5과목 : 시스템분석 및 설계 (산업기사)

Check 1. 시스템 개요 [출제빈도: 上]

1. 시스템 분석 및 설계 이해하기

*** S/W (프로젝트) 개발 절차**
: 요구 분석 -> 설계 -> 구현(코딩) -> 테스트(시험) -> 유지보수

요구분석	어떻게 만들어 줄까?(무엇) -> 분석 도구
설계	요구 분석 결과를 가지고 구체적인 기능과 구조를 체계화 (어떻게) -> 설계 기법
구현(코딩)	프로그램 언어를 선정하고, 설계 명세서를 컴퓨터가 이해할 수 있도록 표현 -> 프로그램 언어 선정 기준, 코딩 표준화
테스트(시험)	요구 사항에 맞게 작동하는가? -> 테스트 기법
유지보수	버전 업데이트 및 새로운 기능 추가 (S/W 개발 비용 70% 차지) -> 유지보수 과정 (유지보수 요구 →현 시스템 이해 →수정, 테스트)

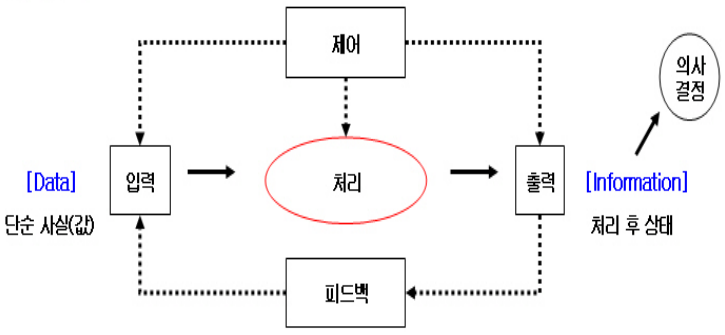
2. 시스템 ★

1) 정의: 어떤 **목적**을 위하여 하나 이상의 **상호 관련된 요소**의 유기적인 **결합체**

2) 특성

- 종 합성 : 시스템은 종합적인 결합체이다.
- 목 적성 : 시스템은 공통의 목적이 있다.
- 자 동성 : 시스템은 자동 조치한다.
- 제 어성 : 정해진 목표를 달성하기 위해 오류가 발생하지 않도록 사태를 감시하는 특성

3) 기본 요소



① 입력 (Input) : 처리할 데이터 및 조건을 부여하는 요소
② 처리 (Process) : 결과를 산출하기 위해 입력 자료를 조건에 맞게 처리하는 요소
③ 출력 (Output) : 처리 결과를 산출하는 요소
④ 제어 (Control) : 각 과정의 제기능이 올바로 수행되는지를 통제하거나 관리하는 요소
⑤ **피드백 (FeedBack)** : 처리 결과를 평가하여 불충분한 경우, 목적 달성을 위해 반복 처리하는 요소

3. 시스템 분석 및 설계자 조건 ★★☆☆☆

1) 기업의 목적 이해
2) 업계 동향 및 관계 법규 등도 파악해야 함
3) 컴퓨터 기술과 관리 기법 알아야 함
4) 창조성
4) 현장 분석 경험 중요
5) 시간 배정과 계획 등을 빠른 시간 내에 파악해야 함
*** 분석가는 기계 중심적이어야 함 (X) -> 인간 중심적 분석**

4. 시스템 개발 주기 (순서) ★

목적 설정→ 현장 **조사** 및 **분석**→ **설계**(시스템,프로그램) → **구현**(프로그래밍) →**테스트, 디버깅**→ **운용** → **유지보수**

1) 조사
- 현 시스템의 상태와 요구사항을 파악하는 단계

[조사 방법]
① 현장 조사 : 작업 현장 방문 조사
② 자료 조사 : 작업 관련 문서를 조사
③ 면담 조사 : 담당자를 만나서 조사
④ 질문서 조사 : 수집할 자료가 여러 사람의 의견으로부터 도출되어야 하거나, 지리적으로 멀리 떨어져 있는 곳의 정보를 수집하고자 할 때, 주로 사용되는 조사 방법

2) 분석
- 조사 내용을 요구 분석 명세서를 작성하는 단계

3) 설계
- 요구 분석 명세서를 구체화하는 단계

4) 구현
- 프로그래밍 언어로 원시 코드를 작성하는 단계 (코딩)
- 설계 단계에서 산출된 설계 사양서에서 내용을 컴퓨터가 인식할 수 있는 프로그램 코드로 변환, 작성하는 단계

5) 테스트
- 사용자의 요구에 맞게 수행이 되는지를 검증하는 단계

6) 운용
- 실제 업무 처리에 적용하는 단계

7) 유지보수
- 시스템을 항상 최상의 상태로 유지하는 단계 (개발 비용의 70% 소요)

* 소프트웨어 비용산출 시 고려해야 할 요소
: 제품의 복잡도, 제품의 크기, 프로그래머의 자질

* 비용 산정 모델 : COCOMO

Check 2. 코드 설계 [출제빈도: 上]

1. 코드(code) 기능 ★★★★★

*** 개념 이해하기 : 왜 주민등록번호(코드)를 이용 할까?**
-> 1977년 5월 5일 부산 김유신 (770505-1693333)

1) 정의 : 컴퓨터에서 자료처리를 쉽게 하기 위해 사용하는 기호

2) 코드의 기능

- 3대 기능 : **배**열, **분**류, **식**별
- 기타 기능 : 표준화, 암호화, 확장성, 연상(표의성), 단순화 (호환성, 중복성, 복잡성 X)

-> **연상성** : 코드에 대한 해독을 쉽게 하는 것으로 코드를 보는 순간 그 코드의 실체를 알 수 있도록 하는 코드
-> **확장성** : 기본 사항을 바꾸지 않고 코드 부여 대상의 신규 발생, 변경, 폐지에 대응할 수 있는 코드의 성질

2. 코드 설계 ★★★☆☆

① 코드화 대상 결정
② 코드화 목적의 명확화
③ 코드 부여 대상 수 확인
④ 사용 범위 결정
⑤ 사용 기간 결정
⑥ 코드화 대상의 특성 분석
⑦ 코드 부여 방식의 결정
⑧ 코드의 문서화 (코드표)

3. 코드 설계 시 유의사항 ★★★☆☆

-기계 처리의 용이성, 취급의 용이성, 분류의 용이성
-단순성, 고유성, 표의성, 공통성, 체계성
(다양성, 비계체성, 복잡성 X)

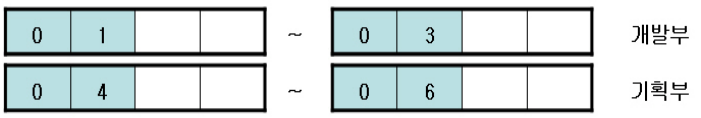
4. 코드의 종류

1) 순서코드((Sequence) : 일련 번호 ★☆☆☆☆

- 코드화 대상 항목을 어떤 일정한 배열로 일련 번호를 배당하는 코드로서 항목 수가 적고 장래에 **다시 작성하는 일이 없는 항목**에 적합한 코드
- 단순하고 이해하기 쉬움
- 발생 순서대로 코드 부여시 확장성이 좋음
- 명확한 분류 기준이 없음 -> 코드 분류 어려움
- 누락된 자료 삽입이 어려움 -> 융통성이 적음
- ex) 교실 입장 순서대로 번호(코드) 붙이기

2) 구분(Block) 코드 ★☆☆☆☆

- 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련 번호를 부여하는 방법
- 적은 자리수로 많은 항목을 표시할 수 있음
- 예비코드를 사용할 수 있어 항목의 추가가 용이
- 공통된 특성별로 분류 및 집계가 용이
- ex) 부서별로 사원번호(코드) 붙이기



3) 그룹 분류 코드 (Group Classification) : 대분류, 중분류, 소분류 ★★☆☆☆

- 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하여 일련번호를 부여하는 방법
- ex) 부서별(지사,사업부,팀)로 사원번호(코드) 붙이기, **학번**

4) 10진 코드(Decimal) : 도서분류코드 ★☆☆☆☆

100 정보처리기사	110 필기	111	전자계산기구조
		112	데이터베이스
		113	운영체제

5) 표의 숫자 코드(Significant Digit) ★★☆☆☆
: 물리적 수치 (길이, 넓이, 부피, 무게)

- 코드화 대상 항목의 길이, 넓이, 부피, 무게 등을 나타내는 문자, 숫자 혹은 기호를 그대로 코드로 사용
- ex) 합판 코드

코 드	두께	폭	길이
110-800-500	110	800	500

6) 연상코드 (Mnemonic, 기호) : 가전제품 ★★☆☆☆

- 코드화 대상의 명칭이나 약호를 코드의 일부에 넣어서 대상을 외우기 쉽도록 하는 코드
- ex) 가전 제품

7) 코드 오류 종류 ★★★★☆

-필사 오류 (transcription error, 오자오류)
: 입력 시 임의의 한자리를 잘못 기록한 경우 (34278 -> 34578)

-전위 오류 (transposition error)
: 입력 시 좌우 자리를 바꾸어 기록하는 경우 (1996 -> 1969)

Check 3. 입력 설계 [출제빈도: 上]

1. 입력 설계 순서 ★★★★☆

1) 정의
: 입력데이터를 어떤 매체를 이용하고 어떤 형태로 입력할 것인지를 설계

2) 입력 설계 순서
(중간고사 OMR 카드 입력 절차를 생각해 보세요.)

① 입력정보 발생 : 입력 정보의 명칭과 목적 결정
② 입력정보 수집 : 입력 정보 수집 주기 및 시기 결정
③ 입력정보 매체화 : 입력 매체, 입력 정보의 형태 선택 (디스켓, 자기 테이프, 디스크, OMR)
④ 입력정보 투입 : 투입 주기 및 시기 결정
⑤ 입력정보 내용: 입력 항목명, 순서, 배열, 입력 정보에 대한 오류 검사

2. 입력 방식 ★

① 집중 입력 방식 : 일정시간 동안 수집 -> 일괄 입력
- 발생한 데이터를 전표상에 기록하고, 일정한 시간단위로 일괄 수집하여 전산부서에서 입력매체에 수록

② 턴 어라운드 방식 : 지로 용지
- 입력된 자료가 처리되어 일단 출력된 후 이용자를 거쳐 다시 입력되는 방식으로, 공과금, 보험료 징수 등의 지로 용지를 처리하는데 사용되는 입력 방식

③ 분산 입력 방식
- 데이터를 발생한 장소에서 입력하는 방식

3. 원시전표 기입 시 고려 사항 ★★☆☆☆

* 개념 이해하기 : 설문 조사 양식을 생각해 보세요.
(원시 전표 : 발생한 자료를 최초로 기록한 종이)
① 가능한 한 기입량은 적고, 간단하게 적을 수 있어야 함
② 일정 순서대로 기입할 수 있어야 함
③ 기입상 혼란을 일으킬 수 있는 경우에는 전표상에 기입 요령을 명시해야 함

Check 4. 출력 설계 [출제빈도: 上]

1. 출력 설계 순서 ★★★★★

1) 정의

: 컴퓨터가 처리한 결과를 어떤 매체를 통해, 어떤 형식으로 출력할 것인지를 설계

2) 출력 설계 순서

* 개념 이해하기
: 중간고사 OMR 카드 처리 후 절차를 생각해 보세요.

① 출력정보 내용 : 출력할 항목 결정 (순서, 크기, 자릿수, 숫자, 영문자, 한글, 한자)
② 출력정보 매체화
③ 출력정보 분배 : 전달 경로 결정 (분배 책임자, 분배 방법, 분배 주기)
④ 출력정보 이용 : 출력 정보명과 사용 목적 결정

2. 출력 매체 ★★★☆☆

*** COM 시스템**(Computre Output Mocroflim)

-처리결과를 마이크로필림에 기록
-축소 보관과 반영구적인 매체로 사용 가능

Check 5. 파일 설계 [출제빈도: 上]

1. 파일 설계 ★★★★☆

1) 성격 (목적) 확인 : 적용 업무 확인 후 파일의 목적, 종류, 명칭을 결정
2) 항목 검토 : 항목 명칭, 항목 배열 순서
3) 특성 조사
4) 매체 검토 : 업무 적합성, 매체 특성 등 검토 후 저장 매체 결정
5) 편성법 검토: 순차, 색인 순차, 랜덤, 리스트 편성 등 결정

2. 파일 ★★★★☆

: 서로 관련 있는 레코드의 집합체

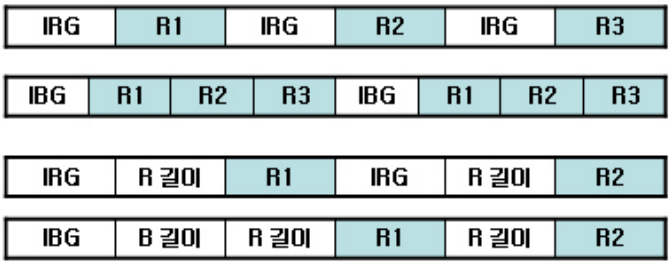
1) 레코드 형식

① 비블록화 고정 길이 레코드

② **블록**화 **고정** 길이 레코드
: **속도** 빠르고 **프로그램** 작성 용이

③ 비블록화 가변 길이 레코드

④ 블록화 가변 길이 레코드



2) 파일의 분류 : 매체, 내용, 편성 방법에 따른 분류

① 매체에 의한 분류
- 직접 저장 장치 : 자기 디스크 등
- 순차 저장 장치 : 자기 테이프

② **파일 편성 방법에 의한 분류**

③ **파일 내용에 의한 분류**

3. 순차 편성(SAM: Sequntial Acses Method) ★☆☆☆☆
: 자기 테이프

-파일의 내용을 추가, 변경, 삭제가 어렵다.
→파일 전체를 복사할 필요가 있다.

4. 색인순차편성(ISAM: Index Sequntial Acses Method) ★
: 자기 디스크

- 인덱스를 통한 랜덤 처리와 데이터의 순차 처리를 병행할 수 있는 파일
-검색 효율적

-삽입, 삭제, 갱신이 용이
-특정 레코드를 처리할 때 여러 단계의 인덱스 처리를 해야 하므로 접근 시간이 느림(랜덤 편성과 비교)

색인 영역 (Index)	**마스터 색인** : 레코드가 어느 실린더 인덱스 상에 기록되어 있는가의 정보를 수록
	실린더 색인 : 레코드가 저장된 실린더의 정보가 기록되는 인덱스
	트랙 색인 : 레코드가 저장된 트랙을 판별
기본영역 (Prime)	실제 레코드 기록
오버플로 영역	실린더 오버플로 : 기본 데이터 구역에서 오버플로된 데이터를 기록
	독립 오버플로 : 실린더 오버플로 구역에 다시 오버플로가 발생할 경우 저장하는 공간

5. 랜덤 편성 : 해싱 함수 ★★★☆☆

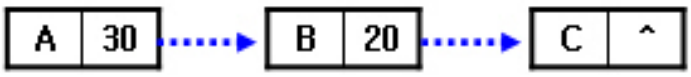
: 처리하고자 하는 레코드를 주소 계산에 의하여 직접 처리

- 키, 주소 변환을 위한 계산 시간 필요
- 충돌(collision) 가능성 있음
- 해싱 함수 선택 시 고려 사항
  : overflow, 충돌, 메모리 낭비의 최소화
- **시노임(Synonym)** : 동일한 버켓 주소를 갖는 레코드들의 집합
- 오버플로 : 버켓 내에 기억 공간이 없는 현상
- 특징 : 검색은 빠르지만 기억공간의 낭비 발생

6. 리스트 편성 : 포인터 ★☆☆☆☆

: 관련되는 데이터 레코드들이 물리적으로는 떨어져 있으나 데이터 레코드에 포함되어 있는 포인터가 순차적으로 데이터 레코드가 저장되어 있는 주소를 지시함으로써 데이터 구조 관계를 유지

- 포인터 내용 변경 -> 삽입, 삭제 쉬움



7. 파일 내용에 의한 분류 ★

1) **마스터 파일 (Master) : 원장**

- 전표 처리에서 원장 또는 대장에 해당되는 파일로서 데이터 처리 시스템에서 중추적 역할을 담당하며 기본이 되는 데이터의 축적 파일
- 트랜잭션 파일에 의해 갱신됨

2) **트랜잭션 파일 (Transaction) : 갱신**

- 마스터 파일의 변경하고자 하는 내용을 검사하거나 갱신할 때 사용되는 정보로서, 일시적인 성격을 지닌 파일

3) **히스토리 파일 (History) : 복구**

- 통계 처리나 파일의 자료에 잘못이 발생하였을 때 파일을 원상 복구하기 위해 사용되는 파일
- 현재까지 변화된 정보를 포함하고 있는 기록 파일

4) 트레일러 파일 (Trailer) : **마스터 파일 끝부분**

- 마스터 파일을 목적에 따라 여러 개의 파일로 나누었을 때 가장 끝부분에 해당하는 파일

Check 6. 프로세스 설계 [출제빈도: 上]

1. 프로세스 설계 시 유의사항 ★★☆☆☆

-입력 정보를 토대로 출력정보를 얻기까지의 처리과정을 설계하는 것

1) 신뢰성과 정확성을 고려
2) 시스템의 상태 및 구성 요소, 기능 등을 종합적으로 표시함
3) 오류 체크 시스템도 고려
4) 예외 사항의 처리 방법에 유의
5) 하드웨어의 기기 구성 및 처리 능력을 고려
6) 프로세스 전개의 사상을 통일할 것 -> 표준화
7) 오퍼레이터의 개입을 많게 할 것 (X) -> 자동화 위배
8) 사용자의 하드웨어와 프로그래밍에 관한 상식 수준을 고려함 (X)
9) 각 부문별 담당자의 책임 범위를 고려함 (X)

2. 입력 단계 오류 체크 ★

1) **유효 범위 체크 (Limit)**

- 입력 자료의 어떤 항목 내용이 논리적으로 정해진 하한치와 상한치 범위 내에 있는가를 체크

2) **일괄 합계 체크 (Batch Total)**

- 입력 자료의 특정 항목 합계 값을 미리 계산해서 이것을 입력 정보와 함께 입력하고, 컴퓨터상에서 계산한 결과와 수동 계산 결과가 같은지를 체크

3) **균형 체크 (Balance)**

- 입력 정보의 두 가지 이상이 특정 항목의 합과 같다는 것을 알고 있을 때, 컴퓨터를 이용해서 계산한 결과와 분명히 같은 지를 체크
- 경리 장부 처리시 차변, 대변의 한계값을 체크하는 데 사용하는 방법으로 대차의 균형이나 가로, 세로의 합계가 일치하는 가를 체크

4) **데이터 수 체크 (Data Count)**
- 컴퓨터로 처리할 데이터의 개수와 컴퓨터로 처리한 데이터의 개수가 같은지의 여부를 체크

5) **체크 디지트 체크 (Check Digit)**

- 오류를 체크할 수 있는 1자리 숫자를 넣어서 오류를 자동으로 체크
- ex) 주민등록번호 : 770101 - 1691833

* 숫자 체크 (Numeric), 순차 체크 (Sequence)

3. 계산 처리 단계 오류 체크 ★☆☆☆☆

1) **불일치 레코드 체크 (Unmatched Record)**

: 마스터 파일과 트랜잭션 파일을 조합하는 경우에 키 항목이 일치 하는지의 여부를 체크

2) **중복 레코드 체크 (Double Record)**

: 마스터 파일을 변경하는 경우 트랜잭션 정보에 동일한 레코드가 여러 개 있는지를 체크

3) **한계 초과 체크 (overflow check)**

: 자리수나 한계를 초과하는지를 검사

4. 처리(Process) 패턴 ★

1) **변환 (Conversion) : 매체 변환**
(ex. 테이프 파일 -> 자기 디스크에 수록하는 처리)

2) **병합 (Merge)**
: 동일한 파일 형식을 가진 두 개 이상의 파일을 하나로 정리하는 처리 패턴

3) **갱신 (Update)**
: 마스터 파일 안의 정보 변동에 의해 추가, 삭제, 교환을 하고 새로운 내용의 마스터 파일을 작성

4) **분배 (Distribution)**
: 어떤 특정한 조건을 부여하여 조건을 만족시키는 정보와 만족시키지 못하는 정보로 분리하는 처리

5. 프로그램 설계서 ★★★★☆

1) 특징

- **프로그램 설계** : 프로그램의 세부사항을 설계하는 단계
- 프로그래머에게 주는 작업 지시서 역할을 함
- 시스템 엔지니어 또는 시스템 분석가가 작성 (프로그래머 작성 X)

2) 포함될 사항

- 입,출력 설계표, 시스템명, 코드표, 프로세스 흐름도 등
- **프로그래밍 지시서**
: 프로그램명, 설계서 작성자명, 처리 개요,
입,출력 일람, 처리 명세, 프로그램의 작성기간 및 비용

3) 작성 효과

- 프로그래머의 인사 이동시 결함을 방지
- 시스템의 수정, 유지보수가 간단
- 비용이 절감되어 장기 계획 수립 가능

Check 7. 시스템 평가 [출제빈도: 中]

1. 시스템 평가 ★★★★★

: 새로운 시스템이 본래의 목적에 만족하는가를 평가

1) 평가 **목적**
- 시스템의 성능과 유용도를 판단
- 처리 비용과 효율 면에서 개선점 파악
- 시스템 운용 관리의 타당성을 파악
- **새로운 프로그래밍 작성 기법을 개발 (X),**
시스템 운영요원의 재훈련 (X)

2) 평가 **방법**

- 이용부분의 만족도 분석
- 프로그램 정확성과 효율성 분석
- 개발비의 운용 비용의 분석
- **시스템 분석가의 만족도 분석 (X)**

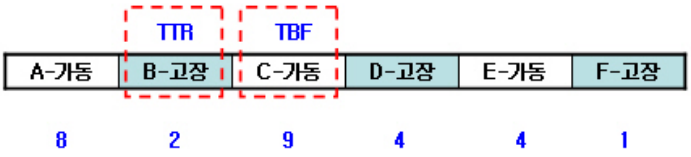
3) 평가 **항목** : 성능(처리시간), 유연성, 기능, 신뢰성, **가격 (X)**

4) 우수한 시스템의 판정 기준

- 시스템 능력
- 시스템 신뢰성
- 시스템 유연성
- **시스템 구축비용 (X)**

2. 평가> 신뢰도 측정 ★★★★☆

-신뢰도: 주어진 시간동안 고장 없이 가동될 확률



1) **MTBF** (평균 고장 간격, Mean Time Between Failure)
- 상호 인접한 고장사이의 가동된 시간 평균
- (A+C+E)/3 = 21/3 = 7

2) **MTTR** (평균 수리 시간, Mean Time To Repair)
- 시스템에 고장이 발생하여 가동하지 못한 시간 평균
- (B+D+F)/3 = 7/3 = 2.3

3) **신뢰도 공식** : MTBF / (MTBF+MTTR)

4) **신뢰도 계산 문제** : 가동시간 / 전체 시스템 운영시간

3. 테스트 종류 ★☆☆☆☆

1) **단위 테스트**
- 개발자나 개발 부서에서 각 모듈에 논리적인 로직이나 인터페이스의 기능을 테스트

2) **통합(결합) 테스트**
- 개발된 모듈들을 통합해 가면서 테스트 것으로 하향식, 상향식, 혼

합식 테스트가 있음
- 프로그램 단위별로 디버깅이 끝난 것을 모아 서로 연관된 프로그램 군(group)을 계통적으로 감시하는 테스트

3) **시스템 테스트**
- 요구 사항을 만족시키는지를 테스트

4. 유지보수 ★☆☆☆☆

- 시스템 개발 단계 중 가장 많은 비용이 투입
- 종류 : 수정 유지 보수, 적응 유지 보수, 예방 유지보수

5. 문서화 목적 및 효과 ★

- 시스템 개발 프로젝트 관리의 효율화
- 개발 진척 관리의 지표가 될 수 있다.
- 소프트웨어 이완의 용이함 (유연성)
- 시스템 유지보수의 효율화
- 시스템의 변경에 따른 혼란을 방지
- 시스템 보수 및 운용하는 그룹에 인계 인수 작업이 용이하다.

- **개발자의 순서도 작성, 코딩, 디버깅, 테스트런만을 위해서이다 (X)**
  -> 전과정
- **시스템 평가를 위해서 필요하다. (X)**
- **개발자 입장에서는 문서가 필요없지만 사용자 입장에서는 반드시 문서가 필요하다. (X)**
- **책임의 명확화 (X)**
- **문서화는 시스템이 모두 개발된 후에 일괄적으로 작업해야 정확하다. (X)**
- **시스템의 전체 개발 시간을 단축할 수 있다. (X)**
- **보안성 (X)**

Check 8. 구조적 개발 방법론 [출제빈도: 上]

1. 개발 방법론 발전 과정

1) 초기 개발 방법 (1950년대)
: 무원칙, 개발자 위주 개발 -> 생산성 저하, 유지보수 어려움

2) 소프트웨어 위기 : H/W 개발 속도 > S/W 개발 속도

3) 소프트웨어 공학 등장
: 신뢰도 높은 S/W를 만들기 위한 방법, 도구와 절차들의 체계화한 학문

4) **구조적 개발 방법론**
① 제어구조 : 순차, 반복, 선택 구조 (GO TO 문 X) -> 순차적 실행 -> 간단하고 이해가 쉬움
② 모듈화 -> 재사용 가능
③ 폭포수 모델이 기본 (계획 -> 요구사항 분석 -> 설계 -> 구현 -> 테스트 -> 유지보수)
④ 소규모 프로젝트 적합

5) **객체 지향 개발 방법론**
① 객체 중심 개발 -> 재사용, 유지보수 우수
-> 소프트웨어 위기 극복
② 객체 특징 : 다형성, 상속성, 추상화, 캡슐화, 정보 은닉

2. 소프트웨어 생명 주기

: 소프트웨어를 개발하기 위해 정의, 개발, 유지보수 과정을 각 단계별로 나눈 것
- 표현 형태 : 폭포수 모형, 프로토타입 모형, 나선형 모형

1) 폭포수 모형 ★★★☆☆

-정의
: 하향식 생명주기 모형으로 각 단계가 끝나는 시점에서 확인, 검증, 검사를 거쳐 다음 단계로 넘어가거나 이전 단계로 환원하면서 구현 및 운영 단계에 이르는 생명주기 모형

-개발 주기
: 계획→요구사항 정의→ **개략(개요,기본)**설계 →상세 설계→구현(코딩)→ 통합 시험 (테스트)→시스템 시행(운용)→유지보수

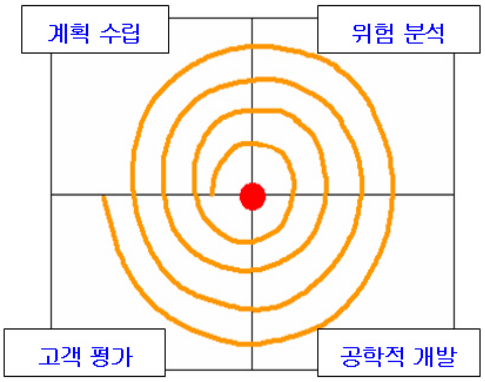
-기본 설계 단계
: 개발될 소프트웨어에 대한 전체적인 하드웨어 및 소프트웨어 구조, 제어구조, 자료 구조의 개략적인 설계를 작성하는 단계

2) 프로토타입 모형 ★☆☆☆☆

- 실제 상황이 나오기전에 가상으로 시뮬레이션을 통하여 최종 결과물에 대한 예측이 가능한 모형
- 시스템 개발 초기에 사용자의 요구 기능을 시제품으로 만들어 사용자로 하여금 기능과 사용성 등에 대해 검증시켜 가면서 시스템을 개발하는 기법

3) 나선형 모형 ★☆☆☆☆

- 시스템 구축시 발생하는 위험을 최소화 할 수 있다.
- 복잡, 대규모 시스템의 소프트웨어 개발에 적합하다.
- 초기에 위험 요소를 발견하지 못할 경우 위험 요소를 제거하기 위해서 많은 비용이 들 수 있다.



3. 구조적 분석 ★★★☆☆

1) 특징

- 시스템을 **하향식으로 분할할 수 있음 (상향식 X)**
- 분석자와 사용자간의 의사 소통이 용이 **(도형 중심)**
- 제어구조 : **순차, 반복, 선택 구조** (GO TO 문 X)

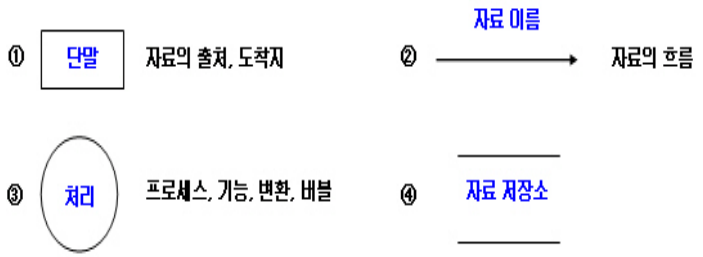
-> 순차적 실행 -> 간단하고 이해가 쉬움
- 폭포수 모델이 기본
(계획 -> 요구사항 분석 -> 설계 -> 구현 -> 테스트 -> 유지보수)

2) 구조적 분석 기법(도구)

- 자료의 흐름과 처리를 중심으로 하는 요구사항 분석 방법
- 종류 : 자료 흐름도, 자료 사전, 소단위 명세서, 개체 관계도, 상태 전이도

① 자료 흐름도 (DFD : Data Flow Diagram) ★★★★☆

*기호와 의미



② 자료 사전 (DD: Data Dictionary) ★★★☆☆

- 시스템과 관련된 모든 자료의 명세와 자료 속성을 파악할 수 있도록 조직화한 도구
- 갱신하기 쉬워야 한다.
- 이름을 가지고 정의를 쉽게 찾을 수 있어야 한다.
- 정의하는 방식이 명확해야 한다.

*기호와 의미

기호	의미	기호	의미	기호	의미
=	정의	+	연결	[|]	선택
{ }	반복	* *	주석, 설명	( )	생략

③ 소단위 명세서 (Mini-Spec.) ★☆☆☆☆

- 구조적 언어나 의사 결정표의 형태로 자료흐름의 최소 단위를 명세화한 도구

* 구조적 언어

1. 고객명세는 고객성명, 고객번호, 고객주소로 구성되어 있으며, 고객성명과 고객번호는 둘 중 하나만 선택이 가능하다.
  1.1 고객성명은....,
  1.2 고객번호는...,
  1.3 고객주소는...,
    1.3.1 .....

* 의사 결정표

		1	2	3	4
조건 항목란	1. 평균 60점 이상인가?	N	N	Y	Y
	2. 과락인 과목은 있는가?	N	Y	N	Y
행동 항목란	1. 합격	N	N	Y	N
	2. 불합격	Y	Y	N	Y

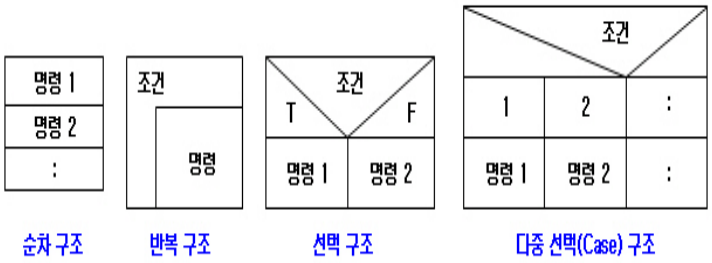
스텁 규칙란

3) 구조적 설계

- 하향식 설계 기법 : 최상위 단계 -> 하위 단계로 설계
- 자료 흐름 중심 설계 기법
- **모듈화** -> 재사용이 쉬움
- 구조적 설계 기법(도구) : N-S Chart, HIPO

① N-S Chart ((Nassi-Shneiderman) ★☆☆☆☆

- 순서도와는 달리 논리기술에 중점을 두고 상자도형을 이용한 도형식 설계도구로 순차, 선택, 반복, 케이스(case) 제어 구조를 표현하는 도구 (GOTO 명령 X)

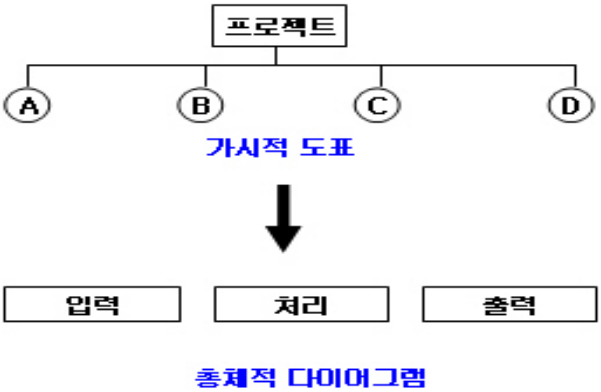


② HIPO (Hierarchy Input Process Output) ★

- 분석, 설계, 문서화에 사용되는 도구이며, 기본 시스템 모델은 입력, 처리, 출력으로 구성됨
- 하향식 소프트웨어 개발을 위한 문서화 도구로서 이해하기 쉬움
- 변경, 유지보수 용이

-종류

가시적 도표 (Visual Table of Contents) =구조도	- 시스템 전체적인 기능과 흐름을 보여주는 계층 구조도
총체적다이어그램 (Overview Diagram) = 개요 도표 집합	- 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표
세부적 다이어그램 (Detail Diagram) =상세도표집합	- 총체적 다이어그램를 상세 기술하는 도표



③ IPT (Improved Programming Technique) ★★★☆☆

- 프로그램의 품질 개선과 생산성 향상을 위한 기법
- 기술적, 관리적 측면에서 모두 우수한 개발 작업이 되도록 함
- HIPO, N-S 차트 등

* 목적
- 개발자의 생산성 향상
- 프로그래밍의 표준화 -> 개인적인 차이 해소

4) 모듈 ★★☆☆☆

- 소프트웨어 구조를 이루는 기본 단위
- 결합되어 통속적으로 실행되지만 컴파일은 독립적임
- 업무 성격이 비슷한 처리에 부품처럼 공통으로 사용할 수 있음
- 모듈 작성은 분담하여 독립적으로 작성할 수 있음
- 사용할 변수를 새로 정의하지 않고 상속하여 사용할 수 있음 (메모리 절약)
- 소프트웨어 복잡도가 감소하고, 변경이 쉬우며 프로그램 구현용이
- 각 모듈은 독립적이며 상호 의존도는 낮도록 설계

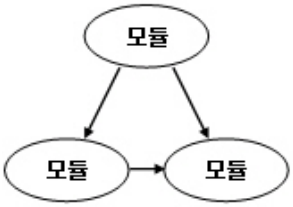
5) 결합도와 응집도 ★★★★☆

결합도	-모듈 간에 상호 의존도 ***좋은 설계** →독립적인 모듈이 되기 위해서는 결합도가 약해야 함 ***종류** : 데이터 결합도 < 스탬프 결합도 < 제어 결합도 < 외부 결합도 < 공통 결합도 < 내용 결합도 ***내용 결합도** - 다른 모듈 내의 외부 선언을 하지 않은 자료를 직접 참조하므로 의존도가 대단히 높고, 순서 변경이 다른 모듈에 영향을 주기 쉬운 모듈 결합도 -가장 강한 결합도를 가지고 있으며, 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 조회하도록 설계되었을 경우와 관계 되는 결합도 ***데이터(자료) 결합도** - 품질이 가장 좋은 결합도
응집도	- 모듈 안의 요소들이 서로 관련되어 있는 정도 - 기능 수행시 모듈간의 최소한의 상호작용을 하여 하나의 기능만을 수행하는 정도를 표현하는 용어 - 한 모듈내에 있는 구성요소의 기능적 관련성을 평가하는 기준으로서, 다른 모듈과의 결합도에 영향을 주는 것 ***좋은 설계** →독립적인 모듈이 되기 위해서는 응집도가 강해야 함 ***종류** 우연적 < 논리적 < 시간적 < 절차적 < 교환적 < 순차적 < 기능적 ***기능적 응집도** : 품질이 가장 좋은 응집도

Check9. 객체지향 개발 방법론 [출제빈도: 上]

1. 구조적 개발 VS 객체지향 개발

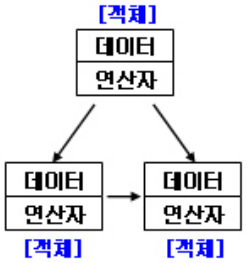
1) 구조적 개발



[장점]
구조 단순 -> 이해 O, 수정 O, 정확 O (C언어)

[단점]
소프트웨어 재사용, 유지보수 어려움 -> 소프트웨어 위기 해결 안됨

2) 객체지향 개발



[장점]
현실 세계를 프로그램에 반영
소프트웨어 재사용, 유지보수 향상 -> 소프트웨어 위기 해결 방안

[관련 용어]
기본 : 객체, 클래스, 메시지
원칙 : 캡슐화, 정보 은폐, 추상화, 상속성, 다형성
데이터 = 상태, 속성(Attribute), 변수, 자료구조

연산자 = 행위, 메소드(Method), 동작(Operation)

2. 객체, 클래스, 메시지 -개념 이해하기 ★

1) 객체 (Object)

- 현실 세계의 개체며 객체들 간의 상호작용은 메시지를 통해 이루어짐

① 데이터
: 객체가 가지고 있는 상태 (속성, Attribute, 변수, 자료구조)

② 연산자
: 객체의 데이터를 처리하는 행위
(메소드, Method, 동작, Operation, 함수, 프로시져)

2) 클래스 (Class)

- 하나 이상의 유사한 객체들을 묶어 공통된 특성을 표현한 데이터 추상화(모델링)를 의미
- 공통된 속성과 연산을 갖는 객체의 집합 (객체의 일반적인 타입)
- 인스턴스 (Instance) : 클래스에 속한 각각의 객체
(객체는 클래스의 인스턴스)

3) 메시지 (Message)

- 객체들 간에 상호작용을 하는데 사용되는 수단
- 객체에서 객체로 메시지가 전달되면 메소드(행위)를 시작함

4) 메소드(method)

- 객체지향 시스템에서 전통적 시스템의 함수(function) 또는 프로시저(procedure)에 해당하는 연산기능
- 객체지향 개념에서 객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산

3. 캡슐화 (Encapsulation) ★★☆☆☆

- 자료 부분과 연산(또는 함수) 부분 등 정보처리에 필요한 기능을 한 테두리로 묶는 것
- 왜? 정보 은폐 -> 외부에서 변경 X
→프로그램 변경에 대한 오류의 파급효과가 적다 (결합도 낮아짐)

-> 재사용 용이, 객체간의 인터페이스 단순화, 응집도 향상

4. 상속 (Inheritance) ★☆☆☆☆

- 상위 클래스의 메소드와 속성을 하위 클래스가 물려받는 것

5. 추상화 (Abstraction) ★☆☆☆☆

6. Booch 기법 ★☆☆☆☆

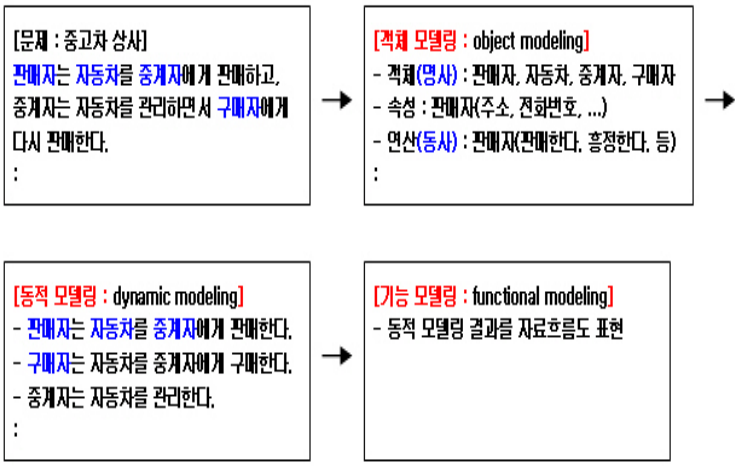
: 데이터 흐름 다이어그램(DFD)을 사용해서 객체를 분해하고, 객체들 간의 인터페이스를 찾아 이것들을 Ada 프로그램으로 변환시키는 기법

- 전체 시스템의 가시화와 실시간처리(real time)에 유용
- 설계를 위한 문서화 기법 강조함
- 분석단계와 구현세부사항 취약함

7. 코드(Coad)와 요돈(Yourdon) 기법

- 구성요소
: 문제 영역 요소, 사람과 상호 작용 요소, 데스크(작업) 관리 요소

8. Rumbaugh 분석 기법 ★★☆☆☆



- **객체 모델링**
: 시스템에서 요구되는 객체를 찾아내어 객체들의 특성을 규명
- **동적 모델링**
: 시간 흐름에 따른 객체들과 객체들 사이의 제어 흐름, 상호 작용, 동작 순서 등을 표현하는 것으로 시스템의 변화를 보여주는 객체 상태 다이어그램을 작성하는 모형
- **기능 모델링**
: 데이터흐름 다이어그램을 이용하여 다수의 프로세스들 간의 데이터 흐름을 중심으로 처리과정 표현