정보처리기사 1장 요구사항 확인

Chapter 01 소프트웨어 개발 방법론

1 소프트웨어 개발 방법론

(1) 소프트웨어 생명주기 모델

1.소프트웨어 생명주기
요구분석 ~ 유지보수까지의 전 공정을 체계화한 절차

보통은
요구분석 -> 설계 -> 구현 -> 테스트 -> 유지보수 로 이뤄짐.

우리가 개발할 때를 생각해볼 것.

입학상담앱을 만든다.

1. 기획
2. 개발
3. 테스팅
4. 유지보수 임

위와 같은 것임

2.소프트웨어 생명주기 모델의 종류

- 폭포수 모델
- 프로토타이핑 모델
- 나선형 모델
- 반복적 모델

"우리 회사는 어떤 모델을 주로 쓰는지도 생각해보면 재밌을 것임"

폭포수모델

> "폭포수는 떨어졌다가 다시 올라가지 않는다."

- 각 단계 확실히 마무리 그 다음 단계
- 고전적

- 당연히 경험, 사례 많음
- 정의 산출물 명확
- 요구사항 변경은 어려움

```
절차 :
타당성 검토 -> 계획 -> 요구사항 분석 -> 설계 -> 구현 -> 테스트 ->   유지보수
```

위의 절차는 조금 세분화 된 것일 뿐 결국엔 같음

프로토타이핑 모델

지금 상담앱이 이런 방식으로 하려고 하고 있음.

프로토타입(구현 골격)을 만들고, 고객의 피드백을 반영하는 방식

발주자, 개발자 모두에게 참조모델이 될 수 있음.

나선형 모델

```
절차 :
계획 및 정의 -> 위험 분석 -> 개발 -> 고개 평가 -> 계획 및 정의
```

폭포수와 차이, 위험을 분석하며 개발하는 것과 평가 후 계획으로 다시 돌아간다는 점

퀄리티가 올라가게 되어있음.

사실 나선형 모델과 프로토 타입 모델의 차이는 그리 명확지 않다고 본다.

그러나 프로토타입은 그래도 하나의 산출물이 나오는 반면
나선형 모델은 조금 더 사용자의 평가를 자주 듣고 반영하는 느낌이 강한듯

반복적 모델

> 협업을 한다면 반복적 모델!

병렬 개발 후 통합(머지)

기능의 세분화 하여
작은 기능을 완성하고 병합하는 식의 반복

예를들면

상담앱의 경우

- 성적산출

- 산출 결과
- 상세 페이지
- 그외 설정 페이지

위와 같이 기능이 나눠져 있음.

또 그걸 나눔.
성적 산출이면

- 상단 바
- 사이드 바
- 메인화면

이를 나눠서 개발
그리고 또 산출결과 개발..
이런식으로 진행해 나가는 것이 반복적 모델

** 여기까지 해서 가장 중요하다 생각되는 건
폭포수 모델이랑 나선형 모델 정도일 듯.

(2) 소프트웨어 개발방법론

1. 소프트웨어 개발 방법론

위의 생명주기 전역에 걸쳐서 어떤 방법으로 개발할 것이냐 하는 것임

결국에는 모델에 따라서 방법론이 달라지고,
모델과 방법론은 비슷하게 따라가는 것이라 봐도 될 것 같음.

2. 종류

- 구조적 방법론
- 정보공학 방법론
- 객체지향방법론
- 컴포넌트 기반 방법론
- 애자일 방법론
- 제품 계열 방법론

구조적 방법론

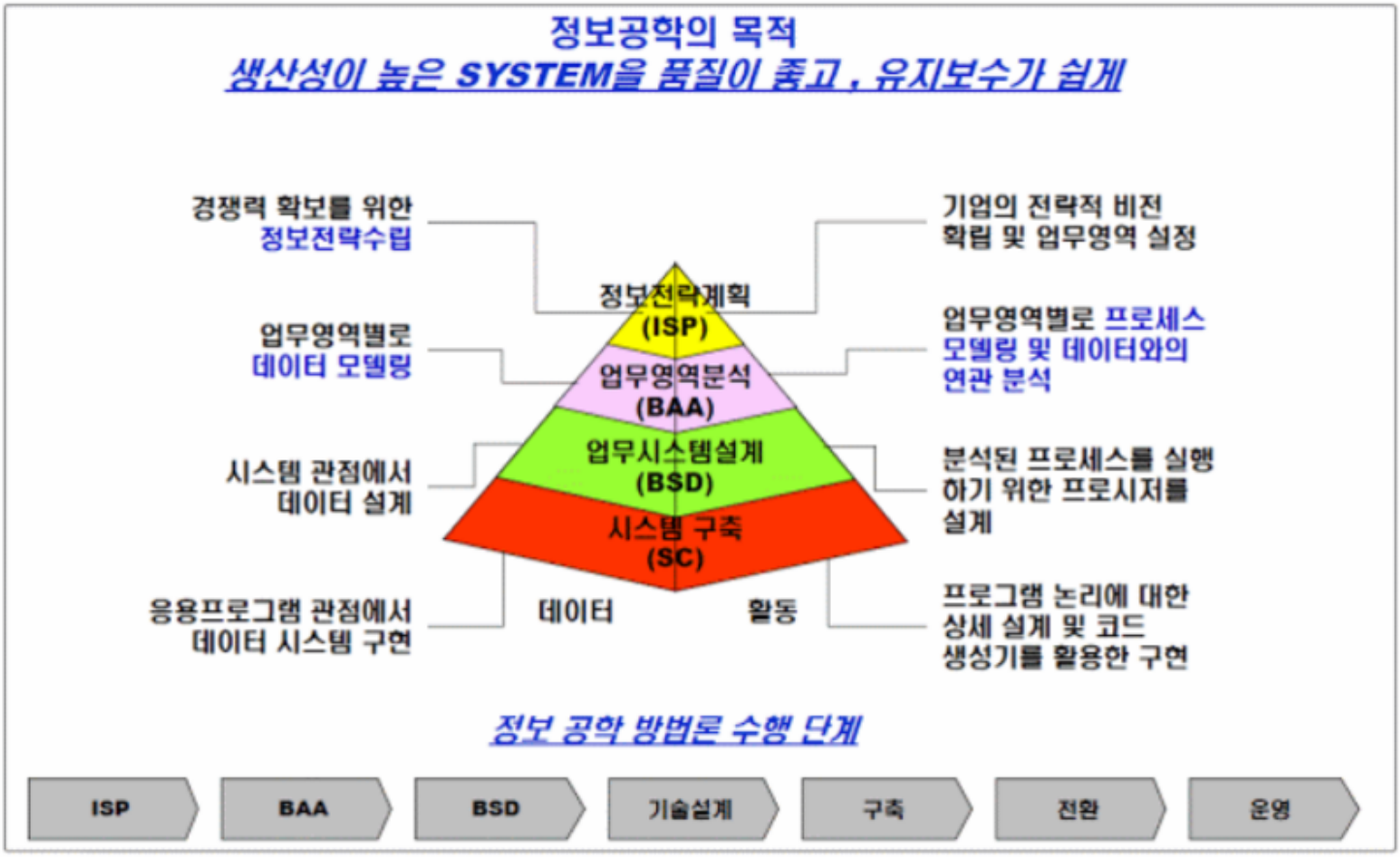
기능에 따라 구조를 나누고 개발함.
위의 반복적 모델과 긴밀한 관계가 있는듯함.
하향식 방법론 (위의 입학상담앱 예시를 참고)

※ 나씨 슈나이더만 차트를 사용

구조적 프로그래밍 표현으로 한눈에 알아보기 쉽게 하기 위함.

- 논리적 도표
- 조건이 복합될 경우 사용하면 좋음

정보공학 방법론



그리 중요할 것 같은 방법론은 아니지만

체계적이라는 것이 중요한 것 같고
그래서 대규모 프로젝트에 사용된다고 하였다.

가장 대표적으로 생성툴 이야기를 하고 싶다.

생성툴은 굉장히 대규모 프로젝트였다
그리고 그 효과는 굉장했고.

그러나 그 과정에서는 체계적인 분석과 계획이 필요하였다.

이를 팀장님이 해내시기까지는 시간이 걸렸다.

- 많은 운영자와의 회의를 통한 계획 수립
- 업무 영역 분석
- 이를 구현하기 전의 설계 (JX 프레임워크 설계 등)
- 시스템 구축 (실제 개발)

작은 프로젝트에서는 절대 사용할 수 없을 것임

객체 지향 방법론

너무나 유명한 방법론임.

클래스를 통해 객체를 만들면
객체에서의 동작을 정의하여
여러가지 활동들을 할 수 있다는 것임.

예를 들면 우리의 결제 모듈을 이야기할 수 있음

- 내국인 결제
- 외국인 결제
- 장학금
- 면제 등등

이를 하나의 결제 객체를만들어 조금씩 수정하여 사용한다면 될 것임

!!객체지향분석방법론
객체 지향 방법론을 사용하기 위해선 모델링이 필수적으로 필요하다

객체, 클래스, 속성 등의 정의와 각 객체간의 연관 관계를 정의하는 것이 필요하기 때문임.

그 종류로는

- OOSE (Object Oriented Software Engineering)
  야콥슨이 만들었음
  유스케이스 모델을 사용함
- OMT (Object Modeling Technology)
  럼바우가 만들었음

그래픽 표기법을 사용함
객체모델링 -> 동적모델링 -> 기능모델링 순서로 하는데
객체(정보)모델링 : 객체들간의 관계 **ER 다이어그램**을 만들기까지의 모델링
동적 모델링 : 시간의 흐름에 따른 동작 순서 **상태 다이어그램**
기능 모델링 : 자료 흐름을 중심으로 **자료흐름도**를 활용하여 표현

- OOD (Object Oriented Design)
  부치가 만들었음
  설계 문서화를 강조함.

이렇게 세가지가 있는데 시험이 나왔다니까 알아둬야할 듯!

컴포넌트 기반 방법론

내가 개인적으로 최근 상담앱 개발을 하면서 많이 사용한 방법론

- 코드가 깔끔해짐
- 개발 기간이 단축됨
- 새로운 기능 추가 쉬움
- 재사용이 가능

다 좋은 얘기 밖에 없음
그리고 다 같은 얘기이긴 함.
편하다

애자일 방법론

절차보다 사람중심!

내가 옛날 회사에서는
뭐 하나를 개발하려고 하면 문서를 많이 만들어야 했음

클래스 구조 문서
프로토콜 데이터 문서 등등

그러나 여기서는 절차가 아닌
사람 중심으로 빠르게 진행하자가 특징인듯하다.

폭포수 모델과는 대립되고
나선형 모델과 유사한 것 같다

안정성이 중요한 대형 시스템에서는 폭포수 모델이 낫겠지만
빠르게 개발을 해야한다면 나선형 모델이 좋겠다는 생각이 든다.

애자일 방법론의 유형

- XP (Extreme Programming)
- 린
- 스크럼

1.XP

익스트림한 개발을 하겠다.(도전적으로 품질을 높이겠다는 것인데..)

1~3주의 반복 개발주기

5가지 가치

- 용기
  코드 작성 전 테스트 모듈을 먼저 만듬
  빠른 피드백
  테스트에 부합하지 못하는 코드를 리팩토링할 수 있는 용기...

아.. 이거 진짜 별로인듯...
테스트하느라 시간 다 잡아먹음...
신속한 개발을 하겠다 하고 이게 핵심가치라면 절대 안될 것 같은데..
(개인 생각임)

- 단순성
  필요한 것만 하고, 그 이상의 것들은 하지 않음.
  예를 들면... 필요하지도 않은 기능을 쓸데 없이 개발한다던지 하는 것들..
- 의사소통
  개발자, 관리자, 고객 3자간의 의사소통 (이건 진짜 중요하지)
  위의 단순성에서 필요하다의 기준은 여기서 결정될 것임
- 피드백
  의사소통에 따른 피드백 (단순성, 의사소통, 피드백 다 이어지는 거네...)
- 존중
  팀원 간의 존중
  다 이어지고 있음. 서로 얘기를 많이하려면 필요함.

12가지 기본원리

- 짝프로그래밍
  개발자 둘이서 짝으로 코딩한다.
  (난 이게 시간 다잡아먹는다고 본다.)
  나 진짜 물어보고 싶음 이거의 장점이 뭐임?

- 공동 코드 소유
  시스템에 있는 코드는 누구든지 수정 가능하다

우리회사에서 제일 안지켜지고 있는 원칙
관리자는 창남피디님만 수정 가능하다.

- 지속적인 통합
  매일 여러번 통합해야 한다.
  깃헙에서의 (푸시, 풀, 머지 등을 자주해야한다는 것임)
  (수정과 동기화가 자주 되어서 코드공유가 빨라야한다는 의미임)
- 계획 세우기
  이거야 당연한거고
- 작은 릴리즈
  작은 시스템을 먼저 만들고 짧은 단위의 업데이트
- 메타포어
  명명 규칙, 주석처리 잘해라
- 간단한 디자인
  쓰잘데기 없이 현란하게 만들지 말고 심플하게!
- 테스트 기반 개발
  테스트 모듈을 먼저 만들고 개발할 것
- 리팩토링
  중복제거,단순화를 수시로 하자.
  (이건 중요하다고 봄)
- 40시간 작업
  (이거 내가 제일 못지키는 거...)
  아니 근데, 속도가 나려면 어쩔수 없지 않나?
  암튼 잘 쉬는 건 중요하다 생각함
- 고객 상주
  개발자들이 언제든지 개발하다가 궁금한 것들을 답변해줄 수 있는 사람이 필요함.
- 코드 표준
  코딩 표준을 정해야 한다.(위의 메타포어랑 비슷한 느낌이네)

※ 결국 여기서는 뭐가 뭐고 설명하라고 하기보다
5가치와 12원리를 외울 수 있는게 필요할 수 있음

다시 정리
5가치 : (용기,단순성,의사소통,피드백,존중) : 용단의피존
12원리 : 짝공지계작메간테리사고코
짝프로그래밍

공동코드소유
지속적인 통합
계획세우기
작은 릴리즈
메타포어
간단한 디자인
테스트기반 개발
리팩토링
40시간 작업
고객상주
코드표준

이건 지금 외우는게 좋을 듯

2. 스크럼

짧은 시간의 개발하는 팀을 지칭함.
상담앱이 한 스크럼이 됨.

ARA같은 경우는 프론트가 한 스크럼이 될 수 있겠음.

백로그 : 요구사항
스프린트 : 2~4주간의 짧은 개발기간
스크럼미팅 : 기강마스터의 회의
스크럼마스터 : 기강마스터
스프린트회고 : 노션 정리
번다운차트 : 남은 일정등을 차트로 보여줌

3. 린

도요타 린 프로세스에서 가져왔다네

칸반보드를 사용한대

그냥 이런개 있구나 정도 하자.

책에보면...
애자일 방법론과 나선형 모델이 다르다고 분류가 되어있는데
나는 거의 같다고 봄. 인터넷에도 거의 비슷하다는 의견이 대부분

제품계열방법론

공통된 기능을 정의하여 개발함.

영역 공학과 응용 공학으로 구분함.

그리 중요하지는 않을 듯.

2 비용산정, 일정관리 모형

여기는 결국 계산식 아닌가

비용산정 모형 분류

- 하향식 산정방법
  경험이 많은 전문가.
  전문가의 경험에 따른 비용을 판단
  (델파이 기법)

뭐 전문가라지만 예를들면
경험이 많은 팀장님이
이 정도 서비스면 얼마정도 걸리겠다 판단하는 것과 같음

- 상향식 산정방법
  세부적인 요구사항과 기능에 따라 계산함.
  - 코드라인 수
  - Man Month
  - COCOMO 기법
  - 푸트남 모형
  - 기능점수(FP) 모형

> 코드라인 수 :
> (낙관치 + 4 * 중간치 + 비관치)/6

> ManMonth : 코드라인수 / 프로그래머의 월간 생산성
> 프로젝트 기간 = ManMonth / 프로젝트 인력

> COCOMO 모형
> 보헴이 제안,
> 5만라인 이하 : 조직형
> 30만 라인 이하 :반분리형
> 30만 라인 이상 : 임베디드형

> 푸트남 모형
> 개발주기의 단계별 인력의 분포를 가정하여 계산

> Rayleigh-Norden 곡선의 노력분포도를 기초로 함.

> 기능점수 모형
> 총 기능점수 * (0.65 + (0.1 * 총 영향도))

일정관리 모델 (넘어감)

Chapter 02 현행 시스템 분석

현행시스템 파악

현행시스템 파악개념, 파악 절차는 넘어감

소프트웨어 아키텍처

소프트웨어의 구성요소
그 구성요소의 외부에 드러나는 특성
구성요소간의 관계를 표현

여기선 중요한건 아래..

소프트웨어 아키텍처 패턴

- 계층화 패턴
  계층으로 구분함
  각 하위 모듈들은 특정한 추상화를 제공, 상위 계층에 서비스를 제공
  계층화 패턴은 서로 마주 보는 두 개의 계층 사이에서만 상호작용이 이뤄짐

컴포넌트같은 걸로 구분할 때, 부모노드와 자식 노드들이 생기니까
그런 경우를 계층화 패턴이라고 할 수 있지 않을까

- 클라이언트 서버 패턴
  하나의 서버, 다수의 클라이언트
  서버에요청, 서버가 응답
  서버는 계속해서 요청대기
- 파이프 필터 패턴
  몰라..
- 브로커 패턴
  서버가 여러개로 분산됨
  서버와 클라이언트 사이에 브로커를 둠

클라이언트의 요청에 맞게 서버를 리다이렉트 시켜서 처리해줌.

- 모델뷰컨트롤러 (MVC 패턴)
  - 모델 : 핵심 기능과 데이터 보관
  - 뷰 : 사용자에게 정보 표시
  - 컨트롤러 : 사용자로 부터 요청을 입력 받음.

예시로 로그인을 할 때 입력하고 로그인 클릭했을 때

1. 뷰 단에서 입력을 감지
2. 컨트롤러로 넘겨서 요청을 모델쪽으로 보냄
3. 모델에서 검사해서 해당 결과를 컨트롤러로 보냄
4. 컨트롤러에서 뷰로 넘기고, 뷰에서 보여줌.

디자인 패턴

디자인 패턴이란?
공통으로 발생하는 문제에 대한 자주 쓰이는 설계 방법을 정리

디자인패턴의 유형

목적

- 생성 : 객체 인스턴스 생성할 때의 패턴
- 구조 : 더큰 구조 형성을 목적으로 함.
- 행위 : 클래스나 객체들의 상호작용하는 방법, 역할 분담 다룸.

범위

- 클래스 : 클래스간의 관련성, 컴파일 타임에 정적으로 결정
- 객체 : 객체 간 관련성, 런타임에 동적으로 결정

위의 내용도 외워야겠지만
결국에는
생성의 종류
구조의 종류
행위의 종류 ..

이런식으로 각 종류가 뭐가 있는지..
뭐가 어떤건지 등을 알고 있어야 한다.

디자인 패턴 종류

생성패턴

- Builder
  객체를 구현하는 방법과 생성하는 방법을 분리하였다.
  예를 들면 Director 객체가 있고
  ToyBuilder 객체가 있음
  PoohBuilder 객체를 만든 상태에서 곧바로 pooh를 만들 수 없고
  Director에게 가서 이거 만들어주세요 해야 함.
  toy를 만들 땐 director에게 가야하지만
  toy의 형태는 내가 설정해줘야 하는 것임.
  동일한 절차로 다양한 애들을 만들어줄 수 있다는 것이 핵심
  참고링크
- Prototype 패턴
  프로토타입을 만들어서 복제를 용이하게 만든 것임
  예를들면 Car라는 객체를 만들었다.
  근데 Car2에는 어떤 데이터를 추가해야하고
  Car3에는 어떤 데이터를 빼주어야 한다면
  Car를 만들고 Car.clone 해서 복제해서
  Car2, Car3를 조작할 수 있도록 만든 것이 Prototype 패턴이다.
  참고링크
- Factory Method
- Abstract Factory
- Singleton

구조패턴

- Bridge
- Decorator
- Facade
- Flyweight
- Proxy
- Composite
- Adapter

행위패턴

- Mediator
- Interpreter
- Iterator
- Template Method

- Observer
- State
- Visitor
- Command
- Strategy
- Memento
- Chain of Responsibility

개발 기술 환경 정의

운영체제

운영체제의 종류 및 특징

- PC :
  - 윈도우
  - 유닉스
  - 리눅스
- 모바일 :
  - 안드로이드
  - IOS

네트워크

OSI 7 Layer

- 응용계층
- 표현계층
- 세션계층
- 전송계층
- 네트워크계층
- 데이터 링크계층
- 물리계층

DBMS

이게 SSMS 임.
기능

- 중복제어
- 접근통제

- 인터페이스 제공
- 관계 표현
- 샤딩/파티셔닝
- 무결성 제약조건
- 백업 및 회복

미들웨어

운영체제와 소프트웨어 사이에 위치하여 원만한 통신이 이뤄질 수있도록 제어해준다.

가장 대표적인 것이 WAS

WAS (Web Application Server)의 개념

아래의 두 부분은 넘어감

Chapter 03 요구사항 확인

Chapter 04 분석 모델 확인하기