

Systems Analysis and Design in the Age of Options

**Gary Spurrier, University of Alabama
Heikki Topi, Bentley University**

Table of Contents

Part I: Essence and Evolution of Systems Analysis and Design Chapter 1: Systems Analysis and Design in an Age of Options

- 1.1. Introduction to Systems Analysis and Design in an Age of Options
- 1.2. The Core Role of the BA: Software Requirements Analysis
 - 1.2.1. Understanding the Importance of Requirements: Features and Designs
 - 1.2.2. Relating Requirements to Systems and the BA's Core Role
 - 1.2.3. Why SA&D is so Challenging
- 1.3. Beyond SA&D Requirements Analysis: Understanding the Big Picture of System Projects in the Plan-Driven Approach
 - 1.3.1. Key Systems Project Activity: Initial Visioning
 - 1.3.2. Key Systems Project Activity: Business Analysis
 - 1.3.3. Key Systems Project Activity: Project Planning and Implementation Approach Selection
 - 1.3.4. Key Systems Project Activity: Specify Functionality Design
 - 1.3.5. Key Systems Project Activity: Specify Initial Technical Architecture and Design
 - 1.3.6. Key Systems Project Activity: Final Project Approval and Execution Planning
 - 1.3.7. Key Systems Project Activity: Finalize Technical Architecture
 - 1.3.8. Key Systems Project Activity: Implementation via Construction and Configuration
- 1.4. A Major Alternative to Plan-Driven: Agile Approaches
 - 1.4.1. Agile Motivations and Assumptions
- 1.5. The BA in an Age of Options
 - 1.5.1. The Expanded Role of the BA Today
 - 1.5.2. Making Sense of Project Activities in an Age of Options
- 1.6. Security: A Critically Important Topic that is Every Team Member's Job
- 1.7. On-going Case Study and Mini-Cases
 - 1.7.1. Mini-Cases

- 1.7.2. Introduction to On-going I2C2.Business Case Study
- 1.8. Summary
- 1.9. Chapter Review
 - 1.9.1. Key Terms
 - 1.9.2. Review Questions
 - 1.9.3. Problems and Exercises

Part II: Business Analysis

Chapter 2: Identifying Opportunities for Business Transformation with IT

- 2.2. Decades of Interest in Process Innovation
- 2.3. Initial Visioning
- 2.4. Business Analysis
 - 2.4.1. Unified Modeling Language (UML)
 - 2.4.2. UML Activity Diagram
 - 2.4.3. Using Activity Diagram for Process Modeling: Basic Structural Elements
 - 2.4.4. Using Activity Diagram for Process Modeling: Advanced Structures
 - 2.4.5. Current State vs. Future State
 - 2.4.6. Business Process Modeling at I2C2
- 2.5. Summary
- 2.6. Chapter Review
 - 2.6.1. Key Terms
 - 2.6.2. Review Questions
 - 2.6.3. Problems and Exercises
- 2.7. References
- 2.8. Other Resources

Chapter 3: Identifying and Documenting Key Concepts of the Domain of Interest

- 3.1. Setting the Stage
- 3.2. Domain Modeling as the Foundation to Understanding a Problem Domain
 - 3.2.1. Two Approaches to Modeling the Problem Domain
 - 3.2.2. Discovering and structuring the domain model
- 3.3. Representing Elements of Problem Domain with the EER Grammar
 - 3.3.1. Entities
 - 3.3.2. Attributes
 - 3.3.2.1. Degree
 - 3.3.2.2. Cardinality
 - 3.3.2.3. Attributes Associated with Relationships
 - 3.3.2.4. Associative Entity

- 3.3.2.5. Generalization/Specialization
- 3.4. Representing Elements of Problem Domain with the UML Class Diagram Grammar
 - 3.4.1. Classes
 - 3.4.2. Attributes
 - 3.4.3. Associations
- 3.5. Illustrating Integrated Domain Modeling
 - 3.5.1. Connections between Process Models and the Domain Model
- 3.6. On-Going I2C2.Case Study: Conceptual Data Models
- 3.7. Summary
- 3.8. Chapter Review
 - 3.8.1. Key Terms
 - 3.8.2. Review Questions
 - 3.8.3. Problems and Exercises
- 3.9. References
- 3.10. Other Resources

Chapter 4: Articulating Future System Capabilities with User Stories and UI Models

- 4.1. Introduction
- 4.2. Identifying High-Level Information Systems Requirements to Enable Business Transformation
- 4.3. User Stories
 - 4.3.1. Constructing your first user story
 - 4.3.1.1. High-abstraction level user stories leave a lot of options open
 - 4.3.1.2. High-abstraction level user stories do not lock in technology solutions
 - 4.3.2. Moving toward more detailed user stories
 - 4.3.3. Acceptance Criteria
 - 4.3.4. Epics as User Stories that Summarize a Range of Activities
- 4.4. First User Interface Models
 - 4.4.1. Background
 - 4.4.2. Creating UI models associated with user stories
 - 4.4.3. Extending UI models with navigation
 - 4.4.4. Creating UI models based on the domain model
- 4.5. Integrative Perspective on Initial Requirements
- 4.6. Feasibility Analysis
- 4.7. Analysis of Implications and Potential Consequences
- 4.8. Discovery Methods
 - 4.8.1. Interviewing
 - 4.8.2. Surveys
 - 4.8.3. Observation
 - 4.8.4. Review of Existing Documents and Systems
 - 4.8.4.1. Documents Written to Describe the Domain of Interest

- 4.8.4.2. Existing Transaction Documents
 - 4.8.4.3. Existing Internal Systems and Their Documentation
 - 4.8.4.4. Existing External Systems
- 4.8.5. Collaborative Discovery Processes
- 4.9. I2C2.Example
- 4.10. Summary
- 4.11. Chapter Review
 - 4.11.1. Key Terms
 - 4.11.2. Review Questions
 - 4.11.3. Problems and Exercises
- 4.12. References

Part III: Planning the Project and Selecting an Implementation Approach

Chapter 5: Selecting the Optimal Project Approach

- 5.1 Introduction
 - 5.1.1. Plan-driven vs.Agile: What It Means to the BA
 - 5.1.2. Plan-driven vs.Agile: Making the Choice
- 5.2 Selecting the Best Project Approach: Fundamental Choices
 - 5.2.1. Summary of the Plan-driven Approach
 - 5.2.2. Summary of the Agile Approach
 - 5.2.3. Plan-driven vs.Agile: The Contrast
- 5.3. Plan-driven vs.Agile vs. Hybrid: Understanding the Differences and Making the Best Choice
 - 5.3.1. The Essence of Plan-driven, Agile, and Hybrid Approaches
 - 5.3.2. Iterative Construction is Always Better
 - 5.3.3. The Real Choice Today is “Agile vs. Hybrid,” not “Agile vs.Plan-Driven”
- 5.4. Making the Choice between Agile and Hybrid Construction using the Home Grounds Model
- 5.5. Using Radar Charts to Plot Project Characteristics
 - 5.5.1. Understanding the Project Characteristics Radar Chart
 - 5.5.2. Making Sense of Project Characteristics at a Glance
- 5.6. Functional Requirements: The Key Determinant of BRUF
 - 5.6.1. Functional Requirements: The Key Determinant of BRUF
 - 5.6.2. Non-Functional Requirements: The Key Determinant of BDUF
 - 5.6.3. Team Characteristics: Driven by and Fitting with Requirements
- 5.7. Summary
- 5.8. Chapter Review
 - 5.8.1. Key Terms
 - 5.8.2. Review Questions
 - 5.8.3. Problems and Exercises
- 5.9. References

Chapter 6: Project Planning and Creating the Product Backlog

- 6.1. Introduction
- 6.2. Project Management in the Age of Options
 - 6.2.1. Classic Project Management Tools for High Predictability Projects
 - 6.2.2. Systems Projects: Managing High UNpredictability
 - 6.2.2.1. House Construction as a High Predictability Project Managing Replication Risk
 - 6.2.2.2. Systems Development as a Low Predictability Project Managing Design Risk
 - 6.2.3. Predicting the Unpredictable with Gantt-lite Charts
- 6.3. Release Planning
 - 6.3.1. Product Scope in Plan-Driven, Hybrid, and Agile Projects
 - 6.3.2. Creating the Product Backlog
 - 6.3.3. Agile vs. Hybrid Release Planning
 - 6.3.3.1. Agile Sprint-by-Sprint Planning
 - 6.3.3.2. Hybrid Release Planning
 - 6.3.3.3. Creating an Agile Release Plan for the I2C2.Pharmacy Enhancements Project
- 6.4. Summary
- 6.5. Chapter Review
 - 6.5.1. Key Terms
 - 6.5.2. Review Questions
 - 6.5.3. Problems and Exercises
- 6.6. References

Chapter 7: Identifying Development Options: Selecting Implementation Approach and Determining Sources of Resources

- 7.1. Introduction
- 7.2. Approaches to Acquiring Software Capabilities
 - 7.2.1. Introduction
 - 7.2.2. Development of Custom, Proprietary Software Solutions
 - 7.2.2.1. Justification for use
 - 7.2.2.2. Costs and risks
 - 7.2.3. Licensing Commercially Available Software Solutions
 - 7.2.3.1. Large-scale enterprise systems
 - 7.2.3.2. Specialized systems for industry verticals
 - 7.2.3.3. Traditional packaged software
 - 7.2.3.4. Delivery modes: On premises vs. Cloud-based systems

- 7.2.3.5. Implementation and Maintenance Processes
 - 7.2.3.6. Justification for use
 - 7.2.3.7. Costs
 - 7.2.4. Open Source Software
 - 7.2.5. Practical Reality: Integrated Systems
 - 7.2.5.1. Achieving Distinctive System Capabilities through Integration and Configuration
- 7.3. Sourcing of Development Resources
 - 7.3.1. Contractual Arrangement Types
 - 7.3.1.1. Salaried employees
 - 7.3.1.2. Independent Contractors
 - 7.3.1.3. Developers contracted through professional services firm
 - 7.3.1.4. Outsourcing
 - 7.3.1.5. Consulting
 - 7.3.2. Geographic Placement of Resources
 - 7.3.2.1. On-site
 - 7.3.2.2. Offshoring, nearshoring and onshore outsourcing
 - 7.3.3. Current reality: geographically distributed teams with multiple organizational arrangements
- 7.4. Managing Relationships with System Vendors
 - 7.4.1. Selecting a vendor
 - 7.4.2. Types of contracts with a vendor
 - 7.4.2.1. Application software development and maintenance
 - 7.4.2.2. Enterprise systems and other third-party software
 - 7.4.2.3. Cloud infrastructure for systems development and deployment
 - 7.4.3. Building and managing a long-term relationship with vendors
- 7.5. Impact of Software and Resource Sourcing on Development Methodology
- 7.6. Summary
- 7.7. Chapter Review
 - 7.7.1. Key Terms
 - 7.7.2. Review Questions
 - 7.7.3. Problems and Exercises
- 7.8. References

Part IV: Functional Design and System Architecture

Chapter 8: Creating Use Case Narratives: When, How, and Why?

- 8.1. Introduction
 - 8.1.1. A Quick Review of Requirements: Features vs. Functional Designs
 - 8.1.2. Specifications Supporting the Core Goal of SA&D: Uncertainty Reduction
 - 8.1.3. The Need for Use Case Narratives
 - 8.1.4. An Example of the Need for Additional Requirements in the I2C2 Pharmacy Project

- 8.1.5. A Note about Non-Functional Requirements and Technical Designs
- 8.2. In-Depth: Use Case Narratives
 - 8.2.1. Use Case Narratives vs. User Stories
 - 8.2.2. Use Case Narratives vs. Use Case Diagrams
 - 8.2.3. The Contents and Format of a Use Case Narrative
 - 8.2.4. Use Case Level of Detail
 - 8.2.5. Use Case Goal Level
 - 8.2.6. A Use Case by Any Other Name: The More General Notion of the Software Requirement Specification
 - 8.2.7. How do Use Cases Relate to Agile vs. Plan-Driven Techniques?
 - 8.2.8. Determining Sections to Use in Fully Dressed Use Cases
 - 8.2.8.1. The Minimum Contents of a “Fully Dressed” Use Case
 - 8.2.9. Maximizing Requirements Value by Intelligently Combining Different Requirements Artifacts: The Example of Use Case Narratives and UI/UX Prototypes
 - 8.2.9.1. Project Factors Determining Which Use Cases Should Be Created and the Level of Detail They Should Contain
 - 8.2.10. Functional Requirements Characteristics
 - 8.2.11. Non-Functional Requirements Characteristics
 - 8.2.12. Team Characteristics
 - 8.2.13. Use Case Formats for Supporting Different Levels of Being “Dressed”
 - 8.2.14. When to Create Features and Functional Designs
- 8.3. Identifying Use Cases from User Stories
 - 8.3.1. One-to-One Correspondence
 - 8.3.2. One-to-None Correspondence
 - 8.3.3. Single User Story Generates Multiple Use Cases
 - 8.3.4. Multiple Features/Stories Generate a Single Use Case
 - 8.3.5. Techniques for Splitting Up Individual Use Cases to Fit into Implementation Sprints
 - 8.3.5.1. Use Case Slices
 - 8.3.5.2. Decomposition to Engineering Tasks
- 8.4. I2C2 case study: Examples of Use Case Narratives
 - 8.4.1. Pharmacy Enhancement Project
 - 8.4.2. Medical/War Evacuation Business
 - 8.4.2.1. Elaborating on Features/Stories in an Agile Development Approach
 - 8.4.2.2. Example “Casual” Use Case with Acceptance Criteria
- 8.5. Summary
- 8.6. Chapter Review
 - 8.6.1. Key Terms
 - 8.6.2. Review Questions
 - 8.6.3. Problems and Exercises
- 8.7. References

Chapter 9: Architectural Context for Systems Development

- 9.1. Introduction: Architecture as context for systems development
- 9.2. Approaches to Specifying Enterprise Architecture
 - 9.2.1. Zachman Framework
 - 9.2.2. The Open Group Architectural Framework (TOGAF)
 - 9.2.3. Alternative Perspectives on Enterprise Architecture
- 9.3. Non-functional Requirements
- 9.4. Technology Architecture
 - 9.4.1. Introduction
 - 9.4.2. Computing Devices and System Software
 - 9.4.2.1. Client devices
 - 9.4.2.2. Servers
 - 9.4.2.3. Data centers
 - 9.4.2.4. System software
 - 9.4.2.5. Scalability
 - 9.4.3. Networks and Protocols
 - 9.4.3.1. Introduction
 - 9.4.3.2. Layered models for networking
 - 9.4.3.3. Network types
 - 9.4.4. Architectural Issues in Software Design
 - 9.4.4.1. Layered models for software design
 - 9.4.4.2. Software architecture
 - 9.4.5. Cloud-based Services
- 9.5. Data Architecture
- 9.6. Application Architecture
 - 9.6.1. Introduction
 - 9.6.2. Modeling Application Architecture
 - 9.6.3. Managing Application Portfolio
- 9.7. Summary
- 9.8. Chapter Review
 - 9.8.1. Key Terms
 - 9.8.2. Review Questions
 - 9.8.3. Problems and Exercises
- 9.9. References

Part V: Building a Business Case for the Project

Chapter 10: Estimating Software Projects Effectively

- 10.1 The Importance of Creating Meaningful and Accurate Systems Costs and Benefits Estimates

- 10.2 Go/No Go Decisions: Justifying a Systems Project
 - 10.2.1. Costs vs. Benefits in Business Terms
 - 10.2.2. The Challenges of Estimating Software Costs
 - 10.2.3. The Challenges of Estimating Business Benefits
 - 10.2.4. Tuning Estimation to the Needs of Different Organizations and Projects
- 10.3. Key Concepts in Estimating Systems Project Costs
 - 10.3.1. Key Systems Project Cost Categories: The Primacy of Software Labor Construction Costs
 - 10.3.2. The Planning Fallacy
 - 10.3.3. Impacts of Project Size
 - 10.3.4. Tension between Agility and Planning in Software Cost Estimation
 - 10.3.5. A Key to Success or Failure: Accurately Estimating Development Costs
 - 10.3.6. Hallmarks of a Good Software Estimate: Accurate rather than Precise
 - 10.3.7. Quantifying Minimum Estimation Error: The Cone of Uncertainty
- 10.4. Software Project Cost Estimating Approaches
 - 10.4.1. T-Shirt Sizing
 - 10.4.2. Planning Poker
 - 10.4.3. Expert Judgment
 - 10.4.4. Function Point Analysis
 - 10.4.5. Dealing with the Issue of Analyzing Complexity
 - 10.4.6. Beyond Function Points: Adjusting for Impacts of People, Product, and Process Characteristics
 - 10.4.7. Concluding Thoughts regarding Function Point Analysis and COCOMO II
- 10.5. Software Cost Estimating Summary: Concluding Thoughts and Advice
- 10.6. Summary
- 10.7. Chapter Review
 - 10.7.1. Key Terms
 - 10.7.2. Review Questions
 - 10.7.3. Problems and Exercises
- 10.8. References

Chapter 11: Estimating Business Benefits and Analyzing the Systems Investment

- 11.1. Cost/Benefit Analysis to Justify an Information Technology Project
 - 11.1.1. Benefits Estimation Challenge 1: Confusing Business Benefits with System Capabilities
 - 11.1.1.1. Using the Project Vision Statement to Get Clear about Capabilities vs Benefits
 - 11.1.1.2. Monetizing System Capabilities

- 11.1.2. Benefits Estimation Challenge 2: Many Different Sources of Benefits
- 11.1.3. Benefits Estimation Challenge 3: Incremental Benefits and Long-range Planning Horizons
- 11.1.4. Benefits Estimation Challenge 4: Different Approaches to Comparing Costs and Benefits
- 11.2. Case Study: Identifying and Estimating Business Benefits
 - 11.2.1. Business Benefits Source: Increasing Efficiencies
 - 11.2.1.1. A Specific Efficiencies Example: The Current State and Problems
 - 11.2.1.2. A Specific Efficiencies Example: The Future State Opportunity
- 11.3. Cost/Benefit Analysis Approaches for Software Projects: The Ideal vs. the Real
 - 11.3.1. Financial Analysis Approaches: The NPV “Ideal” vs. the ROI “Real”
 - 11.3.2. An Ideal Investment Decision Approach: Net Present Value Analysis
 - 11.3.2.1. Quick Overview: The Time Value of Money
 - 11.3.2.2. The Time Value of Money: Why It Matters to Systems Project Investment Decisions
 - 11.3.2.3. NPV: Investment Decision Rule and Practical Pitfalls for Software Projects
 - 11.3.3. A Pragmatic Investment Decision Approach: Return on Investment Analysis
 - 11.3.3.1. ROI Using a Cost/Benefit Analysis Grid
 - 11.3.3.2. ROI Decision Rules Recognizing the Estimating Risk
 - 11.3.4. Bottom Line of the Bottom Line: NPV vs. ROI Analysis
- 11.4. Summary
- 11.5. Chapter Review
 - 11.5.1 Key Terms
 - 11.5.2 Review Questions
 - 11.5.3. Problems and Exercises
- 11.6. References

Chapter 12: Planning to Succeed Using Project Documents

- 12.1. Introduction
 - 12.1.1. Intelligent Project Planning
 - 12.1.2. The Value of Writing Out the Project Plan
 - 12.1.3. Key Project Planning Questions
- 12.2. Key Project Planning Documents
 - 12.2.1. Key Project Planning Documents: Flow and Interrelationship
 - 12.2.2. Thumbnail Sketch of the Four Key Project Planning Documents
- 12.3. Project Planning Sections in Depth
 - 12.3.1. Project Vision
 - 12.3.2 Relationship between Statement of Work and Project Charter
 - 12.3.3. Scope Definition
 - 12.3.4. Definition of Done

- 12.3.5. Sprint Plan, Milestones, and Key Dates
- 12.3.6. Team Roles and Organization
- 12.3.7. Risks and Issues
 - 12.3.7.1. Risks
 - 12.3.7.2. Issues
- 12.3.8. Assumptions and Constraints
- 12.3.9. Authority to Proceed: Bringing It All Together in a Business Case
- 12.4. Summary
- 12.5. Chapter Review
 - 12.5.1 Key Terms
 - 12.5.2 Review Questions
 - 12.5.3 Problems and Exercises
- 12.6 References

Part VI: Technical Design and Construction of the System

Chapter 13: Designing the User Experience and User Interfaces

- 13.1. Introduction
- 13.2. Role of UX/UI activities in the Systems Development Process
 - 13.2.1. Background
 - 13.2.2. Inclusion of UX/UI Throughout the SA&D Process
 - 13.2.3. User-Centered Design
 - 13.2.4. User Stories and Use Case Narratives as Foundation of UX Design
- 13.3. General UX/UI Design Principles/Heuristics
 - 13.3.1. Nielsen's 10 heuristics
 - 13.3.2. Shneiderman's Eight Golden Rules of Interface Design
 - 13.3.3. Human-Computer Collaboration
- 13.4. Interaction Design: Designing Interfaces for Operational Use
 - 13.4.1. Identifying and structuring the features of the application
 - 13.4.2. Core terminology
 - 13.4.3. Navigation controls
 - 13.4.4. Input controls
 - 13.4.5. Information components
 - 13.4.6. Providing support in error situations
 - 13.4.7. User's status
 - 13.4.8. Searches related to transactions
 - 13.4.9. Organizing user interface elements on a screen
 - 13.4.10. Example
- 13.5. Interaction Design: Designing Interfaces for Retrieving and Analyzing Data
- 13.6. Interaction Design: Dashboards
- 13.7. Achieving the Same Goal with Multiple Interaction Channels

- 13.7.1. Specific issues to address in design for mobile devices
- 13.8. Designing User Experiences for Global Organizations
- 13.9. Usability Evaluation
 - 13.9.1. Usability testing
 - 13.9.2. Usability inspection
- 13.10. Summary
- 13.11. Chapter Review
 - 13.11.1. Key Terms
 - 13.11.2. Review Questions
 - 13.11.3. Problems and Exercises
- 13.12. References
- 13.13. Additional Resources

Chapter 14: Role of the Business Analyst in Technical Design

- 14.1. Introduction
- 14.2. Context Specificity of Technical Design
- 14.3. Layered Architecture
- 14.4. Data(base) Layer Design
 - 14.4.1. Relational Model
 - 14.4.2. Converting a Domain Model into a Relational Database Design
 - 14.4.3. Normalization
 - 14.4.3.1. From a logical model to physical database design and database implementation
 - 14.4.4. Using Domain Model as a Foundation for Other Data Management Solutions
 - 14.4.5. Integrating Relational Data Management with Object-Oriented Development
- 14.5. Designing Application Logic Using UML Interaction Diagrams
 - 14.5.1. UML Sequence and Communication Diagrams
 - 14.5.2. Specifying User-System Communication Further with a System Sequence Diagram
 - 14.5.3. Designing Internal System Structure Using the Sequence Diagram
 - 14.5.3.1. Layered application architecture
 - 14.5.4. Identifying Class Behaviors based on the Sequence Diagram
- 14.6. Design Patterns and Frameworks
 - 14.6.1. Introduction
 - 14.6.2. MVC, MVP, and MVVM Patterns
 - 14.6.3. Full-stack Development Frameworks: Two Examples
 - 14.6.4. GRASP Patterns
 - 14.6.5. Gang of Four Patterns
- 14.7. Summary
- 14.8. Chapter Review
 - 14.8.1. Key Terms

- 14.8.2. Review Questions
- 14.8.3. Problems and Exercises
- 14.9. References
- 14.10. Additional Resources

Chapter 15: Leading Iterative Software Construction

- 15.1. Introduction
- 15.2. Leading a Construction Sprint
 - 15.2.1. Construction Sprints in Agile and Hybrid Projects
 - 15.2.2. Expressing the Sprint Plan using a Burndown Chart
 - 15.2.3. The Day-by-Day Process of Running a Sprint
 - 15.2.3.1. Updating the Burndown Chart and Using It in Stand-up Meetings
 - 15.2.3.2. Interpreting Burndown Charts
 - 15.3.2.3. Diagnosing Problems with a Burndown Chart
 - 15.2.4. Issues Impacting Sprint Capacity
 - 15.2.4.1. Real-life Interruptions and Absences
 - 15.2.4.2. Testing and Revisions
 - 15.2.4.3. Technical Debt
 - 15.2.5. Using Task Boards and Kanban Boards to Manage the Sprint
- 15.3. Evaluating Construction Sprints and Reporting Status
 - 15.3.1. Assessing Velocity
 - 15.3.1.1. Velocity as a Measure of Stories Completed
 - 15.3.1.2. Caveats to Sprint Velocity Measurement: Zooming in on Engineering Tasks
 - 15.3.1.3. Caveats to Sprint Velocity Measurement: Are We Paying on Technical Debt?
 - 15.3.1.4. Caveats to Sprint Velocity Measurement: Are We Creating More Technical Debt?
 - 15.3.2. Estimates and Effort Remaining: Ideal Days (or Hours) vs. Story Points
 - 15.3.3. Sprint Reviews for Software Feedback and Requirements Course Corrections
 - 15.3.4. Sprint Retrospectives
 - 15.3.5. Project Reporting
 - 15.3.5.1. Weekly Status Report
 - 15.3.5.2. Monthly Steering Committee Report
- 15.4. Summary
- 15.5. Chapter Review
 - 15.5.1. Key Terms
 - 15.5.2. Review Questions
 - 15.5.3. Problems and Exercises
- 15.6 References

Chapter 16: Making the Changes Real: Systems Change Management and Deployment

16.1 Introduction

16.2 Change Management

16.2.1. Key Change Management Areas

16.2.2. Interaction of Project Activities and Sorting Out Who does What

16.2.3. Change Management: Data Preparation

16.2.4. User Acceptance Testing

16.2.4.1. UAT: Creating Environments, Populating Data, and Anonymizing Sensitive Data

16.2.4.2. UAT: Test Plans, Test Cases, and Test Scripts

16.2.4.3. Coordinating Testing between IT and the Business

16.2.4.4. Implementing Testing Automation

16.2.5. Change Management: Policies and Procedures

16.2.6. Change Management: Training Manuals

16.3. Understanding and Choosing Options for Deployment

16.3.1. Traditional Release Approaches

16.3.1.1. The Character of Traditional Software Deployment Approaches

16.3.1.2. Direct Deployment

16.3.1.3. Parallel Deployment

16.3.1.4. Pilot Deployment

16.3.1.5. Phased Deployment

16.3.2. DevOps: Pushing Agile Principles to Deployment

16.3.2.1. DevOps Concepts and Motivations

16.3.2.2. The Evolution of IT Projects: Making DevOps Possible

16.3.2.3. Outlining the Path to DevOps

16.3.2.4. The Starting Point: Agile Development with Traditional Deployment

16.3.2.5. The First Step toward DevOps: Continuous Integration

16.3.2.6. Continuous Delivery: Routinely Deploying Tested Code into “Production-Like” Environments

16.3.2.7. Continuous Deployment: Releasing Features at the Speed of Agile

16.4. Change Management and Deployment in an Age of Options: Implications for the BA

16.4.1. DevOps: Cultural as Well as Technical Changes

16.4.2. The Challenge of Planning in a World without Major Releases

16.4.3. A New Requirements Dimension for the BA

16.4.4. The Paradox of DevOps: Agile Deployment takes a Lot of Planning!

16.4.5. DevOps as an Option in the Age of Options

- 16.4.5.1. Low Change Management Needs: Environments Compatible with DevOps
- 16.4.5.2. High Change Management Needs: Environments where DevOps is More Challenging

16.5. Summary