

학습 1 인터페이스 설계서 확인하기

학습 2 인터페이스 기능 구현하기

학습 3 인터페이스 구현 검증하기

1-1. 외부, 내부 모듈 간 공통 기능 및 데이터 인터페이스 확인

학습 목표

- 인터페이스 설계서를 기반으로 외부 및 내부 모듈 간 공통적으로 제공되는 기능과 각 데이터의 인터페이스를 확인할 수 있다.

필요 지식 /

① 인터페이스 설계서

인터페이스 설계서는 이 기종 시스템 또는 컴포넌트 간 데이터 교환 및 처리를 위한 목적으로 각 시스템의 교환 데이터 및 업무, 송수신 주체 등이 정의되어 있다. 일반적인 내용이 포함된 인터페이스 설계서(정의서) 외에도 다양한 다이어그램 및 데이터 포맷을 포함한 형태의 인터페이스 설계서가 있다.

1. 인터페이스 설계서(정의서)

시스템의 인터페이스 현황을 한눈에 확인하기 위하여 한 시스템이 갖는 인터페이스 목록 및 각 인터페이스의 상세 데이터 명세와 각 기능의 세부 인터페이스 정보를 정의한 문서이다.

제계정이력

날짜	버전	작성자	승인자	내용
18년 6월 1일	1.0	XXX	YYY	인터페이스 정의서 1.0 작성

인터페이스 정의서

시스템명	인사노무관리	서비스시스템명	전표작성
단계명	설계	작성일자	18년 6월 1일
		버전	1.0

1. 인터페이스 목록

송신				전달			수신				관련 요구사항 ID	비고
인터페이스 번호	일련번호	송신 시스템명	프로그램 ID	처리형태	인터페이스 방식	발생빈도	상대 담당자 확인	프로그램 ID	수신 시스템명	수신번호		
HR_INV_01	1	전표발생	P_XSW_001	ON-LINE	URL 호출	수시	YYY	P_ERP_001	매입 시스템	RCV-001	REQ-IF-004	

2. 인터페이스 명세

인터페이스 번호	데이터송신시스템					송신 프로그램 ID	데이터수신시스템					수신 프로그램 ID
	시스템명	데이터 저장소명	속성명	데이터타입	길이		데이터 저장소명	속성명	데이터 타입	길이	시스템명	
HR_INV_01	전표발생	급여결과	사번	CHAR	9	P_XSW_001	매입전표	거래자	CHAR	9	전표매입	P_ERP_001
HR_INV_01	전표발생	급여결과	NUMBER	NUMBER	10	P_XSW_001	매입전표	거래금액	NUMBER	10	전표매입	P_ERP_001

※ 실제 더 많은 인터페이스 속성이 필요하나 일부데이터를 샘플로 추출하여 보여줌

[그림 1-1] 시스템 인터페이스 설계서(정의서) 예시

(1) 시스템 인터페이스 정의서

한 시스템의 인터페이스 현황을 확인하기 위하여 시스템이 갖는 인터페이스 목록과 인터페이스 명세를 보여 주는 설계 문서이다.

(가) 인터페이스 목록

시스템에서 가지고 있는 인터페이스 목록을 보여 준다. 인터페이스 번호 및 인터페이스 되는 시스템의 정보 및 관련 요구 사항 ID(요구 사항 정의서의 요구 ID)를 리스트(목록) 형태로 보여 준다.

(나) 인터페이스 명세

인터페이스 목록에 있는 각 인터페이스의 상세 정보를 보여 준다. 각 인터페이스 번호당 인터페이스 되는 데이터, 데이터 형식, 송수신 시스템의 정보 등을 구체화한다.

(2) 상세 기능별 인터페이스 정의서

인터페이스를 통한 각 세부 기능의 개요, 세부 기능이 동작하기 전에 필요한 사전 조건, 사후 조건 및 인터페이스 파라미터(데이터), 호출 이후 결과를 확인하기 위한 반환값 등을 정의한 문서이다.

인터페이스 ID	HR_INV_01	인터페이스 명	급여전표발생 인터페이스
오퍼레이션 명	request_Generate_Invoice		
오퍼레이션 개요	정기급여를 회계 전표로 발행하는 프로세스 프로세스를 requestInvoice 로 호출하고 처리된 결과를 returnInvoiceDTO로 리턴한다.		
사전조건	정기급여 결과가 완료가 되어야 함 (calculatePayroll) 전표를 발행하는 회계시스템으로 이관하는 컴포넌트가 동작 중이어야 함		
사후조건	전표 인터페이스 후 전표발생 결과값을 return 해야 한다.		
파라미터	각 급여결과, 전표번호, 급여일자, 식발자		
반환값	전표발생결과, 전표발생금액(검증용)		

[그림 1-2] 각 상세 기능에 대한 인터페이스 정의

2. 정적·동적 모형, 데이터 포맷 형태에 따른 인터페이스 설계서

정적, 동적 모형으로 각 시스템의 구성 요소를 표현한 다이어그램을 통해 시스템, 컴포넌트별 인터페이스와 요구 조건을 확인할 수 있다.

(1) 정적, 동적 모형을 통한 인터페이스 설계서

시스템을 구성하는 주요 구성 요소 간 트랜잭션을 보여 주고, 이를 통해 시스템에서 인터페이스는 어디에 속하고 어떤 트랜잭션이 인터페이스를 통해 상호 교환되는지 확인할 수 있다.

1-2. 외부 및 내부 모듈 연계를 위한 인터페이스 기능 식별

학습 목표

- 개발하고자 하는 응용소프트웨어와 관련된 외부 및 내부 모듈 간의 연계가 필요한 인터페이스의 기능을 식별할 수 있다.

필요 지식 /

① 내부, 외부 모듈 연계 방법(EAI, ESB 연계 방법)

시스템 인터페이스를 위해 외부 및 내부 모듈을 연계하는 대표적인 방법은 EAI(Enterprise Application Integration) 방식과 ESB(Enterprise Service Bus) 방식이 있다.

1. EAI(Enterprise Application Integration)

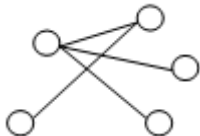
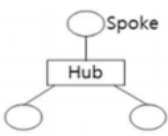
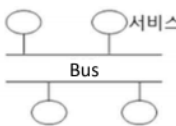
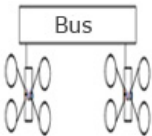
(1) EAI의 개념

기업에서 운영되는 서로 다른 플랫폼 및 애플리케이션들 간의 정보 전달, 연계, 통합을 가능하게 해 주는 솔루션이다. EAI를 사용함으로써 각 비즈니스 간 통합 및 연계성을 증대시켜 효율성을 높여 줄 수 있으며 각 시스템 간의 확정성을 높여 줄 수 있다.

(2) EAI의 구축 유형

EAI는 크게 Peer to Peer형, Hub & Spoke 방식, Bus 형태의 구성으로 분류될 수 있다.

<표 1-2> EAI의 구축 유형

유형	개념도	설명	특징
Point-to-Point		- 중간에 미들웨어를 두지 않고 각 애플리케이션 간 Point to Point 형태로 연결	- 솔루션 구매 없이 통합 - 상대적 저렴하게 통합 가능 - 변경, 재사용 어려움
Hub & Spoke		- 단일 접점이 허브 시스템을 통해 데이터를 전송하는 중앙 집중적 방식	- 모든 데이터 전송 보장 - 확장, 유지 보수 용이 - 허브 장애 시 전체 영향
Message Bus (ESB 방식)		- 애플리케이션 사이 미들웨어(버스)를 두어 처리 - 미들웨어 통한 통합	- 어댑터가 각 시스템과 버스를 두어 연결하므로 뛰어난 확정성, 대용량 처리 가능
Hybrid		- 그룹 내에는 Hub & Spoke 방식을 그룹 간 메시징 버스 방식을 사용	- 표준 통합 기술, 데이터 병목 현상 최소화

2-2. 인터페이스 구현

학습 목표

- 개발하고자 하는 응용소프트웨어와 연계 대상 모듈 간의 세부 설계서를 확인하여 공통적인 인터페이스를 구현할 수 있다.

필요 지식 /

① 인터페이스 구현을 위한 도구

여러 가지 방법으로 인터페이스는 구현될 수 있지만 대표적인 방법으로는 데이터 통신을 이용한 인터페이스 구현 방법과 인터페이스 테이블을 이용한 인터페이스 구현 방법으로 나눌 수 있다.

1. 데이터 통신을 통한 인터페이스 구현

애플리케이션 영역에서 인터페이스 형식에 맞춘 데이터 포맷을 인터페이스 대상으로 전송하고 이를 수신 측에서 파싱(Parsing)하여 해석하는 방식이다. 주로 JSON 및 XML 형식의 데이터 포맷을 사용하여 인터페이스를 구현한다.

(1) JSON(JavaScript Object Notation)

JSON은 속성-값 쌍(attribute-value pairs)으로 이루어진 데이터 오브젝트를 전달하기 위해 사용하는 개방형 표준 포맷이다. AJAX(Asynchronous JavaScript and XML)에서 많이 사용되고 XML을 대체하는 주요 데이터 포맷이다. 언어 독립형 데이터 포맷으로 다양한 프로그래밍 언어에서 사용되고 있다.

(가) JSON의 기본 자료형

JSON의 기본 자료형은 다음과 같이 사용된다.

<표 2-3> JSON 자료형 예시

구분	예시	설명
수(Number)	- 정수: 75, 1974, -114 - 실수(고정 소수점): 3.14, -2.712 - 실수(부동 소수점): 1e4, 2.5e12	기본 자료형, 8진수나 16진수 표현 방법은 지원하지 않음.
문자열(String)	"1234", "Lovd", "문자", "\", json\""	큰 따옴표로 묶어야 한다. \는 특수 기호 문자를 표현하기 위해 사용한다(\, 탭 등).
배열(Array)	[10, {"v": 20}, [30, "마흔"]]	배열은 대괄호([])로 나타냄. 배열의 각 요소는 기본 자료형이거나 배열, 객체임. 각 요소는 쉼표(,)로 구분됨.
객체(Object)	{"name2": 50, "name3": "값3", "name1": true}	객체는 이름 : 값 쌍의 집합으로 중괄호({ })를 사용함. 이름은 문자열이기 때문에 반드시 따옴표로 표현하며 값은 기본 자료형임.

(나) JSON의 사용 예시

다음은 JSON을 통하여 데이터 전송 포맷을 만든 예제이다.

<표 2-4> JSON을 활용한 데이터 전송 포맷 예시

예시
<pre>{ "이름": "홍길동", "나이": 38, "성별": "남 ", "주소": "경기도 용인시 기흥구 중동", "특기": ["프로그래밍", "영어"], "가족관계": {"#": 3, "아내": "전지현", "딸": "김고은"}, "회사": "서울시 강남구 논현동" }</pre>

(2) XML(eXtensible Markup Language)

다른 특수한 목적을 갖는 마크업 언어를 만드는 데 사용하도록 권장하는 다목적 마크업 언어이다. 다른 많은 종류의 데이터를 기술하는 데 사용될 수 있으며, 다른 종류의 시스템끼리 데이터를 쉽게 주고받을 수 있게 하는 목적이 있다.

(가) XML의 주요 특징

XML 언어가 가지는 주요 특징은 다음과 같다.

<표 2-5> XML의 주요 특징

구분	예시
유니코드 문자	정의상 XML 문서는 문자로 이루어져 있다. 거의 모든 올바른 유니코드 문자는 XML 문서에 나타날 수 있다.
XML 파서(Parser)	파서는 마크업을 분석하고 필요한 정보를 추출하여 애플리케이션에 넘긴다.
마크업(Mark up)과 내용(Content)	XML 문서를 구성하는 문자들은 마크업과 내용으로 구분되며 간단한 문법 규칙으로 이루어진다. 마크업으로 구성되는 문자열은 '<'로 시작하여 '>'로 끝나거나 '&'로 시작하여 문자 ';'로 끝나며 마크업이 아닌 문자열은 내용이다. * 태그(Tag): '<'로 시작하여 '>'로 끝나는 마크업 구조
엘리먼트(element)	문서의 논리 요소로서 시작 태그로 시작하여 짝이 되는 끝 태그로 끝나거나 빈 엘리먼트 태그만으로 이루어진다. 자식 엘리먼트를 포함할 수 있음. * 예시: <Greeting><child>Hello world</child></Greeting>

2-4. 인터페이스 보안 기능 적용

학습 목표

- 응용소프트웨어와 관련된 내외부 모듈 간의 연계 데이터의 중요성을 고려하여 인터페이스 보안 기능을 적용할 수 있다.

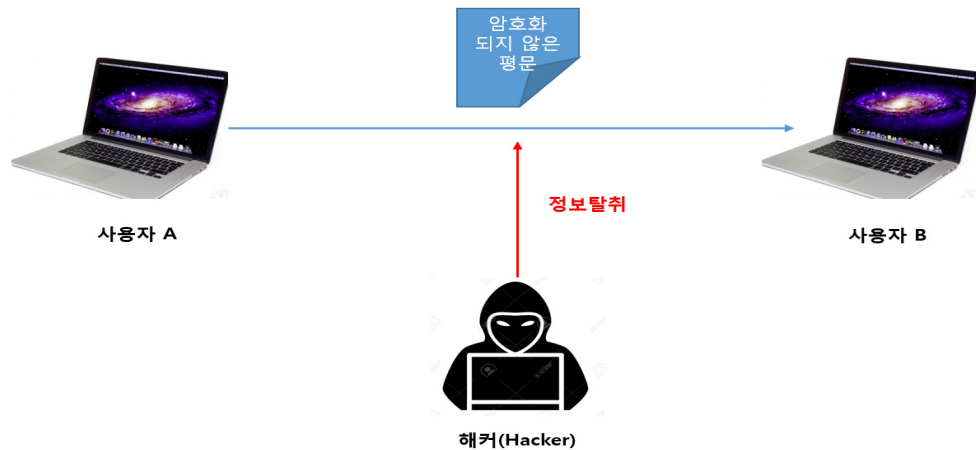
필요 지식 /

① 구현된 인터페이스의 주요 보안 취약점

인터페이스는 시스템 모듈 간 통신 및 정보 교환을 지원하므로 데이터 변조·탈취 및 인터페이스 모듈 자체의 보안 취약점이 있을 수 있다.

1. 데이터 통신 시 데이터 탈취 위험

데이터 통신 내역을 중간에서 감청하여 기밀성을 훼손할 수 있는 기법으로 스니핑(Sniffing)이라고 불리기도 한다. 스푸핑(Spoofing)처럼 공격 대상을 직접적으로 속이는 해킹이 아닌 수동적(Passive) 해킹 공격 기법이며, 도청할 수 있도록 중간에 설치되는 도구를 스니퍼(Sniffer)라고 한다. 주로 패킷 분석기 같은 툴을 통해서 진행된다.



[그림 2-6] 스니핑 공격 개념도

2. 시큐어 코딩

대표적인 웹 애플리케이션의 보안 취약점 발표 사례인 OWASP(Open Web Application Security Project) Top 10을 참고하여 KISA(한국 인터넷 진흥원)에서 SW 보안 약점 가이드를 발표하였고 SW 보안 취약점, 약점 및 대응 방안이 구체적으로 서술되어 있다.

<표 2-18> 보안 취약점에 대한 시큐어 코딩 항목

구분	내 용
입력 데이터 검증 및 표현	XSS, SQL 인젝션을 방지하기 위해 소스코드 취약점 점검
API 이용	gets(), system.exit() 등 시스템 접근 API 오용
보안 특성	인증, 접근 제어, 기밀성, 암호화, 권한 관리, 취약한 알고리즘, 부적절 인가로 인한 취약점
시간 및 상태	프로세스 동시 수행 시, system call 등을 동시 수행 시 잘못된 권한 위임 가능성
에러 처리	에러 처리가 부적절하거나 에러에 정보가 과도하게 많이 포함된 경우
코드 품질	복잡한 소스코드가 가독성과 유지 보수성을 저하함.
캡슐화	중요 데이터의 불충분한 캡슐화로 악의적 접근 가능

3. 데이터베이스 암호화

데이터베이스의 기밀성을 유지하기 위해 중요 민감 데이터는 암호화하는 기법을 사용한다. 다양한 암호 알고리즘을 활용하여 중요 데이터는 암호화한다.

(1) 데이터베이스 암호화 알고리즘

데이터베이스 암호화 알고리즘은 크게 대칭 키, 해시, 비대칭 키 알고리즘이 사용된다.

<표 2-19> 주요 데이터베이스 암호화 알고리즘

구분	내 용
대칭 키 암호 알고리즘	ARIA 128/192/256, SEED
해시 알고리즘	SHA -256/384/512, HAS-160
비대칭 키 알고리즘	RSA, ECDSA

(2) 데이터베이스 암호화 기법

데이터베이스 암호화 기법으로는 애플리케이션에서 암호화를 수행하는 API 방식과 데이터베이스에서 암호화를 수행하는 Plug-in 방식, API 방식과 Plug-in 방식을 혼합한 Hybrid 방식이 있다.

학습 1	인터페이스 설계서 확인하기
학습 2	인터페이스 기능 구현하기
학습 3	인터페이스 구현 검증하기

3-1. 인터페이스 구현 검증

학습 목표

- 구현된 인터페이스 명세서를 참조하여 구현 검증에 필요한 감시 및 도구를 준비할 수 있다.
- 인터페이스 구현 검증을 위하여 외부 시스템과의 연계 모듈 상태를 확인할 수 있다.

필요 지식 /

① 인터페이스 구현 검증도구, 감시 도구

구현된 인터페이스의 동작을 검증하기 위해 인터페이스 구현 및 감시 도구를 통해서 동작 상태를 검증 및 감시(monitoring)할 수 있다.

1. 인터페이스 구현 검증도구

인터페이스 구현을 검증하기 위해서는 인터페이스 단위 기능 및 시나리오에 기반한 통합 테스트가 필요하다. 테스트 자동화 도구를 이용하여 단위 및 통합 테스트의 효율성을 높일 수 있다.

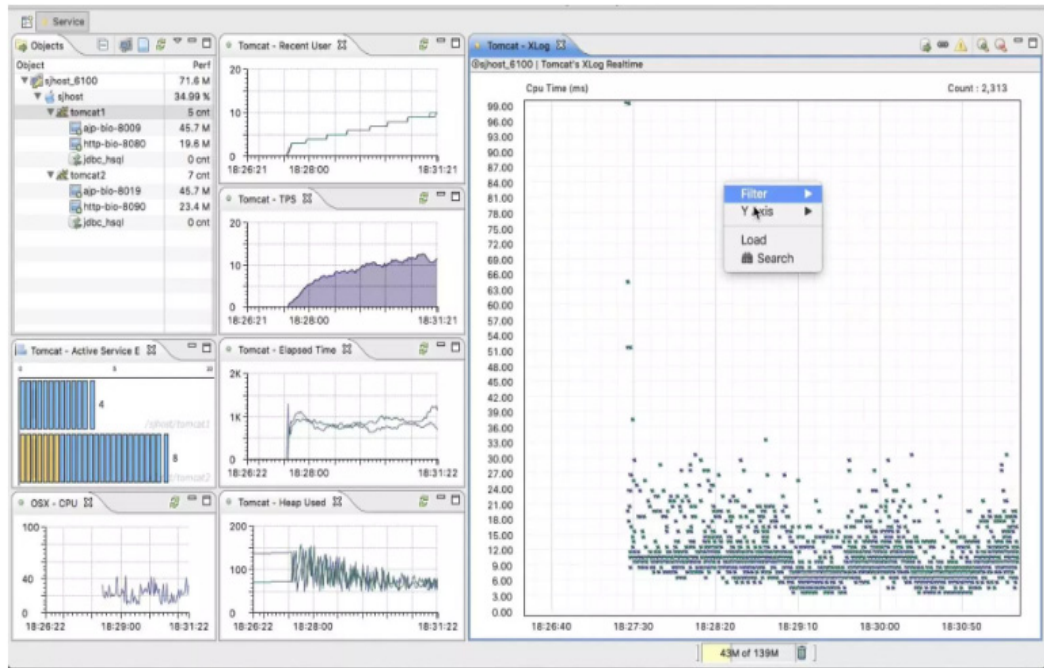
<표 3-1> 인터페이스 구현 검증도구

도구	설명
xUnit	java(Junit), C++(Cppunit), .Net(Nunit) 등 다양한 언어를 지원하는 단위 테스트 프레임워크
STAF	서비스 호출, 컴포넌트 재사용 등 다양한 환경을 지원하는 테스트 프레임워크
FitNesse	웹 기반 테스트 케이스 설계/실행/결과 확인 등을 지원하는 테스트 프레임워크
NTAF	Naver 테스트 자동화 프레임워크이며, STAF와 FitNesse를 통합
Selenium	다양한 브라우저 지원 및 개발언어를 지원하는 웹 애플리케이션 테스트 프레임워크
watir	Ruby 기반 웹 애플리케이션 테스트 프레임워크

출처: https://www.oss.kr/info_test/show/b3f50bf5-7d67-486f-bc70-d426d6f01dc4 테스트 자동화 프레임워크

2. 인터페이스 감시 도구

인터페이스의 동작이 잘 진행되는지 확인하기 위해서는 애플리케이션 모니터링 툴(APM: Application Performance Management)을 사용하여 동작 상태를 감시할 수 있다. 상용 제품 및 오픈소스를 이용한 애플리케이션 모니터링 툴이 있다. 데이터베이스, 웹 애플리케이션의 트랜잭션과 변수값, 호출 함수, 로그 및 시스템 부하 등 종합적인 정보를 조회하고 분석할 수 있다.



[그림 3-1] 오픈소스 감시 도구 예시(스카우터)

3-2. 인터페이스 오류 처리 확인 및 보고서 작성

학습 목표 • 인터페이스 오류 처리 사항을 확인하고 보고서를 작성할 수 있다.

필요 지식 /

① 인터페이스 오류 처리 방법 및 오류 처리 보고서

인터페이스는 이기종 시스템이기 때문에 오류 처리 시 사용자, 관리자에게 오류 처리 상태를 보여 주는 방법이 다소 복잡하고 관리하기 불편할 수 있다. 인터페이스 오류는 중요한 오류(장애)일 경우가 많으므로 오류 발생 시 오류 처리 보고서를 작성하여 관리 조직에 보고하여야 한다.

1. 인터페이스 오류 처리 방법

인터페이스 오류 처리 방법은 크게 사용자 화면에서 오류를 인지하게 구현하는 방법, 인터페이스 오류 시스템 로그를 별도로 작성하여 파일로 보관하는 방법, 별도 데이터베이스에 인터페이스 관련 오류 사항을 기록하는 방법이 있다.

(1) 사용자 화면에서 오류를 발생

사용자 화면에서 인터페이스 오류를 인지하는 방법은 가장 직관적으로 오류를 인지할 수 있어 가장 많이 쓰이는 방법이다. 인터페이스 오류가 발생하였을 경우 알람 형태로 화면에 표시되며, 주로 즉시적으로 데이터가 인터페이스되는 경우에 사용된다.

(2) 인터페이스 오류 로그 생성

시스템 운영 로그에 인터페이스 오류 시 관련 에러 로그가 생성되도록 할 수 있다. 인터페이스 오류의 자세한 내역을 알기 위해 사용되며, 시스템 관리자나 운영자가 오류 로그를 확인할 수 있다.

<표 3-4> 인터페이스 오류 로그 생성 예시

예시

```
[2018-07-27 10:01:070001][ERRORCODE100] 인사발령번호 = 2018-444  
(EMP_NOTC_NM) = xxxxxxxx 인사발령 구분  
Length Exceed Exception : 발령내역 길이가 초과하였습니다.
```

(3) 인터페이스 관련 테이블에 오류 사항 기록

테이블을 통한 인터페이스 기능을 구현할 경우나 인터페이스 트랜잭션 기록을 별도로 보관하는 경우 테이블에 오류 사항을 기록할 수 있다. 이력을 직관적으로 보기 쉬워 운영자가 관리하기 용이한 장점이 있다.

송신일시	변경구분	발령번호	사번	발령내용	처리일시	처리상태	오류코드	오류내용
18.7.27	입력	2018-111	18-001,18-002	신규채용	18.7.27	실패	E-003	수신 데이터베이스 연결실패

[그림 3-2] 인터페이스 관련 테이블에 오류 사항 기록 예시

2. 인터페이스 오류 처리 보고서

인터페이스에서 오류가 발생 시 관련 사항을 조직에서 정의된 보고 라인으로 인터페이스 오류 처리 보고서를 작성하여 즉각적으로 보고하여야 한다.

(1) 인터페이스 오류 처리 보고서 형식

정형화된 형식은 없으며 조직 및 상황에 맞는 보고서를 작성하여 활용한다.

장애(발생/진행/완료)보고서

2018.7.27 담당: ... 대리

장애처리 (발생/진행/완료)보고서		보고서 번호	
장애 발생 일시	2018년 7월 27일 09:00	장애환경	XX 시스템
장애 조치 일시	2018년 7월 27일 09:00 ~	종료여부	처리완료
장애 종료 일시	2018년 7월 27일 13:00	장애등급	2등급 ①영향도: B ②중요도: High
장애 내용 및 증상 - 인터페이스 오류			
장애 원인 - 시스템 결함, 네트워크 장애			
조치 사항 - 조치경과 기록			
재발방지 계획 및 의견 - 향후 재발방지 방안			

[그림 3-3] 장애 보고서(인터페이스 오류 보고서) 양식 예시