanks2010] George Fairbanks: "Just Enough Software Architecture: A Risk-Driven Approach", Marshall & Brainerd 2010
- [Ford2017] Neal Ford, Rebecca Parsons, Patrick Kua: “Building Evolutionary Architectures”, O’Reilly Media 2017

H

- [Harmel-Law2024] Andrew Harmel-Law: "Facilitating Software Architecture", O’Reilly Media 2024

K

- [Kniberg2011] Henrik Kniberg: "Lean from the Trenches: Managing Large-Scale Projects with Kanban", Pragmatic Bookshelf 2011

L

- [Larman2008] Craig Larman, Bas Vodde: "Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum", Addison Wesley 2008

R

- [Richardson2005] Jared Richardson, William A. Gwaltney: “Ship It!: A Practical Guide to Successful Software Projects”, Pragmatic Bookshelf 2005
- [Richards2020] Mark Richards, Neal Ford: “Fundamentals of Software Architecture”, O’Reilly Media 2020

S

- [Skelton2019] Matthew Skelton, Manuel Pais: “Team Topologies: Organizing Business and Technology Teams for Fast Flow”, IT Revolution Press 2019
- [Schwaber2013] Ken Schwaber, Jeff Sutherland: "The Definitive Guide to Scrum: The Rules of the Game", http://www.scrumguides.org

T

- [Toth2025] Stefan Toth: "Vorgehensmuster für Softwarearchitektur: Kombinierbare Praktiken in Zeiten von Agile und Lean", 4. Auflage, Hanser Verlag 2025

V

- [Vigenschow2012] Uwe Vigenschow, Björn Schneider, Ines Meyrose: "Soft Skills für IT-Berater: Workshops durchführen, Kunden methodisch beraten und Veränderungen aktiv gestalten", dpunkt.verlag 2012