

---

# Part III. Requirements for specific project classes

CHAPTER 20 Agile

CHAPTER 21 Enhancement and reengineering projects

CHAPTER 22 Packaged solution projects

CHAPTER 23 Outsourced projects

CHAPTER 24 Business process automation projects

CHAPTER 25 Business analytics projects

CHAPTER 26 Embedded and other real-time systems projects

A decorative graphic at the bottom of the slide consisting of diagonal stripes in orange and white.

---

# CHAPTER 20 Agile

- Describes the characteristics of agile approaches as they relate to the requirements activities for a software project, the major adaptations of traditional requirements practices for an agile project, and a road map of where to find more detailed guidance.
- Student should recognize what and how they have to do in every stage of agile projects

1. Limitations of the waterfall
2. The agile development approach
3. Essential aspects of an agile approach to requirements
4. Adapting requirements practices to agile projects
5. Transitioning to agile: Now what?

# Limitations of the waterfall

---

- What it is?
- What are advantages?
- What are disadvantages?
- When we use waterfall development?

- What is agile development approach?
- What are advantages?
- When we use agile development approach?

# Essential aspects of an agile approach to requirements

---

- Customer involvement
- Documentation detail
- The backlog and prioritization
- Timing
- Epics, user stories, and features
- Expect change

# Adapting requirements practices to agile projects

**TABLE 20-1** A road map to chapters that address agile development topics

Chapter	Topic
Chapter 2, "Requirements from the customer's perspective"	Reaching agreement on requirements
Chapter 4, "The business analyst"	The BA's role on agile projects and who is responsible for the requirements artifacts created
Chapter 5, "Establishing the business requirements"	Setting and managing the vision and scope
Chapter 6, "Finding the voice of the user"	User representation
Chapter 8, "Understanding user requirements"	User stories
Chapter 10, "Documenting the requirements"	Specifying requirements for agile development
Chapter 12, "A picture is worth 1024 words"	Modeling on agile projects
Chapter 14, "Beyond functionality"	Identifying quality attributes, especially those needed up front for architecture and design
Chapter 15, "Risk reduction through prototyping"	Agile projects and evolutionary prototyping
Chapter 16, "First things first: Setting requirement priorities"	Prioritization on agile projects
Chapter 17, "Validating the requirements"	Acceptance criteria and acceptance tests
Chapter 27, "Requirements management practices"	Managing requirements on agile projects through backlogs and burndown charts
Chapter 28, "Change happens"	Managing change on agile projects



- Determine what your role is on the team
- Identify suggested agile practices that will work best in your organization
- Implement a small project first as a pilot for agile methods
- ...


---

# **CHAPTER 21 Enhancement and replacement projects**

# Objectives

---

- Exploring some suggestions to practice the enhancement and replacement projects.
- After finish this chapter, student should know what and how they have to do in enhancement and replacement projects.

- 
- What is Enhancement and replacement project?
  - Expected challenges
  - Requirements techniques when there is an existing system
  - Prioritizing by using business objectives
  - When old requirements don't exist
  - Encouraging new system adoption
  - Can we iterate?
- 
- A decorative graphic at the bottom of the slide consisting of a series of parallel diagonal lines in orange and white, creating a striped effect.

# What is enhancement and replacement project?

---

- An enhancement project is one in which new capabilities are added to an existing system
- A replacement (or reengineering) project replaces an existing application with a new custom-built system, a commercial off-the-shelf (COTS) system, or a hybrid of those

# Expected challenges

---

- The changes made could degrade the performance to which users are accustomed.
- Little or no requirements documentation might be available for the existing system.
- Users who are familiar with how the system works today might not like the changes they are about to encounter.
- You might unknowingly break or omit functionality that is vital to some stakeholder group.
- Stakeholders might take this opportunity to request new functionality that seems like a good idea but isn't really needed to meet the business objectives

Technique	Why it's relevant
Create a feature tree to show changes	<ul style="list-style-type: none"> <li>■ Show features being added.</li> <li>■ Identify features from the existing system that won't be in the new system.</li> </ul>
Identify user classes	<ul style="list-style-type: none"> <li>■ Assess who is affected by the changes.</li> <li>■ Identify new user classes whose needs must be met.</li> </ul>
Understand business processes	<ul style="list-style-type: none"> <li>■ Understand how the current system is intertwined with stakeholders' daily jobs and the impacts of it changing.</li> <li>■ Define new business processes that might need to be created to align with new features or a replacement system.</li> </ul>
Document business rules	<ul style="list-style-type: none"> <li>■ Record business rules that are currently embedded in code.</li> <li>■ Look for new business rules that need to be honored.</li> <li>■ Redesign the system to better handle volatile business rules that were expensive to maintain.</li> </ul>
Create use cases or user stories	<ul style="list-style-type: none"> <li>■ Understand what users must be able to do with the system.</li> <li>■ Understand how users expect new features to work.</li> <li>■ Prioritize functionality for the new system.</li> </ul>
Create a context diagram	<ul style="list-style-type: none"> <li>■ Identify and document external entities.</li> <li>■ Extend existing interfaces to support new features.</li> <li>■ Identify current interfaces that might need to be changed.</li> </ul>
Create an ecosystem map	<ul style="list-style-type: none"> <li>■ Look for other affected systems.</li> <li>■ Look for new, modified, and obsolete interfaces between systems.</li> </ul>
Create a dialog map	<ul style="list-style-type: none"> <li>■ See how new screens fit into the existing user interface.</li> <li>■ Show how the workflow screen navigation will change.</li> </ul>
Create data models	<ul style="list-style-type: none"> <li>■ Verify that the existing data model is sufficient or extend it for new features.</li> <li>■ Verify that all of the data entities and attributes are still needed.</li> <li>■ Consider what data has to be migrated, converted, corrected, archived, or discarded.</li> </ul>
Specify quality attributes	<ul style="list-style-type: none"> <li>■ Ensure that the new system is designed to fulfill quality expectations.</li> <li>■ Improve satisfaction of quality attributes over the existing system.</li> </ul>
Create report tables	<ul style="list-style-type: none"> <li>■ Convert existing reports that are still needed.</li> <li>■ Define new reports that aren't in the old system.</li> </ul>
Build prototypes	<ul style="list-style-type: none"> <li>■ Engage users in the redevelopment process.</li> <li>■ Prototype major enhancements if there are uncertainties.</li> </ul>
Inspect requirements specifications	<ul style="list-style-type: none"> <li>■ Identify broken links in the traceability chain.</li> <li>■ Determine if any previous requirements are obsolete or unnecessary in the replacement system.</li> </ul>

Requirements techniques when there is an existing system



# Prioritizing by using business objectives

---

- Mind the gap
- Maintaining performance levels





# When old requirements don't exist

---

- Which requirements should you specify?
- How to discover the requirements of an existing system



# Encouraging new system adoption

---

- Why we do this?
- What are advantages?
- How to do this?

---

# **CHAPTER 22**

## **Packaged solution projects**

# Objectives

---

- Student should understand what is packaged solution project and how to get requirements for this project.

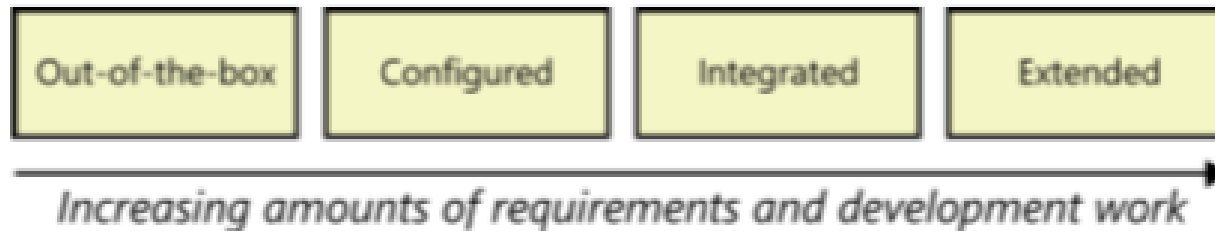
- 
- What is packaged solution project?
  - Requirements for selecting packaged solutions

# Requirements for selecting packaged solutions

---

- Developing user requirements
- Considering business rules
- Identifying data needs
- Defining quality requirements
- Evaluating solutions

# Requirements for implementing packaged solutions



**FIGURE 22-3** A spectrum of implementation effort for packaged solutions.

**TABLE 22-1** COTS package implementation approaches

Type of COTS implementation	Description
Out-of-the-box	Install the software and use it as is.
Configured	Adjust settings in the software to suit your needs without writing new code.
Integrated	Connect the package to existing systems in your application ecosystem; usually requires some custom code.
Extended	Develop additional functionality with custom code to enhance the package's capabilities to close needs gaps.



# Requirements for implementing packaged solutions

---

- Configuration requirements
- Integration requirements
- Extension requirements
- Data requirements
- Business process changes





# Common challenges with packaged solutions

---

- Too many candidates
- Too many evaluation criteria
- Vendor misrepresents package capabilities
- Incorrect solution expectations
- Users reject the solution

---

# CHAPTER 23

## Outsourced projects

SelfStudy

- 
- What is outsourced project?
  - Appropriate levels of requirements detail
  - Acquirer-supplier interactions
  - Change management
  - Acceptance criteria

---

# **CHAPTER 24**

## **Business process automation projects**

SelfStudy

---

# CHAPTER 25

## Business analytics projects

SelfStudy

---

# **CHAPTER 26**

## **Embedded and other real-time systems projects**

SelfStudy