Agile Modeling and Prototyping

6

Systems Analysis and Design, 7e

Kendall & Kendall

Learning Objectives

- **Understand the roots of agile modeling in prototyping and the four main types of prototyping**
- **Be able to use prototyping for human information requirements gathering**
- **Understand the concept of RAD for use in human information requirements gathering and interface design**
- **Understand agile modeling and the core practices that differentiate it from other development methodologies**
- **Learn the importance of values critical to agile modeling**
- **Understand how to improve efficiency for users who are knowledge workers using either structured methods or agile modeling**

Agile Modeling, but First Prototyping

- Agile modeling is a collection of innovative, user-centered approaches to systems development
- Prototyping is an information-gathering technique useful in seeking user reactions, suggestions, innovations, and revision plans

Major Topics

- Prototyping
- Rapid application development (RAD)
- Agile Modeling

Prototyping

- Patched-up
- Nonoperational
- First-of-a-series
- Selected features

Patched-Up Prototype

- A system that works but is patched up or patched together
- A working model that has all the features but is inefficient
- Users can interact with the system
- Retrieval and storage of information may be inefficient

Nonoperational Scale Models

- A nonworking scale mode that is set up to test certain aspects of the design
- A nonworking scale model of an information system might be produced when the coding required by the application is too expensive to prototype but when a useful idea of the system can be gained through prototyping of the input and output only

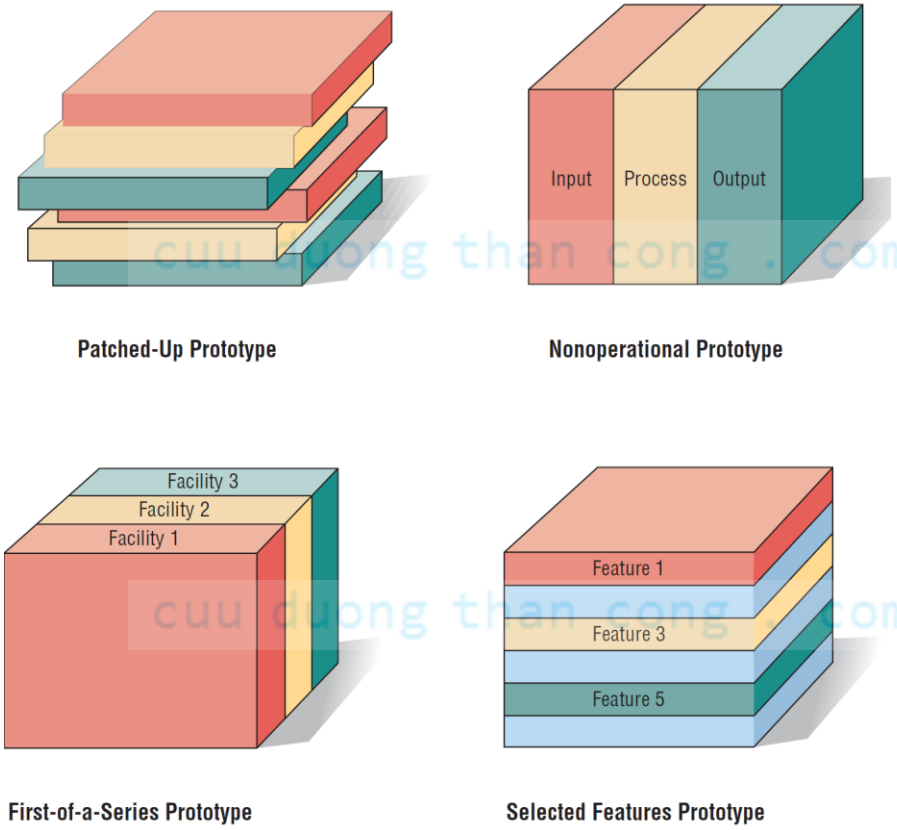
First-of-a-Series Prototype

- Creating a pilot
- Prototype is completely operational
- Useful when many installations of the same information system are planned
- A full-scale prototype is installed in one or two locations first, and if successful, duplicates are installed at all locations based on customer usage patterns and other key factors

Selected Features Prototype

- Building an operational model that includes some, but not all, of the features that the final system will have
- Some, but not all, essential features are included
- Built in modules
- Part of the actual system

Figure 6.1 Four kinds of prototypes (clockwise, starting from the upper left)



Prototyping as an Alternative to the Systems Life Cycle

- Two main problems with the SDLC
  - Extended time required to go through the development life cycle
  - User requirements change over time
- Rather than using prototyping to replace the SDLC use prototyping as a part of the SDLC

Guidelines for Developing a Prototype

- Work in manageable modules
- Build the prototype rapidly
- Modify the prototype in successive iterations
- Stress the user interface

Disadvantages of Prototyping

- It can be difficult to manage prototyping as a project in the larger systems effort
- Users and analysts may adopt a prototype as a completed system

Advantages of Prototyping

- Potential for changing the system early in its development
- Opportunity to stop development on a system that is not working
- Possibility of developing a system that more closely addresses users' needs and expectations

Prototyping Using COTS Software

- Sometimes the quickest way to prototype is through the modular installation of COTS software
- Some COTS software is elaborate and expensive, but highly useful

Users' Role in Prototyping

- Honest involvement
  - Experimenting with the prototype
  - Giving open reactions to the prototype
  - Suggesting additions to or deletions from the prototype

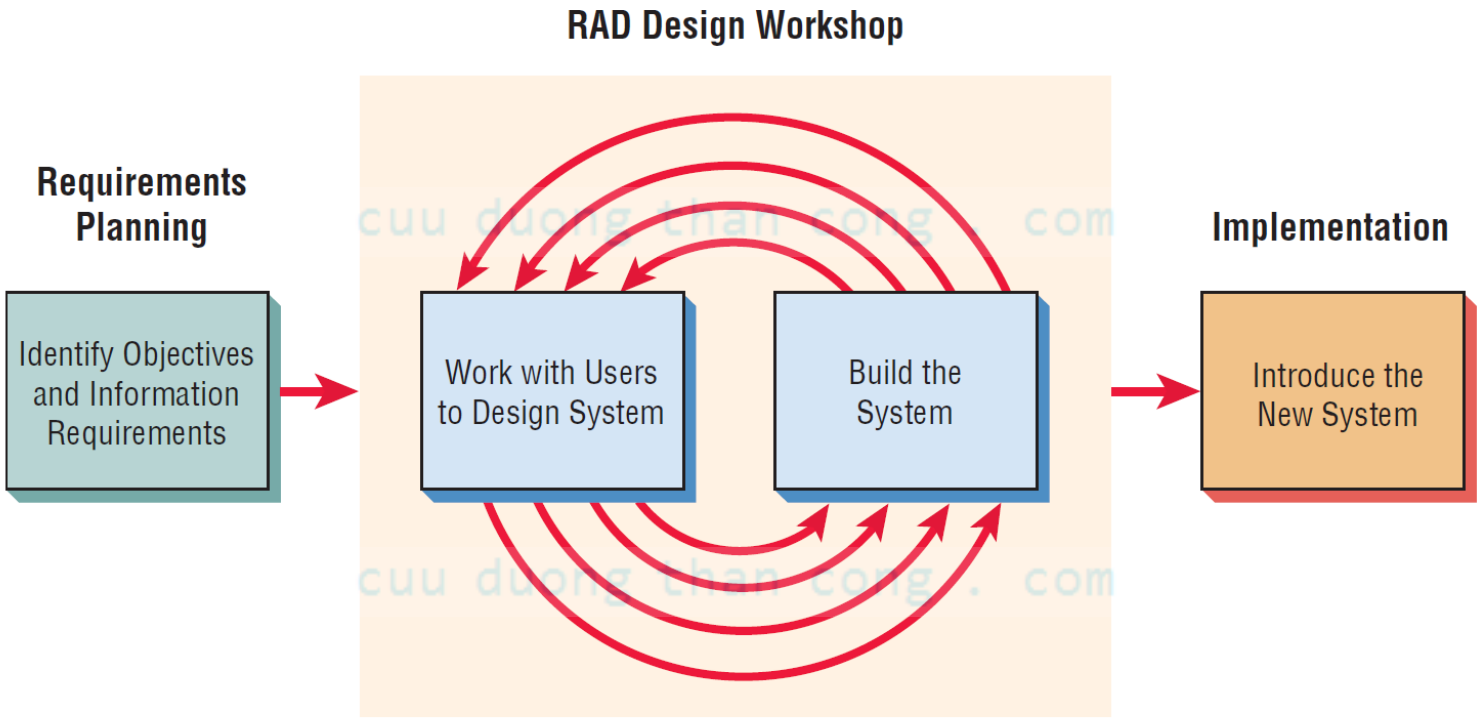
RAD (Rapid Application Development)

- An object-oriented approach to systems development that includes a method of development as well as software tools

RAD Phases

- Requirements planning
- RAD design workshop
- Implementation

Figure 6.4 The RAD design workshop is the heart of the interactive development process



Requirements Planning Phase

- Users and analysts meet to identify objectives of the application or system
- Orientation is toward solving business problems

RAD Design Workshop

- Design and refine phase
- Use group decision support systems room if available
- Users respond to actual working prototypes
- Analysts refine designed modules based on user responses

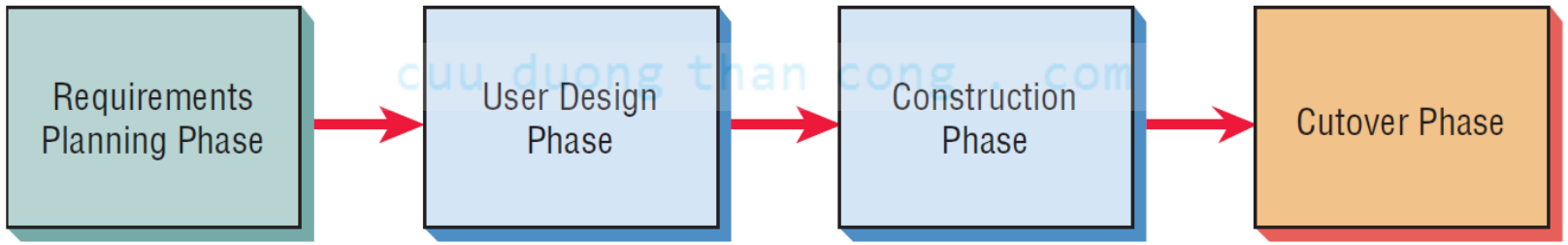
Implementation Phase

- As the systems are built and refined, the new systems or part of systems are tested and then introduced to the organization
- When creating new systems, there is no need to run old systems in parallel

Martin's Pioneering Approaches to RAD

- Requirements planning
- User design
- Construction
- Cutover

Figure 6.5 Martin's phases of RAD



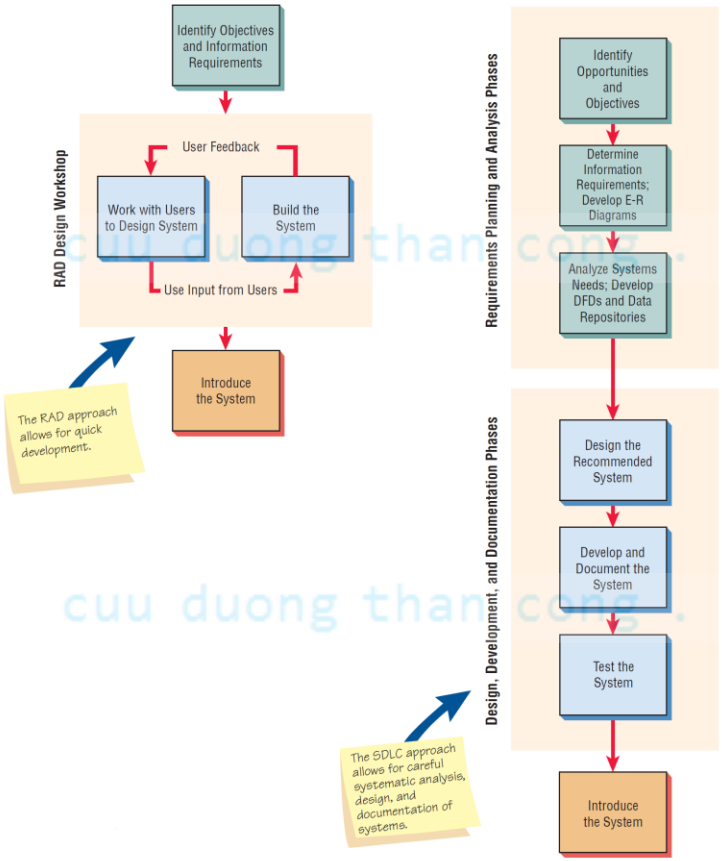
Software Tools for RAD

- Microsoft Access, Microsoft Visual Basic, Visual C++, and Microsoft .NET
- Differ from one another in their:
  - Capabilities to support client/server applications
  - Ease of use and the amount of programming skill that is required

Comparing RAD to the SDLC

- RAD software tools are used to generate screens and exhibit the overall flow of the running of the application
- RAD users are signing off on a visual model representation
- RAD implementation is less stressful because users have helped to design the business aspects of the system

Figure 6.6 The RAD design workshop and the SDLC approach compared



When to Use RAD

- The team includes programmers and analysts who are experienced with it
- There are pressing reasons for speeding up application development
- The project involves a novel ecommerce application and needs quick results
- Users are sophisticated and highly engaged with the goals of the company

Disadvantages of RAD

- Trying to hurry the project too much
- Lack of documentation

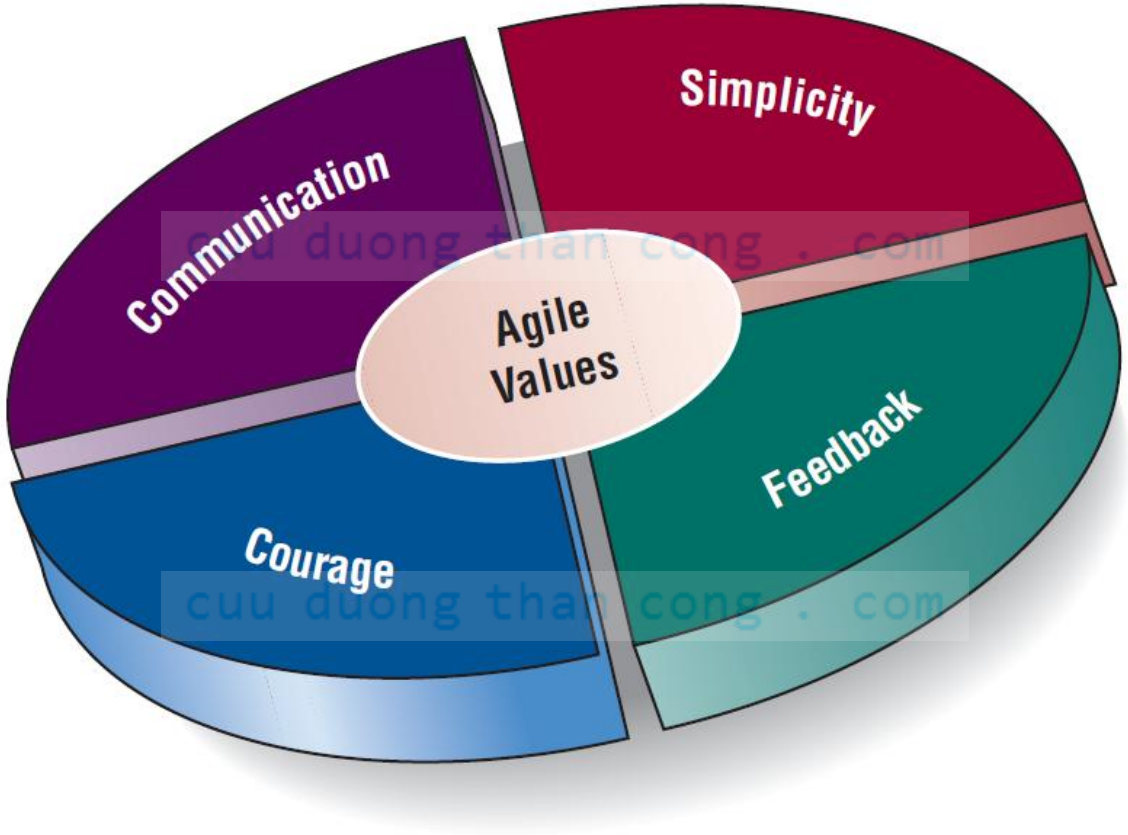
Agile Modeling

- Agile methods are a collection of innovative, user-centered approaches to systems development
- Tries to define an overall system plan quickly, develop and release software quickly, and then continuously revise the software to add additional features

Values and Principles of Agile Modeling

- Communication
- Simplicity
- Feedback
- Courage

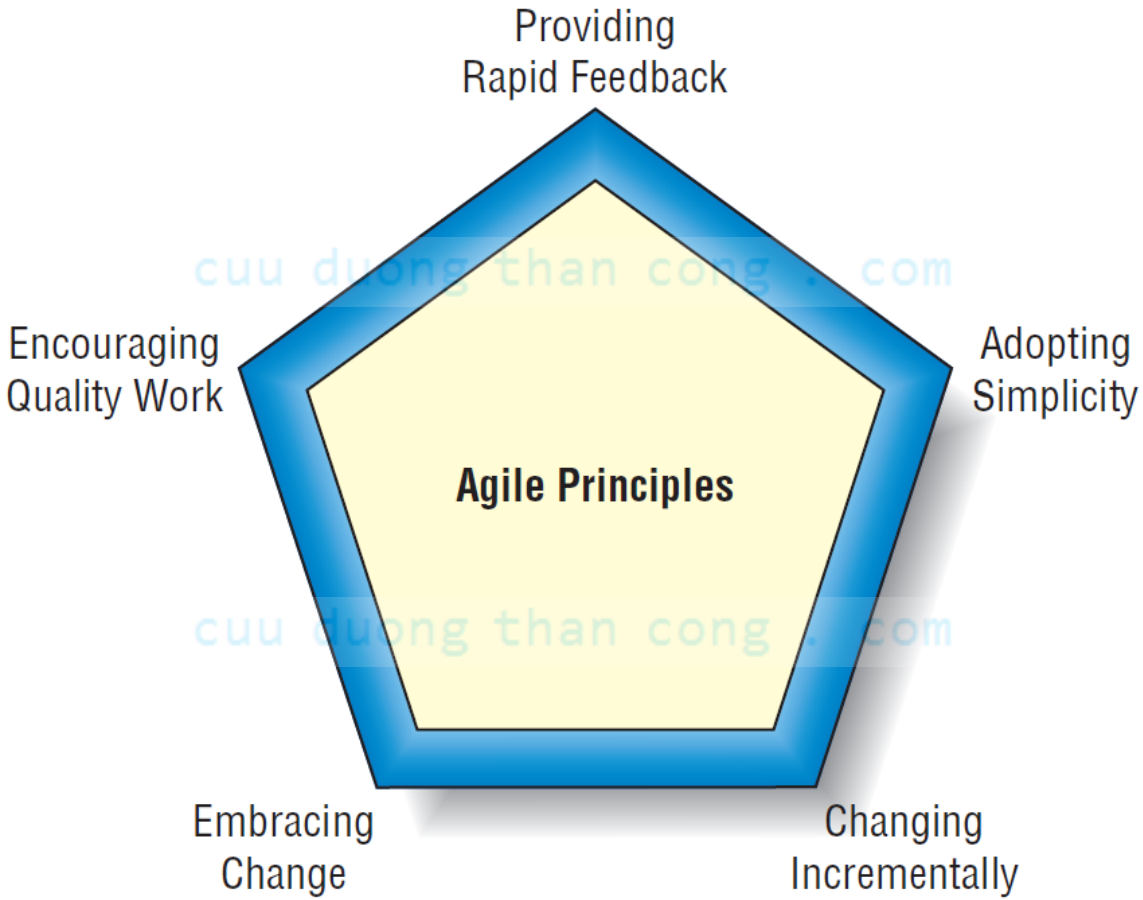
Figure 6.7 Values are crucial to the agile approach



The Basic Principles of Agile Modeling

- Providing rapid feedback
- Assuming simplicity
- Changing incrementally
- Embracing change
- Encouraging quality work

Figure 6.8 Five Agile Principles guide the systems analyst through a successful project



Activities, Resources, and Practices of Agile Modeling

- Coding
- Testing
- Listening
- Designing

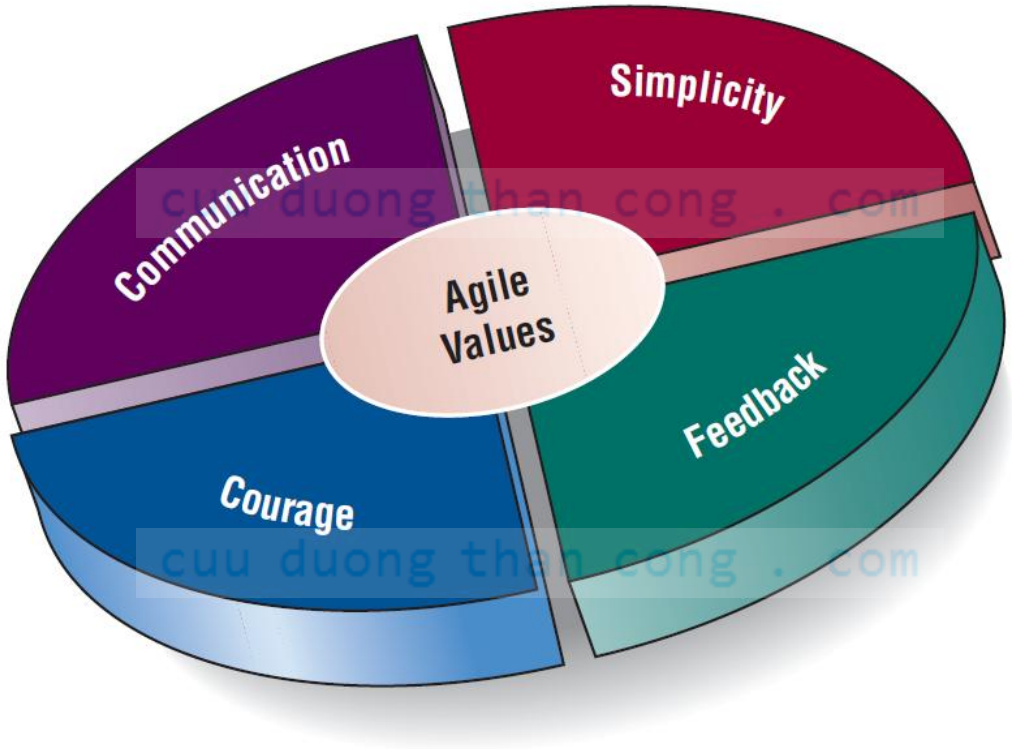
Four Resource Control Variables of Agile Modeling

- Time
- Cost
- Quality
- Scope

Four Core Agile Practices

- Short releases
- 40-hour work week
- Onsite customer
- Pair programming

Figure 6.9 The core practices are interrelated with agile modeling's resources, activities, and values



The Agile Development Process

- Exploration
- Planning
- Iterations to the first release
- Productionizing
- Maintenance

Writing User Stories

- Spoken interaction between developers and users
- Seeking first and foremost to identify valuable business user requirements
- The goal is prevention of misunderstandings or misinterpretations of user requirements

Figure 6.10 User stories can be recorded on cards. The user story should be brief enough for an analyst to determine what systems features are needed

Need or Opportunity:	Apply shortcut methods for faster checkout.					
Story:	If the identity of the customer is known and the delivery address matches, speed up the transaction by accepting the credit card on file and the rest of the customer's preferences such as shipping method.					
		Well Below	Below Average	Average	Above Average	Well Above
Activities:	Coding			✓		
	Testing			✓		
	Listening			✓		
	Designing					
Resources:	Time				✓	
	Cost				✓	
	Quality		✓			
	Scope			✓		

Development Tools for Agile Modeling

- Tools that facilitate collaboration
- Tools that support defect management
- Automated unit testers, acceptance testers, and GUI testers
- Tools for quality assurance
- Measuring system and component performance
- Source code configuration management
- Development environments

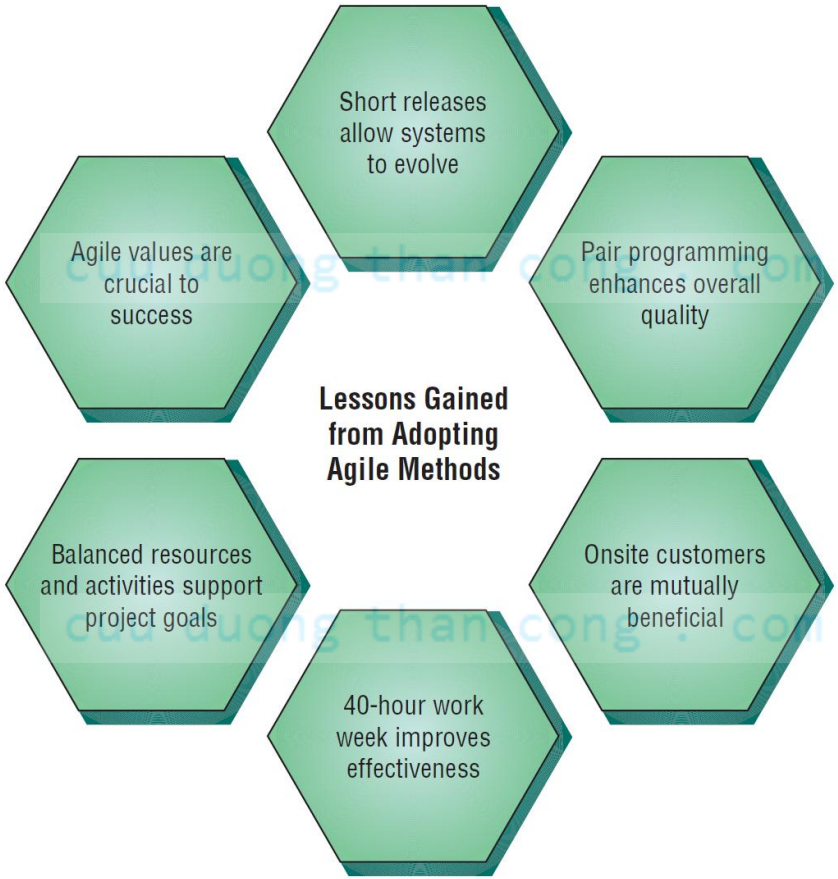
Lessons Learned from Agile Modeling

- Short releases allow the system to evolve
- Pair programming enhances overall quality
- Onsite customers are mutually beneficial to the business and the agile development team

Lessons Learned from Agile Modeling (Continued)

- The 40-hour work week improves worker effectiveness
- Balanced resources and activities support project goals
- Agile values are crucial to success

Figure 6.11 There are six vital lessons that can be drawn from the agile approach to systems



Scrum

- Product backlog
- Sprint backlog
- Sprint
- Daily scrum
- Demo

cuu duong than cong . com

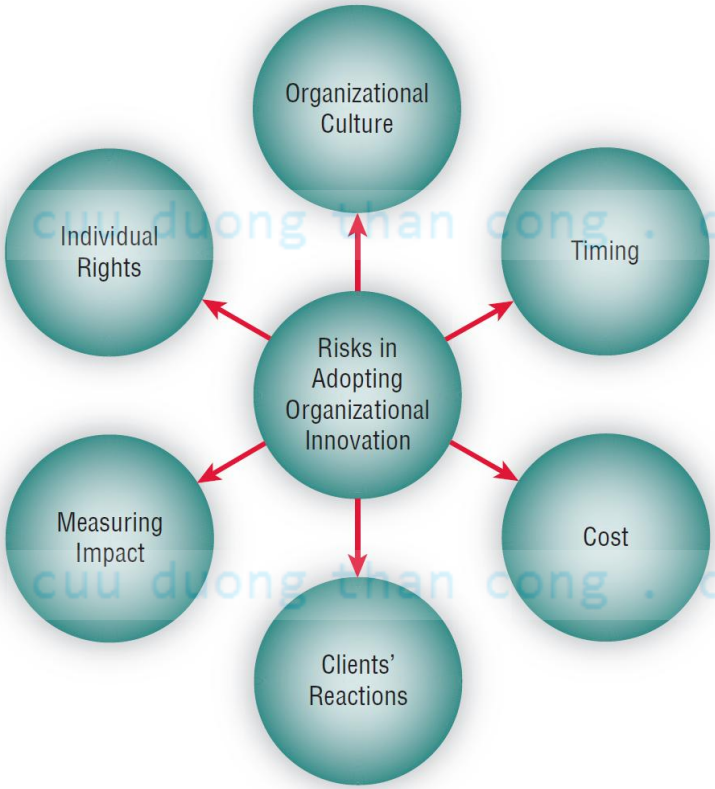
Comparing Agile Modeling and Structured Methods

- Improving the efficiency of systems development
- Risks inherent in organizational innovation

Figure 6.12 How Davis and Naumann's (1999) strategies for improving efficiency can be implemented using two different development approaches

Strategies for Improving Efficiency in Knowledge Work	Implementation Using Structured Methodologies	Implementation Using Agile Methodologies
Reduce interface time and errors	Adopting organizational standards for coding, naming, etc.; using forms	Adopting pair programming
Reduce process learning time and dual processing losses	Managing when updates are released so the user does not have to learn and use software at the same time	Ad hoc prototyping and rapid development
Reduce time and effort to structure tasks and format outputs	Using CASE tools and diagrams; using code written by other programmers	Encouraging short releases
Reduce nonproductive expansion of work	Project management; establishing deadlines	Limiting scope in each release
Reduce data and knowledge search and storage time and costs	Using structured data gathering techniques, such as interviews, observation, sampling	Allowing for an onsite customer
Reduce communication and coordination time and costs	Separating projects into smaller tasks; establishing barriers	Timeboxing
Reduce losses from human information overload	Applying filtering techniques to shield analysts and programmers	Sticking to a 40-hour work week

Figure 6.13 Adopting new information systems involves balancing several risks



Summary

- Prototyping
  - Patched-up system
  - Nonoperational
  - First-of-a-series
  - Selected-features
- Prototype development guidelines
- Prototype disadvantages

Summary (Continued)

- Prototype advantages
- Users' role in prototyping
- Agile modeling
- Five values of the agile approach
- Principles of agile development
- Agile activities
- Agile resources

Summary (Continued)

- Core practices of the agile approach
- Stages in the agile development process
- User stories
- Agile lessons
- Scrum methodology
- Dangers to adopting innovative approaches