Chapter 1: Introduction to Systems Analysis and Design

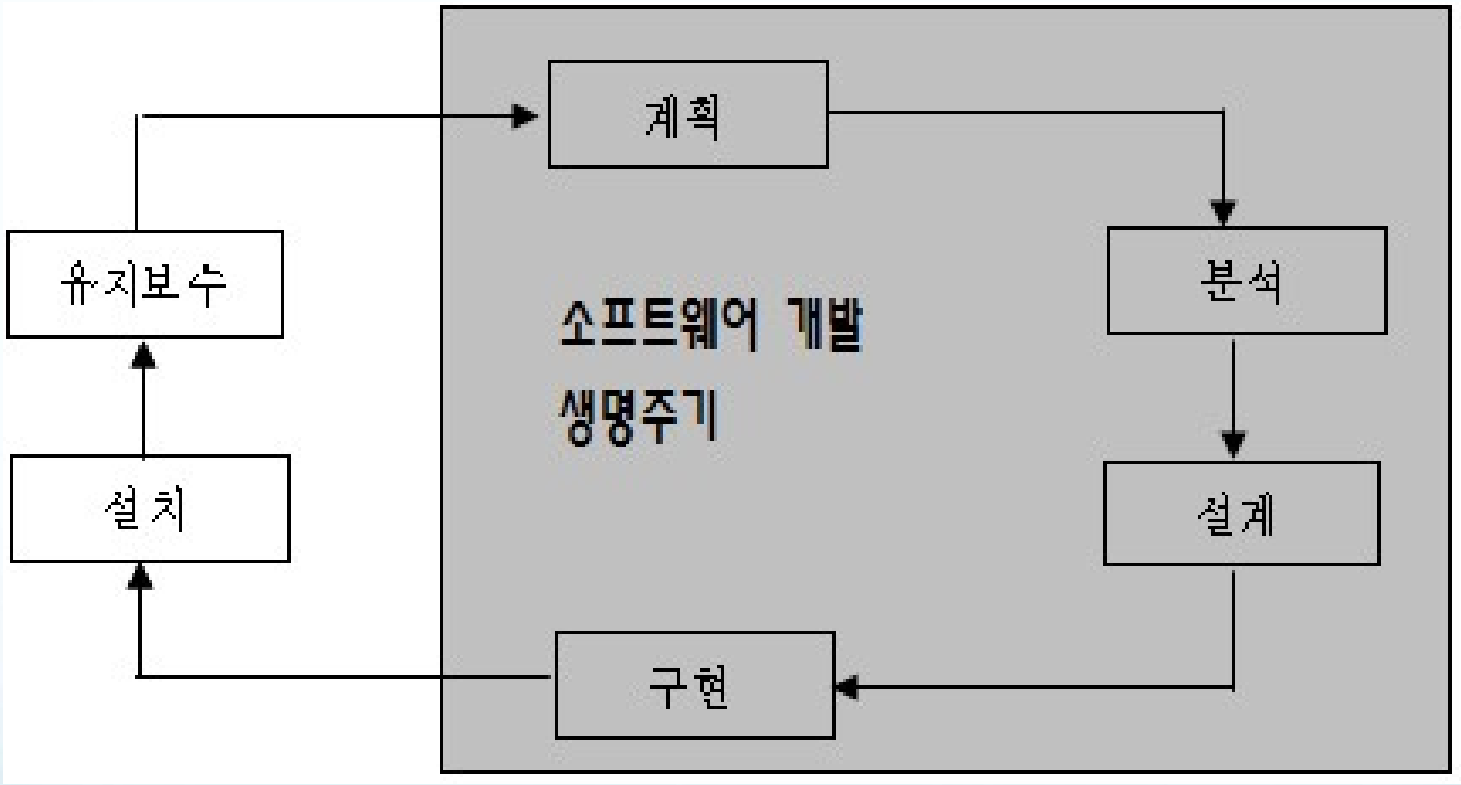
Learning Objectives

- Systems development life cycle(SDLC)
  - Identify the four phases
  - How it came about
  - Methodology alternatives
- Team roles & skill sets
- Object-oriented systems characteristics
- Object-oriented systems analysis & design
- The Unified Process & its extensions
- The Unified Modeling Language (UML)

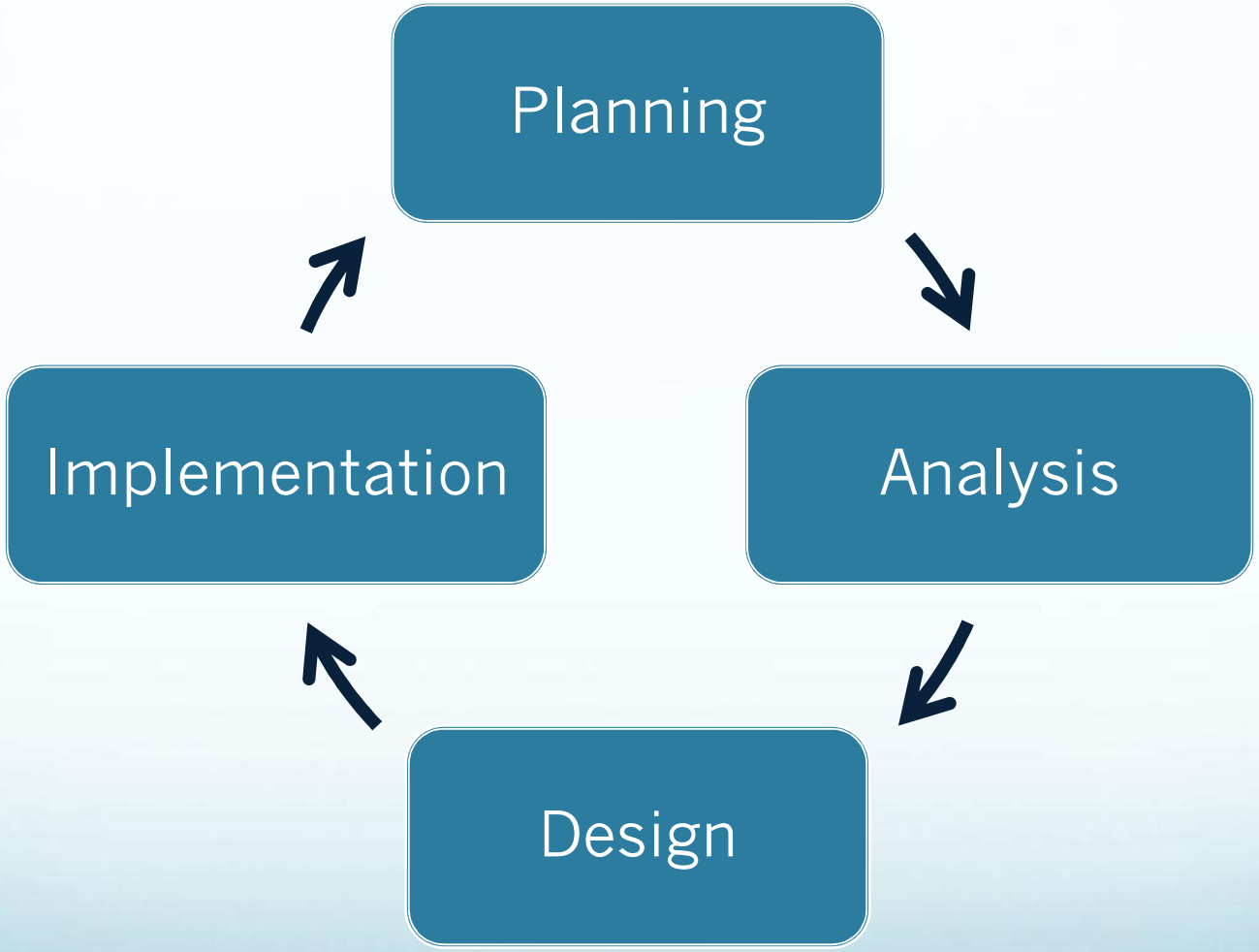
Introduction

- Why do we need a formal process?
  - Failures occur (too) often
  - Creating systems is not intuitive
  - Projects are late, over budget or delivered with fewer features than planned
- The System Analyst is the key person
  - Designs a system to add value
  - Must understand the business processes
  - Job is rewarding, yet challenging
  - Requires specific skill sets

시스템 개발 과정



Systems Development Life Cycle (SDLC)



The SDLC Process

- The process consists of four phases
- Each phase consists of a series of steps
- Each phase is documented (deliverables)
- Phases are executed sequentially, incrementally, iteratively or in some other pattern

Questions to be Answered

- Planning phase
  - Why should we build this system?
  - What value does it provide?
  - How long will it take to build?
- Analysis phase
  - Who will use it?
  - What should the system do for us?
  - Where & when will it be used?
- Design phase
  - How should we build it?

SDLC: The Planning Phase

1. Project Initiation
   - Develop/receive a system request
   - Conduct a feasibility analysis
   - 왜만드나?/ 타당한가?
2. Project Management
   - Develop the work plan
   - Staff the project
   - Monitor & control the project
   - 프로젝트(투입시간/투입인력/투입자원)

SDLC: The Analysis Phase
무엇을 만들 것인가?

1. Develop an analysis strategy
   - Model the current system (현재에서_)
   - Formulate the new system(미래로)
2. Gather the requirements
   - Develop a system concept
   - Create a business model to represent:
     - Business data
     - Business processes
     - 병원업무 : 병원 자료와 병원 프로세스 모델
3. Develop a system proposal(이런 것 만들 래)

SDLC: The Design Phase

1. Develop a design strategy
2. Design architecture and interfaces
3. Develop databases and file specifications
4. Develop the program design to specify:
   - What programs to write
   - What each program will do

- Design Specification 이렇게 만들래

SDLC: The Implementation Phase

1. Construct the system
   - Build it (write the programming code)
   - Test it
2. Install system
   - Train the users
3. Support the system (maintenance)

SDLC: Methodologies

- Methodology: a formalized approach to implementing the SDLC
- Categories
  - Process oriented - Structured
  - Data centered -
  - Object-oriented –
  - Rapid action development
  - Agile development

개발방법론

	Process oriented 구조적 방법론	Data oriented 정보공학 방법론	Object Oriented 객체지향 방법론
계획 단계	타당성 분석	정보 전략 분석	프로젝트 문제 분석
분석 단계	구조적 분석(요구 분석 및 모델링) (DFD,미니스팩,자료 사전)	비즈니스 영역 분석 (E-R diagram 등)	객체지향 분석(요구 분석 및 모델링 (사용사례, 사용 스토리 등)
설계 단계	구조적 설계	비즈니스 시스템 설계	객체지향설계 시스템 설계와 객체 설계
구현 단계	구조적 프로그래밍 c,pascal,fortran, 등	구축 전환	객체지향 프로그래밍 java, c++, c#, python, 등
사용 단계	유지보수	생산	재사용, 유지보수

소프트웨어 개발 프로세스 모델

- 생명 주기 지원
- 소프트웨어를 개발해 나가는 단계나 과정
  - 컨셉트를 정하는 것부터 소멸될 때까지 과정
  - 몇 달 또는 몇 년이 걸릴 수 있음
- 각 단계의 목표
  - 명확한 작업 단계
  - 손에 잡히는 결과
  - 작업의 검토
  - 다음 단계의 명시

Classes of Methodologies

- Structured Development
  - Waterfall Development – 개발 속도 지연
  - Parallel Development
- Rapid Application Development
  - Phased
  - Prototyping
- Agile Development
  - eXtreme Programming
  - SCRUM

Classes of Methodologies

개발 방법론

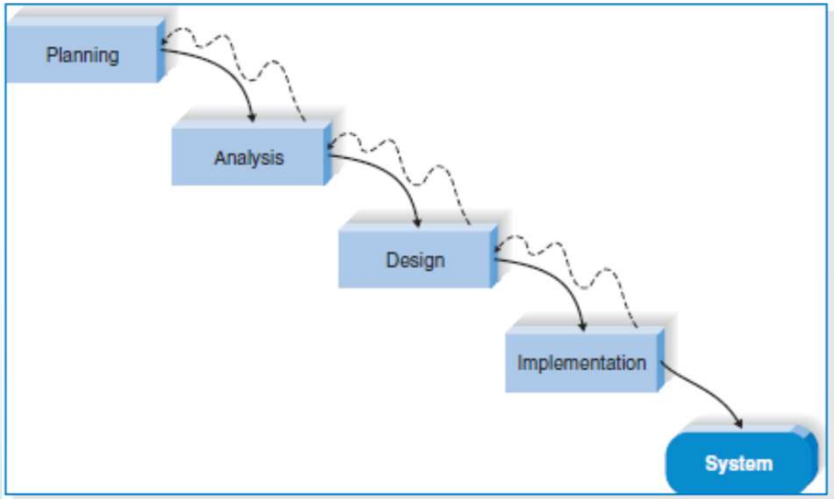
		개발 방법론		
		과정 지향	자료 지향	객체 지향
development process model 개발 과정 모델	폭포수			
	병렬개발			
	phased			
	prototyping			
	Throwaway – prototyping			
	agile			

Classes of Methodologies

- 1. Code-and-Fix Methodology
  - 생명 주기가 없음
  - 일단 프로그램을 짜보고 고침

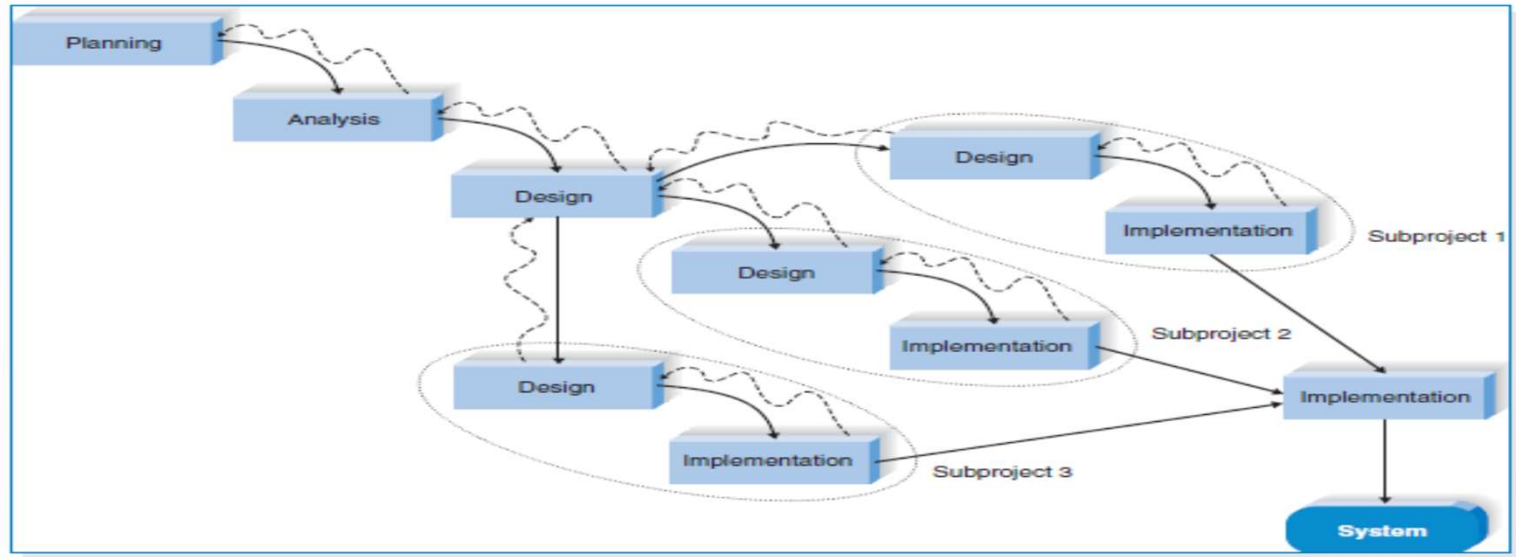
Classes of Methodologies

- 2. Structured Development
  - 2.1 Waterfall Development
    - 개발 전 완전한 설계 필요/분석과 개발 사이(몇개월 ~ 몇 년)
      - 가끔 significant Rework 필요
    - 차 개발을 종이에 명시할 때 차개발-문열면 몇 개의 실내등을 켤까? 명시하기와 차에 몇 개의 실내등이 있는지 명시하기 힘들다.



Classes of Methodologies

- 2. Structured Development
  - 2.2 Parallel Development – 분석과 시스템 납품사이 기간 최소화 목적 – rework 필요성 감소시킴
    - Subproject 사이 의존성 존재/설계결정이 서로 영향/require significant integration efforts필요



Classes of Methodologies

- 3. Rapid Application Development-case 도구, jad, 4gl, visual language, code generator 등 사용
  - 문제점 : 사용자 기대치 높임,요구 증가시킴, 문서철저 프로젝트의 경우 많은 시간 허비시킴
  - 3.1 Phased-version1, version 2… 개발
    - 사용자가 불완전한 시스템 사용 시작함 – 단점
    - 초기에 중요하고 필요한 feature 제공해야함.

A phased development-based mothodology

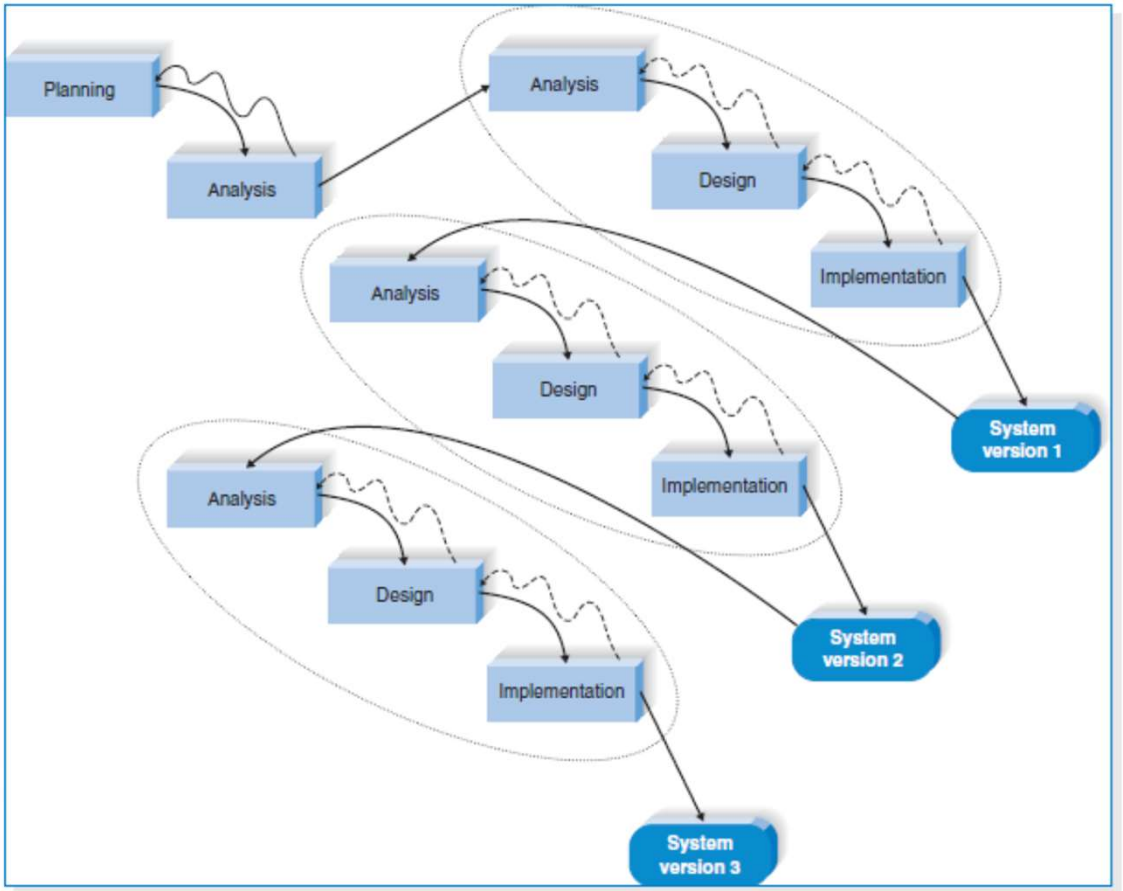
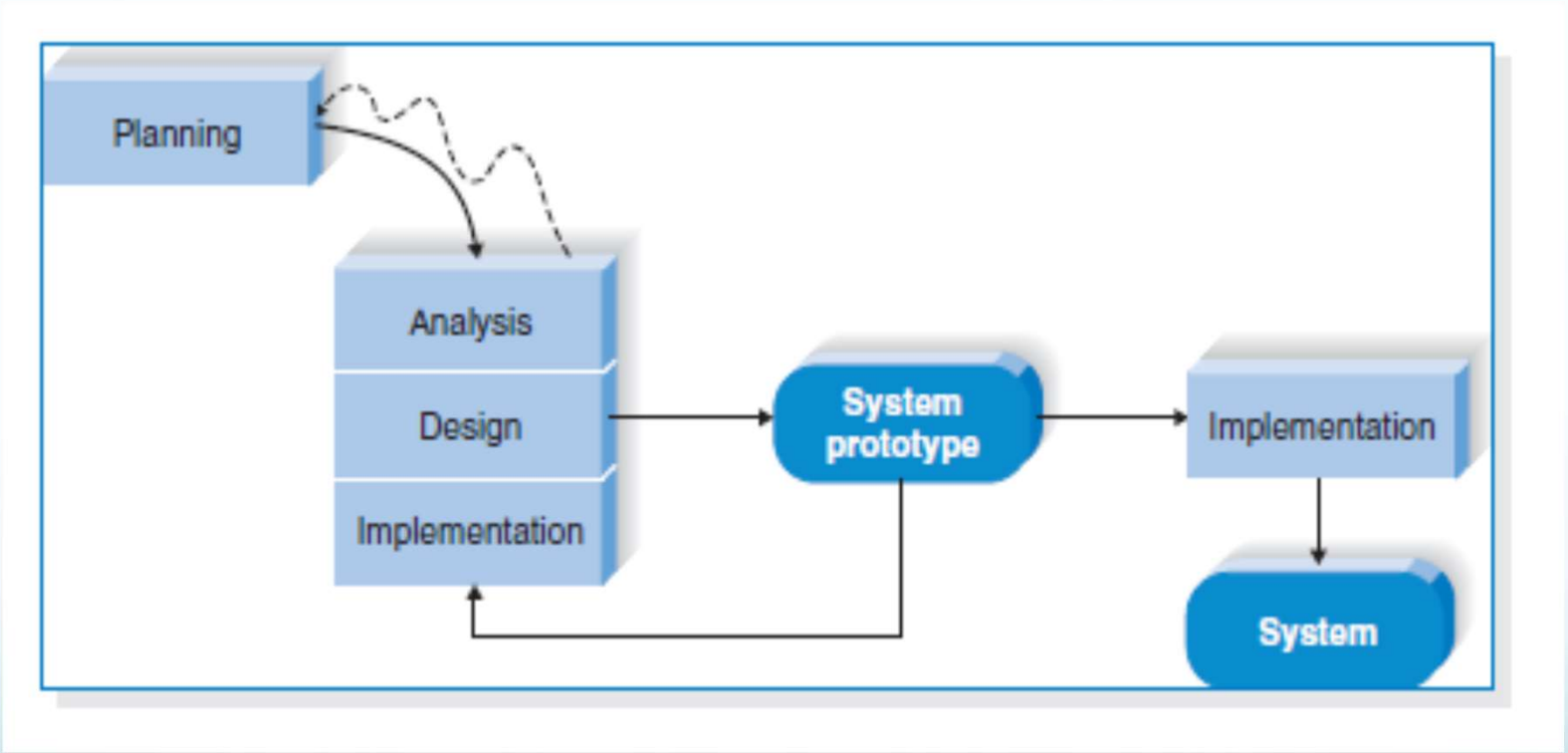


FIGURE 2-5
Iterative Development

Classes of Methodologies

- 3. Rapid Application Development
  - 3.2 Prototyping – 분석, 설계, 구현 동시진행
    - Spiral
    - 초기 주요 문제점 및 이슈 등을 주의 깊고 체계적으로 분석하지 않고 도전 정신으로 극복 / 주요 결정사항 미스가 있을 수 있다. 차를 인수하여 1년 몰다가 엔진오일을 갈려고 하니 차를 다 분해해야 함.

A prototyping based methodology



Classes of Methodologies

- 3. Rapid Application Development
  - 3.3 Throwaway –prototyping
    - 프로토타입을 주요 결정 사항에 사용
    - 결과를 설계에 반영

    - 아파트 분양모델은 분양에 사용
    - 여기서 프로토타입은 설계에 사용

Throwaway prototyping

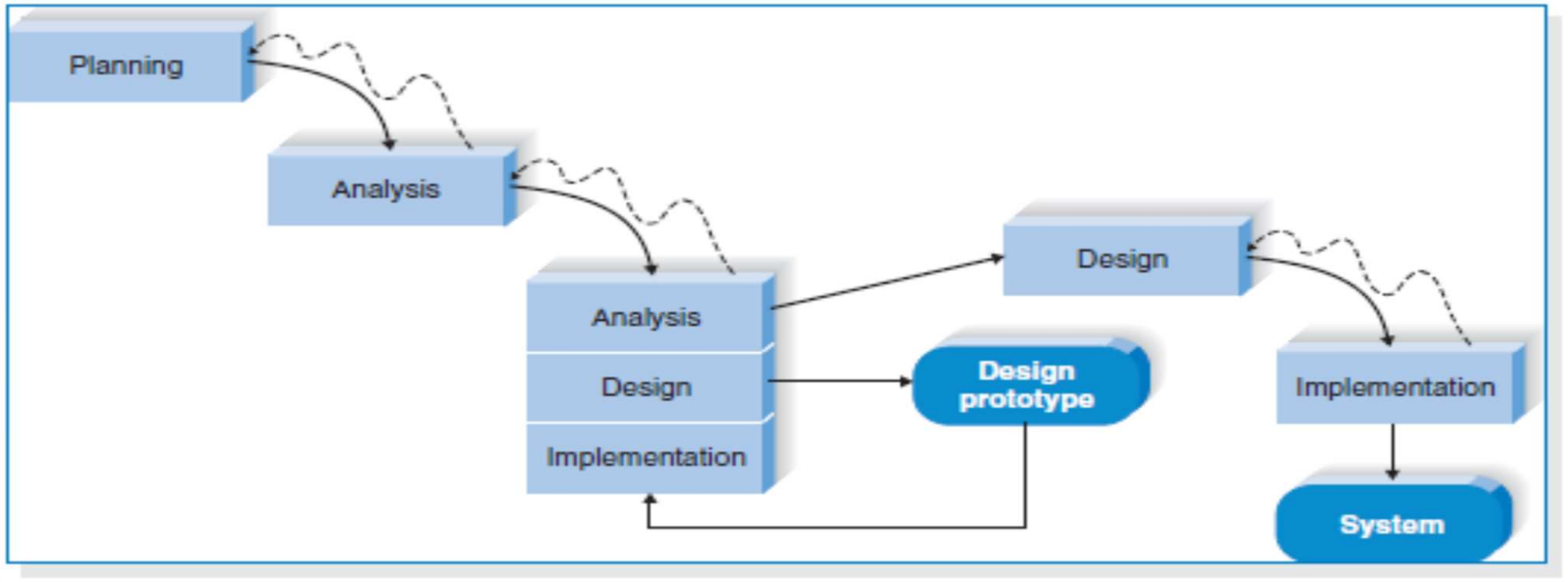


FIGURE 2-7
Throwaway Prototyping

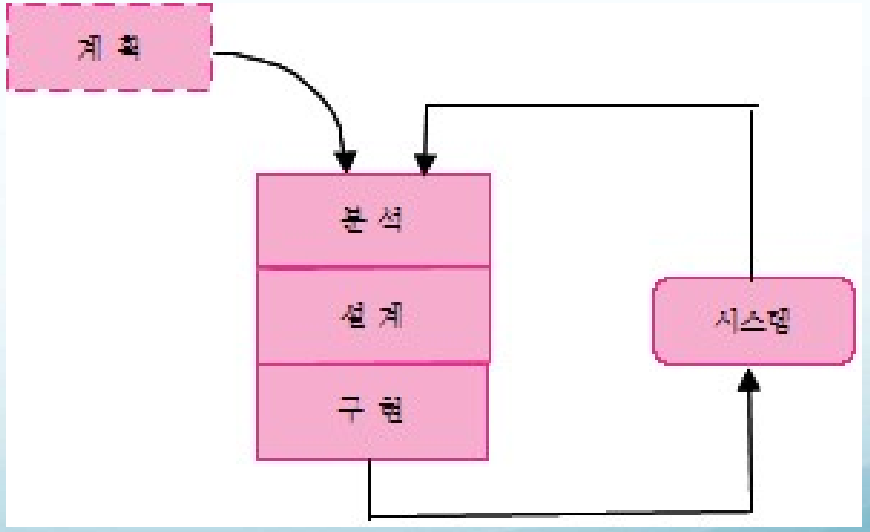
모델하우스 : 분양이 목적
Throwaway prototype : 설계가 목적

Classes of Methodologies

- 4. Agile Development
  - eXtreme Programming
  - SCRUM

애자일 모형

- Heavy 한 프로세스 - 기존방법
  - 과다한 단계
  - 과다한 문서
  - 코드가 나오기까지 시간이 많이 소요됨
- 과도한 모델링과 문서화의 짐을 과감히 생략하고 개발에 집중
  - Extreme Programming, Scrum, DSDM
- Extreme Programming
  - 의사소통
  - 단순함
  - 피드백
  - 격려
  - 테스팅



Which Methodology to Use?

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Excellent	Good	Good
That Are Reliable	Good	Good	Good	Poor	Excellent	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

The Systems Analyst: Skills

- Agents of change
  - Identify ways to improve the organization
  - Motivate & train others
- Skills needed:
  - Technical: must understand the technology
  - Business: must know the business processes
  - Analytical: must be able to solve problems
  - Communications: technical & non-technical audiences
  - Interpersonal: leadership & management
  - Ethics: deal fairly and protect confidential information

The Systems Analyst: Roles

- Business Analyst
  - Focuses on the business issues
- Systems Analyst
  - Focuses on the IS issues
- Infrastructure Analyst
  - Focuses on the technical issues
- Change Management Analyst
  - Focuses on the people and management issues
- Project Manager
  - Ensures that the project is completed on time and within budget

Object-Oriented Systems Analysis & Design

- Attempts to balance data and process
- Utilizes the Unified Modeling Language (UML) and the Unified Process
- Characteristics of OOAD:
  - Use-case Driven
  - Architecture Centric
  - Iterative and Incremental

Characteristics of Object-Oriented Systems

- Classes & Objects
  - Object (instance): instantiation of a class
  - Attributes: information that describes the class
  - State: describes its values and relationships at a point in time
- Methods & Messages
  - Methods: the behavior of a class
  - Messages: information sent to an object to trigger a method (procedure call)

Characteristics of Object-Oriented Systems (cont.)

- Encapsulation & information hiding
  - Encapsulation: combination of process & data
  - Information hiding: functionality is hidden
- Inheritance
  - General classes are created (superclasses)
  - Subclasses can inherit data and methods from a superclass

Characteristics of Object-Oriented Systems (cont.)

- Polymorphism & dynamic binding
  - Polymorphism: the same message can have different meanings
  - Dynamic binding: type of object is not determined until run-time
  - Contrast with static binding

Object-Oriented Systems Analysis & Design

- Use-case driven
  - Use-cases define the behavior of a system
  - Each use-case focuses on one business process
- Architecture centric
  - Functional (external) view: focuses on the user's perspective
  - Static (structural) view: focuses on attributes, methods, classes & relationships
  - Dynamic (behavioral) view: focuses on messages between classes and resulting behaviors

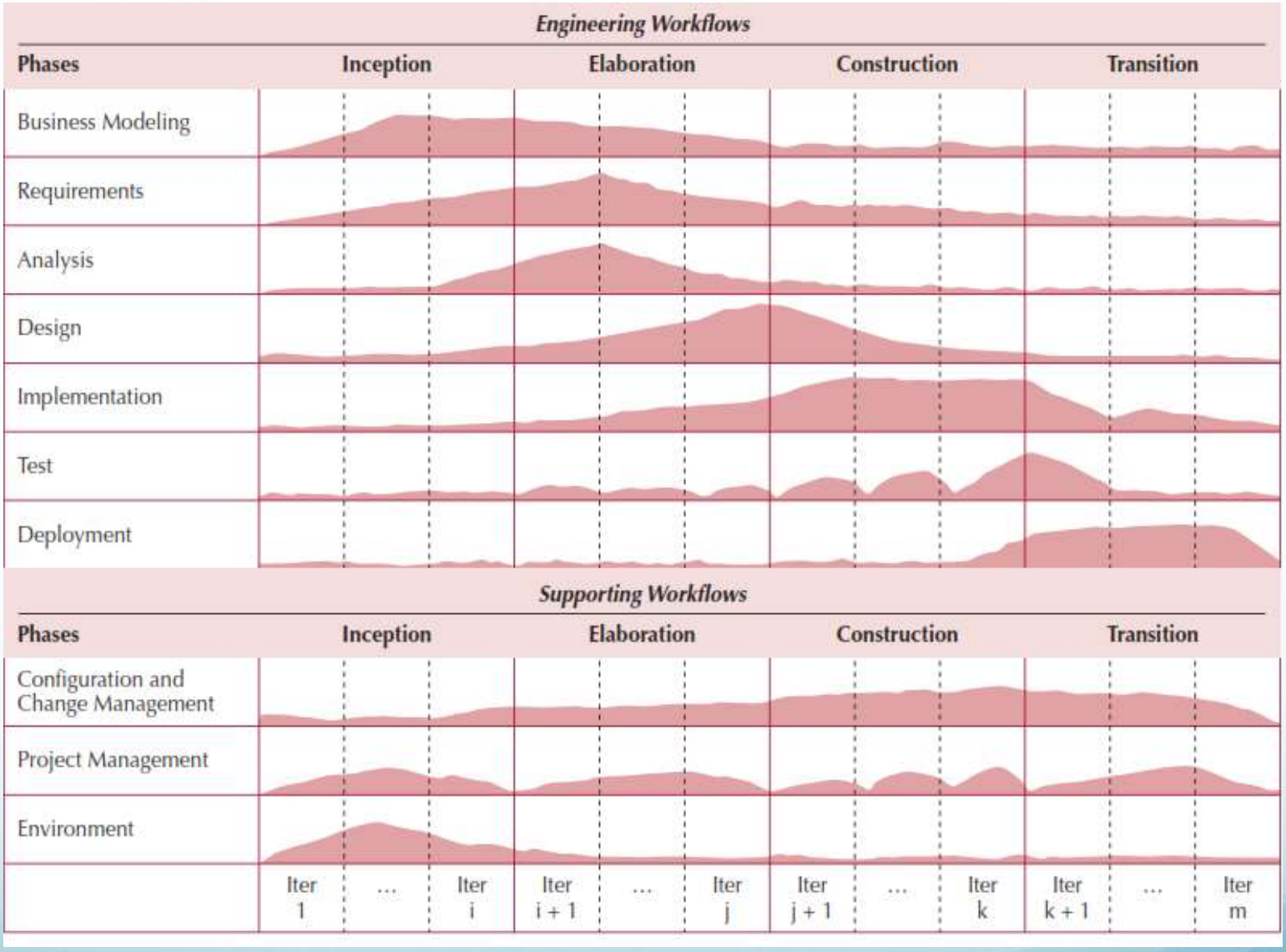
Object-Oriented Systems Analysis & Design (cont.)

- Iterative & incremental
  - Undergoes continuous testing & refinement
  - The analyst understands the system better over time
- Benefits of OOSAD
  - Break a complex system into smaller, more manageable modules
  - Work on modules individually

The Unified Process

- A specific methodology that maps out when and how to use the various UML techniques for object-oriented analysis and design
- A two-dimensional process consisting of phases and workflows
  - Phases are time periods in development
  - Workflows are the tasks that occur in each phase
  - Activities in both phases & workflows will overlap

The Unified Process



Unified Process Phases

- Inception
  - Feasibility analyses performed
  - Workflows vary but focus is on business modeling & requirements gathering
- Elaboration
  - Heavy focus on analysis & design
  - Other workflows may be included
- Construction: Focus on programming (implementation)
- Transition--Focus on testing & deployment

Engineering Workflows

- Business modeling
- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment

Supporting Workflows

- Project management
- Configuration and change management
- Environment
- Operations and support*
- Infrastructure management*

* Part of the *enhanced* unified process

Extensions to the Unified Process

- The Unified Process does not include:
  - Staffing
  - Budgeting
  - Contract management
  - Maintenance
  - Operations
  - Support
  - Cross- or inter-project issues

Extensions to the Unified Process (cont.)

- Add a Production Phase to address issues after the product has been deployed
- New Workflows:
  - Operations & Support
  - Infrastructure management
- Modifications to existing workflows:
  - Test workflow
  - Deployment workflow
  - Environment workflow
  - Project Management workflow
  - Configuration & change management workflow

Unified Modeling Language

- Provides a common vocabulary of object-oriented terms and diagramming techniques rich enough to model any systems development project from analysis through implementation
- Version 2.5 has 15 diagrams in 2 major groups:
  - Structure diagrams
  - Behavior diagrams

UML Structure Diagrams

- Represent the data and static relationships in an information system
  - Class
  - Object
  - Package
  - Deployment
  - Component
  - Composite structure

UML Behavior Diagrams

- Depict the dynamic relationships among the instances or objects that represent the business information system
  - Activity
  - Sequence
  - Communication
  - Interaction overview
  - Timing
  - Behavior state machine
  - Protocol state machine,
  - Use-case diagrams

Summary

- All systems development projects follow essentially the same process, called the system development life cycle (SDLC)
- System development methodologies are formalized approaches to implementing SDLCs
- The systems analyst needs a variety of skills and plays a number of different roles
- Object-oriented systems differ from traditional systems

Summary

- Object-Oriented Systems Analysis and Design (OOSAD) uses a use-case-driven, architecture-centric, iterative, and incremental information systems development approach
- The Unified Process is a two-dimensional systems development process described with a set of phases and workflows
- The Unified Modeling Language, or UML, is a standard set of diagramming techniques

Questions?