Appendix

Summary of Lessons

Requirements

#1. If you don't get the requirements right, it doesn't matter how well you execute the rest of the project.

#2. The key deliverables from requirements development are a shared vision and understanding.

#3. Nowhere more than in the requirements do the interests of all the project stakeholders intersect.

#4. A usage-centric approach to requirements will meet customer needs better than a feature-centric approach.

#5. Requirements development demands iteration.

#6. Agile requirements aren't different from other requirements.

#7. The cost of recording knowledge is small compared to the cost of acquiring knowledge.

#8. The overarching objective of requirements development is clear and effective communication.

#9. Requirements quality is in the eye of the beholder.

#10. Requirements must be good enough to let construction proceed at an acceptable level of risk.

#11. People don't simply gather requirements.

#12. Requirements elicitation must bring the customer's voice close to the developer's ear.

#13. Two commonly used requirements elicitation practices are telepathy and clairvoyance. They don't work.

#14. A large group of people can't agree to leave a burning room, let alone agree on exactly how to word some requirement.

#15. Avoid decibel prioritization when deciding which features to include.

#16. Without a documented and agreed-to project scope, how do you know whether your scope is creeping?

Design

#17. Design demands iteration.

#18. It's cheaper to iterate at higher levels of abstraction.

#19. Make products easy to use correctly and hard to use incorrectly.

#20. You can't optimize all desirable quality attributes.

#21. An ounce of design is worth a pound of recoding.

#22. Many system problems take place at interfaces.

Project Management

#23. Work plans must account for friction.

#24. Don't give anyone an estimate off the top of your head.

#25. Icebergs are always larger than they first appear.

#26. You're in a stronger negotiating position when you have data to build your case.

#27. Unless you record estimates and compare them to what actually happened, you will forever be guessing, not estimating.

#28. Don't change an estimate based on what the recipient wants to hear.

#29. Stay off the critical path.

#30. A task is either entirely done or it is not done: no partial credit.

#31. The project team needs flexibility around at least one of the five dimensions of scope, schedule, budget, staff, and quality.

#32. If you don't control your project's risks, they will control you.

#33. The customer is not always right.

#34. We do too much pretending in software.

Culture and Teamwork

#35. Knowledge is not zero-sum.

#36. No matter how much pressure others exert, never make a commitment you know you can't fulfill.

#37. Without training and better practices, don't expect higher productivity to happen by magic.

#38. People talk a lot about their rights, but the flip side of every right is a responsibility.

#39. It takes little physical separation to inhibit communication and collaboration.

#40. Informal approaches that work for a small colocated team don't scale up well.

#41. Don't underestimate the challenge of changing an organization's culture as it moves toward new ways of working.

#42. No engineering or management technique will work if you're dealing with unreasonable people.

Quality

#43. When it comes to software quality, you can pay now or pay more later.

#44. High quality naturally leads to higher productivity.

#45. Organizations never have time to build software right, yet they find the resources to fix it later.

#46. Beware the crap gap.

#47. Never let your boss or your customer talk you into doing a bad job.

#48. Strive to have a peer, rather than a customer, find a defect.

#49. Software people love tools, but a fool with a tool is an amplified fool.

#50. Today's "gotta get it out right away" development project is tomorrow's maintenance nightmare.

Process Improvement

#51. Watch out for "Management by Businessweek."

#52. Ask not, "What's in it for me?" Ask, "What's in it for us?"

#53. The best motivation for changing how people work is pain.

#54. When steering an organization toward new ways of working, use gentle pressure, relentlessly applied.

#55. You don't have time to make each mistake that every practitioner before you has already made.

#56. Good judgment and experience sometimes trump a defined process.

#57. Adopt a shrink-to-fit philosophy with document templates.

#58. Unless you take the time to learn and improve, don't expect the next project to go any better than the last one.

#59. The most conspicuous repeatability the software industry has achieved is doing the same ineffective things over and over.

General

#60. You can't change everything at once.