

# SOFTWARE ARCHITECT

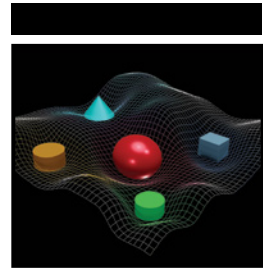
MICHAEL BELL

WILEY



# **Software Architect**





---

# Software Architect

Michael Bell

WILEY

Copyright © 2023 by John Wiley & Sons. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.  
Published simultaneously in Canada and the United Kingdom.

ISBN: 978-1-119-82097-0

ISBN: 978-1-119-82098-7 (ebk.)

ISBN: 978-1-119-82099-4 (ebk.)

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at [www.wiley.com/go/permission](http://www.wiley.com/go/permission).

**Trademarks:** Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

If you believe you've found a mistake in this book, please bring it to our attention by emailing our reader support team at [wileysupport@wiley.com](mailto:wileysupport@wiley.com) with the subject line "Possible Book Errata Submission."

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at [www.wiley.com](http://www.wiley.com).

**Library of Congress Control Number:** 2022951780

Cover image: Courtesy of Michael Bell

Cover design: Wiley



## About the Author

**Michael Bell** is an American novelist, artist, producer, and enterprise solution architect, chiefly recognized for developing the Incremental Software Architecture methodology (ISAM), Service-Oriented Modeling Framework (SOMF), Cloud Computing Modeling Notation (CCMN), and the Multidimensional Software Architecture Construction (MSAC). His innovative research and publications in the fields of software architecture, service-oriented architecture, microservices, artificial intelligence (AI), cloud computing, and big data are recognized internationally for their contribution to the software design and development communities. He has consulted for organizations including J.P. Morgan Chase, Citibank, Bank One, UBS-Paine Webber, American Express, AIG, and the U.S. government. He is the best-selling author of software architecture books, and he offers a variety of enterprise integration solutions for back-end and customer-facing systems.







## About the Technical Editor

**Paul C. Martello** is a technical writer with more than 18 years of experience in IT. Before becoming a technical writer, he held roles ranging from elementary and business teacher in New York to history teacher for Fairfax County Public Schools in Virginia. In 2005, Paul was selected to participate in the Relief International Schools Online Teacher Exchange in Bangladesh, integrating technology in schools. He loves watching his favorite football team, the Buffalo Bills, traveling, and spending time with his family in Bristow, Virginia.

## About the Contributing Editors

**Noreen O'Brien** has spent the better part of her life writing, editing, and creating through various mediums. From her start as a reporter to the freelance editing work she does today, she has been instrumental in the production of a multitude of papers, dissertations, and documents. She lives in Richmond, Virginia, with her son, Liam.

**Monica Gagnier** is an experienced editor who has worked with Michael Bell on previous books. A graduate of Syracuse University's S.I. Newhouse School of Public Communications, Gagnier is a seasoned financial journalist who has worked at such publications as BusinessWeek (now Bloomberg Businessweek), the New York Post, and the Albuquerque Journal. She lives in Santa Fe, N.M.





# Contents at a Glance

<b>Introduction</b>		<b>xxiii</b>
<b>Part 1</b>	<b>Software Architect Capability Model</b>	<b>1</b>
<b>Chapter 1</b>	<b>Software Architect Capability Model</b>	<b>3</b>
<b>Part 2</b>	<b>Software Architecture Career Planning</b>	<b>29</b>
<b>Chapter 2</b>	<b>Types of Software Architects</b>	<b>31</b>
<b>Chapter 3</b>	<b>Career Planning for Software Architects: A Winning Strategy</b>	<b>73</b>
<b>Chapter 4</b>	<b>Self-Assessment for Software Architects</b>	<b>117</b>
<b>Part 3</b>	<b>Software Architecture Toolbox</b>	<b>149</b>
<b>Chapter 5</b>	<b>Employing Innate Talents to Provide Potent Organizational Solutions</b>	<b>151</b>
<b>Chapter 6</b>	<b>Software Architecture Environment Construction</b>	<b>173</b>
<b>Chapter 7</b>	<b>Structural Construction of Software Implementations in Multidimensional Environments</b>	<b>223</b>
<b>Part 4</b>	<b>Software Architecture Interview Preparations</b>	<b>285</b>
<b>Chapter 8</b>	<b>Preparing for a Software Architecture Interview: A Winning Strategy</b>	<b>287</b>
<b>Chapter 9</b>	<b>An Outline for Software Architecture Job Interview Questions</b>	<b>337</b>
<b>Index</b>		<b>369</b>





# Contents

<b>Introduction</b>	<b>xxiii</b>
<b>Part 1      Software Architect Capability Model</b>	<b>1</b>
<b>Chapter 1    Software Architect Capability Model</b>	<b>3</b>
Software Architect Capability Model: Benefits	4
How Should Organizations Utilize the Software Architect Capability Model?	4
Why Create a Personal Software Architect Capability Model?	5
Rudimentary Guiding Principles	6
Software Architect Capability Model Creation Process	6
Requirements Drive Architecture Solutions	7
Requirements Issued by Problem and Solution Domain Entities	7
How Do the Problem and Solution Domains Collaborate?	7
Important Facts to Remember	9
Create a Software Architect Capability Model in Five Steps	9
Step 1: Provide Requirements and Specifications	10
Business Requirements	10
Technical Specifications	11
Ensure Clear Requirements	11
Step 2: Identify Software Architecture Practices	12
Establish Architecture Practices	12
Step 3: Establish Software Architecture Disciplines	13
Apply Architecture Disciplines to Architecture Practices	14
Applying Disciplines to the Application Architecture Practice	14
Applying Disciplines for the Data Architecture Practice	16
Step 4: Add Software Architecture Deliverables	17
About Software Architecture Deliverables	17
Add the Deliverables Section	18

Step 5: Quantify Skill Competencies	21
Quantifying Architecture Skills	22
Measuring the Application Architect Skill Levels	22
Measuring Data Architect Skill Levels	24
Skill Competency Patterns for Architects	25
How Can Organizations Utilize the Skill Competency Pattern?	26
How an Individual Can Utilize the Skill Competency Pattern	27
Interview Questions	28
<b>Part 2 Software Architecture Career Planning</b>	<b>29</b>
<b>Chapter 2 Types of Software Architects</b>	<b>31</b>
Business Needs for Technological Solutions	32
Business Needs for Software Architecture: Strategic Collaboration	32
How Does Software Architecture Respond to Business Needs?	33
Business Needs for Software Architecture: Technological Mediation	33
How Could Technological Mediation Efforts Be Utilized?	34
Business Needs for Software Architecture: Technological Implementation	34
How Does the Implementation of Software Products Meet Business Needs?	34
Organizational Leading Software Architect Levels	35
Ranking Leading Software Architects	35
Collaboration Hierarchy of Leading Software Architects	36
Level I: Enterprise Architect Responsibilities	38
Enterprise Architect Summary of Responsibilities	38
Enterprise Architect Responsibility Table	39
Level II: Solution Architect Responsibilities	40
Solution Architect Summary of Responsibilities	41
Solution Architect Responsibility Table	42
Level III: Application Architect Responsibilities	44
Application Architect Summary of Responsibilities	44
Application Architect Responsibilities Table	46
Comparing Responsibilities of Leading Software Architects	48
Types of Domain Software Architects	49
Data Architect	49
Data Architect Summary of Responsibilities	50
Data Architect Responsibilities Table	51
Cloud Architect	51
Cloud Architect Summary of Responsibilities	54
Cloud Architect Responsibilities Table	55
Security Architect	57
Security Architect Summary of Responsibilities	58
Security Architect Responsibilities Table	60

Business Architect	62
Business Architect Summary of Responsibilities	62
Business Architect Responsibilities Table	63
Collaboration Between Leading Software Architects and Domain Software Architects	65
Use Case I: Collaboration Between an Application Architect and a Data Architect	66
Application Architect and Data Architect Collaboration Table	66
Use Case II: Solution Architect and Security Architect Solution Architect and Security Architect Collaboration Table	68
Use Case III: Business Architect and Enterprise Architect Collaboration	70
Business Architect and Enterprise Architect Collaboration Table	70
<b>Chapter 3 Career Planning for Software Architects: A Winning Strategy</b>	<b>73</b>
Software Architecture Career Planning Process	74
Career Planning Step 1: Conduct Self-Discovery	75
Discovery of Technological and Social Talents	75
Career Planning Self-Discovery Subjects	76
Career Planning Step 2: Pursue Research	76
Formal Education, Training, and Certification	77
Employment Opportunities and Interviews	77
Subjects of Research	77
Career Planning Step 3: Devise an Approach	78
Setting Software Architecture Career Goals	78
Setting Software Architecture Career Milestones	80
Decision-Making	81
Action Planning	82
Career Planning Step 4: Plan Career Execution	85
Use Case I: A Software Architecture Career Execution Plan with Alternative Tasks	85
Use Case II: Optimized Software Architecture Execution Plan	88
Self-Discovery Process: The Six Ws	89
The “Why”	90
The “Who”	91
The “What”	92
Self-Discovery Questions for Software Architecture Candidates	93
Self-Discovery Queries for Software Architects	93
The “Where”	94
The “When”	95
The “How”	96
“How” Self-Queries for Software Architecture Applicants	97
“How” Self-Questions for Practicing Software Architects	97

Carving a Software Architecture Career Path	98
The 4D Software Architecture Career Perspectives	99
Social-Driven Career Perspective	100
Social-Driven Career Chart	100
Carve Out a Social-Driven Career Chart	101
Social-Driven Career Path	102
Create a Social-Driven Career Path	102
Technology-Driven Career Perspective	103
Technology-Driven Career Chart	104
Create a Technology-Driven Career Chart	105
Technical-Driven Career Path	106
Develop a Technical-Driven Career Path	106
Leadership-Driven Career Perspective	107
Leadership-Driven Career Chart	108
Create a Leadership-Driven Career Chart	109
Leadership-Driven Career Path	110
Develop a Leadership-Driven Career Path	110
Strategy-Driven Career Perspective	112
Strategy-Driven Career Chart	112
Create a Strategy-Driven Career Chart	114
Strategy-Driven Career Path	114
Develop a Strategy-Driven Career Path	115
<b>Chapter 4 Self-Assessment for Software Architects</b>	<b>117</b>
Social Intelligence	118
Teamwork	118
Partnership	119
Self-consciousness	119
Communication	120
Networking	120
Soft Skills	120
Trust Building	121
Learning from Others	121
Negotiation	122
Self-presentation	122
Teleworking	123
Fellowship	123
Self-sufficiency	124
Handling Customer Relationships	124
Social Intelligence Skill Assessment	124
Software Architecture Practice	126
Software Architecture Strategy	126
Software Architecture Vision	127
Software Architecture Role	127
System Integration	128
Interoperability	128



Software Reuse	129	
Distributed Architecture Model	129	
Federated Architecture Model	129	
Architecture Styles	130	
Architecture and Design Patterns	130	
Componentization	130	
Software Architecture Frameworks	131	
Software Development	131	
Software Architecture Practice Skill Assessment	132	
Leadership	133	
Managing Time	134	
Decision-Making	134	
Problem-solving	134	
Diversity, Equity, and Inclusion	135	
Responsibility and Accountability	135	
Hiring Preferences	136	
Creative Thinking	136	
Critical Thinking	136	
Being Proactive	137	
Establishment of Trust	137	
Administrative Duties	138	
Coaching and Training	138	
Team Building	139	
Resolving Conflicts	139	
Assessment of Leadership Competencies	140	
Strategy	141	
Software Architecture Strategy	142	
Strategic Thinking	142	
Problem Identification	142	
Problem-solving	143	
Abstraction	143	
Generalization	144	
Visualization	144	
Software Design Approaches	145	
Simplification	145	
Analytical Capabilities	145	
Influencing	146	
Promoting Culture	146	
Strategy Execution Plan	147	
Assessment of Strategic Competencies	147	
<b>Part 3</b>	<b>Software Architecture Toolbox</b>	<b>149</b>
<b>Chapter 5</b>	<b>Employing Innate Talents to Provide Potent Organizational Solutions</b>	<b>151</b>
Innate Skills Promote Software Architecture Effectiveness	152	
Remember: Survival, Survival, Survival	152	
Consequences of Failing to Invoke Innate Talents	153	

Employ Chief Innate Talents to Become an Effective Software Architect	154
The Power of Creativity	154
The Benefits of Unleashing Software Architecture Creativity	155
Unleash the Power of Software Architecture Creativity	155
The Potency of Imagination	157
The Benefits of Harnessing Imagination	158
Unleash the Power of Imagination	159
Software Design Aesthetic	162
Technical Proficiency and Aesthetic Talents Drive Software Design	162
The Chief Contribution of Design Aesthetic Talents to Software Architecture	163
Curiosity Attributes	167
The Contribution of Curiosity to Software Architecture	167
The Influencing Facets of Curiosity on Software Architecture Practices	168
<b>Chapter 6 Software Architecture Environment Construction</b>	<b>173</b>
Benefits of the Software Architecture Environment Construction Discipline	174
Must Haves: Problem Statements and Requirements	174
Never Start a Software Design Project Without Understanding the Problems	175
Never Start a Software Design Project Without Requirements	176
Software Architecture Structures	176
Micro Level: Multidimensional Structures of Software Implementations	176
Macro Level: 3D Software Architecture Environment Structure	177
Software Architecture Environment: Driven by an Uncontrolled Quantum Landscape Behavior	178
Software Architecture Environment: An Intelligent Topological Space	179
Deformation Aspects of a Multidimensional Software Architecture Environment	181
Entanglement Effects in a Software Architecture Environment	182
Software Architecture Environment Forces Drive Software Behavior	183
Probability Assessment of Software Operations and Behavior	184
Software Architecture Environment Positive and Negative Forces	184
Software Architecture Environment Gravitational Forces	185
The Impetus for Granting Software Architecture Gravitational Powers to Software Implementations	186
Software Architecture Gravitational Force Intensity	187

The Cost of Unbalanced Software Architecture	187
Environment Gravitational Forces	187
Competing Software Architecture Environment Forces	188
Software Architecture Environment: A Survival Game	
Space	188
Maintaining a Pragmatic Balance Between Competing	
Software Architecture Forces	189
Mitigating the Competing Forces Challenge	190
Software Architecture Environment Harmonizing and	
Disharmonizing Forces	190
Chief Properties of Harmonizing Forces in a Software	
Architecture Environment	191
Chief Properties of Disharmonizing Forces in a Software	
Architecture Environment	193
Genetic Encoding of a Software Architecture Environment	194
Difficulties of Restructuring a Software Architecture	
Environment	194
Encoding a Software Architecture Environment	195
Influences on Social, Behavioral, and Business Goals	195
Software Architecture Environment Construction Life Cycle	196
Software Architecture Environment Construction Process	197
Creating a Software Architecture Environment Construction	
Balance Table	197
Software Architecture Environment Construction Design	
Activities	199
Use Case I: Software Architecture Environment Composition	
and Decomposition Design Activities	201
Design-Time vs. Runtime Environment Composition and	
Decomposition Design Activities	201
Composition and Decomposition Design Methods	202
Composition and Decomposition Process Outline	203
Use Case II: Software Architecture Environment Integration	
and Disintegration Design Activities	204
When to Apply Integration and Disintegration Design	
Activities	205
Integration and Disintegration Design Methods	205
Integration and Disintegration Process Outline	206
Use Case III: Software Architecture Environment	
Centralization and Decentralization Design Activities	208
When to Employ the Software Environment Centralization	
and Decentralization Design Activities	208
Centralization and Decentralization Design Methods	209
Software Architecture Environment Centralization and	
Decentralization Process Outline	210
Use Case IV: Software Architecture Environment Elasticity	
and Inelasticity Design Activities	211

When to Employ Elasticity and Inelasticity Design Activities	212
Elasticity and Inelasticity Design Methods	213
Software Architecture Elasticity and Inelasticity Design Process Outline	214
Use Case V: Software Architecture Environment Synchronization and Desynchronization Design Activities	215
When to Employ Environment Synchronization and Desynchronization Design Activities	216
Environment Synchronization and Desynchronization Design Methods	216
Software Architecture Environment Synchronization and Desynchronization Design Process Outline	218
Construction Laws of a Software Architecture Environment	219
Best Practices for Software Architecture Environment Construction	220
<b>Chapter 7 Structural Construction of Software Implementations in Multidimensional Environments</b>	<b>223</b>
Software Architecture Solids: Rudimentary Geometrical Design Structures	224
Atomic Solid	225
Composite Solid	227
Monolithic Solid	228
Interface Solid	229
Pipe Solid	230
Inclusive Utilization of Pipe Solids	231
Exclusive Utilization of Pipe Solids	232
Internal Utilization of Pipe Solids	233
Data Solid	234
Software Architecture Solids' Attribute Summary	236
Software Architecture Dimensional Model	237
Software Architecture: Zero Dimension	238
Software Architecture: One Dimension	239
Software Architecture: Two Dimensions	240
What Impacts the Length and Width Dimensions of a 2D Software Structure?	241
Software Architecture: Three Dimensions	242
Volumes of 3D Software Structures	242
Increase in Software Architecture Level of Specificity in a 3D Computing World	243
Software Population Sustainability in an Architecture Environment Space: A Capacity Planning Challenge	245
Comparative Perspectives in a Software Architecture Space	246
3D Software Structures in a Software Architecture Computing Space	247

The Impetus for Establishing a 3D Software Architecture Space	247
Chief Features of Software Architecture Computing Space	249
Influences of Software Structures on Software Architecture Computing Space	250
Relative Positions in a 3D Software Architecture Computing Space	250
Coordinate Axes: Skeleton of a Software Architecture Computing Space	251
Software Architecture Computing Space Logical Coordinate System	252
Cardinal and Intercardinal Physical Directions in Software Architecture Computing Space	253
Applying Cardinal and Intercardinal Directions to Software Architecture Computing Space	254
Marrying a Logical Coordinate System with Cardinal and Intercardinal Physical Directions System	255
Leveraging the Z-Axis to Create Floors in a Software Architecture Computing Space	256
Distribution Styles of 3D Software Implementations in an Architecture Computing Space	257
Federated Distribution Style	258
Flooring Distribution Style	260
Symmetrical and Asymmetrical Distribution Styles	261
Symmetrical Distribution Style	261
Asymmetrical Distribution Style	263
Construction Life Cycle of Software Implementations	264
Software Construction Process	265
Creating a Software Construction Balance Table	265
Software Construction Design Activities	266
Use Case I: Thicken and Contract Design Activities	267
When to Apply Thicken and Contract Design Activities	268
Thicken and Contract Design Methods	269
Software Structure Thickening and Contracting Process Outline	270
Use Case II: Lengthen and Shorten Design Activities	272
When to Apply the Lengthen and Shorten Design Activities	273
Lengthen and Shorten Design Methods	273
Software Structure Lengthening and Shortening Process Outline	275
Use Case III: Layer and Delayer Design Activities	277
When to Apply Layer and Delayer Design Activities	277
Layer and Delayer Design Methods	278
Layer and Delayer Process Outline	279

	Governing Laws for Software Construction in a 3D Computing World	281
	Best Practices for Constructing Software Implementations	282
<b>Part 4</b>	<b>Software Architecture Interview Preparations</b>	<b>285</b>
<b>Chapter 8</b>	<b>Preparing for a Software Architecture Interview: A Winning Strategy</b>	<b>287</b>
	Software Architecture Job Interview Strategy	288
	Preparing a Job Interview Defense Plan	288
	Preparing a Job Interview Attack Plan	289
	Software Architecture Job Interview Preparation Model	290
	Software Architecture Job Interview Defense Plan	291
	Study and Analyze the Job Description	291
	Start with Identifying the Scope of the Software Architecture Job Requirements	292
	Dive Deep into the Software Architect Job Description	293
	Start with Analyzing the Summary Portion of the Job Requirements	294
	Create a Findings Table Version I for the Job Description	295
	Next, Analyze the Responsibilities Portion of the Job Requirements	296
	Then, Update the Findings Table Version II of the Job Description	296
	Last, Analyze the Software Architect Skills Portion of the Job Requirements	297
	Do Not Forget to Update the Findings Table of the Job Description	298
	Create a Software Architect Skill Competency Model for the Job Description	299
	Skill Competency Model's Requirements and Practices	300
	Skill Competency Model's Disciplines	301
	Design Discipline's Deliverables	301
	Cybersecurity Discipline Deliverables	301
	Products Selection and Evaluation Discipline's Deliverables	302
	SDLC Discipline's Deliverables	302
	The Competency Part of the Skill Competency Model	303
	Discover the Personal Knowledge Gap Before Attending a Job Interview	303
	Assess Whether the Next Software Architecture Job Is a Strategic Career Move	304
	Conduct a Software Architecture Mock Interview	305
	Prepare a Software Architecture Interview Cheat Sheet	306
	Prepare for Possible Software Architecture Interview Questions	307
	Software Architecture Job Interview Attack Plan	308
	Study the Hiring Organization's Business	309
	Start by Finding Information About the Hiring Organization	309

Leveraging Business Knowledge During an Interview	311
Understand the Business Model	312
Get Familiar with the Hiring Company’s Culture	314
Conduct a Quick SWOT Analysis	315
Understand the Hiring Organization’s Technology	316
Technological Information Sources	316
Discover the Environment’s Technology Stack	318
Learn About the Development Technology Stack	319
Study the Applications	320
Identify Specific IT Projects	321
Demonstrate Enterprise Architecture Knowledge of the Hiring Organization	321
Adopt Software Architecture Lingo	323
Use Design Patterns Vocabulary	323
Use the Software Architecture Guidelines Lingo to Communicate Solutions	324
Remember Software Architecture Tools	328
Classification of Software Architecture Tools	329
Especially Prepare for Architecture Visualization Tools Questions	332
Get Familiar with Software Architecture Analysis and Evaluation Methods	333
Be Aware of Early Architecture Evaluation Methods	334
Be Aware of Late Architecture Evaluation Methods	335
Talk About Software Architecture Analysis Standards	335
<b>Chapter 9 An Outline for Software Architecture Job</b>	
<b>Interview Questions</b>	<b>337</b>
Behavioral Questions	338
Communication	339
Interpersonal Relationships	340
Software Architecture Leadership	340
Skill Assessment Questions	341
Software Architecture Attributes Questions	342
Software Architecture LifeCycle Questions	343
Software Architecture Concepts Questions	346
Design Building Blocks Concepts	347
Employ Design Building Blocks Concepts to Depict Solutions	347
Prepare for the “How to Design” Interview Questions	348
Software Architecture Environment Concepts	349
Business Concepts	351
Consumer Concepts	352
Architecture Style, Architecture Pattern, and Design Pattern Questions	353
Architecture Patterns vs. Design Patterns	353
Understand Architecture Styles	355

Remember Contextual Hierarchy of Patterns	355
Why Interviewers Ask Architecture and Design Pattern Questions	356
Prepare for Architecture and Design Pattern Questions	357
Problem-solving and decision-making Questions	358
Embrace the Software Architecture Problem-Solving and Decision-Making Process	358
Identifying Business Problems	358
Attend to the Problem-Solving and Decision-Making Process	359
Prepare for Problem-Solving and Decision-Making Questions	360
Data-Related Questions	360
Focus on Data Aspects Related to Software Architecture	361
More Data-Related Interview Questions	361
Production Environment Questions	362
Characteristics of Software Architecture Environment Hosted in Production	363
Production Environment-Related Questions	364
Software Architecture Framework Questions	365
Focus on Array of Framework Contributions	365
Software Architecture Framework Questions	367

**Index**





# Introduction: Software Architect, Who Are You?

As a software architect you've embarked on a career journey in an uncharted and unpredictable territory with no guarantee of successful technological solutions. You are employed as a software architect to participate in a corporate business, technological, and social experiment whose chief thrust is to manufacture software products deployed to virtual environments. It's also arduous to foretell the business performance quality and stability after deploying and integrating software implementations in computing ecosystems.

By no means is this a bleak portrayal of a software architecture career. On the contrary, the uncertainty of your contribution to enterprise solutions only opens the doors to business development and transformation opportunities, technological modernization, and career improvement and growth. Furthermore, your hard work and dedication can be achieved through the power of creativity, imagination, and persistence. Once you are resolved to pursue a software architecture career, or are already a devoted practitioner, you're destined for a highly successful journey.

The following sections draw a picture of an *ideal software architect* whose capability to solve organizational problems is beyond imagination. This profile represents a well-rounded software architect with close-to-perfect professional talents that organizations would most certainly employ if the need existed. However, do not fret or be discouraged. We strive to possess these outlined qualities to make a difference in people's lives by promoting business culture, strategies, mission, and vision.

Figure I.1 illustrates the ideal software architect's attributes: career-oriented, innate traits-driven, strategy-driven, culture promoter, integration-driven, leadership-oriented, solution-driven, domain-driven, and social-driven.

So, ideal software architect, who are you?



**Figure I.1:** An Ideal Software Architect Profile

---

## You Promote Institutional Culture

---

You're hired as a software architect to inspire change, stir up enthusiasm for innovation, stimulate new ideas, affect organizational strategies, combat business and technological stagnation, and make a big difference in people's lives.

### Become an Agent of Cultural Transformation

You are offered a key position to participate in transforming the *old* into the *new*. The former refers to outdated business concepts, traditional ways of doing business, archaic methods of developing software products, and waning technological solutions. The "new," on the other hand, pertains to modern technologies, creative and practical applications and systems, innovative end-to-end software architecture methodologies and life cycles, and partnerships that promote organizational dialogue to secure the business.

By partaking in such ambitious organizational metamorphosis, you're the *de facto institutional agent of cultural transformation*. You are actively engaged in a social and technological experiment that touches lives and instills change in people's behavior. This multifaceted cultural change manifests in how people communicate, interface with applications and systems, form relationships and partnerships, run their daily lives, and manage their careers.

So, how do software architects promote organizational culture? The arsenal of tools and utilities employed to impact the environment profoundly is vast.

Furthermore, the sky is the limit for technological evolution and innovation. The business and technological solutions you're being asked to provide drive the establishment of organizational *policies*, *best practices*, and *standards*. These rules and procedures you're advocating for promote institutional norms of behavior, foster business alliances, and forge new codes of cultural conduct.

## **Contribute, Do Not Follow**

However, the cultural change that you're promoting does not touch only individuals. You are employed to harness the power of your talents and creativity to form a new generation of ideas and find shared values reinforced by members of your organization inspired by your innovative visions. In reality, you are a benefactor at heart, not a follower. Any organizational solution you offer contributes to the institutional knowledge base and the collective memory of your followers, who are ultimately employed to solve enterprise problems.

## **Further Reading**

Although the topic of promoting organizational culture is discussed throughout the book, refer to these chapters to learn about the specific methods that software architects can leverage to impact institutional culture:

- Chapter 3, "Career Planning for Software Architects: A Winning Strategy" depicts four career-driven perspectives that can impact organizational culture: social-driven, technology-driven, management-driven, and strategy-driven.
- Chapter 4, "Self-Assessment for Software Architects" offers a self-scoring questionnaire that contains queries about promoting organizational culture methods.

## **You're an Astute Strategist**

---

Your strategic mindset is the key to the success of your software architecture career. No matter which software architecture scope of solutions you pursue, application or enterprise level, focus on the big picture. You're a generalist by nature. Never rush into details to develop effective solutions. Having a bird's-eye view is what makes you an all-around type of person.

You're also a gifted tactician who incessantly occupies your mind with long-term and sustainable solutions to remediate business problems. The prospect of business prosperity and technological continuity motivates you to carve out complicated schedules, road maps, and product development timetables.

No matter the magnitude of your work, your strategic outlook is driven by a thorough study of business and technological events that occur on the ground. Then, by connecting the dots, you deliver superior software architecture artifacts. In this context, *connecting the dots* pertains to aggregating and utilizing all possible organizational resources, such as subject-matter experts, data, utilities, and facilities, to derive the best possible software and environment implementations.

## **Adopt an Effective Outside-In Strategy to Deliver Synthesized Software Architecture Solutions**

You're an outside-in software architecture strategist attuned to market and industry trends, quality of organizational services, and, most important, customer imperatives. In addition, you are acquainted with advanced product development life-cycle methodologies and often follow business market developments and innovations, valuable knowledge that drives your methodological approach to meeting client requirements. Satisfying these imperatives begins with an effective business discovery and analysis process that leads to software architecture solutions.

Do not be constrained by existing technological limitations. If current organizational technologies tend to narrow the scope of your vision, you must drive change, modernization, and initiatives aligned with your software architecture vision and mission. Furthermore, you drive business and technological transformation through creativity, curiosity, and modernity synthesis. Finally, never deprive yourself of the freedom of imagination when proposing innovative software architecture implementations.

## **Align Software Architecture Strategies with Organizational Imperatives**

As an astute software architecture strategist, you know that your technological vision and mission must align with business strategies. Remember, you're not operating in a vacuum. Your software architecture solutions, therefore, ought to promote business agendas, foster business growth, and ensure business stability and continuity.

However, aligning software architecture strategies with business vision and mission would not promote satisfactory technological solutions. Business cooperation and coordination are indeed primary and compulsory goals for software architects. Their duties, however, must go beyond business imperatives. There are accessorial software architecture strategy alignment necessities to drive a comprehensive enterprise technological balance.

Thus, software architecture strategies must also be aligned with existing deployment environments, supporting infrastructure, development platforms,

data and message exchange mechanisms, architecture styles, design patterns, and integration patterns. Again, promote transformation initiatives to satisfy software architecture vision and mission if software architecture strategies cannot align with existing technologies and environments.

Figure I.2 illustrates a software architecture strategy alignment priority example chart that outlines alignment opportunities with business, technologies, environment, and infrastructure.



**Figure I.2:** Software Architecture Alignment Priorities

## Further Reading

The topic of software architecture strategy alignment with business strategy, vision, and mission to propel technological initiatives across the organization is discussed largely in Chapter 2, “Types of Software Architects.” It introduces three business needs for software architecture to foster organizational transformation and modernization: strategic collaboration, technological mediation, and technological implementation

## You’re a Gifted Leader

You’re a leader, not necessarily a manager. You possess noteworthy interpersonal traits. You’re a person of integrity who instills trust in your co-workers, managers, corporate executives, and partners. You promote institutional social harmony to foster consensus on software architecture strategies, technologies, best practices, standards, and policies. You’re trusting and trustworthy because you have a positive perspective of humankind.

Your natural leadership traits inspire followers. These devoted fans respect your perspectives and are committed to collaborating with you on software architecture projects and business initiatives. As a gifted technological leader and team player, you prefer to collaborate with others. You encourage diversity of

ideas and solutions by fostering the collective creativity of enthusiastic technologists. You never impose your views on others—in contrast, you’re an advisor, a mentor who offers viable guidance to those who seek professional direction.

**TIP** Remember, you’re a leader. You’re not a manager or administrator who signs timesheets and reprimands staff for wrongdoing.

## **Tolerate Errors and Stay Open to Technological Experiences**

Your innate problem-solving and decision-making skills paint a realistic view of your organization’s business and technological contribution. In other words, nothing is perfect! You understand the difficulties and constraints of any proposed software architecture solution. And you’re aware of the impact your technical recommendations have on your organization. You’re wise to understand that ill-designed applications and systems can cause operational chaos, disrupt business continuity, negatively impact productivity, and harm your company’s bottom line.

With all these potential risks to the enterprise, you’re still a natural optimist and idealist, a risk-taker willing to surrender short-term gains in favor of strategic long-term technological success. These traits define a person who tolerates design errors, software implementation mishaps, and software deployment and integration flaws. In reality, you’re not afraid of failure. In your mind, the design experiment journey you’re willing to embark on can only promote successful technological modernization.

## **Build a Circle of Trustful Followers by Uplifting Their Spirits**

As a software architect, you must inspire others and galvanize positive energy among your co-workers and work teams. You’re here to foster creativity—the failure of imagination is not an option. You are here to usher intelligent followers who trust your software architecture judgment and good taste, and who are not afraid of making design mistakes or expressing silly opinions.

**TIP** Remember, you’re an experimentalist whose leadership traits galvanize enthusiasm for collaborative teamwork to offer superb technological solutions for sustaining and accelerating business success.

## **Further Reading**

Take the self-evaluation questionnaire provided in Chapter 4, “Self-Assessment for Software Architects,” to find out if you possess the proper software architecture

leadership talents that can galvanize enthusiasm for business innovation and technological modernization.

## You're an Instrumental Solution Provider

---

At heart, you are a solution provider. Deep inside you, there is a veiled desire to mitigate risks, resolve social conflicts, and provide guidance to tackle organizational challenges. You always rise to the occasion to remediate business shortfalls. Furthermore, you go the extra mile to seek pragmatic technological solutions.

## Promote Business Growth Through Modern Technological Solutions

You are committed to implementing potent strategic foundations for sustainable and viable business growth through technological modernization. You're a risk-taker and venture to support business transformation by leveraging the best-of-breed technical capabilities. Furthermore, you believe that technology is not just a mechanical mean for implementing temporary solutions or for offering Band-Aid remedies that do not withstand time. Simply put, you're a solution provider with technological, strategic agendas that tolerate occasional failures to achieve novel goals.

## Provide Solutions Within the Boundaries of Your Software Architecture Expertise

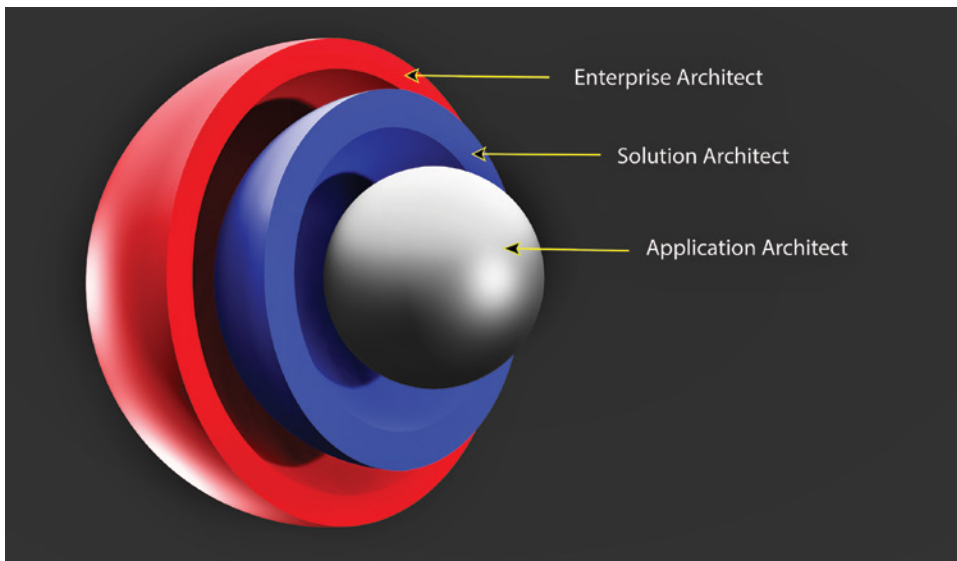
As a software architect, you focus on design solutions—software-oriented remedies, *not hardware*. This is because you understand the boundaries of your occupation. You're aware that the solutions you provide are within the margins of software architecture practices—the field in which you excel. You may collaborate with co-workers specializing in physical infrastructure, hardware servers, and network devices. However, your chief responsibility is to design applications, services, systems, and deployment environments within your software architecture expertise.

Know the boundaries of your responsibilities. Be aware of your software architecture level of contribution. You're wise to understand that the reach of your technical solutions depends on the boundaries of your position. Namely, the job you're holding as a software architect has restricted outreaching responsibilities. This is not because you cannot accomplish tasks beyond your job description. It's simply due to the software architecture duties you're commissioned to pursue.

## Understand the Scope of Your Technological Solutions

So, what would be the scope of your technological solutions?

Nowadays, common organizational practice calls for founding a hierarchical structure of software architecture roles. They are established to address three different levels of solutions. Affiliated with the top layer of a pyramid, enterprise software architects and their technological solutions must meet enterprise-level business imperatives. Then, solutions architects are related to the second layer, just beneath the enterprise architects' level. They are commissioned not only to promote enterprise software architecture strategies, but also to oversee application-level design initiatives. Finally, application architecture roles are the nucleus of any organizational software design initiative. They are positioned at the very bottom of the structural employment hierarchy, assigned to offer solutions for the narrowest range of problems. Figure I.3 illustrates the hierarchical structure of software architecture roles and their solution scope and dependencies within an enterprise.



**Figure I.3:** Software Architecture Roles And Their Organizational Solution Scope

### Further Reading

To learn more about how to scope technological solutions and set boundaries for your professional expertise visit these two chapters:

- Chapter 1, “Software Architect Capability Model,” discusses a method to help scoping technological solutions and setting boundaries to a professional occupation by creating a capability model with five driving sections:



specifications, architecture practices, architecture disciplines, architecture deliverables, and quantification of skill competencies.

- Chapter 2, “Types of Software Architects,” elaborates on two types of software architect roles: leading software architects and domain software architects. Each of these roles are commissioned to focus on specific solution scopes.

## **You’re an Integrator Par Excellence**

---

Integration duties are the bedrock foundation of your occupation. It’s an integral part of your professional daily practice. No matter what level of software architecture contribution to the enterprise you provide, you’re well aware that integration is a compulsory responsibility that you cannot avoid. It’s a software design capability you possess, leverage, and exhibit to satisfy a broad range of business and technological imperatives. Furthermore, integration is a technological, social, and business competence you consistently demonstrate to provide large-scale business remedies. And it’s a software architecture aptitude you employ to aggregate solutions mutually provided by a community of software implementations.

### **Connect the Dots**

You’re dubbed a “software integrator” because every design scheme you devise proves effective partnerships and communications between software implementations. Any design blueprint you provide presents logical views of interaction and collaboration between applications, services, and systems. And it’s starkly apparent that any software architecture environment you design maintains a pragmatic alliance between distributed software assets.

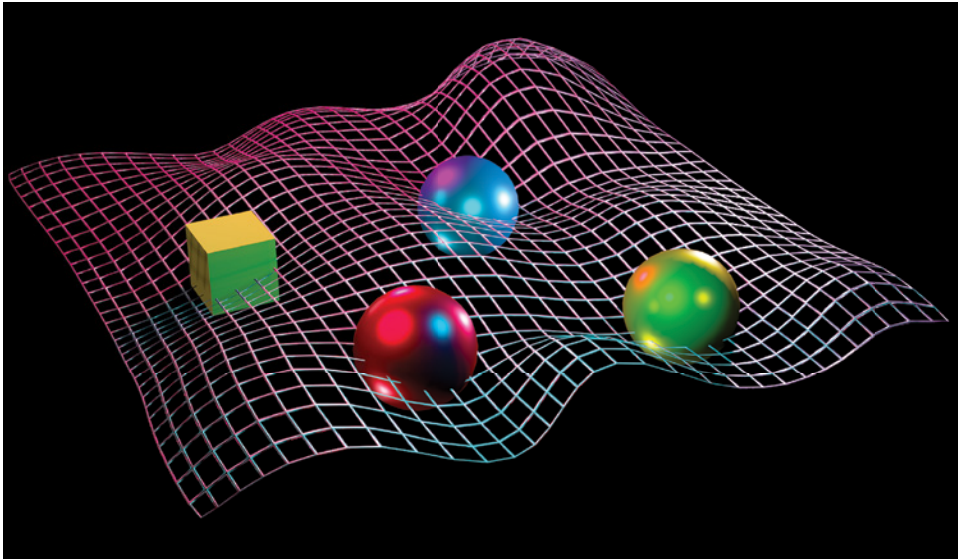
You do not take the term *connecting the dots* lightly in regard to software integration. Namely, you do not sneeze at opportunities to utilize diverse sources of information, combine people’s ideas, and aggregate technological fountains of knowledge to devise powerful software architecture integration solutions. In essence, you’re wise to connect the dots to foster software reuse and optimize the redundancy of business functionality.

### **Integrate Software in a Three-Dimensional Software Architecture Environment**

As a software architect, you’re keenly aware that integration is not only about connecting the dots and not merely about enabling software entities to talk to

each other and exchange information. Indeed, these are vital software architecture tasks that ensure business continuity, ensuing viable organizational solutions.

However, you're also mindful that software implementations do not operate in a vacuum and are deployed to a topological, geometrical, and three-dimensional landscape that offers them adequate architectural conditions to survive in. In Chapter 6, "Software Architecture Environment Construction," and Chapter 7, "Structural Construction of Software Implementations in Multidimensional Environments," this ecosystem is labeled the *software architecture environment*. As illustrated in Figure I.4, this landscape constantly undergoes structural deformation due to the behavior of the hosted software entities.



**Figure I.4:** Structural Deformation Of A Software Architecture Environment

## Mitigate Risks in a Quantum Software Architecture Ecosystem

Your design outcomes consistently demonstrate a compromise between radical software architecture solutions. These negotiated solutions between extreme design approaches contribute vastly to the mitigation of the risks of an unpredictable deployment environment that can negatively impact business execution. Also, you're compelled to adhere to integration best practices, standards, and policies to foster a balanced software architecture environment. Your instrumental integration talents promote a sensible environment balance to minimize the erratic behavior of software. And your surpassing software integration capabilities alleviate the risks of business interruptions.

## Further Reading

The book's Part 3, "Software Architecture Toolbox," represents the Multidimensional Software Architecture Construction (MSAC) methodology. This design approach offers use cases, best practices, and construction laws for software implementations and their affiliated environments:

- Chapter 6, "Software Architecture Environment Construction," is all about integration of software in a multidimensional software architecture environment hosted in production.
- Chapter 7, "Structural Construction of Software Implementations in Multidimensional Environments," represent the 3D structural composition of software entities that are deployed and integrated in a software architecture space.

## You're Domain-Driven

---

You're well prepared to tackle business and technological problems by employing your software architecture talents. There is nothing that can swerve your focus from fulfilling your goals. Furthermore, your uncompromising devotion to offering effective and sustainable software architecture solutions is immeasurable to your organization. Your steadfast resolve to tackle business challenges is attributed to your laser-beam focus on critical problems while avoiding personal agendas and evading trivial issues.

Simply put, the secret of your unwavering commitment to providing best-of-breed software architecture solutions is rooted in your ability to concentrate on what matters. More specifically, your solutions align with corporate business and technological strategies; software architecture vision and mission; leadership directives; and institutional best practices, standards, and policies.

**TIP** In a nutshell, you're a domain-driven software architect familiar with the business environment, the industry, the customers, and the supporting technology.

## Align the Orbit of Your Software Architecture Solutions with Organizational Domains

The alignment of software architecture solutions with business imperatives characteristically yields robust technological solutions. In this context, *business imperatives* refers to different types of requirements. As a pragmatic software architect you can tailor technological solutions to specific business needs. More

explicitly, your solutions to business problems may be affiliated with a specific business sector, business industry, business product, business portfolio, line of business, or business division. These particular business domains drive the technical remedies you propose.

However, business needs do not always drive the domain alignment process. Equally significant is the alignment of *software architecture strategies with business strategies*. The chief reason is that business strategies are the empirical driving forces in the enterprise. Therefore, technological solutions should foster and support long-term business plans, business models, business vision and mission, and business policies.

Moreover, from a domain alignment perspective, you're most certainly aware that the existing technological capabilities of your organization (such as infrastructure, platforms, and networks) must support software architecture solutions. In some cases, the existing technological capacity may not be advanced enough to deliver your proposed design. Therefore, promote technological modernization and transformation initiatives to improve the alignment with your architectural vision and mission.

## Delineate the Scope of Your Software Architecture Solutions

Your devotion to providing software architecture solutions to specific organizational imperatives accelerates time-to-market and ensures business and technological continuity. Pinpointing the sources of business obstacles, devising feasible solutions, and mitigating domain-related issues are prescriptions for software architecture success.

**TIP** By accomplishing this, you're essentially accredited as a domain-driven software architect who is business-driven, strategy-driven, technology-driven, solution-driven, and leadership-driven. Leverage these capabilities to respond to business and technological necessities.

Again, aligning your technological solutions with organizational domains promotes pragmatic software architecture. Therefore, it's highly advisable to create a solution-focused domain diagram similar to the one shown in Figure I.5. Such a depiction will demonstrate the various opportunities for software architecture success. Focus on your organizational domains that require attention. Leverage your leadership talents to focus on particular business and technological problems. Finally, focus only on domain challenges that require solutions.



**Figure 1.5:** Domain-Driven Software Architecture Solution Scope

## Further Reading

As a domain-driven software architect whose duty is to deliver a balanced software architecture, focus on the software architecture construction life cycles, governing laws, and best practices covered in Chapter 6, “Software Architecture Environment Construction” and Chapter 7, “Structural Construction of Software Implementations in Multidimensional Environments.”

Furthermore, to foster a software architecture environment equilibrium, employ your *domain-driven design skills* to meet business, strategy, technology, solution, and leadership imperatives.

## You’re Socially Driven

As a software architect, you’re mindful that social collaboration and partnership with co-workers, industry alliances, customers, and stakeholders yields compelling technological solutions. Technical solutions have never been successful without teamwork and cooperation with subject-matter experts (SMEs). Individuals promoting personal agendas can never deliver substantial software architecture strategies.

In conclusion, software architects should fulfill their duties through the power of social intelligence. Moreover, architects who snub social skills to better accomplish the tasks they were hired for often find that their software architecture solutions ultimately fail to live up to organizational expectations. Simply put,

the respectful and productive interrelationship between technologists and business leaders always demonstrates social capabilities that deliver perceptive technological solutions.

## Leverage the Contribution of Social Intelligence to Your Software Architecture Career

In this context, *social intelligence* pertains to your ability to understand yourself, your needs, and your limitations. However, it's not only about your imperatives or boundaries. This self-acumen is about the capability to know others, the aptitude to understand the environment, and the faculty to develop trustful and sustainable partnerships in the workplace.

Have a look at the chief social intelligence tokens presented in Figure I.6. These represent software architecture social intelligence capabilities leveraged to drive powerful business and technology transformation solutions: *agility*, *adaptability*, and *awareness*.



**Figure I.6:** Software Architecture Social Intelligence Pillars

*Awareness* is a unique social talent that can be leveraged to cope with complex social, business, and technological changes and challenges thrown your way. The term *adaptability* stands for versatility—an attribute that describes a skillful person with multiple talents for tackling organizational problems. Moreover, *agility* is a personal quality of a software architect who knows how to negotiate and compromise on technological solutions, resolve social conflicts, and collaborate with others in good faith.

## Follow a Simple Process to Leverage Your Software Architecture Social Intelligence Skills

Your social intelligence skills can be instrumental in establishing working-related partnerships and alliances. To build a coalition of supporters and collaborators, consider these simple roadmap milestones: *search*, *connect*, *integrate*, and *cooperate*.

At the onset of this exertion, begin searching for candidates who understand your language and objectives and are willing to work together to achieve software architecture goals. While these individuals are typically found in close vicinities, such as in your organization, others can be spotted on social media and at technological conferences.

Once potential social partners are found, devote your time to connecting, raising their interest, and spurring enthusiasm for contributing to the organization and industry. Then share your knowledge. Learn from others. Interface and cooperate on strategies. *And always remember: you're not alone!*

## Further Reading

The topic of software architecture social intelligence skills is covered chiefly in these two chapters:

- Chapter 4, “Self-Assessment for Software Architects,” includes queries to evaluate an individual’s communication, collaboration, and partnership formation skills required to promote software architecture strategies and contribute to technological transformation and innovation.
- Chapter 9, “An Outline for Software Architecture Job Interview Questions,” introduces potential interview behavioral questions, preparing candidates to demonstrate communication, interpersonal relationship, and leadership capabilities.

## You’re Career-Driven

---

It’s critical to carve out a long-term plan, a strategy that reflects your talents and capabilities. Equally important, stay attuned to your individual preferences, such as the types of duties that you’d like to fulfill and contribute to a specific sector and industry. However, focusing merely on your career agenda or promoting individual interests would never contribute to solving organizational problems or boosting your software architecture performance.

Remember that your preliminary duty is to collaborate with co-workers and partners to support business objectives—a vision greater than your aspirations. In software architecture, there is nothing nobler than teaming up with stakeholders to promote organizational culture, influence the outcome of business transformation, and accelerate technological modernization.

## Carve Out a Software Architecture Career Strategy

A software architecture career strategy is a long-term plan that spells out incremental steps to achieving professional milestones and goals. Each milestone

is an important landmark, a checkpoint for evaluating your professional progress and achievements. A career milestone can also mark a turning point, perhaps a change in direction or adjustment to your software architecture employment strategy.

The software architecture career strategy's goal should not be considered your last professional occupation. On the contrary, in a long-term career time span, there may be multiple goals to pursue. Again, each milestone assessment should determine the next career step to conquer.

Moreover, a software architecture career strategy ought to be realistic. And professional development in the field must be gradual and feasible. The journey to achieving career goals should be devoted to knowledge acquisition, self-improvement, and delivering best-of-class software products and the architecture of their hosting environments.

*Knowledge acquisition* refers to the incremental learning and practice of software architecture disciplines during career employment. Specifically, business and technical knowledge are acquired through years of hard work, research, and studies. *Self-improvement* is related to the knowledge acquisition process. But it is affiliated chiefly with self-motivation and the individual appetite for improving software architecture capabilities.

**TIP** Bottom line: a software architecture career strategy encompasses a gradual and self-challenging agenda that should be reevaluated at every milestone.

## Software Architecture Career Strategy Perspectives

Throughout the years, the software architecture field has grown immensely in scope. Professionals choose to focus on different architectural practices and disciplines. Some individuals pursue the leadership and governance route, while others focus merely on the technological aspects of software architecture roles. As illustrated in Figure I.7, this book centers on four chief software architecture career planning perspectives: social, technology, strategy, and leadership.

**Social-Driven Career Perspective** Consider this employment avenue if you seek to promote your professional objectives by forming productive alliances with collaborating partners and executives to provide business and technological solutions.

**Technology-Driven Career Perspective** If you focus merely on your technical skills, pursue this career path by applying software architecture capabilities and experience to provide business and technological solutions.

**Leadership-Driven Career Perspective** Choose this career path if you possess management skills and seek to focus on promoting enterprise culture,



steering technological transformation, and establishing institutional best practices, standards, and policies.

**Strategy-Driven Career Perspective** This role is for you if you look to influence enterprise business and technological evolution, foster digital transformation, develop organization-wide road maps, and align business strategies with software architecture strategies.



**Figure I.7:** Software Architecture Career Strategy Perspectives

## Further Reading

A well-planned career path is a roadmap for a successful software architecture journey. But a career strategy is not the only ingredient for a flourishing occupation. Knowledge acquisition and carving out a winning strategy for software architecture interviews can indeed yield a lifetime of prosperous employment.

- Chapter 3, “Career Planning for Software Architects: A Winning Strategy,” depicts the four career-driven perspectives that can impact organizational culture: social, technology, management, and strategy.
- Chapter 8, “Preparing for a Software Architecture Interview: A Winning Strategy,” introduces a job interview preparation model that includes two different strategies to consider: interview defense and attack plans.
- Chapter 9, “An Outline for Software Architecture Job Interview Questions,” presents potential software architecture questions that can increase the odds of acing an interview. They are grouped into ten different categories, such as technical, behavioral, social, problem-solving and decision-making, software architecture life cycle, and more.

## You Trust Your Innate Talents

You undoubtedly bring a slew of talents instrumental in providing effective software architecture solutions to organizational problems. Moreover, you know

that these personal traits successfully contribute to your employment duties. You may have wondered if these individual aptitudes were with you at birth, or perhaps you've learned them on the job.

Numerous scientific studies submit that the talents you have been carrying since birth are recognized as innate traits—skills not necessarily learned through experience. These are affiliated with primal instincts—natural survival abilities such as endurance, social bonding, adaptability, enthusiasm, and more. We often employ them to endure economic hardships, social challenges, or natural calamities.

However, there is no indication that these survival abilities cannot be learned and honed during a lifetime, career journey, or professional training. And it has become evident that combining innate talents with on-the-job experiences improves the ability of software architects to deliver pragmatic and potent solutions. For example, software architecture capabilities to construct powerful applications and systems typically depend on professional traits such as balanced decision-making, effective problem-solving, and good taste.

## **Employ Innate Traits to Advance Business and Technological Missions**

It is no secret that tempestuous organizational issues often challenge software architects. Some are affiliated with the struggle to advance software architecture roadmaps, visions, missions, and strategies. Fostering and maintaining technological leadership is another difficulty that software architects wrestle with. Facing stiff resistance to business change or technological modernization initiatives is another predicament that must be tackled.

Employ your communication, patience, and self-discipline capacities to alleviate unnecessary conflicts. Respect the diversity of ideas, concepts, and solutions your co-workers, managers, and partners propose. Consider their diverse approaches to solving software development and integration problems. Most importantly, stay tuned with the four innate talents that can enhance your decision-making capabilities (as illustrated in Figure I.8): *creativity*, *imagination*, *software design aesthetic*, and *curiosity*.

## **Avoid Self-Induced Software Architecture Blindness**

Ignoring your innate skills when you need them the most promotes business stagnation, delays technological standardization, and stalls applications and systems modernization. There is nothing riskier to business development than the underutilization of fundamental innate skills, such as creativity and imagination.

Creativity and imagination are all about the enablement of business opportunities. They are the bedrock of every software architecture implementation

that allows the business to flourish and win the competition. On the other hand, curiosity is an essential innate gift that galvanizes research and studies and ultimately encourages perfection. Finally, the design aesthetic is an innate skill that entices consumers to buy goods, acquire services, and look forward to the next line of innovative products.



**Figure 1.8:** Four Leading Innate Talents

**TIP** Do not engage in self-induced software architecture blindness by overlooking your innate traits.

## Further Reading

Refer to Chapter 5, “Employing Innate Talents to Provide Potent Organizational Solutions,” to learn more about the chief innate gifts that software architects should leverage to mitigate enterprise challenges and successfully promote software architecture agendas. This chapter also elaborates on a variety of methods to boost software architecture creativity, imagination, good design taste and aesthetics, and curiosity.



**Part**

**1**

# **Software Architect Capability Model**

---

## **In This Part**

---

**Chapter 1:** Software Architect Capability Model