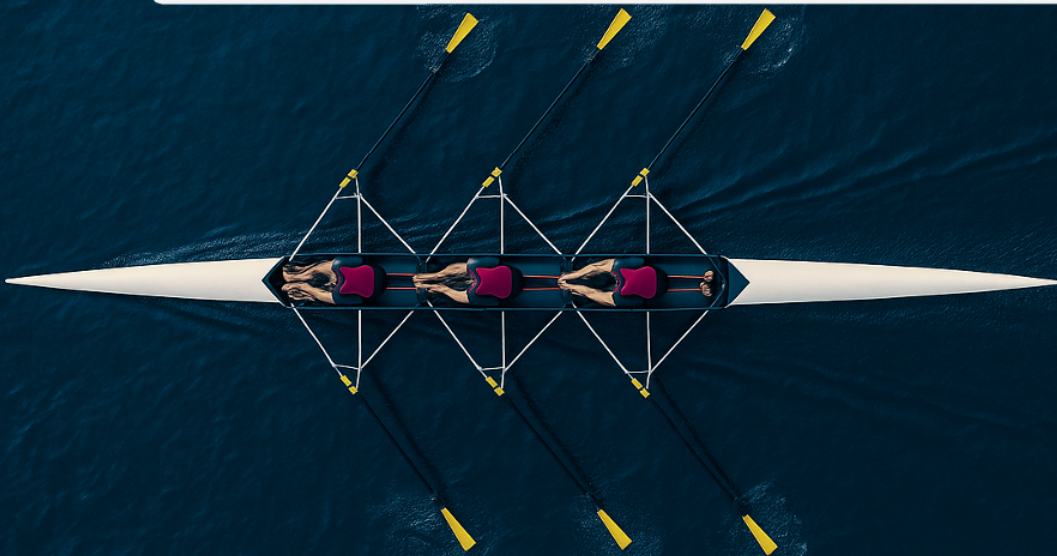




Ronald Melster

The Ultimate Guide to Automotive SPICE®

100+ Best Practices for ASPICE:
Real-World Solutions from 20 Years of Assessment,
Coaching, and Application



BIBLIOGRAPHIC INFORMATION

The Deutsche Nationalbibliothek (German National Library) lists this publication in the German National Bibliography. Detailed bibliographic data is available online at <https://dnb.dnb.de>.

E-Book ISBN: 978-3-9827736-5-0

Automotive SPICE® is a registered trademark of the Verband der Automobilindustrie e.V. (VDA).

The use of trade names, brand names, and product designations in this book, even without specific marking, does not imply that such names are freely available for use under trademark and brand protection law.

All processes and techniques described in this book have been presented with the best possible accuracy. However, errors cannot be completely ruled out.

Neither the author nor the publisher assume any responsibility or liability for any potential damages arising from the application of the methods or data presented.

Copyright and Usage Rights

This work is copyright protected. All rights, including translation, reproduction, and duplication of the book or its parts, are reserved. No part of this work may be used or reproduced without the written permission of the publisher, except for educational purposes within the limits of §53, §54 UrhG.

Publisher: Melster Consulting GmbH

Website: www.melster-consulting.gmbh

© 2025 Ronald Melster. All rights reserved.

Key Message:

This book is not about compliance - it is about real process improvement and engineering success.



Automotive SPICE^{®1} (ASPICE) is often misunderstood. Many engineers and managers see it as an assessment framework, a checklist exercise, or another compliance requirement. However, ASPICE is far more than that - it is a powerful tool for building structured, scalable, and efficient engineering processes.

When ASPICE was introduced, few people were familiar with abbreviations like MAN . 3 or SUP . 1. These terms were known only to a handful of specialists. Today, they are widely recognized across the industry, forming a common language for process names and their corresponding **base practices**.

Despite this shared terminology, significant uncertainties and misconceptions persist, particularly regarding the **practical implementation** of ASPICE expectations. This book addresses those gaps and provides a practical guide for applying ASPICE in real-world engineering environments. It draws from thousands of hours spent conducting **assessments** and **coaching teams**, witnessing both struggling projects and successful implementations.

This book is for those who want to move beyond compliance. It is not just about meeting **assessment criteria** - it is about understanding ASPICE as a tool to create better products and guide organizations toward sustainable **process improvements**.

¹Automotive SPICE[®] is a registered trademark of the Verband der Automobilindustrie e.V. (VDA).

*"Good processes do not slow teams down
- they allow them to move faster with
fewer mistakes."*

– Ronald Melster



Decades of ASPICE implementation and process improvement across various companies and industries have revealed recurring challenges. Many talented teams struggle to realize their full potential when ASPICE is treated as a compliance exercise rather than as a means to enhance product quality, reproducibility, security, and safety.

This book addresses these challenges by providing a practical, real-world guide to implementing, improving, and benefiting from ASPICE.

Who Should Read This Book?

This book is for engineers and developers who want to understand how ASPICE can help them rather than hinder them. It is for project managers who need to balance process discipline with agility. It is for *quality assurance professionals* looking to enhance their ability to support **coaching initiatives, process improvements, and assessment preparation**. It is for executives who want to build a sustainable, high-performance engineering culture.

ASPICE is not about bureaucracy - it is about creating better, more predictable engineering processes that work for both small teams and large organizations.

This book provides practical guidance for implementing process excellence in your engineering environment.

About the Author (Short Version)

Ronald Melster is one of Europe's longest-serving ASPICE assessors and a leading expert in process improvement. Over the past two decades, he has worked with major organizations such as BOSCH, Carmeq, Nippon Seiki, Audi, and Porsche to implement scalable, efficient engineering processes. With this book, his goal is to bridge the gap between ASPICE compliance and real-world success.

For a detailed background, see About the Author on page 333.

Ronald Melster

August 14, 2025

CONTENTS

1.	Motivation	19
1.1.	Why am I writing this book?	19
1.2.	Why am I the one writing this book?	20
1.3.	The Importance of Training and Process Support	21
1.4.	Why am I writing THIS book?	21
2.	Introduction	23
2.1.	Definition of “Process”	23
2.2.	Rationale for ASPICE	27
2.3.	The Evolution of ASPICE: From Early Industry Approaches to Stan- dardization	28
2.4.	The Two Dimensions of ASPICE	29
2.5.	ASPICE is Not Just About Compliance	30
2.6.	Core Processes	31
2.7.	The Engineering Processes	33
2.8.	Process Chapter Structure and Content	34
3.	Project Management (MAN.3)	37
3.1.	Project Definition	38
3.2.	Project Management Rationale	39
3.3.	Sources of Project Management Expectations	41
3.4.	Core Activities of Project Management	42
3.4.1.	Planning	42
3.4.2.	Implement the Plans	45
3.5.	Challenges and Best Practices in Project Management	48
3.5.1.	Challenge: Unclear Project Boundaries	48
3.5.2.	Best Practice: Define Scope Across Three Dimensions	48
3.5.3.	Best Practice: Apply SMART Criteria	49
3.5.4.	Challenge: Maintenance Phase Scope Confusion	50
3.5.5.	Best Practice: Separate Development and Maintenance Activities	50
3.5.6.	Challenge: Activity and Task Terminology Confusion	51
3.5.7.	Best Practice: Apply Work Breakdown Hierarchy	52
3.5.8.	Challenge: Conflicting Agile and ASPICE Terminologies	53
3.5.9.	Best Practice: Establish Clear Terminology Mapping	54
3.5.10.	Challenge: Multi-Level Lifecycle Complexity	55

3.5.11.	Best Practice: Define Lifecycle Hierarchy	55
3.5.12.	Challenge: Multi-Level Project Management Coordination	57
3.5.13.	Best Practice: Define Three Levels of Project Management	58
3.5.14.	Challenge: Competency Management Challenges	62
3.5.15.	Best Practice: Implement Structured Competency Artifacts	63
3.5.16.	Best Practice: Define Standardized Competency Levels	65
3.5.17.	Challenge: Critical Path Management Complexity	66
3.5.18.	Best Practice: Implement Systematic Critical Path Analysis	67
3.5.19.	Best Practice: Critical Path Management in Agile Environments	68
3.5.20.	Challenge: Project Artifact Consistency Management	69
3.5.21.	Best Practice: Implement Structured Consistency Management	70
4.	Quality Assurance (SUP.1)	73
4.1.	Definition of Quality	73
4.2.	Quality Assurance Rationale	74
4.3.	Quality Expectation Sources	75
4.3.1.	Customer Requirements	75
4.4.	Core Activities Quality Assurance	76
4.4.1.	Plan Quality Activities	76
4.4.2.	Conduct Reviews	77
4.4.3.	Perform Process Assessments	77
4.4.4.	Track and Analyze Defects	78
4.4.5.	Measure Quality Performance	78
4.4.6.	Report Quality Status	79
4.4.7.	Follow-up and Verification of Corrective Actions	80
4.4.8.	QA Contribution to Release Decisions	80
4.5.	Challenges and Best Practices in Quality Assurance	81
4.5.1.	Challenge: The QA Independence Paradox	82
4.5.2.	Best Practice: Two-Level Quality Assurance Structure	83
4.5.3.	Best Practice: Disciplinary Separation of Quality Engineers	84
4.5.4.	Best Practice: Establishing QA Independence	85
4.5.5.	Best Practice: Four-Eyes Principle for Quality Verification	86
4.5.6.	Challenge: "End-of-Line" vs. "Built-In" Quality Mindset	87
4.5.7.	Best Practice: Constructive Quality Assurance Approach	88
4.5.8.	Best Practice: Regular Compliance Checks	88
4.5.9.	Best Practice: Self-Organized Reviews in Mature Teams	90
4.5.10.	Challenge: Efficient Reviews	91
4.5.11.	Challenge: Reviews are Misunderstood as Inspections	91
4.5.12.	Best Practice: Comprehensive Review Method Definitions	92

4.5.13.	Best Practice: Tool-Supported Review Implementation	93
4.5.14.	Best Practice: Full Reviews versus Delta Reviews	94
4.5.15.	Best Practice: Review Termination for Immature Artifacts	95
4.5.16.	Challenge: Balancing the Use of Checklists in Reviews	96
4.5.17.	Best Practice: Appropriate Checklist Level and Frequency	97
5.	Configuration Management (SUP.8)	99
5.1.	Definitions	99
5.2.	Configuration Management Rationale	101
5.3.	Configuration Management Requirement Sources	101
5.4.	Core Tasks in Configuration Management	102
5.4.1.	Definition of Configuration Management Strategy	102
5.4.2.	Provision and Setup of Configuration Management System	102
5.4.3.	Identification of Configuration Management Elements	103
5.4.4.	Backup Management	103
5.4.5.	Baselining	103
5.4.6.	Conducting Configuration Management Audits	104
5.5.	Challenges and best practices in Configuration Management	105
5.5.1.	Challenge: Unclear Branching Criteria	105
5.5.2.	Challenge: Undefined Merging Criteria	105
5.5.3.	Best Practice: Establishing Structured Criteria for Branching and Merging	106
5.5.4.	Challenge: Lack of Clear Responsibility for Backup Management	109
5.5.5.	Best Practice: Formalizing Project-Specific Backup Management	109
5.5.6.	Best Practice: Project-Specific Definition of Backup Requirements	110
5.5.7.	Best Practice: Formal Agreement and Monitoring of Backup Im- plementation	111
5.5.8.	Challenge: Archiving at Project and Product End-of-Life	111
5.5.9.	Best Practice: Comprehensive and Systematic Archiving	112
5.5.10.	Challenge: Status Control in Multi-Level Baselining	113
5.5.11.	Best Practice: Baseline of Baselines in Hierarchical System Devel- opment	113
5.5.12.	Best Practice: Integration of Configuration Management and Change Management	114
5.5.13.	Best Practice: Interface to Release Management	115
6.	Problem Resolution Management (SUP.9)	117
6.1.	Definitions	118
6.1.1.	Problem	118

6.1.2.	Distinguishing a Problem from a Risk	119
6.2.	Rationale for Problem Resolution Management	120
6.3.	Sources of Expectations for Problem Resolution Management	121
6.4.	Core Tasks in Problem Resolution Management	123
6.4.1.	Developing a Problem Resolution Strategy	123
6.4.2.	Problem Identification and Documentation	126
6.4.3.	Problem Analysis and Root Cause Investigation	127
6.4.4.	Problem Classification and (Re-)Prioritization	127
6.4.5.	Resolution Planning and Implementation	127
6.4.6.	Communication and Reporting	128
6.4.7.	Verification and Lessons Learned	128
6.4.8.	Trend Analysis and Preventive Measures	128
6.5.	Challenges and Best Practices in Problem Resolution Management (SUP.9)	130
6.5.1.	Challenge: Finding an Appropriate Status Model	130
6.5.2.	Best practice: Status model	131
6.5.3.	Best practice: Status transitions for Problem	132
6.5.4.	Challenge: Confusion Between Urgent Problems and Alert Notifi- cations	133
6.5.5.	Best Practice: Urgent Resolution Strategy	135
6.5.6.	Best Practice: Pre-Analysis	136
6.5.7.	Challenge: Classification of Problems	137
6.5.8.	Best Practice: Classification of Problems Along the Architecture	137
6.5.9.	Best Practice: Classification of Problems Along the Process Steps	138
6.5.10.	Best Practice: Interface to Change Management	140
6.5.11.	Best Practice: Tool Support with Mandatory Fields	141
6.5.12.	Challenge: Trend Analysis	142
7.	Change Request Management (SUP.10)	145
7.1.	Definitions	147
7.1.1.	Change	148
7.1.2.	Modifications That Do Not Qualify as Changes	150
7.2.	Rationale for Change Request Management	151
7.3.	Sources of Expectations for Change Request Management	152
7.4.	Core Tasks in Change Request Management	153
7.4.1.	Create a Change Request Management Strategy	154
7.4.2.	Receive and Register Change Requests	157
7.4.3.	Perform Impact Analysis	158
7.4.4.	Decide on Acceptance, Rejection, or Deferral	159

7.4.5.	Track Implementation	160
7.4.6.	Verify the Implemented Change	161
7.5.	Challenges and Best Practices in Change Request Management . . .	162
7.5.1.	Challenge: Defining a Clear and Practical Status Model	162
7.5.2.	Best Practice: Status Model	162
7.5.3.	Best Practice: Status Transitions for Change Requests	165
7.5.4.	Challenge: Change vs. Heavy Development	167
7.5.5.	Best Practice: Apply Change Request Management Only After Baselin- ing	167
7.5.6.	Best Practice: Two Levels of Approval	168
7.5.7.	Challenge: Implementation Before Approval	169
7.5.8.	Best Practice: Two Levels of Acceptance — Internal and Stakeholder	170
7.5.9.	Challenge: Late Changes	171
7.5.10.	Best Practice: Structured Change Handling Under Time Pressure .	172
7.5.11.	Challenge: Informal Change Requests Over the Phone	173
7.5.12.	Best Practice: Ensure Formalization of All Change Requests	174
7.5.13.	Challenge: Uncontrolled Changes to Project Scope (Scope Creep) .	175
7.5.14.	Best Practice: Apply Change Management to Scope Modifications .	176
7.5.15.	Challenge: Change Request Management Limited to Product Ar- tifacts	177
7.5.16.	Best Practice: Include PMT Changes	178
7.5.17.	Best Practice: Ensure Traceability Between Problems and Change Requests	180
7.5.18.	Challenge: Missing Traceability Between Change Requests and Changed Artifacts	181
7.5.19.	Best Practice: Ensure Traceability Between Change Requests and Changed Artifacts	182
7.5.20.	Best Practice: Document Changes in the Release Notes	183
7.5.21.	Best Practice: Include Change Requests in the Project Scope	183
8.	Requirements Engineering	187
8.1.	Definition “Requirement”	187
8.2.	Requirements Engineering Rationale	189
8.3.	Sources of Expectations for Requirements Engineering	190
8.4.	Core Tasks in Requirements Engineering	191
8.4.1.	Create an approach for requirements engineering	192
8.4.2.	Derive requirements	192
8.4.3.	Analyze requirements	193
8.4.4.	Create verification criteria	194

8.4.5.	Allocation of requirements to elements of the architecture	195
8.4.6.	Create bi-directional traceability	195
8.4.7.	Establish consistency	196
8.4.8.	Agree	197
8.4.9.	Baseline	197
8.4.10.	Communicate	198
8.5.	Challenges and best practices in Requirements Management	199
8.5.1.	Challenge: Limited stakeholder integration	199
8.5.2.	Challenge: Stakeholder expectations	200
8.5.3.	Challenge: Customer defines very detailed software requirements .	201
8.5.4.	Best practice: Derive and agree on a common understanding	202
8.5.5.	Best practice: Accept “solutions” as design proposals	203
8.5.6.	Challenge: Different levels of requirements	204
8.5.7.	Best practice: Define a common approach	206
8.5.8.	Best practice: Lifecycle for requirements	207
8.5.9.	Challenge: System requirements are copied to software require- ments level	208
8.5.10.	Best practice: Define allocation via HSI, not by duplication	209
8.5.11.	Best practice: Create a Requirements Engineering Strategy Across Domains	209
9.	Architecture and Design in ASPICE	211
9.1.	Definitions	212
9.1.1.	Architecture definition and scope	212
9.1.2.	Component definition and characteristics	213
9.1.3.	Detailed design definition and characteristics	214
9.1.4.	Unit definition and characteristics	214
9.1.5.	Distinguishing architecture from detailed design	215
9.2.	Architecture necessity and benefits	215
9.3.	Sources of Expectations for architectural processes	216
9.4.	Core Tasks	217
9.4.1.	Define (static) structure	217
9.4.2.	Define interfaces	218
9.4.3.	Define dynamic behaviour	219
9.4.4.	Agree	220
9.4.5.	Baseline	221
9.4.6.	Communicate	222
9.4.7.	Maintain	223

9.5.	Differences between the architectures	224
9.5.1.	System	225
9.5.2.	Software only	227
9.5.3.	Detailed Design	228
9.6.	Challenges and best practices in Architecture and Detailed Design .	229
9.6.1.	Challenge: Uncertainty about how to start designing the architecture	229
9.6.2.	Best practice: Create a Context Diagram First	230
9.6.3.	Challenge: Architectural decisions	231
9.6.4.	Best practice: Integrate rationale into architecture changes	233
9.6.5.	Challenge: Understanding and representing dynamic behavior . .	233
9.6.6.	Best practice: Represent system-wide dynamic behavior explicitly .	234
9.6.7.	Best practice: Document absence of dynamic behavior explicitly . .	235
9.6.8.	Challenge: How to deal with reuse	235
9.6.9.	Best practice: Clearly define the supplied software components . .	236
9.6.10.	Best practice: Agree on the Responsibility for Testing	237
9.6.11.	Challenge: Interfaces Are Described Only Informally	237
9.6.12.	Best practice: Create a database for all interfaces	238
9.6.13.	Challenge: Estimation of resource consumption	239
9.6.14.	Best practice: Start with Measuring the Actual Consumption	240
9.6.15.	Best practice: Agree on Runtime Consumption	240
9.6.16.	Best practice: Define Component Requirements	241
9.6.17.	Challenge: Implement First	241
9.6.18.	Best practice: Detailed Design as Part of the Source Code	242
9.6.19.	Challenge: Traceability from Architecture to Requirements	243
9.6.20.	Best practice: Define the Components in the Requirements Man- agement Tool	243
10.	Implementation Strategies	245
10.1.	Definitions	245
10.2.	Implementation Rationale	249
10.3.	Sources of Expectations for Implementation	250
10.4.	Core Tasks	250
10.4.1.	Implement	251
10.4.2.	Unit Testing	251
10.4.3.	Static Code Analysis	251
10.4.4.	Baseline	252
10.4.5.	Merging	252
10.5.	Challenges and best practices in Implementation	252
10.5.1.	Challenge: Term “Unit”	252

10.5.2.	Challenge and Best Practice: Term Unit in a Model-Based Development Environment	253
10.5.3.	Challenge: Unit Interfaces	254
10.5.4.	Challenge: Unit Verification	254
10.5.5.	Best Practice: Do Unit Verification Once	255
10.5.6.	Best Practice: Developers as Unit Testers	256
10.5.7.	Challenge: Traceability from Software Units to Software Requirements	256
10.5.8.	Best Practice: Ensure Traceability Between Design Units and Software Requirements	257
11.	Integration Strategies	259
11.1.	Definitions	259
11.1.1.	System Definition	260
11.2.	Integration Rationale	263
11.3.	Sources for Expectations for Integration Processes	264
11.4.	Integration Verification Across Software and System	265
11.5.	Core Tasks	267
11.5.1.	Software: Integration	267
11.5.2.	Software: Integration Verification	268
11.5.3.	Software: Verification of the structure	269
11.5.4.	Software: Verification of the interfaces	270
11.5.5.	SW: Verification of the dynamic behaviour	271
11.5.6.	SW: Measurement of the resource consumption	272
11.5.7.	System: Integration	273
11.5.8.	System: Integration Verification	274
11.5.9.	Create traceability	275
11.5.10.	Report about the verification	276
11.6.	Challenges and best practices in Integration	277
11.6.1.	Challenge: integration Responsibilities across organisational units	278
11.6.2.	Best Practice: Clear definition of responsibilities	279
11.6.3.	Best Practice: Clarify the architecture	280
11.6.4.	Best Practice: Hand over of integration test cases and integration manual	281
11.6.5.	Challenge: Integration test level often misunderstood	282
11.6.6.	Best Practice: Re-use of test cases for SWE.5 and SWE.6	282
11.6.7.	Challenge: System integration – incomplete work products from the domains	283
11.6.8.	Best Practice: Check incoming work products	284

11.6.9.	Challenge: Unclear responsibility for interface verification	285
11.6.10.	Challenge: Mixed-domain integration objects (e.g., microcontroller with software)	286
11.6.11.	Challenge: System contains multiple ECUs	287
11.6.12.	Challenge: Integration tests misunderstood as qualification	288
11.6.13.	Best Practice: Component requirements as test reference	289
11.6.14.	Best Practice: Require availability and approval of SWE.5 artifacts before execution	289
11.6.15.	Challenge: Integration despite incomplete units or component ver- ification	290
11.6.16.	Best Practice: Document integration readiness status per component	291
11.6.17.	Challenge: Logging and traceability – test runs to test logs	292
11.6.18.	Best Practice: External logging reference via file or database	293
11.6.19.	Challenge: Traceability from structure to review protocols	294
11.6.20.	Best practice: Describe the mechanism in a central strategy	295
11.6.21.	Challenge: Regression test selection unclear	296
11.6.22.	Best Practice: Impact-based regression selection	297
11.6.23.	Best Practice: Define verification coverage targets	298
11.6.24.	Best Practice: Document test rationale and context in the report	299
11.6.25.	Best Practice: Define recipient-specific communication strategy	300
12.	Verification	301
12.1.	Definitions	301
12.1.1.	Distinction Between Verification and Validation	302
12.2.	Verification Rationale	303
12.3.	Sources of Expectations for Verification Processes	303
12.4.	Core Tasks	304
12.4.1.	Define Verification Strategy	304
12.4.2.	Review the Requirements	305
12.4.3.	Derive and Specify Verification Cases	306
12.4.4.	Prepare and Maintain Verification Environment	307
12.4.5.	Implement Test Cases	308
12.4.6.	Execute Verification	309
12.4.7.	Baseline the Verification Cases	309
12.4.8.	Document Results and Evaluate Criteria	310
12.4.9.	Ensure Traceability	311
12.4.10.	Create Report	312
12.4.11.	Communicate Results	313
12.4.12.	Handling Deviations	314

12.5.	Challenges and Best Practices in Verification	315
12.5.1.	Challenge: Requirements Are Never Approved	315
12.5.2.	Best Practice: Approve the Set of Requirements Valid for a Specific Release	316
12.5.3.	Challenge: Test Cases Are Derived from Outdated Requirement Versions	316
12.5.4.	Best Practice: Ensure Synchronization of Test Artifacts with Approved Requirements	317
12.5.5.	Challenge: Requirements Are Not Testable	317
12.5.6.	Best Practice: Define Guidelines for the Formulation of Requirements	318
12.5.7.	Best Practice: Involve the Test Team Early	318
12.5.8.	Challenge: Verification of Non-Functional Requirements	319
12.5.9.	Best Practice: Justify Non-Functional Requirements	319
12.5.10.	Best Practice: Exempt the Non-Functional Requirements from the Traceability KPIs	320
12.5.11.	Best Practice: Define a Verification Criterion	321
12.5.12.	Best Practice: Separate Test Specification from Test Implementation	322
12.5.13.	Best Practice: Automate Test Execution and Reporting	323
12.5.14.	Best Practice: Use Full Test Suite Execution as Regression Strategy .	324
12.5.15.	Challenge: Test Reports from CI/CD Are Never Analyzed	325
12.5.16.	Best Practice: Define Verification Monitoring Frequency and Triggers	325
12.5.17.	Challenge: Test engineers are drawn into problem resolution	326
12.5.18.	Best Practice: Clear Separation from SUP.9	327
13.	Conclusion	329
13.1.	Key Takeaways from this Book	329
13.2.	Next Steps - How to Apply ASPICE in Practice	330
13.3.	Assessment Preparation Tools	331
13.4.	Peer Exchange - Community of Practice	331
13.5.	Personal Consultation	332
14.	About the Author	333
A.	List of Best Practices	335
B.	Key ASPICE Concepts	339