Curriculum for

CPSA Certified Professional for Software Architecture®

– Advanced Level –

Module:
AGILA

Agile Software Architecture

Version 1.2 (February 2015)

Table of contents

List of Learning Goals

0 Introduction

0.1 What is covered by the "AGILA" module?

In this module, the participants learn how to design, develop and further develop software systems and architectures in accordance with agile principles. On the one hand, the module covers the application of agile principles and concepts to architecture work, and on the other hand expedient anchoring of architecture practices in an agile approach.
The development of architectures in projects with self-sufficient teams or shared responsibilities demands new skills and capabilities on the part of developers and architects. These in turn cover technical as well as methodical and also communicative aspects, which are addressed here all theoretically and in practical exercises.

0.2 What does an Advanced Level Module teach?

- The iSAQB Advanced Level offers modular training in three areas of competence with flexible academic approaches. It considers individual leanings and focuses.
- The certification is achieved by writing a term paper. Experts designated by the iSAQB perform the assessment and administer the oral examination. Details can be found here: http://isaqb.org.

0.3 What capabilities do graduates of the Advanced Level (CPSA-A) acquire?

CPSA-A graduates are capable of the following:

- Independent and method-based design of medium- to large-scale IT systems.
- Responsibility for technology and content of IT systems of medium to high criticality.
- Development, design and documentation of measures for achieving non-functional requirements. Support of development teams in implementation of these measures.
- Control and implementation of architecture-related communication in medium to large development teams.

0.4 Requirements for CPSA-A certification

- Successful training and certification as a CPSA-F (Certified Professional for Software Architecture, Foundation Level)
- At least three years of full-time career experience in the IT sector, with participation in design and development of at least two different IT systems
    - Exceptions are possible on application (such as participation in open source projects)
- Training and advanced qualification within the framework of iSAQB Advanced Level courses comprising at least 70 credit points from all three different areas of competence.
- Successful completion of the CPSA-A certification examination.

0.5 Structure of the curriculum and the recommended time allocation

Content	Recommended Minimum Time
Basics	90 minutes
Agile approach to architecture	150 minutes
Architecture requirements in agile projects	240 minutes
Designing architectures in a team	300 minutes
Reflection and feedback	180 minutes
Examples of agile architecture work	120 minutes
Total (3 days, each lasting 360 minutes)	1,080 minutes = 18h

0.6 Duration, didactics and other details

The times stated below are recommendations. The minimum duration of a course on AGILA should be 3 days, but it can be longer. Providers can differ with respect to the duration, methodology, type and structure of the exercises as well as the detailed course outline. In particular, the curriculum leaves the type of examples and exercises completely open

In terms of accreditation for the subsequent Advanced Level Certification Examination, licenced courses on the AGILA module contribute the following credit points:

Methodical competence: 20 points
Technical competence: 00 points
Communicative competence: 10 points

0.7 Prerequisites for the AGILA module

Participants should have the following knowledge and/or experience:

- 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.

In addition, the following knowledge is **advantageous** for an understanding of some of the concepts covered:

- Knowledge of agile procedure models:
  - The Agile Manifesto [AgileM+2001]
  - Scrum [Schwaber+2013]
  - 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.

0.8 Structuring of the module units based on learning goals

The individual units of the curriculum are specified using the following structure:

- Terms/concepts: The key terms associated with this topic.
- Lesson/practical exercise duration: Defines the minimum time that must be allocated to the teaching and practical exercises for this topic in an accredited course.
- Learning objectives: Describe the content to be taught, including the associated key terms and concepts.

The sections of the curriculum are structured based on learning goals.

Where necessary, the learning goals include references to further literature, standards or other sources.

1 Introduction to agile software architecture

Duration: 90 minutes

1.1 Terms and concepts

- Software architecture, general
- The Agile Manifesto: Values and principles
- 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

- Definition of architecture work and demarcation from design
- Agile ideas in the context of work on software architectures:
- The agile philosophy in the Agile Manifesto and the Scrum Guide
- Relevant principles of the Agile Manifesto
- Concepts and cornerstones of Lean
- Agile procedure models and their impacts on classic approaches to architecture.

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

- Potential conflict between a fast reaction capability and a good planning and control capability.
    - In the negative extreme case: Conflict between low reactivity (rigidity) and a lack of controllability (chaos).

- Just enough architecture work and supporting concepts; limitation of effort to relevant problems:
  - Focussing of effort based on architecture relevance
  - Focussing of communication
- Lean waste in architecture tasks.

LG 1-4: Knowledge of agile tools for architecture work

- Task management and prioritisation
- Agile alternatives to modelling tools
  - Compromise between a simple capability of creating diagrams and model elements that cannot be searched or are not linked
- Linking continuous code inspection to architecture attributes.

LG 1-5: Knowledge of the capabilities of anchoring architecture as a cross-cutting aspect in agile organisations

Establishing architecture as a basic skill of developers, for example via:

- Open, cross-project communication of best practices
- Establishment of Communities of Practice for architecture and keeping them active
- Architecture katas as a means of disseminating architecture knowledge
- Communicating and assigning architecture responsibility.

2 The agile architecture approach

Duration: 150 minutes

2.1 Terms and concepts

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

2.2 Learning goals

LG 2-1: Being able to iteratively and agilely structure architecture work

- Risk-driven architecture work: Architectural risk as the decisive factor for the use of architecture practices
- Assessment of the urgency of architecture issues and postponement of decisions
- Integration of architecture work into Scrum
- Determining iteration 0 and the necessary preparatory work
- Consideration of architecture work during release planning
- Managing architecture work with Kanban.

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

- Context-based selection of role models for architects, for example:
  - Architecture owner/supporting architect
  - Architecture agents
  - Work without designated architects.
- Responsibilities of Scrum roles
- Interaction between the three Scrum roles (Product Owner, Scrum Master, Development Team) for dealing with architecture tasks.

LG 2-3: Knowledge of ways of involving stakeholders in architecture work

- Creation of methodical docking points for the collaboration
- Motivating stakeholders to contribute to the architecture
- Making architecture aspects and impacts understandable for technical contact persons.

3 Architecture requirements in agile projects

Duration: 240 minutes

3.1 Terms and concepts

- 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 requirements in agile processes
- Technical debt
- Architectural risks incl. their determination
- Requirements management; grooming, 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

- Abstraction levels of quality scenarios
- Formulation of architectural problems and technical compromises in qualitative and technical terms.

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

- Supplementing stories with qualitative aspects (acceptance criteria)
- Anchoring of architecture requirements in backlogs
- Breaking down and assigning architecture tasks.

LG 3-3: Being able to use iterative approaches for continuous definition of architecture requirements

- Continuous collaboration and cooperation with customers (or their representatives):
  - Highlighting benefits for technical points of contact
  - Techniques for effective, regular involvement of technical points of contact.
- Using technical debts as architecture requirements
- Techniques for determination of risks: alone and in groups
- Using tests and static analysis as a source of architecture requirements.

LG 3-4: Being able to effectively organise joint management, evaluation and prioritisation of requirements

- Regular prioritisation of architecture requirements, and where necessary their separation and detailed specification (e. g., within the backlog grooming)
- Joint evaluation of architectural issues
- Prioritisation techniques for architecture requirements.

LG 3-5: Knowing and being able to explain urgency as a driving factor for architecture work

- Late architecture decisions and open options as value (Real Options Theory)
- Methods for recognition of urgently necessary architecture decisions
- Communication with the Product Owner and other roles of relevance to prioritisation.

4 Designing and developing architectures in a team

Duration: 300 minutes

4.1 Terms and concepts

- 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)
- Just-in-time architecture decisions
- Architecture principles
- Architecture wall.

4.2 Learning goals

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

Methodological fundamentals of effectively guiding group members (with equal rights) to reaching decisions:

- Voting and veto methods
- Consensus agreement and how to achieve it
- Separating decision preparation from decision making
- Understanding the dynamics of the delegation of decision-making authority
- Use in conjunction with agile methodologies (Scrum).

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

To be able to reach decisions effectively in groups, methodical and communicative aspects are important:

- Moderation of groups in decision-making processes
- Correctly handling dissent and resistance
- Techniques for steering broad discussion to a desired decision.

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

- Clarification and effective assignment of responsibilities
- Use of principles as a means of achieving consistent decisions and a high level of design integrity
- Technical system configuration to promote local decisions.

LG 4-4: Being familiar with architecture concepts for promoting local decision-making capabilities

The technical architecture also influences the extent to which typical architecture topics influence developers. If for example the technological stack is not specified centrally, but instead defined by teams, the impacts of changes to the team stack are less extensive, the communication load between the teams is reduced, and decisions can more easily be delegated to teams and developers.

- Classification of current architectural concepts for promotion of a local decision-making capability: Vertical architecture styles such as (micro)services or CQRS, polyglot architecture approaches etc.

- Prerequisites for vertical architecture approaches (organisational and technical)
- Compromises in the transition to flexibility of technology decisions (e. g., in the direction of developer know-how, operation or maintenance)
- Link to the principles of the Agile Manifesto (self-organised teams, faster reaction capability, etc.)
- Being aware of and able to evaluate risks associated with team-internal optimisation (for example redundancy, loss of design integrity, excessive diversity).

LG 4-5: Being familiar with methods for just-in-time architecture decisions

- The concept of the last responsible moment
- Technical ways of postponing architectural issues
- Splitting architectural issues into sub-decisions
- Techniques for handling urgent and high-risk decisions (Set-Based Design)

LG 4-6: Being familiar with ways of communicating architecture decisions in agile projects

- Stand-ups and feedback meetings on decisions
- Lightweight architecture evaluation workshops (see also Section 5)
- Architecture wall (informative workplace)
- Communicating architecture concepts with principles in teams
- Recognising the need for explicit decision criteria for some (large, risky) decisions.

5 Reflection and feedback on architecture work in the agile context

Duration: 180 minutes

5.1 Terms and concepts

- Lightweight architecture evaluation and other feedback mechanisms
- Root cause analysis
- Testing of qualitative system attributes
- Quality indicators and metrics.

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

- Lightweight development of architecture evaluation techniques
  - Moderation of small evaluation workshops
  - Correct feedback on drafts and decision submissions (feedback rules)
- Use of metrics and tools for continuous architecture review
- Reality check for architecture objectives.

LG 5-2: Being able to find the reasons for specific architecture problems

- Techniques for group determination of the cause of problems (e. g., root cause analysis)
- Establishing a link between test and/or measurement results and architecture decisions.

LG 5-3: Being familiar with feedback capabilities from the implementation and able to attribute results to architecture objectives

- Testing of qualitative attributes: Techniques, tools and integration into the development activities
- Selection and use of static quality indicators
- Carrying out trend analyses on quality indicators, and derivation of conclusions
- Advantages of Continuous Integration and Continuous Deployment in this context.

6 Examples of agile architecture work

Duration: 120 minutes

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.

LG 6-1: Being familiar with and understanding examples for decision-making procedures in agile projects

Based on at least two examples, participants should be able to understand how agile projects of different sizes make and communicate architecture decisions. For this purpose, the relevant meetings, roles and responsibilities should be described in courses, and the specific embedding into the agile process of the respective project should be made clear.

LG 6-2: 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 of joint work on architecture requirements in real projects (grooming, planning etc.).

LG 6-3: Being familiar with physical characteristics of agile communication concepts

Participants should be able to see and understand, in examples, how agile projects communicate architecture information and decisions (photos of informative workplaces, layouts of information radiators etc.).

LG 6-4: Being able to understand the postponement of architecture decisions

Based on a real example, participants should become familiar with and understand the postponement of architecture decisions.

LG 6-5: Being familiar with and understanding examples of agilely organised architecture groups

In at least two real examples, participants should become familiar with and understand the organisation of architecture communities or guilds, and become familiar with the activities and techniques used by companies to establish these groups and keep them active.

7 Sources and references on AGILA

A

[AgileM+2001] 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

F

[Fairbanks2010] George Fairbanks: "Just Enough Software Architecture: A Risk-Driven Approach ", Marshall & Brainerd 2010

K

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

L

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

R

[Richardson+2005] Jared Richardson, William A. Gwaltney: "Ship It!: A Practical Guide to Successful Software Projects", Pragmatic Bookshelf 2005

S

[Schwaber+2013] Ken Schwaber, Jeff Sutherland: "The Definitive Guide to Scrum: The Rules of the Game", http://www.scrumguides.org

T

[Toth2015] Stefan Toth: „Vorgehensmuster für Softwarearchitektur: Kombinierbare Praktiken in Zeiten von Agile und Lean", 2. Auflage, Hanser Verlag 2015

V

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