1. 요구사항 확인

> 요구사항 확인 단원은 소프트웨어 개발 방법론, 현행 시스템 분석, 요구사항 확인으로 구성되어 있습니다. 핵심 과목인 소프트웨어 개발 방법론에서 애자일 방법론, 객체 지향 분석 방법론, 프로젝트 관리 등은 반복 학습으로 고득점을 노려봅니다.

소프트웨어 생명주기(SDLC)

시스템의 전 공정을 체계화한 절차

SDLC 모델 종류

- 폭포수 모델 : 각 단계를 확실히 마무리 지은 후에 다음 단계로 넘어감, 선형 순차적 모형(고전적 생명주기 모형)
- 프로토타이핑 모델 : 프로토타입을 구현해, 고객의 피드백을 반영하며 만들어 간다
- 나선형 모델 : 위험을 최소화하기 위해 점진적으로 개발
- 반복적 모델 : 구축 대상을 나누어 병렬적으로 개발 후 통합하거나, 반복적으로 개발(SDLC 모델)

소프트웨어 개발방법론

- 구조적 방법론 : 전체 시스템을 기능에 따라 나누어 개발하고, 이를 통합.(하향식 방법론) 나씨-슈나이더만 차트 사용(도형식, 제어 논리 구조, 명확한 식별)
- 정보공학 방법론 : 정보시스템 개발에 필요한 관리 절차와 작업 기반을 체계화
- 객체지향 방법론 : 복잡한 현실 세계를 사람이 이해하는 방식으로 시스템에 적용
- 컴포넌트 기반 방법론(CBD) : 컴포넌트를 조립해 하나의 새로운 응용 프로그램 작성(생산성, 확장성, 재사용)
- 애자일 방법론 : 절차보다는 사람이 중심, 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템 개발
- 제품 계열 방법론 : 특정 제품에 적용하고 싶은 공통된 기능을 정의해 개발, 임베디드에 유용

애자일

- XP : 의사소통 개선과 즉각적 피드백
  - 5가지 가치 : 용기, 단순성, 의사소통, 존중, 피드백
  - 12가지 기본 원리
    - 짝 프로그래밍(Pair Programming) : 개발자 둘이서 짝 코딩
    - 공동 코드 소유(Collective Ownership) : 시스템 코드는 누구든지 언제라도

수정 가능
- 지속적인 통합(Continuous Integration) : 매일 여러 번씩 SW통합, 빌드 해야함
- 계획 세우기(Planning Process) :개발자가 필요한 것은 무엇이며 어떤 부분에서 지연될 수 있는지를 알려줘야함
- 작은 릴리즈(Small Release) : 작은 시스템 먼저 만들고 짧은 단위 업데이트
- 메타포어(Metathor) : 공통적인 이름 체계와 시스템 서술서를 통해 고객과 개발자간의 의사소통을 원활하게
- 간단한 디자인(Simple Design): 요구사항에 적합한 가장 단순한 시스템 설계
- 테스트 기반 개발(Test Driven Develop)
- 리팩토링(Refactoring) : 프로그램의 기능은 바꾸지 않고 중복제거, 단순화 등을 위해 시스템 재구성
- 40시간 작업 : 40시간 이상 일 X
- 고객 상주(On Site Customer) : 개발자들의 질문에 대답할 수 있는 고객 풀타임 상주
- 코드 표준(Coding Standard) : 효과적인 공동 작업을 위해 코딩 표준을 정의

- SCRUM : 매일 정해진 시간, 장소에서 짧은 시간의 개발
  - 백로그(Backlog) : 제품과 프로젝트에 대한 요구사항
  - 스프린트(Sprint) : 2~4주의 짧은 개발 기간 반복적 수행
  - 스크럼 미팅(Scrum Meeting) : 매일 15분 정도 미팅
  - 스크럼 마스터(Scrum Master) : 프로젝트 리더
  - 스프린트 회고(Sprint Retrospective) : 스프린트 주기 되돌아보며 규칙 준수 여부, 개선점 확인
  - 번 다운 차트(Burn Down Chart)
- LEAN : 도요타, 낭비 요소를 제거하여 품질 향상
  - 낭비제거, 품질 내재화, 지식 창출, 늦은 확정, 빠른 인도, 사람 존중, 전체 최적화

비용산정 모형

- 하향식
  - 델파이 기법 : 전문가의 경험적 지식을 통한 문제 해결 및 미래예측을 위한 기법
- 상향식
  - LoC(Lind of Code) : 원시 코드 라인 수의 낙관치, 중간치, 비관치를 측정해 예측치를 구해 비용을 산정하는 방식
  - Man Month : 한 사람이 1개월 동안 할 수 있는 일의 양을 기준으로 프로젝트 비용 산정하는 방식
    - 프로젝트 기간 = man month(LoC/프로그래머 월간 생산성)/프로젝트 인력
  - COCOMO : 보헴이 제안, 프로그램 규모에 따른 비용 산정
    - 조직형(Organic Mode) : 5만 라인 이하
    - 반 분리형(Semi-Detached Mode) : 30만 라인 이하
    - 임베디드형(Embedded Mode) : 30만 라인 이상

    - 푸트남 : 개발주기의 단계별로 요구할 인력의 분포를 가정하는 방식, 생명주기 예측 모형, Rayleigh-Norden 곡선
    - 기능점수(FP) : 요구 기능에 따른 가중치 부여

일정관리 모델

- 주 공정법(CPM) : 여러 작업의 수행 순서가 얽혀 있는 프로젝트의 일정 계산(임계경로는 가장 오래 걸리는 경로)
- PERT : 일의 순서를 계획적으로 정리하기 위한 수렴 기법, 비관치, 중간치, 낙관치의 3점 추정방식 이용
- 주 공정 연쇄법(CCPM) : 자원제약사항을 고려해 일정 작성

현행 시스템 파악

구성/기능/인터페이스 파악 -> 아키텍처 및 소프트웨어 구성 파악 -> 하드웨어 및 네트워크 구성 파악

소프트웨어 아키텍처

여러 가지 소프트웨어 구성요소와 그 구성요소가 가진 특성 중 외부에 드러나는 특성, 그리고 구성요소 간의 관계를 표현하는 시스템의 구조나 구초체

소프트웨어 아키텍처 4+1 뷰

고객의 요구사항을 정리해 놓은 시나리오를 4개의 관점에서 바라보는 소프트웨어적인 접근방법

- 유스케이스 뷰 : 유스케이스 또는 아키텍처 도출, 다른 뷰를 검증하는데 사용
- 논리 뷰 : 시스템의 기능적인 요구사항
- 프로세스 뷰 : 시스템의 비기능적 요구사항
- 구현 뷰 : 모듈의 구성을 보여줌
- 배포 뷰 : 어떻게 배치되는가

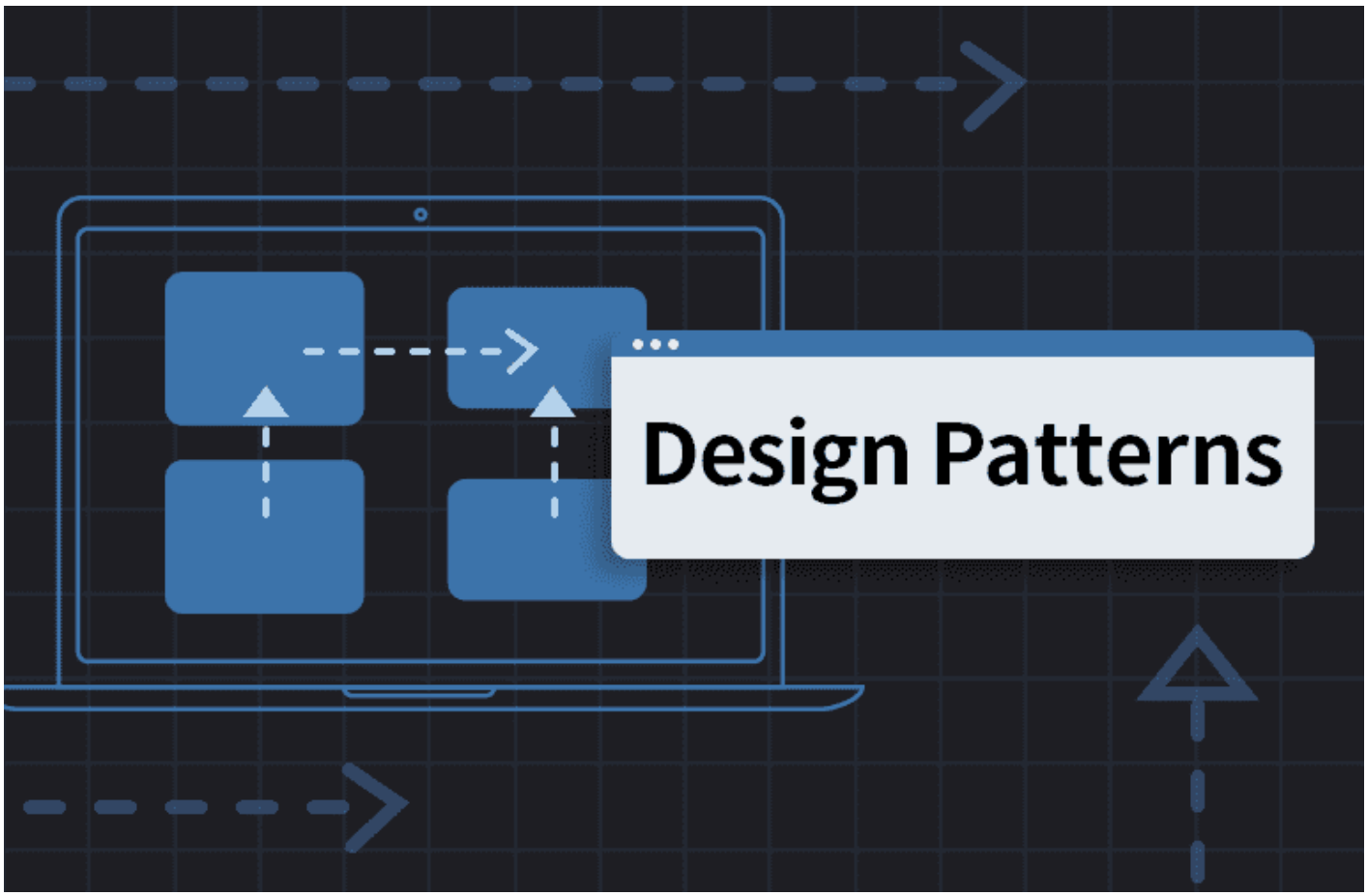
소프트웨어 아키텍처 패턴 유형

- 계층화 패턴(Layered) : 서로 마주 보는 두 개의 계층 사이에서만 상호작용
- 클라이언트-서버 패턴 : 하나의 서버와 다수의 클라이언트
- 파이프-필터 패턴 : 데이터 스트림을 생성하고 처리하는 시스템에서 사용
- 브로커패턴 : 분리된 컴포넌트들로 이루어진 분산 시스템에서 사용되고, 원격 서비스 실행을 통해 상호작용이 가능
- MVC패턴
    - 모델 : 핵심 기능, 데이터 보관
    - 뷰 : 사용자에게 정보 표시

- 컨트롤러 : 사용자로부터 요청 입력 받아 처리

소프트웨어 아키텍처 비용 평가 모델 종류

- SAAM : 변경 용이성, 기능성에 집중
- ATAM : 아키텍처 품질 속성을 만족시키는지 판단
- CBAM : 경제적 의사결정에 대한 요구를 충족하는지
- ADR : 응집도 평가 모델
- ARID : 특정 부분 품질 요소



디자인 패턴 ⭐ 매우 중요 ! ⭐

SW설계에서 공통으로 발생하는 문제에 대해 자주 쓰이는 설계 방법을 정리한 패턴

- 생성 bprofas 비프로파스
  - builder : 복잡한 인스턴스를 조립해 만드는 구조
  - prototype : 처음부터 일반적인 원형을 만들어 놓고, 그것을 복사한 후 필요한 부분만 수정해 사용하는 패턴
  - factory method : 상위 클래스에서 인터페이스 정의, 하위클래스에서 인스턴스 생성

    - abstract factory : 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공
    - singleton : 전역 변수 사용하지 않고 객체 하나만 생성, 그 객체는 어디서든지 참조할 수 있음
- 구조 abcdffp
    - adapter : 기존에 생성된 클래스를 재사용할 수 있도록 중간에서 맞춰주는 역할
    - bridge : 기능 계층과 구현 계층을 연결, 구현부에서 추상 계층 분리
    - composite : 객체들의 관계를 트리 구조로 구성
    - decorator : 기존에 구현되어 있는 클래스에 필요한 기능 추가해 나감
    - facade : 복잡한 시스템에 대해 단순한 인터페이스 제공, 시스템 구조에 대한 파악 쉽게
    - flyweight : 메모리 절약, '클래스의 경량화'목적
    - proxy : 실체 객체에 대한 대리 객체
- 행위
    - Mediator : 중간에 통제, 중재자
    - Interpreter : 언어의 다양한 해석, 구문의 해석을 맡는 클래스 각각 작성
    - Iterator : 컬렉션 구현 방법 노출시키지 않으면서도 그 집합체 안에 들어있는 모든 항목에 접근할 방법을 제공
    - Template Method : 상위 클래스-추상, 하위 클래스-구체
    - Observer : 한 객체의 상태가 바뀌면 그 객체에 의존하는 다른 객체들에 연락
    - State : 상태에 따라 다르게 처리할 수 있도록 행위 내용 변경
    - Visitor : 클래스의 메서드가 각 클래스를 돌아다니며 특정 작업 수행
    - Command : 명령이 들어오면 그에 맞는 서브 클래스 선택되어 실행
    - Strategy : 알고리즘 군 정의, 행위를 클래스로 캡슐화해 동적으로 행위 자유롭게 변환
    - Memento : Undo(작업취소) 기능 개발
    - Chain of Responsibility : 정적으로 어떤 기능에 대한 처리의 연결이 하드 코딩되어 있을 때, 이를 동적으로 연결되어 있는 경우에 따라 다르게 처리될 수 있도록 연결한 디자인

운영체제

컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스 담당

- 윈도즈 : 중/소규모 서버, 일반 pc
- 유닉스 : 대용량 처리, 엔터프라이즈 급 서버
- 리눅스 : 중/대규모 서버 대상, 높은 보안성, 비용 가장 적음
- 안드로이드 : 리눅스 위에서 구동, 자바와 코틀린으로 작성
- IOS : 높은 보안성, 고성능

운영체제 현행 시스템 분석 고려사항(신성기주구)

신뢰도, 성능, 기술 지원, 주변 기기 , 구축 비용

미들웨어

응용 프로그램과 환경 간에 원만한 통신이 이루어질 수 있도록 제어해주는 SW

- WAS : 서버계층에서 애플리케이션이 동작할 수 있는 환경 제공, 트랜잭션 처리, 이기종 시스템 연동

요구공학

사용자 요구사항에 대한 도출, 분석, 명세 , 확인 및 검증하는 구조화된 활동

요구사항

- 기능적 요구사항 : 시스템이 제공하는 기능, 서비스에 대한 요구사항(사용자UI)
- 비기능적 요구사항 : 시스템이 수행하는 기능 이외의 사항(백엔드)

요구사항 개발 단계(도분명확)

- 도출
  - 인터뷰 : 이해 관계자와 직접 대화
  - 설문조사 : 설문지, 여론조사
  - 브레인스토밍 : 말을 꺼내기 쉬운 분위기로 만들어 비판 없이 수용할 수 있도록 하는 회의
  - 델파이 기법 : 전문가의 경험적 지식을 통한 문제 해결 방법
  - 롤 플레잉 : 여러 사람이 각자가 맡은 역을 연기
  - 워크숍 : 단기간에 다양하고 전문적인 정보를 획득하고 공유
- 분석
  - 청취 기술
  - 인터뷰와 질문 기술
- 명세
  - 비정형 명세 기법
    - 자연어 기반
    - 사용자와 개발자 이해 용이
    - 명확성 및 검증 문제
  - 정형 명세 기법
    - 수학적인 원리와 표기법, Z-스키마, Petri Nets
    - 표현 간결, 명확성 및 검증 용이
    - 기법 이해 어려움
- 확인 및 검증
  - 정형 기술 검토
    - 동료 검토 : 2~3명이 진행, 작성자가 명세서 설명하고 이해관계자들이 설명을 들으면서 결함 발견
    - 워크스루 : 검토자료를 회의 전에 배포해서 사전검토한 후 짧은 시간동안 회

의 진행
- 인스펙션 : 저작자 외의 다른 전문가 또는 팀이 검사하여 오류를 찾아내는 공식적 검토 방법