CH1. The Systems Development Environment

- 定義資訊系統分析和設計
- 描述資 information systems development life cycle（SDLC）
  - Planning
  - Analysis
  - Design
  - Implementation
  - Maintenance
- 說明 Rapid Application Development (RAD)和 computer-aided software engineering (CASE) tools
- 說明 Agile Methodologies 和 eXtreme Programming
- 說明 object-oriented analysis 以及 Rational Unified Process（RUP）

1. Information Systems Analysis and Design

- Complex organizational process
- Used to develop and maintain computer-based information systems
- Used by a team of business and systems professionals

2. **Application Software**

   Computer software designed to support organizational functions or processes

3. **An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools**

   - **Methodologies**
     Comprehensive, multiple-step approaches to systems development that will guide your work and influence the quality of your final product – the IS
     (Most methodologies incorporate several development techniques)

   - **Techniques**
     Particular processes that analysts will follow to help ensure that their work is well thought out, complete and comprehensible to others on their project teams

     e.g., conducting interviews, planning and managing the SD activities, diagramming the system's logic, designing the reports

   - **Tools**

Typically computer programs that make it easy to use and benefit from techniques and to faithfully follow the guidelines of the overall development methodology

4. **Systems Analyst**

Organizational role most responsible for analysis and design of information systems

The primary role

- To study the problems and needs of an organization in order to determine how people, methods, and IT can best be combined to bring about improvements in the organization
- To help define users requirements for new or enhanced information services
- An agent of change and innovation

1950s:
efficiency of processing, efficient automation of existing processes be developed from scratch in machine language or assembly language

1960s:
third generation languages (3GL)(C, C++, Java), minicomputers, software industry,
from scratch using their in-house development staffs

1970s:
Database management systems (hierarchical and network models), the storage and retrieval of data, from processes first to data first

1980s:

Microcomputers, off-the-shelf software(現成的軟體), 4GL, CASE

tools, window- or icon-based interfaces, object-oriented methods, developed less software in-house and bought relatively more from software vendor, from builders to integrators

1990s:
system integration, visual programming environments (VB or PB), GUI applications, client/server platforms, relational and OO database, Internet (Web),
purchase its entire enterprise-wide system from external vendors

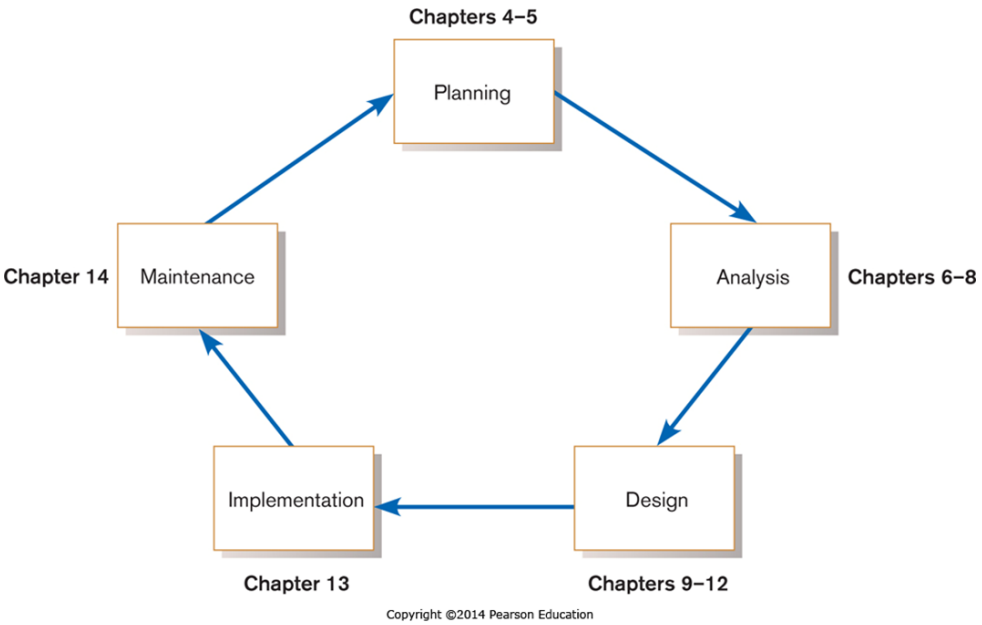
The new century:
Web application development, N-tier design, wireless PDAs and smart phones, component-based applications, application service providers (ASP), cloud computing

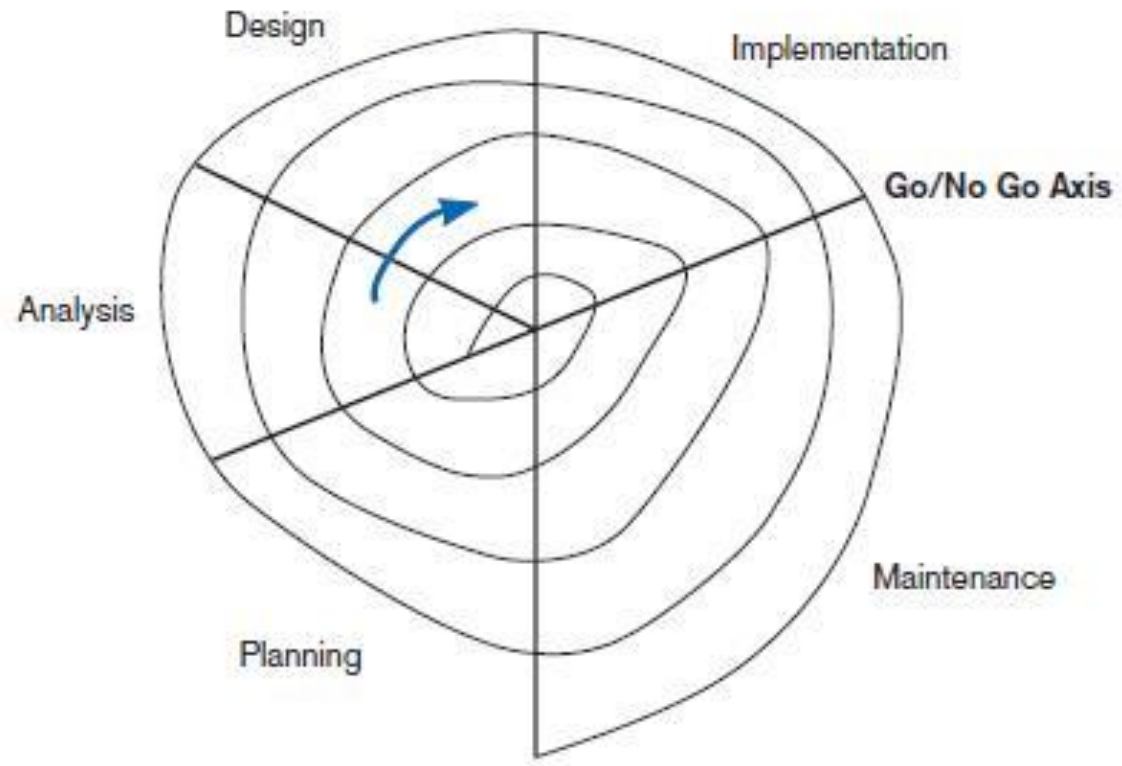
5. **Developing Information Systems**
   System Development Methodology is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

6. **Systems Development Life Cycle (SDLC)**
   Traditional methodology used to develop, maintain, and replace information systems

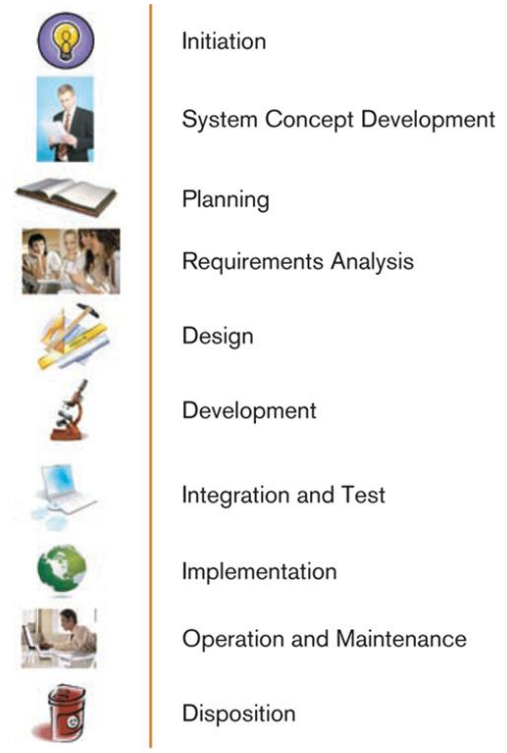
   - **Planning**
   - **Analysis**
   - **Design**
   - **Implementation**
   - **Maintenance**

   Every textbook author and IS development organization uses a slightly different life-cycle model, with anywhere from 3 to almost 20 identifiable phases





Evolutionary model(上圖)



(Source: Diagram based on www.usdoj.gov/jmd/irm/lifecycle/ch1.htm#para1.2.)

U.S. DOJ's SDLC(左圖)

Planning

an organization's total IS needs are identified, analyzed, prioritized, and arranged

- Strategic planning, strategic IS planning, or ad-hoc(特別的) user requests
  - ⇨ Problems, tasks, or opportunities

- Two additional activities: the formal investigation and presentation of reasons

  - ⇨ To determine the scope of the proposed system, to provide a formal definition and specific plan for the proposed project, and to give the final presentation of the business case for proceeding with the subsequent project phases

Analysis

system requirements are studied and structured

- **Two subphases**

  - ⇨ Requirements determination (what do the users want from a proposed system?), carefully study of any current systems, manual or computerized, and procedures

  - ⇨ To study the requirements and structure them according to their inter-relationships, and eliminate any redundancies
- **The output:** a description of (but not a detailed design for) the alternative solution recommended by the analysis team

Design

a description of the recommended alternative solution is converted into logical and then physical system specifications

- **The analysts must**

  - ⇨ Design all aspects of the system, from input and output screens to reports, databases, and computer processes, then
    Provide the physical specifics of the system they have designed, either as a model or as detailed documentation to guide those who will build the new system

- **The output**: the physical system specifications in a form ready to be turn over to programmers and other system builders for construction

- **Logical design** – all functional features of the system chosen for development in analysis are described independently of any computer platform (hardware and system software)

  - ⇨ Concentrates on business aspects of the system and tends to be oriented to a high level of specificity

- **Physical design** – the logical specifications of the system from logical design are transformed into the technology-specific details from

which all programming and system construction can be accomplished

⇨ Determines programming language, database, and hardware platform

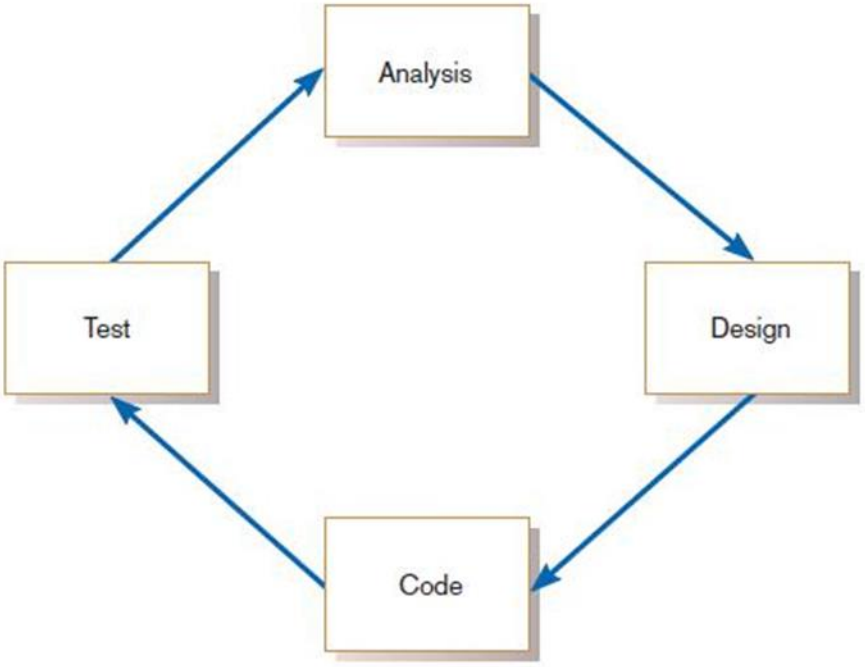
⇨ Designs the various parts of the system to perform the physical operations necessary to facilitate date capture, processing, and information output (working model, prototype, detailed specifications)

Implementation – the information system is coded, tested, installed and supported (documentation, training, and ongoing user assistance) in the organization

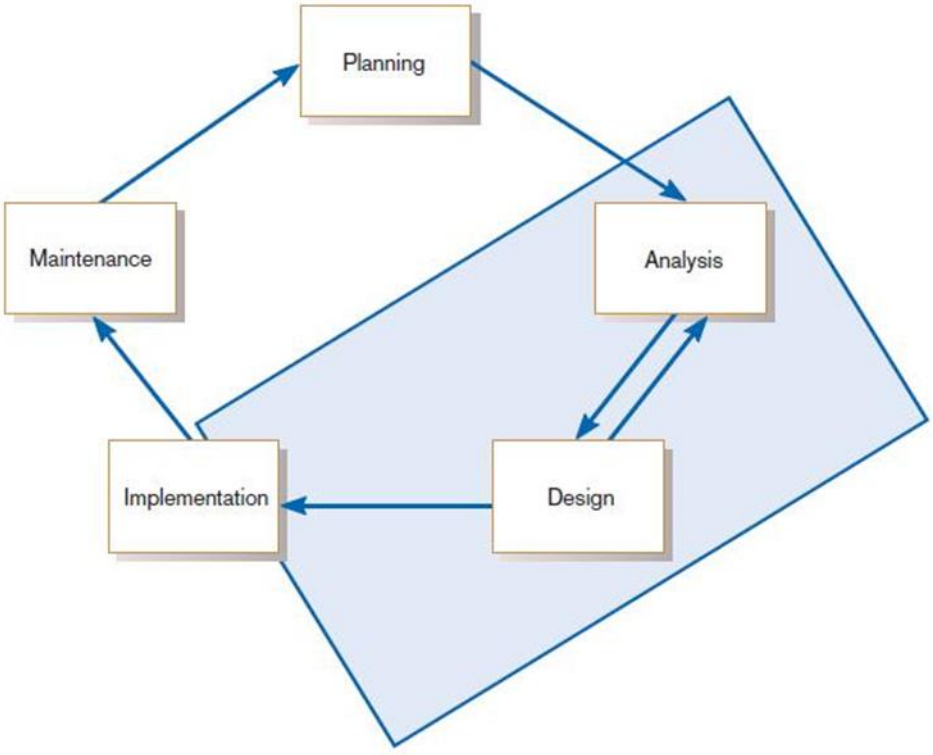
Maintenance – an IS is systematically repaired and improved

TABLE 1-1 Products of SDLC Phases

Phase	Products, Outputs, or Deliverables
Planning	Priorities for systems 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 are 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 all 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

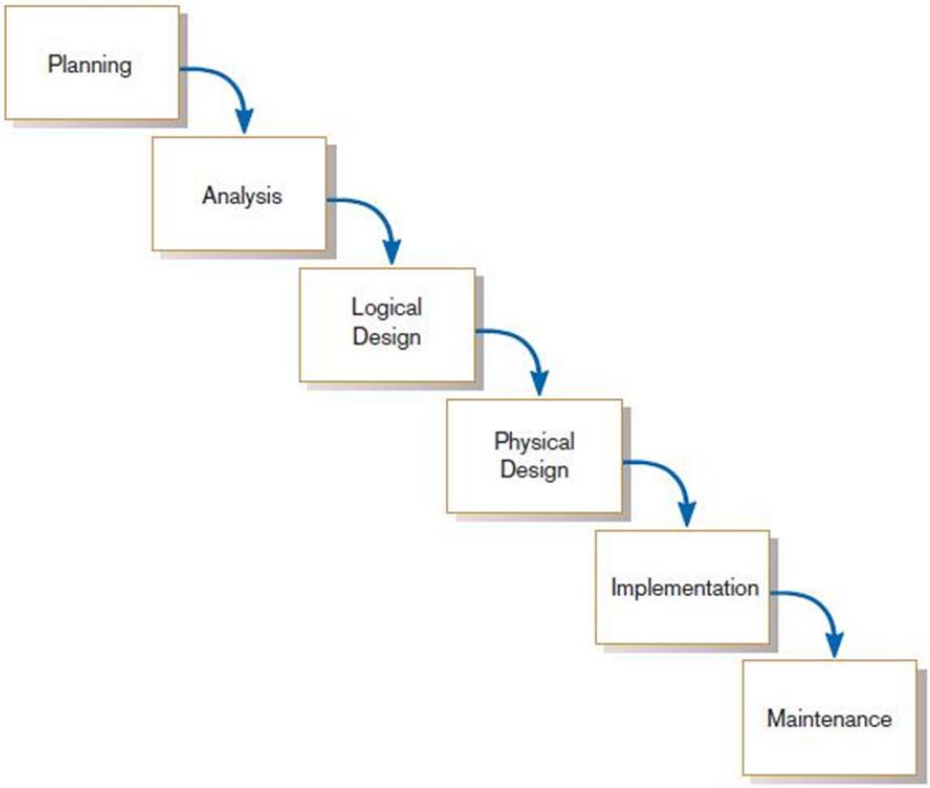


Analysis–design–code–test loop(上圖)



The heart of systems development(上圖)

➢ **Traditional Waterfall SDLC**

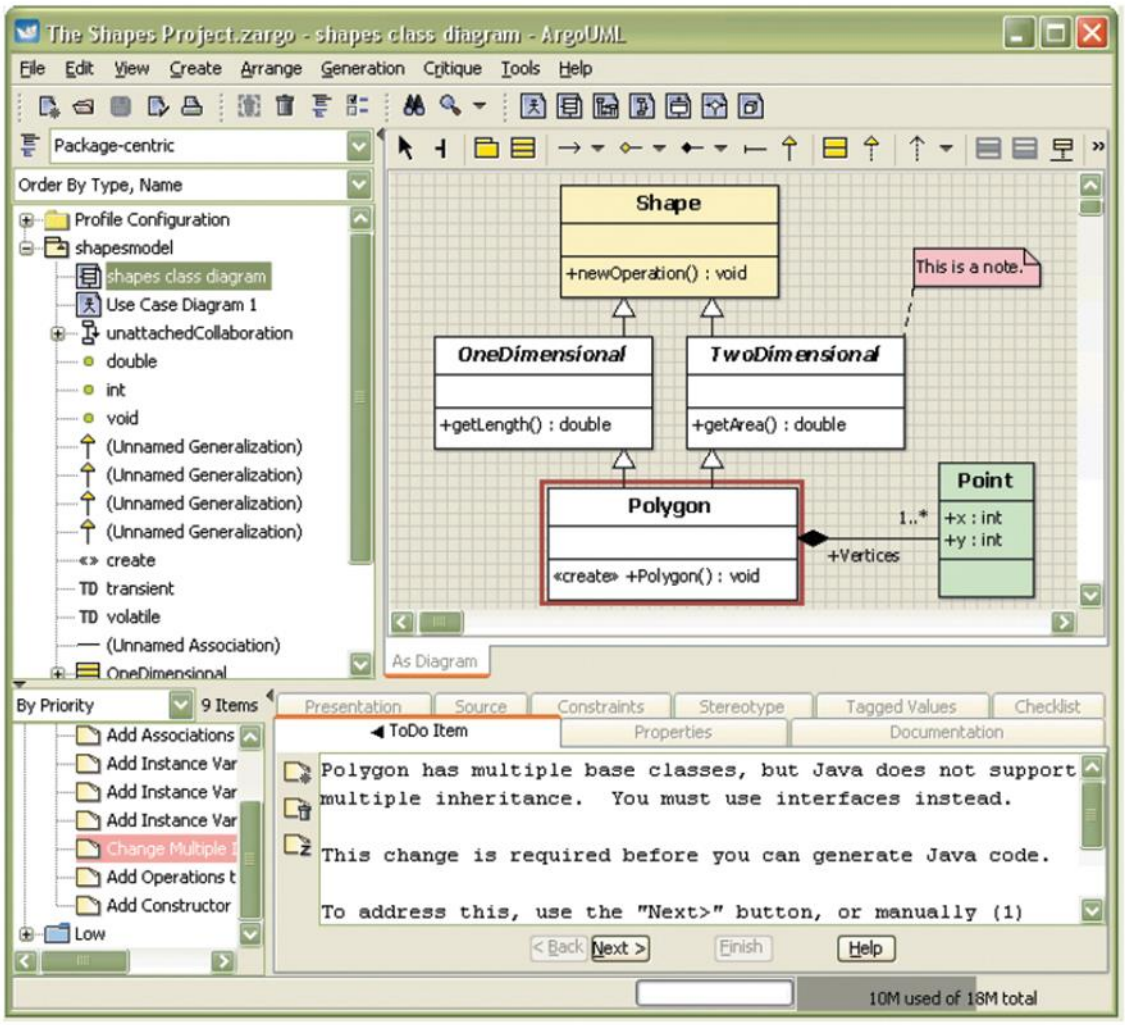


Problems:

- ⇨ Feedback ignored, milestones lock in
- ⇨ Limited user involvement (only in requirements phase)
- ⇨ Too much focus on milestone deadlines of the nebulous and intangible SDLC phases to the detriment of sound development practices

7. Different Approaches to Improving Development

- ⇨ **CASE(Computer-Aided Software Engineering) Tools**



1. A central repository provides integrated storage of specifications, diagrams, reports, and project management information.
2. Documentation generators
3. Code generators
4. Diagramming tools
5. Computer displays and report generators help prototype how systems “look and feel”
6. Analysis tools automatically check for completeness, consistency, and correctness in diagrams, forms, and reports.

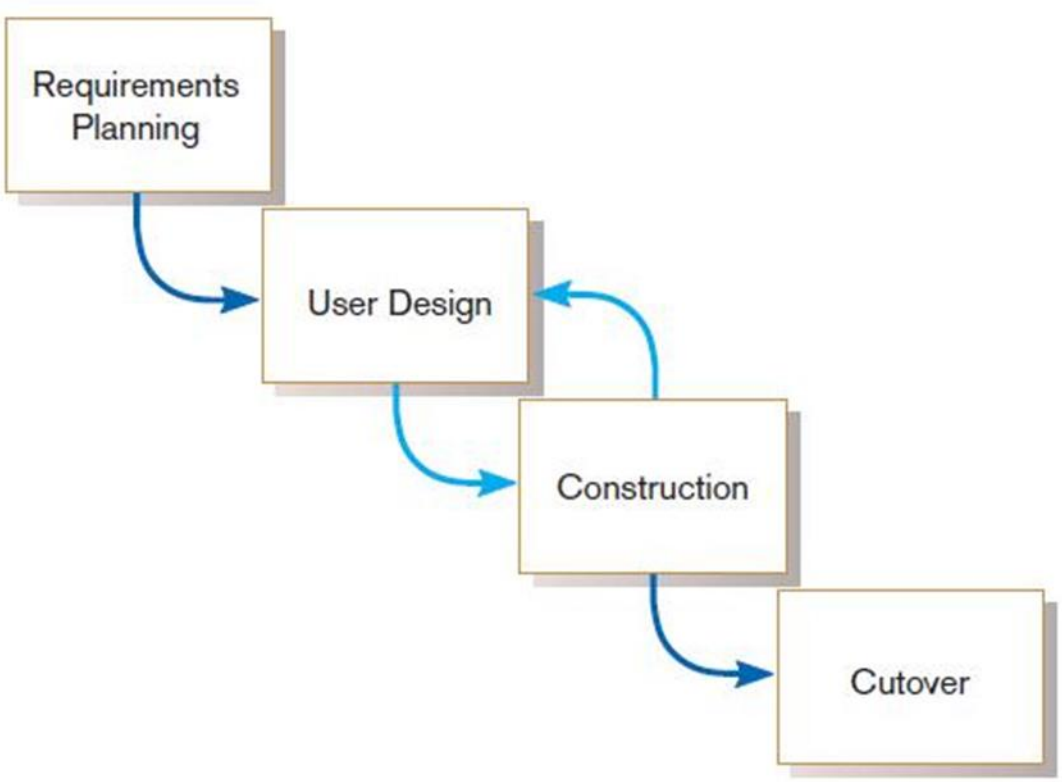
TABLE 1-2 Examples of CASE Usage within the SDLC

SDLC Phase	Key Activities	CASE Tool Usage
Project identification and selection	Display and structure high-level organizational information	Diagramming and matrix tools to create and structure information
Project initiation and planning	Develop project scope and feasibility	Repository and documentation generators to develop project plans
Analysis	Determine and structure system requirements	Diagramming to create process, logic, and data models
Logical and physical design	Create new system designs	Form and report generators to prototype designs; analysis and documentation generators to define specifications
Implementation	Translate designs into an information system	Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation
Maintenance	Evolve information system	All tools are used (repeat life cycle)

⇨ **Rapid Application Development (RAD)**

1. Decreases design and implementation time - Extensive user involvement, prototyping, integrated CASE tools, code generators
2. Planning and design phases are shortened

- More focus on user interface requirements, system function more on doing different tasks in parallel with each other and on using prototype extensively
- less on detailed business analysis and system performance issues, Less emphasis on the sequence and structure of processes in the life cycle



RAD life cycle(上圖)

⇨ **Agile Methodologies**

Four key principle of “The Agile Manifesto” :

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

When to use Agile Methodologies:

- Responding to change over following a plan
- Responsible and motivated developers
- Customers who understand the process and will get involved

TABLE 1-4 Five Critical Factors That Distinguish Agile and Traditional Approaches to Systems Development

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 projects.
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 no-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).

(*Source:* Boehm, Barry; Turner, Richard, Balancing Agility and Discipline: A Guide for the Perplexed, 1st Ed., (c)2004. Reprinted and electronically reproduced by permission of Pearson Education, Inc. Upper Saddle River, New Jersey.

⇨ **Agile Methodology Family**

- The Crystal family of methodologies
- Adaptive Software Development
- Scrum
- Feature Driven Development
- **eXtreme Programming(Beck & Andre, 2004)**
  1. Short cycles
  2. Incremental planning approach(漸進式開發方法)
  3. Automated tests written by customers and programmers
  4. A reliance on an evolutionary approach to development
  5. Two-person programming teams and a customer on-site
  6. Coding, testing, listening, designing

Advantage:

1. More and better communication among developers
2. High level of productivity
3. High-quality code
4. Reinforcement of the other practices in eXtreme programming, such as the code-and-test discipline

⇨ **Object-oriented analysis and design**

Based on objects rather than data or processes

- The 3rd approach to system development
- Reusable system elements and improved system quality and SA&D productivity

Object-Oriented Analysis (OOA):

- To identify objects
- To define their structure and behavior and their relationship

Object-Oriented Design (OOD):

- To model the details of the objects' behavior and communication with other objects so that system requirements are met
- To reexamine and redefine objects to better take advantage of inheritance and other benefits of OO

➢ **Rational Unified Process (RUP)**

- An object-oriented systems development methodology based on an iterative, incremental approach
- Establishes four phase of development:

inception(起始): analysts define the scope, determine the feasibility of the project, understand user requirements, and prepare a software development plan.

elaboration(細述): analysts detail user requirements and develop a baseline architecture.

construction(建構): the software is actually coded, tested, and

documented

transition(轉換): the system is deployed, and the users are trained and supported.

⇨ Each phase is organized into a number of separate iterations