강의노트

중대한 시스템 개발 및 특정분야 소프트웨어 개발 시 고려사항

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

▣ 학습내용

- 신뢰할 수 있는 프로그래밍
- 결함 내성
- 특정분야 시스템 개발 시 고려사항

▣ 학습목표

- 신뢰할 수 있는 소프트웨어를 만들기 위한 프로그래밍 기법을 설명할 수 있다.
- 어느 정도 결함을 이겨낼 수 있는 결함내성 기법을 설명할 수 있다.
- 특정 분야의 시스템 개발 시 고려할 사항을 설명할 수 있다.

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

1. 신뢰할 수 있는 프로세스

◈ 신뢰할 수 있는 프로세스 개념

- 소프트웨어가 설치되기 전 남아있는 결함을 찾고 보완하고 개선함으로써 만들어질 수 있음
- 이를 위하여 다음과 같은 부분을 수행

요구사항 검사	요구사항 관리	모델 점검	설계와 코드 검사	정적 분석	시험계획 수립과 관리

소프트웨어의 결함의 대부분이 요구사항오류에서 기인함으로
이러한 오류를 발견 제거한다면 시스템의 결함이 최소화 될 것임

요구사항 검사	**요구사항 관리**	모델 점검	설계와 코드 검사	정적 분석	시험계획 수립과 관리

요구사항의 변경에 대한 관리를 통하여 시스템의 신뢰도를 높임

요구사항 검사	요구사항 관리	**모델 점검**	설계와 코드 검사	정적 분석	시험계획 수립과 관리

시스템의 모델(아키텍처, 구성도 등)이
내부적인 일관성과 외부적인 일관성을 유지하는지 점검

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

1. 신뢰할 수 있는 프로세스(계속)

◈ 신뢰할 수 있는 프로세스 개념(계속)

▪ 이를 위하여 다음과 같은 부분을 수행

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

2. 신뢰할 수 있는 프로그래밍

◈ 신뢰할 수 있는 프로그래밍 정의

- 신뢰할 수 있는 프로그래밍은 결함을 예측하여 회피하거나 결함을 견딜 수 있도록 프로그래밍함
- 이를 위하여 다음의 프로그래밍작업을 수행

◈ 정보보호

- 프로그램 컴포넌트는 구현에 필요한 데이터만 접근을 허용하여야 함
- Java에서 interface사례, Class에서 public변수와 private변수의 구별은 이 사례에 해당

```
Class sample {
    Public  int a;
    Private int b;
}
```

```
main() {
    class sample ss;
    ss.a= 1; // OK
    ss.b=1; // Error
}
```

◈ 안전한 프로그래밍

- 안전한 프로그래밍을 위하여 다음 사항을 주의하여야 함

1 부동소수점 수

- 프로그램 계산에서 부동소수점 오류를 고려해야 함
- 3.00000000은 실제 2.99999999이나 3.00000001로 표현될 수 있음

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

2. 신뢰할 수 있는 프로그래밍(계속)

◈ 안전한 프로그래밍(계속)

▪ 안전한 프로그래밍을 위하여 다음 사항을 주의하여야 함(계속)

2 포인터

- 포인터를 잘못 사용하면 배열과 다른 구조체의 경계검사를 구현하기 어렵게 만들 수 있음

3 동적 메모리 할당

- 실행 시 할당되는 동적 메모리는 잘 다루어야 하며 사용 후 정확히 반환되지 않으면 예기치 않은 결과를 얻게 됨
- 동적 메모리 할당 alloc 후 free를 하지 않고 프로그램을 종료하는 경우

4 병행성

- 병행 프로세스는 각 프로세스의 진행결과의 연관성이 없는 프로세스이어야 함
- A프로세스의 결과가 B프로세스의 입력으로 사용되는 경우 A 실행 후 B를 실행하여야 하며, A와 B를 같이 실행한다면 정확한 결과를 얻을 수 없음

5 되부름

- 함수 등이 자신을 호출하는 경우 프로그램 오류를 찾기는 더욱 어려움
- 또한 스택 등 메모리 변수의 처리도 유의하여야 함
- 되부름 호출 시 탈출 로직이 반드시 존재해야 함

6 인터럽트

- 인터럽트는 현재 실행중인 코드와 관계없는 특정부분으로 제어를 강제로 이동시킴
- 특히 유의하여야 함

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

2. 신뢰할 수 있는 프로그래밍(계속)

◈ 안전한 프로그래밍(계속)

- 안전한 프로그래밍을 위하여 다음 사항을 주의하여야 함(계속)

7 상속

- 객체지향 언어에서 상속의 문제점은 관련된 코드가 한곳에 모여있지 않다는 것임
- 이런 경우 프로그램의 동작을 이해 하기 어렵고, 오류를 놓치기 쉬움
- 이 경우도 유의하여야 함

8 에일리어싱(Alias, 별명/별칭처리)

- 한 변수, 함수 등을 두 개의 이름으로 이용하고 이를 인지하지 않는다면, 수정과 프로그램 이해를 어렵게 할 수 있음

◈ 예외처리

- 프로그램 실행 중 오류나 예상치 못한 사건이 발생하는 것은 필연적임
- 이러한 오류나 예상치 못한 사건은 프로그램의 결함이나 예측할 수 없는 외부환경의 결과일 수 있음
- 예외(Exception): 프로그램의 실행 중에 발생한 오류나 예상치 못한 사건

예 시스템 전원고장, 존재하지 않는 데이터 접근시도,
수치연산에서의 오버플로우와 언더플로우

- 프로그램을 구현할 경우 이러한 예외처리를 추가하는 노력을 하여야 함

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩

◈ 시큐어 코딩 정의

- 개발하는 소프트웨어가 복잡해짐으로 인해 보안상 취약점이 발생할 수 있는 부분을 보완하여 프로그래밍하는 것
- 시큐어 코딩에는 안전한 소프트웨어를 개발하기 위해 지켜야 할 코딩 규칙과 소스 코드 취약 목록이 포함
- 미국은 2002년 연방정보보안관리법(FISMA)을 제정해 시큐어 코딩을 의무화했고, 마이크로소프트는 윈도 비스타(Windows Vista)를 개발할 때 시큐어 코딩을 도입
- 국내에서는 2012년 12월부터'소프트웨어 개발 보안' 제도를 시행하여 시큐어 코딩을 의무화

◈ 정부의 정보시스템 구축사업

- 20170206(고시2) 정보시스템 구축운영 지침 개정전문 등 여러 지침에서 프로그램 구현 시 47가지의 시큐어 코딩 점검 리스트를 제시함

입력데이터 검증 및 표현	프로그램 입력 값에 대한 검증 누락 또는 부적절한 검증, 데이터의 잘못된 형식지정으로 인해 발생할 수 있는 보안약점
보안기능	보안기능(인증, 접근제어, 기밀성, 암호화, 권한 관리 등)을 부적절하게 구현 시 발생할 수 있는 보안약점
시간 및 상태	동시 또는 거의 동시 수행을 지원하는 병렬 시스템, 하나 이상의 프로세스가 동작되는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안약점
에러처리	에러를 처리하지 않거나, 불충분하게 처리하여 에러정보에 중요정보(시스템 등)가 포함될 때 발생할 수 있는 보안약점

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩(계속)

◈ 정부의 정보시스템 구축사업

코드오류	타입변환 오류, 자원(메모리 등)의 부적절한 반환 등과 같이 개발자가 범할 수 있는 코딩오류로 인해 유발되는 보안약점
캡슐화	중요한 데이터 또는 기능성을 불충분하게 캡슐화 하였을 때, 인가되지 않은 사용자에게 데이터 누출이 가능해지는 보안약점
API 오용	의도된 사용에 반하는 방법으로 API를 사용하거나, 보안에 취약한 API를 사용하여 발생할 수 있는 보안약점

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트

◈ 입력데이터 검증 및 표현 점검 예시

번호	보안약점	설명	비 고
1	SQL 삽입	검증되지 않은 외부 입력 값이 SQL 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점	
2	경로 조작 및 자원 삽입	검증되지 않은 외부 입력 값이 시스템 자원 접근경로 또는 자원제어에 사용되어 공격자가 입력 값을 조작해 공격할 수 있는 보안약점	
3	크로스사이트 스크립트	검증되지 않은 외부 입력 값에 의해 사용자 브라우저에서 악의적인 스크립트가 실행될 수 있는 보안약점	
4	운영체제 명령어 삽입	검증되지 않은 외부 입력 값이 운영체제 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점	
5	위험한 형식 파일 업로드	파일의 확장자 등 파일형식에 대한 검증 없이 업로드를 허용하여 발생할 수 있는 보안약점	

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트(계속)

◈ 입력데이터 검증 및 표현 점검 예시(계속)

번호	보안약점	설명	비 고
6	신뢰되지 않는 URL주소로 자동접속 연결	검증되지 않은 외부 입력 값이 URL 링크 생성에 사용되어 악의적인 사이트로 자동 접속될 수 있는 보안약점	
7	XQuery 삽입	검증되지 않은 외부 입력 값이 XQuery 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점	
8	XPath 삽입	검증되지 않은 외부 입력 값이 XPath 쿼리문 생성에 사용 되어 악의적인 쿼리가 실행될 수 있는 보안약점	
9	LDAP 삽입	검증되지 않은 입력 값이 LDAP 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점	
10	크로스사이트 요청 위조	검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 스크립트가 실행되어 공격자가 원하는 요청(Request)이 다른 사용자(관리자 등)의 권한으로 서버로 전송되는	
11	HTTP 응답분할	검증되지 않은 외부 입력 값이 HTTP 응답헤더에 삽입되어 악의적인 코드가 실행될 수 있는 보안약점	
12	정수형 오버플로우	정수를 사용한 연산의 결과가 정수 값의 범위를 넘어서는 경우, 프로그램이 예기치 않게 동작될 수 있는 보안약점	
13	보안기능 결정에 사용 되는 부적절한 입력 값	검증되지 않은 입력 값이 보안결정(인증, 인가, 권한부여 등)에 사용되어 보안 메커니즘 우회 등을 야기할 수 있는 보안약점	
14	메모리 버퍼 오버플로우	메모리 버퍼의 경계 값을 넘어서 메모리 값을 읽거나 저장하여 예기치 않은 결과를 발생시킬 수 있는 보안약점	
15	포맷 스트링 삽입	printf 등 외부 입력 값으로 포맷스트링을 제어할 수 있는 함수를 사용하여 발생할 수 있는 보안약점	

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트(계속)

◈ 보안기능 점검 예시

번호	보안약점	설명	비 고
1	적절한 인증 없는 중요기능 허용	적절한 인증 없이 중요정보(금융정보, 개인정보, 인증정보등)를 열람(또는 변경)할 수 있게 하는 보안약점	
2	부적절한 인가	적절한 접근제어 없이 외부 입력 값을 포함한 문자열로 중요자원에 접근할 수 있는 보안약점	
3	중요한 자원에 대한 잘못된 권한 설정	중요자원(프로그램 설정, 민감한 사용자 데이터 등)에 대한 적절한 접근권한을 부여하지 않아, 인가되지 않은 사용자 등에 의해 중요정보가 노출·수정되는 보안약점	
4	취약한 암호화 알고리즘 사용	중요정보(금융정보, 개인정보, 인증정보 등)의 기밀성을 보장할 수 없는 취약한 암호화 알고리즘을 사용하여 정보가 노출될 수 있는 보안약점	
5	중요정보 평문 저장	중요정보(비밀번호, 개인정보 등)를 암호화하여 저장하지 않아 정보가 노출될 수 있는 보안약점	
6	중요정보 평문 전송	중요정보(비밀번호, 개인정보 등) 전송 시 암호화하지 않거나 안전한 통신채널을 이용하지 않아 정보가 노출될 수 있는 보안약점	
7	하드코드 된 비밀번호	소스코드 내에 비밀번호가 하드코딩되어 소스코드 유출 시 노출 우려 및 주기적 변경 등 수정(관리자 변경 등)이 용이하지 않는 보안약점	
8	충분하지 않은 키 길이 사용	데이터의 기밀성, 무결성 보장을 위해 사용되는 키의 길이가 충분하지 않아 기밀정보 누출, 무결성이 깨지는 보안약점	
9	적절하지 않은 난수 값 사용	예측 가능한 난수 사용으로 공격자로 하여금 다음 숫자 등을 예상하여 시스템 공격이 가능한 보안약점	
10	하드코드 된 암호화 키	소스코드 내에 암호화 키가 하드코딩되어 소스코드 유출 시 노출 우려 및 키 변경이 용이하지 않는 보안약점	

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트(계속)

◈ 보안기능 점검 예시(계속)

번호	보안약점	설명	비 고
11	취약한 비밀번호 허용	비밀번호 조합규칙(영문, 숫자, 특수문자 등) 미흡 및 길이가 충분하지 않아 노출될 수 있는 보안약점	
12	사용자 하드디스크저장되는 쿠키를 통한 정보 노출	쿠키(세션 ID, 사용자 권한정보 등 중요정보)를 사용자 하드디스크에 저장함으로써 개인정보 등 기밀정보가 노출될 수 있는 보안약점	
13	주석문안에 포함된시스템 주요정보	소스코드내의 주석문에 인증정보 등 시스템 주요정보가 포함되어 소스코드 유출 시 노출될 수 있는 보안약점	
14	솔트 없이 일방향 해쉬함수 사용	공격자가 솔트 없이 생성된 해쉬 값을 얻게 된 경우, 미리 계산된 레인보우 테이블을이용하여 원문을 찾을 수 있는 보안약점	
15	무결성 검사 없는 코드 다운로드	원격으로부터 소스코드 또는 실행파일을 무결성 검사 없이 다운로드 받고 이를 실행하는 경우, 공격자가 악의적인 코드를	
16	반복된 인증시도 제한 기능 부재	인증시도의 수를 제한하지 않아 공격자가 무작위 인증 시도를 통해 계정접근 권한을 얻을 수 있는 보안약점	

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트(계속)

◈ 시간 및 상태 및 에러처리 점검 예시

번호	보안약점	설명	비 고
1	경쟁조건: 검사 시점과 사용 시점(TOCTOU)	멀티 프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안약점	
2	종료되지 않는 반복문 또는 재귀 함수	종료조건 없는 제어문 사용으로 반복문 또는 재귀함수가 무한히 반복되어 발생할 수 있는 보안약점	
3	오류 메시지를 통한 정보 노출	개발자가 생성한 오류메시지에 시스템 내부구조 등이 포함되어 민감한 정보가 노출될 수 있는 보안약점	
4	오류 상황 대응 부재	시스템에서 발생하는 오류상황을 처리하지 않아 프로그램 실행정지 등 의도하지 않은 상황이 발생할 수 있는 보안약점	
5	부적절한 예외 처리	예외에 대한 부적절한 처리로 인해 의도하지 않은 상황이 발생될 수 있는 보안약점	

◈ 코드오류 점검 예시

번호	보안약점	설명	비 고
1	Null Pointer 역참조	Null로 설정된 변수의 주소 값을 참조했을 때 발생하는 보안약점	
2	부적절한 자원 해제	사용된 자원을 적절히 해제 하지 않으면 자원 누수 등이 발생하고, 자원이 부족하여 새로운 입력을 처리할 수 없게 되는 보안약점	
3	해제된 자원 사용	메모리 등 해제된 자원을 참조하여 예기치 않은 오류가 발생될 수 있는 보안약점	
4	초기화되지 않은 변수 사용	변수를 초기화하지 않고 사용하여 예기치 않은 오류가 발생될 수 있는 보안약점	

❖ 학습내용

[1] 신뢰할 수 있는 프로그래밍

3. 시큐어 코딩 사례 - 시큐어 코딩 점검 체크리스트(계속)

◈ 캡슐화 점검 예시

번호	보안약점	설명	비 고
1	잘못된 세션에 데이터 정보 노출	잘못된 세션에 의해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점	
2	제거되지 않고 남은 디버그 코드	디버깅을 위해 작성된 코드를 통해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점	
3	시스템 데이터 정보노출	사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점	
4	Public 메소드부터 반환된 Private 배열	Private로 선언된 배열을 Public으로 선언된 메소드를 통해 반환(Return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열이 수정될 수 있는 보안약점	
5	Private 배열에 Public 데이터 할당	Public으로 선언된 데이터 또는 메소드의 인자가 Private로 선언된 배열에 저장되면, Private 배열을 외부에서 접근할 수 있게 되는 보안약점	

◈ API오용 점검 예시

번호	보안약점	설명	비 고
1	DNS lookup에 의존한 보안결정	DNS는 공격자에 의해 DNS 스푸핑 공격 등이 가능함으로 보안결정을 DNS 이름에 의존할 경우, 보안결정 등이 노출되는 보안약점	
2	취약한 API 사용	취약하다고 알려진 함수를 사용함으로써 예기치 않은 보안위협에 노출될 수 있는 보안약점	

❖ 학습내용

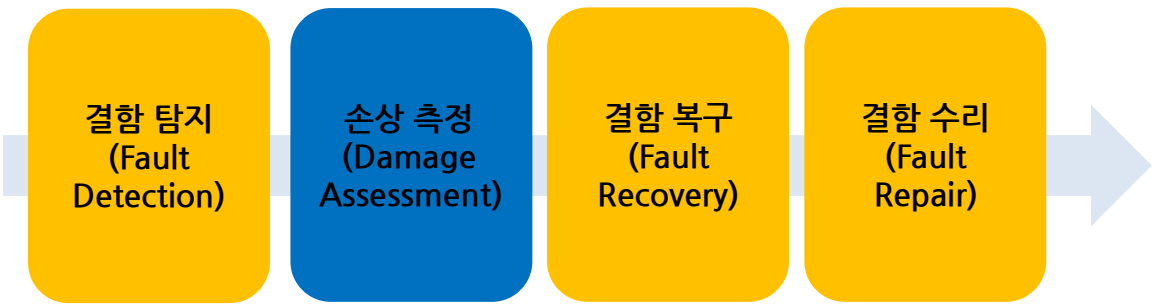
[2] 결함내성

1. 결함내성의 개념

◈ 결함내성

- 시스템이 운영되는 동안, 어느 정도의 결함이 발생하더라도 시스템은 이러한 결함과 상관없이 계속 운영 될 수 있도록 개발되어야 함
- 이러한 결함을 견디어 낼 수 있는 시스템 특성을 결함내성이라고 함
- 결함 내성의 단계

- 시스템은 시스템 장애를 유발할 수 있는 결함을 탐지해야 함
- 일반적으로 결함 탐지는 시스템 상태가 일관성이 있는지를 검사해야 함



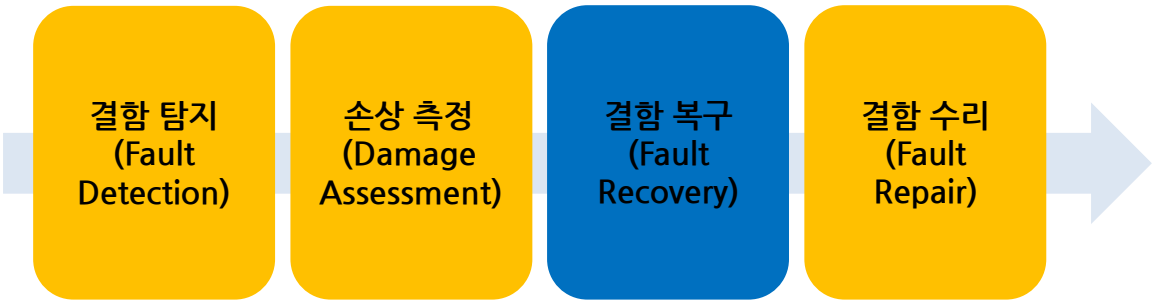
- 결함에 영향을 받은 시스템 상태의 부분을 탐지해야 함

❖ 학습내용

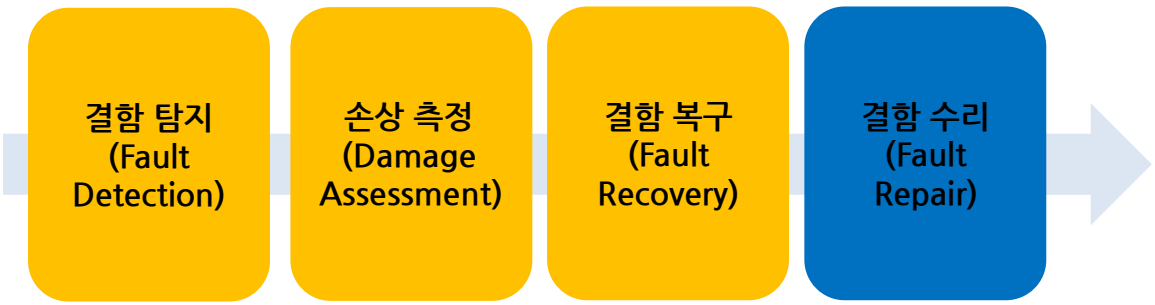
[2] 결함내성

1. 결함내성의 개념(계속)

◈ 결함 내성의 단계(계속)



- 시스템은 알려진 "안전"상태로 복구되어야 함
- 이것은 손상된 상태를 수정(전방 오류 복구)하거나, 알려진 "안전" 상태로 시스템을 회복(후방 오류 복구)함으로써 달성할 수 있음



- 결함 수리는 시스템을 수정해서 결함이 재발하지 않게 하는 것이나 많은 소프트웨어 결함은 일시적인 상태로 나타남
- 이것은 시스템입력의 특별한 조합에 기인
- 수리가 필요하지 않고 결함 복구 후 즉시 정상 처리가 재개될 수 있음

❖ 학습내용

[2] 결함내성

2. 결함내성 구현기술

◈ 2가지 결함내성 구현기술

- 결함내성을 가지기 위한 구현기술은 다음과 같음

방어적 프로그래밍 (Defensive Programming)	결함 허용 아키텍처 (Fault-tolerant Architecture)

◈ 방어적 프로그래밍(Defensive Programming)

- 소프트웨어적으로 결함탐지, 손상측정, 결함복구, 결함수리의 기능을 구현
- 시스템의 코드에 에러가 있다고 가정하여 상태를 검사하는 코드를 삽입
- 즉, 만약 에러가 감지될 때 이를 처리할 수 있는 코드를 미리 삽입해둠
- 하드웨어와 소프트웨어간의 상호작용에서 발생하는 오류를 처리하기 어렵고 구현도 쉽지 않음

◈ 결함 허용 아키텍처(Fault-tolerant Architecture)

- 결함내성을 지원하는 하드웨어 또는 소프트웨어 아키텍처를 사용

3. 장애허용 아키텍처

◈ 결함 허용 아키텍처(Fault-tolerant Architectures)

- 방어적 프로그래밍 기법은 하드웨어와 소프트웨어간의 상호작용에서 발생하는 오류를 처리하기 어렵기 때문에 결함 허용 아키텍처를 사용

하드웨어 결함 허용(Hardware Fault Tolerance)
소프트웨어 결함 허용(Software Fault Tolerance)

◈ 하드웨어 결함 허용(Hardware Fault Tolerance)

- 동일한 입력을 받는 여러 개의 동일한 컴포넌트를 중복 사용하여 결과를 비교
- 보통은 3개의 컴포넌트로 구성하며 이를 TMR(Triple-modular Redundancy) 라고 함
- 만약 하나의 결과가 다르다면, 그것은 무시되고 그 컴포넌트는 오류가 발생한 것으로 간주

❖ 학습내용

[2] 결함내성

3. 장애허용 아키텍처(계속)

◈ 하드웨어 결함 허용(Hardware Fault Tolerance)

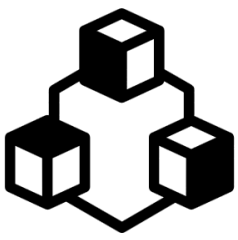
- 설계상의 오류라기 보다는 컴포넌트 자체의 오류를 허용하기 위한 것이며, 동시에 컴포넌트들이 오류 발생을 일으킬 확률은 아주 낮기 때문에 이러한 구조가 구현가능
- 결과 비교기(Output Comparator)가 존재하여 비교적 단순한 하드웨어 유닛으로 다수결 투표를 하는 것과 같은 원리로 처리됨
- 결과 비교기는 결함 관리자(Fault Manager)에 연결되어 처리됨

◈ 소프트웨어 결함 허용(Software Fault Tolerance)

- TMR은 다음을 기본적으로 가정

  - 하드웨어 컴포넌트는 공통적인 설계 결함을 가지지 않음
  - 컴포넌트는 임의적으로 오동작하며, 모든 컴포넌트가 동시에 오동작할 확률은 아주 낮음

- 하지만, 소프트웨어의 경우는 이러한 2가지 가정이 성립하지 않음
- 하드웨어의 경우는 동일한 설계의 컴포넌트라 할지라도 서로 다른 결과를 낼 수 있지만, 소프트웨어의 경우는 설계가 동일한 경우는 동일한 결과를 내게 되므로, TMR의 경우와 같은 비교는 의미가 없음
- 그러므로 어떤 하나의 업무를 여러 팀이 구현하여 결과를 비교하는 방식 등을 사용



N-version Programming

- 동일한 명세에 대해 서로 다른 팀들에 의해 다양한 구현이 이루어짐
- 각각의 버전들은 동시에 입력을 받아 계산을 하고 결과를 모아서 비교
- 각 팀들이 동일한 실수를 범할 확률은 낮지만, 경우에 따라 거의 동일한 알고리즘이 만들어지기도 함

❖ 학습내용

[3] 특정분야 시스템 개발 시 고려사항

1. 웹 시스템 개발

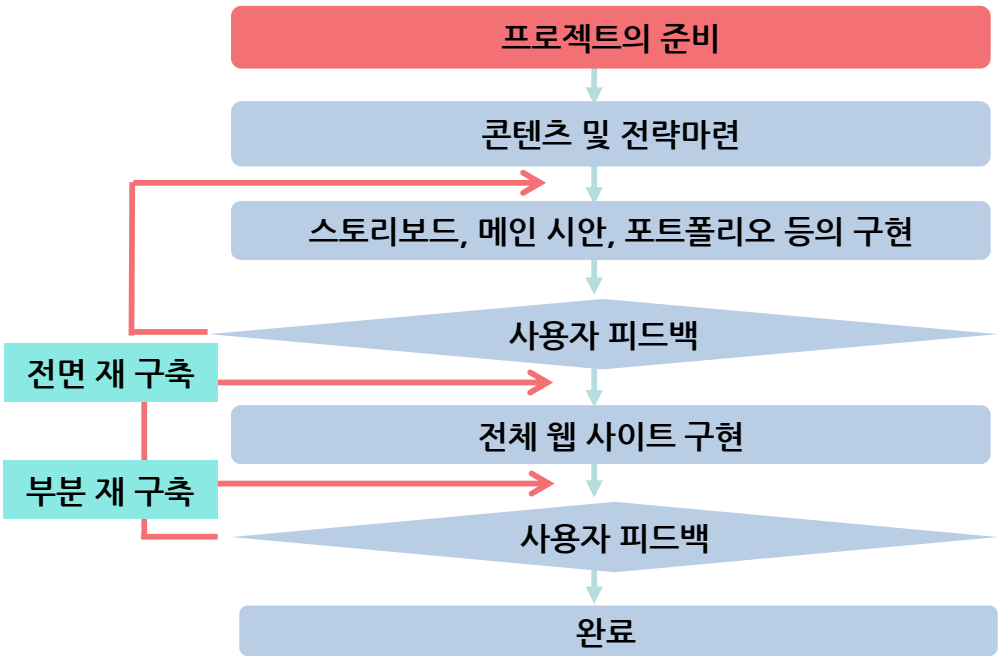
◈ 특정분야 시스템 개발 시 고려사항

- 요즘의 정보시스템을 개발하는 업무 중 웹 시스템, 게임 시스템, 모바일 시스템의 비중이 높음
- 이러한 시스템은 나름대로 특색 있는 부분을 고려하여 개발업무에 임해야 함

◈ 웹 시스템 개발

- 웹 시스템을 개발하기 위해서는 먼저 웹 시스템의 특성을 이해
- 웹 시스템은 UI중심의 개발이 이루어지기 때문에 UI개발을 위한 기술적인 특성을 고려
- 웹 시스템의 특성상 중간개발 단계에서 수시로 관계자의 승인(Conform)이 필요

◈ 웹 시스템 개발 절차



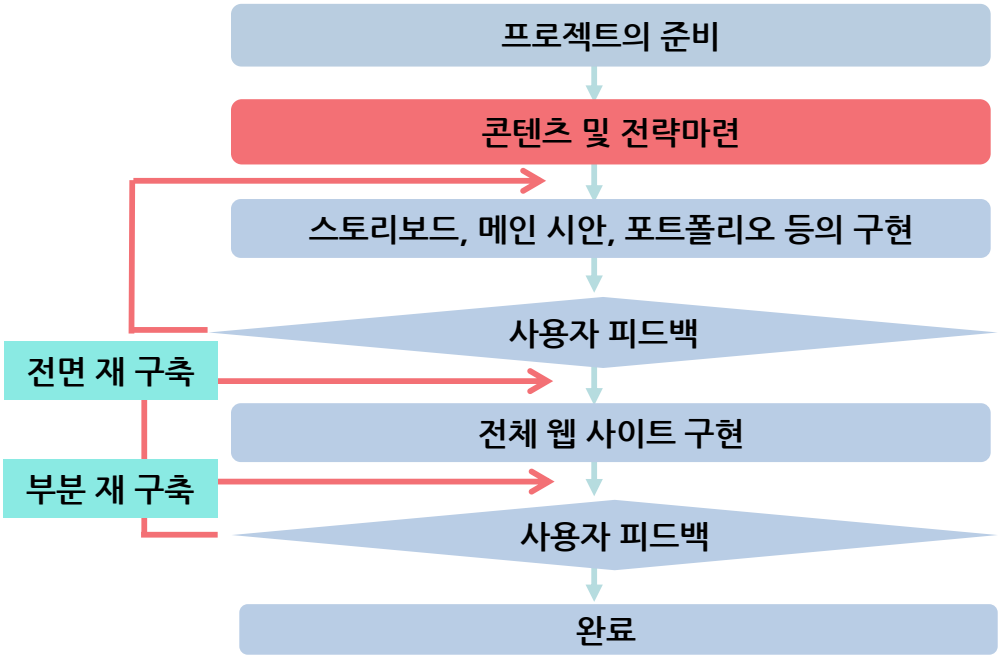
- 프로젝트의 목적 이해, 업무의 범위 산정
- 투입인력과 일정체크,각종 초기의 문서작업 진행

❖ 학습내용

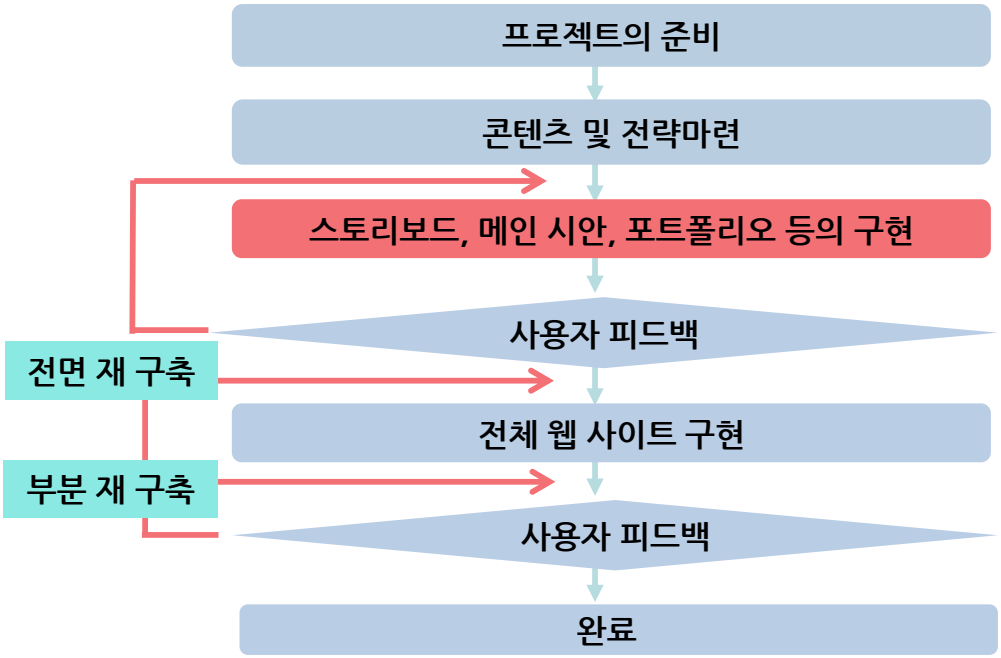
[3] 특정분야 시스템 개발 시 고려사항

1. 웹 시스템 개발(계속)

◈ 웹 시스템 개발 절차(계속)



- 현재의 트렌드와 자사,경쟁사,타사 분석 등을 통하여 현재 진행하려는 프로젝트의 목적성에 대한 기본 방향을 확정



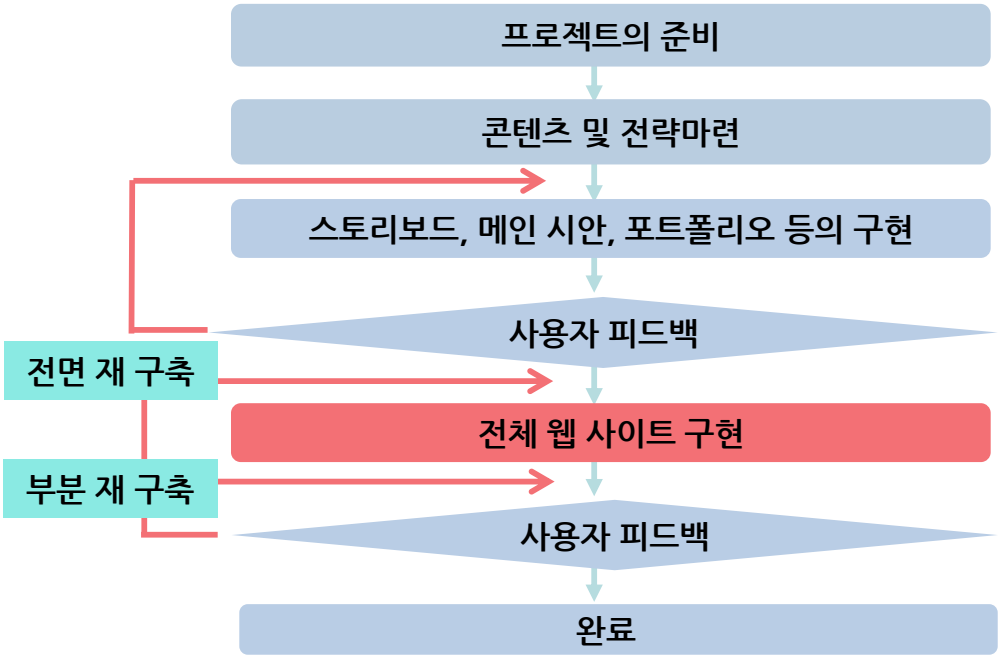
- 초기 시안 등은 사용자의 확답 후 원하는 방향대로 진행

❖ 학습내용

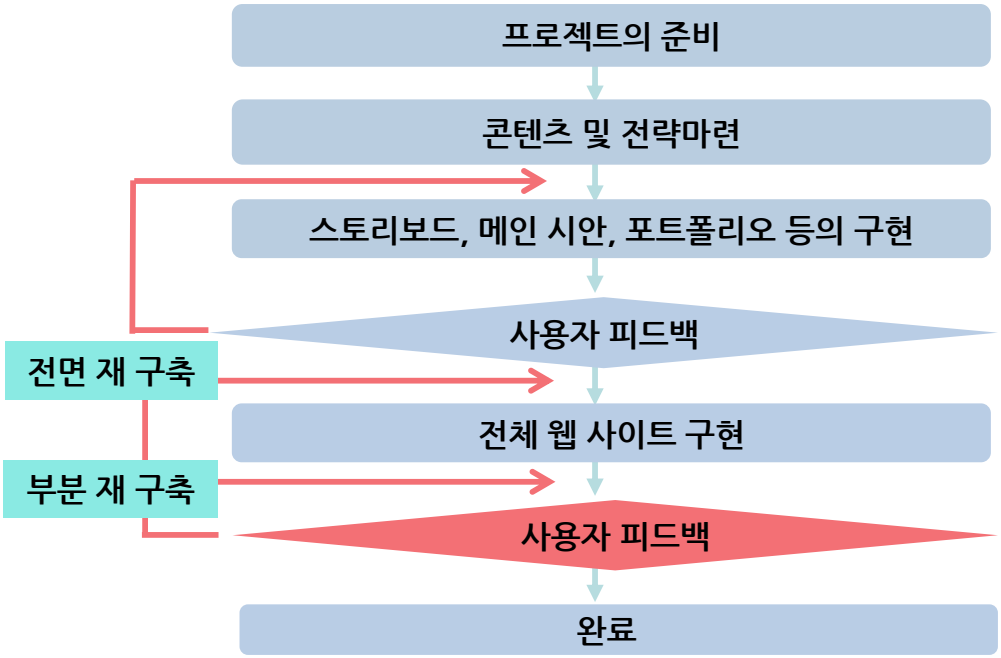
[3] 특정분야 시스템 개발 시 고려사항

1. 웹 시스템 개발(계속)

◈ 웹 시스템 개발 절차(계속)



- 일반적인 개발 프로젝트의 개발방법론과 유사한 부분



- 최종단계에서 사용자의 의도에 따라 프로젝트가 전면 재시작 되는 경우도 많음

❖ 학습내용

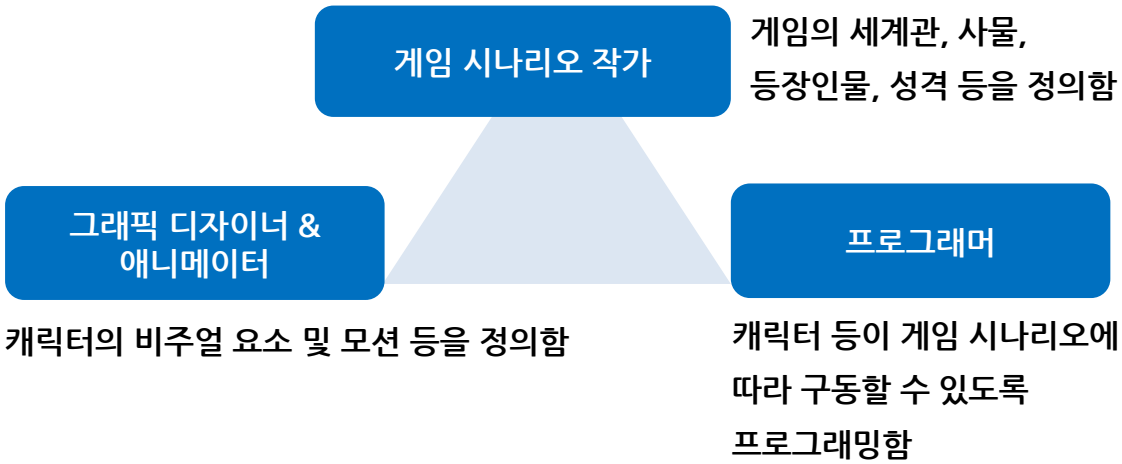
[3] 특정분야 시스템 개발 시 고려사항

2. 게임 시스템 개발

◈ 게임 시스템 개발 개요

- 게임 시스템의 특성을 이해
- 게임은 정적인 UI가 아닌 동적인 그래픽환경의 개발 • 모션 처리 부분 등 여러 부분을 고려
- 실제 게임을 만들기 위하여 게임이 어떻게 진행되는지 게임 시나리오를 먼저 구현

◈ 온라인 게임이 구동되기 위하여 여러 분야의 사람이 협업을 하며, 프로젝트의 성격상 프로젝트 관리자가 전체를 통합 관리하여야 함



3. 모바일 시스템 개발

◈ 모바일 시스템 개발 개요

- 모바일 시스템은 스마트폰, 이동기기 등 제약적인 컴퓨팅 환경에서의 프로그램을 개발
- 개발 프로그램은 각종 시스템자원, 화면 크기 등 많은 제약사항이 있으며 개발 시 이를 고려
- 시스템 제약성

  - CPU능력의 한계
  - 메모리 및 프로세스구동의 한계
  - 화면 크기 등의 제약 및 저전력 소모 고려
  - 다양한 기기(안드로이드, 아이폰, 제조사별 단말) 고려

- 시스템 특징의 고려

  - 이동성(Mobility): 뛰어난 휴대성과 이동 가능성을 가지고 있음
  - 위치확인(Location): 휴대한 단말의 위치를 인지할 수 있음
  - 개인화(Personalization): 단말은 개인의 용도로 사용함
  - 적시성(Timeliness): 모바일 프로그램의 구동시간에 대한 제한이 있음

❖ 학습내용

[3] 특정분야 시스템 개발 시 고려사항

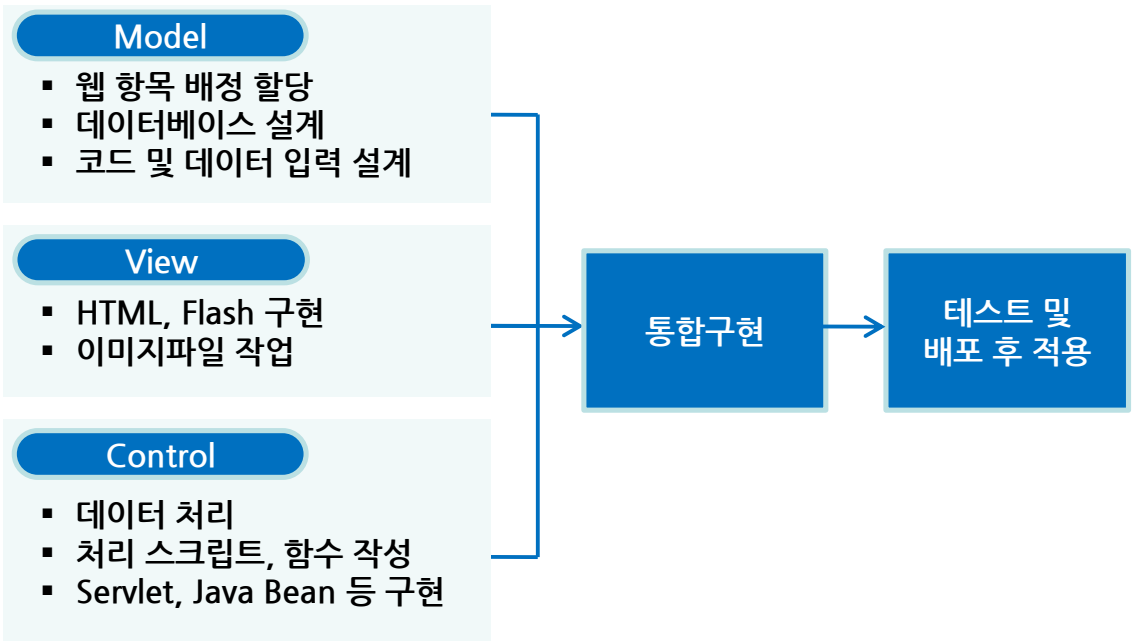
3. 모바일 시스템 개발(계속)

◈ MVC 모델

- 화면 UI가 중심인 웹, 모바일 등 단말채널 어플리케이션 개발을 위하여 일반적으로 MVC(Model-View-Controller)모델을 많이 고려함

1	어떤 웹 항목을 배정하고 할당하는 것에 따라 사용될 데이터베이스를 설계하고 코드 및 데이터를 입력하는 Model부분 개발
2	해당 화면을 HTML이나 플래시 등을 이용하여 화면을 구현하는 View부분 개발
3	데이터를 처리하여 View부분에 데이터를 공급하는 Control부분 개발
4	결국 Model, View, Control부분으로 나누어서 개발된 부분을 다시 통합하여 웹 어플리케이션 구현을 완료 한 후 테스트 및 배포

◈ MVC모델 개발 흐름도



❖ 핵심정리

1. 신뢰할 수 있는 프로그래밍

- 신뢰할 수 있는 프로그래밍은 결함을 예측하여 회피하거나 결함을 견딜 수 있도록 프로그래밍 하며, 이를 위하여 정보보호, 안전한 프로그래밍, 예외 처리 등의 프로그래밍작업을 수행
- 시큐어 코딩은 개발하는 소프트웨어가 복잡해짐으로 인해 보안상 취약점이 발생할 수 있는 부분을 보완하여 프로그래밍하는 것을 의미

2. 결함 내성

- 시스템이 운영되는 동안, 어느 정도의 결함이 발생하더라도 시스템은 이러한 결함과
  상관없이 계속 운영 될 수 있도록 개발되어야 함. 이러한 결함을 견디어 낼 수 있는 시스템 특성을 결함내성이라고 함
- 결함 내성에는 결함 탐지, 손상 측정, 결함 복구, 결함 수리의 단계를 거침

3. 특정분야 시스템 개발 시 고려사항

- 웹 시스템을 개발 한다면 UI중심의 개발이 이루어지기 때문에 UI개발을 위한 기술적인
  특성을 고려.
- 게임 시스템은 정적인 UI가 아닌 동적인 그래픽환경의 개발. 모션 처리 부분 등 여러 부분을 고려
- 모바일 시스템은 각종 시스템자원, 화면 크기 등 많은 제약사항이 있으며 개발 시 이를 고려