# Section 2. 요구사항 분석

## 1. 현행 시스템 분석

### (1) 현행 시스템 파악

#### ① 현행 시스템 파악의 정의

- 현행 시스템이 어떤 하위 시스템으로 구성되어 있는지
- 제공하는 기능이 무엇인지
- 다른 시스템들과 어떤 정보를 주고받는지
- 어떤 기술요소를 사용하고 있는지
- 사용하고 있는 소프트웨어 및 하드웨어는 무엇인지
- 네트워크는 어떻게 구성되어 있는지

#### ② 현행 시스템 파악의 목적

- 향후 개발하고자 하는 시스템의 개발범위 및 이행 방향성 설정에 도움을 주는 것이 목적

#### ③ 현행 시스템 파악 절차

- 1단계 - 현행 시스템의 구성, 기능, 인터페이스 현황을 파악하는 단계
- 2단계 - 현행 시스템의 아키텍처 및 소프트웨어 구성 현황을 파악하는 단계
- 3단계 - 현행 시스템의 하드웨어 및 네트워크 구성 현황을 파악하는 단계

### (2) 플랫폼 기능 분석

#### ① 플랫폼 정의

- 어플리케이션을 구동시키는데 필요한 하드웨어와 소프트웨어의 결합
- 공급자와 수요자 등 복수 그립이 참여하여 각 그룹이 얻고자 하는 가치를 공정한 거래를 통해 교환할 수 있도록 구축된 환경

#### ② 플랫폼 기능

- 소프트웨어 개발과 운영비용이 감소하고 생산성이 향상
- 플랫폼 내의 커뮤니티를 형성하고 네트워크 효과를 유발

#### ③ 플랫폼의 유형

- 싱글 사이드 플랫폼(single-side platform)
  - 제휴 관계를 통해 소비자와 공급자를 연결하는 형태
- 투 사이드 플랫폼(two-side platform)
  - 두 그룹을 중개하고 모두에게 개방하는 형태
- 멀티 사이드 플랫폼(multi-side platform)
  - 다양한 이해관계 그룹을 연결하여 중개하는 형태

### (3) 운영체제 분석

#### ① 운영체제 개념

- 컴퓨터 시스템 자원을 효율적으로 관리하여 사용자가 컴퓨터를 편리하게 사용할 수 있도록 환경을 제공해주는 시스템 소프트웨어
- 컴퓨터 시스템이 제공하는 모든 하드웨어, 소프트웨어를 사용할 수 있도록 해줌
- 사용자와 하드웨어간의 인터페이스를 담당

#### ② 운영체제 종류

- 유닉스(UNIX)
- 리눅스(Linux)
- 윈도우즈(Windows)
- 맥 OS(Mac OS)

PC

- iOS
- Android
- 윈도우폰 OS

Mobile

시스템 ex) OS

응용 ex) 카카오톡

### (4) 네트워크 분석

#### ① 네트워크 개념

- 노드(컴퓨터)들이 자원을 공유할 수 있게 하는 디지털 전기 통신망
- 노드 간 연결을 통해 서로에게 데이터를 교환

#### ② 프로토콜

- 데이터를 교환하기 위해 사용하는 통신 규칙
- 프로토콜의 3요소
  - 구문(Syntax) : 데이터의 형식이나 부호화 및 신호 레벨을 규정
  - 의미(Semantic) : 전송의 조작이나 오류 제어를 위한 제어 정보에 대한 규정
  - 타이밍(Timing) : 접속되어 있는 개체 간의 통신 속도의 조정이나 메시지의 순서 제어 규정

구타일 왜하니 의미구!

### (5) DBMS 분석

#### ① DBMS(Database Management System) 개념

- 사용자, 어플리케이션등의 상호 작용을 위해 데이터를 저장하고 분석하는 소프트웨어
- 데이터베이스 생성, 조회, 변경 등의 관리

#### ② 현행 시스템 데이터베이스 분석

- DBMS의 종류, 버전, 구성방식, 스토리지 크기, 백업 주기 분석
- 테이블 수량, 데이터 증가 추이, 백업 방식 등을 분석

## (6) 미들웨어(Middleware) 분석

### ① 미들웨어 개념

- 양 쪽을 연결하여 데이터를 주고 받을 수 있도록 중간에서 매개 역할을 하는 소프트웨어

### ② 미들웨어 종류

- PRC(Remote Procedure Call)
  - 클라이언트가 원격에서 동작하는 프로시저를 호출하는 시스템
- MOM(Message Oriented Middleware)
  - 응용 소프트웨어 간의 데이터 통신을 위한 소프트웨어
  - 메시지 기반의 비동기형 메시지를 전달하는 방식의 미들웨어
- ORB (Object Request Broker)
  - 객체지향 시스템에서 객체 및 서비스를 요청하고 전송할 수 있도록 지원하는 미들웨어
- DB 접속 미들웨어
  - 애플리케이션과 데이터베이스 서버를 연결해주는 미들웨어
- TP 모니터
  - 온라인 트랜잭션을 처리 및 감시하는 미들웨어
- WAS(Web Application Server)
  - 동적인 콘텐츠를 처리하기 위한 미들웨어
- ESB(Enterprise Service Bus)
  - 메시지 기반으로 느슨한 결합형태의 표준 인터페이스 통신을 지원하는 미들웨어
  - 기업 안팎에 있는 모든 시스템 환경을 연동하는 미들웨어

## 2. 요구 공학

### (1) 요구공학 개념

- 고객 요구를 체계적으로 수집, 분석, 명세화, 검증하고 추적, 변경되는 요구사항을 도출하고 관리하는 기법

### (2) 요구공학의 필요성

- 분석의 어려움
  - 이해부족, 의사소통, 잦은 요구사항의 변경
- 요구사항 변화
  - 요구사항은 개발초기에 불완전하고, 개발 동안 지속적으로 변화
- 관점별 차이 발생
  - 묵시적 요구사항, 변경과 추적에 대한 문제, 해당 업무에 대한 지식

### (3) 요구사항의 분류

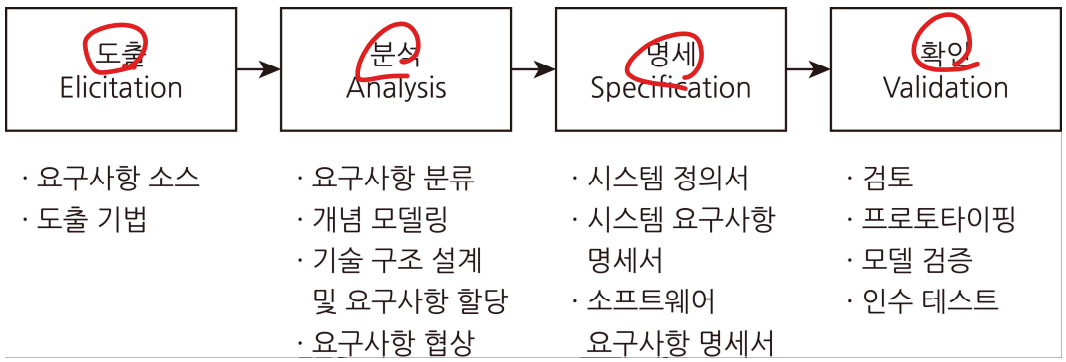
#### ① 참여자 관점

- 사용자 요구사항
  - 사용자의 관점에서 소프트웨어에 대해 원하는 사항들
- 시스템 요구사항
  - 설계자의 관점에서 하드웨어와 소프트웨어가 갖춰야 하는 것들
- 소프트웨어 요구 사항
  - 개발자의 관점에서 소프트웨어가 갖춰야 하는 사항들

#### ② 요구사항 내용의 종류

- 기능적 요구사항
  - 소프트웨어를 구성하는 기능들이 무엇인지를 정의한 것
- 비기능적 요구사항
  - 소프트웨어의 기능들에 대한 조건과 제약 사항들이 무엇인지 정의한 것
  - 보안, 성능, 품질, 안정성 등

### (4) 요구사항 개발 프로세스




#### ① 도출

- 소프트웨어가 해결해야 할 문제를 이해하고 요구사항이 어디에 있고, 어떻게 수집할 것인가를 확인
- 요구사항 도출 기법
  - 이해 관계자 분석
  - 브레인 스토밍
  - 인터뷰
  - 문서 분석 및 검토
  - 포커스 그룹
  - 인터페이스 분석
  - 관찰
  - 프로토타이핑
  - 요구사항 워크숍
  - 설문 조사

#### ② 분석

- 요구사항들 간에 상충 되는 것을 해결
- 소프트웨어의 범위를 파악
- 업무환경과의 상호작용 파악(도메인 분석)
- 구조적 분석 도구 ↓ 하향식
  - DFD(Data Flow Diagram) : 자료 흐름도
  - Data Dictionary : 자료 사전
  - Mini-Spec : 소단위 명세서
  - ERD(Entity Relationship Diagram) : 개체 관계도
  - STD(State Transition Diagram) : 상태 전이도
- 객체지향 분석 도구
  - UML(Unified Modeling Language)
  - 모델링

### ③ 명세

- 체계적으로 검토, 평가, 승인될 수 있는 문서를 작성
- 시스템정의, 시스템 요구사항, 소프트웨어 요구사항을 작성
- 요구사항 명세 기법

| 구분 | 정형 명세 기법 | 비정형 명세 기법 |
|---|---|---|
| 기반 | - 수학, 논리학 | - 자연어, 그림 중심 |
| 작성기법 | - 수학적 기호, 정형화된 표기법 | - 일반 명사, 동사 등의 자연어를 기반으로 서술하거나 다이어그램으로 작성 |
| 장점 | - 명세 오류 및 모호성 쉽게 파악 | - 사용자/개발자간 의사전달 용이 |
| 단점 | - 작성이 어렵고, 시간이 많이 소모됨 | - 내용이 모호하고, 완전한 검증이 곤란 |
| 언어 종류 | - VDM, Z, Petri-net, CSP | - FSM, Decision Table, ER 모델링, State Chart(SADT) 등 |

- 산출물
  - 시스템 정의서
  - 시스템 요구사항 명세서
  - 소프트웨어 요구사항 명세서

### ④ 확인

- 분석가가 요구사항을 이해했는지 확인(Validation)
- 요구사항 문서가 일관성 있고 완전한지 검증(Verification)
- 이해 관계자들이 문서를 검토하고, 형상관리를 수행

## (5) 요구사항 분석 도구

### ① 요구사항 분석 CASE(Computer Aided Software Engineering) 도구

- 요구사항을 자동으로 분석하고, 요구사항 분석 명세서를 기술하는 도구
- 소프트웨어 개발 전반에 걸쳐 적용된다.
- CASE 도구의 분류

| 분류 | 설명 |
|---|---|
| 상위 CASE | - 생명주기 전반부에 사용되며, 소프트웨어의 계획과 요구분석, 설계 단계를 지원한다.<br>- 모순검사, 오류검사, 자료흐름도 작성 등의 기능을 수행한다. |
| 하위 CASE | - 생명 주기 후반부에 사용되며, 코드의 작성과 테스트, 문서화 하는 과정을 지원한다.<br>- 구문 편집기, 코드 생성기 등의 기능을 수행한다. |
| 통합 CASE | - 소프트웨어 생명주기에 포함되는 전체 과정을 지원한다. |

상위: 분석, 설계
하위: 구현, 테스트

- 종류

| 종류 | 설명 |
|---|---|
| SADT | - Structured Analaysis and Design Technique<br>- SoftTech 사에서 개발<br>- 시스템 정의, 요구사항 분석, 시스템/소프트웨어 설계를 이용되는 구조적 분석 및 설계도구 |
| SREM | - Software Requirements Engineering Methodology<br>- 실시간 처리 소프트웨어 시스템에서 요구사항을 명확히 기술하도록 할 목적으로 개발 |
| PSL/PSA | - 미시간 대학에서 개발한 것으로 PSL과 PSA를 사용하는 자동화 도구<br>- PSL(Problem Statement Language) : 문제 기술언어<br>- PSA(Problem Statement Analyzer) : PSL로 기술한 요구사항을 자동으로 분석하여 다양한 보고서를 출력하는 문제 분석기 |
| TAGS | - Technology for Automated Generation of Systems<br>- 시스템 공학 방법 응용에 대한 자동 접근 방법<br>- 개발 주기의 전 과정에 이용할 수 있는 통합 자동화 도구 |

### ② HIPO(Hierarchy Inut Process Output)

- HIPO 의 개념
  - 하향식 소프트웨어 개발을 위한 문서화 도구
  - 시스템의 기능을 여러 개의 고유 모듈들로 분할하여 이들 간의 계층구조를 표현한 도표
- HIPO 의 기능
  - 분석 및 설계 도구로 사용된다.
  - 하향식 개발에 적합하다.
  - 체계적인 문서관리에 효율적이다.
  - 기능과 자료의 의존관계를 명시할 수 있다.
- HIPO Chart 종류

| 종류 | 설명 |
|---|---|
| 가시적 도표<br>(Visual Table of Content) | - 시스템의 전체 기능과 흐름을 보여주는 Tree(계층) 구조<br>- 가시적 도표에는 입력, 처리, 출력 없음 |
| 총체적 도표<br>(Overview Diagram) | - 프로그램을 구성하는 기능을 기술한 것<br>- 입력, 처리, 출력에 대한 전반적인 정보 제공 |
| 세부적 도표<br>(Detail Diagram) | - 총체적 도표에 표시된 기능을 구성하는 기본 요소들을 상세히 기술하는 도표<br>- 총체적 도표와 같은 모양이지만 내용만 좀 더 복잡하게 들어간 형태 |

## 3. 요구사항 분석 모델링

### (1) 모델링의 개념

- 복잡한 시스템을 쉽게 이해하기 위해 불필요한 부분을 생략하고 추상화하여 간단한 모델로 표현하는 것을 의미
- 소프트웨어를 구성하는 모듈들을 식별하고, 이것들의 연결을 그림으로 표현
- 요구분석 과정에 의해 수집된 개념과 정보를 분석하여 UML과 같은 방법을 이용하여 모델로 비주얼화

### (2) 모델링이 주는 도움

- 소프트웨어를 이해하는 데 도움
- 이해관계자들 사이에서 문제를 해결 할 수 있도록 해준다.
- 파악한 개념을 사용자와 고객에게 전달할 때 도움을 준다.
- 설계, 구현, 테스팅, 유지보수에 개념적인 기준을 제공한다.

### (3) 모델링 구분

- 기능적 모델링
- 정적 모델링 – 구조
- 동적 모델링 – 행위

### (4) 분석 모델의 종류

- 구조적 분석 모델
- 객체 지향 분석 모델
- 정보공학 분석 모델
- 정형화 분석 모델

### (5) 구조적 분석 모델

#### ① 구조적 분석 방법론

- 도형화된 도구를 이용해 정형화된 분석 절차에 따라 사용자 요구사항을 파악하고 문서화하는 분석 기법
- 하향식 기능 분해 기법 등을 사용하는 특성

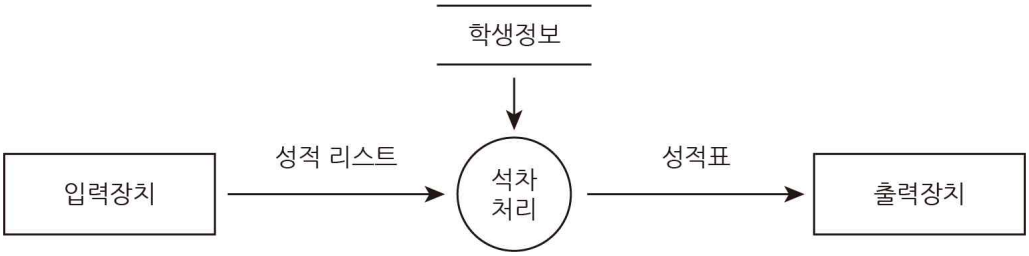
#### ② 구조적 분석 도구

㉠ 자료 흐름도(DFD, Data Flow Diagram)

- 가장 보편적으로 사용되는 시스템 모델링 도구로서, 기능 중심의 시스템을 모델링 하는데 적합
- 자료의 흐름과 처리 과정을 도형 중심으로 기술
- 자료 흐름 그래프 또는 버블 차트라고도 함
- 자료 흐름도 구성요소

| 구성요소 | 설명 | 기호 |
|---|---|---|
| 처리 과정(Process) | 자료를 변환시키는 처리 과정을 나타낸다. | ○ |
| 자료 흐름(Data Flow) | 자료의 이동을 나타낸다. | → |
| 자료 저장소(Data Store) | 파일, 데이터베이스 등 자료가 저장되는 곳을 나타낸다. | ═ |
| 단말(Terminator) | 데이터의 입출력 주체(사용자)를 나타낸다. | □ |

- 자료 흐름도 사례




㉡ 자료사전(Data Dictionary, DD)

- 자료흐름도에 기술된 모든 자료들에 대한 사항을 자세히 정의
- 자료사전 사용 기호

중하

| 기호 | 의미 | 설명 |
|---|---|---|
| = | 자료의 정의 | ~로 구성되어 있다 |
| + | 자료의 연결 | 그리고, 순차(and) |
| ( ) | 자료의 생략 | 생략 가능한 자료 |
| [ \| ] | 자료의 선택 | 여러 대안 중 하나 선택 |
| { } | 자료의 반복 | 자료의 반복 |
| ** | 자료의 설명 | 주석 |

- 자료사전 예시

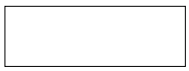
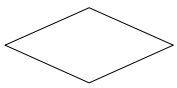
| | |
|---|---|
| 자료 흐름 | 쇼핑몰 회원정보는 회원번호, 회원성명, 전화번호, 휴대폰번호로 구성되어 있고, 전화번호와 휴대폰번호는 둘 중 하나만 선택이 가능하다. |
| 표기형식 | 회원정보 = 회원번호 + 회원성명 + [전화번호 \| 휴대폰번호] |

㉢ 소단위 명세서(Mini-Specification)

- 자료 흐름도에서 어떤 일이 수행되는지를 정의하기 위해 각 처리들이 수행하는 업무를 상세하게 작성
- 프로세스 명세서라고도 한다.
- 소단위 명세서는 구조적 언어이고, 선후 조건문, 의사결정표 등이 사용

㉣ 개체 관계도(Entity Relationship Diagram, ERD)

- 시스템에서 처리되는 개체(자료)와 개체의 구성과 속성, 개체 간의 관계를 표현하여 자료를 모델화 하는데 사용
- 개체 관계도 구성

| 속성 | 설명 | ERD 기호 |
|---|---|---|
| 개체(Entity) | 업무의 중심이 되는 실체 |  |
| 속성(Attribute) | 업무에 속하는 구체적인 항목 |  |
| 관계(Relationship) | 업무와 업무의 연관관계 |  |

㉤ 상태 전이도(State Transition Diagram, STD)

- 시스템에 어떤 일이 발생할 경우 시스템의 상태와 상태 간의 전이를 모델화한 것으로, 상태 전이도를 통해 개발자는 시스템의 행위를 정의

### (6) 객체 지향 분석 모델

준비 ⇆ 실행
↘ 대기 ↙

#### ① 객체 지향 분석

- 사용자의 요구사항을 분석하여 요구된 문제와 관련된 모든 클래스, 이와 연관된 속성과 연산, 그들 간의 관계 등을 정의하여 모델링하는 작업

#### ② 객체지향 분석 방법론

㉠ Rumbaugh(럼바우) 방법

- 가장 일반적으로 사용되는 방법, 분석 활동을 객체 모델, 동적 모델, 기능 모델로 나누어 수행
- 분석 절차

| 종류 | 설명 |
|---|---|
| 객체 모델링<br>(Object Modeling) | - 시스템에서 요구되는 객체를 찾아내어 속성과 연산 식별 및 객체들 간의 관계를 규정해 객체 다이어그램으로 표현<br>- 세 가지 모델 중 가장 선행되어야 함 |
| 동적 모델링<br>(Dynamic Modeling) | - 상태 다이어그램을 이용하여 시간의 흐름에 따라 제어 흐름, 동작 순서 등 동적인 행위 표현<br>- 객체나 클래스의 상태, 사건을 중심으로 표현 |
| 기능 모델링<br>(Functional Modeling) | - 자료 흐름도(DFD)를 이용해 다수의 프로세스들 간의 자료 흐름을 중심으로 처리 과정 표현<br>- 어떤 데이터를 입력하여 어떤 결과를 구할 것인지를 표현 |

㉡ Booch(부치) 방법

- 미시적 개발 프로세스와 거시적 개발 프로세스를 모두 사용하는 분석 방법

㉢ Jacobson 방법

- Use case를 강조하여 사용하는 분석 방법

㉣ Coad와 Yourdon 방법

- E-R 다이어그램을 사용하여 객체의 행위를 모델링, 객체 식별, 구조 식별, 주제 정의 등의 과정으로 구성하는 기법

㉤ Wirfs-Brock 방법

- 분석과 설계간 구분 없음
- 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행

# 03 소프트웨어 설계

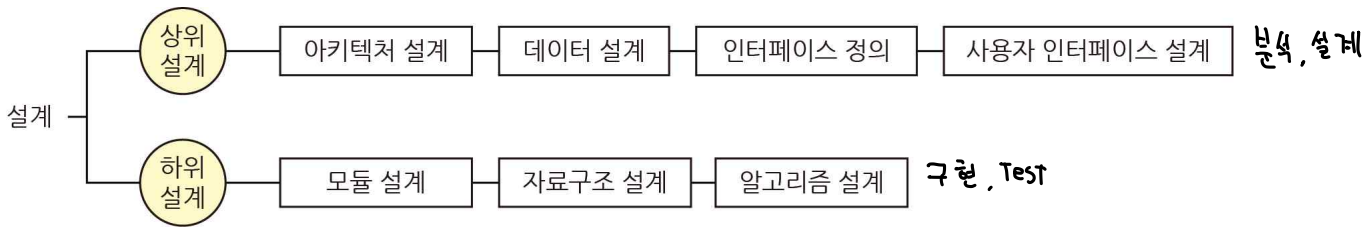
## Section 1. 소프트웨어 설계의 기본 원칙

### 1. 소프트웨어 설계

#### (1) 소프트웨어 설계의 개념

- 요구사항 명세서를 참조하여 소프트웨어의 구체적인 설계서를 작성하는 단계 S/W
- 물리적으로 구현이 가능하도록 시스템을 구체적으로 정의하는 단계 System

#### (2) 소프트웨어 설계의 종류




##### ① 상위설계

- 아키텍처 설계
  - 시스템의 전체적인 구조 설계
- 데이터 설계
  - 시스템에 필요한 정보를 설계
  - 데이터 베이스 설계
- 인터페이스 정의
  - 시스템의 구조와 서브 시스템들 사이의 인터페이스를 명확히 정의
- 사용자 인터페이스 설계
  - 사용자가 익숙하고 편리하게 사용하도록 인터페이스 설계

##### ② 하위설계

- 모듈 설계
  - 각 모듈의 실제적인 내부를 알고리즘 형태로 표현
- 자료구조 설계
  - 자료구조, 변수 등에 대한 상세한 정보를 설계
- 알고리즘 설계
  - 업무의 처리 절차 등을 설계

### (3) 소프트웨어 설계의 원리

- 분할과 정복(Divide & Conquer)
  - 규모가 큰 소프트웨어를 여러 개의 작은 서브 시스템으로 나누어 하나씩 완성 시킨다.
- 추상화(Abstraction)
  - 특정한 목적과 관련된 필수 정보만 추출하여 강조하고, 관련이 없는 세부 사항을 생략함으로써, 본질적인 문제에 집중할 수 있도록 한다.
  - 자세한 구현 전에, 상위 레벨에서 제품의 구현을 먼저 생각해보는 것
  - 추상화 기법

| 추상화 기법 | 설명 |
|---|---|
| 과정 추상화 | 자세한 단계를 고려하지 않고 상위 수준에서 수행 흐름만 먼저 설계 |
| 데이터 추상화 | 데이터 구조를 대표할 수 있는 표현으로 대체하는 것이다. |
| 제어 추상화 | 여러 명령들을 간단한 표현으로 대체하는 것이다. |

- 단계적 분해(Gradual Decomposition)
  - 기능을 점점 작은 단위로 나누어 점차적으로 구체화 하는 방법
- 모듈화(Modulization)
  - 실제로 개발할 수 있는 작은 단위로 나눈다.
- 정보은닉(Information Hiding)
  - 다른 객체에게 자신의 정보를 숨기고, 자신의 연산만을 통해 접근이 가능하도록 한다.
  - 클래스 외부에서 특정 정보에 대한 접근을 막는다는 의미
  - 캡슐화와 밀접한 관계가 있다.

## 2. 설계 모델링

### (1) 설계 모델링 개념

- 소프트웨어를 구성하는 모듈들을 식별하고, 이것들의 연결을 그림으로 표현한 것
- 소프트웨어를 만들기 위한 계획 또는 만들어야 할 물건을 의미있게 표현한 것
- 소프트웨어에 대하여 여러 엔지니어들이 공통된 개념을 공유하는데 도움을 준다.

### (2) 설계 모델링 원칙

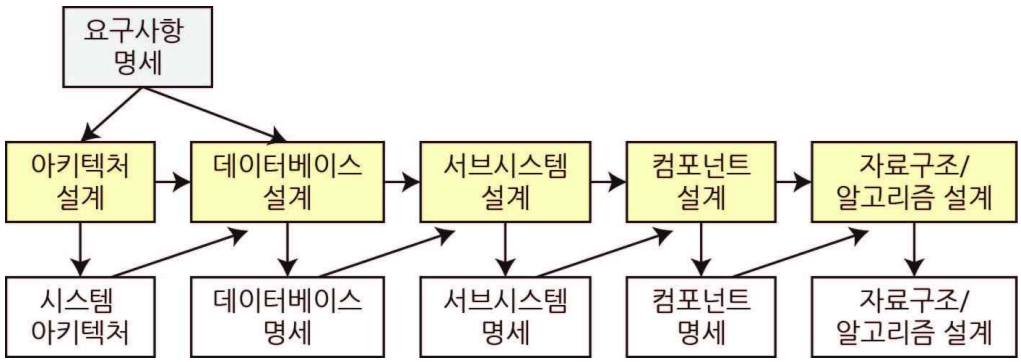
- 소프트웨어 설계는 변경이 용이하도록 구조화되어야 한다.
- 한 함수 안에 특정 기능을 수행하는 데 필요한 자료만을 사용하도록 한다.
- 요구사항 분석에서 얻은 정보를 이용하여 반복적 방법을 통해 이루어져야 한다.
- 독립적이고 기능적인 특성들을 지닌 모듈 단위로 설계되어야 한다.

### (3) 설계 모델링 유형

- 구조 모델링
  - 소프트웨어를 구성하는 컴포넌트들의 유형, 인터페이스, 내부 설계 구조 및 이들의 연결 구조를 모델링
  - 시스템의 구성 요소들과 이들 사이의 구조적인 관계와 특성들의 모델링
  - UML 정적 다이어그램

- 행위 모델링
  - 소프트웨어의 구성요소들의 기능들이 언제, 어떠한 순서로 기능을 수행해야 작용하는지를 모델링
  - 시스템의 구성 요소들이 언제 어떠한 순서로 수행되는가와 같은 동적 특성들의 모델링
  - UML 동적 다이어그램

### (4) 소프트웨어 설계 절차 및 유형




| 유형 | 설명 |
|---|---|
| 아키텍처 설계 | 시스템을 구성하는 서브시스템들과 그들 간의 관계를 파악하고 명세한다. |
| 데이터베이스 설계 | 시스템 구현에 사용되는 데이터의 구조를 자세하게 설계하고 명세한다. |
| 서브시스템 설계 | 각 서브시스템이 담당하는 서비스와 제약사항들에 대해 명세한다. |
| 컴포넌트 설계 | 서브시스템이 수행하는 기능을 여러 컴포넌트에 할당하고, 컴포넌트들의 인터페이스를 설계하고 명세한다. |
| 자료구조와 알고리즘 설계 | 컴퓨터에 자료를 효율적으로 저장하는 방식과, 자료구조 내에서 기본적인 연산방법을 설계하고 명세 |
| 협약에 의한 설계 | 클래스에 대한 여러 가정을 공유하도록 명세<br>- 선행 조건 : 컴포넌트 오퍼레이션 사용 전에 참이 되어야할 조건<br>- 결과 조건 : 사용 후 만족되어야 할 결과조건<br>- 불변 조건 : 오퍼레이션이 실행되는 동안 항상 만족되어야할 조건 |