참고자료

강사: 한동준
handongjoon@gmail.com
dongjoon.han@synetics.kr

설계 및 구현

설계란?

- 정의
  - 설계는 개발될 제품에 대한 의미 있는 공학적 표현
  - 설계는 고객의 요구사항으로 추적 가능해야 하며, 동시에 좋은 설계라는 범주에 들도록 품질에 대해서도 검증되어야 한다
    [IEEE-Std-610]

- 소프트웨어의 설계(design)
  - 본격적인 프로그램의 구현에 들어가기 전에 소프트웨어를 구성하는 뼈대를 정의해 구현의 기반을 만드는 것
  - 종류
    - 상위 설계(High-Level Design) / 아키텍처 설계
    - 하위 설계(Low-Level Design) / 상세 설계

상위 설계와 하위 설계

- 상위 설계(High-Level Design)
  - 의미
    - 아키텍처 설계(Architecture Design), 예비 설계(Preliminary Design)라고 함
    - 시스템 수준에서의 소프트웨어 구성 컴포넌트들 간의 관계로 구성된 시스템의 전체적인 구조
    - 시스템 구조도(Structure Chart), 외부 파일 및 DB 설계도(레코드 레이아웃, ERD), 화면 및 출력물 레이아웃 등이 포함됨

- 하위 설계(Low-Level Design)
  - 의미
    - 모듈 설계(Module Design), 상세 설계 (Detail Design)이라고 함
    - 시스템의 각 구성 요소들의 내부 구조, 동적 행위 등을 결정
    - 각 구성 요소의 제어와 데이터들간의 연결에 대한 구체적인 정의를 하는 것
  - 하위 설계 방법
    - 절차기반(Procedure-Oriented), 자료위주(Data-Oriented), 객체지향(Object-Oriented) 설계 방법

상위 설계와 하위 설계의 구조도

아키텍처 설계 원칙 요건(ISO 26262)

Table 3 — Principles for software architectural design

	Principles	ASIL			
		A	B	C	D
1a	Appropriate hierarchical structure of the software components	++	++	++	++
1b	Restricted size and complexity of software components[a]	++	++	++	++
1c	Restricted size of interfaces[a]	+	+	+	++
1d	Strong cohesion within each software component[b]	+	++	++	++
1e	Loose coupling between software components[b,c]	+	++	++	++
1f	Appropriate scheduling properties	++	++	++	++
1g	Restricted use of interrupts[a,d]	+	+	+	++
1h	Appropriate spatial isolation of the software components	+	+	+	++
1i	Appropriate management of shared resources[e]	++	++	++	++

[a] In principles 1b, 1c, and 1g "restricted" means to minimize in balance with other design considerations.

[b] Principles 1d and 1e can, for example, be achieved by separation of concerns which refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concept, goal, task, or purpose.

[c] Principle 1e addresses the management of dependencies between software components.

[d] Principle 1g can include minimizing the number, or using interrupts with a clear priority, in order to achieve determinism.

[e] Principle 1i applies for shared hardware resources as well as shared software resources in the case of coexistence. Such resource management can be implemented in software or hardware and includes safety mechanisms and/or process measures that prevent conflicting access to shared resources as well as mechanisms that detect and handle conflicting access to shared resources.

** 출처) ISO26262, Part 6*

상세 설계 원칙 요건(ISO 26262)

Table 6 — Design principles for software unit design and implementation

Principle		ASIL			
		A	B	C	D
1a	One entry and one exit point in subprograms and functions[a]	++	++	++	++
1b	No dynamic objects or variables, or else online test during their creation[a]	+	++	++	++
1c	Initialization of variables	++	++	++	++
1d	No multiple use of variable names[a]	++	++	++	++
1e	Avoid global variables or else justify their usage[a]	+	+	++	++
1f	Restricted use of pointers[a]	+	++	++	++
1g	No implicit type conversions[a]	+	++	++	++
1h	No hidden data flow or control flow	+	++	++	++
1i	No unconditional jumps[a]	++	++	++	++
1j	No recursions	+	+	++	++

[a] Principles 1a, 1b, 1d, 1e, 1f, 1g and 1i may not be applicable for graphical modelling notations used in model-based development.

** 출처) ISO26262, Part 6*

SW 상세 설계와 구현

- SW 개발 단계에서 SW 상세 설계와 구현을 별도의 단계로 구분하는 것이 일반적

- 기능 안전 표준에서는 SW 상세 설계와 구현을 하나의 단계로 인식
  - SW 상세 설계의 내용이 소스코드로 그대로 옮겨짐
  - SW 상세 설계가 소스코드의 모든 부분을 설계할 수 없고, 소스코드 구현 과정 자체가 지속적인 상세 설계의 과정임

모듈화

- 모듈의 의미
  - 수행 가능 명령어, 자료구조 또는 다른 모듈을 포함하고 있는 독립 단위
- 특성
  - 이름을 가지며
  - 독립적으로 컴파일 되고
  - 다른 모듈을 사용할 수 있고
  - 다른 프로그램에서 사용될 수 있다
- 모듈의 예
  - 완전한 독립 프로그램, 라이브러리 함수, 그래픽 함수 등
- 모듈의 크기
  - 되도록 쉽게 이해될 수 있도록 가능한 한 작아야 함
  - 너무 작은 모듈로 나눠지지 않도록 함

정보 은닉(Information Hiding)

- 의미
  - 각 모듈 내부 내용에 대해서는 비밀로 묶어두고, 인터페이스를 통해서만 메시지를 전달 할 수 있도록 하는 개념
  - 설계상의 결정 사항들이 각 모듈 안에 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 함

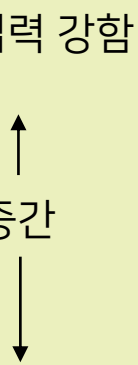
- 장점
  - 모듈의 구현을 독립적으로 맡길 수 있음
  - 설계 과정에서 하나의 모듈이 변경되더라도 설계에 영향을 주지 않음

모듈의 응집력 (1/2)

- 모듈의 응집력이란?
  - 모듈을 이루는 각 요소들의 서로 관련되어 있는 정도
  - 강력한 응집력을 갖는 모듈을 만드는 것이 모듈 설계의 목표

- Myers의 응집력 정도 구분

1. 기능적 응집(Functional cohesion)
   정보적 응집(Informational Cohesion)
2. 교환적 응집(Communication cohesion)
3. 절차적 응집(Procedural cohesion)
4. 시간적 응집(Temporal cohesion)
5. 논리적 응집(Logical cohesion)
6. 우연적 응집(Coincidental cohesion)



모듈의 응집력 (2/2)

응집력의 종류

- 기능적 응집(Functional cohesion)
  - 모듈이 잘 정의된 하나의 기능만을 수행할 때 기능적 응집도가 높아짐
- 교환적 응집(Communication cohesion)
  - 한 모듈 내에 2개 이상의 기능이 존재하고 단계별 순서에 의해서만 수행되는 경우
  - 각 기능은 동일한 입력 자료를 사용하면서도 서로 다른 출력을 생성함
- 절차적 응집(Procedural cohesion)
  - 모듈 안의 작업들이 큰 테두리 안에서 같은 작업에 속하고, 입출력을 공유하지 않지만 순서에 따라 수행될 필요가 있는 경우
- 시간적 응집(Temporal cohesion)
  - 프로그램의 초기화 모듈 같이 한 번만 수행되는 요소들이 포함된 형태
- 논리적 응집(Logical cohesion)
  - 비슷한 성격을 갖거나 특정 형태로 분류되는 처리 요소
- 우연적 응집(Coincidental cohesion)
  - 아무 관련 없는 처리 요소들로 모듈이 형성되는 경우

모듈의 결합도 (1/2)

- 의미
  - 모듈간에 연결되어 상호 의존하는 정도
  - 낮은 결합도를 갖는 모듈(Loosely coupled)을 만드는 것이 모듈 설계의 목표

- 모듈간의 의존도

1. 자료 결합(Data coupling)
2. 구조 결합(Stamp coupling)
3. 제어 결합(Control coupling)
4. 공통 결합(Common coupling)
5. 내용 결합(Content coupling)

결합도 약함

결합도 강함

모듈의 결합도 (2/2)

- 결합도의 종류
  - 자료 결합(data coupling)
    - 모듈 간의 인터페이스가 자료 요소로만 구성된 경우
    - 가장 이상적인 형태의 결합
  - 구조 결합(stamp coupling)
    - 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달되는 경우
  - 제어 결합(control coupling)
    - 한 모듈이 다른 모듈에게 제어 요소(function code, switch, tag 등)를 전달하는 경우
  - 공통 결합(common coupling)
    - 여러 모듈이 공동 자료 영역을 사용하는 경우
  - 내용 결합(content coupling)
    - 한 모듈이 다른 모듈의 일부분을 직접 참조 또는 수정하는 경우

기능 안전 표준의 SW 적용 범위는?

SW 품질

소프트웨어 제품 품질 보증

- Verification
  - 제품이 올바르게 생성되고 있는가?(Are we building the product right?)

『Boehm』

  - 소프트웨어가 정확한 요구사항에 부합하여 구현되었음을 보장하는 활동
  - '요구사항 명세서에 맞게 올바른 방법으로 제품을 만들고 있음'을 보장

- Validation
  - 올바른 제품을 생성하고 있는가?(Are we building the right product?)

『Boehm』

  - 소프트웨어가 고객이 의도한 요구사항에 따라 구현되었음을 보장하는 활동
  - '고객이 의도한 환경이나 사용 목적에 맞게 올바른 제품을 만들고 있음'을 보장

Verification & Validation (테스트) 종류

- 정적(Static)인 방법
  - 소프트웨어를 실행하지 않고 결함을 찾아내는 것
  - 여러 참여자들이 모여 소프트웨어를 검토하여 결함을 찾아내거나 정적 검증 도구 이용
  - 소프트웨어 개발 중에 생성되는 모든 산출물들에 대해서 적용 가능
  - 대표적인 방법:
    - 동료검토(Peer Review)
      - 인스펙션(Inspection)
      - 워크스루(Walk-through)
      - 데스크체크(Desk Check)
    - 도구를 이용한 정적 분석
      - 룰 기반 정적분석(PMD, BugFind 등)

- 동적(Dynamic)인 방법
  - 소프트웨어를 실행하여 결함을 찾아냄
  - 발견된 결함은 디버깅 활동으로 확인하여 수정함
  - 대표적인 방법
    - 테스트
      - 블랙박스 테스트
      - 화이트박스 테스트

소프트웨어 제품 품질 특성

- 소프트웨어 제품 품질이 가져야하는 세부 속성

- ISO 25010 (과거 ISO 9126)에 소프트웨어 품질 특성 정의

기능성	: 요구되는 기능을 만족 시키는 능력
신뢰성	: 규정된 환경에서 결함 없이 의도된 기능 및 작업을 수행하는 능력
사용성	: 사용자가 이해하고 배우기 쉬운 정도
효율성	: 적절한 자원의 사용 및 적정한 반응시간 정도
유지보수성	: 소프트웨어의 수정 및 변경의 용이성
이식성	: 지원하는 다양한 환경에서 운영될 수 있는 능력
상호운영성	: 다른 시스템과의 상호 연동 능력
보안성	: 정보 및 데이터를 보호하는 능력

소프트웨어 제품 품질 특성 - 부특성 (1)

주특성	부특성	내용
기능성	기능 성숙도 Functional Completeness	명시된 요구사항 구현 정도
	기능 정확도 Functional Correctness	정의된 정밀도에 따라 정확하게 결과를 제공하는 정도
	기능 타당성 Functional Appropriateness	사용자의 목적 달성에 소프트웨어가 도움을 주는 정도
효율성	시간 반응성 Time-behavior	기능 수행 시 응답, 처리 시간과 처리율이 요구사항을 충족시키는 정도
	요소 활용 Resource Utilization	기능 수행 시, 사용되는 자원의 유형 및 양이 요구사항을 만족 시키는 정도
	기억 용량 Capacity	제품 혹은 시스템 파라미터(최근 사용자 수, 통신 대역폭, 데이터베이스가 저장할 수 있는 데이터 양 등)의 최대 한계가 요구사항을 만족시키는 정도
상호 운영성	공존성 Co-existence	다른 소프트웨어에 유해한 영향을 주지 않고 환경 및 자원을 공유하면서 요구된 기능을 효과적으로 수행하는 정도
	상호 운영성 Interoperability	둘 혹은 그 이상의 시스템, 제품 혹은 구성요소가 정보를 교환하거나 교환된 정보를 이상 없이 사용할 수 있는 정도

소프트웨어 제품 품질 특성 - 부특성 (2)

주특성	부특성	내용
사용성	타당성 식별력 Appropriateness recognisability	사용자의 요구에 적절한 기능인지 식별할 수 있는 정도
	학습성 Learnability	사용자가 소프트웨어의 사용법을 배워 명시된 목적을 달성할 수 있는 정도
	운용성 Operability	제품 혹은 시스템이 작동 및 제어를 쉽게 할 수 있는 정도
	사용자 오류 보호 User error protection	소프트웨어가 발생한 오류로부터 사용자를 보호하는 정도 (버튼 비활성화, 알림 창 등)
	사용자 인터페이스 미학 User interface aesthetics	사용자 인터페이스가 사용자에게 만족스러운 정도
	접근성 Accessibility	연령과 장애에 관계없이 사용될 수 있는 정도
신뢰성	성숙성 Maturity	소프트웨어 구성요소가 표준적 환경에서 신뢰도 요구를 충족 시키는 정도
	가용성 Availability	사용자가 원하는 시간에 사용 및 접근이 가능한 정도
	결점 완화 Fault tolerance	시스템, 제품 및 구성요소가 하드웨어 혹은 소프트웨어에 결함이 존재하더라도 이를 극복하고 의도한대로 작동해야 함
	회복 가능성 Recoverability	중단 및 실패 발생 시, 제품 혹은 시스템이 데이터를 복구할 수 있는 정도

소프트웨어 제품 품질 특성 - 부특성 (3)

주특성	부특성	내용
보안성	기밀성 Confidentiality	제품 혹은 시스템은 반드시 권한이 있는 데이터에만 접근 가능하도록 해야 함
	무결성 Integrity	시스템, 제품 혹은 구성요소가 컴퓨터 프로그램 혹은 데이터에 대해 무단으로 접근 혹은 변경되는 것을 방지하는 정도
	부인 방지 Non-repudiation	사건 및 행위 후에 부인하지 못하도록 행동 및 사건에 대해 입증되는 정도
	책임성 Accountability	시스템 내의 각 개인을 유일하게 식별하여 언제 어떠한 행동을 하였는지 기록하여 필요 시 그 행위자를 추적할 수 있는 능력
	진본성(인증성) Authenticity	사건 및 행동에 대해 행위자임을 증명할 수 있는 능력
유지 보수성	모듈성 Modularity	최소의 영향을 가진 개별 구성요소로 구성된 정도
	재사용성 Reusability	자신이 하나 이상의 시스템에서 사용될 수 있고, 기타 자신을 구축하 수 있는 정도
	분석성 Analyzability	시스템 변화에 대해 어떠한 영향을 받는지 평가 할 수 있는 보고서를 제공하는 정도
	수정 가능성 Modifiability	제품 혹은 시스템이 장애 없이 효과적이고 효율적으로 수정될 수 있는 정도
	시험 가능성 Testability	제품 사용전, 사용에 필요한 검증 기능 제공 여부
이식성	적용성 Adaptability	제품 혹은 시스템이 다른 하드웨어, 소프트웨어 혹은 기타 사용 환경에 효과적이고 효율적으로 적용될 수 있는 정도
	설치성 Installability	제품 또는 시스템이 성공적으로 설치 및 제거될 수 있는 정도
	대치성 Replaceability	제품이 동일한 환경에서 동일한 목적을 위해 다른 지정 소프트웨어 제품으로 대처될 수 있는 정도

소프트웨어 품질 메트릭

- 정의
  - 사용 목적을 가지고 측정 및 분석하는 메트릭 중, 소프트웨어의 품질을 판단하는데 도움을 주는 지표
  - 활용하지 않을 지표라면, 측정하지 말아야 함

- 측정 및 분석방법
  - 프로세스 및 계획에 따라 측정
  - 모든 메트릭은 정의되어야 함
    - 목적
    - 목표
    - 계산식
    - 측정 대상 및 방법
    - 측정 주기
    - 분석 및 해석 방법
    - 분석 주기

메트릭 정의 예제

지표명	순환 복잡도 위반 건수 (LDRA: Cyclomatic Complexity)	
지표 정의	LDRA 분석 수행을 통해 발견한 함수의 순환 복잡도 10 이상 건수	
산식	A 건	A: 순환 복잡도 10 이상 건수 B: -
수집 단계	구현 단계 이후	
수집 주기	매일(Jenkins 자동 실행)	
수집 대상	LDRA (Jenkins 자동 실행) / *.mts.htm	
검토 주기	매일	
해석	높을수록 소스코드를 이해하기 어려운 코드가 많아, 유지보수 노력과 잠재 결함의 발생 가능성을 증가시킴	
목표 정의	0 건	

제품 품질 메트릭

- 규모 관련
  - 라인 수(LOC: Line of Code)
  - 주석 제외 라인 수
  - 주석 비율
  - 함수 별 라인 수
- 복잡도 관련
  - 순환 복잡도(Cyclomatic Complexity): 함수의 제어 흐름이 얼마나 복잡한지 측정. 분기문 +1
  - 인지 복잡도(Cognitive Complexity): 함수가 얼마나 중첩문이 많은지 측정. Nesting Depth 라고도 표현
- 테스트 관련
  - 기능 커버리지
  - 함수 커버리지
  - 문장/분기/MCDC 커버리지
- 정적분석 관련
  - MISRA 룰 위반 수

제품 품질 메트릭

- 결함 관련
  - 소스코드 라인 대비 결함 비율
  - 유형 별 결함 비율
  - 원인 별 결함 비율
- 의존성 관련
  - 함수 별 호출하는 건수
  - 함수 별 호출되는 건수
  - 변경 영향 비율(Stability): 한 함수가 변경되면 코드의 몇 %가 영향받는가?
  - 상호 참조 비율: 상호 호출하는 파일이 있는가? (변경에 대한 영향도 파악 목적)
- 출시 관련
  - 출시 후 결함 수

제품 품질 메트릭 측정의 자동화

- 제품 품질 메트릭은 최대한 자동화 하는 것이 필요
  - 자동화의 의미: 도구가 지정한 주기 별로 측정하여 보고하는 것 (목표 위반 시 Alert)

- 자동화 가능 메트릭
  - 규모 관련
  - 의존성 관련
  - 복잡도 관련
  - 테스트 관련
  - 정적분석 관련

- 자동화 가능 메트릭의 공통점
  - 자동차 전장에 필수 사용하는 **테스트/정적분석 도구에서 측정 가능**

감사합니다.