정보시스템과 소프트웨어 개발

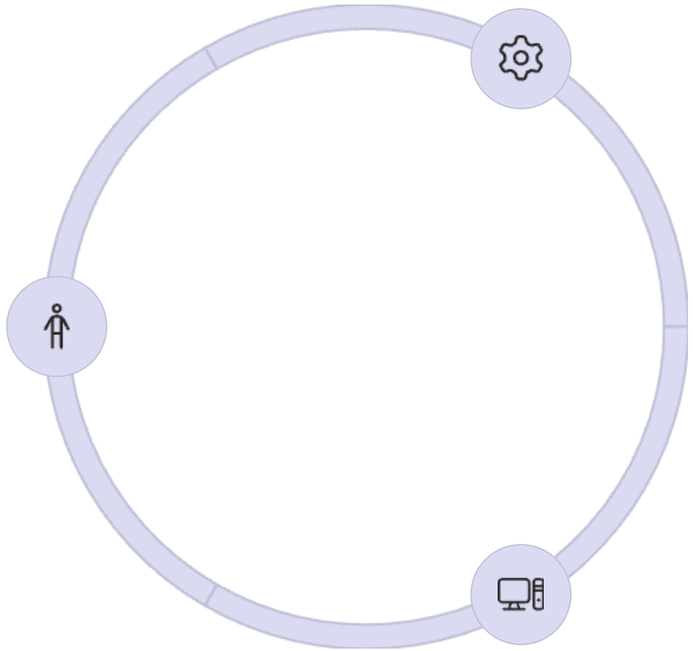
.

김동근
kdgfox@gmail.com

정보 시스템의 구성요소

사람

시스템을 사용하고 관리하는 모든 이해관계자



프로세스

시스템이 작동하는 규칙과 절차

정보 기술

하드웨어, 소프트웨어, 네트워크 등의 기술적 요소

정보 시스템은 세 가지 중요한 요소로 구성됩니다: 사람, 프로세스, 정보 기술(IT). 예를 들어 경연 프로그램의 투표 시스템에서는 투표에 참여하는 시청자(사람), 투표 규칙과 처리 절차(프로세스), 그리고 시청자들이 사용하는 휴대폰, 앱, 웹사이트, 방송국 서버(정보 기술)가 모두 필요합니다.

소프트웨어 개발의 특징

개발 과정

소프트웨어는 제조되는 것이 아니라 개발됩니다. 이는 단순한 코딩 작업이 아닌 구상, 명세화, 설계, 문서화, 테스팅, 버그 수정, 유지 관리 등 복합적인 과정을 포함합니다.

지속적 변화

소프트웨어는 완성 후에도 계속 변경되고 진화합니다. 사용자 요구사항이나 환경의 변화에 따라 지속적인 업데이트가 필요합니다.

오류 발생

소프트웨어는 본질적으로 오류가 많습니다. 복잡한 로직과 다양한 상호작용으로 인해 모든 오류를 사전에 예방하기 어렵습니다.

예측 어려움

개발 일정과 비용을 정확히 예측하기 어렵습니다. 요구사항 변경, 기술적 문제, 팀 역량 등 다양한 변수가 존재합니다.

분석과 설계의 중요성

분석의 정의

소프트웨어 개발을 의뢰한 고객이 요구하는 기능과 성능을 일관성 있게 정리하는 과정입니다. 이는 개발 방향을 명확히 하는 중요한 단계입니다.

설계의 정의

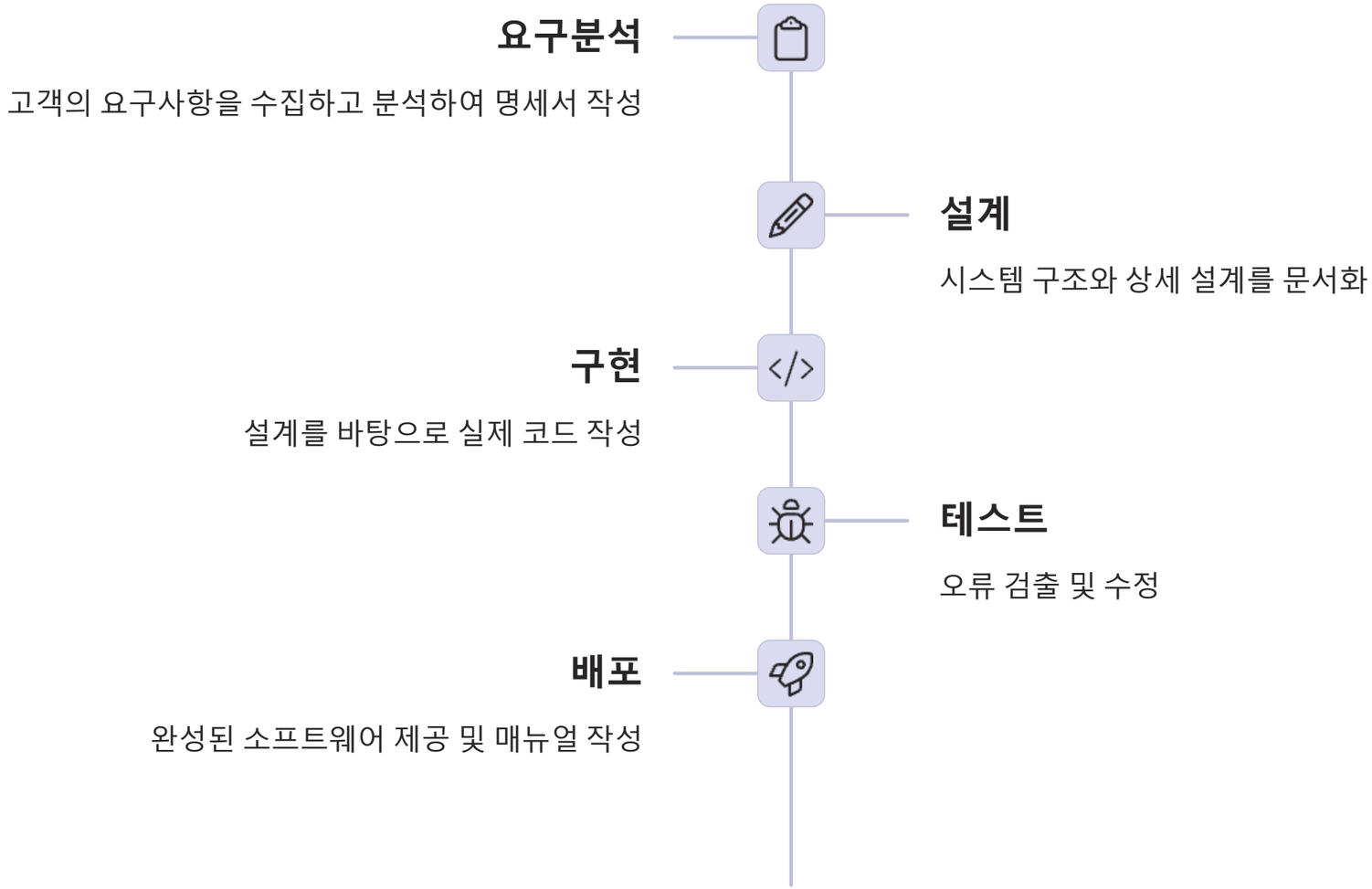
분석을 통해 밝혀진 요구사항을 어떤 형태로 프로그램에 반영할 것인지 결정하는 작업입니다. 이는 실제 구현의 청사진 역할을 합니다.

건축과의 비유

건축가처럼 계획하고 비전을 수립하며 요구를 이해하고 정리하는 작업이 필요합니다. 코딩을 바로 시작할 수 없는 이유가 여기에 있습니다.

분석과 설계는 시스템 전체에서 시작하여 점차 상세화되어 부분으로 진행됩니다. 구조적 분석 설계에서는 단계적 분할 방식을, 객체지향 분석 설계에서는 시스템 외부에서 내부로 상세화하는 방식을 사용합니다.

개발 프로세스와 결과물

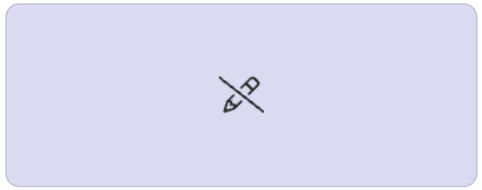


개발 프로세스는 소프트웨어를 만들기 위한 방법, 절차, 공정을 의미합니다. 이는 아이디어 고안부터 실제 소프트웨어로 실현될 때까지의 모든 개발 단계를 포함합니다. 각 단계에서는 요구분석 명세서, 설계서, 프로그램, 매뉴얼 등 다양한 결과물이 산출됩니다.

폭포수 모델

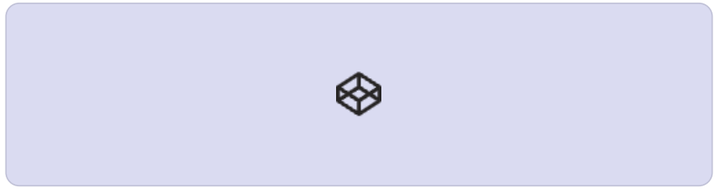
요구분석

시스템 요구사항 정의



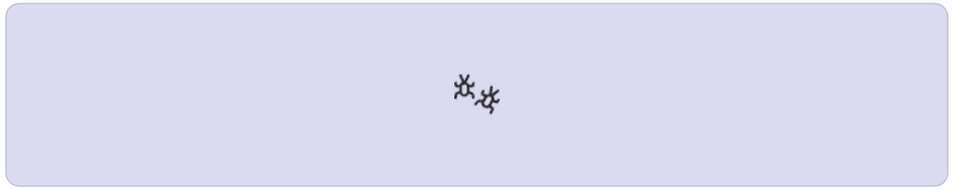
설계

시스템 및 소프트웨어 설계



구현

코딩 및 단위 테스트



테스트

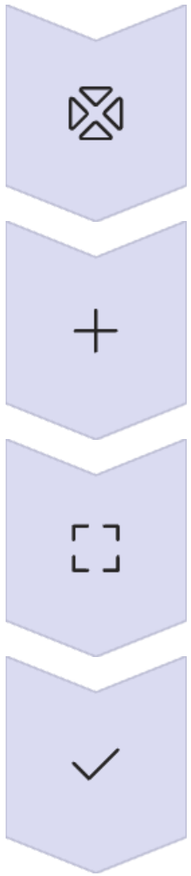
통합 및 시스템 테스트

유지보수

운영 및 유지관리

폭포수 모델은 가장 보편적이고 직관적인 개발 프로세스 모델입니다. 각 단계가 순차적으로 진행되며, 이전 단계가 완료되어야 다음 단계로 넘어갈 수 있습니다. 그러나 결함이 마지막 테스트 단계까지 발견되지 않는 경우가 많고, 분석과 설계의 오류 수정 비용이 많이 들며, 한 공정의 지연이 전체 일정에 영향을 미친다는 단점이 있습니다.

점증적 모델



버전 1 개발

핵심 기능 구현

버전 2 개발

추가 기능 구현

버전 3 개발

기능 확장

최종 시스템

완성된 제품

점증적 모델은 전체 시스템의 콘셉트를 정하고, 요구사항을 여러 버전으로 분할하여 순차적으로 개발하는 방식입니다. 중요하고 기초가 되는 요구사항을 버전 1로, 덜 중요한 사항을 버전 2, 3으로 나누어 개발합니다. 이 모델의 장점은 작동하는 소프트웨어를 조기에 생성할 수 있고, 유연성이 향상되며, 요구사항 변경에 대응하는 비용이 적다는 것입니다.

애자일 모델

유저 스토리

고객 요구사항 정리

스프린트 계획

단기 개발 목표 설정

일일 스크럼

진행 상황 공유

구현 및 릴리스

기능 개발 및 배포

회고 회의

프로세스 개선점 도출

애자일 모델은 신속한 개발이 가능한 프로세스로, 변화하는 고객의 요구를 유연하게 통합합니다. 프로세스와 도구보다 참여 개인 간의 교류를 중시하고, 광범위한 문서 작성보다 제대로 작동하는 소프트웨어 개발에 주력합니다. 실행되는 소프트웨어 기능을 고객에게 가치가 높은 것부터 순서대로 구현하여 확인하고, 단기간에 릴리스를 반복하는 것이 특징입니다.

프로세스, 방법, 원리

.

표준 소프트웨어 프로세스의 필요성

다른 문화를 가진 고객 및 개발 업체

의사소통에 불편을 없애기 위한 표준화

동일한 관점에서 이해

SW 개발의 작업 절차를 일관되게 파악

시행착오 감소

프로젝트 진행에서 불필요한 오버헤드를 줄임

프로젝트 산출물

개발 계획서

배경, 목적, 개요, 시스템의 범위, 기능

요구분석서

문제 정의, 시스템 제안서

기본 설계서

UI, 시스템 아키텍처, SW/HW 아키텍처, 네트워크 아키텍처

상세 설계서

모듈 내부구조, 데이터 구조, 네트워크 구조, 처리 흐름

세 가지 방법론

정보공학 방법론

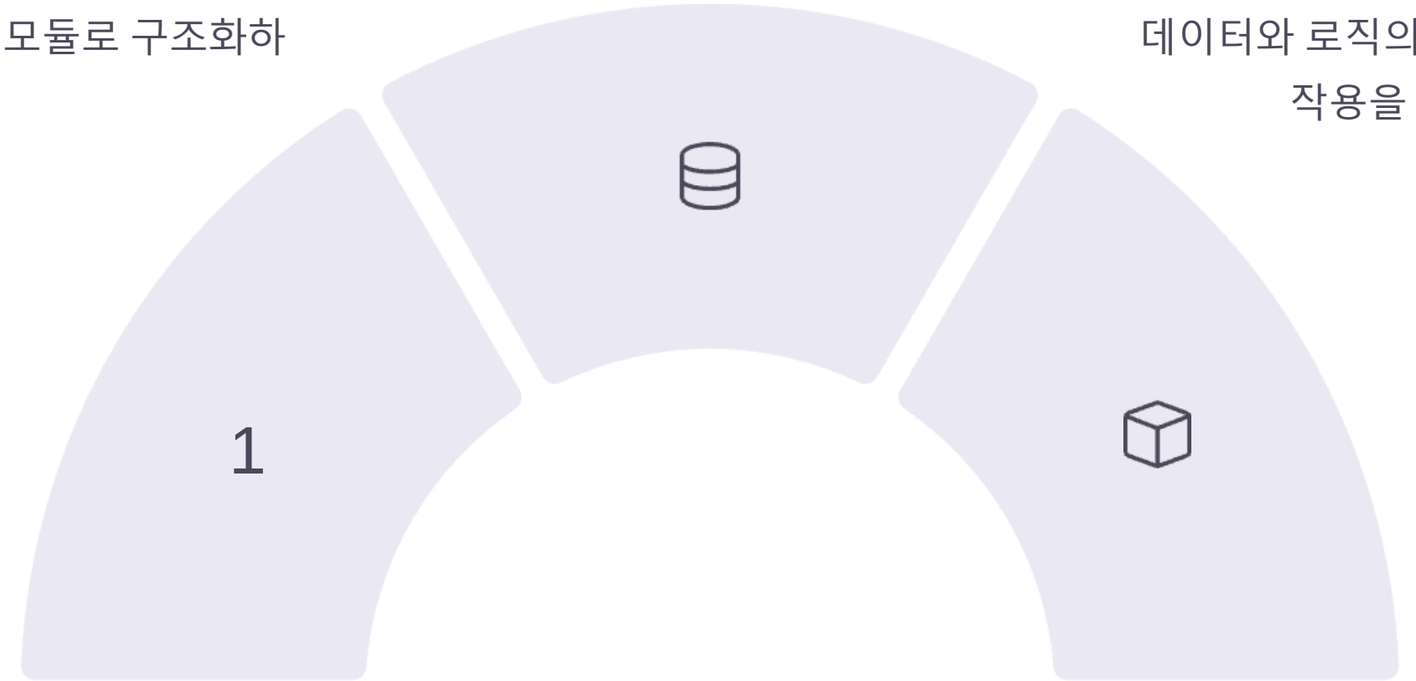
비즈니스 시스템 개발을 공학적으로 접근하는 데이터 중심 개발 방법

구조적 방법론

기능적 분해를 먼저 한 후 모듈로 구조화하는 방법

객체지향 방법론

데이터와 로직의 통합된 객체로 보고 상호작용을 모델링하는 방법



구조적 방법론과 정보공학 방법론

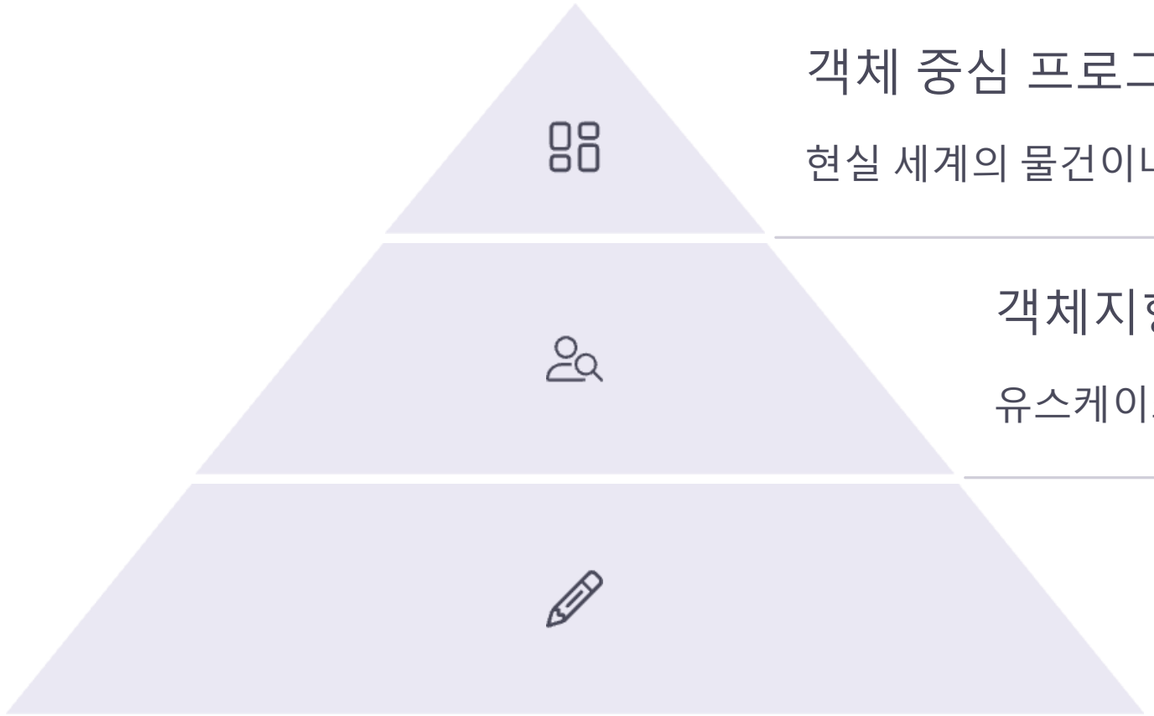
구조적 방법론

- 자료 흐름도(DFD) 활용
- 구조도로 모듈 관계 정의
- 데이터 처리에 주목
- 기능을 명확히 하고 구조 결정

정보공학 방법론

- 기업 중심 접근
- 전략적 시스템 계획 중심
- 데이터 중심 설계
- 분할과 정복 원리 적용

객체지향 방법론



객체 중심 프로그램

현실 세계의 물건이나 개념의 특징을 프로그램화

객체지향 분석

유스케이스 모델 구축, 도메인 클래스 및 관계 식별

객체지향 설계

클래스, 속성, 메서드 및 구조를 설계하고 구체화

As-Is와 To-Be

현재 상태(As-Is)

업무 프로세스의 현재 상태를 파악

문제 정의

차이(Gap) 분석

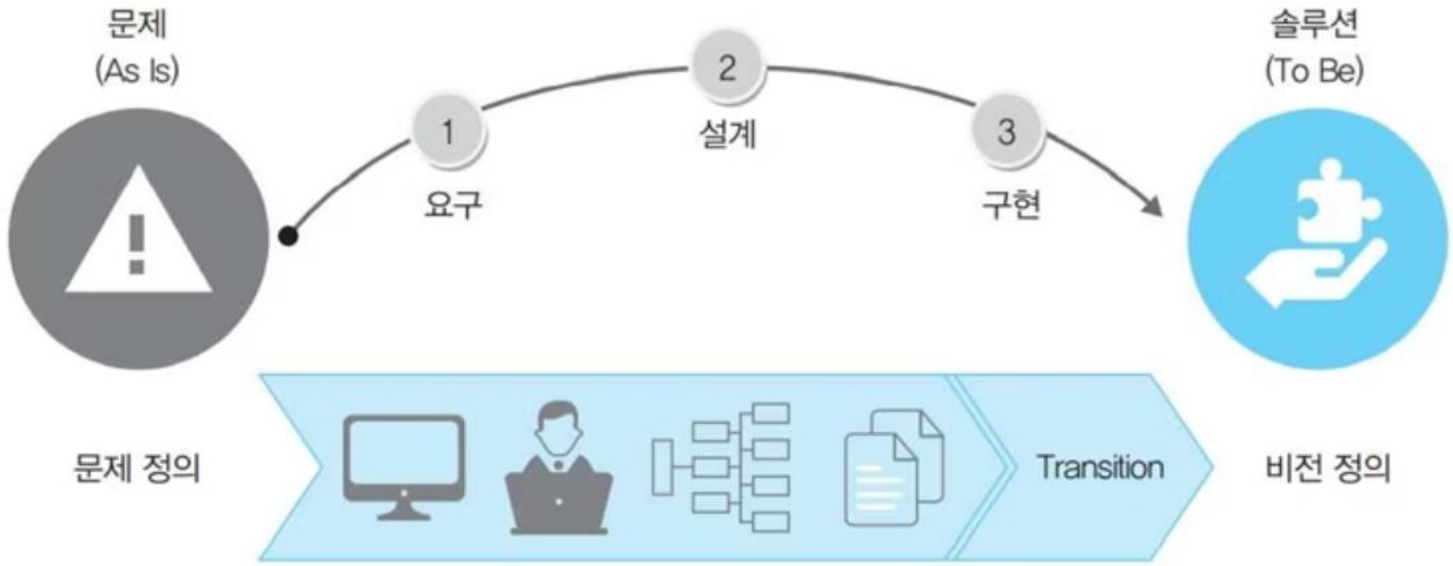
현재와 목표 사이의 차이점을 분석

목표(To-Be)

달성하고자 하는 목표 상태를 계획

솔루션 구축

차이를 해소하기 위한 솔루션 개발



다이어그램의 중요성

효율적 정보 전달

그림이 글보다 더 많은 정보를 집약적으로 표현할 수 있음

UML 다이어그램

객체지향 모델링을 위한 표준화된 다이어그램 언어

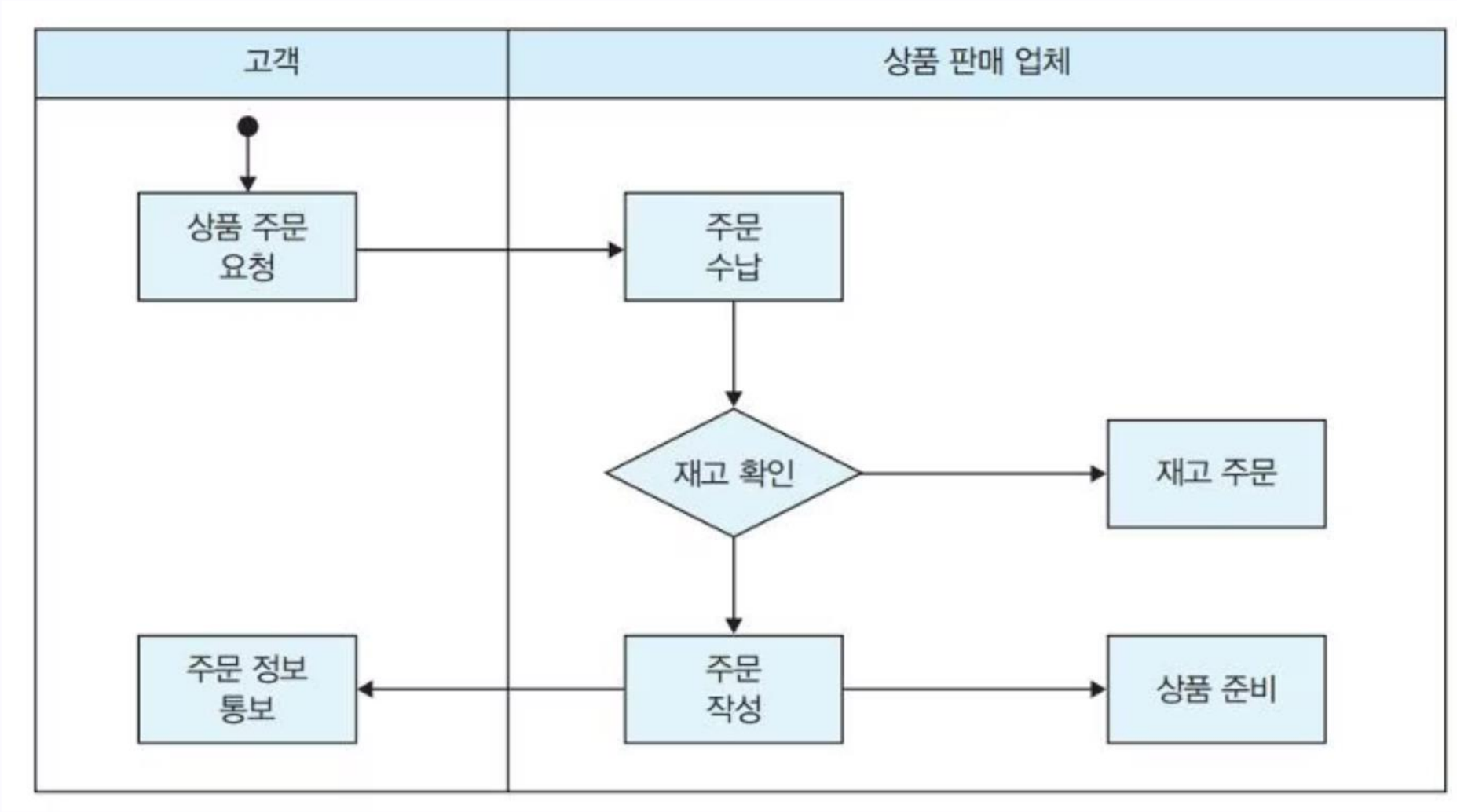
업무 흐름도

비즈니스 프로세스와 데이터 흐름을 시각적으로 표현

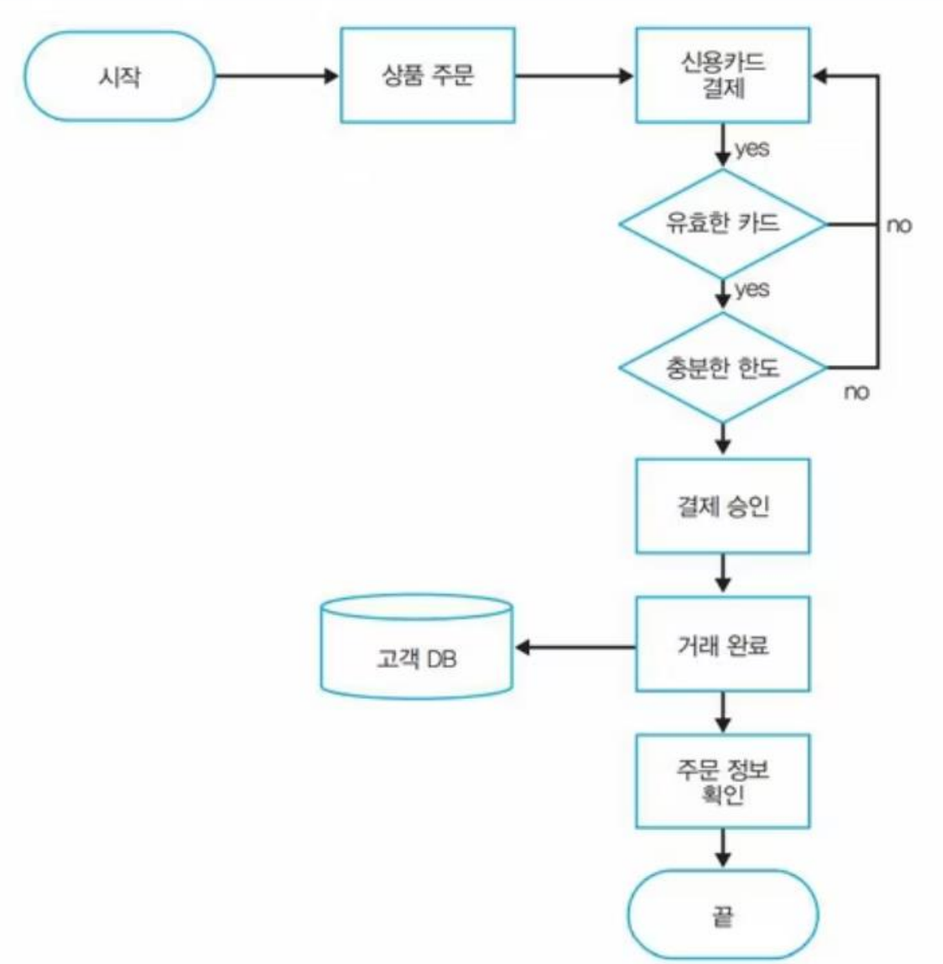
데이터 흐름도(Data Flow Diagram)

처리 프로세스와 데이터의 흐름을 간결하게 표현

업무 흐름도



순서도

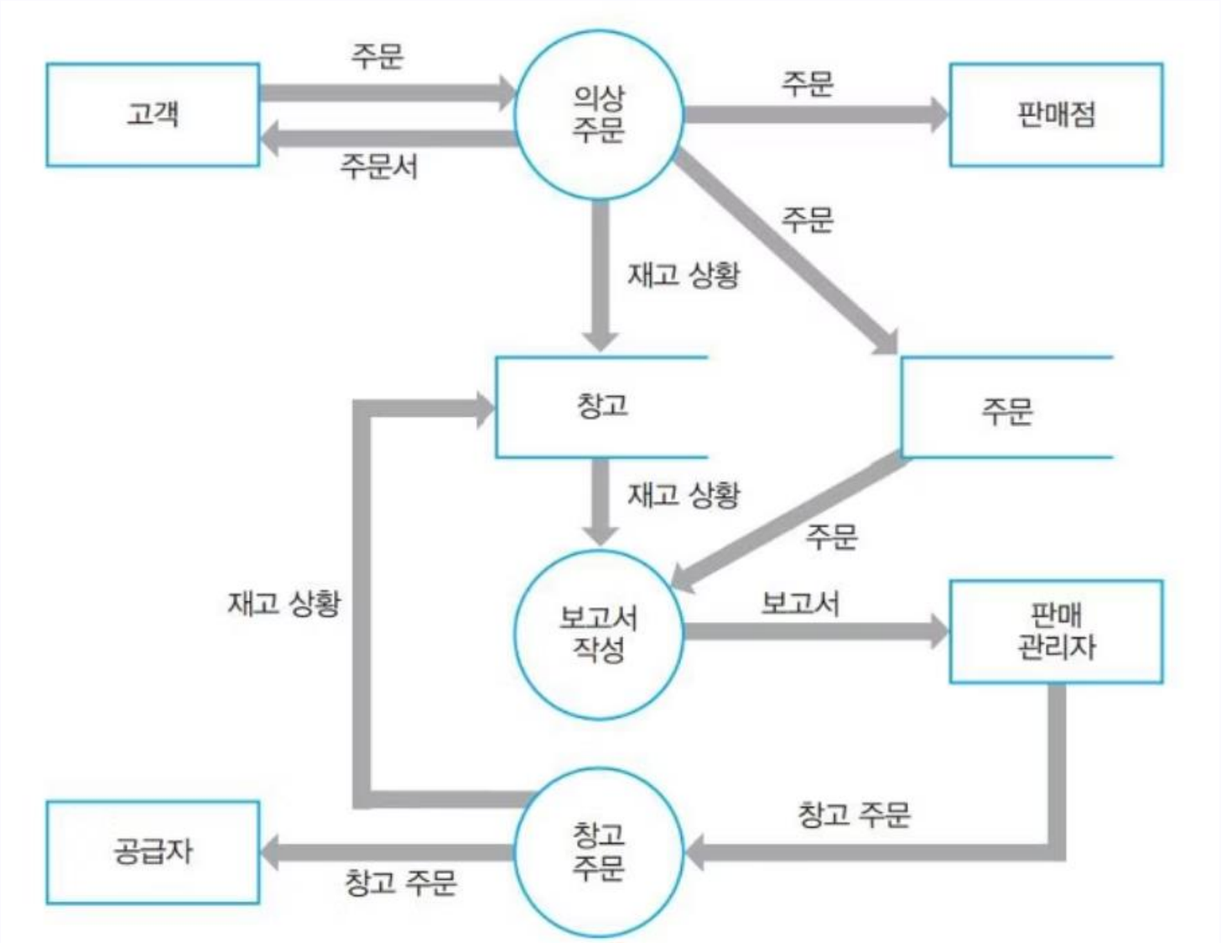


간결한 그림 기호를 사용

처리와 데이터의 흐름을 나타냄

데이타 흐름도 (DFD)

심볼	이름
→	데이터 흐름
1.x	프로세스
	저장소
	터미널



요구 분석

.

요구 분석 과정의 이해

1

요구 분석의 정의

구축하거나 수정할 소프트웨어에 대하여 사용자가 기대하는 것을 정의하는 프로세스로, 사용자와 관련자의 참여가 필요합니다.

2

사용자와의 협력

개발자는 사용자와 협력하여 무엇을 개발할지 논의하고, 이를 통해 요구 사항 명세서(Software Requirement Specification)라는 결과물을 도출합니다.

3

반복적 작업

요구 분석은 반복적이고 병렬적인 작업으로, 원하는 소프트웨어 시스템의 이해에 초점을 맞추며 분할과 정복의 원리 전략이 적용됩니다.

4

문제 영역 다루기

요구 분석은 문제 영역을 다루며, 설계는 해결 영역을 다룹니다. 이 과정에서 DFD, 유스 케이스 등의 기법이 적용됩니다.

요구의 종류와 특성

기능 요구

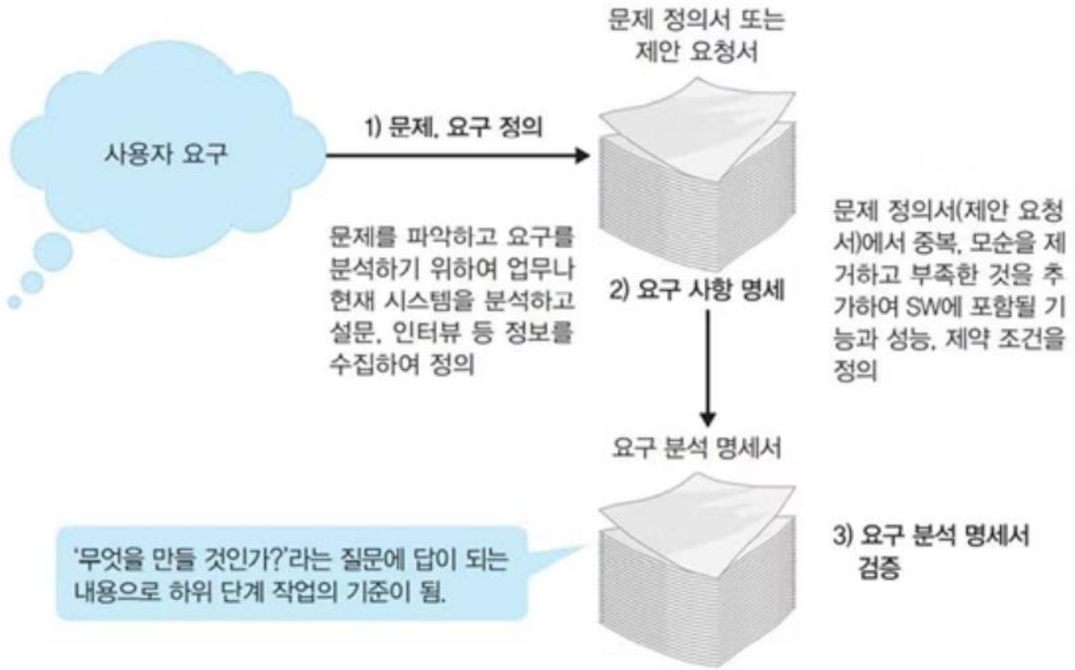
업무 처리와 실행에 필요한 절차를 의미합니다. 예를 들어 '고객의 이름을 등록', '고객의 이름을 주고 주소 찾기' 등 시스템에 요구되는 처리나 서비스를 말합니다.

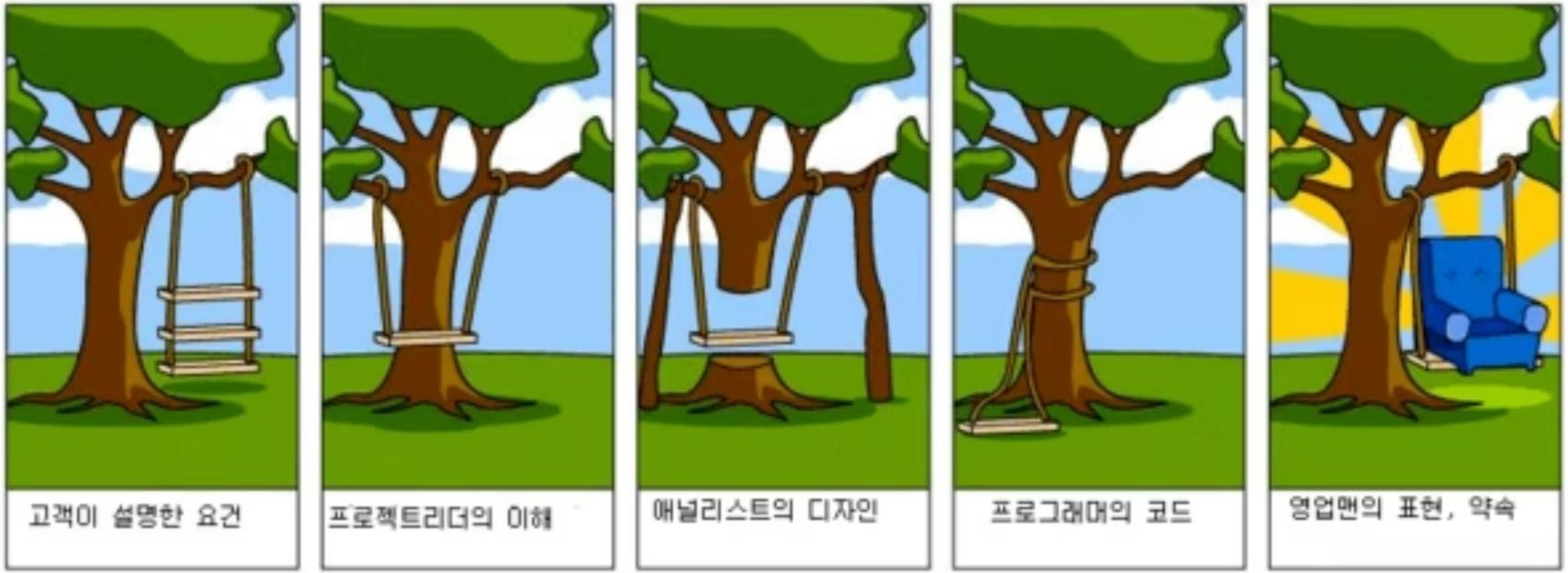
비기능 요구

상호운용성, 보안성, 신뢰성, 사용성, 효율성, 유지보수성 등이 포함됩니다. 이는 시스템이 갖추어야 할 품질적 특성으로, 시스템의 성능과 관련된 요구사항입니다.

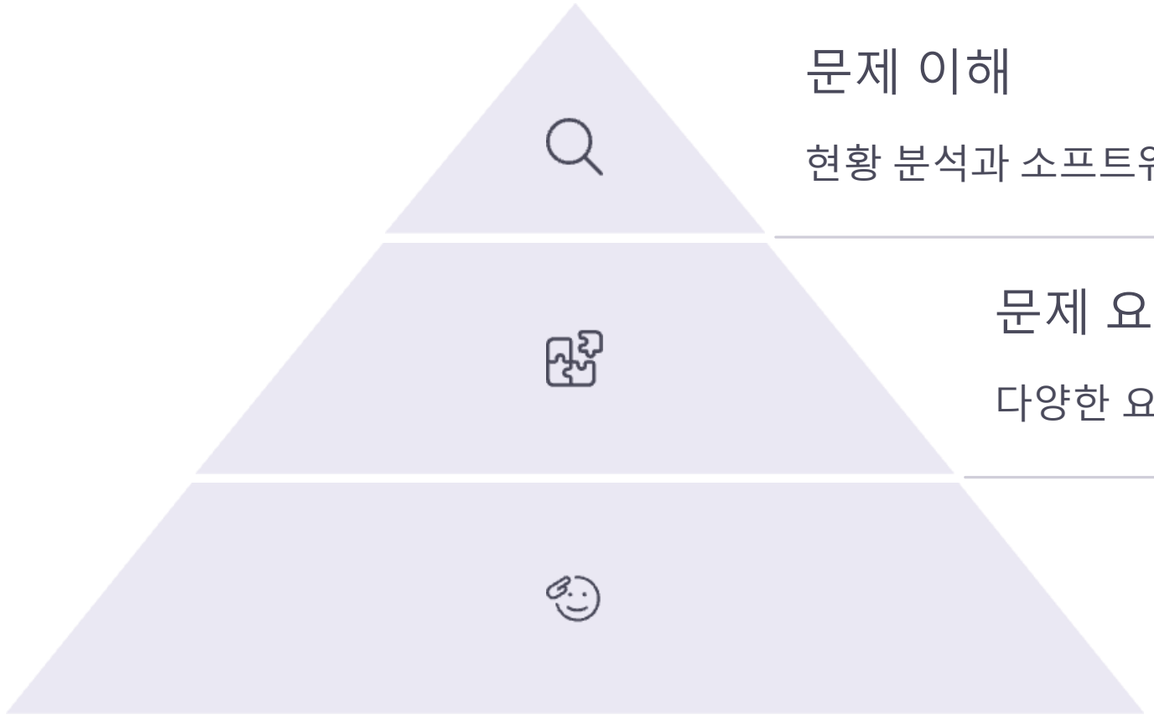
문제 정의와 요구 정의

문제 정의는 고객과 사용자가 원하는 요구와 해결하기 원하는 문제를 다루는 반면, 요구 정의는 소프트웨어 시스템으로 구현 가능한 요구와 해결 가능한 문제를 다룹니다.





문제 분석의 중요성



문제 이해

현황 분석과 소프트웨어 개발 대상 문제를 심층적으로 이해

문제 요소 파악

다양한 요소가 혼합된 문제의 전체 모습 파악

근본적 해결

소프트웨어 개발을 통한 근본적인 문제 해결 추구

문제 분석은 소프트웨어 개발의 핵심 단계로, 현황을 정확히 파악하고 개발 대상 문제를 심층적으로 이해하는 작업입니다. 소프트웨어 개발은 근본적인 문제 해결을 목표로 하기 때문에, 문제를 심층적으로 분석할 필요가 있습니다. 이 과정에서 이슈 트리와 같은 도구를 활용하여 문제의 구조를 체계적으로 파악할 수 있습니다.

기능 요구 분석법

시스템에 요구되는 처리나 서비스

업무 처리와 실행에 필요한 절

언어를 이용한 방법

장점: 배울 필요가 없고(고객이 쉽게 이해), 표현에 제한이 없음

단점: 정확도가 떨어지고, 모호할 수 있으며 도구가 없음

다이어그램을 이용한 방법

장점: 복잡한 문제나 해결책 이해에 도움이 되고, 커뮤니케이션이 용이하며, 구현에 이용할 수 있음

단점: 제한된 표현, 배워야 함

다이어그램 종류

- 흐름 기반 방법 - DFD, 순서도
- 시나리오 기반 방법 - 유스케이스 다이어그램, 액티비티 다이어그램
- 동작 기반 방법 - 상태 다이어그램, 시퀀스 다이어그램

비기능 요구(Non - Functional Requirements)

소프트웨어 기능들의 대한 조건과 제약 사항에 관한 요구사항

제품 요구사항(Product Requirement) 제품의 동작을 규정

조직 요구사항(Organizational Requirement) 고객과 개발자 조직의 정책과 절차

외부 요구사항(External Requirement) 시스템과 개발 프로세스의 외부 요소로 부터 생긴 모든 요구사항

		비기능적 요구사항
제품 요구사항 (Product Requirement)	사용성(Usability) : 사용자가 어떻게 쉽게 사용할 수 있는가	
	효율성(Efficiency)	성능(Performance) : 특정 기능이 특정시간 내에 실행
		공간 (Space) : 특정 기능 수행시 메모리를 최대 얼마까지 사용할 수 있는가
	신뢰성(Reliableility) : 특정 기능 실행시 실패할 가능성이 몇 %보다 낮아야 하는가	
	이식성 (Portability) : 다양한 플랫폼 위에서 작동하는가	
조직 요구사항 (Organizational Requirement)	배포(Delivery) : 소프트 웨어를 어떻게 배포할 것인가	
	구현(Implement) : 소프트웨어 구현 ; 어떤 방법론? 어떤 프로그래밍 언어?	
	표준(Standard) : 소프트웨어 개발 시 어떤 표준을 따를 것인가	
외부 요구사항 (External Requirement)	상호 운용성 (Interoperability) : 구현할 소프트웨어가 다른 소프트웨어와 어떻게 연동할지 정의	
	윤리적 (Ethical) : 소프트 웨어의 내용의 윤리적 범위을 정의 ex) 성인용 게임 19세 이상	
	법적 (Legislative)	사생활(privacy) ex) 공개범위 선택
		안전성(safety) ex) 자료 저장방식, DBMS 어떤것? 자료의 암호화 여부

요구 분석 명세서 작성

명세서 포함 내용

시스템의 목표와 범위, 기능과 비기능 요구, 프로젝트의 제약 조건, 도입 전환 계획, 자료 흐름도나 유스케이스 명세서 등이 포함되어야 합니다.

명세서 작성 원칙

작업을 완료할 수 있게 충분히 상세하게 작성하되, 구현 독립적이어야 하며, 현존하지 않는 미래 기술에 의존해서는 안 됩니다. 또한 현재 상황(예산, 시간, 기술)에 맞게 실현 가능해야 합니다.

필요 충분성과 책임 분담

해야 할 것뿐만 아니라 하지 않아야 할 것도 명확히 기술하고, 기능과 성능을 정량적으로 작성해야 합니다. 또한 책임 분담을 명확히 하여 역할을 분명히 해야 합니다.