



1과목-소프트웨어 설계

(1장.요구사항 확인)

학습 방향 : 정보처리기사의 다른 과목에 비해 생소한 내용이 많고 범위가 넓어 어려운 과목이지만 출제기준이 변경된 이후 약 25%로 가장 많은 문제가 출제된 과목이기 때문에 신경 써 학습해야 한다. 다행히 출제된 문제들이 대부분 개념을 묻는 것들이어서 필요한 점수를 얻는데 그렇게 어렵지는 않다. 출제 비중이 높은 1장과 3장에 집중해서 학습한다. 특히 1장에서는 소프트웨어 생명 주기와 UML을, 3장에서는 객체 지향의 구성 요소와 모듈 부분을 집중하도록 하자.

시험에 관한 안내

1. 정보처리기사 시험은 국가직무능력표준(NCS)을 기반으로 하여 문제가 출제된다.

; 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업 부문별, 수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력을 국가적 차원에서 표준화한 것을 의미하며, NCS의 능력 단위를 교육 및 훈련할 수 있도록 구성한 '교수·학습자료'를 NCS 학습 모듈이라고 한다. 정보처리기사 시험은 NCS 학습 모듈 중 정보통신 분야의 '정보기술' 분류에 포함된 정보기술개발과 정보기술운영에 속한 125개의 학습 모듈을 기반으로 하고 있다.

2. 정보처리기사, 산업기사 자격증 취득 시 독학사 취득을 위한 학점이 인정된다.

- ① 정보처리 기사 : 20학점
- ② 정보처리 산업기사 : 16학점
- ③ 사무자동화 산업기사 : 16학점
- ④ 컴퓨터 활용능력 1급 : 14학점
- ⑤ 컴퓨터 활용능력 2급 : 6학점
- ⑥ 워드 프로세서 : 4학점

시험에 관한 안내

3. 정보처리기사/산업기사 필기 시험은 어디서 접수는 인터넷으로만 접수할 수 있다. q-net.or.kr에 접속하여 신청하면 된다.
4. 필기 시험에 합격한 후 실기 시험에 여러 번 응시할 수 있으며, 필기시험에 합격한 후 실기 시험 응시 횟수에 관계 없이 필기 시험 합격자 발표일로부터 2년 동안 실기 시험에 응시할 수 있다.
5. 정보처리기사/산업기사는 정기 시험만 있다. 1년에 3번이며, 상시 시험은 없다.
6. 정보처리기사 필기 시험에 합격하려면 평균 60점 이상, 과목당 40점 이상 되어야 한다. 즉, 평균 60점 이상이지만 어느 한 과목이라도 40점 미만이면 불합격이다.
7. 정보처리 기사 응시자격

구분	응시 자격	제출 서류
학력 응시	4년제: 졸업 또는 4학년1학기 이상 재학/휴학/제적	졸업/재학/휴학/수료/제적증명서 중 택1
	3년제: 졸업 후 1년 실무 경력	졸업증명서+경력(재직)증명서
	2년제: 졸업 후 2년 실무 경력	졸업증명서+경력(재직)증명서
경력 응시	동일 직무 분야에서 4년 이상 실무경력	경력(재직)증명서
	기능사 취득 후 3년 실무경력	경력(재직)증명서+자격증 사본
	산업기사 취득 후 1년 실무경력	경력(재직)증명서+자격증 사본
기타 응시	학점인정법률의거 106점 이상 취득	한국교육개발원에서 발급한 학점인정증명서

시험에 관한 안내

8. 시험과목

- 필기

1. 소프트웨어 설계
2. 소프트웨어 개발
3. 데이터베이스 구축
4. 프로그래밍 언어 활용
5. 정보 시스템 구축 관리

- 실기

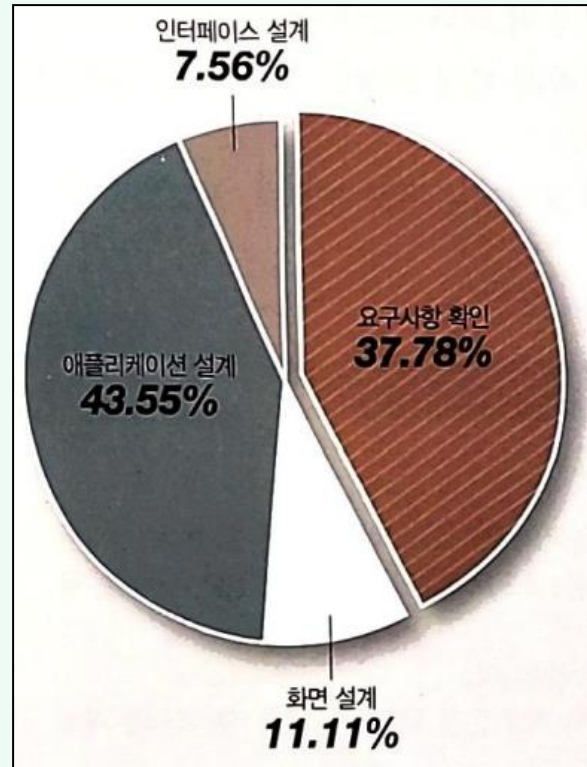
정보처리 실무

9. 검정 방법

- 필기 : 객관식 4지 택일형, 과목당 20문항(과목당 30분)
- 실기 : 필답형(2시간30분)

정보처리 기사(1과목. 소프트웨어 설계 구분)

- 1장. 요구사항 확인
- 2장. 화면 설계
- 3장. 애플리케이션 설계
- 4장. 인터페이스 설계



요구사항 확인 파트

요구사항 확인 파트는 총 10개의 작은 장으로 구성된다.

1. 소프트웨어 생명주기
2. 스크럼(Scrum) 기법
3. XP(eXtreme Programming) 기법
4. 현행 시스템 파악
5. 개발 기술 환경 파악
6. 요구사항 정의
7. 요구사항 분석
8. 요구사항 분석 CASE와 HIPO
9. UML(Unified Modeling Language)
10. 주요 UML 다이어그램

1. 요구사항 확인-소프트웨어 생명주기

- 이 장을 공부하면서 반드시 알아두어야 할 키워드

; 폭포수 모형, 애자일 선언, XP(eXtreme Programming), 요구사항분석, 자료흐름도, 자료사전, 관계, 다이어그램의 종류, 유스케이스 다이어그램, 순차 다이어그램

1) 소프트웨어 생명 주기

; 소프트웨어 생명 주기는 소프트웨어 개발 방법론의 바탕이 되는 것으로, 소프트웨어를 개발하기 위해 정의하고 운용, 유지보수 등의 과정을 각 단계별로 나눈 것이다.

- 소프트웨어 생명 주기는 소프트웨어 개발 단계와 각 단계별 주요 활동, 그리고 활동의 결과에 대한 산출물로 표현한다. 소프트웨어 수명 주기라고도 한다.
- 소프트웨어 생명 주기를 표현하는 형태를 소프트웨어 생명 주기 모형이라고 하며, 소프트웨어 프로세스 모형 또는 소프트웨어 공학 패러다임이라고도 한다.
- 개발자는 문제의 유형이나 개발 방법 등에 따라 특정 모형을 선택하여 사용할 수 도 있고, 개별적인 모형을 사용할 수도 있다.
- 일반적으로 사용되는 소프트웨어 생명 주기 모형에는 폭포수 모형, 프로토타입 모형, 나선형 모형, 애자일 모형 등이 있다.

1. 요구사항 확인-소프트웨어 생명주기

1) 소프트웨어 생명 주기

; 소프트웨어 공학의 개념

소프트웨어 공학(SE: Software Engineering)은 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문이며 여러 가지 방법론과 도구, 관리 기법들을 통하여 소프트웨어의 품질과 생산성을 향상시킬 목적으로 한다. 소프트웨어 공학은 다음과 같이 여러 형태로 정의할 수 있다.

- IEEE의 소프트웨어 공학 표준 용어사전: 소프트웨어의 개발, 운용, 유지보수, 폐기 처분에 대한 체계적인 접근 방안
- Fairley: 지정된 비용과 기간 내에 소프트웨어를 체계적으로 생산하고 유지, 보수하는 데 관련된 기술적이고 관리적인 원리
- Boehm: 과학적인 지식을 소프트웨어 설계와 제작에 응용하는 것이며 이를 개발, 운용, 유지, 보수하는 데 필요한 문서 작성 과정

; 소프트웨어 공학의 기본 원칙

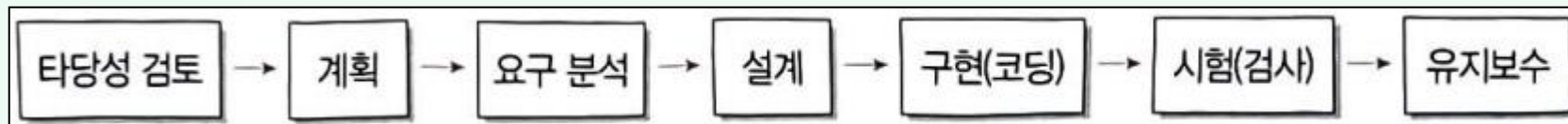
- 현대적인 프로그래밍 기술을 계속적으로 적용해야 한다.
- 개발된 소프트웨어의 품질이 유지되도록 지속적으로 검증해야 한다.
- 소프트웨어 개발 관련 사항 및 결과에 대한 명확한 기록을 유지해야 한다.

1. 요구사항 확인-소프트웨어 생명주기

2) 폭포수 모형(Waterfall Model)

; 폭포수 모형은 폭포에서 한번 떨어진 물은 거슬러 올라갈 수 없듯이 소프트웨어 개발도 이전 단계로 돌아갈 수 없다는 전제하에 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 개발 방법론이다.

- 폭포수 모형은 소프트웨어 공학에서 가장 오래되고 가장 폭넓게 사용된 전통적인 소프트웨어 생명 주기 모형으로, 고전적 생명 주기 모형이라고도 한다.
- 소프트웨어 개발 과정의 한 단계가 끝나야만 다음 단계로 넘어갈 수 있는 선형 순차적 모형이다.
- 모형을 적용한 경험과 성공 사례가 많다.
- 제품의 일부가 될 매뉴얼을 작성해야 한다. 매뉴얼은 프로그램들의 사용과 운영에 대한 내용이 기술되어 있는 문서이다.
- 각 단계가 끝난 후에는 다음 단계를 수행하기 위한 결과물이 명확하게 산출되어야 한다.
- 두 개 이상의 과정이 병행하여 수행되지 않는다.

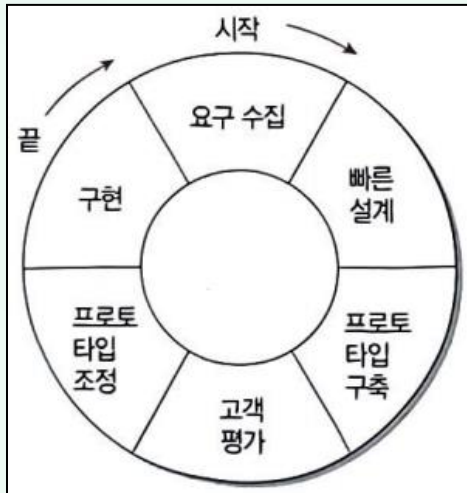


1. 요구사항 확인-소프트웨어 생명주기

3) 프로토타입 모형(Prototype Model, 원형 모형)

; 프로토타입 모형은 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본(시제)품(Prototype)을 만들어 최종 결과물을 예측하는 모형이다.

- 시제품은 사용자와 시스템 사이의 인터페이스에 중점을 두어 개발한다.
- 시스템의 일부 혹은 시스템의 모형을 만드는 과정으로서 요구된 소프트웨어를 구현하는데, 이는 추후 구현 단계에서 사용될 골격 코드가 된다.
- 소프트웨어의 개발이 완료된 시점에서 오류가 발견되는 폭포수 모형의 단점을 보완하기 위한 모형이다.



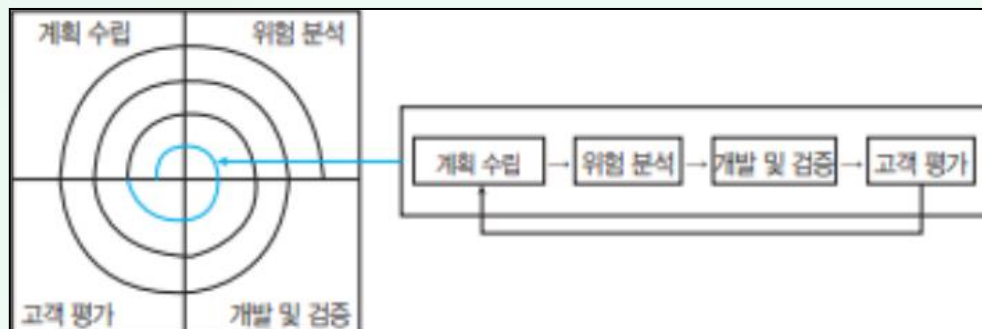
1. 요구사항 확인-소프트웨어 생명주기

4) 나선형 모형(Spiral Model, 점진적 모형)

; 나선형 모형은 보헴(Boehm)이 제안한 것으로, 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형이다.

- 나선을 따라 돌듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로 완벽한 최종 소프트웨어를 개발하는 것으로, 점진적 모형이라고도 한다.
- 소프트웨어를 개발하면서 발생할 수 있는 위험을 관리하고 최소화하는 것을 목적으로 한다.
- 점진적으로 개발 과정이 반복되므로 누락되거나 추가된 요구사항을 첨가할 수 있고, 정밀하며, 유지보수 과정이 필요 없다.

; 나선형 모형은 '계획 → 분석 → 개발 → 평가' 순이라는 것을 기억하자. 나선형 모형은 나선을 따라 돌 듯이 소프트웨어 개발 과정을 여러 번 반복하면서 진행된다는 것을 염두에 두고 생각하면 이해가 쉬울 것이다.



1. 요구사항 확인-소프트웨어 생명주기

5) 애자일 모형(Agile Model)

; 애자일은 '민첩한', '기민한'이라는 의미로, 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발과정을 진행한다.

- 애자일 모형은 어느 특정 개발 방법론이 아니라 좋은 것을 빠르고 낭비 없게 만들기 위해 고객과의 소통에 초점을 맞춘 방법론을 통칭한다.
- 애자일 모형은 기업 활동 전반에 걸쳐 사용된다.
- 애자일 모형은 스프린트(Sprint) 또는 이터레이션(Iteration)이라고 불리는 짧은 개발 주기를 반복 하며, 반복되는 주기마다 만들어지는 결과물에 대한 고객의 평가와 요구를 적극 수용한다.
- 각 개발주기에서는 고객의 요구사항에 우선순위를 부여하여 개발 작업을 진행한다.
- 소규모 프로젝트, 고도로 숙달된 개발자, 급변하는 요구사항에 적합하다.
- 애자일 모형을 기반으로 하는 소프트웨어 개발 모형에는 스크럼(Scrum), XP(eXtreme Programming), 칸반(Kanban), Lean, 크리스탈(Crystal), ASD(Adaptive Software Development), 기능 중심 개발(FDD: Feature Driven Development), DSDM(Dynamic System Development Method), DAD(Disciplined Agile Delivery) 등이 있다.

1. 요구사항 확인-소프트웨어 생명주기

6) 폭포수 모형과 애자일의 비교

구분	폭포수 모형	애자일
새로운 요구사항 반영	어려움	지속적으로 반영
고객과의 의사소통	적음	지속적임
테스트	마지막에 모든 기능을 테스트	반복되는 일정 주기가 끝날 때마다 테스트
개발 중심	계획, 문서(매뉴얼)	고객

1. 요구사항 확인-소프트웨어 생명주기

기출문제 확인

1. 소프트웨어 공학의 기본 원칙이라고 볼 수 없는 것은?

- ① 품질 높은 소프트웨어 상품 개발
- ② 지속적인 검증 시행
- ③ 결과에 대한 명확한 기록 유지
- ④ 최대한 많은 인력 투입

설명 : 인력은 최대한 많이 투입하는 것이 아니라 가능한 효율적으로 인원이 투입되어야 한다.

2. 폭포수 모형의 특징으로 거리가 먼 것은?

- ① 개발 중 발생한 요구사항을 쉽게 반영할 수 있다.
- ② 순차적인 접근방법을 이용한다.
- ③ 단계적 정의와 산출물이 명확하다.
- ④ 모형의 적용 경험과 성공사례가 많다.

설명 : 폭포수 모형은 한번 떨어진 물이 다시 거슬러 올라갈 수 없듯이 한 단계가 끝나면 이전 단계로 돌아갈 수 없다는 전제하에 만들어진 모형을 의미하므로 개발 중 요구사항을 쉽게 반영을 하지 못한다.

3. 소프트웨어 생명 주기 모형 중 고전적 생명 주기 모형으로, 선형 순차적 모델이라고도 하며, 타당성 검토, 계획, 요구사항 분석, 구현, 테스트, 유지보수의 단계를 통해 소프트웨어를 개발하는 모형은?

- ① 폭포수 모형 ② 애자일 모형
- ③ 컴포넌트 기반 방법론 ④ 6GT 모형

설명 : 고전적 생명주기 모형, 선형 순차적 모델이라고 하면 폭포수 모형이라는 것을 기억하도록 한다.

컴포넌트 기반 방법론(CBD)이란 컴포넌트를 조합해 재사용함으로써 개발 생산성과 품질을 높이고 시스템 유지보수 비용을 최소화할 수 있는 방법론을 의미한다.

4. 프로토타입을 지속적으로 발전시켜 최종 소프트웨어 개발까지 이르는 개발 방법으로 위험관리가 중심인 소프트웨어 생명 주기 모형은?

- ① 나선형 모형 ② 델파이 모형
- ③ 폭포수 모형 ④ 기능점수 모형

설명 : 나선을 돌 듯이 여러 번의 소프트웨어 개발 과정을 지속적으로 발전시키는 모형을 의미한다.

1. 요구사항 확인-소프트웨어 생명주기

기출문제 확인

5. 소프트웨어 개발 모델 중 나선형 모델의 4가지 주요 활동이 순서대로 나열된 것은?

- | | |
|-----------|---------|
| Ⓐ 계획 수립 | Ⓑ 고객 평가 |
| Ⓒ 개발 및 검증 | Ⓓ 위험 분석 |

- ① A-B-D-C ② A-D-C-B
③ A-B-C-D ④ A-C-B-D

설명 : 나선형 모형은 계획->분석->개발->평가 과정을 반복하는 모형이다.

6. 애자일 기법에 대한 설명으로 맞지 않은 것은?

- ① 절차와 도구보다 개인과 소통을 중요하게 생각한다.
② 계획에 중점을 두어 변경 대응이 난해하다.
③ 소프트웨어가 잘 실행되는데 가치를 둔다.
④ 고객과의 피드백을 중요하게 생각한다.

설명 : 애자일 모형은 계획에 따르기보다는 시대변화에 반응하는 것에 더 가치를 두는 개발 방법론이고, 의뢰자와의 소통을 가장 중요시 하는 모형이다.

7. 애자일 방법론에 해당하지 않는 것은?

- ① 기능 중심 개발 ② 스크럼
③ 익스트림 프로그래밍 ④ 모듈 중심 개발

설명 : 모듈 중심 개발은 애자일 방법론에는 해당하지 않는다.

8. 소프트웨어 생명주기 모델 중 나선형 모델(Spiral Model)과 관련한 설명으로 틀린 것은?

- ① 소프트웨어 개발 프로세스를 위험 관리(Risk Management) 측면에서 본 모델이다.
② 위험 분석(Risk Analysis)은 반복적인 개발 진행 후 주기의 마지막 단계에서 최종적으로 한 번 수행해야 한다.
③ 시스템을 여러 부분으로 나누어 여러 번의 개발 주기를 거치면서 시스템이 완성된다.
④ 요구사항이나 아키텍처를 이해하기 어렵다거나 중심이 되는 기술에 문제가 있는 경우 적합한 모델이다.

설명 : 나선형 모델에서 위험 분석은 개발 과정에 포함되므로 개발 진행 과정에서 반복적으로 수행된다.

2. 요구사항 확인-스크럼(Scrum) 기법

1) 스크럼의 개요

; 스크럼이란 럭비에서 반칙으로 경기가 중단된 경우 양 팀의 선수들이 럭비공을 가운데 두고 상대팀을 밀치기 위해 서로 대치해 있는 대형을 말한다. 스크럼은 이처럼 팀이 중심이 되어 개발의 효율성을 높인다는 의미가 내포된 용어이다.

- 스크럼은 팀원 스스로가 스크럼 팀을 구성(self-organizing)해야 하며, 개발 작업에 관한 모든 것을 스스로 해결(cross-functional)할 수 있어야 한다.

- 스크럼 팀은 제품 책임자, 스크럼 마스터, 개발팀으로 구성된다.

- 제품 책임자(PO: Product Owner)

- 이해 관계자들 중 개발될 제품에 대한 이해도가 높고, 요구사항을 책임지고 의사 결정할 사람으로 선정하는데, 주로 개발 의뢰자나 사용자가 담당한다.

- 이해 관계자들의 의견을 종합하여 제품에 대한 요구사항을 작성하는 주체다.

- 요구사항이 담긴 백로그(Backlog)를 작성하고 백로그에 대한 우선순위를 지정한다.

- 팀원들이 백로그에 스토리를 추가할 수는 있지만 우선순위를 지정할 수는 없다.

- 제품에 대한 테스트를 수행하면서 주기적으로 요구사항의 우선순위를 갱신한다.

스토리(Story)란 백로그에 담겨질 요구사항은 단어 형태로 표현된 것이 아니라 '고객은 상품 주문을 위해 로그인을 수행해야 한다.'와 같이 이야기를 서술하는 형태로 표현한다. 그래서 백로그에 작성되는 요구사항을 스토리라 한다.

2. 요구사항 확인-스크럼(Scrum) 기법

1) 스크럼의 개요

- 스크럼 마스터(SM: Scrum Master)

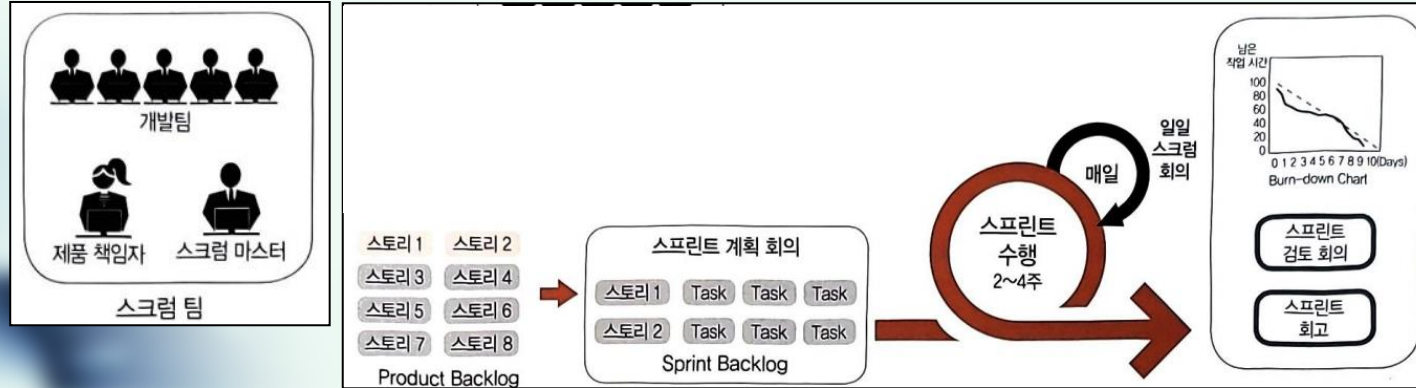
- 스크럼 팀이 스크럼을 잘 수행할 수 있도록 객관적인 시각에서 조언을 해주는 가이드 역할을 수행한다. 팀원들을 통제하는 것이 목표가 아니다.
- 일일 스크럼 회의를 주관하여 진행 사항을 점검하고, 개발 과정에서 발생한 장애 요소를 공론화하여 처리한다.

- 개발팀(DT: Development Team)

- 제품 책임자와 스크럼 마스터를 제외한 모든 팀원으로, 개발자 외에도 디자이너, 테스터 등 제품 개발을 위해 참여하는 모든 사람이 대상이 된다.
- 보통 최대 인원은 7~8명이 적당하다.

2. 요구사항 확인-스크럼(Scrum) 기법

2) 스크럼 개발 프로세스



스프린트의 의미는 팀이 일정량의 작업을 완료하는 시간이 정해진 짧은 기간을 말한다. 스프린트는 스크럼과 애자일 방법론의 핵심이며, 올바른 스프린트는 애자일 팀이 더 적은 수고를 통해 더 나은 소프트웨어를 제공하는 데 도움이 된다.

● 제품 백로그(Product Backlog)

- 제품 개발에 필요한 모든 요구사항(User Story)을 우선순위에 따라 나열한 목록이다.
- 개발 과정에서 새롭게 도출되는 요구사항으로 인해 지속적으로 업데이트 된다.
- 제품 백로그에 작성된 사용자 스토리를 기반으로 전체 일정 계획인 릴리즈 계획록이다.

● 스프린트 계획 회의(Sprint Planning Meeting)

- 제품 백로그 중 이번 스프린트에서 수행할 작업을 대상으로 단기 일정을 수립하는 것이다.
- 스프린트에서 처리할 요구사항(User Story)을 개발자들이 나눠서 작업할 수 있도록 태스크(Task)라는 작업 단위로 분할한 후 개발자 별로 수행할 작업 목록인 스프린트 백로그(Sprint Backlog)를 작성한다.

2. 요구사항 확인-스크럼(Scrum) 기법

2) 스크럼 개발 프로세스

● 스프린트(Sprint)

- 실제 개발 작업을 진행하는 과정으로, 보통 2 ~ 4주 정도의 기간 내에서 진행한다.
- 스프린트 백로그에 작성된 태스크를 대상으로 속도(Velocity)를 추정한 후 개발 담당자에게 할당한다.
- 태스크를 할당할 때는 개발자가 원하는 태스크를 직접 선별하여 담당할 수 있도록 하는 것이 좋다.
- 개발 담당자에게 할당된 태스크는 보통 할 일(To Do), 진행 중(In Progress), 완료(Done)의 상태를 갖는다.
- 여기서 속도란 한 번의 스프린트에서 한 팀이 감당할 수 있는 제품 백로그의 양에 대한 추정치이다.

● 일일 스크럼 회의(Daily Scrum Meeting)

- 모든 팀원이 매일 약속된 시간에 약 15분 정도의 짧은 시간 동안 진행 상황을 점검한다.
- 회의는 보통 서서 진행하며, 남은 작업 시간은 소멸 차트(Burn-down Chart)에 표시한다.
- 스크럼 마스터는 발견된 장애 요소를 해결할 수 있도록 도와준다.
- 소멸 차트(Burn-down Chart)란 해당 스프린트에서 수행할 작업의 진행상황을 확인할 수 있도록 시간의 경과에 따라 남은 작업 시간을 그래프로 표현한 것이다. 초기에 추정했던 전체 작업 시간은 작업이 진행될수록 점점 줄어(Burn-down) 들게 됩니다.

2. 요구사항 확인-스크럼(Scrum) 기법

2) 스크럼 개발 프로세스

- 스프린트 검토 회의(Sprint Review)

- 부분 또는 전체 완성 제품이 요구사항에 잘 부합되는지 사용자가 포함된 참석자 앞에서 테스트를 수행한다.
- 스프린트의 한 주당 한 시간 내에서 진행한다. 제품 책임자(Product Owner)는 개선할 사항에 대한 피드백을 정리한 후 다음 스프린트에 반영할 수 있도록 제품 백로그를 업데이트 한다.

- 스프린트 회고(Sprint Retrospective)

- 스프린트 주기를 되돌아보며 정해 놓은 규칙을 잘 준수했는지, 개선할 점은 없는지 등을 확인하고 기록한다.
- 해당 스프린트가 끝난 시점에서 수행하거나 일정 주기로 수행한다.

2. 요구사항 확인-XP(eXtremeProgramming) 기법

1) XP(eXtreme Programming)

; XP(eXtreme Programming)는 수시로 발생하는 고객의 요구사항에 유연하게 대응하기 위해 고객의 참여와 개발 과정의 반복을 극대화하여 개발 생산성을 향상시키는 방법이다.

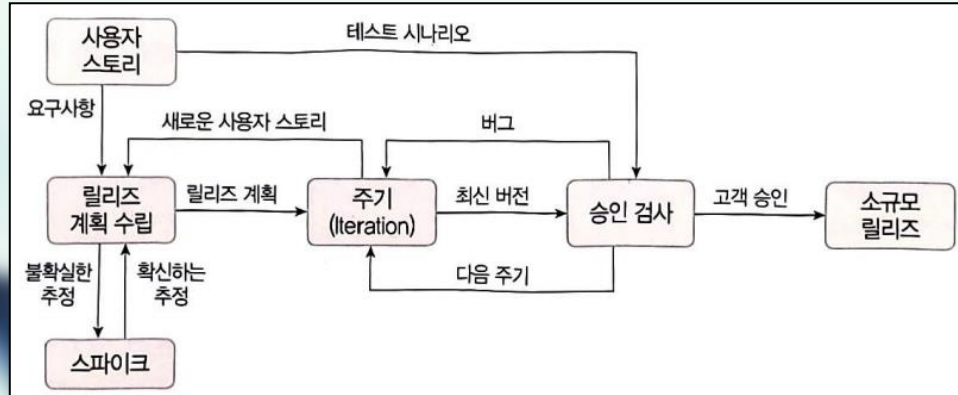
- XP는 짧고 반복적인 개발 주기, 단순한 설계, 고객의 적극적인 참여를 통해 소프트웨어를 빠르게 개발하는 것을 목적으로 한다.
- 릴리즈의 기간을 짧게 반복하면서 고객의 요구사항 반영에 대한 "가시성"을 높인다.
- 릴리즈 테스트마다 고객을 직접 참여시킴으로써 요구한 기능이 제대로 작동하는지 고객이 직접 확인할 수 있다.
- 비교적 소규모 인원의 개발 프로젝트에 효과적이다.
- XP의 5가지 핵심 가치 : 의사소통(Communication), 단순성(Simplicity), 용기(Courage), 존중(Respect), 피드백(Feedback)

릴리즈(Release) : 몇 개의 요구사항이 적용되어 부분적으로 기능이 완료된 제품을 제공하는 것을 말한다.

가시성(Visibility) : 대상을 확인할 수 있는 정도를 의미한다. 릴리즈 기간을 짧게 반복하면서 개발 과정에서 제품 소프트웨어의 일부 기능이 구현될 때마다 고객에게 이를 확인시켜주면, 고객은 요구사항이 잘 반영되고 있음을 직접적으로 알 수 있다는 의미이다.

2. 요구사항 확인-XP(eXtremeProgramming) 기법

2) XP 개발 프로세스



● 사용자 스토리(User Story)

- 고객의 요구사항을 간단한 시나리오로 표현한 것이다.
- 내용은 기능 단위로 구성하며, 필요한 경우 간단한 테스트 사항(Test Case)도 기재한다.

● 릴리즈 계획 수립(Release Planning)

- 몇 개의 스토리가 적용되어 부분적으로 기능이 완료된 제품을 제공하는 것을 릴리즈라고 한다.
- 부분 혹은 전체 개발 완료 시점에 대한 일정을 수립한다.

● 스파이크(Spike)

- 요구사항의 신뢰성을 높이고 기술 문제에 대한 위험을 감소시키기 위해 별도로 만드는 간단한 프로그램이다. 처리할 문제 외의 다른 조건은 모두 무시하고 작성한다.

2. 요구사항 확인-XP(eXtremeProgramming) 기법

2) XP 개발 프로세스

● 주기(Iteration)

- 하나의 릴리즈를 더 세분화 한 단위를 이터레이션(Iteration)이라고 한다.
- 일반적으로 1 ~ 3주 정도의 기간으로 진행된다.
- 이 기간 중에 새로운 스토리가 작성될 수 있으며, 작성된 스토리는 진행 중인 이터레이션 혹은 다음 이터레이션에 포함될 수 있다.

● 승인 검사(Acceptance Test, 인수 테스트)

- 하나의 이터레이션 안에서 계획된 릴리즈 단위의 부분 완료 제품이 구현되면 수행하는 테스트이다.
- 사용자 스토리 작성 시 함께 기재한 테스트 사항에 대해 고객이 직접 수행한다.
- 테스트 과정에서 발견한 오류 사항은 다음 이터레이션에 포함한다.
- 테스트 이후 새로운 요구사항이 작성되거나 요구사항의 상대적 우선순위가 변경될 수 있다.
- 테스트가 완료되면 다음 이터레이션을 진행한다.

● 소규모 릴리즈(Small Release)

- 릴리즈를 소규모로 하게 되면, 고객의 반응을 기능별로 확인할 수 있어, 고객의 요구사항에 좀 더 유연하게 대응할 수 있다. 계획된 릴리즈 기간 동안 진행된 이터레이션이 모두 완료되면 고객에 의한 최종 테스트를 수행한 후 릴리즈, 즉 최종 결과물을 고객에게 전달한다.

2. 요구사항 확인-XP(eXtremeProgramming) 기법

2) XP 개발 프로세스

● XP의 주요 실천 방법

- Pair Programming(짝 프로그래밍) : 다른 사람과 함께 프로그래밍을 수행함으로써 개발에 대한 책임을 공동으로 나눠 갖는 환경을 조성한다.
- Collective Ownership(공동 코드소유) : 개발 코드에 대한 권한과 책임을 공동으로 소유한다.
- Test-Driven Development(테스트 주도 개발)
개발자가 실제 코드를 작성하기 전에 테스트 케이스를 먼저 작성하므로 자신이 무엇을 해야 할지를 정확히 파악한다.
테스트가 지속적으로 진행될 수 있도록 자동화된 테스트 도구(구조, 프레임워크)를 사용한다.
- Whole Team(전체 팀) : 개발에 참여하는 모든 구성원(고객 포함)들은 각자 자신의 역할이 있고 그 역할에 대한 책임을 가져야 합니다.
- Continuous Integration(계속적인 통합) : 모듈 단위로 나눠서 개발된 코드들은 하나의 작업이 마무리될 때마다 지속적으로 통합된다.
- Design Improvement(디자인 개선) 또는 Refactoring(리팩토링) : 프로그램 기능의 변경 없이, 단순화, 유연성 강화 등을 통해 시스템을 재구성한다.

2. 요구사항 확인-XP(eXtremeProgramming) 기법

2) XP 개발 프로세스

- XP의 주요 실천 방법

- Small Releases(소규모 릴리즈) : 릴리즈 기간을 짧게 반복함으로써 고객의 요구 변화에 신속히 대응할 수 있다.

2. 요구사항 확인-스크럼 기법, XP 기출문제

기출문제 확인(스크럼)

1. 다음이 설명하는 프로세스 모델은 무엇인가?

- 팀원들이 스스로 팀을 구성하며, 개발 작업의 모든 것을 스스로 해결할 수 있어야 한다.
- 개발에 필요한 요구사항에 우선순위를 부여한 제품기능 목록(Product Backlog)을 작성한다.
- 개발 주기를 의미하는 스프린트는 2 ~ 4주 정도의 기간으로 진행한다.
- 스프린트 회고(Retrospective)를 통해 스프린트 동안 발생한 문제점을 파악하고 이에 대한 해결 방안을 모색한다.

① 익스트림 프로그래밍(XP)

② 크리스털(Crystal)

③ 칸반(Kanban)

④ 스크럼(Scrum)

설명 : 스프린트, 제품기능 목록, 스프린트 회고만 봐도 어떤 프로세스 모델인지 알 수 있다.

2. 애자일(Agile)기법 중 스크럼(Scrum)과 관련된 용어에 대한 설명이 틀린 것은?

① 스크럼 마스터(Scrum Master)는 스크럼 프로세스를 따르고, 팀이 스크럼을 효과적으로 활용할 수 있도록 보장하는 역할 등을 맡는다.

② 제품 백로그(Product Backlog)는 스크럼 팀이 해결해야 하는 목록으로 소프트웨어 요구사항, 아키텍처 정의 등이 포함될 수 있다.

③ 스프린트(Sprint)는 하나의 완성된 최종 결과물을 만들기 위한 주기로 3달 이상의 장기간으로 결정된다.

④ 속도(Velocity)는 한 번의 스프린트에서 한 팀이 어느 정도의 제품 백로그를 감당할 수 있는지에 대한 추정치로 볼 수 있다.

설명 : 스프린트는 보통 2~4주 정도의 기간으로 결정해 작업을 한다.

3. 다음의 스크럼(Scrum) 개발 과정을 진행 순서에 맞게 올바르게 나열한 것은?

- ㄱ. 스프린트(Sprint)
- ㄴ. 스프린트 회고(Sprint Retrospective)
- ㄷ. 일일 스크럼 회의(Daily Scrum Meeting)
- ㄹ. 스프린트 검토 회의(Sprint Review)
- ㅁ. 스프린트 계획 회의(Sprint Planning Meeting)

① ㄱ → ㄷ → ㄱ → ㄴ → ㄹ

② ㄱ → ㄱ → ㄷ → ㄹ → ㄴ

③ ㄱ → ㄷ → ㄱ → ㄹ → ㄴ

④ ㄱ → ㄹ → ㄱ → ㄴ → ㄷ

설명 : 계획된 내용을 토대로 일정 기간 스프린트 수행하면서 매일 회의하고 하나의 스프린트가 끝이 나면 검토하고, 회고한다.

2. 요구사항 확인-스크럼 기법, XP 기출문제

기출문제 확인(XP)

1. 익스트림 프로그래밍(eXtreme Programming)의 5가지 가치에 속하지 않는 것은?

- ① 의사소통 ② 단순성
- ③ 피드백 ④ 고객 배제

설명 : 의사소통, 단순성, 용기, 존중, 피드백으로 구성됨
고객의 적극적인 참여를 통한 방법이 XP기법이다.

2. XP(eXtreme Programming)의 기본 원리(실천 방법)로 볼 수 없는 것은?

- ① Linear Sequential Method
- ② Pair Programming
- ③ Collective Ownership
- ④ Continuous Integration

설명 : 짝 프로그래밍, 공동 코드 소유, 테스트 주도 개발,
전체 팀, 지속적인 통합, 디자인 개선 및 리팩토링,
스몰 릴리즈

3. 익스트림 프로그래밍(XP)에 대한 설명으로 틀린 것은?

- ① 빠른 개발을 위해 테스트를 수행하지 않는다.
- ② 사용자의 요구사항은 언제든지 변할 수 있다.
- ③ 고객과 직접 대면하며 요구사항을 이야기하기 위해 사용자 스토리 (User Story)를 활용할 수 있다.
- ④ 기존의 방법론에 비해 실용성(Pragmatism)을 강조한 것이라고 볼 수 있다.

4. 익스트림 프로그래밍에 대한 설명으로 틀린 것은?

- ① 대표적인 구조적 방법론 중 하나이다.
- ② 소규모 개발 조직이 불확실하고 변경이 많은 요구를 접하였을 때 적절한 방법이다.
- ③ 익스트림 프로그래밍을 구동시키는 원리는 상식적인 원리와 경험을 최대한 끌어 올리는 것이다.
- ④ 구체적인 실천 방법을 정의하고 있으며, 개발 문서 보다는 소스 코드에 중점을 둔다.

설명 : 스크럼, XP, 칸반 등등은 애자일 개발 방법론을 기반으로 한 대표적인 모형이다.

2. 요구사항 확인-스크럼 기법, XP 기출문제

기출문제 확인

5. 소프트웨어를 보다 쉽게 이해할 수 있고 적은 비용으로 수정할 수 있도록 겉으로 보이는 동작의 변화 없이 내부 구조를 변경하는 것은?

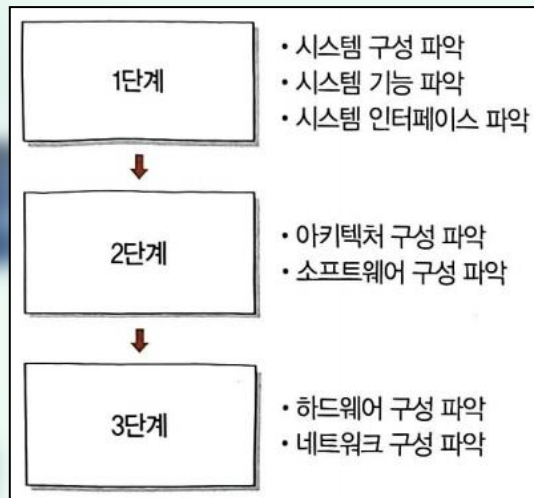
- ① Refactoring ② Architecting
- ③ Specification ④ Renewal

설명 : 핵심은 동작(기능)의 변화 없이 내부 구조를 재구성한다(Refactoring)는 것이다.

2. 요구사항 확인-현행 시스템 파악

1) 현행 시스템 파악 절차

; 새로 개발하려는 시스템의 개발 범위를 명확히 설정하기 위해 현행 시스템의 구성과 제공 기능, 시스템 간의 전달 정보, 사용되는 기술 요소, 소프트웨어, 하드웨어, 그리고 네트워크의 구성 등을 파악한다.



2. 요구사항 확인-현행 시스템 파악

2) 시스템 구성 파악

; 현행 시스템의 구성은 조직의 주요 업무를 담당하는 기간 업무와 이를 지원하는 지원 업무로 구분하여 기술한다.

- 조직 내에 있는 모든 정보 시스템의 현황을 파악할 수 있도록 각 업무에 속하는 단위 업무 정보 시스템들의 명칭, 주요기능들을 명시한다.

; 예) 금융기관의 여신관리 업무와 고객관리 업무 시스템 현황

구분	시스템명	시스템 내용
여신관리 업무	여신기획 관리 시스템	여신기획 관리를 위한 여신요율 책정, 연간 여신운용지침 수립 등의 기능을 제공하는 시스템
	여신상담 관리 시스템	여신상담 관리를 위한 거래처정보 관리, 여신상담, 대출의향서 발급 기능을 제공하는 시스템
고객관리 업무	고객등록 처리 시스템	고객의 기본 정보를 관리하기 위한 등록, 변경, 조회 삭제 등의 기능을 제공하는 시스템

2. 요구사항 확인-현행 시스템 파악

3) 시스템 기능 파악

; 현행 시스템의 기능은 단위 업무 시스템이 현재 제공하는 기능들을 주요 기능과 하부 기능, 세부 기능으로 구분하여 계층형으로 표시한다.

; 예) 여신상담 관리 시스템의 주요 기능과 하부 세부 기능

단위 업무 시스템	Level 1 주요 업무 기능	Level 2 세부 업무 기능	Level 3 세부 업무 기능 활동
여신상담 관리	여신기획 관리	여신요율 책정	
		연간 여신운용지침 수립	
	여신상담 관리	거래처정보 관리	거래처정보 등록
			신용정보 관리
		여신상담	대상거래 파악
			상담결과 보고
			신용조사 의뢰

2. 요구사항 확인-현행 시스템 파악

4) 시스템 인터페이스 파악

; 현행 시스템의 인터페이스에는 단위 업무 시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 연계 유형, 주기 등을 명시한다.

- 데이터를 어떤 형식으로 주고 받는지, 통신규약은 무엇을 사용하는지, 연계 유형은 무엇인지 등을 반드시 고려해야 한다.

; 예) 여신상담 관리 시스템의 인터페이스 현황

송신 시스템	수신 시스템	연동 데이터	연동 형식	통신규약	연계 유형	주기
여신상담 관리 시스템	여신관리센터	연체 정보	XML	TCP/IP	EAI	하루(일)
여신상담 관리 시스템	여신금융협회	부도 정보	XML	X.25	FEP	수시

XML : Extensible Markup Language(XML)를 사용하면 공유 가능한 방식으로 데이터를 정의하고 저장할 수 있다. XML은 웹 사이트, 데이터베이스 및 타사 애플리케이션과 같은 컴퓨터 시스템 간의 정보교환을 지원한다. 사전 정의된 규칙을 사용하면 수신자가 이러한 규칙을 사용하여 데이터를 효율적으로 정확하게 읽을 수 있으므로 모든 네트워크에서 데이터를 XML 파일로 손쉽게 전송할 수 있다.

X.25 : 1970년대 후반에 개발된 공중 데이터 네트워크(패킷교환망)에 대한 최초의 표준이며, 현재는 거의 사라졌다.

EAI : Enterprise Architecture Integration의 약자이다.기업 내 필요한 여러 어플리케이션(채널)이 있을텐데 이런 각종 어플리케이션 간에 상호 연동이 가능하도록 통합하는 솔루션이다.

FEP : Front End Processor의 약자로 원래 메인 프레임에서 통신 과부하를 경감시키기 위해 전처리 작업을 하는 과정을 말하지만, 금융권에서는 의미가 조금 와전되어 B2B 연계(대외계)를 FEP라고 부른다.

2. 요구사항 확인-현행 시스템 파악

5) 아키텍처 구성 파악

- ; 현행 시스템의 아키텍처 구성은 기간 업무 수행에 어떠한 기술 요소들이 사용되는지 최상위 수준에서 계층별로 표현한 아키텍처 구성도로 작성한다. 아키텍처가 단위 업무 시스템 별로 다른 경우에는 가장 핵심이 되는 기간 업무 처리 시스템을 기준으로 표현한다.
- ; 시스템 아키텍처(System Architecture) : 시스템 아키텍처는 시스템 내부에서 각각의 하위 시스템들이 어떠한 관계로 상호 작용하는지 파악할 수 있도록 구성이나 동작 원리를 표현한 것을 말한다.

6) 소프트웨어 구성 파악

- ; 소프트웨어 구성에는 단위 업무 시스템 별로 업무 처리를 위해 설치되어 있는 소프트웨어들의 제품명, 용도, 라이선스 적용 방식, 라이선스 수 등을 명시한다.
- 시스템 구축비용 면에서 소프트웨어 비용이 적지 않은 비중을 차지하므로, 상용 소프트웨어의 경우 라이선스 적용 방식의 기준과 보유한 라이선스의 파악이 중요하다.

구분	시스템명	SW 제품명	용도	라이선스 적용 방식	라이선스 수
여신관리 업무	여신기획 관리 시스템	Apache Tomcat	WAS	오픈 소스 Apache License	1
		MySQL	데이터베이스	GPL 또는 상용	1
		UNIX	운영체제	GNU GPL	1
	여신상담 관리 시스템	Sage	ERP	상용	1
		Oracle	데이터베이스	GPL 또는 상용	1
		Windows 10	운영체제	DSP	5

GNU 일반 공중 사용 허가서(GNU: General Public License, GNU GPL 또는 GPL)는 자유 소프트웨어 재단에서 만든 자유 소프트웨어 라이선스로, 소프트웨어의 실행, 연구, 공유, 수정의 자유를 최종 사용자에게 보장하는 것을 의미한다.

DSP: Demand Side Platform의 약자로, 광고 시장의 수요측인 광고주가 광고 지면을 구매하기 위해 사용하는 소프트웨어 플랫폼이다.

2. 요구사항 확인-현행 시스템 파악

7) 하드웨어 구성 파악

; 하드웨어 구성에는 단위 업무 시스템들이 운용되는 서버의 주요 사양과 수량, 그리고 이중화의 적용 여부를 명시한다. 서버의 이중화는 기간 업무의 서비스 기간, 장애 대응 정책에 따라 필요 여부가 결정된다.

● 현행 시스템에 이중화가 적용된 경우 대부분 새로 구성될 시스템에도 이중화가 필요하므로 이로 인한 비용 증가와 시스템 구축 난이도가 높아질 가능성을 고려해야 한다.

; 서버의 CPU 처리 속도 메모리 크기, 하드 디스크의 용량 등을 파악해서 명시한다.

; 서버의 이중화(Replication)

● 서버의 이중화란 운용 서버의 장애 시 대기 서버로 서비스를 계속 유지할 수 있도록, 운용 서버의 자료 변경이 예비 서버에도 동일하게 복제되도록 관리하는 것을 의미한다.

8) 네트워크 구성 파악

; 네트워크 구성은 업무 시스템들의 네트워크 구성을 파악할 수 있도록 서버의 위치, 서버 간의 네트워크 연결 방식을 네트워크 구성도로 작성한다. 네트워크 구성도를 통해 서버들의 물리적인 위치 관계를 파악할 수 있고 보안 취약성을 분석하여 적절한 대응을 할 수 있다. 네트워크에 장애가 발생한 경우 발생 원인을 찾아 복구하기 위한 용도로 활용될 수 있다.

2. 요구사항 확인-개발 기술 환경 파악

1) 개발 기술 환경의 정의

- ; 개발하고자 하는 소프트웨어와 관련된 운영체제(Operating System), 데이터베이스 관리 시스템(Database Management System), 미들웨어(Middle Ware) 등을 선정할 때 고려해야 할 사항을 기술하고, 오픈소스 사용 시 주의해야 할 내용을 제시한다.
- ; 미들웨어는 운영체제와 해당 운영체제에 의해 실행되는 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어이다.

2) 운영체제(OS, Operating System)

- ; 운영체제는 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어이다.
- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해준다.
- 컴퓨터 운영체제의 종류에는 Windows, UNIX, Linux, Mac OS 등이, 모바일 운영체제에는 iOS, Android 등이 있다.

2. 요구사항 확인-개발 기술 환경 파악

3) 운영체제 관련 요구사항 식별 시 고려사항

; 운영체제와 관련된 요구사항 식별 시 다음과 같은 사항을 고려해야 한다.

① 가용성 : 프로그램이 주어진 시점에서 요구사항에 따라 운영될 수 있는 능력을 의미한다.

- 시스템의 장시간 운영으로 인해 발생할 수 있는 운영체제 고유의 장애 발생 가능성
- 메모리 누수로 인한 성능 저하 및 재가동
- 보안상 발견된 허점을 보완하기 위한 지속적인 패치 설치로 인한 재가동
- 운영체제의 결함 등으로 인한 패치 설치를 위한 재가동

② 성능

- 대규모 동시 사용자 요청에 대한 처리
- 대규모 및 대용량 파일 작업에 대한 처리
- 지원 가능한 메모리 크기(32bit, 64bit)

③ 기술 지원

- 제작 업체의 안정적인 기술 지원
- 여러 사용자들 간의 정보 공유
- 오픈 소스 여부(Linux)

2. 요구사항 확인-개발 기술 환경 파악

3) 운영체제 관련 요구사항 식별 시 고려사항

④ 주변기기

- 설치 가능한 하드웨어
- 여러 주변기기 지원 여부

⑤ 구축 비용

- 지원 가능한 하드웨어 비용
- 설치할 응용 프로그램의 라이선스 정책 및 비용
- 유지관리 비용
- 총 소유 비용(TCO)

; TCO : (TCO; Total Cost of Ownership): 어떤 자산을 획득하려고 할 때 지정된 기간 동안 발생할 수 있는 모든 직간접 비용들로, 하드웨어 구매, 소프트웨어 구매 및 라이선스, 설치, 교육, 지속적인 기술 지원, 유지보수, 가동 중지로 인한 손실, 에너지 등의 비용이 있다.

; 메모리 누수 : 응용 프로그램이 더 이상 사용하지 않는 메모리를 반환하지 않고 계속 점유하고 있는 현상

; 오픈 소스 : 누구나 제한 없이 사용할 수 있도록 소스 코드를 공개해 무료로 사용이 가능한 소프트웨어

2. 요구사항 확인-개발 기술 환경 파악

4) 데이터베이스 관리 시스템(DBMS)

; DBMS(DataBase Management System)는 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어이다.

- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해 준다.
- DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
- DBMS의 종류에는 Oracle, IBM DB2, Microsoft SQL Server, MySQL, SQLite, MongoDB, Redis 등 있다.

5) DBMS 관련 요구사항 식별 시 고려사항

; DBMS와 관련된 요구사항 식별 시 다음과 같은 사항을 고려해야 한다.

① 가용성

- 시스템의 장시간 운영으로 인해 발생할 수 있는 운영체제 고유의 장애 발생 가능성
- DBMS의 결함 등으로 인한 패치 설치를 위한 재가동
- 백업이나 복구의 편의성
- DBMS 이중화 및 복제 지원

2. 요구사항 확인-개발 기술 환경 파악

5) DBMS 관련 요구사항 식별 시 고려사항

② 성능

- 대규모 데이터 처리 성능(분할 테이블 지원 여부)
- 대용량 트랜잭션 처리 성능
- 튜닝 옵션의 다양한 지원
- 최소화된 설정과 비용 기반 질의 최적화 지원

③ 기술 지원

- 제작 업체의 안정적인 기술 지원
- 여러 사용자들 간의 정보 공유
- 오픈소스 여부

④ 상호 호환성

- 설치 가능한 운영체제의 종류
- JDBC, ODBC와의 호환 여부

⑤ 구축 비용

- 라이선스 정책 및 비용, 유지관리 비용, 총 소유비용(TCO)

2. 요구사항 확인-개발 기술 환경 파악

6) 웹 애플리케이션 서버(WAS: Web Application Server)

; 웹 애플리케이션 서버는 정적인 콘텐츠 처리를 하는 웹 서버와 달리 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어이다.

- 데이터 접근, 세션 관리, 트랜잭션 관리 등을 위한 라이브러리를 제공한다.
- 주로 데이터베이스 서버와 연동해서 사용한다.
- 웹 애플리케이션 서버의 종류에는 Tomcat, GlassFish, JBoss, Jetty, JEUS, Resin, WebLogic, WebSphere 등이 있다.

7) 웹 애플리케이션 서버(WAS) 관련 요구사항 식별 시 고려사항

구분	내용
가용성	<ul style="list-style-type: none">• 시스템의 장시간 운영으로 인해 발생할 수 있는 고유의 장애 발생 가능성• WAS의 결함 등으로 인한 패치 설치를 위한 재가동• 안정적인 트랜잭션 처리• WAS 이중화 지원
성능	<ul style="list-style-type: none">• 대규모 트랜잭션 처리 성능• 다양한 설정 옵션 지원• 가비지 컬렉션(GC; Garbage Collection)*의 다양한 옵션
기술 지원	<ul style="list-style-type: none">• 제조업체의 안정적인 기술 지원• 여러 사용자들 간의 정보 공유• 오픈 소스 여부
구축 비용	<ul style="list-style-type: none">• 라이선스 정책 및 비용• 유지관리 비용• 총 소유 비용(TCO)

2. 요구사항 확인-개발 기술 환경 파악

8) 오픈 소스 사용에 따른 고려사항

; 오픈 소스(Open Source)는 누구나 별다른 제한 없이 사용할 수 있도록 소스 코드를 공개한 것으로 오픈 소스 라이선스를 만족하는 소프트웨어이다.

- 오픈 소스를 사용하는 경우에는 라이선스의 종류, 사용자 수, 기술의 지속 가능성 등을 고려해야 한다.

2. 요구사항 확인-현행 시스템 파악, 개발 기술 환경 파악 기출문제

기출문제 확인(현행 시스템 파악)

1. 다음 중 현행 시스템 파악과정에 대한 설명으로 잘못된 것은?

- ① 시스템 구성은 조직의 주요 업무를 담당하는 기간 업무와 이를 지원하는 지원 업무로 구분하여 기술한다.
- ② 소프트웨어 구성을 파악할 때 상용 소프트웨어의 경우 라이선스 적용 방식의 기준과 보유한 라이선스의 파악이 중요하다.
- ③ 아키텍처 구성을 파악할 때는 단위 업무 시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 연계 유형, 주기 등을 명시한다.
- ④ 네트워크 구성을 파악하면 서버들의 물리적인 위치 관계를 파악할 수 있고 보안 취약성을 분석하여 이에 대한 적절한 대응을 할 수 있다.

설명 : 아키텍처 구성은 기간 업무 수행에 어떠한 기술 요소가 사용되었는지 최상위 수준에서 계층별로 표현한 것이다.

단위 업무 시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 연계 유형, 주기 등을 명시하는 것은 시스템 인터페이스 파악할 때, 기술할 내용이다.

2. 현행 시스템 분석에서 고려하지 않아도 되는 항목은?

- ① DBMS 분석
- ② 네트워크분석
- ③ 운영체제 분석
- ④ 인적 자원 분석

설명 : 현행 시스템 파악은 말 그대로 현재 사용하고 있는 정보 시스템에 대한 현황을 파악하는 것을 의미한다.하여 인적 자원 분석은 이에 해당하지 아니한다.

2. 요구사항 확인-현행 시스템 파악, 개발 기술 환경 파악 기출문제

기출문제 확인(개발 기술 환경 파악)

1. DBMS 분석 시 고려사항으로 거리가 먼 것은?

- ① 가용성 ② 성능
- ③ 네트워크 구성도 ④ 상호 호환성

설명 : DBMS 분석 시 고려사항으로는 가용성, 성능, 상호 호환성, 기술 지원, 구축 비용이 있다.

2. WAS(Web Application Server)가 아닌 것은?

- ① JEUS ② JVM
- ③ Tomcat ④ WebSphere

설명 : JVM은 자바 가상머신으로 Java 실행을 위한 프로그램이다.
WAS의 종류가 많아서 외우기 어렵기 때문에 위에 제시된 것들 정도만
알아두고 차차 기출문제를 풀어가면서 좀 더 확인해보도록 하자.

3. 운영체제에 대한 설명으로 잘못된 것은?

- ① 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공한다.
- ② 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 일종의 하드웨어 장치이다.
- ③ 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공한다.
- ④ 종류에는 Windows, UNIX, Linux, iOS 등이 있다.

4. 다음 중 데이터베이스 관리 시스템(DBMS)이 아닌 것은?

- ① Oracle ② MySQL
- ③ Microsoft SQL Server ④ Android

설명 : Android는 모바일 환경의 운영체제이다.

DBMS의 종류는 Oracle, MySQL, DB2, SQL Server, MS-SQL, TIBERO, MongoDB, Redis, SAP 등이 있다.

5. 요구사항 식별 시 고려사항 중 가용성과 관련된 내용이 아닌 것은?

- ① 시스템의 장시간 운영으로 인해 발생할 수 있는 고유의 장애 발생 가능성
- ② DBMS의 결함 등으로 인한 패치 설치를 위한 재가동
- ③ WAS 이중화 지원
- ④ 설치할 응용 프로그램의 라이선스 정책 및 비용

설명 : 가용성이란 프로그램이 주어진 시점에서 요구사항에 따라 운영될 수 있는 능력을 의미한다.

라이선스 정책 및 비용, 유지관리 비용, 총 소유비용(TCO)은 구축 비용에서 고려될 사항이다.

2. 요구사항 확인-요구사항 정의

1) 요구사항의 개념 및 특징

; 요구사항은 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영 되는데 필요한 제약조건 등을 나타낸다.

- 요구사항은 소프트웨어 개발이나 유지 보수 과정에서 필요한 기준과 근거를 제공한다.
- 요구사항은 개발하려는 소프트웨어의 전반적인 내용을 확인할 수 있게 하므로 개발에 참여하는 이해 관계자들 간의 의사소통을 원활하게 하는 데 도움을 준다.
- 요구사항이 제대로 정의되어야만 이를 토대로 이후 과정의 목표와 계획을 수립할 수 있다.

2. 요구사항 확인-요구사항 정의

2) 요구사항의 유형

; 요구사항은 일반적으로 기술하는 내용에 따라 기능 요구사항(Functional requirements)과 비 기능 요구사항(Non-functional requirements)으로 구분하며, 기술 관점과 대상의 범위에 따라 시스템 요구사항(System requirements)과 사용자 요구사항(User requirements)으로 나눈다.

유형	내용
기능 요구사항 (Functional requirements)	<ul style="list-style-type: none"> • 시스템이 무엇을 하는지, 어떤 기능을 하는지에 대한 사항 • 시스템의 입력이나 출력으로 무엇이 포함되어야 하는지, 시스템이 어떤 데이터를 저장하거나 연산을 수행해야 하는지에 대한 사항 • 시스템이 반드시 수행해야 하는 기능 • 사용자가 시스템을 통해 제공받기를 원하는 기능
비기능 요구사항 (Non-functional requirements)	<ul style="list-style-type: none"> • 시스템 장비 구성 요구사항 : 하드웨어, 소프트웨어, 네트워크 등의 시스템 장비 구성에 대한 요구사항 • 성능 요구사항 : 처리 속도 및 시간, 처리량, 동적·정적 적응량, 가용성 등 성능에 대한 요구사항 • 인터페이스 요구사항 : 시스템 인터페이스와 사용자 인터페이스에 대한 요구사항으로 다른 소프트웨어, 하드웨어 및 통신 인터페이스, 다른 시스템과의 정보 교환에 사용되는 프로토콜과의 연계도 포함하여 기술 • 데이터 요구사항 : 초기 자료 구축 및 데이터 변환을 위한 대상, 방법, 보안이 필요한 데이터 등 데이터를 구축하기 위해 필요한 요구사항 • 테스트 요구사항 : 도입되는 장비의 성능 테스트(BMT)나 구축된 시스템이 제대로 운영되는지를 테스트하고 점검하기 위한 테스트 요구사항 • 보안 요구사항 : 시스템의 데이터 및 기능, 운영 접근을 통제하기 위한 요구사항 • 품질 요구사항 : 관리가 필요한 품질 항목, 품질 평가 대상에 대한 요구사항으로 가용성[※], 정확성[※], 상호 호환성[※], 대응성[※], 신뢰성, 사용성, 유지·관리성, 이식성[※], 확장성[※], 보안성 등으로 구분하여 기술

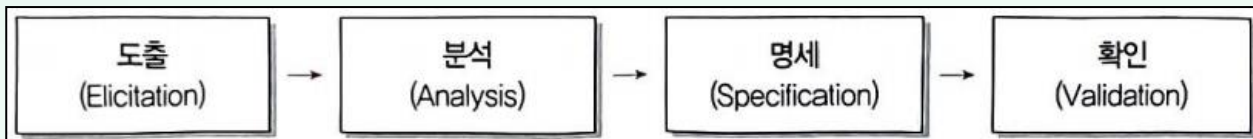
비기능 요구사항 (Non-functional requirements)	<ul style="list-style-type: none"> • 제약사항 : 시스템 설계, 구축, 운영과 관련하여 사전에 파악된 기술, 표준, 업무, 법·제도 등의 제약조건 • 프로젝트 관리 요구사항 : 프로젝트의 원활한 수행을 위한 관리 방법에 대한 요구사항 • 프로젝트 지원 요구사항 : 프로젝트의 원활한 수행을 위한 지원 사항이나 방안에 대한 요구사항
사용자 요구사항 (User requirements)	<ul style="list-style-type: none"> • 사용자 관점에서 본 시스템이 제공해야 할 요구사항 • 사용자를 위한 것으로 친숙한 표현으로 이해하기 쉽게 작성된다.
시스템 요구사항 (System requirements)	<ul style="list-style-type: none"> • 개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항 • 사용자 요구사항에 비해 전문적이고 기술적인 용어로 표현된다. • 소프트웨어 요구사항이라고도 한다.

2. 요구사항 확인-요구사항 정의

3) 요구사항 개발 프로세스

; 요구사항 개발 프로세스는 개발 대상에 대한 요구사항을 체계적으로 도출하고 이를 분석한 후 분석 결과를 명세서(Specification Document)에 정리한 다음 마지막으로 이를 확인 및 검증하는 일련의 구조화된 활동이다.

- 요구사항 개발 프로세스가 진행되기 전에 개발 프로세스가 비즈니스 목적에 부합되는지, 예산은 적정한지 등에 대한 정보를 수집, 평가한 보고서를 토대로 타당성 조사(Feasibility Study)가 선행되어야 한다.
- 요구사항 개발은 요구공학(Requirement Engineering)의 한 요소이다.

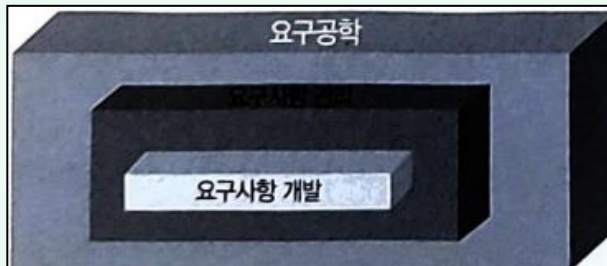


2. 요구사항 확인-요구사항 정의

4) 요구 공학(Requirements Engineering)

; 요구 공학은 무엇을 개발해야 하는지 요구사항을 정의하고, 분석 및 관리하는 프로세스를 연구하는 학문이다.

- 점점 복잡하고 대형화 되어 가는 소프트웨어 개발 환경에 따라 사용자 요구사항도 더욱 복잡해지고 잦은 변경이 발생하는 데, 이는 요구사항에 문제가 발생할 가능성을 높이며 요구사항 관리가 잘못될 수 있는 원인이 된다.
- 요구 공학은 요구사항 변경의 원인과 처리 방법을 이해하고 요구사항 관리 프로세스의 품질을 개선하여 소프트웨어 프로젝트 실패를 최소화하는 것을 목표로 한다.



2. 요구사항 확인-요구사항 정의

5) 요구사항 도출(Requirement Elicitation, 요구사항 수집)

; 요구사항 도출은 시스템, 사용자, 그리고 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항이 어디에 있는지, 어떻게 수집할 것인지를 식별하고 이해하는 과정이다.

- 요구사항 도출은 소프트웨어가 해결해야 할 문제를 이해하는 첫 번째 단계이다.
- 요구사항 도출 단계에서 개발자와 고객 사이의 관계가 만들어지고 이해관계자(Stakeholder)가 식별된다.
- 이 단계에서는 다양한 이해관계자 간의 효율적인 의사소통이 중요하다.
- 요구사항 도출은 소프트웨어 개발 생명 주기(SDLC: Software Development Life Cycle) 동안 지속적으로 반복된다.
- 요구사항을 도출하는 주요 기법에는 청취와 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스 등이 있다.

브레인스토밍(Brain Storming) : 3인 이상이 자유롭게 의견을 교환하면서 독창적인 아이디어를 산출해내는 방법이다.

프로토타이핑(Prototyping) : 프로토타이핑은 프로토타입(견본품)을 통해 효과적으로 요구 분석을 수행하면서 명세서를 산출하는 작업으로 가장 단순한 형태는 설명을 위해 종이에 대략적인 순서나 형태를 그려 보여주는 것이다.

유스케이스(Use Case) : 유스케이스는 사용자의 요구사항을 기능 단위로 표현하는 것이다.

2. 요구사항 확인-요구사항 정의

6) 요구사항 분석(Requirement Analysis)

; 요구사항 분석은 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정이다.

- 사용자 요구사항의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다.
- 내용이 중복되거나 하나로 통합되어야 하는 등 서로 상충되는 요구사항이 있으면 이를 중재하는 과정이다.
- 도출된 요구사항들을 토대로 소프트웨어의 범위를 파악한다.
- 도출된 요구사항들을 토대로 소프트웨어와 주변 환경이 상호 작용하는 방법을 이해한다.
- 요구사항 분석에는 자료 흐름도(DFD: Data Flow Diagram), 자료 사전(DD) 등의 도구가 사용된다.

자료 흐름도(Data Flow Diagram) : 자료의 흐름과 처리 과정을 도형 중심으로 기술자료 흐름 그래프나 버블 차트라고 칭한다. 단계적으로 세분화 자료를 처리할 때마다 새로운 이름을 부여한다.

자료 사전(Data Dictionary) : 자료 흐름도의 자료를 설명하는 것을 말한다. 자료를 설명하는 자료를 메타 데이터라고 부른다.

2. 요구사항 확인-요구사항 정의

7) 요구사항 명세(Requirement Specification)

; 요구사항 명세는 분석된 요구사항을 바탕으로 모델을 작성하고 문서화하는 것을 의미한다.

- 요구사항을 문서화할 때는 기능 요구사항은 빠짐 없이 완전하고 명확하게 기술해야 하며, 비기능 요구사항은 필요한 것만 명확하게 기술해야 한다.
- 요구사항은 사용자가 이해하기 쉬우며, 개발자가 효과적으로 설계할 수 있도록 작성되어야 한다.
- 설계 과정에서 잘못된 부분이 확인될 경우 그 내용을 요구사항 정의서에서 추적할 수 있어야 한다.
- 구체적인 명세를 위해 소단위 명세서(Mini-Spec)가 사용될 수 있다.

; 소프트웨어 요구사항 명세서(SRS: Software Requirement Specification)

- 업계 표준 용어로 소프트웨어가 반드시 제공해야 하는 기능, 특징, 제약조건 등을 명시한다.
- 시스템의 모든 동작뿐만 아니라 성능, 보안, 사용성과 같은 품질도 기술되어야 한다.
- 프로젝트 유형에 맞게 양식을 만들어 사용한다.
- 소프트웨어 요구사항 명세서에 포함되는 시스템 기능, 데이터, 외부 인터페이스, 품질 요구사항은 요구사항 단위 별로 개별 요구사항 명세서를 작성한다.

2. 요구사항 확인-요구사항 정의

7) 요구사항 명세(Requirement Specification)

; 요구사항 명세 기법

- 요구사항 명세 기법은 정형 명세와 비정형 명세로 구분된다.

구분	정형 명세 기법	비정형 명세 기법
기법	수학적 원리 기반, 모델 기반	상태/기능/객체 중심
작성 방법	수학적 기호, 정형화된 표기법	일반 명사, 동사 등의 자연어를 기반으로 서술 또는 다이어그램으로 작성
특징	<ul style="list-style-type: none">• 요구사항을 정확하고 간결하게 표현할 수 있음• 요구사항에 대한 결과가 작성자에 관계없이 일관성이 있으므로 완전성 검증이 가능함• 표기법이 어려워 사용자가 이해하기 어려움	<ul style="list-style-type: none">• 자연어의 사용으로 인해 요구사항에 대한 결과가 작성자에 따라 다를 수 있어 일관성이 떨어지고, 해석이 달라질 수 있음• 내용의 이해가 쉬어 의사소통이 용이함
종류	VDM, Z, Petri-net, CSP 등	FSM, Decision Table, ER모델링, State Chart(SADT) 등

VDM(Vienna Development Method) : 시스템의 비기능적인 요구사항을 제외한 기능적인 요구사항에만 한정되며 이와 관련한 기능 요구 명세와 검증 설계에 관해 적절한 표기법인 검증 방법을 제공한다.

Z : 논리를 기반으로 한 계산적 표현을 사용하여 여러 특성을 VDM보다 함축적으로 표현할 수 있다.

Petri Net : 그래프에 의한 표기법을 제공하며, 병렬 처리를 기술할 때 유한 상태 기계의 한계성을 극복하도록 고안되었다.

FSM(Finite State Machine) : 유한상태기계를 의미한다.

Decision Table : 판단표라고 하며, 복잡한 의사결정논리를 기술하는데 사용되며, 판단표는 4개의 4분원으로 구성된다.

2. 요구사항 확인-요구사항 정의

8) 요구사항 확인(Requirement Validation, 요구사항 검증)

; 요구사항 확인은 개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동이다.

- 분석가가 요구사항을 정확하게 이해한 후 요구사항 명세서를 작성했는지 확인(Validation)하는 것이 필요하다.
- 요구사항이 실제 요구를 반영하는지, 서로 상충되는 요구사항은 없는지 등을 점검한다.
- 개발이 완료된 후 문제가 발견되면 재작업 비용이 발생할 수 있으므로 요구사항 검증은 매우 중요하다.
- 요구사항 명세서의 내용이 이해하기 쉬운지, 일관성은 있는지, 회사의 기준에는 맞는지, 그리고 누락된 기능은 없는지 등을 검증(Verification)하는 것이 중요하다.
- 요구사항 문서는 이해관계자들이 검토해야 한다.
- 요구사항 검증 과정을 통해 모든 문제를 확인할 수 있는 것은 아니다.
- 일반적으로 요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 형상 관리를 수행한다.

형상 관리(SCM: Software Configuration Management): 소프트웨어 개발 단계의 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터 등을 통칭하여 형상이라고 한다. 형상 관리는 소프트웨어의 개발 과정에서 만들어지는 형상들의 변경 사항을 관리하는 일련의 활동을 말한다.

2. 요구사항 확인-요구사항 정의 기출문제

기출문제 확인(요구사항 정의)

1. 요구사항 분석에서 비기능적(Nonfunctional) 요구에 대한 설명으로 옳은 것은?

- ① 시스템의 처리량(Throughput), 반응 시간 등의 성능 요구나 품질 요구는 비기능적 요구에 해당하지 않는다.
- ② '차량 대여 시스템이 제공하는 모든 화면이 3초 이내에 사용자에게 보여야 한다'는 비기능적 요구이다.
- ③ 시스템 구축과 관련된 안전, 보안에 대한 요구사항들은 비기능적 요구에 해당하지 않는다.
- ④ 금융 시스템은 조회, 인출, 입금, 송금의 기능이 있어야 한다'는 비기능적 요구이다.

설명 : 1번의 경우 성능요구나 품질 요구는 비기능 요구사항에 해당

2번의 경우 비기능 요구사항 중 성능 요구사항에 해당한다.

3번의 경우 안전, 보안에 관한 요구사항은 비기능 요구 해당한다.

4번의 경우 기능 요구 사항에 해당한다.

2. 요구사항 명세 기법에 대한 설명으로 틀린 것은?

- ① 비정형 명세 기법은 사용자의 요구를 표현할 때 자연어를 기반으로 서술한다.
- ② 비정형 명세 기법은 사용자의 요구를 표현할 때 Z 비정형 명세 기법을 사용한다.
- ③ 정형 명세 기법은 사용자의 요구를 표현할 때 수학적 원리와

3. 요구 분석(Requirement Analysis)에 대한 설명으로 틀린 것은?

- ① 요구 분석은 소프트웨어 개발의 실제적인 첫 단계로, 사용자의 요구에 대해 이해하는 단계라 할 수 있다.
- ② 요구 추출(Requirement Elicitation)은 프로젝트 계획 단계에 정의한 문제의 범위 안에 있는 사용자의 요구를 찾는 단계이다.
- ③ 도메인 분석(Domain Analysis)은 요구에 대한 정보를 수집하고 배경을 분석하여 이를 토대로 모델링을 하게 된다.
- ④ 기능적(Functional) 요구에서 성능, 보안, 품질, 안정 등에 대한 요구사항을 도출한다.

설명 : 성능, 보안, 품질, 안정 등에 대한 요구사항은 비기능적 요구사항에 해당된다.

4. 요구사항 개발 프로세스의 순서로 옳은 것은?

㉠ 도출(Elicitation)	㉡ 분석(Analysis)
㉢ 명세(Specification)	㉣ 확인(Validation)

- ① ㉠ - ㉡ - ㉢ - ㉣
- ② ㉠ - ㉢ - ㉡ - ㉣
- ③ ㉠ - ㉣ - ㉡ - ㉢
- ④ ㉠ - ㉡ - ㉣ - ㉢

설명 : 개발에 대한 타당성이 충족이 되면 도출된 요구사항을 분석을 하고 문서화(명세)하고, 확인(검증)하는 단계로 진행이 된다.

2. 요구사항 확인-요구사항 정의 기출문제

기출문제 확인(요구사항 정의)

5. 요구사항 분석이 어려운 이유가 아닌 것은?

- ① 개발자와 사용자 간의 지식이나 표현의 차이가 커서 상호 이해가 쉽지 않다.
- ② 사용자의 요구는 예외가 거의 없어 열거와 구조화가 어렵지 않다.
- ③ 사용자의 요구사항이 모호하고 불명확하다.
- ④ 소프트웨어 개발 과정 중에 요구사항이 계속 변할 수 있다.

설명 : 1,3,4 번과 같은 이유로 인해 사용자의 요구는 수시로 예외가 발생할 수 있어 열거와 구조화가 매우 까다롭다.

6. 요구사항 검증(Requirements Validation)과 관련한 설명으로 틀린 것은?

- ① 요구사항이 고객이 정말 원하는 시스템을 제대로 정의하고 있는지 점검하는 과정이다.
- ② 개발 완료 이후에 문제점이 발견될 경우 막대한 재작업 비용이 들 수 있기 때문에 요구사항 검증은 매우 중요하다.
- ③ 요구사항이 실제 요구를 반영하는지, 문서상의 요구사항은 서로 상충되지 않는지 등을 점검한다.
- ④ 요구사항 검증 과정을 통해 모든 요구사항 문제를 발견할 수 있다.

설명 : 요구사항 검증 과정을 통해 모든 요구사항 문제를 발견한다면 그건 100% 완벽한 프로그램에 해당하기에 있을 수 없는 일이다.

하여, 요구사항 검증을 하더라도 모든 요구사항 문제를 발견할 수 없다.

2. 요구사항 확인 - 요구사항 분석

1) 요구사항 분석의 개요

; 요구사항 분석은 소프트웨어 개발의 실제적인 첫 단계로 개발 대상에 대한 사용자의 요구사항을 이해하고 문서화(명세화)하는 활동을 의미한다.

- 사용자 요구의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다.
- 사용자의 요구를 정확하게 추출하여 목표를 정하고, 어떤 방식으로 해결할 것인지를 결정한다.
- 요구사항 분석을 통한 결과는 소프트웨어 설계 단계에서 필요한 기본적인 자료가 되므로 사용자의 요구사항을 정확하고 일관성 있게 분석하여 문서화해야 한다.
- 소프트웨어 분석가에 의해 요구사항 분석이 수행되며, 이 작업 단계를 요구사항 분석 단계라고 한다.
- 요구사항 분석을 위해 UML(Unified Modeling Language), 자료 흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec.), 개체 관계도(ERD), 상태 전이도(STD), 제어 명세서 등의 도구를 이용한다.

UML : Unified Modeling Language의 약자로, 단어 그대로 해석하면 통합 모델링 언어라는 뜻이다. 객체지향 소프트웨어를 개발할 때 시스템과 산출물을 명세화, 시각화, 문서화할 때 사용한다.

ERD : Entity Relationship Diagram의 약자로, 개체-관계 모델, 테이블간의 관계를 설명해주는 다이어그램이라고 볼 수 있으며, 이를 통해 프로젝트에서 사용되는 DB의 구조를 한눈에 파악할 수 있다.

상태 전이도(STD) : State Transition Diagram의 약자로, 시스템에 어떤 일이 발생할 경우 시스템의 상태와 변화를 모델링 하는 것이다.

Mini-Spec : 자료 흐름도 상의 최하위 처리 절차를 상세하게 기술하는 데 사용하는 도구로 프로세스 명세서라고도 한다.

2. 요구사항 확인 - 요구사항 분석

2) 구조적 분석 기법

; 구조적 분석 기법은 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법으로, 다음과 같은 특징이 있다.

- 도형 중심의 분석용 도구와 분석 절차를 이용하여 사용자의 요구사항을 파악하고 문서화한다.
- 도형 중심의 도구를 사용하므로 분석가와 사용자 간의 대화가 용이하다.
- 하향식 방법을 사용하여 시스템을 세분화할 수 있고, 분석의 중복을 배제할 수 있다.
- 사용자의 요구사항을 논리적으로 표현하여 전체 시스템을 일관성 있게 이해할 수 있다.
- 시스템 분석의 질이 향상되고, 시스템 개발의 모든 단계에서 필요한 명세서 작성이 가능하다.

하향식 방법 : 한 장의 종이에 소프트웨어의 모든 기능을 모델링 할 수 없으므로 소프트웨어의 기능을 전체적인 수준에서 상세 수준까지 위에서 아래로 단계별로 분리하여 모델링 하는 것을 의미한다.

2. 요구사항 확인 - 요구사항 분석

3) 자료 흐름도(DFD)


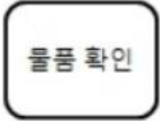
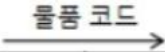
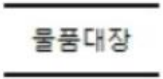
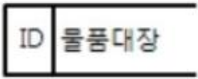


; 자료 흐름도(DFD: Data Flow Diagram)는 요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법으로 자료 흐름 그래프, 버블 차트라고도 한다. 자료의 흐름이란, 자료는 각 절차에 따라 컴퓨터 기반의 시스템 내부를 흘러 다니는데 이를 자료의 흐름이라 한다.

- 시스템 안의 프로세스와 자료 저장소 사이에 자료의 흐름을 나타내는 그래프로 자료 흐름과 처리를 중심으로 하는 구조적 분석 기법에 이용된다.
- 자료 흐름도는 자료 흐름과 기능을 자세히 표현하기 위해 단계적으로 세분화 된다.
- 자료는 처리(Process)를 거쳐 변환될 때마다 새로운 이름이 부여되며, 처리는 입력 자료가 발생하면 기능을 수행한 후 출력 자료를 산출한다.
- 자료 흐름도에서는 자료의 흐름과 기능을 프로세스(Process), 자료 흐름(Flow), 자료 저장소(Data Store), 단말(Terminator)의 네 가지 기본 기호로 표시한다.

2. 요구사항 확인 - 요구사항 분석

3) 자료 흐름도(DFD)

; 자료 흐름도 표기법

기 호	의 미	표기법	
		Yourdon/DeMacro	Gane/Sarson
프로세스 (Process)	<ul style="list-style-type: none">• 자료를 변환시키는 시스템의 한 부분 (처리 과정)을 나타내며 처리, 기능, 변환, 버블이라고도 한다.• 원이나 둥근 사각형으로 표시하고 그 안에 프로세스 이름을 기입한다.		
자료 흐름 (Flow)	<ul style="list-style-type: none">• 자료의 이동(흐름)을 나타낸다.• 화살표 위에 자료의 이름을 기입한다.		
자료 저장소 (Data Store)	<ul style="list-style-type: none">• 시스템에서의 자료 저장소(파일, 데이터 베이스)를 나타낸다.• 도형 안에 자료 저장소 이름을 기입한다.		
단말 (Terminator)	<ul style="list-style-type: none">• 시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받는다(정보의 생산자와 소비자)• 도형 안에 이름을 기입한다.		

; Yourdon/DeMarco와 Gane/Sarson에 의해 두 가지 방법으로 표기할 수 있으나 Yourdon/DeMarco 표기 방법이 주로 사용된다.

2. 요구사항 확인 - 요구사항 분석

4) 자료 사전

; 자료 사전(DD: Data Dictionary)은 자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것이며, 이처럼 데이터를 설명하는 데이터를 데이터의 데이터 또는 메타 데이터(Meta Data)라고 한다.

- 자료 흐름도에 시각적으로 표시된 자료에 대한 정보를 체계적이고 조직적으로 모아 개발자나 사용자가 편리하게 사용할 수 있다.
- 자료 사전에서 사용되는 표기 기호는 다음과 같다.

기 호	의 미
=	자료의 정의: ~로 구성되어 있다(is composed of)
+	자료의 연결: 그리고(and)
()	자료의 생략: 생략 가능한 자료(Optional)
[]	자료의 선택: 또는(or)
{ }	자료의 반복: Iteration of ① { } _n : n번 이상 반복 ② { } ⁿ : 최대로 n번 반복 ③ { } _m ⁿ : m 이상 n 이하로 반복
* *	자료의 설명: 주석(Comment)

2. 요구사항 확인-요구사항 분석 기출문제

기출문제 확인(요구사항 분석)

1. 소프트웨어 개발 방법 요구사항 분석(Requirements Analysis)과 거리가 먼 것은?

- ① 비용과 일정에 대한 제약 설정
- ② 타당성 조사
- ③ 요구사항 정의 문서화
- ④ 설계 명세서 작성

설명 : 요구사항 분석은 설계 단계에서 필요한 기본적인 자료를 준비하여 요구사항 명세서를 만드는 것이다. 요구사항 분석 과정이 아니라 설계과정에서 수행하는 작업이다.

2. 소프트웨어 개발 단계에서 요구분석 과정에 대한 설명으로 거리가 먼 것은?

- ① 분석 결과의 문서화를 통해 향후 유지보수에 유용하게 활용할 수 있다.
- ② 개발 비용이 가장 많이 소요되는 단계이다.
- ③ 자료흐름도, 자료 사전 등이 효과적으로 이용될 수 있다.
- ④ 보다 구체적인 명세를 위해 소단위 명세서(Mini-Spec)가 활용될 수 있다.

설명 : 요구사항 분석은 소프트웨어 개발의 실질적인 첫 단계로 이 단계에서는 사용자의 요구의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다. 비용을 설정하는 단계에서는 비용이 많이 들지

3. DFD(Data Flow Diagram)에 대한 설명으로 틀린 것은?

- ① 자료흐름 그래프 또는 버블(Bubble)차트라고도 한다.
- ② 구조적 분석 기법에 이용된다.
- ③ 시간 흐름을 명확하게 표현할 수 있다.
- ④ DFD의 요소는 화살표, 원, 사각형, 직선(단선/이중선)으로 표시 한다.

설명 : DFD는 자료의 흐름을 표현하는 도구이지 시간 흐름을 명확하게 표시할 수 있는 도구는 아니다.

4. 자료흐름도(Data Flow Diagram)의 구성 요소로 옳은 것은?

- ① process, data flow, data store, comment
- ② process, data flow, data store, terminator
- ③ data flow, data store, terminator, data dictionary
- ④ process, data store, terminator, mini-spec

설명 : 자료 흐름도(DFD)의 구성 요소에는 프로세스(Process), 자료 흐름(Data Flow), 자료 저장소(Data Store), 단말(Terminator)로 이루어진다.

5. 다음 중 자료 사전(Data Dictionary)에서 선택의 의미를 나타내는 것은?

- ① [] ② {} ③ + ④ =

설명 : []는 자료의 선택(또는 or), {}은 자료의 반복(Iteration Of), +는 자료의 연결(그리고, and), =은 자료의 정의(~으로 구성되어 있다.is

2. 요구사항 확인-요구사항 분석 기출문제

기출문제 확인(요구사항 분석)

6. 자료 사전에서 자료의 반복을 의미하는 것은?

- ① = ② () ③ {} ④ []

설명 : ()는 자료의 생략(optional)을 의미하는 기호이다.

7. 자료 사전에서 자료의 생략을 의미하는 기호는?

- ① {} ② ** ③ = ④ ()

설명 : * *는 자료의 설명(주석, comment)이다.

8. 소프트웨어 설계에서 요구사항 분석에 대한 설명으로 틀린 것은?

- ① 소프트웨어가 무엇을 해야 하는가를 추적하여 요구사항 명세를 작성하는 작업이다.
- ② 사용자의 요구를 추출하여 목표를 정하고 어떤 방식으로 해결할 것인지 결정하는 단계이다.
- ③ 소프트웨어 시스템이 사용되는 동안 발견되는 오류를 정리하는 단계이다.
- ④ 소프트웨어 개발의 출발점인 동시에 실질적인 첫 번째 단계이다.

설명 : 소프트웨어 시스템이 사용되는 동안 발견되는 오류를 정리하는 과정은 형상 관리이다.

9. 자료 흐름도(DFD)의 각 요소 별 표기 형태의 연결이 옳지 않은 것은?

- ① Process : 원 ② Data Flow: 화살표

- ③ Data Store: 삼각형 ④ Terminator: 사각형

설명 : Data Store는 자료 저장소를 의미하며 표기법으로 (단선/이중선)으로 표기하며 평행선이라고 불린다. 아울러 평행선 안에 자료 저장소의 이름을 기입한다.

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO)

1) 요구사항 분석을 위한 CASE(자동화 도구)

; 요구사항 분석을 위한 자동화 도구는 요구사항을 자동으로 분석하고, 요구사항 분석 명세서를 기술하도록 개발된 도구를 의미한다. 요구사항 분석을 위한 자동화 도구 사용의 이점은 다음과 같다.

- 표준화와 보고를 통한 문서화 품질 개선
- 데이터베이스가 모두에게 이용 가능하다는 점에서 분석자들 간의 적절한 조정
- 교차 참조도와 보고서를 통한 결함, 생략, 불일치 등의 발견 용이성
- 변경이 주는 영향 추적의 용이성
- 명세에 대한 유지보수 비용의 축소
- S/W 라이프 사이클 전 단계의 연결
- 모델들 사이의 오류 검사
- 모델의 오류 검증
- 자료 흐름도(DFD) 등의 다이어그램(Diagram) 작성
- 다양한 소프트웨어 개발 모형 지원
- 시스템 문서화 및 명세화를 위한 그래픽 지원

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO)

1) 요구사항 분석을 위한 CASE(자동화 도구)

; 요구사항 분석을 위한 자동화 도구의 종류

- SADT(Structured Analysis and Design Technique) : SoftTech사에서 개발한 것으로 시스템 정의, 소프트웨어 요구사항 분석, 시스템/소프트웨어 설계를 위해 널리 이용되어 온 구조적 분석 및 설계 도구이다. 구조적 요구 분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구이다.

블록 다이어그램은 핵심 프로세스가 블록으로 표시되는 시스템의 기본 부분을 나타내는 다이어그램이다.

- SREM(Software Requirements Engineering Methodology, RSL/REVS) : TRW 사가 우주 국방 시스템 그룹에 의해 실시간 처리 소프트웨어 시스템에서 요구사항을 명확히 기술하도록 할 목적으로 개발한 것으로, RSL과 REVS를 사용하는 자동화 도구이다.

RSL(Requirement Statement Language) : 요소, 속성, 관계, 구조들을 기술하는 요구사항 기술 언어

요소	요구사항 명세를 개발하기 위해 사용되는 개체와 개념
속성	요소를 수정하거나 수식(修飾= 장식)하기 위한 것
관계	개체들 간의 관계
구조	정보 흐름을 묘사하기 위한 것

RE(on System) : RSL로 기술된 요구사항들을 자동으로 분석하여 요구사항 분석 명세서를 출력하는 요구사항 분석기

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO)

1) 요구사항 분석을 위한 CASE(자동화 도구)

; 요구사항 분석을 위한 자동화 도구의 종류

- PSL/PSA : 미시간 대학에서 개발한 것으로 PSL과 PSA를 사용하는 자동화 도구이다.

PSL(Problem Statement Language) : 문제(요구사항) 기술 언어

PSA(Problem Statement Analyzer) : PSL로 기술한 요구사항을 자동으로 분석하여 다양한 보고서를 출력하는 문제 분석기

- TAGS(Technology for Automated Generation of Systems) : 시스템 공학 방법 응용에 대한 자동 접근 방법으로, 개발 주기의 전 과정에 이용할 수 있는 통합 자동화 도구이다.

구성 : IORL, 요구사항 분석과 IORL 처리를 위한 도구, 기초적인 TAGS 방법론

IORL: 요구사항 명세 언어

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO)

2) HIPO(Hierarchy Input Process Output)

; HIPO는 시스템의 분석 및 설계나 문서화할 때 사용되는 기법으로, 시스템 실행 과정인 입력, 처리, 출력의 기능을 나타낸다.

- 기본 시스템 모델은 입력, 처리, 출력으로 구성되며, 하향식 소프트웨어 개발을 위한 문서화 도구임
- 체계적인 문서 관리가 가능하다
- 기호, 도표 등을 사용하므로 보기 쉽고 이해하기도 쉽다.
- 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
- 변경, 유지보수가 용이하다.
- 시스템의 기능을 여러 개의 고유 모듈들로 분할하여 이들 간의 인터페이스를 계층 구조로 표현한 것을 HIPO Chart라고 한다.

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO)

2) HIPO(Hierarchy Input Process Output)

; HIPO Chart의 종류

- HIPO Chart의 종류에는 가시적 도표(Visual Table of Contents), 총체적 도표(Overview Diagram), 세부적 도표(Detail Diagram)가 있다.

- 가시적 도표(도식 목차) : 시스템의 전체적인 기능과 흐름을 보여주는 계층(Tree)구조도
- 총체적 도표(총괄도표, 개요 도표) : 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표
- 세부적 도표(상세 도표) : 총체적 도표에 표시된 기능을 구성하는 기본 요소들을 상세히 기술하는 도표

2. 요구사항 확인 - 요구사항 분석(CASE와 HIPO) 기출문제

기출문제 확인(요구사항 분석(CASE와 HIPO))

1. SoftTech사에서 개발한 것으로 구조적 요구 분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구는?

- ① SREM ② PSL/PSA
- ③ HIPO ④ SADT

설명 : SADT는 SoftTech사에서 개발한 것으로 구조적 요구 분석을 하기 위해 블록 다이어그램을 채택한 자동화 도구

SREM : TRW사, 우주 국방 시스템, 실시간 처리, RSL언어, REVS분석 명세서 출력하는 요구사항 분석기

PSL/PSA : 미시간 대학, 문제 기술언어, 문제 분석기

2. HIPO(Hierarchy Input Process Output)에 대한 설명으로 거리가 먼 것은?

- ① 상향식 소프트웨어 개발을 위한 문서화 도구이다.
- ② HIPO 차트 종류에는 가시적 도표, 총체적 도표, 세부적 도표가 있다.
- ③ 기능과 자료의 의존관계를 동시에 표현할 수 있다.
- ④ 보기 쉽고 이해하기 쉽다.

설명 : HIPO는 하향식 소프트웨어 개발을 위한 문서화 도구이다.

3. CASE(Computer Aided Software Engineering)의 주요 기능으로 옳지 않은 것은?

- ① S/W 라이프 사이클 전 단계의 연결

4. HIPO(Hierarchy Input Process Output)에 대한 설명으로 옳지 않은 것은?

① HIPO 다이어그램에는 가시적 도표(Visual Table of Contents), 총체적 다이어그램(Overview Diagram), 세부적 다이어그램(Detail Diagram)의 세 종류가 있다.

② 가시적 도표(Visual Table of Contents)는 시스템에 있는 어떤 특별한 기능을 담당하는 부분의 입력, 처리, 출력에 대한 전반적인 정보를 제공한다.

③ HIPO 다이어그램은 분석 및 설계 도구로서 사용된다.

④ HIPO는 시스템의 설계나 시스템 문서화용으로 사용되고 있는 기법이며, 기본 시스템 모델은 입력, 처리, 출력으로 구성된다.

설명 : 가시적 도표(도식 목차)는 시스템의 전체적인 기능과 흐름을 보여주는 계층(Tree)구조도를 의미하다.

5. 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력을 기술하는 HIPO 패키지에 해당하는 것은?

- ① Overview Diagram ② Detail Diagram
- ③ Visual Table of Contents ④ Index Diagram

2. 요구사항 확인 - UML(Unified Modeling Language)

1) UML(Unified Modeling Language)의 개요

; UML은 시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어이다. 사람, 자동차, 컴퓨터, 동물 등과 같이 우리 주위에서 사용되는 물질적 이거나 개념적인 것을 개체(Entity)라고 한다. 이러한 개체를 컴퓨터 내부에 추상적으로 표현한 것을 사물(Things) 또는 객체(Object)라고 하는데, 다이어그램을 표현할 때는 사물보다는 객체라는 표현을 주로 사용한다.

- UML은 Rumbaugh(OMT), Booch, Jacobson 등의 객체지향 방법론의 장점을 통합하였으며, 객체 기술에 관한 국제 표준화 기구인 OMG(Object Management Group)에서 표준으로 지정하였다.
- UML을 이용하여 시스템의 구조를 표현하는 6개의 구조 다이어그램과 시스템의 동작을 표현하는 7개의 행위 다이어그램을 작성할 수 있다.
- 각각의 다이어그램은 사물과 사물 간의 관계를 용도에 맞게 표현한다.
- UML의 구성 요소에는 사물(Things), 관계(Relationships), 다이어그램(Diagram) 등이 있다.

Rumbaugh(OMT) : 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링 하는 객체지향 분석기법

Booch : 클래스와 객체들을 분석 및 식별하고 클래스의 속성과 연산을 정의하는 기법

Jacobson : 유스케이스를 사용하여 분석하는 방법

2. 요구사항 확인 - UML(Unified Modeling Language)

2) 사물(Things)

; 사물은 모델을 구성하는 가장 중요한 기본 요소로, 다이어그램 안에서 관계가 형성될 수 있는 대상들을 말한다.

- 사물에는 구조 사물, 행동 사물, 그룹 사물, 주해 사물이 있다.

사물	내용
구조사물 (Structural Things)	- 시스템의 개념적, 물리적 요소를 표현 - 클래스, 유스케이스, 컴포넌트, 노드 등
행동사물 (Behavioral Things)	- 시간과 공간에 따른 요소를 행위를 표현 - 상호작용, 상태머신
그룹사물 (Grouping Things)	- 요소들을 그룹으로 묶어서 표현 - 패키지
주해사물 (Annotation Things)	- 부가적인 설명이나 제약조건 등을 표현 - 노트

유스케이스(use case)는 행위자(actor)가 관심을 가지고 있는 유용한 일을 달성하기 위한 시나리오의 집합을 명시한다.
(예. 음료 자판기의 유스케이스: "콜라 사기" / 시나리오: "재고 없음", "금액이 맞지 않음" 등)

컴포넌트(Component)는 컴포넌트를 한마디로 표현하자면 소프트웨어 시스템에서 독립적인 업무 또는 독립적인 기능을 수행하는 '모듈'로서 이후 시스템을 유지보수 하는데 있어 교체 가능한 부품을 의미한다.
문서, 소스코드, 파일, 라이브러리 등과 같은 모듈화된 자원으로 재사용이 가능하다.

노드(Node)는 하나의 기억 공간을 말한다.

2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

; 관계는 사물과 사물 사이의 연관성을 표현하는 것으로, 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계, 실체화 관계 등이 있다.

● 연관(Association) 관계

- 사물 사이를 실선으로 연결하여 표현하며, 방향성은 화살표로 표현한다.
- 서로에게 영향을 주는 양방향 관계의 경우 화살표를 생략하고 실선으로만 연결한다.
- 연관에 참여하는 객체의 개수를 의미하는 다중도(Multiplicity)를 선 위에 표기한다.

다중도	의미
1	1개의 객체가 연관되어 있다.
n	n개의 객체가 연관되어 있다.
0..1	연관된 객체가 없거나 1개만 존재한다.
0..* 또는 *	연관된 객체가 없거나 다수일 수 있다.
1..*	연관된 객체가 적어도 1개 이상이다.
n..*	연관된 객체가 적어도 n개 이상이다.
n..m	연관된 객체가 최소 n개에서 최대 m개이다.

2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

● 연관(Association) 관계 예제

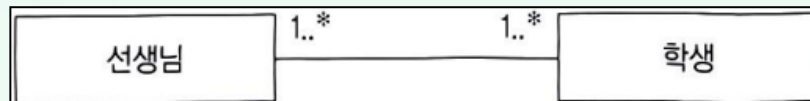
- ① 사람이 집을 소유하는 관계이다. 사람은 자기가 소유하고 있는 집에 대해 알고 있지만 집은 누구에 의해 자신이 소유되고 있는지 모른다는 의미이다.



해설 : '사람' 쪽에 표기된 다중도가 '1'이므로 집은 한 사람에 의해서만 소유될 수 있다.

'집' 쪽에 표기된 다중도가 '1'이므로 사람은 집을 하나만 소유할 수 있다.

- ② 선생님은 학생을 가르치고 학생은 선생님으로부터 가르침을 받는 것과 같이 선생님과 학생은 서로 관계가 있다.



해설 : '선생님' 쪽에 표기된 다중도가 1..* 이므로 학생은 한 명 이상의 선생님으로부터 가르침을 받는다. '학생' 쪽에 표기된 다중도가 1..*이므로 선생님은 한 명 이상의 학생을 가르친다.

2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

- 집합(Aggregation) 관계

; 집합 관계는 하나의 사물이 다른 사물에 포함되어 있는 관계를 표현한다.

- 포함하는 쪽(전체, Whole)과 포함되는 쪽(부분, Part)은 서로 독립적이다.
- 포함되는 쪽(부분, Part)에서 포함하는 쪽(전체, Whole)으로 속이 빈 마름모를 연결하여 표현한다.

- 집합(Aggregation) 관계 예제

- 프린터는 컴퓨터에 연결해서 사용할 수 있으며, 다른 컴퓨터에 연결해서 사용할 수도 있다.



2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

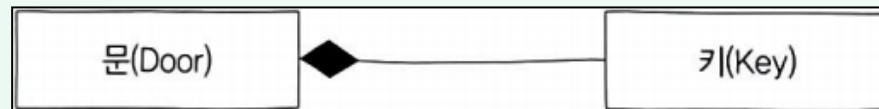
- 포함(Composition) 관계

; 포함 관계는 집합 관계의 특수한 형태로 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 표현한다.

- 포함하는 쪽(전체, Whole)과 포함되는 쪽(부분, Part)은 서로 독립될 수 없고 생명주기를 함께 한다.
- 포함되는 쪽(부분, Part)에서 포함하는 쪽(전체, Whole)으로 속이 채워진 마름모를 연결하여 표현한다.

- 포함(Composition) 관계 예제

- 문을 열 수 있는 키는 하나이며, 해당 키로 다른 문은 열 수 없다. 문이 없어지면 키도 더 이상 필요하지 않다.



2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

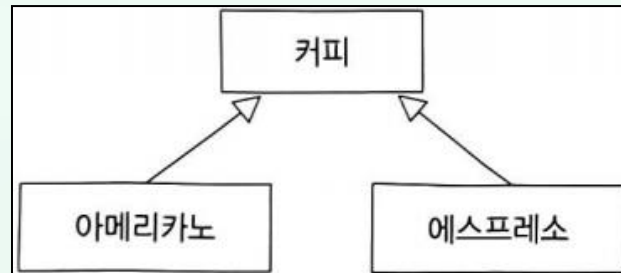
- 일반화(Generalization) 관계

; 일반화 관계는 하나의 사물이 다른 사물에 비해 더 일반적이지 구체적이지를 표현한다.

- 예를 들어 사람은 여자와 남자보다 일반적인 개념이고 반대로 여자와 남자는 사람보다 구체적인 개념이다.
- 보다 일반적인 개념을 상위(부모), 보다 구체적인 개념을 하위(자식)라고 부른다.
- 구체적(하위)인 사물에서 일반적(상위)인 사물 쪽으로 속이 빈 화살표를 연결하여 표현한다.

- 일반화(Generalization) 관계 예제

- 아메리카노와 에스프레소는 커피이다. 다시 말하면, 커피에는 아메리카노와 에스프레소가 있다.



2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

- 의존(Dependency) 관계

; 의존 관계는 연관 관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계를 표현한다.

- 하나의 사물과 다른 사물이 소유 관계는 아니지만 사물의 변화가 다른 사물에도 영향을 미치는 관계이다.
- 일반적으로 한 클래스가 다른 클래스를 오퍼레이션의 매개 변수로 사용하는 경우에 나타나는 관계이다.
- 영향을 주는 사물(이용자)이 영향을 받는 사물(제공자) 쪽으로 점선 화살표를 연결하여 표현한다.

- 의존(Dependency) 관계 예제

- 등급이 높으면 할인율을 적용하고, 등급이 낮으면 할인율을 적용하지 않는다. 등급을 이용해 할인율을 적용하는 것처럼 할인율을 적용하기 위해 등급을 매개 변수로 사용하는 관계를 말한다.



2. 요구사항 확인 - UML(Unified Modeling Language)

3) 관계(Relationships)

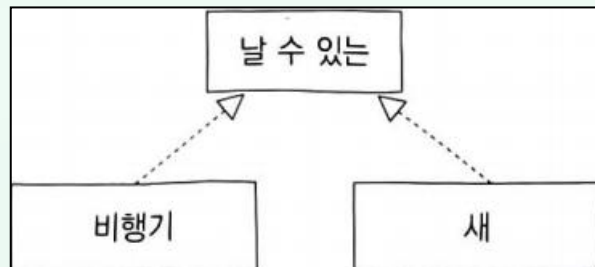
- 실체화(Realization) 관계

; 실체화 관계는 사물이 할 수 있거나 해야 하는 기능(오퍼레이션, 인터페이스)으로 서로를 그룹화할 수 있는 관계를 표현한다.

- 한 사물이 다른 사물에게 오퍼레이션을 수행하도록 지정하는 의미적 관계이다.
- 사물에서 기능 쪽으로 속이 빈 점선 화살표를 연결하여 표현한다.

- 실체화(Realization) 관계 예제

- 비행기는 날 수 있고 새도 날 수 있다. 그러므로 비행기와 새는 날 수 있다는 행위로 그룹화할 수 있다.



2. 요구사항 확인 - UML(Unified Modeling Language)

4) 다이어그램(Diagram)

; 다이어그램은 사물과 관계를 도형으로 표현한 것이다.

- 여러 관점에서 시스템을 가시화한 뷰(View)를 제공함으로써 의사소통에 도움을 준다.
- 정적 모델링에서는 주로 구조적 다이어그램을 사용하고 동적 모델링에서는 주로 행위 다이어그램을 사용한다.

; 구조적(Structural) 다이어그램의 종류

종류	내용
클래식 다이어그램 (Class Diagram)	클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다
객체 다이어그램 (Object Diagram)	- 클래스에 속한 사물들 즉 인스턴스를 특정 시점의 객체와 객체 사이의 관계로 표현한다 - 럼바우(Rumbaugh) 객체지향 분석 기법에서 객체 모델링에 활용된다
컴포넌트 다이어그램 (Component Diagram)	- 실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현한다 - 구현 단계에서 사용한다
배치 다이어그램 (Deployment Diagram)	- 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현한다
복합체 구조 다이어그램 (Composite Structure Diagram)	클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현한다
패키지 다이어그램 (Package Diagram)	유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한다

2. 요구사항 확인 - UML(Unified Modeling Language)

4) 다이어그램(Diagram)

; 행위(Behavioral) 다이어그램의 종류

종류	내용
유스케이스 다이어그램 (Use Case Diagram)	- 사용자의 요구를 분석하는 것으로 기능 모델링 작업에 사용한다 - 사용자와 사용 사례로 구성된다
시퀀스 다이어그램 (Sequence Diagram)	상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현한다
커뮤니케이션 다이어그램 (Communication Diagram)	동작에 참여하는 객체들이 주고받는 메시지와 객체들 간의 연관 관계를 표현한다
상태 다이어그램 (State Diagram)	- 하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는지를 표현한다 - 롬바우 객체지향 분석 기법에서 사용된다
활동 다이어그램 (Activity Diagram)	시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현한다
상호작용 개요 다이어그램 (Interaction Overview Diagram)	상호작용 다이어그램 간의 제어 흐름을 표현한다
타이밍 다이어그램 (Timing Diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다

2. 요구사항 확인 - UML(Unified Modeling Language)

5) 스테레오 타입(Stereotype)

; 스테레오 타입은 UML에서 표현하는 기본 기능 외에 추가적인 기능을 표현하기 위해 사용한다.

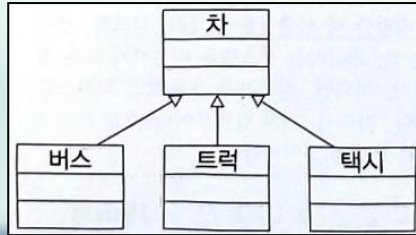
- 길러멧(Guillemet)이라고 부르는 겹화살 괄호(<< >>) 사이에 표현할 형태를 기술한다.
- 주로 표현되는 형태는 다음과 같다.

«include» 연결된 다른 UML요소에 대해 포함관계
«exclude» 확장 관계
«interface» 인터페이스 정의
«exception» 예외 정의
«constructor» 생성자 역할 수행

2. 요구사항 확인 - UML(Unified Modeling Language) 기출문제

기출문제 확인(UML(Unified Modeling Language))

1. 아래의 UML모델에서 '차'클래스와 각 클래스의 관계로 옳은 것은?



- ① 추상화 관계 ② 의존 관계
- ③ 일반화 관계 ④ 그룹 관계

설명 : 차는 버스, 트럭, 택시 등의 일반적인 개념이고 반대로는 버스, 트럭, 택시는 차의 구체적인 개념이다.

2. UML의 기본구성 요소가 아닌 것은?

- ① Things ② Terminal
- ③ Relationship ④ Diagram

설명 : UML의 구성요소에는 사물, 관계, 다이어그램이 존재한다.

3. UML 확장 모델에서 스테레오 타입 객체를 표현할 때 사용하는 기호로 맞는 것은?

- ① << >> ② (()) ③ {{ }} ④ [[]]

설명 : 스테레오 타입을 표현하는 기호는 겹화살표 이다.

4. 럼바우(Rumbaugh) 객체지향 분석 기법에서 동적 모델링에 활용되는 다이어그램은?

- ① 객체 다이어그램(Object Diagram)
- ② 패키지 다이어그램(Package Diagram)
- ③ 상태 다이어그램(State Diagram)
- ④ 자료 흐름도(Data Flow Diagram)

설명 : 럼바우는 그래픽 표기법을 이용하는 모델링 객체지향 분석기법이다.

상태 다이어그램은 하나의 객체가 자신이 속한 클래스의 상태 변화, 다른 객체와의 상호 작용에 따라 상태가 어떻게 변하는지를 표현하는 행위 다이어그램의 하나이다.

5. UML에서 활용되는 다이어그램 중, 시스템의 동작을 표현하는 행위(Behavioral) 다이어그램에 해당하지 않는 것은?

- ① 유스케이스 다이어그램(Use Case Diagram)
- ② 시퀀스 다이어그램(Sequence Diagram)
- ③ 활동 다이어그램(Activity Diagram)
- ④ 배치 다이어그램(Deployment Diagram)

설명 : 배치 다이어그램은 결과물, 프로세스, 컴포넌트 등을 물리적 요소들의 위치를 표현하는 구조적 다이어그램의 한 종류이다.

6. UML 다이어그램 중 정적 다이어그램이 아닌 것은?

- ① 컴포넌트 다이어그램 ② 배치 다이어그램

2. 요구사항 확인 - UML(Unified Modeling Language) 기출문제

기출문제 확인(UML(Unified Modeling Language))

7. UML 모델에서 한 사물의 명세가 바뀌면 다른 사물에 영향을 주며, 일반적으로 한 클래스가 다른 클래스를 오퍼레이션의 매개 변수로 사용하는 경우에 나타나는 관계는?

- ① Association ② Dependency
- ③ Realization ④ Generalization

설명 : 한 사물의 변화가 다른 사물에 영향을 주게 되면, 영향을 받는 사물은 영향을 주는 사물에게 의존할 수 밖에 없다.

8. UML 모델에서 한 객체가 다른 객체에게 오퍼레이션을 수행하도록 지정하는 의미적 관계로 옳은 것은?

- ① Dependency ② Realization
- ③ Generalization ④ Association

설명 : 실체화 관계는 사물이 할 수 있거나 해야 하는 기능(오퍼레이션, 인터페이스)으로 서로를 그룹화할 수 있는 관계를 의미한다.

오퍼레이션을 수행하도록 지정하는 의미적 관계이다. 빈 점선 화살표를 이용한다.

9. UML 다이어그램이 아닌 것은?

- ① 액티비티 다이어그램(Activity Diagram)
- ② 절차 다이어그램(Procedural Diagram)
- ③ 클래스 다이어그램(Class Diagram)
- ④ 시퀀스 다이어그램(Sequence Diagram)

10. UML(Unified Modeling Language)에 대한 설명 중 틀린 것은?

- ① 기능적 모델은 사용자 측면에서 본 시스템 기능이며, UML에서는 Use case Diagram을 사용한다.
- ② 정적 모델은 객체, 속성, 연관관계, 오퍼레이션의 시스템의 구조를 나타내며, UML에서는 Class Diagram을 사용한다.
- ③ 동적 모델은 시스템의 내부 동작을 말하며, UML에서는 Sequence Diagram, State Diagram, Activity Diagram을 사용한다.

④ State Diagram은 객체들 사이의 메시지 교환을 나타내며, Sequence Diagram은 하나의 객체가 가진 상태와 그 상태의 변화에 의한 동작순서를 나타낸다.

설명 : 상태 다이어그램은 하나의 객체가 가진 상태와 그 상태의 변화에 의한 동작 순서를 나타내는 것이고, 시퀀스 다이어그램은 객체들 사이에 메시지 교환을 나타낸다.

11. 다음의 설명에 해당하는 언어는?

객체지향 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화하는 데 사용된다. 즉, 개발하는 시스템을 이해하기 쉬운 형태로 표현하여 분석가, 의뢰인, 설계자가 효율적인 의사소통을 할 수 있게 해 준다. 따라서, 개발 방법론이나 개발 프로세스가 아니라 표준화된 모델링 언어이다.

- ① JAVA ② C ③ UML ④ Python

2. 요구사항 확인 - 주요 UML 다이어그램

1) 유스케이스(Use Case) 다이어그램

; 유스케이스 다이어그램은 개발될 시스템과 관련된 외부 요소들, 즉 사용자와 다른 외부 시스템들이 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점(View)에서 표현한 것이다.

- 외부 요소와 시스템 간의 상호 작용을 확인할 수 있다.
- 사용자의 요구사항을 분석하기 위한 도구로 사용된다.
- 시스템의 범위를 파악할 수 있다.

; 유스케이스 다이어그램의 구성 요소

- 유스케이스 다이어그램은 시스템 범위, 액터, 유스케이스, 관계로 구성된다.

① 시스템(System) / 시스템 범위(System Scope)

- ▶ 시스템 내부에서 수행되는 기능들을 외부 시스템과 구분하기 위해 시스템 내부의 유스케이스들을 사각형으로 묶어 시스템 범위를 표현한다.
- ▶ 사각형 안쪽 상단에 시스템 명칭을 기술한다.

예시) <상품주문>

2. 요구사항 확인 - 주요 UML 다이어그램

1) 유스케이스(Use Case) 다이어그램

; 유스케이스 다이어그램의 구성 요소

② 액터(Actor)

▶ 시스템과 상호작용을 하는 모든 외부요소로, 사람이나 외부시스템을 의미(외부 요소이므로 시스템 밖에 표현)

▶ 주액터 : 시스템을 사용함으로써 이득을 얻는 대상으로, 주로 사람이 해당함
(예시) '비회원', '회원', '고객'이 주액터에 해당된다.

▶ 부액터 : 주액터의 목적 달성을 위해 시스템에 서비스를 제공하는 외부 시스템으로, 조직이나 기관 등이 될 수 있음

예시) '재고 시스템', '결제 시스템', '배송업체' 가 부액터에 해당된다.

③ 유스케이스(Use Case)

▶ 사용자가 보는 관점에서 시스템이 액터에게 제공하는 서비스 또는 기능을 표현한 것이다.

▶ 타원으로 표현하며 타원 안쪽이나 아래쪽에 유스케이스 이름을 기술한다.

▶ 부분적 수행은 허용되지 않으며 유스케이스는 분석, 설계, 테스트 등 개발 전 과정에서 이용된다.

예시)'상품조회', '이름으로 조회', '브랜드로 조회', '상품주문', '배송조회' 등이 있다.

2. 요구사항 확인 - 주요 UML 다이어그램

2) 클래스(Class) 다이어그램

; 클래스 다이어그램은 시스템을 구성하는 클래스, 클래스의 특성인 속성과 오퍼레이션, 속성과 오퍼레이션에 대한 제약조건, 클래스 사이의 관계를 표현한 것이다.

- 클래스 다이어그램은 시스템을 구성하는 요소에 대해 이해할 수 있는 구조적 다이어그램이다.
- 클래스 다이어그램은 시스템 구성 요소를 문서화하는 데 사용된다.
- 코딩에 필요한 객체의 속성, 함수 등의 정보를 잘 표현하고 있어 시스템을 모델링 하는 데 자주 사용된다.

; 클래스 다이어그램의 구성 요소

- 클래스 다이어그램은 클래스, 제약조건, 관계 등으로 구성된다.

① 클래스(Class)

- ▶ 클래스는 각각의 객체들이 갖는 속성과 오퍼레이션(동작)을 표현함
- ▶ 일반적으로 3개의 구획(Compartment)으로 나뉘 클래스의 이름, 속성, 오퍼레이션을 표기함
- ▶ 속성(Attribute) : 클래스의 상태나 정보를 표현함
- ▶ 오퍼레이션(Operation): 클래스가 수행할 수 있는 동작으로, 함수(메소드, Method)라고도 함

2. 요구사항 확인 - 주요 UML 다이어그램

2) 클래스(Class) 다이어그램

② 제약 조건

- ▶ 속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있다면 이를 적음

③ 관계

- ▶ 관계는 클래스와 클래스 사이의 연관성을 표현함
- ▶ 클래스 다이어그램에 표현하는 관계에는 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계가 있음

2. 요구사항 확인 - 주요 UML 다이어그램

3) 순차(Sequence) 다이어그램

; 순차 다이어그램은 시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호 작용하는 과정을 액터, 객체, 메시지 등의 요소를 사용하여 그림으로 표현한 것이다.

- 순차 다이어그램은 시스템이나 객체들의 상호 작용 과정에서 주고받는 메시지를 표현한다.
- 순차 다이어그램을 통해 각 동작에 참여하는 시스템이나 객체들의 수행 기간을 확인할 수 있다.
- 순차 다이어그램은 클래스 내부에 있는 객체들을 기본 단위로 하여 그들의 상호작용을 표현한다. 클래스 내부에 있는 객체들의 상호 작용을 표현한다는 것은 클래스가 수행할 수 있는 동작인 오퍼레이션을 표현한다는 의미이다. 예를 들어 <회원> 클래스에 '로그인 버튼 클릭', '상품 선택', '결제', '정보 입력' 등의 오퍼레이션이 있다면 이들 오퍼레이션이 어느 클래스와 상호 작용하는지를 표현하는데, 순차 다이어그램에서는 오퍼레이션을 메시지로 표현한다.
- 순차 다이어그램은 주로 기능 모델링에서 작성한 유스케이스 명세서를 하나의 표현 범위로 하지만, 하나의 클래스에 포함된 오퍼레이션을 하나의 범위로 표현하기도 한다.

2. 요구사항 확인 - 주요 UML 다이어그램

3) 순차(Sequence) 다이어그램

; 순차 다이어그램의 구성 요소

- 순차 다이어그램은 액터, 객체, 생명선, 실행, 메시지 등으로 구성된다.

① 액터(Actor)

▶ 시스템으로부터 서비스를 요청하는 외부 요소로 사람이나 외부 시스템을 의미함

② 객체(Object)

▶ 메시지를 주고받는 주체

③ 생명선(Lifeline)

▶ 객체가 메모리에 존재하는 기간으로 객체 아래쪽에 점선을 그어 표현함

④ 실행 상자(Active Box)

▶ 객체가 메시지를 주고 받으며 구동되고 있음을 표현함

⑤ 메시지(Message)

▶ 객체가 상호 작용을 위해 주고받는 메시지

2. 요구사항 확인 - 주요 UML 다이어그램 기출문제

기출문제 확인(주요 UML 다이어그램)

1. UML에서 시퀀스 다이어그램의 구성 항목에 해당하지 않는 것은?

- ① 생명선 ② 실행
- ③ 확장 ④ 메시지

설명 : 순차(Sequence) 다이어그램은 시스템이나 객체들이 메시지를 주고 받으며 시간의 흐름에 따라 상호작용하는 과정을 액터, 객체, 메시지 등의 요소를 사용하여 그림으로 표현한 것

순차 다이어그램의 구성 요소 : 액터, 객체, 생명선, 실행 박스, 메시지

2. 유스케이스(Use Case)의 구성 요소 간의 관계에 포함되지 않는 것은?

- ① 연관 ② 확장
- ③ 구체화 ④ 일반화

설명 : 유스케이스 다이어그램은 개발될 시스템과 관련된 외부 요소들, 사용자와 외부 시스템들이 개발될 시스템을 이용해서 수행할 수 있는 기능을 사용자의 관점(View)에서 표현한 것

유스케이스 다이어그램에서는 연관, 포함, 확장, 일반화 관계를 표현

3. 유스케이스(Use case)에 대한 설명 중 옳은 것은?

- ① 유스케이스 다이어그램은 개발자의 요구를 추출하고 분석하기 위해 주로 사용한다.
- ② 액터는 대상 시스템과 상호 작용하는 사람이나 다른 시스템에 의한

4. 클래스 다이어그램의 요소로, 다음 설명에 해당하는 용어는?

- 클래스의 동작을 의미한다
- 클래스에 속하는 객체에 대하여 적용될 메소드를 정의한 것이다.
- UML에서는 동작에 대한 인터페이스를 지칭한다고 볼 수 있다.

- ① Instance
- ② Operation
- ③ Item
- ④ Hiding

설명 : 클래스 다이어그램의 구성요소는 속성, 동작(오퍼레이션, 기능, 함수, 메소드), 제약 조건, 관계가 있다.

5. UML 다이어그램 중 시스템 내 클래스의 정적 구조를 표현하고 클래스와 클래스, 클래스의 속성 사이의 관계를 나타내는 것은?

- ① Activity Diagram
- ② Model Diagram
- ③ State Diagram
- ④ Class Diagram

설명 : 클래스 다이어그램은 시스템을 구성하는 클래스, 클래스의 특성인 속성, 오퍼레이션, 속성과 오퍼레이션에 대한 제약조건, 클래스 간에 관계를 표현한 것

6. 순차 다이어그램과 관련한 설명으로 틀린 것은?

- ① 객체들의 상호 작용을 나타내기 위해 사용한다.



감사합니다.