back to chapter 1

Systems, Roles, and Development Methodologies

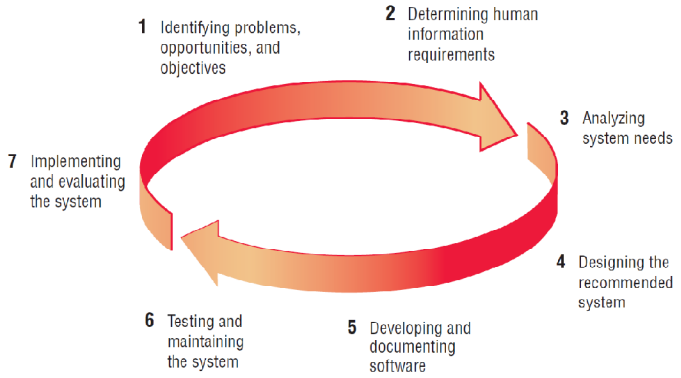
Systems Analysis and Design, 8e

Kendall & Kendall

Systems Development Life Cycle (SDLC)

- The systems development life cycle is a phased approach to solving business problems.
- Developed through the use of a specific cycle of analyst and user activities
- Each phase has unique user activities.

The Seven Phases of the Systems Development Life Cycle (Figure 1.3)



Identifying Problems, Opportunities, and Objectives

- Activity:
  - Interviewing user management
  - Summarizing the knowledge obtained
  - Estimating the scope of the project
  - Documenting the results
- Output:
  - Feasibility report containing problem definition and objective summaries from which management can make a decision on whether to proceed with the proposed project

Determining Human Information Requirements

- Activity:
  - Interviewing
  - Sampling and investing hard data
  - Questionnaires
  - Observe the decision maker's behavior and environment.
  - Prototyping
  - Learn the who, what, where, when, how, and why of the current system.
- Output:
  - The analyst understands how users accomplish their work when interacting with a computer; and begin to know how to make the new system more useful and usable. The analyst should also know the business functions and have complete information on the people, goals, data, and procedure involved.

Analyzing System Needs

- Activity:
  - Create data flow, activity, or sequence diagrams.
  - Complete the data dictionary.
  - Analyze the structured decisions made.
  - Prepare and present the system proposal.
- Output:
  - Recommendation on what, if anything, should be done

Designing the Recommended System

- Activity:
  - Design procedures for data entry.
  - Design the human-computer interface.
  - Design system controls.
  - Design database and/or files.
  - Design backup procedures.
- Output
  - Model of the actual system

Developing and Documenting Software

- Activity:
  - System analyst works with programmers to develop any original software.
  - Works with users to develop effective documentation.
  - Programmers design, code, and remove syntactical errors from computer programs.
  - Document software with help files, procedure manuals, and Web sites with Frequently Asked Questions.
- Output:
  - Computer programs
  - System documentation

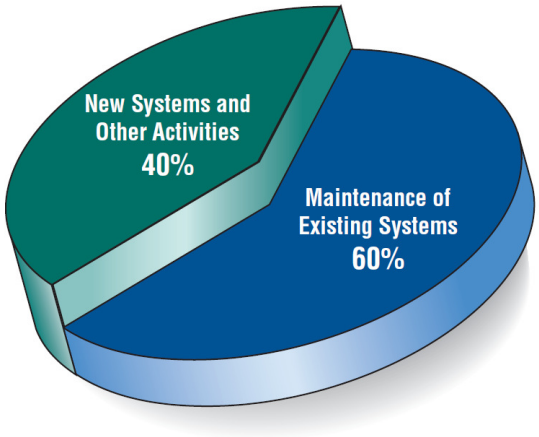
Testing and Maintaining the System

- Activity:
  - Test the information system.
  - System maintenance.
  - Maintenance documentation.
- Output:
  - Problems, if any
  - Updated programs
  - Documentation

Implementing and Evaluating the System

- Activity:
  - Train users.
  - Analyst plans smooth conversion from old system to new system.
  - Review and evaluate system.
- Output:
  - Trained personnel
  - Installed system

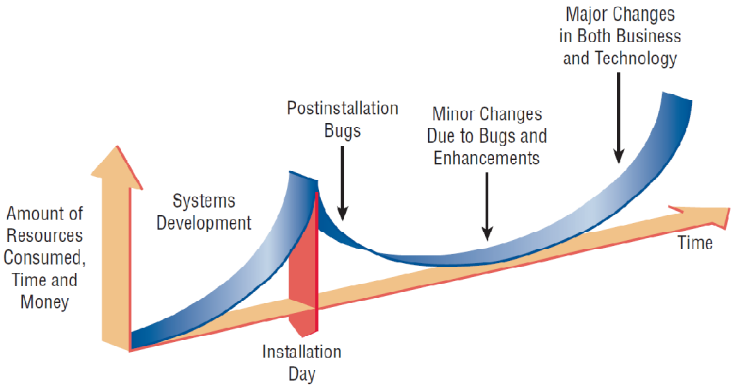
Some Researchers Estimate that the Amount of Time Spent on Systems Maintenance May Be as Much as 60 Percent of the Total Time Spent on Systems Projects (Figure 1.4)



The Impact of Maintenance

- Maintenance is performed for two reasons:
  - Removing software errors
  - Enhancing existing software
- Over time the cost of continued maintenance will be greater than that of creating an entirely new system. At that point it becomes more feasible to perform a new systems study.

Resource Consumption over the System Life (Figure 1.5)



Approaches to Structured Analysis and Design and to the Systems Development Life Cycle

- Traditional systems development life cycle
- CASE (Computer Aided Software Engineering) systems development life cycle
  - Analysts rely on CASE tools to increase productivity, communicate more effectively with users, and integrate the work
- Object-oriented systems analysis and design

Choosing a Method

- Choose either:
  - SDLC
  - Agile
  - Object-oriented methodologies

When to Use SDLC

- Systems have been developed and documented using SLDC.
- It is important to document each step.
- Upper level management feels more comfortable or safe using SDLC.
- There are adequate resources and time to complete the full SDLC.
- Communication of how new systems work is important.

When to Use Agile

- There is a project champion of agile methods in the organization.
- Applications need to be developed quickly in response to a dynamic environment.
- A rescue takes place (the system failed and there is no time to figure out what went wrong).
- The customer is satisfied with incremental improvements.
- Executives and analysts agree with the principles of agile methodologies.

When to Use Object-Oriented

- The problems modeled lend themselves to classes.
- An organization supports the UML learning.
- Systems can be added gradually, one subsystem at a time.
- Reuse of previously written software is a possibility.
- It is acceptable to tackle the difficult problems first.

6

Agile Modeling and Prototyping

Systems Analysis and Design, 8e

Kendall & Kendall

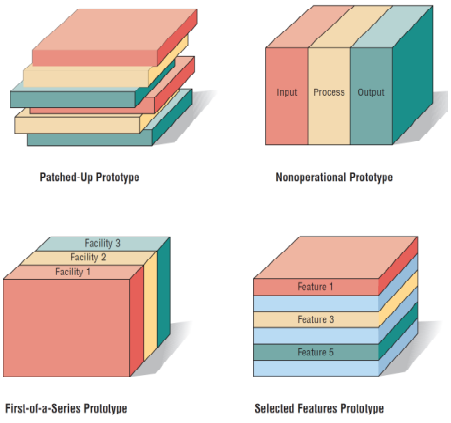
Agile Modeling, but First Prototyping

- 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.

Prototyping

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

Four Kinds of Prototypes Clockwise, Starting from the Upper Left (Figure 6.1)



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

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

Users' Role in Prototyping

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

Rapid Application Development (RAD)

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

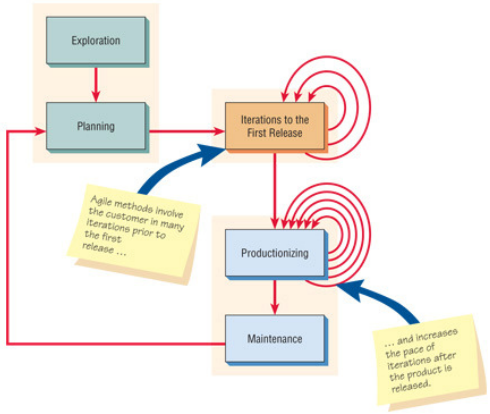
Agile Modeling

- Agile methods are a collection of innovative, user-centered approaches to systems development.

Five Stages of Agile Development

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

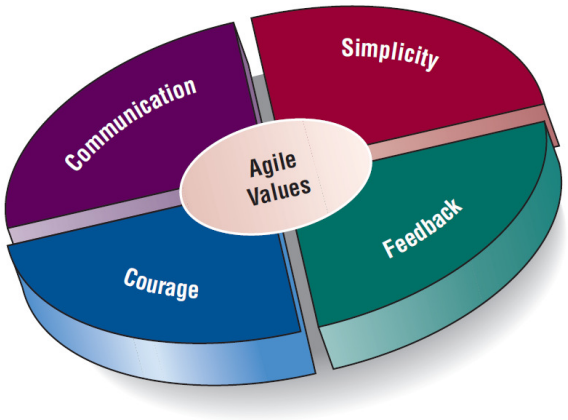
Agile Project Development Process (Figure 1.7)



Values and Principles of Agile Modeling

- Communication
- Simplicity
- Feedback
- Courage

Values Are Crucial to the Agile Approach (Figure 6.6)



The Basic Principles of Agile Modeling

- Satisfy the customer through delivery of working software.
- Embrace change, even if introduced late in development.
- Continue to deliver functioning software incrementally and frequently.
- Encourage customers and analysts to work together daily.
- Trust motivated individuals to get the job done.

The Basic Principles of Agile Modeling (continued)

- Promote face-to-face conversation.
- Concentrate on getting software to work.
- Encourage continuous, regular, and sustainable development.
- Adopt agility with attention to mindful design.
- Support self-organizing teams.

The Basic Principles of Agile Modeling (continued)

- Provide rapid feedback.
- Encourage quality.
- Review and adjust behavior occasionally.
- Adopt simplicity.

Activities, Resources, and Practices of Agile Modeling

- Coding
- Testing
- Listening
- Designing

Four Resource Control Variables of Agile Modeling

- Time
- Cost
- Quality
- Scope

The Agile Development Process

- Listen for user stories.
- Draw a logical workflow model.
- Create new user stories based on the logical model.
- Develop some display prototypes.
- Create a physical data model using feedback from the prototypes and logical workflow diagrams.

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.

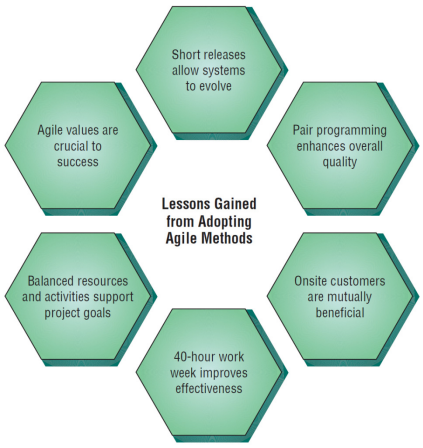
Lessons Learned from Agile Modeling

- Short releases allow the system to evolve.
- Pair programming enhances the 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.

There Are Six Vital Lessons that Can Be Drawn from the Agile Approach to Systems (Figure 6.9)



Comparing Agile Modeling and Structured Methods

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

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

