🥷

[취업폭격기 Zeromini 위클리 개념 폭격 #21]

📕 과목 : 소프트웨어공학

🔥 참고문제 : 2022년 국가직 7급

🌝 문제 수정 버전 : V 1.0

1.소프트웨어 프로젝트의 작업

- 문제: 소프트웨어 프로젝트에서 수행되는 작업 중, 설계 작업이 포함하는 주요 내용은 무엇인가요?
- 해설: 설계 작업은 소프트웨어 개발의 핵심 단계 중 하나로, 요구사항 분석을 바탕으로 어떻게 구현할지에 대한 전략과 방법을 결정하는 과정입니다. 이 단계에서는 시스템의 전체적인 구조와 각 구성 요소의 상세 내용, 그리고 구성 요소 간의 상호작용 방식을 정의합니다. 또한, 서브 시스템의 분할, 인터페이스 설계, 데이터 저장 방식, 알고리즘 선택

등 다양한 설계 활동이 포함됩니다. 이 과정에서는 효율적이고 안정적인 시스템을 구축하기 위한 다양한 기술과 방법론이 사용됩니다.

2.**프로젝트 범위 관리**

- 문제: WBS(Work Breakdown Structure)를 사용하는 프로젝트 범위 관리 프로세스는 무엇인가요?
- 해설: WBS는 프로젝트의 전체 범위를 세부 작업으로 나누는 도구로, 프로젝트의 전체적인 구조와 계층을 명확하게 표현합니다. 프로젝트 범위 관리에서는 WBS를 사용하여 주요 프로젝트 인도물을 더 작고 관리 가능한 구성 요소로 분해하며, 이를 통해 프로젝트의 전체 범위와 각 작업의 관계를 명확히 파악할 수 있습니다. WBS의 최하위 항목인 작업 패키지는 개별 작업 단위를 의미하며, 이를 통해 자원 할당, 일정 계획, 비용 추정 등의 프로젝트 관리 활동을 수행합니다.

3.**프로젝트 상태 정의**

- 문제: 프로젝트나 프로덕트가 특정 상태에 이르렀는지 나타내는 것은 무엇인가요?
- 해설: 프로젝트나 프로덕트의 특정 상태를 나타내는 것은 'Baseline'입니다. Baseline은 프로젝트 진행의 중요한 상태나 단계를 정의하며, 이를 통해 프로젝트의 진척 상황을 체계적으로 관리하고 추적할 수 있습니다. 또한, Baseline은 형상 항목에 대한 변경을 제어하는 중요한 메커니즘으로, 프로젝트의 안정성과 품질을 보장하는 데 중요한 역할을 합니다. 프로젝트의 각 단계마다 설정되는 Baseline을 통해 변경 요청과 이슈 처리, 리스크 관리 등의 활동을 체계적으로 수행할 수 있습니다.

4.**소프트웨어 품질 속성**

- 문제: 소프트웨어의 품질 속성 중 가장 중요한 것은 무엇인가요?
- 해설: 소프트웨어의 품질 속성은 여러 가지가 있으며, 이 중 가장 중요한 것은 '기능성', '신뢰성', '사용성', '효율성', '유지보수성', '이식성' 등이 있습니다. 이러한 품질 속성은 소프트웨어가 사용자의 요구사항과 기대를 만족시키기 위해 가져야 할 특성을 나타냅니다. 각 속성은 소프트웨어의 성능, 안정성, 사용자 경험 등 다양한 측면을 반영하며, 이를 통해 소프트웨어의 전반적인 품질을 평가하고 개선할 수 있습니다.

5.**소프트웨어 테스트 전략**

- 문제: 소프트웨어 테스트의 주요 전략 중 하나인 회귀 테스트의 목적은 무엇인가요?
- 해설: 회귀 테스트는 소프트웨어의 변경점이나 수정된 부분이 기존의 기능에 부정적인 영향을 주지 않는지 확인하는 테스트 전략입니다. 소프트웨어 개발 과정에서 발생하는 버그 수정, 기능 추가, 성능 개선 등의 작업 후에 이전에 테스트된 기능들이 여전히 올바르게 동작하는지 검증하기 위해 수행됩니다. 이를 통해 소프트웨어의 안정성을 유지하고, 새로운 이슈나 문제점이 발생하지 않도록 합니다.

6.**소프트웨어 유지보수**

- 문제: 소프트웨어 유지보수의 주요 유형 중 '적응 유지보수'의 특징은 무엇인가요?
- 해설: 적응 유지보수는 환경의 변화나 요구사항의 변경에 대응하기 위해 소프트웨어를 수정하는 유지보수 유형입니다. 예를 들어, 운영 체제의 업데이트, 하드웨어의 변경, 외부 시스템과의 인터페이스 수정 등의 이유로 소프트웨어를 적응시키기 위한 작업을 포함합니다. 이 유형의 유지보수는 기존의 기능을 보존하면서 새로운 환경에 맞게 소프트웨어를 조정하거나 확장하는 것을 목표로 합니다.

7.**소프트웨어 아키텍처**

- 문제: MVC (Model-View-Controller) 아키텍처 패턴에서 'Controller'의 주요 역할은 무엇인가요?
- 해설: MVC 패턴에서 'Controller'는 사용자의 입력을 처리하고, 해당 입력에 따라 'Model'과 'View'를 업데이트하는 역할을 합니다. 사용자의 요청을 받아 적절한 비즈니스 로직을 실행하거나 데이터를 조작하고, 결과를 'View'에 반영하여 사용자에게 표시합니다. 'Controller'는 'Model'과 'View' 사이의 중재자 역할을 하여, 데이터와 사용자 인터페이스 간의 상호작용을 관리합니다.

8.**소프트웨어 개발 방법론**

- 문제: 애자일(Agile) 방법론에서 강조하는 핵심 가치 중 하나인 '변화에 대한 대응'의 의미는 무엇인가요?
- 해설: 애자일 방법론은 변화하는 요구사항과 환경에 빠르게 대응할 수 있는 유연한 개발 방식을 추구합니다. '변화에 대한 대응'은 고정된 계획을 따르기보다는, 프로젝트 중에 발생하는 요구사항의 변경이나 추가에 적극적으로 대응하는 것을 의미합니다. 이를 통해 고객의 만족도를 높이고, 프로젝트의 성공 확률을 증가시키는 것을 목표로 합니다.

9.**소프트웨어 요구사항**

- 문제: 비기능적 요구사항의 주요 특징은 무엇인가요?
- 해설: 비기능적 요구사항은 시스템이 어떻게 동작해야 하는지, 즉 시스템의 성능, 보안, 사용성 등의 특성을 설명합니다. 이는 시스템의 기능 자체보다는 시스템이 어떤 품질 속성을 가져야 하는지를 나타내며, 이를 통해 시스템의 전반적인 품질과 안정성을 보장합니다.

10.**소프트웨어 디자인 패턴**

- 문제: 싱글턴(Singleton) 패턴의 주요 목적은 무엇인가요?
- 해설: 싱글턴 패턴의 주요 목적은 특정 클래스의 인스턴스가 하나만 생성되도록 보장하고, 그 인스턴스에 쉽게 접근할 수 있도록 하는 것입니다. 이를 통해 시스템 전체에서 해

당 클래스의 인스턴스를 공유하며, 중복 생성을 방지하여 리소스를 효율적으로 사용할 수 있습니다.

11.**소프트웨어 테스팅**

- 문제: 화이트박스 테스팅과 블랙박스 테스팅의 주요 차이점은 무엇인가요?
- 해설: 화이트박스 테스팅은 소프트웨어의 내부 구조와 동작 원리를 기반으로 테스팅하는 방법이며, 코드의 각 라인과 분기점 등을 검사합니다. 반면, 블랙박스 테스팅은 소프트웨어의 내부 구조나 동작 원리를 고려하지 않고, 오직 입력과 출력만을 기반으로 테스팅하는 방법입니다.

12.**소프트웨어 프로젝트 관리**

- 문제: 버니 다운 차트(Burn-down Chart)의 주요 용도는 무엇인가요?
- 해설: 버니 다운 차트는 애자일 프로젝트에서 남아있는 작업의 양을 시간에 따라 그래프로 표현하는 도구입니다. 이를 통해 프로젝트의 진행 상황을 시각적으로 파악하고, 남은 작업량과 일정을 쉽게 추정할 수 있습니다.

13.**소프트웨어 아키텍처**

- 문제: 마이크로서비스 아키텍처의 주요 장점은 무엇인가요?
- 해설: 마이크로서비스 아키텍처는 각 서비스가 독립적으로 배포되고 운영될 수 있어 확장성과 유연성이 높습니다. 또한, 서비스 간의 결합도가 낮아 변경이나 실패가 다른 서비스에 미치는 영향이 적으며, 다양한 기술 스택을 사용할 수 있습니다.

14.**소프트웨어 개발 생명주기**

- 문제: 워터폴(Waterfall) 모델의 주요 단점은 무엇인가요?
- 해설: 워터폴 모델은 각 단계가 순차적으로 진행되므로, 초기 단계에서의 요구사항 변경이 어렵고, 테스팅 단계까지 오류를 발견하기 어렵습니다. 또한, 고객의 피드백을 반영하기 어려워 프로젝트의 유연성이 떨어집니다.

15.**소프트웨어 품질 보증**

- 문제: 소프트웨어 품질 보증(QA)의 주요 목표는 무엇인가요?
- 해설: 소프트웨어 품질 보증의 주요 목표는 소프트웨어 제품의 품질을 보장하고, 제품이 요구사항과 일치하는지 확인하는 것입니다. QA 활동은 테스팅, 검토, 감사 등 다양한 방법을 통해 수행되며, 소프트웨어의 결함을 사전에 발견하고 수정하여 고객 만족도를 높이는 데 중점을 둡니다.

16.**소프트웨어 통합**

- 문제: 빅뱅(Big Bang) 통합 방식의 주요 위험점은 무엇인가요?

- 해설: 빅뱅 통합 방식은 모든 모듈이 완성된 후에 한 번에 통합하는 방식입니다. 이 방식의 주요 위험점은 통합 시점에 발생하는 오류와 문제점이 많아, 디버깅이 어렵고, 오류의 원인을 파악하기 힘들다는 것입니다. 또한, 오류 수정에 많은 시간이 소요될 수 있습니다.

17.**소프트웨어 메트릭스**

- 문제: 코드 복잡도를 측정하는 메트릭스 중 하나인 '사이클로매틱 복잡도'의 기준은 무엇인가요?
- 해설: 사이클로매틱 복잡도는 프로그램 코드 내의 선형적으로 독립된 경로의 수를 측정하는 메트릭스입니다. 이는 코드의 복잡성과 테스트의 어려움을 나타내며, 결정 구조와 루프 등의 구조를 기반으로 계산됩니다.

18.**소프트웨어 리팩토링**

- 문제: 리팩토링의 주요 목적은 무엇인가요?
- 해설: 리팩토링의 주요 목적은 코드의 구조와 디자인을 개선하여 소프트웨어의 유지보수성을 높이는 것입니다. 이를 통해 코드의 가독성을 향상시키고, 중복을 제거하며, 오류를 줄일 수 있습니다. 리팩토링은 기능의 변경 없이 코드의 내부 구조만을 개선하는 작업입니다.

19.**소프트웨어 라이선스**

- 문제: 오픈 소스 라이선스와 프로프라이터리 라이선스의 주요 차이점은 무엇인가요?
- 해설: 오픈 소스 라이선스는 소스 코드를 공개하며, 누구나 코드를 수정하거나 재배포할 수 있습니다. 반면, 프로프라이터리 라이선스는 소스 코드의 접근이 제한되며, 수정이나 재배포가 허용되지 않습니다. 오픈 소스는 공동체 기반의 개발을 지향하며, 프로프라이터리는 상업적 이익을 중심으로 합니다.

20.**소프트웨어 보안**

- 문제: SQL 인젝션 공격의 주요 원리는 무엇인가요?
- 해설: SQL 인젝션은 악의적인 SQL 코드를 웹 어플리케이션의 입력 값으로 주입하여 데이터베이스를 조작하거나 정보를 유출하는 공격 기법입니다. 이 공격은 입력 값 검증의 부재나 적절하지 않은 쿼리 구성으로 인해 발생하며, 데이터베이스의 민감한 정보 유출이나 데이터 조작 등의 위험을 초래합니다.

21.**소프트웨어 팀 관리**

- 문제: 애자일 팀에서 '스크럼 마스터'의 주요 역할은 무엇인가요?
- 해설: 스크럼 마스터는 스크럼 프레임워크를 올바르게 이해하고 실행하는 것을 지원하는 역할을 합니다. 팀의 장애물을 제거하고, 팀원 간의 커뮤니케이션을 원활하게 하며,

스크럼 회의와 같은 리듬을 유지하는데 중점을 둡니다. 또한, 팀과 조직의 스크럼 적용을 지속적으로 개선하는 데 기여합니다.

22.**소프트웨어 문서화**

- 문제: 소프트웨어 개발에서 '기술 사양서(Technical Specification)'의 주요 목적은 무엇인가요?
- 해설: 기술 사양서는 소프트웨어의 설계와 구현에 관한 세부 정보를 제공하는 문서입니다. 이 문서는 개발자들이 어떻게 코드를 작성할지에 대한 지침을 제공하며, 시스템의 구조, 인터페이스, 알고리즘 등의 기술적인 세부 사항을 명시합니다. 이를 통해 일관된 개발 방향을 유지하고, 팀 간의 커뮤니케이션을 강화합니다.

23.**소프트웨어 검증 및 검증(Validation and Verification)**

- 문제: 소프트웨어의 검증(Validation)과 검증(Verification)의 주요 차이점은 무엇인가요?
- 해설: 검증(Validation)은 개발된 소프트웨어가 사용자의 실제 요구사항과 일치하는지 확인하는 과정입니다. 즉, "올바른 제품을 만들었는가?"에 대한 질문에 답합니다. 반면, 검증(Verification)은 소프트웨어가 정확하게 개발되었는지, 즉 "제품을 올바르게 만들었는가?"에 대한 질문에 답하는 과정입니다.

24.**소프트웨어 유용성 테스트**

- 문제: 사용성 테스트의 주요 목적은 무엇인가요?
- 해설: 사용성 테스트는 소프트웨어 제품이 사용자에게 얼마나 효과적이고 효율적이며 만족스러운지를 평가하는 과정입니다. 이 테스트는 사용자의 관점에서 제품의 인터페이스와 상호작용을 평가하며, 사용자의 요구사항과 기대를 충족하는지 확인합니다.

25.**소프트웨어 리스크 관리**

- 문제: 소프트웨어 프로젝트에서 리스크 관리의 중요성은 무엇인가요?
- 해설: 리스크 관리는 프로젝트의 성공을 위해 예상치 못한 문제나 위험 요소를 미리 파악하고 대응하는 과정입니다. 이를 통해 프로젝트의 지연, 비용 초과, 품질 저하 등의 문제를 미리 예방하고, 프로젝트의 목표를 성공적으로 달성할 수 있습니다.

26.**소프트웨어 코드 리뷰**

- 문제: 코드 리뷰의 주요 장점은 무엇인가요?
- 해설: 코드 리뷰는 다른 개발자가 작성한 코드의 품질과 정확성을 평가하는 과정입니다. 이를 통해 코드의 오류나 결함을 미리 발견하고 수정할 수 있으며, 코드의 가독성과 유지보수성을 향상시킬 수 있습니다. 또한, 팀 간의 지식 공유와 협업을 강화하는 데도 기여합니다.

27.**소프트웨어 프로젝트 추정**

- 문제: 소프트웨어 프로젝트의 비용과 일정을 추정하는 데 사용되는 주요 방법은 무엇인가요?
- 해설: 소프트웨어 프로젝트의 비용과 일정을 추정하는 주요 방법에는 전문가 판단, 유사 프로젝트 비교, 알고리즘 기반 추정(예: COCOMO) 등이 있습니다. 각 방법은 프로젝트의 특성과 요구사항에 따라 적절하게 선택되어야 합니다.

28.**소프트웨어 프로젝트 모니터링**

- 문제: 프로젝트 모니터링에서 사용되는 주요 지표 중 하나인 '진척률(Earned Value)'의 의미는 무엇인가요?
- 해설: 진척률(Earned Value)은 프로젝트의 진행 상황을 나타내는 지표로, 지금까지 완료된 작업의 예상 비용을 의미합니다. 이 지표는 프로젝트의 실제 성과와 예상 성과를 비교하여 프로젝트의 진행 상황을 평가하는 데 사용됩니다.

29.**소프트웨어 프로젝트 팀 성**

- 문제: 크로스 펑셔널 팀(Cross-Functional Team)의 주요 특징은 무엇인가요?
- 해설: 크로스 펑셔널 팀은 다양한 전문 분야의 전문가들로 구성된 팀으로, 팀원 각각이 서로 다른 역량과 지식을 가지고 있습니다. 이러한 다양성은 팀의 문제 해결 능력과 창의성을 향상시키며, 프로젝트의 다양한 요구사항에 효과적으로 대응할 수 있습니다.

30.**소프트웨어 프로젝트 커뮤니케이션**

- 문제: 효과적인 프로젝트 커뮤니케이션의 중요성은 무엇인가요?
- 해설: 효과적인 프로젝트 커뮤니케이션은 프로젝트의 목표와 요구사항을 명확하게 전달하고, 팀 간의 협업을 강화하는 데 중요합니다. 좋은 커뮤니케이션은 오해와 갈등을 줄이며, 프로젝트의 효율성과 성공 확률을 높입니다.

31.**소프트웨어 프로젝트 리스크**

- 문제: 소프트웨어 프로젝트에서 리스크 회피(Risk Avoidance) 전략의 주요 특징은 무엇인가요?
- 해설: 리스크 회피는 잠재적인 위험 요소를 완전히 제거하거나 피하는 전략입니다. 이는 해당 리스크가 발생할 가능성을 최소화하기 위해 프로젝트의 특정 부분이나 기능을 변경하거나 제거하는 방식으로 진행됩니다. 이 전략은 리스크의 영향을 완전히 제거하려는 경우에 적용됩니다.

32.**소프트웨어 프로젝트 스테이크홀더 관리**

- 문제: 프로젝트 스테이크홀더의 관리에서 가장 중요한 요소는 무엇인가요?

- 해설: 프로젝트 스테이크홀더 관리에서 가장 중요한 요소는 효과적인 커뮤니케이션입니다. 스테이크홀더의 기대와 요구사항을 명확하게 이해하고, 프로젝트의 진행 상황과 변경 사항을 지속적으로 공유하여 스테이크홀더의 참여와 지원을 유지하는 것이 중요합니다.

33.**소프트웨어 프로젝트 품질 관리**

- 문제: 품질 보증(Quality Assurance)와 품질 관리(Quality Control)의 주요 차이점은 무엇인가요?
- 해설: 품질 보증은 프로세스와 방법론을 중심으로 품질을 보장하는 활동이며, 전체적인 품질 관리 체계의 구축과 유지에 중점을 둡니다. 반면, 품질 관리는 제품의 품질을 직접 검사하고 평가하는 활동으로, 제품의 결함이나 오류를 식별하고 수정하는데 중점을 둡니다.

34.**소프트웨어 프로젝트 리소스 관리**

- 문제: 리소스 레벨링(Resource Leveling)의 주요 목적은 무엇인가요?
- 해설: 리소스 레벨링은 프로젝트의 리소스 할당을 조정하여 리소스의 과부하나 부족을 방지하는 기법입니다. 이를 통해 리소스의 효율적인 사용을 보장하며, 프로젝트 일정과 비용을 최적화하는데 중점을 둡니다.

35.**소프트웨어 프로젝트 변경 관리**

- 문제: 변경 요청(Change Request) 처리 과정에서 가장 중요한 단계는 무엇인가요?
- 해설: 변경 요청 처리 과정에서 가장 중요한 단계는 변경의 영향 분석(Impact Analysis)입니다. 이 단계에서는 변경 요청이 프로젝트의 일정, 비용, 범위 등에 미치는 영향을 평가하며, 변경의 적절성과 필요성을 판단하는 데 중점을 둡니다.

36.**소프트웨어 프로젝트 폐쇄**

- 문제: 프로젝트 폐쇄 단계에서 수행되는 주요 활동은 무엇인가요?
- 해설: 프로젝트 폐쇄 단계에서는 프로젝트의 모든 활동을 종료하고, 프로젝트 결과물을 고객에게 전달합니다. 또한, 프로젝트의 성과와 경험을 정리하고 문서화하며, 팀원들의 성과를 평가하고 피드백을 제공하는 활동을 수행합니다.

37.**소프트웨어 프로젝트 위험 관리**

- 문제: 위험 대응 전략 중 '위험 감소(Risk Mitigation)'의 주요 특징은 무엇인가요?
- 해설: 위험 감소는 잠재적인 위험의 영향을 줄이기 위해 특정 조치나 활동을 수행하는 전략입니다. 이 전략은 위험의 발생 가능성을 낮추거나, 위험의 영향을 최소화하기 위해 적절한 대책을 마련하는 데 중점을 둡니다.

38.**소프트웨어 프로젝트 팀 구성**

- 문제: 효과적인 팀 구성을 위한 주요 원칙은 무엇인가요?
- 해설: 효과적인 팀 구성을 위한 주요 원칙은 다양성, 명확한 역할 및 책임, 올바른 스킬 세트, 팀원 간의 적절한 커뮤니케이션 및 협업 능력, 그리고 지속적인 피드백과 교육입니다. 이 원칙들은 팀의 성과와 효율성을 향상시키는 데 기여합니다.

39.**소프트웨어 프로젝트 커뮤니케이션**

- 문제: 효과적인 커뮤니케이션을 위한 주요 방법론은 무엇인가요?
- 해설: 효과적인 커뮤니케이션을 위한 주요 방법론에는 명확한 메시지 전달, 적절한 피드백 제공, 활발한 청취, 공감 능력, 그리고 지속적인 정보 공유 및 업데이트가 포함됩니다. 이러한 방법론은 팀 간의 협업과 프로젝트의 성공을 지원합니다.

40.**소프트웨어 프로젝트 리더십**

- 문제: 효과적인 프로젝트 리더의 주요 특성은 무엇인가요?
- 해설: 효과적인 프로젝트 리더의 주요 특성에는 명확한 비전과 목표 설정 능력, 의사 결정 능력, 팀원들의 동기 부여와 지원, 효과적인 커뮤니케이션 능력, 그리고 유연성 및 문제 해결 능력이 포함됩니다. 이러한 특성은 프로젝트의 성공적인 진행과 팀의 성과를 향상시키는 데 중요합니다.