# 교환기 소프트웨어를 위한 사업자별 소스 관리 방안

권성희, 김병철, 권경인, 조시철

LG 정보통신

E-mail. shkwon@rex.lgic co kr

# A Source Management Method for Exchange Software

Seong-Hee Kwon, Byungchul Kim, Kyoung-In Kwon, Si-Cheol Cho

LG Information & Communocations

요 약

LG 정보통신에서 개발 중인 교환기는 국내 및 해외의 다양한 통신 서비스 사업자를 납품 대상으로 한다. 통신 서비스 사업자들이 요구하는 교환기의 기본 기능은 공통되지만, 사업자별로 추가적인 기능을 요구하거나, 동일한 기능에서도 요구하는 세부 사항에는 차이가 있기 마련이다. 따라서, 교환기 소프트웨어를 개발하는데 있어 체계적인 형상관리를 하면서 동시에 사업자별로 상이한 요구사항을 수용할 수 있는 방안이 필요하다. 본 논문에서는 모든 사업자들의 요구사항을 동일한 소스에 구현하고, 컴파일 시 사업자 지정을 통해 해당 사업자의 기능만을 추출하는 방안에 대해 설명한다

## 1. 개요

교환기 소프트웨어와 같이 규모가 큰 소프트웨어는 형상관리[1]가 필수적이다. 즉, 다수의 개발자에 의해 병렬적으로 개발이 진행되는 동안 작업이 중복되거나 개발자 실수에 의해 부정확한 패키지를 제작할 가능성이 높고 소스의 변경을 통제하기 어렵기 때문이다. 형상관리는 소스를 하나의 데이터베이스로 관리하고 개발자 작업 공간을 제공하며 소스 변경 절차를 두어 소스에 대한 변경 및 접근을 통제함으로써 이러한 문제의 해결을 가능하게 한다.

LG 정보통신에서 개발 중인 교환기는 국내 및 해외의 다양한 통신 서비스 사업자를 납품 대상으로 한다 그런데, 통신 서비스 사업자들이 요구하는 교환기의 기본 기능은 공통되지만, 사업자별로 추가적인 기능을 요구하거나, 동일한 기능에서도 요구하는 세부 사항에 차이가 있기 마련이다 따라서, 교환기 소프트웨어를 개발하는데 있어 형상관리를 체계적으로 하면서 동시에 사업자별로 상이한 요구사항을 수용할 수 있는 방안이 필요하다.

형상관리 도구에 포함된 버전관리 기능에는 하나의 소스로부터 기능이 다른 변종(variant)을 만들 수 있는 branch 기능이 있다[2][3][4]. 그러나, 사업자별 소소 관리를 위하여 각 사업자별로 branch 를 생성하는 것은 매우 번거로운 작업이다. 또한, 한 사업자의 특정 기능을 다른 사업자에 포함시키기 위해서는 추가적인 작업이 필요하며, 이때 불필요한 오류가 발생할 수 있다.

본 논문에서는 버전관리 도구를 이용하지 않고, 모든 사업자들의 요구사항을 동일한 소스에 구현하고, 컴파일 시 사업자 지정을 통해 해당 사업자의 기능만을 추출하는 방안에 대해 설명한다.

2 장에서는 사업자별 소스 관리에 대한 요구사항을 정리하였다. 형상관리에 포함된 버전관리 도구를 이용한 방법에 대한 문제점은 3 장에서 설명한다. 버전관리 도구를 이용하지 않고 사업자별 소스 관리 방안은 4 장에서 설명한다. 5 장에서는 본 방안이 가지는 장점을 열거하고, 6 장은 결론과 앞으로의 과제이다.

## 2. 사업자별 소스 관리 요구사항

사업자별로 상이한 요구사항을 수용하면서 동시에 효율적으로 교환기 응용 프로그램을 형상관리하기 위해, 소스 관리에 다음과 같은 요구 사항들이 제기되었다.

① 사업자별로 소스를 구분할 수 있어야 한다 즉, 컴파일 시

사업자에 따라 소스 파일을 첨삭하거나, 한 소스 파일에서 코드를 줄단위로 첨삭할 수 있어야 한다.

② 교환기의 주요 기능(function)별로 소스를 구분할 수 있어야 한다

③ 하나의 소스로 관리해야 한다 소스를 하나로 관리하지 않으면 사업자별, 기능별 공통 기능을 구현하거나 변경할 때 일관성을 유지하기 힘들다.

④ 컴파일된 결과물을 저장할 공간을 사업자별로 구분해야 한다 저장 공간을 구분함으로써 사업자별로 이미 컴파일한 결과물을 재사용할 수 있다.

⑤ 개발자가 쉽게 소스 파일을 편집할 수 있고 컴파일할 수 있으며, 컴파일 결과물을 참조할 수 있어야 한다.

⑥ 개발자들이 수행하는 프로젝트 관련 작업, 형상관리 작업, 코딩 작업, 컴파일 작업 등이 소스 관리 방안 적용 전과 동일해야 한다.

## 3. 형상관리 도구를 이용할 경우의 문제점

대부분의 형상관리 도구들은 한 소스를 기본으로 하여 조금씩 다른 변종(variant)들을 생성시키는 branch 기능과 변종들을 다시 하나로 병합하는 merge 기능을 제공한다. 그런데, 여러 사업자를 대상으로 하는 교환기 소프트웨어를 branch 기능을 사용하여 관리할 경우 다음과 같은 단점이 있다

① 사업자별 요구 기능 구현이 소스 파일 내 몇 행의 차이만 있을 경우에 branch 를 두는 것은 관리가 복잡해질 뿐만 아니라 불필요한 컴퓨터 자원이 낭비된다.

② 개발자가 서로 다른 사업자에 대한 작업을 동시에 수행해야 할 경우, 각 branch 마다 추가적인 형상관리 작업을 수행해야 한다.

③ 한 사업자에 대해 구현되었던 특정 기능을 다른 사업자의 branch 에도 적용하려면 branch 간 merge 가 필요하다. 그러나, 형상관리 도구에서 지원하는 merge 기능은 전체 파일의 내용을 합치는 것이기 때문에, 원하는 부분만을 추출하여 merge 할 수 없다. 따라서, 개발자가 수동적인 방법을 이용해야 하는데, 이때 불필요한 오류가 발생할 가능성이 많다.

④ 소스 파일 전체 내용을 merge 해야 하는 경우에도, 개발자의 판단이 필요할 경우, 수동 작업으로 전환해야 한다.

⑤ merge 문제를 고려하여 기능별로 branch 를 생성하면, branch 가 너무 많아져 형상관리 자체가 힘들어진다.

## 4. 사업자별 소스 관리 방안

### 4.1. 기본 방안

개발자는 하나의 소스 파일에 사업자별 또는 기능별로 코드를 구분하기 위하여 cpp(선행처리기)[7]의 #if directive 를 사용한다. cpp 는 CHILL/C/어셈블리 소스 파일을 컴파일할 때 반드시 구동되는 프로그램이다. 다음의 프로그램 예는 CHILL 프로그램에서 #if directive 를 사용하여 통신 사업자인 한국통신(KT), 하나로(HANARO), 데이콤(DACOM)과 교환기 기능 중 하나인 CS-2 에 대한 코드를 구분한 예이다. '__SYSTEM__'은 사업자 이름을 나타내기 위한 식별자이다 기능별 구분을 위해서는 '__FUNCTION__'을 식별자로 사용한다.

```
10 #if __FUNCTION__CS_2
11     /* CS-2 기능에 대한 코드 */
12     DCL b INT;
13 #else
14     /* CS-2 가 아닌 기능에 대한 코드 */
15     DCL b ARRAY (0:3) BYTE;
16 #endif
   . .
20 #if __FUNCTION__CS_2
21     #if __SYSTEM__KT
22         /* CS-2 기능 중 한국통신에 대한 코드 */
23         a := b;
24     #elif __SYSTEM__DACOM || __SYSTEM__HANARO
25         /* CS-2 기능 중 데이콤과 하나로에 대한 코드 */
26         a  = INT(b);
27     #else
28         /* CS-2 기능 중 다른 사업자에 대한 코드 */
29         a := 10;
30     #endif
31 #else
32      /* CS-2 가 아닌 기능에 대한 코드 */
33      a := DEFAULT_VALUE;
34 #endif
```

개발자는 소스 파일을 컴파일할 때 사업자 이름을 지정하면, 자동으로 '__SYSTEM__사업자이름'과 사업자에 정의된 default 기능들에 대한 '__FUNCTION__기능이름'이 1 로 정의되어, 해당 사업자 또는 기능에 해당하는 코드들이 컴파일 결과물에 포함된다

### 4.2. SYSTEM.h

SYSTEM.h 는 사업자별, 기능별 소스 코드 구분을 위한 식별자를 정의한 파일이다. 또한, 사업자마다 default 기능을 정의한다. 다음은 SYSTEM.h 파일의 예이다.

```
#if __SYSTEM__KT
       #define __FUNCTION__STAR_DB        1
       #define __FUNCTION__CS_2           1

#elif __SYSTEM__DACOM
       #define __FUNCTION__CS_2           1

#elif __SYSTEM__HANARO
       #define __FUNCTION__STAR_DB        1
```

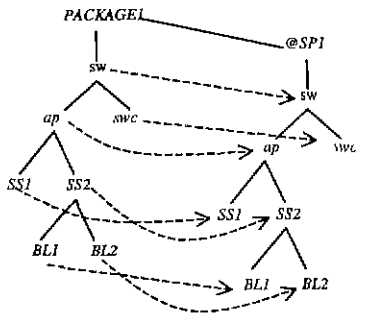
### 4.3. amake

amake는 본 방안의 핵심적인 도구로서, 사업자 이름을 지정할 수 있도록 GNU make[5]를 수정하였다. amake는 개발자가 지정한 사업자이름에 따라, '-D__SYSTEM__사업자이름=1' 옵션과 사업자 default 기능들에 대한 '-D__FUNTION__기능이름=1' 옵션들을 추가하여 소스 프로그램을 컴파일한다. 컴파일 결과물은 다음에 재사용할 수 있도록 target node(4.4. 참조)에 저장한다.

예를 들어, "*amake @KT*"를 수행하면, '-D__SYSTEM__KT=1', '-D__FUNCTION__CS_2=1', '-D__FUNCTION__STAR_ DB=1' 옵션이 지정되어, 결과적으로 12, 23 행의 코드는 컴파일하고, 15, 26, 29, 33 행의 코드는 컴파일하지 않는다.

### 4.4. 디렉토리 구조

각 사업자들에 대해 컴파일 할 경우, 이전의 컴파일 결과물들을 재사용하기 위해서는 각 사업자별로 컴파일 결과물을 저장하는 곳을 따로 마련해야 한다. 그렇지 않을 경우, 컴파일 대상이 되는 사업자가 바뀔 때마다 처음부터 다시 컴파일 해야 하는데, 교환기 소프트웨어는 규모가 크기 때문에 컴파일 시간이 매우 많이 걸린다. 다음 그림은 본 방안에서 제안한 디렉토리 체계이다.




사업자 SP1에 대해 @SP1이라는 새로운 노드(target node)를 생성하고 target node의 directory들을 PACKAGE 노드(source node)의 directory 구조와 동일하게 구성하고, source node의 각 directory 하에 @SP1이란 이름으로 symbolic link 시켰다. Source node는 교환기 소프트웨어의 소스가 있고, target node에는 해당 사업자의 컴파일 결과물이 저장된다. 개발자들은 source node에서 target node를 symbolic link를 통해 쉽게 참조할 수 있다.

### 4.5. 기타 도구들

funclist는 SYSTEM.h에 설정된 사업자별 default 기능들의 리스트를 출력한다. syscheck는 소스 파일을 검사하여 사업자/기능 식별자가 유효한지를 검사한다. sysnode와 unsysnode는 각각 사업자별 target node를 생성 또는 삭제한다.

## 5. 본 방안의 장점

① 컴파일 시 개발자가 지정한 사업자나 기능에 따라 소스 파일에서 코드를 줄단위로 첨삭할 수 있다.

② amake가 사용하는 makefile 내에 사업자나 사업자 요구 기능에 따라 소스 파일을 첨삭할 수 있다.

③ 작업할 소스 파일이 하나이므로 쉽게 파일을 참조하거나 편집할 수 있고, 사업자별, 기능별 공통 기능을 일관성있게 구현하거나 변경할 수 있다.

④ 한 사업자의 기능을 SYSTEM.h의 식별자의 조정만으로 쉽게 다른 사업자에 이식할 수 있다.

⑤ 사업자별 target node에 의해 한번 컴파일된 결과를 재사용함으로써 컴파일 시간을 단축시킨다.

⑥ 기존의 환경에 최소의 변경으로 쉽게 구현할 수 있다.

## 6. 결론

본 논문은 교환기의 사업자별 기능을 하나의 소스로 관리하는 방안에 대하여 기술하였다. 이 방안은 cpp의 '#if' directive를 사용하여 하나의 소스 파일에서 사업자별/기능별로 코드를 구분할 수 있고, target node 디렉토리 구조를 사용하여 사업자별 컴파일 결과물을 쉽게 관리/참조/재사용할 수 있다는 장점이 있다

앞으로 소스 파일에서 특정 사업자 내용만 추출하여 볼 수 있는 편집기를 개발할 예정이다.

## 참고 문헌

[1] Wayne A.Babich, *Software Configuration Management*, Addison-Wesley, 1986

[2] *SABLIME Product Administration System Administrator's Manual*, AT&T, 1996

[3] *ClearCase Concept Manual*, Atria Software. 1994

[4] *CVS-Concurrent Versions System*, Signum Support AB, 1993

[5] Richard M. Stallman and Roland McGrath, *GNU make Version 3.77*, Free Software Foundation, 1998

[6] *NMAKE User's Manual and Reference Guide V3.0*, AT&T, 1994

[7] Richard M. Stallman, *The C preprocessor for GCC version* 2, Free Software Foundation, 1992