**살충제 페러독스**

**정의**: 동일한 테스트 케이스에 의한 반복적 테스트는 새로운 버그를 찾지 못한다는 테스트 원리

**프로토콜의 기본 요소 3가지**

• **구문(Syntax)**: 데이터 형식 및 신택스

• **의미(Semantics)**: 데이터의 의미

• **순서(Timing)**: 데이터 전송 순서

**XML (eXtensible Markup Language)**

**정의**: W3C(World wide Web Consortium)에서 개발되었고, 웹 브라우저 간 호환이 되지 않는 문제와 SGML(Standard Generalized Markup Language)의 복잡함을 해결하기 위해 개발된 다목적 마크업 언어

**JSON (JavaScript Object Notation)**

**정의**: 속성-값(Attribute-value Pair)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷 **설명**: Ajax(Asynchronous JavaScript and XML)에서 많이 사용되고 XML(eXtensible Markup Language)을 대체하는 주요 데이터 포맷. 언어 독립형 데이터 포맷으로 다양한 데이터 프로그래밍 언어에서 사용

**스케줄링 방식에서 HRN(Highest Response ratio Next) 우선순위 계산식**

**계산식**: (대기 시간 + 서비스 시간) / 서비스 시간

**트랜잭션의 특징**

• **원자성**: 트랜잭션이 데이터베이스에 모두 반영되던가, 전혀 반영되지 않아야 한다는 점

• **일관성**: 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다는 점

• **독립성**: 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우, 어떤 하나의 트랜잭션이라도 다른 트랜잭션의 연산에 끼어들 수 없다는 점

• **지속성**: 트랜잭션이 성공적으로 완료되었을 경우, 결과는 영구적으로 반영되어야 한다는 점

**랜드 어택 (Land Attack)**

**정의**: 공격자가 패킷의 출발지 주소나 포트를 임의로 변경해 출발지와 목적지 주소를 동일하게 함으로써 공격 대상 컴퓨터의 실행 속도를 느리게 하거나 동작을 마비시켜 서비스 거부 상태에 빠지도록 하는 공격 방법

**MD5**

**정의**: RFC 1321로 지정되어 있으며, 주로 프로그램이나 파일이 원본 그대로인지를 확인하는 무결성 검사 등에 사용 **설명**: 1991년 로널드 라이베스트가 예전에 쓰이던 MD4를 대체하기 위해 고안된 128비트 암호화 해시 함수

**비정규화 (De-Normalization)**

**정의**: 정규화된 엔티티, 속성, 관계에 대해 성능 향상과 개발 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링 기법

**물리 계층 (Physical Layer)**

**정의**: OSI 계층 중 비트를 전송하는 계층

**재해 복구 시간 또는 RTO (Recovery Time Objective)**

**정의**: 비상사태 또는 업무 중단 시점부터 업무가 복구되어 다시 정상 가동될 때까지의 시간을 의미하는 용어 **설명**: 정보 시스템 운영 중 서버가 다운되거나 자연재해나 시스템 장애 등의 이유로 고객에게 서비스가 불가능한 경우에 발생

**비동기 통신 기법 또는 AJAX (Asynchronous JavaScript and XML)**

**정의**: 브라우저가 가지고 있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고도 페이지의 일부분만을 위한 데이터를 로드하는 기법 **설명**: 하이퍼텍스트 표기 언어(HTML)만으로 어려운 다양한 작업을 웹 페이지에서 구현해 이용자가 웹 페이지와 자유롭게 상호작용할 수 있도록 하는 기술명

**애자일 방법론**

**정의**: 절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템을 개발할 수 있는 신속 적응적 경량 개발 방법론 **설명**: 개발 기간이 짧고 신속하며, 워터폴에 대비되는 방법론으로 최근 회사에서 각광받는 방법론

**트랜잭션 Rollback**

**정의**: 트랜잭션 처리 중 오류가 발생했을 때, 오류 이전의 특정 시점(SAVEPOINT, CHECKPOINT) 상태로 되돌려주는 제어어(명령문)

**IPSec (Internet Protocol Security)**

**정의**: 무결성과 인증을 보장하는 인증 헤더(AH)와 기밀성을 보장하는 암호화(ESP)를 이용한 프로토콜 **설명**: 네트워크 계층(Network Layer)인 인터넷 프로토콜(IP)에 보안성을 제공해주는 표준화된 기술

**정적 분석 도구**

**정의**: 애플리케이션을 실행하지 않고, 소스 코드에 대한 코딩 표준, 코딩 스타일, 코드 복잡도 및 남은 결함을 발견하기 위해 사용하는 도구

**Observer 패턴**

**정의**: 한 객체의 상태가 바뀌면 그 객체에 의존하는 다른 객체들이 연락을 받아 자동으로 내용이 갱신되는 방법 **설명**: 일대 다의 의존성을 가지며 상호작용하는 객체 사이에서는 가능하면 느슨하게 결합하는 디자인을 사용해야 한다

**안드로이드**

**정의**: Linux 운영체제 위에서 구동하며 휴대폰 전화를 비롯한 휴대용 장치를 위한 운영체제와 미들웨어, 사용자 인터페이스 그리고 표준 응용프로그램(웹 브라우저 등)을 포함하고 있는 소프트웨어 스택 **설명**: 리눅스 모바일 운영체제로, 개발자들이 자바와 코틀린 언어로 응용 프로그램을 작성할 수 있게 했고, 컴파일 된 바이트 코드를 구동할 수 있는 런타임 라이브러리를 제공하는 운영체제

**SOAP (Simple Object Access Protocol)**

**정의**: HTTP, HTTPS, SMTP를 통해서 XML 기반의 데이터를 주고받는 프로토콜로 웹 서비스 방식에 HTTP 기반의 **SOAP**을 사용하여 송수신한다. **SOAP**대신 **레스트풀(RESTful)**프로토콜로 대체할 수 있다.

**SQL Injection**

**정의**: 응용 프로그램의 보안 취약점을 이용해서 악의적인 SQL 구문을 삽입, 실행시켜서 데이터베이스에 접근하여 정보를 탈취하거나 조작 등의 행위를 하는 공격 기법

**chmod 751 a.txt**

**정의**: 사용자에게 읽기 / 쓰기 / 실행 권한을 부여하고 그룹에게는 읽기 / 실행을 부여하고 그 이외에는 실행 권한을 a.txt에 부여하는 명령어를 8진법을 사용하여 한 줄로 작성한 명령어

**UI 설계 원칙**

• **직관성**: 누구나 쉽게 이해하고 사용할 수 있어야 한다.

• **학습성**: 누구나 쉽게 배우고 익힐 수 있어야 한다.

• **유연성**: 사용자의 요구사항을 최대한 수용하며 오류를 최소화해야 한다.

• **유효성**: 사용자의 목적을 정확하게 달성해야 한다.

**LOD (Linked Open Data)**

**정의**: 전 세계 오픈된 정보를 하나로 묶는 방식으로, **Link Data**와 **Open Data**의 합성어

**데이터 모델링 과정**

1. 요구사항 분석

2. 개념적 데이터 모델링

3. 논리적 데이터 모델링

4. 물리적 데이터 모델링

**형상관리기법**

**정의**: 소프트웨어 개발 과정에서 변경 사항을 관리하는 기법 **도구**: CVS, SVN, Git

**리팩토링의 목적**

**정의**: 복잡한 코드의 단순화, 소스의 가독성을 통해 유지보수성 향상, 유연한 시스템, 생산성 향상, 품질 향상

**OSPF (Open Shortest Path First)**

**정의**: 대표적인 내부 라우팅 프로토콜로, 다익스트라 알고리즘을 이용한 대규모 네트워크에 적합한 링크 상태 라우팅 프로토콜

**형상 통제**

**정의**: 형상 항목의 버전 관리를 위해서 변경 여부와 변경 활동을 통제하는 활동

**프로토콜**

**정의**: 심리학자 톰 마릴은 컴퓨터가 메시지를 전달하고, 메시지가 제대로 도착했는지 확인하며, 도착하지 않았을 경우 메시지를 재전송하는 일련의 방법을 '기술적 은어'를 뜻하는 **프로토콜**이라는 용어로 정의

**TCP/IP 오류 처리**

**정의**: TCP/IP에서 오류가 발생하면 ICMP(Internet Control Message Protocol) 메시지를 보내서 오류가 발생했음을 알린다.

**관계 대수 연산**

• **셀렉트(σ)**: 릴레이션에 존재하는 튜플들 중에서 특정 조건을 만족하는 튜플들의 부분집합을 구하여 새로운 릴레이션을 만듭니다.

• **프로젝트(π)**: 주어진 릴레이션에서 속성 리스트에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만듭니다. (연산 결과에 중복이 발생하면 중복이 제거됩니다.)

• **조인(⋈)**: 공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만듭니다.

• **디비전(÷)**: 릴레이션 A, B가 있을 때 릴레이션 B 조건에 맞는 것들만 릴레이션 A에서 튜플을 꺼내 프로젝션하는 관계 대수의 기호

**블랙박스 테스트**

**정의**: 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 테스트로, 기능 테스트라고 불립니다.

**동치 분할 검사**

**정의**: 입력 자료에 초점을 맞춰 테스트 케이스를 만들고 검사하는 방법 (= 동등 분할 기법)

**경계값 분석**

**정의**: 입력 자료에만 치중한 동치 분할 기법을 보완한 기법으로, 입력 조건의 중간값보다 경계값에서 오류가 발생될 확률이 높다는 점을 이용하여 입력 조건의 경계값을 테스트 케이스로 선정하여 검사하는 기법

**원인-효과 그래프 검사**

**정의**: 입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법

**오류 예측 검사**

**정의**: 과거의 경험이나 확인자의 감각으로 테스트하는 기법으로, 다른 블랙박스 테스트 기법으로는 찾아낼 수 없는 오류를 찾아내는 일력의 보충적 검사 기법 (= 데이터 확인 검사)

**비교 검사**

**정의**: 여러 버전의 프로그램에 동일한 테스트 자료를 제공하여 동일한 결과가 출력되는지 테스트하는 기법

**DB 스키마**

**정의**: 데이터베이스의 구조, 제약조건 등의 정보를 담고 있는 기본적인 구조

**UI 설계 원칙 중 직관성**

**정의**: 누구나 쉽게 이해하고, 쉽게 사용할 수 있어야 하며, 쉬운 검색, 쉬운 사용성, 일관성이 부특성을 가지고 있는 UI 설계 원칙

**EAI (Enterprise Application Integration)**

**정의**: 기업 내외부 정보 시스템 통합 **유형 (데이터 전송 모델)**:

• **메시지 버스**: 애플리케이션과 미들웨어 간 웹 서비스 인터페이스를 통해 전송

• **하이브리드**: 허브 앤 스포크와 메시지 버스 혼합

• **포인트 투 포인트**: 1:1 방식으로 애플리케이션 통합 수행

• **허브 앤 스포크**: 단일 접점인 허브 시스템을 통해 데이터를 전송하는 중앙 집중 방식

**C++에서 생성자**

**정의**: 해당 클래스의 객체가 생성될 때 자동으로 호출되는 특수한 종류의 메서드

**IPv6**

**정의**: 현재 IPv4의 확장형으로, IPv4가 가지고 있는 주소 고갈, 보안성, 이동성 지원 등의 문제점을 해결하기 위해 개발된 128비트 주소 체계를 갖는 차세대 인터넷 프로토콜

**디자인 패턴**

**목적**: 생성 / 구조 / 행위 **범위**: 클래스 / 객체

**UML (Unified Modeling Language) 다이어그램**

**정의**: UML 다이어그램은 통합 모델링 언어를 사용하여 시스템 상호작용, 업무 흐름, 시스템 구조, 컴포넌스 관계 등을 그린 도면 **사용 이유**: 프로그래밍을 단순화시켜 표현하여 의사소통하기 좋고, 대규모 프로젝트 구조의 로드맵을 만들거나 개발을 위한 시스템 구축에 기본을 마련 **종류**:

• 클래스 다이어그램

• 객체 다이어그램

• 유스케이스 다이어그램

• 상태 다이어그램

• 시퀀스 다이어그램

• 활동 다이어그램

• 통신 다이어그램

• 컴포넌트 다이어그램

• 배포 다이어그램

• 복합체 구조 다이어그램

• 교류 개요 다이어그램

• 타이밍 다이어그램

• 패키지 다이어그램

**즉각 갱신 회복 기법**

**정의**: 데이터베이스의 회복(Recovery) 기법 중 Rollback 시 Redo, Undo가 모두 실행되는 트랜잭션 처리법으로, 트랜잭션 수행 중 갱신 결과를 바로 DB에 반영하는 기법

**지연 갱신 회복 기법**

**정의**: 트랜잭션의 부분 완료 상태에서는 변경 내용을 로그 파일에만 저장

**체크 포인트 회복 기법**

**정의**: 장애 발생 시 검사점(checkpoint) 이전에 처리된 트랜잭션은 회복에서 제외하고, 검사점 이후에 처리된 트랜잭션은 회복 작업을 수행

**그림자 페이징 회복 기법**

**정의**: 트랜잭션이 실행되는 메모리상의 current page table과 하드디스크의 shadow page table을 이용

**미디어 회복 기법**

**정의**: 디스크와 같은 비휘발성 저장 장치가 손상되는 장애 발생을 대비한 회복 기법

**스니핑 (Sniffing)**

**정의**: 암호화되지 않은 패킷들을 수집하여 순서대로 재조합 후 ID, PW와 같은 중요한 정보를 유출하기 위한 수동적인 형태의 공격

**IP 스푸핑 (IP Spoofing)**

**정의**: 발신지 IP나 목적지 IP를 위조하여 공격하는 기법

**ARP 스푸핑 (ARP Spoofing)**

**정의**: MAC 주소를 위조하여 LAN 상에서의 통신 흐름을 왜곡시키는 공격 기법

**트로이 목마 (Trojan Horse)**

**정의**: 사용자가 원하는 무언가로 변장해 시스템 방어망을 뚫고 들어가는 공격 기법

**NAT (Network Address Translation)**

**정의**: IP 패킷에서 외부의 공인 IP 주소와 포트 주소에 해당하는 내부 IP 주소를 재기록하여 라우터를 통해 네트워크 트래픽을 주고받는 기술

**블록체인 (Blockchain)**

**정의**: 분산 컴퓨팅 기술 기반의 데이터 위변조 방지 기술로, P2P 방식을 기반으로 하여 소규모 데이터들이 연결되어 형성된 '블록'이라는 분산 데이터 저장 환경에 관리 대상 데이터를 저장함으로써 누구도 임의로 수정할 수 없고, 누구나 변경의 결과를 열람할 수 있게끔 만드는 기술

**하둡 (Hadoop)**

**정의**: 오픈 소스 기반의 분산 컴퓨팅 플랫폼으로, 일반 PC급 컴퓨터들로 가상화된 대형 스토리지를 형성하고 그 안에 보관된 거대한 데이터 세트를 병렬로 처리할 수 있도록 개발된 자바 소프트웨어 프레임워크로, 구글, 야후 등에 적용한 기술

**이상 현상의 종류**

**정의**: 삽입 이상, 삭제 이상, 갱신 이상

**테스트 오라클**

**정의**: 테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참값을 입력하여 비교하는 기법 **유형**:

• **참 오라클**: 모든 입력값에 대해 기대하는 결과를 생성하여 오류 검출

• **샘플링 오라클**: 특정한 몇 개의 입력값에 대해서만 기대하는 결과를 제공

• **휴리스틱 오라클**: 샘플링 오라클을 개선하여 특정 입력값에 올바른 결과를 제공하고 나머지 값은 추정으로 처리

• **일관성 검사 오라클**: 애플리케이션 변경 시 수행 전과 후의 결괏값이 동일한지 확인

**유닉스 (UNIX)**

**정의**: 데니스 리치와 켄 톰슨 등이 벨 연구소를 통해 만든 운영체제 **특징**:

• 90% 이상이 C언어로 구현됨

• 시스템 프로그램이 모듈화되어 다른 하드웨어 기종으로 쉽게 이식 가능

• 계층적 트리 구조로 통합적인 파일 관리가 용이

**RARP (Reverse Address Resolution Protocol)**

**정의**: 물리 네트워크(MAC) 주소에 해당하는 IP 주소를 알려주는 프로토콜 **설명**: 역순 주소 결정 프로토콜

**물리적 설계**

**정의**: 특정 DBMS의 특성 및 성능을 고려하여 데이터베이스 저장 구조로 변환하는 과정 **결과물**: 테이블 정의서 등

**개념적 설계**

**정의**: 현실 세계에 대한 인식을 추상적, 개념적으로 표현하여 개념적 구조를 도출하는 과정 **산출물**: E-R 다이어그램

**논리적 설계**

**정의**: 목표 DBMS에 맞는 스키마 설계, 트랜잭션 인터페이스를 설계하는 정규화 과정 수행

**요구사항**

**정의**: 시스템 개발 분야에서 어떤 과제를 수행하기 위해 필요한 조건이나 능력

• **기능적 요구사항**: 시스템이 제공하는 기능, 서비스에 대한 요구사항

• **비기능적 요구사항**: 시스템이 수행하는 기능 이외의 사항, 시스템 구축에 대한 제약사항에 관한 요구사항

**WSDL (Web Services Description Language)**

**정의**: 웹 서비스명, 제공 위치, 메시지 포맷, 프로토콜 정보 등 웹 서비스에 대한 상세 정보가 기술된 XML 형식으로 구성된 언어

**시스템 테스트**

**정의**: 구현된 시스템이 정해진 요건에 적합한지 여부를 평가하기 위해 실제 운용과 같은 환경에서 시스템 전체에 대해서 행하는 테스트

**인수 테스트**

**정의**: 계약상의 요구 사항이 만족되었는지 확인하기 위해, 설치 후 구입자의 현장에서 납품자도 참가하여 구입자에 의해 실시되는 시스템 또는 기능 단위 테스트

**알파 테스트**

**정의**: 특정 사용자들에 의해 개발자 관점에서 수행되는 테스트 (가장 자주 사용)

**회귀 테스트**

**정의**: 애플리케이션에 변경, 결함 수정 또는 기능 개선 등으로 인해 프로그램 코드에 물리적인 변경이 발생했을 경우, 새로운 결함에 대비하여 이미 실시했던 테스트케이스를 재시험하는 테스트

**단위 테스트**

**정의**: 개별 모듈, 서브루틴이 정상적으로 실행되는지 확인하는 테스트

**통합 테스트**

**정의**: 인터페이스 간 시스템이 정상적으로 실행되는지 확인하는 테스트

**IPC (Inter Process Communication)**

**정의**: 프로세스 간 통신하는 기술 **방법**: 공유 메모리, 소켓, 세마포어, 메시지 큐 등

**EAI (Enterprise Application Integration)**

**정의**: 시스템 통합에 사용되는 솔루션 **유형**:

• **포인트 투 포인트**: 가장 기초적인 애플리케이션 통합 방법 (1:1 단순 통합 방법) / 개발자 간 대화를 통해 통합 가능

• **허브 앤 스포크**: 단일한 접점의 허브 시스템을 통해 데이터를 전송하는 중앙 집중적 방식

• **메시지 버스**: 애플리케이션 사이에 미들웨어를 두어 연계하는 통합 방식 / 뛰어난 확장성과 대용량 데이터 처리 가능

• **하이브리드**: 그룹 내는 허브 앤 스포크 방식, 그룹 간에는 메시지 버스 방식을 사용하는 통합 방식

**제약조건**

**정의**: 데이터 무결성 유지를 위한 DB의 보편적 방법으로 릴레이션의 특정 컬럼에 설정하는 제약 **종류**: 개체 무결성, 참조 무결성 등

**접근 통제 모델**

• **임의적 접근 통제 (DAC, Discretionary Access Control)**: 시스템 객체의 접근을 개인 또는 그룹의 식별자에 기반을 둔 방법. 접근 권한을 가진 사용자가 다른 사용자에게 권한을 허용할 수 있음.

• **강제적 접근 통제 (MAC, Mandatory Access Control)**: 미리 정해진 정책과 보안 등급에 의거하여 접근을 통제하는 모델.

• **역할기반 접근 통제 (RBAC, Role Based Access Control)**: 사용자에게 할당된 역할에 기반하여 접근을 통제하며 중앙에서 집중적으로 관리.

**결합도 (Coupling)**

• **내용 결합도**: 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우의 결합도

• **스탬프 결합도**: 모듈 간의 인터페이스로 배열이나 객체, 구조 등이 전달되는 경우의 결합도

• **공통 결합도**: 파라미터가 아닌 모듈 밖에 선언된 전역 변수를 참조하고 전역 변수를 갱신하는 식으로 상호작용하는 경우의 결합도

• **외부 결합도**: 어떤 모듈에서 반환한 값을 다른 모듈에서 참조해서 사용하는 경우

• **제어 결합도**: 단순히 처리를 해야 할 대상인 값만 전달되는 것이 아니라 어떻게 처리를 해야 한다는 제어요소가 전달

• **자료 결합도**: 모듈 간의 인터페이스 전달되는 파라미터를 통해서만 모듈 간의 상호작용이 일어나는 경우 **결합도와 품질**(낮은 품질에서 좋은 품질 순): 내용 결합도 > 공통 결합도 > 외부 결합도 > 제어 결합도 > 스탬프 결합도 > 자료 결합도

**세션 하이재킹**

**정의**: '세션을 가로채다' 라는 의미로, 정상적 연결을 RST 패킷을 통해 종료시킨 후 재연결 시 희생자가 아닌 공격자에게 연결하는 공격 방법

**무선 네트워크 구성 방식**

**애드혹 네트워크(Ad-hoc Network)**:

• **정의**: 네트워크 장치를 필요로 하지 않고 네트워크 토폴로지가 동적으로 변화되는 특징이 있으며 응용 분야로는 긴급 구조, 긴급 회의, 전쟁터에서의 군사 네트워크에 활용되는 네트워크

**Infrastructure 네트워크**:

• **정의**: 이동통신망, Wi-Fi에 사용되며 중앙집중형 방식

**메쉬 네트워크**:

• **정의**: 무선 백본망에 사용되며 신뢰성이 우수하고, 계층적인 구조를 가지고 있음

**트랜잭션의 특징**

• **원자성**: 모두 반영되거나 아니면 전혀 반영되지 않아야 한다.

• **일관성**: 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다.

• **독립성**: 둘 이상의 트랜잭션이 동시에 실행될 경우, 어떤 하나의 트랜잭션이라도 다른 트랜잭션의 연산에 끼어들 수 없다.

• **지속성**: 트랜잭션이 성공적으로 완료되었을 경우, 결과는 영구적으로 반영되어야 한다.

**AES (Advanced Encryption Standard)**

**정의**: 미국 국립 표준 기술연구소(NIST)에서 DES를 대체하기 위해 개발된 128비트 블록 크기와 128, 192, 256비트 키 크기의 대칭 키 암호화 방식

**대칭키 알고리즘**

**정의**: 동일한 키를 사용하여 암호화와 복호화를 하는 알고리즘으로, 가장 많이 사용하는 알고리즘이 AES입니다.

**응집도 순서 (낮은 것부터 높은 순서)**

• **우연적 응집도**: 모듈 내부의 각 구성 요소들이 연관이 없을 경우

• **논리적 응집도**: 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우

• **시간적 응집도**: 연관된 기능이 아니더라도 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우

• **절차적 응집도**: 입출력 간 연관성은 없으나 순서에 따라 수행되는 것

• **교환적 응집도**: 동일한 입력과 출력을 사용하는 경우

• **순차적 응집도**: 모듈 내에서 한 활동으로부터 나온 출력값을 다른 활동이 사용할 경우

• **기능적 응집도**: 하나의 기능에 모두 기여하고 밀접하게 연관되어 있는 것

**패킷교환방식 (저장 방식)**

**정의**: 패킷이라는 단위를 사용하여 데이터를 송신하고 수신하는 방식 **설명**: 패킷은 정보를 일정한 크기로 분할한 뒤 각각의 패킷에 송수신 주소 및 부가 정보를 입력한 것으로, 현재 컴퓨터 네트워크에서 주로 사용하는 방식 (예: 인터넷)

**가상 회선 방식**

**정의**: 목적지 호스트와 미리 연결한 후 통신하는 연결형 교환 방식

**데이터그램 방식**

**정의**: 헤더에 붙어서 개별적으로 전달하는 비연결형 교환 방식

**디자인 패턴 중 행위 패턴**

**정의**: 반복적으로 사용되는 객체들의 상호작용을 패턴화 한 것으로, 클래스나 객체들이 상호작용하는 방법 **알고리즘 패턴 예시**: Interpreter, Observer, Command

**로킹 (Locking)**

**정의**: 병행 제어 기법 중 하나로, 접근한 데이터에 대한 연산을 모두 마칠 때까지 상호 배제하는 기법

**럼바우 객체지향 분석 기법**

**정의**: 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링하는 분석 기법 **분석 활동**: 객체 모델링(Object Modeling), 동적 모델링(Dynamic Modeling), 기능 모델링(Functional Modeling) **절차**: 객체 모델링 > 동적 모델링 > 기능 모델링

**AAA**

**정의**: 유무선 이동 및 인터넷 환경에서 가입자에 대한 안전하고 신뢰성 있는 인증, 권한 검증 등의 기능을 체계적으로 제공하는 정보 보호 기술 **구성 요소**:

• **인증(Authentication)**: 신분을 확인하는 기능

• **인가(Authorization)**: 접근 및 허가를 결정하는 기능

• **계정(Accounting)**: 리소스 사용 정보를 수집 및 관리하는 기능

**Grant의 기능**

**정의**: 사용자(User)에게 접속 권한, 오브젝트 생성 권한, DBA 권한 등을 부여할 수 있는 명령어

**ARP 스푸핑**

**정의**: 근거리 통신망 하에서 ARP 메시지를 이용하여 상대방의 데이터 패킷을 중간에서 가로채는 중간자 공격 기법 **설명**: 이 공격은 데이터 링크 상의 프로토콜인 ARP를 이용하기 때문에 근거리상의 통신에서만 사용할 수 있는 공격이다.

**OSI 7 Layer**

**정의**: 네트워크에서 통신이 일어나는 과정을 단계별로 파악하기 위해 7단계로 나눈 것

1. **물리계층(Physical Layer)**: 단순 데이터를 전기적인 신호로 변환(on/off)하여 주고받는 기능. 케이블, 리피터, 허브를 통해 데이터 전송.

2. **데이터 링크계층(DataLink Layer)**: 물리계층을 통해 송수신되는 정보의 오류와 흐름을 관리하여 안전한 정보의 전달 수행. MAC 주소를 사용.

3. **네트워크 계층(NetWork Layer)**: 전송 데이터를 목적지까지 경로를 찾아 전송. 주소(IP) 설정 및 경로(route) 선택.

4. **전송 계층(Transport Layer)**: 데이터 전송 및 전송 속도 조절, 오류 발생 부분 수정. 주로 TCP 프로토콜 사용.

5. **세션 계층(Session Layer)**: 네트워크의 양쪽 연결을 관리하고 지속적으로 연결. TCP/IP의 세션을 생성 및 해제.

6. **표현 계층(Presentation Layer)**: 데이터 인코딩 및 디코딩. JPEG, TIFF, GIF 등의 포맷 지원.

7. **응용 계층(Application Layer)**: 사용자가 네트워크에 접근할 수 있도록 도와줌. 메일 전송, 인터넷 접속 등.

**UML (Unified Modeling Language)**

**정의**: 통합 모델링 언어로, 시스템을 모델로 표현해주는 대표적인 모델링 언어 **다이어그램 종류**:

1. **구조 다이어그램 (Structure Diagram)**:

o 클래스 다이어그램

o 객체 다이어그램

o 복합체 구조 다이어그램

o 배치 다이어그램

o 컴포넌트 다이어그램

o 패키지 다이어그램

2. **행위 다이어그램 (Behavior Diagram)**:

o 활동 다이어그램

o 상태 머신 다이어그램

o 유즈 케이스 다이어그램

o 상호작용 다이어그램

**클래스 다이어그램 (Class Diagram)**

**정의**: 시간에 따라 변하지 않는 시스템의 정적인 면을 보여주는 대표적인 UML 구조 다이어그램. 시스템을 구성하는 클래스들 사이의 관계를 표현. **관계 유형**:

• 연관 관계 (association)

• 일반화 관계 (generalization)

• 집약 관계 (aggregation)

• 합성 관계 (composition)

• 의존 관계 (dependency)

• 실체화 관계 (realization)

**Factory Method 패턴**

**정의**: 객체지향 디자인 패턴 중 하나로, 부모(상위) 클래스에 알려지지 않은 구체 클래스를 생성하는 패턴. 자식(하위) 클래스가 어떤 객체를 생성할지를 결정하도록 함. **목적**: 부모(상위) 클래스 코드에 구체 클래스 이름을 감추기 위한 방법으로 사용.

**파일 구조**

**정의**: 파일을 구성하는 레코드들이 보조 기억 장치에 편성되는 방식 **접근 방식**: 순차 접근, 인덱스 접근, 해싱 접근

**통합 테스트**

**정의**: 모듈을 통합하는 과정에서 모듈 간 호환성 문제를 찾아내기 위해 수행되는 테스트. 모듈 간의 인터페이스가 올바르게 작동하는지 테스트. **방식**:

1. **빅뱅(Big Bang) 접근법**: 모든 구성 요소들을 한꺼번에 통합 후 테스트 수행

2. **점증적(incremental) 접근법**: 논리적으로 연관된 두 개 또는 그 이상의 모듈을 조합하여 테스트

o **하향식(Top Down) 접근법**: 소프트웨어 시스템의 제어 흐름이 위에서 아래로 발생 (스텁 필요)

o **상향식(Bottom Up) 접근법**: 모든 모듈들이 테스트될 때까지 더 낮은 레벨에 있는 각 모듈은 더 위에 있는 모듈과 함께 테스트 (드라이버 필요)

o **샌드위치 접근법**: 하향식과 상향식 접근법의 조합

**Redo**

**정의**: 오류가 발생하기 전까지의 사항을 로그로 기록해 놓고, 이전 상태로 되돌아간 후, 실패가 발생하기 전까지의 과정을 그대로 따라가는 현상

**Undo**

**정의**: 작업을 취소하여 트랜잭션을 이전 상태로 되돌리는 것

**RAID (Redundant Array of Independent Disks)**

**정의**: 여러 개의 하드 디스크 드라이브(HDD)나 SSD를 결합하여 하나의 논리적 단위로 만드는 데이터 저장 기술로, 성능 향상, 데이터 중복성, 데이터 가용성을 제공하기 위해 개발됨. 여러 RAID 레벨이 있으며, 각 레벨은 특정한 방식으로 데이터를 분산하고 중복함.

**주요 RAID 레벨**:

• **RAID 0 (스트라이핑)**:

o 데이터는 여러 디스크에 스트라이핑(분산)되어 저장됨.

o 성능이 향상되지만, 데이터 중복성이 없기 때문에 하나의 디스크가 고장나면 모든 데이터가 손실됨.

• **RAID 1 (미러링)**:

o 데이터는 동일하게 두 개 이상의 디스크에 복제됨.

o 데이터 중복성을 제공하여 하나의 디스크가 고장나더라도 다른 디스크에서 데이터를 복구할 수 있음.

o 성능은 향상되지 않지만, 안전성이 높음.

• **RAID 5 (패리티 스트라이핑)**:

o 데이터와 패리티(오류 검출 및 수정 정보)를 여러 디스크에 분산 저장함.

o 최소 3개의 디스크가 필요하며, 하나의 디스크가 고장나더라도 패리티를 통해 데이터를 복구할 수 있음.

o 성능과 데이터 중복성을 모두 제공.

• **RAID 6 (이중 패리티 스트라이핑)**:

o RAID 5와 유사하지만, 패리티 블록이 두 개 있어 두 개의 디스크가 고장나더라도 데이터를 복구할 수 있음.

o 최소 4개의 디스크가 필요.

• **RAID 10 (미러링 + 스트라이핑)**:

o RAID 1과 RAID 0의 결합으로, 데이터는 먼저 미러링되고 그 후에 스트라이핑됨.

o 성능과 데이터 중복성을 모두 제공하지만, 많은 디스크가 필요.

**리스트 함수**

• **extend**: 요소를 확장해준다는 의미로, 모든 항목을 하나의 요소로 추가

• **pop**: 리스트 내부 요소를 꺼내주는 함수로, 그 요소는 리스트 안에서 삭제하고 그 값을 반환

• **reverse**: 리스트 내부의 요소의 순서를 뒤집는 역할

**TKIP (Temporal Key Integrity Protocol)**

**정의**: 임시 키 무결성 프로토콜

**코드 분석 기법**

• **static**: 소스 코드의 실행 없이, 코드의 의미를 분석해 결함을 찾아내는 원시적 코드 분석 기법

• **dynamic**: 소스 코드를 실행하여 프로그램 동작이나 반응을 추적하고 코드에 존재하는 메모리 누수, 스레드 결함 등을 분석

**JUnit**

**정의**: 자바 프로그래밍 언어를 이용한 xUnit의 테스트 기법으로, 단위 테스트를 쉽게 해주는 테스트용 Framework

**ISMS (Information Security Management System)**

**정의**: 정보보호 관리체계의 영문 약자

**데이터베이스 키**

• **슈퍼키**: 유일성 속성을 가짐.

• **후보키**: 유일성과 최소성 속성을 가짐.

**watering hole**

**정의**: APT 공격에서 주로 쓰이는 공격으로, 공격 대상이 방문할 가능성이 있는 합법적인 웹 사이트를 미리 감염시킨 뒤, 잠복하고 있다가 공격 대상이 방문하면 대상의 컴퓨터에 악성코드를 설치하는 방식

**관계해석**

**정의**: 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때 계산 수식을 사용. 수학의 predicate calculus에 기반을 두고 있으며, 관계 데이터 모델의 제안자인 Codd가 수학에 가까운 기반을 두고 특별히 관계 데이터베이스를 위해 제안. **특성**: 비절차적 **종류**: 튜플 관계해석, 도메인 관계해석

**IDEA (International Data Encryption Algorithm)**

**정의**: Xuejia Lai와 James Messey가 만든 알고리즘으로, 1991년에 제작된 블록 암호 알고리즘 **특성**: 64비트 블록을 128비트의 키를 이용하여 8개의 라운드로 구성됨.

**SKIPJACK**

**정의**: 미국의 NSA에서 개발한 Clipper 칩에 내장되는 블록 알고리즘 **특성**: 전화기와 같은 음성을 암호화하는 데 주로 사용되며, 64비트 입출력에 80비트의 키, 총 32 라운드를 가짐.

**VPN (Virtual Private Network)**

**정의**: 인터넷을 통해 디바이스 간에 사설 네트워크 연결을 생성하며, 퍼블릭 네트워크를 통해 데이터를 안전하게 익명으로 전송하는 데 사용됨. 사용자 IP 주소를 마스킹하고 데이터를 암호화하여 수신 권한이 없는 사람이 읽을 수 없도록 함.

**베타 테스트**

**정의**: 하드웨어나 소프트웨어의 개발 단계에서 상용화하기 전에 실시하는 제품 검사 작업. 선발된 잠재 고객으로 하여금 일정 기간 무료로 사용하게 한 후에 나타난 여러 가지 오류를 수정, 보완함. 공식적인 제품으로 발매하기 이전에 최종적으로 실시하는 검사 작업.

**알파 테스트**

**정의**: 새로운 제품 개발 과정에서 이루어지는 첫 번째 테스트. 시제품이 운영되는 동안의 신제품 연구와 개발 과정 단계에서 초기 작동의 결과를 평가하는 수단이며, 개발 회사 내부에서 이루어지는 테스트로서 단위 테스트, 구성 테스트, 시스템 테스트 등을 포함.

**Regression**

**정의**: 오류를 제거하거나 수정한 시스템이나 시스템 컴포넌트 또는 프로그램이 오류 제거와 수정에 의해 새로이 유입된 오류가 없는지를 확인하는 일종의 반복 시험. 수정·변경된 시스템이나 시스템 컴포넌트 또는 프로그램이 명세된 요구 사항을 충족시키는지를 확인하는 시험의 한 형태.

**SOLID 원칙**

**정의**: 객체 지향 프로그래밍(OOP)에서 소프트웨어 설계를 개선하기 위한 다섯 가지 기본 원칙으로, 로버트 C. 마틴(Robert C. Martin)에 의해 제안되었으며 유지보수성, 확장성, 이해 용이성을 높이는 데 도움을 줌. SOLID는 다섯 가지 원칙의 머리글자를 따서 만든 약어.

1. **Single Responsibility Principle (SRP) - 단일 책임 원칙**:

o 클래스는 하나의 책임만 가져야 함. 즉, 클래스는 단 하나의 기능만을 가져야 하며, 변경의 이유도 하나여야 함.

o 예시: Invoice 클래스는 청구서 관련 작업만을 처리해야 하며, 프린트 관련 작업은 다른 클래스에서 처리해야 함.

2. **Open/Closed Principle (OCP) - 개방/폐쇄 원칙**:

o 소프트웨어 개체(클래스, 모듈, 함수 등)는 확장에 대해서는 열려 있어야 하지만, 수정에 대해서는 닫혀 있어야 함.

o 예시: 새로운 기능을 추가할 때 기존 코드를 수정하지 않고 확장할 수 있도록 설계해야 함. 예를 들어, 새로운 할인 정책을 추가할 때 기존 코드 수정 없이 정책 클래스를 추가하는 방식.

3. **Liskov Substitution Principle (LSP) - 리스코프 치환 원칙**:

o 자식 클래스는 언제나 부모 클래스를 대체할 수 있어야 함. 즉, 자식 클래스는 부모 클래스가 기대하는 모든 동작을 수행해야 함.

o 예시: Rectangle 클래스와 이를 상속받은 Square 클래스가 있을 때, Square 클래스가 Rectangle 클래스의 동작을 모두 수행할 수 있어야 함.

4. **Interface Segregation Principle (ISP) - 인터페이스 분리 원칙**:

o 클라이언트는 자신이 사용하지 않는 인터페이스에 의존하지 않아야 함. 즉, 하나의 일반적인 인터페이스보다는 여러 개의 구체적인 인터페이스가 더 나음.

o 예시: Worker 인터페이스를 IWorker와 IEater로 분리하여, 작업 관련 기능과 식사 관련 기능을 각각의 인터페이스로 나누는 것이 좋음.

5. **Dependency Inversion Principle (DIP) - 의존 역전 원칙**:

o 고수준 모듈은 저수준 모듈에 의존해서는 안 되며, 둘 다 추상화에 의존해야 함. 추상화는 구체적인 사항에 의존해서는 안 됨.

o 예시: OrderProcessor 클래스가 PaymentProcessor 인터페이스에 의존하고, 실제 구현 클래스는 이 인터페이스를 구현하도록 설계함.

**네트워크 라우팅 프로토콜**

• **IGP (Interior Gateway Protocol)**:

o 자율 시스템(AS) 내의 라우터 간 경로 설정.

o 같은 도메인 내에 존재하는 라우터는 도메인 내부 라우터가 되고, 도메인 외부에 존재하는 라우터는 도메인 외부 라우터가 됨. 도메인 내부 경로 설정을 가리킴.

• **EGP (Exterior Gateway Protocol)**:

o 다른 도메인 간 경로 설정 정보 교환.

o 다른 도메인 사이에 라우팅 시 정리된 관리가 거의 없고 많은 경우에 신용도가 매우 낮아 빠른 수행보다는 보안과 제어가 본래의 목적임.

• **OSPF (Open Shortest Path First)**:

o RIP보다 큰 네트워크에서 사용 가능한 라우팅 프로토콜.

o 규모가 크고 복잡한 TCP/IP 네트워크에서 RIP의 단점을 개선한 라우팅 프로토콜로써 RIP에 비해 자세한 제어가 가능하고, 관리 정보의 트래픽도 줄일 수 있음.

• **BGP (Border Gateway Protocol)**:

o 다른 AS 간 라우팅 정보 교환을 위한 외부 게이트웨이 프로토콜.

o 각 목적지에 대한 전체 경로가 포함되며, 다른 시스템과 교환하는 네트워크 도달 가능성 정보의 데이터베이스를 유지. 네트워크 도달 가능성 정보를 사용하여 AS 연결 그래프를 구성하며, 이를 통해 라우팅 루프를 제거하고 AS 수준에서 정책 결정을 실행할 수 있음.

**함수적 종속성**

1. **Full Functional Dependency (완전 함수적 종속성)**:

o 정의: 속성 집합 A가 속성 집합 B를 결정할 때, B가 A의 모든 속성에 종속되어 있고 A의 부분 집합에는 종속되지 않는 경우.

2. **Partial Functional Dependency (부분 함수적 종속성)**:

o 정의: 속성 집합 A가 속성 집합 B를 결정할 때, B가 A의 모든 속성에 종속되는 것이 아니라, A의 부분 집합에도 종속되는 경우.

3. **Transitive Functional Dependency (이행 함수적 종속성)**:

o 정의: 속성 집합 A가 속성 집합 B를 결정하고, 속성 집합 B가 속성 집합 C를 결정할 때, 속성 집합 A가 직접적으로 속성 집합 C를 결정하지 않지만, A → C가 성립하는 경우.

**HTTP (Hypertext Transfer Protocol)**

**정의**: 인터넷에서, 웹 서버와 사용자의 인터넷 브라우저 사이에 문서를 전송하기 위해 사용되는 통신 규약. 하이퍼텍스트 문서를 교환하기 위해 사용되며, 이 규약에 맞춰 개발된 시스템끼리 정보를 교환할 수 있음.

**Hypertext**

**정의**: 문자, 그래픽, 음성 및 영상을 하나의 연상 거미집(Web of Association)과 같이 서로 연결시켜, 제시된 순서에 관계없이 이용자가 관련된 정보를 검색할 수 있도록 하는 정보 제공 방법. 한 페이지에서 링크된 순서에 상관없이 사용자들이 원하는 정보를 클릭함으로써 원하는 정보에 쉽게 접근하는 방식.

**HTML (Hypertext Markup Language)**

**정의**: 웹 페이지 표시를 위해 개발된 지배적인 마크업 언어. 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타내며, 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공함.

**관계대수**

• **Selection (σ, 시그마)**: 특정 조건을 만족하는 튜플(행)을 선택합니다.

• **Projection (π, 파이)**: 특정 속성(열)만을 선택합니다.

• **Union (∪)**: 두 릴레이션의 합집합을 구합니다.

• **Set Difference (−)**: 한 릴레이션에서 다른 릴레이션을 뺀 차집합을 구합니다.

• **Cartesian Product (×)**: 두 릴레이션의 모든 가능한 튜플 쌍을 구합니다.

• **Rename (ρ, 로)**: 릴레이션의 이름이나 속성 이름을 변경합니다.

• **Join (⨝)**: 두 릴레이션을 특정 조건에 따라 결합합니다.

• **Division (÷)**: 한 릴레이션을 다른 릴레이션으로 나누어, 나누어진 릴레이션의 모든 튜플을 포함하는 튜플을 찾습니다.

**사회공학 (Social Engineering)**

**정의**: 보안학적 측면에서 기술적인 방법이 아닌 사람들 간의 기본적인 신뢰를 기반으로 사람을 속여 비밀 정보를 획득하는 기법.

**다크 데이터 (Dark Data)**

**정의**: 빅데이터와 비슷하지만, 구조화돼 있지 않고 더는 사용하지 않는 ‘죽은’ 데이터를 의미. 일반적으로 정보를 수집해 저장한 이후 분석이나 특별한 목적을 위해 활용하지 않으며, 저장공간만 차지하고 이러한 이유로 심각한 보안 위험을 초래할 수 있음.

**디자인 패턴**

• **Bridge**:

o 기능을 처리하는 클래스와 구현을 담당하는 추상 클래스로 구별.

o 구현뿐 아니라 추상화도 독립적 변경이 필요할 때 사용.

o 기존 시스템에 부수적인 새로운 기능들을 지속적으로 추가할 때 유용.

o 새로운 인터페이스를 정의하여 기존 프로그램의 변경 없이 기능을 확장할 수 있음.

• **Observer**:

o 한 객체의 상태가 변화하면 객체에 상속되어 있는 다른 객체들에게 변화된 상태를 전달.

o 일대다 관계를 가지며, 주로 분산된 시스템 간에 이벤트를 생성·발행(Publish)하고, 이를 수신(Subscribe)해야 할 때 이용.

**SIEM (Security Information and Event Management)**

**정의**: 머신러닝 기술을 이용하여 IT 시스템에서 발생하는 대량의 로그를 통합 관리 및 분석하여 사전에 위협에 대응하는 보안 솔루션. **기능**:

• 서로 다른 기종의 보안 솔루션 로그 및 이벤트를 중앙에서 통합 수집하여 분석.

• 네트워크 상태의 모니터링 및 이상 징후 미리 감지.

• 네트워크 이기종 장비 간 로그의 상관관계 분석.

• 이상 징후의 행위 기반 및 문맥 기반 분석 기능.

• 각 로그의 상관관계를 조건식에 따라 검색하여 분석.

• 이벤트 및 로그의 이상 패턴을 인식해 잠재적 위협 발생 시 알림 기능.

**SSO (Single Sign-On)**

**정의**: 여러 개의 사이트에서 한 번의 로그인으로 여러 가지 다른 사이트들을 자동적으로 접속하여 이용하는 방법. **설명**:

• 서로 다른 시스템 및 사이트에서 각각의 사용자 정보를 관리.

• 하나의 사용자 정보를 기반으로 여러 시스템을 하나의 통합 인증을 사용하게 함.

• 하나의 시스템에서 인증할 경우 타 시스템에서는 인증 정보가 있는지 확인하고, 있으면 로그인 처리, 없는 경우 다시 통합 인증.

**Trustzone**

**정의**: ARM사에서 개발한 하드웨어 기반의 보안 기술로, 프로세서 안에 독립적인 보안 구역을 따로 두어 중요한 정보를 보호.

**Typosquatting**

**정의**: 사용자들이 사이트에 접속할 때 주소를 잘못 입력하거나 철자를 빠뜨리는 실수를 이용하기 위해 유사한 유명 도메인을 미리 등록하는 일. URL 하이재킹(hijacking)이라고도 함.

**스케줄링**

1. **선입선출 스케줄링 (FCFS, First-Come, First-Served)**

o **설명**: 프로세스가 도착한 순서대로 CPU를 할당받는 방식.

o **특징**:

 **장점**: 구현이 간단하고 이해하기 쉬움.

 **단점**: 긴 작업이 먼저 도착하면 짧은 작업들이 오랫동안 기다리게 되는 "Convoy Effect"가 발생할 수 있음.

2. **최단 작업 우선 스케줄링 (SJF, Shortest Job First)**

o **설명**: 실행 시간이 가장 짧은 작업을 먼저 실행하는 방식.

o **특징**:

 **장점**: 평균 대기 시간을 최소화할 수 있음.

 **단점**: 실행 시간을 미리 알기 어렵고, 긴 작업이 계속해서 뒤로 밀리는 "Starvation" 문제가 발생할 수 있음.

3. **라운드 로빈 스케줄링 (RR, Round Robin)**

o **설명**: 모든 프로세스가 동일한 시간 할당량(Time Quantum)을 가지고, 순환하며 CPU를 할당받는 방식.

o **특징**:

 **장점**: 모든 프로세스가 공평하게 CPU를 할당받을 수 있음.

 **단점**: 시간 할당량이 너무 짧으면 Context Switching 오버헤드가 커지고, 너무 길면 대기 시간이 늘어날 수 있음.

4. **우선순위 스케줄링 (Priority Scheduling)**

o **설명**: 각 프로세스에 우선순위를 부여하고, 우선순위가 높은 프로세스에 CPU를 할당하는 방식.

o **특징**:

 **장점**: 중요한 작업을 빨리 처리할 수 있음.

 **단점**: 우선순위가 낮은 작업이 계속해서 실행되지 않는 "Starvation" 문제가 발생할 수 있음.

5. **다단계 큐 스케줄링 (Multilevel Queue Scheduling)**

o **설명**: 프로세스들을 여러 개의 큐로 분류하고, 각 큐에 다른 스케줄링 알고리즘을 적용하는 방식.

o **특징**:

 **장점**: 서로 다른 특성을 가진 작업들을 효율적으로 관리할 수 있음.

 **단점**: 큐 간의 우선순위 조정이 어렵고, 특정 큐가 과도하게 자원을 사용할 수 있음.

6. **다단계 피드백 큐 스케줄링 (Multilevel Feedback Queue Scheduling)**

o **설명**: 프로세스가 CPU를 사용한 시간에 따라 큐를 이동하며, 각 큐에 다른 시간 할당량과 스케줄링 알고리즘을 적용하는 방식.

o **특징**:

 **장점**: 다양한 유형의 작업을 효율적으로 처리할 수 있음.

 **단점**: 설정이 복잡하고, 잘못된 파라미터 설정 시 성능 저하가 발생할 수 있음.

7. **최단 잔여 시간 우선 스케줄링 (SRT 또는 SRTF, Shortest Remaining Time First)**

o **설명**: 현재 실행 중인 프로세스와 준비 큐에 있는 프로세스 중에서 남은 실행 시간이 가장 짧은 프로세스를 선택하여 실행하는 선점형 스케줄링 알고리즘.

o **특징**:

 **장점**: 평균 대기 시간을 최소화할 수 있음.

 **단점**: 긴 작업들이 계속해서 중단될 수 있어 "Starvation"이 발생할 수 있음. 프로세스의 남은 실행 시간을 정확하게 예측하기 어렵고, 선점이 자주 발생하면 Context Switching 오버헤드가 증가할 수 있음.

8. **기한 기반 스케줄링 (Deadline Scheduling)**

o **설명**: 각 작업에 기한을 부여하고, 기한을 맞추기 위해 스케줄링하는 방식.

o **특징**:

 **장점**: 실시간 시스템에서 기한을 맞춰 작업을 완료할 수 있음.

 **단점**: 모든 작업의 기한을 맞추는 것이 불가능할 수 있으며, 우선순위가 낮은 작업이 무시될 수 있음.

**스케줄링의 목표**:

• **공정성(Fairness)**: 모든 프로세스가 공평하게 CPU를 할당받아야 함.

• **효율성(Efficiency)**: CPU 사용률을 최대화하고, 프로세스 처리 시간을 최소화해야 함.

• **응답 시간(Response Time)**: 프로세스가 처음 실행되기까지의 시간을 최소화해야 함.

• **대기 시간(Waiting Time)**: 프로세스가 준비 큐에서 대기하는 시간을 최소화해야 함.

• **처리량(Throughput)**: 단위 시간당 완료된 프로세스의 수를 최대화해야 함.

**UML (Unified Modeling Language)**

**정의**: 통합 모델링 언어로써, 시스템을 모델로 표현해주는 대표적인 모델링 언어. **구성 요소**:

• **사물**: UML 다이어그램을 구성하는 요소.

• **관계**: 사물 간의 상호작용을 나타냄.

• **다이어그램**: 시스템의 구조와 행위를 표현.

**구조 다이어그램 중 클래스 다이어그램**:

• **정의**: 시스템에서 사용되는 객체 타입을 정의하고, 그들 간의 존재하는 정적인 관계를 다양한 방식으로 표현한 다이어그램.

• **인터페이스**: 클래스와 같은 기타 모델 요소 또는 컴포넌트가 구현해야 하는 오퍼레이션 세트를 정의하는 모델 요소.

**L2TP (Layer 2 Tunneling Protocol)**

• **정의**: 2계층(데이터링크 계층)에서 구현되는 터널링 기술 중 하나.

• **특징**: L2F와 PPTP가 결합된 프로토콜로 VPN과 인터넷 서비스 제공자가 이용.

• **보안**: IPsec과 함께 사용하면 PPTP보다 훨씬 안전하지만, 보안보다는 익명화에 더 적합.

**SSH (Secure Shell)**

• **정의**: 네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 파일을 복사할 수 있도록 해주는 응용 프로그램 또는 프로토콜.

• **기능**: 보안 접속을 통한 rsh, rcp, rlogin, rexec, telnet, ftp 등을 제공하며, IP spoofing을 방지하기 위한 기능 제공.

• **포트 번호**: 기본적으로 포트 22번 사용.

**악성 프로그램 종류**

• **웜 (Worm)**:

o **정의**: 감염된 컴퓨터 시스템에서 스스로 복제해 다른 컴퓨터로 복사본을 확산시킬 수 있는 악성 프로그램.

o **특징**: 독자적으로 실행되며 다른 실행 프로그램이 필요하지 않음. 자가 복제 및 네트워크를 통한 전파 가능.

• **트로이 목마 (Trojan Horse)**:

o **정의**: 정상적인 프로그램으로 가장해 사용자가 직접 컴퓨터에 설치하도록 위장하는 악성 프로그램.

o **특징**: 백도어를 만들어 공격자가 침입할 수 있도록 하거나 시스템을 파괴함. 스스로를 복제하는 능력은 없음.

• **바이러스 (Virus)**:

o **정의**: 대상(정상 파일)을 감염시키는 형태로 실행되는 악성코드.

o **특징**: 감염시킬 대상이 존재하지 않으면 실행되지 않으며, 다른 정상 파일로의 자기 복제 기능 있음. 네트워크를 통해 전파되지 않음.

**Proxy (프록시)**

• **정의**: 다른 무언가와 이어지는 인터페이스 역할을 하는 클래스.

• **기능**: 실제 객체를 호출하면 행위를 중간에 가로채서 다른 동작을 수행하는 객체로 변경. 객체를 정교하게 제어해야 하거나 객체 참조가 필요한 경우 사용.

• **용도**: 대리 작업을 중간 단계에 삽입하거나 객체의 실행 시점을 관리.

**튜플 (Tuple)**

• **정의**: 셀 수 있는 수량의 순서 있는 열거.

• **특징**:

o 요소의 집합, 혹은 테이블에서의 행을 가리킴.

o 중복이 허용될 수 있음.

o 숫자, 문자, 객체, 배열, 튜플 안의 튜플 등 여러 객체를 담을 수 있음.

**릴레이션 인스턴스 (Relation Instance)**

• **정의**: 어느 한 시점에 릴레이션의 내용(상태), 즉 저장된 데이터 전체를 의미.

• **특징**: 단순히 릴레이션 또는 릴레이션 외연(Relation Extension)라고도 함.

**카디널리티 (Cardinality)**

• **정의**: 특정 데이터 집합의 유니크(Unique)한 값의 개수.

• **특징**: 전체 행에 대한 특정 컬럼의 중복 수치를 나타내는 지표.

**외부 스키마 (External Schema)**

• **정의**: 사용자나 응용 프로그래머가 개인의 입장에서 필요한 데이터베이스의 논리적 구조를 정의.

• **특징**:

o 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있기 때문에 서브 스키마라고도 함.

o 하나의 데이터베이스 시스템에는 여러 개의 외부 스키마가 존재할 수 있음.

o 하나의 외부 스키마를 여러 응용 프로그램 혹은 사용자가 공유할 수 있음.

**개념 스키마 (Conceptual Schema)**

• **정의**: 데이터베이스의 전체적인 논리적 구조로, 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재.

• **특징**:

o 개체 간의 관계(Relationship)와 제약 조건을 나타내고 데이터베이스의 접근 권한, 보안 및 무결성 규칙에 관한 명세를 정의.

o 데이터베이스 파일에 저장되는 데이터의 형태를 나타냄.

o 기관이나 조직체의 관점에서 데이터베이스를 정의.

**내부 스키마 (Internal Schema)**

• **정의**: 물리적인 저장장치 입장에서 데이터가 저장되는 방법을 기술.

• **특징**:

o 실제 데이터베이스에 저장될 레코드의 물리적인 구조를 정의.

o 저장 데이터 항목의 표현방법, 내부 레코드의 물리적 순서, 인덱스 유/무 등을 나타냄.

o 시스템 프로그래머나 시스템 설계자가 관리.

**테스트 커버리지 유형**

1. **구문 커버리지 (Statement Coverage)**

o **정의**: 소스 코드의 각 구문(명령문)이 최소한 한 번 실행되었는지를 측정.

o **장점**: 간단하고 직접적인 방법으로 테스트의 범위를 평가할 수 있음.

o **단점**: 모든 구문이 실행되었다고 해서 모든 논리적 경로가 테스트된 것은 아님.

2. **경로 커버리지 (Path Coverage)**

o **정의**: 코드 내의 모든 가능한 실행 경로가 테스트되었는지를 측정.

o **장점**: 가장 포괄적인 테스트 커버리지 방법으로, 모든 경로를 테스트하여 결함을 최소화할 수 있음.

o **단점**: 가능한 경로의 수가 많아질수록 테스트의 복잡도와 비용이 증가.

3. **조건/결정 커버리지 (Condition/Decision Coverage)**

o **정의**: 조건 커버리지와 결정 커버리지를 결합한 형태로, 각 조건이 참과 거짓을 한 번씩 갖도록 하고, 각 결정(분기)이 참과 거짓을 한 번씩 갖도록 테스트.

o **장점**: 조건과 결정을 동시에 테스트하여 더 많은 결함을 발견할 수 있음.

o **단점**: 모든 조건 조합을 테스트하지는 않으므로 여전히 일부 결함이 남을 수 있음.

4. **변형 조건/결정 커버리지 (Modified Condition/Decision Coverage, MC/DC)**

o **정의**: 각 조건이 결정을 독립적으로 영향을 미치는지 테스트하여 모든 조건의 모든 결과 조합이 최소한 한 번씩 실행되도록 함.

o **장점**: 안전-critical 시스템에서 요구되는 높은 수준의 테스트 커버리지 방법.

o **단점**: 매우 복잡하고 비용이 많이 들 수 있음.

5. **다중 조건 커버리지 (Multiple Condition Coverage)**

o **정의**: 각 조건이 모든 가능한 조합(참/거짓)을 갖도록 테스트.

o **장점**: 모든 조건의 조합을 테스트하여 가장 철저한 검사를 수행.

o **단점**: 가능한 조건의 조합 수가 기하급수적으로 증가하여 현실적으로 모든 조합을 테스트하는 것이 어렵음.

6. **결정 커버리지 (Decision Coverage)**

o **정의**: 코드 내의 모든 결정(분기점)이 참과 거짓을 모두 한 번씩 갖도록 테스트.

o **장점**: 분기점에서 발생할 수 있는 오류를 줄일 수 있음.

o **단점**: 결정 내의 개별 조건까지는 테스트하지 않음.

7. **조건 커버리지 (Condition Coverage)**

o **정의**: 조건문의 각 개별 조건이 참과 거짓을 모두 한 번씩 갖도록 테스트.

o **장점**: 조건문의 모든 경우를 테스트하여 결함을 줄일 수 있음.

o **단점**: 각 조건의 조합까지는 고려하지 않으므로 일부 결함이 남을 수 있음.

**템퍼프루핑 (Tamper-Proofing)**

• **정의**: 소프트웨어가 불법으로 변경되었을 경우 정상 수행되지 않게 하는 기법.

• **특징**:

o 소프트웨어 변조 공격을 방어하기 위해 프로그램에 변조 검증 코드를 삽입.

o 변조 검증 코드의 용도는 프로그램이 변경되었는지를 탐지하고, 변조가 탐지되었을 경우 프로그램이 실행되지 않게 함.

o 디지털 콘텐츠 보호, 전자 상거래, 소프트웨어 보호 등 다양한 산업 분야에 적용.

**생성 패턴 (Creational Patterns)**

1. **Singleton (싱글턴 패턴)**

o **정의**: 클래스의 인스턴스를 하나만 생성하고, 그 인스턴스에 대한 전역 접근을 제공.

o **장점**: 인스턴스가 하나임을 보장하여 자원을 절약하고, 글로벌 접근점을 제공.

o **예시**: 데이터베이스 연결 관리, 설정 관리 객체 등.

2. **Factory Method (팩토리 메서드 패턴)**

o **정의**: 객체 생성 작업을 서브클래스에서 수행하도록 하는 패턴. 상위 클래스는 객체 생성을 위한 인터페이스만 제공하고, 실제 객체 생성은 하위 클래스에서 처리.

o **장점**: 객체 생성 코드를 캡슐화하여 코드의 유연성과 확장성을 높임.

o **예시**: 다양한 제품군을 생성하는 객체 생성 코드.

3. **Builder (빌더 패턴)**

o **정의**: 복잡한 객체를 단계별로 생성할 수 있도록 하는 패턴. 다양한 구성 요소를 사용하여 동일한 객체를 다르게 생성할 수 있음.

o **장점**: 객체 생성의 복잡성을 줄이고, 다양한 구성 요소를 독립적으로 다룰 수 있음.

o **예시**: 복잡한 제품의 구성 요소를 설정하는 객체 생성 코드.

**구조 패턴 (Structural Patterns)**

1. **Adapter (어댑터 패턴)**

o **정의**: 인터페이스 호환성이 없는 클래스들을 함께 동작할 수 있도록 변환하는 패턴.

o **장점**: 기존 코드를 수정하지 않고도 인터페이스를 호환시킬 수 있음.

o **예시**: 서로 다른 인터페이스를 사용하는 라이브러리 통합.

2. **Bridge (브리지 패턴)**

o **정의**: 기능의 구현과 추상을 분리하여 각기 독립적으로 확장할 수 있게 하는 패턴.

o **장점**: 추상화와 구현을 분리하여 독립적인 개발과 변경이 가능.

o **예시**: 운영체제와 독립적인 그래픽 라이브러리 설계.

3. **Decorator (데코레이터 패턴)**

o **정의**: 객체에 추가적인 기능을 동적으로 추가할 수 있도록 하는 패턴.

o **장점**: 서브클래스를 만들지 않고도 객체에 기능을 추가할 수 있음.

o **예시**: 입력 스트림에 다양한 기능을 추가하는 Java I/O 라이브러리.

**행위 패턴 (Behavioral Patterns)**

1. **Observer (옵서버 패턴)**

o **정의**: 객체의 상태 변화에 따라 의존 관계에 있는 다른 객체들이 통지(알림)를 받도록 하는 패턴.

o **장점**: 객체 간의 느슨한 결합을 유지하면서도 이벤트를 통지할 수 있음.

o **예시**: GUI 이벤트 처리기, 뉴스 구독 시스템.

2. **Strategy (전략 패턴)**

o **정의**: 알고리즘을 클래스별로 캡슐화하여, 런타임에 전략을 바꿀 수 있게 하는 패턴.

o **장점**: 알고리즘을 독립적으로 교체할 수 있어 코드의 유연성과 재사용성이 높음.

o **예시**: 정렬 알고리즘 선택, 로그 기록 방법 설정.

3. **Visitor (비지터 패턴)**

o **정의**: 객체 구조를 변경하지 않으면서 객체에 수행할 연산을 추가할 수 있도록 하는 패턴.

o **장점**: 데이터 구조와 연산을 분리하여 새로운 연산을 추가할 때 구조를 변경할 필요가 없음.

o **예시**: 컴파일러에서 각 노드에 대해 수행하는 작업 분리.

**Hamming Code**

**정의**: 데이터 전송 시 1 비트의 에러를 정정할 수 있는 오류 정정부호의 일종으로 미국의 Bell 연구소의 Hamming에 의해 고안됨. 선형 블록 부호 및 순회 부호에 속함.

**FEC (Forward Error Correction)**

**정의**: 송신측이 전송할 문자나 프레임에 부가적 정보(Redundancy)를 첨가하여 전송하고, 수신측이 이 부가적 정보를 이용하여 에러 검출 및 에러 정정을 하는 방식.

**BEC (Business Email Compromise)**

**정의**: 공격자가 전자 메일을 사용하여 상대방이 금전을 보내거나 기밀 회사 정보를 누설하도록 유도하는 사이버 범죄의 일종. 공격자는 신뢰할 수 있는 인물로 가장한 다음 가짜 청구서의 지불을 요청하거나 다른 사기에서 사용할 중요한 데이터를 요구함.

**Parity**

**정의**: 데이터가 저장장치 내의 한 장소에서 다른 장소로 이동되거나, 컴퓨터들 간에 전송될 때 데이터가 유실 또는 손상되었는지 여부를 점검하는 기술과 관련된 용어.

**CRC (Cyclic Redundancy Check)**

**정의**: 네트워크 등을 통하여 데이터를 전송할 때 전송된 데이터에 오류가 있는지를 확인하기 위한 체크값을 결정하는 방식.

**대칭키 및 비대칭키 알고리즘**

• **대칭키**: DES, AES, ARIA, SEED

• **비대칭키**: RSA, ECC

**해시 (Hash)**

• **정의**: 임의의 크기를 가진 데이터(Key)를 고정된 크기의 데이터(Value)로 변화시켜 저장하는 것.

• **특징**:

o 키에 대한 해시값을 사용하여 값을 저장하고, 키-값 쌍의 갯수에 따라 동적으로 크기 증가.

o 해시 값 자체를 인덱스로 사용해 평균 시간복잡도가 O(1)로 매우 빠름.

o 해시 함수는 임의의 길이의 데이터를 고정된 길이의 데이터로 매핑.

o 큰 파일에서 중복되는 레코드를 찾아 데이터베이스 검색이나 테이블 검색의 속도를 가속.

**HDLC 프로토콜**

• **정보 프레임**:

o Seq: 송신용 순서번호

o Next: 응답용 순서번호

o P/F: P가 1로 설정된 경우 주국에서 종국에 데이터 전송을 허용, F가 1로 설정된 경우 종국에서 주국으로 데이터 전송을 의미.

• **감독 프레임**:

o 맨 앞의 필드가 1로 되어 있어 정보 프레임이 아님을 나타냄.

o Type의 경우 2비트로 4가지 종류로 나뉨.

o Seq에 대한 값은 없고, 다음 프레임을 요구하는 Next만 존재.

• **비번호 프레임**:

o 순서 번호가 없는 프레임.

o 첫 번째 비트와 두 번째 비트가 모두 1로 설정.

o Type의 2비트와 Modifier의 3비트를 합쳐 5비트로 종류 구분.

• **비동기균형**: 두 호스트 모두 혼합국으로 동작, 양쪽에서 명령과 응답 전송 가능.

• **비동기응답**: 불균형 모드로 주국의 허락 없이 종국에서 데이터를 전송 가능.

**여러 용어의 정의**

• **NFT (Non-Fungible Token)**:

o **정의**: 대체 불가능 토큰.

o **설명**: 블록체인 기술을 기반으로 한 디지털 자산으로, 각 토큰이 고유한 값을 가지고 있어 상호 교환이 불가능. 예술 작품, 음악, 게임 아이템 등의 소유권을 디지털화하여 표현하는 데 사용됨.

• **JWT (JSON Web Token)**:

o **정의**: JSON 웹 토큰.

o **설명**: JSON 객체를 사용하여 두 시스템 간에 정보를 안전하게 전송하기 위한 개방형 표준(RFC 7519). 주로 사용자 인증 및 정보 교환에 사용되며, 세 부분(헤더, 페이로드, 서명)으로 구성됨.

• **SAML (Security Assertion Markup Language)**:

o **정의**: 보안 주장 마크업 언어.

o **설명**: XML 기반의 개방형 표준으로, 주로 웹 브라우저 단일 로그인(SSO)을 위해 사용됨. SAML을 통해 인증과 권한 부여 정보를 교환 가능.

• **Access Token (액세스 토큰)**:

o **정의**: 접근 토큰.

o **설명**: 사용자 인증 후 발급되는 토큰으로, 사용자가 보호된 리소스에 접근할 수 있는 권한을 부여. 보통 OAuth 2.0 프레임워크 내에서 사용됨.

• **SSO (Single Sign-On)**:

o **정의**: 단일 로그인.

o **설명**: 사용자가 한 번의 로그인으로 여러 애플리케이션이나 시스템에 접근할 수 있도록 하는 인증 방식. 사용자 편의성을 높이고 보안을 강화할 수 있음.

• **OIDC (OpenID Connect)**:

o **정의**: 오픈ID 커넥트.

o **설명**: OAuth 2.0을 기반으로 한 인증 프로토콜로, OAuth 2.0의 권한 부여 기능에 사용자 인증을 추가. 사용자 정보(ID 토큰)를 안전하게 제공하여 신원 확인에 사용.

• **Authentication (인증)**:

o **정의**: 인증.

o **설명**: 사용자의 신원을 확인하는 과정. 비밀번호, 생체 인식, 인증 토큰 등을 통해 수행.

• **OAuth (Open Authorization)**:

o **정의**: 오픈 인증.

o **설명**: 권한 부여를 위한 개방형 표준으로, 사용자가 비밀번호를 제공하지 않고도 타사 애플리케이션이 사용자 정보를 접근할 수 있도록 허용. 주로 접근 토큰을 발급하여 권한 부여.

• **Cookie (쿠키)**:

o **정의**: 쿠키.

o **설명**: 웹 브라우저와 서버 간의 상태 정보를 저장하고 전송하기 위한 작은 데이터 조각. 세션 관리, 사용자 맞춤 설정, 추적 등의 목적으로 사용.

• **Refresh Token (리프레시 토큰)**:

o **정의**: 리프레시 토큰.

o **설명**: 액세스 토큰의 수명을 연장하기 위해 사용되는 토큰. 액세스 토큰이 만료되었을 때 새 액세스 토큰을 발급받기 위해 사용됨.

**ATM (Asynchronous Transfer Mode)**

• **정의**: 자료를 일정한 크기로 정하여 순서대로 전송하는 자료의 전송 방식.

• **특징**:

o 셀이라 부르는 고정 길이 패킷(53바이트)을 이용하여 처리.

o 가상 채널 (Virtual Channel) 기반의 연결 지향 서비스.

o 통계적 다중화 방식을 이용.

**RIP (Routing Information Protocol)**

• **정의**: 최소 Hop count를 파악하여 라우팅하는 프로토콜.

• **특징**:

o 거리와 방향으로 길을 찾아가는 Distance Vector 다이나믹 프로토콜.

o 최단 거리, 즉 Hop count가 적은 경로를 택하여 라우팅.

o 최대 Hop count는 15로, 거리가 짧기 때문에 IGP로 많이 이용.

**PaaS (Platform as a Service)**

**정의**: 서비스로서의 플랫폼 **설명**:

• **플랫폼 제공**: PaaS는 애플리케이션 개발, 실행, 관리에 필요한 플랫폼을 제공. 개발자들은 인프라를 관리할 필요 없이 애플리케이션을 개발, 테스트, 배포할 수 있음.

• **주요 구성 요소**: 운영 체제, 개발 도구, 데이터베이스 관리 시스템, 미들웨어 등.

• **장점**: 개발 시간 단축, 인프라 관리 비용 절감, 확장성 및 유연성 향상.

• **예시**: Google App Engine, Microsoft Azure App Services, Heroku.

**SaaS (Software as a Service)**

**정의**: 서비스로서의 소프트웨어 **설명**:

• **소프트웨어 제공**: SaaS는 클라우드 환경에서 소프트웨어 애플리케이션을 제공하는 모델. 사용자는 소프트웨어를 설치하거나 관리할 필요 없이 인터넷을 통해 접근하고 사용할 수 있음.

• **주요 구성 요소**: 사용자 인터페이스, 애플리케이션 로직, 데이터 저장소 등.

• **장점**: 초기 설치 비용 없음, 유지보수 부담 감소, 접근성과 이동성 향상.

• **예시**: Google Workspace (Gmail, Docs, Sheets), Microsoft Office 365, Salesforce, Dropbox.

**IaaS (Infrastructure as a Service)**

**정의**: 서비스로서의 인프라 **설명**:

• **인프라 제공**: IaaS는 가상화된 컴퓨팅 리소스를 제공하는 모델. 이는 물리적 데이터 센터 대신 가상 서버, 스토리지, 네트워크를 통해 인프라를 제공.

• **주요 구성 요소**: 가상 서버, 스토리지, 네트워킹, 데이터센터 공간 등.

• **장점**: 물리적 하드웨어 구매 및 관리 비용 절감, 확장성 및 유연성 향상, 사용한 만큼 비용 지불.

• **예시**: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP).

**요약**

• **PaaS (Platform as a Service)**: 개발자에게 애플리케이션 개발 및 배포를 위한 플랫폼을 제공. 운영 체제, 개발 도구, 데이터베이스 관리 시스템 등을 포함.

• **SaaS (Software as a Service)**: 사용자가 소프트웨어를 인터넷을 통해 접근하고 사용하는 서비스 모델. 소프트웨어 설치 및 관리를 필요로 하지 않음.

• **IaaS (Infrastructure as a Service)**: 가상화된 컴퓨팅 리소스(서버, 스토리지, 네트워크)를 제공. 물리적 하드웨어 대신 가상 인프라를 사용하여 비용 절감 및 확장성 제공.

• **IaaS < PaaS < SaaS**

**참조 무결성**

**정의**: 관계 데이터베이스 관계 모델에서 두 개의 관련된 관계 변수(테이블) 간의 일관성(데이터 무결성). **설명**:

• 기본 키 또는 키가 아닌 후보 키와 외래 키의 조합으로 강제 적용.

• 외래 키가 선언된 관계 변수의 외래 키를 구성하는 속성(열)의 값은 그 관계 변수의 부모가 되는 관계 변수의 기본 키 값 또는 기본 키가 아닌 후보 키 값으로 존재해야 함.

**조인의 종류**

• **세타 조인 (Theta Join)**: 조인에 참여하는 두 릴레이션의 속성 값을 비교하여 조건을 만족하는 튜플만 반환.

• **동등 조인 (Equi Join)**: 조건이 정확하게 '=' 등호로 일치하는 결과를 반환.

• **자연 조인 (Natural Join)**: 동등 조인에서 조인에 참여한 속성이 두 번 나오지 않도록 중복된 속성을 제거한 결과를 반환.

**Rootkit**

**정의**: 인터넷 공격자의 존재를 숨기면서 이 공격자에게 시스템에 대한 무제한 접근 권한을 부여하는 악성 프로그램. **설명**:

• 해커가 자신의 존재를 숨기면서 허가되지 않은 컴퓨터나 소프트웨어에 접근할 수 있도록 설계된 도구.

• 펌웨어, 가상화 계층 등의 다양한 시스템 영역에서 작동하며, 운영체제의 시스템 콜을 해킹하여 악성코드의 실행 여부를 숨겨 안티바이러스 탐지를 우회할 수 있음.

**APT (Advanced Persistent Threat)**

**특징**:

1. 불특정 다수가 아닌 명확한 표적을 정하여 지속적인 정보 수집 후 공격 감행.

2. 시스템에 직접 침투하는 것뿐 아니라 표적 내부 직원들이 이용하는 다양한 단말을 대상으로 함.

3. Zero-day 취약점, 악성코드 등 다양한 보안 위협 공격 기술 사용.

4. 공격은 침투, 검색, 수집 및 유출의 4단계로 실행되며, 각 단계별로 다양한 공격 기술 사용.

**Abstract Factory**

**정의**: 구체적인 클래스에 의존하지 않고 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공하는 패턴. **설명**:

• 연관성이 있는 객체 군이 여러 개 있을 경우 이를 묶어 추상화하고, 구체적인 상황이 주어지면 팩토리 객체에서 집합으로 묶은 객체 군을 구현화하는 생성 패턴.

• 관련성 있는 여러 종류의 객체를 일관된 방식으로 생성하는 경우에 유용.

• "kit"라고도 불림.