Char2. 요구사항 확인

현행 시스템 파악 절차(for 새로 개발하려는 시스템의 개발 범의를 명확히 설정하기 위해)

1단계	시스템 구성 파악	조직 내 모든 정보시스템 현황을 파악할 수 있도록 단위업무정보시스템 명시
	시스템 기능 파악	주요기능 하부기능 세부기능으로 구분하여 계층형으로 표시
	시스템 인터페이스 파악	단위업무시스템 간에 주고받는 데이터의 종류, 형식, 프로토콜, 주기 등을 명시
2단계	아키텍처 구성 파악	기간업무수행에 대한 어떤 기술요소가 사용되었는지 최상위수준에서 계층별로 표현한 아키텍처 구성도 작성 (시스템별 다른 경우 기간업무처리시스템을 기준으로)
	소프트웨어 구성 파악	상용소프트웨어인 경우 라이선스 적용방식 기준과 보유한 라이선스 파악 중요
3단계	하드웨어 구성 파악	서버의 이중화는 기간업무의 서비스기간, 장애대응정책에 따라 필요여부 결정
	네트워크 구성 파악	네트워크 구성도를 통해 서버간의 물리적 위치관계와 보안 취약성 분석 가능

개발 기술 환경 정의

운영체제 OS Operating System

컴퓨터 시스템의 자원들을 효율적으로 관리하며 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 있도록 환경을 제공하는 소프트웨어

컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공

- 컴퓨터 운영체제 종류: Windows UNIX Linux MacOS
- 모바일 운영체제 종류: iOS Android Tizen
- 요구사항 식별 시 고려사항: 가용성, 성능, 기술 지원, 주변 기기, 구축 비용

데이터베이스 관리 시스템 DBMS DataBase Management System

사용자와 데이터베이스 사이에서 사용자 요구에 따라 정보생성과 데이터베이스 관리해 주는 소프트웨어

기존 파일 시스템의 **데이터 종속성과 중복성 문제를 해결**하기 위해 제안된 시스템

- 종류: Oracle, IBM DB2, MySQL, SQLite, MongoDB, Redis, Microsoft SQL Server
- 요구사항 식별 시 고려사항: 가용성, 성능, 기술 지원, 상호 호환성, 구축 비용

웹 애플리케이션 서버 WAS Web Application Server

정적인 콘텐츠 처리를 하는 웹 서버와 달리 **사용자의 요구에 따라 변하는 동적 콘텐츠를 처리하기 위한 미들웨어**

- 종류: Tomcat, GlassFish, JBoss, Jetty, JEUS, WebLogic, WebSqhere
- 요구사항 식별 시 고려사항: 가용성, 성능, 기술 지원, 구축 비용

- 요구사항은 기술하는 내용에 따라 기능 **요구사항 Functional requirement**와 **비기능 요구사항 Non-functional requirement**로 나뉜다.
- 요구사항은 기술관점과 대상범위에 따라 **시스템 요구사항 System requirement**와 **사용자 요구사항 User requirement**로 나뉜다.

요구사항 개발 프로세스

요구사항 개발 프로세스가 진행되기 전에 타당성 조사 Feasibility Study가 선행되어야 한다.

요구사항 개발은 요구공학 Requirement Engineering의 한 요소이다.(요구공학⊃요구사항 관리⊃요구사항 개발)

요구사항 도출 Rrquirement Elicitation

소프트웨어 개발 생명 주기SDLC Software Development Life Cycle동안 지속적으로 반복

주요기법: 인터뷰 설문 워크샵 브레인스토밍 프로토타이핑 유스케이스

↓

요구사항 분석 Rrquirement Analysis

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

- 요구사항 분류 Requirement Classification
- 개념 모델링 Conceptual Modeling (주요표기는 UML Unified Modeling Language을 사용)
- 요구사항 할당 Requirement Allocation (요구사항을 만족시키기 위한 구성요소를 식별하는 것)
- 요구사항 협상 Requirement Negotitation (요구사항이 서로 충돌할 경우 이를 적절히 해결하는 과정)

  우선순위를 부여하면 문제해결에 도움
- 정형분석 Formal Analysis(구문 Syntax과 의미Semantics를 갖는 정형화된 언어를 이용하여 요구사항을 수학적 기호로 표현한 후 이를 분석하는 요구사항 분석의 마지막 단계)

↓

요구사항 명세 Rrquirement Specification

요구사항을 체계적으로 분석한 후 승인될 수 있도록 문서화하는 것

기능 요구사항은 빠짐없이 완전 명확하게 기술, 비기능 요구사항은 필요한 것만 명확하게 기술

↓

요구사항 확인 Rrquirement Validation

개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동

일반적으로 요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 형상 관리를 수행

요구사항검토(Requirement Reviews) 프로토타이핑(Prototyping) 모델검증(Model Verification) 인수테스트(Acceptance Tests)

UML Unified Modeling Language

시스템 분석, 설계, 구현 등 시스템 개발과정에서 시스템 개발자와 고객 또는 개발자 상호 간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 모델링 언어

- 구성요소

1. 사물 Things: 다이어그램 안에서 관계가 형성될 수 있는 대상들

사물	내용	예시
구조사물 Structural Things	시스템의 개념적, 물리적 요소를 표현	클래스 유스케이스 컴포넌트 노드
행동사물 Behavioral Things	시간과 공간에 따른 요소들의 행위를 표현	상호작용 Interaction 상태 머신 State Machine
그룹사물 Grouping Things	요소들을 그룹으로 묶어서 표현	패키지 Package
주해사물 Annotation Things	부가적인 설명이나 제약조건 등을 표현	노트 Note

2. 관계 Relationships: 사물과 사물 사이의 연관성을 표현하는 것

관계	내용
연관관계 Association	2개 이상의 사물이 서로 관련되어 있음을 표현
집합관계 Aggregation	하나의 사물이 다른 사물에 포함되어 있는 관계를 표현
포함관계 Composition	집합관계의 특수한 형태로 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계
일반화 Generalization	하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지 표현(커피, 아메리카노)
의존관계 Dependency	연관관계와 같이 사물 사이에 서로 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계
실체화 관계 Realization	사물이 할 수 있으나 해야 하는 기능으로 서로를 그룹화 할 수 있는 관계를 표현

3. 다이어그램 Diagram: 사물과 관계를 도형으로 표현한 것

구조적 다이어그램 (정적모델링에 사용)	**클래스 다이어그램 Class**	클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현
	객체 다이어그램 Object	클래스에 속한 사물들, 즉 인스턴스를 특정시점의 객체 사이의 관계로 표현
	컴포넌트 다이어그램 Component	실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현 구현단계에서 사용되는 다이어그램
	배치 다이어그램 Deployment	결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현 노드와 의사소통(통신) 경로로 표현 구현단계에서 사용되는 다이어그램
	복합체 구조 다이어그램 Composite Structure Diagram	클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현
	패키지 다이어그램 Package	유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계
행위 다이어그램 (동적모델링에 사용)	**유스케이스 다이어그램 Use Case Diagram**	사용자 요구를 분석하는 것으로 기능 모델링 작업에 사용 사용자Actor와 사용사례UseCase로 구성
	시퀀스 다이어그램 Sequence	상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현
	커뮤니케이션 다이어그램 Communication Diagram	시퀀스 다이어그램과 같이 동작에 참여하는 객체들이 주고받는 메시지를 표현하는데 메시지 뿐만 아니라 객체들 간의 연관까지 표현
	상태 다이어그램 State	하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호작용에 따라 상태가 어떻게 변화하는지를 표현
	활동 다이어그램 Activity	시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현
	상호작용 개요 다이어그램 Interaction Overview Diagram	상호작용 다이어그램 간의 제어 흐름을 표현
	타이밍 다이어그램 Timing	객체 상태 변화와 시간 제약을 명시적으로 표현

기능 모델링

사용자의 요구사항을 분석하여 개발될 시스템이 갖춰야 할 기능들을 정리한 후 사용자와 함께 정리된 내용을 공유하기 위해 표현하는 것을 말한다.

UML의 기능 다이어그램에는 **유스케이스 다이어그램**과 **활동 다이어그램**이 있다.

유스케이스 다이어그램 Use Case Diagram

개발될 시스템과 관련된 외부 요소들, 즉 사용자와 다른 외부 시스템들이 개발될 시스템을 이용해 수행할 수 있는 기능을 사용자의 관점에서 표현한 것

시스템 범위, 액터, 유스케이스, 관계로 구성

시스템 범위 System Scope	시스템 내부에서 수행되는 기능들을 외부 시스템과 구분하기 위해 시스템 내부의 유스케이스들을 사각형으로 묶어 시스템의 범위를 표현 사각형 안쪽 상단에 시스템 명칭 기술
액터 Actor	시스템과 상호작용을 하는 모든 외부 요소로 사람이나 외부 시스템을 의미 시스템에 대해 수행할 수 있는 역할을 의미 액터 이름이 구체적이면 안됨 - 주액터 Primary Actor 시스템을 사용함으로써 이득을 얻는 대상으로 주로 사람이 해당 사람 형태를 간략화하여 표현하며 주로 시스템 왼쪽에 배치 - 부액터 Secondary Actor 주액터의 목적달성을 위해 시스템에 서비스를 제공하는 외부 시스템으로 조직이나 기관이 될 수 있음 주로 시스템의 오른쪽에 배치하며 시스템명을 사각으로 묶은 후 상단에 **<<Actor>>** 라고 표기
유스케이스 Use Case	사용자가 보는 관점에서 시스템이 액터에게 제공하는 서비스 또는 기능을 표현 타원으로 표현하며 타원 안쪽이나 아래쪽에 유스케이스 이름을 기술
관계 Relationship	액터와 유스케이스, 유스케이스와 유스케이스사이에서 관계가 나타날 수 있다. - 포함관계 원래 유스케이스에서 새롭게 만든 유스케이스로 점선화살표연결 후 화살표 위에 <<include>> 두 개 이상의 유스케이스에 공통적으로 적용되는 기능을 별도 분리하여 새로운 유스케이스로 만든 경우 원래의 유스케이스와 새롭게 분리된 유스케이스와의 관계를 의미 - 확장관계 확장될 유스케이스에서 원래 유스케이스로 점선화살표 연결 후 화살표 위에 <<extends>> 유스케이스가 특정 조건에 부합되어 유스케이스의 기능을 확장될 때 원래의 유스케이스와 확장된 우스케이스와의 관계를 의미 - 일반화관계 Generalization 상속 받는 관계 하위 액터나 유스케이스에서 상위 액터나 유스케이스 쪽으로 속이 빈 삼각형 화살표를 실선으로 연결

유스케이스 명세서

유스케이스 안에서의 액터와 시스템 간의 상호작용과정을 글로 자세히 표현한 것

각 **유스케이스 명세서**를 참고하여 **활동 다이어그램** 작성

활동 다이어그램 Activity Diagram

자료 흐름도와 유사한 것으로 사용자 관점에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현 한 것

하나의 유스케이스 안에서 혹은 유스케이스 사이에 발생하는 복잡한 처리의 흐름을 명확하게 표현 가능

액션 action **액티비티 activity**	**액션은 더 이상 분해 할 수 없는 단일 작업** **액티비티는 몇 개의 액션으로 분리될 수 있는 작업** 테두리가 있는 둥근 사각형으로 표현하고 그 안에 명칭 기술
제어 흐름	실행의 흐름을 표현 화살표로 표현
노드 Node	**시작노드** - 액션이나 액티비티 시작됨 의미 - 하나의 다이어그램 안에는 하나의 시작점만 존재 - **검은색 원**으로 표현 **종료노드** - 액티비티 안의 모든 흐름이 종료됨을 의미 - 하나의 다이어그램 안에 여러 개의 종료 노드가 있을 수 있으나 일반적으로 하나만 표현 - **검은색 원을 포함한 원**으로 표현 **조건(판단)노드** - 조건에 따라 제어의 흐름이 분리됨을 표현 - **마름모로 표현. 들어오는 제어흐름은 하나이고 나가는 제어흐름은 여러 개** **병합노드** - 여러 경로의 흐름이 나라로 합쳐짐을 표현 - **마름모로 표현. 들어오는 제어흐름은 여러 개, 나가는 제어흐름은 한 개** **포크Fork노드** - 액티비티의 흐름이 분리되어 수행됨을 표현 - **굵은 가로선을 표현, 들어오는 제어흐름은 한 개, 나가는 제어흐름은 여러 개** **조인Join노드** - 분리되어 수행되던 액티비티의 흐름이 다시 합쳐짐을 표현 - **굵은 가로선을 표현, 들어오는 제어흐름은 여러 개, 나가는 제어흐름은 한 개**
스웸레인 **Swim Lane**	액티비티 수행을 담당하는 주체를 구분 가로 또는 세로 실선을 그어 구분

클래스 다이어그램 Class Diagram

클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다.

UML을 이용한 정적 모델링의 대표적인 것이 클래스 다이어그램이다.

클래스 Class	각각의 객체들이 갖는 속성과 오퍼레이션(동작)을 표현 반점으로 3 개의 구획(Compartment)으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기함 - 속성(Attribute) : 클래스의 상태나 정보를 표현 [접근 제어자]속성명:자료형[다중성][=초기값] ([]는 생략가능) - 오퍼레이션(Operation, 연산) : 클래스가 수행할 수 있는 동작으로, 함수(Method)라고도 함 [접근 제어자]오퍼레이션명(매개변수 1:자료형 1, 매개변수 2:자료형 2,...):반환자료형
제약조건 operation	속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있다면 이를 적음
관계 Relationships	클래스와 클래스 사이의 연관성을 표현 연관관계, 집합관계, 포함관계, 일반화 관계, 의존관계

- 다중도 Multiplication

다중도	의미
1	1개의 객체와 연관
n	n개의 객체와 연관
0..1	연관된 객체가 없거나 1개 존재
0..* 또는 *	연관된 객체가 없거나 다수일 수도 있다.
1..*	연관된 객체가 적어도 1개 이상
n..*	연관된 객체가 적어도 n개 이상
n..m	연관된 객체가 최소 n개에서 최대m개

- 접근제어자

접근제어자	표현법	내용
Public	+	어떤 클래스에서라도 접근이 가능
Private	-	해당 클래스 내부에서만 접근 가능
Protected	#	동일 패키지 내의 클래스 또는 해당 클래스를 상속받은 외부 패키지의 클래스에서 접근 가능
Package	~	동일 패키지 내부에 있는 클래스에서만 접근 가능

동적 모델링

시스템의 내부 구성 요소들의 상태가 시간의 흐름에 따라 변화하는 과정과 변화하는 과정에서 발생하는 상호작용을 표현한 것

UML의 **시퀀스 다이어그램, 커뮤니케이션 다이어그램, 상태 다이어그램**이 포함

시퀀스 다이어그램 Sequence diagram

시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호작용하는 과정을 액터, 객체, 메시지 등의 요소를 사용하여 그림으로 표현한 것

주로 기능 모델링에서 작성한 **유스케이스 명세서**를 하나의 표현범위로 하지만, 하나의 클래스에 포함된 **오퍼레이션**을 하나의 범위로 표현하기도 한다.

<table>
<tr><td>액터 Actor</td><td>시스템으로부터 서비스를 요청하는 외부요소로 사람이나 외부 시스템을 의미</td></tr>
<tr><td>객체
Object</td><td>메시지를 주고받는 주체
콜론(:)을 기준으로 앞쪽에는 객체명을 뒤쪽에는 클래스명을 기술 (객체명 : 클래스명)</td></tr>
<tr><td>라이프라인
Lifeline</td><td>객체가 메모리에 존재하는 기간으로, 객체 아래쪽에 점선을 그어 표현
객체 소멸(X)이 표시된 기간까지 존재</td></tr>
<tr><td>활성상자
Activation Box</td><td>객체가 메시지를 주고받으며 구동되고 있음을 표현
라이프라인 상에 겹쳐 직사각형 형태로 표현</td></tr>
<tr><td>메시지
Message</td><td>객체가 상호작용을 위해 주고받는 메시지
메시지에 번호를 표기하여 전달 순서를 표현할 수 있다.
<table>
<tr><td>⟶</td><td>동기</td><td>메시지를 보낸 후 결과 반환될 때까지 기다린다.</td></tr>
<tr><td>→</td><td>비동기</td><td>메시지를 보낸 후 결과 반환될 때까지 기다리지 않고 다른 작업수행</td></tr>
<tr><td>--></td><td>생성</td><td>메시지를 받는 새로운 객체를 생성</td></tr>
<tr><td>----▸</td><td>응답</td><td>동기 메시지에 대한 수행 결과</td></tr>
</table></td></tr>
<tr><td>객체 소멸</td><td>라이프라인 상에서 객체 소멸 표시를 만나면 해당 객체는 더 이상 메모리에 존재하지 않음
객체 라이프라인 마지막에 X로 표시</td></tr>
<tr><td>프라임
Frame</td><td>다이어그램의 전체 또는 일부를 묶어서 표현
프라임 왼쪽 위에 다이어그램의 종류와 제목 표기</td></tr>
</table>

커뮤니케이션 다이어그램 Communication Diagram

시스템이나 객체들이 메시지를 주고받으며 시간의 흐름에 따라 상호작용하는 과정을 그림으로 표현한 것

시퀀스 다이어그램과 같이 메시지를 표현하는데 커뮤니케이션은 **객체들 간의 관계**까지 표현한다.

초기에 현업 다이어그램 Collaboration Diagram이라고 불렸다.

<table>
<tr><td>액터 Actor</td><td>시스템으로부터 서비스를 요청하는 외부요소로, 사람이나 외부 시스템을 의미</td></tr>
<tr><td>객체 Object</td><td>메시지를 주고받는 주체
콜론을 기준으로 앞쪽에는 객체명, 뒤쪽에는 클래스명 기술 (객체명:클래스명)</td></tr>
<tr><td>링크 Link</td><td>객체들 간의 관계를 표현
액터와 객체, 객체와 객체 간에 실선을 그어 표현
링크에 메시지 표현</td></tr>
<tr><td>메시지
Message</td><td>객체가 상호 작용을 위 주고받는 메시지
화살표 방향은 메시지를 받는 쪽으로 향하게 표현
일정한 순서에 의해 처리되는 메시지의 경우 숫자로 순서를 표시
메시지의 종류는 시퀀스 다이어그램에서 표현하는 방법과 동일</td></tr>
</table>

상태 다이어그램 State Diagram

객체들 사이에 발생하는 **이벤트**에 의한 **객체들의 상태 변화**를 그림으로 표현한 것

시스템에서 **시스템 변환 이벤트**를 확인할 필요가 있는 객체만을 대상으로 그린다.

- A이벤트에 의해 B상태로 전환된다

상태 State	객체의 상태를 표현 객체 상태를 둥근 사각형 안에 기술 - 시작 상태 ● - 종료 상태 ◉
상태 전환	상태 사이의 흐름, 변화를 화살표로 표현 **화살표에 이벤트 표현**
이벤트 Event	상태에 변화를 주는 현상
프레임 Frame	상태 다이어그램의 범위를 표현