

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Software architecture as an aspect of software engineering. . . . .	2
1.2	iSAQB: The International Software Architecture Qualification Board. .	4
1.3	Certified Professional for Software Architecture – Foundation and Advanced Level. . . . .	5
1.4	The aim of this book. . . . .	7
1.5	Prerequisites . . . . .	8
1.6	Reader’s guide. . . . .	8
1.7	Target audience. . . . .	9
1.8	Acknowledgements. . . . .	10
<b>2</b>	<b>Software Architecture Fundamentals</b>	<b>11</b>
2.1	Integration with the iSAQB curriculum. . . . .	12
2.2	Software-intensive systems and software architectures . . . . .	13
2.3	Fundamental software architecture concepts . . . . .	19
2.4	A bird’s-eye view of software architecture design . . . . .	39
2.5	Test your knowledge. . . . .	47
<b>3</b>	<b>Designing Software Architectures</b>	<b>51</b>
3.1	Integration with the iSAQB curriculum. . . . .	52
3.2	Overview of the architecture design process . . . . .	52
3.3	Design principles and heuristics . . . . .	59
3.4	Architecture-centric development approaches. . . . .	65
3.5	Techniques for a good design . . . . .	72
3.6	Architectural patterns. . . . .	80
3.7	Design patterns. . . . .	90
3.8	Test your knowledge. . . . .	97

<b>4</b>	<b>Description and Communication of Software Architectures</b>	<b>101</b>
4.1	Integration with the iSAQB curriculum. . . . .	101
4.2	The CoCoME example. . . . .	102
4.3	Views and templates. . . . .	105
4.4	Technical/cross-cutting concepts in software architectures . . . . .	134
4.5	Architecture and implementation . . . . .	137
4.6	Common document types for software architectures . . . . .	139
4.7	Best-practice rules for documentation. . . . .	143
4.8	Examples of alternative architecture frameworks . . . . .	145
4.9	Test your knowledge. . . . .	149
<b>5</b>	<b>Software Architectures and Quality</b>	<b>151</b>
5.1	Integration with the iSAQB curriculum. . . . .	152
5.2	Evaluating software architectures . . . . .	153
5.3	Prototypes and technical proof of concept . . . . .	161
5.4	Architecture analysis. . . . .	163
5.5	Test your knowledge. . . . .	170
<b>6</b>	<b>Tools for Software Architects</b>	<b>173</b>
6.1	Integration with the iSAQB curriculum. . . . .	173
6.2	General information . . . . .	173
6.3	Requirements management tools . . . . .	175
6.4	Modeling tools . . . . .	176
6.5	Generation tools . . . . .	177
6.6	Static code analysis tools. . . . .	178
6.7	Dynamic analysis tools . . . . .	179
6.8	Build management tools . . . . .	180
6.9	Configuration and version management tools. . . . .	182
6.10	Code management tools . . . . .	183
6.11	Testing tools . . . . .	184
6.12	Documentation tools . . . . .	185
6.13	Test your knowledge. . . . .	186

---

## Appendix

---

<b>A</b>	<b>Sample Questions</b>	<b>189</b>
A.1	Excerpts from the examination regulations . . . . .	189
A.2	Sample Questions . . . . .	191
<b>B</b>	<b>List of Abbreviations</b>	<b>193</b>
<b>C</b>	<b>Glossary</b>	<b>195</b>
<b>E</b>	<b>References</b>	<b>205</b>
	<b>Index</b>	<b>211</b>



---

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Software architecture as an aspect of software engineering. . . . .	2
1.2	iSAQB: The International Software Architecture Qualification Board. . .	4
1.3	Certified Professional for Software Architecture – Foundation and Advanced Level. . . . .	5
1.4	The aim of this book. . . . .	7
1.5	Prerequisites . . . . .	8
1.6	Reader’s guide. . . . .	8
1.7	Target audience. . . . .	9
1.8	Acknowledgements. . . . .	10
<b>2</b>	<b>Software Architecture Fundamentals</b>	<b>11</b>
2.1	Integration with the iSAQB curriculum. . . . .	12
2.1.1	Learning goals . . . . .	12
2.2	Software-intensive systems and software architectures . . . . .	13
2.2.1	What is a software-intensive system? . . . . .	13
2.2.2	Types of software-intensive systems . . . . .	14
2.2.3	The importance of software architecture for a software-intensive system. . . . .	18
2.3	Fundamental software architecture concepts . . . . .	19
2.3.1	What is a software architecture? . . . . .	20
2.3.2	Building blocks, interfaces, and configurations. . . . .	21
2.3.3	Concepts for describing software architectures. . . . .	29
2.3.4	Architectural description and architectural levels . . . . .	33
2.3.5	Interactions between software architecture and environment. . .	35
2.3.6	Quality and value of a software architecture . . . . .	37
2.4	A bird’s-eye view of software architecture design . . . . .	39
2.4.1	Objectives and functions of software architecture design . . . . .	40
2.4.2	Overview of software architecture design. . . . .	41

2.4.3	Interplay between activities and abstraction levels within the design . . . . .	43
2.4.4	A software architect's tasks and relationships with other roles . . . . .	46
2.5	Test your knowledge. . . . .	47
<b>3</b>	<b>Designing Software Architectures</b>	<b>51</b>
3.1	Integration with the iSAQB curriculum. . . . .	52
3.1.1	Learning goals . . . . .	52
3.2	Overview of the architecture design process . . . . .	52
3.3	Design principles and heuristics . . . . .	59
3.3.1	Top-down and bottom-up . . . . .	60
3.3.2	Hierarchical (de)composition . . . . .	61
3.3.2.1	Divide and conquer . . . . .	61
3.3.2.2	Decomposition principles . . . . .	62
3.3.2.3	The "as-simple-as-possible" principle . . . . .	62
3.3.2.4	Separation of concerns . . . . .	63
3.3.3	Lean interfaces and information hiding . . . . .	63
3.3.3.1	Information hiding . . . . .	63
3.3.3.2	Use of interfaces . . . . .	64
3.3.4	Regular refactoring and redesign . . . . .	64
3.4	Architecture-centric development approaches. . . . .	65
3.4.1	Domain-driven design . . . . .	65
3.4.1.1	Functional models as the basis for a design. . . . .	65
3.4.1.2	Systematic management of domain objects. . . . .	66
3.4.1.3	Structuring of the functional domain . . . . .	67
3.4.1.4	Types of domains . . . . .	67
3.4.1.5	Integration of domains . . . . .	68
3.4.2	MDA . . . . .	68
3.4.3	Reference architectures. . . . .	70
3.4.3.1	Generative creation of system building blocks . . . . .	70
3.4.3.2	Aspect orientation. . . . .	70
3.4.3.3	Object orientation. . . . .	71
3.4.3.4	Procedural approaches . . . . .	72
3.5	Techniques for a good design . . . . .	72
3.5.1	Degenerated design. . . . .	73
3.5.2	Loose coupling. . . . .	74
3.5.3	High cohesion . . . . .	75

---

3.5.4	The open/closed principle . . . . .	76
3.5.5	Dependency inversion. . . . .	77
3.5.6	Separation of interfaces . . . . .	78
3.5.7	Resolving cyclic dependencies . . . . .	78
3.5.8	Liskov's substitution principle . . . . .	78
3.6	Architectural patterns . . . . .	80
3.6.1	Adaptable systems . . . . .	80
3.6.1.1	Dependency Injection . . . . .	80
3.6.2	Interactive systems . . . . .	81
3.6.2.1	Model-view-controller pattern . . . . .	81
3.6.2.2	Model-view-presenter pattern. . . . .	82
3.6.2.3	Presentation-abstraction-control. . . . .	83
3.6.3	From chaos to structure . . . . .	84
3.6.3.1	Layered architecture . . . . .	84
3.6.3.2	Pipes and filters. . . . .	85
3.6.3.3	Blackboard . . . . .	86
3.6.4	Distributed systems . . . . .	87
3.6.4.1	Broker. . . . .	87
3.6.4.2	Service orientation . . . . .	88
3.6.4.3	Modularization. . . . .	89
3.6.4.4	Microservices . . . . .	89
3.7	Design patterns. . . . .	90
3.7.1	Adapter . . . . .	90
3.7.2	Observer. . . . .	91
3.7.3	Decorator . . . . .	92
3.7.4	Proxy . . . . .	93
3.7.5	Facade . . . . .	94
3.7.6	Bridge. . . . .	94
3.7.7	State . . . . .	95
3.7.8	Mediator . . . . .	96
3.8	Test your knowledge. . . . .	97
<b>4</b>	<b>Description and Communication of Software Architectures</b>	<b>101</b>
4.1	Integration with the iSAQB curriculum. . . . .	101
4.1.1	Learning goals . . . . .	102
4.2	The CoCoME example. . . . .	102
4.2.1	Use cases in the CoCoME system. . . . .	103
4.2.2	Overview of the structure of the CoCoME system. . . . .	104

4.3	Views and templates . . . . .	105
4.3.1	Well-established views as defined by the iSAQB . . . . .	106
4.3.2	UML diagrams as a notation tool in view descriptions. . . . .	107
4.3.3	View description: high-level structure and an example. . . . .	110
4.3.3.1	High-level structure: template-type view description . . . . .	111
4.3.3.2	Example: Excerpt from a view description for a building block view . . . . .	112
4.3.4	Context view (or context diagram). . . . .	114
4.3.5	Building block view . . . . .	118
4.3.6	Runtime view . . . . .	122
4.3.7	Deployment/infrastructure view . . . . .	126
4.3.8	Interdependencies of architecture views . . . . .	130
4.3.9	Hierarchical refinement of architecture views. . . . .	131
4.4	Technical/cross-cutting concepts in software architectures . . . . .	134
4.4.1	Technical/cross-cutting concepts - sample dimensions . . . . .	135
4.4.2	Error handling . . . . .	135
4.4.3	Security. . . . .	136
4.5	Architecture and implementation . . . . .	137
4.5.1	Sample implementation . . . . .	138
4.6	Common document types for software architectures . . . . .	139
4.6.1	Central architecture description . . . . .	139
4.6.2	Architecture overview. . . . .	140
4.6.3	Document overview . . . . .	141
4.6.4	Overview presentation . . . . .	141
4.6.5	Architecture wallpaper . . . . .	141
4.6.6	Documentation handbook . . . . .	141
4.6.7	Technical Information . . . . .	141
4.6.8	Documentation of external interfaces. . . . .	142
4.6.9	Template. . . . .	142
4.7	Best-practice rules for documentation. . . . .	143
4.7.1	Rule 1: Write from the readers' perspective . . . . .	143
4.7.2	Rule 2: Avoid unnecessary repetition . . . . .	143
4.7.3	Rule 3: Avoid ambiguity. . . . .	143
4.7.4	Rule 4: Standardized organizational structure or templates . . . . .	144
4.7.5	Rule 5: Justify important decisions in writing. . . . .	144
4.7.6	Rule 6: Check the documentation's suitability for use . . . . .	144



---

4.7.7	Rule 7: Uncluttered diagrams . . . . .	145
4.7.8	Rule 8: Regular updates . . . . .	145
4.8	Examples of alternative architecture frameworks . . . . .	145
4.8.1	The 4+1 framework . . . . .	146
4.8.2	RM-ODP . . . . .	147
4.8.3	SAGA . . . . .	148
4.9	Test your knowledge . . . . .	149
<b>5</b>	<b>Software Architectures and Quality</b>	<b>151</b>
5.1	Integration with the iSAQB curriculum . . . . .	152
5.1.1	Learning goals . . . . .	152
5.2	Evaluating software architectures . . . . .	153
5.2.1	Qualitative evaluation . . . . .	153
5.2.1.1	DIN ISO/IEC 25010 . . . . .	153
5.2.1.2	Quality characteristics . . . . .	153
5.2.1.3	Additional quality characteristics . . . . .	155
5.2.1.4	Effects of specific quality characteristics . . . . .	156
5.2.1.5	Tactics and practices for fulfilling quality requirements . . . . .	156
5.2.2	Quantitative evaluation . . . . .	158
5.2.2.1	Checking architecture compliance . . . . .	159
5.2.2.2	Metrics . . . . .	159
5.2.2.3	Cyclomatic complexity . . . . .	160
5.3	Prototypes and technical proof of concept . . . . .	161
5.3.1	Technical proof of concept . . . . .	161
5.3.2	Prototype . . . . .	161
5.3.2.1	Benefits and disadvantages of software prototypes . . . . .	162
5.3.2.2	Types of software prototypes . . . . .	162
5.4	Architecture analysis . . . . .	163
5.4.1	The ATAM method . . . . .	163
5.5	Test your knowledge . . . . .	170
<b>6</b>	<b>Tools for Software Architects</b>	<b>173</b>
6.1	Integration with the iSAQB curriculum . . . . .	173
6.1.1	Learning goals . . . . .	173
6.2	General information . . . . .	173
6.2.1	Costs . . . . .	174
6.2.2	Licenses and licensing conditions . . . . .	174

6.3	Requirements management tools . . . . .	175
6.3.1	Requirements and decision criteria. . . . .	175
6.3.2	Challenges faced by requirements management tools. . . . .	175
6.3.3	Examples . . . . .	175
6.4	Modeling tools . . . . .	176
6.4.1	Requirements and decision criteria. . . . .	176
6.4.2	Challenges faced by modeling tools . . . . .	177
6.4.3	Examples . . . . .	177
6.5	Generation tools . . . . .	177
6.5.1	Requirements and decision criteria. . . . .	178
6.5.2	Challenges faced by code generators . . . . .	178
6.5.3	Examples . . . . .	178
6.6	Static code analysis tools. . . . .	178
6.6.1	Requirements and decision criteria. . . . .	179
6.6.2	Challenges faced by static code analysis tools. . . . .	179
6.6.3	Examples . . . . .	179
6.7	Dynamic analysis tools . . . . .	179
6.7.1	Requirements and decision criteria. . . . .	180
6.7.2	Challenges faced by dynamic analysis tools . . . . .	180
6.7.3	Examples . . . . .	180
6.8	Build management tools . . . . .	180
6.8.1	Requirements and decision criteria. . . . .	181
6.8.2	Challenges faced by build management tools . . . . .	181
6.8.3	Examples . . . . .	181
6.9	Configuration and version management tools. . . . .	182
6.9.1	Requirements and decision criteria. . . . .	182
6.9.2	Challenges faced by configuration and version management tools . . . . .	182
6.9.3	Examples . . . . .	183
6.10	Code management tools . . . . .	183
6.10.1	Challenges faced by code management tools . . . . .	183
6.10.2	Examples . . . . .	183
6.11	Testing tools . . . . .	184
6.11.1	Requirements and decision criteria. . . . .	184
6.11.2	Challenges faced by test tools. . . . .	184
6.11.3	Examples . . . . .	185

6.12 Documentation tools . . . . .	185
6.12.1 Requirements and decision criteria. . . . .	185
6.12.2 Challenges faced by documentation tools. . . . .	185
6.12.3 Examples . . . . .	186
6.13 Test your knowledge. . . . .	186

## Appendix

---

<b>A Sample Questions</b>	<b>189</b>
A.1 Excerpts from the examination regulations . . . . .	189
A.2 Sample Questions . . . . .	191
<b>B List of Abbreviations</b>	<b>193</b>
<b>C Glossary</b>	<b>195</b>
<b>E References</b>	<b>205</b>
<b>Index</b>	<b>211</b>