# 중간고사 - 암기

1. 데이터 수집 방법 - 대면, 비대면
2. 소프트웨어 비용 산정 기능 점수 - 산정 절차 자세한 설명
3. 팀 구성 방식 외우기
4. 프로젝트 요구사항, 기능, 제품 베이스라인
5. 기술 부채에 대한 내용과 해결 방안
6. 데브옵스 린 사고방식 7가지
7. 데브옵스 모델 iaas 비교 설명 (p 154)
8. 데브옵스 핵심 원리
9. 소프트웨어 공학적 기법 기본 원리
10. 데이터의 기능과 처리기능 원리 알아두기
11. 기능적, 비기능적 요구 사항 - 분별할 수 있어야한다. - 정의서 확인하기
12. 소프트웨어 다양한 모델 어떤 사항일 때 어떤 모델을 사용할지 알아두기
13. 데브옵스 프로세스 절차
14. 품질 요소 中 외적 요소, 내적 요소 디테일하게
15. 퍼트 차트에서 임계 경로

---

## 1. 데이터 수집 방법 - 대면, 비대면

### 인터뷰

시스템 개발과 관련된 이해 관계자들에게 대면 인터뷰를 신청한다.

1. 인터뷰 준비단계
   인터뷰 대상자 선정 → 인터뷰 계획 수립 → 인터뷰 준비
2. 인터뷰 실시단계
   인터뷰 시작 → 인터뷰 실시 → 인터뷰 종료
3. 인터뷰 정리단계
   인터뷰 결과 정리 → 인터뷰 결과 공지 및 평가

### JAD 세션

프로젝트 관리자, 사용자, 개발자가 모여 요구사항 도출을 위해 상호 토론하는 방법
3주에 걸쳐 총 5일 정도 수행한다.

1. 참석자를 선정한다.
2. JAD 세션에 대한 전반적인 사항을 설계한다.
3. JAD 세션을 준비한다
4. JAD 세션을 진행한다.
5. 세션 종료 후, 후속 조치를 한다.

## 비대면 수집 방법

### 문서 분석

개발 대상 소프트웨어를 사용할 업무에서 현재 사용하고 있는 다양한 종류의 분서를 분석해 구체적인 요구사항을 확보한다.

### 설문지 활용

개인에게서 필요한 정보를 수집하기 위하여 작성한 질문지로서, 다수 사용자에게 배포하고 수집하여 요구사항을 확보한다.

### 관찰

관찰은 소프트웨어의 미래 사용자가 수행하는 활동을 살펴보는 활동이다.
매일 반복되는 일이라서 중요하지 않다고 생각하는 업무 처리 과정을 찾아내는것이 목표이다.

### 소셜 네트워크

시간과 공간의 제약 없이 다수 사용자에게서 요구사항을 수집하고자 할 때, 소셜 네트워크 서비스는 효과적인 요구사항 수집 방법이된다.

## 2. 소프트웨어 비용 산정 기능 점수 - 산정 절차 자세한 설명

### 기능 점수 구성 요소

**데이터 기능(Data):** 내부 데이터 파일, 외부 데이터 파일

**처리 기능(Transaction)**: 사용자에게 의미있는 데이터를 처리하는 기능을 제공하는 프로세스의 집합을 말하며 외부 입력(EI), 외부출력(EO), 외부 질의(EQ)에 대해 DET와 FTR를 식별해 계산한다.

**기능 점수 산정**: 각 기능의 양을 Count한 후, 각 기능을 정의한 복잡도의 가중치를 곱한다.

### 기능 점수 산정 절차 !!!!

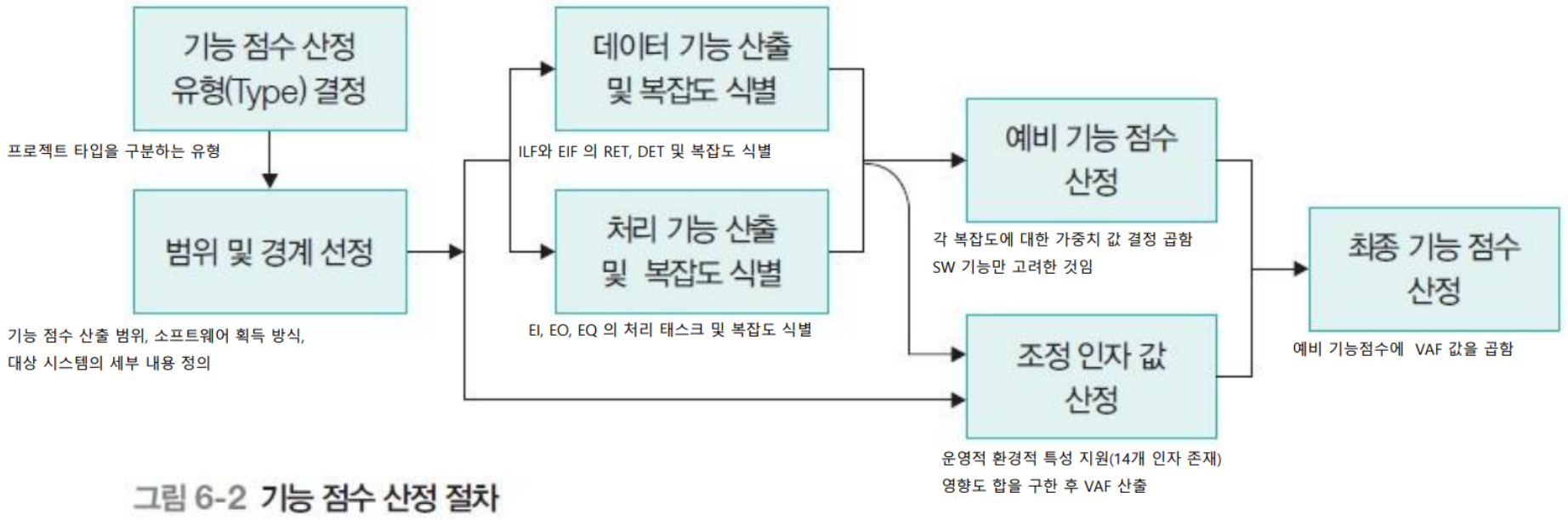


그림 6-2 기능 점수 산정 절차

1. **기능 점수 산정 유형 결정**: 프로젝트 타입을 구분하는 유형
2. **범위 및 경계 선정**: 기능 점수 산출 범위, 소프트웨어 획득 방식, 대상 시스템의 세부 내용을 정의한다.
3. **데이터 기능 산출 및 복잡도 식별**: 내부 논리 파일(ILF)과 외부 인터페이스 파일(EIF)을 식별한다.
4. **처리 기능 산출 및 복잡도 식별**: 응용 소프트웨어가 데이터를 처리하여 사용자에게 제공하는 기능인 처리 기능을 산출하고 복잡도를 식별한다.
5. **예비 기능 점수 산정**: 3~4단계를 거쳐 모든 데이터 기능과 처리기능에 대한 복잡도가 산정되면 각 복잡도에 대한 가중치 값을 결정해 곱해준다.
6. **조정 인자 값 산출**: 운영적, 환경적 특성 지원을 분석한다.
7. **최종 기능 점수 산출**: 앞 단계에서 결정된 VAF 값을 예비 기능 점수에 곱해준다.

## 3. 팀 구성 방식 외우기

### 프로젝트 팀 구조

#### 1. 중앙 집중형 팀 구조

프로젝트에서 수행해야 할 작업 목록이 단순하거나 충분히 이해된 경우에 적합한 팀 구성 방식이다.

유능한 프로젝트 리더가 필요하다.

문제 해결이 신속하게 이루어질 수 있고, 의사소통의 패턴이 매우 단순하다.

#### 2. 분산형 팀 구조

프로젝트의 주요 의사결정이 팀 구성원의 합의에 의해 이루어지는 민주주의적 팀 구성이다.

문제가 복잡하여 해결 방안을 모색해야 하는 경우에 적합하다.

대규모 구성원을 포함하는 프로젝트에는 적합하지 않을 수 있다.

#### 3. 하이브리드 팀 구조

중앙 집중형 팀 구조와 분산형 팀 구조를 통합한 계층형 구조이다.

프로젝트 관리자(PM)는 각 팀의 리더와 의사 결정을 위한 중앙집중형 구조를 취하고, 팀 내부의 운영은 분산형 구조를 채택하여 의사소통한다.

팀원을 소규모로 구성해 팀별 문제 해결이 용이하다.

---

# 4. 베이스 라인과 그 종류

# 1. 일정 관리 기법

## 베이스 라인

어떤 변경이 발생했을 때, 모든 관련자의 합의를 거쳐 변경사항을 결정한 다음, 후속 작업을 수행하기 위한 출발점이라고 정의한다.

## 베이스 라인 관리

1. **요구사항 베이스 라인**: 요구사항을 수집하고 프로젝트 계획을 수립한 후에 정의하는 베이스라인
2. **기능 베이스 라인**: 요구사항 분석 단계 후에 설정하는 베이스 라인
3. **제품 베이스 라인**: 소프트웨어 개발을 완료한 후에 정의하는 베이스 라인이며 개발 과정의 모든 산출물이 이 베이스 라인 기준선에 포함된다.

---

# 5. 기술 부채에 대한 내용과 해결 방안

기술 부채 절감을 위한 DevOpㄴ 전략

1. 동기부여
   개발에서 배포까지 소프트웨어 개발의 전체 수명주기를 관리하는 DevOps 팀에 동기를 부여하는것
2. DevOps 자동화 활용
   자동화는 DevOps 접근 방식의 핵심이다. 일상적, 오류가 발생하기 쉽고, 시간이 많이 소요되는 프로세스를 자동화하면 개발 팀은 기술 부채를 완화하기 위한 활동에 시간을 쓸 수 있다.
3. 마이크로서비스 모델 채택
   상태 비저장 마이크로서비스를 사용한다. 마이크로 서비스 모델로 이동함으로써 단일 서비스 또는 어플리케이션 기능이 변경되더라도 DevOps의 다른 서비스 또는 기능에 영향을 미치지 않아 개발자는 변경으로 인한 기술 부채에서 자유로워진다.
4. 컨테이너화된 배포
   코드 개발 과정을 자동화하고, 배포를 컨테이너화하고, 소프트웨어 개발 수명주기와 배포 프로세스의 초기에 보안 품질 인증 과정을 구축하면 많은 시간이 단축된다.
5. API 중심 모델 구축
   DevOps 팀에서 개발 코드의 인터페이스를 잘 정의하고 버전 관리를 수행함으로써, 기술 부채를 피한다.

- 마이크로 서비스는 단일 애플리케이션이 다수의 느슨하게 결합되고 독립적으로 배치 가능한 더 작은 구성요소 또는 서비스로 구성되는 클라우드 네이티브 아키텍처 접근 방식이다.
- 마이크로 서비스를 사용했을 경우, 서비스를 독립적으로 배치할 수 있고, 정확한 스케일링이 가능하다는 이점이 있다.

---

# 6. 데브옵스 린 사고방식 7가지

린 사고방식은 낭비 요소를 제거하면서 사회에 더 많은 혜택을 제공하고, 개인이 일을 통한 가치를 얻을 수 있또록 인간 활동을 체계화하는 방법을 제공하는 데 목표를 둔 혁신적 접근 방법이다.

1. 낭비를 제거하라
2. 학습을 확대하라
3. 결정은 최대한 늦게하라
4. 최대한 빨리 배포하라
5. 팀에 권한을 위임하라
6. 통합 체계를 구축하라
7. 전체를 최적화하라

---

# 7. 데브옵스 모델 iaas 비교 설명 (p 154)

### Iaas 모델의 DevOps

Iaas에서는 클라우드 서비스 제공자가 네트워크, 저장소, 서버 등을 포함하는 가상화된 인프라만 제공한다.  따라서 OS, 미들웨어, 어플리케이션 설치, 패치 등의 관리 및 책임은 사용자에게 있따. 즉, 어플리케이션 개발과 인프라 관리자 간 사일로가 존재하며, DevOps가 제공하는 큰 이점을 얻기 어렵다.

### Paas 모델의 DevOps

DevOps에서 Paas 클라우드 모델을 채택한다면, 사용자는 어플리케이션과 데이터의 생성 및 관리에 대해서만 고민하면된다. 다른 모든 계층은 클라우드 플랫폼에서 서비스로 제공된다. 특히 Paas모델에서는 개발 지원 도구, 테스트 도구, 배포를 위한 지원도 제공하기 때문에 Iaas 서비스 모델보다 개발자는 어플리케이션 개발이라는 핵심 임무에 더 집중할 수 있다.

---

## 8. 데브옵스 핵심 원리

DevOps는 **개발, 운영, 품질 보증 부서 및 서비스 제공자가 상호 협력하여 SW 시스템을 개발 배포하고, 시장 기회의 포착과 고객 피드백을 신속하게 수행함으로써, 서비스를 적시에 제공할 수 있는 린 및 애자일 원칙에 기반한 소프트웨어 개발 접근 방식**

### DevOps 핵심 원리

1. 자동화: 소프트웨어 동작이 배포, 운영 환경에서도 동일하게 보장될 수 있또록 자동화된 환경을 구축한다.
2. 반복: 개발 ↔ 운영을 순환구조로 연결해 반복 개발하는 형식이다.
3. 자기 서비스: 개발자와 운영자가 독립적으로 일할수 있도록 진행한다.
4. 지속적 개선: 사용자의 피드백을 SW를 개선하기 위해 활용
5. 협업: 개발, 운영 팀 간의 지속적인 협업 능력을 요구한다.
6. 지속적 테스트: 단위 테스트, 통합 테스트와 같은 많은 종류의 테스트 들을 통합과 배포 과정에서 반복한다.
7. 지속적 통합과 지속적 배포: 통합이 주기적으로 이루어지듯 배포도 문제 없는 한 지속적으로 이루어진다.

---

## 9. 소프트웨어 공학적 기법 기본 원리

### 소프트웨어 공학의 원리

**엄격성과 정형성**: SW 개발은 주어진 시간과 비용에서 명확하게 개발되어야한다.

**관심사의 분할**: 복잡한 문제를 단순한 문제들로 분리하여 적용한다.

**모듈화**: 응집력은 높이고 결합력은 낮추는 소프트웨어 구조 설계

**추상화**: 세부사항은 감추고 대표적인 속성으로 객체를 정의한다.

**변경의 예측**: 변경은 필히 일어나므로 변경이 일어날 것 같은 부분을 모듈화로 분리해 유지 보수에 용이하게 함

**일반화**: 특정한 곳이 아니라 다양한 환경, 사용자가 사용할 수 있게 지원한다.

**점진성**: 단계, 순차적으로 SW를 개발한다.

**명세화**: SW개발 과정 및 정보를 체계적으로 기술한다. (like SW 이력서)

### 소프트웨어 공학의 정의

1. 소프트웨어의 설계, 구현, 테스트, 문서화를 위한 과학적이고 기술적인 지식, 방법, 경험의 체계적인 적용을 한 것을 소프트웨어 공학이라고 한다.
2. 소프트웨어의 개발, 운용, 유지보수와 같은 수명주기 전반을 체계적, 서술적, 정량적으로 다루는 활동의 총체적인 모임

#### 소프트웨어 공학의 목표

다양한 소프트웨어 공학 기법을 활용해 개발 대상을 명확화, 개발 과정을 체계화, 개발 수명주기를 지원하여 사용자의 요구사항을 충족시키는 **품질 좋은 소프트웨어**를 개발하는 것.

---

## 10. 데이터 기능과 처리 기능 원리 알아두기

### 기능 점수 구성 요소

**데이터 기능(Data):** 내부 데이터 파일, 외부 데이터 파일

**처리 기능(Transaction)**: 사용자에게 의미있는 데이터를 처리하는 기능을 제공하는 프로세스의 집합을 말하며 외부 입력(EI), 외부출력(EO), 외부 질의(EQ)에 대해 DET와 FTR를 식별해 계산한다.

**기능 점수 산정**: 각 기능의 양을 Count한 후, 각 기능을 정의한 복잡도의 가중치를 곱한다.

---

# 11. 기능적, 비기능적 요구 사항 - 분별할 수 있어야한다. - 정의서 확인하기

## 1- 1. 요구사항

소프트웨어 시스템이 수행해야 할 것과 소프트웨어 시스템에 있어야 할 특성을 기술한 문장

## 1 - 2. 요구사항의 분류

### 기능적 요구사항

사용자의 업무 처리와 직접 관련되어 소프트웨어 시스템이 수행해야 하는 요구 내용

소프트웨어 개발에서 반드시 구현되어야 하는 항목

### 비기능적 요구사항

소프트웨어 시스템이 제공해야 하는 행위적 속성이다. 즉, 고장이 나면 안 되며 적절한 성능을 제공해야한다.

### 인터페이스 요구사항

시스템을 사용하는 과정에서 지원해야하는 GUI와 기존 시스템과의 연동도 포함해야한다.

비기능적 요구사항 예시 - 나머지 예시는 기능적 요구사항

| |
|---|
| 시스템은 네트워크 연결 시에만 작동한다. |
| 시스템은 리눅스에서 동작한다. |
| 시스템은 모든 사용자의 요청에 대해 3초 내로 응답한다. |
| 카메라 모듈은 500만 화소를 제공한다. |
| 시스템은 모든 사용자의 개인정보를 암호화하여 저장한다. |
| 시스템은 영어 및 한국어를 지원한다. |
| 데이터베이스는 실시간으로 업데이트 된다. |

---

# 12. 소프트웨어 다양한 모델 어떤 사항일 때 어떤 모델을 사용할지 알아두기

---

# 13. 데브옵스 프로세스 절차

### DevOps 프로세스!!!

과정 잘 알아두자 (폭포수도 비슷하니까 외우기)

1. **계획 단계**

   SW개발자, 설계자, 담당자, 인프라 관리자등이 함께 참여해, 어플리케이션의 비지니스 가치와 사용자 요구사항을 정의한다.

2. **코딩**

   SW설계, SW 제품 형상 설계 및 정의하고 코드 품질과 개발 생산성을 고려한 소스코드를 작성하고 단위 테스트를 수행한다.

3. **빌드**

   작성된 코드를 통합하고 제품 형상을 정의한다.

4. **테스트 단계**

   사용자의 요구사항, 회귀, 성능, SW형상 등을 테스트하고 보안과 취약성을 분석한다.

5. **어플리케이션 배포를 승인하는 릴리즈 단계**

6. **설치 단계**

   사용자 환경에 설치하는 설치 단계

7. **운영 및 모니터링 단계**

   지속적인 성능에 대한 모니터링을 하는 운영 및 모니터링 단계

## 14. 품질 요소 中 외적 요소, 내적 요소 디테일하게

1. **외적 품질 요소 (보여지는 증상, 기능적 측면, 사용자 위주 측면)**

   정확성, 신뢰성, 견고성, 성능, 사용자 친숙성, 가용성, 보안성, 안전성, etc..

   1. 정확성: 주어진 명세서의 내용을 하나씩 테스트하여 원하는 결과를 생성하는지 여부로 판
   2. 신뢰성: SW를 사용하는 동안 나타나는 오류 발생 정도
   3. 견고성: 사용자가 제시한 요구 사항 명세에 정의하지 않은 조건이나 환경에서도 소프트웨어가 합리적으로 동작해야 견고하다고 한다.
   4. 성  능: SW의 효율성 - 메모리 사용량, 총 실행 시간 등으로 측정한다.
   5. 사용자 친숙성: 소프트웨어가 사용하기 편리한 지를 나타내는 품질 요소
   6. 가용성: 서버 네트워크 프로그램의 정보 시스템이 정상적으로 사용 가능한 정도 (SW를 정상적으로 사용 가능한 시간 / SW 전체 운영 시간)
   7. 무결성: 프로그램이나 데이터에 대한 불법, 잘못된 접근을 막는 정도
   8. 외부의 악의적인 공격이나 해커의 위협을 소프트웨어가 막아낼 수 있도록 구현 해 잠재적인 공격이 예측되는 상황에서도 SW가 올바르게 동장

---

2. **내적 품질 요소 (감춰진 증상, 개발자 위주 측면)**

   검증 가능성, 유지 보수성, 재사용성, 이식성, 가독성, 생산성, 상호 운용성, 적시성, 가시성, 코드 규칙, etc..

   1. 검증 가능성: 소프트웨어가 지닌 속성이 올바르다는 것을 안전하게 확인할 수 있다면 검증 가능성을 갖는다고 한다.
      정형 검증과 테스트로 평가한다.
   2. 유지 보수성
      a. 수정 유지 보수: 개발된 소프트웨어를 사용하는 동안 오류가 발생하는 경우에 이루어지는 활동
      b. 적응 유지보수: 소프트웨어 운영 조건에 대한 변화를 수용하는 유지보수 활동
      c. 완전 유지 보수(리뉴얼하기): 사용 중인 SW에 대한 가독성 및 이해성을 높이는 재구조화를 목적으로 수행
      d. 예방 유지 보수: 소프트웨어가 정확하게 동작할 것인지를 테스트하고 점검

      **임무 중심** 및 **안정 중심 소프트웨어**의 경우에 매우 중요한 요소이다.
   3. 재사용성: 새로운 소프트웨어를 개발하기 위해 기존 SW 컴포넌트를 사용하는 정도
      재사용성 = LOC(R) / LOC(S) = 재사용에 의해 개발된 부분의 양 / 전체 개발된 산출물의 양
   4. 생산성: 외적 품질 요소인 성능의 속성을 적용한 내적 품질 속성, 주어진 시간 내에 얼만큼의 성과를 내는가
   5. 상호 운용성: 서로 다른 소프트웨어들이 협업을 수행할 수 있는 능력을 충분히 제공하는 것을 의미한다.
   6. 가시성: SW 개발 단계의 상태 정보와 산출물 등을 체계적으로 유지 및 관리 (ex 문서화해서 SW 개발 과정을 관리)

---

## 15. 퍼트 차트에서 임계 경로

프로젝트에서 수행되어야 하는 태스크 간의 **의존성**을 표현하는 차트
프로젝트 전체 수행 일정에서 핵심적인 수행 경로를 식별 관리한다.

- 임계 경로: 각 경로 상에 있는 태스크 수행 일수의 합이 가장 많은 핵심 경로 → 임계 경로를 기준으로 프로젝트 수행 기간을 정하면 된다.