**1. 현행 시스템 파악**

- 구성/기능 및 인터페이스 파악

- 아키텍처 및 소프트웨어 구성 파악

- 하드웨어 및 네트워크 구성 파악

**2. 소프트웨어 생명주기(SDLC) 모델**

시스템의 요구분석 부터 유지보수까지

전 공정을 체계화한 절차이다.

**3. TDD(Test Driven Development) 테스트 기반 개발**

작성해야 하는 프로그램에 대한 테스트를 먼저 수행하고 해당 테스트를 통과할 수 있도록

실제 프로그램의 코드를 작성한다.

**4. 요구공학**

사용자 요구사항에 대한 도출, 분석, 명세, 확인 및 검증하는 구조화된 활동이다.

**5. 짝 프로그래밍(Pair Programming)**

개발자 둘이서 짝으로 코딩하는 원리이다.

**6. Man Month**

한 사람이 1개월 동안 할 수 있는 일의 양을 기준으로 프로젝트 비용을 산정하는 방식이다.

**7. 소프트웨어 아키텍처**

여러 가지 소프트웨어 구성요소와

그 구성요소가 가진 외부에 드러나는 특성

그리고 구성요소 간의 관계를 표현하는 시스템

구조이다.

**8. 요구사항 확인 기법 종류**

요구사항 검토 / 프로토타이핑 / 모델 검증 / 인수 테스트

**9. 프로토타이핑**

새로운 요구사항을 도출하기 위한 수단이다.

**10. 상호 운용성**

다른 목적을 가진 2개 이상의 시스템들이 서로 정보 및 서비스를 교환하면서 효과적으로 운용될 수 있는 시스템 능력이다.

**11. 기능적 요구사항**

시스템에서 필수적으로 제공해야 하는 특정 기능을 정의한다.

**12. 비기능적 요구사항**

시스템의 전체적인 품질이나 기능적 요구사항 구현 시 고려해야 하는 제약사항을 정의한다.

**13. 요구사항 협상**

두 명의 이해관계자가 서로 상충되는 내용을 요구하는 경우, 어느 한쪽을 지지하기 보다는 적절한 시점에서 합의하기 위한 기법이다.

**14. 프로젝트 관리자**

요구사항의 기술적 타당성 분석 결과에 대해

이해관계자들의 이견이 있는 경우

중재하여 합의를 도출해 내는 사람

**15. Rumbaugh(럼바우) 방법론**

객체모델링 - 동적모델링-기능모델링

**16. 애자일 방법론**

고객의 요구사항 변화에 유연하게 대응하기 위해서

일정한 주기를 반복하면서 개발하는 방법이다.

**17. 디자인 패턴**

소프트웨어 공학의 소프트웨어 설계에서

공통으로 발생하는 문제에 대해

자주 쓰이는 설계 방법을 정리한 패턴이다.

**18. 디자인 패턴**

생성 / 구조 / 행위

**19. MVC 패턴**

모델, 뷰, 컨트롤러로 구성되어진 아키텍처이다.

Model은 애플리케이션 정보를 나타내며,

View는 사용자 인터페이스 요소를 나타내고

Controller는 데이터와 비즈니스 로직 사이의 상호동작을 관리한다.

**20. 소프트웨어 아키텍처 패턴**

소프트웨어를 설계할 때 참조할 수 있는

전형적인 해결 방식이다.

**21. 소프트웨어 아키텍처 프레임워크**

소프트웨어 집약적인 시스템에서 아키텍처가 표현해야 하는 내용 및 관계를 제공하는 아키텍처 기술 표준이다.

**22. 분석 모델 검증 방법**

유스케이스 모델 검증 /

개념 수준의 분석 클래스 검증 /

분석 클래스 검증

**23. 분석 모델의 기술적 타당성 검토 항목**

시스템 간 상호 운용성 /

기술적 위험분석 /

성능 및 용량 산정의 적정성 /

IT 시장 성숙도 및 트렌드 부합성

**24. 기술적 위험 분석**

분석 모델을 구현하기 위해서 특정 업체의 기술, 특허, 라이선스에 의존해야 하는지 확인 하는 기법이다.

**25. 유스케이스 모델 검증 방법**

시스템 기능에 대한 유스케이스 모형 상세화 수준 및 적정성 검증을 위해서

액터, 유스케이스, 유스케이스 명세서를 점검하는 기법이다.

**26. 요구사항 검토 방법 종류**

요구사항 목록 확인 /

요구사항 정의서 작성 여부 확인

비기능적 요구사항의 확인 /

타 시스템 연계 및 인터페이스 요구사항 확인

**27. 웹 애플리케이션 서버(WAS)**

서버 계층에서 애플리케이션이 동작 할 수 있는 환경을 제공하고, 안정적인 트랜잭션 처리 와 관리, 다른 이기종 시스템과의 애플리케이션 연동을 지원하는 서버이다.

**28. Protected**

클래스를 구성하는 객체에 대해 동일한 패키지 내에서만 접근이 가능하다.

**29. 자료 결합도**

모듈 간의 인터페이스로 전달되는 파라미터를 통해서만 모듈 상호 작용이 일어나는 결합도를 의미한다.

**30. 형상통제 위원회(Configuartion Control Board)**

형상 관리에 대한 주요 방침을 결정하고

산출물을 검토하며 단계별 의사결정을 수행하는

조직이다.

**31. UI 설계 원칙**

직관성 / 학습성 / 유연성 / 유효성

**32. UI 설계 원칙 - 직관성**

누구나 쉽게 이해하고 사용할 수 있어야 한다.

**33. UI 설계 원칙 - 유연성(Flexibility)**

사용자의 인터렉션을 최대한 포용하고

실수를 방지할 수 있도록 제작해야 한다.

**34. UI(User Interface)**

사용자와 시스템 사이에서 의사소통 할 수 있도록 하는 물리적 가상의 매개체이다.

**35. UI 요구사항**

사용자가 정보시스템을 구축하여 얻고자 하는 최종 목적의 기준이다.

**36. UX(User Experience ; 사용자 경험)**

사용자가 직간접 적으로 경험하면서 느끼고 생각하는 총체적 경험이다.

**37. UI 지침(Guideline)**

UI 표준에 따라 사용자 인터페이스 설계, 개발 시 지켜야 할 세부 사항을 규정하는 것을 의미한다.

**38. UI 흐름 설계**

업무의 흐름이나 업무 수행과 관련된 일련의 클릭에 의한 화면의 위치와 흐름을 흐름도 형식으로 표현하는 활동이다.

**39. UML**

산출물을 명세화, 시각화, 문서화할 때

사용되는 모델링 기술과 방법론을 통합하여 만든

표준화된 범용 모델링 언어이다.

**40. UML 구성요소**

사물 / 관계 / 다이어그램

**41. 사용성 테스트(Usability Test)**

사용자가 직접 제품을 사용하면서 미리 작성된 시나리오에 따라 과제를 수행하고 질문에 답하는 테스트이다.

**42. 프로토타입(Prototype)**

요구사항을 좀 더 잘 이해하고 결정하기 위해서

전체적인 기능을 간략한 형태로 구현한 시제품이다.

**43. SWOT 분석**

기업의 내부 환경과 위부 환경을 분석하여

strength(강점), weakness(약점), opportunity(기회), threat(위협) 요인을 규정하고 이를 토대로 경영전략을 수립한다.

**44. 스토리보드(Story Board)**

UI 화면 설계를 위해서 대부분의 정보가 수록된 문서로서 디자이너와 개발자가 최종적으로 참고하는 산출물이다.

**45. 추상 클래스(Abstract Class)**

객체 인스턴스를 생성하지 않고 유사 클래스들의 공통 특징을 정의한다.

**46. 논리 데이터 모델링 속성**

개체 / 속성 / 관계

**47. 데이터 제어어(DCL)**

데이터베이스 관리자가 데이터 보안, 무결성 유지, 병행 제어, 회복을 위해 관리자가 사용하는 제어용 언어이다.

**48. 데이터베이스**

다수의 인원, 시스템 또는 프로그램이 사용할 목적으로 통합하여 관리되는 데이터 집합이다.

**49. 데이터베이스 이상현상**

데이터베이스 내의 데이터들이 불필요하게 중복되어 릴레이션 조작 시 문제가 발생하는 현상이다.

**50. 데이터베이스 이상현상**

삽입이상 / 갱신이상 / 삭제이상

**51. 데이터베이스 이상 현상 – 삭제 이상**

정보 삭제 시 원치 않는 다른 정보가 같이 삭제되는 경우

**52. ERD**

업무 분석 결과로 도출된 개체와 개체간의 관계를 도식화한 다이어그램이다.

**53. 데이터베이스 함수종속성**

하나의 릴레이션을 구성하는 속성들의 부분집합을 X와Y라고 할때 어느 시점에서든 릴레이션 내의 모든 튜플을 대상으로

한 X값에 대한 Y값이 항상 하나이면 X가 Y를 함수적으로 결정한다.

**54. 데이터베이스 정규화**

데이터 중복성을 제거하여 무결성을 보존한다.

**55. 데이터베이스 비정규화(반정규화)**

정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로 성능개선을 위해서 의도적으로 정규화 원칙을 위배하는 행위이다.

**56. 정규화 - BCNF**

모든 결정자가 후보키가 되도록 하여

결정자 함수 종속성을 제거하는 단계이다.

**57. 데이터베이스 뷰**

사용자에게 허용된 자료만을 보여주기 위해서

하나 이상의 기본 테이블로 부터 유도된 이름을 가지는 가상의 테이블이다.

**58. 데이터베이스 CRUD Matrix 개념과 점검 기준**

프로세스와 엔티티 간의 상관관계를 이용하여 구축한 엔티티 관계도를 검증하는 도구이다.

**점검기준 :**

- 모든 엔티티에 CRUD가 한번 이상 표기되어야 한다.

- 모든 엔티티에 C가 한번 이상 존재해야 한다.

- 모든 엔티티에 R가 한번 이상 존재해야 한다.

- 모든 단위 프로세스가 하나 이상의 엔티티에

표기되어야 한다.

**59. 데이터베이스 슈퍼/서브 타입**

여러 엔티티에서 공통 속성이 존재하는 경우

속성들을 하나로 묶어 통합하는 데이터 모델링 기법이다.

**60. 물리 데이터 모델링**

논리 모델을 적용하고자 하는 기술에 맞도록 상세화해 가는 과정이다.

**61. 데이터 마이닝**

대용량의 데이터 간 관계, 패턴, 추세를 발견하고

이를 의미 있는 정보로 변환하여 기업의

의사결정에 활용하는 기술

**62. 텍스트 마이닝(Text Mining)**

대량의 텍스트 데이터로부터 패턴 또는 관계를 추출하여 의미 있는 정보를 찾아내는 기법이다.

**63. 데이터베이스 인덱스**

검색 연산의 최적화를 위해서 데이터베이스 내에 튜플에 대한 정보를 구성하는 데이터 구조이다.

**64. 인덱스 컬럼 선정 기준**

- 분포도가 좋은 컬럼은 단독적을 생성한다.

- 자주 조합되어 사용되는 컬럼은 결합인덱스로

생성한다.

**65. 데이터베이스 파티셔닝**

대용량의 테이블이나 인덱스를 유지보수 하기 쉬운 단위로 분리하는 방법이다.

**66. 파티셔닝 수행 장점**

성능향상 / 가용성 향상 / 백업 가능 / 경합 감소

**67. 레인지 파티셔닝(Range Partitioning)**

연속적인 숫자나 날짜를 기준으로 하는 파티셔닝 기법이다.

**68. DBMS**

데이터 관리의 복잡성을 해결하고 데이터 추가, 삭제, 갱신 등의 기능을 지원하는 소프트웨어이다.

**69. NoSQL**

데이터 저장에 고정된 테이블 스키마가 필요하지 않고 조인 연산을 사용할 수 없으며

수평적으로 확장이 가능한 DBMS이다.

**70. NoSQL의 데이터 저장 구조**

- Key/Value Store

- Ordered Key/Value Store

- Document Key/Value Store

**71. JDBC(Java Database Connectivity)**

- Java에서 데이터베이스를 사용할 수 있도록 연결해주는 응용프로그램 인터페이스이다.

- JAVA언어를 이용하여 개발하며 SQL을 사용하여 DBMS에 질의하고 데이터를 조작하는 API를 제공하는 기술이다.

**72. ORM 도구 - Mybatis**

JAVA의 관계형 DB 프로그래밍을 좀 더 쉽게 할 수 있도록 도와주는 개발 프레임워크이다.

- 복잡한 JDBC 코드 단순화

- SQL을 거의 그대로 사용 가능

**73. WITH GRANT OPTION**

WITH GRANT OPTION을 주어 권한을 부여한 후 권한 회수 시 이후에 부여된 모든 권한도 연속하여 회수된다.

**74. 저장점(SAVEPOINT)의 개념**

현 시점에서 SAVEPOINT까지 트랜잭션의 일부만 ROLLBACK 하는 명령어이다.

**75. 데이터베이스 서브쿼리**

다른 SQL문 안에 포함되어 사용되는 또 다른 SQL 문이다.

**76. 데이터베이스 트랜잭션**

데이터베이스의 상태를 변화시키기 위해서 수행하는 최소 작업의 단위이다.

**77. ROLLBACK**

트랜잭션 실패 시 작업을 취소하고 이전 상태로 되돌릴 때 사용하는 명령어이다.

**78. 스키마**

데이터베이스의 전체적인 구조와 제약조건에 대한 명세를 기술 정의한 것이다.

**79. 프로시저**

일련의 쿼리들을 하나의 함수처럼 실행하기 위한 쿼리의 집합이다.

**80. 사용자 정의함수**

일련의 SQL 처리를 수행하고 수행 결과를 단일 값으로 반환 할 수 있는 절차형 SQL이다.

**81. 트리거**

데이터베이스 시스템에서 삽입, 삭제, 갱신 등의 이벤트가 발생할 때 마다 관련 작업이 자동으로 수행되는 절차형 SQL이다.

**82. 데이터베이스 옵티마이저**

SQL을 가장 빠르고 효율적으로 처리하기 위한

실행계획을 생성해주는 DBMS 내부의 핵심 엔진이다.

**83. 실행계획(Execution Plan)**

옵티마이저가 생성한 SQL 처리 경로

**84. 힌트**

옵티마이저가 생성한 실행계획을 사용자가 의도하는 방향으로 수동으로 제거하는 도구이다.

**85. 커밋(COMMIT)**

트랜잭션을 메모리에 영구적으로 저장하는 제어어**86. DBMS\_OUTPUT 패키지**

메시지를 버퍼에 저장하고 버퍼로부터 메시지를 불러오기 위한 인터페이스를 제공하는 패키지이다.

**87. SET SERVEROUTPUT ON**

PL/SQL의 처리결과를 화면에 출력하기 위한 SQL 명령어이다.

**88. SHOW ERRORS**

PL/SQL의 처리결과에 대한 오류 내용을 출력한다.

**89. 커넥션 풀(Connection Pool)**

데이터베이스와 연결된 커넥션을 미리 만들어서

풀에 저장해 두고 필요할 때 가져와 쓰고 다시 반환하는 기법이다.

**90. 개체-관계(E-R) 모델**

현실 세계에 존재하는 데이터와 그들 간의 관계를

사람이 이해할 수 있는 형태로 명확하게 표현하기 위한 모델이다.

**91. 온톨로지(Ontology)**

실세계에 존재하는 모든 개념과 속성, 관계 정보를

컴퓨터가 이해할 수 있도록 서술해 놓은 지식 베이스이다.

**92. 빅데이터(Big Data)**

주어진 비용 시간 내에 처리 가능한 데이터 범위를 넘어서는 크기의 비정형 데이터이다.

**93. 맵 리듀스(Map Reduce)**

구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅에서 처리하기 위한 소프트웨어 프레임워크이다.

**94. 개체(Entity) 정의서**

데이터베이스 개념 모델링 단계에서 도출한 개체의 타입과 속성 등의 정보를 명세화한 정의서이다.

**95. 연계 요구사항 분석 기법**

인터뷰 / 체크리스트 / 설문지 / 델파이 기법 /

브레인스토밍

**96. 브레인스토밍**

소속된 인원들이 자발적으로 자연스럽게 제시된 아이디어 목록을 통해서 연계요구사항을 도출하는 요구사항 분석 기법이다.

**97. 델파이 기법(Delphi Method)**

전문가의 경험적 지식을 통한 문제 해결 및 미래 예측을 위한 기법이다.

**98. 연계 메커니즘**

응용 소프트웨어 연계 대상 모듈간의 데이터 연계 시 요구사항을 고려한 연계 방법과 주기를 설계하기 위한 흐름이다.

**99. 연계 서버 장애 시 주요 점검 사항**

연계 서버의 정상 작동 여부 확인

연계 서버 엔진에 기록되는 로그 파일 내용 확인

**100. ESB 방식**

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간을 하나의 시스템으로 관리 운영할 수 있도록 서비스 중심의 통합을 지향하는 아키텍처 또는 기술이다.

**101. EAI 방식**

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보전달, 통합, 연계를 가능하게 해주는 솔루션이다.

**102. EAI 유형**

메시지 버스(Message Bus)

하이브리드(Hybrid)

포인트 투 포인트(point-to-point)

허브 앤 스포크(Hub & Spoke)

**103. 하이브리드(Hybrid) 방식**

그룹 내 허브 앤 스포크 방식을 사용하고

그룹 간에는 메시지 버스 방식을 사용한다.

**104. 포인트 투 포인트(point-to-point)**

가장 기초적인 애플리케이션 통합 방법으로 1:1 단순 통합이다.

**105. 메시지 버스(Message Bus)**

애플리케이션 사이 미들웨어(버스)를 두어 연계하는 방식으로 뛰어난 확장성과 대용량의 데이터 처리가 가능하다.

**106. Hub & Spoke (허브 앤 스포크)**

단일한 접점의 허브 시스템을 통하여 데이터를 전송하는 중앙 집중식 방식

-허브 장애 시 전체 장애 발생

**107. 웹 서비스 방식**

네트워크에 분산된 서비스 정보를 서비스 형태로 개방하여 표준화된 방식으로 공유하는 기술이다.

**108. API**

응용프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 규격을 정해놓은 인터페이스이다.

**109. DB링크**

수신시스템에서 DB링크를 생성하고,

송신시스템에서 해당 DB링크를 직접 참조하는 방식이다.

**110. SOAP(Simple Object Access Protocol)**

HTTP, HTTPS, SMTP등을 사용하여 XML 기반의 메시지를 네트워크 상태에서 교환하는 프로토콜이다.

**111. WSDL(Web Service Description Language)**

웹 서비스에 대한 상세 정보를 기술한 파일이다.

**112. UDDI**

WSDL을 등록하고 검색하기 위한 저장소로

공개적으로 접근, 검색이 가능한 레지스트리이다.

**113. 인터페이스 설계서**

이기종 시스템 및 컴포넌트 간 데이터 교환 및 처리를 위해 각 시스템의 교환되는 데이터, 업무, 송수신 주체 등이 정의된 문서

**114. 인터페이스 구현 방법**

- 데이터 통신을 사용하여 인터페이스를 구현하는 방법

- 인터페이스 개체를 사용하여 인터페이스를 구현하는 방법

**115. 인터페이스 데이터 표준 확인**

상호 연계하고자 하는 시스템 간 인터페이스가 되어야 할 범위의 데이터 형식과 표준을 정의하는 것이다.

**116. 내부 · 외부 모듈의 기능 확인 방법**

시스템 인터페이스 정의서를 통한 기능 확인

상세 기능 인터페이스 정의서를 통한 기능 확인

정적 동적 모형을 통한 기능 확인

**117. 인터페이스 오류 처리 방법**

- 사용자 화면에서 오류를 인지하도록 구현

- 인터페이스 오류 로그 생성

- 인터페이스 관련 테이블에 오류 사항 기록

**118. 스카우터(Scouter)**

대표적인 인터페이스 감시 도구로

애플리케이션에 대한 모니터링 및 DB Agent를 통해

오픈 소스 DB 모니터링 기능, 인터페이스 감시 기능을 제공한다.

**119. XML**

HTML의 단점을 보완한 인터넷 언어로

SGML의 복잡한 단점을 개선한 특수한 목적을 갖는 마크업 언어이다.

**120. AJAX**

Javascript를 사용한 비동기 통신기술

클라이언트와 서버 간에 XML 데이터를 주고받는 기술이다.

**121.JSON**

속성-값 쌍 또는 키-값 쌍으로 이루어진 데이터 오브젝트를

전달하기 위해 사용하는 개방형 표준 포멧이다.

**122. REST**

웹과 같은 분산 하이퍼미디어 환경에서

자원의 존재/상태 정보를 표준화된 HTTP 메서드로

주고받는 웹 아키텍처이다.

**123. Selenium**

다양한 브라우저 및 개발언어를 지원하는

웹 애플리케이션 테스트 프레임워크이다.

**124. Plug-In 방식**

DBMS에 Plug-In 모듈로 동작하는

데이터베이스 암호화 방식이다.

**125. 연산자**

프로그램 실행을 위해 연산을 표현한 기호

**126. 연산자 - %**

왼쪽 값을 오른쪽 값으로 나눈 나머지를 계산

**127. 연산자 - &&**

두 개의 논리값이 모두 참이면 참을 반환하고

그렇지 않으면 거짓을 반환하는 연산자이다.

**128. 사용자 정의 자료형 - 구조체**

사용자가 기본 타입을 가지고 새롭게 정의 할 수 있는 사용자 정의 자료형이다.

**129. 사용자 정의 자료형 – 열거체**

서로 연관된 정수형 상수들의 집합

정수형 상수에 이름을 붙여서 코드를 이해하기 쉽게 한다.

**130. 사용자 정의 자료형 – 공용체**

모든 멤버 변수가 하나의 메모리 공간을 공유하는

사용자 정의 자료형이다.

**131. 데이터 타입**

변수가 기질 수 있는 속성 및 속성 값의 길이

**132. 헝가리언 표기법**

접두어에 자료형을 붙이는 표기법이다.

**133. 카멜 표기법(Camel Case)**

식별자 표기 시에 여러 단어가 이어지면

첫 단어 시작만 소문자료 표기하고

각 단어의 첫 글자는 대문자로 지정한다.

**134. C++언어 생성자**

객체 생성 시 호출되는 메소드로

멤버변수의 초기화를 위해 사용된다.

**135. 명령형 언어**

컴퓨터에 저장된 명령어들이 순차적으로 실행되는

프로그래밍 방식이다.

**136. 논리형 언어**

논리 문장을 이용하여 프로그램을 표현하고

계산을 수행하는 개념에 기반한 프로그래밍 방식이다.

**137. 함수형 언어**

수학적 수식과 같은 함수들로 프로그램을 구성하여

호출하는 방식이다.

**138. 파이썬 언어 특징**

인터프리터 방식의 프로그래밍 언어

강력한 문자열 처리 기능 제공

**139. 라이브러리**

프로그램 개발을 위해 필요한 프로그램을 모아 놓은 집합체이다.

**140. 라이브러리 - 구성요소**

도움말 / 설치 파일 / 샘플 코드

**141. 라이브러리 - 기능**

문자열 연산 / 수치 계산 / 난수 생성 /

디렉터리 생성 / 날짜 조작 / 로그 출력 /

문자열 탐색/치환

**142. 패키지**

모듈을 디렉터리 형식으로 구조화한 라이브러리이다.

**143. 루틴(Routine)**

소프트웨어에서 특정 동작을 수행하는 일련의 코드로서 기능을 가진 명령들의 모임이다.

**144. 모듈(Module)**

그 자체로 하나의 완전한 기능을 수행할 수 있는

독립된 실체이다.

**145. 집계 함수**

열 행 또는 테이블 전체 행으로 부터 하나의 결과값을 반환하는 함수이다.

- COUNT / SUM / AVG / MAX / MIN / STDDEV / VARAN

**146. 윈도우 함수**

데이터베이스를 사용한 온라인 분석 처리 용도로 사용하기 위해서 표준 SQL에 추가된 함수이다.

**147. 비율 함수**

백분율을 보여주거나 행의 순서 별 백분율 등 비율과 관련된 통계를 보여주는 함수이다.

**148. 컴퓨터 과학 - 추상화**

복잡한 자료, 모듈, 시스템 등으로부터

핵심적인 개념 또는 기능을 간추려내는 기법이다.

**149. 소프트웨어 형상 관리**

소프트웨어의 변경 사항을 체계적으로 추적하고 통제하는 관리 기법이다.

**150. 형상 관리 도구**

- 소프트웨어 프로젝트에서 작성한 코드와 리소스 등의 산출물에 대한 버전 관리를 위해 사용되는 도구이다.

- 개발자들이 작성한 코드와 리소스 등 산출물에 대한 버전 관리를 위해 사용하는 도구이다.

**151. 소프트웨어 형상관리 베이스라인**

개발 과정의 각 단계의 산출물을 검토, 평가, 조정, 처리 등 변화를 통제하는 시점의 기준이다.

**152. 소프트웨어 버전 관리 도구**

형상 관리 지침을 활용하여 제품 소프트웨어의 신규 개발, 변경, 개선과 관련된 수정 사항을 관리하는 도구이다.

**153. 구현 도구**

개발자의 코드 작성과 디버깅, 수정 등과 같은 작업을 지원하는 도구이다.

**154. 형상 통제**

형상에 대한 변경 제안을 검토, 승인하여 현재 기준선에 반영할 수 있도록 통제하는 것을 말한다.

**155. 접근 통제**

접근 객체에 대한 접근 권한 확인 및 접근 제어를 통해 자원에 대한 비인가된 사용을 방지하는 기능이다.

**156. 배치 프로그램**

사용자와의 상호작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나

정해진 규칙에 따라 일괄처리 하는 프로그램이다.

**157. 정보 보안 구성요소**

기밀성 / 무결성 / 가용성

**158. 정보보안 요소 – 기밀성**

인가되지 않은 개인 혹은 시스템 접근에 다라 정보 공개 및 노출을 차단하는 특성

**159. 정보보안 - 가용성**

권한을 가진 사용자나, 애플리케이션이 원하는 서비스를 지속적으로 사용할 수 있도록 보장하는 특성이다.

**160. 암호화 알고리즘**

평문을 암호문으로 만드는 과정이다.

**161. 일방향 암호화**

암호화는 수행하지만 복호화가 불가능한 알고리즘이다.

**162. 솔트(Salt)**

일방향 해시 함수에서 다이제스트를 생성할 때 추가되는 바이트 단위 임의의 문자열이다.

**163. 대칭 키 암호화 알고리즘**

암·복호화에 같은 암호키를 쓰는 알고리즘이다.

**164. 공개키 암호화 기법**

공개키로 암호화한 평문을 개인키로 복호화하는 비대칭키 구조를 가지고 있는 암호화 알고리즘이다.

**165. 데이터베이스 암호화 - TDE**

DB서버의 DBMS 커널이 자체적으로 암복호화 기능을 수행하는 방식이다.

**166. 고유 식별 정보**

개인을 고유하게 구별하기 위하여 부여된 식별 정보이다.

**167. 민감 정보**

정보 주체의 사생활을 현저하게 침해할 수 있는 정보이다.

**168. 소프트웨어 보안 약점**

보안 취약점의 근본 원인이 되는 소프트웨어의 결함이나 버그

- 소스코드에 존재하는 잠재적인 위험을 의미한다.

**169. 소프트웨어 보안 취약점**

해킹 등 실제 침해사고에 이용될 수 있는 외부에 노출된 보안약점이다.

- 보안 약점 중에 침해사고로 연결되는 위험을 의미한다.

**170. CWE Seven Pernicious Kingdoms에서 정의한**

**소프트웨어 보안약점**

- 입력데이터 검증 및 표현

- 보안기능

- 시간 및 상태

- 에러 처리

- 코드품질

- 캡슐화

- API 약용

**171. 소프트웨어 개발보안**

소프트웨어 개발 생명주기의 각 단계별로 수행하는 일련의 보안활동을 통하여 안전한 소프트웨어를 만들기 위한 개발 체계

**172. SW 개발 보안 - 무결성**

시스템내의 정보는 오직 인가된 사용자만 수정 할 수 있다.

**173. 소프트웨어 개발 시 설계 단계에서의 보안 요구사항**

- 중요 정보 및 개인정보에 대한 보안 요구사항

- 법규에 근거한 중요 정보 및 보호조치 기준

- 소프트웨어의 안전성 확보를 위한 기능상의 보안 요구사항

**174. 소프트웨어 개발 보안 방법론**

MS-SDL / Seven Touch Points / CLASP

**175. (OWASP) CLASP**

프로그램 설계나 코딩 오류를 찾아내어 개선하기 위해 개발팀에 취약점 목록을 제공하는 모델이다.

**176. 시큐어 코딩**

설계 및 구현 단계에서 해킹 등의 공격을 유발할 가능성이 있는 잠재적인 보안 취약점을 사전에 제거하고, 외부 공격으로부터 안전한 소프트웨어를 개발하는 기법이다

**177. 시큐어 코딩 가이드 적용 대상**

입력 데이터 검증 및 표현/ 보안 기능 / 시간 및 상태 / 에러처리 / 코드 오류 / 캡슐화 / API 오용

**178. 시큐어 코딩 – 입력데이터 검증 및 표현**

SQL 삽입 / 크로스 사이트 스크립트(XSS)

**179. 크로스 사이트 스크립트(XSS)**

웹 페이지에 악의적인 스크립트를 포함해 사용자 측에서 실행되게 유도 할 수 있는 공격기법이다.

**180. 코드 오류 취약점**

**- Null Pointer 역참조**

NULL로 설정된 변수의 주소 값을 참조했을 때 발생하는 보안약점 의도적으로 널 값을 유발해 예외 상황을 확인하고 추후 공격에 활용

**- 부적절한 자원 해제**

사용된 자원을 적절히 해제하지 않으면 자원 누수 등이 발생하고 자원이 부족하여 새로운 입력을 처리할 수 없게 되는 보안약점

**- 해제된 자원 사용**

메모리 등 해제된 자원을 참조하여 예기치 않은 오류가 발생될 수 있는 보안약점

**- 초기화 되지 않은 변수 사용**

변수를 초기화 하지 않고 사용하여 예기치 않은 오류가 발생될 수 있는 보안약점

**181. DoS 공격**

특정 서버에 수많은 접속시도를 보내어

다른 사용자가 정상적으로 서비스 이용을 하지 못하게 하거나 서버의 자원을 소진시켜 원래 용도로 사용하지 못하게 하는 공격이다.

**182. Dos 공격 - 랜드 어택(Land Attack)**

출발지 IP와 목적지 IP를 같은 주소로 만들어

수신자가 자기 자신에게 응답을 보내게 하는 공격 기법이다.

**183. DoS 공격 - PoD(Ping of Death)**

ICMP 패킷을 정상적인 크기보다 아주 크게 만들어 전송하면 버퍼 오버플로우가 발생하여 정상적인 서비스를 하지 못하도록 하는 공격 기법이다.

**184. DRDoS 공격**

공격자는 출발지 IP를 공격대상 IP로 위조하여

다수의 반사 서버로 요청 정보를 전송하고

공격 대상자는 반사 서버로부터 다량의 응답을 받아 서비스 거부가 되는 공격이다.

185. ARP 스푸핑(ARP Spoofing)

MAC 주소를 사용자의 컴퓨터가 아닌 다른 사용자의 컴퓨터의 MAC 주소인 것처럼 조작하는 공격이다.

**186. APT 공격**

특정 타깃을 목표로 하여 다양한 수단을 통해

지속적이고 지능적이 맞춤형 공격기법이다.

**187. SQL Injection**

강제로 SQL문을 삽입하여 내부 데이터베이스 서버의 데이터를 유출 및 변조하고 관리자 인증을 우회하는 공격 기법이다.

**188. SQL 삽입**

웹 애플리케이션에서 입력 데이터에 대한 유효성 검증을 하지 않을 경우, 공격자가 입력 창 및 ULR에 SQL문을 삽입하여 DB로부터 정보를 열람, 조작 할 수 있는 공격기법이다.

**189. 스니핑**

- 네트워크상에서 자신이 아닌 다른 상대방들의 패킷 교환을 도청하는 행위이다.

- 공격 대상에게 직접 공격하지 않고 데이터만 몰래 들여다보는 수동적 공격 기법이다.

**190. 메모리 버퍼 오버플로우**

할당된 메모리 범위를 넘어선 위치에 자료를 읽거나 쓰려고 할 때 프로그램 오동작을 유발시키거나 악의적인 코드를 실행시켜 공격자 프로그램을 통제할 수 있는 권한을 획득한다.

**191. SYN 플러딩**

서버의 동시 가용 사용자 수를 SYN 패킷만 보내 점유하여 다른 사용자가 서버를 사용 불가능하게 하는 공격이다.

**192. RUDY (Slow HTTP POST DoS) 공격기법**

요청헤더의 content-length를 비정상적으로 크게 설정하여 메시지 바디 부분을 매우 소량으로 보내

계속 연결 상태를 유지시키는 공격 기법이다.

**193. BIA**

재난 및 재해로 인해 운영상의 손실을 가정하여

시간 흐름에 따른 영향도 및 손실평가를 조사하는

비즈니스 영향 분석이다.

**194. 워터링홀(Watering Hole)**

특정인이 잘 방문하는 웹 사이트에

악성코드를 심거나 악성코드를 배포하는 URL로

자동으로 유인하여 감염시키는 공격 기법이다.

**195. 에러(Error)/오류**

결함의 원인이 되는 것

**196. 애플리케이션 테스트**

알려지지 않은 에러를 발견하기 위한 활동이다.

**197. 애플리케이션 디버깅**

이미 알고 있는 에러의 수정을 위해서 원인을 파악하는 활동이다.

**198. 소프트웨어 테스트(Test)**

사용자 요구사항 만족여부 확인 및 노출되지 않은 소프트웨어 결함을 찾아내는 활동이다.

**199. 소프트웨어 테스트 원리**

결함이 존재 / 완벽한 테스팅은 불가능 / 초기에 테스팅 시작 / 결함집중 / 살충제 패러독스 / 정황에 의존 / 오류-부재의 궤변

**200. 테스팅은 정황에 의존한다.**

테스트는 정황과 비즈니스 도메인에 따라 다르게 수행해야 한다.

**201. 살충제 패러독스**

동일한 테스트 케이스로 동일한 테스트를 반복하면 더 이상 결함이 발견되지 않는 현상

**202. 오류-부재의 궤변**

요구사항을 만족시키지 못하면, 결함이 없다고 해도 품질이 높다고 할 수 없다.

**203. 경과시간**

애플리케이션에 사용자가 요구를 입력한 시점부터 틀랜잭션 처리 후 그 결과의 출력이 완료할 때까지 걸리는 시간

**204. 결함 에이징 분석**

등록된 결함에 대해 특정한 결함 상태의 지속 시간을 측정하여 분석하는 것이다.

**205. 하향식 통합 방법(Top-down)**

메인 제어 모듈로부터 아래 방향으로 제어의 경로를 따라 하향식으로 통합하면서 테스트를 진행하는 방법이다

.

**206. 상향식 통합 방법(Bottom-up)**

최하위 레벨의 모듈 또는 컴포넌트로부터 위쪽 방향으로 제어의 경로를 따라 이동하면서

구축과 테스트를 수행하는 방법이다.

**207. 테스트 드라이버**

상향식 통합 테스트 수행 시 허용되는 더미 모듈이다.

**208. 동적 테스트**

프로그램의 실행을 요구하는 테스트 기법

**209. 정적 테스트**

프로그램 실행 없이 소스 코드의 구조를 분석하여 논리적으로 검증하는 테스트 기법이다.

**210. 정적 분석 도구**

작성된 소스코드를 실행시키지 않고 코드 자체만으로 분석하는 도구이다.

**211. 정적 분석 도구 유형**

- 구조 검사 도구

- 데이터 분석 도구

- 순서 검사 도구

**212. 블랙박스 테스트(Black-box Test)**

프로그램 외부 사용자의 요구사항 명세를 보면서 수행하는 테스트이다.

**213. 블랙박스 테스트 기법 유형**

- 동등분할 기법

- 경계값 분석 기법

- 원인 효과 그래프

- 의사결정 테이블 기법

**214. 경곗값 분석 테스트**

경곗값을 포함하여 테스트 케이스를 설계하여 테스트 하는 기법이다.

**215. 페어와이즈 테스트**

테스트 데이터 값들 간에 최소한 한 번씩을 조합하는 방식이다.

**216. 도메인 테스트**

입력 변수들 간에 상관관계가 존재하는 경우 그 관계에 따라 영역을 분할하고 테스트케이스를 도출하는 방법이다.

**217. 화이트박스(Whitebox) 테스트**

응용프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트 방식이다.

**218. 화이트박스 테스트 기법 유형**

- 데이터 흐름 검사

- 루프 검사

- 제어 구문 검사

- 기본 경로 검사

**219. 상태 전이 테스트**

테스트 대상이 되는 시스템이나 객체의 상태를 구분하고 이벤트에 의해 다른 상태로 전이되는

경우의 수를 수행하는 테스트 기법이다.

**300. 회귀 테스트**

오류를 제거하거나 수정한 시스템에서

오류 제거와 수정에 의해 새로이 유입된 오류가 없는지 확인하는 반복 테스트이다.

**301. 구조 기반 테스트**

각 응용프로그램의 내부 구조와 동작을 검사하는

소프트웨어 테스트이다.

**302. 강도 테스트**

시스템에 과다 정보량을 부과하여 과부하 시에도 시스템이 정상적으로 작동되는지를 검증하는 테스트이다.

**303. 시스템 테스트**

통합된 단위 시스템의 기능이 시스템에서 정상적으로 수행되는지를 기능/비기능 요구사항을 검증하는 테스트이다.

**304. 통합 테스트**

단위 테스트가 끝난 모듈 또는 컴포넌트 단위의 프로그램이 설계 단계에서 제시한 애플리케이션과 동일한 구조와 기능으로 구현된 것인지 전체적으로 확인하는 테스트이다.

**305. 코드 리팩토링**

소프트웨어 외부 동작을 바꾸지 않으면서 내부 구조를 개선하는 방법이다.

**306. 리팩토링의 목적**

복잡한 코드의 단순화, 소스의 가독성을 통해

유지보수성 향상, 생산성 향상, 품질 향상이 있다.

- 가독성을 높이고 유지보수의 편리성을 높일 수 있다.

- 소프트웨어를 쉽게 이해할 수 있고 버그를 빨리 찾을 수 있다.

- 프로그램을 빨리 작성할 수 있도록 도와준다.

- 유지보수 및 소프트웨어 생산성을 향상 시킬 수 있다.

**307. 테스트 커버리지**

소프트웨어의 소스코드가 얼마만큼 테스트가 되었는가를 나타내는 지표이다.

**308. 기본 경로 커버리지 테스트**

- 맥케이브(MacCabe) 순환 복잡도

제어 흐름의 복잡한 정보를 정량적으로 표시하는 기법이다.

**309. 분기 커버리지(Branch Coverage)**

전체 조건식이 적어도 한번은 참과 거짓의 결과를 수행하는 테스트 커버리지이다.

**310. 테스트 시나리오**

테스트를 수행하기 위한 여러 테스트 케이스의 집합이다.

**311. 테스트 슈트**

시스템에 사용되는 테스트 케이스의 집합이다.

**312. 테스트 오라클**

테스트의 결과가 참인지 거짓인지를 판단하기 위해서 사전에 정의된 참 값을 입력하여 비교하는 기법이다.

**313. 목 오프젝트(Mock Object)**

사용자의 행위를 조건부로 사전에 입력해두면, 그 상황에 예정된 행위를 수행하는 객체이다.

**314. 결함 추이 분석**

애플리케이션의 어떤 모듈 또는 컴포넌트에서 결함이 발생할지를 추정하는 작업이다.

**315. 외계인 코드(Alien Code)**

아주 오래되거나 참고 문서 또는 개발자가 없어

유지보수 작업이 어려운 코드이다.

**316. 스파게티 코드**

작동은 정상적으로 하지만 사람이 코드를 읽으면서

코드의 작동을 파악하기 어려운 코드이다.

**317. 클린 코드(Clean Code)**

가독성이 높고, 단순하며, 의존성을 줄이고, 중복을 최소화하여 깔끔하게 잘 정리된 코드이다.

**318. 인스펙션(Inspection)**

저작자 외의 다른 전문가 또는 팀이 검사하여

문제를 식별하고 해결 방법을 찾아내는 형식적인 검토이다.

**319. 워크스루(Walk Throughs)**

검토 자료를 회의 전에 배포해서 사전 검토한 후

짧은 시간 동안 회의를 진행하는 형태이다.

**320. 관계 대수**

원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적 언어이다.

**321. 관계 해석**

원하는 정보가 무엇인지 정의하는 비절차적인 언어이다.

**322. 운영체제**

사용자가 컴퓨터의 하드웨어를 쉽게 사용할 수 있도록 인터페이스를 제공해주는 소프트웨어이다.

**323. 윈도즈 운영체제 특징**

- GUI 제공

- 자동감지 기능 제공

- 선점형 멀티태스킹 방식 제공

- OLE 제공

**324. 유닉스 운영체제**

교육 및 연구기관에서 사용되는 범용 다중 사용자 방식의 시분할 운영체제이다.

**325. 가상화**

물리적인 리소스들을 사용자에게 하나로 보이게 하거나 하나의 물리적인 리소스를 여러 개로 보이게 하는 기술이다.

**326. 클라우드 컴퓨팅**

인터넷을 통해 가상화된 컴퓨터 시스템 리소스를 제공하고 정보를 인터넷에 연결된 컴퓨터로 처리하는 기술이다.

**327.IaaS**

서버, 스토리지 같은 시스템 자원을 클라우드로 제공하는 서비스이다.

**328. 가상 메모리**

물리적인 메모리 용량보다 더 큰 용량의 프로그램을 실행할 수 있도록 보조기억장치를 메모리처럼 사용할 수 있도록 하는 가상 메모리 영역이다.

**329. 네트워크**

원하는 정보를 원하는 수신자 또는 기기에 정확하게 전송하기 위한 기반 인프라이다.

**330. 네트워크 프로토콜**

컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고받는 양식과 규칙의 체계이다.

**331. 프로토콜**

서로 다른 시스템에 있는 두 개체 간의

데이터 교환을 원활히 하기 위한 일련의 통신 규약이다.

**332. 프로토콜 기본 요소**

구문 / 의미 / 시간

**333. SSL /TLS**

전송계층과 응용계층 사이에서 클라이언트와 서버 간의 웹 데이터 암호화, 상호 인증 및

전송 시 데이터 무결성을 보장하는 보안 프로토콜이다.

**334. IPSec**

무결성과 인증을 보장하는 인증 헤더와

기밀성을 보장하는 암호화를 이용한 IP 보안 프로토콜이다.

**335. 거리 백터 알고리즘**

인접 라우터와 정보 공유 하며 목적지까지의 거리와 방향을 결정하는 라우팅 프로토콜이다.

**336. UDP 프로토콜**

비연결성, 비신뢰성, 순서화되지 않은 데이터 그램 서비스를 제공하는 전송 계층의 통신 프로토콜이다.

**337. UDP 특징**

- 비신뢰성

- 단순헤더

- 순서화되지 않은 데이터 프로그램 서비스 제공

- 실시간 응용 및 멀티캐스팅 가능

**338. TCP 특징**

- 신뢰성 보장

- 흐름제어

- 혼잡제어

- 연결 지향적 특징

**339. IPv6 특징**

- IP 주소 확장

- 단순헤더 적용

- 이동성

- 인증 및 보안 기능

- 개선된 QoS 지원

- Plug & Play 지원

- Ad-hoc 네트워크 지원

**340. 선점형 스케줄링**

하나의 프로세서가 다른 CPU를 차지하고 있을 때

우선순위가 높은 다른 프로세스가 현재 프로세스를 중단시키고 CPU를 점유하는 스케줄링이다.

**341. 비선점형 스케줄링 - SJF(Shortest Job First)**

프로세스가 도착하는 시점에 따라 가장 적은 서비스시간을 갖는 프로세스가 종료 시까지 자원을 점유한다.

**342. 데이터베이스 교착상태**

어떤 한 트랜잭션이 사용하기 위해 잠가 놓은 자원을 사용하기 위해서 모든 트랜잭션들이 실행을 못하고 서로 무한정 기다리는 상태이다.

**333. 가상사설망(VPN)**

공중망에서 사설망을 구축하여 전용망을 사용하는 효과를 주는 보안 솔루션이다.

**334. 온프레미스 방식**

외부 인터넷망이 차단된 상태에서 인트라넷 망을 사용하여 개발환경을 구축하는 방식이다.

**335. 패킷 스위칭**

컴퓨터 네트워크와 통신의 방식 중 하나로 작은

블록의 패킷으로 데이터를 전송하며 데이터를 전송하는 동안만 네트워크 자원을 사용하도록 하는 통신 방식이다.

**336. 서킷 스위칭**

네트워크 리소스를 특정 사용층이 독점하도록 하는 통신 방식이다.

**337. 모듈화**

소프트웨어의 성능을 향상 시키거나

시스템의 디버깅, 시험 통합 및 수정을 용이하도록 하는 모듈 중심의 소프트웨어 설계 기법이다.

**338. 모듈화 – 장점**

전체적인 소프트웨어 이해가 용이 /

모듈 재사용 가능으로 개발과 유지보수 용이

**339. 데이터베이스의 회복**

데이터베이스 트랜잭션을 실행하는 도중 장애가 발생하여 데이터베이스가 손상된 경우

손상되기 이전의 상태로 복구하는 작업이다.

**340. 데이터베이스 검사점(Check point) 회복 기법**

시스템 장애가 발생한 경우 검사점 이후 장애발생 전에 완료된 경우 REDO 수행 하거나

장애 발생 시점까지 완료하지 못한 경우 UNDO 수행한다.

**341. REDO**

데이터베이스 내용 자체가 손상된 경우, 가장 최근의 복제본을 적재 후 이후에 일어난 변경만을 로그를 이용하여 재실행하여 데이터베이스를 회복한다.

**342. UNDO**

데이터베이스 내용 자체는 손상되지 않았지만 변경된 내용에 대한 신뢰성을 잃은 경우, 모든 변경 내용을 취소하여 데이터베이스를 회복한다.

**343. 데이터베이스 병행제어**

여러 트랜잭션들이 성공적으로 동시에 실행될 수 있도록 지원하는 기능이다.

**344. 로킹(Locking) 기법**

하나의 트랜잭션에 사용되는 데이터를 다른 트랜잭션이 접근하지 못하게 제어하는 동시성 제어 기법이다.

**345. 제품 소프트웨어 패키징**

개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 포장하는 과정이다.

**346. 제품 소프트웨어 매뉴얼**

사용자 측면에서 패키징 이후 설치

제품 소프트웨어를 사용하는데 필요한 주요 내용을 기록한 문서이다.

**347. 제품 소프트웨어 사용자 매뉴얼**

개발이 완료된 제품 소프트웨어를 고객에게 전달하기 위한 형태로 패키징 하고, 설치 와 사용에 필요한 제반 절차 및 환경 등 전체 내용을 포함하는 매뉴얼이다.

**348. 제품 소프트웨어 설치 매뉴얼**

제품 소프트웨어 개발 단계부터 적용한 기준이나 패키징 이후 설치 및 사용자 측면의 주요 내용 등을 기록한 문서이다.

**349. 제품 소프트웨어 배포본**

개발된 컴포넌트 또는 패키지에 대해 제품화하고 배포 정보를 포함한 문서이다.

**350. 릴리즈 노트**

최종 사용자인 고객에게 개발 과정에서 정리된

제품의 릴리즈 정보를 제공하는 문서이다.

**351. 릴리즈 노트 예외 케이스**

테스트 단계에서 베타버전 출시 /

긴급 버그 수정 시

**352. 저작권**

창작물인 저작물에 대한 배타적인 독점적 권리로 타인의 침해를 받지 않을 고유한 권한이다.

**353. 클리어링 하우스**

소비자와 유통업자 사이에서 발생하는 거래에 대한

디지털 저작권 라이선싱을 중개하고

라이선스 발급을 수행하는 정산소이다.

**354. SSO**

한 번의 시스템 인증으로 재인증 절차 없이

여러 정보 시스템에 접근할 수 있는 통합 로그인이다.

**355. 증분 백업**

정해진 시간을 기준으로 그 이후에 변화된 파일만을 백업한다.