**정보처리기사 실기 4장. 개발 환경 구축**

***최종수정일 – 2023/04/13***

**중요사항(암기사항)**은 **볼드체**

**시나공 정보처리기사 실기2023 요약.**

**62. 개발 환경 구축 중요도 - D**

**하드웨어 환경**

클라이언트와 서버로 구분

클라이언트 – 스마트폰, PC

서버 – 웹 서버, 웹 애플리케이션 서버(WAS), 데이터베이스 서버, 파일 서버

**소프트웨어 환경**

시스템 소프트웨어(OS, WAS 서버 프로그램, DB 등), 개발 소프트웨어(요구사항 관리 도구, 설계, 모델링 도구, 구현 도구, 빌드 툴, 테스트 도구, 형상(버전) 관리 도구)

**웹 서버 기능**

HTTP/HTTPS 지원, 통신기록, 정적파일 관리, 대역폭 제한, 가상 호스팅, 인증

**63. 소프트웨어 아키텍처 – 중요도 A**

모듈화, 추상화, 단계적 분해, 정보 은닉을 통해 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 것.

**모듈화**

성능향상, 수정 및 재사용, 유지 관리 등을 위해 시스템의 기능을 모듈 단위로 잘게 나누는 것. 모듈의 크기를 너무 크게 나누면 개수가 적어 통합 비용은 적게 들지만 모듈 하나의 개발 비용이 많이 든다.

**추상화**

과정 추상화 – 전반적인 흐름만 파악할 수 있게 추상화

데이터 추상화 – 데이터의 세부 속성이나 용도를 정의하지 않고 데이터 구조를 대표할 수 있는 표현으로 대처

제어 추상화 – 이벤트 발생의 절차나 방법 정의 x, 대표할 수 있는 표현으로 대처

**단계적 분해(Stepwise Refinement)**

상위의 중요 개념으로부터 하위의 개념으로 구체화시켜 나가는 분할 기법. Niklaus Wirth가 제안한 하향식 설계 전략이다.

**정보 은닉(Information Hiding)**

모듈 내부의 절차와 데이터가 감춰져 다른 모듈에 접근하거나 변경하지 못하도록 하는 기법. 한 모듈이 변경돼도 다른 모듈에 영향을 주지 않아 수정, 유지보수가 용이하다.

**상위 설계와 하위 설계**

**상위 설계** – 아키텍처 설계, 예비 설계라고 한다. 시스템의 전체적 구조에서 구조, DB, 인터페이스 등

**하위 설계** – 모듈 설계, 상세 설계. 시스템의 내부 구조 및 행위 설계이다. 컴포넌트, 자료구조, 알고리즘 등

**소프트웨어 아키텍처의 품질 속성**

평가 요소에는 크게 **시스템 측면, 비즈니스 측면, 아키텍처 측면(개념적 무결성, 완결성)이 있다.**

**아키텍처 설계 과정**

설계 목표 설정 – 시스템 타입 결정 – 아키텍처 패턴 적용 – 서브시스템 구체화 – 검토

**협약(Control)에 의한 설계**

컴포넌트를 설계할 때 클래스에 대한 여러 가정을 공유할 수 있도록 명세한 것이다.

**선행 조건 – 오퍼레이션 호출 전에 참이 될 조건**

**결과 조건 – 오퍼레이션 수행 후 만족되어야 할 조건**

**불변 조건 – 오퍼레이션 실행 중 항상 만족되어야 할 조건**

**아키텍처 패턴**

패턴이라는 말에서 알 수 있듯 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식을 말한다.

**주요 아키텍처 패턴**

* **레이어 패턴**
* **클라이언트 – 서버 패턴**
* **파이프 – 필터 패턴**
* **모델 – 뷰 – 컨트롤러 패턴**

**레이어 패턴(Layers pattern)**

시스템을 계층으로 구분해 구성하는 고전적 패턴, 하위 계층은 상위 계층에 대한 서비스 제공자가 된다. 상위 계층은 하위 계층 클라이언트가 된다. 서로 마주보는 두 계층 사이에서만 상호작용이 이루어진다. OSI 참조 모델이 대표적인 레이어 패턴이다.

**클라이언트 – 서버 패턴(Client – Server Pattern)**

하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴이다. 사용자가 클라이언트로 서버에 요청하면 클라이언트가 응답을 받아 사용자에게 제공하는 방식

**파이프 – 필터 패턴(Pipe – Filter Pattern)**

데이터 스트림 절차의 각 단계를 필터로 캡슐화해 파이프를 통해 전송한다. 쉘에서 파이프 생각하면 된다.

**MVC 패턴**

서브 시스템을 모델, 뷰, 컨트롤러로 구조화하는 패턴이다. 한 모델에 대한 여러 개의 뷰를 만들어 사용할 수 있다.

**기타 패턴**

**마스터 – 슬레이브 패턴:** 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식으로 작업 수행 ex. 장애 허용 시스템, 병렬 컴퓨팅 시스템

**브로커 패턴:** 사용자가 원하는 서비스와 특성을 브로커 컴포넌트에 요청하면 브로커 컴포넌트가 사용자와 상응하는 컴포넌트를 연결시켜준다. Ex. 분산 환경 시스템

**피어 투 피어 패턴:** 피어라 불리는 한 컴포넌트가 동시에 클라이언트와 서버가 될 수 있는 패턴 ex. 파일공유

**이벤트 버스 패턴:** 소스가 특정 채널에 이벤트 메시지를 발행하면 해당 채널을 구독한 리스터들이 메시지를 받아 이벤트를 처리 ex. 알림 서비스

**블랙보드 패턴:** 모든 컴포넌트가 공유 DB와 블랙보드 컴포넌트에 접근할 수 있다.

Ex. 음성인식, 차량 식별, 신호 해석

**인터프리터 패턴:** 코드의 각 라인을 수행하는 방법을 지정하고 기호마다 클래스를 갖도록 구성한다. Ex. 번역기, 컴파일러, 인터프리터

**65. 객체지향(Object – Oriented) -A`**

소프트웨어의 각 요소를 객체로 만들고 객체를 조립해서 소프트웨어를 개발하는 기법.

**구성요소**

* **객체, 클래스, 메시지**

**객체지향의 특징**

* **캡슐화, 상속, 다형성, 연관성**

**객체**

데이터와 메서드(함수)로 구성된 소프트웨어 모듈

**클래스**

공통된 속성과 연산을 갖는 객체의 집합. 클래스의 요소를 편집해 객체를 생성하므로 객체가 가질 수 있는 속성과 연산을 정의하는 틀이다.

**메시지**

객체들 간의 상호작용을 하는데 사용되는 수단. 메시지를 받은 객체는 특정 연산을 수행하거나 연산을 통한 값을 반환한다.

**캡슐화**

객체 내부로의 접근을 제한하기 위해 인터페이스를 제외한 세부 내용을 은닉하는 것

**상속**

상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것이다. 하위 클래스는 물려받은 데이터와 연산을 추가로 정의하지 않아도 즉시 자신의 속성으로 사용할 수 있다. 또한 첨가도 가능하다.

**다형성**

하나의 메시지에 객체들이 각자의 고유한 방식으로 반응할 수 있는 능력. 객체들은 서로 동일한 메서드를 가지고 각기 다르게 반응할 수 있다.

**연관성**

두 개 이상의 객체들이 상호 참조하는 관계를 의미한다.

**연관성의 종류**

Is member of(연관화) – 2개 이상의 객체가 상호 관련되어 있다.

Is instance of(분류화) – 동일한 형의 특성을 갖는 객체들을 모아 구성

Is part of(집단화) – 관련 있는 객체들을 묶어 하나의 상위 객체를 구성

Is a(일반화) – 공통적인 성질들로 추상화한 상위 객체를 구성

Is a(특수화/상세화) – 상위 객체를 구체화하여 하위 객체를 구성

**66. 객체지향 분석 및 설계 -A**

**객체지향 분석(OOA; Object Oriented Analysis)**

사용자의 요구사항과 관련된 **객체, 속성, 연산, 관계** 등을 정의하여 모델링하는 작업. **클래스 식별이 주요 목적이다.**

**객체지향 분석 방법론**

**Rumbaugh(럼바우) 방법:** 객,동,기(객체모델, 동적모델, 기능 모델)로 나누어 분석

**Booch(부치) 방법: 미시적, 거시적 개발 프로세스를** 모두 사용. 클래스와 객체들을 분석 및 식별, 클래스의 속성과 연산을 정의

**Jacobson 방법:** 유스케이스를 강조해 사용

**Coad와 Yourdon 방법:** E-R 다이어그램 사용해 객체 행동 모델링. 객체 식별, 구조 식별, 주체 정의, 속성과 인스턴스 연결 정의, 연산과 메시지 연결 정의

**Wirfs – Brock 방법:** 분석과 설계 간 구분 x , 고객 명세서를 평가해 설계 작업까지 연속적으로 수행

**럼바우 분석법**

모든 소프트웨어 구성 요소를 그래픽으로 모델링

객체 모델링기법(OMT, Object – Modeling Technique)이라고도 한다. 분석활동 순서는 객, 동, 기

**객체 모델링 – 정보 모델링이라고도 함.** 객체 간의 관계를 규정해 **객체 다이어그램**으로 표현

**동적 모델링 – 상태 다이어그램 사용.** 시간의 흐름에 따라 객체 간의 제어 흐름, 상호 작용, 동작 순서 등 동적 행위 표현

**기능 모델링 – 자료흐름도(DFD) 사용.** 다수의 프로세스들 간 자료흐름을 중심으로 처리과정 표현

**객체지향 설계 원칙**

**SOLID(SRP, OCP, LSP, ISP, DIP)**

**67. 모듈 – A**

서브루틴, 서브시스템, 소프트웨어 내의 프로그램, 작업 단위

모듈의 독립성은 결합도와 응집도에 의해 측정된다.

**결합도**

모듈간 상호의존도. 결합도가 높을수록 저품질, 낮을수록 고품질

**강함 -> 약함**

**내용결합도 – 공통 결합도 – 외부 결합도 – 제어 결합도 – 스탬프 결합도 – 자료 결합도**

**내용결합도** – 내부 기능 및 자료를 직접 참조할 수 있는 수준의 결합

**공통 결합도** – 공유되는 공통 데이터를 여러 모듈이 사용할 때의 결합도. 파라미터가 아닌 모듈 밖의 전역변수를 사용해 전역변수를 갱신할 때의 결합도

**외부 결합도** – 타 모듈에서 선언한 변수를 외부의 다른 모듈에서 참조

**제어 결합도** – 타 모듈의 제어 흐름에 간섭하기 위해 제어 신호나 제어 요소를 전달하는 결합도. 하위 모듈에서 상위 모듈로 제어 신호가 이동해 하위 모듈이 상위 모듈에게 명령하는 권리 전도 현상 발생.

**스탬프(검인) 결합도 –** 모듈 간 인터페이스로 배열이나 레코드 등 자료구조가 전달될 때 결합도

**자료 결합도 –** 인터페이스가 자료 요소로만 구성될 때 결합도

**응집도**

모듈의 내부 요소들이 서로 관련된 정도.

응집도는 결합도와 반대로 높을수록 품질이 좋고 낮을수록 품질이 떨어진다.

**강함 -> 약함**

**기능적 응집도 – 순차적 응집도 – 교환적 응집도 – 절차적 응집도 – 시간적 응집도 – 논리적 응집도 – 우연적 응집도**

**기능적 응집도 –** 모든 기능 요소들이 **단일 문제**와 연관되어 수행

**순차적 응집도 –** 메서드 체이닝같이 한 오퍼레이션의 결과가 다름 오퍼레이션의 파라미터로 사용.

**교환(통신)적 응집도 –** 동일 입출력을 사용해 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우 응집도

**절차적 응집도 –** 모듈이다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우 응집도

**시간적 응집도 –** 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우 응집도

**논리적 응집도 –** 유사한 성격을 갖거나 특정 형태로 처리되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도

**우연적 응집도 –** 서로 관련 없는 요소로 이루어진 모듈

**팬인/팬아웃**

팬인은 한 모듈을 제어하는 모듈의 수, 팬아웃은 한 모듈에 의해 제어되는 모듈의 수

팬인이 높다는 것은 재사용 측면에서 설계가 잘 되어있다는 것, 높을수록 좋다. 하지만 팬인이 높으면 단일 장애점이 발생할 수 있어 중점적 관리 및 테스트가 필요하다. 단일 장애점은 한 부분의 장애점으로 전체 시스템을 먹통을 만드는 장애점이다.

**N – S 차트(Nassi – Schneiderman Chart)**

논리의 기술에 중점을 두고 도형을 이용해 표현. 박스 다이어그램, Chapin Chart라고도 한다. GOTO와 화살표를 사용하지 않고 연속, 선택 및 다중 선택, 반복의 3가지 논리 구조로 표현한다.

**68. 단위 모듈 – A**

독립 컴파일이 가능한 단위 기능 모듈. 단위 기능 명세서 작성 – 입출력 구현 – 알고리즘 구현 순으로 이뤄진다.

**IPC(Inter – Process Communication)**

모듈 간 통신을 구현하기 위한 인터페이스 집합. 복수의 프로세스 수행, 이뤄지는 프로세스 간 통신까지 구현 가능

**대표 메서드 5 가지**

* **Shared Memory:** 공유 가능한 메모리 구성, 다수의 프로세스가 통신
* **Socket:** 네트워크 소켓을 이용해 네트워크를 경유하는 프로세스 간 통신
* **Semaphores:** 공유 자원에 대한 접근 제어를 통해 통신
* **Pipes & named Pipes:** 파이프는 하나가 이용 중이면 다른 프로세스가 접근할 수 없다.
* **Message Queueing:** 메시지를 전달해 통신

**단위 모듈 테스트**

유닛 테스트, 시스템 수준 오류는 못 잡는다.

**테스트 케이스**

여기서 테스트 케이스는 사용자 요구사항 준수했나 체크하기 위한 명세서(체크리스트)이다.

**표준 구성 요소**

* **식별자, 테스트 항목, 입출력 명세, 환경 설정, 특수 절차 요구, 의존성 기술**

**69. 공통 모듈 – B**