

대분류/20
정보통신

중분류/01
정보기술

소분류/02
정보기술개발

세분류/05
NW엔지니어링

능력단위/08

NCS학습모듈

네트워크 프로그래밍

구현

LM2001020508_14v2



교육부

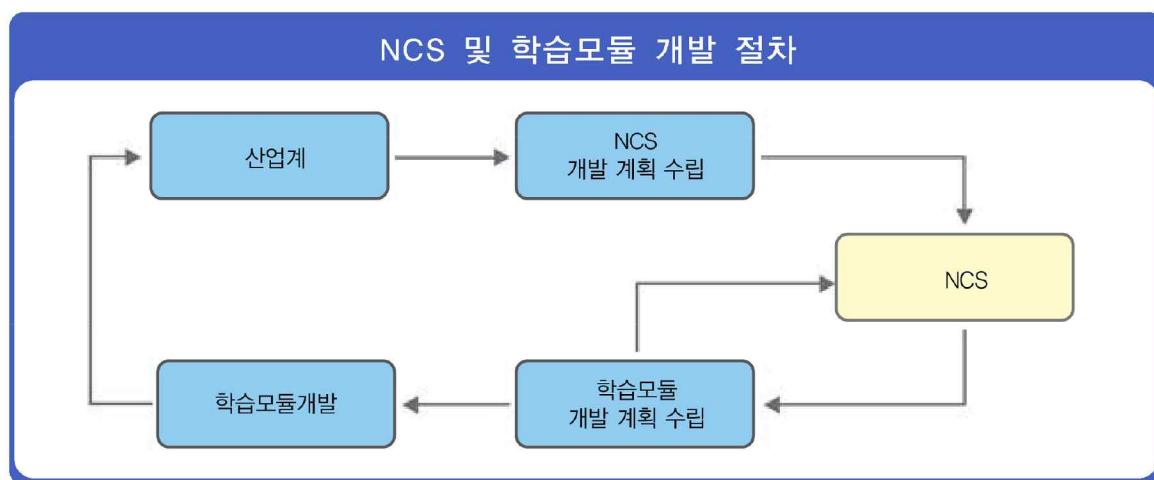
NCS 학습모듈은 교육훈련기관에서 출처를 명시하고 교육적 목적으로 활용할 수 있습니다. 다만 NCS 학습모듈에는 국가(교육부)가 저작재산권 일체를 보유하지 않은 저작물들(출처가 표기되어 있는 도표, 사진, 삽화, 도면 등)이 포함되어 있으므로 이러한 저작물들의 변형, 복제, 공연, 배포, 공중 송신 등과 이러한 저작물들을 활용한 2차 저작물의 생성을 위해서는 반드시 원작자의 동의를 받아야 합니다.

NCS 학습모듈의 이해

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

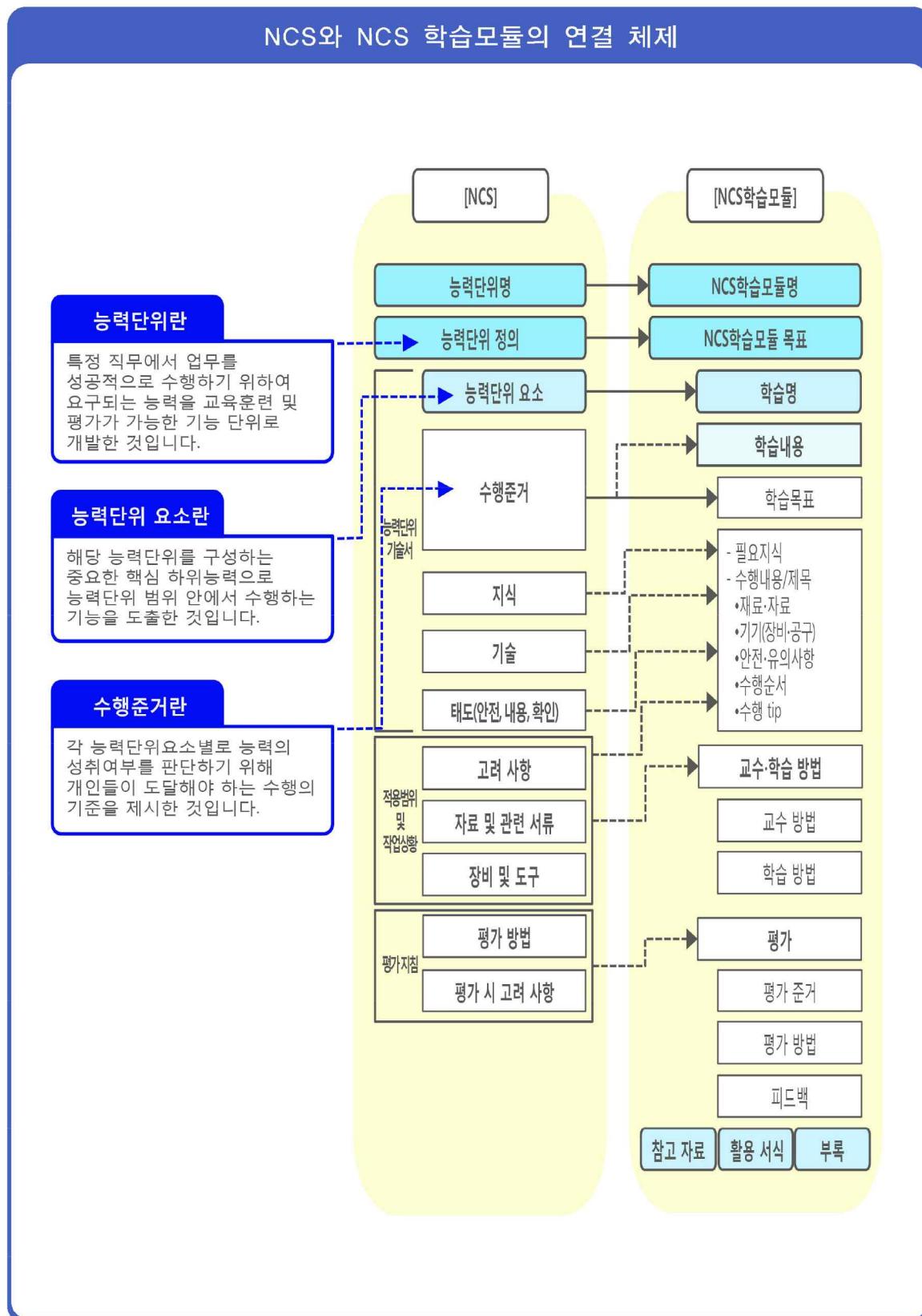
(1) NCS 학습모듈이란?

- 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 구체적 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 체계를 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/부록으로 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	이·미용 서비스

세분류

헤어미용	능력단위	학습모듈명
피부미용	네일 샵 위생 서비스	네일숍 위생서비스
메이크업	네일 화장을 제거	네일 화장을 제거
네일미용	네일 기본 관리	네일 기본관리
이용	네일 랩	네일 랩
	네일 팁	네일 팁
	젤 네일	젤 네일
	아크릴릭 네일	아크릴 네일
	평면 네일아트	평면 네일아트
	융합 네일아트	융합 네일아트
	네일 샵 운영관리	네일숍 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 학습 모듈로 나누어 개발할 수도 있습니다.

2. NCS 학습모듈의 개요



구성

- 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.



활용 안내

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, 큐티를 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

선수학습

네일숍 위생서비스(LM1201010401_14V2)

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 프리에지 형태 만들기	1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링	1201010403_12v2.1	프리에지 모양 만들기
2. 큐티를 정리하기	2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티를 관리	1201010403_14v2.2	큐티를 정리하기
3. 컬러링하기	3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업	1201010403_14v2.3	컬러링
4. 보습제 도포하기	4-1. 보습제 선정과 도포 4-2. 각질제거	1201010403_14v2.4	보습제 바르기
5. 네일 기본관리 마무리하기	5-1. 유분기 제거 5-2. 네일 기본관리 마무리와 청리	1201010403_14v2.5	마무리하기

학습모듈의 목표는

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

선수 학습은

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.

핵심 용어

프리에지, 니퍼, 퓨셔, 폴리시, 네일 파일, 스웨어형, 스웨어 오프형, 라운드형, 오발형, 포인트형

3. NCS 학습모듈의 내용 체계



- 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 상호작용이 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.



예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

학습 1	프리에지 형태 만들기(LM1201010403_14v2.1)
학습 2	큐티클 정리하기(LM1201010403_14v2.2)
학습 3 컬러링하기(LM1201010403_14v2.3)	
학습 4	보습제 도포하기(LM1201010403_14v2.4)
학습 5	네일 기본관리 미무리하기(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 기초적인 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.

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 일plex 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

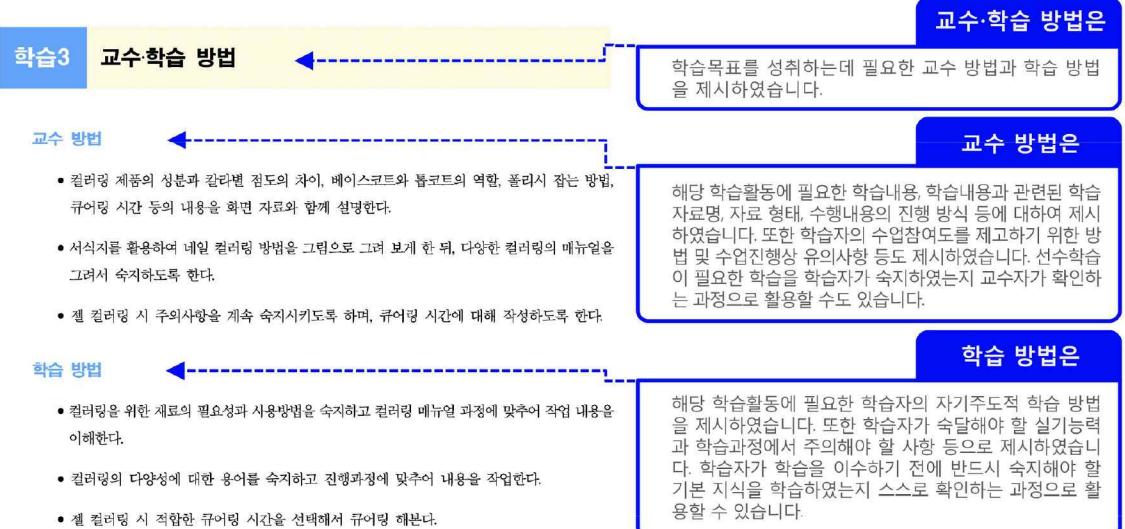
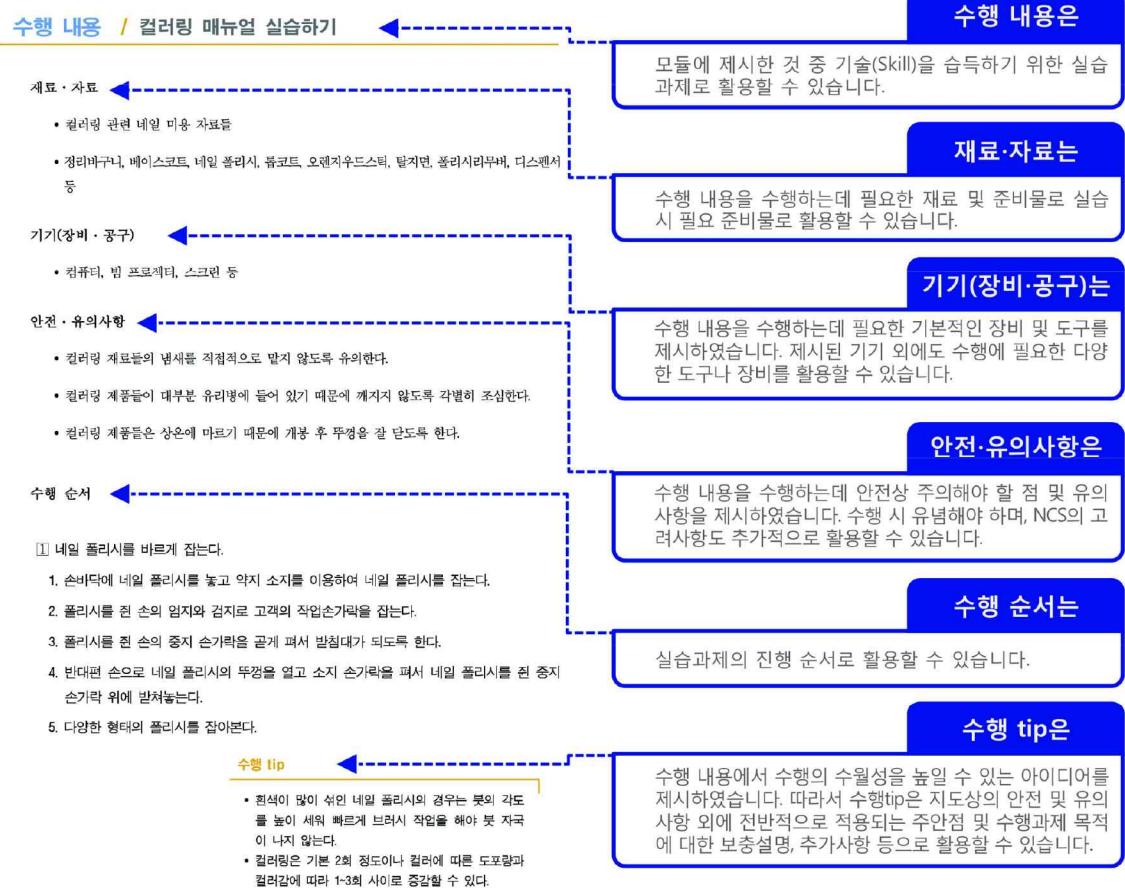
필요 지식 /

① 컬러링 매뉴얼

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

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.



학습3 평가

평가 준거

- 평가는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.

- 평가는 다음 사항을 평가해야 한다.

학습내용	학습목표	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

평가 방법

- 작업장 평가

학습내용	평가 항목	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

피드백

1. 작업장 평가

- 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

4. 참고 자료

참고 자료

• 김미원(2011).『Nail Study』. 서울: 사)한국네일자식서비스협회.

• 민방경(2015).『미용사(네일)필기』. 서울: 예문사.

• 박은주(2014).『네일미용』. 서울: 정담미디어.

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. - 발톱의 형태 활용 - 내인성 발톱의 보정시에 적용

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.

참고자료는

해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

활용서식은

평가 서식, 실험시트 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 미스트	흰색	작업자 착용
보호안경	투명한 렌즈 (안경으로 대체 가능)	작업자 착용
재료정리함	제질, 색상 무관	작업대

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

부록은

[NCS-학습모듈의 위치]

대분류	정보통신
중분류	정보기술
소분류	정보기술개발

세분류	능력단위	학습모듈명
SW아키텍처	네트워크 환경 분석	네트워크 환경 분석
응용SW 엔지니어링	네트워크 프로토콜 분석	네트워크 프로토콜 설계
임베디드SW 엔지니어링	네트워크 프로토콜 설계	네트워크 토폴로지 설계
DB엔지니어링	네트워크 토폴로지 설계	네트워크 자원 관리 설계
NW엔지니어링	네트워크 QoS 제어 설계	네트워크 QoS 제어 설계
보안 엔지니어링	네트워크 소프트웨어 아키텍처 수립	네트워크 소프트웨어 아키텍처 수립
UI/UX 엔지니어링	네트워크 소프트웨어 개발 방법 수립	네트워크 소프트웨어 개발 방법 수립
시스템SW 엔지니어링	네트워크 프로그래밍 구현	네트워크 프로그래밍 구현
빅데이터플랫폼 구축	네트워크 품질 평가	네트워크 품질 평가
핀테크 엔지니어링	네트워크 프로젝트 관리	네트워크 프로젝트 관리

차 례

학습모듈의 개요 1

학습 1. 개발환경 분석하기

1-1. 네트워크 개발환경 구축	3
1-2. 표준개발도구 사용 및 형상 수정, 보완	12
• 교수·학습 방법	20
• 평가	21

학습 2. 기능 구현하기

2-1. 네트워크 응용프로그램 및 프로토콜 구현	23
2-2. 데이터베이스 및 에이전트 구현	30
2-3. 네트워크 QoS(Quality of Service) 제공방안 구현	38
• 교수·학습 방법	46
• 평가	47

학습 3. 프로그램 디버깅하기

3-1. 네트워크 프로그램 시험	49
3-2. 네트워크 프로그램 디버깅 및 수정	59
• 교수·학습 방법	69
• 평가	70

학습 4. 프로그램 최적화하기

4-1. 삽입문구 작성 및 오류 최소화	72
4-2. 네트워크 프로그램 코드 최적화	82
• 교수·학습 방법	93

• 평가 ----- 94

참고 자료 ----- 96

활용 서식 ----- 97

네트워크 프로그래밍 구현 학습모듈의 개요

학습모듈의 목표

네트워크 프로그램을 구현하기 위한 네트워크 개발환경을 분석하고, 각 단계별로 요구되는 기능을 구현(코딩)하고, 테스트를 하는 능력과 프로그래밍 개발과정에서 발생할 수 있는 오류를 디버깅하여 프로그램을 최적화할 수 있다.

선수학습

SW아키텍처 이행(2001020106_14v2), 제품소프트웨어 패키징(2001020209_14v2), 데이터베이스 구현(2001020405_14v2), SQL활용(2001020410_14v2)

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 개발환경 분석하기	1-1. 네트워크 개발환경 구축	2001020508_14/21	개발환경 분석하기
	1-2. 표준개발도구 사용 및 형상 수정, 보완		
2. 기능 구현하기	2-1. 네트워크 응용프로그램 및 프로토콜 구현	2001020508_14/22	기능 구현하기
	2-2. 데이터베이스 및 에이전트 구현		
3. 프로그램 디버깅하기	2-3. 네트워크 QoS(Quality of Service) 제공방안 구현	2001020508_14/23	프로그램 디버깅하기
	3-1. 네트워크 프로그램 시험		
4. 프로그램 최적화하기	3-2. 네트워크 프로그램 디버깅 및 수정	2001020508_14/24	프로그램 최적화하기
	4-1. 삽입문구 작성 및 오류 최소화		
	4-2. 네트워크 프로그램 코드 최적화		

핵심 용어

네트워크 개발 환경분석, 네트워크 기능 구현, 네트워크 프로그램 디버깅, 네트워크 프로그램 최적화, 형상관리, 네트워크 QoS(Quality of Service), 에이전트 소프트웨어, 소프트웨어 테스트, 네트워크 프로토콜, 네트워크 모델

학습 1

개발환경 분석하기

학습 2

기능 구현하기

학습 3

프로그램 디버깅하기

학습 4

프로그램 최적화하기

1-1. 네트워크 개발환경 구축

학습 목표

- 개발방법 기준에 따라서 네트워크 프로그래밍 구현을 위한 하드웨어(PC, Workstation, Server 등) 및 소프트웨어(Unix, Windows, IOS 등) 개발환경을 구축할 수 있다.

필요 지식

/

① 네트워크 프로그래밍의 작업 단계

네트워크 프로그래밍을 위한 전체적인 작업 단계는 개발환경 구축, 요구된 기능의 구현, 구현된 프로그램 테스트 및 디버깅, 프로그램 최적화의 단계로 구성된다.

개발환경 구축 단계에서는 개발환경을 분석하고 서버, 운영체제, 표준개발도구 등에 대한 작업이 병행되어야 한다. 기능 구현의 단계는 정의된 설계서에 따라 기능을 구현한다.

프로그램의 테스트 및 디버깅에서는 시간 및 자원 제약을 고려하여 우선순위 기반 테스트 전략을 수립하여 테스트한다. 프로그램 최적화는 소스코드 및 실행코드 측면에서 최적화한다.

② 네트워크 개발환경

1. 하드웨어 개발환경

PC, Workstation, Server 등의 설치 작업 및 환경 설정을 통해 하드웨어와 소프트웨어의 호환성 등을 사전에 점검한다.

2. 소프트웨어 개발환경

Unix, Windows 등 운영체제, Eclipse 등 개발 도구, DBMS, 형상관리 소프트웨어 등을 설치하고 환경을 설정한다.

3. 통합개발환경 (Integrated Development Environment, IDE)

통합 개발 환경은 코딩, 디버그, 컴파일, 배포 등 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어이다. 기존 소프트웨어 개발에서는 컴파일러, 텍스트 편집기, 디버거 등을 별도로 사용했으나 현재 통합개발환경에서는 프로그램들을 하나로 묶어 대화형 인터페이스를 제공한다.

③ 개발 프레임워크

개발 프레임워크는 특정 도메인, 기능군에 속한 응용 소프트웨어 개발에 공통적으로 사용되는 요소들을 일반화한 반제품의 소프트웨어이다. 주요 기능으로는 예외처리, 트랜잭션처리, 서비스관리, 데이터 소스관리, 메모리공유, 리소스매니저, 로깅 서비스, 쿼리 서비스, 사용자인증서비스 등을 수행한다.

1. 스프링 프레임워크

스프링 프레임워크는 EJB(Enterprise Java Beans) 기반의 복잡함과 무거움을 극복하고 개발 생산성 향상과 고품질의 시스템 개발을 위한 자바 플랫폼 상의 경량화된 오픈소스 웹 애플리케이션 프레임워크이다.

2. 전자정부 프레임워크

전자정부 프레임워크는 대한민국의 공공부문 정보화 사업 시 플랫폼별 표준화된 개발 프레임워크로서 응용 소프트웨어의 표준화, 품질 및 재사용성을 향상을 목표로 하며 특정 업체의 종속성 심화와 사업별 공통 컴포넌트 중복 개발을 막기 위해 개발한 프레임워크이다.

3. 닷넷 프레임워크

닷넷 프레임워크는 마이크로소프트에서 개발한 윈도 프로그램 개발 및 실행 환경이다. 네트워크 작업, 인터페이스 등의 많은 작업을 캡슐화하였고, 공통 언어 런타임(Common Language Runtime)이라는 이름의 가상 머신 위에서 작동한다.

④ 운영체제(Operating System, OS)

운영체제는 시스템 하드웨어를 관리, 응용 소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어이다. 핵심 기능으로는 프로세스 관리, 인터럽트, 메모리 관리, 파일 시스템, 장치 드라이버, 네트워킹(TCP/IP, UDP), 보안(프로세스 및 메모리 보호), 입출력 관리등이다. 주요 운영체제로는 BSD(Berkeley Software Distribution), OS X, 리눅스(우분투, 오픈 수세, 페도라, 리눅스 민트, 센트OS), Windows 등이 대중적으로 사용되고 있고 최근 보안 위협 증대로 Secure OS의 중요성도 증대되고 있다.

수행 내용 / 네트워크 개발환경 구축하기

재료 · 자료

- 시스템 개발계획서, 형상관리 계획서, 하드웨어 및 소프트웨어 시스템 구성도, IEEE(Institute of Electrical and Electronics Engineers) 1471의 소프트웨어 구조, 하드웨어 및 소프트웨어 제품 명세서, 사내 기술 및 개발 표준내역서, 개발 관련 소프트웨어, 서버장비-운영체제 호환리스트, 범정부 기술참조모델 2.3(서비스 접근 및 전달, 플랫폼 및 기반구조, 요소기술, 인터페이스 및 통합, 보안)

기기(장비 · 공구)

- 개발서버, 개발PC, 프로젝트관리 도구, 제품 라이선스 관리 시스템, 문서작성도구, 제품 설치 CD

안전 · 유의사항

- 기술 문서는 하드웨어 제품사양서, 회로도, 부품 리스트, 데이터시트 등을 포함한다
- 운영체제와 응용 프로그램 구성 계획은 소비자의 동향을 적극 반영하여 계획하도록 한다.
- 이식성을 고려하여 소프트웨어 구성 및 선정을 하도록 한다.
- 지속적으로 신기술을 파악하고, 제품에 적용할 수 있는지를 검토해야 한다.
- 개발환경은 운영환경과 분리하여 구축해야 한다.

수행 순서

① 네트워크 프로그래밍을 구현하기 위한 개발환경을 분석한다.

1. 네트워크 프로그래밍을 위한 시스템 아키텍처를 분석한다.

(1) Technical Architecture에 따른 네트워크 응용 프로그램의 시스템 구성도를 분석한다.

(2) 서버 구성에 따른 Tier를 분석한다.

웹서버, 웹 애플리케이션 서버, DB서버 등의 구성에 따른 1 Tier, 2 Tier, 3 Tier 형태를 분석한다.

2. 네트워크 프로그램을 위한 서버의 시스템 요구사항을 분석한다.

서버에 프로세서, 메모리, 하드디스크, 그래픽 카드, 설치가능 운영체제 등을 확인한다.

3. 네트워크 프로그래밍을 위한 네트워크 환경을 분석한다.

(1) 네트워크 프로그래밍 구현에 따른 트래픽을 예측하고 사전 분석한다.

(2) 네트워크 응용프로그램의 QoS(Quality of Service)를 분석한다.

QoS(Quality of Service) 구현 모델인 IntServ, DiffServ를 활용하여 트래픽을 사전 예측하고 분석한다.

(3) 네트워크 응용프로그램의 원활한 서비스 위한 전용회선 적용 여부를 분석한다.

4. 네트워크 응용 프로그램의 유형을 분석한다.

응용계층 프로그램, 트랜스포트 계층 프로그램, 디바이스 드라이버 계층 프로그램 등으로 유형을 분류하여 분석한다.

5. 네트워크 프로그래밍을 위한 개발언어를 분석한다.

대표적으로 C, C++, Java 등을 이용한 네트워크 프로그래밍 사례(소켓 등)를 분석한다.

6. 네트워크 프로그래밍 위한 보안 관련 사항을 분석한다.

(1) 소스코드 보안 점검 도구를 분석한다.

상용 보안 점검 도구인 코드시큐어 등의 사용 가능성을 분석한다.

(2) 소프트웨어 개발 보안가이드를 분석한다.

전자정부 소프트웨어 개발 운영자를 위한 소프트웨어 개발보안 가이드를 분석한다.

(3) 네트워크 응용 프로그램의 보안 구성을 분석한다.

네트워크 통신 시 암호화 적용 여부, SSL(Secure Sockets Layer) 적용가능성을 분석한다.

(4) 개발 서버의 보안 시스템을 분석한다.

개발 서버의 방화벽(웹 방화벽 포함), DB 암호화 솔루션 적용가능성을 분석한다.

수행 tip

- 하드웨어, 소프트웨어를 포괄하는 관점에서 개발환경을 분석하고 기존 네트워크 프로그램 개발환경 분석 사례를 참조한다.

② 네트워크 프로그래밍을 구현하기 위한 하드웨어(PC, Workstation, Server 등) 개발환경을 구축한다.

1. 네트워크 프로그래밍을 구현하기 위한 서버 개발환경을 구축한다.

(1) 운영체제 및 개발 소프트웨어 설치에 필요한 사양을 점검한다.

서버는 적절한 사양 확보를 통해 네트워크 프로그래밍 환경 구축을 최적화할 수 있다.

<표 1-1> Windows Server 2012 R2와 Ubuntu의 설치 사양 비교

구분	Windows Server 2012 R2	Ubuntu
프로세서	1.4GHz 64비트	300MHz (x86)
RAM	512MB	192MB
디스크	32GB	1GB

(2) 네트워크 프로그래밍을 구현하기 위한 서버의 일반적인 고려사항을 도출한다.

서버는 안정적인 서비스를 제공해야 하므로 성능, 안정성, 확장성, 표준성, 개방성, 보안성, 최신성, 이식성 등의 주요 특성을 만족하여야 한다.

<표 1-2> 서버 설치 시 고려사항

구분	설명
성능	CPU, Cache, 시스템 버스 및 I/O, TPM-C
안정성	Hot Swap 기능, 고가용성(High Availability), Fault Tolerant System
확장성	CPU, 메모리, 디스크, I/O 등의 확장성
표준성	산업계 표준을 준수하는 서버
개방성	다양한 네트워크 프로토콜 지원
보안성	보안 수준, Secure OS 고려
최신성	단종 되지 않은 최신장비(버전)의 서버

(3) 네트워크 프로그래밍을 구현하기 위한 서버를 설치하기 전 필요 사항을 확인한다.

- (가) 서버, 네트워크 등의 최적 구성 위한 토플로지 구성 형태를 확인한다.
- (나) 신규 서버 추가, 서버 내부 장치 등의 확장성 지침을 확인한다.
- (다) 서버 및 데이터베이스에 대해 지원되는 운영체제가 실행되는지 확인한다.
- (라) 하드웨어(CPU, 메모리, 하드디스크, 그래픽 등)의 권장사항을 확인한다.
- (마) 서버 및 데이터베이스의 공간 요구사항을 확인한다.
- (바) 관리 서버의 사전 설치 소프트웨어가 설치되는지 확인한다.
- (사) 보안 고려사항(Secure OS 적용가능성 등)을 점검한다.

(4) Technical Architecture에 따라 서버를 배치한다.

Tier를 고려하고 스토리지 서버, 웹 서버 등을 상호 연계하여 서버를 배치한다.

(5) 대용량 데이터 저장이 발생하는 경우 스토리지 시스템을 추가로 구축한다.

개발서버와 스토리지 시스템은 DAS(Direct Attached Storage), NAS(Network Attached Storage), SAN(Storage Area Network) 방식을 이용하여 연결한다.

2. 네트워크 프로그래밍을 구현하기 위한 PC(로컬) 개발환경을 구축한다.

(1) 개발자가 사용하기 편리하고 표준 개발도구와 호환성을 가지는 운영체제를 설치한다.

.Net Framework 4.0 버전의 경우 지원하는 지원 운영체제는 Windows 7, Windows Server 2003, Windows Server 2008, Windows Server R2, Windows R2 SP1, Windows Vista, Windows XP 등이다.

(2) 네트워크 프로그래밍을 구현하기 위한 표준 개발도구 설치에 필요한 사양을 확인한다.

표준개발도구가 개발자 PC의 프로세서, 메모리, 하드디스크 용량 등에 최적화되어야 개발, 컴파일, 빌드, 디버깅, 배포 등의 작업이 지체되지 않고 가능하게 된다.

<표 1-3> 안드로이드 기반 네트워크 응용 프로그램 개발의 최소 요구사항

구분	윈도우	리눅스	Mac OS X
OS 버전	Window XP (32bit)	Ubuntu, RedHat and Others	OS X (10.4.9+)
하드디스크 공간	25GB	25GB	25GB
시스템 메모리	3GB	2GB	4GB
프로세서	Dual Core +	Dual Core +	X86 Only
USB	USB 2.0+	USB 2.0+	USB 2.0+

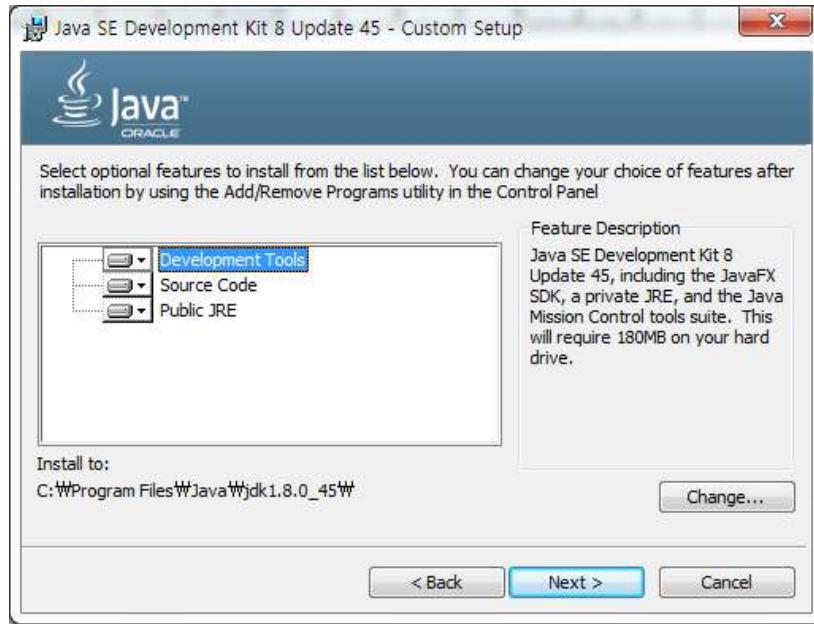
(3) 네트워크 프로그래밍을 구현하기 위한 표준 개발도구를 설치한다.

.Net Framework Control, Visual Studio, Visual C++, Commands, Eclipse, Eclipse+CDT, Kdevelop, JDK, Tomcat, 버전관리시스템(Subversion, GitHub), AmaterasUML(UML Editor), PMD(Code Inspection), JUnit(단위 테스트), 데이터베이스 접속 도구(Plsql Developer, Toad, Orange, SQLGate for Mysql) 등을 설치한다.

수행 tip

- 서버, 운영체제, 개발도구 등 필요한 목록을 사전에 작성하여 누락되지 않도록 하고, 환경 설정까지 완료해야 한다.

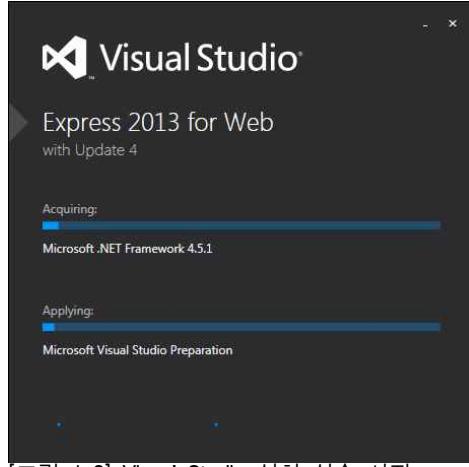
(가) JDK 설치 실습 사진



[그림 1-1] JDK 설치 실습 사진

설치경로를 확인하고 필요한 기능을 설치한다.

(나) Visual Studio 설치 실습 사진



[그림 1-2] Visual Studio 설치 실습 사진

Visual Studio, .Net Framework를 설치한다.

(4) DBMS(Database Management System) 클라이언트를 설치하고 설정한다.

(가) TNSNames.ora 파일을 수정하여 데이터베이스 사용을 원활하게 한다.

(나) TNSNames.ora 파일을 수정하는 방법은 다음과 같다.

<표 1-4> TNSNames.ora 작성 사례

```
Test = (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP)(HOST = test.com)(PORT = 1521))
        (CONNECT_DATA = (SERVICE_NAME= TEST)))
```

[3] 네트워크 프로그래밍 구현 위한 소프트웨어(Unix, Windows, IOS 등) 개발환경을 구축한다.

1. 네트워크 프로그래밍을 구현하기 위한 운영체제를 설치한다.

(1) 네트워크 프로그래밍 구현 위한 서버에 운영체제가 호환하는지 파악한다.

(가) 서버별 운영체제 호환리스트를 작성하고 점검한다.

<표 1-5> 호환리스트 예시

<http://www-03.ibm.com/systems/info/x86servers/serverproven/compat/us/nos/redchat.html>

(나) 서버에 클라이언트 운영체제(예-Windows XP)를 설치하지 않도록 주의한다.

(2) 네트워크 프로그램 구현 위한 운영체제를 서버에 설치한다.

리눅스 설치 예제

(가) 리눅스 설치 이전에 BIOS(Basic Input and Output System)를 설정한다.

(나) 리눅스(여러 가지 리눅스 배포판 중에서 선택)를 설치한다.

(다) 다른 하드디스크에 OS가 설치된 하드디스크를 복사하여 장애에 대비한다.

(라) SSH(Secure Shell) 보안 설정을 통해 사용자 인증, 접근통제를 구현한다.

(마) YaST(Yet Another Setup Tool)와 사용자 계정을 설정하여 정의된 사용자의 접근을 관리한다.

(3) 네트워크 프로그램을 구현하기 위한 운영체제 설치 후 최신 패치를 실시한다.

보안패치, 최신 운영체제 기능이 적용되도록 패치하여 서버를 최적화해야 한다.

2. 네트워크 프로그래밍을 구현하기 위한 컴파일러를 설치한다.

리눅스 운영체제에서 gcc(GNU Compiler Collection) 컴파일러를 활용할 경우 gcc(GNU Compiler Collection) 설치여부를 점검한다.

<표 1-6> gcc 명령어 사용법

```
gcc 설치 여부 확인: #gcc --version
gcc 설치 명령: #sudo apt-get install gcc
```

3. 타겟시스템의 형상관리를 위해 형상관리시스템을 설치하고 환경을 설정한다.

(1) 형상관리시스템(Subversion, GitHub 등)을 설치한다.

(2) 소스, 산출물 등 형상 아이템이 저장될 형상관리 저장소를 만든다.

(3) 형상관리 관리자, 사용자의 인증 및 권한 정보를 설정한다.

4. 네트워크 프로그래밍을 구현하기 위한 DBMS를 설치한다.

(1) Oracle 데이터베이스, SQL Server, MySql, DB2 유니버설 데이터베이스, 인포믹스 다이나믹 서버, 사이베이스 어댑티브 서버 등을 설치할 DBMS를 선택한다.

(2) 개발서버의 운영체제(32비트, 64비트)를 확인하여 설치한다.

5. 네트워크 프로그래밍 구현위한 데몬 서버를 구축한다.

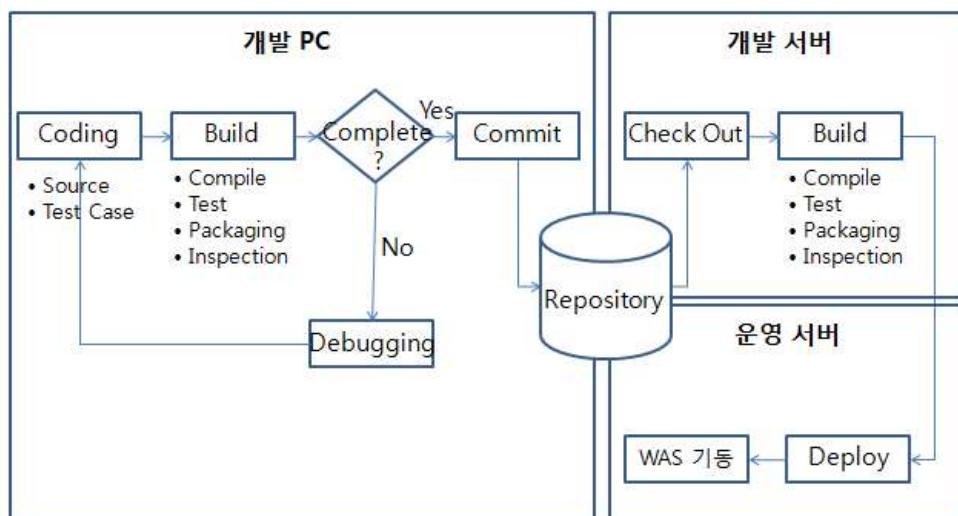
백그라운드로 항상 실행되고 있으면서 서비스를 제공하는 데몬(Httpd, Sendmail, Named 등) 프로세스를 생성한다.

수행 tip

- 소프트웨어 개발환경은 설치뿐만 아니라 환경설정 작업까지 완료해야 한다.

④ 네트워크 프로그래밍을 위한 개발환경 흐름도를 도출한다.

개발환경 흐름도는 전체적인 개발 프로세스를 가시화하고 개발PC, 개발서버, 운영서버 간 역할을 명확하게 정의할 수 있다. 개발 PC에서는 코딩(소스코드, 테스트 케이스 생성), 빌드, 디버깅 등의 작업을 수행, 개발 서버에서는 형상관리시스템과 연동하여 소스를 체크아웃하고 운영서버로 이관하기 위해 서버 빌드 작업을 수행한다.



[그림 1-3] 개발환경 흐름도

1-2. 표준개발도구 사용 및 형상 수정, 보완

학습 목표

- 네트워크 개발환경의 프로그래밍 구현을 위하여 표준 개발 도구(Commands, .Net Framework Control, Eclipse 등)를 사용할 수 있다.
- 개발 방법론에 따라서 네트워크 프로그래밍 구현을 위한 타겟시스템 형상(Configuration)을 수정하여 보완할 수 있다.

필요 지식 /

① 자바 가상 머신(Java Virtual Machine, JVM)

자바가상머신은 컴파일된 자바 바이트 코드와 실제로 프로그램의 명령어를 실행하는 마이크로 프로세서 (또는 하드웨어 플랫폼) 간에 인터페이스 역할을 하는 소프트웨어이다.

1. 클래스 로더(Class Loader)

클래스 파일을 메모리에 올려서 실행하는 부분을 담당한다. Class Loader로 인해서 '.class' 확장자의 파일은 플랫폼에 독립적으로 JVM(Java Virtual Machine)이 설치된 환경에서는 자바 프로그램을 실행할 수 있다.

2. 실행 데이터 영역(Run-time Data Area)

클래스 로더에서 준비한 Test.class에서 수행하면서 활용할 데이터를 보관한다. 메소드영역(Method Area), 힙영역(Heap Area), 스택영역(Stack Area), Native Method 스택, PC 레지스터(Register) 영역으로 구성된다.

3. 실행엔진(Execution Engine)

클래스 파일이 클래스 로더에 의해서 메모리에 정상적으로 로딩되고, 실행 엔진에 의해서 프로그램이 수행된다. JVM(Java Virtual Machine)이 읽을 수 있도록 만들어진 '.class' 파일 해석기를 의미한다.

4. 가비지 컬렉터(Garbage Collector)

멀티 프로세스 환경에서 다수의 클래스 파일이 수행되면서 힙 메모리에 남아 있는 데이터를 효율적으로 관리(삭제)하는 역할을 한다. 프로그램 환경에서 메모리 관리 기능 담당한다.

② 네트워크 형상관리

1. 형상관리는 소프트웨어 생명주기의 단계적 산출물에 대한 가시성과 추적가능성을 체계화하는 품질보증 활동이다. 형상관리를 위한 핵심 요소로는 형상항목, 베이스라인(기준선), 형상관리위원회, CMDB(Configuration Management Database) 등이 있다.
2. 형상관리 프로세스는 형상식별 -> 형상통제 -> 형상감사 -> 형상기록의 절차를 가지고 수행하게 된다. 특히, 형상통제 단계에서 변경관리를 연계하여 수행해야 한다.

③ 객체지향방법론

1. 객체지향방법론 원리

<표 1-7> 객체지향의 원리

구분	설명
캡슐화	객체의 상세한 내용을 객체 외부에 철저히 숨기고 단순히 메시지만으로 객체와의 상호 작용을 하게 하는 것
추상화	현실세계의 사실을 그대로 객체로 표현하기보다는 문제의 중요한 측면을 주목하여 상세내역을 없애 나가는 과정
다형성	하나의 인터페이스를 이용하여 서로 다른 구현 방법을 제공하는 것
상속성	슈퍼클래스가 갖는 성질을 서브클래스에 자동으로 부여하는 개념

2. 객체지향 방법론 수행절차

<표 1-8> 객체지향방법론의 절차별 수행내용

구분	절차	수행작업
객체 지향 분석	요건정의	- 사용자 요구사항에 적합한 업무요건을 정의
	객체 모델링	- 시스템 정적 구조 포착, 추상화, 분류화, 일반화
	동적 모델링	- 시간의 흐름에 따라 객체 사이의 변화를 조사 - 상태, 사건, 동작
객체 지향 설계	기능 모델링	- 입력에 대한 처리 결과에 대한 확인
	시스템 설계	- 시스템 구조를 서브 시스템으로 분해 - 성능 최적화 방안, 자원 분배 방안
객체 지향 구현	객체 설계	- 상세 내역을 모형으로 개발의 상세화 - 구체적 자료구조와 알고리즘 구현
	구현	- 객체지향언어(C++, Java)를 이용하여 구현

④ 표준개발도구

소프트웨어 개발자가 다른 프로그램과 응용 프로그램을 만들고 오류를 고치고 유지보수하는데에 사용하는 프로그램이나 응용 프로그램이다. 다양한 언어 및 플랫폼을 지원해야 하며 통합 개발 환경(IDE, Integrated Development Environment)을 지원해야 한다. 통합 개발 환경은 많은 도구의 기능을 하나의 패키지에 묶은 것을 말한다.

수행 내용 / 표준개발도구 사용 및 형상 수정, 보완하기

재료 · 자료

- 개발도구 사용 매뉴얼(Eclipse, Visual Studio 등), 전자정부 표준프레임워크 포털 (<http://www.egovframe.go.kr/>), 데이터베이스 연결 프로그램, 개발도구별 추가 기능 구현 위한 Plug In, 소프트웨어 개발 방법론, 소프트웨어 수명주기(Software Development Life Cycle), ISO(International Organization for Standardization) 12207의 기본 생명주기, 형상관리 계획서, 형상항목, 형상관리 위원회

기기(장비 · 공구)

- 개발서버, 개발PC, CMDB(Configuration Management Database), 소프트웨어 버전관리 도구, 프로젝트관리 도구, 제품 라이선스 관리 시스템, 컴파일 도구

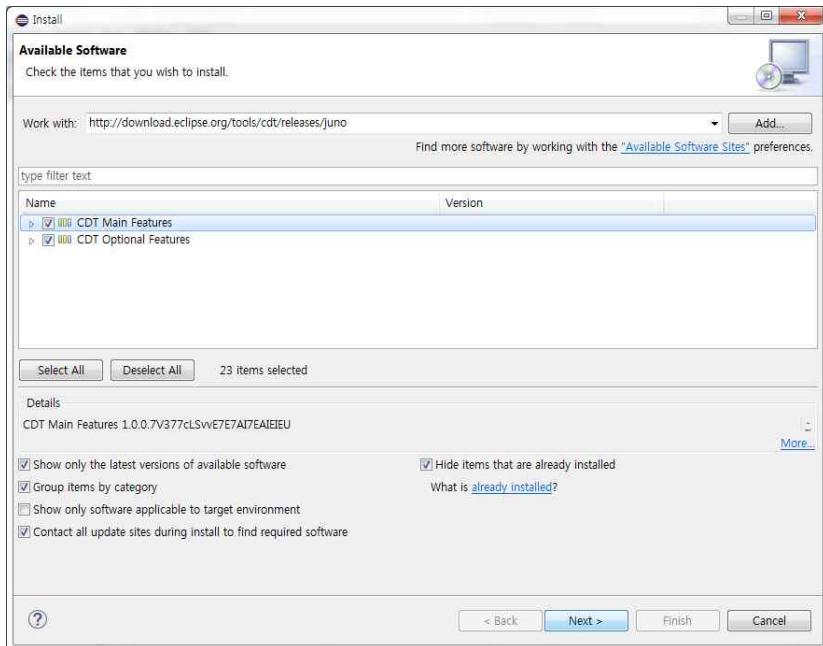
안전 · 유의사항

- 소프트웨어 개발 공정은 순서도 작성, 프로그램 구현, 개별 테스트, 종합 디버깅, 이미지 생성, 이미지 다운로드, 최종 소프트웨어 배포 등의 절차를 포함한다.
- 기술 문서를 참고하여 시스템적인 접근 방식으로 소프트웨어를 개발한다.
- 제품개발을 위한 전 과정의 이해와 분석을 바탕으로 하여 개발기준 수립에 일관성을 유지하여야 한다.
- 호스트시스템 - 타겟시스템의 구성 프레임워크를 이해해야 한다.
- 개발환경과 실행환경, 운영환경을 연동해야 한다.
- 표준개발도구는 개방된 오픈소스나 네트워크 프로그램 개발에 최적화된 개발도구를 선택하여 개발을 진행하여야 한다.

수행 순서

- ① 네트워크 프로그래밍을 구현하기 위한 표준 개발 도구를 사용한다.
 - 프로젝트의 개발 표준 및 아키텍처를 표준 개발 도구에 적용한다.
아키텍처 스타일, 소스 코딩 방법(메소드 및 변수 선언, 오류처리) 등이 표준 개발 도구에 적용 되어야 한다.

2. 네트워크 프로그래밍을 구현하기 위한 표준개발도구가 필수 요소 간에 상호 연계한다.
표준개발도구가 구현도구, 배포도구, 테스트도구, 형상관리도구와 연계되어야 한다.
3. 네트워크 프로그래밍 개발환경의 표준개발도구를 실행환경, 관리환경, 운영환경과 연계한다.
개발서버 상의 실행환경, 관리환경뿐만 아니라 향후 운영 및 유지보수를 위한 운영환경과도 연계되도록 하여 표준개발도구를 최적화하도록 해야 한다.
4. 프로젝트의 요구사항과 연계하여 요구사항 추적성을 구현한다.
요구사항분석, 시스템설계, 시스템구현(개발), 단위/통합/인수 테스트가 연계되어야 한다.
5. 표준개발도구에 필요한 기능을 구현하기 위해 Plug In을 설치한다.
Eclipse의 경우 Help -> Install New Software를 선택하여 Plug In 설치가 가능하다.



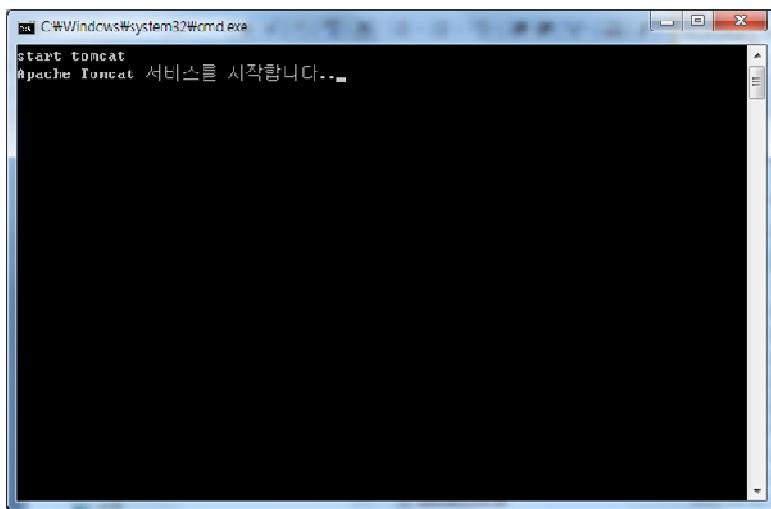
[그림 1-4] Eclipse에 CDT Plug In 설치하는 사례

6. 개발된 네트워크 프로그램의 소스코드를 빌드할 수 있다.
 - (1) 소스코드가 정상적으로 작동되는지 컴파일을 수행한다.
 - (2) 구현된 기능을 테스트한다.
 - (3) 소스코드에 결함이 존재할 경우 디버깅(Debugging)하여 결함 위치를 검색한다.
 - (4) 디버깅이 완료된 네트워크 프로그램 소스코드를 패키징(EAR, WAR 등)한다.
7. 네트워크 프로그래밍을 구현하기 위한 표준개발도구와 형상관리 서버와 연계하고 개발자 간 협업을 수행한다

공통 사용 및 참조되는 라이브러리 모듈의 경우 형상관리를 필수적으로 수행해야 한다.

8. 네트워크 프로그래밍을 구현하기 위한 표준개발도구와 개발환경 서버를 연결한다.

톰캣(Tomcat) 등을 연계하여 구현된 기능의 자체 테스트 및 개발 용이성을 고려한다.



[그림 1-5] Tomcat 실행 실습 사진

서비스 실행은 설치폴더\bin\start.bat 파일을 수행한다.

9. 네트워크 프로그래밍을 구현하기 위한 표준개발도구와 프로젝트 관리를 연계한다.

표준개발도구와 프로젝트 일정 연계하여 프로젝트의 지연이 발생하지 않도록 한다.

10. 네트워크 프로그래밍을 구현하기 위한 표준 개발 도구를 활용하여 성과를 평가한다.

구현해야 할 기능의 구현 정도를 파악하여 현재 개발 진척도를 실시간으로 점검한다.

수행 tip

- 표준개발도구는 코딩, 컴파일, 자체실행, 테스트, 디버깅, 빌드를 수행할 수 있어야 한다.

② 네트워크 프로그래밍 구현을 위한 타겟시스템 형상(Configuration)을 수정하여 보완한다.

1. 네트워크 프로그래밍 구현을 위한 타겟시스템의 형상을 식별한다.

프로그래밍 과정의 대표적인 형상물은 소스코드, 개발 산출물 등이다.

2. 형상관리 서버의 디렉토리(Directory)의 버전을 관리한다.

디렉토리(Directory) 트리 전체의 변경을 확인할 수 있도록 가시적으로 구현 가능하며 디렉토리(Directory)에도 버전정보가 추가된다.

3. 형상관리시스템의 메타데이터 버전을 관리한다.

파일과 디렉토리(Directory)는 각각 관련한 속성 키와 값의 조합을 생성해 보존하며, 속성도 파일의 내용과 동일하게 버전 관리한다.

4. 형상관리 저장소에서 전체 소스의 최종버전을 로컬의 표준 개발 도구로 받아온다.

일반적으로 프로젝트 개발 초기에 1회성으로 실시하며, 개발 중간에 전체 소스를 다시 받아오고자 하는 특수한 경우에 수행하는 절차이다

5. 형상관리 저장소로부터 동기화한 이후 타인에 의한 형상들의 변경 사항을 받아온다.

형상관리 저장소에 있는 형상들 중에서 로컬과 비교하여 변경된 항목의 최신 버전의 형상을 가져온다.

6. 로컬 환경에 체크아웃한 형상들을 수정, 추가, 삭제한다.

수정, 추가, 삭제된 형상들을 형상관리 서버에 적용할 때 형상들의 충돌 여부를 확인한다.

7. 형상들의 충돌을 방지하기 위해서 잠금 기능을 사용한다.

동일한 형상들을 서로 다른 사람이 동시에 변경하지 않도록 형상들을 Locking해야 한다.

8. 형상들의 변경사항을 다른 버전과 병합한다.

변경된 형상들을 이전의 다른 버전과 통합하여 관리해야 하는 경우 병합한다.

9. 변경된 형상들을 형상관리 저장소에 저장하고 갱신한다.

체크아웃, 체크인 기능을 활용하여 형상관리 저장소에 적용한다.

10. 형상들의 버전 및 내용이 충돌할 경우 문제를 해결한다.

다음은 형상들이 서버에서 충돌할 경우 해결 방안이다.

(1) 즉시 반영 없이, 소스 코드를 계속 수정하고 수정이 완료된 시점에 다시 시도한다.

(2) 충돌된 상황을 비교하고 로컬과 저장소의 소스의 차이를 비교하여 해결한다.

(3) 로컬의 파일을 다시 열어 수정을 계속한다.

(4) 저장소의 버전 내용을 무시하고, 로컬의 내용으로 커밋을 수행한다.

(5) 로컬의 수정 내역을 무시하고, 저장소의 버전으로 대체하여 업데이트한다.

11. 형상물의 과거의 모든 변경 이력을 조회한다.

현재 버전과 과거의 버전을 비교하여 변경내용을 확인하거나 변경내역을 관리한다.

수행 tip

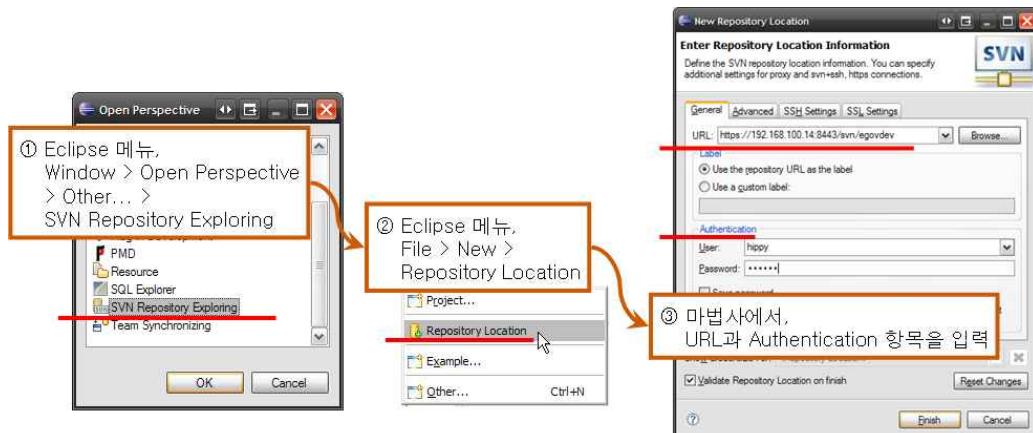
- 형상관리시 형상관리절차를 준수하고, 구조도, 프로그램 코드, 설계 사양서, 지침서 등 개발에 필요한 모든 것을 관리한다.

③ 네트워크 프로그래밍을 구현하기 위한 타겟시스템 형상관리를 수행한다.

표준개발 도구와 형상관리 시스템을 연동하여 개발자 간 협업, 소스 버전 관리, 소스 백업 등을 원활하게 하기 위해 수행 사례를 참조하여 형상관리를 직접 수행해 보아야 한다.

1. 사용자의 PC에 저장소 정보를 생성(연결)한다.

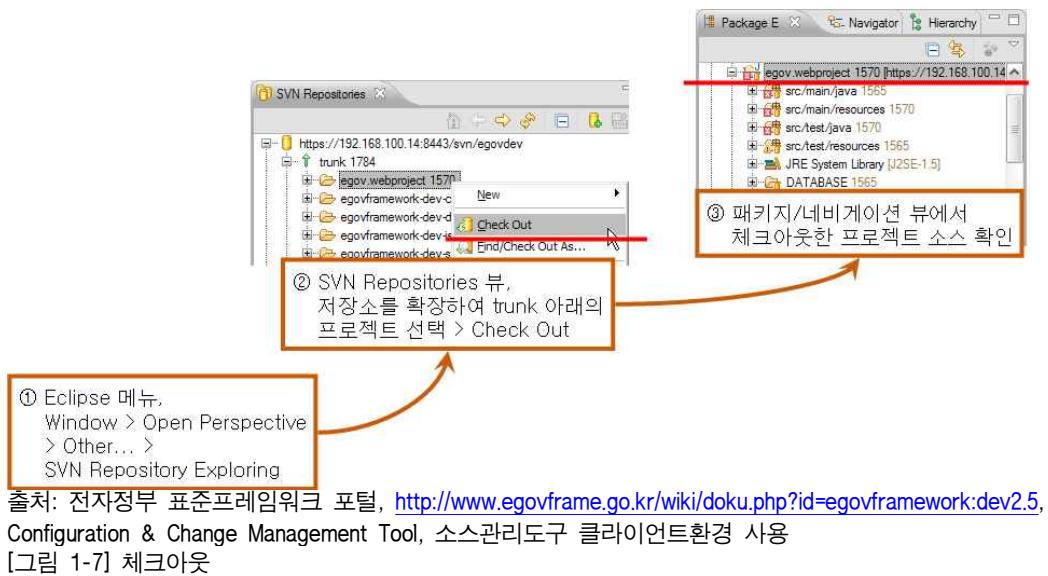
저장소 접근을 위한 URL과 인증정보를 입력하여 저장소 정보를 생성한다. Authentication 항목에 있는 User와 Password에 사용자의 SVN 저장소 접근 ID와 패스워드를 입력한다.



출처: 전자정부 표준프레임워크 포털, <http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2.5>, Configuration & Change Management Tool, 소스관리도구 클라이언트환경 사용
[그림 1-6] 사용자의 PC에 저장소 정보 생성(연결)하기

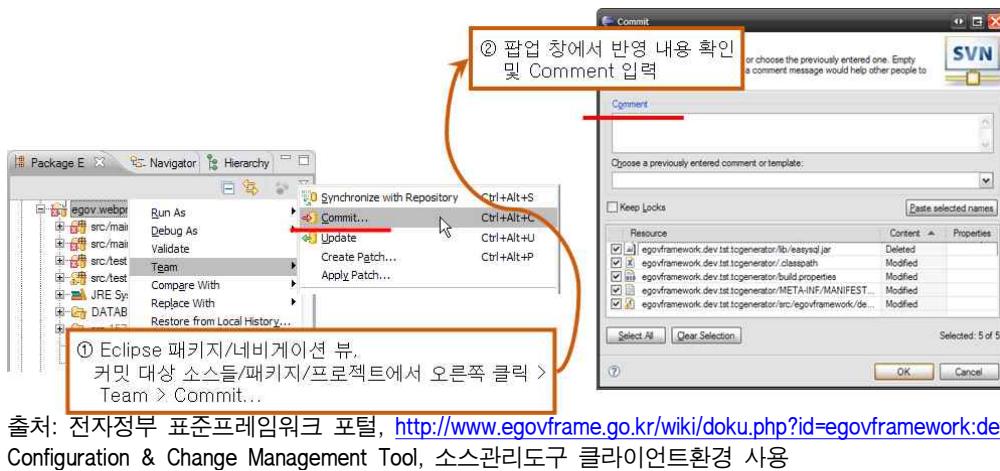
2. 저장소에서 전체 소스의 최종 버전을 받아온다.

저장소에서 전체 소스의 최종 리비전을 받아오는 것으로써 저장소 인증 정책에 따라 ID, Password 입력 여부가 결정된다. 일반적으로 프로젝트 개발 초기에 1회성으로 실시하며, 개발 중간에 전체 소스를 다시 받아오고자 하는 특수한 경우에 수행하는 절차이다.



3. 로컬에 체크아웃 한 소스를 수정한 뒤 저장소에 저장하여 갱신한다.

로컬에 체크아웃 한 소스를 수정, 파일 추가, 삭제 등을 한 뒤 저장소에 저장하여 갱신 하는 것으로 커밋을 하면 전체 리비전이 1 증가(CVS의 경우 수정한 각각 파일의 리비전이 증가)



수행 tip

- 표준개발도구와 호환성이 우수한 형상관리시스템을 선택하여 협업이 원활하도록 한다.

학습 1 교수 · 학습 방법

교수 방법

- 네트워크 프로그래밍의 개발환경 분석 사례를 사전에 파일, 출력물 등의 형태로 제공한다.
- 네트워크 프로그래밍의 개발환경을 구축하기 위해 실제 서버 등 장비를 활용하여 학습의 몰입도를 높인다.
- 네트워크 프로그래밍의 개발환경을 분석하기 위해 시스템 아키텍처에 대해 설명한다.
- 네트워크 프로그래밍의 개발환경 구축을 위해 서버의 개념 및 유형을 설명하다.
- 네트워크 프로그래밍의 개발환경 구축을 위해 운영체제의 유형과 기능에 대해 설명한다.
- 네트워크 프로그래밍을 위해 개발 프레임워크 및 개발언어에 대해 설명한다.
- 네트워크 프로그래밍 위한 표준개발도구의 설치 및 환경설정 방법에 대해 설명한다.
- 타겟시스템의 형상을 수정하고 보완하기 위해 형상관리에 대해 설명한다.
- 학습자의 성과향상, 수업참여도를 제고하기 위해 스스로 학습내용을 평가하고 피드백한다.

학습 방법

- 네트워크 프로그래밍의 개발환경을 분석하거나 구축할 때 의문점은 적극적으로 의견을 개진하여 확실하게 학습한다.
- 선수학습을 통해 학습 체크리스트를 작성하고 성과를 평가해 본다.
- 네트워크 프로그래밍의 개발환경을 분석하기 위해 시스템 아키텍처에 대해 학습한다.
- 네트워크 프로그래밍의 개발환경 구축을 위해 서버의 개념 및 유형을 학습하다.
- 네트워크 프로그래밍의 개발환경 구축을 위해 서버의 설치 요구사항에 대해 학습한다.
- 네트워크 프로그래밍의 개발환경 구축을 위해 운영체제의 유형과 기능에 대해 학습한다.
- 네트워크 프로그래밍을 위해 개발 프레임워크 및 개발언어에 대해 학습한다.
- 네트워크 프로그래밍을 위한 표준개발도구의 설치 및 환경설정 방법에 대해 학습한다.
- 타겟시스템의 형상을 수정하고 보완하기 위해 형상관리에 대해 학습한다.

학습 1 평 가

평가 준거

- 평가자는 피평가자가 수행 준거 및 평가 내용에 제시되어 있는 내용을 성공적으로 수행할 수 있는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
네트워크 개발환경 구축	- 개발방법 기준에 따라서 네트워크 프로그래밍 구현을 위한 하드웨어(PC, Workstation, Server 등) 및 소프트웨어(Uinx, Windows, IOS 등) 개발 환경을 구축할 수 있다.			
표준개발도구 사용 및 형상 수정, 보완	- 네트워크 개발환경의 프로그래밍 구현을 위하여 표준 개발 도구(Command, .Net Framework Control, Eclipse 등)를 사용할 수 있다. - 개발 방법론에 따라서 네트워크 프로그래밍 구현을 위한 타겟시스템 형상(Configuration)을 수정하여 보완할 수 있다.			

평가 방법

- 서술형시험

학습내용	평가항목	성취수준		
		상	중	하
네트워크 개발환경 구축	- 네트워크 프로그래밍 구현을 위한 하드웨어 및 소프트웨어 개발환경 구축 - 개발환경 분석 사례 작성 - 서버 운영체제 설치 및 환경 설정 방법			
표준개발도구 사용 및 형상 수정, 보완	- 표준개발도구 환경 설정 방법 - 표준개발도구 사용방법 - 형상관리 절차서 작성 - 형상관리 구성요소 나열			

• 사례연구

학습내용	평가항목	성취수준		
		상	중	하
네트워크 개발환경 구축	- 네트워크 프로그래밍 구현을 위한 하드웨어 및 소프트웨어 개발환경 구축			
표준개발도구 사용 및 형상 수정, 보완	- 네트워크 프로그래밍 구현을 위한 표준 개발 도구 사용 - 네트워크 프로그래밍 구현을 위한 타겟시스템 형상을 수정하여 보완			

• 평가자 체크리스트

학습내용	평가항목	성취수준		
		상	중	하
표준개발도구 사용 및 형상 수정, 보완	- 네트워크 프로그래밍 구현을 위한 표준 개발 도구 사용 - 네트워크 프로그래밍 구현을 위한 타겟시스템 형상을 수정하여 보완			

피드백

1. 서술형시험

- 네트워크 개발환경 구축 및 표준개발도구 사용, 형상관리 절차서 작성 등에 대한 서술형 시험문제를 제출하여 평가하고, 평가결과 일정 점수 이하인 학생들은 추가 학습한 후 그 결과를 제출하도록 한다.

2. 사례연구

- 피평가자로 하여금 네트워크 개발환경 구축 사례를 분석하고 이를 적용하는 과정을 설명하고 분석한 사례를 가지고 본인의 경우 대처할 수 있는 방안을 제시하도록 하고, 수행하지 못하는 경우에는 사례분석을 정확하게 실시하고 이를 본인의 경우에 맞추어 이에 대처할 수 있도록 지도한다.

3. 평가자 체크리스트

- 피평가자가 수행하는 표준개발도구 사용 및 형상관리의 학습내용의 체크리스트를 통하여 체크하고 수행이 원활하지 못한 경우에는 수행하지 못한 이유를 설명하고, 이를 바로 지도한다.

학습 1	개발환경 분석하기
학습 2	기능 구현하기
학습 3	프로그램 디버깅하기
학습 4	프로그램 최적화하기

2-1. 네트워크 응용프로그램 및 프로토콜 구현

학습 목표

- 개발방법 기준에 따라서 네트워크 프로그래밍 응용프로그램을 구현할 수 있다.
- 프로그래밍 방법론에 따라서 설계내용을 바탕으로 네트워크 프로토콜을 구현할 수 있다.

필요 지식 /

① 네트워크 프로토콜

컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고받는 양식과 규칙의 체계이다. 통신 프로토콜은 신호 체계, 인증, 그리고 오류 감지 및 수정 기능을 포함할 수 있다. 프로토콜은 형식, 의미론, 그리고 통신의 동기 과정 등을 정의하기는 하지만 구현되는 방법과는 독립적이다. 따라서 프로토콜은 하드웨어 또는 소프트웨어 그리고 때로는 모두를 사용하여 구현되기도 한다.

1. HTTP(HyperText Transfer Protocol)

HTTP는 WWW(World Wide Web)상에서 정보를 주고받을 수 있는 프로토콜이다. 주로 Html 문서를 주고받는 데에 쓰인다. TCP와 UDP를 사용하며, 80번 포트를 사용한다.

2. FTP(File Transfer Protocol)

FTP는 TCP/IP 프로토콜을 가지고 서버와 클라이언트 사이의 파일 전송을 하기 위한 프로토콜이다. 연결의 종류에는 명령 연결, 데이터 전송용 연결이 존재한다.

3. SMTP(Simple Mail Transfer Protocol)

SMTP는 인터넷에서 이메일을 보내기 위해 이용되는 프로토콜이다. 사용하는 TCP 포트번호는 25번이다. 상대 서버를 지시하기 위해서 DNS의 MX레코드가 사용된다.

4. TCP(Transmission Control Protocol)

TCP는 근거리 통신망이나 인트라넷, 인터넷에 연결된 컴퓨터에서 실행되는 프로그램 간에 일련의 옥텟을 안정적으로, 순서대로, 에러 없이 교환할 수 있게 한다

5. IP(Internet Protocol)

IP는 송신 호스트와 수신 호스트가 패킷 교환 네트워크에서 정보를 주고받는 데 사용하는 규약이다. OSI 모형에서 호스트의 주소지정과 패킷 분할 및 조립 기능을 담당한다

6. ARP(Address Resolution Protocol)

ARP는 네트워크상에서 IP주소를 물리적 네트워크 주소로 대응시키기 위해 사용되는 프로토콜이다.

7. RARP(Reverse Address Resolution Protocol)

RARP는 IP호스트가 자신의 물리 네트워크 주소(MAC)는 알지만 IP주소를 모르는 경우, 서버로부터 IP주소를 요청하기 위해 사용한다.

8. DHCP(Dynamic Host Configuration Protocol)

DHCP는 호스트 IP 구성 관리를 단순화하는 IP 표준이다. 동적 호스트 구성 프로토콜 표준에서는 DHCP 서버를 사용하여 IP 주소 및 관련된 기타 구성 세부 정보를 네트워크의 DHCP 사용 클라이언트에게 동적으로 할당하는 방법을 제공한다.

② 네트워크 소켓

인터넷 소켓(Internet Socket, Socket 또는 Network Socket 라고 부르기도 한다.)은 네트워크로 연결되어 있는 컴퓨터 통신의 접점에 위치한 통신 객체이다. 네트워크 통신을 위한 프로그램들은 소켓을 생성하고, 이 소켓을 통해서 서로 데이터를 교환한다.

1. 인터넷 소켓의 구성요소

인터넷 프로토콜(TCP, UDP, Raw IP), 로컬 IP 주소, 로컬 포트, 원격 IP 주소, 원격 포트

2. 인터넷 소켓의 타입 분류

UDP 프로토콜(SOCK_DGRAM)을 사용하는 경우와 TCP 프로토콜(SOCK_STREAM)을 사용하는 경우의 인터넷 소켓의 타입이 존재한다.

수행 내용 / 네트워크 응용프로그램 및 프로토콜 구현하기

재료 · 자료

- 시스템 개발계획서, 프로그램 설계서(DB설계서, 화면설계서 등), 성능 및 기능 수준 정의서, 네트워크 프로토콜 구성 내역서, OSI 7 Layer 구성 아키텍처, IETF(Internet Engineering Task Force)의 RFC(Request for Comments) 표준 문서

기기(장비 · 공구)

- 개발서버, 개발 PC, 표준개발도구, 컴파일 도구, 프로젝트 관리 도구, 소프트웨어 버전관리 도구, 제품 라이선스 관리시스템, 문서작성도구

안전 · 유의사항

- 이 능력단위는 요구분석을 통하여 도출된 기술문서를 기초로 네트워크 프로그래밍을 개발하는 업무에 적용한다.
- 소프트웨어 개발 공정은 순서도 작성, 프로그램 구현, 개별 테스트, 종합 디버깅, 이미지 생성, 이미지 다운로드, 최종 소프트웨어 배포 등의 절차를 포함한다.
- 기술 문서를 참고하여 시스템적인 접근 방식으로 소프트웨어를 개발한다.
- 기 개발된 지침 및 표준을 조사 및 분석하여 즉시 활용할 수 있는 부분과 보완이 필요한 부분을 구분하고 보완이 필요한 부분은 자료의 수집 및 분석을 통해 적극적으로 보완하거나 새로운 지침 및 표준을 작성하는 노력이 필요하다.
- 소프트웨어 개발표준을 준수하여 네트워크 응용프로그램을 개발하고, 네트워크 프로토콜을 적용한다.

수행 순서

- ① 네트워크 응용프로그램의 프로그래밍을 구현하기 위한 설계를 한다.
 - 네트워크 응용프로그램을 구현하기 위해 프로그램 설계서를 분석한다.
 - 프로그램 사양서, 기능목록 정의서, 데이터베이스 설계서 등을 분석한다.
 - 작성된 다이어그램을 분석한다.
- Class Diagram, Activity Diagram, Sequence Diagram 등 주요 다이어그램을 분석한다.

2. 네트워크 응용프로그램을 구현하기 위해 프로그램 로직을 설계한다.

조건문, 분기문, 반복문, 호출구조, 데이터 입출력 구조, 멀티태스킹 처리, 스레드 적용, 프로토콜(TCP, UDP 등)의 프로그램 로직을 설계한다.

3. 네트워크 응용프로그램을 구현하기 위해 프로그램 순서도를 작성한다.

(1) 프로그램 순서도를 작성하기 위한 기호를 도출한다.

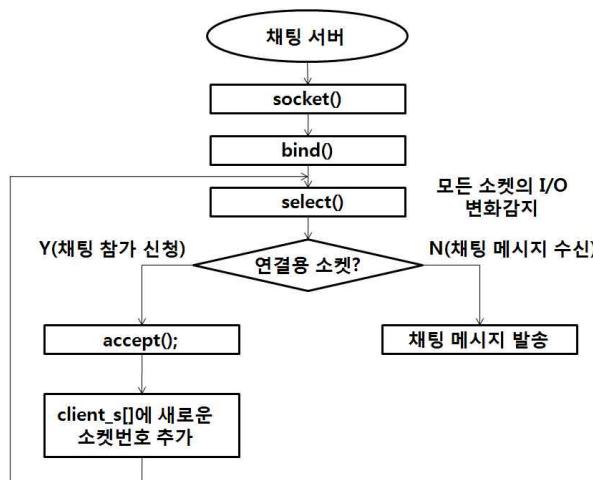
처리, 입출력, 연결자, 흐름선, 단말기, 자기디스크, 자기테이프, 통신연결, 수동입력, 비교 판단 등 기호를 통하여 프로그램의 로직 순서도를 작성할 때 활용한다.

명칭	기호	의미	명칭	기호	의미
입/출력		데이터의 입출력 기능	처리		산술연산, 데이터의 이동 편집 처리
서류/문서		서류를 매체로 하는 입출력 기능	흐름선		처리의 흐름과 실행순서 표시
단말기		처리의 시작과 종료	연결자		다음에 처리할 순서가 있는 곳 연결
비교/판단		조건의 비교 판단	화면표시		영상화면 통한 출력
자기테이프		자기테이프의 입출력	자기디스크		자기디스크의 입출력
수동입력		수작업 입력	페이지연결자		다음 페이지로 연결

[그림 2-1] 순서도 작성 위한 기호

(2) 프로그램 순서도를 작성한다.

프로그램의 순서는 일종의 알고리즘으로서 시스템의 기능을 분석하여 기중별 처리흐름을 구조화하여 작성한다. 네트워크 프로그램의 경우 네트워크 응용프로그램에서 주로 사용되는 함수를 활용하는 것도 좋은 방법이다. 추가적으로 프로그램 순서도는 Pseudo Code 생성 시 활용하여 네트워크 프로그램 구현을 원활하게 한다.



[그림 2-2] 소켓 프로그램 구현 위한 순서도 예시

(3) 네트워크 응용프로그램을 구현하기 위한 Pseudo Code를 작성한다.

Pseudo Code는 특정 프로그래밍 언어의 문법을 따라 쓰여진 것이 아니라, 일반적인

언어로 코드를 흉내 내어 알고리즘을 써 놓은 코드를 밀하는데 프로그램 순서도를 바탕으로 실제 네트워크 응용프로그램을 구현하기 위한 관점으로 구체적으로 작성한다.

<표 2-1> Pseudo Code 작성 예시

```
IF connect_socket then  
    accept()  
else  
    send_message()  
end if;
```

수행 tip

- 프로그램 로직 설계는 Pseudo Code 작성을 병행하여 개발의 일관성을 유지하도록 한다.

② 네트워크 프로그래밍을 위한 응용프로그램을 구현한다.

프로그램 구현을 위하여 프로그래밍 기본 함수를 파악하여 실전에 코딩할 수 있도록 숙지 및 관련 문서를 준비한다.

<표 2-2> 네트워크 프로그래밍 시 필요한 기본 함수

구분	설명
socket()	통신에 사용할 소켓을 개설한다.
bind()	소켓번호와 소켓주소를 결합한다.
listen()	특정 소켓을 클라이언트의 접속을 받을 수 있도록 만들어 준다.
connect()	서버와 클라이언트를 연결하는 함수이다.
accept()	TCP 연결방식에서 서버 쪽에서만 쓰이는 함수이며, 서버와 클라이언트의 연결고리를 만드는 함수이다.
send()	데이터를 해당 소켓으로 보내는 함수이다.
recv()	해당 소켓에서 데이터를 받아오는 함수이다.
close()	서비스 완료 후 연결을 종료하다.
fork()	프로세스를 생성하는 함수이다.
pthread_create()	스레드를 생성하는 함수이다.

1. 개발할 네트워크 응용프로그램의 기능을 세분화한다.

분할정복 알고리즘 등을 활용하여 구현하기 쉬운 기능 단위로 분할한다.

2. 네트워크 프로그래밍에 필요한 기본 함수 목록을 작성하고 기본 함수의 사용법을 숙지한다.

네트워크 응용프로그램에서는 socket(), bind(), listen(), connect(), accept(), send(), close(), fork(), pthread_create() 등의 함수가 주로 사용된다.

3. 설계된 로직 및 순서도를 이용하여 네트워크 응용프로그램을 작성한다.

(1) 네트워크 응용프로그램 분류에 따른 대표적인 API를 적용한다.

(가) 디바이스 드라이버 계층 응용프로그램의 API를 적용한다.

LAN에서 MAC 프레임 단위의 송수신을 다루는 API를 적용한다. (예:FTP사의 패킷 드라이버, 마이크로소프트사의 NDIS, 노벨사의 ODI)

(나) 트랜스포트 계층 응용프로그램의 API를 적용한다.

소켓(Socket) API를 적용한다. (예:Unix의 BSD소켓, 윈도우 소켓)

(2) 네트워크 응용프로그램을 작성할 때 다음 유의사항을 적용한다.

(가) 변수명, 함수명 등에 코딩 표준에 기반하여 Naming Rule을 적용한다.

(나) Clean Code 구현 관점에서 사용하지 않는 코드는 작성하지 않는다.

(다) 함수는 한 가지 기능 수행, 작은 인수 사용, 중복되지 않도록 작성한다.

(라) 함수, 클래스 등은 모듈화 원리를 적용하여 응집도 최대화, 결합도 최소화 관점에서 구현한다.

(마) 필요한 주석을 간단, 명료하게 작성하여 프로그램의 가독성을 확보한다.

(3) 네트워크 응용프로그램을 구현할 때 예외처리를 적용한다.

(가) Try-catch 문을 이용한 예외처리를 통해 로직과 오류처리를 분리한다.

(나) Error page 작성을 통해 오류 발생 시 해당 페이지로 이동한다.

(4) 소프트웨어 개발보안 가이드를 적용한다.

입력데이터 검증 및 표현, 보안기능, 시간 및 상태, 에러처리, 코드오류, 캡슐화, API 오용 등 전체 7가지 요소를 적용한다.

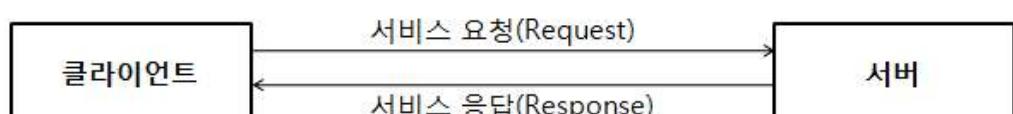
수행 tip

- 개발표준, 코딩스타일을 반드시 준수해야 하며 기 작성된 Pseudo Code를 기반으로 하여 구현한다.

[3] 설계 내용을 바탕으로 하여 네트워크 프로토콜을 구현한다.

1. 네트워크 프로토콜을 구현하기 위한 클라이언트-서버 모델을 이해한다.

클라이언트는 서비스를 이용하기 위한 요청(Request)을 서버로 보내고, 서버는 서비스를 제공하기 위한 응답(Response)을 클라이언트에게 보낸다. 일반적으로 서버를 먼저 설계 및 구현하고 클라이언트는 서버가 제공하는 서비스를 이용하도록 설계 및 구현한다.



[그림 2-3] 클라이언트-서버 모델

2. 네트워크 프로토콜을 구현하기 위한 서버 구현 기술을 이해한다.

(1) 연결형 서버와 비연결형 서버의 구현 기술을 이해한다.

(가) 연결형 서버의 구현 기술을 이해한다.

종점 간 연결 설정 및 해제, 데이터 송수신 등 세 단계의 절차를 거친다.

(나) 비연결형 서버의 구현 기술을 이해한다.

종점 간 연결 설정 및 해제 작업 없이 바로 데이터를 주고받는 방식이다.

(2) Stateful 서버와 Stateless 서버의 구현 기술을 이해한다.

(가) Stateful 서버의 구현 기술을 이해한다.

클라이언트와의 통신 상태를 계속 추적하여 서비스를 제공한다.

(나) Stateless 서버의 구현 기술을 이해한다.

상태 정보를 이용하지 않고 독립적인 요청에 의해 서비스를 제공한다.

(3) Iterative 서버와 Concurrent 서버의 구현 기술을 이해한다.

(가) Iterative 서버의 구현 기술을 이해한다.

서버가 여러 가지 클라이언트의 요청을 순서대로 처리한다.

(나) Concurrent 서버의 구현 기술을 이해한다.

서버가 여러 가지 클라이언트 요청을 동시에 처리한다.

3. 네트워크 프로토콜을 구현한다.

네트워크 프로토콜의 경우 서버 프로그램, 클라이언트 프로그램을 모두 구현해야 한다.

(1) 구현할 네트워크 프로토콜의 작동원리를 이해한다.

DHCP, DNS, FTP, HTTP, Telnet, SMTP, SMB, AFP, SSH, TLS, TCP, UDP, ARP, IP, ICMP, IGMP, MAC, PPP 등의 프로토콜의 작동 원리를 이해한다.

(2) 네트워크 응용프로그램상에 네트워크 프로토콜을 구현한다.

<표 2-3> 소켓 프로그램에서 TCP/UDP 예시

구분	내용
TCP 프로토콜	socket(PF_INET, SOCK_STREAM, 0);
UDP 프로토콜	socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);

수행 tip

- 구현되는 네트워크 프로그램의 특성에 적합한 프로토콜의 원리를 숙지하고 IETF의 RFC 표준 문서 등을 참조한다.

2-2. 데이터베이스 및 에이전트 구현

학습 목표

- 구축 계획에 따라 자원관리를 위하여 데이터베이스를 구현할 수 있다.
- 효율적인 자원관리를 위하여 에이전트(Agent)를 구현할 수 있다.

필요 지식 /

① 구축계획

제안요청서, 제안서, 프로젝트 계획서, 프로젝트 Charter, 아키텍처, 설계서 등에서 파생되는 프로젝트의 전체 일정과 범위, 비용관리 계획, 자원 관리 계획, 품질관리 계획, 의사소통 계획, 위험관리 계획 등을 통칭하여 구축 계획이라고 한다.

② 자원관리

자원관리는 다양한 의미로 해석해 볼 수 있으나 해당 학습목표의 자원관리는 기능구현으로 생성되는 데이터에 대한 자원 관리 관점과 네트워크 프로그램 구동 시 에이전트를 통한 자동화되고 효율적인 자원을 관리하는 것을 말한다.

③ 데이터베이스 정규화 및 반정규화

1. 데이터베이스 정규화

이상현상을 야기하는 함수적 종속성을 제거하기 위해 릴레이션을 작은 여러 릴레이션으로 무손실 분해(Lossless Decomposition) 하는 과정이다. 데이터 일관성 유지, 자료구조의 안전성 확보, 자료 불일치성 최소화, 데이터 중복 제거 등을 위해 실시하게 된다. 정규화는 1차 정규화, 2차 정규화, 3차 정규화, BCNF 정규화, 4차 정규화, 5차 정규화 등이 있다.

2. 데이터베이스 반정규화

정규화된 엔티티 타입, 속성, 관계를 시스템의 성능 향상, 개발과 운영을 단순화하기 위해 데이터 모델을 통합하는 프로세스이다. 반정규화의 수행절차는 반정규화 대상 조사 → 다른 방법 유도 검토 → 반정규화 실시의 과정으로 이루어진다. 주요 반정규화 기법으로는 테이블 통합/분할/추가, 중복컬럼 추가, 계산컬럼 추가, 이력컬럼 추가, 중복관계 허용 등의 기법이 있다.

④ 에이전트(Agent)

시스템, 모바일 기기, 인터넷 등에서 사용자를 대신하여 특정 목적의 작업을 수행하는 소프트웨어이며, 자신의 감각기관(Sensor)을 통해 환경(Environments)을 인지(Percept)하여 작용기(Effectors)를 통해 그 환경에 대해 반응(Action)하는 시스템을 말한다.

1. 단순 반사형 에이전트

‘조건부-결론부 관계 규칙’에 의해 자신의 지식 베이스에서 인지된 상태와 정확히 일치하는 반응만을 수행한다.

2. 외부 지식 기억형 에이전트

인지된 상태의 범위에 관한 내부 지식을 계속적으로 기억한다. 즉, 인지된 상태가 지식 베이스 조건부와 정확히 일치하는 것이 없더라도 인지된 상태의 범위로부터 유사한 결론부를 찾아낼 수 있다.

3. 목표기반 에이전트

인지에 대한 반응이 목표가 주어졌을 때 정확히 수행된다는 것을 기본 전제로 하는 에이전트로서, 탐색(Search) 문제나 계획(Planning) 문제 등에 적용된다.

4. 함수 기반 에이전트

목표 기반 에이전트가 수행할 목표들을 선택할 때 문제가 발생하기 때문에 이를 해결하기 위해 에이전트로서, 인지한 반응을 목표에 대해 얼마나 만족하는지 수치화한다.

⑤ 데이터베이스

데이터베이스란 어느 한 조직의 여러 응용 시스템이 공용할 수 있도록 통합, 저장된 운영 데이터의 집합이다. 데이터베이스는 통합 데이터, 저장 데이터, 운영 데이터, 공용 데이터여야 한다.

데이터베이스는 실시간 접근성(Real-time Accessibility), 계속적인 변화(Continuous Evolution), 동시 공용(Concurrent Sharing), 내용에 의한 참조(Contents Reference) 등의 특성을 지녀야 한다.

수행 내용 / 데이터베이스 및 에이전트 구현하기

재료 · 자료

- 시스템 개발계획서, 성능 및 기능 수준 정의서, 데이터베이스 개론, 데이터 아키텍처, 데이터 품질관리 체계, 인터넷 정보기술, 에이전트 기반 인공지능 기술, 기계학습 알고리즘, ISO(International Organization for Standardization, 국제표준화기구) 8000 데이터 품질관리 프레임워크

기기(장비 · 공구)

- 데이터베이스 서버, 데이터베이스 접속 도구, 개발서버, 개발 PC, 에이전트 테스트 서버, ER 다이어그램 작성 도구

안전 · 유의사항

- 개발 데이터베이스는 운영 데이터베이스와 분리하여 구축해야 한다.
- 테스트 데이터를 충분히 확보하여 데이터베이스 설계의 품질을 확보한다.
- 에이전트는 별도의 환경에서 테스트가 가능하도록 별도의 서버를 확보한다.
- 기술 문서를 참고하여 시스템적인 접근 방식으로 에이전트 소프트웨어를 개발한다.

수행 순서

① 데이터베이스를 구현한다.

- 데이터베이스를 구현하기 위한 데이터의 요건을 분석한다.

현행 시스템 분석, 사용자 요구사항 수집, 제안 요청서, 사업수행 계획서 등을 이용하여 데이터의 요건을 분석한다.

- 데이터베이스를 구현하기 위해 데이터를 표준화한다.

(1) 데이터의 명칭을 표준화한다.

유일성, 업무적 관점의 보편성, 의미 전달의 충분성 등의 원칙에 부합해야 한다.

(2) 데이터를 정의한다.

해당 데이터가 의미하는 범위 및 자격 요건을 규정하여 데이터를 정의한다.

(3) 데이터 형식(타입)을 표준화한다.

데이터 길이 및 소수점 자리, 도메인 정의를 통해 성격이 유사한 데이터 간의 데이터 형식(타입)을 통일하여 표준화한다.

(4) 데이터 규격을 표준화한다.

기본 값, 허용 값, 허용범위 등을 사전에 정의하여 데이터의 규격을 표준화한다.

(5) 데이터 표준화 구성요소를 정의하고 관리한다.

데이터 표준, 데이터 표준 관리 조직, 데이터 표준화 절차 등의 데이터 표준화 구성요소를 식별하고 ISO 8000 등의 표준을 연계한다.

3. 데이터베이스를 구현하기 위하여 데이터 모델링을 수행한다.

(1) 개념 데이터 모델링(Conceptual Data Modeling)을 수행한다.

(가) 데이터의 최상위 집합인 주제영역(Subject Area)을 정의한다.

(나) 데이터의 보관 단위인 핵심 데이터 집합을 정의한다.

(다) 핵심 데이터 집합 간의 관계(외래키)를 정의한다.

(라) ER(Entity-Relationship, 개체관계) 다이어그램, EER(Extended-ER) 다이어그램으로 표현한다.

(2) 논리 데이터 모델링(Logical Data Modeling)을 수행한다.

(가) 개념 데이터 모델링에서 추출된 Entity 속성을 검증 및 확정한다.

(나) Entity에 대해 유일성을 식별할 수 있는 주식별자를 확정한다.

(다) 이상현상을 야기하는 속성 간의 종속관계를 제거하기 위해 정규화를 수행한다.

(라) M:M 관계인 Entity의 관계를 1:M 관계로 해소한다.

(마) 개체 무결성(기본키) 및 참조 무결성(외래키) 규칙을 정의한다.

(바) ER(Entity-Relationship) 다이어그램을 검증한다.

(사) 도출된 데이터 모델이 사용자가 원하는 트랜잭션을 만족하는지 검증한다.

(3) 물리 데이터 모델링(Physical Data Modeling)을 수행한다.

(가) 물리요소에 대해서 조사 및 분석을 실시한다.

(나) 논리모델을 물리모델로 변환한다.

데이터가 물리적으로 컴퓨터에 어떻게 저장될 것인지를 정의하는 물리적 스키마를 생성한다.

(다) 성능향상, 개발 및 운영의 단순화를 위해 반정규화를 실시한다.

4. 데이터베이스를 구현하기 위해 데이터베이스를 설계한다.

(1) 데이터베이스의 저장 공간을 설계한다.

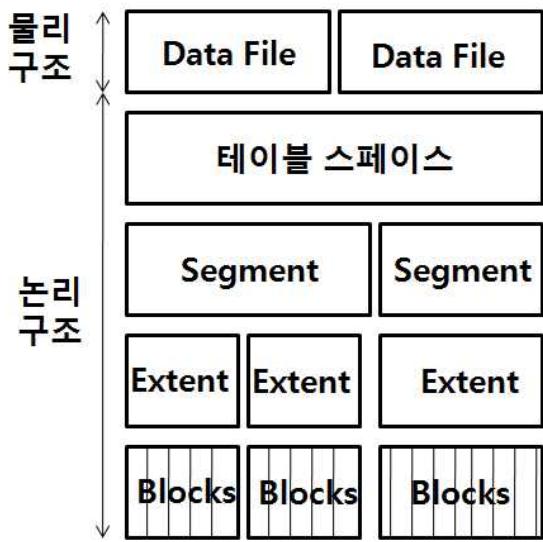
(가) 오브젝트별 용량 및 디스크 용량을 산정하기 위한 용량 설계를 수행한다.

<표 2-4> 용량 설계

구분	내용
용량 분석	데이터 증가 예상 건수, 주기, 행 길이(Row Length) 등을 고려한다.
오브젝트별 용량 산정	테이블, 인덱스에 대한 크기를 산정한다.
테이블 스페이스별 용량 산정	테이블 스페이스별 오브젝트 용량 합계를 계산한다.
디스크 용량 산정	테이블 스페이스에 따른 디스크 용량과 I/O 분산 설계를 수행한다.

(나) 테이블 스페이스(Table Space)를 설계한다.

테이블은 테이블 스페이스라는 논리적인 단위를 이용하여 관리하고, 테이블 스페이스는 물리적인 데이터 파일을 지정하여 저장된다.



[그림 2-4] 물리/논리 저장 구조 계층 (오라클 예시)

(다) 테이블(Table)을 설계한다.

테이블은 행(Row)과 칼럼(Column)으로 구성되는 가장 기본적인 데이터베이스 객체로 데이터베이스 내에서 모든 데이터는 테이블을 통해 저장된다.

1) 테이블(Table)을 설계한다.

테이블은 데이터의 저장 형태, 파티션 여부, 데이터의 유지 기간 등에 따라 다양하게 분류할 수 있다.

2) 칼럼(Column)을 설계한다.

데이터 타입(Data Type)과 길이(Length)를 정의한다.

(2) 데이터베이스의 무결성을 설계한다.

개체 무결성, 참조 무결성, 속성 무결성, 사용자 정의 무결성, 키 무결성을 구현하다.

(3) 데이터베이스의 인덱스를 설계한다.

(가) 프로그램 설계서, 프로그램 처리 조건 등을 고려하여 접근 경로를 수집한다.

(나) 분포도 조사(10 ~ 15% 정도)에 의한 후보 칼럼을 선정한다.

(다) 인덱스 후보 목록을 이용하여 접근 유형에 따라 접근 경로를 결정한다.

(라) 인덱스를 구성할 칼럼(Column)을 조합하고 순서를 결정한다.

(마) 설계된 인덱스를 적용하고 접근 경로별로 인덱스가 사용되는지 시험한다.

(4) 데이터베이스의 보안을 설계한다.

(가) 비인가자의 불법적인 자원접근을 차단하기 위해 접근통제를 구현한다.

임의적 접근통제(DAC, Discretionary Access Control), 강제적 접근통제(MAC, Mandatory Access Control), 역할기반 접근통제(RBAC, Role Based Access Control) 기법 등을 구현한다.

(나) 데이터베이스의 모든 활동을 감사 및 추적한다.

애플리케이션 및 사용자가 데이터베이스에 접근하여 수행한 모든 활동을 기록한다.

(다) 필요시 데이터베이스 암호화 솔루션을 적용한다.

Plug-In, API 방식, Secure Proxy 등의 암호화 방식을 적용한다.

수행 tip

- 데이터베이스의 무결성, 독립성, 품질 확보를 고려하여 데이터베이스를 구현한다.

② 효율적인 자원관리를 위하여 에이전트(Agent)를 구현한다.

1. 구현된 네트워크 응용프로그램을 분석한다.

(1) 구현된 네트워크 응용프로그램의 유형을 분석한다.

(가) 이메일, 웹 브라우저, 텔넷 클라이언트 프로그램, 메신저 같은 전통적인 프로그램

(나) FTP 서버에 연결하여 파일을 열거나 저장하는 프로그램

(다) 웹 사이트에 연결하거나 백신 서버에 연결하여 바이러스를 체크하고 백신을 업데이트하는 프로그램

(라) 원격지의 음악을 재생하는 음악 재생기 프로그램

(마) 실시간으로 게이머들과 커뮤니케이션하는 게임프로그램

(2) 구현된 네트워크 응용프로그램의 운용환경을 분석한다.

유무선 인터넷 환경, 모바일 환경, 원격지 네트워크 환경 등 운용환경을 분석한다.

2. 구현할 에이전트(Agent)의 핵심 기능을 도출한다.

<표 2-5> 에이전트(Agent) 핵심기능

구분	내 용
자율성(Autonomy)	에이전트는 사람이나 다른 사물의 직접적인 간섭 없이 스스로 판단하여 동작하고, 그들의 행동이나 내부 상태에 대한 어떤 종류의 제어를 가진다.
사회성(Social Ability)	에이전트는 에이전트 통신 언어를 사용하여 사람과 다른 에이전트들과 상호 작용할 수 있다.
반응성(Reactivity)	에이전트는 실세계, 그래픽사용자 인터페이스를 경유한 사용자, 다른 에이전트들의 집합, 인터넷 같은 환경을 인지하고 그 안에서 일어나는 변화에 시간상 적절히 반응한다.
능동성(Proactivity)	에이전트는 단순히 환경에 반응하여 행동하는 것이 아니라 주도권을 가지고 목표 지향적으로 행동한다.
시간 연속성 (Temporal Continuity)	에이전트는 단순히 한번 주어진 입력을 처리하여 결과를 보여 주고 종료하는 것이 아니라, 전면에서 실행하고 이면에서 잠시 휴식하는 연속적으로 수행하는 데몬 같은 프로세스이다.
목표지향성 (Goal-orientedness)	에이전트는 복잡한 고수준 작업들을 수행한다. 작업이 더 작은 세부 작업으로 나뉘고 처리 순서가 결정되어 처리되는 등의 책임을 에이전트가 진다

3. 전트(Agent)의 작동 원리를 도출한다.

(1) 성공의 정도를 평가하는 성능 평가 척도를 사용한다.

(2) 에이전트가 지금까지 인지한 사실들, 인지 서열 (Percept Sequence)을 목록화한다.

(3) 에이전트가 환경에 대해 아는 모든 지식을 추출한다.

(4) 에이전트가 수행할 수 있는 가능한 행동을 예측한다.

4. 에이전트(Agent)가 구동될 환경을 분석한다.

접근 가능과 접근 불가능한 환경, 결정적과 비결정적 환경, 에피소드적과 비에피소드적 환경, 정적과 동적 환경, 이산적과 연속적 환경 등 에이전트와 환경의 작동 관계를 분석한다.

5. 효율적인 자원관리를 위하여 에이전트(Agent)의 구성요소를 설계한다.

(1) 에이전트 엔진을 설계한다.

(가) Rule DB(Database)를 설계한다.

지식을 추론하고 연역하기 위한 추론 로직을 구현하는 Rule DB를 설계한다.

(나) 추론엔진(Inference Engine)을 설계한다.

센서로부터 들어온 정보를 기반으로 상황을 분석하기 위한 추론엔진을 설계한다.

(2) 에이전트 지식영역을 설계한다.

(가) Knowledge Map을 설계한다.

특정 분야 작업 수행을 위하여 체계화한 지식 영역 체계를 구현한다.

(나) Ontology를 설계한다.

에이전트가 필요로 하는 지식에 대한 사전을 구현한다.

(3) 에이전트 통신언어를 설계한다.

KQML(Knowledge Query Manipulation Language), ACL(Agent Communication Language, 에이전트통신언어) 등 에이전트 통신언어를 활용한다.

수행 tip

- 자원관리를 위한 에이전트(Agent)는 구현된 네트워크 응용프로그램의 분석을 통해 구현할 에이전트의 종류 및 핵심 기능을 도출한 후 에이전트(Agent)를 구현한다.

2-3. 네트워크 QoS(Quality of Service) 제공방안 구현

학습 목표

- 효과적인 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 제공방안을 구현할 수 있다.

필요 지식 /

① QoS(Quality of Service)

TCP/IP의 Best effort 한계 극복하고 인터넷 트래픽 특성에 따라 패킷을 분류하여 차별화된 서비스 품질을 제공하기 위한 지표 또는 정책을 말한다. 주요 측정 요소로 대역폭(Bandwidth), 딜레이(Delay/Latency), 지터(Jitter), 패킷 손실(Packet Loss)을 이용하여 측정하게 된다. 주요 구현 모델로서 IntServ와 DiffServ가 있다.

1. IntServ

송신자에서 수신자로 Path Message를 전송하고, 각 라우터를 지나며 Path State를 설정한 후 수신자가 송신자로 RSVP(Resource Reservation Protocol, 자원예약프로토콜) 메시지를 전송하여 라우터를 예약한다. RSVP(Resource Reservation Protocol) 시그널링 프로토콜을 기본적으로 사용하며, GS(Guaranteed Service, 보장서비스)와 CL(Controlled Load Service)의 두 가지 서비스 클래스를 제공한다. 여기서 GS는 실시간 애플리케이션에 해당하고, CL은 사전에 정의된 기준 부하하에서 기존 BE(Best Effort) 트래픽을 처리하는 것과 유사한 서비스 등급을 의미한다.

2. DiffServ

RSVP의 단점인 확장성 문제를 해결하기 위해 제안된 DiffServ 구조에서는 6비트의 DSCP(Differentiated Services Code Point)값에 의해 패킷을 서로 다른 등급으로 나누어 차별화된 서비스를 제공한다. DiffServ 모델에서는 개별적인 플로(Flow)들이 사전에 미리 정의된 서비스들 중에서 하나로 처리되고 네트워크상에서 동일한 등급의 플로(Flow)들은 같은 방식으로 처리된다. 또한 DiffServ모델에서는 SLA(Service level agreement, 서비스수준 협약서) 적용이 가능하다.

② Queue 관리기술

패킷의 입력과 출력 Queue에 대한 정책을 설정하고 제어하기 위한 기술이다.

1. 선입선출처리(FIFO)

대기열(Queue) 등에서 앞에 입력된 것부터 순서적으로 처리하여 출력하는 방법이다. 복수의 호 또는 잡(Job)이 처리 대기로 되어 있을 경우 처리의 우선순위를 붙이지 않고 먼저 도착한 순서대로 처리하는 방식이다.

2. 우선순위 큐잉(Priority Queuing)

우선순위가 높고 중요한 패킷에 높은 우선순위를 부여하여 빠르게 처리하도록 하는 방식이다.

3. 맞춤예약 큐잉(Custom Queuing)

가용 대역폭 범위 내에서 일정 비율을 특정한 프로토콜에 맞추어 사용하는 큐잉 방식을 말하며, 사전에 트래픽의 종류 및 특성을 알아야 한다.

4. WFQ (Weighted Fair Queuing)

소량의 트래픽이 대량의 트래픽에 의해 손해를 보지 않도록 패킷 플로(Flow) 별로 서로 다른 큐를 두어 트래픽을 조절하고 (공정성 측면), 특정 기준에 따라 가중치를 정하여 같은 양의 트래픽을 가진 패킷 플로 간에서도 차별을 두는 방식을 (가중치 측면) 혼합한 방식이다.

③ 혼잡제어(Congestion Control)

통신망 혼잡 발생 시 회피 및 제어하는 기술로서 네트워크 내 대기하는 패킷 수를 줄여 혼잡을 미연에 방지하거나 제어하는 기법이다. 주요 해결 방식으로 네트워크 자원을 늘리는 방식, 네트워크 부하를 줄이는 방식, 예약 기반의 혼잡제어(네트워크 쪽에다가 자원을 미리 예약하는 방식), 피드백 기반의 혼잡제어(네트워크 쪽에서 보내는 피드백 정보에 따라 속도조절) 방식 등이 있다.

1. RED(Random Early Detection)

폭주가 발생하기 이전에 이를 감지해서, 사전에 폭주의 가능성을 줄이기 위해 TCP 등에 채용된 방법이다.

2. WRED(Weighted Random Early Detection)

RED의 단점을 보완한 기법으로 서비스 차별성을 유지하면서도 혼잡제어가 가능한 방법이다. 혼잡 발생 시 탈락시킬 플로(Flow)를 특정 기준/정책에 준하는 값에 따라 우선순위를 두고 선택하도록 하는 방법이다.

재료 · 자료

- 네트워크 성능 정의서(패킷크기, 트래픽 사양 등), 스위치/라우터 등의 제품 사양서, QoS(Quality of Service) 지표, QoS(Quality of Service) 구현 모델, 기술환경 분석 자료, 스위치/라우터 등의 트래픽 최대 용량 사양

기기(장비 · 공구)

- 트래픽 수집기, 트래픽 분석기, 트래픽 대시보드, 개발서버, 개발 PC

안전 · 유의사항

- 구현된 네트워크 응용프로그램을 구동하면서 트래픽 양을 분석하고 QoS(Quality of Service)를 제공 방안을 구현해야 한다.
- 스위치 및 라우터 등의 최대 처리 트래픽을 사전에 파악하고 가용한 대역폭 설정, 패킷손실 등의 지표를 정량화해야 한다.
- 지속적으로 신기술을 항상 파악하고, 제품에 적용할 수 있는지를 검토하여 트래픽 분석 및 관리를 최적화해야 한다.

수행 순서

① 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 측정 기준을 도출한다.

1. 대역폭(Bandwidth)에 대한 기준을 도출한다.
 - (1) 서비스에 대해 가용한 사용자 데이터를 전달하는 속도 또는 달성될 수 있는 목표의 Throughput을 측정한다. (Mbps, Gbps 등으로 측정)
 - (2) Sustained Data Rate, Peak Data Rate, Min/max Data Rate 등의 파라미터로 사용된다.
2. 딜레이(Delay/Latency)에 대한 기준을 도출한다.
 - (1) End-to-end 또는 특정 네트워크 구성요소에서 고려되어야 한다.
 - (2) End-to-end 딜레이는 패킷 처리와 관련하여 Serialization(패킷 조립/분해), Propagation(전달), Switching(스위칭/라우팅) 지연의 합으로 구성된다.

3. 지터(Jitter)에 대한 기준을 도출한다.

(1) 송신 쪽에서 패킷 전송할 때 패킷들 사이의 시간에 대해 수신 쪽에서 수신되는 패킷들 사이의 시간의 왜곡 정도를 측정한다.

(2) End-to-end 또는 특정 네트워크 구성 요소에서 고려되어야 한다.

4. 패킷 손실(Packet Loss)에 대한 기준을 도출한다.

패킷 전달 과정에서 패킷 손실의 정도를 나타내며, 보내진 전체 패킷에 대한 전달되지 않은 패킷 수의 비율을 측정한다.

5. 기타 사항들을 도출한다.

(1) 패킷의 순서(Packet Order)를 결정한다.

(2) 네트워크 가용성 및 신뢰성(Network Availability/Reliability)을 지표화한다.

<표 2-6> QoS(Quality of Service) 측정기준

구분	내 용
대역폭	전송 구간이 다수 존재하고 대역폭들이 다른 값을 갖고 있을 경우 가장 낮은 대역폭으로 결정한다. 3개의 전송구간이 존재하는 경우 대역폭이 1Mbps, 256Kbps, 512Kbps이면 256Kbps가 대역폭이 된다.
딜레이	3개의 전송구간이 존재, 대역폭은 1Mbps, 256Kbps, 512Kbps이며, 패킷 사이즈가 125 Byte(1,000 bit)인 경우 구간별 연속지연은 1구간; 1,000 bit / 1,000,000 bps = 1 msec 2구간; 1,000 bit / 256,000 bps = 3.9 msec 3구간; 1,000 bit / 512,000 bps = 1.9 msec이다.
지터	발신지에서 송신지까지 구간에서 최초 라우터에서는 20ms(20,20,20)의 동일한 딜레이를 가졌다면 목표 구간에서도 20ms를 가져야한다.
패킷손실	패킷 전송 성공률을 99.5%로 하여 패킷손실을 최소화한다.

대역폭, 딜레이, 지터, 패킷손실 정량적인 지표를 측정기준으로 선정한다.

수행 tip

- QoS(Quality of Service)의 측정 기준은 정량적 지표를 통해 수치화하고 향후 트래픽 측정에 활용한다.

[2] 트래픽 분석을 위하여 네트워크 QoS(Quality of Service)의 요구사항을 도출한다.

1. 트래픽의 종류를 식별한다.

구현된 네트워크 응용프로그램에서 발생되는 트래픽의 유형을 도출하고 전체 시스템에 많은 영향을 미치는 트래픽을 선정한다.

2. 식별된 트래픽을 대상으로 대역폭, 딜레이, 지터, 패킷손실의 요구사항을 구체화한다.

<표 2-7> QoS(Quality of Service) 요구사항

트래픽 종류	QoS(Quality of Service) 요구사항			
	대역폭	딜레이	지터	패킷손실
트래픽 A	정해진 대역폭 보장	낮음, 150ms 미만	낮음	낮음
트래픽 B	모두 수용	중간	중간	낮음
트래픽 C	정해진 대역폭 보장	중간	중간	낮음
트래픽 D	20 MHz 보장	낮음, 100ms 미만	낮음	낮음

구현된 네트워크 응용프로그램의 트래픽 종류별로 QoS(Quality of Service) 측정 기준과 평하여 QoS(Quality of Service)에 대한 요구사항을 도출한다.

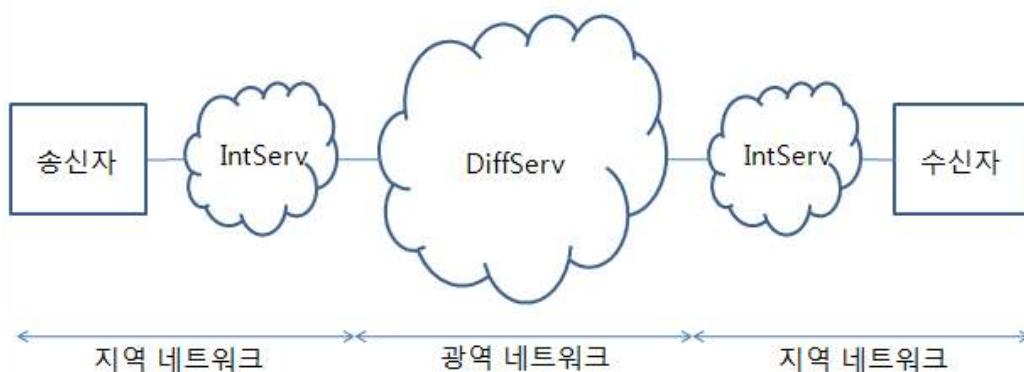
수행 tip

- QoS(Quality of Service)의 요구사항은 실제 트래픽을 가정하여 구체적으로 도출한다.

③ 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 제공방안을 구현한다.

적정한 QoS(Quality of Service)를 제공하기 위하여 QoS(Quality of Service) 구현 개념을 이해하여 지역 네트워크, 광역 네트워크에서의 QoS(Quality of Service) 구현 모델(IntServ, DiffServ)을 도출한다. QoS(Quality of Service) 구현 기술(Metering, Marking, Shaping 등)을 실제 적용하여 트래픽을 최적화하고 관리한다.

1. 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 구현 개념도를 설계한다.



[그림 2-5] IntServ와 DiffServ가 결합된 QoS(Quality of Service) 구현 개념도

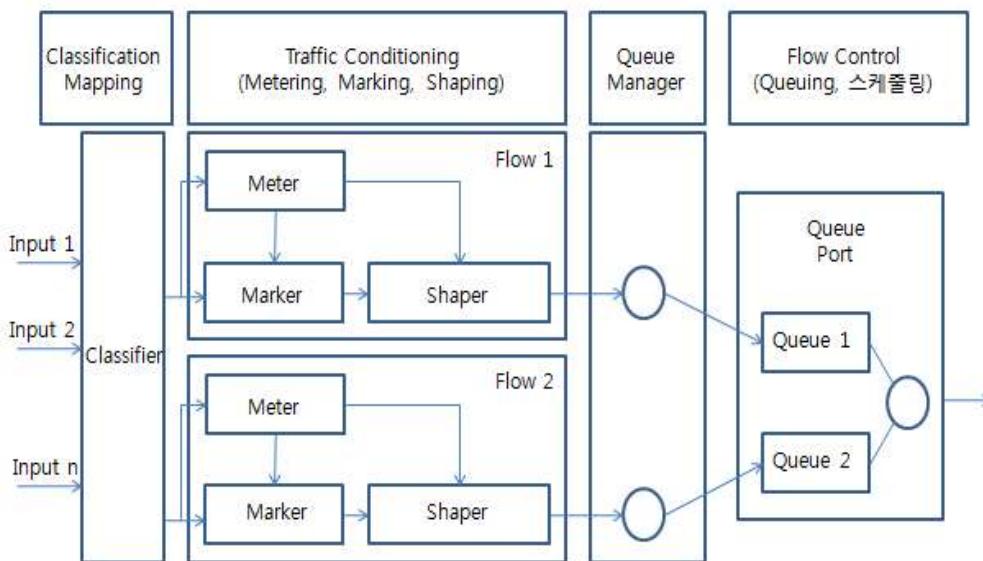
지역 네트워크와 광역 네트워크의 특성에 적합한 QoS(Quality of Service) 구현모델을 적용해야 하며, 지역 네트워크에는 IntServ를 적용하고 광역 네트워크에는 DiffServ를 적용하여 트래픽을 최적화하고 서비스의 품질을 보장해야 한다.

2. 트래픽 분석을 위하여 네트워크 QoS(Quality of Service)의 구현 기술을 적용한다.

QoS(Quality of Service) 구현 기술 아키텍처를 구축하고 아키텍처 요소마다 트래픽 관리, 큐 관리, 트래픽 흐름 제어를 연계 접목하여 트래픽을 정량화, 가시화하여 패킷 손실을 최소화하고, 트래픽의 지연을 사전에 방지한다.

(1) 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 구현 기술 아키텍처를 구축한다.

Input 트래픽을 Classifier를 통해 분류하고 Metering, Marking, Shaping 과정을 통해 Drop 여부를 결정한다. 최종적으로 큐 관리를 통해 트래픽 품질을 확보한다.



[그림 2-6] QoS(Quality of Service) 구현 기술 아키텍처

(2) 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 구현 기술을 적용한다.

QoS(Quality of Service)를 구현하기 위한 기술을 적용하기 위해 사전에 정의된 트래픽 정책과 QoS(Quality of Service) 측정 지표인 대역폭, 딜레이, 지터, 패킷 손실의 지표를 활용해야 한다.

(가) Classification 기술을 적용한다.

1) 정의된 기준에 따라 입력 패킷을 여러 등급으로 분류한다.

업무의 중요도나 패킷의 종류(음성, 문서, 그래픽 등)에 따라서 패킷을 분류한다.

2) Best Effort 등급을 포함하여 일반적으로 4개의 등급으로 분류한다.

3) TOS(Type of Service) 또는 DSCP(Differentiated Services Code Point) 필드를 이용하여 구분한다.

(나) Mapping 기술을 적용한다.

트래픽을 매핑하는 것은 일관된 QoS(Quality of Service) 특성을 유지시켜 주는 데 아주 중요한 역할을 한다.

1) 서로 다른 계층(Layer), 프로토콜, 서비스 도메인들 사이에서 QoS(Quality of

Service) 특성을 서로 연결해 주는 기능을 수행한다.

- 2) IP 네트워크에서 주로 Layer 2와 Layer 3 사이에서 수행되며 일관된 QoS(Quality of Service) 특성을 유지시켜 준다.

(다) Metering 기술을 적용한다.

패킷이 정해진 트랙을 준수하는지 여부를 체크하고 대역폭을 측정하게 하여 Metering 과정을 수행한다.

- 1) 입력되는 트래픽 플로의 속도를 통해 대역폭을 측정한다.
- 2) 입력 플로의 버스트(Burst) 정도를 측정한다.
- 3) 사전에 정의된 트래픽 프로파일과 비교한다.

(라) Marking 기술을 적용한다.

트래픽의 Marking을 통하여 트래픽의 우선순위를 부여하여 서비스 및 트래픽의 중요도에 따른 품질을 보장한다.

- 1) Metering 결과에 따라 마킹을 실시한다.
- 2) 트래픽 프로파일 만족(Committed), 일정 범위 내 초과(Excess), 일정 범위 초과(Violated)의 세 가지로 분류한다.
- 3) DiffServ의 경우 sr-TCM(Single-rate Three-color Marker) 또는 tr-TCM(Two-rate Three-color Marker)을 이용한다.

(마) Policing/Shaping 기술을 적용한다.

패킷에 Shaping을 적용하는 것은 패킷의 폐기(Drop) 여부를 결정하기 위함이다.

- 1) 네트워크의 트래픽 대역폭을 제어한다.
- 2) 초과하는 과도한 트래픽을 제한한다.
- 3) Policing의 경우 버퍼링을 사용하지 않고 초과 트래픽을 폐기하며, Shaping의 경우 버퍼링 기능을 사용하여 어느 정도 버스트 트래픽을 수용한다.
- 4) Leaky bucket, Token-bucket 등의 기법을 사용한다.

(바) Queueing 기술을 적용한다.

패킷의 입력과 출력 Queue에 대한 정책을 설정하고 제어를 수행하며 실제 서비스에는 다양한 패킷이 존재하므로 버퍼를 활용해야 한다.

- 1) 한 번에 처리할 수 없는 패킷들을 버퍼에 저장한다.
- 2) FIFO, Priority, Round-Robin, Fair Queueing, Weighted Fair Queueing(WFQ) 등의 기법을 사용한다.
- 3) WFQ의 경우 각 큐에 Weight를 할당하여 이 값에 비례하도록 서비스하며, 보장 대역폭과 연계하여 동작한다.

(사) Scheduling 기술을 적용한다.

- 1) 버퍼에 저장된 패킷들을 서비스하는 방식이다.

2) 일반적으로 Queueing 기법에 따라 스케줄링 방식이 결정된다.

(아) Congestion/Flow Control 기술을 적용한다.

입력되는 트래픽을 제어하고, 네트워크에서 혼잡이 발생하는 경우 미리 회피하거나 제어한다.

1) 네트워크 노드들이 계속해서 과부하 상태에 있는 것을 방지한다.

2) 대표적으로 TCP 트래픽을 제어하기 위한 RED(Random Early Discard) 및 WRED(Weighted RED) 기법을 이용한다.

RED 기법에서는 큐에 두 가지 임계값을 설정하여 구간별 서로 다른 패킷 drop 확률을 적용하고, WRED 기법에서는 트래픽 등급에 따라 서로 다른 RED 함수를 적용한다.

수행 tip

- QoS(Quality of Service) 제공을 통해 네트워크 성능을 포함한 QoE(Quality of Experience) 관점으로 구현해야 한다.

학습 2 교수 · 학습 방법

교수 방법

- 네트워크 프로그래밍의 설계, 코딩, 처리의 과정을 자연스럽게 습득하도록 순서도 등을 직접 작성하게 한다.
- 네트워크 응용프로그램을 구현하기 전에 설계서를 분석할 수 있도록 설명한다.
- 네트워크 응용프로그램을 구현하기 위해 순서도를 작성할 수 있도록 설명한다.
- 네트워크 프로토콜의 원리 및 응용프로그램과 프로토콜의 관계를 이해할 수 있도록 설명한다.
- 데이터베이스 설계를 통해 기본키, 외래키 등을 생성할 수 있도록 설명한다.
- 데이터베이스 설계 과정에서 정규화 및 반정규화를 수행할 수 있도록 설명한다.
- 에이전트의 구성요소를 이해하고 에이전트의 유형을 설명한다.
- QoS(Quality of Service) 제공 방안을 구현하기 위해 지표, QoS(Quality of Service) 구현 모델 등을 설명한다.
- 학습자의 성과향상, 수업참여도를 제고하기 위해 스스로 학습내용을 평가하고 피드백한다.

학습 방법

- 실습실의 개발 PC를 이용하여 직접 기능 구현, 코딩에 적극적으로 참여한다.
- 선수학습을 통해 학습 체크리스트를 작성하고 성과를 평가해 본다.
- 네트워크 응용프로그램을 구현하기 전에 설계서를 분석할 수 있도록 학습한다.
- 네트워크 응용프로그램을 구현하기 위해 순서도를 작성할 수 있도록 학습한다.
- 네트워크 프로토콜의 원리 및 응용프로그램과 프로토콜의 관계를 학습한다.
- 데이터베이스의 기본키, 외래키 등의 개념을 학습한다.
- 데이터베이스 설계 과정에서 정규화 및 반정규화를 학습한다.
- 에이전트의 구성요소 및 유형을 학습한다.
- QoS(Quality of Service)의 측정 지표 및 구현 모델을 학습한다.

학습 2 평 가

평가 준거

- 평가자는 피평가자가 수행 준거 및 평가 내용에 제시되어 있는 내용을 성공적으로 수행할 수 있는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
네트워크 응용프로그 램 및 프로토콜 구현	- 개발방법 기준에 따라서 네트워크 프로그래밍 응용프로그램을 구현할 수 있다. - 프로그래밍 방법론에 따라서 설계내용을 바탕 으로 네트워크 프로토콜을 구현할 수 있다.			
데이터베이스 및 에이 전트 구현	- 구축 계획에 따라 자원관리를 위하여 데이터 베이스를 구현할 수 있다. - 효율적인 자원관리를 위하여 에이전트(Agent) 를 구현할 수 있다.			
네트워크 QoS(Quality of Service) 제공방안 구현	- 효과적인 트래픽 분석을 위하여 네트워크 QoS(Quality of Service) 제공방안을 구현할 수 있다.			

평가 방법

- 문제해결 시나리오

학습내용	평가항목	성취수준		
		상	중	하
네트워크 응용프로그 램 및 프로토콜 구현	- 네트워크 프로그래밍 응용프로그램 구현 - 네트워크 프로토콜 구현			
데이터베이스 및 에이 전트 구현	- 자원관리를 위한 데이터베이스 구현 - 자원관리를 위한 에이전트(Agent) 구현			

• 사례 구현

학습내용	평가항목	성취수준		
		상	중	하
네트워크 응용프로그램 및 프로토콜 구현	<ul style="list-style-type: none"> - 네트워크 프로그래밍 응용프로그램 구현 - 소켓 프로그램 작성방법 - 네트워크 프로토콜 구현 - 프로토콜 유형별 특성 및 적용 사례 - 자원관리를 위한 데이터베이스 구현 - 데이터베이스 설계서 작성 			
데이터베이스 및 에이전트 구현	<ul style="list-style-type: none"> - 자원관리를 위한 데이터베이스 구현 - 에이전트(Agent) 구현 - 에이전트와 온톨로지 연계 방법 			
네트워크 QoS(Quality of Service) 제공방안 구현	<ul style="list-style-type: none"> - 트래픽 분석을 위한 네트워크 QoS(Quality of Service) 구현 모델 - 트래픽 분석을 위한 네트워크 QoS(Quality of Service) 지표 			

• 평가자질문

학습내용	평가항목	성취수준		
		상	중	하
네트워크 응용프로그램 및 프로토콜 구현	<ul style="list-style-type: none"> - 네트워크 프로그래밍 응용프로그램 구현 - 네트워크 프로토콜 구현 			
데이터베이스 및 에이전트 구현	<ul style="list-style-type: none"> - 자원관리를 위한 데이터베이스 구현 - 자원관리를 위한 에이전트(Agent) 구현 			
네트워크 QoS(Quality of Service) 제공방안 구현	- 트래픽 분석을 위한 네트워크 QoS(Quality of Service) 제공방안 구현			

피드백

1. 문제해결 시나리오
 - 네트워크 응용 프로그램 및 프로토콜을 구현할 때 발생 가능한 문제를 제시하고 문제에 대한 해결 시나리오를 작성하게 하고 평가한 후 모범 사례를 선정하여 부연 설명한다.
2. 사례 구현(기타 - 프로그램 구현)
 - 네트워크 응용 프로그램을 실제 구현하게 하고 요구된 기능대로 평가하여 작동이 되지 않는 경우 다시 프로그래밍하게 하고 문제점은 무엇이었는지 제출하게 한다.
3. 평가자질문
 - 피 평가자가 수행하는 학습에 대하여 4~5개 정도의 간단한 질문을 통해 학습의 이해 여부를 평가하고 일정 기준에 미달하는 학생은 추가 학습하게 하고 재 질문을 하도록 한다.

학습 1	개발환경 분석하기
학습 2	기능 구현하기
학습 3	프로그램 디버깅하기
학습 4	프로그램 최적화하기

3-1. 네트워크 프로그램 시험

학습 목표

- 명확한 기능구현을 위하여 결과를 시험(Test)할 수 있다.

필요 지식 /

① 테스트 설계기법

테스트 설계기법은 테스트케이스를 도출하고 수행하여 테스트 대상이 어느 수준까지 테스팅되었는지 확인(보장성 확보)하기 위해 사용된다.

1. 명세기반 테스트 설계기법

<표 3-1> 명세기반 테스트 설계기법

구분	내용
동등분할 기법	입력값, 출력값 영역을 구분하고, 등가집합의 대푯값을 통해 테스트케이스를 도출
경계값 분석 기법	동등분할 기법의 경계값을 포함하도록 테스트케이스를 도출
상태전이 테스팅	현재상황이나 이전 이력의 상태 및 변화를 상태 전이 다이어그램으로 작성하고 테스트케이스를 도출

2. 구조기반 테스트 설계기법

<표 3-2> 구조기반 테스트 설계기법

구분	내용
구문 커버리지	테스트스위트에 실행된 구문이 몇 퍼센트인지를 측정
결정 커버리지	결정포인트 내 전체 조건식이 최소한 참/거짓 한번의 값을 갖도록 측정
조건 커버리지	전체 조건식의 결과와 관계없이 각 개별 조건식이 참/거짓 한번 모두 갖도록 개별 조건식을 조합

3. 경험기반 테스트 설계기법

<표 3-3> 경험기반 테스트 설계기법

구분	내용
탐색적 테스트	테스트 엔지니어의 경험을 기반으로 테스트 설계/수행/계획/기록 및 학습을 동시에 진행하는 휴리스틱한 테스팅 기법
체크리스트	테스트하고 평가해야 할 내용과 경험을 분류하여 테스팅하는 기법
특성테스팅	ISO/IEC 9126의 품질모델에 있는 품질특성을 근간으로 하여 경험적으로 테스트케이스를 도출하는 기법

② 테스트케이스

특정한 프로그램 부분 및 경로를 실행해 보거나 특정한 요구사항에 준수하는지를 확인하기 위해 개발된 입력값, 실행 조건, 그리고 예상된 결과의 집합이다.

<표 3-4> 테스트케이스의 구성요소

구분	내용
ID	테스트케이스를 식별하기 위한 고유번호이다.
사전조건	테스트가 수행되기 위해 선결되어야 하는 조건에 대한 정의이다.
테스트 프로시저	테스트가 수행되어야 하는 순서이다. (일반적으로 7단계 이하가 적절함.)
기대결과	테스트 실행 후 의도대로 동작하였는지를 판단하는 근거이다.
합격/불합격	테스트케이스를 위한 결과에 대한 최종 판결을 의미한다.

출처: 권원일 외 3인(2008). 『개발자도 알아야 할 소프트웨어 테스팅 실무』. STA.

③ ISO/IEC/IEEE 29119

테스트 활동의 생산성 극대화를 위해 테스팅 개념, 용어, 테스팅 프로세스, 테스팅 문서화 및 기법에 관한 국제표준이다. 테스팅에 대한 평가 기준 제시, 품질 확보, 테스트 생산성 확보, 테스트 프로세스 향상을 위해 필요하다.

수행 내용 / 네트워크 프로그램 시험하기

재료 · 자료

- 테스트케이스(입력데이터, 테스트 조건, 예상결과), 테스트데이터, 테스트 하네스/스텝/드라이버, Mock Object(DB Mock, Web Mock), 요구사항 추적 매트릭스, 요구사항 명세서, 테스트 V모델, ISO/IEC/IEEE 29119 테스트 표준 문서

기기(장비 · 공구)

- 개발서버, 개발 PC, 테스트 수행 도구, 테스트 서버, 테스트 로그 수집기, 버그 트래킹 시스템

안전 · 유의사항

- 테스트케이스는 생성 후 지속적으로 개선하여 살충제 패러독스를 예방한다.
- 제3의 테스트 조직을 운영하여 테스트의 독립성을 확보한다.
- 테스트 수행 결과를 디버깅 단계와 연계하여 오류 수정과 연계해야 한다.

수행 순서

① 네트워크 응용프로그램의 테스트 전략을 수립한다.

테스트 전략은 테스트 레벨, 유형, 사람, 도구, 절차, 방법, 자원 등과 같은 테스트 필요 요소들에 대한 접근 방법을 타당한 근거를 기반으로 결정하기 위한 것으로 테스트 활동에서 가장 우선적으로 수행되어야 하는 중요한 활동이다.

1. 어떤 테스트 접근법을 적용할지 결정한다.

(1) 예방적 접근법

가능한 프로젝트 초반에, 즉 개발이 완료되기 전에 테스트를 설계한다.

(2) 사후적 접근법

소프트웨어 또는 시스템이 개발된 이후에 테스트 설계한다.

2. 리스크 기반 테스팅 기법을 활용하여 전략을 수립한다.

개발 초반부터 리스크를 식별하고 분석하여 우선순위에 기반을 두고 효과성과 효율성을 극대화하는 테스트 전략을 수립한다. 테스트의 전략을 수립할 때에는 ISO/IEC 29119 표준을 활용하여 체계적인 전략 수립이 가능하도록 한다.

<표 3-5> 테스트 수행 전략

구분	내용
상위레벨 테스트	인수테스트를 중심으로 사업적인 리스크를 중점적으로 점검한다.
하위레벨 테스트	개발테스트를 중심으로 기술적인 리스크를 중점적으로 점검한다.

수행 tip

- 테스트 접근법은 예방적/사후적 접근법을 모두 적용하여 테스트 ROI(Return On Investment, 투자대비수익률)를 확보해야 한다.

② 네트워크 응용프로그램의 테스트 완료 조건을 미리 결정한다.

하나의 테스트 레벨 또는 특정 목적의 테스팅이 언제 종료할지 정의하며, 테스트 완료 조건을 미리 결정하는 것은 효과적이고 효율적인 테스트 수행에 필수 요소이다.

1. 코드 커버리지, 기능 커버리지, 리스크 커버리지와 같은 보장성, 충분함에 대한 측정치를 만족한 경우 테스트를 종료한다.

커버리지는 작성한 테스트 코드가 대상 소스 코드에 대해 테스트하는 코드를 작성했는지 그 커버하는 정도를 백분율과 코드 라인을 통해 알려주는 것이므로 정량적인 수치를 통해 측정치를 만족하는지 검사해야 한다.

2. 추정된 결함 밀도 또는 신뢰성 측정치를 만족한 경우 테스트를 종료한다.

추정된 결함 밀도는 기존 유사 개발 사례 등을 활용하여도 무방하다.

3. 잔존 리스크(예: 수정되지 않은 결함 또는 일부 테스트 대상에 대한 불충분한 테스트 커버리지 등)가 모두 제거되면 테스트를 종료한다.

기존 테스트 케이스를 활용한 회귀테스트, 확인테스트를 활용하여 리스크가 제거 되었는지 평가한다.

4. 사전에 할당된 테스트 비용과 예산이 모두 소요된 경우 테스트를 종료한다.

5. 네트워크 응용프로그램의 시장 출시 시기에 따른 스케줄로 인해 테스트를 종료한다.

결함이 응용 프로그램 수행에 있어 사소한 것이라면 시장 출시 시기를 고려해 본다.

6. 결함의 심각도를 고려하여 수정되지 않은 결함수를 측정하고 테스트를 종료한다.

결함의 심각도가 큰 경우 해당 결함은 반드시 제거해야 하며 중요도가 낮은 결함 심각도라면 프로젝트의 비용, 일정을 고려해 본다.

7. 네트워크 응용프로그램의 시간당 결함수가 0에 근접할 때 테스트를 종료한다.

시간당 결합 수는 테스트 자동화 도구 등을 활용하여 실시간 발생하는 결함을 체크해야 한다.

8. 사전에 정의된 재 테스트 횟수를 모두 수행한 경우 테스트를 종료한다.

테스트 케이스를 개선하면서 재테스트의 횟수를 사전에 정의하여 수행해야 한다.

9. 모든 테스트 케이스를 한번 이상 수행하고 심각한 결함이 없거나 특정 수 이하일 때 테스트를 종료한다.

10. 예방된 피해와 소요되는 테스팅 비용을 비교하고 테스트를 종료한다.

테스트를 통해 결함이 제거되고 예방된 피해를 정량화하고 테스팅 비용과 비교한다.

11. 위 테스트 완료 조건의 몇 가지 조합을 만족하는 경우 테스트를 종료한다.

수행 tip

- 테스트 완료 조건의 테스트 과정의 효율성, 결과의 효과성 판단에 적합하도록 정의한다.

③ 네트워크 응용프로그램의 테스트 프로세스를 수립한다.

테스트 프로세스는 전체 테스트를 가시화하고 절차, 산출물, 방법론, 관리기법, R&R를 구체화하기 위해 반드시 필요한 절차이다.

1. 테스트의 목표와 임무를 달성하기 위해 테스트 계획과 제어(통제) 체계를 구축한다.

테스팅의 목표와 임무를 달성하기 위해 목표와 임무를 면밀히 확인하고, 테스팅의 목표 달성을 위해 필요한 활동 내역을 정의해야 한다.

(1) 테스트의 범위와 리스크를 결정하고 테스팅의 목적을 식별한다.

다른 시스템과의 인터페이스 정도, 품질특성 포함여부, 기술적/비지니스 리스크 고려, 품질요구수준, 테스트 레벨별 목적을 확인해야 한다.

(2) 테스트 레벨과 각 테스트 레벨별 시작과 종료 조건의 정의를 포함하는 테스트의 총 체적 접근법(전략)을 정의한다.

테스트 커버리지, 테스트 케이스, 결합밀도 등을 활용하여 시작과 종료 조건을 정의한다.

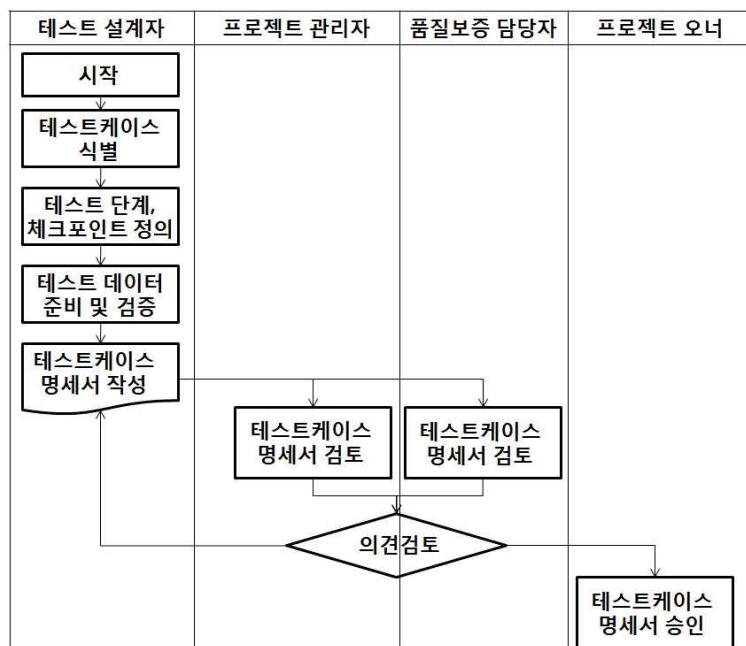
(3) 테스트 활동과 소프트웨어 획득, 공급, 개발, 운영, 유지보수 단계의 수명주기에 통합한다.

테스트 활동이 개발 프로세스와 연계되도록 설계 되어야 하며 ISO/IEC 12207 표준 등을 활용해 보아야 한다.

(4) 테스트의 내용, 주체, 방법, 평가 기준을 사전에 결정한다.

(5) 테스트 분석과 설계, 테스트 구현, 테스트 실행 및 평가 일정을 계획한다.

다음 단계에서 수행될 테스트 프로세스를 위한 전체 일정을 구체화해야 한다.



[그림 3-1] 테스트 설계활동

테스트 수행 단계와 단계별 R&R를 상호 매핑하여 테스트 전체 일정을 가시화하고 체계적으로 수행하도록 한다.

(6) 정의된 테스트 활동에 자원을 할당하고 자원관리 계획을 수립한다.

테스트 활동과 수행 일정 및 담당자의 단계별 역할을 지정하여 자원관리 계획을 수립하여 테스팅의 추적성을 연계 확보한다.

<표 3-6> 테스트 자원관리 계획

활동	시작	종료	테스트 엔지니어	도메인 전문가	테스트 컨설턴트
테스트 계획	1.1.	1.14.	V		V
리스크 분석	1.15.	1.21.	V	V	V
테스트 전략	1.22.	1.24.	V		V
테스트 설계	1.25.	2.5.	V	V	
테스트 케이스 작성	2.6.	2.20.	V	V	
테스트 환경 구축	2.21.	3.1.	V	V	
테스트 실행	3.2.	3.30.	V	V	V
테스트 마감	4.1.	4.14.	V		V

(7) 테스트 문서의 분량, 상세함 정도, 구조, 템플릿을 정의하고 현행화 방안을 수립한다.

(8) 테스트 준비와 실행, 결함 해결과 리스크 이슈를 모니터링하고 제어하기 위한 메트릭을 선정한다.

(9) 테스트 프로시저(스크립트)에 대한 상세함 정도를 결정한다.

테스트 프로시저는 전이트리 등을 활용하여 수행 깊이, 상세함을 정의해야 한다.

2. 테스트를 분석하고 설계한다.

(1) 요구사항 명세서, 아키텍처, 개발 설계 문서 등 테스트 베이시스를 리뷰한다.

(2) 테스트 대상 아이템 또는 제품, 명세, 동작과 구조의 분석을 통해 테스트 상황을 식별하고 우선순위를 결정한다.

(3) 테스트케이스를 설계하고 테스트케이스의 우선순위를 결정한다.

명세기반, 구조기반, 경험기반 등의 테스트 설계기법을 중첩하여 적용하고 테스트 케이스를 설계해야 한다.

(4) 테스트 상황과 테스트케이스에 필요한 테스트 데이터를 식별한다.

(5) 테스트 자동화를 위한 테스트 도구를 식별한다.

테스트 설계도구, 정적 분석도구, 커버리지 측정도구, 동적 분석도구, 테스트 수행도구 등 소프트웨어 수명주기 전 과정에 적용하여 테스팅의 품질을 확보할 수 있다.

3. 테스트를 구현하고 실행한다.

테스트 구현과 실행은 특별한 순서로 테스트 케이스를 결합하고 테스트 실행에 필요한 다른 정보를 포함하는 테스트 프로시저 또는 테스트 스크립트를 명세화한다.

(1) 테스트케이스를 개발하고 구현한 후 우선순위를 선정한다.

테스트케이스는 구현되는 기능에 따라 사전조건, 수행순서, 기대결과를 정의하고 핵심 기능별로 우선순위를 지정한다.

<표 3-7> 테스트케이스 작성 사례

ID	구분	사전조건	수행순서 (테스트 조건)	기대결과	추적성	중요도	비고
1	로그인	-테스트 데이터 : next, @next, 89144	-ID, PWD를 입력하고 로그인 버튼을 클릭	-정상 로그인 실패	-프로그램 사용자	중간	ID, PWD의 정상 여부 확인
2		-테스트 데이터 : @next, 89144	-3번 이상 시 시 자동으로 ID, PWD 찾기 이벤트 수행	-ID 찾기 -PWD 찾기	-프로그램 사용자	중간	PWD 등을 초기화한다.

테스트 케이스는 사전조건(테스트 데이터 포함), 수행순서(테스트 조건), 기대결과는 반드시 필요한 구성요소이며 추적성을 통해 요구사항의 구현 여부를 검증한다.

(2) 테스트 프로시저를 작성하고 우선순위를 선정하여 테스트 데이터를 생성한다.

테스트 하네스, 테스트 스텝을 준비하고 자동화 테스트 스크립트를 작성한다.

(3) 효율적인 테스트 실행을 위해 테스트스위트(테스트케이스 묶음)를 생성한다.

(4) 테스트 환경이 올바르게 구축되었는지 점검하고 미비한 부분은 재구축한다.

(5) 계획된 순서에 의거하여 수동 또는 테스트 실행 툴로 준비된 테스트 프로시저를 수행한다.

(6) 테스트 수행 결과를 기록해 두고, 테스트 중인 소프트웨어, 테스트 툴, 테스트웨어의 식별과 버전을 기록한다.

(7) 테스트 오라클을 활용하여 테스트의 예상 결과와 실제 결과를 비교한다.

테스트 오라클은 네트워크 응용프로그램의 유형 및 기능의 중요도에 따라 참 오라클, 샘플링 오라클, 휴리스틱 오라클, 일관성 검사 오라클 등을 활용해야 한다.

(8) 예상 결과와 실제 결과 간의 차이에서 오는 불일치를 인시던트 또는 결함으로 보고 한다.

(9) 불일치의 원인을 알아내기 위해서 실제 결과나 현상을 분석한다.

(10) 각각의 불일치를 조치한 결과를 확인하기 위해 테스트 활동을 반복한다.

4. 테스트의 완료조건을 평가하고 리포팅한다.

- (1) 테스트 실행 결과가 테스트 계획에 명시된 완료 조건을 만족하는지 확인한다.
- (2) 추가적인 테스트가 필요한지, 아니면 명세된 테스트 완료 조건을 변경해야 하는지에 대한 평가를 수행한다.
- (3) 이해관계자에게 배포할 테스트 요약 보고서를 작성한다.

<표 3-8> 테스트 요약보고서

프로젝트명: 날짜 : 2015년
XX개발 프로젝트 10월9일 저자: 홍길동 승인자: 이순신 비고:

요약: XX 개발 프로젝트 시 수행된 테스트에 대한 요약보고서로서 전체 테스트 일정과 비용을 준수하여 실시되었음.

특이사항: 없음.

테스트 진행 결과: 전체 1000개 테스트 케이스를 대상으로 자동화 도구를 활용하여 테스트를 진행하였고, 오류가 발생된 기능을 안정화 기간을 통해 보완함.

추가 관리 사항

- 리스크 : 테스트 일정 지연 가능성에 대한 리스크를 식별하고 대응한다.
 - 이슈 : 오류가 발생된 기능은 설계서와 비교하여 검토하고 개선한다.
 - 제약사항 : 외부 인터페이스 사항은 Mock Object를 통한 테스트가 가능함.
 - 의존성 : 특이사항 없음.
-

테스트 수행 관련 사항을 간략하여 요약하여 이해관계자에게 배포한다.

- (4) 테스트 요약 보고서에는 다음의 내용을 포함해야 한다.

발견된 결함과 미해결 결함의 추이 및 우선순위, 테스트 진척도, 리스크 및 메트릭으로 실증된 조언, 테스트 환경의 가용성, 테스트 커버리지 등이 포함되어야 한다.

5. 테스트의 마감 활동을 수행한다.

- (1) 예정된 산출물을 확인, 인시던트 리포트 종료 등을 통해 테스트의 결과를 마감한다.
- (2) 테스트웨어를 유지보수 조직에 이관하여 연속성을 유지한다.
- (3) 테스트 프로세스의 심사 또는 평가 및 개선 사항을 제안한다.

테스트 프로세스 심사 또는 평가는 TPI, TMMI 등의 표준 모델을 활용하여 전체적인 테스트 프로세스를 향상시키도록 하여야 한다.

수행 tip

- 테스트케이스는 생성 후 지속적으로 개선하여 테스트 커버리지를 최대화해야 한다.

④ 네트워크 응용프로그램의 테스트 결과를 평가한다.

테스트 평가는 테스트 매니지먼트의 일환으로, 테스트 프로세스에 관여하는 모든 작업에 대한 비용, 노력, 기간 등을 메트릭 기반으로 정량적으로 측정하는 것이다.

1. 테스트 평가 기법을 선정한다.

<표 3-9> 테스트 평가 기법

구분	내용
• 메트릭 기반 접근법	<ul style="list-style-type: none">과거 프로젝트나 유사 프로젝트의 메트릭을 근거로 또는 전형적인 가치를 근거로 테스트 노력 또는 업무량을 평가한다.
• 전문가 기반 접근법	<ul style="list-style-type: none">전문가나 업무 수행 주체에 의해 테스트를 평가한다.

메트릭 기반 접근법과 전문가 기반 접근법을 접목하여 테스트 결과를 평가하는 것이 네트워크 응용프로그램의 품질을 확보하고 테스트 조직의 성숙도를 높일 수 있다.

2. 테스팅 결과 평가 시 고려요소를 선정한다.

테스트를 시작할 수 있는 시점, 테스트 용이성, 개발자 및 관련자들의 숙련도, 잠재 결함의 양, 개발자 문제 해결 시간, 테스트 환경의 가용성과 안정성, 테스트웨어의 재사용성, 개발 프로세스의 특성, 테스트 결과, 리스크 수준 및 테스트 깊이 등을 고려한다.

<표 3-10> 테스트 결과 평가 중 결함심각도 분석 사례

구분	내용
• 0. None	<ul style="list-style-type: none">-부정적인 영향을 주지 않는 결함
• (영향을 미치지 않음.)	<ul style="list-style-type: none">-결함이 아니지만 개선 사항으로 분류될 수 있는 항목들-일부 미미한 결함을 포함하고 있으나 기능 수행의 대안이 있음.
1. Minor	<ul style="list-style-type: none">-기능이 전반적으로 제대로 작동되나 일부 불완전
• (사소한 결함)	<ul style="list-style-type: none">-해당 기능 수행은 가능하나 사용들의 혼동, 불편을 유발-화면 표준으로부터의 불일치, 시스템 미관상의 문제점
2. Major	<ul style="list-style-type: none">-주요 기능이 작동하지 않거나 정확하게 작동하지 않음.
• (주요 결함)	<ul style="list-style-type: none">-요구되는 주요 기능이나 성능 특성이 누락됨.-업무상 필수적인 예외상황 핸들링이 부적절하거나 누락됨.
3. Critical	<ul style="list-style-type: none">-애플리케이션이 작동되지 않음.
• (치명적 결함)	<ul style="list-style-type: none">-시스템 운영이 멈춤-데이터 손실 또는 훼손

테스트 결과 평가는 결함발견 현황, 결함심각도 분석, 결함 유형 분석을 실시한다.

수행 tip

- 테스트 결과 평가는 정량적으로 수치화하여 향후 개선 지표로 활용한다.

3-2. 네트워크 프로그램 디버깅 및 수정

학습 목표

- 계획된 기능구현을 위하여 결과를 디버깅(Debugging)할 수 있다.
- 개발 방법론에 따라서 기능 구현 결과를 수정(Modify)할 수 있다.

필요 지식 /

① 디버깅(Debugging)

디버깅은 오류의 존재를 확인하는 것부터 시작하여 프로그램 내에서 의심스러운 오류의 정확한 성질과 있는 위치를 찾아내는 것, 오류를 수정하는 작업으로 나누어진다. 오류의 위치를 발견할 때에는 오류의 징후(徵候)와 관계되는 정보를 충분히 분석하고 디버깅 도구는 두 번째 수단으로 사용하다.

② 회귀테스트(Regression Test)

회귀테스트는 소프트웨어의 복잡성 증가, 성능과 기능 관계, 결합 조치 확인, 정합성 테스트를 위해 승인된 변경 이후 기존 테스트케이스를 활용하여 변경 및 영향 모듈을 테스트하여 결함을 발견하는 기법이다.

1. 전체 리테스트(Retest All) 기법

전체 리테스트(Retest All) 기법은 기 축적된 테스트 케이스 및 데이터 전부 사용하는 방법이다. 테스트 커버리지가 향상되고, 테스트의 완전성이 향상되나 테스트 비용이 증가하고 사전 테스트 데이터의 준비 오버헤드가 존재한다.

2. 선택적(Selective) 기법

선택적(Selective) 기법은 변경대상을 위주로 영향 범위 결정하여 테스트를 수행하는 기법이다. 테스트 수행 범위의 최소화가 가능하고 비용 투자 대비 가장 효과적이나, 테스트의 완전성이 부족하고 선정 대상이 부정확할 때 결함발견의 어려움이 있다.

3. 우선순위(Priority) 기법

우선순위(Priority) 기법은 시스템의 핵심 기능을 위주로 우선순위를 결정하고 테스트하는 기법이다. 중요도 및 위험도에 의한 테스트 수행으로 테스트 비용이 최소화되나 우선순위가 부정확할 때 결함발견의 어려움이 존재한다. 위험도가 적은 시스템에 적용하면 효과적인 기법이다.

③ 인스펙션

인스펙션은 개발 제품 및 개발 영역에 익숙한 다수의 기술 전문가가 개발 제품의 기술적 정확성을 확인하여 소프트웨어 관련 문제들을 최대한 발견하기 위한 활동이다(IEEE 1028). 조기에 결함을 발견하고, 결함을 확인한 후 발견된 결함을 기록하고 수정하는 과정을 거치게 된다.

1. 시스템 설계 인스펙션

요구명세, 성능명세, 인터페이스 설계를 점검하여 모듈의 전반적인 설계 및 기능상의 결함을 도출한다. 설계 작업의 입력(계약 등), 기능 할당 및 흐름, 인터페이스를 검사한다.

2. 상세 설계 인스펙션

시스템 전체 설계의 목적을 반영하도록 각 모듈이 설계(상세 설계)되었는지 점검한다. 프로시저 사이의 인터페이스나 모듈의 성능 영향도 등을 검사한다.

3. 코드 인스펙션

코드가 요구 명세와 설계 명세 및 인터페이스 명세에 적합한지 검사한다. 설계의 정확한 구현여부, 제품의 국제표준 및 규격성 등을 중점 점검한다.

④ 테스트 기반 프로그램법(TDD, Test Driven Development)

테스트 기반 프로그램법은 매우 짧은 개발 사이클을 반복하는 소프트웨어 개발 프로세스이다. 우선 개발자는 바라는 향상 또는 새로운 함수를 정의하는(초기적 결함을 점검하는) 자동화된 테스트 케이스를 작성한다. 그런 후에 그 케이스를 통과하기 위한 최소한의 양의 코드를 생성한다. 그리고 마지막으로 그 새 코드를 표준에 맞도록 리팩토링한다.

1. 사용자 요구사항을 정의한다.

사용자, BA, 제품 개발자 등이 요구사항의 User Story 작성한다.

2. 테스트 케이스를 생성한다.

사용자, 테스터 등 관련자들이 테스트케이스를 상세하게 작성한다.

3. 코드를 작성한다.

개발자가 개발 단계로 테스트케이스를 통과하는 코드를 작성한다.

4. 클린 코드를 구현하기 위해 리팩토링을 수행한다.

Bad smell을 제거하여 리팩토링을 수행한 후 Simple Code에 대한 테스트를 수행한다.

수행 내용 / 네트워크 프로그램 디버깅 및 수정하기

재료 · 자료

- 오류보고서, 오류수정 계획서, 오류수정 결과 보고서, 오류수정 대상물, 오류 수정 보고서, 결함/버그/오류 리스트, 디버깅 매뉴얼, 프로그램 컴파일 방법, 소프트웨어 시험 절차서, 인스펙션 절차, 테스트 로그 파일

기기(장비 · 공구)

- 컴파일러, 디버깅 툴, 코드 검사기, 테스트 Log 수집기, 개발서버, 개발PC, 버그 트래킹 시스템, 표준개발도구

안전 · 유의사항

- 체계적인 형상관리를 위해 문서수정 여부, 소스 수정 여부를 기록하고 이력을 관리해야 한다.
- 형상관리된 형상물은 유지보수 조직과 연계하여 인수 인계해야 한다.
- 소스 수정 시 코딩 표준, 코딩 스타일을 준수하여 코드 최적화 측면에서 소스를 수정하여 프로그램의 전체적인 품질을 확보해야 한다.
- 수정된 소스는 회귀테스트를 반드시 실시하여 결함제거 여부를 확인해야 한다.
- 디버깅 수행 시 반드시 절차를 준수해야 한다.
- 정적분석도 병행하여 실시하여 결함 발생 가능성을 최소화해야 한다.

수행 순서

① 네트워크 응용프로그램의 디버깅(Debugging)을 수행하기 위한 활동을 도출한다.

1. 테스트 수행 중에 발생한 결함 및 예상 가능한 버그를 도출한다.
 - (1) 데이터 오류, 예외처리 미준수, 프로그램 일시 중단, 이상 결과 생성 등을 목록화한다.
 - (2) 요구사항 정의서, 요구사항 추적 매트릭스 등을 연계하고 형상관리를 수행한다.
2. 디버깅(Debugging)을 수행하기 위한 도구 및 기능을 숙지한다.

표준개발도구 사용 시 제공하는 디버깅 기능 및 컴파일 시 디버그 도구 활용법을 숙지한다.

3. 디버깅(Debugging)을 수행하기 전에 소스코드에 대한 정적분석을 실시한다.

Code Inspection 등을 수행하여 예측되는 결함 및 버그를 도출하고 개선한다.

4. 계획된 기능 구현을 위해 디버깅(Debugging)을 수행한다.

클래스, 메소드, 변수, Loop문, 조건문, 호출, 메모리 등을 디버깅한다.

5. 디버깅(Debugging)을 수행한 결과를 리포팅하여 이력을 관리한다.

디버깅(Debugging) 리포트와 이슈리포트 등을 연계하여 관리한다.

6. 디버깅(Debugging)을 수행한 후 문제가 발생한 코드에 대해서는 소스코드를 수정한다.

수정된 소스코드는 확인 테스트, 회귀테스트 등을 수행해야 한다.

수행 tip

- 디버깅 계획은 디버깅, 수정, 테스트 과정이 반복되도록 수립하고 단계별 수행내용을 반드시 리포팅한다.

② 테스트 수행 중에 발생한 결함 및 예상 가능한 버그를 도출한다.

1. 처리되는 이상 데이터를 도출한다.

- (1) 선언된 변수를 초기화하는 데 실패하여 데이터의 이상이 발생한 경우를 식별한다.
- (2) 정수 형식에 정수 형식의 범위를 초과(실수 등)하는 형식이 대입된 경우를 식별한다.
- (3) 배열의 크기를 초과하여 배열의 색인 표현에서 에러가 발생한 경우를 식별한다.
- (4) 루프 조건에서 데이터가 변조되는 에러가 발생한 경우를 식별한다.
- (5) 동적으로 할당된 배열의 크기에서 에러가 발생한 경우를 식별한다.
- (6) 클래스 Copy 컨структор, 할당 연산자, 디스트럭터를 구현하는 데 실패한 경우를 식별한다.

2. 예외 처리가 프로그램상에 구현되지 않은 경우를 도출한다.

- (1) 유효하지 않은 포인터 또는 레퍼런스를 참조하여 예외가 발생한 경우를 식별한다.
- (2) Catch 핸들러(Handler)를 누락하여 Exception에 대한 Throw를 누락한 경우를 식별한다.

3. 구현된 프로그램이 수행 중에 중단되거나 오류가 발생하는 경우를 도출한다.

- (1) 변수를 초기화하는데 실패하여 프로그램이 중단된 경우를 식별한다.
- (2) 루프문에서 완료 조건이 누락되어 무한 루프를 수행하는 프로그램을 식별한다.
- (3) 유효하지 않은 포인터로 인해 프로그램이 수행 중에 중단되는 경우를 식별한다.

- (4) 동일한 Free Store 메모리를 두 번 해제시킨 경우를 식별한다.
- (5) 클래스 디스트럭터의 구현에 실패했거나, 그 디스트럭터에서 에러가 발생한 경우를 식별한다.

4. 스트림 입력 데이터가 잘못된 경우를 도출한다.

추출 연산자(Extraction Operator)와 getline() 함수를 사용하여 스트림 데이터를 입력받아서 버그가 발생된다.

5. 프로그램을 수행한 이후 이상한 결과가 생성된 경우를 도출한다.

- (1) 입력 에러, == 대신에 -=를 입력했거나, j 대신에 i를 입력한 경우 등을 식별한다.
- (2) 변수를 초기화하는 데 실패하여 잘못된 결과가 생성된 경우를 식별한다.
- (3) 유효하지 않은 포인터로 인해 잘못된 결과가 생성된 경우를 식별한다.
- (4) Switch에서 Break를 누락하여 잘못된 결과가 생성된 경우를 식별한다.

수행 tip

- 예상 가능한 버그 리스트를 프로젝트 관리 도구와 연계하여 관리한다.

③ 네트워크 응용프로그램의 계획된 기능구현을 위하여 결과를 디버깅(Debugging)한다.

1. 디버깅(Debugging)을 수행하기 위한 중단점(Breakpoint)을 설정한다.

- (1) 디버깅(Debugging)을 수행하기 위해 중단점(Breakpoint)을 추가하거나 삭제한다.
- (2) 디버깅(Debugging)을 수행하기 위해 중단점(Breakpoint)을 활성화 또는 비활성화한다.
Disable Breakpoint를 활용하여 중단점(Breakpoint)이 비활성화되면 특정 중단점(Breakpoint)에서 실행을 멈추지 않고 수행하거나 중단점(Breakpoint)을 삭제하지 않아도 된다.

2. 스텝 단위 디버깅(Debugging)을 수행하다.

- (1) 호출하는 메소드 내부 동작을 디버깅하기 위해 프로그램을 한 스텝씩 진행하되, 메소드 호출부라면 실행 코드를 메소드 안으로 옮기고 디버깅을 수행한다.
- (2) 호출하는 메소드 내부 동작에는 관심이 없고, 현재 코드 블록에 디버깅을 집중할 때에는 메소드 호출부라도 메소드 안으로 들어가지 않고 현재 코드에서 한 스텝씩 진행한다.
- (3) 현재의 메소드에서 리턴한 후, 메소드 호출부에서 다시 멈춘다.
- (4) 멈춰있던 쓰레드를 다시 진행한다. 다음 중단점(Breakpoint)까지 진행한다.

(5) 선택한 스택 프레임의 첫 행으로 실행 포인트를 옮긴다. 특정 메소드를 수행 중에 그 메소드의 처음부터 다시 디버깅을 수행할 때 사용한다.

(6) 스텝단위 디버깅이 완료되면 프로그램을 종료한다.

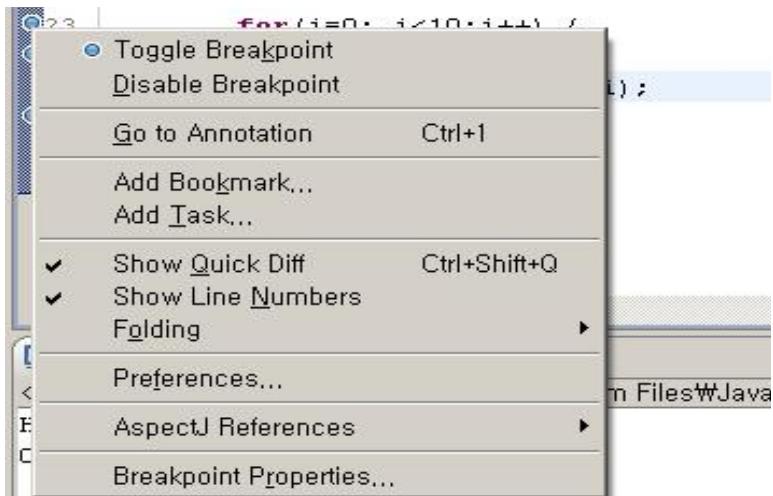
3. 디버깅 도구를 이용한 디버깅 수행 방법을 이해한다.

(1) 중단점(Breakpoint)을 설정한다.

디버깅 중에 프로그램의 의심되는 부분을 집중적으로 분석하기 위해 중단점(Breakpoint)을 설정해 디버깅 포인트를 지정한 부분을 하이라이트한다.

(가) 에디터의 왼쪽에 있는 Marker bar를 더블클릭하거나 팝업 메뉴를 통해 Toggle Breakpoint를 선택하여 추가할 수 있다.

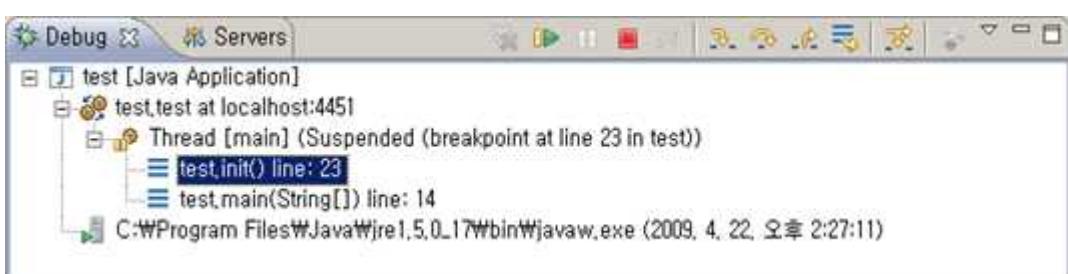
(나) Disable Breakpoint를 선택하면 중단점(Breakpoint)을 비활성화시킨다.



출처: 전자정부 표준프레임워크 포털 (<http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2.5>), Local Debug
[그림 3-2] 중단점(Breakpoint) 설정

(2) 스텝 단위 디버깅(Debugging)을 수행한다.

클래스 내부의 메소드별로 하이라이트를 표시하고 단계적 디버깅을 한다.



출처: 전자정부 표준프레임워크 포털 (<http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2.5>), Local Debug
[그림 3-3] 스텝 단위 디버깅

(3) 스텝 필터링을 수행한다.



출처: 전자정부 표준프레임워크 포털 (<http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2.5>),
Local Debug
[그림 3-4] 스텝 필터링

스텝 필터링 기능을 통해 필터링 대상 프로그램의 경우 Step Over와 같이 동작하고, 필터링 대상이 아니면 Step Into와 같이 동작한다. Debug View의 Use Step Filters 버튼을 눌러 활성화시킬 수 있다

수행 tip

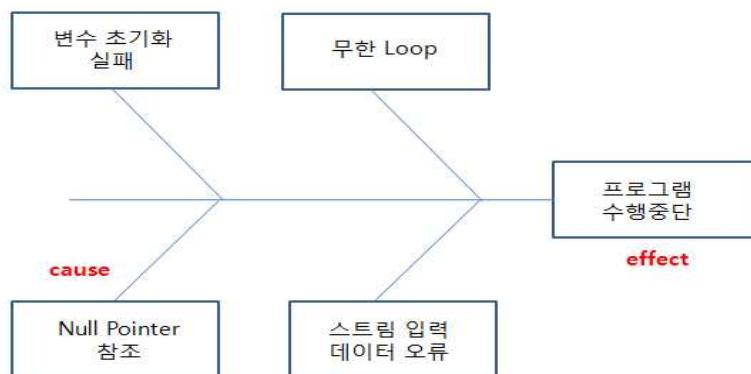
- 디버깅은 전용 디버거를 활용하거나 표준개발도구에서 제공하는 기능을 이용한다.

④ 네트워크 응용프로그램의 기능 구현 결과를 수정한다.

1. 테스트 및 디버깅 수행 과정에서 발생한 결함 및 오류에 대한 Causal Effect를 분석한다.

물고기뼈 도표(Fishbone Diagram)를 활용하여 원인-결과의 근본 요인을 분석한다.

Cause(변수 초기화 실패) -> Effect(프로그램 수행중단) 과정을 순차적으로 거친다.



[그림 3-5] 프로그램 결함에 대한 물고기뼈 도표

2. 기능 구현 결과의 오류 수정 계획을 수립한다.

- (1) 프로젝트 버퍼를 활용하여 수정 일정을 수립한다.

프로젝트 버페, 자원버페, 용량버페 등 프로젝트 관리 시 CCPM(Critical Chain Project Management)을 활용한다.

- (2) 소스코드의 수정에 대한 코드 및 문서의 수정 범위를 정의한다.

- (3) 오류보고서를 작성한다.

<표 3-11> 오류 보고서

• 오류번호	•	•	•
• 테스트 항목	•	• 오류 형태	•
• 추적성	•	• 발견 시점	•
• (발생 시점)	•	• 재현여부	•
• 처리상태	•	•	•
• 심각도	• 사업적	• 우선순위	•
	• 기술적	•	•
• 보고일	•	• 발생버전/빌드	•
• 수정일	•	• 수정버전/빌드	•
• 완료일	•	• 확인자	•
• 작성자	•	• 수정담당자	•

오류번호, 오류형태 등 발생한 오류를 상세하게 기술한다.

(4) 오류 수정 계획서를 작성한다.

오류 발생한 테스트 아이템을 기준으로 오류 수정 계획서를 작성한다.

<표 3-12> 오류 수정 계획서(ISO/IEC/IEEE 29119-3 기반)

구분	내용
1. 개요	1.1 범위 1.2 참고자료 1.3 용어
2. 테스트 컨텍스트	2.1 프로젝트 단계/유형 2.2 오류 발생 테스트 아이템 2.3 오류 발생 테스트 범위 2.4 가정과 제약
3. 테스팅 커뮤니케이션	
4. 리스크	4.1 제품 리스크 4.2 프로젝트 리스크
5. 테스트 전략	5.1 테스트 서브 프로세스 5.2 재테스팅과 Regression 테스팅 5.3 테스트 설계 기법 5.4 테스트 종료 기준 5.5 테스트 데이터 요구사항 5.6 테스트 환경 요구사항 5.7 테스트 도구 요구사항 5.8 테스트 산출물
6. 테스트 업무산정	
7. 일정	7.1 중단과 재개기준
8. 리스크 추적	

(5) 소스코드 수정 이후에 재테스트 수행 방안을 수립한다.

수정된 소스코드를 대상으로 기존 테스트케이스를 개선하여 회귀테스트 및 확인테스트를 실시하고 발생된 오류가 제거 되었는지 반드시 확인한다.

3. 좋은 품질의 네트워크 응용프로그램 구현위해 결과를 수정한다.

(1) 수정 및 변경되는 내용은 반드시 문서화 통해 현행화를 구현한다.

(2) 코딩표준을 준수하고 코딩스타일을 접목하여 소스코드를 수정한다.

안전한 소프트웨어를 개발하기 위하여 소스코드 보안가이드를 참조하여 보안 부분도 누락이 없어야 한다.

4. 기능 구현을 위해 결과를 수정한 이후 검토하여 최적화를 진행한다.

(1) 변경된 문서 및 소스코드는 형상관리를 통해 이력을 관리한다.

(2) 확인 테스트, 회귀테스트를 실시한다.

(가) 살충제 패러독스 예방을 위하여 테스트 케이스를 변경, 개선한다.

(나) 전체 재테스트 실시, 선택된 부분만 테스트, 우선순위 기반 테스트 등 구현된 기능의 중요도를 파악하여 회귀 테스트를 실시한다.

(3) 오류수정 결과보고서를 작성하여 네트워크 응용프로그램에 이상이 없음을 확인한다.

오류수정 결과보고서는 오류보고서에서 수정 관련 정보를 추가 작성하여 활용한다.

수행 tip

- 수정된 프로그램은 형상관리를 통해 변경 이력을 반드시 관리해야 한다.

학습 3 교수 · 학습 방법

교수 방법

- 테스트와 디버깅의 차이점을 설명하고 테스트 설계자, 테스트 실행자 등 역할 분담을 통해 학습의 참여도를 제고한다.
- 학습자의 성과 향상, 수업 참여도를 제고하기 위해 스스로 학습 내용을 평가하고 피드백한다.
- 프로그램의 모듈과 기능을 테스트하기 위한 테스트케이스를 생성할 수 있도록 설명한다.
- 테스트 설계기법을 이해하고 테스트를 수행할 때 직접 기법을 선정할 수 있도록 설명한다.
- 테스트 오라클의 유형을 이해하고 실제 테스트케이스와 연계할 수 있도록 설명한다.
- 디버깅을 수행하기 위한 결합 및 버그 유형을 도출할 수 있도록 설명한다.
- 발견된 결합 및 잠재된 결합의 위험도를 계량화할 수 있도록 설명한다.
- 디버깅을 수행하는 방법과 디버깅 도구를 사용할 수 있도록 설명한다.
- 디버깅을 수행하기 위해 중단점(Breakpoint) 기술 설치 및 해제할 수 있도록 설명한다.

학습 방법

- 선수학습을 통해 학습 체크리스트를 작성하고 성과를 평가해 본다.
- 구현된 기능에 따른 테스트케이스를 직접 작성하고 커버리지를 도출해 본다.
- 테스트의 전체적인 프로세스를 학습하고 네트워크 응용프로그램 특성에 맞게 설계해 본다.
- 테스트 설계기법을 학습하고 테스트를 수행할 때 기법을 적용할 수 있도록 학습한다.
- 테스트 오라클의 유형을 학습하고 실제 테스트케이스와 연계할 수 있도록 학습한다.
- 디버깅을 수행하기 위한 결합 및 버그 유형을 도출할 수 있도록 학습한다.
- 발견된 결합 및 잠재된 결합의 위험도를 계량화할 수 있도록 학습하고 계량화 기법을 학습한다.
- 디버깅을 수행하는 방법과 디버깅 도구를 사용할 수 있도록 학습한다.
- 디버깅을 수행하기 위해 중단점(Breakpoint) 기술을 설치 및 해제할 수 있도록 학습한다.

학습 3 평 가

평가 준거

- 평가자는 피평가자가 수행 준거 및 평가 내용에 제시되어 있는 내용을 성공적으로 수행할 수 있는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
• 네트워크 프로그램 시험	- 명확한 기능구현을 위하여 결과를 시험(Test) 할 수 있다.			
• 네트워크 프로그램 디버깅 및 수정	- 계획된 기능구현을 위하여 결과를 디버깅(Debugging)할 수 있다. - 개발 방법론에 따라서 기능 구현 결과를 수정(Modify)할 수 있다.			

평가 방법

- 포트폴리오

학습내용	평가항목	성취수준		
		상	중	하
• 네트워크 프로그램 시험	- 네트워크 프로그램 시험(Test) 보고서			

- 문제해결 시나리오

학습내용	평가항목	성취수준		
		상	중	하
• 네트워크 프로그램 시험	- 네트워크 프로그램 시험(Test) 절차			
• 네트워크 프로그램 디버깅 및 수정	- 네트워크 프로그램 디버깅(Debugging) 과정 - 네트워크 프로그램 수정(Modify) 과정			

• 서술형시험

학습내용	평가항목	성취수준		
		상	중	하
• 네트워크 프로그램 시험	<ul style="list-style-type: none"> - 테스트 계획서 작성 방법 - 네트워크 프로그램 시험(Test) 레벨 별 수행 활동 - 네트워크 프로그램 시험(Test) 커버리지 			
• 네트워크 프로그램 디버깅 및 수정	<ul style="list-style-type: none"> - 중단점(Breakpoint) 설정 방법 - 디버깅 툴 사용법 - 오류보고서 작성 - 네트워크 프로그램 수정(Modify) - 프로그램 수정 시 형상관리 연계 방법 			

피드백

1. 포트폴리오

- 피평가자에게 평가항목에서 제시한 경우를 작성하고 이를 본인의 포트폴리오로 작성하게 한다. 원활하게 작성하지 못하는 경우에는 이를 바로잡아 주고, 포트폴리오의 기능을 지도한다.

2. 문제해결 시나리오

- 네트워크 프로그램 시험 및 디버깅할 때에 발생 가능한 문제를 제시하고 문제에 대한 해결 시나리오를 작성하게 하고 평가한 후 모범 사례를 선정하여 부연 설명한다.

3. 서술형시험

- 네트워크 프로그램 시험 및 디버깅에 대한 서술형 시험문제를 제출하여 평가하고, 평가 결과 일정 점수 이하인 학생들은 추가 학습한 후 그 결과를 제출하도록 한다.

학습 1	개발환경 분석하기
학습 2	기능 구현하기
학습 3	프로그램 디버깅하기

학습 4

프로그램 최적화하기

4-1. 삽입문구 작성 및 오류 최소화

학습 목표

- 개발 방법 기준에 따라서 네트워크 프로그래밍 시 삽입문구를 작성할 수 있다.
- 개발 방법 기준에 따라서 네트워크 프로그래밍 시 오류를 최소화할 수 있다.

필요 지식 /

① 코드 스멜(Code Smell)

소스코드에 상당히 문제가 있음을 나타내는 어떤 짐새를 나타내는 것으로 리팩토링을 통해서 코드 스멜(Code Smell)을 제거하고 품질 좋은 코드를 작성해야 한다.

- 같은 코드가 여러 곳에 존재하는 중복된 코드로 코드 스멜(Code Smell)이 발생한다.
- 클래스, 메소드, 함수, 프로시저의 길이가 매우 길어져서 발생한다.
- 다른 클래스의 메소드들을 너무 많이 사용하는 기능 집착으로 발생한다.
- 다른 클래스의 구현에 세밀하게 의존하는 클래스로서 부적절한 관계를 형성한다.
- 상속을 거부하는 현상으로 부모 클래스의 규약을 지키고 않은 채, 메소드 오버라이드 (Method Override)를 하는 경우에 발생한다.

② 프로그램 삽입문구

프로그램 수행에는 영향을 주지 않고 사람이 명확하게 판독하는 데 도움을 주기 위해 컴퓨터 프로그램과 제어 언어 또는 데이터 집합 내에 삽입해 넣는 정보를 말한다. 프로그램의 삽입 문구는 개발표준에 따라서 작성 원칙을 준수하여 적용하여야 한다.

③ 시큐어 코딩

소프트웨어 개발 과정에서 개발자의 지식 부족이나 실수, 프로그래밍 언어의 고유 특징 등으로 발생할 수 있는 취약점을 최소화하기 위하여, 설계 단계부터 보안을 고려하여 코드를 작성하는 개발 방식이다.

<표 4-1> 주요 보안 취약점

구분	내용
입력 데이터 검증 및 표현	프로그램 입력값에 대한 검증 누락, 데이터의 잘못된 형식지정, 일관되지 않은 언어셋 사용 등으로 인해 발생되는 보안약점 대응
보안기능	보안기능(인증, 접근제어, 기밀성 등)을 부적절하게 구현할 때 발생할 수 있는 보안약점으로 적절한 인증 없는 중요기능 허용 등을 대응
시간 및 상태	동시 수행을 지원하는 병렬 시스템, 여러 프로세스가 동작되는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안약점 대응
에러처리	에러를 처리하지 않거나, 불충분하게 처리하여 에러 정보에 중요 정보가 포함될 때 발생하는 취약점을 대응한다.
코드오류	타입변환, 자원의 부적절한 반환 등과 같은 개발자가 범할 수 있는 코딩 오류로 발생되는 취약점을 대응한다.
캡슐화	중요한 데이터 또는 기능성을 불충분하게 캡슐화하거나 잘못 사용함으로써 발생되는 보안약점으로 정보노출, 권한문제 등의 발생 오류에 대응
API 오용	의도된 사용에 반하는 방법으로 API를 사용하거나, 보안에 취약한 API를 사용하여 발생할 수 있는 보안약점에 대응한다.

④ 프로그램 오류 최소화 방안

네트워크 응용프로그램의 소스코드 오류, 실행 시 오류, 보안 오류 등이 발생할 수 있는 상황 및 케이스를 사전에 도출하여 오류가 최소화되도록 하며, 발생한 오류가 정상적인 기능과 Side Effect가 발생하지 않도록 해야 한다.

수행 내용 / 삽입문구 작성 및 오류 최소화하기

재료 · 자료

- 네트워크 응용프로그램 코드, 오류 프로그램 코드, 코딩표준, 코딩스타일, 소프트웨어 개발 보안 가이드

기기(장비 · 공구)

- 개발서버, 개발PC, 형상관리 시스템, 표준개발도구, 소프트웨어 버전관리도구, 데이터베이스 연결 도구

안전 · 유의사항

- 수정이 필요한 코드는 변경하되 형상관리 시스템을 통해 버전을 관리한다.
- 코딩표준, 코딩스타일을 준수하는 코드로 수정되어야 한다.
- 기준 작성 결과의 오류가 발생할 경우를 대비하여 전문가 회의를 통한 교차 확인 방법을 강구해야 한다.
- 개발타당성 분석 결과 및 개발기준을 반영하여 오류를 최소화해야 한다.
- 지속적으로 신기술을 파악하고 제품에 적용할 수 있는데 검토해야 한다.
- 네트워크 응용프로그램 개발을 위한 전 과정의 이해와 분석을 바탕으로 하여 오류 최소화에 일관성을 유지하여야 한다.
- 네트워크 프로그래밍의 오류 최소화는 코드 최적화와 연계되어 수행되어야 한다.

수행 순서

① 네트워크 프로그래밍 시 삽입문구 작성을 위한 원칙을 정의한다.

- 저작권 및 소유권에 대한 내용을 삽입문구로 작성한다.

프로그램 소스의 저작권 정보와 소유권 정보를 표시하고 모든 조항과 조건을 열거하는 대신에 표준 라이선스나 외부 문서에 대한 링크를 적어 놓아 둔다.

- 프로그램 코드의 정보를 제공하는 삽입문구를 작성한다.

코드의 기본적인 정보를 제공하면 프로그래머의 이해를 도울 수 있다. 단, 이때 삽입문구

로 전달하려는 정보는 나쁘게 작성된 코드를 보완하기 위한 것이어서는 안 된다.

3. 개발자의 의도를 명확하게 표현하는 삽입문구를 작성한다.

구현 방식을 결정하게 된 배경, 즉 구현 의도를 설명함으로써 프로그래머의 이해를 도울 수 있다.

4. 프로그램의 의미를 명료하게 밝히는 삽입문구를 작성한다.

인수 또는 반환 값을 명확하게 만들 수 없는 경우 의미를 명료하게 밝히기 위해서 삽입문구를 작성한다.

5. 메소드, 프로그램의 결과를 경고하는 삽입문구를 작성한다.

코드에 담긴 결함이나 특정 상황에서 예상치 못한 실행 결과를 돌려줄 수 있는 코드를 다른 프로그래머에게 경고할 목적으로 삽입문구를 작성한다.

6. 프로그램의 TODO(해야 할 작업)에 대한 삽입문구를 작성한다.

필요 없는 기능을 삭제하라는 알림, 문제를 봐 달라는 요청, 더 좋은 이름을 떠올려 달라는 부탁 등의 필요하다고 여기지만 당장 구현하기 어려운 업무를 기술한다.

7. 중요성을 강조하는 삽입문구를 작성한다.

중요하지만 대수롭지 않게 넘길 수 있는 부분을 강조하기 위하여 삽입문구를 이용할 수 있다.

8. 소설식의 장황한 삽입문구는 피해야 한다.

장황하거나 주절거리는 삽입문구는 삽입문구 자체를 이해하기 위해 시간을 소비하게 되므로 삽입문구를 작성하기로 했다면 시간을 들여 최고의 삽입문구를 작성한다.

9. 동일한 내용을 반복하는 삽입문구는 피한다.

프로그램의 코드를 정당화하는 삽입문구가 아닌 코드의 내용을 설명하는 삽입문구는 코드보다 읽고 이해하기까지 시간이 더 걸린다. 삽입문구가 코드보다 더 많은 정보를 제공할 수 없다.

수행 tip

- 프로그램 삽입문구의 작성 원칙은 개발 표준을 통해 사전에 정의하고 적용한다.

② 네트워크 프로그래밍 시 삽입문구를 작성한다.

1. 파일에 대한 정보를 소스의 가장 위 상단에 작성한다.

삽입문구는 소스의 상단에 작성하여 소스를 변경하기 전에 이력 정보를 확인하고 일관된 방식으로 소스를 변경하고 기능의 변화를 확인할 수 있어야 한다.

<표 4-2> 삽입문구 작성 예시

```
/**  
 * @FileName : socket.c (선택된 편집 파일 이름을 작성한다.)  
 * @Project : 네트워크 엔지니어링 (선택된 프로젝트 이름을 작성한다.)  
 * @Date : 2015-01-01 (현재의 날짜를 작성한다.)  
 * @작성자 : 홍길동 (작성자의 이름을 기록한다.)  
 * @변경이력 : 2015-01-01 초기버전 작성 (프로그램이 변경되는 경우 해당 이력을 간략하게 작성한다.)  
 * @프로그램 설명 : socket을 이용한 채팅프로그램 (프로그램의 기능을 간략하게 기술한다.)  
 */
```

2. 메소드의 바로 상단에 메소드에 대한 정보를 작성한다.

메소드에 삽입문구는 파리미터, Return 타입, 예외처리 등에 대해 상세히 기술하여 메소드 호출할 때 발생할 수 있는 오류를 사전에 예방할 수 있다.

<표 4-3> 삽입문구 작성 예시

```
/**  
 * @Method Name : getChatMemList (선택된 메소드의 이름을 작성한다.)  
 * @작성일 : 2015-01-01 (메소드를 작성한 일자를 기록한다.)  
 * @작성자 : 홍길동 (작성자의 이름을 작성한다.)  
 * @변경이력 : 2015-01-01 초기버전 작성 (메소드가 변경되는 경우 해당 이력을 간략하게 작성한다.)  
 * @Method 설명 : 연결된 채팅 멤버조회 (메소드가 수행하는 간단한 기능을 간략하게 기술한다.)  
 * @Parameter : int year, String name (메소드의 입력 파라미터를 기술한다.)  
 * @return : ArrayList (메소드의 리턴타입을 작성한다.)  
 * @예외처리 : IOException, Null Pointer Exception (메소드가 수행되는 도중에 발생할 수 있는 예외사항을 기술한다.)  
 */
```

3. 변수 또는 메소드 내부의 특정 라인에 대한 삽입문구를 작성한다.

특정 라인에 대한 삽입문구는 핵심 내용만 간략하게 작성하여 가시성을 확보한다.

<표 4-4> 삽입문구 작성 예시

```
int numMem = CountMems(); // CountMems 함수는 채팅 정보 데이터를 순환하여 현재 해당 채팅에 등록된 멤버수를 반환한다.
```

삽입문구를 달아 놓고 해당 코드를 보는 사람들에게 무엇을 하는 함수/변수인지 정확하게 표현하고 향후 변경할 때 영향도를 최소화 시켜야 한다..

수행 tip

- 프로그램의 삽입문구는 압축적이고 간략하게 작성하여 프로그램의 가독성 확보에 주력한다.

- ③ 네트워크 프로그래밍의 오류를 최소화를 위해 주로 발생하는 오류를 식별한다.
1. 프로그램을 구현할 때 사용되는 전용 함수를 사용하여 발생할 수 있는 오류를 파악한다.
 - (1) `connect()` 함수를 사용할 때 발생하는 오류를 식별한다.
 - (가) TCP 소켓의 경우 IP와 PORT 할당의 오류 또는 연결 요청 진행(Three-way Handshaking) 상에 발생하는 오류를 파악한다.
 - (나) UDP 소켓의 경우 IP와 PORT 할당의 오류가 발생한다.
 - (2) `close()` 함수를 사용할 때 발생하는 오류를 식별한다.

연결 설정되지 않았거나 닫아 줄 파일 디스크립터가 존재하지 않는 경우 발생되는 오류를 파악한다.
 - (3) `bind()` 함수를 실행할 때 발생하는 오류를 식별한다.

프로그램에서 소켓을 Close시킬 때 소켓을 소멸시켜도 바로 죽이지 않고 대기시키는 상태를 TIME_WAIT 상태라고 하는데, bind()된 소켓이 아직 죽지도 않았는데 같은 주소, 같은 포트로 또 다른 소켓이 bind() 요청을 해 오니 커널에서는 오류를 발생시킨다.
 - (4) `listen()` 함수를 사용할 때 발생하는 오류를 식별한다.

해당 소켓의 연결 대기 시 연결이 큐의 길이를 초과하는 경우 발생하는 오류이다.
 - (5) `read()` 함수를 사용할 때 발생하는 오류를 식별한다.

읽어 들인 파일을 데이터의 최대 길이를 초과하거나 읽을 파일이 존재하지 않을 경우 발생하는 오류를 파악한다.
 2. 일반적인 코딩 과정에서 발생할 수 있는 오류이다.
 - (1) 메모리 누수(Memory Leak)로 인한 오류를 식별한다.

메모리가 프리되지 않고 프로그램에 계속 할당하는 현상이다. 가비지 컬렉션이 자동으로 되지 않는 C, C++ 언어에서 자주 발생될 수 있는 오류이다.
 - (2) 중복된 프리 선언으로 인한 오류를 식별한다.

프로그램 안에서 사용하는 자원은 먼저 할당되고 사용한 후에는 프리로 선언하게 된다. 이미 프리로 선언된 자원을 다시 프리로 선언하게 되어 발생하는 오류이다.
 - (3) Null의 사용으로 인한 오류를 식별한다.

Null을 포인트로 하고 있는 곳의 Content를 접근하려고 하는 경우 발생하는 오류이다.
 - (4) 별칭(Alias)의 남용을 인한 오류를 식별한다.

서로 다른 주소값을 예상하고 사용한 두 개의 변수의 값이 별칭 선언으로 인하여 같은 값이 되었을 때 오류가 발생한다.
 - (5) 배열의 인덱스로 인한 오류를 식별한다.

배열의 인덱스가 배열의 크기 한도를 벗어나면 예외 오류가 발생한다.

(6) 하나 차이로 인한 오류를 식별한다.

'0'으로 시작해야 하는데 '1'로 시작하는 경우, ' ≤ 0 '을 ' < 0 '로 프로그래밍 하는 경우에 오류가 발생한다.

(7) 사용자 정의 자료형의 오류를 식별한다.

사용자 정의 자료형을 다룰 때 오버플로나 언더플로 오류가 발생하기 쉽다.

(8) 문자열 처리에 의한 오류를 식별한다.

프로그래밍 언어의 문자열 처리 함수에서 매개 변수가 Null이면 오류가 발생한다.

(9) 버퍼 크기 등으로 인한 오류를 식별한다.

버퍼의 크기보다 더 큰 입력값을 고의로 주입하여 스택 버퍼 오버플로를 발생시킨다.

(10) 동기화의 오류를 식별한다.

공통 자원에 접근하는 다수의 스레드가 있는 병렬 프로그램에서 발생하는 오류이다.

3. 네트워크 프로그래밍 보안 미준수로 발생할 수 있는 오류이다.

(1) (입력데이터 검증 및 표현) SQL 삽입으로 인한 오류를 식별한다.

데이터베이스와 연동된 애플리케이션에서 입력된 데이터에 대한 유효성을 검증하지 않을 경우, DB로부터 정보를 열람하거나 조작할 수 있다.

(2) (보안기능) 적절한 인증 없는 중요기능을 허용하는 오류를 식별한다.

적절한 인증 과정 없이 중요 정보를 열람하거나 변경할 때 발생하는 오류이다.

(3) (시간 및 상태) 제어문을 사용하지 않는 재귀 함수로 인한 오류를 식별한다.

재귀의 순환횟수를 제어하지 못하면 할당된 메모리나 프로그램 스택 등의 자원을 과다하게 사용하여 발생하는 오류이다.

(4) (에러처리) 오류메시지를 통한 정보 노출로 인한 오류를 식별한다.

예외발생시 예외이름이나 스택 트레이스를 출력하는 경우, 프로그램 내부 구조가 노출되는 오류이다.

(5) (코드오류) 부적절한 자원해제로 인한 오류를 식별한다.

열린 파일 디스크립터, 힙 메모리, 소켓 등은 유한한 자원이므로 할당받아 사용한 후 해당 자원을 반납하지 않아서 발생하는 오류이다.

(6) (캡슐화) 잘못된 세션에 의한 데이터 정보노출로 인한 오류를 식별한다.

다중 스레드 환경에서는 싱글턴 객체 필드에 경쟁조건이 발생하는 오류이다.

(7) (API 오용) DNS Lookup에 의존한 보안결정으로 인한 오류를 식별한다.

공격자가 DNS 엔트리를 속일 수 있으므로 도메인명에 의존하여 보안결정(인증 및 접

근통제 등)을 하는 경우 발생하는 오류이다.

<표 4-5> 보안 미준수로 인해 발생할 수 있는 오류

구분	내용
취약한 소스	#include <stdio.h> #include <stdlib.h> #define BUFSIZE 100; void f() { char buf[BUFSIZE]; gets(buf); }
오류 내용	입력데이터 검증 및 표현 누락의 버퍼 오버플로 문제를 유발한다. buf는 BUFSIZE의 배열, gets()함수는 그 크기와 상관없이 입력 값을 buf에 저장하기 때문에 버퍼 오버플로우를 유발 할 수 있다.

출처: 소프트웨어 개발보안 가이드

수행 tip

- 주로 발생하는 오류 목록은 인시던트 리포트와 연계하여 관리한다.

④ 네트워크 프로그래밍 시 발생하는 오류를 최소화한다.

1. 네트워크 프로그램을 구현할 때 사용되는 전용 함수를 사용 발생하는 오류를 최소화한다.

(1) connect() 함수를 사용할 때 발생 오류를 최소화한다.

(가) TCP 소켓의 경우 정해진 IP와 PORT를 할당하고 연결 요청 진행(Three-way Handshaking)을 점검한다.

(나) UDP 소켓의 경우 정해진 IP와 PORT 할당하여 오류를 최소화한다.

(2) close() 함수를 사용할 때 발생 오류를 최소화한다.

(가) 연결의 설정 여부를 확인하고 close() 함수를 사용한다.

(나) 열린 파일 디스크립터가 존재하지 확인하고 close() 함수를 사용한다.

(2) bind() 실행 에러를 최소화한다.

SO_REUSEADDR를 지정해 주면 같은 포트에 대해 다른 소켓이 bind()되는 것을 허락해 주기 때문에 bind()에러 없이 프로그램을 실행할 수 있다.

(3) listen() 함수를 사용할 때 발생하는 오류를 최소화한다.

소켓 디스크립터를 정의하고 최대 연결 가능한 큐의 길이를 지정한다.

(4) read() 함수를 사용할 때 발생하는 오류를 최소화한다.

- (가) 읽어 들일 파일이 존재하는 미리 확인하고 `read()` 함수를 사용한다.
- (나) 읽어 들일 파일의 데이터 최대 길이를 변수로 선언한다.
2. 일반적인 코딩 과정에서 발생하는 오류를 최소화한다.
- (1) 메모리 누수(Memory Leak)로 인한 오류를 최소화한다.
- 메모리를 사용한 후 반드시 반납하고 프로그램에 할당해야 한다.
- (2) 중복된 프리 선언으로 인한 오류를 최소화한다.
- 자원이 할당되고 프리 선언된 자원은 또다시 프리를 선언하지 않는다.
- (3) Null의 사용으로 인한 오류를 최소화한다.
- Null이 될 수 있는 레퍼런스(Reference)는 참조하기 전에 Null 값인지를 검사하여 안전한 경우에만 사용한다.
- (4) 별칭(Alias)의 남용으로 인한 오류를 최소화한다.
- 변수 등에서 별칭(Alias)을 사용할 때 중복되어 별칭을 생성하지 않도록 한다.
- (5) 배열의 인덱스로 인한 오류를 최소화한다.
- 배열 인덱스의 최대 크기를 비교하여 입력되는 값을 제한하여야 한다.
- (6) 하나 차이로 인한 오류를 최소화한다.
- 프로그램 설계서를 정확히 분석하고, 디버깅을 통해 오류를 신속히 발견하고 개선한다.
- (7) 사용자 정의 자료형으로 인한 오류를 최소화한다.
- 사용자 정의 자료형은 삽입문구를 정확히 작성하고 자료형의 반환 타입 등을 확인한다.
- (8) 문자열 처리에 의한 오류를 최소화한다.
- 매개변수가 널이지 여부 등을 체크하고 문자열 처리를 수행한다.
- (9) 버퍼 사용으로 인한 오류를 최소화한다.
- 입력값 크기를 버퍼의 최대 크기로 제한하여 버퍼 오버플로를 예방한다.
- (10) 동기화로 인한 오류를 최소화한다.
- 세마포어, 모니터, 상호배제 기법을 활용하여 자원 경쟁 동기화를 예방한다.
3. 네트워크 프로그래밍 보안 미준수로 발생하는 오류를 최소화한다.
- (1) (입력데이터 검증 및 표현) Sql 삽입의 오류를 최소화한다.
- Prepared Statement 객체를 이용하여 DB에 컴파일된 쿼리문을 전달하거나 특수 문자 필터링을 실시한다.
- (2) (보안기능) 적절한 인증 없는 중요기능 허용의 오류를 최소화한다.
- 중요한 정보는 채 인증 실시하고 클라이언트를 우회하여 서버에 접속하지 못하도록 설계한다.

- (3) (시간 및 상태) 제어문을 사용하지 않는 재귀 함수의 오류를 최소화한다.
재귀 호출할 때, 재귀 호출 수를 제한하거나, 초기값을 설정하여 재귀 호출을 제한한다.
- (4) (에러처리) 오류메시지를 통한 정보노출의 오류를 최소화한다.
오류메시지는 정해진 사용자에게 유용한 최소한의 정보만 포함하여 출력한다.
- (5) (코드오류) 부적절한 자원 해제의 오류를 최소화한다.
자원을 획득하여 사용한 후에는 반드시 자원을 해제하여 반환해야 한다.
- (6) (캡슐화) 잘못된 세션에 의한 데이터 정보 노출의 오류를 최소화한다.
싱글턴 패턴을 사용하는 경우, 변수 범위(Scope)에 주의를 기울여야 한다.
- (7) (API 오용) DNS Lookup에 의존한 보안 결정의 오류를 최소화한다.
보안 결정에서 도메인명을 이용한 DNS lookup을 하지 않도록 한다.

<표 4-6> 보안 미준수로 인해 발생할 수 있는 오류 제거

구분	내용
개선 소스	#include <stdio.h> #include <stdlib.h> #define BUFSIZE 100; void f() { char buf[BUFSIZE]; /* buf 할당된 메모리 이내에서만 문자들을 입력받음 */ fgets(buf, BUFSIZE, stdin); }
오류 개선 사항	버퍼 오버플로 공격에 취약한 gets() 함수 대신 fgets() 함수 사용

출처: 소프트웨어 개발보안 가이드

수행 tip

- 프로그램 코드 오류는 사전에 자주 발생하는 오류 유형을 정의하고 개발한 후 발생된 오류를 점검하고 코드를 개선한다.

4-2. 네트워크 프로그램 코드 최적화

학습 목표

- 개발 방법 기준에 따라서 네트워크 프로그래밍 시 코드를 최적화할 수 있다.

필요 지식 /

① 리팩토링

리팩토링은 Code Smell을 제거하기 위해 외부적 동작의 변화 없이 내부 구조를 변경하는 기법으로 설계개선, 이해용이성, 품질향상, 생산성 향상에 핵심적인 품질확보 기술이다.

1. 메소드 추출(Extract Method)

메소드 추출(Extract Method)는 그룹으로 함께 묶을 수 있는 코드 조각이 있으면, 코드의 목적이 잘 드러나도록 메소드의 이름을 지어 별도의 메소드를 추출하는 방법이다.

2. 임시변수교체(Replace Temp With Query)

임시변수교체(Replace Temp With Query)는 어떤 수식의 결과값을 저장하기 위해서 임시 변수를 사용하고 있다면, 수식을 추출해서 메소드로 만들고, 임시변수를 참조하는 곳을 찾아 모두 메소드 호출로 교체하는 방법이다.

3. 메소드 이동(Move Method)

메소드 이동(Move Method)는 메소드가 정의된 클래스보다 다른 클래스의 기능을 더 많이 사용하고 있다면, 이 메소드를 가장 많이 사용하고 있는 클래스에 비슷한 몸체를 가진 새로운 메소드를 생성한다.

4. 클래스 추출(Extract Class)

클래스 추출(Extract Class)는 두 개의 클래스가 해야 할 일을 하나의 클래스가 하고 있는 경우 새로운 클래스를 만들어 관련 있는 필드와 메소드를 예전 클래스에서 새로운 클래스로 이동한다.

5. 메소드 재명명(Rename Method)

- 메소드 이름이 그 목적을 드러내지 못하고 있다면 메소드의 이름을 변경한다.
- 두 서브 클래스가 동일한 필드를 가지고 있다면 해당 필드를 슈퍼클래스로 변경한다.

② 클린 코드(Clean Code)

논리가 간단하고 버그를 원천 차단할 수 있는 효율적인 코드작성을 위해 의존성을 최소로 하고 사람이 이해할 수 있는 가독성, 목적성이 뛰어난 명확한 코드이다. Clean Code는 의존성 최소화, 단일 책임의 원칙, 버그유입 최소화, 가독성 향상, 중복코드 최소화, 변경용이, 작은 코드 등의 특징이 있다.

③ 디자인 패턴

소프트웨어 엔지니어의 경험, 프로그래머들이 유용하다고 생각되는 객체들 간의 일반적인 상호작용 방법들을 모은 목록이다. Interface Class를 사용, Delegation 사용, Coupling 최소화를 통해 유연하고 응집도가 높은 소프트웨어를 만들 수 있다.

1. 생성패턴

객체의 생성방식을 결정하는 패턴으로 Factory Method, Singleton, Builder, Abstract Factory, Prototype 패턴 등이 있다.

2. 구조패턴

객체를 조직화하기 위한 패턴으로 Decorator, Adapter, Bridge, Composite, Façade, Flyweight, Proxy 패턴 등이 있다.

3. 행위패턴

객체의 행위를 조정, 관리하기 위한 패턴으로 Template Method, Interpreter, Iterator, Strategy, Visitor, Chain of Responsibility, Mediator, Observer, Memento, State, Command 패턴 등이 있다.

④ 소프트웨어 아키텍처

시스템의 품질 보장, 개발생산성 향상 위해 시스템을 구성하는 컴포넌트 및 그들 사이의 관계를 설계한 시스템의 전체적인 구조이다. 사용자의 요구사항과 구현될 시스템 간의 Gap을 최소화한다. 주요 구성요소로는 Component(구체화된 시스템을 구성하는 기본이 되는 조직), Relationship(각종 소프트웨어의 컴포넌트 간의 관계 정의), Principal(소프트웨어 디자인과 진화를 이끄는 기본적인 이론) 등이 있다.

수행 내용 / 네트워크 프로그램 코드 최적화하기

재료 · 자료

- 코딩 표준, 코딩 스타일, 디자인패턴 유형, Smell Code, 리팩토링 기법, 시스템 개발계획서, 프로그래밍 언어, IEEE 1471 기반 소프트웨어 아키텍처, 테스트 케이스

기기(장비 · 공구)

- 개발서버, 개발PC, 프로젝트 관리도구, 버그 트레킹 시스템, 컴파일러, 표준개발도구

안전 · 유의사항

- 기술 문서를 참고하여 시스템적인 접근 방식으로 소프트웨어를 최적화한다.
- 개발 표준, 코딩 가이드를 참조하여 코드를 최적화해야 한다.
- 형상관리를 통해 변경된 프로그램 코드를 변경하고 이력을 관리해야 한다.

수행 순서

① 프로그램 표준 코딩규칙을 적용하여 코드를 최적화한다.

1. 네트워크 응용프로그램의 클래스, 메소드, 변수 명칭을 최적화한다.

(1) 클래스, 메소드, 변수 명칭의 길이는 31자 이내로 작성해야 한다.

명칭의 길이는 컴파일러에 따라서 제한하는 경우가 있으며, 명칭의 길이가 필요 이상으로 길어질 경우 오타 발생 확률이 증가되며 가독성이 떨어진다.

<표 4-7> 클래스, 메소드, 변수 명칭 길이 작성 예시

```
public class InfoMgtFromDBServer { //클래스명이 31자 초과하지 않음.  
    String collectInfoFromDBServer() { //메서드명이 31자 초과하지 않음.  
        int init_dbmgt_sd_status=0; //변수명이 31자 초과하지 않음.  
        ...  
    }  
}
```

(2) 동일한 변수명과 메소드명을 사용하지 않는다.

동일한 이름을 가진 변수명과 메소드명은 가독성이 떨어진다. 개발자가 변수에 접근 시에 메소드를 호출할 수 있기 때문에 혼란을 줄 수 있다.

<표 4-8> 상이한 변수명과 메소드명 작성 예시

```
public class Foo {  
    //멤버변수명과 메소드명이 상이.  
    private String name;  
    public String getName() {  
        return name;  
    }  
}
```

- (3) 명칭은 “_”이외의 특수 문자를 사용하지 않는다

다른 프로그래밍 언어와 명칭에 대한 호환을 위해서 특수 문자는 “_”만 허용한다.

<표 4-9> 변수명에 특수 문자 미사용 예시

```
int count_email_others=0; //변수명에 “_”이외의 특수문자 사용하지 않음.
```

2. 네트워크 프로그램의 소스 형식을 최적화한다.

- (1) 하나의 소스 파일은 2000줄 이내로 작성한다.

하나의 파일에 너무 많은 코드를 작성할 경우, 프로그램의 문맥 파악이 어렵고, 파일 관리 및 유지 보수 측면에서도 효율성이 떨어진다.

- (2) 한 줄의 길이는 80자 이내의 문자로 한다.

한 줄에 많은 코드를 작성할 경우 가독성이 및 유지보수성이 떨어진다.

- (3) 메소드나 함수의 내용은 70줄 이내로 작성한다.

한 화면에 전체 내용이 보이지 않아, 메소드(또는 함수)의 앞부분과 뒷부분의 문맥을 파악하기 어려워 가독성이 떨어진다.

- (4) 중괄호는 시작 문장의 마지막 열에 삽입하고 닫는 중괄호는 새로운 시작 열에 삽입 한다.(if문, elseif문, else문, for문, while문, doWhile문, switch문, trycatch문)

중괄호를 별도의 줄에 삽입하는 것은 코드 길이가 더 길어져서 아래위로 스크롤해가며 읽어야 함으로 가독성이 떨어진다.

- (5) 하나의 문장이 여러 줄로 작성될 경우 다음과 같은 규칙에 의해 행을 나눈다.

(가) 80자가 넘을 경우, 쉼표 다음 문자부터 새로운 행을 시작한다.

(나) 이전 행과 동일한 수준의 표현식과 열을 맞춘다.

하나의 문장을 여러 줄로 작성할 경우 명확하게 구분하지 않으면 소스코드의 가독성이 떨어진다.

<표 4-10> 표현식에 열 맞춤 예시

```
int result = getEmployees(records_from_employee_table,read_count,
    records_to_employee_table,write_count);
//한 문장이 여러 줄에 작성될 경우 인자의 “,” 이후 내용을 다음 줄에 위치.
//두번째 줄에서 들여쓰기가 이전 행과 동일 수준의 열을 맞춤.
```

- (6) 동일 수준의 문장은 같은 위치에서 시작하고 끝난다.

문장의 시작/끝을 명확하게 구분하기 위해서는 시작과 끝의 위치를 맞추어야 한다. 같은 수준의 문장에 동일한 들여쓰기를 적용하지 않을 경우 가독성이 저하된다.

<표 4-11> 동일 수준 문장의 동일 위치 종결 예시

```
if(score==MAX) {
    score--;
    return score;//동일 수준의 들여쓰기를 적용.
}
```

3. 네트워크 응용프로그램의 삽입문구를 최적화한다.

- (1) 프로그램은 최초작성자, 최초작성일, 최종변경일, 목적, 개정이력 및 저작권을 기술하는 삽입문구로 시작해야 한다.

각 파일의 목적을 정확히 파악할 수 있도록 주석 처리하여 소스코드의 가독성과 유지보수성을 높인다.

<표 4-12> 프로그램의 시작 부분 삽입문구 작성 예시

```
/*
 *최초작성자 : 김길동
 *최초작성일 : 2009.10.10.
 *최종변경일 : 2009.10.20.
 *목적 : 문자열을 조작하고 처리하는 유틸리티
 *개정이력 : 홍길동,2009.10.15,utf-8지원
 *
 *          홍길동,2009.10.20,checksum 기능 추가
 *저작권 : 대한민국(주)
 */
public class String Helper {  
    ...  
}
```

- (2) 메소드나 함수 주석은 목적, 매개변수, 반환값 및 변경 이력을 기술하는 주석으로 시작한다. 메소드, 함수 정의 앞부분에 다음의 정보를 기술하면, 유지보수성, 가독성이 향상된다.

4. 네트워크 응용프로그램의 변수 선언을 최적화한다.

- (1) 같은 용도의 변수는 같은 선언에 둔다.

하나의 문장에 구분자를 통해 서로 다른 용도의 변수를 선언하면 각 변수에 대한 의미를 정확하게 전달하기 어렵다.

(2) 불필요한 변수를 선언하지 않는다.

불필요한 변수 선언으로 개발자로 하여금 코드에 대한 이해를 저하시켜 유지보수가 어렵다.

(3) 배열을 선언하는 경우 반드시 요소 수를 명시적으로 선언하거나 초기화에 의해 둑 시적으로 결정되도록 한다.

요소 수가 결정되지 않은 배열은 배열 범위를 예측할 수 없으므로 배열 범위를 벗어나는 참조를 통한 보안 문제를 유발할 수 있다.

<표 4-13> 배열 초기화 예시

```
int arry[] = {1,2}; //배열을 초기화.
```

(4) 지역 변수는 선언과 동시에 초기화한다.

지역 변수는 선언과 동시에 초기화하여 초기화되지 않은 변수의 사용을 사전에 방지해야 한다.

※ C인 경우 지역 변수 초기화가 필수이지만, Java인 경우는 컴파일러가 이를 탐지하기 때문에 해당 사항이 없다.

5. 네트워크 응용프로그램의 상수를 최적화한다.

(1) 8진수로 표현된 상수를 사용하지 않는다.

8진수는 가독성이 매우 떨어지므로, 10진수나 16진수 표기법을 사용한다.

<표 4-14> 10진수나 16진수 표기법 예시

```
int hexa = 0xFF; //HEX 표현  
int decimal = 255; //10진수 표현
```

(2) 숫자 리터럴을 직접적으로 소스코드 안에 삽입하지 않는다.

반복문의 카운터로 사용될 수 있는 (-1,0,1)을 제외한 숫자는 가독성을 저하시킨다. 소스코드에 직접 기술한 숫자 리터럴은 그 의미를 파악하기 어렵다.

6. 네트워크 응용프로그램의 수식을 최적화한다.

(1) 단항 연산자는 피연산자와 붙여 쓴다.

단항 연산자는 피연산자가 무엇인지 명확하게 알 수 있도록 붙여 쓰지 않을 경우 가독성이 저하된다.

(2) (.dot)연산자를 제외한 모든 이항 연산자 전후는 공백으로 구분한다.

이항 연산자 전후에 공백을 삽입하는 것은 연산자와 피연산자의 구분을 명확하게 할 수 있으며, 소스코드의 가독성을 높일 수 있다.

(3) 조건부 연산자에서 “?” 연산자 앞에 이항 연산식이 나타날 경우 괄호로 구분한다.

조건부 연산자는 3개의 피연산자를 갖기 때문에 복잡한 구조를 가진다. 또한 첫 번째 피연산자가 이항 연산식이 될 경우 혼란을 가중시킬 수 있다.

<표 4-15> 조건부 연산자 ‘?’ 표기법 예시

($x \geq 0$) ? $x : -x$; //이항 연산식에 괄호 붙임.

(4) 증감 연산자는 수식에서 다른 연산자와 결합하여 사용하지 않는다.

증감 연산자가 다른 연산자와 결합하여 사용되면 직관적으로 이해할 수 없게 되어 가독성이 떨어진다.

(5) .(dot),->(포인터)연산자를 제외한 모든 이항 연산자 전후는 공백으로 구분한다.

이항 연산자와 피연산자 전후에 공백을 두는 것은 구분을 명확하게 하지 않을 경우 가독성이 떨어진다.

(6) 3개 이상의 연산자를 사용하는 경우 괄호로 연산의 우선순위를 표현한다.

연산자의 우선순위와 결합법칙에 적합한 수식도 3개 이상의 연산자를 사용하는 경우 가독성이 떨어진다.

(7) 비트 연산자는 부호 있는 자료에 사용하지 않는다.

비트 연산자는 부호의 의미를 포함하지 않으므로, 부호 있는 연산자에 시프트 연산을 하게 되면 부호 비트와 숫자값을 가지는 비트가 그 의미를 잃게 된다.

<표 4-16> 비트 연산자 사용 예시

unsigned int j = 10;
unsinged int k = j << 2; //양수에 비트연산자 적용.

7. 네트워크 응용프로그램의 문장을 최적화한다.

(1) Switch-case 문장에서 Case문에 Break문이 없는 경우 주석을 작성한다.

Switch-case 문장에서 Case문에 Break문이 없는 경우, 다음 Case문으로 제어의 흐름이 넘어간다. 주석이 없는 경우 프로그램 이해도가 떨어져 가독성이 저하된다.

(2) Switch-case문에서는 반드시 Default문을 작성하고 마지막 항목에 위치한다.

Default문을 작성하지 않으면, 모든 Case문을 만족하지 않을 경우에 대한 처리가 누락이 되어 프로그램 오류가 발생할 확률이 높다

(3) Goto문을 사용하지 않는다.

Goto문은 프로그램 제어의 흐름을 복잡하게 하여, 가독성이 저하되며, 제어의 흐름이 예측할 수 없는 방향으로 진행되어 오류발생 가능성이 높다.

(4) For문을 제어하는 수식에 실수 값을 사용하지 않는다.

실수 같은 표현 한계로 인하여 부정확한 값을 가지는 경우가 대부분이므로 의도하지 않은 동작을 유발할 수 있다.

<표 4-17> For 문 사용 예시

```
int i = 0; //for문의 제어 인자로 정수를 사용  
for(i=0; i<10; i++)
```

(5) For문을 제어하는 수치 변수는 루프 내에서 변화되지 않아야 한다.

루프 내에서 루프 제어 변수가 변화할 경우 반복 횟수를 예측할 수 없어 프로그램 이해도가 떨어지고, 잘못 계산된 루프제한 변수는 무한루프를 발생시킨다.

(6) 반복문 내부에서 반복중단을 위한 Break문은 되도록 한번만 사용하도록 한다.

Break문이 산재되어 있는 경우 프로그램의 동작을 예측하기 어렵다.

(7) If ~ Else If 문은 반드시 Else 문으로 끝나도록 한다.

Else 문으로 종료하지 않는 경우, 처리가 누락되는 경우가 발생하여 오류를 발생시킬 수 있다.

<표 4-18> If 문 사용 예시

```
//If ~ Else If 문에 Else 문이 존재.  
if (value == 0) {  
    ...  
} else if (value < 0) {  
    ...  
} else {  
    ...  
}
```

수행 tip

- 코딩표준을 참조하여 코드 최적화를 수행하고 코드 인스펙션 수행을 병행해야 한다.

② 컴파일 수행 관점에서 네트워크 응용프로그램의 코드를 최적화한다.

1. 네트워크 응용프로그램의 지역(기본 블록 내) 최적화를 수행한다.

(1) 공통 부분식(Common Subexpression Elimination)을 제거한다.

프로그램에서 공통된 부분이 반복되어 나타나는 경우를 제거하는 방법이다.

<표 4-19> 공통 부분식 제거

AS-IS	TO-BE
$A := B + C + D;$	$X := B + C;$
$E := B + C + F;$	$A := X + D;$
$H := (B + C) * G;$	$E := X + F;$ $H := X * G;$

(2) 연산 강도를 경감(Strength Reduction)시킨다.

연산자의 비용이 적은 연산자로 바꾸는 방법이다.

<표 4-20> 연산 강도 경감

AS-IS	TO-BE
$Y = 2 * A;$	$Y = A + A;$
$X = A / 10;$	$X = A * 0.1;$

(3) 상수 풀딩(Constant Folding)을 수행하다.

컴파일 시간에 상수식을 직접 계산하여 그 결과를 사용하는 방법이다.

<표 4-21> 상수풀딩

AS-IS	TO-BE
$X := X1 + 4 * 5$	$X := X1 + 20$

(4) 상수 전파(Constant Propagation)를 수행한다.

고정된 값을 갖는 변수를 상수로 대치하는 방법이다.

<표 4-22> 상수 전파

AS-IS	TO-BE
$x := 3 * 4$	$X := 12$

(5) 대수학적 간소화(Algebraic Simplification)를 수행한다.

수학적인 대수 법칙을 이용하여 식을 간소화하는 방법이다.

2. 네트워크 응용프로그램의 루프 최적화를 수행한다.

전체 코드의 10%가 실행시간의 90%를 차지하는데 대부분의 실행 시간을 루프 내에서 소모하는 경우가 존재한다.

(1) 루프의 반복 횟수를 감소시키는 Loop Unrolling을 수행한다.

(2) 루프의 종료 조건을 검사하는 경우 0인지 아닌지를 검사한다.

3. 네트워크 응용프로그램의 전역 최적화를 수행한다.

기본 블록 간의 정보와 흐름 그래프를 이용하여 프로그램의 전체적인 흐름 분석을 통한 최적화를 수행한다.

수행 tip

- 컴파일 관점의 코드 최적화는 성능 등과 연관성이 높으므로 구현 코드를 인스펙션을 통해 점검한다.

③ 네트워크 응용프로그램에 리팩토링 기법을 적용하여 코드를 최적화한다.

1. 리팩토링할 대상을 선정한다.

(1) Smell Code, 가장 중요하고 자주 추가 및 변경되는 프로그램을 선정한다.

(2) 주요 리팩토링 대상은 다음과 같다.

반복되는 코드, 긴 메소드, 큰 클래스, 긴 매개변수 목록, 확산을 위한 변경, 샷건(산탄총) 수술, 기능에 대한 욕심, 큰 데이터 덩어리, 기본 자료형에 대한 집착, Switch문장, 임시 필드, 메시지 연결, 중간자적 입장, 높은 연관성 제거, 약간 부족한 라이브러리, 잘못된 상속, 불필요한 주석

2. 리팩토링을 수행할 유형을 선정한다.

<표 4-23> 리팩토링 수행 유형

구분	내용
물리적 재구성	코드의 이름을 변경하고 전체 구조를 변경한다.
물리적 재명명	필드, 변수, 메소드명에 대한 명명 규칙을 변경한다.
클래스 관계 재정의	클래스의 관계(호출 구조)를 변경하고 Dependency를 최소화한다.
클래스 계층화	익명클래스(추상클래스 및 인터페이스), 중첩클래스(동일 유형 클래스로)를 구현한다.

3. 선정된 대상의 테스트 코드를 작성한다.

리팩토링하기 전부터 프로그램에 오류가 있을 수도 있고, 리팩토링을 한 이후에 프로그램이 제대로 수행되는지를 점검하려면 테스트 코드를 작성해야 한다.

4. 코드를 분해한 후 재조립한다.

메소드 정리, 객체나 클래스 사이의 이동, 데이터의 재구성, 조건문을 이해하기 쉽게 단순화, 메소드 호출 단순화, 상속관계 재구성 등의 리팩토링 기법을 적용한다.

5. 재조립한 코드를 테스트한다.

사전에 작성된 테스트 코드를 통과하도록 지속적으로 반복하여 Clean Code를 구현한다.

6. 코드 재조립과 테스트를 반복하여 수행한다.

지속적으로 Smell Code를 찾아보고 제거해야 품질 좋은 소프트웨어를 제작할 수 있다.

수행 tip

- 리팩토링은 Clean Code가 구현될 때까지 반복적으로 수행해야 코드의 품질이 높아진다.

학습 4 교수 · 학습 방법

교수 방법

- 네트워크 프로그래밍 시 발생할 수 있는 오류 목록을 파일, 출력물 등의 형태로 사전에 제공하고 직접 오류 목록을 작성하게 한다.
- 학습자의 성과향상, 수업참여도를 제고하기 위해 스스로 학습내용을 평가하고 피드백한다.
- 리팩토링 기법을 이해하고 Smell Code를 제거하기 위해 리팩토링을 수행할 수 있도록 설명한다.
- 소프트웨어 아키텍처, IEEE 1471을 이해하고 소프트웨어 아키텍처 구성요소를 도출할 수 있도록 설명한다.
- 소스코드의 오류를 수정하기 위해 버깅, 뮤테이션 테스트할 수 있도록 설명한다.
- 소스코드를 수정한 후 기존 테스트케이스를 활용하여 회귀테스트를 할 수 있도록 설명한다.
- 프로그래밍 코딩 시 발생할 수 있는 오류 유형을 도출할 수 있도록 설명한다.
- 개발보안 가이드를 적용하여 프로그래밍 소스의 보안 오류를 취소화할 수 있도록 설명한다.

학습 방법

- 선수학습을 통해 학습 체크리스트를 작성하고 성과를 평가해 본다.
- 요구사항을 적극적으로 수용하려는 태도를 가지고 프로그램 최적화 작업에 돌입한다.
- 리팩토링 기법을 학습하고 Smell Code를 제거하기 위해 리팩토링을 수행할 수 있도록 학습한다.
- 소프트웨어 아키텍처, IEEE 1471을 학습하고 소프트웨어 아키텍처 구성요소를 학습한다.
- 소스코드의 오류를 수정하기 위해 버깅, 뮤테이션 테스트할 수 있도록 학습한다.
- 소스코드를 수정한 후 기존 테스트케이스를 활용하여 회귀테스트를 할 수 있도록 학습한다.
- 프로그래밍 코딩 시 발생할 수 있는 오류 유형을 도출할 수 있도록 학습한다.
- 개발보안 가이드를 학습하여 프로그래밍 소스의 보안 오류를 취소화할 수 있도록 학습한다.

학습 4 평 가

평가 준거

- 평가자는 피평가자가 수행 준거 및 평가 내용에 제시되어 있는 내용을 성공적으로 수행할 수 있는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습내용	평가항목	성취수준		
		상	중	하
삽입문구 작성 및 오류 최소화	- 개발 방법 기준에 따라서 네트워크 프로그래밍 시 삽입문구를 작성할 수 있다. - 개발 방법 기준에 따라서 네트워크 프로그래밍 시 오류를 최소화할 수 있다.			
네트워크 프로그램 코드 최적화	- 개발 방법 기준에 따라서 네트워크 프로그래밍 시 코드를 최적화할 수 있다.			

평가 방법

- 문제해결 시나리오

학습내용	평가항목	성취수준		
		상	중	하
삽입문구 작성 및 오류 최소화	- 네트워크 프로그래밍 시 삽입문구 작성과정 - 네트워크 프로그래밍 시 오류 최소화 과정			
네트워크 프로그램 코드 최적화	- 네트워크 프로그래밍 시 코드 최적화 과정			

• 사례연구

학습내용	평가항목	성취수준		
		상	중	하
삽입문구 작성 및 오류 최소화	- 네트워크 프로그래밍 시 삽입문구 작성 사례			
	- 네트워크 프로그래밍 시 오류 최소화 사례			
네트워크 프로그램 코드 최적화	- 네트워크 프로그래밍 시 코드 최적화 사례			

• 평가자 질문

학습내용	평가항목	성취수준		
		상	중	하
삽입문구 작성 및 오류 최소화	- 네트워크 프로그래밍 시 삽입문구 작성법 질의			
	- 네트워크 프로그래밍 시 오류 최소화 과정 질의			
네트워크 프로그램 코드 최적화	- 네트워크 프로그래밍 시 코드 최적화 과정 질의			

피 드 백

- 문제해결 시나리오
 - 삽입문구 작성 및 오류 최소화, 코드를 최적화할 때에 발생할 수 있는 문제를 제시하고 문제에 대한 해결 시나리오를 작성하게 하고 평가한 후 모범 사례를 선정하여 부연 설명한다.
- 사례연구
 - 피평가자로 하여금 기존 유사 사례를 분석하고 이를 적용하는 과정을 설명하고 분석한 사례를 가지고 본인의 경우 대처할 수 있는 방안을 제시하도록 하고, 수행하지 못하는 경우에는 사례분석을 정확하게 실시하고 이를 본인의 경우에 맞추어 이에 대처할 수 있도록 지도 한다.
- 평가자질문
 - 피평가자가 수행하는 학습에 대하여 4~5개 정도의 간단한 질문을 통해 학습의 이해 여부를 평가하고 일정 기준에 미달하는 학생은 추가 학습하게 하고 재질문을 한다.

참고자료



- 강성원(2015). 『소프트웨어 아키텍처로의 초대』. 홍릉과학출판사.
- 권월일 외 3인(2008). 『개발자도 알아야 할 소프트웨어 테스팅 실무』. STA.
- 양대일(2011). 『정보보안 개론과 실습』. 한빛미디어.
- 원유현(2011). 『프로그래밍 언어 개념』. 정의사.
- 윤상배(2011). 『(뇌를 자극하는) TCP/IP 소켓 프로그래밍』. 한빛미디어.
- 이석호(2012). 『데이터베이스 시스템』. 정의사.
- 차동완 외 2인(2004). 『개념으로 풀어본 인터넷 정보기술』. 홍릉과학출판사.
- 최은만(2010). 『소프트웨어공학』. 정의사.
- 전자정부 표준 프레임워크 포털(작성일 불명). 개발 가이드. <http://www.egovframe.go.kr/>. 2015.7.01.
- 행정안전부(작성일 불명). 소프트웨어 개발보안 가이드. http://www.moi.go.kr/frt/bbs/type001/commonSelectBoardArticle.do;jsessionid=Gjt5CRV42fOznQGGNqdBA7xyP9lNB4TxlF5kox1pgoMxp9kZObm6GY1HeXSUMtMs.mopwas52_servlet_engine1?bbsId=BBSMSTR_00000000012&nttId=42149. 2015.8.10.
- Abraham Silberschatz 외 2인(2013). 『Operating System Concepts』. 홍릉과학출판사.

활용서식



작업 포트폴리오

20 년 월 일	학년 반 번 성명:
[개발환경 분석하기]	개발환경 분석 내용을 기재해 주세요. • • • •
[기능 구현하기]	기능 구현 내용을 기재해 주세요. • • • •
[프로그램 디버깅하기]	프로그램 디버깅 내용을 기재해 주세요. • • • • •
[프로그램 최적화하기]	프로그램 최적화 내용을 기재해 주세요. • • • •

교수자 확인 : _____

작업장 평가 서식

20 년 월 일	학년	반	번	성명:		
평가 관점	교수자의 평가					
	충분함	보통임	노력 요			
<ul style="list-style-type: none"> ■ 실습 도구 준비 및 정리 <ul style="list-style-type: none"> · 수업에 사용하는 실습 도구 준비를 함. · 수업에 사용한 실습 도구 정리를 함. 						
<ul style="list-style-type: none"> ■ 실습 과정 <ul style="list-style-type: none"> · 주의 깊게 실습하고 수업의 주요 내용을 노트에 기록함. · 실습할 때 아이디어를 냄. · 문제를 해결하는 과정에 참여함. (프로젝트를 수행할 때 주체가 되어 수행함.) 						
<ul style="list-style-type: none"> ■ 태도 <ul style="list-style-type: none"> · 선정한 문제(선정지 및 프로젝트)에 대해 질문과 호기심을 나타냄. · 인내심을 보임. 						
<ul style="list-style-type: none"> ■ 행동 반응 <ul style="list-style-type: none"> · 수업 시간에 시간을 잘 활용함. · 과제와 단계별 해결 방안을 제시간에 완수함. 						
<ul style="list-style-type: none"> ■ 협동 기능 <ul style="list-style-type: none"> · 팀별 과제를 나누는데 모둠원과 협동함. · 분담된 과제를 완성하고 협동함. 						

교수자 확인: _____

NCS학습모듈 개발이력

발행일	2015년 12월 31일		
세분류명	NW엔지니어링(20010205)		
개발기관	(사)한국정보통신기술사협회, 한국직업능력개발원 오병택(문엔지니어링)* 권태호(기술사협회) 김경구(한국고용정보원) 김태형(KB투자증권) 박대우(이씨오) 양전성(대영유비텍) 오영배(수원여자대학교) 오창현(새롬씨앤씨) 윤용식(특허정보원)		
집필진	검토진	도기철(농협은행) 박유현(주)엔투엠) 유은숙(한국정보화진흥원) 조동(중원대학교) 최경훈(한국IBM)	

* 표시는 대표집필자임

네트워크 프로그래밍 구현(LM2001020508_14v2)

저작권자	교육부
연구기관	한국직업능력개발원
발행일	2018. 12. 31.

* 이 학습모듈은 자격기본법 시행령(제8조 국가직무능력표준의 활용)에 의거하여 개발하였으며, NCS통합포털사이트(<http://www.ncs.go.kr>)에서 다운로드 할 수 있습니다.



www.ncs.go.kr