2장 요구사항 확인

19. 현행 시스템 파악

1. 현행 시스템 파악 절차

시스템 구성 파악 ->시스템 기능 파악 ->시스템 인터페이스 파악 ->아키텍처 구성 파악 ->소프트웨어 구성 파악 ->하드웨어 구성 파악 ->네트워크 구성 파악

2. 시스템 구성 파악 –주요 업무를 담당하는 기간업무 / 이를 지원하는 지원 업무

3. 시스템 기능 파악 – 단위 업무 시스템이 현재 제공하는 기능들을 주요 기능/하부 기능/세부 기능 구분

4. 시스템 인터페이스 파악 – 단위 업무 시스템 간 주고받는 데이터 종류,형식,프로토콜,연계 유형,주기 명시

5. 아키텍처 구성 파악 –기간 업무 수행에 어떠한 기술 요소들이 사용 되는지 최상위 수준에서 계층별로 표현한 아키텍처 구성도로 작성,아키텍처가 단위 업무 시스템별로 다른 경우 핵심 기간 업무 처리 시스템 기준

6. 소프트웨어 구성 파악 –소프트웨어들의 제품명,용도,라이선스 적용 방식,라이선스 수 등을 명시

7. 하드웨어 구성 파악 – 단위 업무 시스템들이 운용되는 서버의 주요 사양,수량,이중화의 적용 여부 명시

서버의 이중화 : 운영 서버의 장애 시 대기 서버로 서비스가 계속 유지될 수 있도록 운용 서버의 자료 변경이 예비 서버에도 동일하게 복제되도록 관리하는 것을 의미

8. 네트워크 구성 파악 – 서버의 위치,서버 간의 네트워크 연결 방식을 네트워크 구성도로 작성

20. 개발 기술 환경 파악

1. 개발 기술 환경의 정의 – OS, DBMS, 미들웨어(Middle Ware)등을 선정할 때 고려해야 할 사항을 기술

2. 운영체제(Operating System)

컴퓨터 시스템의 자원들을 효율적으로 관리

사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어 // iOS, Tizen, UNIX

3. 운영체제 관련 요구사항 식별 시 고려사항

가용성 : 주어진 시점에서 사용자의 요구사항에 따라 운영될 수 있는 능력/성능/기술 지원/주변 기기/구축 비용

4. 데이터베이스 관리 시스템(DBMS)

사용자의 요구에 따라 정보를 생성해 주고 데이터베이스를 관리해 주는 소프트웨어

5. DBMS 관련 요구사항 식별 시 고려사항 – 가용성, 성능,기술 지원,상호 호환성,구축 비용

6. 웹 애플리케이션 서버(WAS; Web Application Server)

사용자의 요구에 따라 변하는 동적인 컨텐츠를 처리하기 위해 사용되는 미들웨어

7. 웹 애플리케이션 서버(WAS) 관련 요구사항 식별 시 고려사항 – 가용성,성능,기술지원,구축비용

8. 오픈 소스 사용에 따른 고려사항 – 라이선스의 종류,사용자수,기술의 지속가능성 등을 고려

21. 요구사항 정의

1. 요구사항의 개념 및 특징

소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 제약 조건

2. 기능 요구사항,비기능 요구사항,사용자 요구사항,시스템 요구사항

3. 요구사항 개발 프로세스 –개발 대상에 대한 요구사항을 체계적으로 도출하고 이를 분석한 후 분석 결과를 명세서(Specification Document)에 정리한 다음이를 확인 및 검증하는 일련의 구조화된 활동

타당성 조사(Feasibility Study) 선행 필요,요구공학(Requirement Engineering)의 한 요소

도출(Elicitation) ->분석(Analysis) ->명세(Specification) ->확인(Validation)

4. 요구사항 도출(Requirement Elicitation, 요구사항 수집)

시스템,사용자,개발자들이 서로 의견을 교환하여 요구사학을 식별하고 이해하는 과정

주요 기법 : 인터뷰,설문,브레인스토밍,워크샵,프로토타이핑,유스케이스

5. 요구사항 분석(Requirement Analysis)

사용자의 요구사항 중 명확하지 않거나 모호하여 이해되지 않는 부분을 발견하고 이를 걸러내기 위한 과정

상충되는 요구사항 발생 시 이를 해결,타당성 조사,비용/일정에 대한 제약 설정,범위 파악,상호작용 방법 이해

6. 요구사항 명세(Requirement Specification)

요구사항을 체계적으로 분석한 후 승인될 수 있도록 문서화하는 것을 의미

기능 요구사항은 완전하고명확하게 기술,비기능 요구사항은 필요한 것만 명확하게 기술

사용자가 이해하기 쉬우며,개발자가 효과적으로 설계할 수 있도록 작성

설계 과정에서 잘못된 부분이 확인될 경우 그 내용을 요구사항 정의서에서 추적할 수 있어야 한다.

7. 요구사항 확인(Requirement Validation, 요구사항 검증)

개발 자원을 요구사항에 할당하기 전에 요구사항 명세서를 검토하는 활동

22. 요구사항 분석기법

2. 요구사항 분류(Requirement Classification)

기능 / 비기능 , 상위요구사항에서 유도 / 이해관계자로부터 직접 발생,개발 제품 / 개발 과정,영향의 범위

소프트웨어 생명 주기 동안에 변경될 가능성이 있는지 여부에 따라 분류

3. 개념 모델링(Conceptual Modeling)

현실 세계의 상황을 단순화하여 개념 적으로 표현한 것을 모델이라고 하며 모델을 만드는 과정을 모델링이라 함

개념 모델은 문제의 주체인 개체들과 그들 간의 관계 및 종속성을 반영한다.

4. 요구사항 할당(Requirement Allocation)

요구사항을 만족시키기 위한 구성 요소를 식별하는 것, 분석 과정에서 추가적인 요구사항이 발견될 수 있음

5. 요구사항 협상(Requirement Negotiation)

요구사항이 서로 충돌될 경우 이를 적절히 해결하는 과정

두명의 이해관계자가 요구하는 요구사항이 충돌,요구사항과 자원의 충돌,기능/비기능 요구사항의 충돌

* 적절한 기준점을 찾아 합의.우선 순위 부여 시 문제 해결에 도움

6. 정형 분석(Formal Analysis)

구문(Syntax)과 의미(Semantics)를 갖는 정형화된 언어를 이용해 요구사항을 수학적 기호로 표현 후 분석

23. 요구사항 확인 기법

요구사항 개발 과정을 거쳐 문서화된 요구사항 관련 내용을 확인하고 검증하는 방법,자원 배정 전 수행 필요

2. 요구사항 검토(Requirement Reviews)

문서화된 요구사항을 훑어보면서 확인하는 것,가장 일반적인 요구사항 검증 방법

명확하지 않은 내용,잘못된 가정,정해놓은 기준을 벗어나지 않는지 등을 찾아냄

3. 프로토타이밍(Prototyping)

초기 도출된 요구사항을 토대로 프로토타입을 만든 후대상 시스템의 개발 진행 동안 도출되는 요구사항을 반영하면서 지속적으로 프로토타입을 재 작성하는 과정

개발 대상 시스템 또는 그 일부분을 개략적으로 만든 원형을 프로토타입이라고 한다.

수행하면서 새로운 요구사항 도출 가능,소프트웨어 요구사항에 대한 개발자의 해석이 맞는지 확인하기위한 수단

4. 모델 검증(Model Verification)

요구사항 분석 단계에서 개발된 모델이 요구사항을 충족시키는지 검증하는 것

5. 인수 테스트(Acceptance Tests)

사용자가 실제로 사용될 환경에서 요구사항들이 모두 충족되는지 사용자 입장에서 확인하는 과정

종류 : 사용자 인수 테스트,운영상의 인수 테스트,계약 인수 테스트,알파 검사,베타 검사,규정 인수 테스트

24. UML(Unified Modeling Language)

시스템 개발 과정에서 개발자와 고객 또는 상호 간의 원활한 의사소통을 위해 표준화한 객체지향 모델링 언어

시스템의 구조를 표현하는 6개의 구조 다이어그램 / 시스템의 동작을 표현하는 7개의 행위 다이어그램

2. 사물(Things)

모델을 구성하는 가장 중요한 기본 요소,다이어그램 안에서 관계가 형성될 수 있는 대상들을 말한다.

구조 사물 : 시스템의 개념적,물리적 요소를 표현,클래스,유스케이스(Use Case),컴포넌트(Component),노드 등

행동 사물 : 시간과 공간에 따른 요소들의 행위를 표현, 상호작용(Interaction), 상태 머신(State Machine) 등

그룹 사물(Grouping Things) : 요소들을 그룹으로 묶어서 표현,패키지(Package)

주해 사물(Annotation Things) : 부가적인 설명이나 제약조건 표현,노트(Note)

3. 관계(Relationships)

연관(Association)관계

2개 이상의 사물이 서로 관련되어 있음을 표현한다.실선으로 연결하여 표현,방향성은 화살표로 표현

양방향 관계의 경우 화살표를 생략하고 실선으로만 연결,객체의 개수를 의미하는 다중도를 선위에 표기

집합(Aggregation) 관계

하나의 사물이 다른 사물에 포함되어 있는 관계를 표현

포함하는 쪽(전체), 포함되는 쪽(부분)은 서로 독립적,부분에서 전체로 속이 빈 마름모를 연결하여 표현함

포함(Composition) 관계

집합 관계의 특수한 형태,포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 표현

전체와 부분은 서로 독립x, 생명주기를 함께함.부분에서 전체로 속이 채워진 마름모를 연결하여 표현

일반화(Generalization) 관계

하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현한다.

자식 사물에서 부모 사물 쪽으로 속이 빈 화살표를 연결하여 표현

의존(Dependency) 관계

사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계

영향을 주는 사물이 받는 사물 쪽으로 점선 화살표를 연결하여 표현

실체화(Realization) 관계

실체화 사물이 할 수 있거나 해야 하는 기능(행위,인터페이스)으로 서로를 그룹화 할 수 있는 관계를 표현

사물에서 기능 쪽으로 속이 빈 점선 화살표를 연결하여 표현한다.

4. 다이어그램(Diagram)

사물과 관계를 도형으로 표현한 것

정적 모델링 – 구조적 다이어그램,동적 모델링 – 행위 다이어그램

구조적(Structural) 다이어그램의 종류

클래스 다이어그램 : 클래스와 클래스의 속성,클래스 사이의 관계를 표현,시스템 구조파악 및 문제점 도출

객체(Object) 다이어그램 : 클래스에 속한 사물(객체)들,인스턴스를 특정 시점의 객체와 객체 사이의 관계로 표현

컴포넌트(Component) 다이어그램 : 컴포넌트 간의 관계나 인터페이스 표현,구현 단계에서 사용되는 다이어그램

배치(Deployment) : 결과물,프로세스,컴포넌트 등의 물리적 요소들의 위치를 표현,노드와 의사소통 경로로 표현

복합체 구조(Composite Structure) 다이어그램 : 클래스/컴포넌트가 복합 구조를 갖는 경우 내부 구조 표현

패키지(Package) : 유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한다.

행위(Behavioral) 다이어그램의 종류

유스케이스(Use Case) 다이어그램 : 사용자의 요구 분석,기능 모델링 작업에 사용, Actor와 Use Case로 구성

시퀀스(Sequence) 다이어그램 : 상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현

커뮤니케이션 : 동작에 참여하는 객체들이 주고받는 메시지를 표현,메시지 외에도 객체들 간의 연관까지 표현

상태 :하나의 객체가 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 변화하는 상태를 표현

활동 다이어그램 : 시스템의 수행 기능,객체의 처리 로직또는 조건에 따른 처리의 흐름을 순서에 따라 표현

상호작용 개요(Interaction Overview) 다이어그램 : 상호작용 다이어그램 간의 제어 흐름 표현

타이밍(Timing) 다이어그램 : 객체 상태 변화와 시간 제약을 명시적으로 표현

25. 유스케이스(Use Case) 다이어그램

1. 기능 모델링의 개념

사용자의 요구사항을 분석하여 개발될 시스템이 갖춰야 할 기능들을 정리한 후 내용 공유를 위해 표현하는 것

개발될 시스템의 전반적인 형태를 기능에 초점을 맞춰 표현, Use Case/Activity 다이어그램

2. 유스케이스(Use Case) 다이어그램의 개념

개발될 시스템과 관련된 외부 요소들이 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점에서 표현

3. 유스케이스(Use Case)다이어그램의 구성 요소

시스템(System)/시스템 범위(System Scope)

시스템 내부의 기능들을 외부 시스템과의 구분을 위해 유스케이스들을 사각형으로 묶어 시스템의 범위를 표현

액터(Actor)

시스템과 상호작용을 하는 모든 외부 요소, 사람이나 외부 시스템을 의미, 역할, 구체적x, 주액터/부액터

주액터(Primary Actor)

시스템을 사용함으로써 이득을 얻는 대상, 주로 사람. 사람형태를 간략화하여 표현, 시스템의 왼쪽에 배치

부액터(Secondary Actor)

주액터의 목적 달성을 위해 시스템에 서비스를 제공하는 외부 시스템, 조직이나 기관. 오른쪽에 배치

시스템명을 사각형으로 묶은 후 상단에 <<Actor>>라고 표기

유스케이스(Use Case)

시스템이 액터에게 제공하는 서비스 또는 기능을 표현, 타원, 안쪽이나 아래쪽에 이름을 기술

더 이상 분할되지 않는 기능의 단위, 기능적 요소 중심, 액터에 의해 수행되며 액터가 관찰할 수 있는 결과 산출

관계(Relationship)

액터-유스케이스, 유스케이스-유스케이스 사이에서 나타남. 포함 관계, 확장 관계, 일반화 관계

포함(Include) 관계

두개 이상의 유스케이스에 공통적으로 적용되는 기능을 별도로 분리하여 새로운 유스케이스로 만든 경우

원래의 유스케이스와 새롭게 분리된 유스케이스와의 관계, 원래 --> 신규 점선화살표 연결

확장(Extend) 관계

유스케이스의 기능이 확장될 때 원래의 유스케이스와 확장된 유스케이스와의 관계

확장 유스케이스 쪽에서 원래의 유스케이스 쪽으로 점선 화살표를 연결한 후 화살표 위에 <<extends>> 표기

일반화(Generalization) 관계

유사 액터,유스케이스를 하나의 그룹으로 묶고 싶을 때 일반적인 액터, 유스케이스를 만들고 연결하여 표현하는 관계 / 하위 액터, 유스케이스가 상위 액터, 유스케이스에게 역할과 기능을 상속(Inheritance)받는 관계

4. 유스케이스 명세서(기술서)

유스케이스 안에서의 액터와 시스템 간의 상호 작용 과정을 글로 자세히 표현한 것

각각의 유스케이스 개별 작성, 명세서에 작성된 사건의 흐름을 참고하여 활동(Activity) 다이어그램 작성

26. 활동(Activity) 다이어그램

1. 활동 다이어그램의 개념

자료 흐름도와 유사한 것, 사용자의 관점에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현

2. 활동(Activity) 다이어그램의 구성 요소 (액션, 액티비티, 노드, 스윔레인)

액션(Action) – 더 이상 분해할 수 없는 단일 작업

액티비티(Activity) – 몇 개의 액션으로 분리될 수 있는 작업 둥근 사각형으로 표현, 안에 명칭 기술

제어 흐름 – 실행의 흐름을 표현, 화살표로 표현

시작 노드 – 액션, 액티비티의 시작, 하나의 다이어그램 안에는 하나의 시작점만 존재, 검은색 원으로 표현

종료 노드 – 액티비티 내 모든 흐름 종료, 여러 개 있을 수 있지만 하나만 표현, 검은색 원을 포함한 원 표현

조건(판단 노드) – 조건에 따라 제어의 흐름 분리를 표현, 마름모, 들어오는 제어흐름 한 개, 나가는것 여러 개

병합 노드 – 여러 경로의 흐름이 하나로 합쳐짐, 마름모, 들어오는 제어흐름 여러 개, 나가는 것 하나

포크(Fork) 노드 – 액티비티의 흐름이 분리되어 수행됨을 표현, 굵은 가로선, 들어오는것 1 나가는 것 여러 개

조인 노드 – 분리되어 수행되던 액티비티의 흐름이 다시 합쳐짐, 굵은 가로선, 들어오는것 다수 나가는 것 하나

스윔레인(Swin Lane) – 액티비티 수행을 담당하는 주체를 구분, 가로 또는 세로 실선을 그어 구분

27. 클래스(Class) 다이어그램

1. 정적 모델링의 개념

사용자가 요구한 기능을 구현하는데 필요한 자료들의 논리적인 구조를 표현한 것

2. 클래스 다이어그램의 개념

시스템을 구성하는 클래스, 클래스의 특성인 속성과 오퍼레이션, <-제약조건, 클래스 사이의 관계 표현

시스템 구성요소를 문서화, 시스템을 모델링하는 데 자주 사용, 클래스/제약조건/관계 등으로 구성

3. 클래스(Class)

객체들이 갖는 속성과 오퍼레이션(동작)을 표현

3개의 구획(Compartment)으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기

속성이나 오퍼레이션이 생략된 경우에는 구획선을 그리지 않아도 됨, 이름은 반드시 명시

속성(Attribute) – 클래스의 상태나 정보를 표현

접근제어자 – 속성과 오퍼레이션을 외부에 노출시키는 정도를 제어(public +, private -, protected #, package ~)

속성명 – 속성의 이름으로, 사용자가 임의로 작성

자료형 – 프로그램마다 제공하는 자료형을 사용할 수 있다

다중성 – 동일한 속성명으로 여러 개의 속성 값을 가질 수 있는 것으로, 배열과 같은 의미

초기값 – 데이터를 입력하지 않았을 때 기본적으로 입력되는 값을 지정한다.

오퍼레이션(Operation, 연산) – 클래스가 수행할 수 있는 동작, 함수(메소드, Method) / [접근제어자]:반환자료형

4. 제약조건 – 속성에 입력될 값에 대한 제약조건, 주석 도형 안에 제약조건 기록, 속성이나 함수에 점선 연결

5. 관계(Relationships)

클래스와 클래스 사이의 연관성을 표현. 관계에 참여하는 객체의 수(다중도)를 연관 관계 선 위에 표기

연관(Association)관계 – 두 클래스 간의 관계를 명확하게 표현하기위해 관계 표현 실선의 중간 지점에 이름 표기

집합(Aggregation)관계 – 집합 관계에 있는 클래스의 객체 변수를 매개변수로 사용할 수 있다

포함(Composition)관계 – 포함 관계에 있는 클래스를 통해 생성된 객체 변수를 이용하여 새로운 객체 변수 생성

일반화(Generalization)관계 – 일반적 개념 상위클래스 구체적 개념 하위 클래스라고 부름

의존(Dependency)관계 – 영향을 주는 클래스의 특정 오퍼레이션이 수행될 때만 영향을 받는 클래스가 사용됨

28. 시퀀스(Sequence) 다이어그램

1. 동적 모델링의 개념

시스템 내부 구성 요소들의 상태가 시간의 흐름에 따라 변화하는 과정과 그 과정에서 발생하는 상호작용을 표현

동작이라는 관점에서 표현, 시스템이 실행될 때 구성 요소들 간 메시지 호출, 오퍼레이션을 통한 상호작용 초점

2. 시퀀스(Sequence) 다이어그램의 개념

시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호 작용하는 과정을 액터, 객체 메시지 등의 요소를 사용하여 그림으로 표현

3. 시퀀스 다이어그램의 구성 요소

액터(Actor) – 시스템으로부터 서비스를 요청하는 외부 요소, 사람이나 외부 시스템을 의미

객체(Object) – 메시지를 주고받는 주체, 콜론을 기준으로 앞쪽에는 객체명, 뒤쪽에는 클래스명 기술

라이프라인(Lifeline) – 객체가 메모리에 존재하는 기간, 객체 아래에 점선을 그어 표현, 소멸 표시 기간까지 존재

활성 상자(Activation Box) – 객체가 메시지를 주고받으며 구동되고 있음을 표현, 직사각형 형태로 표현

메시지(Message) – 객체가 상호작용을 위해 주고받는 메시지, 번호를 표기하여 전달 순서 표현 가능

객체 소멸 – X로 표현, 해당 객체는 더 이상 메모리에 존재하지 않음을 의미

프레임(Frame) – 다이어그램의 전체 또는 일부를 묶어 표현, 프레임의 왼쪽 위에 종류와 제목 표기

29. 커뮤니케이션(Communication) 다이어그램

1. 커뮤니케이션 다이어그램의 개념

시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호 작용하는 과정을 그림으로 표현한 것

동작에 참여하는 개체들이 주고받는 메시지를 표현, 객체들 간의 관계까지 표현

2. 커뮤니케이션 다이어그램의 구성 요소 - 액터(Actor), 객체(Object), 링크(Link), 메시지(Message)

30. 상태(State) 다이어그램

1. 상태(State) 다이어그램의 개념

객체들 사이에 발생하는 이벤트에 의한 객체들의 상태 변화를 그림으로 표현한 것

객체 자신이 속한 클래스의 상태 변화나 객체가 다른 객체와 상호작용하는 과정에서의 상태변화를 의미

객체의 상태 = 객체가 갖는 속성 값의 변화를 의미

2. 상태 다이어그램의 구성 요소

상태(State) – 객체의 상태를 표현, 둥근 사각형 안에 기술

시작 상태 – 상태의 시작을 표현, 속이 채워진 원 / 종료 상태 – 상태의 종료를 표현, 이중 원

상태 전환 – 상태 사이의 흐름, 변화를 화살표로 표현한다.

이벤트(Event) – 상태에 변화를 주는 현상, 이벤트에는 조건, 외부신호, 시간의 흐름 등이 있다.

프레임(Frame) – 상태 다이어그램의 범위를 표현