97 Things Every Software Architect Should Know

Reference Guide - Version 1.0.0

1. Don't Put Your Resume Ahead of the Requirements
2. Simplify Essential Complexity; Diminish Accidental Complexity
3. Chances Are, Your Biggest Problem Isn't Technical
4. Communication Is King; Clarity and Leadership, Its Humble Servants
5. Application Architecture Determines Application Performance
6. Seek the Value in Requested Capabilities
7. Stand Up!
8. Everything Will Ultimately Fail
9. You're Negotiating More Often Than You Think
10. Quantify
11. One Line of Working Code Is Worth 500 of Specification
12. There Is No One-Size-Fits-All Solution
13. It's Never Too Early to Think About Performance
14. Architecting Is About Balancing
15. Commit-and-Run Is a Crime
16. There Can Be More Than One
17. Business Drives
18. Simplicity Before Generality, Use Before Reuse
19. Architects Must Be Hands On
20. Continuously Integrate
21. Avoid Scheduling Failures
22. Architectural Tradeoffs
23. Database As a Fortress
24. Use Uncertainty As a Driver
25. Warning: Problems in Mirror May Be Larger Than They Appear
26. Reuse Is About People and Education, Not Just Architecture
27. There Is No 'I' in Architecture
28. Get the 1,000-Foot View
29. Try Before Choosing
30. Understand the Business Domain
31. Programming Is an Act of Design
32. Give Developers Autonomy
33. Time Changes Everything
34. "Software Architect" Has Only Lowercase a's; Deal with It
35. Scope Is the Enemy of Success
36. Value Stewardship Over Showmanship
37. Software Architecture Has Ethical Consequences
38. Skyscrapers Aren't Scalable
39. Heterogeneity Wins
40. It's All About Performance
41. Engineer in the White Spaces
42. Talk the Talk
43. Context Is King
44. Dwarves, Elves, Wizards, and Kings
45. Learn from Architects of Buildings
46. Fight Repetition
47. Welcome to the Real World
48. Don't Control, but Observe
49. Janus the Architect
50. Architects' Focus Is on the Boundaries and Interfaces
51. Empower Developers
52. Record Your Rationale
53. Challenge Assumptions, Especially Your Own
54. Share Your Knowledge and Experiences
55. Pattern Pathology
56. Don't Stretch the Architecture Metaphors
57. Focus on Application Support and Maintenance
58. Prepare to Pick Two
59. Prefer Principles, Axioms and Analogies to Opinion and Taste
60. Start with a Walking Skeleton
61. It Is All About The Data
62. Make Sure the Simple Stuff Is Simple
63. Before Anything, an Architect Is a Developer
64. The ROI Variable
65. Your System Is Legacy; Design for It
66. If There Is Only One Solution, Get a Second Opinion
67. Understand the Impact of Change
68. You Have to Understand Hardware, Too
69. Shortcuts Now Are Paid Back with Interest Later
70. Perfect Is the Enemy of "Good Enough"
71. Avoid "Good Ideas"
72. Great Content Creates Great Systems
73. The Business Versus the Angry Architect
74. Stretch Key Dimensions to See What Breaks
75. If You Design It, You Should Be Able to Code It
76. A Rose by Any Other Name Will End Up As a Cabbage
77. Stable Problems Get High-Quality Solutions
78. It Takes Diligence
79. Take Responsibility for Your Decisions
80. Don't Be Clever
81. Choose Your Weapons Carefully, Relinquish Them Reluctantly
82. Your Customer Is Not Your Customer
83. It Will Never Look Like That
84. Choose Frameworks That Play Well with Others
85. Make a Strong Business Case
86. Control the Data, Not Just the Code
87. Pay Down Your Technical Debt
88. Don't Be a Problem Solver
89. Build Systems to Be Zuhanden
90. Find and Retain Passionate Problem Solvers
91. Software Doesn't Really Exist
92. Learn a New Language
93. You Can't Future-Proof Solutions
94. The User Acceptance Problem
95. The Importance of Consomm
96. For the End User, the Interface Is the System
97. Great Software Is Not Built, It Is Grown

Compiled from the book *97 Things Every Software Architect Should Know*, 2009, O'Reilly Media