

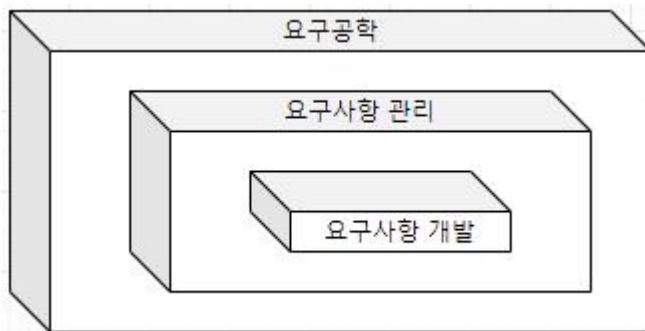
과정명		콘텐츠명	스마트훈련 활동지					담당교사	훈련생명
Spring기반자바(Java) 융합개발자_스마트훈련과정		요구사항 확인 part 2	관련교과 : 실전프로젝트		평가일 : 2021. 10. 8			김상우	
차시	1/10	콘텐츠 회차명	요구사항 정의	원격수업 시간	32분	재량활동 시간	18분	예제제출 (O,X)	

【보충학습자료 : 요구사항 정의】

◆ 요구공학(Requirements Engineering)

요구공학은 무엇을 개발해야 하는지 요구사항을 정의하고, 분석 및 관리하는 프로세스를 연구하는 학문이다.

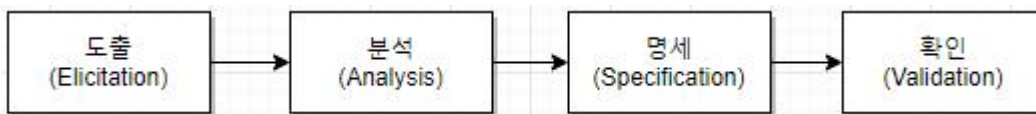
- 점점 복잡하고 대형화되어가는 소프트웨어 개발 환경에 따라 사용자 요구사항도 더욱 복잡해지고 잦은 변경이 발생하는데, 이는 요구사항에 문제가 발생할 가능성을 높이며 요구사항 관리가 잘못될 수 있는 원인이 된다.
- 요구공학은 요구사항 변경의 원인과 처리 방법을 이해하고 요구사항 관리 프로세스의 품질을 개선하여 소프트웨어 프로젝트 실패를 최소화하는 것을 목표로 한다.



1. 요구사항 개발 프로세스

요구사항 개발 프로세스는 개발 대상에 대한 요구사항을 체계적으로 도출하고 이를 분석한 후 분석 결과를 명세서(Specification Document)에 정리한 다음 마지막으로 이를 확인 및 검증하는 일련의 구조화된 활동이다.

- 요구사항 개발 프로세스가 진행되기 전에 개발 프로세스가 비즈니스 목적에 부합되는지, 예산은 적정한지 등에 대한 정보를 수집, 평가한 보고서를 토대로 타당성 조사(Feasibility Study)가 선행되어야 한다.
- 요구사항 개발은 요구공학(Requirement Engineering)의 한 요소이다.



2. 요구사항 도출(Requirement Elicitation, 요구사항 수집)

요구사항 도출은 시스템, 사용자, 그리고 시스템 개발에 관련된 사람들이 서로 의견을 교환하여 요구사항이 어디에 있는지, 어떻게 수집할 것인지를 식별하고 이해하는 과정이다.

- 요구사항 도출은 소프트웨어가 해결해야 할 문제를 이해하는 첫 번째 단계이다.
- 요구사항 도출 단계에서 개발자와 고객 사이의 관계가 만들어지고 이해관계자(Stakeholder)가 식별된다.
- 이 단계에서는 다양한 이해관계자 간의 효율적인 의사소통이 중요하다.
- 요구사항 도출은 소프트웨어 개발 생명 주기(SDLC; Software Development Life Cycle) 동안 지속적으로 반복된다.
- 요구사항을 도출하는 주요 기법에는 인터뷰, 설문, 브레인스토밍, 워크샵, 프로토타이핑, 유스케이스 등이 있다.

*** 브레인스토밍(Brain Storming)**

브레인스토밍은 3인 이상이 자유롭게 의견을 교환하면서 독창적인 아이디어를 산출해 내는 방법이다.

*** 프로토타이핑(Prototyping)**

프로토타이핑은 프로토타입(견본품)을 통해 효과적으로 요구 분석을 수행하면서 명세서를 산출하는 작업으로, 가장 단순한 형태는 설명을 위해 종이에 대략적인 순서나 형태를 그려 보여주는 것이다.

*** 유스케이스(Use Case)**

유스케이스는 사용자의 요구사항을 기능 단위로 표현하는 것이다.

3. 요구사항 분석(Requirement Analysis)

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

- 사용자 요구사항의 타당성을 조사하고 비용과 일정에 대한 제약을 설정한다.
- 내용이 중복되거나 하나로 통합되어야 하는 등 서로 상충되는 요구사항이 있으면 이를 해결한다.
- 도출된 요구사항들을 토대로 소프트웨어의 범위를 파악한다.
- 도출된 요구사항들을 토대로 소프트웨어와 주변 환경이 상호 작용하는 방법을 이해한다.

4. 요구사항 명세(Requirement Specification)

요구사항 명세는 요구사항을 체계적으로 분석한 후 승인될 수 있도록 문서화하는 것을 의미한다.

- 요구사항을 문서화할 때는 기능 요구사항은 빠짐없이 완전하고 명확하게 기술해야 하며, 비기능 요구사항은 필요한 것만 명확하게 기술해야 한다.
- 요구사항은 사용자가 이해하기 쉬우며, 개발자가 효과적으로 설계할 수 있도록 작성되어야 한다.
- 설계 과정에서 잘못된 부분이 확인될 경우 그 내용을 요구사항 정의서에서 추적할 수 있어야 한다.

5. 요구사항 확인(Requirement Validation, 요구사항 검증)

요구사항 확인은 개발 자원을 요구사항에 할당하기 전에 요구사항 명세서가 정확하고 완전하게 작성되었는지를 검토하는 활동이다.

- 분석가가 요구사항을 정확하게 이해한 후 요구사항 명세서를 작성했는지 확인(Validation)하는 것이 필요하다.
- 요구사항 명세서의 내용이 이해하기 쉬운지, 일관성은 있는지, 회사의 기준에는 맞는지, 그리고 누락된 기능은 없는지 등을 검증(Verification)하는 것이 중요하다.
- 요구사항 문서는 이해관계자들이 검토해야 한다.
- 일반적으로 요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 *형상 관리*를 수행한다.

*** 형상 관리(SCM; Software Configuration Management)**

소프트웨어 개발 단계의 각 과정에서 만들어지는 프로그램, 프로그램을 설명하는 문서, 데이터 등을 통칭하여 형상이라고 한다. 형상 관리는 소프트웨어의 개발 과정에서 만들어지는 형상들의 변경 사항을 관리하는 일련의 활동을 말한다.

◆ 소프트웨어 요구사항 명세서

소프트웨어 요구사항 명세서(SRS; Software Requirement Specification)는 업계 표준 용어로 소프트웨어가 반드시 제공해야 하는 기능, 특징, 제약조건 등을 명시한다.

- 시스템의 모든 동작뿐만 아니라 성능, 보안, 사용성과 같은 품질도 기술되어야 한다.
- 프로젝트 유형에 맞게 양식을 만들어 사용한다.
- 소프트웨어 요구사항 명세서에 포함되는 시스템 기능, 데이터, 외부 인터페이스, 품질 요구사항은 요구사항 단위별로 개별 요구사항 명세서를 작성한다.
- 다음은 여러 유형의 프로젝트에 유용하게 사용할 수 있는 소프트웨어 요구사항 명세서 양식이다.

- | |
|--|
| <ol style="list-style-type: none">1. 소개<ol style="list-style-type: none">1.1 목적1.2 문서 규칙1.3 프로젝트 범위1.4 참조2. 전반적인 설명<ol style="list-style-type: none">2.1 제품 관점2.2 사용자 클래스 및 특징2.3 운영환경2.4 설계 및 구현 제약 조건2.5 가정 및 의존성3. 시스템 기능<ol style="list-style-type: none">3.1 시스템 기능<ol style="list-style-type: none">3.1.1 설명3.1.2 기능적 요구사항4. 데이터 요구사항<ol style="list-style-type: none">4.1 논리 데이터 모델4.2 데이터 사전4.3 보고서4.4 데이터 수집, 무결성, 보존 및 폐기5. 외부 인터페이스 요구사항<ol style="list-style-type: none">5.1 사용자 인터페이스5.2 소프트웨어 인터페이스5.3 하드웨어 인터페이스5.4 통신 인터페이스6. 품질 속성<ol style="list-style-type: none">6.1 사용성6.2 성능6.3 보안6.4 안전6.X 기타7. 국제화 및 현지화 요구8. 기타 요구 사항 <p>부록A : 용어 사전
부록B : 분석 모델</p> |
|--|

【문제】

1. 요구분석 단계를 순서대로 바르게 나열한 것은? (④)

- | |
|---|
| 가. 요구사항 검증
나. 요구사항 명세화
다. 타당성 조사
라. 요구사항 추출 및 분석 |
|---|

- ① 다 → 라 → 가 → 나
- ② 라 → 다 → 나 → 가
- ③ 라 → 가 → 다 → 나
- ④ 다 → 라 → 나 → 가

해설 : 개발에 대한 타당성이 충족되면 작성된 요구사항을 분석하여 정리한 후 검증하는 과정을 진행한다.

2. 사용자 요구사항 추출(Elicitation) 방법 중 시스템 수행 결과를 설명하기 위해 종이에 화면 순서를 기술하여 고객과 사용자에게 보여주는 것과 관련된 것은? (④)

- ① 인터뷰
- ② 설문
- ③ 브레인스토밍
- ④ 프로토타이핑

3. 요구 분석에 대한 설명으로 옳지 않은 것은? (①)

- ① 고객이나 개발자 누구나 알 수 있는 내용은 요구사항에서 생략하여도 무방하다.
- ② 요구사항은 소프트웨어를 개발하고 검증하는 기반이 된다.
- ③ 요구사항은 명확하고 구체적이며 검증이 가능해야 한다.
- ④ 요구사항은 크게 기능적인 요구사항과 비기능적 요구사항으로 분류된다.

해설 : 요구사항은 제품 개발 과정에서 검증 테스트를 위한 기반이 되므로 개발에 필요한 모든 요소가 빠짐없이 완전하게 기술되어야 한다.

차시	2/10	콘텐츠 회차명	요구사항 확인	원격수업 시간	28분	재량활동 시간	22분	예제제출 (O,X)	
----	------	---------	---------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 요구사항 확인 기법】

1. 요구사항 확인 기법

요구사항 확인 기법은 요구사항 개발 과정을 거쳐 문서화된 요구사항 관련 내용을 확인하고 검증하는 방법이다.

- 요구사항에 자원이 배정되기 전에 문제 파악을 위한 검증을 수행해야 한다.
- 요구사항 확인 기법에는 요구사항 검토(Requirement Reviews), 프로토타이핑(Prototyping), 모델 검증(Model Verification), 인수 테스트(Acceptance Tests) 등이 있다.

2. 요구사항 검토(Requirement Reviews)

요구사항 검토는 문서화된 요구사항을 훑어보면서 확인하는 것으로 가장 일반적인 요구사항 검증 방법이다.

- 요구사항 검토자들은 요구사항 검토를 통해 명확하지 않은 내용은 없는지, 가정이 잘못되지는 않았는지, 정해놓은 기준을 벗어나지는 않는지 등을 찾아낸다.
- 요구사항 검토자 그룹을 구성할 때는 구성 방법이 중요하다. 예를 들어 고객 중심 프로젝트에 대한 검토자 그룹에는 고객 대표자가 꼭 포함되어야 하기 때문이다.
- 검토는 시스템 정의서(System Definition Document), 시스템 사양서(System Specification), 소프트웨어 요구사항 명세서(SRS; Software Requirements Specification Document) 등을 완성한 시점에 이루어진다.

3. 프로토타이핑(Prototyping)

프로토타이핑은 초기 도출된 요구사항을 토대로 프로토타입(Prototype)을 만든 후 대상 시스템의 개발이 진행되는 동안 도출되는 요구사항을 반영하면서 지속적으로 프로토타입을 재작성하는 과정이다.

- 상품이나 서비스가 출시되기 전에 개발 대상 시스템 또는 그 일부분을 개략적으로 만든 원형을 프로토타입이라고 한다.
- 프로토타이핑을 수행하면서 새로운 요구사항이 도출될 수 있다.
- 소프트웨어 요구사항에 대한 소프트웨어 엔지니어의 해석이 맞는지 확인하기 위한 수단으로 주로 사용된다.
- 프로토타이핑은 다음과 같은 장·단점이 있다.

장점	단점
<ul style="list-style-type: none"> • 빠르게 제작할 수 있으며, 반복되는 제작을 통해 발전된 결과물을 얻을 수 있다. • 최종 시스템을 완성하기 전에 추가/변경 요구사항이나 아이디어 등에 대한 피드백이 가능하다. • 이해하기 쉬워 사용자와 개발자 또는 개발자 사이의 의사소통이 원활해진다. • 개발될 시스템의 사용에 대한 문제점을 시스템 완성 전에 식별할 수 있다. • 프로토타입이 개선될수록 변동 가능한 요구사항들이 감소한다. 	<ul style="list-style-type: none"> • 사용자의 관심이 핵심에서 벗어나 프로토타입 제작에만 집중될 수 있다. • 개발 대상의 일부만을 대상으로 프로토타입이 제작된 경우 대상 범위를 잘못 이해하여 사용성이 과대평가 될 수 있다. • 지속적이고 반복적인 프로토타입의 개선으로 인한 비용이 부담될 수 있다.

4. 모델 검증(Model Verification)

모델 검증이란 요구사항 분석 단계에서 개발된 모델이 요구사항을 충족시키는지 검증하는 것이다.

- 객체 모델의 경우 객체들 사이에 존재하는 의사소통 경로(Communication Path)를 검증(Verify)하기 위하여 정적 분석(Static Analysis)을 수행하는 것이 유용하다.

* 정적 분석(Static Analysis)

정적 분석은 실행을 통해서 확인하는 것이 아니라 명세서의 정확성이나 일관성 등을 확인하거나 분석 도구를 사용해 확인하는 방법이다. 직접 실행을 통해 확인하는 방법은 동적 분석(Dynamic Analysis)이라고 한다.

5. 인수 테스트(Acceptance Tests)

인수 테스트는 사용자가 실제로 사용될 환경에서 요구사항들이 모두 충족되는지 사용자 입장에서 확인하는 과정이다.

- 각각의 요구사항을 어떻게 확인할 것인지에 대한 계획을 세워야 한다.
- 인수 테스트의 종류에는 사용자 인수 테스트, 운영상의 인수 테스트, 계약 인수 테스트, 규정 인수 테스트, 알파 검사, 베타 검사가 있다.

【문제】

1. 요구사항 확인에 대한 설명으로 잘못된 것은? (④)

① 요구사항 확인 기법은 요구사항 개발 과정을 거쳐 문서화된 요구사항 관련 내용을 확인하고 검증하는 방법이다.

② 일반적으로 요구사항 관리 도구를 이용하여 요구사항 정의 문서들에 대해 형상 관리를 수행한다.

③ 요구사항 문서는 이해관계자들이 검토해야 한다.

④ 요구사항에 자원이 배정된 후에 문제 파악을 위한 검증을 수행한다.

해설 : 요구사항에 대한 분석과 확인이 완료된 후에 자원을 배정한다.

2. 다음 중 요구사항 확인 기법이 아닌 것은? (②)

① 요구사항 검토

② 정형 분석

③ 모델 검증

④ 인수 테스트

3. 요구사항 확인 기법 중 하나인 프로토타이핑의 장점이 아닌 것은? (③)

① 빠르게 제작할 수 있으며, 반복되는 제작을 통해 발전된 결과물을 얻을 수 있다.

② 이해하기 쉬워 사용자와 개발자 또는 개발자 사이의 의사소통이 원활해진다.

③ 지속적이고 반복적인 프로토타입의 개선으로 인해 비용이 증가한다.

④ 프로토타입이 개선될수록 변동 가능한 요구사항들이 감소한다.

차시	3/10	콘텐츠 회차명	요구사항 검증	원격수업 시간	28분	재량활동 시간	22분	예제제출 (O/X)	
----	------	---------	---------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 요구사항 정의】

1. 요구사항의 개념 및 특징

요구사항은 소프트웨어가 어떤 문제를 해결하기 위해 제공하는 서비스에 대한 설명과 정상적으로 운영 되는데 필요한 제약조건 등을 나타낸다.

- 요구사항은 소프트웨어 개발이나 유지 보수 과정에서 필요한 기준과 근거를 제공한다.
- 요구사항은 개발하려는 소프트웨어의 전반적인 내용을 확인할 수 있게 하므로 개발에 참여하는 *이해관계자들* 간의 의사소통을 원활하게 하는 데 도움을 준다.
- 요구사항이 제대로 정의되어야만 이를 토대로 이후 과정의 목표와 계획을 수립할 수 있다.

* 이해관계자(利害關係者)

소프트웨어 개발과 관련해서 이해관계자는 소프트웨어 개발 의뢰자, 소프트웨어 개발자, 소프트웨어 사용자 등이 있다.

2. 요구사항의 유형

요구사항은 일반적으로 기술하는 내용에 따라 기능 요구사항(Functional requirements)과 비기능 요구사항(Non-functional requirements)으로 구분하며, 기술 관점과 대상의 범위에 따라 시스템 요구사항(System requirements)과 사용자 요구사항(User requirements)으로 나눈다.

유형	내용
기능 요구사항 (Functional requirements)	<ul style="list-style-type: none"> • 시스템이 무엇을 하는지, 어떤 기능을 하는지에 대한 사항 • 시스템의 입력이나 출력으로 무엇이 포함되어야 하는지, 시스템이 어떤 데이터를 저장하거나 연산을 수행해야 하는지에 대한 사항 • 시스템이 반드시 수행해야 하는 기능 • 사용자가 시스템을 통해 제공받기를 원하는 기능
비기능 요구사항 (Non-functional requirements)	<ul style="list-style-type: none"> • 시스템 장비 구성 요구사항 : 하드웨어, 소프트웨어, 네트워크 등의 시스템 장비 구성에 대한 요구사항 • 성능 요구사항 : 처리 속도 및 시간, 처리량, 동적·정적 적용량, 가용성 등 성능에 대한 요구사항 • 인터페이스 요구사항 : 시스템 인터페이스와 사용자 인터페이스에 대한 요구사항으로 다른 소프트웨어, 하드웨어 및 통신 인터페이스, 다른 시스템과의 정보 교환에 사용되는 프로토콜과의 연계도 포함하여 기술 • 데이터 요구사항 : 초기 자료 구축 및 데이터 변환을 위한 대상, 방법, 보안이 필요한 데이터 등 데이터를 구축하기 위해 필요한 요구사항 • 테스트 요구사항 : 도입되는 장비의 성능 테스트(BMT)나 구축된 시스템이 제대로 운영되는지를 테스트하고 점검하기 위한 테스트 요구사항 • 보안 요구사항 : 시스템의 데이터 및 기능, 운영 접근을 통제하기 위한 요구사항 • 품질 요구사항 : 관리가 필요한 품질 항목, 품질 평가 대상에 대한 요구사항으로 <i>가용성, 정합성, 상호 호환성, 대응성, 신뢰성, 사용성, 유지·관리성, 이식성, 확장성</i>, 보안성 등으로 구분하여 기술 • 제약사항 : 시스템 설계, 구축, 운영과 관련하여 사전에 파악된 기술, 표준, 업무, 법·제도 등의 제약조건 • 프로젝트 관리 요구사항 : 프로젝트의 원활한 수행을 위한 관리 방법에 대한 요구사항 • 프로젝트 지원 요구사항 : 프로젝트의 원활한 수행을 위한 지원 사항이나 방안에 대한 요구사항
사용자 요구사항 (User requirements)	<ul style="list-style-type: none"> • 사용자 관점에서 본 시스템이 제공해야 할 요구사항 • 사용자를 위한 것으로 친숙한 표현으로 이해하기 쉽게 작성된다.
시스템 요구사항 (System requirements)	<ul style="list-style-type: none"> • 개발자 관점에서 본 시스템 전체가 사용자와 다른 시스템에 제공해야 할 요구사항 • 사용자 요구사항에 비해 전문적이고 기술적인 용어로 표현된다. • 소프트웨어 요구사항이라고도 한다.

*** 비기능 요구사항에서 품질 요구사항 관련 용어 정리**

- 가용성 : 사용하고자 할 때 언제라도 사용할 수 있는 정도
- 적합성 : 데이터의 값이 서로 모순 없이 일관되게 일치하는 정도
- 상호 호환성 : 다른 소프트웨어와 정보를 교환할 수 있는 정도
- 대응성 : 발생한 상황에 대처하는 정도
- 이식성 : 다양한 하드웨어 환경에서도 운용 가능하도록 쉽게 수정될 수 있는 정도
- 확장성 : 규모나 범위를 넓힐 수 있는 정도

【문제】

1. 병원 진료관리시스템의 기능적 요구사항으로 옳지 않은 것은? (④)

- ① 담당 의사는 자신이 담당한 환자의 진료 내용을 입력 또는 수정한다.
- ② 환자 정보 관리자는 환자의 정보를 등록, 삭제할 수 있다.
- ③ 환자는 자신이 진료한 내역을 조회할 수 있다.
- ④ 시스템 장애로 인한 정지시간이 한 달에 1시간을 넘지 않아야 한다.

해설 : 성능에 관한 내용은 외적인 품질 속성, 즉 비기능 요구사항에 해당된다.

2. 다음은 '호텔 예약 시스템'의 요구사항을 나열한 것이다. 기능적 요구사항에 대한 설명으로 가장 옳지 않은 것은? (④)

- ① 예약 시 고객의 정보를 입력하는 방법을 결정해야 한다.
- ② 영수증과 예약 확인서에 어떤 정보를 표시할지 결정해야 한다.
- ③ 예약 대행 여행사와 고객이 호텔 정보 데이터베이스에 접근할 때 어떤 정보를 얻을 수 있는지를 결정해야 한다.
- ④ 해외 분점 호텔의 고객 정보까지 관리하기 위해 시스템을 확장할 수 있도록 설계해야 한다.

해설 : 확장성, 가용성, 보안성, 신뢰성, 호환성 등은 모두 비기능 요구사항에 해당한다.

3. 비기능 요구사항에 대한 설명으로 옳지 않은 것은? (④)

- ① 예산의 범위, 조직의 비전, 상호 호환성, 보안성, 안전성과 같은 사용자의 필요에 의해 발생한다.
- ② 시스템이 제공하는 서비스의 품질에 관한 것이다.
- ③ 시스템에서 제공되는 서비스나 기능에 대한 제약 사항에 관한 것이다.
- ④ 시스템이 특정 입력에 대해 어떻게 반응하는지, 사용자의 요구에 대해 시스템이 어떻게 동작해야 하는지에 관한 사항이다.

해설 : 시스템이 무엇을 하는지, 어떻게 하는지 등에 관한 기능이나 동작에 관한 것은 기능 요구사항이다.

4. 다음은 서점 시스템의 요구사항에 대한 내용이다. 비기능 요구사항에 대한 설명은? (③)

- ① 사용자는 로그인 또는 비로그인을 통해 책을 구매할 수 있어야 한다.
- ② 사용자가 책을 현금으로 구매하였을 경우 현금영수증 처리를 할 수 있어야 한다.
- ③ 동시에 100명 이상이 주문을 요청해도 처리할 수 있어야 한다.
- ④ 사용자가 마이페이지에 저장해 놓은 도서 목록은 일정기간 동안 그대로 저장되어 있어야 한다.

해설 : ③번은 시스템의 품질에 관한 요구사항으로 비기능 요구사항에 해당한다.

차시	4/10	콘텐츠 회차명	요구사항 시스템화	원격수업 시간	28분	재량활동 시간	22분	예제제출 (O,X)	
----	------	---------	-----------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 현행 시스템 분석】

1. 플랫폼 기능 분석

1) 플랫폼의 개념

• 일반적으로 플랫폼이란 응용 소프트웨어 프로그램을 구동시키는데 쓰이는 하드웨어와 소프트웨어의 결합을 말한다. 플랫폼은 소프트웨어 개발과 운영을 쉽게 하고 한번 만들어진 소프트웨어는 동일한 플랫폼에서는 언제, 어디서 실행시키더라도 손쉽게 구동이 될 수 있도록 만들어진 결합체를 의미한다.

• 예를 들면, 자바 플랫폼, 닷넷 플랫폼을 들 수 있고, 모바일 플랫폼은 iOS와 안드로이드 플랫폼을 들 수 있다.

• 요구사항을 확인하기 위하여 현행 시스템을 분석하기 위해서는 현재 시스템의 구성요소인 소프트웨어 및 하드웨어 플랫폼을 상세하게 분석해야 한다.

2) 플랫폼의 기능

- 소프트웨어 개발 및 운영 비용을 감소한다.
- 동일한 플랫폼간 커뮤니티를 형성하여 네트워크 효과를 유발한다.
- 소프트웨어 개발의 생산성을 향상시킨다.

3) 플랫폼의 기능 특성 확인 방법

- 기능 테스트 : 현재 시스템의 플랫폼을 평가할 수 있는 기능 테스트를 수행한다.
- 사용자 인터뷰 : 현재 시스템 사용자를 대상으로 플랫폼 기능의 불편함을 인터뷰 한다.
- 문서 점검 : 현재 시스템의 플랫폼과 유사한 플랫폼의 기능 자료를 분석한다.

2. 플랫폼 성능 특성 분석

1) 현행 시스템 분석하기에서 플랫폼의 성능 특성을 알아야 하는 이유

• 현재의 시스템에 구성된 플랫폼의 성능을 분석해야 사용자가 사용하기에 속도가 느린지 빠른지 알 수 있기 때문이다.

• 사용자 요구사항 중에 성능에 대한 요구사항이 있는데 이는 현재 시스템의 플랫폼 성능이 느려서 제기되는 요구사항일 가능성이 높기 때문이다.

2) 플랫폼 성능 특성 확인 방법

- 성능 테스트 : 현재 시스템의 플랫폼을 대상으로 성능/부하 테스트를 수행한다.
- 문서 점검 : 현재 시스템의 플랫폼과 유사한 플랫폼의 성능 자료를 분석한다.
- 사용자 인터뷰 : 현재 시스템 사용자와의 인터뷰를 통하여 성능을 확인한다.

3. 운영체제 분석

1) 운영체제(OS : Operating System)의 개념

• 운영체제는 하드웨어와 소프트웨어 자원을 관리하고 컴퓨터 프로그램을 위한 공통 서비스를 제공하는 소프트웨어를 의미한다.

2) 현재 시스템의 운영체제를 분석한다.

- 현재 운영체제의 종류, 버전, 패치 일자, 백업 주기 등을 분석한다.
- 운영체제의 종류에는 윈도우, 리눅스, 유닉스, IBM AIX 등 매우 다양하다.

4. 네트워크 분석

1) 네트워크의 개념

-
- 컴퓨터 네트워크 또는 컴퓨터망은 노드들이 자원을 공유할 수 있게 하는 디지털 전기통신망의 하나이다. 즉, 분산되어 있는 컴퓨터를 통신망으로 연결한 것을 말한다.
 - 컴퓨터 네트워크에서 컴퓨팅 장치들은 노드 간 연결(데이터 링크)을 사용하여 서로에게 데이터를 교환한다. 이 데이터 링크들은 유선, 광케이블과 같은 케이블 매체, 또는 와이파이와 같은 무선 매체를 통해 확립된다.

2) 현재 시스템의 네트워크를 분석한다.

- 현재 시스템이 구성된 네트워크 구조를 분석한다.
- 사내 인터넷 데이터 센터(IDC), 백본망, 라우터, 스위치, 방화벽 등을 분석한다.

3) 현재 시스템의 네트워크 구성도를 작성한다.

- 현재 시스템의 서버의 위치, 서버 간의 네트워크 연결 방식, 논리 및 물리 네트워크 구성도를 작성한다.

5. DBMS 분석

1) 데이터베이스의 개념

- 사용자와 다른 애플리케이션, 데이터베이스 등과 상호 작용하여 데이터를 저장하고 분석하기 위한 컴퓨터 소프트웨어로 데이터베이스 생성, 조회, 변경 등의 관리가 주요 기능이다.

2) 데이터베이스의 기능

- 데이터 저장과 개발 및 유지보수 측면에서 중복서어 통제
- 다중 사용자간의 데이터 공유
- 권한 없는 사용자의 데이터 접근 통제
- 다양한 사용자에게 다양한 형태의 인터페이스 제공
- 데이터 사이에 존재하는 복잡한 관련성 표현
- 데이터베이스의 무결성 보장
- 백업과 복구 기능 제공

3) 현재 시스템의 데이터베이스 시스템을 분석한다.

- DBMS의 종류, 버전, 구성방식, 스토리지 크기, 백업 주기 등을 분석한다.
- 테이블 수량, 데이터 증가 추이, 백업 방식 등을 분석한다.

4) 논리/물리 테이블의 구조 파악

- 각 테이블의 정규화 정도, 조인의 난이도를 파악한다.
- 조인(JOIN)이란 여러 개의 테이블을 결합하여 데이터를 검색하는 것을 의미한다.
- 각종 프로시저, JOB, 트리거 등을 분석한다.

6. 비즈니스 융합 분석

1) 비즈니스 융합의 개념

① 비즈니스(Business)

- 재화나 서비스 등 유무형의 가치를 제공하고 그에 상응하는 대가를 보상받는 등 영리를 목적으로 행하는 모든 활동(임춘성, 2011)을 말한다.

- 재화나 서비스의 개발 및 제공을 통해 영리를 추구하는 기업활동 또는 경영활동(임춘성, 2011)이다.

② 비즈니스 모델(Business Model)

- 고객의 가치를 창출하고 시장에서 성공적인 경쟁을 하기 위해 고안된 조직 목표, 전략, 프로세스, 기술, 구조 등을 포함하는 요소들의 구성체이다.

③ 비즈니스 융합(Convergence)

- 산업 또는 시장 간의 경계를 허물고 ICT 등을 통한 새로운 전달방식을 도입함으로써 비즈니스 모델의 적용범위를 확대시키는 것을 의미한다.
-

2) 비즈니스 융합 유형

비즈니스 융합 유형	
제품 융합	• 2가지 이상 제품의 기능과 속성을 하나로 모음
서비스 융합	• 2가지 이상 서비스의 기능과 속성을 하나로 모음
제품과 IT 융합	• 기존 제품에 IT부품 또는 자재, SW 등을 추가함
서비스와 IT 융합	• 기존 서비스에 IT부품 또는 자재, SW 등을 추가함
제품의 서비스화	• 제품에 자사 또는 타사의 서비스를 부가하여 서비스 제공
서비스의 제품화	• 서비스를 제품화 또는 장비, 기기로 전환
제품과 서비스 통합	• 사용자의 요구에 부합하는 시스템 또는 솔루션

3) 비즈니스 융합 분석

① 고객 분석

- 비즈니스 모델 상에서 사업자에게 수익을 제공하는 참여자(고객)를 식별하고 분석한다.

② 제품 및 서비스 분석

- 비즈니스 모델 상에서 자사가 제공하는 상품 또는 서비스를 식별하고 분석한다.
- 비즈니스 융합 참여자간 제공하는 서비스와 제공받는 서비스를 식별하고 분석한다.

③ 사업구조 분석

- 상품 및 서비스의 제공자, 소비자 등 참여자간의 관계와 구조를 식별하고 분석한다.

【문제】

1. 다음 중 현행 시스템 분석하기에서 수행하는 활동으로 옳지 않은 것은? (④)

- ① 현재 시스템의 DBMS 분석
- ② 현재 시스템의 네트워크 분석
- ③ 현재 시스템의 운영체제 분석
- ④ 현재 시스템의 미래 형상 분석

2. 다음 중 플랫폼이 제공하는 기술적, 관리적 기능으로 옳지 않은 것은? (②)

- ① 소프트웨어 개발 및 운영 비용을 감소
- ② 모든 소프트웨어의 성능을 향상
- ③ 동일한 플랫폼간 커뮤니티를 형성하여 네트워크 효과를 유발
- ④ 소프트웨어 개발의 생산성을 향상

차시	5/10	콘텐츠 회차명	요구사항 타당성 분석	원격수업 시간	28분	재량활동 시간	22분	예제제출 (O,X)	
----	------	---------	-------------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 요구사항 분석 기법】

1. 요구사항 분석 기법

요구사항 분석 기법은 개발 대상에 대한 사용자의 요구사항 중 명확하지 않거나 모호한 부분을 걸러내기 위한 방법이다.

- 요구사항 분석 기법에는 요구사항 분류, 개념 모델링, 요구사항 할당, 요구사항 협상, 정형 분석 등이 있다.

2. 요구사항 분류(Requirement Classification)

요구사항을 명확히 확인할 수 있도록 다음과 같은 기준으로 분류한다.

- 기능 요구사항과 비기능 요구사항으로 분류한다.
- 하나 이상의 상위 요구사항에서 유도된 것인지 또는 이해관계자나 다른 원천(Source)으로부터 직접 발생한 것인지 분류한다.
- 개발할 제품에 관한 것인지 개발 과정(프로세스)에 관한 것인지 분류한다.
- 우선순위에 따라 분류한다.
- 소프트웨어에 미치는 영향의 범위에 따라 분류한다.
- 소프트웨어 생명 주기 동안에 변경될 가능성이 있는지 여부에 따라 분류한다.

3. 개념 모델링(Conceptual Modeling)

요구사항을 보다 쉽게 이해할 수 있도록 현실 세계의 상황을 단순화하여 개념적으로 표현한 것을 모델이라고 하며, 이러한 모델을 만드는 과정을 모델링이라고 한다.

- 모델은 문제가 발생하는 상황을 쉽게 이해시키고 해결책을 설명할 수 있으므로 실세계 문제에 대한 모델링은 소프트웨어 요구사항 분석의 핵심이다.
- 개념 모델은 문제의 주체인 *개체(Entity)*들과 그들 간의 관계 및 종속성을 반영한다.
- 요구사항을 이해하는 이해관계자별로 관점이 다양하므로 그에 맞게 개념 모델도 다양하게 표현되어야 한다.
- 개념 모델의 종류에는 유스케이스 다이어그램(Use Case Diagram), 데이터 흐름 모델(Data Flow Model), 상태 모델(State Model), 목표기반 모델(Goal-Based Model), 사용자 인터랙션(User Interactions), 객체 모델(Object Model), 데이터 모델(Data Model) 등이 있다.
- 모델링 표기는 주로 UML(Unified Modeling Language)을 사용한다.

* 개체

현실 세계에서는 사람, 자동차, 컴퓨터, 고양이 등과 같이 우리 주위에서 사용되는 물질적이거나 개념적인 것으로, 요구사항 분석에서는 요구 기능이 적용되는 시스템이나 기능을 사용하는 사람 또는 그러한 기능과 연동되는 다른 시스템을 의미한다.

* 종속성(從屬性, Dependency)

A가 어떤 행동을 할 때 반드시 B를 통해서만 수행할 수 있다면 A는 B에 종속적이라고 표현하는데, 이와 같이 서로 의존적인 관계를 의미한다.

4. 요구사항 할당(Requirement Allocation)

요구사항 할당은 요구사항을 만족시키기 위한 구성 요소를 식별하는 것이다.

- 식별된 구성 요소들 간에 어떻게 작용하는지 분석하는 과정에서 추가적인 요구사항이 발견될 수 있다.

5. 요구 협상(Requirement Negotiation)

요구사항 협상은 요구사항이 서로 충돌될 경우 이를 적절히 해결하는 과정이다.

- 요구사항이 다음과 같은 이유로 서로 충돌되는 경우 어느 한 쪽으로 맞추기보다는 적절한 기준점을 찾아 합의해야 한다.
 - 두 명의 이해관계자가 요구하는 요구사항이 서로 충돌되는 경우
 - 요구사항과 자원이 서로 충돌되는 경우
 - 기능 요구사항과 비기능 요구사항이 서로 충돌되는 경우
- 요구사항이 서로 충돌되는 경우에 각각에 우선순위를 부여하면, 무엇이 더 중요한지를 인식할 수 있으므로 문제 해결에 도움이 될 수 있다.

6. 정형 분석(Formal Analysis)

정형 분석은 구문(Syntax)과 의미(Semantics)를 갖는 정형화된 언어를 이용해 요구사항을 수학적 기호로 표현한 후 이를 분석하는 과정이다.

- 정형 분석(Formal Analysis)은 요구사항 분석의 마지막 단계에서 이루어진다.

* 정형 명세(Formal Specification)

정형화된 언어를 이용해 수학적 기호로 기술하는 것을 정형 명세라고 한다.

【문제】

1. 다음 중 요구사항 분석 기법이 아닌 것은? (④)

- ① 요구사항 분류
- ② 개념 모델링
- ③ 정형 분석
- ④ 요구사항 제거

2. 요구사항 분석 기법 중 다음이 설명하는 것은 무엇인가? (③)

- 구문(Syntax)과 의미(Semantics)를 갖는 정형화된 언어를 이용해 요구사항을 수학적 기호로 표현한 후 이를 분석하는 과정이다.
 - 요구사항 분석의 마지막 단계에서 이루어진다.

- ① 요구사항 분류
- ② 요구사항 협상
- ③ 정형 분석
- ④ 개념 모델링

3. 요구사항 분석 기법 중 개념 모델링에 대한 설명이 아닌 것은? (②)

- ① 개념 모델은 소프트웨어 요구사항 분석의 핵심이다.
- ② 요구사항을 이해하는 이해관계자별로 다른 의견이 도출되지 않도록 하나의 통일된 개념 모델로 표현한다.
- ③ 개념 모델은 문제의 주체인 개체(Entity)들과 그들 간의 관계 및 종속성을 반영한다.
- ④ 모델링 표기는 주로 UML(Unified Modeling Language)을 사용한다.

4. 요구사항 협상은 요구사항이 서로 충돌하는 경우 이를 적절히 해결하는 과정이다. 다음 중 요구사항이 서로 충돌되어 적절한 기준점을 찾아 합의해야 하는 경우가 아닌 것은? (②)

- ① 두 명의 이해관계자가 요구하는 요구사항이 서로 충돌되는 경우
 - ② 요구사항이 서로 우선순위가 다른 경우
 - ③ 요구사항과 자원이 서로 충돌되는 경우
 - ④ 기능 요구사항과 비기능 요구사항이 서로 충돌되는 경우
-

차시	6/10	콘텐츠 회차명	요구사항 관리	원격수업 시간	35분	재량활동 시간	15분	예제제출 (O,X)	
----	------	---------	---------	------------	-----	------------	-----	---------------	--

【보충학습자료 : UML(Unified Modeling Language)】

1. UML(Unified Modeling Language)의 개요

UML은 시스템 분석, 설계, 구현 등 시스템 개발 과정에서 시스템 개발자와 고객 또는 개발자 상호간의 의사소통이 원활하게 이루어지도록 표준화한 대표적인 객체지향 *모델링 언어*이다.

- UML은 Rumbaugh(OMT), Booch, Jacobson 등의 객체지향 방법론의 장점을 통합하였으며, 객체 기술에 관한 국제표준화기구인 OMG(Object Management Group)에서 표준으로 지정하였다.
- UML을 이용하여 시스템의 구조를 표현하는 6개의 구조 다이어그램과 시스템의 동작을 표현하는 7개의 행위 다이어그램을 작성할 수 있다.
- 각각의 다이어그램은 사물과 사물 간의 관계를 용도에 맞게 표현한다.
- UML의 구성 요소에는 사물, 관계, 다이어그램 등이 있다.

* 모델링 언어

모델링 언어란 우리가 만들고자 하는 것을 시각적으로 표현할 수 있는 표기법, 도구 등을 의미한다.

2. 사물(Things)

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

- 사물에는 구조 사물, 행동 사물, 그룹 사물, 주해 사물이 있다.

사물	내용
구조 사물 (Structural Things)	<ul style="list-style-type: none"> • 시스템의 개념적, 물리적 요소를 표현 • 클래스(Class), 유스케이스(Use Case), 컴포넌트(Component), 노드(Node) 등
행동 사물 (Behavioral Things)	<ul style="list-style-type: none"> • 시간과 공간에 따른 요소들의 행위를 표현 • 상호작용(Interaction), 상태 머신(State Machine) 등
그룹 사물 (Grouping Things)	<ul style="list-style-type: none"> • 요소들을 그룹으로 묶어서 표현 • 패키지(Package)
주해 사물 (Annotation Things)	<ul style="list-style-type: none"> • 부가적인 설명이나 제약조건 등을 표현 • 노트(Note)

* 컴포넌트(Component)

컴포넌트는 문서, 소스코드, 파일, 라이브러리 등과 같은 모듈화된 자원으로, 재사용이 가능하다.

3. 관계(Relationships)

관계는 사물과 사물 사이의 연관성을 표현하는 것으로, 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계, 실체화 관계 등이 있다.

연관(Association) 관계

연관 관계는 2개 이상의 사물이 서로 관련되어 있음을 표현한다.

- 사물 사이를 실선으로 연결하여 표현하며, 방향성은 화살표로 표현한다.
- 서로에게 영향을 주는 양방향 관계의 경우 화살표를 생략하고 실선으로만 연결한다.
- 연관에 참여하는 객체의 개수를 의미하는 다중도(Multiplicity)를 선 위에 표기한다.

다중도	의미
1	1개의 객체가 연관되어 있다.
n	n개의 객체가 연관되어 있다.
0..1	연관된 객체가 없거나 1개만 존재한다.
0..* 또는 *	연관된 객체가 없거나 다수일 수 있다.
1..*	연관된 객체가 적어도 1개 이상이다.
n..*	연관된 객체가 적어도 n개 이상이다.
n..m	연관된 객체가 최소 n개에서 최대 m개이다.

* 사물, 즉 객체는 유스케이스, 클래스, 컴포넌트와 같이 별도의 표현 형태가 있는 경우를 제외하고는 기본적으로 사각형으로 표현된다.

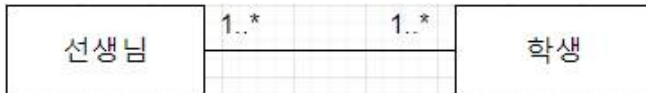
(예제1) 사람이 집을 소유하는 관계이다. 사람은 자기가 소유하고 있는 집에 대해 알고 있지만 집은 누구에 의해 자신이 소유되고 있는지 모른다는 의미이다.



(해설)

- '사람' 쪽에 표기된 다중도가 '1'이므로 집은 한 사람에 의해서만 소유될 수 있다.
- '집' 쪽에 표기된 다중도가 '1'이므로 사람은 집을 하나만 소유할 수 있다.

(예제2) 선생님은 학생을 가르치고 학생은 선생님으로부터 가르침을 받는 것과 같이 선생님과 학생은 서로 관계가 있다.



(해설)

- '선생님' 쪽에 표기된 다중도가 '1..*'이므로 학생은 한 명 이상의 선생님으로부터 가르침을 받는다.
- '학생' 쪽에 표기된 다중도가 '1..*'이므로 선생님은 한 명 이상의 학생을 가르친다.

집합(Aggregation) 관계

집합 관계는 하나의 사물이 다른 사물에 포함되어 있는 관계를 표현한다.

- 포함하는 쪽(전체, Whole)과 포함되는 쪽(부분, Part)은 서로 독립적이다.
- 포함되는 쪽(부분, Part)에서 포함하는 쪽(전체, Whole)으로 속이 빈 마름모를 연결하여 표현한다.

(예제) 프린터는 컴퓨터에 연결해서 사용할 수 있으며, 다른 컴퓨터에 연결해서 사용할 수도 있다.

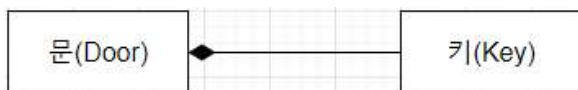


포함(Composition) 관계

포함 관계는 집합 관계의 특수한 형태로, 포함하는 사물의 변화가 포함되는 사물에게 영향을 미치는 관계를 표현한다.

- 포함하는 쪽(전체, Whole)과 포함되는 쪽(부분, Part)은 서로 독립될 수 없고 생명주기를 함께 한다.
- 포함되는 쪽(부분, Part)에서 포함하는 쪽(전체, Whole)으로 속이 채워진 마름모를 연결하여 표현한다.

(예제) 문을 열 수 있는 키는 하나이며, 해당 키로 다른 문은 열 수 없다. 문이 없어지면 키도 더 이상 필요하지 않다.

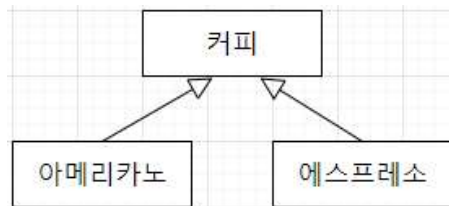


일반화(Generalization) 관계

일반화 관계는 하나의 사물이 다른 사물에 비해 더 일반적인지 구체적인지를 표현한다.

- 예를 들어 사람은 여자와 남자보다 일반적인 개념이고 반대로 여자와 남자는 사람보다 구체적인 개념이다.
- 보다 일반적인 개념을 상위(부모), 보다 구체적인 개념을 하위(자식)라고 부른다.
- 구체적(하위)인 사물에서 일반적(상위)인 사물 쪽으로 속이 빈 화살표를 연결하여 표현한다.

(예제) 아메리카노와 에스프레소는 커피이다. 다시 말하면, 커피에는 아메리카노와 에스프레소가 있다.



의존(Dependency) 관계

의존 관계는 연관 관계와 같이 사물 사이에 연관은 있으나 필요에 의해 서로에게 영향을 주는 짧은 시간 동안만 연관을 유지하는 관계를 표현한다.

- 하나의 사물과 다른 사물이 소유 관계는 아니지만 사물의 변화가 다른 사물에도 영향을 미치는 관계이다.
- 영향을 주는 사물(이용자)이 영향을 받는 사물(제공자) 쪽으로 점선 화살표를 연결하여 표현한다.

(예제) 등급이 높으면 할인율을 적용하고, 등급이 낮으면 할인율을 적용하지 않는다.

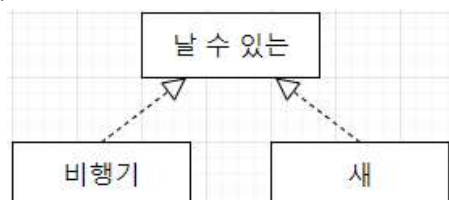


실체화(Realization) 관계

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

- 사물에서 기능 쪽으로 속이 빈 점선 화살표를 연결하여 표현한다.

(예제) 비행기는 날 수 있고 새도 날 수 있다. 그러므로 비행기와 새는 날 수 있다는 행위로 그룹화 할 수 있다.







【문제】

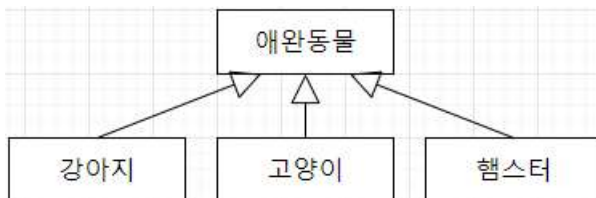
1. UML(Unified Modeling Language)의 구성 요소 중 사물(Thing)에 속하지 않는 것은? (②)

- ① 구조(Structural) 사물
- ② 분석(Analysis) 사물
- ③ 행동(Behavioral) 사물
- ④ 주해(Annotation) 사물

2. 객체지향 개발방법론 UML의 관계 표시법 중 의존(Dependency)을 표현하는 기호는? (②)

- ① 
- ② 
- ③ 
- ④ 

3. UML의 다이어그램에서 관계를 완성하고자 한다. 다음 관계의 표현으로 가장 적합한 것은? (②)



- ① 연관(association) 관계
- ② 일반화(generalization) 관계
- ③ 집단화(aggregation) 관계
- ④ 의존(dependency) 관계

차시	7/10	콘텐츠 회차명	분석 참고모델	원격수업 시간	29분	재량활동 시간	21분	예제제출 (O/X)	
----	------	---------	---------	------------	-----	------------	-----	---------------	--

【보충학습자료 : UML(Unified Modeling Language)】

4. 다이어그램(Diagram)

다이어그램은 사물과 관계를 도형으로 표현한 것이다.

- 여러 관점에서 시스템을 가시화한 뷰(View)를 제공함으로써 의사소통에 도움을 준다.
- 정적 모델링에서는 주로 구조적 다이어그램을 사용하고 동적 모델링에서는 주로 행위 다이어그램을 사용한다.

• 구조적(Structural) 다이어그램의 종류

구조적(Structural) 다이어그램의 종류	
클래스 다이어그램 (Class Diagram)	<ul style="list-style-type: none"> • 클래스와 클래스가 가지는 속성, 클래스 사이의 관계를 표현한다. • 시스템의 구조를 파악하고 구조상의 문제점을 도출할 수 있다.
객체 다이어그램 (Object Diagram)	클래스에 속한 사물(객체)들, 즉 인스턴스(Instance)를 특정 시점의 객체와 객체 사이의 관계로 표현한다.
컴포넌트 다이어그램 (Component Diagram)	<ul style="list-style-type: none"> • 실제 구현 모듈인 컴포넌트 간의 관계나 컴포넌트 간의 인터페이스를 표현한다. • 구현 단계에서 사용되는 다이어그램이다.
배치 다이어그램 (Deployment Diagram)	<ul style="list-style-type: none"> • 결과물, 프로세스, 컴포넌트 등 물리적 요소들의 위치를 표현한다. • 노드와 의사소통(통신) 경로로 표현한다. • 구현 단계에서 사용되는 다이어그램이다.
복합체 구조 다이어그램 (Composite Structure Diagram)	클래스나 컴포넌트가 복합 구조를 갖는 경우 그 내부 구조를 표현한다.
패키지 다이어그램 (Package Diagram)	유스케이스나 클래스 등의 모델 요소들을 그룹화한 패키지들의 관계를 표현한다.

• 행위(Behavioral) 다이어그램의 종류

행위(Behavioral) 다이어그램의 종류	
유스케이스 다이어그램 (Use Case Diagram)	<ul style="list-style-type: none"> • 사용자의 요구를 분석하는 것으로 기능 모델링 작업에 사용한다. • 사용자(Actor)와 사용 사례(Use Case)로 구성되며, 사용 사례 간에는 여러 형태의 관계로 이루어진다.
시퀀스 다이어그램 (Sequence Diagram)	상호 작용하는 시스템이나 객체들이 주고받는 메시지를 표현한다.
커뮤니케이션 다이어그램 (Communication Diagram)	시퀀스 다이어그램과 같이 동작에 참여하는 객체들이 주고받는 메시지를 표현하는데, 메시지뿐만 아니라 객체들 간의 연관까지 표현한다.
상태 다이어그램 (State Diagram)	하나의 객체가 자신이 속한 클래스의 상태 변화 혹은 다른 객체와의 상호 작용에 따라 상태가 어떻게 변화하는지를 표현한다.
활동 다이어그램 (Activity Diagram)	시스템이 어떤 기능을 수행하는지 객체의 처리 로직이나 조건에 따른 처리의 흐름을 순서에 따라 표현한다.
상호작용 개요 다이어그램 (Interaction Overview Diagram)	상호작용 다이어그램 간의 제어 흐름을 표현한다.
타이밍 다이어그램 (Timing Diagram)	객체 상태 변화와 시간 제약을 명시적으로 표현한다.

【문제】

1. 결과물, 프로세스, 컴포넌트 등 물리적인 자원의 위치를 표시하는 것으로 구현 단계에서 사용되는 UML 다이어그램은? (③)

- ① 컴포넌트(Component) 다이어그램
- ② 통신(Communication) 다이어그램
- ③ 배치(Deployment) 다이어그램
- ④ 상태(State) 다이어그램

2. 다음 중 동적인 행위를 표현하기 위한 UML 다이어그램이 아닌 것은? (④)

- ① 시퀀스(Sequence) 다이어그램
- ② 상태(State) 다이어그램
- ③ 활동(Activity) 다이어그램
- ④ 배치(Deployment) 다이어그램

3. 다음 중 UML 다이어그램이 아닌 것은? (④)

- ① 사용사례 다이어그램(Use Case Diagram)
- ② 순차 다이어그램(Sequence Diagram)
- ③ 클래스 다이어그램(Class Diagram)
- ④ 변화 다이어그램(Change Diagram)

4. UML 다이어그램 중 구현 단계에서 사용하기에 가장 적합한 것은? (②)

- ① 유즈케이스 다이어그램
- ② 컴포넌트 다이어그램
- ③ 활동 다이어그램
- ④ 클래스 다이어그램

해설 : 구현 단계에서 사용하는 다이어그램 2개는 배치 다이어그램과 컴포넌트 다이어그램이다.

시스템(System) / 시스템 범위(System Scope)

- 시스템 내부에서 수행되는 기능들을 외부 시스템과 구분하기 위해 시스템 내부의 유스케이스들을 사각형으로 묶어 시스템의 범위를 표현한다.
- 사각형 안쪽 상단에 시스템 명칭을 기술한다.
- (예제)에서는 사각형을 통해 <상품주문> 시스템의 범위를 알 수 있다.

액터(Actor)

- 시스템과 상호작용을 하는 모든 외부 요소로, 사람이나 외부 시스템을 의미한다.
- 액터는 시스템에 대해 수행할 수 있는 역할을 의미하기도 한다.
- 액터 이름이 구체적이면 안된다. ("홍길동"처럼 구체적이면 안됨. 역할명으로 지정함)
- 액터에는 주액터와 부액터가 있다.

주액터(Primary Actor)

- 시스템을 사용함으로써 이득을 얻는 대상으로, 주로 사람이 해당된다.
- 사람 형태를 간략화 하여 표현하며, 주로 시스템의 왼쪽에 배치한다.
- (예제)에서는 '비회원', '회원', '고객'이 주액터에 해당된다.

부액터(Secondary Actor)

- 주액터의 목적 달성을 위해 시스템에 서비스를 제공하는 외부 시스템으로, 조직이나 기관 등이 될 수 있다.
- 주로 시스템의 오른쪽에 배치하며, 시스템명을 사각형으로 묶은 후 상단에 《Actor》라고 표기한다.
- (예제)에서는 '재고 시스템', '결제 시스템', '배송업체'가 부액터에 해당된다.

유스케이스(Use Case)

- 사용자가 보는 관점에서 시스템이 액터에게 제공하는 서비스 또는 기능을 표현한 것이다.
- 타원으로 표현하며, 타원 안쪽이나 아래쪽에 유스케이스 이름을 기술한다.
- 유스케이스 이름은 액터와 시스템 사이에서 이뤄지는 상호 작용의 목적을 내포하되 단순 명료하게 기술해야 한다.
- 기능적 요소를 중심으로 기술하지만 비기능적 요소를 포함할 수 있다.
- 더 이상 분할되지 않는 기능의 단위이다.
- 액터에 의해 수행되며, 액터가 관찰할 수 있는 결과를 산출한다.
- 하나의 유스케이스 안에서 수행되는 동작은 유스케이스 수행 시 모두 수행되어야 하며, 부분적인 수행은 허용되지 않는다.
- 유스케이스는 분석, 설계, 테스트 등 개발 전 과정에서 이용될 수 있다.
- (예제)에서는 '상품조회', '이름으로 조회', '브랜드로 조회', '상품주문', '배송조회', '리뷰작성', '로그인', '사진 업로드'가 유스케이스에 해당된다.

관계(Relationship)

- 유스케이스 다이어그램에서 관계는 액터와 유스케이스, 유스케이스와 유스케이스 사이에서 나타날 수 있으며, 포함 관계, 확장 관계, 일반화 관계의 3종류가 있다.
- (해석) : '상품주문' 시스템에서 '회원' 액터가 '상품주문' 유스케이스와 연관이 있으므로, "회원은 상품주문이라는 목적을 달성하기 위해 상품주문 시스템과 상호 작용한다."라고 해석한다.

포함(Include) 관계

- 두 개 이상의 유스케이스에 공통적으로 적용되는 기능을 별도로 분리하여 새로운 유스케이스로 만든 경우, 원래의 유스케이스와 새롭게 분리된 유스케이스와의 관계를 포함 관계라고 한다.
 - 원래의 유스케이스에서 새롭게 만든 포함되는 유스케이스 쪽으로 점선 화살표를 연결한 후 화살표 위에 《include》라고 표기한다.
 - (해석) : '상품주문' 시스템에서 '상품주문', '배송조회', '리뷰작성' 유스케이스가 '로그인' 유스케이스와 포함 관계에 있으므로, "회원은 상품주문, 배송조회, 리뷰작성을 수행하기 위해서는 로그인을 수행해야 한다."라고 해석한다.
-

확장(Extend) 관계

- 유스케이스가 특정 조건에 부합되어 유스케이스의 기능이 확장될 때 원래의 유스케이스와 확장된 유스케이스와의 관계를 확장 관계라고 한다.
- 확장될 유스케이스에서 원래의 유스케이스 쪽으로 점선 화살표를 연결한 후 화살표 위에 《extends》라고 표기한다.
- (해석) : '상품주문' 시스템에서 '리뷰작성' 유스케이스가 '사진 업로드' 유스케이스와 확장 관계에 있으므로, "회원은 리뷰작성을 수행하는 도중에 경우에 따라 사진 업로드를 수행한다."라고 해석한다.

일반화(Generalization) 관계

- 유사한 액터나 유스케이스를 하나의 그룹으로 묶고 싶을 때 그보다 일반적인 액터나 유스케이스를 만들어 이들을 연결하여 표현하는 관계를 일반화 관계라고 한다.
- 하위 액터나 유스케이스가 상위 액터나 유스케이스로 일반화되는 관계는 상위 액터나 유스케이스가 하위 액터나 유스케이스로 구체화되는 관계라고도 할 수 있다.
- 하위 액터나 유스케이스가 상위 액터나 유스케이스에게 역할이나 기능을 상속(Inheritance)받는 관계이다.
- 하위 액터나 유스케이스에서 상위 액터나 유스케이스 쪽으로 속이 빈 삼각형 화살표를 실선으로 연결한다.
- (해석1) : '상품주문' 시스템에서 '고객' 액터가 '회원', '비회원' 액터와 일반화 관계에 있으므로, "고객의 종류에는 회원, 비회원이 있다."라고 해석한다.
- (해석2) : '상품주문' 시스템에서 '상품조회' 유스케이스가 '이름으로 조회', '브랜드로 조회' 유스케이스와 일반화 관계에 있으므로, "상품조회는 이름으로 조회, 브랜드로 조회가 있다."라고 해석한다.

4. 유스케이스 명세서(기술서)

유스케이스 명세서는 유스케이스 안에서의 액터와 시스템 간의 상호작용 과정을 글로 자세히 표현한 것이다.

- 유스케이스 다이어그램에 있는 모든 유스케이스에 대해 개별적으로 작성해야 한다.
- * 유스케이스 명세서는 유스케이스 다이어그램에 있는 모든 유스케이스에 대해 개별적으로 작성하지만 유스케이스 다이어그램의 구성 요소는 아니다.
- 각 유스케이스 명세서에 작성된 사건의 흐름을 참고하여 활동(Activity) 다이어그램을 작성한다.
- 다음은 '상품주문'에 대한 유스케이스 명세서이다.

유스케이스명	상품주문
액터명	• 주액터 : 회원 • 부액터 : 재고 시스템, 결제 시스템
목표(개요)	회원이 상품을 주문하는 과정에서 재고 시스템과 결제 시스템을 이용한다. ※ 액터가 시스템을 통해 얻으려는 목적을 설명한다.
시작 조건	상품 주문을 하려면, 회원 가입이 되어 있어야 한다. ※ 유스케이스가 실행되기 위해 사전에 만족되어야 할 조건이 있다면 작성한다.
이후 조건	상품 주문이 정상적으로 완료되면, 상품 주문이 완료되었다는 메시지를 화면에 표시한다. ※ 유스케이스 실행이 완료되면 처리되어야 할 조건이 있다면 작성한다.
정상적인 흐름	1. 회원은 메인 화면에서 로그인을 클릭한다. 2. 로그인 창에서 회원 정보를 입력한 후 확인 단추를 클릭한다. 3. 로그인이 완료되면 주문할 상품을 선택한다. 4. 선택된 상품에 대한 재고를 확인한다. 5. 선택된 상품에 대한 결제를 진행한다. 6. 주문을 완료한다.
대체(대안) 흐름	2A : 입력된 정보가 잘못된 경우 1. 회원정보를 다시 입력하도록 요청하는 메시지를 표시한다. 4A : 선택된 상품의 재고가 없는 경우 1. 사용자에게 재고가 없으니 다른 상품을 선택하도록 요청하는 메시지를 표시한다. 5A : 선택된 상품에 대한 결제가 실패한 경우 1. 다시 한 번 결제 인증을 시도한다. 2. 재시도한 결제 인증이 성공하면 주문을 완료한다. 3. 재시도한 결제 인증이 실패하면 인증 실패로 상품을 주문할 수 없다는 메시지를 표시한다. ※ 정상적인 흐름으로부터 예외 사항이 생긴 경우에 예외 사항을 처리하는 흐름을 기술한다.

차시	9/10	콘텐츠 회차명	분석모델 타당성 분석	원격수업 시간	32분	재량활동 시간	18분	예제제출 (O,X)	
----	------	---------	-------------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 활동(Activity) 다이어그램】

1. 활동(Activity) 다이어그램의 개념

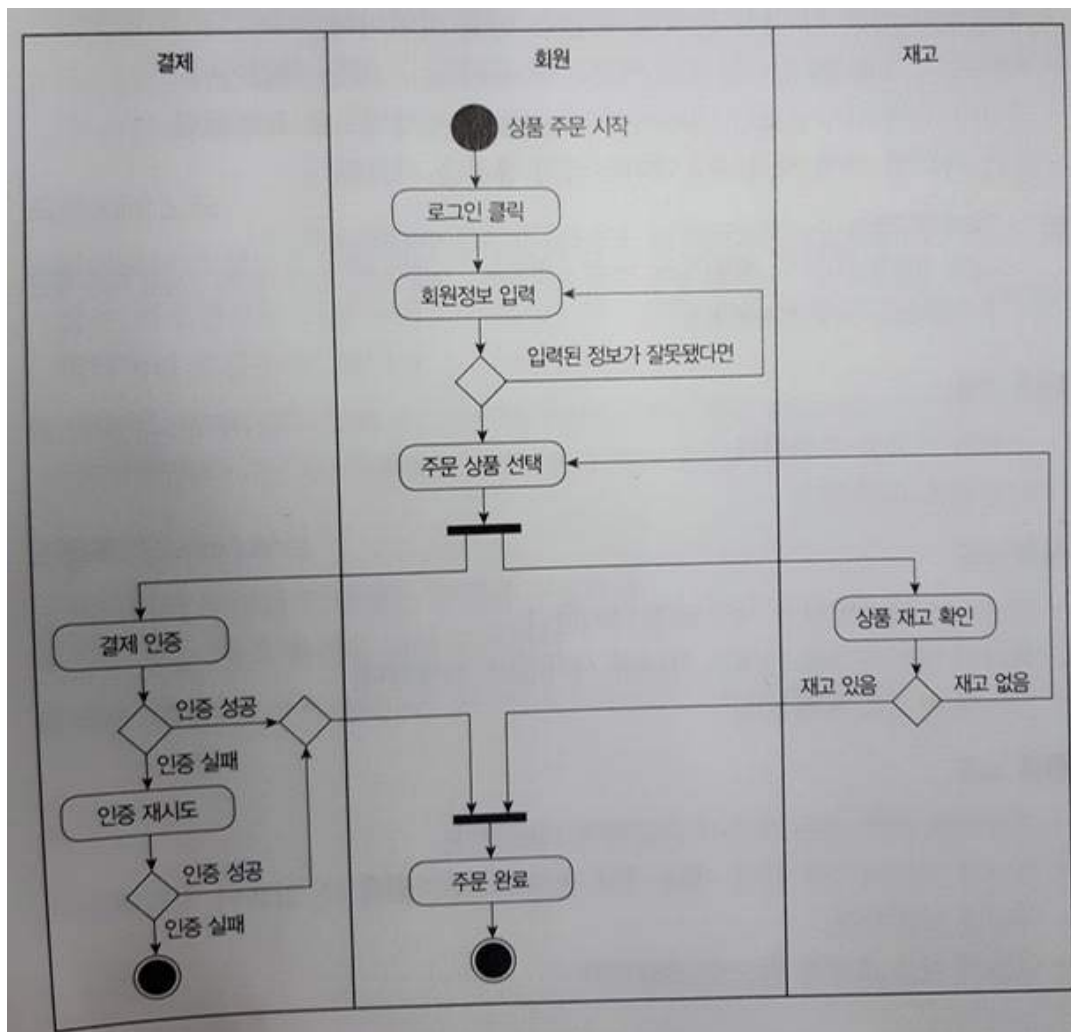
활동 다이어그램은 자료 흐름도와 유사한 것으로, 사용자의 관점(View)에서 시스템이 수행하는 기능을 처리 흐름에 따라 순서대로 표현한 것이다.

- 활동 다이어그램은 하나의 유스케이스 안에서 혹은 유스케이스 사이에 발생하는 복잡한 처리의 흐름을 명확하게 표현할 수 있다.

2. 활동(Activity) 다이어그램의 구성 요소

활동 다이어그램은 액션, 액티비티, 노드, 스왐레인 등으로 구성된다.

(예제) 다음은 회원의 상품 주문 과정에 결제 시스템과 재고 시스템이 관계되어 발생하는 처리의 흐름을 표현한 활동 다이어그램이다.



(해석)

〈회원〉 액터

- 회원이 상품을 주문하기 위해 로그인 단추를 클릭한 후 회원 정보를 입력한다.
- 입력된 정보가 잘못됐으면 회원 정보를 다시 입력받고, 그렇지 않으면 '주문 상품 선택'으로 이동한다.
- 회원이 주문할 상품을 선택하면 흐름이 '결제 인증'과 '상품 재고 확인'으로 분할되어 진행된다.
- 상품 주문을 완료하고 액티비티를 종료한다.

〈결제〉 시스템

- 주문한 상품에 대한 결제 인증을 진행한다.
- 인증에 성공하면 '주문 완료'로 이동하고, 인증에 실패하면 '인증 재시도'로 이동한다.
- 다시 진행한 결제 인증이 성공하면 '주문 완료'로 이동하고, 이번에도 인증이 실패하면 결제 인증에 대한 처리 흐름을 종료한다.

〈재고〉 시스템

- 주문한 상품에 대한 재고를 확인한다.
- 재고가 있으면 액티비티의 흐름을 '주문 완료'로 이동하고, 재고가 없으면 회원이 다른 상품을 선택할 수 있도록 '주문 상품 선택'으로 이동한다.

액션(Action) / 액티비티(Activity)

- 액션(Action)은 더 이상 분해할 수 없는 단일 작업이다.
- 액티비티(Activity)는 몇 개의 액션으로 분리될 수 있는 작업이다.
- 액션이나 액티비티 모두 테두리가 있는 둥근 사각형으로 표현한다.
- 둥근 사각형 안에 액션이나 액티비티의 명칭을 기술한다.
- (예제)에서는 '로그인 클릭'과 '주문 완료'가 액션(Action)에 해당된다.
- (예제)에서는 '회원정보 입력', '주문 상품 선택', '결제 인증', '인증 재시도', '상품 재고 확인'이 액티비티(Activity)에 해당된다.

제어 흐름

- 실행의 흐름을 표현한다.
- 화살표로 표현한다.

시작 노드

- 액션이나 액티비티가 시작됨을 의미한다.
- 하나의 다이어그램 안에는 하나의 시작점만 존재한다.
- 검은색 원으로 표현한다.

종료 노드

- 액티비티 안의 모든 흐름이 종료됨을 의미한다.
- 하나의 다이어그램 안에 여러 개의 종료 노드가 있을 수 있으나 일반적으로 하나만 표현한다.
- 검은색 원을 포함한 원으로 표현한다.

조건(판단) 노드

- 조건에 따라 제어의 흐름이 분리됨을 표현한다.
- 마름모로 표현하며, 들어오는 제어 흐름은 한 개이고 나가는 제어 흐름은 여러 개이다.
- 〈회원〉이 '회원정보 입력'에서 입력한 회원 정보가 맞으면 '주문 상품 선택'으로 이동하고, 정보가 틀리면 회원이 회원정보를 다시 입력할 수 있도록 '회원정보 입력'으로 이동한다.

병합 노드

- 여러 경로의 흐름이 하나로 합쳐짐을 표현한다.
- 마름모로 표현하며, 들어오는 제어 흐름은 여러 개이고 나가는 제어 흐름은 한 개다.
- 〈결제〉 시스템에서 결제 인증에 성공하여 '주문 완료'로 이동하기 위한 흐름과 초기 결제 인증이 실패하여 인증 재시도로 결제 인증에 성공하여 '주문 완료'로 이동하기 위한 흐름이 하나로 합쳐진다.

포크(Fork) 노드

- 액티비티의 흐름이 분리되어 수행됨을 표현한다.
 - 굵은 가로선으로 표현하며, 들어오는 액티비티 흐름은 한 개이고 나가는 액티비티 흐름은 여러 개이다.
 - '주문 상품 선택'에서 주문할 상품이 선택되면 상품에 대한 결제 인증과 상품에 대한 재고 확인이 수행되어야 하므로 액티비티 흐름이 '결제 인증'과 '상품 재고 확인'으로 분할된다.
-

조인(Join) 노드

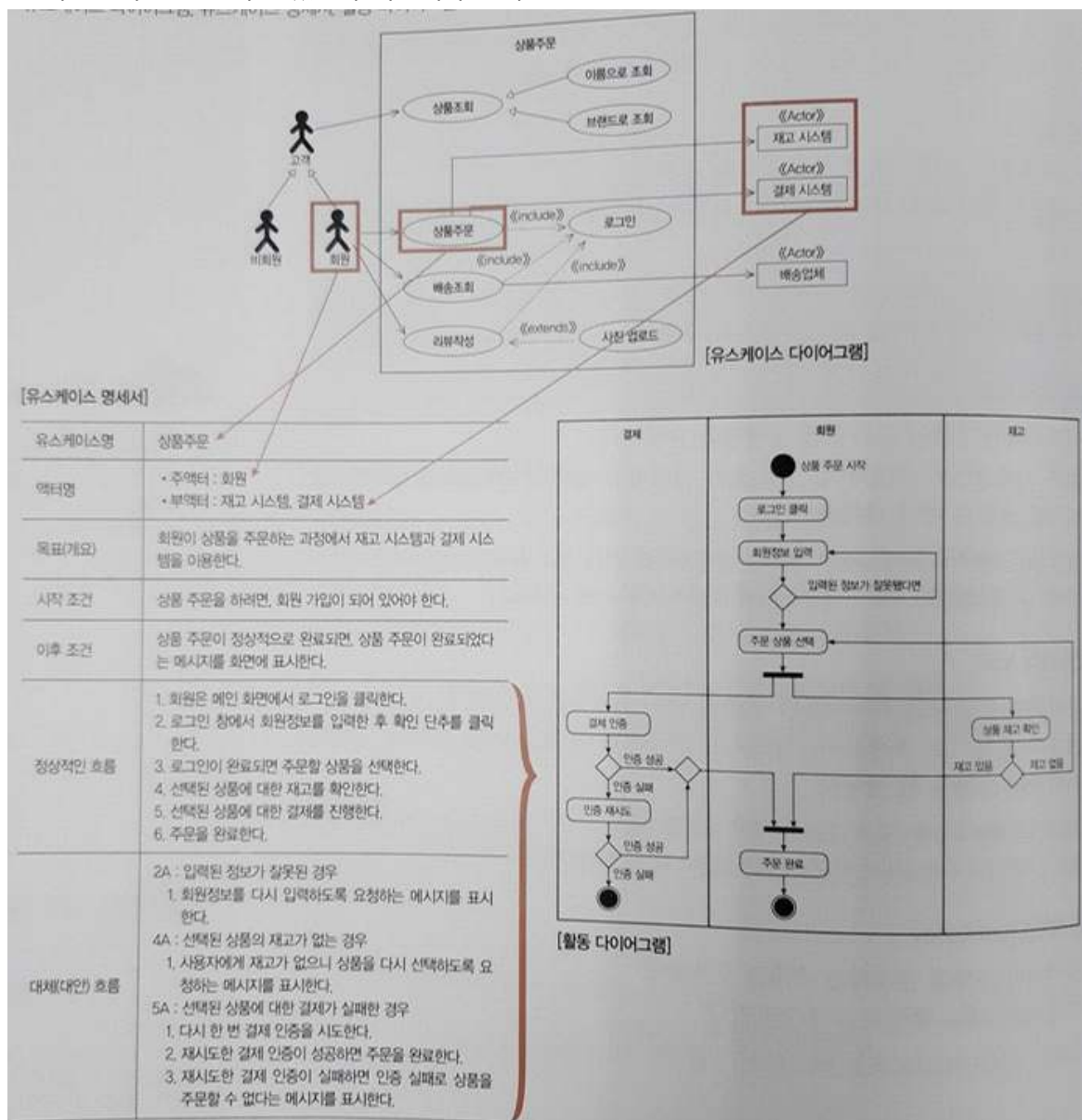
- 분리되어 수행되던 액티비티의 흐름이 다시 합쳐짐을 표현한다.
- 굵은 가로선으로 표현하며, 들어오는 액티비티 흐름은 여러 개이고 나가는 액티비티 흐름은 한 개이다.
- <결제> 시스템에서 결제 인증에 성공하여 '주문 완료'로 이동하기 위한 흐름과 <재고> 시스템에서 상품 재고가 확인되어 '주문 완료'로 이동하기 위한 흐름이 하나로 합쳐진다.

스вим레인(Swim Lane)

- 액티비티 수행을 담당하는 주체를 구분한다.
- 가로 또는 세로 실선을 그어 구분한다.
- <결제>, <회원>, <재고> 시스템을 서로 구분하기 위한 세로 선이 스вим레인에 해당된다.

◆ 기능적 모델 검증

기능 모델링이 끝나고 다음 단계인 구조 모델링이나 동적 모델링으로 넘어가기 전에 기능 모델링이 제대로 작성되었는지 확인할 필요가 있다. 유스케이스 다이어그램, 유스케이스 명세서, 활동 다이어그램이 모두 같은 기능 요구를 표현하고 있는지 확인해야 한다.



※ '상품주문' 유스케이스에 대한 유스케이스 명세서의 정상적인 흐름과 대체 흐름을 기반으로 활동 다이어그램이 작성되었음을 확인할 수 있다.

차시	10/10	콘텐츠 회차명	요구사항 추적서 명세화	원격수업 시간	31분	재량활동 시간	19분	예제제출 (O,X)	
----	-------	---------	--------------	------------	-----	------------	-----	---------------	--

【보충학습자료 : 클래스(Class) 다이어그램】

1. 정적 모델링의 개념

정적 모델링은 사용자가 요구한 기능을 구현하는데 필요한 자료들의 논리적인 구조를 표현한 것이다.

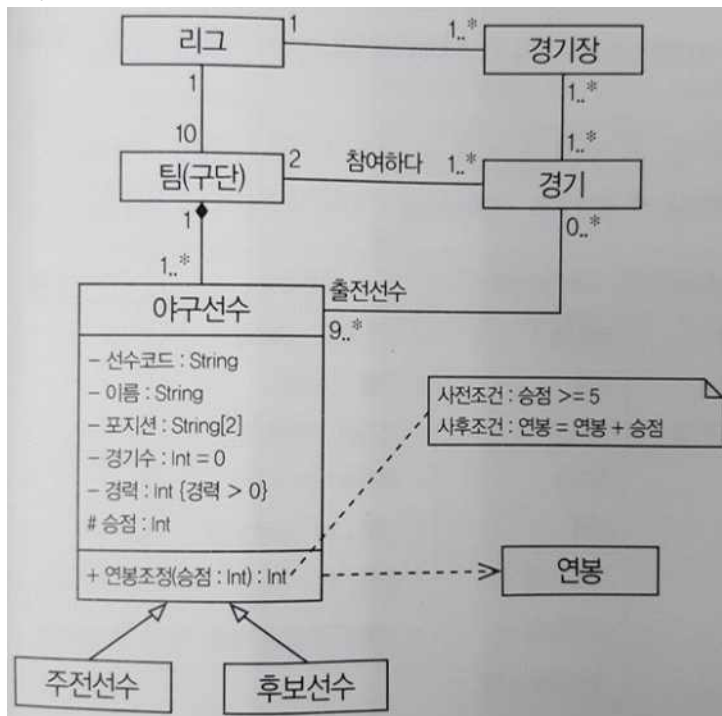
- 정적 모델링은 시스템에 의해 처리되거나 생성될 객체들 사이에 어떤 관련이 있는지를 구조적인 관점(View)에서 표현한다.
- 정적 모델링은 객체(Object)들을 클래스(Class)로 추상화하여 표현한다.
- UML을 이용한 정적 모델링의 대표적인 것이 클래스 다이어그램이다.

2. 클래스 다이어그램의 개념

클래스 다이어그램은 시스템을 구성하는 클래스, 클래스의 특성인 속성과 오퍼레이션, 속성과 오퍼레이션에 대한 제약조건, 클래스 사이의 관계를 표현한 것이다.

- 클래스 다이어그램은 시스템을 구성하는 요소에 대해 이해할 수 있는 구조적 다이어그램이다.
- 클래스 다이어그램은 시스템 구성 요소를 문서화하는 데 사용된다.
- 코딩에 필요한 객체의 속성, 함수 등의 정보를 잘 표현하고 있어 시스템을 모델링하는 데 자주 사용된다.
- 클래스 다이어그램은 클래스, 제약조건, 관계 등으로 구성된다.

(예제) 다음은 프로야구 리그에 필요한 정보의 일부를 표현한 클래스 다이어그램이다.



(해석)

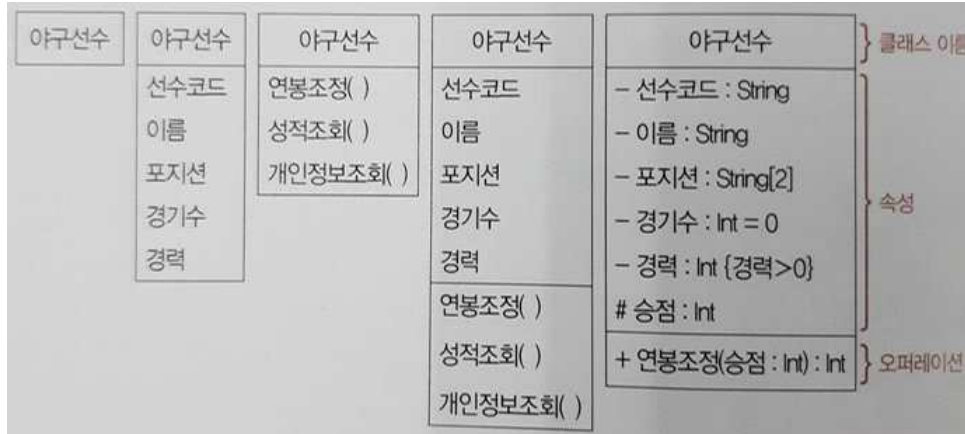
- 리그에는 10개의 팀이 참가한다.
- 리그는 한 개 이상의 경기장에서 경기가 진행된다.
- 매 경기는 두 팀이 진행하며, 두 팀은 한 번 이상의 경기를 진행한다.
- 한 팀에는 한 명 이상의 선수가 있다.
- 한 경기장에서는 한 경기 이상이 진행되며, 경기는 여러 경기장에서 진행된다.
- 한 팀에서 최소 9명 이상의 선수가 한 경기에 출전하며, 경기에 출전하지 못하는 선수도 있고 여러 경기에 출전하는 선수도 있다.
- 선수는 주전선수와 후보선수가 있다.
- 선수의 경력은 0보다 큰 값이 입력되어야 한다.
- 승점이 5 이상인 선수들은 연봉을 조정할 수 있으며, 연봉은 승점만큼 증가한다.

3. 클래스(Class)

클래스는 각각의 객체들이 갖는 속성과 오퍼레이션(동작)을 표현한다.

- 일반적으로 3개의 구획(Compartment)으로 나눠 클래스의 이름, 속성, 오퍼레이션을 표기한다.
- 속성이나 오퍼레이션은 생략할 수 있지만 이름은 반드시 명시해야 한다.
- 속성이나 오퍼레이션이 생략된 경우에는 구획선을 그리지 않아도 된다.

- 클래스의 다양한 표현



- 야구선수 클래스의 Java 코드 표현

```
class 야구선수 {
    private String 선수코드;
    private String 이름;
    private String[] 포지션 = new String[2];
    private int 경기수 = 0;
    private int 경력;
    protected int 승점;

    public int 연봉조정(int 승점) { ... }
}
```

속성(Attribute)

속성은 클래스의 상태나 정보를 표현한다.

- 일반 형식

[접근제어자] 속성명 : 자료형 [다중성] [=초기값]

- 접근제어자 : 속성과 오퍼레이션을 어느 정도까지 클래스 외부에 노출시킬지를 제어한다.
- 속성명 : 속성의 이름으로, 사용자가 임의로 작성한다.
- 자료형 : UML에서 기본적으로 제공하는 자료형 또는 사용자가 필요에 의해 새롭게 정의한 자료형을 사용할 수 있다. 사용할 프로그램 언어가 지정된 경우 해당 프로그램 언어에서 제공하는 자료형을 사용할 수 있다.
- 다중성 : 동일한 속성명으로 여러 개의 속성 값을 가질 수 있는 것으로, 배열과 같은 의미이다.
- 초기값 : 데이터를 입력하지 않았을 때 기본적으로 입력되는 값을 지정한다.
- '야구선수' 클래스의 속성
 - 속성에 접근제어자를 private과 protected로 설정했다.
 - '포지션'은 공격 포지션과 수비 포지션으로 구분하기 위해 다중성을 2로 지정했다.
 - '경기수' 속성은 선수들이 경기를 출전하기 전에는 경기수가 모두 0이므로 초기값을 0으로 지정했다.

접근제어자

접근제어자는 속성과 오퍼레이션에 동일하게 적용되며, 표현법은 다음과 같다.

접근제어자	표현법	내용
public	+	어떤 클래스에서라도 접근이 가능하다.
private	-	해당 클래스 내부에서만 접근이 가능하다.
protected	#	동일 패키지 내의 클래스 또는 해당 클래스를 상속 받은 외부 패키지의 클래스에서 접근이 가능하다.
package	~	동일 패키지 내부에 있는 클래스에서만 접근이 가능하다.

자료형

UML에서 기본적으로 제공하는 자료형은 다음과 같다.

종류	자료형
문자형	String
정수형	Integer, Int
자연수	UnlimitedNatural
논리형	Boolean
실수형	Real, Float

오퍼레이션(Operation, 연산)

오퍼레이션은 클래스가 수행할 수 있는 동작으로, 함수(메소드, Method)라고도 한다.

• 일반 형식

[접근제어자] 오퍼레이션명(매개변수1 : 자료형1, 매개변수2 : 자료형2, ...) : 반환자료형

- 오퍼레이션명 : 오퍼레이션의 이름으로, 사용자가 임의로 작성한다.
- 매개변수 : 오퍼레이션으로 어떤 작업을 할 때 해당 오퍼레이션 수행에 필요한 값을 전달하기 위해 사용된다.
- 반환자료형 : 오퍼레이션 수행 후 반환되는 값에 대한 자료형으로, 반환되는 값이 없으면 반환자료형을 void로 지정한다.
- '야구선수' 클래스의 '연봉조정' 오퍼레이션
 - 어떤 클래스에서라도 접근이 가능하도록 접근제어자를 public으로 지정했다.
 - 매개변수 '승점'에 정수형 자료를 담아 전달하면, 조정된 연봉이 정수형 자료로 반환된다.

4. 제약조건

속성에 입력될 값에 대한 제약조건이나 오퍼레이션 수행 전후에 지정해야 할 조건이 있다면 이를 적는다.

- 주석(Note) 도형 안에 제약조건을 적은 후 제약조건이 적용될 속성이나 오퍼레이션을 점선으로 연결한다.
- 클래스 안에 제약조건을 적을 때는 중괄호 { }를 이용한다.

5. 관계(Relationships)

관계는 클래스와 클래스 사이의 연관성을 표현한다.

- 클래스 다이어그램은 클래스가 서로 연결되어 있음을 의미하는 연관 관계를 기본으로 한다.
- 관계에 참여하는 객체의 수(다중도)를 연관 관계 선 위에 표기한다.
- 클래스 다이어그램에 표현하는 관계에는 연관 관계, 집합 관계, 포함 관계, 일반화 관계, 의존 관계가 있다.

연관(Association) 관계

- 두 클래스 간의 관계를 명확하게 표현하기 위해 관계 표현 실선의 중간 지점에 관계의 이름을 표기할 수 있다.
 - '팀(구단)'이 '경기'에 참여하는 관계에 있다는 것을 '참여하다'는 이름으로 좀 더 명확하게 알 수 있다.
- 해당 클래스의 관계 표현 실선 바로 옆에 클래스의 역할을 표기할 수 있다.
 - '야구선수' 클래스가 출전선수 역할을 한다는 것을 '야구선수' 클래스 옆에 작성된 역할로 알 수 있다.
- 연관 관계에서 추가적으로 표현해야 할 속성이나 오퍼레이션이 있는 경우 연관 클래스를 사용한다.
- 두 클래스가 서로 연관 관계에 있을 때는 클래스 안에 연관된 클래스를 이용하여 객체 변수를 생성할 수 있다.

(예제1) 야구선수가 유니폼을 소유하는 관계이다. 야구선수는 자기가 소유하고 있는 유니폼에 대해 알고 있지만 유니폼은 누구에 의해 자신이 소유되고 있는지 모른다는 의미이다.



```
public class Player {
    // Uniform 클래스를 이용하여 객체 변수 theUniform을 생성한다.
    private Uniform theUniform;
}

public class Uniform {
}
```

(코드 해설)

Player 클래스는 Uniform 클래스를 이용하여 객체 변수 theUniform을 생성하였지만 Uniform 클래스는 Player 클래스를 이용하지 않는다. 즉 Player 클래스는 Uniform 클래스를 알고 있지만 Uniform 클래스는 Player 클래스를 알지 못하므로 한 쪽 방향으로만 관계가 있다.

(예제2) 야구선수는 경기에 출전하고 경기는 야구선수들에게 경기 결과를 제공하므로 야구선수와 경기는 서로 관계가 있다.



```
public class Player {
    // Game 클래스를 이용하여 객체 변수 theGame을 생성한다.
    private Game theGame;
}

public class Game {
    // Player 클래스를 이용하여 객체 변수 thePlayer를 생성한다.
    private Player thePlayer;
}
```

(코드 해설)

Player 클래스는 Game 클래스를 이용하여 객체 변수 theGame을 생성하고, Game 클래스도 Player 클래스를 이용하여 객체 변수 thePlayer를 생성한다. 즉 두 클래스는 서로를 알고 있으므로 서로 관계가 있으며, 서로 사용할 수 있다.

집합(Aggregation) 관계

두 클래스가 서로 집합 관계에 있을 때는 집합 관계에 있는 클래스의 객체 변수를 매개변수로 사용할 수 있다.

(예제) 야구선수에게는 팬이 있으며, 팬은 다른 야구선수의 팬이 될 수도 있다.



```
public class Player {
    private Pan thePan;

    public Player(Pan thePan) {
        this.thePan = thePan;
    }
}

public class Pan { ... }
```

(코드 해설)

Player 클래스의 Player() 메소드는 Pan 클래스의 객체 변수 thePan을 매개변수로 사용하고 있다. 이와 같이 외부 클래스의 객체 변수에 대한 주소만 받아 사용하는 관계를 집합 관계라고 한다.

포함(Composition) 관계

두 클래스가 서로 포함 관계에 있을 때는 포함 관계에 있는 클래스를 이용하여 생성된 객체 변수를 이용하여 새로운 객체 변수를 생성할 수 있다.

(예제) 야구선수는 하나의 팀(구단)에 소속되며, 팀(구단)이 해체되면 해당 야구선수도 더 이상 필요하지 않다.



```
public class Team {
    private Player thePlayer;

    public Team() {
        this.thePlayer = new Player();
    }
}

public class Player { ... }
```

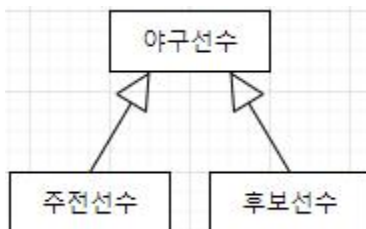
(코드 해설)

Team 클래스의 Team() 메소드는 Player 클래스를 이용하여 생성한 객체 변수 thePlayer를 이용하여 Player 객체를 포함하고 있다. 이와 같은 관계를 포함 관계라고 한다.

일반화(Generalization) 관계

- 일반적인 개념을 상위(부모) 클래스, 구체적인 개념을 하위(자식) 클래스라고 부른다.
- 두 클래스가 서로 일반화 관계에 있을 때는 하위 클래스가 상위 클래스의 속성이나 메소드를 사용할 수 있다.

(예제) 주전선수와 후보선수는 야구선수이다. 다시 말하면, 야구선수에는 주전선수와 후보선수가 있다.



```
public class Player {
    ...
}

public class M_Player extends Player {
    ...
}

public class C_Player extends Player {
    ...
}
```

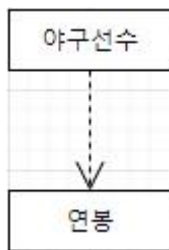
(코드 해설)

M_Player와 C_Player 클래스가 Player 클래스를 extends, 즉 상속받는 관계이다. 이와 같이 표현된 경우 M_Player와 C_Player 클래스를 하위(자식) 클래스라고 하고 Player 클래스를 상위(부모) 클래스라고 한다.

의존(Dependency) 관계

두 클래스가 의존 관계에 있을 때는 영향을 주는 클래스(이용자)의 특정 오퍼레이션(메소드)이 수행될 때만 영향을 받는 클래스(제공자)가 사용된다.

(예제) 야구선수의 승점이 높으면 연봉을 조정하고, 승점이 낮으면 연봉을 조정하지 않는다.



```
public class Player {  
    private int point;  
  
    public void findSalary(SalaryCount s) {  
        s.getSalary(point);  
    }  
}  
  
public class SalaryCount { ... }
```

(코드 해설)

SalaryCount 클래스는 Player 클래스의 findSalary() 메소드가 수행되는 경우에만 사용된다. 이와 같이 필요에 의해 짧은 시간 동안만 연관을 유지하는 관계를 의존 관계라고 한다.

◆ 연관 클래스

연관 클래스는 두 클래스가 연관 관계에 있을 때 추가적으로 표현해야 할 속성이나 오퍼레이션이 있는 경우 사용된다.

- 두 클래스의 연관 관계를 나타내는 선의 가운데로부터 점선을 연관 클래스로 이어 표시한다.
- 연관 클래스의 이름은 연관 관계의 이름을 이용해 지정한다.

(예) 팀이 경기에 참여하는 상황에서 참여횟수와 참여결과 속성을 어디에 두어야 할지를 생각해 보자. 먼저 '팀' 클래스에 참여횟수와 참여결과 속성을 둔다면 팀이 어느 경기에 참여한 것인지 모호해진다. 또한 '경기' 클래스에 참여횟수와 참여결과 속성을 둔다면 어느 팀이 참여한 것인지 모호해진다. 이런 경우 '팀'이나 '경기' 클래스의 속성이 아닌 '참여하다'라는 연관 관계에 대한 연관 클래스를 만들어 '참여횟수'와 '참여결과' 속성을 연관 클래스의 속성으로 다루면 된다.

