

정보처리기사 필기

소프트웨어 설계 4 화면설계 ①

양문자 선생님

출처 : ncs 학습모듈(NCS능력단위 화면+설계)

소프트웨어 설계

차례

1. 요구사항 확인

2. 화면 설계

1) UI 요구사항 확인

2) UI 설계

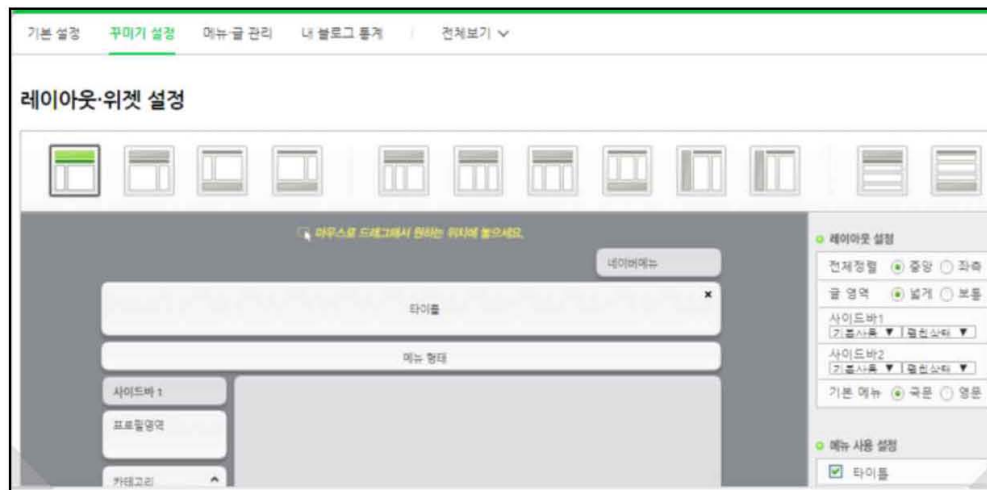
3. 애플리케이션 설계

4. 인터페이스 설계

소프트웨어 설계

1. UI 요구사항 확인

- 소프트웨어 아키텍처란?
 - 개발하고자 하는 소프트웨어의 사전 작업을 통하여 소프트웨어 개발을 쉽게 하도록 기본 틀을 만드는 것.
- 아키텍처 패턴이란?
 - 소프트웨어 아키텍처에서 일반적으로 발생하는 문제점들에 대한 일반화되고 재사용 가능한 솔루션

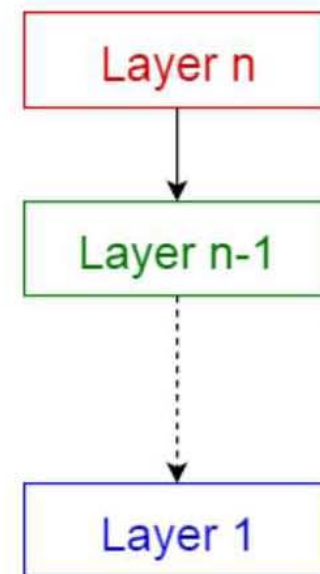


소프트웨어 설계

< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

1. 계층화 패턴 (Layered pattern) 레이어드 패턴

- 하위 모듈들의 그룹으로 나눌 수 있는 구조화된 프로그램
- 일반적인 데스크톱 애플리케이션, 전자상거래 웹 애플리케이션



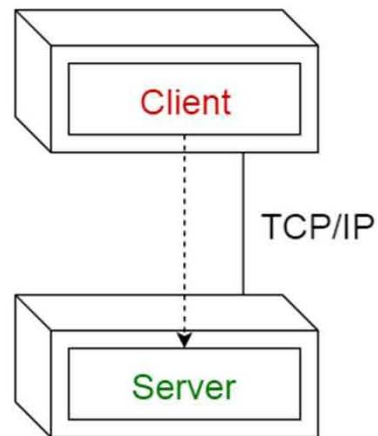
출처 : <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

소프트웨어 설계

< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

2. 클라이언트-서버 패턴 (Client-server pattern)

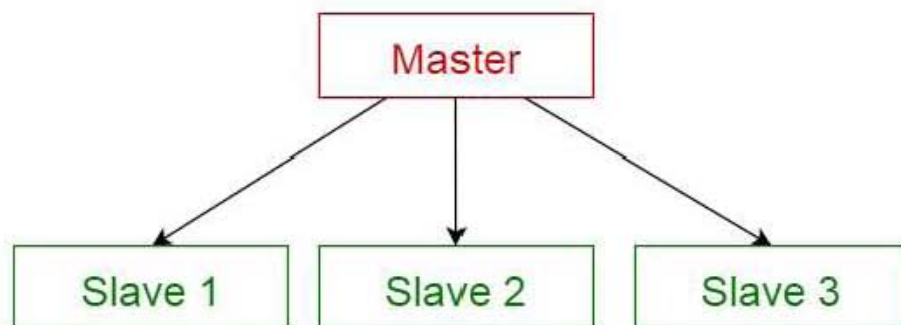
- 클라이언트가 서버에 서비스를 요청하면 서버는 클라이언트에게 적절한 서비스를 제공
- 이메일, 문서 공유 및 은행 등의 온라인 애플리케이션



< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

3. 마스터-슬레이브 패턴 (Master-slave pattern)

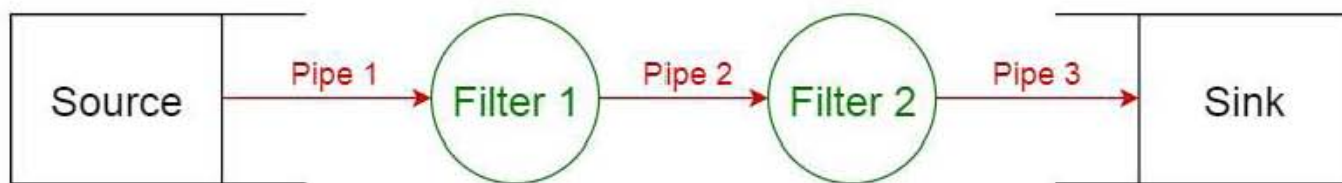
- 마스터 컴포넌트는 동등한 구조를 지닌 슬레이브 컴포넌트들로 작업을 분산하고, 슬레이브가 반환한 결과값으로부터 최종 결과값을 계산
- 컴퓨터 시스템에서 버스와 연결된 주변장치



< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

4. 파이프-필터 패턴 (Pipe-filter pattern)

- 각 처리 과정은 필터 (filter) 컴포넌트에서 이루어지며, 처리되는 데이터는 ****파이프 (pipes)****를 통해 흐른다. 이 파이프는 버퍼링 또는 동기화 목적으로 사용될 수 있다.
- 컴파일러. 연속한 필터들은 어휘 분석, 파싱, 의미 분석



< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

5. 브로커 패턴 (Broker pattern)

- 서버는 자신의 기능들(서비스 및 특성)을 브로커에 넘겨주며(publish), 클라이언트가 브로커에 서비스를 요청하면 브로커는 클라이언트를 자신의 레지스트리에 있는 적합한 서비스로 리디렉션한다.

6. 피어 투 피어 패턴 (Peer-to-peer pattern)

- 각 컴포넌트를 ****피어 (peers)****라고 부른다. 피어는 클라이언트로서 피어에게 서비스를 요청할 수도 있고, 서버로서 각 피어에게 서비스를 제공할 수도 있다.

7. 이벤트-버스 패턴 (Event-bus pattern)

- 소스는 이벤트 버스를 통해 특정 채널로 메시지를 발행하며 (publish), 리스너는 특정 채널에서 메시지를 구독한다
- 안드로이드 개발, 알림 서비스

< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

8. 모델-뷰-컨트롤러 패턴 (Model-view-controller pattern)

- MVC 패턴

- 모델 (model) — 핵심 기능과 데이터를 포함
- 뷰 (view) — 사용자에게 정보를 표시
- 컨트롤러 (controller) — 사용자로부터의 입력을 처리

9. 블랙보드 패턴 (Blackboard pattern)

- 결정 가능한 해결 전략이 알려지지 않은 문제에 유용

10. 인터프리터 패턴 (Interpreter pattern)

- 특정 언어로 작성된 프로그램을 해석하는 컴포넌트를 설계할 때 사용
- SQL과 같은 데이터베이스 쿼리 언어, 통신 프로토콜을 정의하기 위한 언어

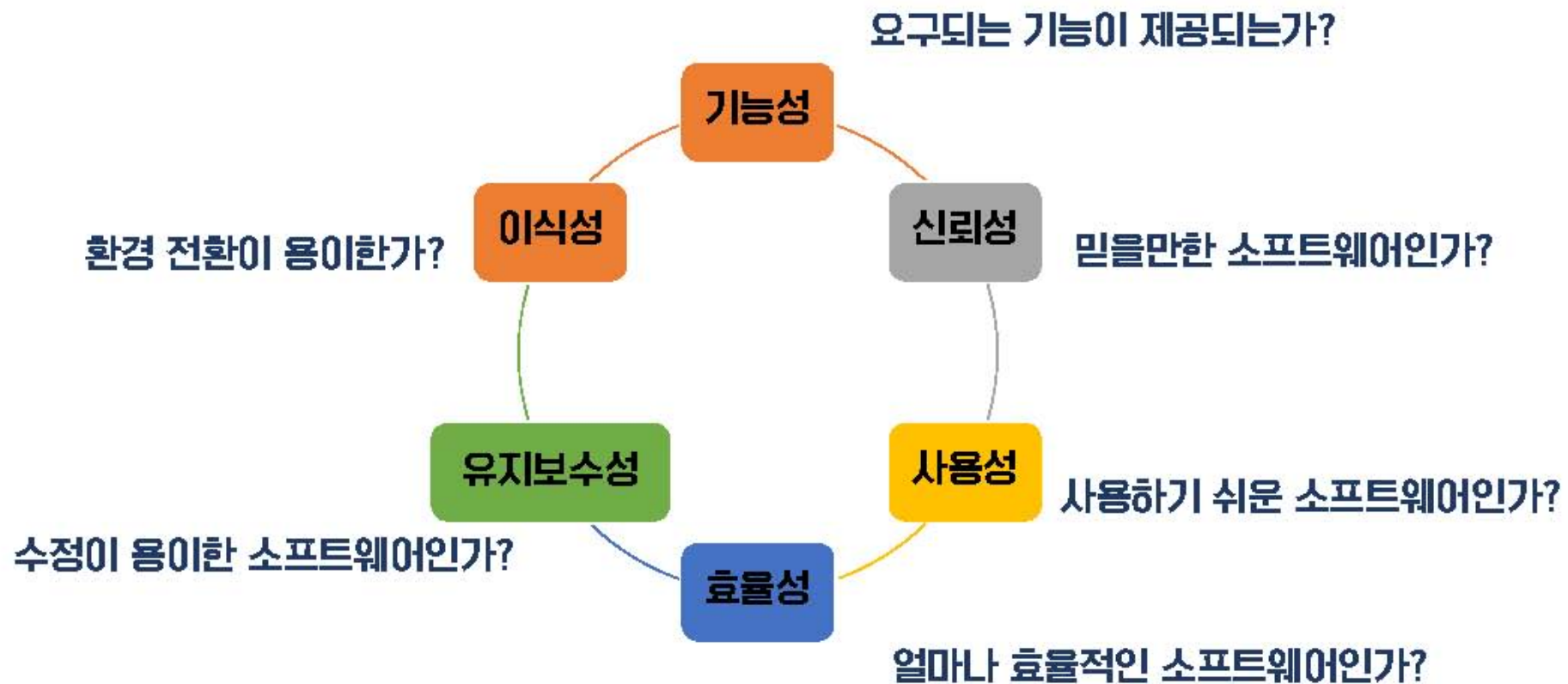
소프트웨어 설계

< 10가지 일반적인 소프트웨어 아키텍처 패턴 >

- 1) 소프트웨어 시스템의 구조를 구성하기 위한 기본 틀이 제시되어 개발 시간을 단축할 수 있다.
- 2) 검증된 구조로 개발되어 있어 안정적 개발이 가능하다.
- 3) 공통 아키텍처가 공유되므로 의사소통이 간편해진다.
- 4) 시스템 구조를 이해하기 쉬워 유지보수도 쉬워진다.

소프트웨어 설계

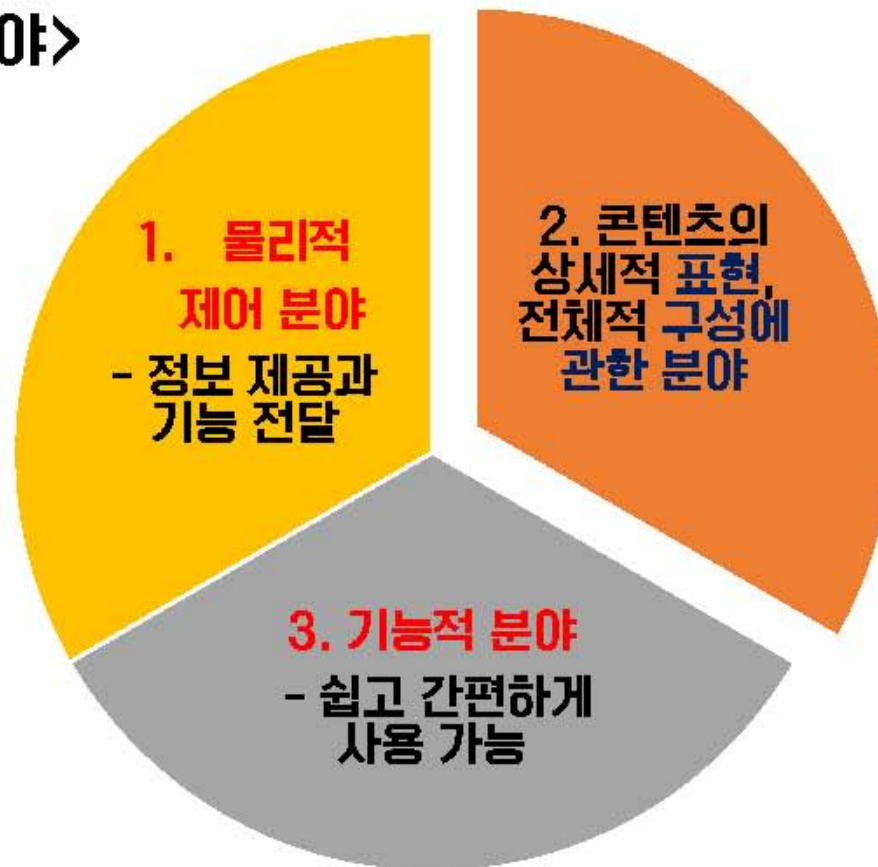
<소프트웨어 아키텍처 품질요구사항>



<UI란?>

- User Interface
 - 사용자가 시스템을 원활히 사용하도록 돕는 장치/소프트웨어
- UI의 종류
 - CLI(Command Line Interface) : 텍스트 기반 인터페이스
 - GUI(Graphic User Interface) : 그래픽 반응 기반 인터페이스
 - NUI(Natural User Interface) : 직관적 사용자 반응 인터페이스(터치, 음성등)

<UI의 세 가지 분야>



<UI의 설계 원칙>

- 직관성 : 누구나 쉽게 이해하고 사용할 수 있도록
- 유효성 : 사용자의 목적을 정확하게 달성할 수 있도록
- 학습성 : 누구나 쉽게 배우고 익힐 수 있도록
- 유연성 : 사용자의 요구사항을 최대한 수용, 오류를 최소화 하도록

소프트웨어 설계

<UI의 설계 지침 : UI 개발과정에서 꼭 지켜야 할 공통의 조건>

- **사용자 중심** : 사용자가 이해하기 편하고 쉽게 사용할 수 있는 환경을 제공하며 실사용자에 대한 이해가 바탕이 되어야 한다.
- **일관성** : 버튼이나 조작 방법을 사용자가 기억하기 쉽고 빠른 습득이 가능하게 설계하여야 한다.
- **단순성** : 조작 방법은 가장 간단하게 작동이 가능하도록 하여 인지적 부담을 감소시켜야 한다.
- **결과 예측 가능** : 작동시킬 기능만 보고도 결과 예측이 가능하여야 한다.
- **가시성** : 주요 기능을 메인 화면에 노출하여 조작이 쉽도록 하여야 한다.

소프트웨어 설계

- **표준화** : 디자인을 표준화하여 기능 구조의 선행 학습 이후 쉽게 사용할 수 있어야 한다.
- **접근성** : 사용자의 직무, 연령, 성별 등 다양한 계층을 수용하여야 한다.
- **명확성** : 사용자가 개념적으로 쉽게 인지하여야 한다.
- **오류 발생 해결** : 사용자가 오류에 대한 상황을 정확히 인지할 수 있어야 한다.

〈UI가 필요한 이유〉

- 오류 최소화, 적은 노력으로 원하는 결과를 얻도록
- 막연한 기능에 대해 구체적인 방법 제시
- 작업 시간 단축, 이해도 향상
- 정보 제공자와 공급자의 매개 역할

소프트웨어 설계

<UI 요구사항 확인>

1) 기능적 요구사항

- 1) 시스템의 입력
- 2) 시스템의 출력
- 3) 시스템이 저장할 데이터
- 4) 시스템이 수행할 연산
- 5) 기타 요구사항(ex: 동기화)

2) 비기능적 요구사항

- 1) 품질 - 사용성, 효율성, 신뢰성, 유지 보수성, 재사용성
- 2) 환경 - 플랫폼, 사용 기술
- 3) 프로젝트 계획 - 비용, 일정

<UI 표준 설계 과정>

UI 표준 : 전체 시스템에 공통으로 적용되는 화면구성등에 관한 규약

1. UI 스타일 가이드를 정의한다.

(1) 구동 환경을 정의한다.

(가) 컴퓨터 OS를 확인한다.

: 기업이 운영하고 있는 업무와 보안성이 높은 운영 체제(OS)를 확인한다.

(나) 웹 브라우저를 확인한다.

: 웹 브라우저는 익스플로러, 크롬, x-internet 등 기업 환경에 가장 적합한 것으로 확정한다.

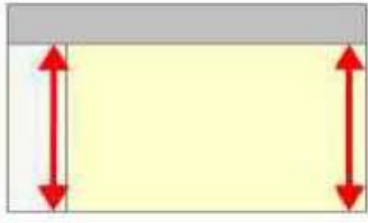


(다) 모니터 해상도를 확인한다.

: 모니터 해상도는 1280px*1024px을 표준으로 한다.

소프트웨어 설계

(라) 프레임 세트를 확인한다.

: 각 영역별(Top, Left,Contents 영역) 프레임을 구분해 적용한다.

구분	Frame 구분	One Frame	Web Application 경우
Contents 구성	<ul style="list-style-type: none"> 프레임별 Contents 구성 각 프레임 페이지에서 메뉴, 배너 구성등 일괄 적용됨 	<ul style="list-style-type: none"> 전체 페이지에서 각 area별 Contents를 자유롭게 구성 	<ul style="list-style-type: none"> page별 구성 Contents에 구성이 동일한 Pattern
Design	<ul style="list-style-type: none"> 각 Frame별 이미지 적용 	<ul style="list-style-type: none"> 전체 페이지에 적용되는 이미지 가능 	<ul style="list-style-type: none"> 업무처리가 주목적으로 페이지 전체에 이미지 적용하는 경우는 적음
속도	<ul style="list-style-type: none"> 변경되는 프레임만 새로 로딩 	<ul style="list-style-type: none"> 페이지 전체가 새로 로딩 	<ul style="list-style-type: none"> 메뉴 변경 시 페이지 로딩이 빨라야함
스크롤			

소프트웨어 설계

(2) 레이아웃을 정의한다.

(가) 화면 구조를 정의한다.

: 기본 배치(Layout)는 크게 Top, Left, Contents 영역의 3개 부분으로 설계

(나) 상단 메뉴 구성(Top Area)을 정의한다.

: 필수적으로 적용하는 사항.

시스템 로고(System Logo), 로그인 사용자(Login User), 바로가기 메뉴(Quick Menu), 주 메뉴(Main Navigation).



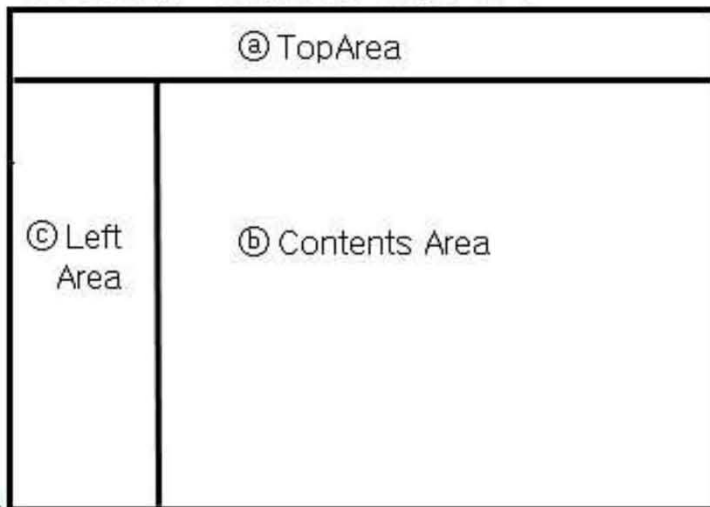
소프트웨어 설계

(다) 좌측 메뉴 구성(Left Area)을 정의한다.

: 선택적으로 적용하는 사항.

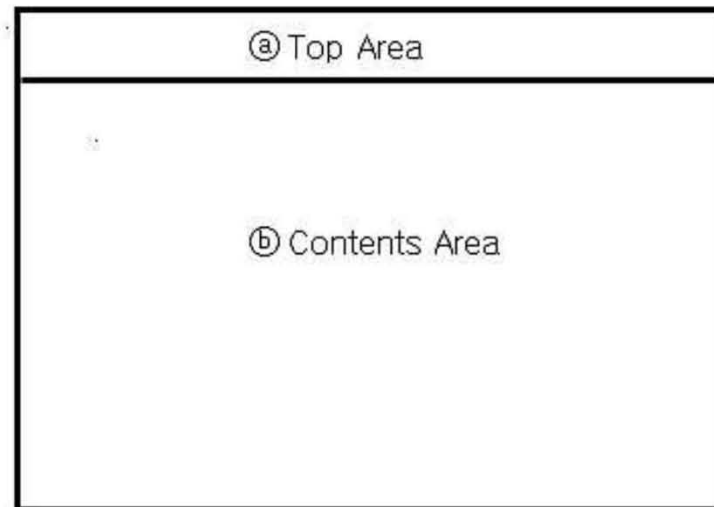
서브 메뉴(Sub Menu), 배너(Banner)

• layout 1안: Left Area 적용할 경우



12

• Layout 2안 : Left Area 적용 안 할 경우



소프트웨어 설계

(라) 내용 구성(Contents Area)을 정의한다.

: 필수적으로 적용하는 사항.

구성 요소로는 메인 이미지(Main Image), 시스템별 구성 콘텐츠.

(마) 하단 메뉴 구성(Footer Area)을 정의한다.

: 선택적으로 적용하는 사항.

구성 요소로는 회사 CI, Copyright가 있다.

(바) 사용 환경에 맞춰 페이지 폭을 정의한다.

소프트웨어 설계

(3) 네비게이션을 정의한다.

(가) 메뉴 네비게이션(Menu navigation)을 정의한다.

: 4가지 타입 - 기본(상단,왼쪽)과 변형



(4) 기능을 정의한다.

- (가) 프로세스 모델링(Process Modeling)을 정의한다. - 업무 과정에서 일어나는 모든 활동
- (나) 데이터 모델(Data Model)을 정의한다. - 적용할 업무 영역에 필요한 데이터 엔티티 식별

(5) 구성 요소를 정의한다.

- (가) 그리드를 정의한다. - 테이블 형태로 구성
- (나) 버튼을 정의한다. - 기능 버튼, 검색 버튼, 그리드(Grid) 관련 버튼, 기타 버튼

소프트웨어 설계

2. UI 패턴 모델을 정의한다.

(1) 업무 화면 클라이언트(Client)를 정의한다.

(2) 서버 컨트롤러(Controller)를 정의한다.

(3) 서버 메시지 및 예외(Exception) 처리를 정의한다.

유형	설명
S(System)	시스템 오류로 인해 발생. 런타임 예외를 전달할 때 사용되며, 모든 트랜잭션은 자동으로 복원(Rollback)
E(Error)	업무 처리 로직의 일환으로 애플리케이션 예외(Application Exception)를 throw할 때 사용한다. 이때 모든 트랜잭션은 자동으로 복원한다. (ex: E0001: 금액이 마이너스 값입니다.)
I(Information)	정상적인 업무 처리 결과나 관련 정보에 대한 확인 메시지를 사용자에게 알려주고자 할 때 사용한다. 이때 모든 트랜잭션은 실행(commit)한다. (ex : 고객 정보가 성공적으로 조회되었습니다.)

(4) 클라이언트(Client) - 서버 간 데이터 변환을 정의한다.

소프트웨어 설계

(5) EP(Enterprise Portal) 연계를 정의한다.

- EP-SSO(Single Sign ON)-Client 간 연계 방안

(6) 보고서를 정의한다.

- 클라이언트와 리포트 솔루션 간의 연계 방식

(7) 신 클라이언트(Thin Client)에 외부 컴포넌트 연계를 정의한다.

- 외부 UI컴포넌트를 도입할 시 서버와의 연계 방식을 결정

소프트웨어 설계

3. UI표준 수립을 위한 조직을 구성한다.

- (1) 조직 구성 및 역할을 정의한다.
- (2) 커뮤니케이션 방안을 수립한다.

4. UI표준을 위한 환경을 분석한다.

- (1) 사용자 트렌드를 분석한다.
- (2) 기능 및 설계를 분석한다.

<UI 프로토타입 이해>

- 프로토타입(Prototype)의 뜻

: “새로운 컴퓨터 시스템이나 소프트웨어의 설계 또는 성능, 구현 가능성, 운용 가능성을 평가하거나 요구 사항을 좀 더 잘 이해하고 결정하기 위하여 전체적인 기능을 간략한 형태로 구현한 시제품”(한국어사전)

- 프로토타입의 의미

: 사전에 프로토타입을 먼저 제작하고 이를 바탕으로 향후 설계될 UI의 적정성을 평가해 수정 보완함으로써 추후 발생 가능한 오류들을 사전에 방지하여 시스템 설계 및 개발에 소요되는 총 비용 및 노력을 절감할 수 있다.

〈UI 프로토타입의 장점과 단점〉

(1) 장점

- (가) 사용자 설득과 이해가 쉽다.
- (나) 개발 시간이 감소한다.
- (다) 오류를 사전에 발견할 수 있다.

(2) 단점

- (가) 너무 많은 수정 과정을 거친다면 오히려 작업 시간이 늘어날 수 있다. 사용자의 요구사항은 가능한 들어주되 적절한 타협이 필요하다.
- (나) 자원 효율성 관점에서 보면 필요 이상으로 자원을 많이 소모한다.
- (다) 정확한 문서 작업이 생략될 수 있다.

<UI 프로토타입 제작, 검토하기>

- I. 응용 소프트웨어 개발을 위한 UI 표준 및 지침에 의거하여, 소프트웨어 아키텍처의 설계 원리를 확인한다.

<도서 대출 예약 시스템의 도서 반납 유스케이스(USE CASE)>

1. 개요 : 사용자는 대출반납기기를 통하여 원하는 도서를 반납한다.

2. 액터 : 사용자

3. 이벤트 흐름

(1) 기본 사항

(가) 사용자는 '도서대출/반납메인화면'에서 반납 버튼을 누른다.

(나) 시스템은 반납할 도서의 인식을 요청하는 '반납도서인식요청화면'을 출력한다.

(다) 사용자는 반납하고자 하는 도서를 대출반납 기기에 인식시킨다.

(라) 시스템은 데이터베이스에서 대출 정보를 수정한다.

(마) 시스템은 반납 결과를 보여주는 '반납결과화면'을 출력한다.

(바) 사용자는 확인 버튼을 누른다.

(2) 추가 사항

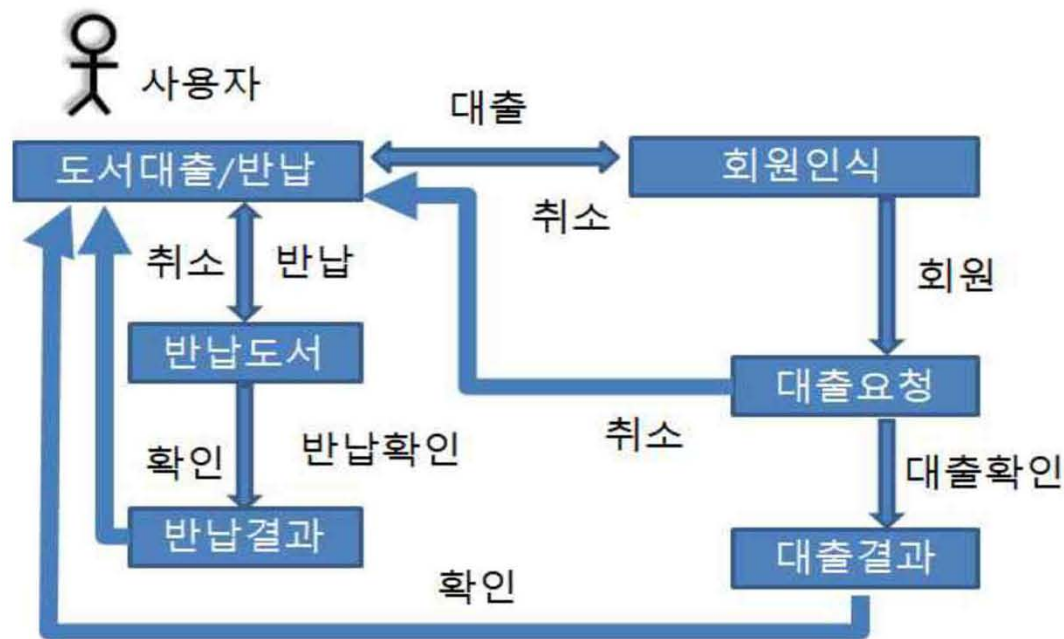
(가) 사용자가 '반납도서인식요청화면'에서 취소를 누를 경우

(나) 시스템은 '도서대출/반납메인화면'을 출력한다.

4. 처리 내용 : 데이터베이스에 대출 정보가 수정된다.

소프트웨어 설계

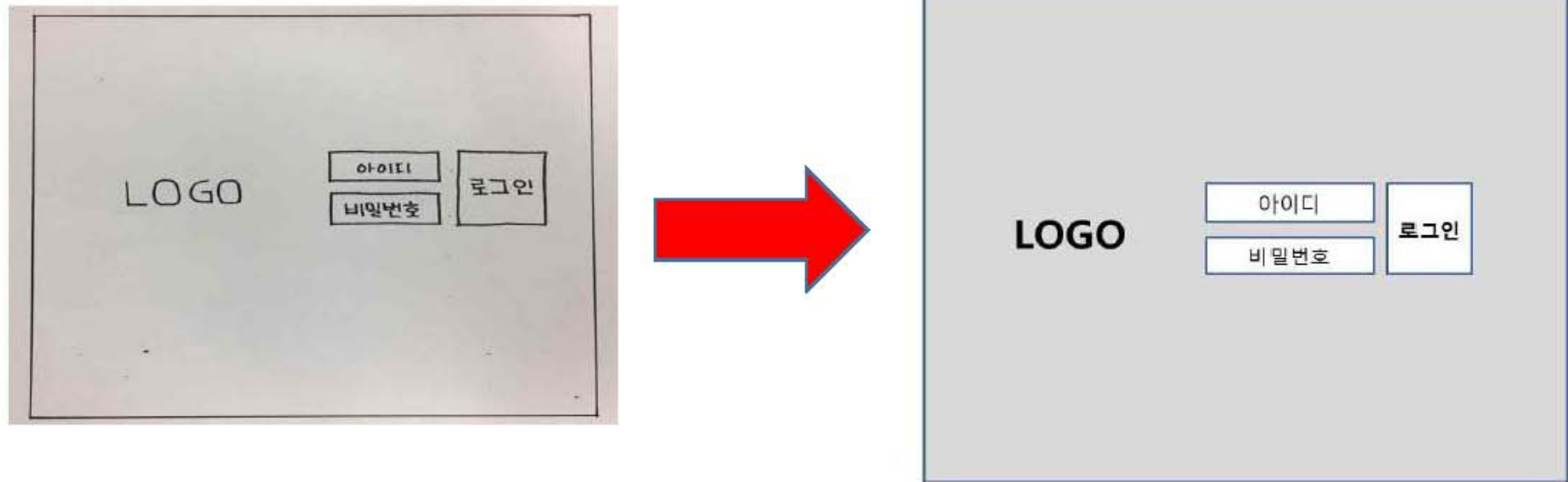
II. UI 설계 원리를 바탕으로 프로토타입 유스케이스(USE CASE)를 작성한다.



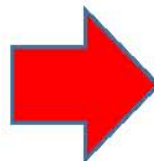
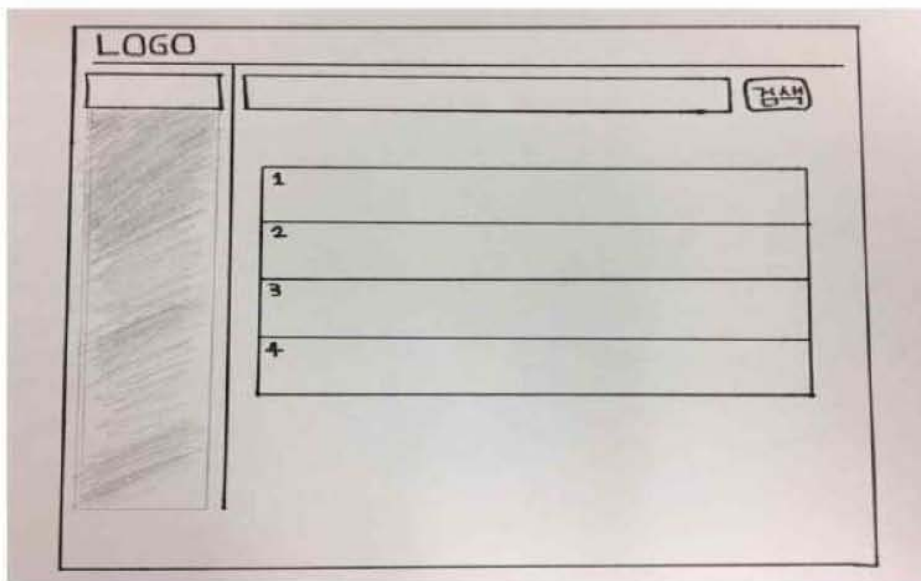
도서대출 예약시스템

소프트웨어 설계

III. UI 요구사항을 반영한 프로토타입을 제작한다.



소프트웨어 설계



The final software design of the interface. It features a header with the text "LOGO" and a "로그아웃" (Logout) button. Below the header is a search bar with a "검색" (Search) button. To the left of the search bar is a vertical sidebar. Below the search bar is a table with four rows, numbered 1 to 4. Each row has a text input field and a "선택" (Select) button. The table also includes a "모기 정렬방식" (Sort Method) dropdown menu.

번호	도서명/색인정보	대출가능	선택
1	도서명/색인정보	대출중	선택
2	도서명/색인정보	대출가능	선택
3	도서명/색인정보	대출가능	선택
4	도서명/색인정보	대출가능	선택

IV. 작성한 프로토타입을 활용하여 UI/UX 엔지니어와 향후 적용할 UI의 적정성에 대해 검토한다.

- UI 전체 구조에 대해 먼저 고려한 뒤 세분화된 페이지에 대해 고려하는 것이 효율적
- UI 요구사항을 만족시키기 위해서는 유스케이스를 이용

문제풀이

1. UI 설계 원칙 중 누구나 쉽게 이해하고 사용할 수 있어야 한다는 원칙은?

- ① 최소성
- ② 유연성
- ③ 직관성
- ④ 멀티운용성

(2020년 3회 정보처리기사 필기 기출문제 소프트웨어 설계)

2. UI 설계 원칙에서 누구나 쉽게 이해하고 사용할 수 있어야 한다는 것은?

- | | |
|-------|-------|
| ① 유효성 | ② 직관성 |
| ③ 무결성 | ④ 유연성 |

(2020년 1,2회 정보처리기사 필기 기출문제 소프트웨어 설계)

문제풀이

3. 파이프 필터 형태의 소프트웨어 아키텍처에 대한 설명으로 옳은 것은?

- ① 노드와 간선으로 구성된다.
- ② 서브시스템이 입력데이터를 받아 처리하고 결과를 다음 서브시스템으로 넘겨주는 과정을 반복한다.
- ③ 계층 모델이라고도 한다.
- ④ 3개의 서브시스템(모델, 뷰, 제어)으로 구성되어 있다.

(2020년 4회 정보처리기사 필기 기출문제 소프트웨어 설계)

소프트웨어 설계

문제풀이

4. 소프트웨어 아키텍처를 설계할 때 따라야 할 프로세스를 가장 적절하게 나열한 것은?

- 가. 서브시스템 사이의 인터페이스를 정의 하고 서브시스템 사이의 상호작용을 위한 동작을 작성한다.
- 나. 전체 시스템에 대한 설계 목표를 파악 하고 결정한다. 즉, 전체 시스템의 목표를 요구에서 발견하고 추출한다.
- 다. 설계 목표와 시스템의 타입을 고려하여 아키텍처 스타일을 선택한다.
- 라. 설계한 아키텍처가 요구, 설계 목표, 설계 원리를 만족시켰는지를 검토한다.
- 마. 적용할 수 있는 아키텍처 스타일이 있다면 이를 적용하여 시스템의 표준 아키텍처를 설계하고, 없다면 맞춤형 아키텍처를 설계한다.

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

(2015년 제16회 정보시스템관리사 필기 소프트웨어공학)

문제풀이

5. 소프트웨어 아키텍처 스타일과 응용분야가 가장 적절하지 않게 연결된 것은?

- ① 클라이언트/서버 아키텍처 - 웹 기반의 수강 신청 시스템
- ② N-tier 아키텍처 - 운영체제
- ③ 계층형 아키텍처 - 마이크로웨이브 오븐 제어 소프트웨어
- ④ MVC 아키텍처 - 모바일 애플리케이션

(2015년 제16회 정보시스템감리사 필기 소프트웨어공학)