

대분류 / 20
정보통신

중분류 / 01
정보기술

소분류 / 02
정보기술개발

세분류 / 02
응용SW엔지니어링

학습모듈 / 02

02

애플리케이션 설계

LM2001020202_14v2

응용SW엔지니어링 학습모듈

01. 요구사항 확인



02. 애플리케이션 설계



03. 애플리케이션 구현



04. 화면 구현



05. 데이터 입출력 구현



06. 통합 구현



07. 개발자 테스트



08. 정보시스템 이행



09. 제품소프트웨어 패키징



10. 소프트웨어공학 활용

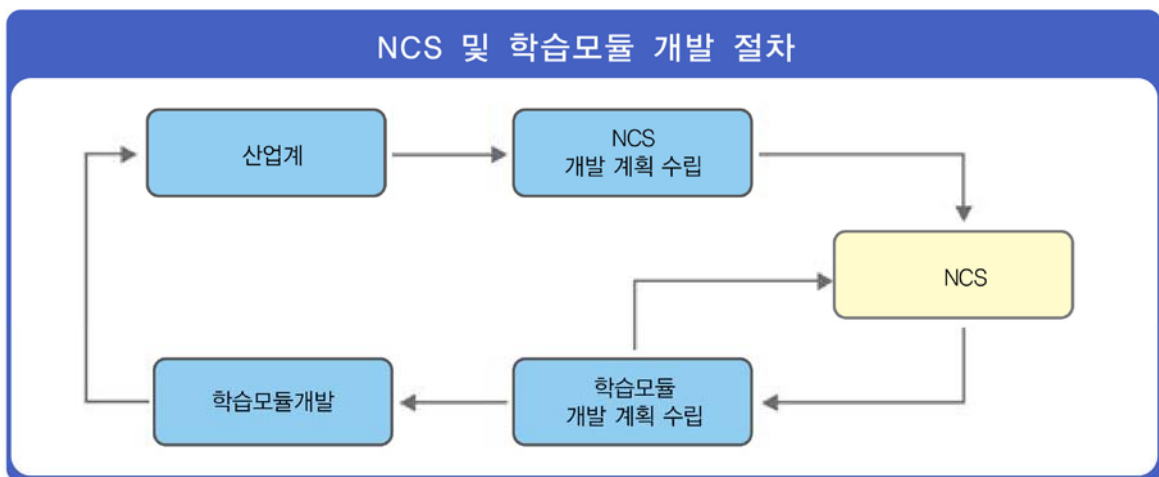


NCS 학습모듈의 이해

※ 본 학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>) 에서 확인 및 다운로드 할 수 있습니다.

(1) NCS 학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS 학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS 학습모듈은 구체적인 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.

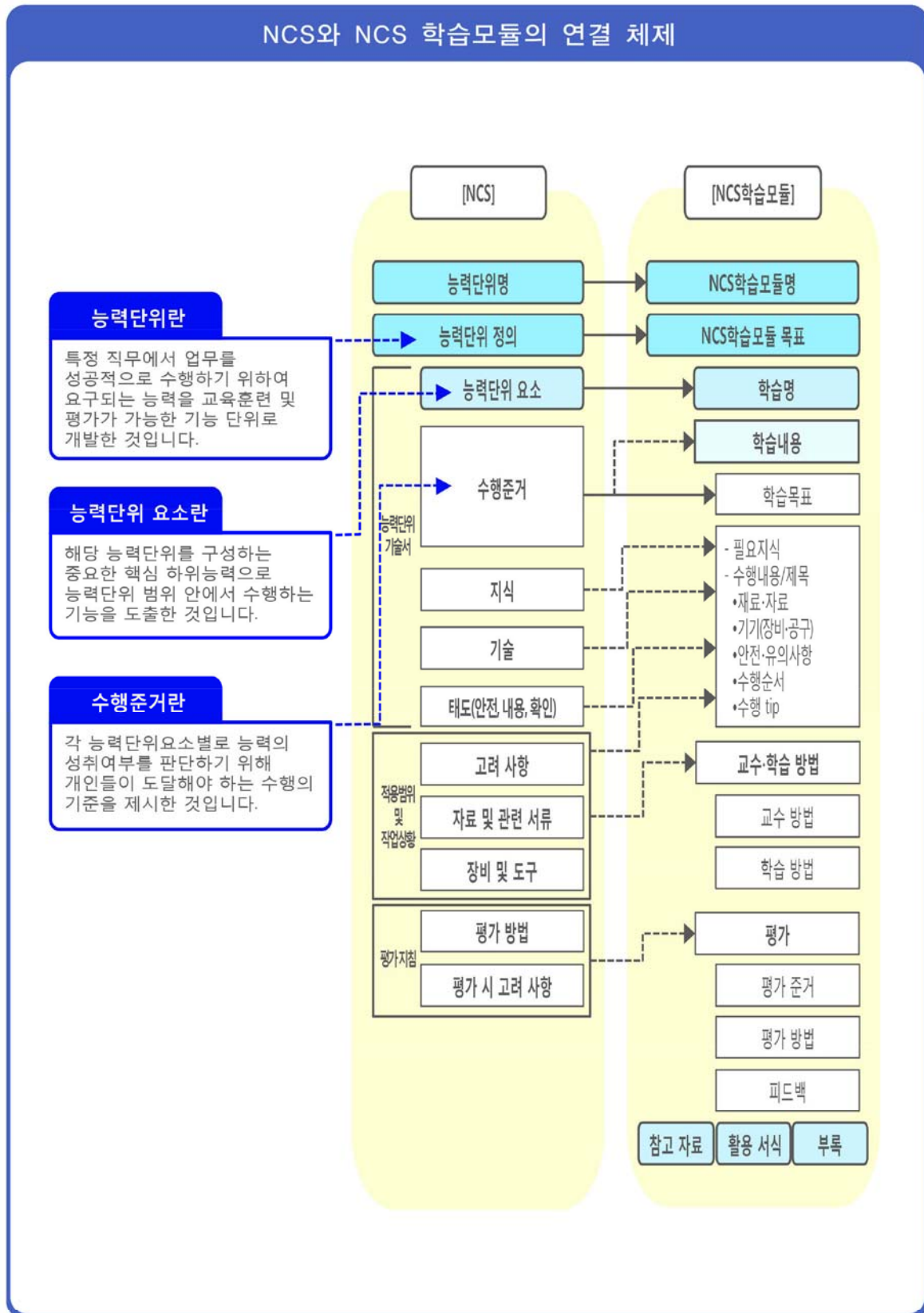


- NCS 학습모듈은 다음과 같은 특징을 가지고 있습니다.

첫째, NCS 학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.

둘째, NCS 학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS 학습모듈 간의 연결 체제를 살펴보면 아래 그림과 같습니다.



(2) NCS 학습모듈의 체계

- NCS 학습모듈은 1.학습모듈의 위치, 2.학습모듈의 개요, 3.학습모듈의 내용 체계, 4.참고 자료, 5.활용 서식/부록 으로 구성되어 있습니다.

1. NCS 학습모듈의 위치

- NCS 학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

예시 : 이·미용 서비스 분야 중 네일미용 세분류

NCS-학습모듈의 위치

대분류	이용·숙박·여행·오락·스포츠
중분류	이·미용
소분류	아·미용 서비스

세분류	능력단위	학습모듈명
헤어미용	네일 샵 위생 서비스	네일샵 위생서비스
피부미용	네일 화장을 제거	네일 화장을 제거
메이크업	네일 기본 관리	네일 기본관리
네일미용	네일 랩	네일 랩
이용	네일 팁	네일 팁
	젤 네일	젤 네일
	아크릴릭 네일	아크릴 네일
	평면 네일아트	평면 네일아트
	융합 네일아트	융합 네일아트
	네일 샵 운영관리	네일샵 운영관리

학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용 단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어서 1개의 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습모듈로 나누어 개발할 수도 있습니다.

2. NCS 학습모듈의 개요

구 성

- NCS 학습모듈 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로서 **학습모듈의 목표**, **선수 학습**, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

학습모듈의 목표	해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다.
선수 학습	해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다.
학습모듈의 내용 체계	해당 NCS 능력단위요소가 학습모듈에서 구조화된 방식을 제시한 것입니다.
핵심 용어	해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다.

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈

네일 기본관리 학습모듈의 개요

학습모듈의 목표

고객의 네일 보호와 미적 요구 충족을 위하여 효과적인 네일 관리로 프리에지 형태 만들기, 큐티클 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

선수학습

네일숍 위생서비스(JM1201010401_14v2)

학습모듈의 내용체계

학습	학습내용	NCS 능력단위요소		
		코드번호	요소명칭	수준
1. 프리에지 형태 만들기	1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링	1201010403_12v2.1	프리에지 모양 만들기	3
2. 큐티클 정리하기	2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티클 관리	1201010403_14v2.2	큐티클 정리하기	3
3. 컬러링하기	3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업 3-3. 컬러링 작업	1201010403_14v2.3	컬러링	3
4. 보습제 도포하기	4-1. 보습제 선정과 도포 4-2. 각질제거	1201010403_14v2.4	보습제 바르기	2
5. 네일 기본관리 마무리하기	5-1. 유분기 제거 5-2. 네일 기본관리 마무리와 정리	1201010403_14v2.5	마무리하기	3

핵심 용어

프리에지, 니퍼, 푸시, 플리시, 네일 파일, 스웨이형, 스웨이 오프형, 라운드형, 오발형, 포인트형

학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악할 수 있도록 지도하는 것이 필요합니다.

선수학습은

교수자나 학습자가 해당 모듈을 교수 또는 학습하기 이전에 이수해야 할 학습내용, 교과목, 핵심 단어 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것이 필요합니다.

핵심 용어는

학습모듈을 통해 학습되고 평가되어야 할 주요 용어입니다. 또한 당해 모듈 또는 타 모듈에서도 핵심 용어를 사용하여 학습내용을 구성할 수 있으며, 「NCS 국가 직무능력표준」 사이트(www.ncs.go.kr)에서 색인(찾아보기) 중 하나로 이용할 수 있습니다.

3. NCS 학습모듈의 내용 체계

구 성

- NCS 학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가** 로 구성되어 있습니다.

학습	해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다. 학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 학습 내용을 제시한 것입니다.
학습 내용	학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성하였으며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 업무의 표준화된 프로세스에 기반을 두고 실제 산업현장에서 이루어지는 업무활동을 다양한 방식으로 반영한 것입니다.
교수·학습 방법	학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간의 상호 작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.
평가	평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거, 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.

활 용 안 내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

학습 1	프리에지 형태 만들기(LM1201010403_14v2.1)
학습 2	큐티를 정리하기(LM1201010403_14v2.2)
학습 3	컬러링하기(LM1201010403_14v2.3)
학습 4	보습제 도포하기(LM1201010403_14v2.4)
학습 5	네일 기본관리 마무리하기(LM1201010403_14v2.5)

학습은

해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 학습은 일반교과의 '대단원'에 해당되며, 모듈을 구성하는 가장 큰 단위가 됩니다. 또한 완성된 직무를 수행하기 위한 가장 기본적인 단위로 사용할 수 있습니다.

학습내용은

요소 별 수행준거를 기준으로 제시하였습니다. 일반교과의 '중단원'에 해당합니다.

학습목표는

모듈 내의 학습내용을 이수했을 때 학습자가 보여줄 수 있는 행동수준을 의미합니다. 따라서 일반 수업시간의 과목목표로 활용할 수 있습니다.

필요지식은

해당 NCS의 지식을 토대로 해당 학습에 대한 이해와 성과를 높이기 위해 알아야 할 주요 지식을 제시하였습니다. 필요지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행순서 내용과 연계하여 교수·학습으로 진행할 수 있습니다.

3-1. 컬러링 매뉴얼 이해

학습목표

- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 얇게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시를 열매 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

필요 지식 /

□ 컬러링 매뉴얼

컬러링 작업 전, 이세론 또는 네일 폴리시 리무버를 사용하여 손톱표면과 큐티클 주변, 손톱 밑 부분까지 깨끗하게 유분기를 제거해야 한다. 컬러링의 순서는 Base coating 1회 → Polishing 2회 → 컬러수정 → Top coating 1회 → 최종수정의 순서로 한다. 베이스코트는 착색을 방지하고 발림성 향상을 위해 가장 먼저 도포하며 컬러링의 마지막에 컬러의 유지와 광택을 위해 톱코트를 도포한다. 네일 보강제(Nail Strengthner)를 바를 시에는 베이스코트를 도포하기 전에 사용한다.

수행 내용 / 컬러링 매뉴얼 실습하기

재료·자료

- 컬러링 관련 네일 미용 자료들
- 컬러바구니, 베이스코트, 네일 폴리시, 톱코트, 오렌지우드스티, 단자면, 폴리시리무버, 디스펜서 등

기기(장비·공구)

- 컴퓨터, 빔 프로젝터, 스크린 등

안전·유의사항

- 컬러링 재료들의 냄새를 직접적으로 맡지 않도록 유의한다.
- 컬러링 제품들이 대부분 유리병에 들어 있기 때문에 깨지지 않도록 각별히 조심한다.
- 컬러링 제품들은 상온에 마르기 때문에 개봉 후 뚜껑을 잘 닫도록 한다.

수행 순서

[1] 네일 폴리시를 바르게 짚는다.

1. 손바닥에 네일 폴리시를 놓고 약지 소지를 이용하여 네일 폴리시를 짚는다.
2. 폴리시를 왼 손의 엄지와 검지로 고객과 작업손가락을 짚는다.
3. 폴리시를 왼 손의 중지 손가락을 굳게 펴서 받침대가 되도록 한다.
4. 반대편 손으로 네일 폴리시의 뚜껑을 열고 소지 손가락을 펴서 네일 폴리시를 왼 중지 손가락 위에 받쳐놓는다.
5. 다양한 형태의 폴리시를 잡아본다.

수행 tip

- 흰색이 많이 섞인 네일 폴리시의 경우는 붓의 각도를 높이 세워 빠르게 브러시 작업을 해야 붓 자국이 나지 않는다.
- 컬러링은 기본 2회 정도이나 컬러에 따른 도포량과 컬러감에 따라 1~3회 사이로 증감할 수 있다.

수행 내용은

모듈에 제시한 것 중 기술(Skill)을 습득하기 위한 실습 과제로 활용할 수 있습니다.

재료·자료는

수행 내용을 수행하는데 필요한 재료 및 준비물로 실습 시 필요 준비물로 활용할 수 있습니다.

기기(장비·공구)는

수행 내용을 수행하는데 필요한 기본적인 장비 및 도구를 제시하였습니다. 제시된 기기 외에도 수행에 필요한 다양한 도구나 장비를 활용할 수 있습니다.

안전·유의사항은

수행 내용을 수행하는데 안전상 주의해야 할 점 및 유의사항을 제시하였습니다. 수행 시 유념해야 하며, NCS의 고려사항도 추가적으로 활용할 수 있습니다.

수행 순서는

실습과제의 진행 순서로 활용할 수 있습니다.

수행 tip은

수행 내용에서 수행의 수월성을 높일 수 있는 아이디어를 제시하였습니다. 따라서 수행tip은 지도상의 안전 및 유의사항 외에 전반적으로 적용되는 주안점 및 수행과제 목적에 대한 보충설명, 추가사항 등으로 활용할 수 있습니다.

학습3 교수·학습 방법

교수·학습 방법은

학습목표를 성취하는데 필요한 교수 방법과 학습 방법을 제시하였습니다.

교수 방법

- 컬러링 제품의 성분과 컬러별 점도의 차이, 베이스코트와 톱코트의 역할, 폴리시 짚는 방법, 큐어링 시간 등의 내용을 화면 자료와 함께 설명한다.
- 서식지를 활용하여 네일 컬러링 방법을 그림으로 그려 보게 한 뒤, 다양한 컬러링의 매뉴얼을 그려서 숙지하도록 한다.
- 겔 컬러링 시 주의사항을 계속 숙지시키도록 하며, 큐어링 시간에 대해 작성하도록 한다.

교수 방법은

해당 학습활동에 필요한 학습내용, 학습내용과 관련된 학습 자료명, 자료 형태, 수행내용의 진행 방식 등에 대하여 제시하였습니다. 또한 학습자의 수업참여도를 제고하기 위한 방법 및 수업진행상 유의사항 등도 제시하였습니다. 선수학습이 필요한 학습을 학습자가 숙지하였는지 교수자가 확인하는 과정으로 활용할 수도 있습니다.

학습 방법

- 컬러링을 위한 재료의 필요성과 사용방법을 숙지하고 컬러링 매뉴얼 과정에 맞추어 작업 내용을 이해한다.
- 컬러링의 다양성에 대한 용어를 숙지하고 진행과정에 맞추어 내용을 작업한다.
- 겔 컬러링 시 적합한 큐어링 시간을 선택해서 큐어링 해본다.

학습 방법은

해당 학습활동에 필요한 학습자의 자기주도적 학습 방법을 제시하였습니다. 또한 학습자가 숙달해야 할 실기능력과 학습과정에서 주의해야 할 사항 등으로 제시하였습니다. 학습자가 학습을 이수하기 전에 반드시 숙지해야 할 기본 지식을 학습하였는지 스스로 확인하는 과정으로 활용할 수 있습니다.

학습3 평가

평가 준거

- 평가자는 학습자가 학습 목표 및 평가 항목에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
컬러링 매뉴얼 이해	<ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얹어 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. 			

평가 방법

- 작업장 평가

학습내용	평가항목	성취수준		
		상	중	하
컬러링 매뉴얼 이해	<ul style="list-style-type: none"> 고객의 요구에 따라 네일 폴리시 색상의 질감을 만들기 위한 베이스코트를 아주 얇게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시를 일찍 얹어 균일하게 도포할 수 있다. 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다. 			

피드백

- 작업장 평가
 - 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

평가는

해당 NCS 능력단위 평가방법과 평가 시 고려 사항을 준용하여 작성하였습니다. 교수자 및 학습자가 평가항목 별 성취수준을 확인하는데 활용할 수 있습니다.

평가 준거는

학습자가 해당 학습을 어느 정도 성취하였는지를 평가하기 위한 기준을 제시하고 있습니다. 학습목표와 연계하여 단위수업 시간에 평가항목 별 성취수준을 평가하는데 활용할 수 있습니다.

평가 방법은

NCS 능력단위의 평가방법을 준용하였으며, 평가 준거에 따른 평가방법을 2개 이상 제시하였습니다. 평가방법으로는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS의 능력단위 요소 별 수행 수준을 평가하는데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 부족한 부분을 알려주고, 학습 결과가 미진한 경우, 해당 부분을 다시 학습하여 학습목표를 달성하는 데 활용할 수 있습니다.

4. 참고 자료

참고자료

- 김미원(2011). 『Nail Study』. 서울: 사)한국네일저서서비스협회.
- 민광경(2015). 『미용사(네일)평가』. 서울: 예문사.
- 박은주(2014). 『네일미용』. 서울: 정담미디어.

참고자료는


해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

5. 활용 서식/부록

활용서식

프리에지 형태 실습지

1. 프리에지 형태의 이해

모양	이름	특징
	{ Square nail }	-강한 느낌의 사각형태 -네일의 양끝 모서리 부분이 90° 사각의 형태이다. { } -발톱이 형태 활용 -내인성 발톱의 보정시에 적용

활용서식은

평가 서식, 실습시트 등 교수학습 시 활용 가능한 다양한 서식들로 구성하였습니다. 과제 진행에서 평가에 이르기까지 필요한 서식을 해당 학습모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

부록

네일 기본관리 도구와 재료 목록

목록	비고	준비
위생가운	흰색	작업자 착용
위생 마스크	흰색	작업자 착용
보호안경	투명한 렌즈 (안경으로 대체 가능)	작업자 착용
재용접반함	재용, 색상 무관	작업대

부록은

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

[NCS-학습מוד의 위치]

대분류	정보통신	
중분류	정보기술	
소분류	정보기술개발	

세분류		
SW아키텍처	능력단위	학습מוד명
응용SW 엔지니어링	요구사항 확인	요구사항 확인
시스템엔지니어링	애플리케이션 설계	애플리케이션 설계
DB엔지니어링	애플리케이션 구현	애플리케이션 구현
NW엔지니어링	화면 구현	화면 구현
보안 엔지니어링	데이터 입출력 구현	데이터 입출력 구현
UI/UX 엔지니어링	통합 구현	통합 구현
시스템SW 엔지니어링	개발자 테스트	개발자 테스트
	정보시스템 이행	정보시스템 이행
	제품소프트웨어 패키징	제품소프트웨어 패키징
	소프트웨어공학 활용	소프트웨어공학 활용

차 례

학습모듈의 개요	1
학습 1. 정적모델 상세설계하기	
1-1 정적모델 검증	3
1-2 정적 설계모델 상세화	12
• 교수·학습 방법	22
• 평가	23
학습 2. 동적모델 상세설계하기	
2-1 동적모델 검증	25
2-2 동적 설계모델 상세화	30
• 교수·학습 방법	40
• 평가	41
학습 3. 공통 모듈 설계하기	
3-1 공통 모듈 상세화	43
• 교수·학습 방법	51
• 평가	52
학습 4. 타 시스템 연동하기	
4-1 시스템 연동 상세화	54
4-2 시스템 연동 오류 관리	59
• 교수·학습 방법	62
• 평가	63
참고 자료	65

애플리케이션 설계 학습모듈의 개요

학습모듈의 목표

요구사항 확인을 통한 상세 분석 결과, SW아키텍처 가이드라인 및 소프트웨어 아키텍처 산출물에 의거하여 이에 따른 애플리케이션 구현을 수행하기 위해 정적모델 상세설계, 동적모델 상세설계, 공통 모듈 설계, 타 시스템 연동에 대한 상세 설계를 수행할 수 있다.

선수학습

업무 관련 용어 및 관련 업무 이해, 객체지향 개념 이해, UML 표기법 및 도식 기술, 설계 모델링 기법 이해 등

학습모듈의 내용체계

학습	학습내용	NCS 능력단위요소		
		코드번호	요소명칭	수준
1. 정적모델 상세설계하기	1-1 정적모델 검증 1-2 정적 설계모델 상세화	2001020202_14v2.1	정적모델 상세설계하기	5
2. 동적모델 상세설계하기	2-1 동적모델 검증 2-2 동적 설계모델 상세화	2001020202_14v2.2	동적모델 상세설계하기	5
3. 공통 모듈 설계하기	3-1 공통 모듈 상세화	2001020202_14v2.3	공통 모듈 설계하기	6
4. 타 시스템 연동설계 하기	4-1 시스템 연동 상세화 4-2 시스템 연동 오류 관리	2001020202_14v2.4	타 시스템 연동설계하기	6

핵심 용어

정적모델, 동적모델, 공통 모듈, 타 시스템 연동

학습 1

정적모델 상세설계하기 (LM2001020202_14v2.1)

학습 2 동적모델 상세설계하기(LM2001020202_14v2.2)

학습 3 공통 모듈 설계하기(LM2001020202_14v2.3)

학습 4 타 시스템 연동하기(LM2001020202_14v2.4)

1-1. 정적모델 검증

학습 목표

- 소프트웨어 아키텍처 설계 가이드라인을 참조하여 정적모델 상세설계 내역을 확인하고 검증할 수 있다.

필요 지식 /

① 모델과 모델링

1. 개념

모델(Model)이란 현실의 단순화, 가시화를 통해 개발할 시스템에 대한 계획/구상에 대한 내용을 나타낸 것이다. 모델링(Modeling)은 양질의 소프트웨어를 개발하기 위해 모델을 만드는 작업에 들어가는 모든 활동이라고 볼 수 있다.

2. 모델링 역사

시스템 구축을 위하여 분석 및 설계 단계에서 활용하는 모델과 표기법은 IT 패러다임 진화에 따라, 그에 대응되어 지원하는 방법론(구조적방법론-정보공학방법론-객체지향방법론-CBD방법론-SOA방법론 등) 유형이 존재하고, 각 방법론마다 다른 모델 및 모델링 기법이 다양하게 활용된다. 본 학습 교재에서는 IT 트렌드와 시장 점유율을 고려하여 객체지향방법론을 기준으로 모델링 방법을 살펴보도록 하겠다.

- (1) 1980년대 말부터 나타나기 시작한 객체지향 모델링 방법은 OMT(Object Modeling Technique), 부치 방법론(Booch Method), OOSE(Object-Oriented Software Engineering), RDD (Responsibility Driven Design) 등 50여 가지 정도로 다양하게 존재한다.
- (2) 그중 가장 많이 사용된 방법으로 OMT, 부치 방법론, OOSE가 대표적인데, 1994년 말에 OMT방법론의 제임스 럼버(James Rumbaugh) 박사가 부치 방법론의 그래디 부치(Grady Booch)가 근무하던 래쇼날 소프트웨어(Rational Software)사로 합류하였고 1995

년에는 OOSE의 이바 야콥슨(Ivar Jacobson) 박사도 같이 합류하여 객체지향 분석 및 설계 표준화를 시작하게 된다. 마침내 1997년 1월에 세 명의 노력으로 OMT, 부치 방법론, OOSE 세 방법론이 융화된 UML 1.0을 발표하게 된다.

- (3) 이듬해 UML 1.1을 발표하여 기술표준 기구인 OMG에 표준 인증을 받게 되고, 1999년에는 UML 1.3, 2001년에는 UML 1.4를 발표하였고, 현재(2015년 8월)는 UML 2.5가 최종 승인을 받아 release 된 후, 지속적으로 계속 업데이트 중이다.

3. 객체지향 모델링 방법

여러 가지 방법이 있겠지만 그중 가장 대표적인 제임스 럼버의 OMT, 그라디 부치의 부치 방법론, 이바 야콥슨의 OOSE, 최근 가장 각광받는 래쇼날 소프트웨어사의 RUP(Unified Process), 새롭게 등장하고 있는 OMG의 MDA, 켄트 벡, 워드 커닝햄의 XP 등의 방법론이 있다.

② UML(Unified Modeling Language)

1. 정의

UML은 소프트웨어 청사진을 작성하는 표준 언어로서, 소프트웨어 중심 시스템의 산출물을 가시화하고, 명세화하고, 구축하고, 문서화하는 데 사용될 수 표준 표기법이라 할 수 있다.

(1) 가시화 언어

UML은 소프트웨어의 개념모델을 가시적인 그래픽 형태로 작성하여 참여자들의 오류 없고 원활한 의사소통이 이루어지게 하는 언어이다.

(2) 명세화 언어

UML은 소프트웨어 개발과정인, 분석, 설계, 구현 단계의 각 과정에서 필요한 모델을 정확하고 완전하게 명세할 수 있게 하는 언어이다.

(3) 구축 언어

UML 언어는 다양한 객체지향 프로그래밍 언어로 변환 가능하다. 따라서 UML로 명세된 설계 모델은 구축하려는 프로그램 코드로 순변환하여(순공학이라고 함) 구축에 사용 가능하다. 또한 기 구축된 코드를 UML 모델로 역변환(역공학이라고 함)하여 분석을 하게 할 수도 있다.

(4) 문서화 언어

UML은 여러 개발자들 간의 통제, 평가, 및 의사소통을 위한 문서화를 위한 언어이다.

2. UML 필요성

소프트웨어 시스템을 만들기 전에 모델을 만드는 것이 빌딩을 만들기 위한 설계도처럼 아주 중요한 역할을 한다. 따라서 소프트웨어 시스템을 만들기 위해서도 어휘와 규칙을 두어 시스템을 개념적 및 물리적으로 표현하는 모델이 필요하게 되었다. 이 모델로써 시스템의 구조적 문제와 프로젝트 팀내의 의사소통, 그리고 소프트웨어 구조의 재사용 문제를 해결할 수 있게 되었다. 이로써 성공적으로 시스템을 만들기 위해서는 객체지향적인 분석과 설계를 위한 표준으로 인정받은 모델링 언어인 UML이 필요하게 되었다.

3. UML 역사

1980년대 후반과 1990년대 초반에 등장하기 시작한 객체지향 분석 및 설계 방법론들 간의 치열한 경쟁 속에서 가장 유력한 방법론이었던 제임스 럼버의 OMT, 부치의 부치 방법론, 이바 야콥슨의 OOSE가 UML이라는 하나의 모델링 언어로 통합되게 되고 1997년 9월 래쇼날 사는 UML버전 1.1을 발표하고 OMG에 의해 표준 표기법으로 받아들여지게 된다. <표 1-1>은 UML의 역사를 요약한 것이다.

<표1-1> UML 역사

시 기	내 용
1980년대 말~ 1990년대 초반	50여 가지의 다양한 표현 기법과 방법론 존재
1995년	부치의 부치 방법론과 제임스 럼버의 OMT 방법론 통합 OOPSLA'95에 발표
1996년	이바 야콥슨의 OOSE 방법론이 추가 통합, UML 0.9 정의
1997년	1월: UML 1.0 발표 9월: UML 1.1 발표 (OMG에 표준화 안 상정) 11월: OMG 표준 인증
2001년	UML 1.4 발표
2004년	UML 2.0 발표
2010년	UML 2.3 발표
2011년	UML 2.4 발표
2015년	UML 2.5 승인

4. UML 2.X

(1) UML 구성

UML 구성은 크게 다음의 4가지로 구성되어 있다.

(가) Infrastructure

- 1) MOF & XML, CWM과 같은 메타모델을 정의하는 데 활용될 수 있는 메타 언어 규약이 정의되어 있다.

- 2) 프로파일(Profile)을 활용하여 UML을 조정(Customization)할 수 있게 하였다.
- 3) 메타모델 간 상호 호환을 지원한다.

(나) Superstructure

- 1) 메타모델을 사용하여 사용자 정의 모델의 구조와 행위를 정의하는 요소이다.
- 2) 구조 및 행위 다이어그램으로 구성된다.
- 3) 일반화, 종속성, 연관 관계의 의미를 보다 명확히 하였다.

(다) OCL(Object Constraint Language)

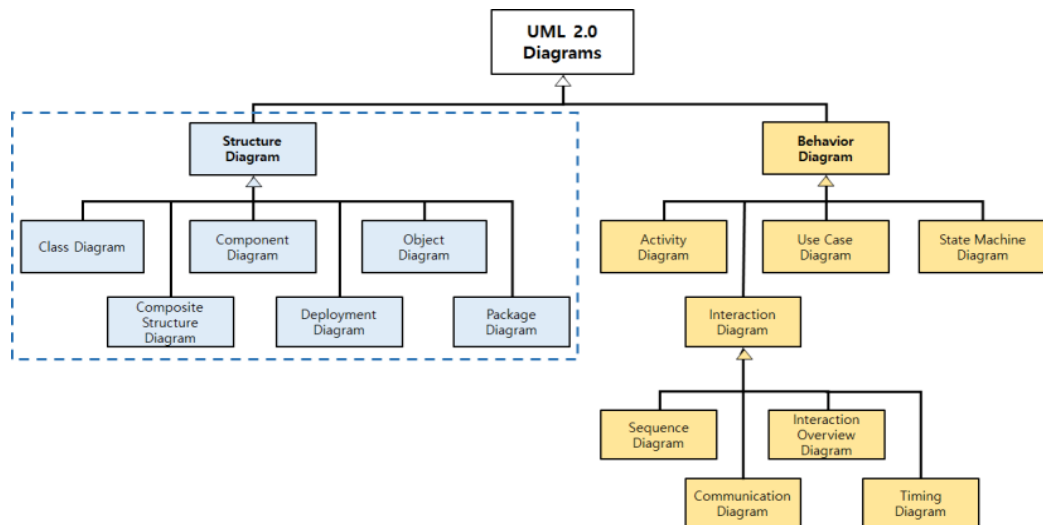
- 1) 모델 요소에 대한 제어와 제약을 위한 간략한 규칙을 정의하였다.
- 2) 특정 도메인에 대한 제한을 명시화하기 위해 사용된다.

(라) Diagram Interchange

서로 다른 모델링 도구 업체 간 UML 모델 호환을 위한 정의이다

(2) UML 제공 다이어그램

UML에서 제공하는 다이어그램은 크게 구조(정적) 다이어그램과 행위(동적) 다이어그램으로 나누어지는데, 본 장에서는 구조(정적) 다이어그램을 중심으로 살펴보기로 한다.



[그림 1-1] UML 2.0 Diagram과 구조(정적) 다이어그램

(3) 구조(정적) 다이어그램 유형

(가) 클래스(Class) 다이어그램

- 1) UML 모델링에서 가장 일반적으로 사용하는 다이어그램으로 시스템의 구조와 구조 간 상호 관계를 나타낸다.
- 2) 시스템을 구성하는 클래스와 인터페이스 사이의 유지하는 지속적인 정적인 관계를 나타낸 다이어그램이다.
- 3) 시스템의 논리적 및 물리적 구성요소 설계 시 주로 활용된다.

(나) 컴포넌트(Component) 다이어그램

컴포넌트 내의 물리적 단위의 구성과 컴포넌트 간 의존관계를 보여주며, 구현되는 시스템과 시스템 내에서 부품들이 어떻게 상호 작용하는지를 보여주는 다이어그램이다.

(다) 객체(Object) 다이어그램

시스템에 생성된 객체들 사이의 관계를 보여 주는데, 특정 시점에 시스템의 스냅샷을 보여준다고 할 수 있다.

(라) 복합구조(Composite Structure) 다이어그램

모델링 요소들의 내부적 구조를 보여 주고, 각 구성 요소들과 그 요소들이 어떻게 분리 및 연결되는지를 표현하는 다이어그램이다.

(마) 배포(Deployment) 다이어그램

실행 시점의 물리적 시스템에 존재하는, 노드(하드웨어)와 노드에 존재하는 컴포넌트의 물리적 구성을 표현한 다이어그램이다.

(바) 패키지(Package) 다이어그램

연관된 모델링 요소들을 그룹화하는 데 사용되는 패키지와 패키지 사이의 의존 관계를 나타내는 다이어그램이다.

③ 설계 클래스 유형(예시)

정적 다이어그램 중 대표적으로 사용하는 클래스 다이어그램을 대상으로, 일반적으로 애플리케이션에서 사용하고 있는 설계 클래스 유형과 해당 설계 클래스가 수행하는 책임을 살펴보면 다음과 같다.

1. JSP 클래스

사용자 인터페이스 클래스로서 사용자와 시스템 간의 상호작용을 위한 인터페이스를 제공한다. 구현을 위하여 JSP 구현기술을 적용한다.

2. Page Controller 클래스

(1) 사용자 인터페이스를 담당하는 JSP 클래스로부터 발생한 이벤트를 해석하여 비즈니스 로직 구현을 담당하는 클래스에 해당 이벤트를 전달한다.

(2) JSP 클래스와 비즈니스 로직 구현을 담당하는 클래스 사이에 정의하는 데이터 구조가 틀린 경우 데이터를 변환하는 책임을 가진다.

3. Biz Logic Service 클래스

모든 비즈니스 로직에 대한 구현을 담당하는 클래스로서 JSP 클래스로부터 Page Controller 클래스를 통하여 전달된 이벤트를 처리하기 위하여 다른 Biz Logic Service 클래스나 Data

Access Object 클래스와 상호작용 흐름을 제어한다.

4. Data Access Object 클래스

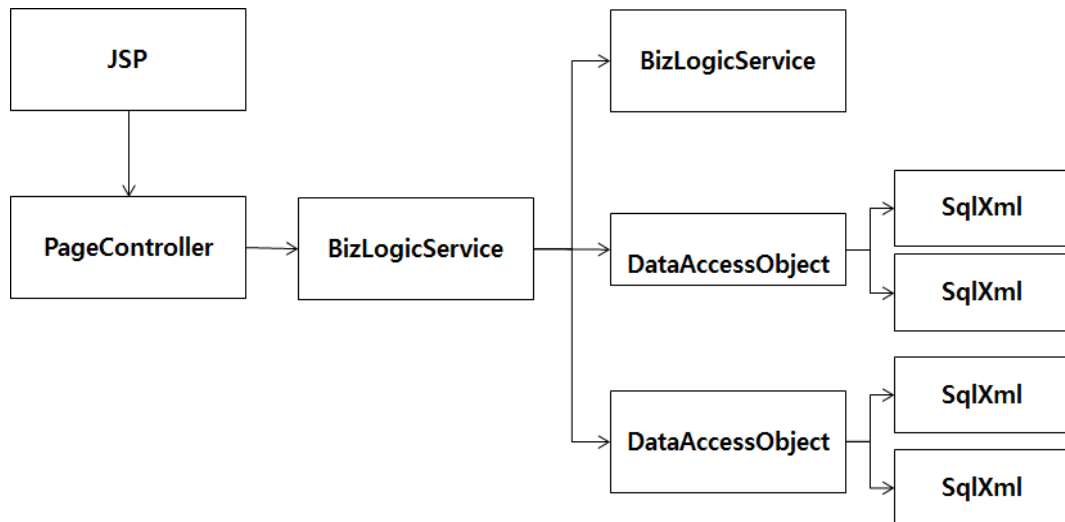
데이터베이스에 접근하여 관리 대상 정보에 대한 생성, 수정, 삭제, 조회 등의 기능을 수행하는 책임을 가진다.

5. Sql Xml 클래스

서로 밀접한 관련이 있는 SQL 문장들을 포함하고 있는 XML 파일이다.

④ 설계 클래스 상호작용 패턴(예시)

일반적인 애플리케이션에서 사용하고 있는 설계 클래스들 간의 상호작용 패턴은 [그림 1-2]와 같고, 이를 표준으로 정의했다면 이와 같은 상호작용 패턴은 모든 설계 클래스 간의 상호작용에 적용되어야 한다.



[그림 1-2] 설계 클래스 상호작용 패턴(예시)

1. JSP 클래스에서 발생한 이벤트는 PageController 클래스로 전달된다.
2. Page Controller 클래스는 JSP 클래스로부터 전달된 이벤트를 해석하고 필요 시 JSP 클래스로부터 전달되어 온 데이터를 Biz Logic Service 클래스가 처리할 수 있는 형태로 변환한 후에 해당 이벤트를 처리할 Biz Logic Service 클래스의 특정 오퍼레이션을 호출한다.
3. Biz Logic Service 클래스는 비즈니스 로직을 처리하기 위하여 다른 Biz Logic Service 클래스나 Data Access Object 클래스와 상호작용을 수행한다.
4. Data Access Object 클래스는 데이터베이스에 접근할 필요가 있는 작업을 수행해야 할 경우에 Sql Xml에 포함된 적절한 SQL문을 실행하여 작업을 처리한다.

⑤ 설계 클래스 및 오퍼레이션의 명명규칙(예시)

설계 클래스 유형에 따른 일반적인 클래스 및 오퍼레이션의 명명규칙은 다음과 같다.

1. 설계 클래스

(1) JSP 클래스: 의미있는 명칭

예) Product

(2) Page Controller 클래스: 의미있는 명칭 + Controller

예) ViewItemController

(3) Biz Logic Service 클래스: 의미있는 명칭

예) PetStore

(4) Data Access Object 클래스: SqlMap + 의미있는 명칭 + Dao

예) SqlMapItemDao

(5) SqlXml 클래스: 의미있는 명칭

예) Item

2. 오퍼레이션: 동사 + 명사 (의미있는 명칭)

예) getItem

⑥ 정적모델 검증을 위한 체크리스트

1. 작성된 정적모델이 해당 업종이나 업무에서 중시하는 개념 Class들이 잘 파악되었는지 확인한다.
2. 참여자들이 정적모델을 보고 업무(개념, 용어, 관계 등)을 쉽게 이해할 수 있는지 확인한다.
3. 핵심 클래스들의 주요 속성(Attribute)과 오퍼레이션이 파악되었는지 확인한다.
4. 유스케이스 기술서(Description)에 표현된 내용(특히 속성)이 클래스 다이어그램에 누락되지 않고 모두 반영되었는지 확인한다.
5. 클래스들 간 연관성이 잘 정의되었는지 검토한다.
6. 클래스명, 속성명, 오퍼레이션명, 연관명 등이 일관성 있고, 명명된 이름이 이해할 수 있을 정도로 충분한 의미를 담고 있는지 확인한다.
7. 클래스의 수준(Granularity)이 적절하고, 너무 작거나 큰 클래스가 존재하는지 확인한다.
8. 클래스간 통합 또는 추상화가 필요 없는지 리뷰한다.
9. 프로젝트 참여자들이 작성된 클래스 모델을 이해하고 공감하는지 파악한다.

재료 · 자료

- 모델링 검토 기준

기기(장비 · 공구)

- 컴퓨터, 인터넷, 설계용 소프트웨어

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

정적모델 설계는, 분석 모델링 결과를 토대로 실제로 구현을 위한 설계 모델을 작성하는 작업으로, 이를 위해서는 먼저 소프트웨어 아키텍처를 분석하여 구현을 위한 아키텍처 표준을 이해하여야 하고 작성된 정적모델을 대상으로 검증하는 순서로 진행된다.

① 소프트웨어 아키텍처 설계 가이드라인을 확인한다.

1. 소프트웨어 아키텍처는 구현을 위한 시스템의 구조로서, 아래와 같은 구현을 위한 제반 표준을 제공한다.
 - (1) 분류코드: 시스템 코드, 프로그램 코드
 - (2) 디렉토리 및 파일 구조
 - (3) 명명 규칙: 디렉토리, 파일, 변수, 함수, DB관련 오브젝트
 - (4) 주석
 - (5) SQL 코딩표준
 - (6) 메시지 표준
 - (7) 문서 및 소스관리 표준: 형상관리, 버전관리, 배포관리 등
2. 정적분석 모델로부터 정적설계 모델을 생성하기 위해서는 특히 설계 클래스의 유형과 상호작용 패턴을 정의하고 있는 계층 아키텍처를 명확히 이해하여야 한다.
3. 이외에도 구현을 위한 명명규칙 등 개발 표준을 이해하여야 한다.

② 상세설계된 정적모델을 대상으로 체크리스트를 정의하여 그에 따라 검증한다.

수행 tip

- 소프트웨어 아키텍처에서 정의된 레이어별 역할분담 기준을 전제하여 각 클래스가 식별 되었는지와, 아키텍처에서 제시하는 명명규칙에 따르고 있는지가 중요한 검증 포인트이다.

1-2. 정적 설계모델 상세화

학습 목표

- 정적모델 설계를 상세 분석하여 도출된 엔터티 클래스 또는 데이터 엔터티를 상세화 할 수 있다.
- 정적모델 설계를 상세 분석하여 경계 클래스 또는 사용자와 시스템 간의 상호작용 케이스를 상세화 할 수 있다.
- 정적모델 설계를 상세 분석하여 제어 클래스 또는 사용자와 시스템 간 상호작용에 관련된 제어관계를 상세화 할 수 있다.
- 정적모델 설계를 상세 분석하여 클래스 모델 또는 시스템의 정적 구조를 상세화 할 수 있다.

필요 지식 /

① 정적 모델의 클래스 유형

정적 모델을 표현하는 클래스의 유형은 정보를 관리하는 역할을 하는 엔터티 클래스, 액터와 상호작용하는 역할을 담당하는 경계 클래스, 그리고 경계 클래스와 엔터티 클래스 사이에서 허브(Hub) 역할을 하면서 유스케이스 행위를 조정하는 역할을 하는 컨트롤 클래스로 분류할 수 있다.

1. 엔터티(Entity) 클래스

엔터티 클래스는 정보를 관리하는 클래스로, 잘 만들어진 유스케이스 기술서, 추상 개념 클래스 다이어그램, 운영중인 시스템에 대한 데이터모델들 중 어느 하나만이라도 있다면 엔터티 클래스를 쉽게 파악할 수 있고, 다음 두가지 지식을 보유한다면 보다 정확한 엔터티 클래스를 식별할 수 있다.

(1) 도메인 지식과 시스템 특성 보유

엔터티 클래스 식별은 도메인 또는 시스템의 특성을 잘 모를 경우, 속성에 해당하는 것도 클래스로 파악할 수 있고, 클래스에 해당하는 중요한 것도 속성으로 파악하는 오류를 범할 수 있으니 유의해야 하는데 이는 시스템과는 전혀 관계가 없는 요소(액터, 프로젝트 범위 외)를 엔터티 클래스로 찾을 수 있기 때문이다.

(2) 패턴 관련 지식 보유

구축하는 시스템이 확장성, 유지보수성, 유연성 등이 전제되는 시스템이라면 엔터티 클래스들에 대한 추상화 과정이 반드시 필요한데, 이를 위해서는 업무 패턴이나 모델 구조 패턴 등을 어느 정도 이해하고 있어야 함을 인지하여야 한다.

2. 경계(Boundary) 클래스

경계 클래스는 액터가 시스템과 상호작용하는 부분 또는 다른 시스템과 연계되는 부분을 표현한다.

(1) 속성

속성은 화면에 표현된 내용이나 다른 시스템과 연동하기 위한 정보를 가질 수도 있고, 단순 스트링(String)으로 처리할 수 있는 부분이기도 하다.

(2) 오퍼레이션

화면에 입력된 내용을 컨트롤 클래스에 전달하거나, 컨트롤 클래스로부터 받은 정보를 화면에 출력하는 오퍼레이션 또는 타 시스템에 정보를 보내거나 받기 위한 오퍼레이션을 포함할 수 있다.

3. 컨트롤(Control) 클래스

컨트롤 클래스는 경계 클래스와 엔터티 클래스 사이에서 중간자 역할을 하면서 유스케이스 행위를 조정하는 역할을 하는 담당하는 클래스이다.

(1) 컨트롤 클래스 도출 기법

컨트롤 클래스를 효과적으로 도출하기 위해 일반적으로 유스케이스 모델을 활용한다. 이는 유스케이스가 서로 밀접하게 관련된 기능적 요구사항들로 기술되어 있기 때문에 이를 토대로 컨트롤 클래스를 도출한다면 해당 컨트롤 클래스가 수행하는 모든 책임이 서로 밀접하게 관련있게 되어 하나의 컨트롤 클래스는 매우 응집성이 높아지고, 서로 다른 컨트롤 클래스 간에는 의존성이 최소화 되는 장점이 있다.

(2) 컨트롤 클래스 크기

유스케이스 기준으로 몇 개의 컨트롤 클래스를 도출하는지에 대한 크기 기준 정립이 필요한데, 유스케이스 모델링 수행 시 유스케이스의 단위를 잘 정하고, 이를 모든 유스케이스에 대하여 일관성 있게 적용했다면 이로부터 도출한 컨트롤 클래스도 그 단위가 적절하고 일관성 있는 품질을 유지할 수 있다.

② 설계용 소프트웨어 선정 시 고려사항

설계용 소프트웨어 선정을 위해 후보군이 되는 도구들은 상용 및 오픈소스를 포함하여 다양하고도 많이 존재한다. 이들 도구들 중 사용해야 할 도구는 다음과 같은 항목을 충분히 고려하여 선정하여야 한다.

1. 기본적으로 제공되어야 할 기능

(1) 순(Foward)/역(Reverse) 엔지니어링 및 UML 표준 최신 스펙(2.x) 지원 여부

(2) 사용 편의성

2. 제품 환경

- (1) 사용을 위한 최소 시스템 사양
- (2) 적용 플랫폼(Windows, Linux, MacOS 등)의 지원 여부

3. 제품 업그레이드

제품의 상태(주기적인 업그레이드 여부, 오픈소스의 경우 해당 프로젝트의 활성화 상태)

4. 비용

라이선스 비용 및 유지보수 요율

수행 내용 / 정적 설계모델 상세화 하기

재료 · 자료

- UML 작성표준
- 아키텍처 정의서
- 모델링 검토기준

기기(장비 · 공구)

- 컴퓨터, 프린터
- 설계용 소프트웨어

안전 · 유의사항

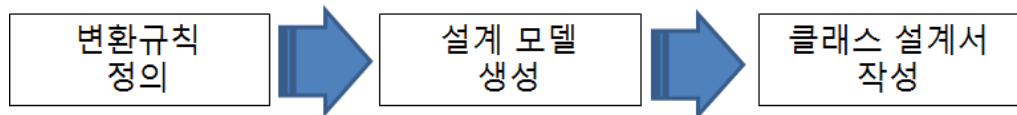
- 실습 후에는 실습 기기 및 컴퓨터 전원을 끈다.

수행 순서

정적모델 설계는, 분석 모델링 결과를 토대로 실제로 구현을 위한 설계 모델을 작성하는 작업이다.
소프트웨어 아키텍처에 대한 이해를 바탕으로 분석 모델에 정의된 분석 클래스를, 구현하기 위한

설계 클래스로 변환규칙을 정의하고, 이 변환 규칙에 의거하여 설계 클래스를 생성하고 설계 객체 상호작용 모델을 작성한다.

상호작용 모델이 완성되고 나면 객체설계 모델에 정의되어 있는 모든 설계 클래스와 이들 간의 상호작용을 문서화한 클래스 설계서를 작성함으로써 정적모델 설계작업이 종료된다. 실무에서는 이러한 일련의 순서가 여러 차례 반복을 통해 수행되며, 이러한 반복을 통하여 정적 설계 모델이 점진적으로 완성되어 간다고 볼 수 있다.



[그림 1-3] 정적모델 설계 순서

① 변환규칙을 정의하여 정적설계 모델로 변환한다.

1. 소프트웨어 아키텍처 분석이 완료된 후, 정적분석 모델을 정적설계 모델로 변환하기 위한 변환규칙을 정의해야 하는데, 변환 규칙은 정적분석 클래스 유형들을 설계 클래스의 어떤 유형으로 변환할 것인가에 대한 규칙을 정의하는 것을 의미한다.
2. 변환 규칙이 완성되면 정적분석 모델은 변환규칙에 의거하여 정적설계 모델로 변환할 수 있게 된다. <표 1-2>는 일반적인 분석 클래스 유형을 대상으로 설계 클래스 유형으로 변환하는 변환규칙을 예시한 내용이다.

<표 1-2> 설계 클래스 상호작용 패턴(예시)

분석 클래스 유형	설계 클래스 유형	변환 규칙
분석 패키지	설계 패키지	분석 패키지 당 하나의 설계 패키지를 생성
경계 클래스	JSP 클래스	Boundary 클래스 당 하나의 JSP 클래스를 생성
경계 클래스의 오퍼레이션	PageController 클래스	경계클래스의 오퍼레이션 당 하나의 PageController 클래스를 생성
컨트롤 클래스	BizLogicService	Control 클래스 당 하나의 BizLogicService 클래스 생성
엔터티 클래스	DataAccessObject 클래스	Entity 클래스 당 하나의 DataAccessObject 클래스를 생성
	SqlXml 클래스	Entity 클래스 당 하나의 SqlXml 클래스를 생성
분석 오퍼레이션	설계 오퍼레이션	분석 오퍼레이션 당 하나의 설계 오퍼레이션을 생성
분석 객체 상호작용	설계 객체 상호작용	분석 객체 상호작용 당 하나의 설계 객체 상호작용을 생성

② 설계모델 클래스를 생성한다.

1. 변환규칙이 완성되면 변환규칙에 따라 객체분석 모델을 객체설계 모델로 변환한다.
2. 객체설계 모델에 설계 클래스 및 해당 오퍼레이션을 생성한 후, 생성된 모든 클래스들을 클래스 다이어그램으로 도식한다.
3. 분석객체 상호작용 다이어그램을 설계객체 상호작용 다이어그램으로 변환한다.

③ 클래스 설계서를 작성한다.

1. 정적설계 모델을 토대로 클래스 설계서를 작성한다.

업무영역(시스템)		단위업무(시스템)	
단계	설계	작성일자	버전

1. 설계 클래스 목록

설계클래스ID	설계클래스명	관련유즈케이스ID

2. 시퀀스도

시퀀스ID		시퀀스도명	
관련유즈케이스ID			
주요액터		주요클래스	

3. 설계 클래스도

설계클래스도ID		설계클래스도명	
관련유즈케이스ID			

4. 설계 클래스 정의

설 계클래스ID				설 계클래스명	
속성					
속성명	가시성	타입		기 본값	설 명
오퍼레이션					
오퍼레이션명	가시성	파라미터		변 환타입	설 명

[그림 1-4] 클래스 설계서 양식

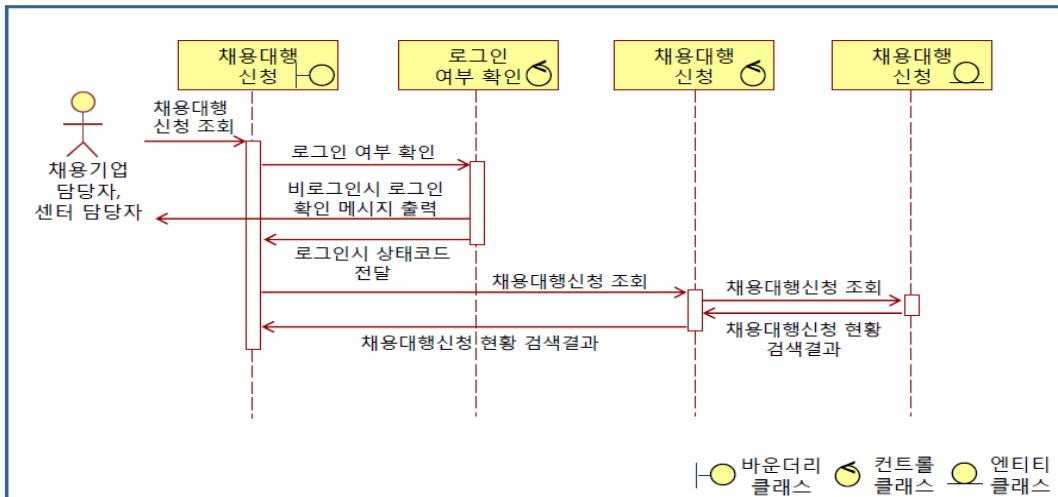
업무영역(시스템)	채용대행서비스	단위업무(시스템)	기업회원서비스
단계	설계	작성일자	2011.11.07
		버전	1.0

1. 설계 클래스 목록

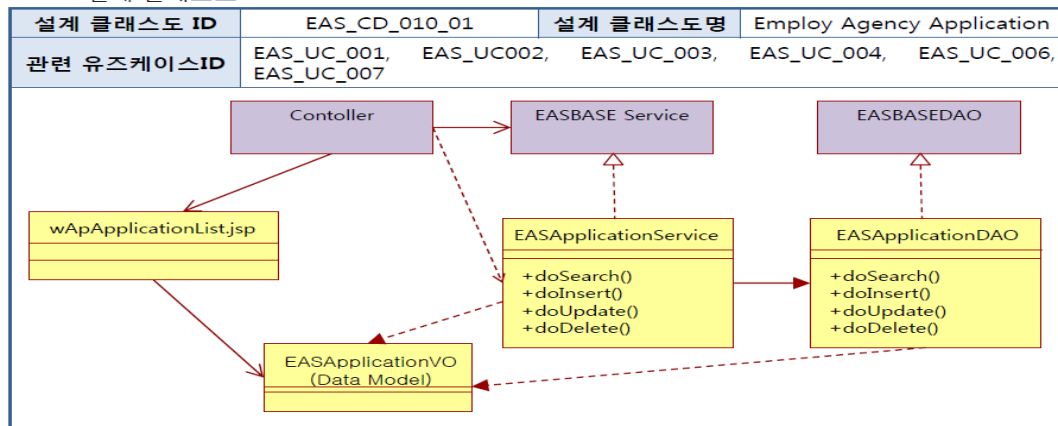
설계클래스ID	설계클래스명	관련유즈케이스ID
EASAppliactionService	채용대행신청	EAS_UC_001,EAS_UC002,EAS_UC_003, EAS_UC_004,EAS_UC_006,EAS_UC_007

2. 시퀀스도

시퀀스ID	EAS_SD_010_01	시퀀스도명	채용대행신청
관련 유즈케이스ID	EAS_UC_001,EAS_UC002,EAS_UC_003,EAS_UC_004,EAS_UC_006, EAS_UC_007		
주요액터	채용기업 담당자, 센터 담당자	주요클래스	채용대행신청,로그인, 회원가입



3. 설계 클래스도



4. 설계 클래스 정의

설 계클래스ID		EAS_CD_010_01		설 계클래스명		EASApplicationService	
속 성							
속 성명		가 시성	타 입	기 본값	설 명		
신청시작일		public	string	현재일 -7일	채용대행신청시작일자		
신청종료일		public	string	현재일	채용대행신청종료일자		
진행상태		public	string	전 체	T:전체,W:작성,R:접수,P:승인,F:반려		
신청 제목명		public	text	-	채용대행신청현황을검색하기위한 제목		
오 퍼레이 셴							
오 퍼레이 셴명		가 시성	파라미터		반 환타입	설 명	
doSearch		public	request:HttpServletRequest response:HttpServletResponse		List	입 력한검 색조건 에 맞는채용대행신청 현황조회	
doInsert		public	request:HttpServletRequest response:HttpServletResponse		boolean	입 력한채용대행신청 을등록	
:	:	:	:	:	:	:	:

[그림 1-5] 클래스 설계서 작성 사례

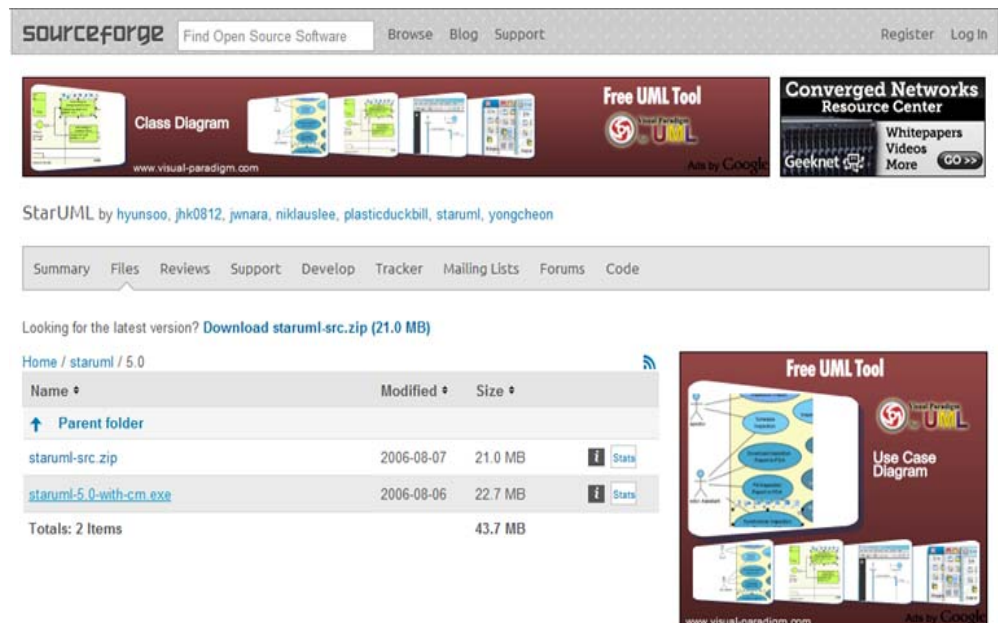
- (1) 객체와 클래스를 식별한다.
 - (2) 클래스에 대한 자료사전을 작성한다.
 - (3) 클래스 간의 관계를 정의한다.
 - (4) 객체 속성 및 연결 관계를 정의한다.
 - (5) 클래스를 계층화하고 모듈로 정의한다.
 - (6) 생성된 모형을 반복적으로 검증한다.
2. 클래스 설계서에는 모든 클래스와 해당 클래스의 오퍼레이션에 대한 개요가 기술되며 특정 이벤트를 처리하기 위한 설계 객체 상호작용 모델도 포함된다.
 3. 이후 구현단계에서 프로그래머가 코딩 작업 수행 시에는 이때 작성한 클래스 설계서를 주요 문서로 참고하여 개발을 수행한다.

④ 설계용 소프트웨어(모델링 도구)를 활용한 정적 모델 작성

현존하는 모델링 도구 중 사용하기 편하고, 오픈 소스로 배포되는 도구 중 StarUML을 기준으로 정적모델을 작성해 보도록 한다.

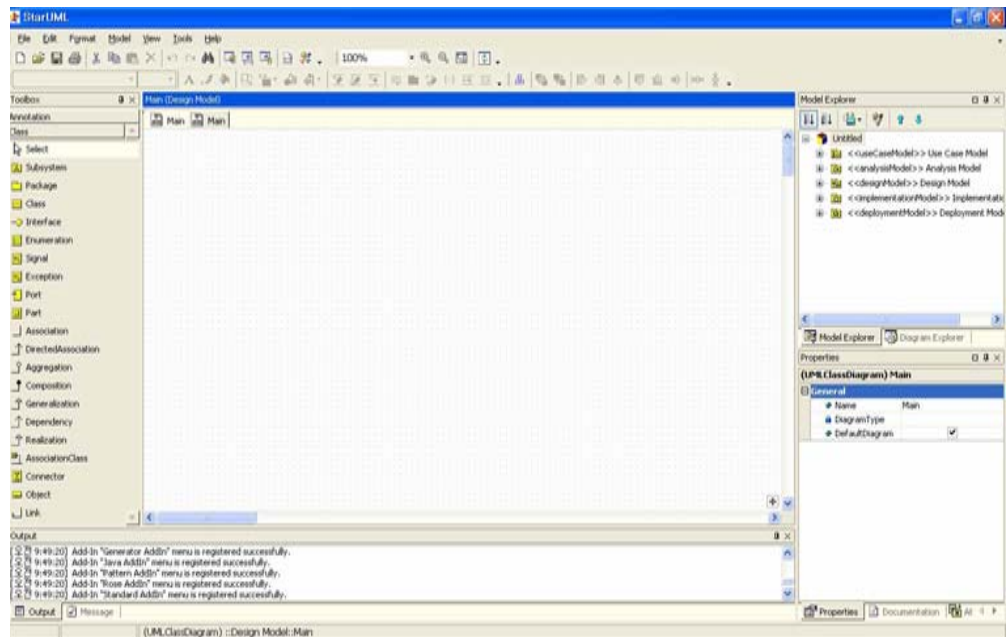
1. StarUML 설치

- (1) 다운로드 경로(<http://sourceforge.net/projects/staruml/files/staruml/5.0/>)로 접속한 후, 화면에서 'staruml-5.0-with-cm.exe' 를 클릭한다.



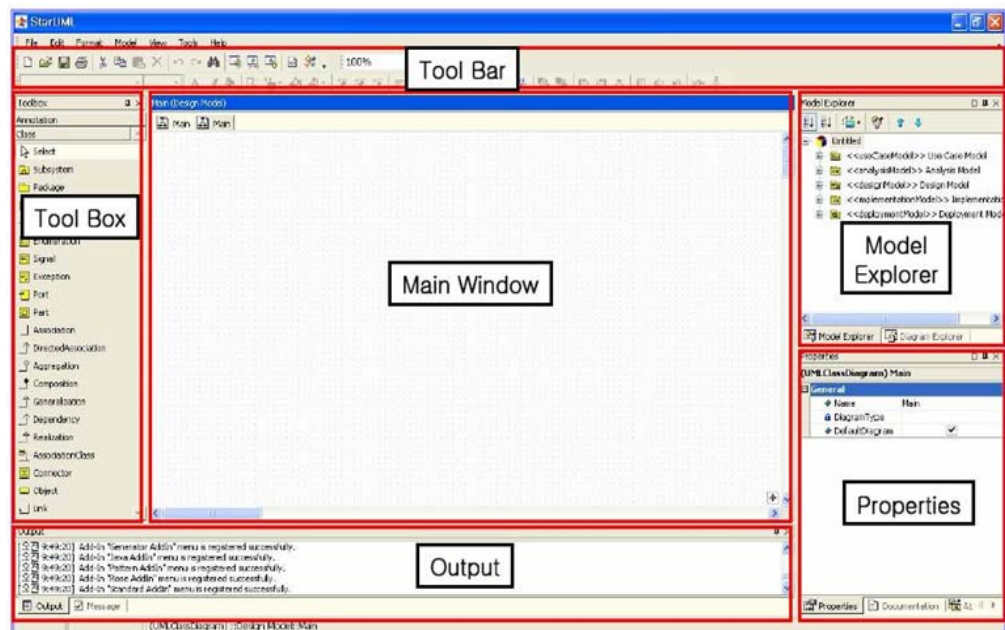
[그림 1-6] StarUML 다운받기

- (2) 다운로드받은 'staruml-5.0-with-cm.exe' 파일을 더블 클릭하여 실행하고, 실행 후 설치되면 아래와 같은 메인화면이 나타난다.



[그림 1-7] StarUML 메인화면

(3) 메인화면은 다음 [그림 1-8]과 같이 6개의 블록영역으로 구성된다.



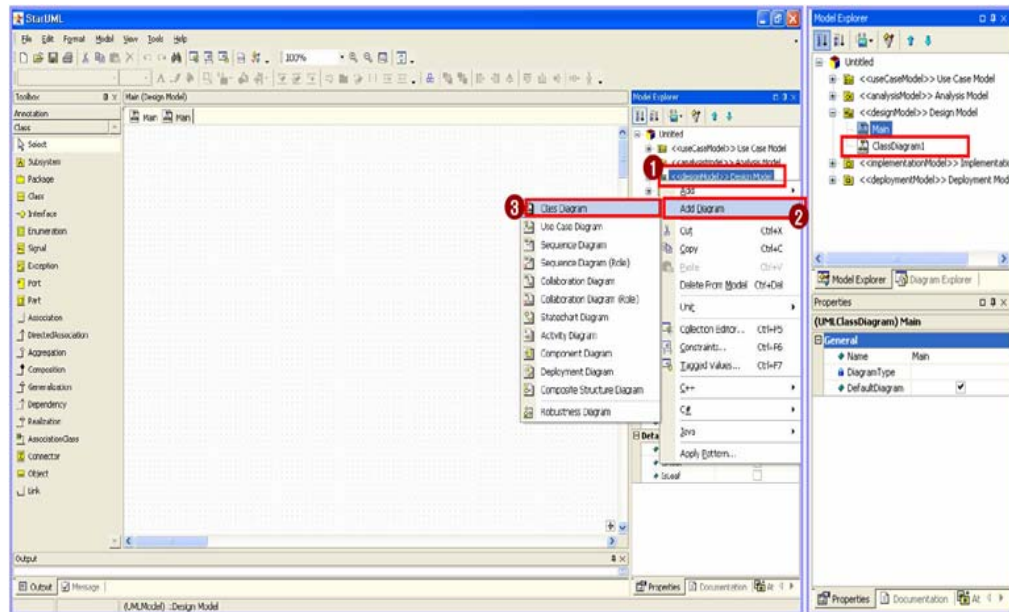
[그림 1-8] StarUML 메인화면 구성

- (4) 메인화면의 6개 블록영역 중 상단에 위치한 공통 Tool Bar는 표준, 서식, 보기, 정렬 도구모음 기능을 제공한다.
- (5) 정상적으로 설치된 도구의 모델 작성과 세부 활용법은 한글화된 사용자 가이드 [url\(http://staruml.sourceforge.net/docs/user-guide\(ko\)/toc.html\)](http://staruml.sourceforge.net/docs/user-guide(ko)/toc.html)를 참조하면 된다.

2. 정적 모델 작성

(1) 정적 모델의 대표적 다이어그램인 클래스 다이어그램을 작성하기 위해 다이어그램 유형을 선택한다.

도구의 메인화면 우측 상단에 있는 Model Explorer의 <<DesignModel>> Design Model 선택 -> 마우스 오른쪽 버튼 클릭 -> Add Diagram-> Class Diagram 선택



[그림 1-9] 클래스 다이어그램 선택

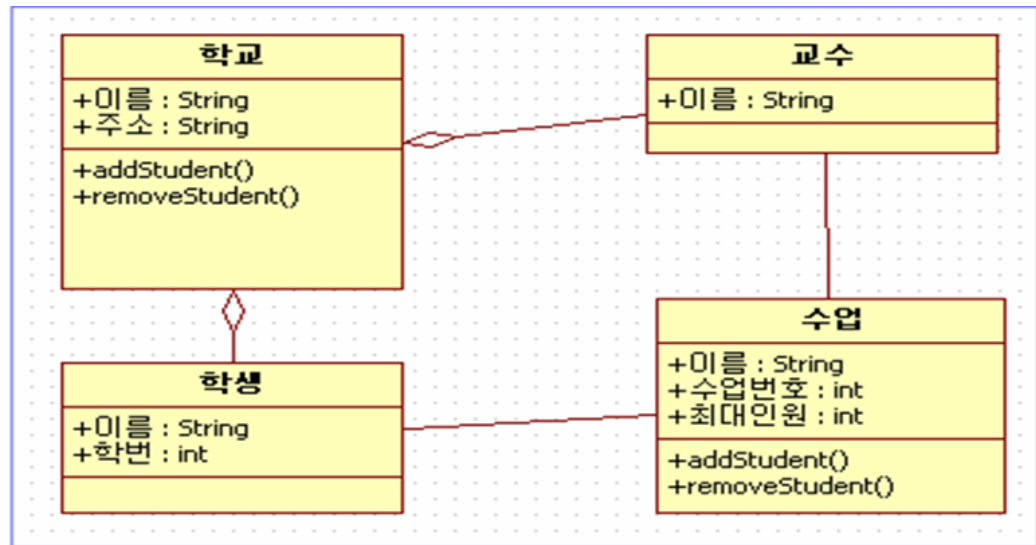
(2) 클래스 다이어그램을 작성하기 위해 사용해야 할 Tool Bar의 유형과 세부 기능은 다음과 같다.

<div>Class</div> <ul style="list-style-type: none"> Select Subsystem Package Class Interface Enumeration Signal Exception Port Part Association DirectedAssociation Aggregation Composition Generalization Dependency Realization AssociationClass Connector Object Link 	<table border="1"> <tr> <td>Select</td><td>Diagram 요소를 선택합니다.</td></tr> <tr> <td>Subsystem</td><td>물리적인 시스템의 부분 또는 전체를 의미합니다.</td></tr> <tr> <td>Package</td><td>관련된 클래스들을 모아서 모델링 한 것을 의미합니다.</td></tr> <tr> <td>Class</td><td>객체의 Attribute와 Method들을 모델링 한 것을 의미합니다.</td></tr> <tr> <td>Interface</td><td>클래스에서 Method의 선언부분만 모델링 한 것을 의미합니다.</td></tr> <tr> <td>Enumeration</td><td>미리 정의된 값을 리스트로 가지는 데이터타입을 의미합니다.</td></tr> <tr> <td>Signal</td><td>객체간의 비동기적 통신 신호를 의미합니다.</td></tr> <tr> <td>Exception</td><td>실행 오류시 Operation에 의해 발생하는 신호를 의미합니다.</td></tr> <tr> <td>Port</td><td>인터페이스와 클래스의 연결 단자를 의미합니다.</td></tr> <tr> <td>Part</td><td>클래스 내부의 특정 부분을 의미합니다.</td></tr> <tr> <td>Association</td><td>한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 가능합니다.</td></tr> <tr> <td>DirectedAssociation</td><td>한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 불가능합니다.</td></tr> <tr> <td>Aggregation</td><td>한 클래스가 다른 클래스를 포함하는 관계일 때 사용합니다.</td></tr> <tr> <td>Composition</td><td>한 클래스가 다른 클래스에 완전히 종속되는 관계일 때 사용합니다.</td></tr> <tr> <td>Generalization</td><td>일반적인 요소와 더 구체적인 요소의 관계일 때 사용합니다.</td></tr> <tr> <td>Dependency</td><td>한 클래스의 변화가 다른 클래스의 변화에 영향을 주는 관계를 의미합니다.</td></tr> <tr> <td>Realization</td><td>인터페이스와 클래스와의 연결에 사용합니다.</td></tr> <tr> <td>AssociationClass</td><td>클래스와 연과의 연결에 사용합니다.</td></tr> <tr> <td>Connector</td><td>Part 사이의 연결에 사용합니다.</td></tr> <tr> <td>Object</td><td>특정 클래스의 인스턴스를 의미합니다.</td></tr> <tr> <td>Link</td><td>객체 사이의 연결에 사용합니다.</td></tr> </table>	Select	Diagram 요소를 선택합니다.	Subsystem	물리적인 시스템의 부분 또는 전체를 의미합니다.	Package	관련된 클래스들을 모아서 모델링 한 것을 의미합니다.	Class	객체의 Attribute와 Method들을 모델링 한 것을 의미합니다.	Interface	클래스에서 Method의 선언부분만 모델링 한 것을 의미합니다.	Enumeration	미리 정의된 값을 리스트로 가지는 데이터타입을 의미합니다.	Signal	객체간의 비동기적 통신 신호를 의미합니다.	Exception	실행 오류시 Operation에 의해 발생하는 신호를 의미합니다.	Port	인터페이스와 클래스의 연결 단자를 의미합니다.	Part	클래스 내부의 특정 부분을 의미합니다.	Association	한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 가능합니다.	DirectedAssociation	한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 불가능합니다.	Aggregation	한 클래스가 다른 클래스를 포함하는 관계일 때 사용합니다.	Composition	한 클래스가 다른 클래스에 완전히 종속되는 관계일 때 사용합니다.	Generalization	일반적인 요소와 더 구체적인 요소의 관계일 때 사용합니다.	Dependency	한 클래스의 변화가 다른 클래스의 변화에 영향을 주는 관계를 의미합니다.	Realization	인터페이스와 클래스와의 연결에 사용합니다.	AssociationClass	클래스와 연과의 연결에 사용합니다.	Connector	Part 사이의 연결에 사용합니다.	Object	특정 클래스의 인스턴스를 의미합니다.	Link	객체 사이의 연결에 사용합니다.
Select	Diagram 요소를 선택합니다.																																										
Subsystem	물리적인 시스템의 부분 또는 전체를 의미합니다.																																										
Package	관련된 클래스들을 모아서 모델링 한 것을 의미합니다.																																										
Class	객체의 Attribute와 Method들을 모델링 한 것을 의미합니다.																																										
Interface	클래스에서 Method의 선언부분만 모델링 한 것을 의미합니다.																																										
Enumeration	미리 정의된 값을 리스트로 가지는 데이터타입을 의미합니다.																																										
Signal	객체간의 비동기적 통신 신호를 의미합니다.																																										
Exception	실행 오류시 Operation에 의해 발생하는 신호를 의미합니다.																																										
Port	인터페이스와 클래스의 연결 단자를 의미합니다.																																										
Part	클래스 내부의 특정 부분을 의미합니다.																																										
Association	한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 가능합니다.																																										
DirectedAssociation	한 클래스와 다른 클래스가 연관 관계가 있을 때 사용합니다. Qualifier의 사용이 불가능합니다.																																										
Aggregation	한 클래스가 다른 클래스를 포함하는 관계일 때 사용합니다.																																										
Composition	한 클래스가 다른 클래스에 완전히 종속되는 관계일 때 사용합니다.																																										
Generalization	일반적인 요소와 더 구체적인 요소의 관계일 때 사용합니다.																																										
Dependency	한 클래스의 변화가 다른 클래스의 변화에 영향을 주는 관계를 의미합니다.																																										
Realization	인터페이스와 클래스와의 연결에 사용합니다.																																										
AssociationClass	클래스와 연과의 연결에 사용합니다.																																										
Connector	Part 사이의 연결에 사용합니다.																																										
Object	특정 클래스의 인스턴스를 의미합니다.																																										
Link	객체 사이의 연결에 사용합니다.																																										

[그림 1-10] 클래스 다이어그램 Tool Bar

(3) 아래의 수강신청 관련 지문 내용을 토대로 클래스 다이어그램을 도식해 보면 다음과 같다.

학교는 0명 이상의 학생으로 구성된 집합연관이다.
 학생은 여러 수업을, 수업은 여러 명의 학생을 수용할 수 있는 다대다 관계이다.
 교수 한 명당 0개 이상의 수업을 담당한다.



[그림 1-11] 클래스 다이어그램 작성 예시

수행 tip

- 도구선택시 고려사항을 고려하여 모델링 도구를 선정하고, UML 표준에 따라 업무지식을 반영한 정적 설계모델이 작성될 수 있도록 한다.

학습 1 교수 · 학습 방법

교수 방법

- 교수자의 주도로 소프트웨어 아키텍처 중 설계와 관련된 항목을 파악하여 제시하고, 정적 모델을 검증하는 등의 내용을 PPT 자료로 작성하여 설명한다.
- 정적모델 설계를 상세 분석하여 도출된 엔터티 클래스 또는 데이터 엔터티를 상세화할 수 있도록 관련 내용을 정리하여 설명한다.
- 정적모델 설계를 상세 분석하여 경계 클래스 또는 사용자와 시스템 간의 상호작용 케이스를 상세화할 수 있도록 관련 내용을 정리하여 설명한다.
- 정적모델 설계를 상세 분석하여 제어 클래스 또는 사용자와 시스템 간 상호작용에 관련된 제어관계를 상세화할 수 있도록 관련 내용을 정리하여 설명한다.
- 모델링 도구들에 대한 이해 및 이전 사용 경험 여부에 대해 파악한 후 수업을 진행하고, 필요시 모델링 도구 매뉴얼 및 관련 홈페이지 등을 방문하여 관련 자료를 획득하여 제시한 후 설명한다.
- 모델링 도구의 설치 과정과 다이어그램 작성에 대해 단계적 실습이 이루어질 수 있도록 지도한다.

학습 방법

- UML과 모델링 도구의 필요성 및 활용 용도에 대하여 이해한다.
- 정적모델 설계를 상세 분석하여 도출된 엔터티 클래스 또는 데이터 엔터티를 상세화하는 과정에 대해 이해하고 실습한다.
- 정적모델 설계를 상세 분석하여 경계 클래스 또는 사용자와 시스템 간의 상호작용 케이스를 상세화하는 과정에 대해 이해하고 실습한다.
- 정적모델 설계를 상세 분석하여 제어 클래스 또는 사용자와 시스템 간 상호작용에 관련된 제어관계를 상세화하는 과정에 대해 이해하고 실습한다.
- 모델링을 위한 각종 다이어그램을 작성하는 과정과 도구 활용법을 이해하고 실습한다.
- 모델링 도구를 여러 개발자들이 공유하여 형상관리할 수 있는 가이드와 자료를 찾아 실습할 수 있도록 한다.

학습 1 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
정적모델 검증	- 소프트웨어 아키텍처 설계 가이드라인을 참조하여 정적모델 상세설계 내역을 확인하고 검증할 수 있다.			
정적 설계모델 상세화	- 정적모델 설계를 상세 분석하여 도출된 엔터티 클래스 또는 데이터 엔터티를 상세화할 수 있다.			
	- 정적모델 설계를 상세 분석하여 경계 클래스 또는 사용자와 시스템 간의 상호작용 케이스를 상세화할 수 있다.			
	- 정적모델 설계를 상세 분석하여 제어 클래스 또는 사용자와 시스템 간 상호작용에 관련된 제어관계를 상세화할 수 있다.			
	- 정적모델 설계를 상세 분석하여 클래스 모델 또는 시스템의 정적 구조를 상세화할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
정적모델 검증	- 소프트웨어 아키텍처 설계 가이드라인의 이해			
	- 설계 가이드라인을 고려한 정적모델 상세 설계내용 이해			
정적 설계모델 상세화	- 정적 설계모델 구체화를 통한 완전성			

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
정적모델 검증	- 소프트웨어 아키텍처 프로토타이핑 결과 이해			
	- 소프트웨어 아키텍처를 고려한 정적모델 설계 이해			
정적 설계모델 상세화	- 정적 설계모델 완전성			
	- 정적 설계모델의 요구사항 추적성 확보			

피드백

1. 체크리스트를 통한 관찰

- 실습 과정에서 체크리스트에 따라 평가한 후, 개선해야 할 사항 등을 정리하여 돌려준다.
- 소프트웨어 아키텍처 가이드라인 이해도에 대해 파악한 뒤, 이해도가 미비한 영역을 정리하여 돌려준다.
- 정적 설계모델의 완전성을 검토하여 미비사항에 대해 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 개선사항에 대하여 표시하여 돌려준다.
- 소프트웨어 아키텍처에 기반한 정적모델 설계인지를 파악한 후, 미비사항에 대해 돌려준다.
- 정적 설계모델의 요구사항 반영 및 추적성을 파악하여, 미비사항에 대해 돌려준다.

학습 1	정적모델 상세설계하기(LM2001020202_14v2.1)
학습 2	동적모델 상세설계하기 (LM2001020202_14v2.2)
학습 3	공통 모듈 설계하기(LM2001020202_14v2.3)
학습 4	타 시스템 연동하기(LM2001020202_14v2.4)

2-1. 동적모델 검증

학습 목표

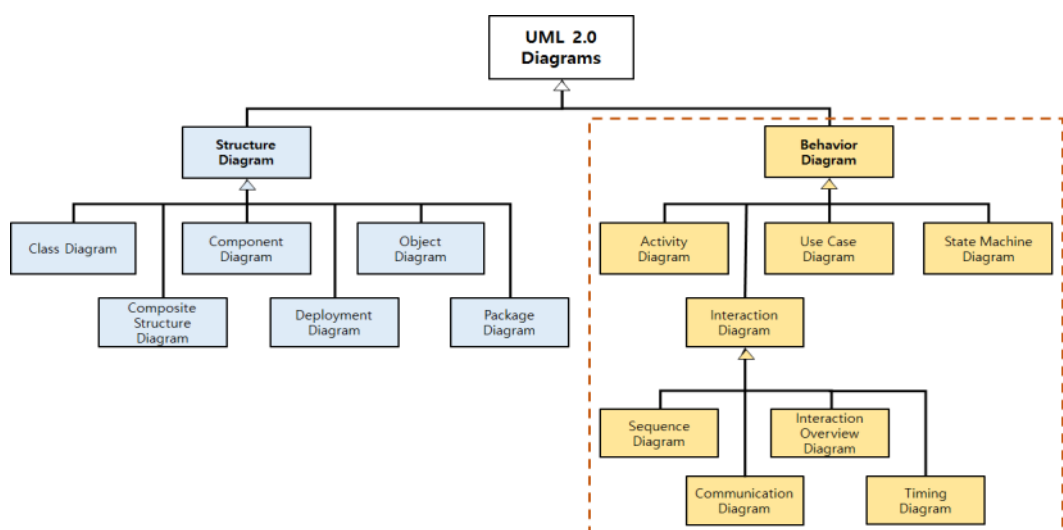
- 소프트웨어 아키텍처 설계 가이드라인을 참조하여, 동적모델 상세설계의 내역 확인 및 검증할 수 있다.

필요 지식 /

① UML(Unified Modeling Language)

1. UML 제공 다이어그램

UML에서 제공하는 다이어그램은 크게 구조(정적) 다이어그램과 행위(동적)다이어그램으로 나누어지는데, 본 장에서는 행위(동적) 다이어그램을 중심으로 살펴보기로 한다.



[그림 2-1] UML 2.0 Diagram과 행위(동적) 다이어그램

2. 행위(동적) 다이어그램 유형

(1) 액티비티(Activity) 다이어그램

시스템 내의 활동들의 흐름을 보여주는 다이어그램으로, 여러 업무 프로세스들을 설명하는데 자주 사용된다.

(2) 유스케이스(Usecase) 다이어그램

시스템이 구현할 업무프로세스를 다루는 다이어그램으로, 시스템을 사용하는 역할이라 할 수 있는 액터와 요구사항 중 기능 요구사항을 대상으로 표현하는 유스케이스와의 관계를 표현한다.

(3) 상태머신(State Machine) 다이어그램

개체의 상태와 이 개체가 상태들 사이를 어떻게 전이하는지 보여주는 다이어그램으로, 이벤트에 따라 순차적으로 발생하는 한 객체의 상태변화를 표현함으로써 동적 뷰를 제공하여, 이벤트 중심의 행동을 모델링할 때 자주 사용된다.

(4) 인터랙션(Interaction) 다이어그램

인터랙션 다이어그램은 아래 4가지 다이어그램으로 구성된다.

(가) 시퀀스(Sequence) 다이어그램

시스템 내부에서 동작하는 객체들 사이의 주고받는 메시지를 시간 순서를 강조하여 표현한 다이어그램이다.

(나) 인터랙션 오버뷰(Interaction Overview) 다이어그램

인터랙션 시퀀스 사이의 제어흐름에 대한 개요를 보여 주는 데 사용되며, 제어 흐름의 이해를 돕기 위한 방법으로 액티비티 다이어그램의 변형을 통해 인터랙션을 표현하는 다이어그램이다.

(다) 커뮤니케이션(Communication) 다이어그램

객체들이 어떻게 협력하고 교류하는지를 객체 간 구조를 강조한 다이어그램으로, UML 1.x의 협력(Collaboration) 다이어그램과 유사하다.

(라) 타이밍(Timing) 다이어그램

장치의 동작이나 회로 동작에서 타이밍과의 상호관계를 나타내며, 교류하는 요소들의 상태 또는 조건 정보에 대한 변경 사항을 나타낸 다이어그램이다.

② 동적모델 검증을 위한 체크리스트

1. 모델링 목적에 맞는 다이어그램을 선택했는지 검토한다.

시간적 흐름을 강조한다면 Sequence, 구조적 흐름을 강조하면 Communication 다이어그램으로, 또는 사용용도에 맞도록 Interaction Overview, Timing 다이어그램으로 도식했는지 체크한다.

2. Interaction 내의 객체들이 정의된 Class 로부터 찾아진 것인지 확인한다.
3. Usecase Description에 나타난 시스템 행위들이 충분히 반영되었는지 검토한다.
4. Class 다이어그램에 정의된 Class의 Operation들과 Message들이 일치하는지 확인한다.
5. Class 다이어그램에 정의된 Class들 간의 관계 표현이 Interaction Diagram의 메시지들과 일치하는지 확인한다.
6. 다이어그램 내에 표기한 구성요소가 본래의 의미를 충분히 파악하여 도식했는지 확인한다.
7. 도식된 다이어그램이 가독성(Readability)과 이해용이성(Understandability)이 보장되는지 확인한다.
8. 일반적으로 제시하는 아래와 같은 관점에서 좋은 동적 다이어그램 요건을 만족하는지 검토한다.
 - (1) 여러 측면이 아닌 한 가지 측면에만 초점을 맞춘 동적인 측면을 포함한다.
 - (2) 해당 부분을 이해하는 데 필수적인 요소만을 표현한다.
 - (3) 추상화 수준에 맞으며 이해하기에 필수적인 장식(Adornment)만 표현한다.
 - (4) 적절한 단위까지 최소화시켜 작성한다.
 - (5) 아래 내용을 반영하여 적절하게 도식한다.
 - (가) 목적을 전달할 수 있는 명칭 부여한다.
 - (나) 교차선이 최소화되도록 구성요소를 배치한다.
 - (다) Note 등을 이용한 시각 효과를 표현한다.
 - (라) 분기의 존재 시 최소화하여 표현한다.

재료 · 자료

- 모델링 검토 기준

기기(장비 · 공구)

- 컴퓨터, 인터넷, 설계용 소프트웨어

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

동적모델 설계는, 분석 모델링 결과를 토대로 실제로 구현을 위한 설계 모델을 작성하는 작업이다. 이를 위해서는 먼저 요구사항 모델이나 정적 모델 내용을 기반으로 작성한 동적 모델의 특성 및 동적 모델 작성시 활용된 여러 다이어그램이 갖는 고유 목적과 모델들 간 연관 관계를 충분히 이해하는 것이 필요하다. 특히 유스케이스 기술서에 나타난 액터와 시스템 간 상호작용과 유스케이스 내부 흐름이 동적 모델에 얼마나 제대로 반영되었는지를 검토하는 것이 선행되어야 하며, 검토 후 작성된 동적모델을 대상으로 검증하는 순서로 진행된다.

① 작성 동적모델을 검토한다.

일반적인 동적 모델은 시간적 흐름에 따른 사항을 표현하게 되고, 이와 관련한 이벤트나 메시지와 같은 오퍼레이션 관련 요소들이 주요 검토 대상이 된다. 이때 이들 동적 모델은 크게 두 가지로 구분해 볼 수 있는데, 하나는 객체를 중심으로 한 모델과 객체를 중심으로 하지 않은 모델로 나눌 수 있다.

1. 객체를 중심으로 한 모델의 검토

정적 모델과 매우 밀접한 관계를 갖고 있기 때문에 정적 모델과 상호 보완적인 특성이 있다.

2. 객체를 중심으로 하지 않은 모델의 검토

대부분 유스케이스 모델을 기반으로 유스케이스와 밀접한 관계를 가지고 있는 다른 다이어그램과 연관관계를 가지는 특성이 있다.

(1) 유스케이스 모델 기반 액티비티 다이어그램

- (가) 업무관련 자료(매뉴얼, 기존 시스템,...)와 유스케이스 다이어그램을 보면서 비즈니스 프로세스를 확인하고 비즈니스 프로세스를 수행하기 위한 주체가 스웸레인에 적절히 표현되었는지 확인한다.
- (나) 유스케이스 수준 액티비티 다이어그램을 검토할 때는 유스케이스 기술서를 바탕으로 액터와 시스템간 상호작용에 대한 주요 내용이 액티비티로 충분히 표현되었는지 확인한다.

(2) 유스케이스 모델 기반 인터랙션(특히 시퀀스) 다이어그램

- (가) 유스케이스 다이어그램과 정적 모델을 기반으로 액터 및 객체가 찾아지는지 확인하고, 유스케이스 기술서에 있는 시나리오대로 표현되었는지 순서대로 메시지들을 따라가 보며 검토한다.
- (나) 인터랙션 다이어그램은 시나리오대로 따라가는 것 외에 동적 모델에 충분한 표현이 되었어도 정적 모델에 반영이 안된 사항이 있다면 이를 반영해야한다. 즉 필요한 시나리오가 인터랙션 다이어그램에 표현되었다면 인터랙션 다이어그램에 정의된 메시지가 클래스 다이어그램에 오퍼레이션으로 정의되었는지. 그리고 메시지를 주고 받을 수 있는 관계가 설정되었는지 확인해야한다.

(3) 유스케이스 모델 기반 스테이트차트 다이어그램

- (가) 스테이트차트 다이어그램에 표현된 상태 내부에 액티비티가 정의되어 있는 경우, 해당 액티비티가 정의된 다른 클래스를 찾아 확인해야 한다. 대체로 상태 내부 액티비티는 해당 클래스에서 생성한 객체가 다른 클래스에서 생성된 객체에게 메시지를 보내기 때문이다.
- (나) 특정 이벤트에 의해 상태가 변경될 경우 이벤트가 별도의 오퍼레이션 형태로 구현되어야 한다면 해당 클래스에 오퍼레이션이 정의되었는지도 확인해야한다.

② 상세설계된 동적모델을 대상으로 체크리스트를 정의하여 그에 따라 검증한다.

수행 tip

- 동적 모델 작성 시 활용된 여러 다이어그램이 갖는 고유 목적과 모델들 간 연관 관계를 충분히 이해하여 선택된 다이어그램의 적정성과 유스케이스 내부 흐름이 동적 모델에 제대로 반영되었는지를 검증하는 것이 중요하다.

2-2. 동적 설계모델 상세화

학습 목표

- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위한 상태차트를 작성할 수 있다.
- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위해 시퀀스 다이어그램을 작성할 수 있다.
- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위해 협동 다이어그램을 통해 유스케이스를 상세화 할 수 있다.

필요 지식 /

① 동적 모델링

1. 정의

시스템이 제공하는 유스케이스가 어떤 객체와 교류를 통하여 동적측면에서 기능을 수행하는지를 분석하는 모델링을 의미하고, 이는 메시지를 통해서 오브젝트들이 어떻게 상호작용하는지 표현된다.

2. 동적 모델링 특징

(1) 강조하고자 하는 내용에 따라 다양한 다이어그램으로 표현할 수 있다.

(가) 시퀀스 다이어그램은 시간의 흐름을 강조하는 경우 활용한다.

(나) 커뮤니케이션 다이어그램은 객체 조직의 구조를 강조하는 경우 활용한다.

(2) 유스케이스가 실행환경에서 어떻게 동작되는지를 표현한다.

(3) 클래스 모델링에서 도출되지 않은 객체를 추가로 발견할 수 있는 기회를 제공한다.

(4) 인터랙션 모델은 프로그래밍 도중에 바뀔 수 있기는 하지만, 프로그래밍을 시작할 수 있는 출발점이 된다.

(5) 유스케이스, 클래스, 인터랙션 모델링을 병행하여 진행하면, 더욱 완성도 있고 좋은 설계모델을 작성할 수 있다.

3. 유스케이스 실체화(Realization)

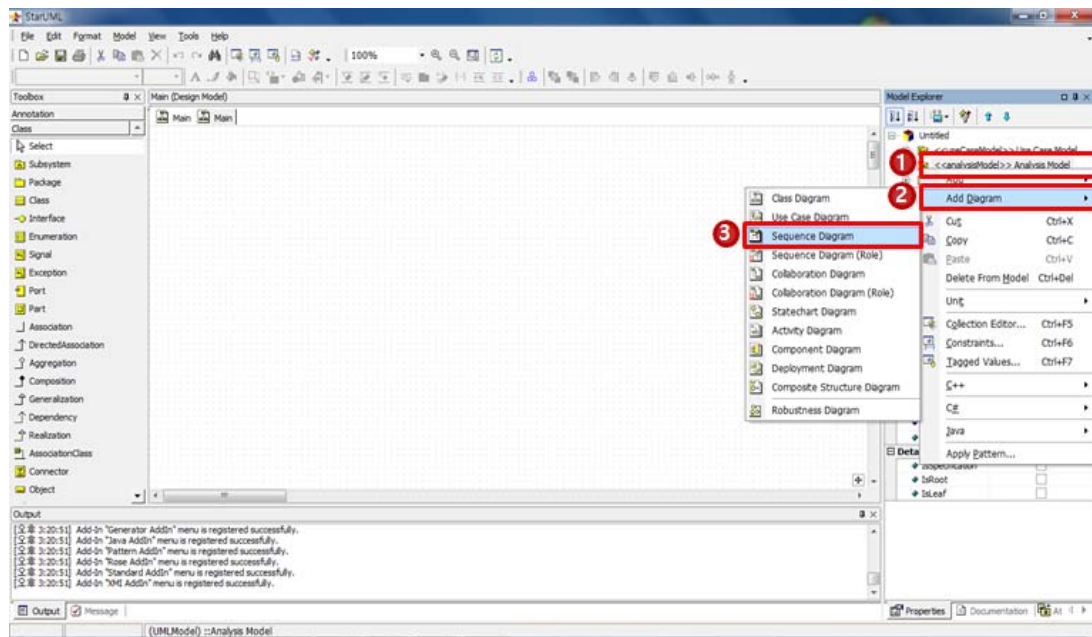
유스케이스 기술서의 내용 분석을 통해 그 내부에 표현된 책임을 클래스에 할당하고 객체들간의 상호작용을 다이어그램으로 표현할 수 있음을 의미한다.

② 설계용 소프트웨어(모델링 도구)를 활용한 동적 모델 작성

현존하는 모델링 도구 중 사용하기 편하고, 오픈 소스로 배포되는 도구 중 StarUML을 기준으로 동적모델 중 대표적 다이어그램인 시퀀스 다이어그램을 작성해 보도록 한다.

1. 시퀀스 다이어그램을 작성하기 위해 다이어그램 유형을 선택한다.

도구의 메인화면 우측 상단에 있는 Model Explorer의 <<AnalysisModel>> Analysis Model 선택 -> 마우스 오른쪽 버튼 클릭-> Add Diagram -> Sequence Diagram 선택



[그림 2-2] 시퀀스 다이어그램 선택

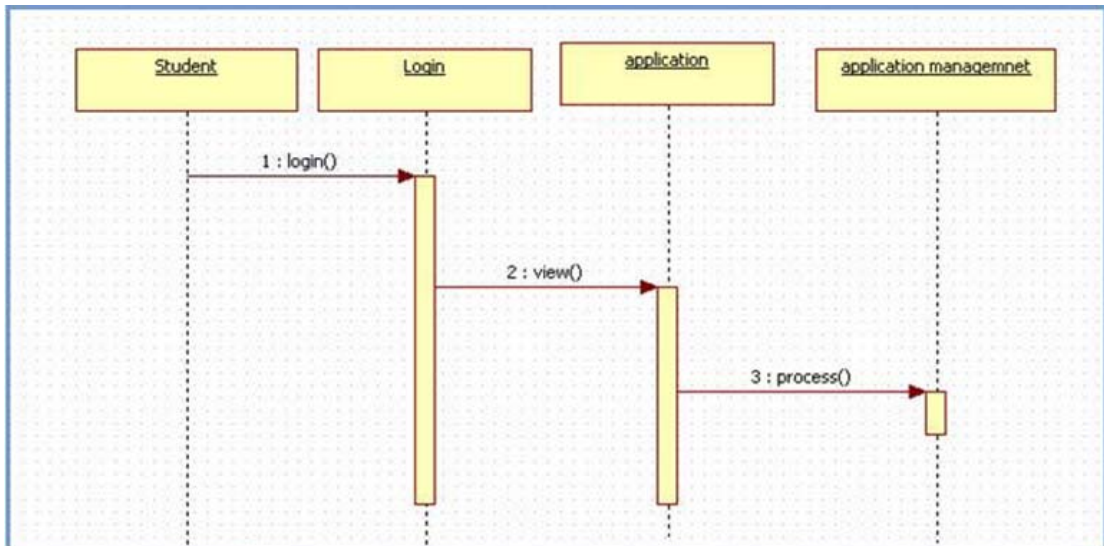
2. 시퀀스 다이어그램을 작성하기 위해 사용해야 할 Tool Bar의 유형과 세부 기능은 [그림 2-3]과 같다.

	<table border="1"> <tr> <td>Select</td><td>Diagram 요소를 선택합니다.</td></tr> <tr> <td>Object</td><td>클래스로부터 생성된 객체를 의미합니다.</td></tr> <tr> <td>Stimulus</td><td>두 객체간의 커뮤니케이션을 의미합니다.</td></tr> <tr> <td>SelfStimulus</td><td>한 객체가 스스로 커뮤니케이션 할을 의미합니다.</td></tr> <tr> <td>Combined Fragment</td><td>Fragment 영역을 의미합니다.</td></tr> <tr> <td>Interaction Operand</td><td>조각 내부를 구분할 때 사용합니다.</td></tr> <tr> <td>Frame</td><td>시퀀스 다이어그램을 특정 영역으로 구분할 때 사용합니다.</td></tr> </table>	Select	Diagram 요소를 선택합니다.	Object	클래스로부터 생성된 객체를 의미합니다.	Stimulus	두 객체간의 커뮤니케이션을 의미합니다.	SelfStimulus	한 객체가 스스로 커뮤니케이션 할을 의미합니다.	Combined Fragment	Fragment 영역을 의미합니다.	Interaction Operand	조각 내부를 구분할 때 사용합니다.	Frame	시퀀스 다이어그램을 특정 영역으로 구분할 때 사용합니다.
Select	Diagram 요소를 선택합니다.														
Object	클래스로부터 생성된 객체를 의미합니다.														
Stimulus	두 객체간의 커뮤니케이션을 의미합니다.														
SelfStimulus	한 객체가 스스로 커뮤니케이션 할을 의미합니다.														
Combined Fragment	Fragment 영역을 의미합니다.														
Interaction Operand	조각 내부를 구분할 때 사용합니다.														
Frame	시퀀스 다이어그램을 특정 영역으로 구분할 때 사용합니다.														

[그림 2-3] 시퀀스 다이어그램 Tool Bar

3. 아래의 수강신청 관련 지문내용을 토대로 클래스 다이어그램을 도식하면 [그림 2-4]와 같다.

대기 상태에서 수강신청을 하기 위해 수강신청 페이지에 접속한다.
로그인을 하고 과목목록을 확인한 뒤 과목을 선택한다.
선택한 과목의 수업을 선택하면 수강신청이 완료된다.



[그림 2-4] 시퀀스 다이어그램 작성 예시

수행 내용 / 정적 설계모델 상세화 하기

재료 · 자료

- UML 작성표준, 화이트 보드, 모델링 검토기준

기기(장비 · 공구)

- 컴퓨터, 프린터, 설계용 소프트웨어

안전 · 유의사항

- 실습 후에는 실습 기기 및 컴퓨터 전원을 끈다.

수행 순서

객체 지향 시스템에서 일을 수행하는 주체는 객체이며, 객체들은 메시지를 주거나 받음으로써 서로 상호작용을 한다. 시퀀스 다이어그램은 객체들간 상호작용을 시간적인 흐름으로 하나씩 나열한 것이며, 커뮤니케이션 다이어그램은 객체들 간 주고받는 메시지들을 함께 표현하면서 상호작용하는 형태를 일괄적으로 보여준다.

객체들이 상호작용을 할 때 주고받는 메시지는 객체 내부 상황을 변화시키는데, 메시지에 따른 객체 내부 상황 변화에 중점을 둔 다이어그램이 스테이트차트 다이어그램이다. 이때 메시지에 대한 오퍼레이션이 복잡할 경우 이를 액티비티 다이어그램으로 복잡한 내부 로직을 자세히 표현한다.

따라서 동적 모델은 객체가 어떻게 서로 상호작용하면서 내부 상태를 유지 또는 변화시키는지를 구체적으로 표현하는 것이라 할 수 있으며, 동적 모델링 순서는 표현하고자 하는 내용에 따라 그 내용이 포함되어 표현될 적절한 다이어그램을 선택하여 도식하는 것이 무엇보다 중요하다고 할 수 있다.

① 동적 모델링 순서

1. 모델링 목적에 따라 어떤 다이어그램을 그릴 것인지를 선택한다.
동적(시퀀스, 커뮤니케이션, 액티비티, 상태머신 등) 다이어그램 특성을 고려하여 선택하여야 한다.
2. 유스케이스 기술서 또는 텍스트 형태의 시나리오를 기반으로 관련된 액터와 객체를 나열하고 객체들 간의 메시지를 표현한다.
3. 클래스 다이어그램과의 상호 연계성을 검토한다.
4. 상호작용을 통해 파악된 정보를 통해 클래스 다이어그램을 개선한다.
5. 클래스 다이어그램에 Operation이 정의된 것이 있다면 인터랙션 다이어그램에 표현할 메시지 이름과 동일하게 사용하고 있는지 점검하여 상이할 경우, 일치시킨다.

② 메시지의 시간적 흐름을 강조하고자 할 경우, 시퀀스 다이어그램을 활용한다.

1. 시퀀스 다이어그램은 유스케이스 다이어그램, 유스케이스 기술서, 클래스 다이어그램, 아키텍처 기술서 등을 기반으로 액터와 여러 객체들 간 이벤트 또는 메시지 흐름을 시간적인 순서로 나열한다. 이때 메시지 이름에는 얻고자 하는 정보 또는 서비스에 대한 구체적인 사항을 잘 표현해야하며, 필요한 경우에는 매개변수나 반환값을 함께 표현한다.
2. 분석 단계에는 각 객체들이 액터로부터 어떤 이벤트를 받고 어떤 메시지를 전달하는지에 대한 표현이 중요하지만, 설계 단계에서는 얼마나 효과적으로 상호작용하는

지를 나타낼 필요가 있다. 즉 분석단계에서는 유스케이스가 수행하는 기능에 대한 논리 수준 상호작용이 중요하다면 설계단계에서는 실행 환경을 바탕으로 때로는 가장 단순한, 때로는 가장 체계적인 행위 표현이 중요하다.

3. 분석단계에는 액터로부터 관련된 모든 객체들을 나열하는 것이 일반적이며, 설계 단계에서는 전체를 표현할 수도 있지만 관심 영역에 대한 부분 모델링도 필요하다. 따라서 행위 모델을 만들기 전에 어떤 행위 표현이 필요한지를 명확히 이해하고 목적에 맞는 모델을 작성해야 한다.

4. 시퀀스 다이어그램 작성 순서

분석이나 설계 모델은 몇 가지 점에서 차이가 있지만, 일반적으로 시퀀스 다이어그램을 작성하는 순서는 몇 가지 단계를 거칠 필요가 있다.

(1) 먼저 상황을 이해한다.

(가) 주어진 상황에 유스케이스 다이어그램이나 유스케이스 기술서와 같은 요구사항 모델이 있는지, 클래스 다이어그램과 같은 구조 모델이 있는지를 파악한다.

(나) 요구사항 모델만 있다면 이를 토대로 초기 분석 시퀀스 다이어그램을 그릴 수 있고, 분석 구조 모델이 있다면 시퀀스 다이어그램을 분석 단계에 필요한 수준까지 그릴 수 있고, 설계 구조 모델과 아키텍처 또는 품질 관련 이슈에 대한 정책이 있다면 구체적인 설계 수준의 시퀀스 다이어그램을 그릴 수 있다.

(2) 주어진 상황에서 UML 모델링 요소들을 파악한다.

(가) 액터, 유스케이스, 클래스 등은 시퀀스 다이어그램을 그릴 때 가장 기본적인 요소이다. 만약 설계 단계 시퀀스 다이어그램을 그린다면 아키텍처 정의에 나타난 레이어나 티어 또는 컴포넌트 등 아키텍처에서 고려된 UML 모델링 요소에 속하지 않은 요소들도 파악해야 한다.

(나) 예로 들어, 유스케이스 다이어그램에서는 액터가 필요한 모델링 요소이고, 클래스 모델에서는 바운더리 클래스, 컨트롤 클래스들, 엔터티 클래스들 모두가 모델 작성에 필요한 모델링 요소들이다.

(3) 목적에 맞는 모델을 작성한다.

(가) 상황에 대한 이해와 파악된 모델링 요소를 근간으로 목적에 맞는 시퀀스 다이어그램에 꼭 필요한 모델링 요소를 나열하고, 그들 간의 메시지 전달 과정을 시간 순으로 표현하는 것이다.

(나) 모델링 목적이 논리적인 수준에서 액터로부터 시스템 내부를 구현하는 여러 클래스들 간 행위를 특정 시나리오에 대해 표현하는 것이라 가정해 보자. 이 경우에는 유스케이스 다이어그램과 클래스 다이어그램에 표현된 모델링

요소들 중 특정 시나리오와 관련 있는 액터와 클래스들을 배치하고 유스케이스 기술서 또는 클래스 내부에 표현된 오퍼레이션을 기반으로 메시지를 표현한다.

(다) 주어진 모델에 없지만 특정 시나리오에 대한 흐름을 표현할 때 필요한 메시지가 있다면 시퀀스 다이어그램에 추가적인 메시지를 표현한다.

(라) 여러 다양한 업무에 대한 클라이언트들에 필요한 업무라면 모든 클라이언트들에 대한 행위를 표현할 필요까지는 없다. 이 경우에는 모델링 목적이 클라이언트 상황과는 상관없이 비즈니스 레이어 내부 흐름에 대한 특정 업무에 대한 구체적인 행위만을 표현하는 것이 될 것이고, 레이어 내부에 있는 모델링 요소(인터페이스, 클래스 등)만을 선택해서 시퀀스 다이어그램을 그린다.

(마) 반대로 특정 업무를 구현한 여러 업무 로직이 있고 클라이언트는 내부 업무 로직 구현이 어떤 것이 되어도 상관없이 일을 해야 한다면 액터로부터 내부 로직 전까지만 표현 할 필요가 있다.

③ 객체간 상호작용하는 구조를 강조할 경우, 커뮤니케이션 다이어그램을 활용한다.

1. 시퀀스 다이어그램과 협동 다이어그램을 함께 인터랙션 다이어그램이라 부르는 데는 이유가 있다. 모델링하는 근본 요소가 객체이면서 상호작용에 대한 표현 정도가 메시지 수준으로 공통적이며, 시나리오를 표현하는 표현력도 매우 비슷하다.
2. 하지만 UML 모델링에 별도 다이어그램으로 존재하는 이유는 분명히 있다. 앞에서 살펴본 시퀀스 다이어그램은 시간적 흐름을 중심으로 메시지를 나열하지만 커뮤니케이션 다이어그램은 객체들 간 메시지들을 함께 표현하면서 객체들간 상호작용하는 패턴 또는 관계를 한눈에 직관적으로 파악할 수 있게 해 준다. 즉 시간 변화에 따른 상호작용 표현이 필요한 것이 아닌 '함께 일하는 상황'에 대한 표현이 필요한 것이다. 이같은 이유 때문에 커뮤니케이션 다이어그램은 독자적인 모델로서 존재하는 것이다.
3. 따라서 커뮤니케이션 다이어그램을 작성하는 이유는 단지 메시지 교환만을 보고 싶어서가 아닌 협업에 대한 유형 또는 상황을 파악하려는 데 있는 것이고, 구조 모델에 갖고 있는 관계에 대한 체계를 정제하려는 의도까지 함께 내포하고 있는 것으로 보아야 한다.
4. 커뮤니케이션 다이어그램 작성 순서
커뮤니케이션 다이어그램은 일반적으로 시퀀스 다이어그램을 작성하는 요령과 동일한 작성 순서에 따라 작성하면 된다.

- (1) 시퀀스 다이어그램과 마찬가지로 커뮤니케이션 다이어그램을 그리려면 먼저 모델을 그리기 전에 주어진 상황을 이해해야 한다.
 - (가) 커뮤니케이션 다이어그램을 그리기 위해 유스케이스 다이어그램이나 유스케이스 기술서와 같은 요구사항 모델이 있는지 클래스 다이어그램과 같은 구조 모델이 있는지를 파악한다.
 - (나) 요구사항 모델만 있다면 이를 토대로 초기 클래스 다이어그램을 작성하는 작업을 선행하고, 분석 구조 모델이 있다면 시퀀스 다이어그램을 분석 단계에 필요한 수준까지 그릴 수 있다.
 - (다) 설계 구조 모델과 아키텍처 또는 품질 관련 이슈에 대한 정책이 있다면 구체적인 설계 수준의 커뮤니케이션 다이어그램까지도 그릴 수 있다.
- (2) 다음 단계는 주어진 상황에서 커뮤니케이션 다이어그램을 그릴 때 필요한 UML 모델링 요소를 파악하는 것이다. 여기에는 액터, 바운더리 클래스, 컨트롤 클래스들, 엔터티 클래스들이 있다.
- (3) 마지막 단계는 목적에 맞는 모델을 작성하는 것이다. 필요한 모델은 액터와 설계 상황에 관여하는 모든 객체들이 나열되어야 하고, 메시지에 대한 표현은 구체적인 매개 변수 타입이나 반환값 등이 개발환경에 맞춘 표현이어야 한다.

④ 업무의 흐름을 파악하고자 하는 경우, 액티비티 다이어그램을 활용한다.

1. 액티비티 다이어그램은 예전부터 사용해왔던 흐름도(Flow-Chart)와 매우 유사한 모양을 띄고 있다. 흐름도는 어떤 행위에 대한 워크플로(Workflow)를 표현하는 대표적인 수단이지만 특정 객체에 대한 행위를 나타낼 수 없기 때문에 객체지향 관련 모델링에는 적합하지 않다. 이를 객체지향 스타일로 개선시킨 형태가 액티비티 다이어그램이라 할 수 있다.
2. 객체지향 스타일 모델을 상위 수준뿐만 아니라 하위 수준으로도 표현하기 위해 객체나 조직 또는 역할에 대한 행위를 표현할 수 있도록 스웜레인(Swimlane)을 추가했다. 이때 액티비티 다이어그램에 표현된 행위를 액티비티라고 하는데, 액티비티란 어느 정도 시간이 걸리는 행위를 말한다.
3. 프로젝트 초기에는 주로 유스케이스 수준 또는 그 상위 수준에서 비즈니스 프로세스를 표현하고, 분석 단계에서는 유스케이스 내부에 대한 구체적인 흐름을 표현하며, 설계 단계에서는 클래스 내부 오퍼레이션에 대한 알고리즘이나 구체적인 로직을 표현할 수도 있다. 따라서 액티비티 다이어그램을 잘 활용하면 다양한 목적에 대한 모델을 만들 수 있다.

4. 액티비티 다이어그램은 시퀀스 다이어그램이나 커뮤니케이션 다이어그램에서 표현하기 어려운 상황을 묘사하거나 보완사항을 표현하는 다이어그램이고, 이후에 설명할 상태머신 다이어그램과도 많은 관련을 갖고 있는 다이어그램이다.

5. 액티비티 다이어그램 작성 순서

액티비티 다이어그램은 동적 모델에 속하기 때문에 앞에서 설명한 일반적인 동적 모델링 작성 요령과 매우 유사하지만 다양한 목적에 따라 액티비티 다이어그램을 작성할 수 있기 때문에 이에 대한 작성 순서는 모델을 작성하는 목적별로 나누어 기술하도록 하겠다.

(1) 유스케이스 상위 수준 액티비티 다이어그램 작성 순서

(가) 액티비티 다이어그램을 작성할 때 다른 동적 모델을 만드는 과정과 가장 다른 특징 중 하나는 요구사항을 정의하는 단계에 활용할 수 있다는 것이다. 즉 유스케이스 다이어그램을 그리는 시점과 유사하게 비즈니스 프로세스를 파악하면서 액티비티 다이어그램으로 표현하는 것은 중요한 작업이 될 수 있다.

(나) 요구사항을 정의하는 시점은 UML 기반 모델이 만들어지기 전이거나 유스케이스 모델이 만들어지고 있는 시점이다. 따라서 참조할 수 있는 근거 및 상황도 UML 관련 모델이 아니라 컨설팅 결과, 업무 매뉴얼, 현행 시스템 등 다양한 것들이 될 수 있고, 인터뷰와 같은 보다 적극적인 활동도 필요할 수 있다. 이 점이 여태까지 소개했던 행위 모델 작성 요령과 근본적으로 다른 부분이다.

(다) 유스케이스 상위 수준 액티비티 다이어그램을 만들거나 모델링을 더욱 명확하게 하려면 현업 담당자 또는 시스템 사용자와 지속적으로 대화하면서 UML로 만들어지지 않은 다른 관련 자료들을 참조하여야 하고, 일반적인 모델링 기법보다는 업무에 대해 잘 알거나 의사소통 기술이 뛰어난 사람이 유스케이스 상위 수준 액티비티 다이어그램을 작성할 필요가 있다.

(라) 만약 컨설팅 결과나 업무 매뉴얼 내에 작성하고자 하는 비즈니스 흐름과 유사한 내용이 있다면, 이들을 토대로 액티비티 다이어그램을 작성하고 현업 또는 사용자에게 확인하는 방법을 사용한다면 보다 쉽고 명확하게 유스케이스 상위 수준 액티비티 다이어그램을 작성할 수 있다.

(2) 유스케이스 수준 액티비티 다이어그램 작성 순서

(가) 일반적으로 유스케이스 수준 액티비티 다이어그램을 모든 유스케이스에 대해 그릴 필요는 없고, 업무 흐름이 비교적 복잡하거나 중요한 유스케이스에 한해 그리는 것이 바람직하다.

(나) 유스케이스 모델링에는 유스케이스를 파악한 이후 구조화를 하거나 중요도를 파악하는 절차가 있는데, 이 결과를 바탕으로 모델 작성 여부를 결정하면 된다.

(다) 액티비티 다이어그램 작성여부를 결정하는 사항은 중요도뿐만 아니라 유스케이스 크기와의 관련성 문제이다. 유스케이스 모델링에 있어 액터에게 의미있는 일의 단위라는 말은 매우 모호한 경우가 많다. 예를 들어 어떤 액터에게 매우 간단한 기능이 하나의 작업단위가 되지만 또 다른 액터에게는 여러 기능이 하나로 묶여지지 않으면 의미있는 작업단위가 되지 못하는 경우도 있기 때문이다. 이런 이유로 아주 사소한 정보 관리는 조회, 생성, 갱신 등 여러 업무를 통합해서 한 유스케이스로 만들기도 하고 한 유스케이스에 여러 유스케이스를 포함(include)시키기도 하는 것이다.

(라) 따라서 한 유스케이스에 대해 여러 유스케이스가 관계를 맺고 있거나 한 유스케이스 기술서 내용이 아주 많아 이를 한눈에 파악하기 어려운 경우가 있다면 유스케이스 수준 액티비티 다이어그램을 작성할 필요가 있다.

(3) 클래스 수준 액티비티 다이어그램 작성 순서

(가) 클래스 수준 동적 모델은 상황에 따라 상태머신 다이어그램으로 표현하는 것이 더 좋을 수도 있고, 액티비티 다이어그램을 사용하는 것이 더 좋을 수도 있다.

(나) 시퀀스 다이어그램과 커뮤니케이션 다이어그램이 매우 유사한 행위 모델을 표현하기는 하지만 각 모델들이 갖는 의미가 있듯이 액티비티 다이어그램과 상태머신 다이어그램도 나름대로 의미를 갖고 있다. 그 차이 중의 하나는 액티비티 중심이고 하나는 상태중심이라는 것이다.

(다) 액티비티 다이어그램은 이벤트나 클래스가 갖는 속성에는 크게 관심을 두지 않지만 객체들이 상태를 유지하거나 상태를 변경시킬 수 있는 것은 속성과 속성값을 변화시킬 수 있는 오퍼레이션 때문이므로 상태머신 다이어그램에서는 속성과 이벤트에 대해 매우 큰 관심을 두고 있음을 이해하여야 한다.

(라) 클래스 수준에서 작성되는 액티비티 다이어그램은 클래스에 정의된 오퍼레이션 수준에서 액티비티들을 표현하므로 인터랙션 다이어그램처럼 여러 객체들의 행위 활동보다는 한 클래스가 행하는 일만을 구체적으로 알고 싶을 때에 액티비티 다이어그램을 사용하면 좋다.

⑤ 객체의 상태 변화를 강조할 경우, 상태머신 다이어그램을 활용한다.

1. 상태머신 다이어그램은 앞에서 소개한 여러 행위 모델들과 많은 관련이 있는데, 특히 클래스 수준 액티비티 다이어그램과 표현 형식은 다르지만 표현 수준은 매우

비슷하다.

2. 상태머신 다이어그램은 주로 클래스가 수행하는 행위 관점을 표현하지만, 유스케이스 수준 또는 서브 시스템, 컴포넌트 등에 대한 상태 변화를 표현할 수도 있다. 이처럼 스테이트차트는 특정 요소가 외부 요소들로부터 다양한 이벤트를 받는 경우 복잡한 상태 변화를 한눈에 알 수 있도록 보여주는 역할을 한다.
3. 대체로 상태머신 다이어그램은 가장 활용빈도가 낮은 모델이고, 일반적으로 클래스 수준의 상태머신을 많이 활용하므로 이를 중심으로 알아보기로 한다.

4. 상태머신 다이어그램 작성 순서

- (1) 상태머신 다이어그램을 작성할때 파악해야 할 상황으로 가장 중요한 것은 클래스와 클래스가 갖는 속성들이다.

정적 모델이 적절하게 작성되었다면 대부분 클래스가 맡은 역할을 수행하기 위해 필요한 속성들을 이미 파악했을 것이므로, 이를 충분히 활용하여 작성 시 참조하여야 한다.

- (2) 상태머신 다이어그램을 그릴 때 가장 먼저 해야 할 것은 상태표현이 필요한 클래스를 선정하는 것이다.

대체로 복잡한 처리 절차나 많은 속성을 갖는 클래스들이 선정되는데, 상태머신 다이어그램을 그리면서 상태 관리에 필요한 속성 또는 오퍼레이션을 추가로 찾기도 한다.

- (3) 상태머신 다이어그램 작성에서 가장 어려운 점은 파악해야 할 상태 수준이 어떤 정도인가 하는 것과 상태 변화를 일으키는 이벤트 파악이다.

- (4) 특히 상태를 어느 정도로 자세히 표현할 것인지는 모델을 작성하는 목적에 따라 다르다.

(가) 만약 클래스가 갖는 속성 및 오퍼레이션을 검증하기 위한 목적이 강하다면 가급적 자세한 상태 표현이 필요하지만, 클래스가 취할 수 있는 주요 상태들을 참고하는 것이라면 상위 수준의 상태 표현만으로 충분할 수도 있다.

(나) 하지만 클래스 수준 상태머신 다이어그램을 그릴 때는 대체로 세부적인 표현까지 필요한 경우가 많다. 따라서 클래스가 가질 수 있는 모든 하부 상태를 파악하고 상태 전이에 관련된 이벤트와 조건을 모두 파악하는 수준으로 상태머신 다이어그램을 표현하는 것이 필요하다.

수행 tip

- UML 표준에 따라서 업무지식과 유스케이스 실체화를 위한 작성목적에 부합하는 다이어그램을 선택하여 동적 설계모델이 작성될 수 있도록 한다.

학습 2 교수 · 학습 방법

교수 방법

- 교수자의 주도로 업무 흐름에 따라 설계와 관련된 항목을 파악하고, 동적 모델을 검증하는 등의 내용을 PPT 자료로 작성하여 설명한다.
- 소프트웨어 아키텍처 설계 가이드라인을 참조하여 동적모델 상세설계의 내역을 확인하고, 검증하는 과정을 정리하여 설명한다.
- 모델링 도구들에 대한 이해 및 이전 사용 경험 여부에 대해 파악한 후 수업을 진행하고, 필요시 모델링 도구 매뉴얼 및 관련 홈페이지 등을 방문하여 관련 자료를 제시한 후 설명한다.
- 모델링 도구의 설치 과정과 다이어그램 작성에 대해 단계적 실습이 이루어질 수 있도록 지도한다.
- 특히 동적 모델링을 위한 다양한 다이어그램의 용도를 명시적으로 제시함으로써, 동적 모델 작성 목적에 부합하는 다이어그램을 직관적으로 선택할 수 있도록 지도한다.

학습 방법

- UML과 모델링 도구의 필요성 및 활용 용도에 대해 이해한다.
- 소프트웨어 아키텍처 설계 가이드라인을 참조하여 동적모델 상세설계의 내역을 확인하고, 검증하는 과정을 학습하고 실습한다.
- 동적 모델링을 위한 다양한 다이어그램의 특성과 차이점에 대해 이해한다.
- 모델링을 위한 각종 다이어그램을 작성하는 과정과 도구활용법을 이해하고 실습한다.
- 모델링 도구를 여러 개발자들이 공유하여 형상관리할 수 있는 가이드와 자료를 찾아 실습할 수 있도록 한다.

학습 2 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
동적모델 검증	- 소프트웨어 아키텍처 설계 가이드라인을 참조하여, 동적모델 상세설계의 내역 확인 및 검증할 수 있다.			
동적 설계모델 상세화	- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위한 상태차트를 작성할 수 있다.			
	- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위해 시퀀스 다이어그램을 작성할 수 있다.			
	- 동적모델 설계를 상세 분석하여, 응용소프트웨어의 구현을 위해 협동 다이어그램을 통해 유스케이스를 상세화 할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
동적모델 검증	- 업무흐름을 명확히 작성하기 위한 모델 도식 목적에 부합하는 다이어그램 선택의 적정성			
	- 설계 가이드라인을 고려한 동적모델의 상세 설계내용 이해			
동적 설계모델 상세화	- 동적 설계모델의 구체화를 통한 완전성			

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
동적모델 검증	- 업무흐름을 명확히 작성하기 위한 모델 도식 목적에 부합하는 다이어그램 선택의 적정성			
	- 소프트웨어 아키텍처를 고려한 동적모델 설계 이해			
동적 설계모델 상세화	- 동적 설계모델의 완전성			
	- 동적 설계모델의 요구사항 추적성 확보			

피드백

1. 체크리스트를 통한 관찰

- 실습 과정에서 체크리스트에 따라 평가한 후, 개선해야 할 사항 등을 정리하여 돌려준다.
- 소프트웨어 아키텍처 설계 가이드라인을 참조하여, 동적모델 상세설계의 내역을 확인하고 검증하여 미비사항을 정리하여 돌려준다.

2. 포트폴리오

- 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
- 소프트웨어 아키텍처에 기반한 동적모델 설계인지를 파악한 후, 미비사항에 대해 돌려준다.
- 동적 설계모델의 요구사항 반영 및 추적성을 파악하여 미비사항에 대해 돌려준다.

학습 1	정적모델 상세설계하기(LM2001020202_14v2.1)
학습 2	동적모델 상세설계하기(LM2001020202_14v2.2)
학습 3	공통 모듈 설계하기 (LM2001020202_14v2.3)
학습 4	타 시스템 연동하기(LM2001020202_14v2.4)

3-1. 공통 모듈 상세화

학습 목표

- 재사용성 확보와 중복개발을 회피하기 위하여, 전체 시스템 차원과 단위 시스템 차원의 공통 부분을 식별하여 이에 대한 상세 명세를 작성할 수 있다.
- 개발할 응용소프트웨어의 전반적인 기능과 구조를 이해하기 쉬운 크기로 공통 모듈을 설계할 수 있다.
- 소프트웨어 측정지표 중 모듈 간의 결합도는 줄이고, 개별 모듈들의 내부 응집도는 높이기 위한 공통 모듈을 설계할 수 있다.
- 전반적인 처리 논리 구조에 예기치 못한 영향을 끼치지 않도록 공통 모듈 인터페이스의 인덱스 번호나 기능 코드를 설계할 수 있다.

필요 지식 /

① 모듈화

모듈화란, 시스템을 분해하고 추상화하여 소프트웨어의 성능을 향상시키거나 시스템의 디버깅, 시험, 통합 및 수정을 용이하도록 하는 소프트웨어 설계 기법으로서, 모듈은 서브시스템, 서브루틴, 소프트웨어 내의 프로그램 혹은 작업단위 등을 의미한다.

1. 원칙

- (1) 프로그램을 모듈로 세분화시킬 때에 지켜야 할 원칙으로는 우선 모듈 간의 독립성과 모듈 내의 응집성을 들 수 있다. 모듈 간의 독립성이란 모듈끼리의 커플링(Coupling) 관계를 낮게 하는 것으로서 이를 지향하는 이유는 다음과 같다.

(가) 파급효과 방지

어떤 모듈에 잘못이 있으면 이의 영향이 그 모듈에 국한되어야 한다.

(나) 유지보수의 용이

하나의 모듈을 변경할 경우 다른 모듈의 세부사항까지 알아야 한다면 그만큼 유지보수가 어려워진다.

- (2) 모듈 간의 연결을 최소화하기 위해서는 모듈 간의 관계가 데이터 교환에 국한되어야 한다. 모듈 내의 응집성은 한 모듈 내에 있는 요소들의 기능적 연관성을 뜻하는 것으로 여러 가지 잡다한 기능들을 한 모듈에 놓아두어서는 안된다는 것이다. 따라서 한 모듈에 한 가지 문제의 해결식으로 프로그램을 모듈화시켜 나가는 것이 필요하다.

2. 효과적인 모듈화 설계 방안

- (1) 결합도는 줄이고 응집도는 높여서 모듈의 독립성을 높인다.
- (2) 모듈의 제어 영역 안에서 그 모듈의 영향 영역을 유지시킨다.
- (3) 모듈의 기능은 예측이 가능해야 하며 지나치게 제한적이어서는 안된다.
- (4) 복잡도와 중복성을 줄이고 일관성을 유지시킨다.
- (5) 모듈 크기는 시스템 전반적인 기능과 구조를 이해하기 쉬운 크기로 분해하여야 한다.
- (6) 하나의 입구와 하나의 출구를 갖도록 해야 한다.
- (7) 인덱스 번호나 기능 코드들이 전반적인 처리 논리 구조에 예기치 못한 영향을 끼치지 않도록 모듈 인터페이스를 설계해야 한다.
- (8) 유지보수가 용이해야 한다.

② 모듈화를 위한 SW설계를 위한 지침

1. 결합도

- (1) 모듈 사이는 가능한 한 약하게 결합되어야 한다.(낮은 결합도)
- (2) 모듈 사이의 자료교환은 명시적이어야 한다.

2. 응집도

- (1) 모듈 내의 명령문들은 강력하게 응집되어야 한다.(높은 응집도)
- (2) 가능한 한 모듈은 기능 응집을 갖도록 설계하여야 한다.

3. 시스템 구조의 구현

- (1) 시스템의 구조는 균형을 유지해야 한다.
- (2) 전체적으로 각 경로의 계층이 비슷하게 설계되어야 한다.
- (3) 입력, 처리, 출력의 세부분으로 구분하여 배치하여야 한다.
- (4) 상위 모듈은 관리, 하위 모듈은 자료를 우선하여 처리한다.

4. 모듈 크기 : 모듈의 크기는 한번에 처리하기 쉬워야 한다.

5. 제어폭 : 호출하는 하위 모듈의 수가 적정해야 한다.

6. 근접성

- (1) 관련된 모듈은 가까이 배치해야 한다.
- (2) 자료는 입력된 근처에서 처리한다.
- (3) 오류가 발생한 모듈에서 오류를 처리한다.
- (4) 작업의 결정과 실행 부분을 가까이 배치한다.

③ 모듈 결합도와 응집도

모듈 설계에 대한 평가의 기준으로, 모듈 사이의 관계를 결정하는 데 도움을 주는 결합도와 응집도가 있다. 일반적으로 각각의 모듈은 결합도가 낮을수록 또한 응집도는 높을수록 모듈분할이 잘 이루어졌다고 말할 수 있다.

1. 모듈 결합도

모듈 결합도는 소프트웨어 구조 내에서 모듈 간 관련성을 측정하는 척도로서, 모듈 간의 상호 연계되는 의존도를 측정하는 요소이다. 결합도에 영향을 미치는 요소는 세 가지로서 모듈 간에 전달되는 항목 수, 제어의 자료량, 그리고 모듈 간에 공유하고 있는 광역 자료 요소의 수 등이며, 낮은 결합도를 구현하는 방법으로는 불필요한 관련성을 제거하고, 불필요한 인터페이스를 가능한 한 제거하며, 필요한 관계를 연결하는 등의 방법이 있다.

모듈 결합도의 종류를 모듈의 독립성이 높은 순서(결합도가 약한 순서)로 나타내면, 자료 / 스탬프 / 제어 / 외부 / 공통 / 내용 결합도 순이라 할 수 있다.

(1) 자료 결합도(Data Coupling)

- (가) 자료 결합도는 모듈 간 인터페이스가 자료 요소로만 구성될 때의 결합도이다.
- (나) 어떤 모듈이 다른 모듈을 호출하면서 매개 변수나 인수로 데이터를 넘겨주고, 호출받은 모듈은 받은 데이터에 대한 처리 결과를 다시 돌려주는 것이다.
- (다) 자료 결합도는 모듈 간의 내용을 전혀 알 필요가 없는 상태로서 한 모듈의 내용을 변경하더라도 다른 모듈에는 전혀 영향을 미치지 않는 가장 바람직한 결합도이다.
- (라) 유실 자료(Trap Data)의 발생 가능성이 있다.

(2) 스탬프 결합도(Stamp Coupling)

- (가) 스탬프 결합도는 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도이다.
- (나) 두 모듈이 동일한 자료 구조를 조회하는 경우의 결합도이며 자료 구조의 어떠한 변화, 즉 포맷이나 구조의 변화는 그것을 조회하는 모든 모듈 및 변화되는 필드

를 실제로 조회하지 않는 모듈에도 영향을 미치게 된다.

- (다) 관련 없는 모듈 간 의존성이 발생할 수 있고, 불필요한 레코드가 존재할 수 있다.

(3) 제어 결합도(Control Coupling)

(가) 제어 결합도는 한 모듈에서 다른 모듈로 논리적인 흐름을 제어하는 데 사용하는 제어 요소(Function Code, Switch, Tag, Flag)가 전달될 때의 결합도이다.

(나) 상위 모듈이 하위 모듈의 상세한 처리 절차를 알고 있어 이를 통제하는 경우나 처리 기능이 두 모듈에 분리되어 설계된 경우에 발생한다.

- (다) 권리 전도(Inversion of Authority) 현상이 발생할 수 있다.

(4) 외부 결합도(External Coupling)

(가) 외부 결합도는 어떤 모듈에서 외부로 선언한 데이터(변수)를 다른 모듈에서 참조할 때의 결합도이다.

(나) 참조되는 데이터의 범위를 각 모듈에서 제한할 수 있다.

(5) 공통 결합도(Common Coupling)

(가) 공통 결합도는 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도이다.

(나) 공통 데이터 영역의 내용을 조금만 변경하더라도 이를 사용하는 모든 모듈에 영향을 미치므로 모듈의 독립성을 약하게 만든다.

- (다) 타 모듈로 오류의 전파 가능성이 크다.

(6) 내용 결합도(Content Coupling)

(가) 내용 결합도는 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도이다.

(나) 한 모듈에서 다른 모듈의 중간으로 분기되는 경우에도 내용 결합도에 해당된다.

- (다) 한 모듈의 작은 변화가 다른 모듈에 중대한 영향을 끼칠 수 있다.

2. 모듈 응집도

응집도는 정보은닉의 개념을 확장시켜 놓은 것으로, 하나의 모듈 내부의 처리 요소들간의 기능적 연관성을 측정하는 척도이자 한 모듈 내에 있는 구성요소의 기능적 관련성을 평가하는 기준이며, 모듈에 있는 기능적인 응집력을 측정하는 방법이다. 이상적인 모듈은 프로시저 내에서 단일작업만을 수행하는 모듈을 의미하며, 응집도는 높을수록 좋은 모듈이 된다.

응집도가 높은 순서로 그 종류를 나타내면 기능적 / 순차적 / 통신적 / 절차적 / 시간적 / 논리적 / 우연적 응집도 순이라 할 수 있다.

(1) 기능적 응집도(Functional Cohesion)

모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도이다.

(2) 순차적 응집도(Sequential Cohesion)

모듈 내 하나의 활동으로부터 나온 출력 데이터를 그다음 활동의 입력 데이터로 사용할 경우의 응집도이다.

(3) 교환(통신)적 응집도(Communication Cohesion)

동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도이다.

(4) 절차적 응집도(Procedural Cohesion)

모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도이다.

(5) 시간적 응집도(Temporal Cohesion)

특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우의 응집도이다.

(6) 논리적 응집도(Logical Cohesion)

유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도이다.

(7) 우연적 응집도(Coincidental Cohesion)

모듈 내부의 각 구성 요소들이 서로 관련없는 요소로만 구성된 경우의 응집도이다.

④ 모듈별 구현 실체화 유형

개발 프로젝트에서 자주 활용되는 모듈의 구현 실체화 유형은 다음과 같다.

1. 라이브러리(Library)

(1) 링킹(Linking) 단위로서, 오브젝트(Object) 형태로 존재한다.

(2) 주체가 아닌 보조자 역할을 수행하여, 단독적으로 수행이 불가능하다.

(3) 일반적으로 언어(Language)에 종속적인 함수 형태로 구현된다. (예) Classes, Interfaces

(4) 공통 함수 또는 기능을 구현하는 실체화 유형이다.

(예) String 처리 함수, Data Format 함수, 일자처리 함수, 공통 스크립트 등

(5) 일반적으로 C 언어에서 자주 사용하는 공통 모듈을 구현하는 수단으로 활용된다.

2. 공통 클래스(Class)

(1) 객체(Object)란 실세계에 존재하거나 또는 개념적인 것을 표현한 것으로, 상태(State),

행위(behavior), 유일성(identity)을 갖는다.

(2) 구조나 행위가 유사한 객체들은 같은 공통 클래스로 정의하여 재사용한다.

3. 컴포넌트(Component)

(1) 인터페이스를 통해 서비스를 제공하는 연관관계가 많은 오브젝트들의 집합이다.

(2) 런타임 수준의 대표적인 재사용 기술이다.

(3) 독립적으로 실행 가능한 단위이다.

(4) 개발된 컴포넌트를 조립하여 소프트웨어를 개발하는 CBD(Component Based Development) 개발기법의 재사용 기초가 된다.

4. 프레임워크(Framework)

(1) 응용 프로그램 표준 구조를 구현하는 클래스와 라이브러리의 모임이다.

(2) 대규모 재사용을 통해 개발생산성을 보장하고, 품질 향상 및 비용을 절감할 수 있다.

(3) 오픈 소스 프레임워크로 스프링 프레임워크가 자주 활용된다.

수행 내용 / 공통 모듈 상세화 하기

재료 · 자료

- 프로그램 분할 기준, 재사용모듈 정의 및 도출 기준, 재사용 모듈에 대한 개발표준이나 지침

기기(장비 · 공구)

- 컴퓨터, 인터넷, 설계용 소프트웨어

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

재사용을 위한 공통 모듈 상세화를 위해서는 먼저, 모듈화를 위한 프로그램 분할 기준을 명확히 정의한 후, 확정된 사용자 요구사항을 정의된 모듈화 분할 기준에 따라 모듈단위로 적절하게

분할될 수 있도록 하며, 분할된 각 모듈에 대한 구현 실체화 유형을 정의함으로써 구현모듈을 상세화 하게 된다.

① 모듈화를 위한 프로그램 분할 기준을 정의한다.

시스템 개발을 위해 프로그램 분할 기준이 적절하지 못한 경우 공통 모듈의 도출과 처리방식 결정이 부적절하게 되어 시스템 개발의 효율성이 저하된다. 또한 사용자 편의성 측면에서 불편함이 야기되므로, 프로그램의 분할이 적절하게 이루어지도록 사전에 분할기준을 명확하게 정의하는 것이 필요하다.

1. 공통 모듈 도출에 대한 기준을 정의한다.

- (1) 기존의 재사용 모듈 라이브러리와 재사용 모듈에 대한 개발 표준이나 지침이 존재하는지 검토한다.
- (2) 새로운 재사용 모듈에 대한 정의와 도출 기준을 검토 및 정의하고, 검토 결과가 개발 표준 및 지침에 반영될 수 있도록 한다.

2. 응용아키텍처 구조를 반영하여 프로그램 분할 기준을 정의한다.

응용아키텍처 구조를 반영하여 프로그램 분할 기준을 정의하고, 응용아키텍처 정의서, 개발 표준 지침서에 이를 반영하도록 한다.

② 사용자와 합의한 요구사항을 정의된 모듈화 분할 기준에 따라 모듈단위로 적절하게 분할될 수 있도록 한다.

1. 공통 모듈 도출 기준에 따라 적정하게 도출한다.

- (1) 공통 모듈 도출 기준(지침)에 따라 공통 모듈을 적정하게 도출하고, 설계지침, 프로그램 사양서에 이에 대한 내용이 반영될 수 있도록 한다.
- (2) 업무의 흐름에 따라 업무기능단위의 처리과정에서 재사용이 가능한 업무 로직이 공통 모듈로 모두 도출되었는지 응용흐름, 프로그램 사양서를 통해 확인한다.

2. 분할된 프로그램 단위가 서로 일관성이 있도록 확인한다.

- (1) 프로그램 사양서에서 프로그램 단위가 표준 및 지침의 내용을 준수하여 일관성을 확보하고 있는지 확인한다.
- (2) 분할된 프로그램 단위가 낮은 모듈 결합도와 높은 모듈 응집도를 기준으로 모듈 기능의 독립성을 확보하고 있는지 확인한다.
- (3) 분할된 개개의 프로그램 단위가 시스템 구조상에서 상호 균형을 유지하고 있는지 확인한다.

③ 분할된 각 모듈에 대한 구현 실체화 유형을 정의한다.

전통적으로 분할된 모듈에 대한 구현 실체화 유형은 다양한 형태로 존재하고, 진화 방향성은 점차 대규모 재사용을 위한 큰 규모의 구현 실체화 유형으로 진화하고 있음을 고려하여 각 모듈별 특성과 재사용 정도를 염두에 두고 아래와 같은 구현 실체화 유형을 정의하여야 한다.

1. 라이브러리(Library)
2. 공통 클래스(Class)
3. 컴포넌트(Component)
4. 프레임워크(Framework)

수행 tip

- 공통 모듈 상세화를 위해서는 모듈화를 위한 분할 기준을 정의하고, 사용자와 합의된 요구사항을 대상으로 정의된 기준에 따라 모듈단위로 적절하게 분할하는 과정을 통해 공통 모듈을 도출하는 것이 중요하다.

학습 3 교수 · 학습 방법

교수 방법

- 교수자의 주도로 모듈화를 위한 분할 기준을 정의하고, 사용자와 합의된 요구사항을 대상으로 정의된 기준에 따라 모듈단위로 적절하게 분할하는 과정을 통해 공통 모듈을 도출하는 등의 내용을 PPT 자료로 작성하여 설명한다.
- 도출된 공통 모듈에 대한 상세 명세화하는 과정을 정리하여 설명한다.
- 모듈화를 위한 결합도와 응집도에 대한 이해를 목적으로 실습이 이루어질 수 있도록 지도한다.
- 식별된 공통 모듈을 실체화할 수 있는 구현 유형별 특성을 이해함으로써, 해당 모듈의 목적에 부합하는 구현 유형을 직관적으로 선택할 수 있도록 지도한다.
- 공통 모듈 인터페이스의 인덱스 번호나 기능코드 필요성 및 설계시 고려사항을 정리하여 설명한다.

학습 방법

- 모듈화를 위한 프로그램 분할 기준에 이해한다.
- 사용자 요구사항을 대상으로 정의된 분할 기준에 따라 모듈단위로 적절히 분할하는 과정에 대해 이해한다.
- 도출된 공통 모듈에 대한 상세 명세화하는 과정에 대해 이해하고 실습한다.
- 모듈화 원칙이라 할 수 있는 결합도와 응집도에 대해 이해하고 실습한다.
- 식별된 공통 모듈을 실체화하는 구현 유형별 특성을 이해하여 구현 유형을 선택할 수 있도록 한다.

학습 3 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 상세화	- 재사용성 확보와 중복개발을 회피하기 위하여, 전체 시스템 차원과 단위 시스템 차원의 공통 부분을 식별하여 이에 대한 상세 명세를 작성할 수 있다.			
	- 개발할 응용소프트웨어의 전반적인 기능과 구조를 이해하기 쉬운 크기로 공통 모듈을 설계할 수 있다.			
	- 소프트웨어 측정지표 중 모듈 간의 결합도는 줄이고 개별 모듈들의 내부 응집도는 높이기 위한 공통 모듈을 설계할 수 있다.			
	- 전반적인 처리 논리 구조에 여기치 못한 영향을 끼치지 않도록 공통 모듈 인터페이스의 인덱스 번호나 기능 코드를 설계할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 상세화	- 공통 모듈을 식별기준의 적절성			
	- 공통 모듈 크기의 적절성			
	- 공통 모듈 식별 원칙인 결합도와 응집도 고려 적절성			
	- 공통 모듈 설계서의 완전성			

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
공통 모듈 상세화	- 공통 모듈을 식별기준의 적절성			
	- 공통 모듈 크기의 적절성			
	- 공통 모듈 식별 원칙인 결합도와 응집도 고려 적절성			
	- 공통 모듈 설계서의 완전성			

피드백

1. 체크리스트를 통한 관찰
 - 실습 과정에서 체크리스트에 따라 평가한 후, 개선해야 할 사항 등을 정리하여 돌려준다.
 - 결합도와 응집도 판단을 위한 가이드라인을 정리하여 돌려준다.
2. 포트폴리오
 - 제출한 보고서를 평가한 후에 주요 사항에 대하여 표시하여 돌려준다.
 - 도출된 공통 모듈에 대한 상세 명세화하는 과정을 검토해 보고, 미비사항을 정리하여 돌려준다.

학습 1	정적모델 상세설계하기(LM2001020202_14v2.1)
학습 2	동적모델 상세설계하기(LM2001020202_14v2.2)
학습 3	공통 모듈 설계하기(LM2001020202_14v2.3)

학습 4

타 시스템 연동하기 (LM2001020202_14v2.4)

4-1. 시스템 연동 상세화

학습 목표

- 소프트웨어 아키텍처에서 정의한 타 시스템 연동 리스트 및 연동 방안을 참조하여, 타 시스템 연동 상세 설계의 가이드라인을 작성할 수 있다.
- 소프트웨어 아키텍처의 정의를 반영한 연동 상세 설계 가이드라인에 따라, 타 시스템 연동 상세 설계할 수 있다.
- 소프트웨어 아키텍처에 따라 선정된 개발 및 운영 환경에 사용될 기술영역별 미들웨어/솔루션에 대하여 명세를 작성할 수 있다.

필요 지식 /

① 내외부 시스템 인터페이스 방식 유형

1. 인터페이스 방식 유형
1. 공유 데이터베이스
2. 인터페이스 파일
3. 메시지
4. EAI 솔루션 활용

② 내외부 시스템 인터페이스 방식 결정 시 고려사항

1. 데이터베이스를 공유하는 것은 인터페이스하는 외부시스템과의 사이를 가장 가깝게 연결함으로써 메시지 오버헤드를 피하여 자원 이용과 경과시간을 절약할 수 있으며, 내부 시스템간의 연계를 위한 인터페이스 설계는 주로 데이터베이스 공유로 이루어지는 경우가 많다. 이런 경우에는 주로 테이블과 프로그램간의 상호연관관계(CRUD 매트릭스) 확인을 통해 설계의 적정성을 판단하여야 하며, 데이터의 참조관계가 명확하게 정의되어 있는지 확인하여야 한다.

2. 메시지를 통한 온라인 통신은 유연성과 시기적절한 응답을 필요로 할 때 적절하다.
3. 인터페이스 파일은 분 또는 초 단위보다는 일 단위로 측정되는 일괄처리 반환에 적합하다.
4. EAI 솔루션 사용은 자동화 프로그램을 이용하고 있으므로, 인터페이스에 대한 상호검증이 편리하다.
 - (1) 분산된 시스템 간에 걸쳐 실시간 이벤트 처리방식으로 정보 흐름이 가능하도록 설계하는 것이 중요하다.
 - (2) 메시지 전송(Messaging), 데이터 인터페이스/교환(Adapters), 변환 및 라우팅(Transformation & Routing), 비즈니스 프로세스와 오류처리(Business Flow) 등을 위한 초기 데이터 세팅이 적절하게 수행하여야 한다.

③ 신규 시스템과 기존 시스템과의 인터페이스를 위해서 기존 시스템에 변경을 요구하게 될 때의 고려사항

1. 인터페이스를 위한 변경사항이 교환 프로토콜과 제어에 대한 각각의 시스템이 가지는 책임에 대해 구체적으로 기술하여야 한다.
2. 기존 시스템의 변경이 수용될 수 없는 경우를 위한 대책이 마련되어 있어야 한다.

④ 내외부 인터페이스 설계서 작성시 포함 항목

1. 내외부 인터페이스 목록

- (1) 송신 인터페이스번호 : 송신 시스템의 인터페이스 일련번호를 기입한다.
- (2) 송신 일련번호 : 한 개 송신단위에서 여러 개의 서브시스템으로 동시에 전송되는 경우에는 순차적으로 기술한다.
- (3) 송신 시스템명 : 송신 시스템이름을 기술한다.
- (4) 송신 프로그램 ID : 송신에 해당하는 프로그램 ID를 기입한다.
- (5) 전달 처리형태 : 인터페이스를 처리하는 형태를 기술한다. Batch / Online 등
- (6) 전달 인터페이스방식 : 통신 프로토콜 및 통신 기술 방식을 기술한다.
- (7) 전달 발생빈도 : 인터페이스 발생빈도를 기술한다. “회수/주기”의 형식으로 기술한다.
- (8) 수신 상대 담당자 : 수신 시스템의 업무담당자명을 기술한다.
- (9) 수신 프로그램 ID : 수신과 관련된 프로그램 ID를 기입한다.
- (10) 수신시스템명: 인터페이스 수신 시스템명을 기술한다.

- (11) 수신 일련번호: 수신 시스템의 동일 인터페이스가 여러 시스템에서 동시에 수신을 받는 경우에 순차적으로 번호를 부여한다.
- (12) 관련 요구사항 ID: 해당 인터페이스와 관련된 분석단계의 “사용자 요구사항 정의서”의 요구사항 ID를 기입한다.
- (13) 비고: 특이사항 등을 기입한다.

2. 내외부 인터페이스 명세

- (1) 인터페이스 번호: 송신 시스템의 인터페이스 일련번호를 기입한다.
- (2) 데이터송신시스템 시스템명: 송신 시스템명을 기술한다.
- (3) 데이터송신시스템 데이터저장소명: 인터페이스 송신과 관련된 엔터티 또는 파일명을 기술한다.
- (4) 데이터송신시스템 속성명: 관련 엔터티의 속성명 또는 파일의 항목명을 기술한다.
- (5) 데이터송신시스템 데이터타입: 엔터티 속성 또는 항목 타입을 기술한다.
- (6) 데이터송신시스템 길이: 데이터의 길이를 기술한다.
- (7) 송신 프로그램 ID: 송신에 해당하는 프로그램 ID를 기입한다.
- (8) 데이터수신시스템 데이터저장소명: 인터페이스 송수신과 관련된 엔터티 또는 파일명을 기술한다.
- (9) 데이터수신시스템 속성명: 관련 엔터티의 속성명 또는 파일의 항목명을 기술한다.
- (10) 데이터수신시스템 데이터타입: 엔터티 속성 또는 항목 타입을 기술한다.
- (11) 데이터수신시스템 길이: 데이터의 길이를 기술한다.
- (12) 데이터수신시스템 시스템명: 수신 시스템명을 기술한다.
- (13) 수신 프로그램 ID: 수신에 해당하는 프로그램 ID를 기입한다.

수행 내용 / 시스템 연동 상세화하기

재료 · 자료

- 표준 및 절차 매뉴얼, 아키텍처 기술서

기기(장비 · 공구)

- 컴퓨터, 인터넷, 시스템 연동 소프트웨어(EAI, ESB 등)

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

시스템 연동 상세화를 위해서는 아키텍처에서 표준으로 정의한 연동해야 하는 내외부 시스템의 인터페이스 목록과 연동 방식에 대한 내용을 파악하는 작업이 선행되며, 각 인터페이스별 특성을 고려하여 연계방식(연계주기, 연계방법 등)을 구체적으로 설계하고, 이에 대한 구현가능성을 검토한 후 인터페이스 설계서를 작성하는 순서로 진행된다.

① 소프트웨어 아키텍처에서 정의한 내외부 시스템 인터페이스 목록 및 연동 방식, 설계 가이드라인을 참조하여 내외부 시스템 연계방식을 설계한다.

- 인터페이스 설계서에 아키텍처 정의서의 시스템 구성 및 네트워크 구성을 반영하여 내외부 시스템 연계 시 고려해야 하는 전제조건 및 제약사항을 정의한다.
- 인터페이스 설계서에 전제조건 및 제약사항을 반영하여 각각의 인터페이스를 위한 조직적(타 기관 또는 타 시스템의 운영 및 유지보수 조직), 기술적 사항(통신 프로토콜의 문제점, 운영체제 시스템간의 발생가능한 문제점 파악)들을 도출한다.
- 인터페이스 설계서에 각각의 인터페이스별로 연계방식을 정의한다.
- 인터페이스 설계서에서 정의한 연계방식과 아키텍처 정의서에서 정의된 시스템 간의 연관관계와의 일관성을 확인하고 검토하여 조정한다.
- 인터페이스 설계서에서 인터페이스를 위해 관련된 내외부 시스템 별 프로세스(업무 기능)와 대상자료, 제공자 등을 구체적으로 정의하고, 연계를 위한 사전 준비작업을 수행한다.

② 내외부 시스템과의 인터페이스를 위한 연계주기 및 방법 등에 따라 인터페이스를 세부적으로 설계하고, 구현 가능한지 검토한다.

1. 인터페이스 설계서에서 각각의 인터페이스별로 내외부 시스템간의 연계주기 및 송수신 데이터, 관련 프로그램 등을 정의한다.
2. 프로그램 목록 및 프로그램 사양서에 내외부 인터페이스 기능을 구현할 수 있도록 해당 프로그램을 정의한다.
3. 프로그램 사양서에 내외부 인터페이스를 구현하기 위해 정의된 대로 연계주기 및 세부 수행방법에 대한 처리 로직이 반영되도록 구체적으로 설계한다.
4. 프로그램 사양서에 내외부 인터페이스를 통해 송수신 되는 데이터 구조, 입출력형식 등이 정확하게 반영하고, 관련 테이블 및 파일을 정의한다.
5. 프로그램 사양서에서 내외부 인터페이스 데이터의 형식 및 구조와 해당 프로시저의 처리 로직이 일관성 있도록 설계한다.

③ 내외부 인터페이스 설계서를 작성한다.

식별된 내외부인터페이스를 송신측과 수신측으로 구분하여 기술하고, 내외부 시스템 인터페이스 연계 방식을 기술하며, 데이터 송신 시스템과 수신 시스템 간의 데이터 저장소와 속성 등의 상세 내역을 기술한다.

수행 tip

- 소프트웨어 아키텍처에서 표준으로 정의한 인터페이스 대상 목록과 방식에 대한 충분한 검토가 선행되어야 하며, 각 인터페이스별 특성을 고려하여 연계방식을 정의하는 것이 중요하다.

4-2. 시스템 연동 오류 관리

학습 목표

- 소프트웨어 아키텍처에 따른 시스템 간의 연동 시, 발생할 수 있는 오류를 예측하고, 이의 대응 방안에 대해 제시할 수 있다.

필요 지식 /

① 내외부 시스템 인터페이스 검증 방안

내외부 시스템 인터페이스에 대한 검증은 궁극적으로 관련 프로그램(업무기능)에 대한 단위 및 통합시험을 통해서 하는 것이 용이하지만, 설계단계에서는 내외부 인터페이스 정의서 및 프로그램 사양서를 바탕으로 시험 계획과 결과가 수행될 수 있을지 확인하는 것이 중요하며, 개발시 직접 내외부 시스템 간의 인터페이스 기능을 시험함으로써 구현의 적정성 및 충분성을 최종 확인할 수 있다.

1. 내부 시스템간 인터페이스의 최종 검증은 단위시험으로, 외부 시스템 간의 인터페이스는 통합 시험 수준으로 수행하는 것이 적합하다.
2. 요구사항 추적표를 통해 산출물간의 연관성을 확인하여 내외부 시스템 간의 인터페이스 정상 작동여부를 확인하는 것이 선행되는 것도 바람직하다.

② 비정상적인 상황에서의 장애대책과 복구대책을 적용한 내외부 시스템 인터페이스 기능의 수행여부 확인

1. 프로그램 소스(구현된 응용시스템) 또는 단위(통합) 시험을 통해 장애 및 비정상적 결과에 대해서 장애대책 및 복구대책에 의해 재처리가 이루어지는지 확인한다.
2. 프로그램 소스(구현된 응용시스템) 또는 단위(통합) 시험을 통해 장애 및 복구대책에 의해 재처리가 수행되는 경우, 인터페이스가 비정상적인 상황 이전의 정상적인 작업결과를 이어받아 연속적으로 작업이 수행되도록 하고 있는지 확인한다.

③ 내외부 시스템 인터페이스에 대한 상호검증체계로 자동화 도구 활용방안

1. 내외부 시스템 인터페이스에 대한 상호검증체계로 EAI나 ETL(ETCL)과 같은 자동화 도구를 사용하는 경우에는 사용자 또는 운영자가 인지할 수 있도록 메시지를 전달하는 메커니즘을 구현하는 것이 일반적이다.

2. 직접 응용프로그램에 의해 내외부 시스템 인터페이스를 구현하는 경우에는 실제로 시스템을 운영하면서 그 중요성을 인식하는 경우가 많으므로, 내외부 시스템 인터페이스에 대한 비정상적 상황에 대한 문제점과 위험을 인식시켜 장애대책과 복구대책을 마련할 수 있도록 하는 것이 필요하다.

수행 내용 / 시스템 연동 오류 관리 하기

재료 · 자료

- UML 작성표준, 아키텍처 정의서, 화이트 보드

기기(장비 · 공구)

- 컴퓨터, 프린터, 설계용 소프트웨어

안전 · 유의사항

- 실습 후에는 실습 기기 및 컴퓨터 전원을 끈다.

수행 순서

내외부 시스템 인터페이스 연동시 발생할 수 있는 오류는 무엇보다도 빠른 시간내에 식별하는 것이 가장 중요하므로, 이를 위해 인터페이스에 대한 상호검증체계를 설계하여 반영하여야 하며, 검증체계를 통해 발견된 장애 및 비정상적 결과에 대해서는 각 상황에 적절한 장애대책과 복구대책을 마련하여야 한다.

① 내외부 시스템 인터페이스에 대한 상호 검증체계를 설계한다.

내외부 시스템 인터페이스에 대한 결과를 확인할 수 있도록 다음 내용을 반영하여 설계한다.

1. 인터페이스 설계서에서 각각의 인터페이스별로 인터페이스 결과를 확인하는 방안을 정의한다.
2. 인터페이스 설계서에서 장애 및 비정상적인 상황을 시스템 사용자 또는 운영자가 인지할 수 있도록 설계한다.

3. 인터페이스 설계서에서 정의된 인터페이스 결과 확인 방안 중 자동화가 필요한 것에 대해 해당 처리 로직을 프로그램 사양서에 반영하여 설계한다.

② 내외부 인터페이스 시 발견된 장애 및 비정상적 결과에 대한 장애대책과 복구대책을 마련한다.

1. 인터페이스 설계서에 각각의 인터페이스 별로 장애 및 비정상적인 결과에 대한 대안 및 문제발생 후 재처리 방법 등을 고려하도록 한다.
2. 인터페이스 설계서 및 프로그램 사양서에서 장애 및 비정상적인 결과에 대한 처리방안이 실현가능하도록 조직 및 기술적 요소들을 고려하여 정의한다.
3. 프로그램 사양서에서 내외부 인터페이스 기능이 비정상적으로 종료되었을 경우, 관련 프로그램에서 불필요한 작업을 피할 수 있는 대안에 대한 처리로직을 반영하여 설계한다.

③ 내외부 시스템 인터페이스 검증 절차

내외부 시스템 인터페이스에 대한 상호 검증결과를 확인하는 체계를 정의함으로써 사전에 검증결과를 예측할 수 있다.

1. 향후 개발될 프로그램 소스(구현된 응용시스템)에서 내외부 시스템 인터페이스에 대한 결과를 확인할 수 있도록 상호검증 체계가 구현되도록 정의하고, 최종적으로는 단위(통합)시험을 수행하여 확인한다.
2. 단위(통합)시험 계획 및 결과서를 가지고 단위(통합) 시험을 수행하여 내외부 시스템의 인터페이스에 대한 정상 또는 장애, 비정상적인 상황에 대한 인지가 가능한지 예측한다.
3. 최종적 검증단계인 프로그램 소스(구현된 응용시스템)와 단위(통합)시험 결과에서 상호검증 체계에 대한 구현이 내외부 인터페이스 정의서 및 프로그램 사양서를 반영하여 분석단계, 설계단계의 내용과 일관성을 이루었는지 확인해야 하므로 설계단계에서 작성하는 인터페이스 정의서와 프로그램 사양서의 완성도를 높여야 한다.

수행 tip

- 내외부 시스템 인터페이스 시 발생가능한 오류는 사전에 이를 식별할 수 있는 검증체계가 필요하며, 이를 통해 파악된 오류에 대하여 장애 및 복구대책을 미리 마련하여 대응하는 것이 무엇보다 중요하다고 할 수 있다.

학습 4 교수 · 학습 방법

교수 방법

- 교수자의 주도로 업무 흐름에 따라 내외부 인터페이스 설계와 관련된 항목을 파악하고, 각 항목에 대한 구체적 내용을 PPT 자료로 작성하여 설명한다.
- 소프트웨어 아키텍처에 따라 선정된 개발 및 운영 환경에 사용될 기술영역별 미들웨어/솔루션에 대해 설명하고, 개발환경 구축을 위해 명세화 과정에 대해 정리하여 설명한다.
- 내외부 시스템 연계방식에 대해 관련 자료를 제시하고, 실제 사례를 통해 실무능력을 확보할 수 있도록 설명한다.
- 내외부 인터페이스시 발생가능한 오류에 대한 상호검증체계에 대해 구체적으로 설명한다.
- 내외부 인터페이스시 발견된 장애가 어떤 영향을 끼치는지와 복구의 중요성에 대해 상세히 설명한다.

학습 방법

- 내외부 시스템 연계방식에 이해한다.
- 소프트웨어 아키텍처에 따라 선정된 개발 및 운영 환경에 사용될 기술영역별 미들웨어/솔루션에 대해 이해하고, 개발환경 구축을 위해 명세화 과정에 대해 실습한다.
- 내외부 시스템 연계방식의 특성을 고려하여 방식 결정 시 고려사항을 숙지하여 상황별 적절한 연계방식을 선택할 수 있도록 한다.
- 내외부 인터페이스시 발생가능한 오류에 대한 상호검증체계에 대해 이해하여 이를 설계하기 위해 반영해야 할 항목을 숙지한다.
- 내외부 인터페이스 시 발견된 장애에 대하여 어떤 절차와 기법으로 복구하는지 이해한다.

학습 4 평가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
시스템 연동 상세화	- 소프트웨어 아키텍처에서 정의한 타 시스템 연동 리스트 및 연동 방안을 참조하여, 타 시스템 연동 상세 설계의 가이드라인을 작성할 수 있다.			
	- 소프트웨어 아키텍처의 정의를 반영한 연동 상세 설계 가이드라인에 따라, 타 시스템 연동 상세 설계를 할 수 있다.			
	- 소프트웨어 아키텍처에 따라 선정된 개발 및 운영 환경에 사용될 기술영역별 미들웨어/솔루션에 대하여 명세를 작성할 수 있다.			
시스템 연동 오류 관리	- 소프트웨어 아키텍처에 따른 시스템 간의 연동 시, 발생할 수 있는 오류를 예측하고 이의 대응 방안에 대해 제시할 수 있다.			

평가 방법

- 평가자 질문

학습내용	평가항목	성취수준		
		상	중	하
시스템 연동 상세화	- 소프트웨어 가이드라인에 따라 타 시스템 연동 상세 설계의 적정성			
	- 각 인터페이스 별 특성을 반영한 연계방식의 적정성			
시스템 연동 오류 관리	- 인터페이스 시 발생가능한 오류에 대한 사전 검증체계의 적정성			
	- 연동 오류로 인한 장애 및 복구대책의 완전성			

• 구두 발표

학습내용	평가항목	성취수준		
		상	중	하
시스템 연동 상세화	- 소프트웨어 가이드라인에 따라 타 시스템 연동 상세 설계의 적정성			
	- 각 인터페이스 별 특성을 반영한 연계방식의 적정성			
시스템 연동 오류 관리	- 인터페이스 시 발생가능한 오류에 대한 사전 검증체계의 적정성			
	- 연동 오류로 인한 장애 및 복구대책의 완전성			

피드백

1. 평가자 질문

- 학습 과정에서 제시한 내용을 중심으로 질문하고, 답변시 미비한 내용을 정리하여 돌려준다.
- 소프트웨어 가이드라인에 따라 설계된 타 시스템 연동 상세설계 내용을 검토하여 미비사항을 정리하여 돌려준다.

2. 구두 발표

- 제출한 발표자료와 발표내용을 평가한 후에 미비사항에 대해 정리하여 돌려준다.
- 인터페이스 시 발생가능 오류에 대한 사전 검증체계를 파악하여 미비사항을 정리하여 돌려준다.
- 인터페이스 연동 오류로 인한 장애 및 복구 대책을 검토하여 미비사항을 정리하여 돌려준다.



- 한국정보화진흥원(2009.5.28). 『정보시스템 감리지침』.
- 한국정보화진흥원(2011.12). 『CBD SW개발 표준 산출물 관리 가이드』.
- StarUML 다운받기, <http://sourceforge.net/projects/staruml/files/staruml/5.0/>에서 2015.10.10. 인출
- StarUML 한글 사용자 가이드, [http://staruml.sourceforge.net/docs/user-guide\(ko\)/toc.html](http://staruml.sourceforge.net/docs/user-guide(ko)/toc.html)에서 2015.10.10. 인출.

NCS 학습모듈 개발진

(대표집필자)

강석진(이비스툼)

(집필진)

김보운(이화여자대학교)

김홍진(LG CNS)

유은희

장현섭((주)커리텍)

주선태(T3Q)

진권기(이비스툼)

최재준

(검토진)

김승현(경희대학교)

엄기영(우리에프아이에스)

장온순(한국IT컨설팅)

조상욱(세종대학교)

조성호(삼성카드)

(개발기관)

최기원(한국소프트웨어기술진흥협회)

이두현(한국소프트웨어기술진흥협회)

(연구기관)

옥준필(한국직업능력개발원)

김상진(한국직업능력개발원)

김성남(한국직업능력개발원)

김지영(한국직업능력개발원)

문한나(한국직업능력개발원)

홍서희(한국직업능력개발원)

*표시는 NCS 개발진임

※ 본 학습모듈은 자격기본법 시행령 제8조 국가직무능력표준의 활용에 의거하여 개발하였으며
저작권법 25조에 따라 관리됩니다.

※ 본 학습모듈은 <http://www.ncs.go.kr>에서 확인 및 다운로드할 수 있습니다.



www.ncs.go.kr