컴퓨터소프트웨어학과
인덕대학교

소프트웨어 요구사항

컴퓨터소프트웨어학과

김병국 교수

구성

- 시스템 분석의 중요성
- IPT 기법
- 객체지향
- 소프트웨어 요구사항

참고자료

IT CookBook, 시스템분석과 설계(개정판) **-** 효과적인 비즈니스 정보시스템 개발, 허원실 저, 한빛출판네트워크, **2015**

1. 시스템 분석/설계의 중요성

❑ 유지보수 단계에서 오류가 발생하면 더 많은 추가비용이 발생함

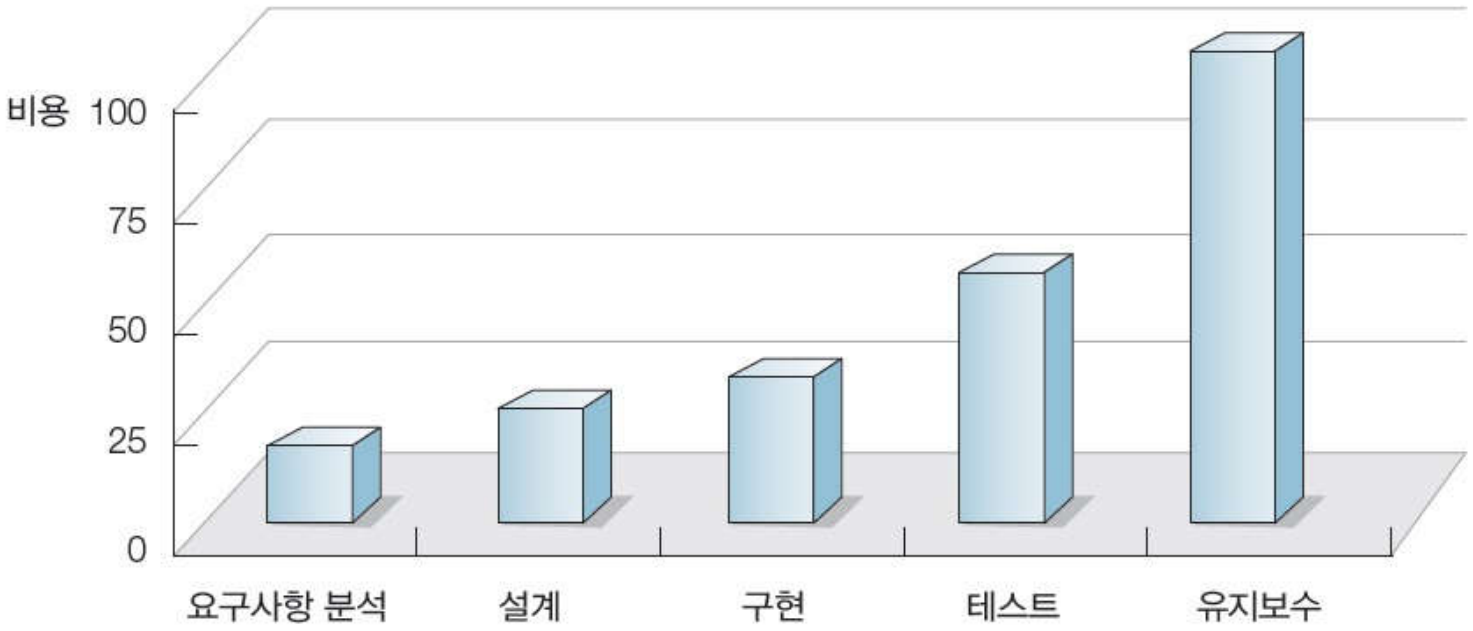


그림 2-1 오류의 발견 시점과 그에 따른 비용

1. 시스템 분석/설계의 중요성

❑ 소프트웨어 개발 비용

- 프로그래밍 이전 단계에서 40~50%가 소요됨
- 소프트웨어 개발에 있어 요구사항 분석과 설계가 체계적으로 이루어지지 않으면 좋은 품질을 기대하기 어려움

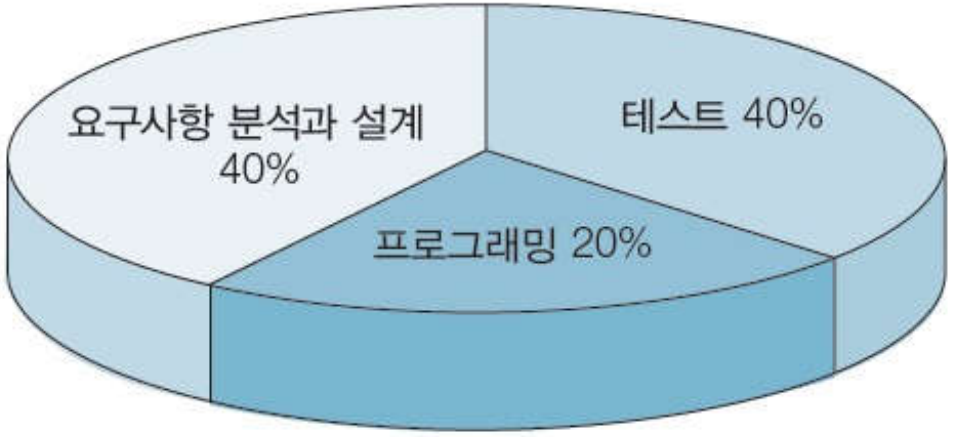


그림 2-2 개발비용 분포

2. 이상적인 SDLC 모형

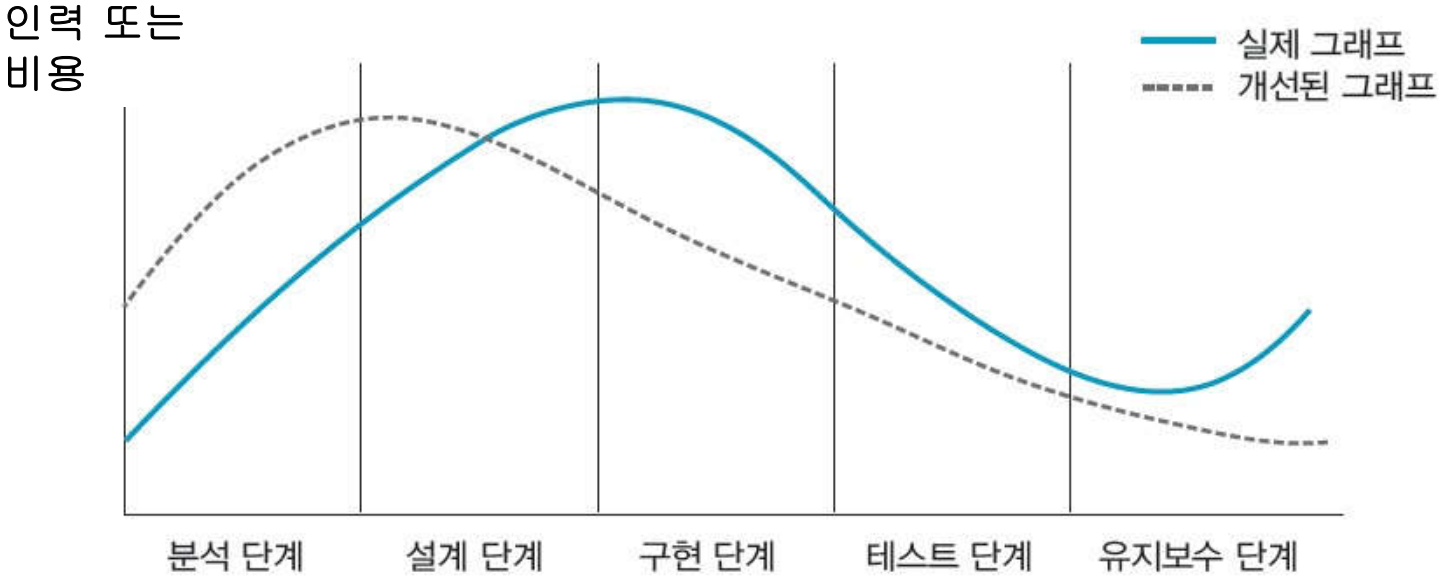


그림 2-3 개선해야 할 SDLC 모형의 단계별 인력소요

3. IPT 기법 [1/12]

❑ 개념

- IPT : Improved Programming Technique
- 주요 목적 :
  - 개발자의 생산성을 향상
  - 프로그램의 품질을 향상 → 유지보수 용이
  - 프로그램의 표준화 → 개인 격차 해소, 교대근무, 인수인계 용이
  - 프로그램의 가독성 향상
  - 경제적인 개발 및 유지보수

3. IPT 기법 [2/12]

❑ 개요

- 기술적 기법
  - 설계 분야: 복합 설계(Composite Design)
  - 코딩 분야: 구조적 프로그래밍(Structured Program)
  - 테스트 분야: 하향식 프로그래밍(Top-Down Programming)
  - 지원 요소 :
    - N-S Chart
    - 프로그램 기술 언어(PDL: Program Description Language) 또는 의사 언어(Pseudo Language)
    - HIPO(Hierarchy plus Input Process Output)
    - 모듈 설계(Module Design)
- 관리적 기법
  - 개발 조직 : 선임 프로그래머 팀(Chief Programmer Team)
  - 품질 관리 :
    - 구조적 검토회(Walk-Through)
    - 검증회(Inspection)
    - 라이브러리(Library)

3. IPT 기법 [3/12]

❑ 기술적 기법 [1/10]

- 복합(Composite) 설계
- 구조적(Structured) 프로그램
- **하향식(Top-Down) 프로그래밍**
- N-S(Nassi-Scheiderman) Chart
- 프로그램 기술 언어(PDL : Program Description Language)
- HIPO(Hierarchy plus Input Process Output)
- 모듈(Module) 설계

❑ 기술적 기법 [2/10]

- 복합(Composite) 설계
  - 상세설계단계에 해당
  - 구조(Structural) 설계 또는 기능(Functional) 설계하고도 함
  - 프로그램의 각 기능을 상세화된 계층 구조로 하여 하향식으로 모듈화

❑ 기술적 기법 [3/10]

▪ 구조적(Structured) 프로그램 [1/2]

- 프로그램의 구조를 단순하게 하기 위한 기법
- 이해와 수정이 쉬움
- 정확성을 검증하기 쉬움
- 결론적으로 제어 구조가 명확한 프로그램 제작이 가능
- 구조 종류 :
  - **순차(Sequence, 순서) 구조** : 기술된 명령문을 순서대로 수행하는 구조
  - **조건(Condition, Selection) 구조** : 일련의 동작 중 조건에 따라 두 가지 이상의 경우 중 한 가지만 선택하여 수행하는 구조
  - **반복(Iteration, Repetition) 구조** : 조건에 따라 임의 동작들을 반복 실행하는 구조

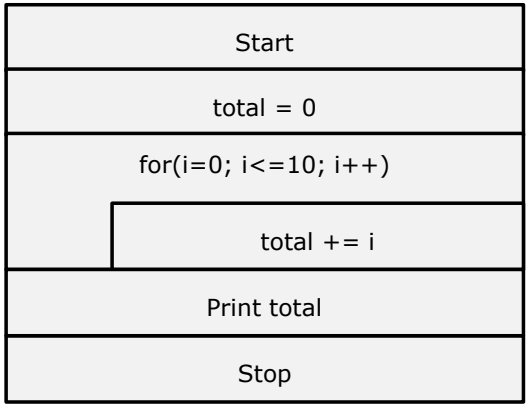
❑ 기술적 기법 [4/10]

▪ 구조적(Structured) 프로그램 특징 [2/2]

- Dijkstra가 제안
- GOTO 문 배제를 통한 프로그램의 가독성 향상
- 논리적 구조를 명확화
- 쉬운 유지보수
- 프로그램 검증 용이
- 신뢰성과 생산성 향상
- 프로그래밍의 효율성 증진
- 하나의 입력을 통한 하나의 출력을 갖도록 설계

❑ 기술적 기법 [5/10]

- N-S(Nassi-Scheiderman) Chart
  - 절차적인 논리적 흐름을 BOX 모양으로 표시
  - 시각적 표현을 중심으로 한 기법
  - 주요 특징 :
    - 화살표가 없음
    - 분기 및 반복 구조를 시각적으로 표시
    - 기능 표현보다는 논리적 표현을 중요시함
    - 순차, 선택, 반복, 케이스 제어 구조를 가짐
    - 구조적 코딩이 용이

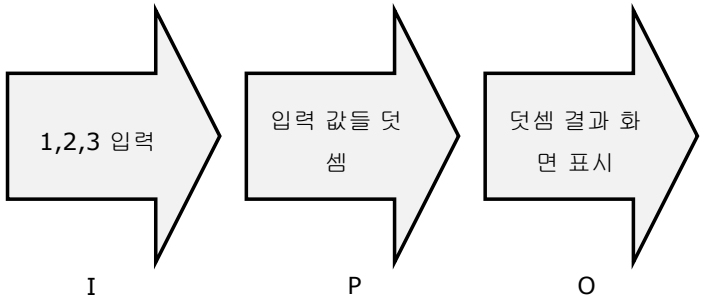


❑ 기술적 기법 [7/10]

- HIPO(Hierarchy plus Input Process Output) [1/2]
  - 문서화와 설계의 효율성을 강화
  - 표준화된 문서 작성 기법을 사용
  - 프로그램의 기능을 계층 구조로 도식화
  - **하향식(Top-Down) 기법**
  - 기능과 자료의 의존 관계를 동시에 표현
  - 기능 중심 설계
  - IPO(입력, 처리, 출력)의 기능을 명확히 설계
  - **도식 목차(Visual Table of Contents), 총괄 다이어그램(Overview Diagram), 상세 다이어그램(Detail Diagram)으로 표현**

❑ 기술적 기법 [8/10]

- HIPO(Hierarchy plus Input Process Output) [2/2]
  - 도식 목차(Visual Table of Contents)
    - 시스템이나 프로그램을 여러 기능으로 분해
    - 분해된 기능 간 관계와 계층 구조를 도표(계층도), 범례(Legend), 보충 설명서(Extended Description)로 표시
  - 총괄 다이어그램(Overview Diagram)
    - 개괄적인 기능을 IPO의 형태로 나타냄
  - 상세 다이어그램(Detail Diagram)으로 표현
    - 상세 사항을 IPO의 형태로 나타냄



❑ 기술적 기법 [9/10]

- 모듈(Module) 설계 특징
  - 프로그램을 여러 개의 작은 단위(모듈, module, 서브루틴)들로 구성
  - 모듈은 독립적으로 컴파일 가능
  - 다른 모듈의 호출에 의해 종속적으로 실행
  - 재사용 가능
  - 유사 업무들에 대해 부품처럼 공통 사용가능
  - 분담 작성이 가능
  - 메모리를 효율적으로 사용할 수 있음

❑ 기술적 기법 [10/10]

- 모듈(Module) 설계 유의 사항
  - 적절한 크기(size)로 작성
  - 모듈내 응집도(Cohesion)는 되도록 강하게 설계
  - 모듈간 결합도(Coupling)는 약하게 설계
  - 다른 곳에서도 적용이 가능하도록 표준화
  - 가독성 고려
- 모듈(Module) 설계 효과
  - 개발 시 시간과 노동력 절감
  - 비용 절감
  - 신뢰도 향상
  - 복잡성 해결
  - 이해 용이

3. IPT 기법 [12/12]

❑ 관리적 기법

- 선임 프로그래머 팀(Chief Programmer Team)
  - 생산성과 품질 향상을 위한 개발 조직의 구성 방법
  - 선임프로그래머(Chief Programmer) : 팀의 대표자로써 설계, 코딩, 테스팅에도 참여 가능
  - 백업프로그래머(Back-up Programmer) : 선임프로그래머 임무 대행
  - 라이브러리언(Librarian) : 프로그램 리스트, 설계 문서, 테스트 계획 등을 관리하는 프로그래머
- 구조적 검토회(Structured Walk-Through)
  - 오류 및 문제를 조기 발견하기 위한 소프트웨어 관리 기법
- 검증회(Inspection)
  - 오류를 문서화, 오류 관리용 데이터를 사용하여 관리하는 기법
- 라이브러리(Library)
  - 공통적인 사항들을 라이브러리로 만듦
  - 개개인의 업무 중복 회피 및 생산성 향상

4. 객체지향 기법 [1/4]

❑ 개념

- 객체(Object), 메시지(Message), 상속(Inheritance)으로 실세계의 모든 엔티티(Entity)들을 시스템에서 객체로 모델링
- 각 엔티티들은 인스턴스(Instance)로 표현
- 각 인스턴스는 변수와 메소드(method)로 규정
- 메시지에 의해 객체 내 프로시저를 실행

❑ 장/단점

- 대형 프로젝트 개발에 유용
  - 객체의 규모가 크기 때문에 실행 속도가 떨어짐
- 재사용률, 확장성이 높아짐
- 신속한 개발 및 유지보수 향상
- 자연적인 모델링이 가능
  - 단, 객체 설계가 어려움
- 대화식 프로그램 개발에 용이

4. 객체지향 기법 [2/4]

❑ 객체(Object)

- 속성과 이를 처리하기 위한 연산이 포함된 실체
- 하나의 독립된 기능을 수행
- 객체마다 각각의 상태(State)와 행위(Behaviour)를 가짐
  - 일정한 기억 장소를 가지고 있음
  - 상태는 변할 수 있음

❑ 클래스(Class)

- 같은 유형의 객체나 유사한 객체들을 묶어서 공통된 특성이나 특징을 정의한 것
- 객체의 타입(Object Type)을 의미
- 클래스로부터 새로운 객체를 생성하는 행위를 인스턴스화(Instantiation)라 함
- 클래스에는 객체의 특성을 기억할 저장 장소와 함수들을 정의

4. 객체지향 기법 [3/4]

❑ 속성(Attribute)

- 객체들이 가지고 있는 데이터들의 값의 정의
- 상태 정보로 성질, 분류, 식별, 수량 등을 가짐

❑ 메소드(Method)

- 객체에 정의된 연산
- 객체의 상태를 변경하는 맴버 함수를 의미

❑ 메시지(Message)

- 객체간 상호작용을 위한 정보 전달 기법
- 전달 형태: 수신 객체/활동 함수(파라미터)

4. 객체지향 기법 [4/4]

❑ 객체의 특징

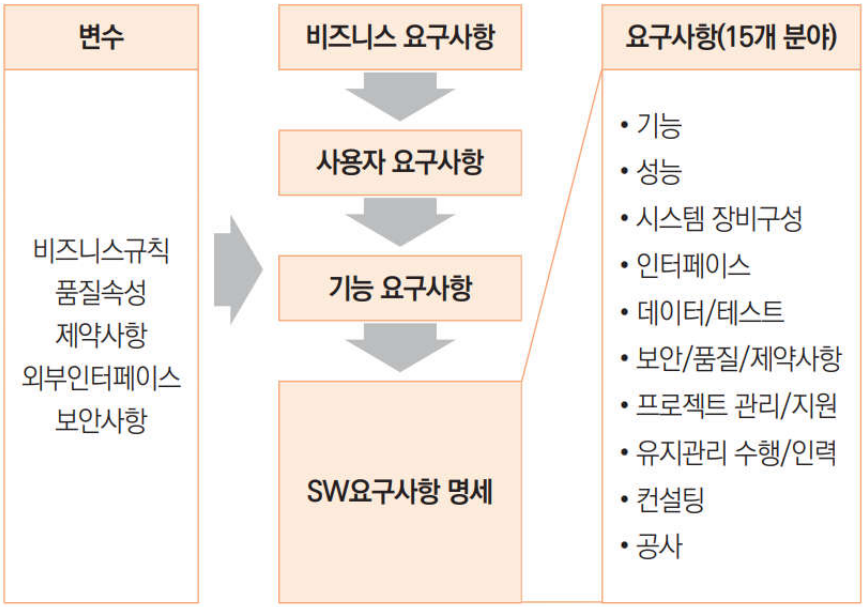
- 주체성(Identity) : 다른 객체와 식별할 수 있는 속성
- 분류성(Classification) : 동일 속성과 행위를 갖는 객체들을 하나의 클래스로 분류
- 다형성(Polymorphism) : 하나의 메시지에 대해 각 클래스가 가지고 있는 고유한 방법으로 응답
- 상속성(Inheritance) : 상위 클래스의 메소드와 속성을 물려 받는 것
- 추상화(Abstraction) :
  - 불필요 부분을 생략
  - 객체의 특징적인 중요사항을 개략화
  - 데이터의 공통된 성질을 추출하여 슈퍼클래스를 선정
- 캡슐화(Encapsulation)
  - 하나의 모듈 내에 결합
  - 객체의 자료가 변조되는 것을 막음
  - 내부적인 구현이나 동작 등의 세부적 사항을 은폐
- 정보 은닉(Information Hidden)
  - 캡슐화된 정보를 외부로부터 감추는 것

5. 요구사항

❑ 정의

- 어떠한 것을 요구하는 권리나 권한을 의미
- 기대치에 대한 포괄적인 뜻을 담고 있음
- **소프트웨어 요구사항(Software Requirement)**
  - 특정 목적을 위해 사용자가 필요로 하는 조건이나 능력을 명시
  - 계획서, 제안 요청서, 명세서 등에 표현되는 **소프트웨어가 갖추어야할 조건이나 능력을 기술**
- 비즈니스 요구사항을 토대로 관련 이해관계자 및 사용자 요구사항, 기능 요구사항을 거쳐 품질 속성, 비즈니스 규칙, 제약사항, 외부인터페이스, 보안사항 등을 고려

〈요구사항 명세 도출 과정〉



5. 요구사항

❑ 조사방법

- 관찰 조사
  - 실제 현업부서를 방문하여 부서의 작업 환경, 현업의 처리 절차, 개선할 사항 등을 관찰
  - 정량적인 정보(빈도, 수량, 비용 등)를 수집하는 방법

- 질문지 조사
  - 체계적으로 설계된 질문지를 이용해 필요한 정보를 수집하는 방법
  - 직접 관찰하거나 면담하기 어려운 부서의 담당자에게서도 손쉽게 정보를 수집할 수 있음

- 면담(인터뷰) 조사
  - **가장 보편적이며 중요한 정보수집 방법**
  - 시스템 분석가와 현업부서 담당자 간의 직접 대화를 통해 현행 시스템의 문제점 및 개선 요구사항 등을 파악할 수 있는 방법

5. 요구사항

요구사항 분류

기능

- 목표 시스템이 수행해야 할 기능(동작)은?
  - ① 기능 요구사항

비기능

- 목표 기능의 원활한 실행 및 성능 확보 방안은?
  - ② 성능 요구사항
  - ③ 시스템 장비구성 요구사항
  - ④ 인터페이스 요구사항
  - ⑤ 데이터 요구사항
  - ⑥ 테스트 요구사항
  - ⑦ 보안 요구사항
  - ⑧ 품질 요구사항
  - ⑨ 제약사항
- 프로젝트의 원활한 수행·관리 방안은?
  - ⑩ 프로젝트 관리 요구사항
  - ⑪ 프로젝트 지원 요구사항

기타

- 시스템 유지관리 단계에서 안정적 운영 방안은?
  - ⑫ 유지관리 수행 요구사항
  - ⑬ 유지관리 인력 요구사항
- 시스템 구축 시 컨설팅 및 공사 관련 사항은?
  - ⑭ 컨설팅 요구사항
  - ⑮ 공사 요구사항

5. 요구사항

❑ 기능

- 기능 요구사항(SFR)
  - 목표 시스템이 반드시 수행해야 하거나 목표 시스템을 이용하여 **사용자가 반드시 수행할 수 있어야 하는 기능(동작)에 대하여 기술**
  - 개별 기능 요구사항은 전체 시스템의 계층적 구조 분석을 통해 세부 기능별 상세 요구사항을 작성해야 하며, 기능 수행을 위한 데이터 요구사항과 연계를 고려하여 기술

5. 요구사항

❑ 비기능 (1/2)

- 성능 요구사항(PER)
  - 목표 시스템의 처리속도 및 시간, 처리량, 동적/정적 용량, 가용성 등 성능에 대한 요구사항을 기술

- 시스템 장비 구성 요구사항(ECR)
  - 기능 수행을 위해 필요한 HW, SW, NW 등의 도입 장비 내역 등 시스템 장비 구성에 대한 기술

- 인터페이스 요구사항(IFR)
  - 목표시스템과 외부를 연결하는 시스템인터페이스와 사용자 인터페이스에 대한 요구사항을 기술
  - 사용자 편의성, 사용자 경험 등의 사용자 중심의 요구사항을 기술

5. 요구사항

❑ 비기능 (2/2)

- 데이터 요구사항(DAR)
  - 목표시스템의 서비스에 필요한 초기자료 구축 및 데이터 변환을 위한 대상, 방법, 보안이 필요한 데이터 등 데이터 구축하기 위해 필요한 요구사항을 기술

- 보안 요구사항(SER)
  - 정보 자산의 기밀성, 무결성을 확보가기 위해 목표 시스템의 데이터 및 기능, 운영 접근을 통제하기 위한 요구사항을 기술

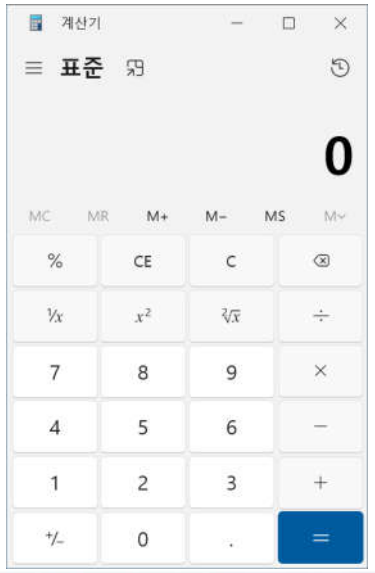
- 품질 요구사항(QUR)
  - 원활한 수행 및 운영을 위해 관리가 필요한 품질 항목, 품질 평가 대상 및 목표에 대한 요구사항을 기술
  - 신뢰성, 사용성, 유지보수성, 이식성, 보안성으로 구분하여 기술

- 제약사항(COR)
  - 사전에 파악된 기술, 표준, 업무, 법제도 등 제약 조건 등을 파악하여 기술

6. 요구사항 작성

❑ 요구사항 작성 예:

- CALC-SFR-001 : 계산기능을 가져야 한다.
- CALC-SFR-002 : 사칙연산 등을 제공해야 한다.
- CALC-SFR-003 : 0~9의 숫자들을 키보드 또는 UI 버튼으로 입력 받아야 한다.
- CALC-SFR-004 : 연속적으로 입력된 숫자요소들은 하나의 숫자를 구성해야 한다.
- CALC-SFR-005 : 사칙연산 명령은 키보드 또는 UI 버튼으로 입력 받아야 한다.
- CALC-SFR-006 : “=“ 버튼 또는 “Enter Key”를 입력 받으면 연산을 수행해야 한다.
- CALC-SFR-007 : 직전 입력숫자에 대한 취소 기능을 제공해야 한다.
- CALC-SFR-008 : 현재 값을 0으로 초기화하는 버튼을 제공해야 한다.
- CALC-SFR-009 : 창의 크기변경이 가능해야 한다.
- CALC-ECR-001 : 윈도우10에서 동작되어야 한다.
- CALC-PER-001 : 사칙연산은 1초 이내에 수행이 완료되어야 한다.



수고하셨습니다.