Journal of Korea Multimedia Society Vol. 15, No. 2, February 2012(pp. 266-272)

# 3-TIER 구조 소프트웨어의 다국어 지원 방식의 설계와 구현

고 정 국†

## 요 약

해외 시장을 겨냥한 소프트웨어의 상품화를 위해서는 여러 나라의 언어를 지원하는 다국어 지원 기능이 필요하다. 3-tier 구조는 2-tier 구조의 문제점을 해결하기 위해 애플리케이션을 분리하여 응용 계층을 두고 프리젠테이션 로직과 데이터베이스를 미들웨어로 연결하는 형태이다. 3-tier 구조의 장점은 애플리케이션의 부하 분산으로 성능이 향상되며 확장이 쉽고, 유지보수와 재사용이 용이하다는 점이다. 본 논문에서는 3-tier 구조의 기업용 소프트웨어를 대상으로 소프트웨어 개발과 유지보수, 지원 언어의 추가가 용이한 공통 리소스 활용 방식을 제안하고 빌링 솔루션의 다국어 버전 개발에 적용하여 유용성을 확인한다. 제안하는 방식은 기능 모듈마다 언어별 리소스 파일을 별도로 유지하는 닷넷의 기존 방식을 개선하여 언어별로 하나만 유지하고 다국어 지원 클래스 라이브러리 형태로 제공하여 메모리와 디스크의 공간 낭비를 줄인다. 또한 다국어 지원 클래스 라이브러리를 응용 계층에 배치하여 소프트웨어 개발과 유지보수, 지원 언어 추가가 용이하다. 한편 리소스 파일에 대한 부적절한 변경을 막기 위해 다국어 지원 클래스 라이브러리는 dll 파일로 제공한다.

# Design and Implementation of Multilingual support method for 3-tiered softwares

Jeong-Gook Koh†

## ABSTRACT

Multilingual support of software is necessary for entering global market. 3-tier architecture is a solution for problems of 2-tier architecture. It divides an application into a client-tier and an application-tier, and presentation logic and database are connected by middleware. The advantage of 3-tier architecture is the enhanced performance through load balancing, scalability, easier maintenance and reusability. This paper proposes a multilingual support method that utilizes common resource files for 3-tier enterprise softwares, applies the proposed method to development of multilingual version of billing solution, and verify the usefulness of it. It is easy for development and maintenance of software, the addition of language supported. Proposed method holds a resource file for each language and provides a multilingual support class library. Therefore this method reduces a waste of memory and disk space. Deployment of a class library into an application tier makes development and maintenance of software, the addition of new language supported easy. To avoid inappropriate modification of a resource file, a multilingual support class library is provided in a dll file.

**Key words:** 3-tier, multilingual support(다국어 지원), resource file(리소스 파일)

※ 교신저자(Corresponding Author) : 고정국, 주소 : 부산광역시 남구 신선로 428번지(608-711), 전화 : 051)629-1171, FAX : 051)629-1169, E-mail : jgkoh@tu.ac.kr
접수일 : 2011년 8월 30일, 수정일 : 2011년 10월 18일
완료일 : 2011년 11월 22일
† 정회원, 동명대학교 컴퓨터공학과
※ 본 연구는 중소기업청의 2010년도 산학연협력 기업부설연구소 지원사업(No.00044198)의 지원으로 수행되었음.

## 1. 서 론

최근 국제화 추세에 따라 해외 시장을 겨냥한 소프트웨어를 개발하여 상품화하려는 시도가 증가하고 있다. 해외 시장에 소프트웨어를 출시하려면 여러 종류의 언어를 지원하는 다국어 지원 기능과 다국적 요구사항을 반영한 국제화가 필요하다[1,2]. 한편 3-tier 구조는 클라이언트와 서버로 구성된 2-tier 구조의 문제점을 해결하는 방안으로서 애플리케이션을 분리하여 중간 계층에 두고 프리젠테이션 로직과 데이터베이스를 미들웨어로 연결하는 형태이다. 3-tier 구조는 애플리케이션의 부하 분산으로 시스템 성능이 향상되며 서버나 데이터베이스에 관계없이 확장이 용이하고, 애플리케이션의 집중 관리로 유지보수와 재사용성이 용이하다[3,4].

본 논문에서는 3-tier 구조의 기업용 소프트웨어를 대상으로 소프트웨어 개발과 유지보수, 지원 언어의 추가가 용이한 다국어 지원 소프트웨어 개발 방식을 제안하고 빌링 솔루션의 다국어 버전 개발에 적용하여 유용성을 확인한다. 본 논문의 구조는 다음과 같다. 2장에서는 2-tier와 3-tier 구조의 장단점을 비교하고 언어 정보의 제공 형태에 따른 다국어 지원 방식과 다국어 지원 프로그램의 형태를 설명한다. 3장에서는 3-tier 구조의 기업용 소프트웨어에 대한 다국어 지원 방식을 제안하고 공통 리소스 파일의 구조를 설명한다. 4장에서는 제안한 방식을 B2B(Business to Business) 방식의 빌링 솔루션 개발에 적용하여 구현한 내용을 기술한다. 마지막으로 5장에서는 본 논문의 결론과 향후 연구방향을 제시한다.

## 2. 관련 연구

### 2.1 2-tier와 3-tier 구조

기존의 소프트웨어들은 서버와 클라이언트로 구성된 2-tier 구조를 많이 채택하고 있는데, 2-tier 구조는 다음과 같은 단점이 있다. 첫째, 클라이언트 수가 증가하면 네트워크 트래픽이 증가하고 서버에 과도한 부하를 유발하여 시스템이 느려진다. 둘째, 처리 기능이 클라이언트와 서버에 분산되어 있는 경우 클라이언트와 서버가 상호 협력을 해야 하므로 프로그램의 변경이나 배포가 어렵다[4,5].

3-tier 구조는 2-tier 구조의 단점에 대한 해결책으로 등장하였다[3-6]. 3-tier 구조는 그림 1과 같이 사용자 인터페이스를 제공하는 클라이언트 계층(client tier)과 서버의 기능을 수행하는 데이터 접근 계층(data access tier), 그리고 클라이언트의 요청을 서버에게 전송하고 처리 결과를 클라이언트에게 제공하는 응용 계층(Application tier)으로 구성된다. 이 구조는 확장성과 안정성, 그리고 효율 측면에서 유리하므로 대용량 서비스에 적합하며, 클라이언트의 요청이 서버에 직접 전달되지 않기 때문에 서버의 부하가 클라이언트 수의 증가에 비례하지 않는다.

3-tier 구조의 장점은 다음과 같다. 첫째, 클라이언트에 DB 접속을 위한 계정이나 참조 권한 등의 정보를 저장할 필요가 없어서 보안 측면에서 유리하다. 둘째, 클라이언트가 서버에 서비스를 요청한 후 연결을 차단하는 비연결형(connectionless) 방식을 사용하므로 적은 수의 연결로 다수의 클라이언트들을 서비스할 수 있다. 셋째, 프로그램의 배포와 유지보수가 용이하다. 클라이언트에 DB 연결 라이브러리를 설치할 필요가 없으므로 응용 프로그램을 설치한 후 주기적인 업데이트만 하면 된다. 넷째, 다중 DB 활용이 가능하다. DB 서버의 변경이나 추가시 클라이언트의 DB 활용 프로그램을 모두 수정해야 하는 2-tier 구조에 비해 3-tier 구조에서는 응용 계층만 수정하면 된다. 따라서 최근 사용자 수가 많고 대용량의 데이터를 다루는 기업용 소프트웨어들은 대부분 3-Tier 구조를 채택하고 있다.

### 2.2 다국어 지원 방식

소프트웨어의 다국어 지원 방식은 언어 정보의 제

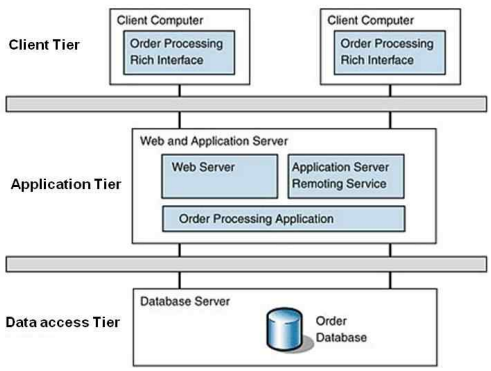


그림 1. 3-tier 구조

공 형태에 따라 다음과 같이 세 가지로 분류할 수 있다[2,7].

첫 번째 형태는 텍스트 파일로 제공하며, 오픈 소스 소프트웨어에서 많이 채택하고 있다. 실행 파일은 메뉴나 메시지 등의 언어 정보를 보유한 초기화(.ini) 파일을 활용한다. 장점은 다국어 정보를 언어별로 제공할 수 있으며 사용자가 초기화 파일을 수정할 수 있다. 단점은 사용자가 초기화 파일을 부적절하게 변경할 수 있으므로 오류 방지 코드가 추가적으로 필요하며, 이미지와 같은 복잡한 리소스는 별도의 파일로 제공해야 한다. 또한 프로그램 업데이트시 실행 파일과 리소스 파일을 함께 배포해야 한다.

두 번째 형태는 데이터베이스를 활용하며, 비즈니스 서버 응용에서 많이 채택하고 있다. 장점은 언어 정보가 데이터베이스에 저장되어 있으므로 응용 프로그램에서 손쉽게 활용할 수 있으며 정보 검색도 용이하다. 단점은 응용 프로그램이 항상 데이터베이스에 접속가능해야 한다.

세 번째 형태는 리소스(resource) 파일로 제공하며, 마이크로소프트사의 닷넷 프레임워크에서 제공하는 리소스 관리자(resource manager)를 활용한다. 장점은 텍스트뿐만 아니라 아이콘, 이미지, 파일도 리소스로 활용할 수 있으며, 리소스 파일을 수정할 수 있다. 또한 컴파일이 가능하므로 파일 크기가 작고 실행 속도도 높일 수 있다. 단점은 닷넷 프레임워크에 의존하므로 리소스 파일을 수정하려면 Visual Studio .NET 툴을 사용해야 하며, 일반 사용자는 리소스 파일의 변경과 관리가 어렵다.

### 2.3 다국어 지원 프로그램의 형태

다국어 지원 프로그램의 형태는 실행 파일과 리소스 파일의 배포 방식에 따라 그림 2와 같이 세 가지로 구분할 수 있다[1,8,9].

첫 번째 형태는 실행 파일을 언어별로 제공하는 개별 실행 파일 방식이다. 장점은 언어별로 별도의 실행 파일을 제공하므로 구현이 용이하다. 단점은 언어별로 프로그램 코드를 관리해야 하고, 프로그램이 일부 변경되면 소스 코드를 다시 컴파일해야 한다.


(a) 개별 실행 파일 방식 (b) 단일 리소스 방식 (c) 복수 리소스 방식

그림 2. 3-tier 구조

지원 언어를 추가하려면 프로그램을 새로 작성해야 하며, 프로그램의 내부 로직이 변경되면 모든 프로그램을 수정해야 한다.

두 번째 형태는 실행 파일과 하나의 리소스 파일을 제공하는 단일 리소스 방식이다. 장점은 언어 정보를 하나의 리소스 파일에 담기 때문에 복수 리소스 방식보다 구현이 용이하며, 프로그램 실행 중 언어 전환이 자유롭다. 또한 언어 정보 변경시 리소스 파일만 수정하면 된다. 단점은 리소스 파일에 사용하지 않은 언어 정보까지 담겨 있으므로 메모리와 디스크 공간의 낭비가 발생한다. 또한 언어마다 특성이 상이하므로 언어별로 프로그램의 정상 동작 여부를 검사해야 한다.

세 번째 형태는 실행 파일과 다수의 리소스 파일을 제공하는 복수 리소스 방식이다. 장점은 리소스 파일이 언어별로 존재하므로 지원 언어의 추가가 용이하다. 그리고 특정 언어의 정보를 변경할 때 관련된 리소스 파일만 수정하므로 유지보수가 용이하다. 단점은 다수의 리소스 파일을 유지하기 때문에 기능 시험 과정이 가장 복잡하다.

## 3. 다국어 지원 소프트웨어의 개발 방식

### 3.1 공통 리소스 파일의 활용 방식

리소스는 문자열과 아이콘, 이미지처럼 프로그램의 실행에 필요한 데이터이다. 닷넷에서는 리소스들을 효율적으로 관리할 수 있도록 .resx 파일 형식을 지원하고 있다. resx 형식의 리소스 파일은 데이터를 XML로 구성하여 관리하는 XML 파일이다[7,9,10].

3-tier 구조의 기업용 소프트웨어에서 클라이언트 계층에는 다수의 기능 모듈이 존재한다. 닷넷에서는 다국어 지원을 위해 그림 3과 같이 기능 모듈마다 .resx 형식의 리소스 파일들을 유지하므로 특정 언어의 리소스 파일이 변경되면 해당 리소스 파일을 사용하는 모든 기능 모듈들의 소스 코드를 재컴파일해야 한다. 또한 지원 언어 추가시 리소스 파일을 기능 모듈마다 추가해야 하는 번거로움이 있다. 따라서 기존 방식은 기능 모듈의 수가 적은 소규모 소프트웨어에 적합하다.

본 논문에서는 다국어 지원이 가능한 3-tier 구조의 기업용 소프트웨어 개발 방식으로 닷넷의 리소스 파일 활용 방식을 개선한 공통 리소스 파일 활용 방

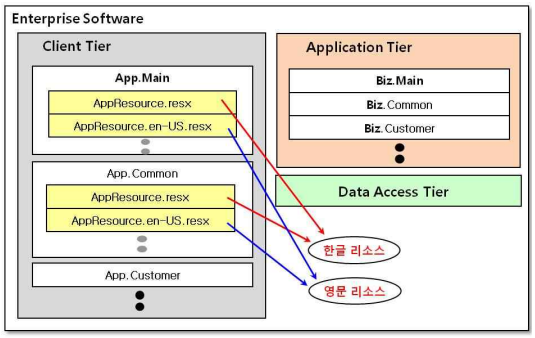


그림 3. 닷넷의 리소스 파일 활용 형태

식을 제안한다. 이 방식은 그림 4와 같이 기능 모듈에 존재하던 다수의 언어 리소스 파일들을 통합하여 언어별로 리소스 파일을 하나만 유지하고 이들을 묶어서 다국어 지원 클래스 라이브러리를 구성한다. 다국어지원 클래스 라이브러리는 리소스 파일과 클라이언트 계층의 인터페이스 역할을 하는 클래스 파일로 구성되며, 응용 계층에 배치한다. 클라이언트 계층에서는 클래스 파일의 메소드를 호출하여 다국어 지원 기능을 활용한다.

제안한 방식은 다수의 언어 리소스 파일들을 하나의 다국어 지원 클래스 라이브러리로 대체함으로써 기존 방식에 비해 메모리와 디스크 공간의 낭비를 줄이고 소프트웨어 개발과 유지보수가 용이하다. 또한 지원 언어를 새로 추가하려면 해당 언어의 리소스 파일을 작성한 후 다국어지원 클래스 라이브러리에 추가하면 되므로 기능 확장도 용이하다. 한편 공통 리소스 파일은 XML 문서이므로 열람과 수정이 가능하다. 본 논문에서는 리소스 파일에 대한 부적절한

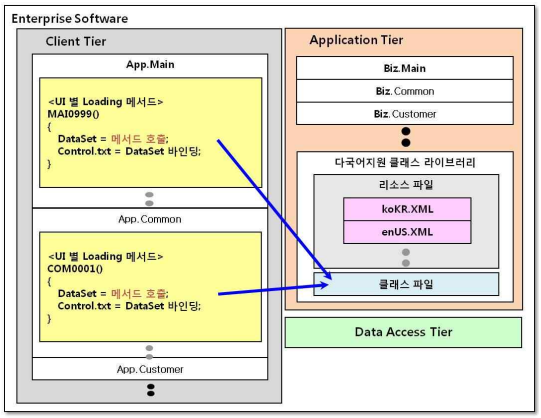


그림 4. 공통 리소스 파일의 활용 방식

변경을 막기 위해 다국어 지원 클래스 라이브러리를 dll(dynamic linking library) 파일로 제공한다.

### 3.2 공통 리소스 파일의 구조

본 논문에서는 공통 리소스 파일의 구조를 그림 5와 같이 설계하였다. 공통 리소스 파일은 기능 모듈의 세부 기능을 담당하는 사용자 인터페이스들을 정의한다. 그리고 각 사용자 인터페이스마다 컨트롤 명칭을 정의하는 id 속성, 컨트롤 명칭에 대입할 값을 정의하는 value 속성, 그리고 사용자 인터페이스의 명칭을 정의하는 ui_name 필드로 구성된 엘리먼트들을 기술한다.

```
<?xml version="1.0" encoding="utf-8" ?>

<!-- 루트 엘리먼트 선언 -->
<!ELEMENT ctlkeylist (ctlkey+)>
   <!ELEMENT ctlkey (id, value, ui_name)>
      <!ELEMENT id (#PCDATA)>
      <!ELEMENT value (#PCDATA)>
      <!ELEMENT ui_name (#PCDATA)>
```

그림 5. 공통 리소스 파일의 구조

## 4. 다국어 지원 소프트웨어의 구현

### 4.1 구현 환경 및 개발 도구

본 논문에서는 제안하는 방식을 'WideUse-Billing'이라는 B2B 방식의 빌링 솔루션의 다국어 버전 개발에 적용하였다. 빌링 솔루션의 다국어 버전을 구현하기 위해 윈도우 2003 서버 버전이 설치된 PC에서 개발 플랫폼으로 닷넷 프레임워크 3.5를, 개발 언어로 C#을 사용하였다. 개발 도구로 Visual Studio 2008과 Firebird Maestro 7.7을 사용하였고 형상 및 버전 관리 도구로 Visual SourceSafe 2005를 사용하였다. 데이터베이스는 Firebird 2.0[11]을 사용하였다.

### 4.2 구현 내역

#### 4.2.1 빌링 솔루션의 제공 기능

WideUseBilling 빌링 솔루션의 클라이언트 계층은 9개의 기능 모듈로 구성되어 있다. 기능 모듈별 제공 기능과 사용자 인터페이스의 명칭은 표 1과 같다.

9개 기능 모듈의 소스 코드 구성은 그림 6과 같다.

표 1. 빌링 솔루션의 기능 모듈

| 대분류 | 중분류 | 소분류 | UI 명칭 |
|---|---|---|---|
| 메인 | 메인 화면 | 메인 화면 | MAI0000 |
| | | 로그인 | MAI0001 |
| | | … | … |
| | | 언어 설정 변경 | MAI0999 |
| 기초 정보 설정 | 공통 코드 | 코드 추가/삭제 | COM0000 |
| | | 코드 조회 | COM0001 |
| | … | … | … |
| | 청구항목 관리 | | COM0003 |
| | … | … | … |
| 그룹 관리 | 그룹 관리 | | GRU0001 |
| | 권한 관리 | | GRU0002 |
| 고객 관리 | 고객 등록 | | CUS0001 |
| | … | … | … |
| | 상담 내역 조회 | | CUS0007 |
| 수납 관리 | 청구원장 | 일괄 생성 관리 | PAY0000 |
| | | 개별 생성 관리 | PAY0001 |
| | 수납관리 대장 | | PAY0002 |
| 전자 금융 | CMS 관리 | 통합 승인 관리 | CMS0000 |
| | | … | … |
| | | 자료 전송 | CMS0004 |
| 지로 관리 | 지로 관리 | 지로 출력 | GIR00000 |
| | | … | … |
| | | EDI 수납 처리 | GIR0205 |
| 부가 서비스 | DM 발행 | | SER0000 |
| | 전자세금계산서 발행 | | SER0001 |
| | 현금영수증 발행 | | SER0002 |
| 통합 조회 | 미수 내역 조회 | | TOT0000 |
| | … | … | … |
| | 출금항목별 입금대장 | | TOT0010 |
| 기타 | 데이터 관리 | 백업 | ETC0000 |
| | | … | … |
| | | A/S 데이터 전송 | ETC0003 |
| | … | … | … |
| | e-giro 사이트 연결 | | ETC0007 |

각 모듈은 기능을 담당하는 코드와 사용자 인터페이스를 정의하는 코드로 구성된다.

### 4.2.2 다국어 지원 클래스 라이브러리

본 논문에서는 한국어와 영어의 공통 리소스 파일들로 구성된 다국어 지원 클래스 라이브러리를 그림 7과 같이 구현하였다. 그리고 공통 리소스 파일이 닷넷의 기존 리소스 파일(.resx)과 구분되도록 공통 리

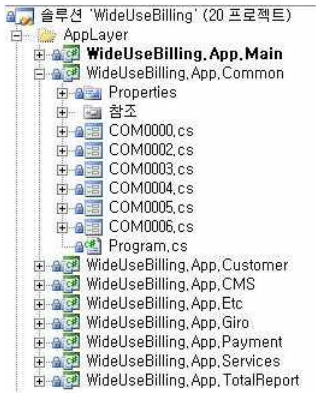


그림 6. 기능 모듈들의 코드 구성


그림 7. 다국어 지원 클래스 라이브러리

소스 파일의 확장자를 .xml로 설정하였다.

9개 기능 모듈 중 하나인 기초정보설정 모듈에서 '청구항목 관리' 기능을 수행하는 사용자 인터페이스에 대한 한국어 리소스 파일의 내용은 그림 8과 같다. 첫 번째 엘리먼트는 ctlkey 태그의 컨트롤 명칭과 컨트롤 명칭에 대입할 값이 각각 'COM0003'과 '청구항목 관리'이며 사용자 인터페이스의 명칭은 'COM0003'이라는 것을 정의하고 있다.

한편 지원 언어를 새로 추가할 때는 해당 언어의 공통 리소스 파일을 작성한 후 다국어 지원 클래스 라이브러리에 추가하면 되므로 기능 확장이 용이하다.

### 4.2.3 사용자 인터페이스 구현 내역

'청구항목 관리' 기능에 대한 언어별 사용자 인터페이스 구현 내역은 그림 9와 같다. 청구항목 관리 기능을 담당하는 모듈에서는 언어 설정 내역에 따라

```
<!-- WideUseBilling.App.Common(COM) -->
<ctlkey id="COM0003" value="청구항목 관리"> COM0003 </ctlkey>
<ctlkey id="groupBox2" value="관리"> COM0003 </ctlkey>
<ctlkey id="selfaLabel3" value="청구코드"> COM0003 </ctlkey>
<ctlkey id="selfaLabel5" value="청구명"> COM0003 </ctlkey>
<ctlkey id="selfaLabel6" value="부과단위"> COM0003 </ctlkey>
<ctlkey id="selfaLabel4" value="비고"> COM0003 </ctlkey>
<ctlkey id="selfaLabel8" value="청구금액"> COM0003 </ctlkey>
<ctlkey id="selfaLabel9" value="연체료부과 \n여부"> COM0003 </ctlkey>
<ctlkey id="slfbtn_Select" value="조회"> COM0003 </ctlkey>
<ctlkey id="slfbtn_New" value="신규"> COM0003 </ctlkey>
<ctlkey id="slfbtn_Update" value="수정"> COM0003 </ctlkey>
<ctlkey id="slfbtn_Delete" value="삭제"> COM0003 </ctlkey>
<ctlkey id="slfgv_List" value="청구코드,청구명,청구방법,부과단위,청구금액,
연체료부과,비고"> COM0003 </ctlkey>
```

그림 8. 한국어 리소스 파일의 내용

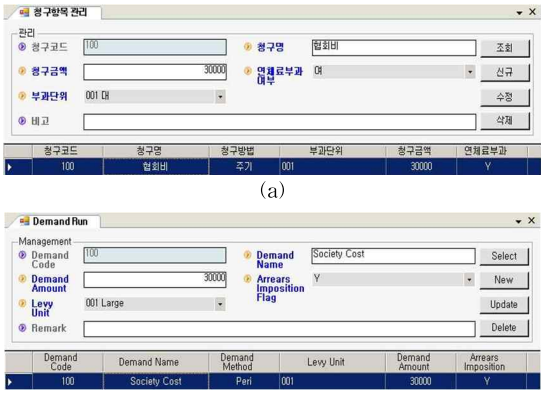


그림 9. 청구항목 관리 기능 모듈의 언어별 사용자 인터페이스 (a) 한국어 사용자 인터페이스, (b) 영어 사용자 인터페이스

다국어 지원 클래스 라이브러리로부터 해당 리소스 파일의 내용을 불러와서 사용자 인터페이스를 설정하며, 언어 설정을 변경하면 사용자 인터페이스가 다른 언어로 전환된다.

## 5. 결 론

본 논문에서는 3-tier 구조의 기업용 소프트웨어를 대상으로 소프트웨어 개발과 유지보수, 지원 언어의 추가가 용이한 공통 리소스 활용 방식을 제안하고 빌링 솔루션의 다국어 버전 개발에 적용하여 유용성을 확인하였다.

제안한 방식은 단일 리소스 방식을 채택하기 때문에 소스 코드를 언어별로 관리하고 지원 언어 추가시 프로그램을 새로 구현하는 개별 실행파일 방식이나 특정 언어의 정보 변경시 관련된 리소스 파일만 수정하므로 유지보수가 쉽지만 기능 시험 과정이 가장 복잡한 복수 리소스 방식에 비해 소프트웨어의 개발 비용이 절감되고 새로운 지원 언어의 추가가 용이하다. 그리고 기능 모듈마다 언어별 리소스 파일을 유지하는 닷넷의 기존 방식의 단점을 개선하여 언어별로 공통 리소스 파일을 하나만 유지하고 이들을 묶어서 다국어 지원 클래스 라이브러리로 제공함으로써 메모리와 디스크 공간의 낭비를 줄이고, 다국어 지원 클래스 라이브러리를 기존의 클라이언트 계층에서 응용 계층에 배치함으로써 소프트웨어 개발기간을 단축하고 유지보수도 용이하다. 한편 다국어 지원 클래스 라이브러리를 dll 파일로 제공함으로서 공통 리소스 파일에 대한 부적절한 변경도 방지할 수 있다.

현재 구현된 빌링 솔루션은 한국어와 영어만 지원하고 있으나, 해외 시장을 겨냥한 소프트웨어의 상품화를 위해서는 보다 다양한 언어의 지원이 필요하다. 따라서 사용자층이 두터운 언어들을 대상으로 다국어 지원 클래스 라이브러리들을 보강하면서 다른 한편으로는 소프트웨어의 개발과 유지보수가 용이한 효율적인 다국어 지원 방안을 지속적으로 연구해 나갈 예정이다. 이러한 기술은 국내 소프트웨어들의 국제화 외에도 해외 소프트웨어들의 지역화에도 많은 도움이 될 것이다.

## 참 고 문 헌

[1] 박문목, 김진수, “효율적인 다국어 프로그래밍 방법에 관한 연구,” 한국정보기술응용학회 춘계학술대회논문집, pp. 67-74, 2007.

[2] 박영진, 고정국, “3-Tier 환경에서 다국어 지원 소프트웨어의 개발,” 한국멀티미디어학회 춘계학술대회논문집, 제14권, 제1호, pp. 145-146, 2011.

[3] Rahman Mahmoodi, 3-tier architecture in C#, http://www.codeproject.com/11128/3-tier-architecture-in-C.aspx, 2005.

[4] Channu Kambalyal, 3-Tier Architecture, http://channukambalyal.tripod.com/NTierArchitecture.pdf, 2010.

[5] Scott Clinton, Developing for Multi-tier Distributed Computing Architectures with Delphi Client/Server Suite 2.0, http://edn.embarcadero.com/article/10343, 1996.

[6] Mai-lan Tomsen, “Build Reliable and Scalable N-Tier Applications that Run on Both Windows NT and UNIX,” *Microsoft Systems Journal*, Vol.13, No.12, pp. 20-27, 1998.

[7] 디밥, RCL: 다국어 지원을 위한 방안, http://debop.egloos.com/2304744, 2009.

[8] MSDN 자습서, Windows 응용 프로그램의 다국어 클라이언트 지원 구성, http://msdn.microsoft.com/ko-kr/library/ms227565(VS.90).aspx, 2011.

[9] Go Global 개발자 센터, Win32 Multilingual User Interface 응용 프로그램 작성, http://msdn.microsoft.com/ko-kr/goglobal/bb688160.aspx, 2011.
[10] 김상형, 닷넷 프로그래밍 정복, 사이텍미디어, 2006.
[11] Firebird Foundation Incorporated, Firebird: True Universal Open Source Database, http://www.firebirdsql.org, 2011.

고 정 국

1992년 부산대학교 컴퓨터공학과 공학사
1994년 부산대학교 컴퓨터공학과 공학석사
1999년 부산대학교 컴퓨터공과 공학박사

1999년 3월~1999년 8월 위덕대학교 컴퓨터공학과 전임강사
1999년 9월~현재 동명대학교 컴퓨터공학과 부교수
관심분야 : 운영체제, 임베디드 시스템, 컴퓨터활용교육