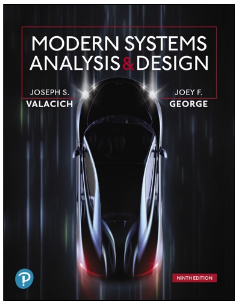
Modern Systems Analysis and Design

Ninth Edition



Chapter 1

The Systems Development Environment

Pearson

1

Learning Objectives

1.1 Define information systems analysis and design

1.2 Describe the information systems development life cycle (SDLC)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

1.4 Explain object-oriented analysis and design and the Rational Unified Process (RUP)

Pearson

2

Introduction

- **Information Systems Analysis and Design**
  - Defined as the complex, challenging, and simulating organizational process that a team of business and systems professionals uses to develop and maintain information systems
- **Application Software**
  - Software designed to support organizational function or process
- **Systems Analyst**
  - Organizational role most responsible for analysis and design of information systems

Pearson

3

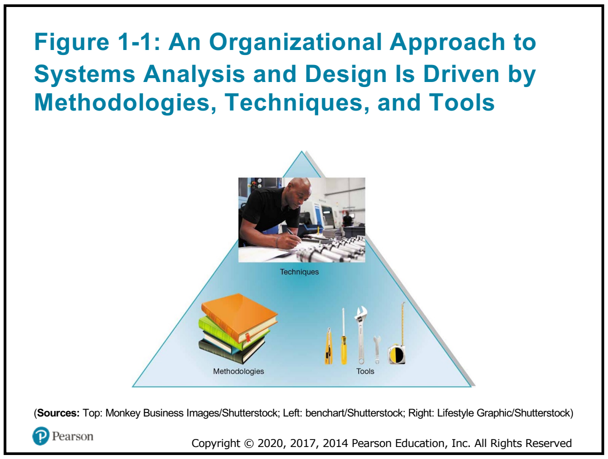
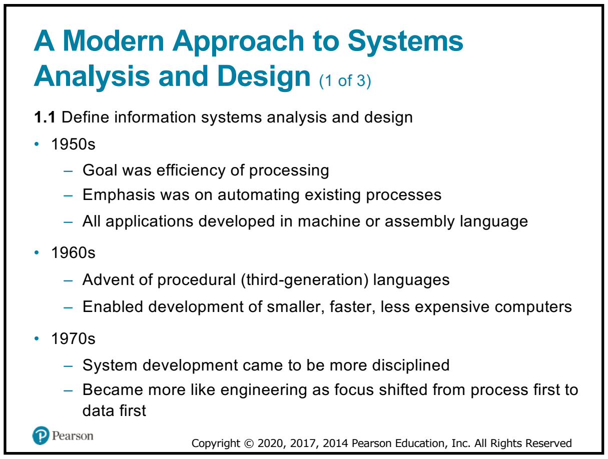


Figure 1-1: An Organizational Approach to Systems Analysis and Design Is Driven by Methodologies, Techniques, and Tools

(**Sources:** Top: Monkey Business Images/Shutterstock; Left: benchart/Shutterstock; Right: Lifestyle Graphic/Shutterstock)

Pearson

4



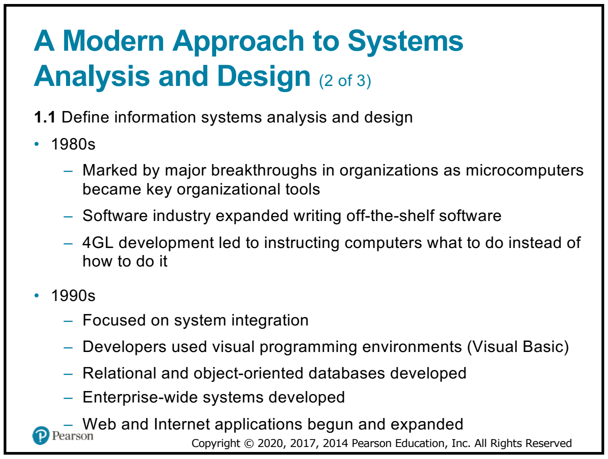
A Modern Approach to Systems Analysis and Design (1 of 3)

1.1 Define information systems analysis and design

- 1950s
  - Goal was efficiency of processing
  - Emphasis was on automating existing processes
  - All applications developed in machine or assembly language
- 1960s
  - Advent of procedural (third-generation) languages
  - Enabled development of smaller, faster, less expensive computers
- 1970s
  - System development came to be more disciplined
  - Became more like engineering as focus shifted from process first to data first

Pearson

5



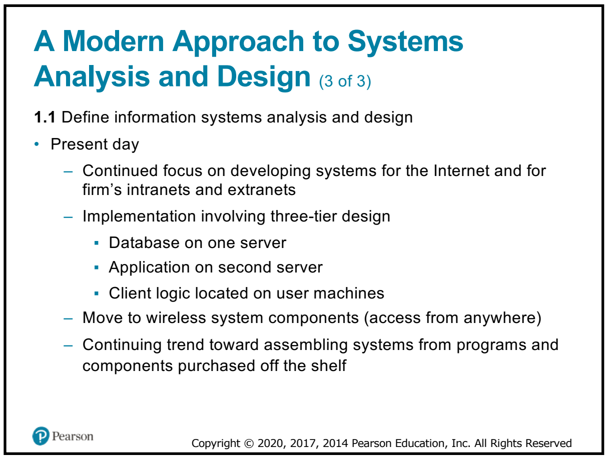
A Modern Approach to Systems Analysis and Design (2 of 3)

1.1 Define information systems analysis and design

- 1980s
  - Marked by major breakthroughs in organizations as microcomputers became key organizational tools
  - Software industry expanded writing off-the-shelf software
  - 4GL development led to instructing computers what to do instead of how to do it
- 1990s
  - Focused on system integration
  - Developers used visual programming environments (Visual Basic)
  - Relational and object-oriented databases developed
  - Enterprise-wide systems developed
  - Web and Internet applications begun and expanded

Pearson

6

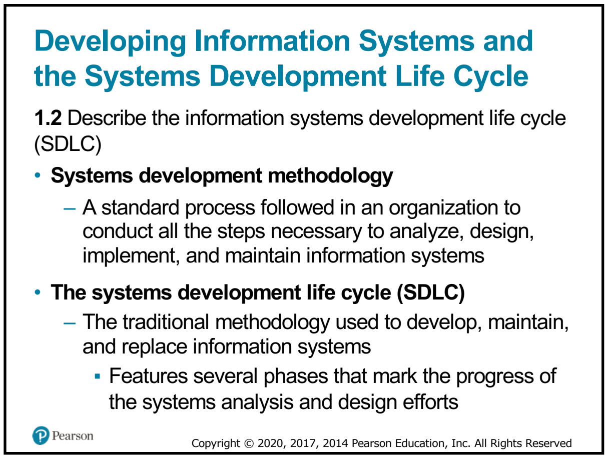


A Modern Approach to Systems Analysis and Design (3 of 3)

1.1 Define information systems analysis and design

- Present day
  - Continued focus on developing systems for the Internet and for firm's intranets and extranets
  - Implementation involving three-tier design
    - Database on one server
    - Application on second server
    - Client logic located on user machines
  - Move to wireless system components (access from anywhere)
  - Continuing trend toward assembling systems from programs and components purchased off the shelf

7

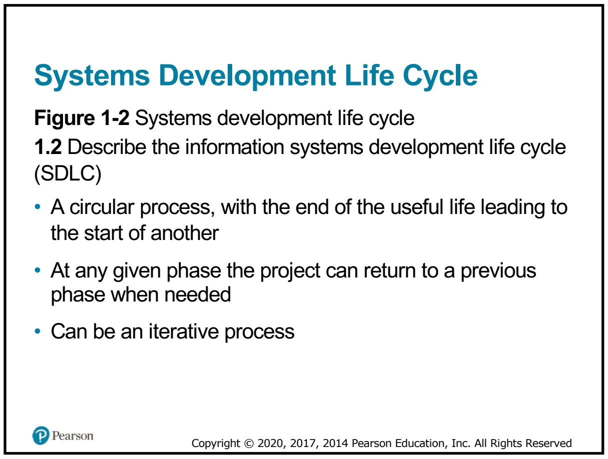


Developing Information Systems and the Systems Development Life Cycle

1.2 Describe the information systems development life cycle (SDLC)

- **Systems development methodology**
  - A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems
- **The systems development life cycle (SDLC)**
  - The traditional methodology used to develop, maintain, and replace information systems
    - Features several phases that mark the progress of the systems analysis and design efforts

8



Systems Development Life Cycle

Figure 1-2 Systems development life cycle

1.2 Describe the information systems development life cycle (SDLC)

- A circular process, with the end of the useful life leading to the start of another
- At any given phase the project can return to a previous phase when needed
- Can be an iterative process

9

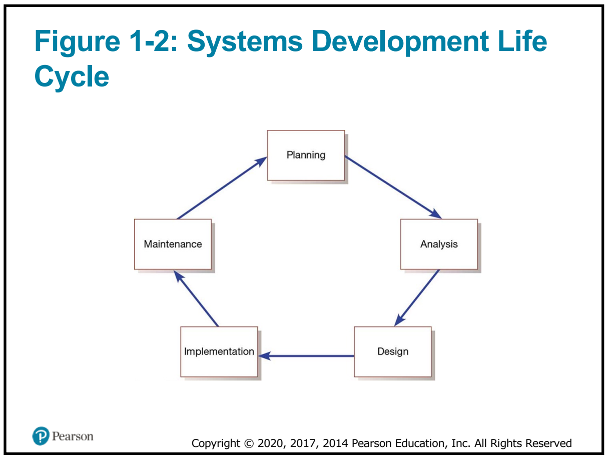
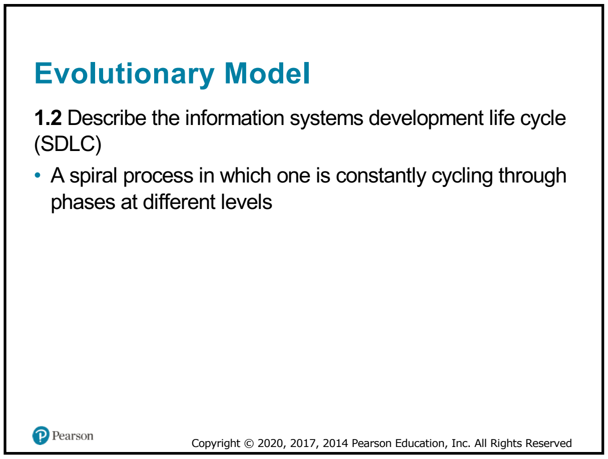


Figure 1-2: Systems Development Life Cycle

10



Evolutionary Model

1.2 Describe the information systems development life cycle (SDLC)

- A spiral process in which one is constantly cycling through phases at different levels

11

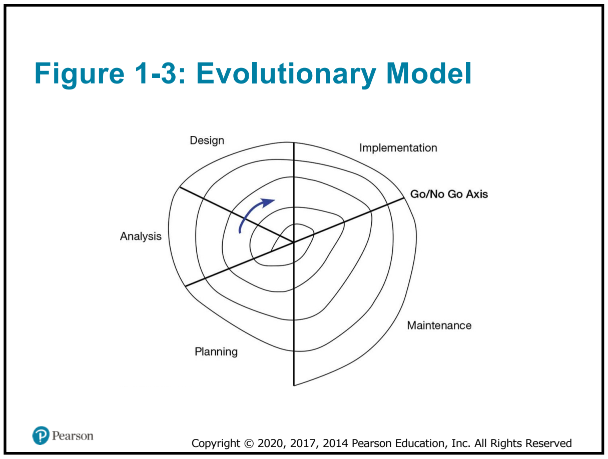
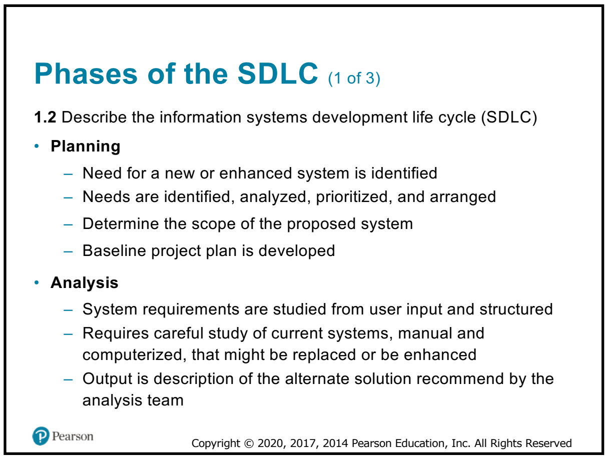


Figure 1-3: Evolutionary Model

12

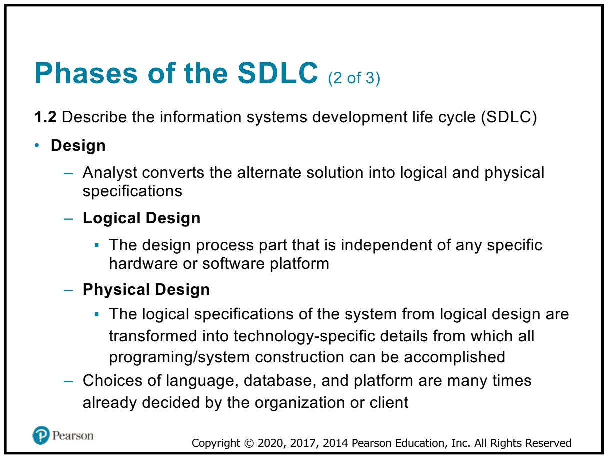


Phases of the SDLC (1 of 3)

1.2 Describe the information systems development life cycle (SDLC)

- **Planning**
  - Need for a new or enhanced system is identified
  - Needs are identified, analyzed, prioritized, and arranged
  - Determine the scope of the proposed system
  - Baseline project plan is developed
- **Analysis**
  - System requirements are studied from user input and structured
  - Requires careful study of current systems, manual and computerized, that might be replaced or be enhanced
  - Output is description of the alternate solution recommend by the analysis team

13

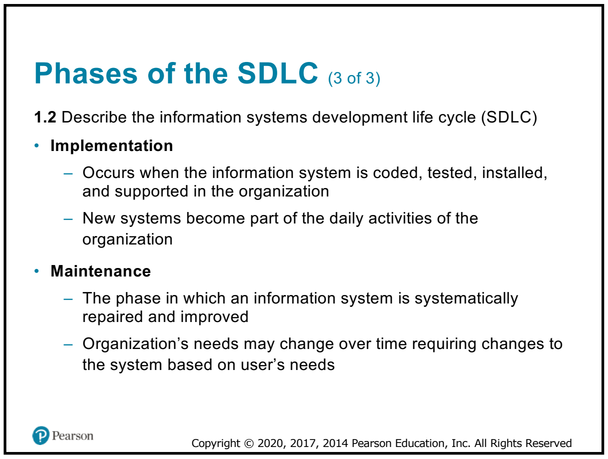


Phases of the SDLC (2 of 3)

1.2 Describe the information systems development life cycle (SDLC)

- **Design**
  - Analyst converts the alternate solution into logical and physical specifications
  - **Logical Design**
    - The design process part that is independent of any specific hardware or software platform
  - **Physical Design**
    - The logical specifications of the system from logical design are transformed into technology-specific details from which all programing/system construction can be accomplished
  - Choices of language, database, and platform are many times already decided by the organization or client

14



Phases of the SDLC (3 of 3)

1.2 Describe the information systems development life cycle (SDLC)

- **Implementation**
  - Occurs when the information system is coded, tested, installed, and supported in the organization
  - New systems become part of the daily activities of the organization
- **Maintenance**
  - The phase in which an information system is systematically repaired and improved
  - Organization’s needs may change over time requiring changes to the system based on user’s needs

15

Table 1-1: Products of SDLC Phases

1.2 Describe the information systems development life cycle (SDLC)

Phase	Products, Outputs, or Deliverables
Planning	• Priorities for system and projects; an architecture for data, networks, and selection hardware, and information systems management are the result of associated systems • Detailed steps, or work plan, for project • Specification of system scope and planning and high-level system requirements or features • Assignment of team members and other resources • System justification or business case
Analysis	• Description of current system and where problems or opportunities exist, with a general recommendation on how to fix, enhance, or replace current system • Explanation of alternative systems and justification for chosen alternative
Design	• Functional, detailed specifications of system elements (data, processes, inputs, and outputs) • Technical, detailed specifications of all system elements (programs, files, network, system software, etc.) • Acquisition plan for new technology
Implementation	• Code, documentation, training procedures, and support capabilities
Maintenance	• New versions or releases of software with associated updates to documentation, training, and support

Pearson

16

Analysis-Design-Code-Test Loop

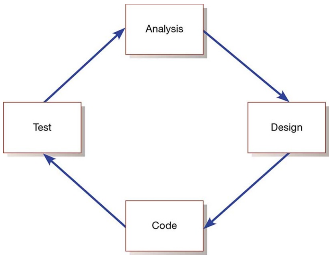
1.2 Describe the information systems development life cycle (SDLC)

- The Analysis-Design-Code-Test Loop is an example of traditional practice

Pearson

17

Figure 1-6: Analysis-Design-Code-Test Loop



Pearson

18

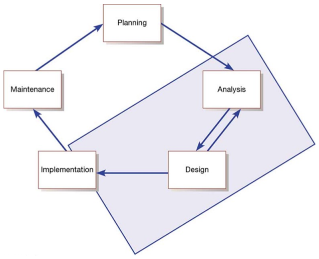
Heart of Systems Development

1.2 Describe the information systems development life cycle (SDLC)

- Current practice combines analysis, design, and implementation into a single process

19

Figure 1-7: Heart of Systems Development



20

The SDLC Traditional Waterfall Problems

1.2 Describe the information systems development life cycle (SDLC)

- Once one phase ends another begins, going downhill until complete
- Makes it difficult to go back
- Results in great expense to make changes
- Role of system users or customers narrowly defined
- Focused on deadlines

21

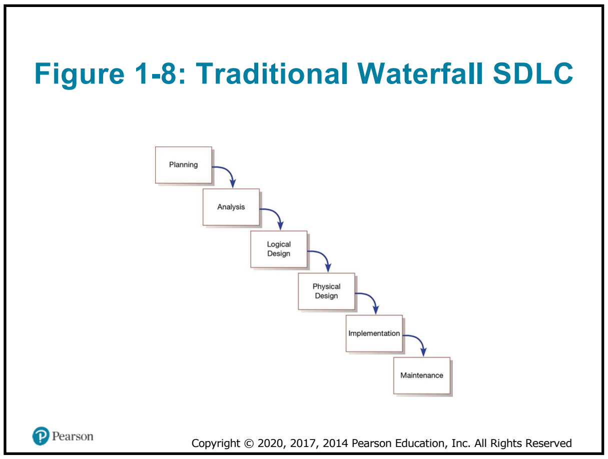
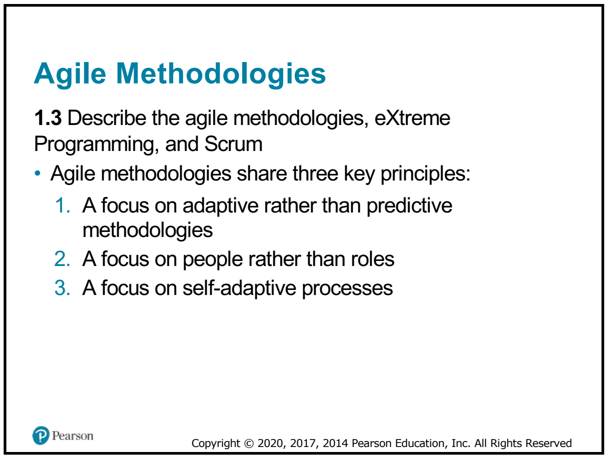


Figure 1-8: Traditional Waterfall SDLC

Pearson

22



Agile Methodologies

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Agile methodologies share three key principles:
  1. A focus on adaptive rather than predictive methodologies
  2. A focus on people rather than roles
  3. A focus on self-adaptive processes

Pearson

23

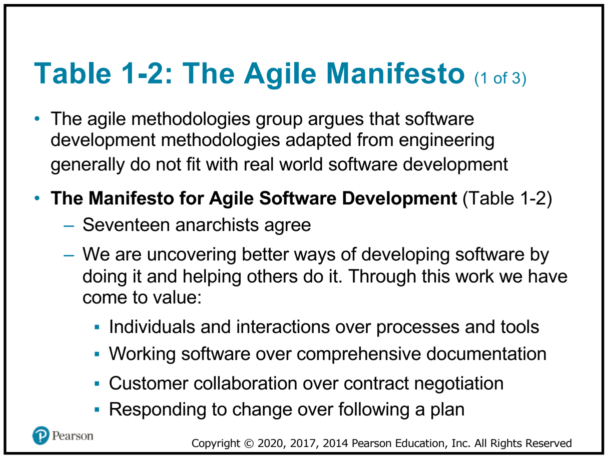


Table 1-2: The Agile Manifesto (1 of 3)

- The agile methodologies group argues that software development methodologies adapted from engineering generally do not fit with real world software development
- **The Manifesto for Agile Software Development** (Table 1-2)
  - Seventeen anarchists agree
  - 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

Pearson

24

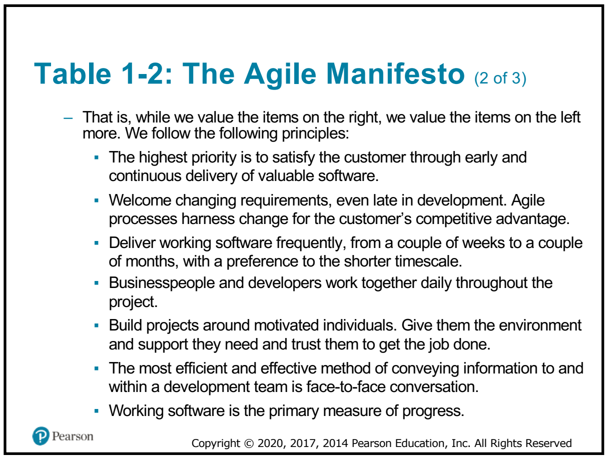


Table 1-2: The Agile Manifesto (2 of 3)

- That is, while we value the items on the right, we value the items on the left more. We follow the following principles:
  - The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  - Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
  - Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
  - Businesspeople and developers work together daily throughout the project.
  - Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
  - The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
  - Working software is the primary measure of progress.

Pearson

25

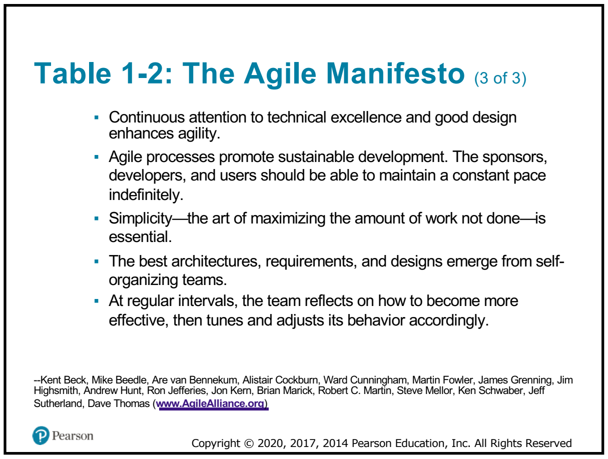


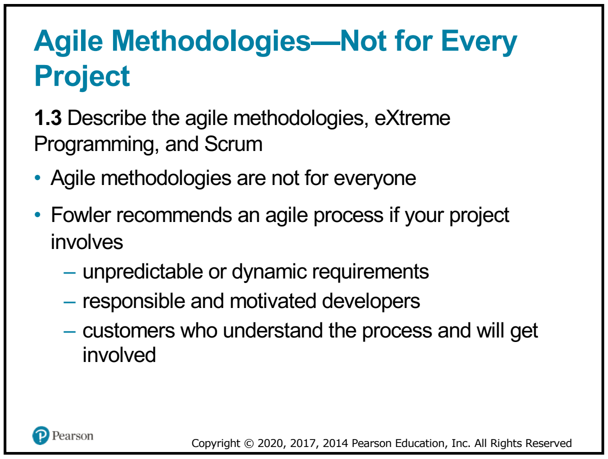
Table 1-2: The Agile Manifesto (3 of 3)

- Continuous attention to technical excellence and good design enhances agility.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

--Kent Beck, Mike Beedle, Are van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jefferies, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas (www.AgileAlliance.org)

Pearson

26



Agile Methodologies—Not for Every Project

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Agile methodologies are not for everyone
- Fowler recommends an agile process if your project involves
  - unpredictable or dynamic requirements
  - responsible and motivated developers
  - customers who understand the process and will get involved

Pearson

27

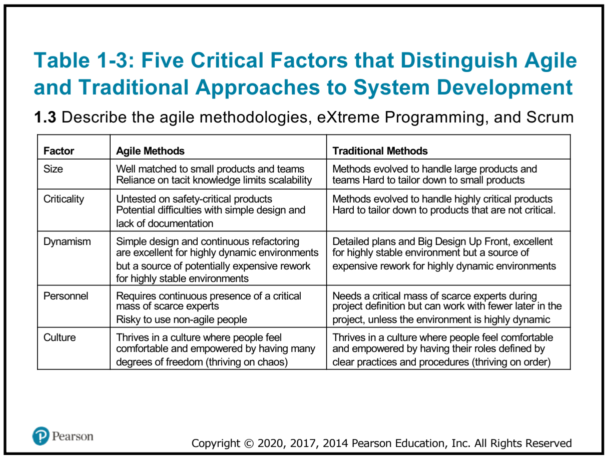


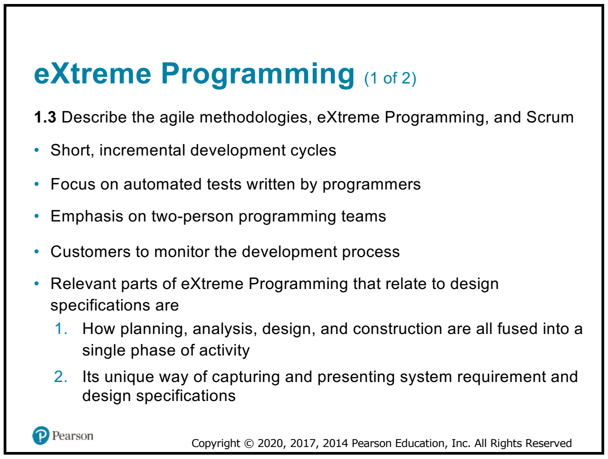
Table 1-3: Five Critical Factors that Distinguish Agile and Traditional Approaches to System Development

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

Factor	Agile Methods	Traditional Methods
Size	Well matched to small products and teams Reliance on tacit knowledge limits scalability	Methods evolved to handle large products and teams Hard to tailor down to small products
Criticality	Untested on safety-critical products Potential difficulties with simple design and lack of documentation	Methods evolved to handle highly critical products Hard to tailor down to products that are not critical.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments but a source of potentially expensive rework for highly stable environments	Detailed plans and Big Design Up Front, excellent for highly stable environment but a source of expensive rework for highly dynamic environments
Personnel	Requires continuous presence of a critical mass of scarce experts Risky to use non-agile people	Needs a critical mass of scarce experts during project definition but can work with fewer later in the project, unless the environment is highly dynamic
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom (thriving on chaos)	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear practices and procedures (thriving on order)

Pearson

28



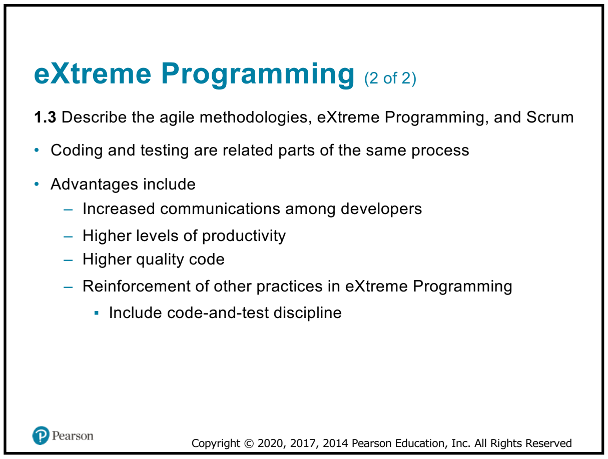
eXtreme Programming (1 of 2)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Short, incremental development cycles
- Focus on automated tests written by programmers
- Emphasis on two-person programming teams
- Customers to monitor the development process
- Relevant parts of eXtreme Programming that relate to design specifications are
  1. How planning, analysis, design, and construction are all fused into a single phase of activity
  2. Its unique way of capturing and presenting system requirement and design specifications

Pearson

29



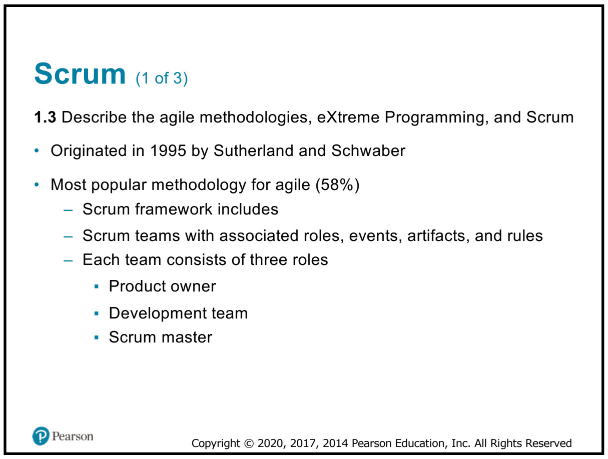
eXtreme Programming (2 of 2)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Coding and testing are related parts of the same process
- Advantages include
  - Increased communications among developers
  - Higher levels of productivity
  - Higher quality code
  - Reinforcement of other practices in eXtreme Programming
    - Include code-and-test discipline

Pearson

30

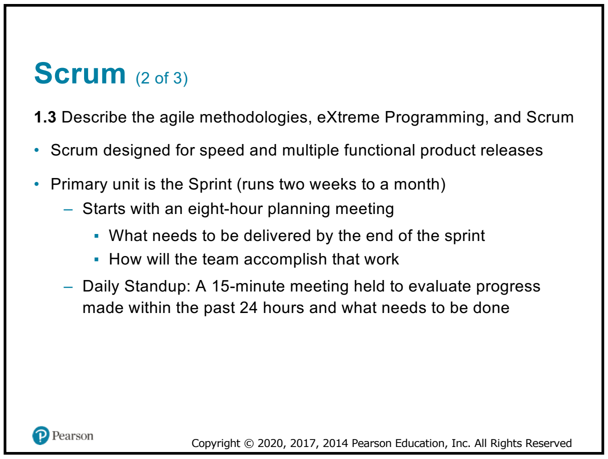


Scrum (1 of 3)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Originated in 1995 by Sutherland and Schwaber
- Most popular methodology for agile (58%)
  - Scrum framework includes
  - Scrum teams with associated roles, events, artifacts, and rules
  - Each team consists of three roles
    - Product owner
    - Development team
    - Scrum master

31

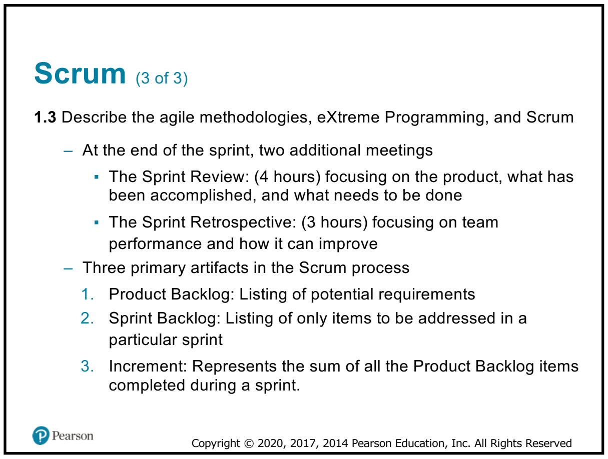


Scrum (2 of 3)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Scrum designed for speed and multiple functional product releases
- Primary unit is the Sprint (runs two weeks to a month)
  - Starts with an eight-hour planning meeting
    - What needs to be delivered by the end of the sprint
    - How will the team accomplish that work
  - Daily Standup: A 15-minute meeting held to evaluate progress made within the past 24 hours and what needs to be done

32

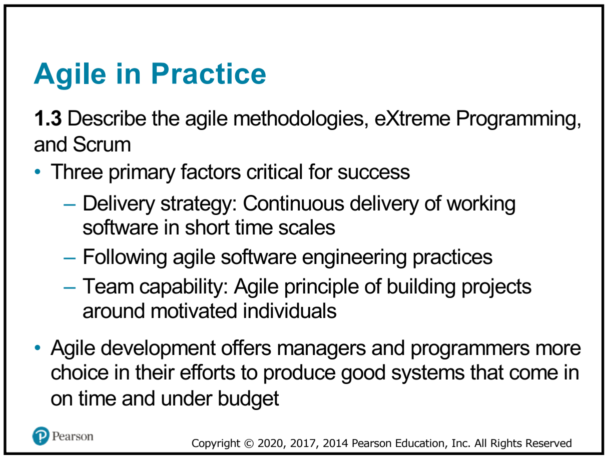


Scrum (3 of 3)

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- At the end of the sprint, two additional meetings
  - The Sprint Review: (4 hours) focusing on the product, what has been accomplished, and what needs to be done
  - The Sprint Retrospective: (3 hours) focusing on team performance and how it can improve
- Three primary artifacts in the Scrum process
  1. Product Backlog: Listing of potential requirements
  2. Sprint Backlog: Listing of only items to be addressed in a particular sprint
  3. Increment: Represents the sum of all the Product Backlog items completed during a sprint.

33

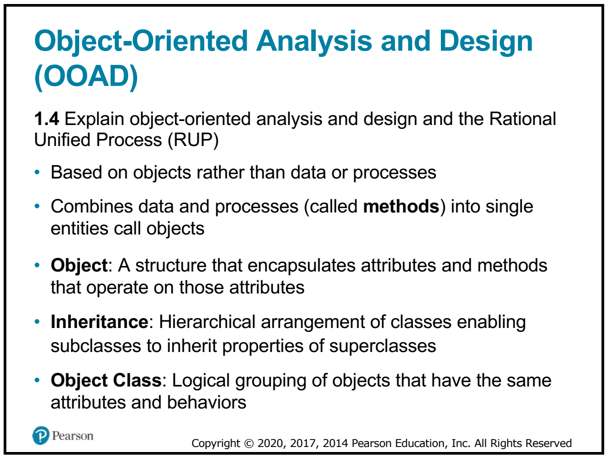


Agile in Practice

1.3 Describe the agile methodologies, eXtreme Programming, and Scrum

- Three primary factors critical for success
  - Delivery strategy: Continuous delivery of working software in short time scales
  - Following agile software engineering practices
  - Team capability: Agile principle of building projects around motivated individuals
- Agile development offers managers and programmers more choice in their efforts to produce good systems that come in on time and under budget

34

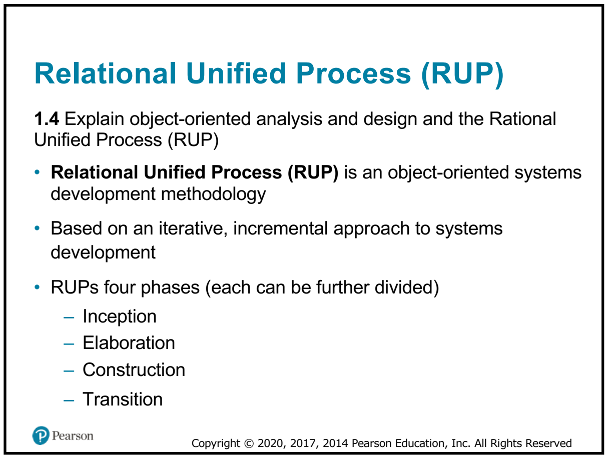


Object-Oriented Analysis and Design (OOAD)

1.4 Explain object-oriented analysis and design and the Rational Unified Process (RUP)

- Based on objects rather than data or processes
- Combines data and processes (called **methods**) into single entities call objects
- **Object**: A structure that encapsulates attributes and methods that operate on those attributes
- **Inheritance**: Hierarchical arrangement of classes enabling subclasses to inherit properties of superclasses
- **Object Class**: Logical grouping of objects that have the same attributes and behaviors

35



Relational Unified Process (RUP)

1.4 Explain object-oriented analysis and design and the Rational Unified Process (RUP)

- **Relational Unified Process (RUP)** is an object-oriented systems development methodology
- Based on an iterative, incremental approach to systems development
- RUPs four phases (each can be further divided)
  - Inception
  - Elaboration
  - Construction
  - Transition

36

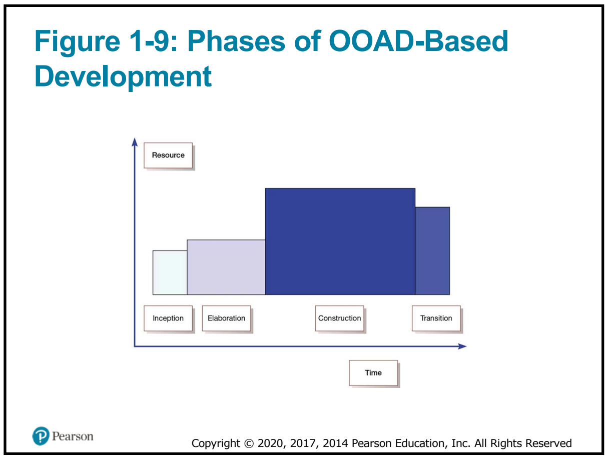
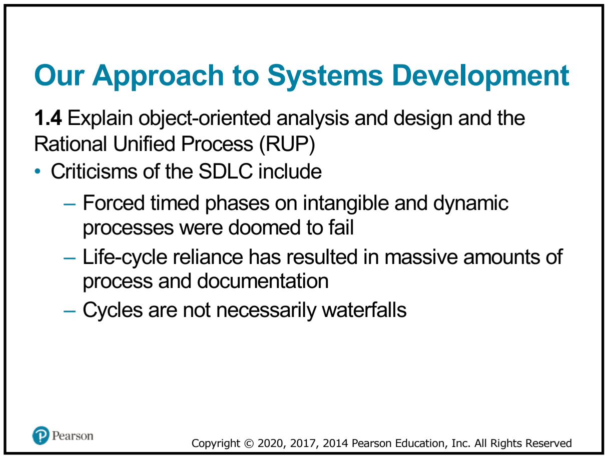


Figure 1-9: Phases of OOAD-Based Development

Pearson

37



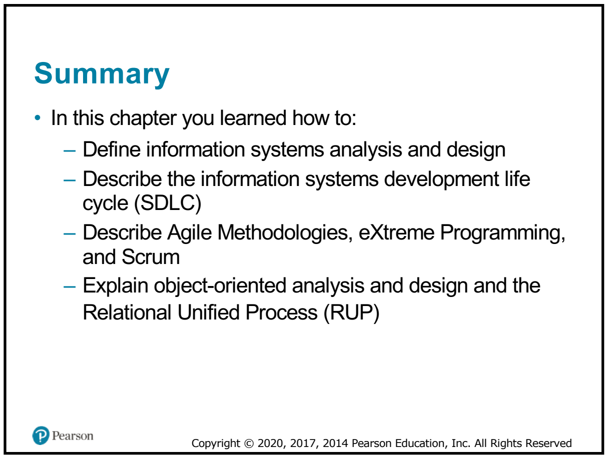
Our Approach to Systems Development

1.4 Explain object-oriented analysis and design and the Rational Unified Process (RUP)

- Criticisms of the SDLC include
  - Forced timed phases on intangible and dynamic processes were doomed to fail
  - Life-cycle reliance has resulted in massive amounts of process and documentation
  - Cycles are not necessarily waterfalls

Pearson

38

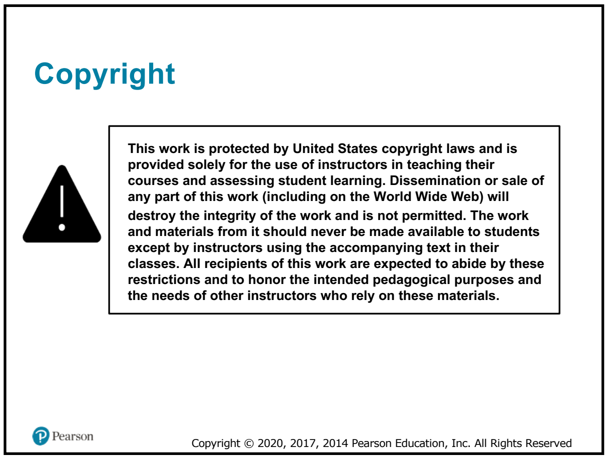


Summary

- In this chapter you learned how to:
  - Define information systems analysis and design
  - Describe the information systems development life cycle (SDLC)
  - Describe Agile Methodologies, eXtreme Programming, and Scrum
  - Explain object-oriented analysis and design and the Relational Unified Process (RUP)

Pearson

39



Copyright

Pearson

40