Curriculum for

Certified Professional for Software Architecture (CPSA)

– Foundation Level –

Version 3.01 (May 5th 2015)

Table of contents

0 INTRODUCTION 6

0.1 WHAT DOES FOUNDATION LEVEL TRAINING INCLUDE? 6
0.2 OUTLINE OF THE CURRICULUM AND RECOMMENDED ALLOCATION OF STUDY TIMES 7
0.3 DURATION, TEACHING METHODS AND FURTHER DETAILS ON ACCREDITED TRAINING 7
0.4 PREREQUISITES 7
0.5 CPSA-F CURRICULUM CHAPTERS, LEARNING GOALS AND RELEVANCE FOR THE EXAMINATION 8
0.6 WHAT IS AND WHAT IS NOT COVERED BY THE CURRICULUM 8
0.7 INTRODUCTION TO THE ISAQB CERTIFICATION PROGRAMME 8

1 BASIC CONCEPTS OF SOFTWARE ARCHITECTURES 10

1.1 TERMS AND CONCEPTS 10
1.2 LEARNING GOALS 10

2 DESIGN AND DEVELOPMENT OF SOFTWARE ARCHITECTURES 13

2.1 TERMS AND CONCEPTS 13
2.2 LEARNING GOALS 13

3 SPECIFICATION AND COMMUNICATION OF SOFTWARE ARCHITECTURES 17

3.1 TERMS AND CONCEPTS 17
3.2 LEARNING GOALS 17

4 SOFTWARE ARCHITECTURES AND QUALITY 19

4.1 TERMS AND CONCEPTS 19
4.2 LEARNING GOALS 19

5 TOOLS FOR SOFTWARE ARCHITECTS 21

5.1 TERMS AND CONCEPTS 21
5.2 LEARNING GOALS 21

6 EXAMPLES OF SOFTWARE ARCHITECTURES 22

7 SOURCES AND REFERENCES FOR SOFTWARE ARCHITECTURE 23

LIST OF LEARNING GOALS

LG 1-1: Being able to discuss definitions of software architecture (R1) 10
LG 1-2: Being able to understand and identify the benefits and objectives of software architecture (R1) 10
LG 1-3: Understanding the place of software architecture within the software life cycle (R2) 10
LG 1-4: Being able to understand software architects' tasks and responsibilities (R1) 11
LG 1-5: Being able to relate the role of software architects to other stakeholders (R1) 11
LG 1-6: Being able to explain the correlation between development approaches and software architecture (R1) 11
LG 1-7: Being able to differentiate between architecture and project objectives (R1) 11
LG 1-8: Distinguish explicit statements and implicit assumptions (R1) 12
LG 1-9: Knowing the roles and responsibilities of software architects within enterprise architecture (R3) 12
LG 1-10: Understanding the differences between types of software-intensive systems (R2) 12
LG 2-1: Being able to select and adhere to approaches and heuristics for architecture development (R1) 13
LG 2-2: Being able to design architectures (R1) 13
LG 2-3: Being able to identify and rank influencing factors upon software architecture (R1) 14
LG 2-4: Being able to select and develop cross-cutting concepts (R1) 14
LG 2-5: Describe, explain and appropriately use important architectural patterns and architectural styles (R1-R3) 14
LG 2-6: Being able to explain and use design principles (R1) 15
LG 2-7: Being able to plan correlations and dependencies between modular components (R1) 15
LG 2-8: Being able to design building blocks / structural elements of software architectures (R1) 15
LG 2-9: Being able to design and define interfaces (R1) 16
LG 2-10: Understanding and using architecture-relevant design patterns (R2) 16
LG 3-1: Being able to explain and consider quality attributes of technical documentation (R1) 17
LG 3-2: Being able to describe and communicate software architectures appropriately to stakeholders (R1) 17
LG 3-3: Understanding how to explain and use notations / models to describe software architecture (R2) 17
LG 3-4: Being able to explain and use architectural views (R1) 17
LG 3-5: Being able to explain and use the system context (R1) 18
LG 3-6: Being able to explain and use cross-cutting and technical architecture concepts (R1) 18
LG 3-7: Being able to describe interfaces (R1) 18
LG 3-8: Understanding the explanation and documentation of architecture decisions (R2) 18
LG 3-9: Understanding the use of documentation as written communication (R2) 18
LG 3-10: Knowing additional resources and tools for documentation (R3) 18
LG 4-1: Being able to discuss quality models and quality characteristics (R1) 19
LG 4-2: Being able to define quality requirements for software architectures (R1) 19
LG 4-3: Understanding the qualitative evaluation of software architectures (R2) 19
LG 4-4: Understanding the quantitative evaluation of software architectures (R2) 20
LG 4-5: Understanding how quality objectives are achieved using appropriate approaches and techniques (R2) 20
LG 5-1: Being able to name and rank important tool categories (R1) 21

LG 5-2: Understanding how tools are selected as required (R2) 21
LG 6-1: Knowing how the relation between requirements and solutions is established (R3) 22
LG 6-2: Knowing the rationale of a solution's technical implementation (R3) 22

0 Introduction

0.1 What does Foundation Level training include?

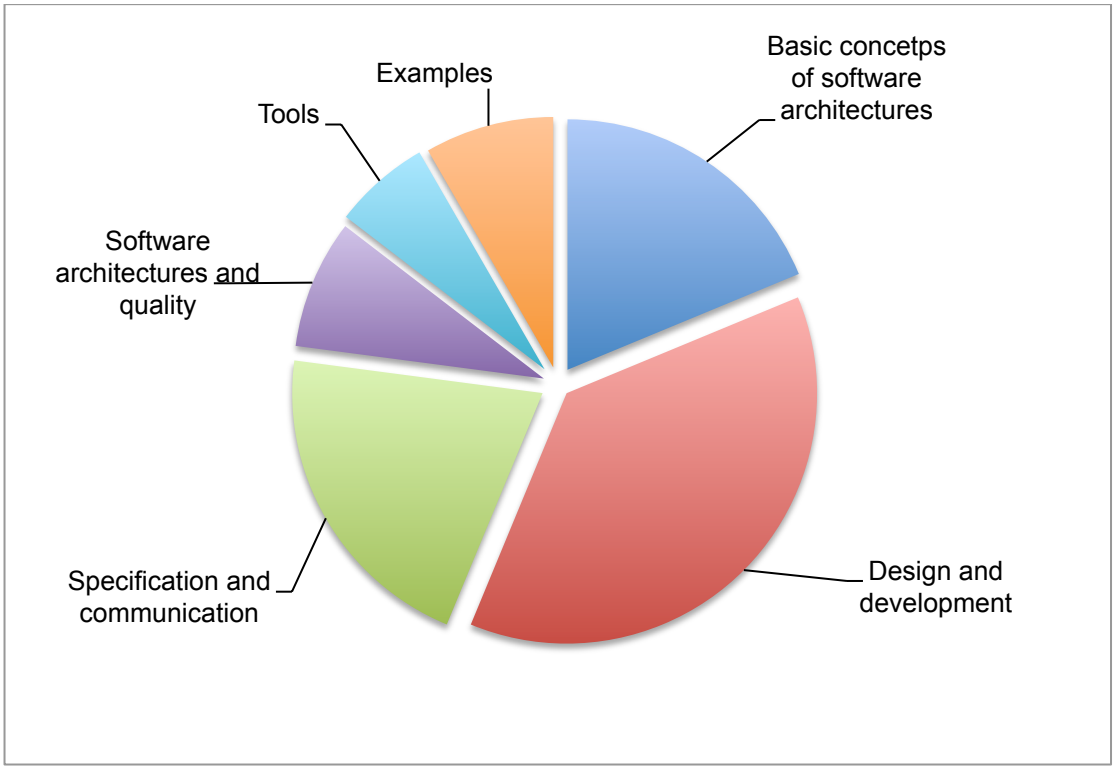
Licensed **Certified Professional for Software Architecture – Foundation Level** (CPSA-F) training will provide participants with the knowledge and skills required for designing and documenting a problem-specific software architecture based on a sufficiently detailed requirements specification for small and medium-sized systems. Participants will be provided with a set of tools enabling them to make problem-specific design decisions on the basis of their previously acquired practical experience.

In particular, such training shall include:

- The term software architecture and its meaning
- The tasks and responsibilities of software architects
- The roles of software architects within projects
- State-of-the-art methods and techniques for developing software architectures, as well as the following competences:
  - Consulting other parties involved in a shared project, in particular representatives from requirements management, development, project management and testing, in order to make essential decisions on software architecture
  - Documenting and communicating software architectures on the basis of views, cross-cutting concepts, decisions as well as architectural patterns and styles
  - Understanding the main steps necessary for developing software architectures and implementing these independently in small and medium-sized systems

Participants of CPSA-F training will learn that the design and documentation of a software architecture are consistent with its actual implementation. Depending on approaches, processes and organisation, one set of results may influence another and vice versa, and ideally it will be possible for these results to be constantly updated and improved.

0.2 Outline of the curriculum and recommended allocation of study times



0.3 Duration, teaching methods and further details on licensed training

Study times given are recommendations. The duration of a training course should be at least three days, but may as well be longer. Providers may vary in their approach to duration, teaching methods, the type and structure of exercises as well as the detailed course outline. In particular, the types (domains and technologies) of examples and exercises can be determined individually by the training provider.

0.4 Prerequisites

Participants should have the following prior knowledge and/or experience:

- More than 18 months practical experience with software development, gained through programming a variety of projects or systems outside of formal education
- Knowledge of and practical experience with at least one higher programming language
- Basics of modelling and abstraction
- Basics of UML (class, package, component and sequence diagrams) and their relation to source code
- Practical experience with technical documentation, in particular with documenting source code, system designs or technical concepts

Furthermore, the following will be useful for understanding certain concepts:

- Knowledge of object-oriented concepts
- Practical experience with a minimum of one object-oriented programming language
- Practical experience with designing and implementing distributed applications, such as client-server systems or web applications

0.5 CPSA-F curriculum chapters, learning goals and relevance for the examination

The structure of the curriculum's chapters follows a set of prioritised learning goals. For each learning goal, relevance for the examination of this learning goal or its sub-elements is clearly stated (through the use of the R-1/R-2/R-3 classification, see table).

Every learning goal describes the contents to be taught including their key terms and concepts.

With regard to relevance for the examination, the following categories are used in this curriculum:

Learning-goal category	Identifier	Meaning	Relevance for examination
Being able to ...	R-1	These are the contents participants will be expected to be able to put into practice independently upon completion of the course. Within the course, these contents will be covered through exercises and discussions.	Contents will be part of the examination.
Understanding ...	R-2	These are the contents participants are expected to understand in principle. They will normally not be the main focus of exercises in training.	Contents may be part of the examination.
Knowing ...	R-3	These contents (terms, concepts, methods, practices or similar) can enhance understanding and motivate the topic. They may be covered in training if required.	Contents will not be part of examination.

If required, the learning goals include references to further reading, standards or other sources.

The sections "Terms and Concepts" of each chapter list word that are associated with the contents of the chapter. Some of them are used in the descriptions of learning goals.

In certification examinations, iSAQB may use relevant questions to test whether the prerequisites listed above have been met.

0.6 What is and what is not covered by the curriculum

This curriculum reflects the contents currently considered by iSAQB members to be necessary and useful for achieving the learning goals of CPSA-F. It is not a comprehensive description of the entire domain of 'software architecture'.

The following topics or concepts are **not** part of CPSA-F:

- Concrete implementation technologies, frameworks or libraries
- Programming or programming languages
- Basics or notations of modelling (such as UML)
- System analysis and requirements engineering (please refer to the education and certification program by IREB e.V., http://ireb.org, International Requirements Engineering Board)
- Test (please refer to the education and certification program by ISTQB e.V., http://istqb.org, International Software Testing Board)
- Project or product management
- Introduction to specific software tools

0.7 Introduction to the iSAQB certification programme

Duration: 15 min	Exercises: n/a

This section is not relevant for the examination.

Participants will become familiar with the context of the iSAQB certification programme as well as its examinations or examination modalities such as:

- iSAQB as an association
- iSAQB's responsibility for the curriculum as well as corresponding examination questions
  - Organisational separation between training and examination
  - Procedures and formal constraints of the CPSA-F examination
- Foundation Level versus Advanced Level
- Optional: other certification schemes

0.8 Change history

Date	Editor	Description
2014-07-11	pg	Added reference to Dern2006
2014-07-31	MISO	• References to POSA series on LZ 2-10 corrected. The reference also corrected (4 digit year) • References with spaces like [Zörner 2012] replaced with the correct version without space.
2014-12-12	GStarke	• Added test to section 0.6 (what is NOT covered) • Formatting • Minor corrections in preparation of Jan-2015 board meeting, see Trello and Subversion commit history. • Fixed some translation issues (e.g. LG 1.8)
2014-12-26	MISO	Changed overview diagram to 2D like the DE version (section 0.2)
2015-01-16	WG Foundation	Finalized V 3.0, several bugfixes
2015-04-14	GS, PG + UB	Corrected several translation issues
2015-04-30	AR, GS, PG	Minor corrections

1 Basic concepts of software architectures

Duration: 135 min	Exercises: 45 min

1.1 Terms and concepts

Software architecture; structure; building blocks/components; interfaces; relationships; cross-cutting concerns/aspects; architecture objectives; software architects and their responsibilities; tasks and required skills; non-functional and functional requirements of systems; constraints; influencing factors; types of IT systems (embedded systems, real-time systems, information systems etc)

1.2 Learning goals

LG 1-1: Being able to discuss definitions of software architecture (R1)

- Comparison of several definitions of software architecture (incl. ISO 42010/IEEE 1471, SEI, Booch etc) and identification of their similarities:
  - Components / building blocks with interfaces and relationships
  - Building blocks as a general term, components as specific types of building blocks
  - Structures and cross-cutting concerns, principles
  - Cross-cutting design decisions and their consequences both across the system and concerning the entire life cycle

LG 1-2: Being able to understand and identify the benefits and objectives of software architecture (R1)

- Objectives of software architects and software architectures
  - Software architecture focuses on quality attributes such as durability, maintainability, changeability, robustness more than on pure functionality.
- Software architecture supports the creation and maintenance of software, in particular its implementation.
- Software architecture supports the achievement of quality requirements.
- Advandages and limitations of the metaphor “architecture for buildings” versus “architecture for software”. (R2)

LG 1-3: Understanding the place of software architecture within the software life cycle (R2)

- The place and role of software architecture within the overall development of IT systems
- The correlation with business and operational processes for information systems
- The correlation with business and operational processes for decision support systems (data warehouse, management information systems)

LG 1-4: Being able to understand software architects' tasks and responsibilities (R1)

- Software architects are responsible for achieving the required or necessary quality and functionality of a solution. Depending on the actual approach or process model used, they must align this responsibility with the overall responsibilities for project management and/or other roles.
- Tasks and responsibilities of software architects:
  - Clarify and question requirements as well as constraints and refine them if required. Together with functional requirements (required features), this includes the required quality attributes in particular (required constraints).
  - Decide how to decompose the system into building blocks, while determining dependencies and interfaces between the building blocks.
  - Determine and, if necessary, implement cross-cutting technical concerns (for instance persistence, communication, GUI etc.)
  - Communicate and document software architecture based on views, architectural patterns and technical concepts
  - Accompany the realisation and implementation of the architecture; integrate feedback from relevant stakeholders into the architecture if necessary; review and ensure the consistency of source code and software architecture
  - Evaluate software architecture, especially with regard to risks involving the required quality characteristics
- It is the responsibility of software architects to identify and highlight the consequences of architectural decisions and discuss these with other stakeholders.
- Their role involves recognising the necessity of iterations in all tasks and pointing out possibilities for relevant feedback.

LG 1-5: Being able to relate the role of software architects to other stakeholders (R1)

- Participants should be familiar with the role of software architects in relation to other stakeholders and be able to explain this role, in particular any potential connections with:
  - Requirements analysis (system analysis, requirements management, specialist field)
  - Implementation
  - Project management
  - Quality assurance
  - IT operations (production, data centres), applies primarily to information systems
  - Hardware development

LG 1-6: Being able to explain the correlation between development approaches and software architecture (R1)

- Explain the influence of iterative approaches on architectural decisions (with regard to risks and predictability)
- Due to inherent uncertainty, software architects often have to work and decide iteratively. To do so, they must systematically seek feedback from other stakeholders.

LG 1-7: Being able to differentiate between architecture and project objectives (R1)

- Participants should be able to demonstrate the significance of architectural objectives, constraints and influencing factors for the design of software architectures.

- Explanation of (long-term) architectural objectives and the distinction between them and (short-term) project objectives
- Identify and specify architectural objectives on the basis of existing requirements
- Explain the connection between requirements and solutions

LG 1-8: Distinguish explicit statements and implicit assumptions (R1)

- Software architects should present assumptions or prerequisites explicitly and avoid implicit assumptions.
- Implicit assumptions can lead to misunderstandings between stakeholders.

LG 1-9: Knowing the roles and responsibilities of software architects within enterprise architecture (R3)

- Additional levels of architecture, e.g. enterprise IT architecture/business architecture, infrastructure architecture (according to [Dern 2006] for instance): several different levels of architectures exist within the IT of information systems:
  - Infrastructure architecture: the technical infrastructure's structure, hardware, networks etc
  - Hardware architecture (for hardware-related systems)
  - Software architecture: the structure of individual software systems. This is what iSAQB and CPSA-F identify as the focus for software architects.
  - Corporate IT architecture, Enterprise IT architecture: the structure of application landscapes. CPSA-F does not focus on this topic.
  - Business process architecture, business architecture: the structure of business processes. CPSA-F does not focus on this topic.

LG 1-10: Understanding the differences between types of software-intensive systems (R2)

- Understand how software architecture is differentiated according to different types of IT systems:
  - Relation to the system architecture or overall architecture for embedded or hardware-related systems
  - Understand the characteristics of hardware/software design (and code) (dependencies between hardware and software design in relation to time and content)

2 Design and development of software architectures

Duration: 270 min	Exercises: 90 min

2.1 Terms and concepts

Design; design approach; design decision; views, technical and crosscutting concepts; architectural patterns; design principles; domain-related and technical architectures; model-based design; iterative/incremental design; domain-driven design; top-down and bottom-up approaches, pattern-languages, stereotypes, tools-and-material-approach

2.2 Learning goals

LG 2-1: Being able to select and adhere to approaches and heuristics for architecture development (R1)

- Participants can name, explain and apply basic methods of architecture development
- Model- and view-based architecture development
- Domain-driven design
- Iterative and incremental design
  - Necessity of iterations, especially when decision-making is affected by uncertainties
  - Feedback on design decisions on the basis of source code as well as qualitative considerations
- Top-down and bottom-up approaches to design
- Influencing factors and constraints for architecture design (global analysis – see [Hofmeister+2000])

LG 2-2: Being able to design architectures (R1)

- Participants can design a software architecture on the basis of known functional and non-functional requirements for software systems that are neither safety nor business-critical and document these appropriately
- Identify and give reasons for interdependencies between design decisions
- Make structure-relevant decisions with regard to system decomposition and the building-block structure while determining dependencies and interfaces between the building blocks
- Explain the terms 'black box' and 'white box' and use them in a targeted manner
- Apply gradual refinement (hierarchisation) as well as the exact specification of building blocks
- Design of individual architecture views, especially distribution, building-block and runtime views, and describe their consequences for the relevant source code
- Define how the architecture is mapped to source code and assess or evaluate the associated consequences
- Justify and apply the separation of technical and domain-related components of architectures
- Design and justify domain-related structures

  - Identify, justify and document domain-related building blocks (entities, services)
  - Design and explain the interaction between domain-related and technical components of systems
- Know the considerable impact of non-functional requirements (such as changeability, robustness, efficiency) and consider these when making architecture and design decisions

LG 2-3: Being able to identify and rank influencing factors upon software architecture (R1)

- Participants are expected to gather and consider constraints and influencing factors that restrict the freedom in design decisions
- Recognise and consider the impact of quality requirements on architectures
- Recognise and consider the impact of technical decisions and concepts on architectures
- Recognise and consider the (potential) impact of organisational structures on building-block structures (R2)

LG 2-4: Being able to select and develop cross-cutting concepts (R1)

- Decide / design cross-cutting / technical concepts, for example persistence, communication, GUI, error handling, concurrency
- Recognise and assess any potential interdependencies between these decisions

LG 2-5: Describe, explain and appropriately use important architectural patterns and architectural styles (R1-R3)

- Dataflow or data-centred architectural styles (R1)
  - Pipes and filters
- Hierarchical architectural styles (R1)
  - Layers
- Architectural styles for interactive systems (R2)
  - Model view controllers
  - Model view presenters
- Heterogeneous architectural styles (R1)
- Architectural styles for asynchronous systems (R3)
  - According to Hohpe (messaging, async pattern)
- Architecture styles for distributed systems (R3)
- Further architectural patterns and styles (R3), e.g.:
  - Event-based systems
  - CQRS
  - According to Fowler/PoEAA
  - According to POSA (e.g. Blackboard or Microkernel)
- Know essential sources for architectural patterns, for instance POSA literature and PoEAA (for information systems) (R3)
- Know examples of further pattern languages (R3)

- Organisational patterns
- Reengineering patterns
- Security patterns
- Client/server patterns
- Patterns for distributed systems
- Further patterns depending on the focus of the training

LG 2-6: Being able to explain and use design principles (R1)

- Information hiding
- Coupling and cohesion
- Separation of concerns
- Open/closed principle
- Inversion of dependencies through interfaces
- Dependency injection in order to externalise dependencies
- Relationships between dependencies in the model and in the source code of programming languages

LG 2-7: Being able to plan correlations and dependencies between modular components (R1)

- Participants are expected to understand the dependencies between and coupling of building blocks and apply these in a targeted manner.
- Be able to demonstrate types of coupling (structural, temporal, via data types, via hardware etc.)
- Identify consequences of coupling
- Understand ways of removing or reducing coupling
- Implementation of relationships in (object-oriented) programming languages
  - Constructors
  - Factory patterns
  - Dependency injection

LG 2-8: Being able to design building blocks / structural elements of software architectures (R1)

- Participants are expected to know and use desirable characteristics (encapsulation, information hiding) of building blocks and structural elements.
- Black-box and white-box building blocks
- Types of composition of building blocks (nesting, usage/delegation, inheritance)
- UML notation for various building blocks and their compositions

- Packages as a semantically weak type of building-block aggregation
- Components with clearly defined interfaces as a semantically more precise type of aggregation

LG 2-9: Being able to design and define interfaces (R1)

- Participants are expected to recognise the importance of interfaces. They should to be able to design or determine interfaces between architectural building blocks and external interfaces between architectural building blocks and elements outside of the system.
- Participants know desired characteristics of interfaces:
  - Easy to learn, easy to use, easy to expand
  - Hard to misuse
  - Functionally complete from the perspective of users or building blocks using them
- Participants can describe and document interfaces
- Participants know different ways of looking at interfaces: (R3)
  - Resource-oriented approach (see Representational State Transfer)
  - Service-oriented approach (see WS-*/SOAP-based webservices)

LG 2-10: Understanding and using architecture-relevant design patterns (R2)

- Understand language-independent design patterns, such as: adapter, wrapper, gateway, facade, registry, broker
- Be familiar with further design patterns, which do not necessarily need to be part of licensed training

References

[Fowler2003]

[Gharbi+2014]

[Martin2003]

POSA series, in particular [Buschmann+1996] and [Buschmann+2007]

[Starke2014]

3 Specification and communication of software architectures

Duration: 150 min	Exercises: 90 min

3.1 Terms and concepts

Views; structures; (technical) concepts; documentation; communication; description; meta structures for description and communication; building blocks; building-block view; run-time view; deployment view; node; channel; deployment units; mapping building blocks onto deployment units; description of interfaces

3.2 Learning goals

LG 3-1: Being able to explain and consider quality attributes of technical documentation (R1)

- Understandability, accuracy, efficiency, adequacy, maintainability
- Form, content and level of detail of documentation chosen in accordance with its target group
- Readers alone can assess the understandability of technical documentation.

LG 3-2: Being able to describe and communicate software architectures appropriately to stakeholders (R1)

- Due to the diversity of stakeholders, the description of software architectures presents particular requirements.
  - Diverse audiences: management, developers, quality assurance as well as other software architects
  - Diverse authors: software architects, developers and potentially more

LG 3-3: Understanding how to explain and use notations / models to describe software architecture (R2)

- Participants should know the following UML diagrams for the notation of architectural views:
  - Class, package and component diagrams
  - Deployment diagrams
  - Sequence and activity diagrams
  - State machines, state diagrams
- Benefits of template-based documentation

LG 3-4: Being able to explain and use architectural views (R1)

- Building-block or component view (how the system is composed of software building blocks)
- Run-time view (dynamic view, interaction between software building blocks at run time, state machines)
- Deployment view (mapping software building blocks on hardware or execution environments)

LG 3-5: Being able to explain and use the system context (R1)

- Differentiate between domain-related and technical contexts

LG 3-6: Being able to explain and use cross-cutting and technical architecture concepts (R1)

- Explain the significance of cross-cutting technical concerns (principles, architectural aspects or concepts)
- Name some typical concepts (e.g. persistence, workflow management, GUI, deployment/integration)

LG 3-7: Being able to describe interfaces (R1)

- Description and specification of interfaces
- Differentiation between internal and external interfaces

LG 3-8: Understanding the explanation and documentation of architecture decisions (R2)

- Document and give reasons for the systematic derivation of architecture decisions

LG 3-9: Understanding the use of documentation as written communication (R2)

- Means for describing software architectures also support their design and creation.
- The wording and notation used in technical documentation should be chosen in accordance with the readers' skills and objectives.

LG 3-10: Knowing additional resources and tools for documentation (R3)

- The basics of several of the published frameworks for describing software architectures, such as:
  - TOGAF, RM/ODP, ISO/IEEE-42010 (previously 1471)
  - arc42, FMC, Tigris ReadySet
- Ideas and examples of check lists for the creation, documentation and evaluation of software architectures
- Potential tools for creating and maintaining architectural documentation

References

[Clements+2002]

[Gharbi+2014]

[Starke2014]

[Zörner2012]

4 Software architectures and quality

Duration: 60 min	Exercises: 60 min

4.1 Terms and concepts

Quality; quality attributes; DIN/ISO 9126 resp. 25010; ATAM; scenarios; quality tree; compromises (when implementing quality attributes); evaluation of software architectures (qualitative and quantitative)

4.2 Learning goals

LG 4-1: Being able to discuss quality models and quality characteristics (R1)

- Explain the concepts of quality (following ISO/IEC 25010, previously 9126) and quality attributes
- Explain quality models (such as ISO/IEC 25010)
- Explain relationships and interactions between quality attributes, for instance:
  - Flexibility versus robustness
  - Memory usage versus run time consumption

LG 4-2: Being able to define quality requirements for software architectures (R1)

- Participants are expected to be able to express the quality requirements for software and software architectures in concrete terms and present them using scenarios and quality trees for instance.
- Explain and carry out the creation of scenarios and quality trees
- Express exemplary quality requirements for software

LG 4-3: Understanding the qualitative evaluation of software architectures (R2)

- Participants are expected to be familiar with the methodical approach to analysing and evaluating software architectures and be able to apply it to smaller examples.
- ATAM approach to qualitative evaluation of software architectures
- For the qualitative evaluation of architectures, the following sources of information may be helpful:

- Requirements, in particular in the form of quality trees and scenarios
- Architecture documentation
- Architecture and design models
- Source code
- Metrics

LG 4-4: Understanding the quantitative evaluation of software architectures (R2)

- The quantitative evaluation (metrics) of software, in particular of source code, can help to identify critical parts within systems.
- Further information can be consulted for assessment and evaluation of architectures, such as:
  - Source code (e.g. metrics like lines of code, cyclomatic complexity, incoming and outgoing dependencies, instability, abstractness, distance)
  - Known errors in source code, in particular error clusters
  - Test cases and test results
  - Requirement and solution models

LG 4-5: Understanding how quality objectives are achieved using appropriate approaches and techniques (R2)

- Participants are expected to explain and apply tactics, best practices and technical possibilities of attaining important quality objectives of software systems (different for embedded systems and information systems), for instance:
  - Efficiency/performance
  - Maintainability, changeability, expandability, flexibility
  - Identification of corresponding risks

References

[Bass+2003]

[Clements+2002]

[Gharbi+2014]

[Martin2003]

[Starke2014]

5 Tools for software architects

Duration: 45 min	Exercises: n/a

5.1 Terms and concepts

Modelling tools, static analysis tools, dynamic analysis tools, generation tools, requirements-management tools, build systems/tools, configuration management tools

5.2 Learning goals

LG 5-1: Being able to name and rank important tool categories (R1)

- Participants are expected to be able to name and explain the most important categories of tools in relation to the work of software architects.

LG 5-2: Understanding how tools are selected as required (R2)

- The working environment and tools of software architects depend on the relevant constraints and influencing factors.

6 Examples of software architectures

Duration: 60 min	Exercises: n/a

This section is not examination relevant.

LG 6-1: Knowing how the relation between requirements and solutions is established (R3)

- Participants are expected to recognise and comprehend the correlation between requirements/architectural objectives and chosen solutions using at least one example.

LG 6-2: Knowing the rationale of a solution's technical implementation (R3)

- With reference to one example, participants are expected to comprehend the technical realisation (implementation, technical concepts, products used, solution strategies) of a solution.

7 Sources and references for software architecture

This section contains a list of sources used for the development of this curriculum.

B

[Bachmann2000]
Bachmann, Felix/Bass, Len/Carriere, Jeromy/Clements, Paul/Garlan, David/Ivers, James/Nord, Robert/Little, Reed. *Software Architecture Documentation in Practice: Documenting Architectural Layers,* Special Report CMU/SEI-2000-SR-004, March 2000, CMU, 2000.

[Bass+2012]
Bass, L/Clements, P/Kazman, R. *Software Architecture in Practice,* 3rd edition, Addison-Wesley, Reading, Mass., 640 pp, 2012, ISBN10 978-0-3218-1573-6.

[Berns+2010]
Berns, K/Schürmann, B/Trapp, M. *Eingebettete Systeme, Systemgrundlagen und Entwicklung eingebetteter Software,* 1st edition, Springer, 270 pp, ISBN13 978-3-8348-9661-2 [only available in German].

[Bosch2000]
Bosch, Jan. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach,* ACM Press, 7 June 2000, Addison-Wesley Longman, Amsterdam, 368 pp, 2000, ISBN10 201674947, ISBN13 978-0201674941.

[Buschmann+1996]
Buschmann, Frank/Meunier, Regine/Rohnert, Hans/Sommerlad, Peter. *A System of Patterns: Pattern-Oriented Software Architecture 1*, Wiley Software Patterns, 1st edition, 12 July 1996, John Wiley & Sons, 476 pp, 1996, ISBN10 471958697, ISBN13 978-0471958697.

[Buschmann+2007]
Buschmann, Frank/Henney, Kevlin/Schmidt, Douglas C. *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing,* Volume 4: *Pattern Language for Distributed Object Computing,* 1st edition, 16 March 2007, John Wiley & Sons, 636 pp, 2007, ISBN10 470059028, ISBN13 978-0470059029.

C

[Clements+2002]
Clements, Paul/Kazman, Rick/Klein, Mark. *Evaluating Software Architectures Methods and Case Studies,* Addison-Wesley Longman, Amsterdam, 368 pp, 2001, ISBN10 020170482X, ISBN13 978-0201704822.

[Clements+2010]
Clements, Paul/Bachmann, Felix/Bass, Len/Garlan, David/Ivers, James/Little, Reed/Merson, Paulo/Nord, Robert L. *Documenting Software Architectures: Views and Beyond,* 2nd edition, 5 October 2010, Addison Wesley, 608 pp, 2010, ISBN10 321552687, ISBN13 978-0321552686.

D

[Dern2006]
Gernot Dern (Autor): Management von IT-Architekturen: Leitlinien für die Ausrichtung, Planung und Gestaltung von Informationssystemen (Edition CIO), Edition 3., durchges. Aufl. 2009 (12. March 2009), Vieweg+Teubner Verlag, p. 343, 2009, ISBN10 3834807184, ISBN13 9783834807182.

E

[Evans2004]
Evans, Eric J. *Domain-Driven Design: Tackling Complexity in the Heart of Software,* 1st edition, 20 August 2003, Addison-Wesley Longman, Amsterdam, 529 pp, 2003, ISBN10 321125215, ISBN13 978-0321125217.

F

[Fowler2003]
Fowler, Martin. *Patterns of Enterprise Application Architecture,* 1st edition, 5 November 2002, Addison-Wesley Longman, Amsterdam, 560 pp, 2002, ISBN10 321127420, ISBN13 978-0321127426.

G

[Gamma+1995]
Gamma, Erich/Helm, Richard/Johnson, Ralph/Vlissides, John M. *Design Patterns: Elements of Reusable Object-Oriented Software,* 1st edition, 10 November 1994, Addison-Wesley Professional, 416 pp, 1994, ISBN10 201633612, ISBN13 978-0201633610.

[Gharbi+2014]
Gharbi M., Koschel, A., Rausch, A., Starke, G.: Basiswissen für Softwarearchitekten. 2. Auflage, Dpunkt Verlag, 2014, ISBN10 3864901650 [only available in German].

[Gorton2011]
Gorton, I. *Essential Software Architecture,* 2nd edition, 2011, Springer, 242 pp, 2011, ISBN13 978-3-642-19176-3.

H

[Hargis+2004]
Hargis, Gretchen/Carey, Michelle/Hernandez, Ann. *Developing Quality Technical Information: A Handbook for Writers and Editors,* IBM Press Series-Information Management, 2nd edition, 6 April 2004, Prentice Hall, 432 pp, 2004, ISBN10 131477498, ISBN13 978-0131477490.

[Hofmeister+2000]
Hofmeister, Christine/Nord, Robert/Soni, Dilip. *Applied Software Architecture,* 1st edition, 14 November 1999, Addison-Wesley Professional, 432 pp, 1999, ISBN10 321643348, ISBN13 978-0321643346.

[Hofmeister+2005]
Hofmeister, C/Nord, R/Soni, D. "Global Analysis: Moving from Software Requirements Specification to Structural Views of the Software Architecture", *IEE Proceedings Software*, volume 152, issue 4, August 2005, 11 pp, pp. 187-197, 2005.

J

[Josuttis2008]
Josuttis, N M. *SOA in der Praxis - System-Design für verteilte Geschäftsprozesse*, Dpunkt Verlag, 408 pp, 2008, ISBN13 978-3-89864-476-1 [only available in German].

K

[Koschel+2008]
Dunkel, J/Eberhart, A/Fischer, S/Kleiner, C/Koschel, A. *Systemarchitekturen für Verteilte Anwendungen - Client-Server, Multi-Tier, SOA, Event-Driven Architectures, P2P, Grid, Web 2.0*, 1st edition, Carl Hanser Verlag Munich, 3005 pp, 2008, ISBN13 978-3-446-41321-4 [only available in German].

[Kruchten1995]
Kruchten, Philippe. "The 4+1 View Model of Architecture*", IEEE Software*, volume 12 (6), 16 pp, pp 45-50, 1995, DOI: 10.1109/52.469759.

M

[Martin2003]
Martin, R C. *Agile Software Development, Principles, Patterns and Practices*, Prentice Hall, 529 pp, 2002, ISBN10 135974445, ISBN13 978-0135974445.

[MVP]
http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter

P

[Parnas1972]
Parnas, David. "On the criteria to be used in decomposing systems into modules", *Communications of the ACM, CACM Homepage archive,* volume 15 issue 12, Dec 1972, pp 1053-1058 1972.

Q

[Quian+2010]
Qian, K/Fu, X/Tao, L/Xu, C/Herrera, J. *Software Architecture and Design Illuminated,* 1st edition, Jones and Bartlett, 387 pp, 2010, ISBN13 9780763754204.

R

[Reussner+2008]
Reussner, Ralf/Hasselbring, Wilhelm. *Handbuch der Software-Architektur*, hardback, 2nd edition, 15 December 2008, dpunkt Verlag, 575 pp, 2008, ISBN10 3898645592, ISBN13 978-3898645591 [available in German only].

[Rupp+2012]
Rupp, C/Queins, S., die Sophisten. *UML 2 glasklar,* 4th edition, Carl Hanser Verlag Munich, XX pp, 2012, ISBN13 978-3-446-43057-0.

S

[Schmidt+2000]
Schmidt, Douglas C/Stal, Michael/Rohnert, Hans/Buschmann, Frank. *Pattern-Oriented Software Architecture,* volume 2: *Patterns for Concurrent and Networked Objects,* 1st edition, 15 August 2000, John Wiley & Sons, 666 pp, 2000, ISBN10 471606952, ISBN13 978-0471606956.

[Shaw+1996]
Shaw, Mary/Garlan, David. *Software Architecture: Perspectives on an Emerging Discipline,* October 1996, Prentice Hall, 242 pp, 1996, ISBN10 131829572, ISBN13 978-0131829572.

[Siedersleben2004]
Siedersleben, Johannes. *Moderne Software-Architektur: Umsichtig planen, robust bauen mit Quasar,* 1st edition, July 2004, Dpunkt Verlag, 281 pp, 2004, ISBN10 3898642925, ISBN13 978-3898642927 [available in German only].

[Starke+2011]
Starke, Gernot/Hruschka, Peter. *Software-Architektur kompakt,* 2nd edition, Spektrum Akademischer Verlag, 121 pp, 2011, ISBN13 978-3-8274-2835-6 [available in German only].

[Starke2014]
Starke, Gernot. *Effektive Software-Architekturen,* 6th edition, January 2014, Carl Hanser Verlag GmbH & Co. KG, 409 pp, 2014, ISBN13 978-3-446-43614-5 [available in German only].

T

[Tilkov2011]
Tilkov, S. *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien,* 2nd edition (expected April 2014), Dpunkt Verlag, 2011 pp, ISBN13 978-3-86490-120-1 [available in German only].

[Toth2013]
Toth, S. *Vorgehensmuster für Softwarearchitektur - Kombinierbare Praktiken in Zeiten von Agile und Lean*. 1st edition, Carl Hanser Verlag Munich, 249 pp, 2013, ISBN13 978-3-446-43615-2 [available in German only].

V

[Vernon2013]
Vernon, V. *Implementing Domain Driven Design,* 1st edition, February 2013, Addison Wesley, 656 pp, 2013, ISBN13 978-0-3218-3457-7.

Z

[Zörner2012]
Zörner, S. *Softwarearchitekturen dokumentieren und kommunizieren,* 1st edition, Carl Hanser Verlag Munich, 280 pp, 2012, ISBN13 978-3-446-42924-6 [available in German only].

