

Session061 모듈 연계를 위한 인터페이스 기능 식별

1 모듈 연계의 개요

1.1 내부 모듈과 외부 모듈 또는 내부 모듈 간 데이터의 교환을 관계를 설정, 방법에는 EAI, ESB가 있다

1.2 EAI(Enterprise Application Integration)

- 기업 내 애플리케이션 및 플랫폼 간의 정보 전달, 연계, 통합 등 상호 연동이 가능하게
- 비즈니스 간 통합 및 연계성 증대, 효율성 및 확정성(Determinacy) 를 높임
- 구축 유형
 - Point to Point : 1:1 연결, 변경 및 재사용이 어렵다
 - Hub & Spoke : 허브 시스템을 통해 데이터 전송, 중앙 집중형, 확장 및 유지 보수 용이, 허브 장애 시 전체 영향
 - Message Bus(ESB 방식) : 애플리케이션 사이 버스를 두어 처리, 확장성 뛰어남, 대용량 처리 가능
 - Hybrid : Hub & Spoke와 Message Bus 혼합, 그룹 내 Hub & Spoke, 그룹 간 Message Bus, 데이터 병목 현상 최소화

1.3 ESB(Enterprise Service Bus)

- 애플리케이션 간 연계, 데이터 변환, 웹 서비스 지원 등 표준 기반의 인터페이스 지원 솔루션
- 서비스 중심의 통합 지향
- 범용적으로 사용, 결합도(Coupling) 약하게(Loosely) 유지
- 관리 및 보안 유지 쉽다. 높은 수준 품질 지원 가능

2 모듈 간 연계 기능 식별

2.1 모듈 간 공통 기능 및 데이터 인터페이스를 기반으로 연계된 기능을 시나리오 형태로 구체화하여 식별

2.2 연계 기능은 인터페이스 기능 식별에 사용

3 모듈 간 인터페이스 기능 식별

3.1 모듈 간 관련 기능 검토, 인터페이스 동작에 필요한 기능을 식별

3.2 인터페이스 동작은 외부 모듈 결과 또는 요청에 의해 수행, 외부 및 인터페이스 모듈 간 동작 기능을 통해 인터페이스 기능 식별

3.3 내부 모듈 동작은 외부 모듈에서 호출 된 인터페이스에 의해 수행됨. 시나리오를 통해 내부 모듈과 관련된 인터페이스 기능을 식별

3.4 식별된 인터페이스 기능 중 실제로 필요한 인터페이스 최종 선정

3.5 인터페이스 기능 구현 정의에 사용

Session063 인터페이스 기능 구현 정의

1 인터페이스 기능 구현의 정의에 대한 개요

1.1 기능에 대한 구현 방법 기능별로 기술한 것

1.1.1 순서

- 컴포넌트 명세서 확인
- 인터페이스 명세서
- 일관된 인터페이스 기능 구현 정의
- 정의된 인터페이스 기능 구현 정형화

2 모듈 세부 설계서

2.1 모듈의 구성 요소와 세부적인 동작 등을 정의

2.2 컴포넌트 명세서

- 컴포넌트의 개요 및 내부 클래스의 동작, 인터페이스를 통해 외부와 통신하는 명세 등을 정의

■ 컴포넌트 ID

■ 컴포넌트명

■ 컴포넌트 개요

■ 내부클래스

◆ ID

◆ 클래스명

◆ 설명

■ 인터페이스 클래스

◆ ID

◆ 인터페이스명

◆ 오퍼레이션명

◆ 구분

2.3 인터페이스 명세서

- 컴포넌트 명세서의 항목 중 인터페이스 클래스의 세부 조건 및 기능 등 정의

■ 인터페이스 ID

■ 인터페이스명

■ 오퍼레이션명

■ 오퍼레이션 개요

■ 사전조건

■ 사후조건

■ 파라미터

■ 반환값

3 모듈 세부 설계서 확인

- 3.1 각 모듈의 컴포넌트 명세서와 인터페이스 명세서를 기반으로 인터페이스에 필요한 기능 확인
- 3.2 컴포넌트 명세서의 컴포넌트 주요 기능 확인
- 3.3 컴포넌트 명세서의 인터페이스 주요 기능 확인
- 3.4 인터페이스 명세서의 인터페이스의 세부 조건 및 기능 확인

4 인터페이스 기능 구현 정의

- 4.1 인터페이스 기능, 데이터 표준, 모듈 세부 설계서 기반으로 일관성 있고 정형화된 인터페이스 기능 구현에 대해 정의
- 4.2 일관성 있는 인터페이스 기능 구현 정의
 - 기능, 데이터 표준, 모듈 세부 설계서를 통해 기능 구현 정의
 - 송 수신 측에서 해야 할 절차까지 다시 세부적 정의
- 4.3 정의된 인터페이스 기능 구현 정형화
 - 소프트웨어에 의존적이지 않게 보기 쉽고 표준화되도록 정형화
 - 가독성 높이기 위해 유스케이스 다이어그램 형태로 정형화

Session065 인터페이스 예외 처리

1 인터페이스 예외 처리의 개요

- 1.1 기능상 예외 상황 발생 시 처리하는 절차
- 1.2 데이터 통신 이용
- 1.3 인터페이스 엔티티 이용

2 데이터 통신을 이용한 인터페이스 예외 처리

- 2.1 인터페이스 객체(JSON, XML 등)를 이용해 구현한 동작이 실패할 경우 대비한 것
- 2.2 객체 송수신시 발생 가능한 예외 케이스 정의, 예외 처리 방법 기술
- 2.3 시스템 환경, 송 수신 데이터, 프로그램 자체 원인

예1 인터페이스 객체 송신 실패 시 예외 처리 방안

구분	예외 상황	예외 처리 방안
시스템 환경	네트워크 불안정	<ul style="list-style-type: none"> POST 이후 오류 메시지를 확인한다. 서버를 찾지 못하는 404 오류일 경우 네트워크 또는 서버 상태를 확인한다.
송신 데이터	송신 데이터 크기, 데이터 정합성* 체크 오류 발생	<ul style="list-style-type: none"> 송신 데이터의 원활한 전송을 위해 사전에 데이터를 정제한다. 데이터 송신 시 데이터 크기 및 정합성을 체크하는 기능을 추가하여 미리 예방한다.
프로그램 자체 원인	송신 데이터 생성 시 프로세스의 논리적 결함	<ul style="list-style-type: none"> 논리적 결함을 수정한다. 충분한 테스트를 통해 사전 예방한다. 프로세스에 따라 예상되는 예외를 사용자에게 알람을 통해 알려 준다.

예2 인터페이스 객체 수신 실패 시 예외 처리 방안

구분	예외 상황	예외 처리 방안
시스템 환경	네트워크 및 서버 불안정	입력 대기 큐에 요청을 적재한 후 순차적으로 처리하여 서버가 정상적으로 가동될 때 동작할 수 있도록 한다.
수신 데이터	특수문자 등으로 파싱 시 오류 발생	특수문자 입력 케이스를 미리 파악한 다음 파싱 시 오류가 발생하지 않는 문자로 우선 대처하고 이후에 다시 처리한다.
프로그램 자체 원인	수신 데이터 처리 시 프로그램의 논리적 결함	<ul style="list-style-type: none"> 논리적 결함을 수정한다. 충분한 테스트를 통해 사전 예방한다. 프로세스에 따라 예상되는 예외를 사용자에게 알람을 통해 알려 준다. 예외사항이 수신되지 않도록 송신 측 프로그램을 수정한다.

3 인터페이스 엔티티를 이용한 인터페이스 예외 처리

3.1 이인터페이스 동작이 실패할 경우 대비

3.2 해당 엔티티에 실패 상황과 원인 등 기록, 사용자 및 관리자에게 알려주는 방식

예1 송신 인터페이스 테이블을 이용한 인터페이스 기능 실패 시 예외 처리 방안

프로세스	예외 상황	예외 처리 방안
인터페이스 데이터 생성	<ul style="list-style-type: none"> 선택 SQL, 프로그램 오류 데이터 객체 생성 오류 	<ul style="list-style-type: none"> 오류 발생 시 사용자에게 알람을 통해 알려준다. 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.
인터페이스 테이블에 입력	<ul style="list-style-type: none"> 입력 SQL 오류 데이터 정합성 오류 	<ul style="list-style-type: none"> 입력 실패 결과와 원인을 인터페이스 테이블에 기록한다. 입력 실패 결과를 사용자에게 알람을 통해 알려준다. 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.
인터페이스 데이터 전송	DB Connection* 오류	<ul style="list-style-type: none"> 통신 결과를 통해 인터페이스 실패 결과와 원인을 인터페이스 테이블에 기록한다. 인터페이스 실패 결과와 원인을 사용자와 관리자에게 이메일 등으로 전송한다.
	데이터 전송 주체의 논리적 오류	<ul style="list-style-type: none"> 인터페이스 실패 결과와 원인을 인터페이스 테이블에 기록한다. 인터페이스 실패 결과를 사용자와 관리자에게 이메일 등으로 전송한다. 예외 케이스의 재발 방지를 위해 프로그램을 개선한다.

예2 수신 인터페이스 테이블을 이용한 인터페이스 기능 실패 시 예외 처리 방안

프로세스	예외 상황	예외 처리 방안
인터페이스 데이터 읽기	데이터 선택 시 오류	<ul style="list-style-type: none"> 수신 측 사용자에게 알람을 통해 예외사항을 알려준다. 인터페이스 테이블에 예외사항을 기록한다. 재발되지 않도록 프로그램을 개선한다.
데이터 트랜잭션	데이터 트랜잭션 시 프로그램의 논리상 오류	<ul style="list-style-type: none"> 사용자에게 알람을 통해 예외사항을 알려준다. 인터페이스 테이블에 예외사항을 기록한다. 재발되지 않도록 프로그램을 개선한다.
처리 결과 응답	DB Connection 오류	<ul style="list-style-type: none"> 인터페이스 테이블에 예외사항을 기록한다. 송·수신자에게 이메일 등으로 예외사항을 알려준다.

Session066 인터페이스 보안

1 인터페이스 보안의 개요

1.1 모듈 간 통신 및 정보 교환 통로, 충분한 보안 기능 필요

1.2 보안 취약점 분석 후 보안 기능 적용

2 인터페이스 보안 취약점 분석

2.1 각 구간에 어떤 보안 취약점 있는지 분석

2.2 각 구간의 구현 현황은 송 수신 영역의 구현 기술 및 특징 등을 구체적으로 확인

2.3 확인된 기능을 기반으로 송신 데이터 선택, 송신 객체 생성, 인터페이스 송 수신, 데이터 처리 결과 전송 등 영역별 발생 가능 보안 취약점을 시나리오 형태로 작성

- 3 인터페이스 보안 기능 적용
 - 3.1 취약점 기반으로 보안 기능 적용
 - 3.2 일반적으로 네트워크, 애플리케이션, 데이터베이스 영역에 적용
 - 3.3 네트워크 영역
 - 스니핑(Sniffing) 등을 이용 데이터 탈취 및 변조 위협 방지 위해 트래픽의 암호화
 - IPSec, SSL, S-HTTP 등 다양한 방식
 - 3.4 애플리케이션 영역
 - 개발 보안 가이드 참조, 코드 상 보안 취약점 보완
 - 3.5 데이터베이스 영역
 - 접근 권한, 데이터베이스 동작 객체의 보안 취약점에 보안 기능 적용
 - 민감한 데이터를 암호화, 익명화 등 데이터 자체의 보안 방안 고려

Session067 연계 테스트

- 1 연계 테스트의 개요
 - 1.1 연계 시스템, 구성 요소가 정상적으로 동작하는지 확인
 - 연계 테스트 케이스 작성, 연계 테스트 환경 구축, 연계 테스트 수행, 연계 테스트 수행 결과 검증
- 2 연계 테스트 케이스 작성
 - 2.1 연계 시스템 간의 데이터 및 프로세스의 흐름 분석 필요 테스트 항목 도출하는 과정
 - 2.2 송 수신용 연계 응용 프로그램의 단위 테스트 케이스와 연계 테스트 케이스 각각 작성
 - 2.2.1 송 수신용 연계 응용 프로그램 단위 테스트 케이스
 - 송 수신 시스템에서 확인해야 할 항목 도출
 - 단순 개별 데이터의 유효값 확인 경우의 수와 데이터간 연관 관계를 확인하는 경우의 수로 구분
 - 2.2.2 연계 테스트 케이스
 - 송 수신용 연계 응용 프로그램의 기능상 결함을 확인 하는 단위 테스트 케이스 형태로 작성
 - 단위 테스트 케이스는 연계 테이블 간 송 수신 절차 앞뒤로 연결, 흐름 확인할 수 있게
- 3 연계 테스트 환경 구축
 - 3.1 일정, 방법, 절차, 소요 시간 등 송 수신 기관과 협의를 통해 결정
 - 3.2 연계 서버, 송 수신용 어댑터 설치, 연계 위한 IP, Port 허용 신청, 디비 계성 및 테이블, 데이터 생성 등 테스트 환경 구축

- 4 연계 테스트 수행
 - 4.1 시험 항목 및 처리 절차 등 실제로 진행
 - 4.2 송 수신용 연계 응용 프로그램의 단위 테스트를 먼저 수행
 - 4.3 완료 후 연계 테스트 케이스에 따라 데이터 추출, 송 수신, 데이터 반영 과정 등 수행
- 5 연계 테스트 수행 결과 검증
 - 5.1 예상 결과와 동일한지 확인
 - 5.2 방법
 - 운영 DB 테이블의 건수 확인
 - 테이블 또는 파일 열어 데이터 확인
 - 파일 생성 위치에서 파일 생성 여부 및 파일 크기 확인
 - 연계 서버에서 제공하는 모니터링 현황 확인
 - 로그를 확인

Session068 인터페이스 구현 검증

- 1 인터페이스 예외 처리의 개요
 - 1.1 문제없이 작동하는지 확인
 - 1.2 검증 도구와 감시 도구 이용 동작 상태 확인
- 2 인터페이스 구현 검증 도구
 - 2.1 인터페이스 단위 기능, 시나리오 등을 기반으로 하는 통합 테스트 필요
 - 2.2 자동화 도구 이용하면 효율적 수행
 - xUnit : juni, cppunit, nunit 등 다양한 언어 지원 단위 테스트 프레임 워크
 - STAF : 서비스 호출 및 컴포넌트 재사용 등 다양한 환경 지원
 - FitNesse : 웹 기반 테스트케이스 설계, 실행, 결과 확인 지원 프레임 워크
 - NTAF : 협업 기능 재사용 및 확장성 통합한 Naver의 테스트 자동화 프레임 워크
 - Selenium : 다양한 브라우저, 개발 언어 지원 웹 애플리케이션 테스트 프레임 워크
 - Water : ruby를 사용하는 애플리케이션 테스트 프레임워크
- 3 인터페이스 구현 감시 도구
 - 3.1 APM을 사용 Monitoring 가능
 - 3.2 애플리케이션 성능 관리 도구를 통해 데이터베이스와 트랜잭션, 변수값, 호출 함수, 로그 및 시스템 부하 등 종합적인 정보 조회 분석 가능
 - 3.3 Scouter, Jennifer

- 4 인터페이스 구현 검증 도구 및 감시 도구 선택
 - 4.1 인터페이스 명세서의 세부 기능을 참조, 정상 동작 여부 확인 위한 검증 도구와 감시 도구의 요건 분석
 - 4.2 시장 및 솔루션 조사를 통해 인터페이스 구현 검증, 감시에 필요한 구현 검증 도구, 감시 도구 선택
- 5 인터페이스 구현 검증 확인
 - 5.1 구현 검증 도구 이용 외부 시스템과 연계 모듈의 동작 상태 확인
 - 5.2 인터페이스 동작 프로세스상에서 예상 되는 결과값, 실제 검증 값 동일한지 비교
 - 5.3 오류 처리도 적절 한지 확인
- 6 인터페이스 구현 감시 확인
 - 6.1 구현 감시 도구 이용 외부 시스템과 연결 모듈이 서비스 제공 할 동안 정상 동작하는 지 확인
 - 6.2 동작 여부, 에러 발생 여부 등 감시 도구에서 제공해 주는 리포트 활용