Leadership Guidelines for an Agile Team

Part 2: Development

Paul Alfred Elling

Contents

Leadership Guidelines for an Agile Team........1
Part 2: Development........1
Introduction........5
Part 2: Development........7
Chapter 1: Software Engineering........8
Software Development Lifecycle (SDLC)........8
Software Requirements........10
Software Design and Architecture........16
Programming........17
Testing and Quality Assurance........17
Deployment........18
Maintenance........18
Chapter 2: Programming........19
Programming........19
Debugging........19
Refactoring........19
Source Code Control........20
Continuous Integration........20
Documentation........20
Unit Tests........21
Code Reviews........21
Chapter 3: Object-Oriented Programming........22
Chapter 4: Design Patterns........23
Chapter 5: Database Design........24
Chapter 6: User Experience........25
Chapter 7: User Interface Design........26
Chapter 8: Agile Leadership........27
Inclusive Leadership........30
Bill Clinton........30
Chapter 9: Agile Project Management........31
Project Management........31
Waterfall Project Management........31

Iterative Development 32
Agile Project Management / Scrum 32
User Stories 33
Waterfall Project Management with Agile Sprints 33
Appendix A: Attributes of Leaders 34
Sources 35

This book is dedicated to my family: my wife, Janelle and my son, Isaac.

Paul Alfred Elling

Introduction

Not enough education is provided to people, including students, about the subject of leadership. It can be considered art and science and has been practiced by people in various situations and organizations throughout the world and through the course of history. Leadership is available to anyone from any walk of life. It is not limited to gender, race, religion, physical attributes, or economic means. Rather, leadership is about galvanizing oneself and others to accomplish something greater than can be achieved by individual efforts. It is the goal of this book to provide guidelines of leadership that can be learned and practiced accomplishing the goals of an organization.

"Leadership can be defined as the art, science, or gift by which a person is enabled and privileged to direct the thoughts, plans, and actions of others in such a manner as to obtain and command their obedience, their confidence, their respect, and their loyal cooperation. Leadership is the profession of the officer in which proficiency can only be obtained through a constant study of leadership principles and practice in applying them in day-to-day relationships with juniors, seniors, and peers. The most successful leaders have a keen understanding of human behavior and know how to get the most from their followers." (p. 1 – 3, Navy)

"Leadership is not about a position or title, as many young people think. It is about choices you make throughout your life—with the goal of making the situations and places you find better because you were there. Great leadership is not about making the leader look good but about how individuals use leadership in service to others to make the people and groups around them better. Leadership is the art of mobilizing others to want to struggle for shared aspirations." (p. XV – 1, Kouzes and Posner)

I obtained an MBA in Strategic Leadership from Amberton University because I wanted to learn what it takes to lead. I have learned about leadership from my career experiences as well. What I have realized over time is that learning about leadership happens throughout life as a never-ending process. Members at all levels of an organization can lead. The role of the leader is to define reality and give hope. Followers can be leaders when they embrace their leaders' goals and when they lead by example. Leadership begins when leaders seize opportunities, and it is all about behavior. "A leader is someone who takes responsibility. Leadership is born when we become active rather than passive." (p. 5, Sacks) Leaders must have the courage to be themselves and not someone else. "The core task of leadership is to create the conditions for other people to thrive." (p. 9, Frei and Morriss) "Leadership is the influencing process of leaders and followers to achieve organizational objectives through change." (p. 6, Lussier and Achua)

These leadership books will focus on the intangibles, the fundamentals, and the principles necessary for an Agile team to succeed. The first book will focus on the process, or the building blocks, that are needed. The second book will focus on development, or the fundamental skills, that are needed. The third and fourth books will focus on the principles, or the bedrock, that are needed.

Part 2: Development

Chapter 1: Software Engineering

"The most fundamental problem in computer science is problem decomposition: how to take a complex problem and divide it up into pieces that can be solved independently." (p. vii, Ousterhout)

Software engineering should utilize an engineering process just as much as any other engineering discipline uses. It is more than simply writing code and deploying that code to a production environment. Quality assurance is essential to ensuring that the code does not cause any harm or breakdowns of the software applications. Software engineers need to be detail-oriented just as much as engineers in other disciplines need to be.

Software Development Lifecycle (SDLC)

Amazon Web Services (AWS) defines the Software Development Lifecycle with the following:

> "The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure."

The SDLC is composed of the following phases:

- Planning
  - Planning means project planning and allocation of resources for the development of the software.
- Analysis and Design
  - Requirements are determined and the design of the software is conducted.
- Implementation
  - Implementation means programming and unit testing.
- Testing
  - Testing is for quality assurance, or system-wide testing, user acceptance testing, and regression testing.
- Deployment
  - Deployment means releasing changes to an environment after testing is completed.
- Maintenance
  - Maintenance involves resolving issues that are found after release of the software.

One of the goals of software development is reuse. Parts of projects that can be reused include the following:

- Software architecture
- Design patterns
- Reusable components
- Requirements
- User interface elements
- Project plans

- Test cases
- Technical review procedures
- Source code
- Configuration management controls
- Team structures

Software Requirements

Requirements can be defined by the following. Ian Sommerville and Pete Sawyer stated, "Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the 10development process of the system." (p. 6, Wiegers and Beatty) Types of requirements include the following:

- Business requirements
- Business rules
- Constraints
- External interface requirements
- Features
- Functional requirements
- Nonfunctional requirements
- Quality attributes
- System requirements
- User requirements

Determining the requirements of software is key to delivering a successful software solution. The drawbacks of not determining the correct requirements up front include bugs being created in development, which might even surface in production. If the end customer can be involved in requirements gathering, software engineers are more likely to develop the features of the system that are expected. Business analysts typically draft requirements documentation and are primarily responsible for gathering requirements. The business analyst's tasks include the following:

- Define business requirements
- Plan the requirements approach
- Identify project stakeholders and user classes
- Elicit requirements
- Analyze requirements
- Document and communicate requirements
- Lead requirements validation
- Facilitate requirements prioritization
- Manage requirements

Successful requirements gathering involves getting input from end users, software engineers, and quality assurance professionals. The levels of requirements are:

- Business
- User
- Functional

Agile projects maintain requirements in user stories in a product backlog. Each iteration of a project is planned by selecting user stories for development during the iteration. The user stories are selected based on prioritization. Changes that are requested during an iteration are added to the product backlog for later development. Iterative development, such as Agile, allows changes to be incorporated within a project more easily than the traditional Waterfall model. In Agile projects, a product owner represents the needs of the users and assumes responsibility for prioritizing the product backlog. The product owner “straddles the project champion and business analyst functions, representing the customer, defining product features, prioritizing them, and so forth.” (p. 115, Wiegers and Beatty)

“Business requirements refers to a set of information that, in the aggregate, describes a need that leads to one or more projects to deliver a solution and the desired ultimate business outcomes. Business opportunities, business objectives, success metrics, and a vision statement make up the business requirements.” (p. 78, Wiegers and Beatty) It’s important to prioritize requirements to deliver the most important software features first in projects. Product backlogs help to prioritize requirements. Requirements documentation should provide the following:

- Business opportunity
- Business objectives
- Success metrics
- Vision statement
- Business risks
- Business assumptions and dependencies
- Scope and limitations
- Major features
- Project priorities
- Software quality metrics

Requirements documentation should spell out the different roles that users play in the use of the software. Software requirements are gathered through the following methods:

- Interviews
- Workshops
- Focus groups
- Observations
- Questionnaires
- System interface analysis
- User interface analysis
- Document analysis

Use cases and user stories are very useful for documenting user interaction with the system. "A use case is a written description of how a user will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled." (Usability.gov) A simple use case would be something like the following:

Use Case Title	Add items to a shopping cart
Actor	Customer
Basic Flow	1. User browses merchandise 2. User selects product 3. User views product 4. User adds product to shopping cart

Use cases can also include diagrams, such as the following:

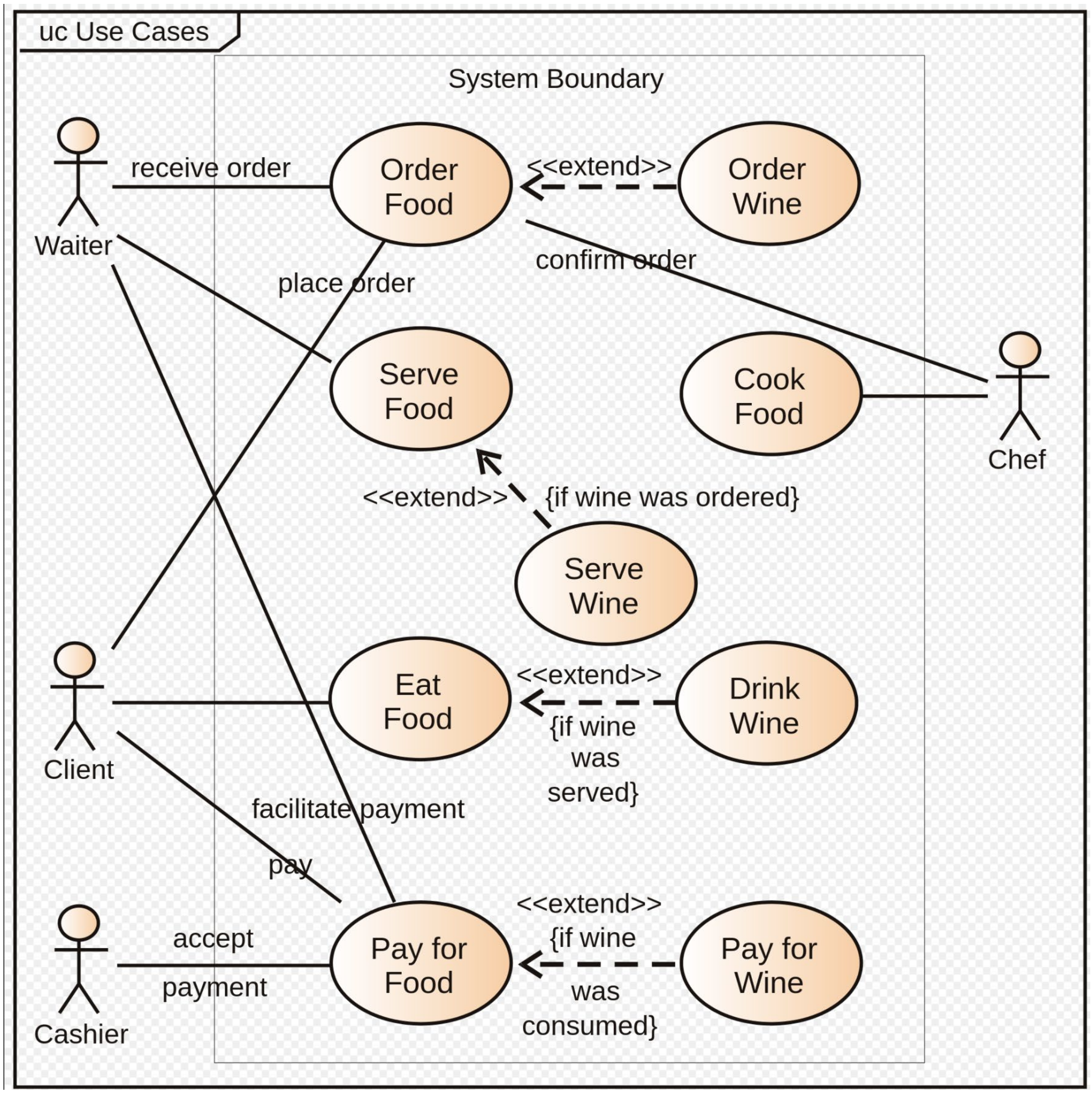

Figure 1 Use Case Diagram (Source: Wikipedia)

Software requirements can also spell out requirements of data flow for software. Data flow diagrams are useful for understanding how information is processed in software.

Figure 4: A sample data flow diagram – bank operations

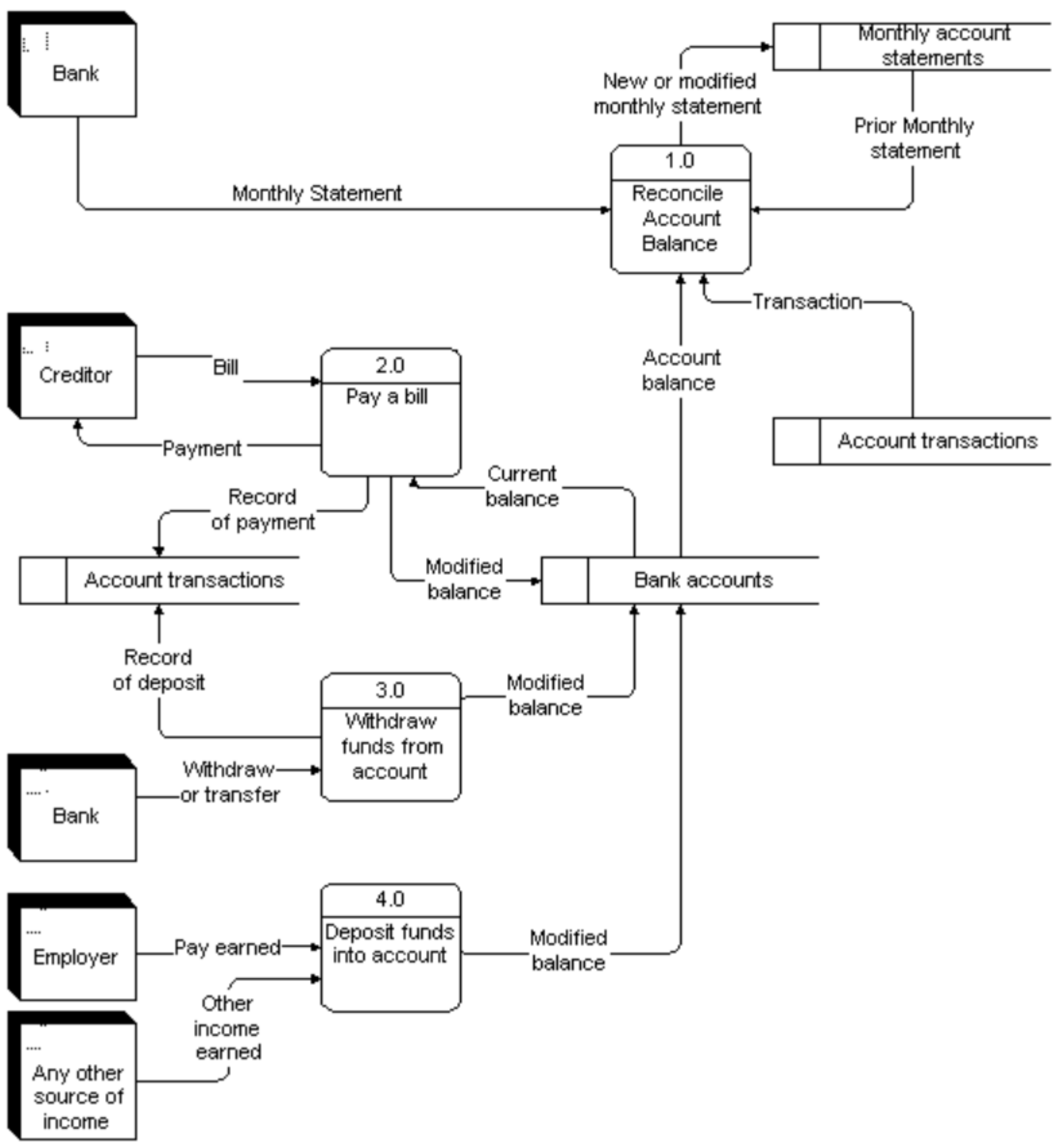

Figure 2 Data Flow Diagram (Source: Babson College)

Another important diagram for understanding databases and the modelling that is required is the Entity Relationship Diagram (ERD). ERDs are essential for understanding the relationships between tables in a database:

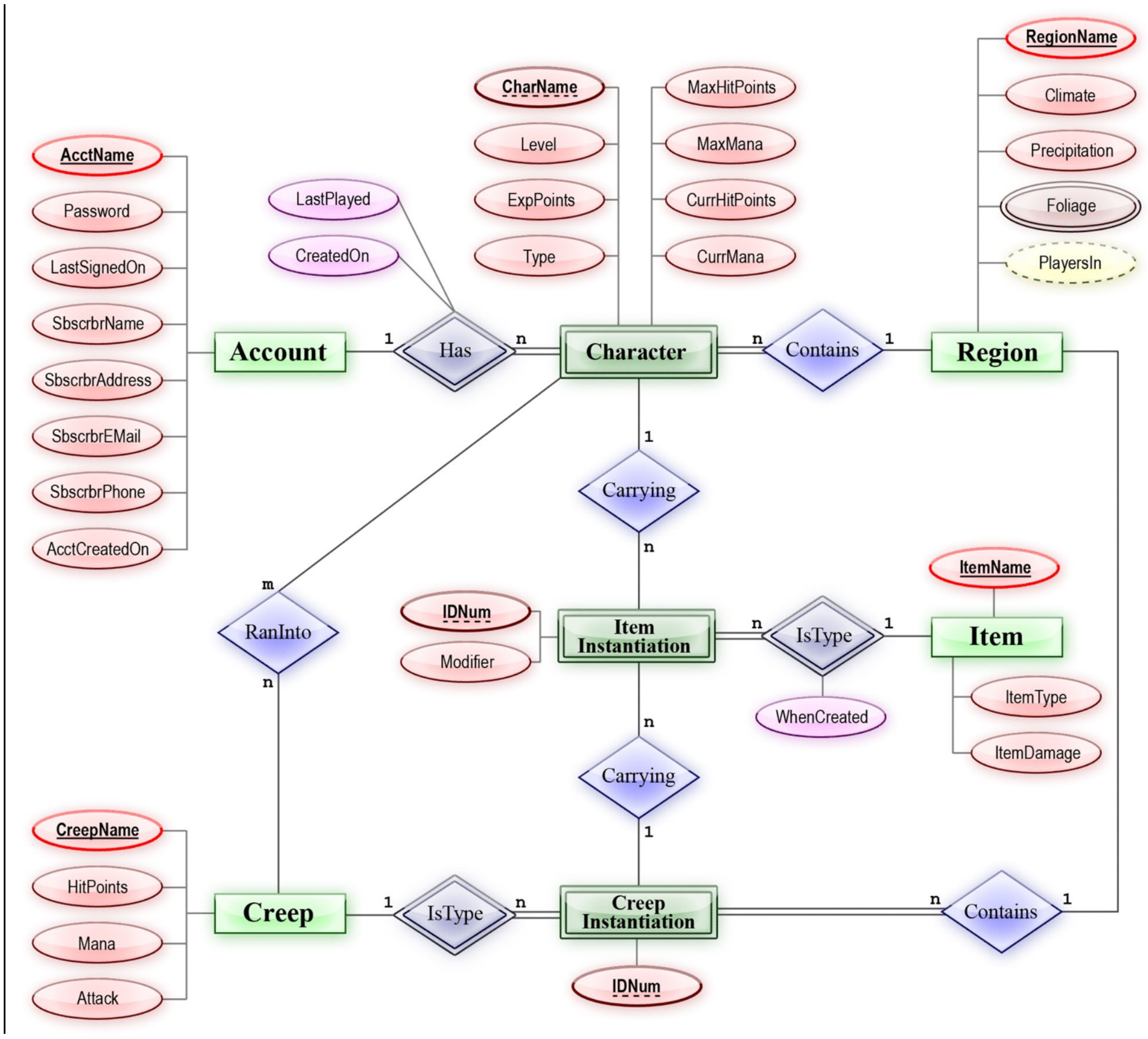

Figure 3 Entity Relationship Diagram (Source: Wikipedia)

Software Design and Architecture

Well-designed software is the goal of software engineering. “The best design for a system is the simplest design that supports all the required features of that system while simultaneously affording the greatest flexibility for change. A simple design in which high-level policies are ignorant of low-level details. Those high-level policies are sequestered and isolated from low-level details such that changes to the low-level details have no impact on the high-level policies." (p. 224 – 225, Martin, Clean Craftsmanship) It's important to keep software design simple to avoid overwhelming complexity. Software design is complex. It is the job of the software engineer to simplify that complexity. "Once complexity has accumulated, it is hard to eliminate, since fixing a single dependency or obscurity will not by itself, make a big difference." (p. 11, Ousterhout) Software engineers need to take a strategic approach to programming. They must focus on the long-term structure of the code and the application. "An abstraction is a simplified view of an entity, which omits unimportant details." (p. 21, Ousterhout)

"When designing modules such as classes and methods, one of the best ways to produce a deep API is to make it general-purpose (general-purpose APIs result in more information hiding)." (p. 39, Ousterhout) "It is more important for a module to have a simple interface than a simple implementation." (p. 61, Ousterhout) An important problem to solve in software design is to either combine functionality in a single class or to separate functionality into more than one class. Functionality should be brought together to eliminate duplicate code. General-purpose code should be separated from code that serves a specific purpose.

When implementing a method, ensure that it does only one thing.

Programming

See Chapter 2 for more information on programming.

Testing and Quality Assurance

Testing and test design are essential to delivering an effective, correct, and efficient software solution. Much time should be devoted to testing to eliminate potential bugs from appearing in the software in production.

Deployment

Deployment is the process of releasing software to a designated environment, whether it is a test environment or production environment. Releasing software to a production environment can be complex, and it is best to streamline the process.

Maintenance

Maintenance involves monitoring software after it has been release to a production environment.

Chapter 2: Programming

Programming

It’s essential to properly name variables, methods, and classes to communicate exactly what the intent of the code is. When naming classes, methods, and variables, it’s important to use concise and purposeful names. “Good names are a form of documentation: they make code easier to understand.” (p. 121, Ousterhout)

It’s important to maintain consistency when designing software. Style guidelines provide criteria for developers to write code effectively. “Modern style guides address a range of issues, such as indentation, curly-brace placement, order of declarations, naming, commenting, and restrictions on language features considered dangerous.” (p. 143 – 144, Ousterhout)

Make judicious use of white space in code. Blank lines are especially useful in separating blocks of code.

Code should be continuously improved every time programmers work on it.

Debugging

Refactoring

Refactoring code is essential to delivering a well-oiled and functioning software systems. It's important to first get the code to work as expected and then refactor the code to be efficient. Refactoring helps to eliminate duplicate code throughout the application. "Refactoring is a sequence of small changes that improve the structure of the software without changing its behavior—as proven by passing a comprehensive suite of tests after each change in the sequence." (p. 199, Martin, Clean Craftsmanship)

Source Code Control

Continuous Integration

Continuous integration (CI) is made up of frequent commits of code.

Documentation

It's essential for programmers to document their work through comments in the code, test designs, ReadMe files, and more. However, the code itself should be self-documenting. When developers write comments, those "comments should describe things that aren't obvious from the code." (p. 101, Ousterhout) When writing comments, it is important not to repeat the code.

Unit Tests

Unit tests should provide sufficient and ample coverage of code.

Developers can embrace Test-Driven Design (TDD) and write test cases before developing well-functioning and clean code. TDD improves the design of code and encourages the developer to write de-coupled code.

Acceptance tests are necessary to satisfy the business requirements of developed software.

Code Reviews

"Code reviews provide an opportunity for enforcing conventions and for educating new developers about the conventions. The more nit-picky that code reviewers are, the more quickly everyone on the team will learn the conventions, and the cleaner the code will be." (p. 145, Ousterhout)

- Questions to ask during a code review

Chapter 3: Object-Oriented Programming

Object-oriented programming takes advantage of modular design to encapsulate complexity.

The pillars of Object-Oriented Programming include the following:

- Abstraction
    - The process of exposing essential features of an entity by hiding the other irrelevant details
    - Reduces complexity and increases efficiency
    - Can have multiple abstractions
- Encapsulation
    - The process of putting data and the operations (functions) that can be performed on that data into a single container called class
    - Any programmer can use this class without knowing how it is implemented
    - Data is insulated and not directly accessible from the outside world
    - Leads to data hiding
- Inheritance
    - The definition of new classes as extensions of existing classes
    - The new class may have some additional properties and functionality
- Polymorphism
    - The ability to take multiple forms and allows the ability to invoke derived class methods through a base class reference during run-time

Chapter 4: Design Patterns

Chapter 5: Database Design

Chapter 6: User Experience

Chapter 7: User Interface Design

Chapter 8: Agile Leadership

The Agile Business Consortium offers the Nine Principles of Agile Leadership. Here they are:

1. Actions speak louder than words

   "Agile leadership is about not only driving and promoting change, it is also about being the change. Those who lead by example and actively engage in their own development, inspire people. This is through action rather than words; as Gandhi said, "Be the change you want to see". Agile Leaders develop themselves to be humble and empathetic by demonstrating virtues such as compassion, kindness and care for their colleagues. Inspiring leaders work on themselves first before working on others."

2. Improved quality of thinking leads to improved outcomes

   "Agile leaders value high quality thinking which will result in meaningful action. Agile leaders view problems from many different angles. They take input from those closest to the problem and this goes some way to ensuring that they are in touch with reality rather than relying solely on electronic information to inform their decision making. This also means allowing thinking time and focusing on the highest priorities at any given time."

3. Organizations improve through effective feedback

   "Receiving feedback can often be perceived as a negative experience, so Agile Leaders lead the way by courageously soliciting meaningful, useful and timely feedback from peers and other colleagues. While requesting feedback is important, agile leaders take time to ensure that they are visibly responding to the suggestions made by their colleagues in order to close the feedback loop. Agile leaders model giving effective feedback that is open, honest and respectful."

4. People require meaning and purpose to make work fulfilling

   "Agile leaders focus on building and sharing a common understanding and purpose. There is a vision of change that is meaningful and applicable to the organization. The work of the agile leader is to be aware of what is in the hearts and minds of their colleagues, and then to unify and align those values into inspired action."

5. Emotion is a foundation to enhanced creativity and innovation

   "Agile leaders inspire others to bring their best selves to their work. They understand that emotion is an important part of the human experience, and when individuals work with their emotions, they achieve more of their potential. Innovation and creativity rely heavily on respect that the agile leader encourages by being accessible, open, honest and transparent whilst expecting the same from others."

6. Leadership lives everywhere in the organization

   "Agile leadership should permeate all aspects of an organization or change initiative. Realizing the leadership potential of all its people helps accelerate the organization's ability to learn and adapt. The work of an agile leader is to develop depth in the organization's leadership capability by providing opportunities for their people to lead. Mentoring tomorrow's leaders in the principles and practices of servant leadership sows the seeds for the agile culture to thrive."

7. Leaders devolve appropriate power and authority

   "Agile leaders recognize that people work best when they are enabled, engaged and energized. Empowering individuals is a necessary skill of the agile leader as they balance the emerging needs and tensions of the organization. Agile leaders recognize that empowerment is not an "all or nothing" concept. Instead, it is a continuum of leadership behavior that responds to the current context for change."

8. Collaborative communities achieve more than individuals

   "Agile leaders build communities based on high trust, respect and meaningful working relationships. Their role is to provide those communities with all that they need to operate efficiently but then to let them function autonomously within their boundaries. The Agile Leader understands that forgiveness, positivity, generosity and gratitude are important parts of a healthy working environment. The healthy functioning of the group together with the preservation of psychological safety, allows the agile leader to encourage learning and development whilst also balancing sustained output and performance for the benefit of the organization."

9. Great ideas can come from anywhere in the organization

   "People who are close to a problem usually have the best ideas about how to solve it. Agile leaders allow themselves to be open to the influence and ideas of others, regardless of their status or position. To this end, the agile leader stops, listens and gives time to really hear the thoughts and ideas for improvement from their colleagues. Even if some ideas are not used, the agile leader encourages a continuous flow of creativity by helping people to understand which ideas were useful and which were not."

Inclusive Leadership

Bill Clinton

Chapter 9: Agile Project Management

Managing legacy systems while maintaining new versions of those systems

Project Management

Well-run software projects meet the following objectives (p.33, McConnell, Professional Software Development):

- Minimal defects
- Maximum user satisfaction
- Minimal response time
- Good maintainability
- Good extendability
- High robustness
- High correctness

- Joining a project in the middle of development

Waterfall Project Management

The Waterfall model of project management involves delivering all of the features of software to the end user at the end of a project. "Waterfall development attempts to minimize the amount of risk in a project by doing extensive planning and documentation prior to initiating construction of the software." (p. 383, Wiegers and Beatty)

Iterative Development

Iterative development aims to deliver software in incremental development, meaning that features are delivered periodically to the end user during the project.

Agile Project Management / Scrum

Agile project management, or Scrum, is a form of iterative project management. It comes from Lean Manufacturing. Agile development involves rapid and frequent delivery of incremental functionality. Agile development is appropriate when requirements inevitably change during a project. Agile project management breaks up the software development process into iterations, or sprints. In each sprint, the requirements are re-prioritized so that the most critical functionality is delivered first. "The goal is to have a body of potentially shippable software at the end of each iteration, even if it constitutes just a small portion of the ultimately desired product." (p. 385, Wiegers and Beatty) Customers are involved continuously during Agile development. Much of the documentation used by developers and QA professionals is contained in user stories. Product backlogs are used to prioritize requirements.

The Agile Manifesto is stated as:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

(Agile Manifesto)

User Stories

Waterfall Project Management with Agile Sprints

Some organizations that develop software in large multi-year projects do so in a combined Waterfall/Agile project management approach, where features are delivered incrementally over an expanded period of time.

Appendix A: Attributes of Leaders

1. Leaders take the initiative to do things.
2. Leaders lead by setting an example.
3. Leaders have the courage to change things when it is necessary.
4. Leaders must be willing to ask questions and pursue answers to those questions.
5. Leaders maintain goodwill when it is necessary.
6. Leaders instill in workers the kind of drive and creativity and innovative spirit more commonly found among entrepreneurs.
7. Leaders make an effort to communicate effectively.
8. Leaders work to build trust through honesty and integrity.
9. Leaders who aim for perfection can achieve greatness.
10. Leaders come up with several solutions to each problem and select the best one.
11. Leaders express clearly their expectations of others.
12. Leaders set clear objectives.

Sources

Software Engineering

Software Engineering for Absolute Beginners: Your Guide to Creating Software Products – Nico Loubser

Engineers Survival Guide – Merih Taze

Software Engineering – Ian Sommerville

AWS – What Is SDLC (Software Development Lifecycle)? - https://aws.amazon.com/what-is/sdlc/#:~:text=The%20software%20development%20lifecycle%20(SDLC,expectations%20during%20production%20and%20beyond.

Software Requirements – Karl Wiegers and Joy Beatty

More About Software Requirements – Karl Wiegers

Use Cases – Usability.gov - https://www.usability.gov/how-to-and-tools/methods/use-cases.html

A Beginner's Guide to Data Flow Diagrams – Hubspot – Clifford Chi - https://blog.hubspot.com/marketing/data-flow-diagram

A Philosophy of Software Design – John Ousterhout

Clean Architecture: A Craftsman's Guide to Software Structure and Design – Bob Martin

Microsoft .NET: Architecting Applications for the Enterprise – Dino Esposito and Andrea Saltarello

Software Architecture in Practice – Len Bass, Paul Clements, and Rick Kazman

Software Engineering Economics – Barry W. Boehm

Programming

Code Complete – Steve McConnell

Clean Code – Bob Martin

Clean Craftsmanship: Disciplines, Standards, and Ethics – Bob Martin

Coders at Work – Peter Seibel

Rapid Development – Steve McConnell

Refactoring: Improving the Design of Existing Code – Martin Fowler

The Nature of Software Development: Keep It Simple, Make It Valuable, Build It Piece by Piece – Ron Jeffries

The Pragmatic Programmer: Your Journey to Mastery – David Thomas and Andrew Hunt

Programming Pearls – Jon Bentley

Soft Skills: The Software Developer's Life Manual – John Sonmez

Object-Oriented Programming

The Object-Oriented Thought Process – Matt Weisfeld

Head First Object-Oriented Analysis & Design – Brett D. McLaughlin, Gary Pollice, and David West

Object-Oriented Analysis – Peter Coad and Edward Yourdon

Object-Oriented Design – Peter Coad and Edward Yourdon

Designing Object-Oriented Software – Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener

Object Design: Roles, Responsibilities, and Collaborations – Rebecca Wirfs-Brock and Alan McKean

Object-Oriented Programming – Peter Coad and Jill Nicola

Object-Oriented Software Construction – Bertrand Meyer

Design Patterns

Adaptive Code via C#: Agile Coding with Design Patterns and SOLID Principles – Gary McLean Hall

Head First Design Patterns: A Brain-Friendly Guide – Eric Freeman and Elisabeth Robson

Dependency Injection in .NET – Mark Seemann

Design Patterns: Elements of Reusable Object-Oriented Software – Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides

Professional ASP.NET Design Patterns – Scott Millett

Patterns, Principles, and Practices of Domain-Driven Design – Scott Millett and Nick Tune

Enterprise Integration Patterns – Gregor Hohpe and Bobby Woolf

The Art of Unit Testing – Roy Osherove

Software Testing Techniques: Finding the Defects that Matter – Scott Loveland, Geoffrey Miller, Richard Prewitt, and Michael Shannon

Database Design

A First Course in Database Systems – Jeffrey D. Ullman

SQL Fundamentals – John J. Patrick

Microsoft SQL Server 2008 T-SQL Fundamentals – Itzik Ben-Gan

Programming SQL Server 2008 – Leonard Lobel, Andrew J. Brust, and Stephen Forte

Inside Microsoft SQL Server 2008: T-SQL Programming – Itzik Ben-Gan

SQL Server 2008 Query Performance Tuning Distilled – Grant Fritchey and Sajal Dam

Inside SQL Server 2008: T-SQL Querying – Itzik Ben-Gan

Microsoft SQL Server 2008 Internals – Kalen Delaney

User Experience

The Elements of User Experience – Jesse James Garrett

The Design of Everyday Things – Don Norman

User Interface Design

Don’t Make Me Think: A Common Sense Approach to Web and Mobile Usability – Steve Krug

A Guide to Usability: Human Factors in Computing – Jenny Preece

Agile Leadership

Theory of Constraints – Eliyahu M. Goldratt

Agile Business Consortium – The Nine Principles of Agile Leadership - https://www.agilebusiness.org/resource/the-nine-principles-of-agile-leadership.html

Strategic Doing: Ten Skills for Agile Leadership – Edward Morrison, Scott Hutcheson, Elizabeth Nilsen, Janyce Fadden, and Nancy Franklin

Doing Agile Right: Transformation Without Chaos – Darrell Rigby, Sarah Elk, and Steve Berez

The Rise of the Agile Leader: Can You Make the Shift – Chuck Mollor

The Professional Agile Leader: The Leader's Journey Toward Growing Mature Agile Teams and Organizations – Ron Eringa, Kurt Bittner, and Laurens Bonnema

Agile Leadership Toolkit: Learning to Thrive with Self-Managing Teams – Peter Koning

The Epic Guide to Agile: More Business Value on a Predictable Schedule with Scrum – Dave Todaro

Inclusive Leadership: The Definitive Guide to Developing and Executing an Impactful Diversity and Inclusion Strategy – Charlotte Sweeney and Fleur Bothwick

The Inclusive Leadership Journal: A 90-Day Journey to Becoming a More Inclusive People Leader – Adriele Parker

LinkedIn – Inclusive Leadership for Agile Teams – Rene Carayol - https://www.linkedin.com/pulse/inclusive-leadership-agile-teams-ren%C3%A9-carayol-mbe/ - Accessed September 2, 2024

Harvard Business Review – Why Inclusive Leaders Are Good for Organizations, and How to Become One – Juliet Bourke and Andrea Titus - https://hbr.org/2019/03/why-inclusive-leaders-are-good-for-organizations-and-how-to-become-one - Accessed September 30, 2023

Heidrick & Struggles – Six Actions Leaders Can Take to Nurture Inclusive Teams – Alice Breeden and TA Mitchell - https://www.heidrick.com/en/insights/diversity-inclusion/six-actions-leaders-can-take-to-nurture-inclusive-teams - Accessed September 15, 2024

My Life – Bill Clinton

The Survivor: Bill Clinton in the White House – John F. Harris

First in His Class: A Biography of Bill Clinton – David Maraniss

The Agenda: Inside the Clinton White House – Bob Woodward

Bill Clinton: Mastering the Presidency – Nigel Hamilton

Bill Clinton: An American Journey – Nigel Hamilton

Bill Clinton: New Gilded Age President – Patrick J. Maney

Agile Project Management

HBR on Leadership – NASA's Former Head of Science on What It Takes to Manage Complex, High-Risk Projects

Harvard Business Review – Why Big Projects Fail – and How to Give Yours a Better Chance of Success – Te Wu and Ram B. Misra - https://hbr.org/2023/11/why-big-projects-fail-and-how-to-give-yours-a-better-chance-of-success - Accessed November 3, 2023

Harvard Business Review – How Project Managers Can Better Navigate Setbacks – Amy Shoenthal - https://hbr.org/2023/10/how-project-managers-can-better-navigate-setbacks?ab=HP-hero-latest-text-2 – Accessed October 30, 2023

Harvard Business Review – Keep Your Team Motivated When a Project Goes Off the Rails – Rebecca Zucker - https://hbr.org/2023/10/keep-your-team-motivated-when-a-project-goes-off-the-rails - Accessed November 5, 2023

Harvard Business Review – Project Managers, Focus on Outcomes – Not Deliverables – Andrea Belk Olson - https://hbr.org/2023/11/project-managers-focus-on-outcomes-not-deliverables - Accessed November 3, 2023

Harvard Business Review – Project Managers Should Think Like Startup Founders – Ron Ashkenas - https://hbr.org/2023/11/project-managers-should-think-like-startup-founders - Accessed November 5, 2023

Harvard Business Review – What the Next Generation of Project Management Will Look Like – Rachel Longhurst and Woojin Choi - https://hbr.org/2023/11/what-the-next-generation-of-project-management-will-look-like - Accessed November 6, 2023

Software Project Survival Guide – Steve McConnell

The Mythical Man-Month – Frederick P. Brooks

Working Effectively with Legacy Code – Michael C. Feathers

Agile Manifesto - https://agilemanifesto.org/

Harvard Business Review – Planning Doesn't Have to Be the Enemy of Agile – Alessandro Di Fiore - https://hbr.org/2018/09/planning-doesnt-have-to-be-the-enemy-of-agile - Accessed September 18, 2023

Agile Project Management with Scrum – Ken Schwaber

Succeeding With Agile – Mike Cohn

Agile Estimating and Planning – Mike Cohn

Agile Testing – Lisa Crispin and Janet Gregory

User Stories Applied – Mike Cohn

Additional Sources

Professional Software Development – Steve McConnell